Project Milestone 4 – Algorithm Refinement Answer Sheet

Instructions

1. Read this document carefully. You are responsible for following all instructions in this document.
2. Read the Learning Objectives at the end of the document to understand how your work will be graded.
3. Use professional language in all written responses and format all plots for technical presentation. See EPS01 and EPS02 for guidelines.
4. Good programming standards apply to all m-files.
5. Submit deliverables to Gradescope and to Blackboard. Name your files to match the format in the table below, where *SSS_TT* is your section and team ID (e.g., 001_03 is Section 001, Team 3)

Item	Deliverables
M4 Answer Sheet	Project_M4_AnswerSheet_*SSS_TT*.docx
M4 Algorithm	M4_Algorithm_*SSS_TT*.m
M4 Regression model	M4_Regression_*SSS_TT*.m
M4 Executive Function	M4_exec_*SSS_TT*.m
Technical Brief Draft	M4_TechnicalBrief_*SSS_TT*.docx
Gradescope Submission	M4_*SSS_TT*.pdf

   See submission requirements on the last page of this answer sheet.

6. Complete the Assignment Header before starting the answer sheet.

Assignment Header

Section and Team ID (SSS_TT):	001_30

Team Member Name	**Purdue Career Account Login**
Alex Norkus	anorkus
Surya Manikhandan	smanikha
Julius Mesa	jmesa
Vincent Lin	lin971

Part 0: M3 Feedback Review

Reflect on your M3 feedback for the purpose of improvement. Your reflection should provide a clear, useful summary of your M3 feedback and provide a clear and practical plan to address the issues. Complete table 1 below.

Table 1. Feedback summary and plan

Part A: Summarize the feedback you received on M3 that could lead to improvements in your work.

As part of our M3 grading, we were given many potential avenues for improvements which we plan to implement in this particular milestone document. Each feedback here corresponds to the same number on the implementation

1. We needed significantly more elaboration when we are explaining the effects of our improvements on parameter identification (v0, Vmax, and Km).
2. We needed to elaborate how our executive function worked and also use appropriate technical language to describe the process.
3. A significant piece of feedback we received was to further explain how our improvements affected the specific parameters we are examining (v0, Vmax, Km).
4. Although we have followed the same method of citation in the previous milestones without any issues, we were counted off on the references in this milestone. This likely has something to do with the in-text citation according to a Peer Teacher's comment.
5. Regarding the M3 Regression Assignment, we had incorrect function call and we didn't have units in the explanation of the input data, which may cause mistakes when using the function.
6. We received multiple programming standards violation comments. These range from errors in the function header, having outputs on the executive function, not properly calling functions, etc.
7. We had incorrect values for SSE and SST on the regression function.
8. The header for the executive function needed to better explain how the regression function was implemented into the executive function.
9. The executive function produced an output when we were not suppose to put any input or output included in the function.
10. Linear regression equation was not displayed on the graph.

Part B: Explain how you will incorporate the M3 feedback to improve your parameter identification (do not just reword your response from Part A).

The methods for implementation shown below correspond to the feedback item in the cell above.

1. We will look over our document after we are done with it to ensure that we have connected how our changes have affected the parameter identification and explain in as much detail as possible. We can obtain 3d party verification by asking another team to peer-review our document for thoroughness. To show that we have improved our findings to v0, Vmax, and Km we will compare how much percent better our new SSE values compared to that of the PGO-X50 Reference Values AND the values we produced in the previous iteration of the program.
2. To change our technical language to be more suitable to class standards, we will use a thesaurus to better describe our thoughts for either changes or explanations we are trying to make in the document.
3. We will specifically state the parameter being improved, and elaborate on how the changes we made affect that specific parameter mathematically. We will make references to formulas where possible to better describe this.

4. To have a more formal citation, we will exclusively use APA in-text citations and references. We will ensure this by double-checking our citations with Purdue Online Writing lab, a fantastic resource for technical writing.
5. After inspecting the grading, it seems we were counted off for not including the pricing in our executive function, which led to the incorrect function call deduction. Therefore, we will address this by calling the regression function with our exec function
6. We will very closely refer to the course programming standards to check that our programs are in full compliance.
7. We will examine how the SSE & SST were calculated in our M3 submissions and attempt to find the mistakes we made in the calculations. We will fix these errors in calculations as we see fit. EDIT: The issue was the SSE was being calculated with the linearized data, which is incorrect. WE will change the code to calculate the SSE with the non-linearized data.
8. The regression function was NOT integrated in the executive function in our initial submission. We will fix this by ensuring the executive function will call the regression function appropriately.
9. The executive function will NOT use output arguments anymore. Instead, we will print out all of the relevant information directly to the command window, which we were told is the correct solution instead of the output arguments used before. This way, we can also make it more user friendly by stylizing the output.
10. We will read MATLAB documentation in order to find out how to display text on the graph. Then, we will implement a solution in order to display the proper equation on the chart.

Part 1: Algorithm Refinements Plan

Respond to each of the prompts below in the space provided. Your goal is to introduce **two refinements** to your M3 algorithm, and these refinements must improve your solution to the NovelEnzymes 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:

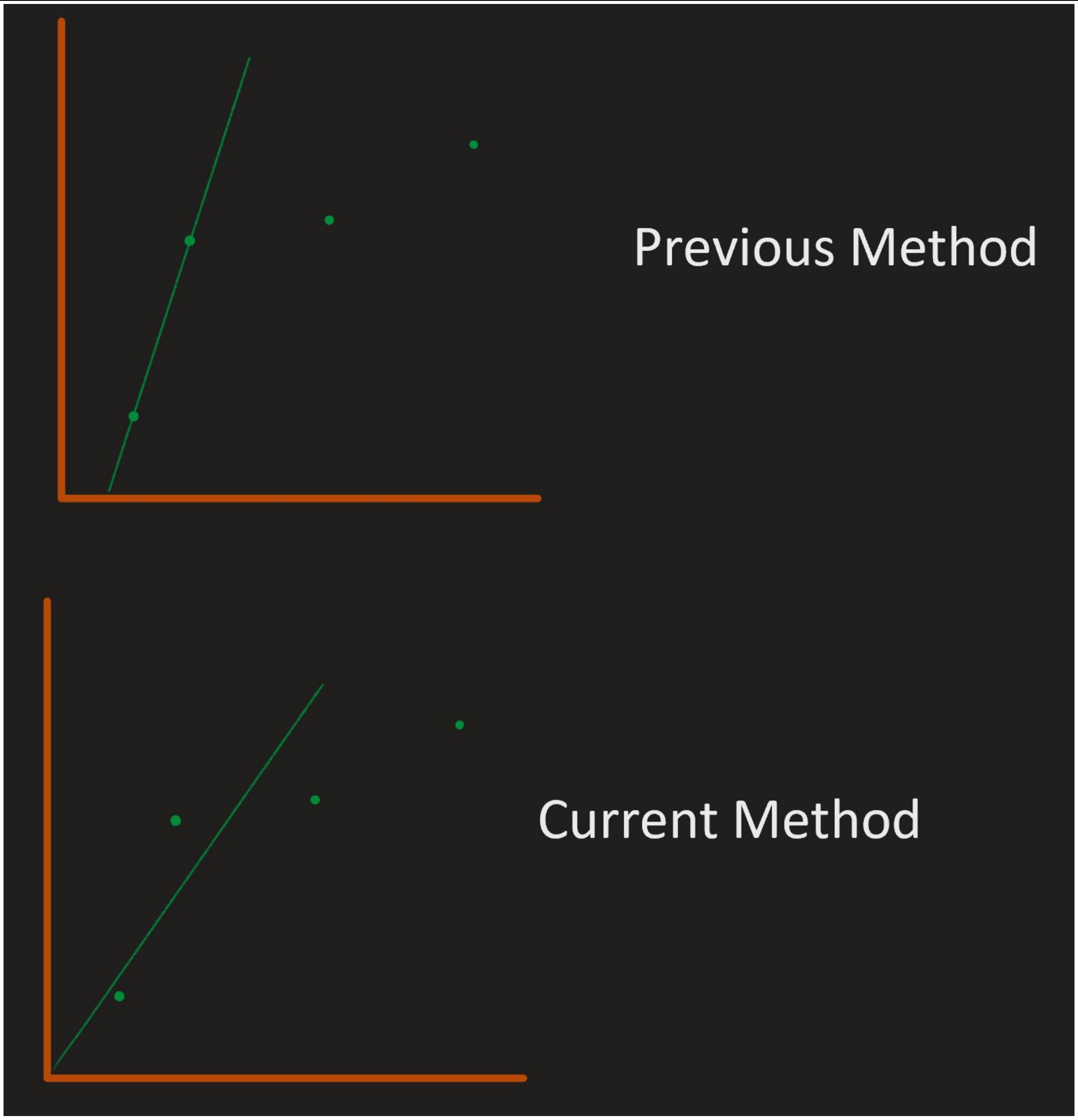
- **Refinement Category 1: Parameter Identification (Required)**
  An improvement that changes the way you are doing parameter identification, and that improves your parameter identification results.
- **Refinement Category 2: Algorithm Efficiency**
  An improvement that improves the efficiency of your code by (for example) removing unneeded 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 **two** refinements. One refinement must be in Category 1. The second refinement can come from either Category 2 or Category 3. Use your ideas from

