Project Milestone 2 – Algorithm Development

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
M2 Answer Sheet	M2_AnswerSheet_*SSS_TT*.pdf
M2 Algorithm	M2_Algorithm_*SSS_TT*.m

   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):	<019-24>

Team Member Name	**Purdue Career Account Login**
Gregory Szymchack	gszymcha
Sergio Monge	smonge
Seena Pourzand	spourzand
Nathan Thorson	njthorso

Role of Each Team Member

In this section, put each team member's name who worked on this milestone. In the Detailed Description of Work, each person on the team should write their own description of how they contributed to this milestone. Be very detailed here. Then in the last column, your team should estimate the percentage of the work that each team member did on the milestone. This column needs to add up to 100%. We know that on any given milestone that this will vary, but one person in the team should not be doing significantly more than the others throughout the whole project. Use this column as a way for you to make sure your workload is balanced throughout the project.

Team Member Name	**Detailed Description of Work**	**Percent of Work**
Gregory Szymchack	Worked on the v(0) estimation and created the figure creation for the main function. Helped with debugging code. Helped on the answer sheet.	25%

Sergio Monge	Revision and debugging of code. Helped on the vMax and Km section, revising the process used and any additional problems. Helped on the answer sheet.	25%
Seena Pourzand	Lead the primary effort on creating the UDF, worked on both v0 estimates and finding vMax and Km. Additionally helped throughout answer sheet.	25%
Nathan Thorson	Revision and de-bugging of code, testing breaking points and the conciseness of code.	25%

Part 0: M1 Feedback Review

Reflect on your M1 feedback for the purpose of improvement. Your reflection should provide a clear, useful summary of your M1 feedback and provide a clear and practical plan to address the issues. Complete table 1 below.

Table 1. Feedback summary and plan

Part A: Based on your feedback from M1, identify at least one strength and one limitation of your team's approach or process you created in M1. Consider how the feedback from M1 could lead to improvements in your work. Our team looked at the comments for each section of our earlier assignment. One of the strengths of our approach is that there should not be any part throughout the code that is too difficult to calculate. All our steps are straightforward. On the other hand, one of our limitations is that we are assuming that certain MATLAB functions (smoothing ones) should work well with our data, but we are not certain about this. Some questions we were given to consider regarding our approach for V0s was "Is this method robust or prone to error? Would this approach hold up given randomness/noise in the data? How will you deal with this? Does changing where the points are affect the accuracy of the result? Can you take advantage of this?" These questions do raise valid concerns that we did not consider initially of how taking the slope of only the first two points could be prone to error even if we smooth the data/reduce noise as it is not inclusive of the rest of the data near zero as there are so many data points.
Part B: Explain how you will incorporate the M1 feedback to improve your parameter identification (do not just reword your response from Part A, include concrete actions you will take to improve). The main concern from our M1 feedback seems to be how our smoothing functions will work. Our approach will begin by testing one of the MATLAB functions first and seeing how the data is affected by this. The function we are using first is smoothdata(). If this function does not work, we will investigate other functions such as smooth() or if there is any need of using even more functions such as rmoutliers(), filter(), etc. This process is easy on paper, since it is a process simply based on testing,

analyzing our results, and depending on the results work from there. Having said that, we do expect there to be unexpected new problems as we go through all our options.

As for the method of determining v0, like we said for smoothing, this will involve testing different methods of attempting to estimate v0 and see how well each approach works.

Part 1: Algorithm Development

In Milestone 1, you developed approaches for identifying v_{0_i} and approaches for identifying V_{max} and K_m. Now, you will take your best ideas from M1 and use them to create a single algorithm that is coded in MATLAB to perform parameter identification on enzyme reaction data.

You must develop a plan for your algorithm before you start coding. Outline your algorithm using pseudocode (i.e., plain English text, not MATLAB code). Remember, it is valuable to develop and organize your programming ideas and solutions *before* you code. A well-developed plan reduces coding frustrations.

As you develop your function parameter identification plans and algorithm, keep in mind that it will be called in future milestones by a main function. This main function will load your data, assign the correct variables, and call your user-defined function parameter identification algorithm to determine the parameters for the data. Use this description of the main function to help you determine appropriate inputs and outputs for your algorithm.

