When approaching this project, we wanted to improve the speed of a process that is used to solve nearly every problem in our industry. Chemical Engineers work to improve and optimize chemical process that are in both everyday life such as water treatment and more specialized such as nuclear power plants. While optimizing these processes we are often presented with a reactor or multiple that have a chemical reaction occurring inside. In order to determine the deficiencies in the process we must first convert multiple inputs or outputs from moles. A mole is a universal unit for the number of particles in a substance, this may be atoms, molecules or ions. In one mole there are 6.022 x 1023 particles. Converting to mass allows for the outcome or origin to be determined from its opposite. Most often, the amount of each reactant or product is presented as mass or mole fractions of the entire substance. Converting mole fractions to mass fractions or vice versa is a simple yet tedious process that leaves room for human error and can take a large amount of time depending on the number of reactants and products. To convert from mole fractions to mass fractions one must first multiply the fraction by the molar mass of each reactant. This gives the grams (or another unit of measurement) of each reactant or product. Following this step, each element must be converted to mass fractions. To do this, the grams are divided by the total mass. To convert mass fractions to mole fractions the process above is similar. As you can see, the process is long and there are many steps that mistakes can be made. For our GUI project we decided to design a program that will take in both mole fraction and molar mass, then convert them to the other. This program will both speed up the trivial step and will eliminate much of the human error that can occur.

With the steps of the process being pretty straightforward, the structure of the code was able to be loosely determined by breaking functions down into each step. First, the molar fraction and molar mass are required for the next steps. To allow more general coding, we designated each part with letters A to H instead of writing explicit components. This is to avoid lack of application in many different conditions, the user will just need to be aware of which is which. Also, if there are to be less than eight components, the user just needs to leave all unused fields as zero. Once these are inputted, the fraction and molar mass are multiplied in order to find the mass of each component. A basis of 1 mole allows the program to keep the molar fractions as a decimal, whereas if we set a basis of 100 moles, the fractions would look like percentages. Although it may look more appealing, doing so causes more work to be done in our later calculations as chemical engineering majors. The masses are then added and the fraction for each is found.

For the functions that require the user to input values, there were not many obstacles. Slight difficulty came with finding the total of the mole fraction column, but once it was figure out, we were able to then implement some error notices. Inputs that would not allow calculations to be run correctly include values less than 0 or nonnumeric inputs. There are notices in place, including an error popup and a general box under the column that prompts the user when they have correctly entered their numbers. Overall, the total must equal 1.00 for the values to be correct and applicable in real life applications.

The true difficulties of this code occurred when the inputted values needed to be multiplied in order to find the mass of each component. Because we had already used them in a different callback, help was sought out from LA’s and the instructor after research into the issue was of no assistance. Five separate LA office hours were attended for this program, but sadly each one did not help us resolve the problems that were occurring. It seemed as if the issue we were running into was one that they had never seen or were not sure where for the program was encountering obstacles. We tried to use two callbacks for one component, but when that was done, previous functions were no longer running correctly. Finally, one of us was able to meet with Quinn in her office hours and we were able to fix the issue, along with completely finishing the code with some assistance. In hindsight, what we thought would be a pretty straightforward and simple task ended up being surprisingly difficult and, in Quinn’s words, “a very ambitious project.” However, as a result, our understanding of GUI and its applications in MATLAB increased greatly. Additionally, we now have more knowledge of how to manipulate GUI’s many capabilities and troubleshoot in ways that were not discussed in class.

GUI programming can be very useful in our industry. Many issues in the Chemical industry can be solved through multiple simple steps and calculations. While it is necessary for an engineer to decide the order that these steps are completed in order to ensure the correct outcome, many of the steps could and should be completed by a computer program to reduce human error and increase workplace efficiency. GUI programs are also very user friendly due to the display options that programmers have. Some specific examples of ways we may use GUI coding in chemical engineering throughout both our undergrad experience and in industry are, developing programs to solve simple mass balances, converting ideal gas behavior at standard conditions to non-ideal behavior at nonstandard conditions and creating dimensionless groups using the Buckingham Pi method to solve problems with dimensional analysis considerations. Simple non-reactive mass balances are often done over a separator or other functions inside a chemical processor. These are simple to solve but leave room for human error much like the problem we solved in this project. Solving problems with non-ideal gases is very common in the chemical industry. It would be very useful to have a program that would link non-ideal behavior to ideal gas behavior. The Buckingham Pi method is a complex system for solving problems involving dimensional analysis. The method is complex due to the number of rules and steps. This system is solved using matrices which is ideal for GUI coding and MATLAB. The rules, while difficult for humans to remember would be no problem for a computer program. These three problems and many more could be solved using GUI coding and it would optimize much of the work done by chemical engineers on a daily basis.

Overall, GUI coding is very useful in our industry due to its easy-to-use nature and ability to solve complex problems. Our code is able to convert mass and mole fractions using an easy-to-understand interface and will help reduce the human error that often occurs in these steps. We both look forward to using our knowledge of GUI coding and MATLAB throughout our time in undergrad and in industry to help us solve problems and overall optimize our performance.