Part 3 of M3 to help formulate ideas. 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.

Table 2. Algorithm refinement plans

Refinement 1: Category 1. Parameter Identification
Parameter(s) Targeted: *V0*
Description *Our Algorithm uses the tangent line method in order to calculate v0 at each substrate concentration. Essentially, the algorithm starts by plotting product concentration [P] in uM on the Y axis and the time in minutes on the X-axis. Before this plotting step, the raw data for [P] vs time is smoothed using the moving-average method discussed in detail in the previous milestone documents.* *Once the smoothed data has been plotted, the algorithm will then calculate v0. Previously, this was done taking the first two points form the plot and using the formula rise/run in order to calculate the slope, which is our v0 value. Although this worked fairly well, the datasets given in this assignment had significantly more noise and sometimes systematic bias.* *Therefore, we will be implementing a new system for calculating the initial slope and therefore v0. M4 Algorithm will pick the first 3 smoothed data points and find the linear regression of those points. The intercept for the regression has been forced to the (0,0) origin point.*
Rationale for Refinement *This improvement is necessary because the datasets given to us in this assignment have significantly more noise and therefore bias. For example, some datasets had spikes in the middle, while others had increasing variability as time went on.* *For this particular assignment, it was clear that our method of choosing the first two points for the slope was not a great idea as it was being affected by the bias in the datasets, particularly at the lower substrate concentrations. The increased noise had started to affect the smoothness of the curve and therefore the integrity of the first two points. This means that the first two points being used to draw the tangent lines were not producing an acceptable tangent line, even though out smoothing function has done its best to iron out the noise in the data.* *Using three points allows us to be better resistant to bias in the dataset and produce a significantly better tangent line as it provides a buffer rather than just the first two points. Fitting a best fit line to these first 3 points allows us to make a much better tangent line than just the first two points. While it is hard to explain, the diagram below shows why we believe this is a great idea.*



As shown in the picture above, the two point method is easily fooled if either of the points are not perfectly on the tangent vector, which is an assumption we made for the previous milestone. Additionally, we noticed that the tangent lines do not pass through the origin in almost all instances when using this method, which produces an inaccurate tangent line as the raw data always starts off at the origin. Our improved 3-point method is able to account for any unexpected variation in the data points which would have been a significant threat to our previous algorithm, and is therefore superior.

Refinement 3: Category 2. Algorithm Efficiency
Parameter(s) Targeted: *Execution Time*
Description

Given the extensive amount of data needed to be imported in the code (>7000 unique lines with over 100 columns), increasing the efficiency of the data imports would shave off significant amounts of time. Therefore, we will have to explore alternative methods of importing data OR examine ways to cut down the amount of information we ae importing (just a few rows instead of the whole column, for example).
Rationale for Refinement *According to the MATLAB Profiler feature, the M3 code spent more than 90% of the time importing the data. Specifically, it took a significant amount of time to import small data sets using read matrix, which seems contradictory. For example, It took longer for the code to import the 100 substrate concentration [S] values than it did to read in all the thousands of [P] values. We ran tests of several import functions such as readmatrix, csvread, dlmread, and xlsread on a small dataset (100 elements to simulate the [S] values). This showed that for small datasets, the xlsread function was superior compared to the others in execution time. However, when trying to import the complete m4 dataset, readmatrix was the winner. Therefore, we will be using xlsread wherever we are importing small amounts of data such as [S], and readmatrix for importing large data such as [P].*

Part 2: Algorithm Refinements Implementation

Before you make any changes to your code, resave your M3 code files as

- M4_Algorithm_*SSS_TT*.m
- M4_Regression_*SSS_TT*.m
- M4_exec_*SSS_TT*.m

Category 1 Refinement (Required)

Implement your Category 1 refinements in M4_Algorithm_*SSS_tt*.m. Clearly comment the refinement changes within the code, using the text 'Category 1' and a concise, meaningful description of the change.

Evaluate the improvement in your algorithm by using the clean and noisy data for the reference enzyme PGO-X50 from M2. Compare the parameters identified for the PGO-X50 data using the algorithm you submitted in M3 and your refined algorithm for M4. Report your results in Table 3. Take care with decimal places.

Table 3. Algorithm refinement comparison

Parameter (µM/min)	PGO-X50 Reference Values		M3_Algorithm		M4_Algorithm	
	Clean	Noisy	Clean	Noisy	Clean	Noisy
v_{0_1}	0.028	0.028	.0283	.0305	.0281	.0282
v_{0_2}	0.056	0.055	.0567	.0572	.0562	.0570
v_{0_3}	0.110	0.11	.1121	.1022	.1112	.1111
v_{0_4}	0.193	0.19	.1959	.1657	.1944	.1979
v_{0_5}	0.360	0.338	.3674	.3546	.3650	.3591
v_{0_6}	0.6	0.613	.6059	.6395	.6026	.6060
v_{0_7}	0.883	0.917	.8915	.9439	.8878	.8790

v_{0_8}	1.212	1.201	1.2192	1.2364	1.2157	1.1796
v_{0_9}	1.376	1.282	1.3806	1.2812	1.3783	1.3269
$v_{0_{10}}$	1.584	1.57	1.5873	1.5274	1.5858	1.5788
V_{max}	1.72	1.61	1.7605	1.6782	1.7604	1.7375
K_m (μM)	226.92	214.28	235.8686	224.6648	237.8632	237.4837
SSE $(\mu M/min)^2$	0.0041	0.0251	0.0301	0.0224	0.02826	0.0079
* Verify your SSE values by comparing them to the provided SSE values for the reference parameters.						

Next, use your M4 algorithm to analyze the full 100 enzyme test data sets and obtain the parameters V_{max} and K_m. In Table 4, copy your enzyme parameter and model goodness of fit results from M3 (i.e., the values from M3 Table 3) and record your results from your M4 algorithm. Take care with decimal places.