Complete your plans in Table 2 below.

Table 2. Algorithm plans

Plan for Algorithm
Our plan for our UDF is to pass in two parameters. The first being a vector containing the time column vector and the second being a matrix containing both the original tests and duplicate tests for an enzyme. The basic idea is each time the function is called, it is meant to perform analysis and find the V(max) and K(m) of the passed in enzyme. The three outputs of our function will be the V(max), K(m) and a vector containing the 10 v(0)s determined for the enzyme that maybe serve a purpose in the future. The way we find the v0s is by first smoothing the data by using MATLAB's built-in functions (we will have to test to see which one works best, i.e. smooth, filter,etc) and then finding the average slope between two adjacent data points and repeat that process for the first 100 data points. We then have 100 slopes for our current concentration. We then average those 100 into 1 slope to represent the current concentration. We repeat this process for the duplicate test and average between the first and duplicate test to get one final slope for that concentration. We repeat this process for every concentration for our given enzyme, resulting in a final vector of v0s that contains 10 elements. We then use those v0s to create a regression line that will help use determine Vmax and Km. Following the steps outlined by Prof.Branco for a Lineweaver-Burke plot we linearize the data, find a regression line for the linearized data and then algebraically alter the linear regression line coefficients to create a non-linear regression line to represent the data of our Michaelis-Menten plot. Once finding the values of our non-linear regression line, we can calculate our Vmax and Km from it.

Finally, we then assign the values of the 10 v0s, vMax, and Km to the output variables that are specified by the user and output them.

After you complete your plan, <u>translate your plan into a user-defined function</u>. Use the filename **M2_Algorithm_SSS_tt.m**. Make sure that your algorithm follows ENGR 132 Programming Standards and is clearly commented throughout.

Part 2: Algorithm Reflection

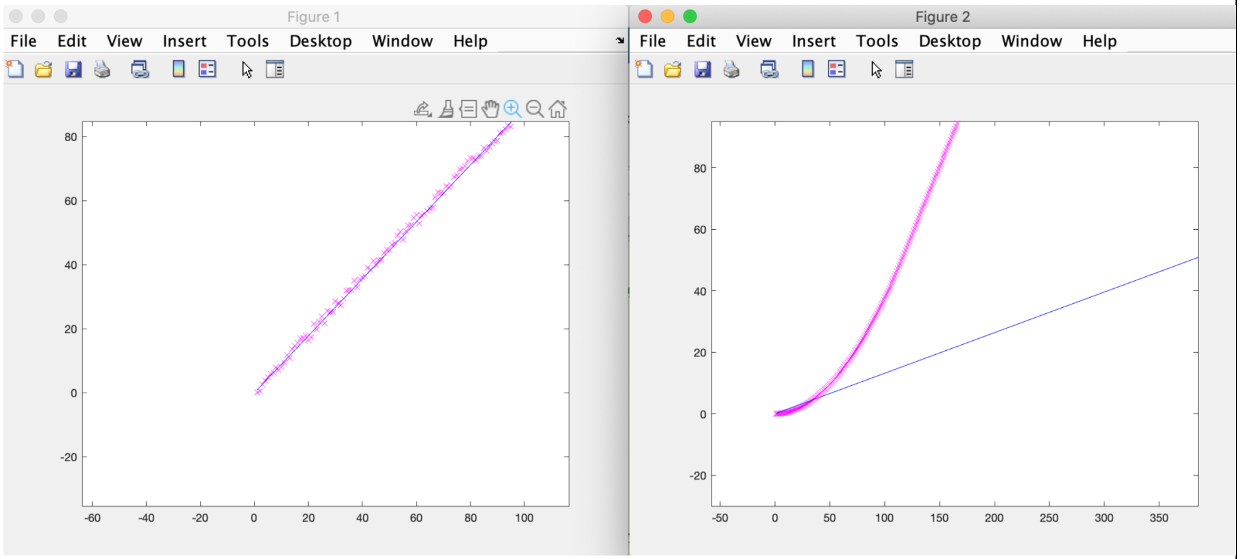
In Table 3, discuss your choice of algorithm, your process for debugging your algorithm, and the strengths and limitations of your algorithm at this point. See the directions in each part of the table.