Table 4. M3 and M4 algorithm comparison of experimental data parameters

Enzyme	M3 Algorithm			M4 Algorithm		
	Enzyme Parameters		SSE $(\mu M/s)^2$	Enzyme Parameters		SSE $(\mu M/s)^2$
	V_{max} (μM/s)	K_m (μM)		V_{max} (μM/s)	K_m (μM)	
NextGen-A	0.9174	155.0399	.0003	.9737	150.3015	.00016
NextGen-B	0.8988	355.4164	.0025	.8763	337.2487	.00195
NextGen-C	1.2349	187.2996	.0010	1.2471	185.0924	.00002
NextGen-D	1.6574	304.0967	.0001	1.6277	288.6323	.00025
NextGen-E	1.7222	179.1636	.0010	1.7027	167.2828	.00065
% decrease in SSE from M3 to M4 (Average)						84.17%

Category 2 Refinement (Optional—Choose one of either Category 2 or 3)

Implement your Category 2 refinements to increase the efficiency of your algorithm. Run your code and document the effect of the refinements. Use the MATLAB functions `tic` and `toc` to measure how long it takes your program to execute. Efficiency refinements must be clearly commented in your M4 code with the text 'Category 2' and a concise, meaningful description.

Do not delete any code as you implement the refinements: comment out unnecessary code and comment on the change. New code must be designated as such.

Record the execution time of your M3 program and your refined M4 program in Table 5.

Table 5. Program execution times

Program	Execution Time (s)
M3 (before refinement)	9.321
M4 (after refinement)	2.760
% reduction in execution time	108.604%

NOTE: Time Measurements were done with laptopped unplugged and the graphics card disabled. Laptop was running in the 'balanced' power configuration to prevent CPU turbo boost.

Category 3 Refinement (Optional—Choose one of either Category 2 or 3)

After refining 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 other 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 a concise, meaningful description.

If you need guidance or other suggestions about how to execute this refinement, be sure to ask the teaching team.

Table 6. Algorithm insight results

Refinement 3 Insight Plot(s)
<insert plot(s) here>
Description of Insights Gained
<write description here>

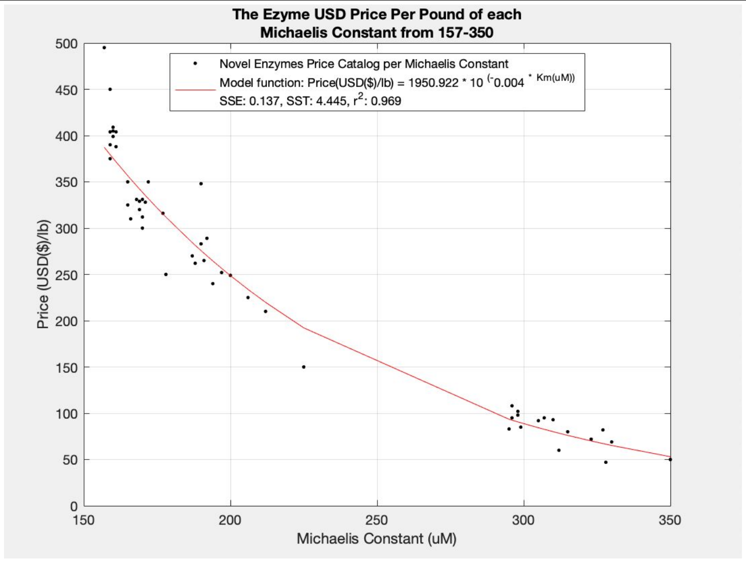
Part 3: Enzyme Pricing Recommendations

Now that you have implemented your algorithm refinements, you have a set of parameters with which to determine an initial price point for each of the five enzymes. Considering the parameters found by your M4 algorithm, decide what changes, if any, you need to make to your regression algorithm. These could be corrections, response to M3 feedback, or changes to model function. Implement changes as needed.

Report the regression model information in Table 7. Take care with decimal places.

Table 7. Regression results

Parameter	Value	Plot

Model	Exponential Function	
SSE ($\$^2$)	0.137	
SST ($\$^2$)	4.445	
r^2	0.969	

Using your model, provide a price for each enzyme. Report your values in Table 8. Add an indicator to any extrapolated value.

Table 8. Price prediction for each enzyme using regression model

Enzyme	**Price ($/lb)**
NextGen-A	414.94
NextGen-B	60.51
NextGen-C	289.99
NextGen-D	99.83
NextGen-E	348.36

In Table 9, justify your regression function selection and any extrapolation required to provide a using evidence-based rationales.

Table 9. Regression model justification

The regression function linearizes measured price using four different kinds of equations, and then calculates SSE value, find the lowest SSE value which indicates the best equation to model the relationship between price and Km value of the enzyme. After running the function, we got the SSE,SST and r^2 value as shown in the chart below.

- By using power equation and exponential equation, we can model the data with a low SSE value of 0.1397(power) and 0.1367(exponential), which indicates these two equations best fit the data.
- We also take r^2 value into consideration to make sure that the model can explain the percent of the variability in the data well. While r^2 value of power equation(0.9686) is very close to the r^2 value of exponential equation(0.9693), it’s hard to decide which model to choose.
- What we consider further is that if the model is reasonable in reality and if the model predicts a reasonable price for extreme Km value. When the Km value is close to 0, the exponential function predicts that the price would be 1950.92$/lb, while the power function predicts the price to be ∞ (indefinite). After some research, the maximum price of bulk detergent enzyme is 575$/lb. (Thermo Scientific™, 2011) , so it is unreasonable that the price will reach unlimited. Thus, we chose the exponential equation to model the price.

No extrapolation was done in finding the price values for the enzymes. In the reference data, the minimum and maximum Km values being 150 and 350, respectively. While estimating the price, the minimum Km value is 150.3015 microMoles for Enzyme-A, and the maximum Km value is 337.487 microMoles for Enzyme-B. Since both values are within the range of 150 and 350, no extrapolation is needed while estimating the price for all 5 enzymes.

	SSE	r^2
Linear equation	80307	0.903
Power equation	0.1397	0.9686
Exponential equation	0.1367	0.9693
Logarithmic equation	58649	0.929

	SSE	r^2
Linear equation	80307	0.903

Power equation	**0.1397**	**0.9686**
Exponential equation	**0.1367**	**0.9693**
Logarithmic equation	**58649**	**0.929**

Part 4: Technical Brief Draft

Consult the M4 memo from NovelEnzymes, Inc. for the details concerning your technical brief. Use the provided template M4_TechnicalBrief_template.docx to respond to the memo. You may find the original introduction memo and the project background documents helpful when composing your technical brief.

Table 10. References used in evidence-based rationales (answer sheet only)

Thermo Scientific B-PER Bacterial Protein Extraction Reagents - BioPharmaceutical Production, Proteomics and Immunoassays. (n.d.). Retrieved from https://www.fishersci.com/shop/products/thermo-scientific-pierce-b-per-bacterial-protein-extraction-reagents-reagent-165ml/pi78243#?keyword=true.

How to Submit

1. Rename this answer sheet to be **Project_M4_AnswerSheet_*SSS_TT*.docx** where ***SSS*** is your section number (e.g., 001 for section 001) and ***TT*** is your team number (e.g., 07 for team 7). Save the answer sheet as a PDF.
2. Create a PDF of your technical brief document.
3. Publish each function you created in this milestone. Note: you may publish sub-functions (such as M4_Algorithm_*SSS_TT*.m) without filling in appropriate inputs or outputs – this will generate an error in the published code for that function that will not affect your grade. Note: these functions must run properly when called by your executive function.
4. Merge the answer sheet PDF, the technical brief PDF, and the published code PDFs into one PDF file named **M4_*SSS_TT*.pdf**.
5. Select one person to submit the PDF for the team. That person should
    a. Log into Gradescope and submit **M4_*SSS_TT*.pdf** to the **M4** assignment.
    b. Indicate which pages correspond to each part of the milestone.

       Failure to tag the items appropriately will result in your work receiving losing credit.
    c. Select all team members for the group assignment.
    d. Double-check that all team members are assigned to the submission.
6. Select one person to submit all m-files, data files, and answer sheet to the **M4 dropbox** on Blackboard.

   Failure to submit your files to Blackboard may result in every team member receiving a zero on this assignment.
7. Each team member should confirm that they are part of the submission and that all parts of the answer sheet were properly tagged.
8. After submission, distribute the submitted files to all team members. *Ensure all members of the team have copies of the submitted files.*

Learning Objectives

Process Awareness (PA)
Reflect on both personal and team's problem solving/design approach and process for the purpose of continuous improvement.
PA02. Identify limitations in the approach used.
PA03. Identify potential behaviors to improve approach in future problem solving/design projects.

Idea Fluency (IF)

Generate ideas fluently. Take risks when necessary.

IF03. Generate testable prototypes (including process steps) for a set of potential solutions.

Evidence-Based Decision Making (EB)

Use evidence to develop and optimize solution. Evaluate solutions, test and optimize chosen solution based on evidence.

EB01. Test prototypes and analyze results to inform comparison of alternative solutions.

EB03. Clearly articulate reasons for answers with explicit reference to data to justify decisions or to evaluate alternative solutions.

EB05. Present findings from iterative testing or optimization efforts used to further improve aspect or performance of a solution.

EB06. Clearly articulate reasons for answers when making decisions or evaluating alternative solutions.

Solution Quality (SQ)

Design final solution to be of high technical quality. Design final solution to meet client and user needs.

SQ01. Use accurate, scientific, mathematical, and/or technical concepts, units, and/or data in solutions.

Information Literacy (IL)

Seek, find, use and document appropriate and trustworthy information sources.

IL04. Include citations within the text (in-text citations) that show how the references at the end of the text are used as evidence to support decisions.

IL05. Format reference list of used sources that is traceable to original sources (APA or MLA are recommended)

Engineering Professional Skills

PC05. Fully address all parts of assignment by following instructions and completing all work.

EPS01. Use professional written and oral communication.

EPS02. Format plots for technical presentation.

Programming

MAT01. Develop code that follows good programming standards.

MAT03. Perform mathematical operations and calculations within MATLAB.

MAT05. Create and use MATLAB scripts and functions.

MAT08. Debug scripts and functions to ensure programs execute properly, perform all required tasks, and produce expected results.

MOD01. Create mathematical models to describe relationships between data.

Project Milestone 4 – Technical Brief Draft

To: John P. Watson, President, NovelEnzymes Inc.

From: ENGR 132 Team 30

RE: Data Analysis

Date: 12/1/19

Introduction

NovelEnzymes Inc. has sent a memo that asked us to create algorithms in order to help them determine an initial starting point for enzyme pricing which is done by our function that is supposed to be able to create a model to determine the price of each of the different types of enzymes tested. We can ensure that our algorithm is correct by comparing our v0 values with the PGO-X50 Reference Values given to us. Some constraints to this project are that the executive function must be fully automated, the executive function must have no inputs or outputs, and this technical brief is due by 1:30 pm on 12/2/19.

The algorithm works by taking a tested optimized range of data points of the reacted enzyme (17 unit range for noisy and 11 for clean) and averaging the values within that range in order to produce smooth data to plot on a graph. It then moves onto the next set of data points to smooth until it reaches the end of the enzyme data. For example, the first set of data includes 1st-17th data points(for noisy data), and the second set of data includes 2nd -18th data points.

Our first key decision in improving the algorithm was how to approach noisy data; we decided to find a moving average for all of the data or to group data and then average them. This presented another issue: how large should the data grouping be without losing data while also minimizing the effects of outlying data? This impacted the quality of our solution by also potentially removing any crucial data points that could have potentially been able to create a more accurate graph. This can be seen within M2 Figures 3.2-3.6 where, although there is an accurate graph that gives the general model for the velocity of the equation, it is seen that there is a loss of data as a result of our current method. That brings us to our next critical decision to decrease data loss.

A second critical decision was optimizing the width range that the algorithm averaged either the noisy or clean data. We ran a test to calculate the r^2 value of a model using data grouped by different sizes and we determined that the optimal averaging size would be the one with the corresponding r^2 value closest to one. This width value ended up to be a value of 17 for the noisy data and 12 for the clean data. What we also did to improve the moving average algorithm was to instead make the width algorithm move one index value at a time. To elaborate, our original method moved the width range for the noisy data from 1-17 to 18-34. This created a tremendous loss of data, so to account for this we changed the algorithm to move the width range by one index at a time such that it would move from 1-17 to 2-18 and so on. It can be clearly seen in M4 Table 3 that the algorithm has improved by 6.11% and 64.73% for clean and noisy data respectively.

The final key decision we made to improve our algorithm was in calculating the Km and Vmax using the Hanes-Woolf plot[1]. Using a linearized model and v0 values, the slope of the model is equivalent to the inverse of Vmax, while the y-intercept is Km/Vmax as seen in Appendix A of this document.

Parameter Identification Procedure

Our algorithm starts off by importing all the test results for the enzymes and, based on the enzyme inputted into the algorithm, selects the columns to use from the data. The algorithm then begins a looping structure for each column of the test results, sending both the original test and duplicate test data into our smoothing function, which returns a smoothed column for the test results and smoothed column for time. The looping structure continues to calculate the v0 values for the test by dividing each value in the smoothed time column by each value in the smoothed test results, this is done with both the original and corresponding duplicate test. These v0 values are then used to calculate Km and Vmax in the executive function. The executive function linearizes the v0 values using the Hannes-Wolf method, then plots them against the Substrate data (Marasović et al., 2017). The Vmax is then calculated by dividing 1 by the slope of the plot, and the Km value is found by dividing the y-intercept of the plot by the slope of the plot. For a visual of the Hannes method, Appendix A illustrates this linearization method.

Results

As you can see from the example execution located in Appendix B, our algorithm returns ten initial reaction velocity values, the maximum reaction velocity, and Michaelis-Menten constant for each of the five enzymes, with the highest maximum reaction velocity being that of enzyme-E at 1.7027, though it is also the second most expensive in suggested price, the most being enzyme-A at $414.94 per pound. We have attached the suggested prices for each enzyme and displayed the equation for the model used to interpret price points in the figure. The goodness of fit statistics for this model has also been included, with the r^2 value being 0.969 and the SSE and SST values being 0.137 and 4.445, respectively. With the calculated statistics, we believe the model should be effective in predicting prices accurately and, since none of the enzymes fell outside the data range, you should find all the suggested prices for the enzymes to be useful.

Interpretation

The error that we can account for in the algorithm is the loss of data with our current smoothing model. Because it is a moving average, there is data loss in the creation of smooth data because it must take the average of the width and combine them into one value. As a result of this, the total data will be decreased by a factor of how many iterations the code runs until it has gone and averaged all the data. In essence, it can be defined as *(smoothed data points) = (number of data points) / iterations*. Evidence of this is seen within our code when it takes the file Data_PGOX50_clean or Data_PGOX50_noisy with an index value of 1222 per test in the file and creates a resulting smoothed dataset that will have an index value of 1222/iterations. The correct optimized width will vary depending on if it is working with either smooth or noisy data.

What NovelEnzymes can honestly say about their products performance is the enzymes function to a specific precision. This goes by saying that their margin of error is low so they have consistent results to show for their customers. Based on the data gathered the noisy data isn't too bad with an r^2 value of 0.969. This being said, they can say that they can guarantee that their product is not going to stray too far from any data collected.