Table 3. Algorithm reflections

Your choice of your algorithm in M2
Describe your process for choosing how you would develop your algorithm? How did you use your data in this process? Initially, our process of choosing how to develop our algorithm was influenced by our answers in M1 and feedback we received on it. We also looked at the data to see what we would need from the large dataset to be passed in as a parameter and what we would not need to pass in. We came to the conclusion our algorithm would need to have a matrix containing the 10 original tests and 10 duplicate tests for an enzyme that a user selects. For example, if the user wanted to use the UDF for enzyme A, they would pass in a matrix containing all of the tests (duplicates included) for enzyme A, that being from row 4: to the last row and column 2 to column 21 or in MATLAB terms dataset(4:end,2:21). We start from row 4 as everything up until this is Null values when you read in the data and we just want to pass in the concentrations only. 2:21 covers from original test 1 with concentration 3.75 microMolarity to duplicate test 10 with concentration 2000 microMolarity. We also decided to pass a vector representing time, which happens to be the first column of the dataset. We realized this could be hardcoded in the UDF but we felt allowing the option to change the time vector could be beneficial if they want to only analyze the first 1000 datapoints, they could specify that length in both the time vector and enzyme matrix length, something that would not be possible if it was hardcoded.
Debugging your algorithm in M2
Describe your process for making sure your algorithm is meeting the needs of the client and running smoothly. What did you do to debug your algorithm? How did you use your data in this process? To ensure that our group was meeting the client's needs and running smoothly, we had a drafting process, a logical analysis of what the outcome of our code should look like, revision of our M1 answer sheet and team revisions. We quickly realized that some of our initial attempts were flawed in their design when attempting to make our UDF so we had to alter our algorithm plans accordingly. First was the issue we encountered

with finding the v0s. We had initially planned to only use the first and second values in the dataset to find the v0. But when we attempted this, even after smoothing the data, we found it did not account for enough data values as it did not create a tangent line that was tangent to the actual data. We soon determined that we would need to include more datapoints not just the first two points. So once doing that we began getting v0s that accurately fit to the data. The debugging process including adding several breaking points in our code to analyze the looping process and plotting process, coming to logical conclusions that are un related to the mathematical outcome of our code, and determining what the outcome of our code should look like before going through the coding process.

Smoothing was another aspect that we spent a long time debugging. Initially, we attempted to smooth our data with smooth() & smoothdata() functions, both provided by MATLAB's curve fitting toolbox. However, upon using these functions they wildly distorted our data as we quickly decided to look towards other options. Then we began using a function called filter() where you pass in a variety of parameters to change the number of values in which you captured in the window of the function, the larger the window the more smoothed the data becomes. For a while we used filter() until we realized it was warping our data and causing incorrect v0 calculations. In the screenshot below you can see how filter was changing our data and interpolating points , changing the overall change of the graph at the very beginning of our data(left being unfiltered/normal and right being filtered). This curve caused incorrect calculations for v0 so we scrapped filter() and looked towards using movemean(). Movemean() struck the balance of smoothing our data enough where it wasn't causing issue calculating our desired values while also not causing issues in calculations by warping/smoothing the data too much. It took a moving average our data so it kept the same overall shape of our data while reducing outliers and jagged parts of our graphs.



When it came to our regression line, we also had to make some adjustments to our original approach. Initially, we planned on calculating Km by plugging in our corresponding substrate concentrations and velocities to find an approximation of Km. However, after class 13A and 13B, where Professor Branco introduced the idea of using linearization to help us obtain an accurate regression line, and how that regression line can be easily used to find Km and Vmax, we as a team decided to shift our approach to that method. We found more accurate results this way, when utilizing a line weaver burke method of linearization and then creating a regression line as opposed to our previous approach. We also used the regression line to solve for Vmax, as opposed to our initial approach of simply using the largest velocity, as both methods yielded similar results. However, we still opted to use the regression line method for Vmax as it was slightly better in our tests and was consistent with how we obtained Km.

Strengths and limitations to your algorithm in M2

Identify at least one strength and one limitation of your team's algorithm you created in M2.