References

Marasović, M., Marasović, T., & Miloš, M. (2017). Robust Nonlinear Regression in Enzyme Kinetic Parameters Estimation. *Hindawi Journal of Chemistry*, *2017*, 1–12. DOI: 10.1155/2017/6560983

APPENDIX A:

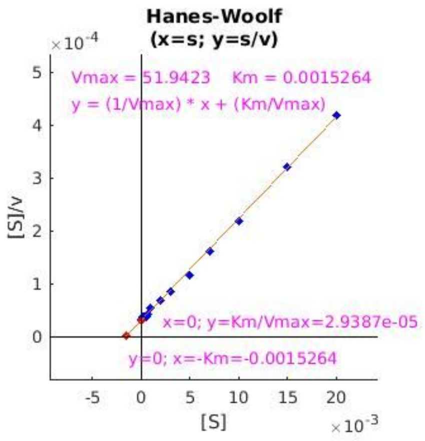
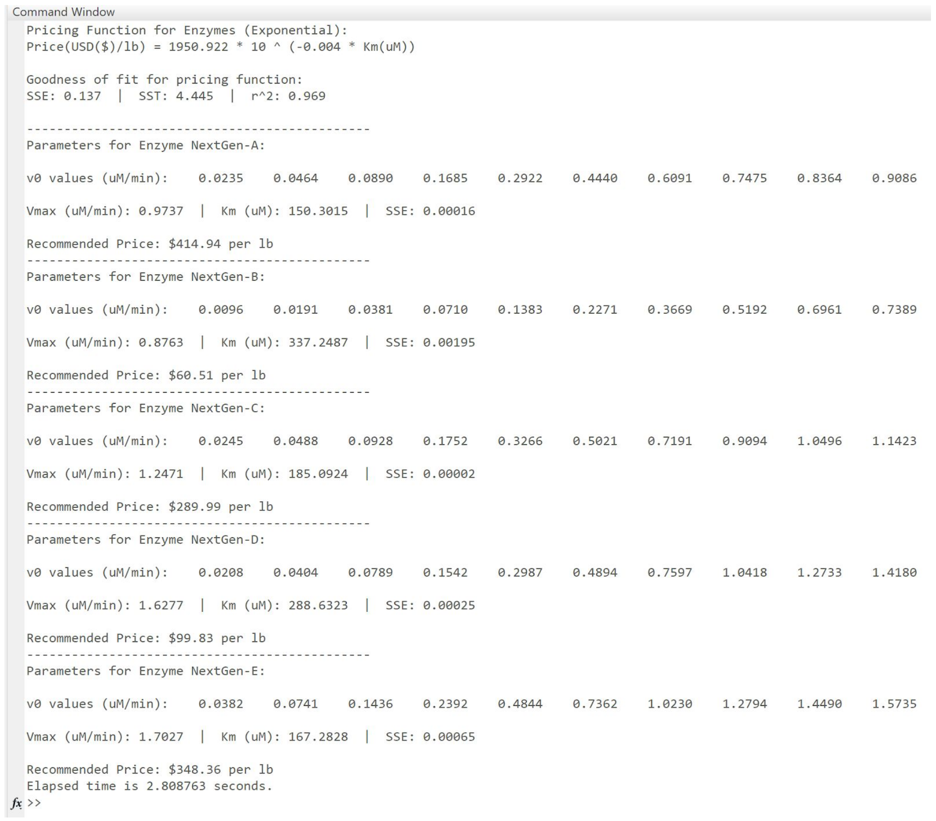


Image Source: https://bit.ly/2quI9YQ

APPENDIX B



```
Command Window
  Pricing Function for Enzymes (Exponential):
  Price(USD($)/lb) = 1950.922 * 10 ^ (-0.004 * Km(uM))

  Goodness of fit for pricing function:
  SSE: 0.137  |  SST: 4.445  |  r^2: 0.969

  ---------------------------------------------
  Parameters for Enzyme NextGen-A:

  v0 values (uM/min):    0.0235    0.0464    0.0890    0.1685    0.2922    0.4440    0.6091    0.7475    0.8364    0.9086

  Vmax (uM/min): 0.9737  |  Km (uM): 150.3015  |  SSE: 0.00016

  Recommended Price: $414.94 per lb
  ---------------------------------------------
  Parameters for Enzyme NextGen-B:

  v0 values (uM/min):    0.0096    0.0191    0.0381    0.0710    0.1383    0.2271    0.3669    0.5192    0.6961    0.7389

  Vmax (uM/min): 0.8763  |  Km (uM): 337.2487  |  SSE: 0.00195

  Recommended Price: $60.51 per lb
  ---------------------------------------------
  Parameters for Enzyme NextGen-C:

  v0 values (uM/min):    0.0245    0.0488    0.0928    0.1752    0.3266    0.5021    0.7191    0.9094    1.0496    1.1423

  Vmax (uM/min): 1.2471  |  Km (uM): 185.0924  |  SSE: 0.00002

  Recommended Price: $289.99 per lb
  ---------------------------------------------
  Parameters for Enzyme NextGen-D:

  v0 values (uM/min):    0.0208    0.0404    0.0789    0.1542    0.2987    0.4894    0.7597    1.0418    1.2733    1.4180

  Vmax (uM/min): 1.6277  |  Km (uM): 288.6323  |  SSE: 0.00025

  Recommended Price: $99.83 per lb
  ---------------------------------------------
  Parameters for Enzyme NextGen-E:

  v0 values (uM/min):    0.0382    0.0741    0.1436    0.2392    0.4844    0.7362    1.0230    1.2794    1.4490    1.5735

  Vmax (uM/min): 1.7027  |  Km (uM): 167.2828  |  SSE: 0.00065

  Recommended Price: $348.36 per lb
  Elapsed time is 2.808763 seconds.
fx >>
```

Table of Contents

```
function [bf, mf, SSE_final, SST] = M3_Regression_001_30

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
% The program finds a model for estimating price of enzyme according
 to
% their Km value and output the estimate price.
%
% Function Call
% [SSE_final, SST] = M3_Regression_001_30;
%
% Input Arguments
% No input argument
%
% Output Arguments
% bf: the intercept value gathered from the regression (unitless - not
 in general form)
% mf: the slope value gathered from the regression (unitless - not in
 general form)
% SSE_final: the SSE value of the linearized data (unitless)
% SST: the SST value of the linearized data (unitless)
%
% Assignment Information
%   Assignment:     M3
%   Team Mmeber:    Luming Lin, lin971@purdue.edu
%                   Surya Manikhandan, smanikha@purdue.edu
%                   Julius Mesa, jmesa@purdue.edu
%                   Alex Norkus, anorkus@purdue.edu
%   Team ID:        001-30
%   Academic Integrity:
%     [] We worked with one or more peers but our collaboration
%        maintained academic integrity.
%     Peers we worked with: Name, login@purdue [repeat for each]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INITIALIZATION

```
clc;clearvars;
data = csvread('Data_NovelEnzymes_priceCatalog.csv',2,0); %import the
 data from the csv file
price_measured = data(:,2); % separate the price from the data
 (USD($)/lbs)
Km_measured = data(:,1); %separates the Km value from the data (uM)
```

Linearizing the data

Linearize the data using linear function

```
coe1 = polyfit(Km_measured,price_measured,1); %find the coefficient of
 trendline of the model
M1 = coe1(1);
B1 = coe1(2);
m1 = M1; %calculate the original model coefficient
b1 = B1;
linearized_predict1 = M1 .* Km_measured + B1; %calculate the predict
 value of price using the linearized model
predict1 = polyval(coe1,Km_measured); % calculate the predict value of
 price using the original model
SSE1 = sum((price_measured - linearized_predict1).^2); %calculate the
 SSE value of the model
```

Linearize the data using power function

```
coe2 = polyfit(log10(Km_measured),log10(price_measured),1); %find the
 coefficient of trendline of the model
M2 = coe2(1);
B2 = coe2(2);
m2 = M2; %calculate the original model coefficient
b2 = 10 ^ B2;
linearized_predict2 = M2 .* log10(Km_measured) + B2; %calculate the
 predict value of price using the linearized model
predict2 = b2 .* Km_measured .^ m2; % calculate the predict value of
 price using the original model
SSE2 = sum((log10(price_measured) -
 linearized_predict2).^2); %calculate the SSE value of the model
```

Linearize the data using exponential function