One strength of our algorithm is how well our resulting non-linear regression model (that was used to compute our kM and Vmax) fits to the data. We decided to determine the R Squared for our data just so we could see how well our model fit numerically speaking and we got values of 0.9993, 0.9971, 0.9798, 0.9934, and 0.9995 for NextGen Enzymes A-E.

We also consider that our model has a high level of consistency in our processes. We consider that although there were several ways of calculating the parameters, it was important to stick to a certain process to minimize any errors caused by a change of approach. We felt it was best to computer our parameters (Km & Vmax) both through the regression line as opposed to have different methods for each one.

Finally, we also think that our approach to create our final algorithm took a lot of work in simply testing and comparing results. We compared various approaches and their accuracy so that we could choose what we thought was the most accurate method. For instance, we managed to have two working algorithms that calculated V0 values. We were able to see that one of the two approaches seemed to be more accurate, which makes us feel confident that the one we chose is the best method we could think of.

On the other hand, there are also limitations to our approach. One of the main ones is that we used the Lineweaver-Burk method on our Vmax and Km. It is a known error for this method that due to the fact the Y-axis is the reciprocal of the rate of reaction, any small error is incremented. Additionally, the calculation error is only shown in the dependent variable and least accurate values occur at the high end of the curve. These sorts of limitation could prove potentially harmful to our algorithm in the future in terms of accuracy but as of right now we cannot say definitively as we have no rubric to grade our results against.

Another limitation is the rigid guidelines in which the parameters must be given. If the enzyme matrix given are not 20 columns, the algorithm will likely produce inaccurate results. Also, if the values of the provided enzyme matrix are not the product concentration and instead a NaN values, the algorithm will likely crash.

Additionally, our decision to average the first 100 slopes of the data in our v0 identification could be a limitation depending on the dataset entered. We chose 100 as it felt appropriate given the sheer number of data points in the given csv file that it would get a good estimate of the beginning of the data where it would capture the approximate initial velocity. However, if a user were to input a dataset with let's say 150-200 datapoints (as opposed to ours that has 7000+), the slope calculated would no longer be that of initial velocity but would be the average velocity for half or more of the function which is not what we would want. Maybe a solution to this would be the number of datapoints used in the v0 calculation to be based on a percentage of the total datapoints.

Lastly, a possible limitation could be the decision to hardcode the substrate concentrations in our UDF as opposed to passing them in as a parameter. This change would be very easy to make but we opted to have them predefined in the UDF as every enzyme test shared the same substrate concentrations and we felt that since our client, NaturalCatalysts, asked us to analyze this data in particular, it would not pose a problem. However, if NaturalCatalysts were to use the UDF on other datasets that perhaps had different substrate concentrations, our hardcoding could pose a problem. But granted the scope and dataset we were given, we felt it was appropriate and believed it would reduce the number of parameters that would need to be passed in, especially as it would need to be repeat for each and every function call.

How to Submit

1. Save this answer sheet as a PDF named **M2_AnswerSheet_SSS_*TT*.pdf.**
2. Select one person to submit the assignment for the team. That person should
    a. Log into Gradescope and submit **M2_AnswerSheet_SSS_*TT*.pdf** and **M2_Algorithm_*SSS_TT*.m** to the **M2** assignment.
    b. Select all team members for the group assignment.
    c. Double-check that all team members are assigned to the submission.
3. Each team member should confirm that they are part of the submission and that all parts of the answer sheet were properly tagged.
4. After submission, distribute the submitted files to all team members. *Ensure all members of the team have copies of the submitted files.*

Learning Objectives

Teamwork (TW)
Contribute to team products and discussions
TW02. Document all contributions to the team performance with evidence that these contributions are significant.

Process Awareness (PA)
Reflect on both personal and team's problem solving/design approach and process for the purpose of continuous improvement.
PA01. Identify strengths in problem solving/design approach.
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.
EB03. Clearly articulate reasons for answers with explicit reference to data to justify decisions or to evaluate 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.

Engineering Professional Skills
PC05. Fully address all parts of assignment by following instructions and completing all work.
EPS01. Use professional written and oral communication.

Programming
MAT01. Develop code that follows good programming standards
MAT08. Debug scripts and functions to ensure programs execute properly, perform all required tasks, and produce expected results.