```
coe3 = polyfit(Km_measured,log10(price_measured),1); %find the
 coefficient of trendline of the model
M3 = coe3(1);
B3 = coe3(2);
m3 = M3; %calculate the original model coefficient
b3 = 10 ^ B3;
linearized_predict3 = M3 .* Km_measured + B3; %calculate the predict
 value of price using the linearized model
predict3 = b3 .* 10 .^ (Km_measured .* m3); % calculate the predict
 value of price using the original model
SSE3 = sum((log10(price_measured) -
 linearized_predict3).^2); %calculate the SSE value of the model
```

Linearize the data using logarithmic function

```
coe4 = polyfit(log10(Km_measured),price_measured,1); %find the
 coefficient of trendline of the model
M4 = coe4(1);
B4 = coe4(2);
m4 = M4; %calculate the original model coefficient
b4 = B4;
linearized_predict4 = M4 .* log10(Km_measured) + B4; %calculate the
 predict value of price using the linearized model
predict4 = m4 .* Km_measured + b4; % calculate the predict value of
 price using the original model
SSE4 = sum((price_measured - linearized_predict4).^2); %calculate the
 SSE value of the model
```

FORMATTED TEXT/FIGURE DISPLAYS

```
SSE_value = [SSE1,SSE2,SSE3,SSE4]; %combine the SSE value of all four
 models
SSE_final = min(SSE_value); %find the minimum SSE value which
 indicates the best model
choose = find(SSE_value == SSE_final); %find which model is the best

figure(1)
if choose == 1 % if the linear function best fits the data
    SST = sum((price_measured - mean(price_measured)).^2); %calculate
 the SST value of the linearized data
    r2_1 = 1-(SSE1/SST); %calculate the r^2 value of the linearized
 data
    r2 = r2_1;
    plot(Km_measured, price_measured,'k.');%plot the original data
    hold on;
    plot(Km_measured, predict1,'r-'); %plot the model
    title({'The Ezyme USD Price Per Pound of each';'Michaelis Constant
 from 157-350'})
    xlabel('Michaelis Constant (uM)');
    ylabel('Price (USD($)/lb)')
    function_output = sprintf("Model function:Price(USD($)/
lb) = %.3f * Km(uM) + %.3f\nSSE: %.3f, SST: %.3f, r^2:
 %.3f",b1,m1,SSE_final,SST,r2); %display the funciton of the model
 also the SST,SSE and r^2 value
    legend('Novel Enzymes Price Catalog per Michaelis Constant',
 function_output, 'location', 'best')
    grid on;
    %assign return parameters for the exec function
    bf = b1;
    mf = m1;

elseif choose == 2 % if the power function best fits the data
    SST = sum((log10(price_measured) -
 mean(log10(price_measured))).^2); %calculate the SST value of the
 linearized data
    r2_2 = 1-(SSE2/SST); %calculate the r^2 value of the linearized
 data
    r2 = r2_2;
    plot(Km_measured, price_measured,'k.') %plot the original data
    hold on;
    plot(Km_measured, predict2,'r-'); %plot the model
    title({'The Ezyme USD Price Per Pound of each';'Michaelis Constant
 from 157-350'})
    xlabel('Michaelis Constant (uM)');
    ylabel('Price (USD($)/lb)')
    function_output = sprintf("Model function:Price(USD($)/
lb) = %.3f * Km(uM) ^ (%.3f)\nSSE: %.3f, SST: %.3f, r^2:
 %.3f",b2,m3,SSE_final,SST,r2); %display the funciton of the model
 also the SST,SSE and r^2 value
    legend('Novel Enzymes Price Catalog per Michaelis Constant',
 function_output, 'location', 'best')
    grid on
    %assign return parameters for the exec function
    bf = b2;
    mf = m2;

elseif choose == 3 % if the exponential function best fits the data
    SST = sum((log10(price_measured) -
 mean(log10(price_measured))).^2); %calculate the SST value of the
 linearized data
    r2_3 = 1-(SSE3/SST); %calculate the r^2 value of the linearized
 data
    r2 = r2_3;
    plot(Km_measured, price_measured,'k.') %plot the original data
    hold on;
    plot(Km_measured, predict3,'r-'); %plot the model
    title({'The Ezyme USD Price Per Pound of each';'Michaelis Constant
 from 157-350'})
    xlabel('Michaelis Constant (uM)');
    ylabel('Price (USD($)/lb)')
```

```
   function_output = sprintf("Model function: Price(USD($)/lb) =
 %.3f * 10 ^(^%.3f ^* ^K^m^(^u^M^)^) \nSSE: %.3f, SST: %.3f, r^2:
 %.3f",b3,m3,SSE_final,SST,r2); %display the funciton of the model
 also the SST,SSE and r^2 value
   legend('Novel Enzymes Price Catalog per Michaelis Constant',
 function_output, 'location', 'best')
   grid on
   %assign return parameters for the exec function
   bf = b3;
   mf = m3;

elseif choose == 4 % if the power logarithmic best fits the data
   SST = sum((price_measured - mean(price_measured)).^2); %calculate
 the SST value of the linearized data
   r2_4 = 1-(SSE4/SST); %calculate the r^2 value of the linearized
 data
   r2 = r2_4;
   plot(Km_measured, price_measured,'k.') %plot the original data
   hold on;
   plot(Km_measured, predict4,'r-'); %plot the model
   title({'The Ezyme USD Price Per Pound of each';'Michaelis Constant
 from 157-350'})
   xlabel('Michaelis Constant (uM)');
   ylabel('Price (USD($)/lb)')
   function_output = sprintf("Model function:Price(USD($)/
lb)= %.3f * log10(Km(uM)) + %.3f\nSSE: %.3f, SST: %.3f, r^2:
 %.3f",b4,m4,SSE_final,SST,r2); %display the funciton of the model
 also the SST,SSE and r^2 value
   legend('Novel Enzymes Price Catalog per Michaelis Constant',
 function_output,'location', 'best')
   grid on
   %assign return parameters for the exec function
   bf = b4;
   mf = m4;

end
```

COMMAND WINDOW OUTPUT

```
fprintf("Pricing Function for Enzymes (Exponential):
\nPrice(USD($)/lb) = %.3f * 10 ^ (%.3f * Km(uM))\n\nGoodness
 of fit for pricing function:\nSSE: %.3f  |  SST: %.3f  |  r^2:
 %.3f",b3,m3,SSE_final,SST,r2);
```

ACADEMIC INTEGRITY STATEMENT

We have not used source code obtained from any other unauthorized source, either modified or unmodified. Neither have we provided access to my code to another. The function we are submitting is our own original work.

Published with MATLAB® R2019a

Table of Contents

```
function M3_exec_001_30

tic
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
% this executive funciton will use the algorithm to automatically
 calculate
% vmax and km parameters for each enzyme using the v0 calculated
 values from
% the algorithm. Additionally, the SSE values are calculated between
 the ideal
% and expected Michales-menten curve in order to judge goodness of
 fit.
% Finally, this function will price all the emzymes. The parameters
 and the
% recommended price are printed to the command window in a neat
 manner.
%
% Note: changed or depreciated code is commented as such. New or
 unmodified
% code will remain uncommented.
%
% Function Call
% M3_exec_001_30();
%
% Input Arguments
% N/A
%
% Output Arguments
% N/A
%
% Assignment Information
%   Assignment:     Milestone 3
%   Team member:    Surya Manikhandan, smanikha@purdue.edu
%                   Julius Mesa, jmesa@purdue.edu
%                   Alex Norkus, anorkus@purdue.edu
%                   Luming Lin, lin971@purdue.edu
%   Team ID:        001-30
%   Academic Integrity:
%     [] We worked with one or more peers but our collaboration
%        maintained academic integrity.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INITIALIZATION

```
% figureNumber = 1; % holds the number of the figure window being
 plotted [General Change - no need to keep track of figure windows
 anymore as all plots have been eliminated]

fileName = "Data_nextGen_KEtesting_allresults.csv"; % the name of the
 datafile

% import the [S] values used to produce the plots
%substrateData = readmatrix(fileName, "range", "B3:CW3"); [Change -
 xlsread is significantly faster than readatrix]
% Category 2 - using xlsread for small data imports saves time as
 readmatrix is only good at large dataset imports
substrateData = xlsread(fileName, "B3:CW3");
```

USE REGRESSION ALGORITHM TO GET ENZYME FUNCTION

```
[b, m, SSE, SST] = M3_Regression_001_30; % call regression function to
 get function constants
fprintf("\n\n"); % move to next line for aesthetics
```

CALCULATIONS OF MICHALES-MENTEN CONSTANTS & PRICE ENZYMES

```
for enzymeNum = 1:5 % loop through all enzymes

    % use algorithm to find the v0 values
    v0Vals = M3_Algorithm_001_30(enzymeNum);

    % find the starting column of [S] values for the given enzyme
    sColumn = 11 + (20 * (enzymeNum - 1));
    % compute [S] for reaction velocity plots
    sData = substrateData(sColumn : sColumn + 9);

    % linearize [S] and v values for the linear regression through
 Hanes-Wolf Method (explanation can be found in M2 exec function)
    linearSData = sData; % [s] vals do not need transformation to be
 linear
    linearV0Array = linearSData ./ v0Vals;

    % find a regression line
    linearCoeffs = polyfit(linearSData, linearV0Array, 1);

    % seperate the slope and the intercept from the data
    linearSlope = linearCoeffs(1);
    linearYIntercept = linearCoeffs(2);

    % calc Vmax and Km values through the Hanes-Wolf Method
 (explanation can be found in M2 exec function)
    % general change - In the previous M2 submission, kM was
 calculated
    % through the use of the y-intercept value, which was equal to
 vmax/km.
    % However, using this method to solve for the kM means that any
 errors
    % in the vmax calculation will compund over to the kM calculation.
    % Therefore, we now based our calculations off the x intercept.
    vMax = 1 / linearSlope;
    kM = linearYIntercept / linearSlope;

    % calculate SSE values between expected and actual values for raw
 data
    idealV = (vMax .* sData) ./ (kM + sData); % calculate ideal v0
 vals using Michaelis-Menten equation
    SSE = sum((idealV - v0Vals) .^ 2);

    % Price the enzyme (exponential equation)
    recPrice = b * 10^(m * kM);

    % display the values to the command window in a neat manner
    fprintf("----------------------------------------------\n");
    fprintf("Parameters for Enzyme NextGen-%c:\n\nv0 values (uM/
min):", 'A' + (enzymeNum-1))
    disp(v0Vals);
    fprintf("Vmax (uM/min): %.4f  |  Km (uM): %.4f  |  SSE: %.5f \n
\nRecommended Price: $%.2f per lb\n", vMax, kM, SSE, recPrice);

    % --- ALL CODE BELOW IN THIS SECTION IS DEPRECIATED ---
    % Add params to the output variables [General Change - executive
 function no longer has parameters, this step is unnecessary]
    % vMaxArray = [vMaxArray, vMax];
    % kSubMArray = [kSubMArray, kM];
    % sseArray = [sseArray, SSE];
```

```
   % [General Change - eliminate all plots because it is not
necessary for this assignment and provides a large speed boost
(Category 2)]
   % figure displays
   % figure(figureNumber);

   % First, plot the raw function output for reaction velocities
   % subplot(1, 2, 1);
   % plot(sData, v0Vals, 'ro');
   % xlabel("Substrate Concentration [S] (uM)");
   % ylabel("Reaction Velocity v (uM/min)");
   % title("Unaltered Data");
   % grid on
   % hold on
   % overlay the ideal reaction velocity vector
   % plot(sData, idealV, "-b");
   % legend("Raw Reaction Velocity", "Michaelis-Menten Expected
Vector", "location", "south");
   % hold off


   % Next, plot the linearized data according to Hanes-Wolf method
   % subplot(1, 2, 2);
   % plot(linearSData, sData ./ v0Vals, 'ro');
   % xlabel("Linearized Substrate Concentration");
   % ylabel("Linearized Reaction Velocity");
   % title("Linearized Data");
   % grid on
   % hold on
   % overlay the linear regression on to that line
   % plot(linearSData, (linearSlope * linearSData) +
linearYIntercept, '-b');
   % legend("Linerized Reaction Velocity", "Hanes-Wolf Best Fit
Line", "location", "north");
   % hold off
   % title the overall figure and increment figure number
   % figureTitle = sprintf("Reaction Velocity Plots for Enzyme %d
(NextGen-%c)", enzymeNum, ('A' + (enzymeNum - 1)));
   % sgtitle(figureTitle);
   % figureNumber = figureNumber + 1;

end
```

FORMATTED TEXT/FIGURE DISPLAYS

COMMAND WINDOW OUTPUT

ACADEMIC INTEGRITY STATEMENT

We have not used source code obtained from any other unauthorized source, either modified or unmodified. Neither have we provided access to my code to another. The function we are submitting is our own original work.

```
toc
```

```
end
```

Published with MATLAB® R2019a

Table of Contents

```
function v0Vals = M3_Algorithm_001_30(enzNum);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
% This program will use a method similar to Algorithm 1 of M2 in order
% to calculate v0 constants for a given enzyme at 10 different
 substrate
% concentrations.
%
% Note: changed or depreciated code is commented as such. New or
 unmodified
% code will remain uncommented.
%
% Function Call
% v0Vals = M3_Algorithm_001_30(enzNum);
%
% Input Arguments
% enzNum - the number of the enzyme being examined as integer 1 - 5.
 Other
% values will throw an error.
%
% Output Arguments
% v0vals - an array containing all 10 v0 values for a given enzyme at
 each
% substrate concentration. (Units: uM/minute)
%
% Assignment Information
%   Assignment:     Milestone 3, Algorithm
%   Team member:    Surya Manikhandan, smanikha@purdue.edu
%                   Julius Mesa, jmesa@purdue.edu
%                   Alex Norkus, anorkus@purdue.edu
%                   Luming Lin, lin971@purdue.edu
%   Team ID:        001-30
%   Academic Integrity:
%     [] We worked with one or more peers but our collaboration
%        maintained academic integrity.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INPUT VALIDATION

```
% --- ALL CODE BELOW IN THIS SECTION IS DEPRECIATED ---
% CHANGE: Input validation has been removed as Algorithm is not meant
 to be used
% by other users and is only implemented in the executive function.
 Since
% the values for enzyme number parameter is set properly, we have no
 need
% for input validation

% Category 2 - By removing the unnecessary input validation step, the
% program runtime is shortened due to the reduction of CPU cycles
 spent on
% comparisons. However, this means we will need to be more
% careful and ensure proper parameter passes when using this function

% inval = 0; % this flag value will hold whether or not any of the
 params are invalid

% if((floor(enzNum) ~= enzNum) | (enzNum > 5) | (enzNum < 0)) % check
 if width is a positive integer
%    fprintf(2, "ERROR: enzNum parameter must be an integer between
 1-5 inclusive\n");
%    inval = 1; % toggle flag
% end

% if(inval) % quit if any parameter is invalid
%    return;
% end
```

INITIALIZATION

```
fileName = "Data_nextGen_KEtesting_allresults.csv"; % the name of the
 datafile

% import data vals crucial to the calculation
timeAxis = readmatrix(fileName, "range", "A4:A");
concentrationData = readmatrix(fileName, "range", "B5:CW7488");


% compute the starting column of the data given the enzyme number
origColumn = 1 + (20 * (enzNum - 1));
dupeColumn = 11 + (20 * (enzNum - 1));
```

CALCULATIONS OF v0 VALUES

```
passWidth = 17; % width for the moving average smoothing function

% v0ValsTest = []; % initialize v0s to void array to begin (for orig
 test)
% v0ValsDupe = []; % initialize v0s to void array to begin (for dupe
 test)

% [Category 2 Change - Previously, the array was not preallocated with
 zeros, which causes
% significant performance losses when adding elaments to the array.
 Therefore,
% we are preallocating the arays with zeros to increase performance]
v0ValsTest = zeros();
v0ValsDupe = zeros();

for product = 0:9

    % Code in next 5 lines will isolate test and dupe data without NAN
 vals
    % NOTE: BASED OFF CODE FOUND IN https://bit.ly/2OyqAQj
    testData = concentrationData(:, origColumn + product)';
    testData = testData(~isnan(testData))';

    dupeData = concentrationData(:, dupeColumn + product)';
    dupeData = dupeData(~isnan(dupeData))';

    % Find v0 values for original test and duplicate seperately. Then,
    % average those v0 values to get the final result.

    % __________ Original test __________
    [timeArray, dataArray] = M3_Smooth_001_30(testData, timeAxis,
 passWidth);

    % Category 1 Change - In M2, we used the first two points in the
    % reaction velocity plot and used the rise/run equation to finf
 the
    % vmax. However, we found that this does NOT create the best
 tangent
    % line. Instead of only considering the first two points, which
 leaves
    % room for error, we found that finding the linear regression of
 the
    % first 3 points created a much better tangent line and therefore
    % better vmax values. Comprehensive explanation located in M4
 docs.
    % v0 = (dataArray(2) - dataArray(1))/ (timeArray(2) -
 timeArray(1));
```

```
   timeArray = timeArray(1:3);
   dataArray = dataArray(1:3);
   v0 = timeArray(:) \ dataArray(:); % find slope, which is equal to
v0 value

   % [Category 2 Change - Previously, we added the element to the end
of the array using
   % concatenation, which is not effecient. Now using indexes instead
for direct adding of v0 values where needed]
   %v0ValsTest = [v0ValsTest, v0];
   v0ValsTest(product + 1) = v0; % General change - fit new array
scheme (use direct assignment)

   % __________ Duplicate test __________
   % NOTE: changes implemented here are identical to original test
changes
   [timeArray, dataArray] = M3_Smooth_001_30(dupeData, timeAxis,
passWidth);

   % [OLD] use rise/run to find the first slope value which is our v0
val
   %v0 = (dataArray(2) - dataArray(1))/ (timeArray(2) -
timeArray(1));

   timeArray = timeArray(1:3);
   dataArray = dataArray(1:3);
   v0 = timeArray(:) \ dataArray(:); % find slope, which is equal to
v0 value

   %v0ValsDupe = [v0ValsDupe, v0];
   v0ValsDupe(product + 1) = v0; % fit new array scheme (use direct
assignment)

 end
```

FORMATTED TEXT/FIGURE DISPLAYS

COMMAND WINDOW OUTPUT

```
% final v0 array is the average of the test and dulplicate data
v0Vals = (v0ValsTest + v0ValsDupe) ./ 2;
```

ACADEMIC INTEGRITY STATEMENT

We have not used source code obtained from any other unauthorized source, either modified or unmodified. Neither have we provided access to my code to another. The function we are submitting is our own original work.

```
end
```

Published with MATLAB® R2019a

Table of Contents

```
function [truncatedTime, smoothedData] = M3_Smooth_001_30(dataArray,
 timeArray, segmentWidth);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ENGR 132
% Program Description
% This user-defined functin will use the moving average method to
 smooth
% an array of data and return the smoothed array back to the calling
% function. Averages are done sequentially with the width given by
 user.
%
% Note: changed or depreciated code is commented as such. New or
 unmodified
% code will remain uncommented.
%
% Function Call
% [truncatedTime, smoothedData] = M3_Smooth_001_30(dataArray,
 timeArray, passWidth);
%
% Input Arguments
% dataArray -    a one dimensional array containing the data for the
 product
%                conc. data of an enzyme at a given substrate conc.
 value.
% timeArray -    the time data array. Will be returned at an
 appropriate length.
% segmentWidth - the width the function will use to calculate the
 moving
%                average.
%
% Output Arguments
% truncatedTime - the array of the time elements corresponding to
 smoothed
```

```
%                  data values.
% smoothedData - the array of smoothed data determined through the
 moving
%                 average method.
%
% Assignment Information
%   Assignment:     Milestone 3
%   Team member:    Surya Manikhandan, smanikha@purdue.edu
%                   Julius Mesa, jmesa@purdue.edu
%                   Alex Norkus, anorkus@purdue.edu
%                   Luming Lin, lin971@purdue.edu
%   Team ID:        001-30
%   Academic Integrity:
%     [] We worked with one or more peers but our collaboration
%        maintained academic integrity.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

INPUT VALIDATION

```
% --- ALL CODE BELOW IN THIS SECTION IS DEPRECIATED ---
% CHANGE: Input validation has been removed as Smooth is not meant to
 be used
% by other users and is only implemented in the algorithm. Since
% the values for width parameter is set properly, we have no need for
 this.

% Category 2 - By removing the unnecessary input validation step, the
% program runtime is shortened due to the reduction of CPU cycles
 spent on
% comparisons. However, this means we will need to be more
% careful and ensure proper parameter passes when using this function

% for input validation
% inval = 0; % this flag value will hold whether or not any of the
 params are invalid

% if((floor(segmentWidth) ~= segmentWidth) | (segmentWidth <= 0)) %
 check if width is a positive integer
%    fprintf(2, "ERROR: passWidth parameter must be an integer greater
 than zero\n");
%    inval = 1; % toggle flag
% end

% if(inval) % quit if any parameter is invalid
%    return;
% end
```

INITIALIZATION

CALCULATIONS

```
% general change - parameters do not need to be initialized
% void anymore as we will do that later with the zeros function
% smoothedData = [];
% truncatedTime = [];

% [Category 2 Change - Previously, the array was not preallocated with
 zeros, which causes
% significant performance losses when adding elaments to the array.
 Therefore,
% we are preallocating the arays with zeros to increase performance]
smoothedData = zeros();
truncatedTime = zeros();

arrayindex = 1; % will keep track of the array index

for index = 1:ceil(segmentWidth/2):(length(dataArray) - segmentWidth)

    % isolate segment of width from dataset
    dataSegment = dataArray(index : index+segmentWidth);
    timeSegment = timeArray(index: index+segmentWidth);

    % sum all elements in the segment
    sumDataSegment = sum(dataSegment);
    sunTimeSegment = sum(timeSegment);

    % take average of elements in that array
    avgDataSegment = sumDataSegment / (segmentWidth + 1);
    avgTimeSegment = sunTimeSegment / (segmentWidth + 1);

    % [Category 2 Change - Previously, we added the element to the end
 of the array using
    % concatenation, which is not effecient. Now using indexes instead
 for direct adding of elements]

    % add the averaged value to the smoothed array
    %smoothedData = [smoothedData, avgDataSegment];
    %truncatedTime = [truncatedTime, avgTimeSegment];

    % general change - fit new array scheme (direct assignment)
    smoothedData(arrayindex) = avgDataSegment;
    truncatedTime(arrayindex) = avgTimeSegment;
    arrayindex = arrayindex + 1; % increment the index of the array
end
```

FORMATTED TEXT/FIGURE DISPLAYS

COMMAND WINDOW OUTPUT

ACADEMIC INTEGRITY STATEMENT

We have not used source code obtained from any other unauthorized source, either modified or unmodified. Neither have we provided access to my code to another. The function we are submitting is our own original work.

```
end
```

Published with MATLAB® R2019a