# CSCE 155N
# Spring 2018

# Final Project

Assigned: March 26, 2018
Due: April 25, 2018

## Objectives

The main objective of this final project is to put together all the skills and integrate your understanding of the concepts that you have learned about programming using Matlab during this course, to create an efficient, useful program. An important secondary objective is to for you learn to work in teams and communicate with your teammates to complete the assignment.

## Groups

We will assign groups such that each group has a group leader. The group leader will receive additional 5% bonus points of the total points received of the final project. The group leader will be responsible for managing his or her group

## Project Proposal (Due: April 02, 2018, 8:00 AM)

In order to check and approve the suitability of your project as a final project, each group is required to submit a brief project proposal by 8:00 AM April 02, 2018. A list of topics is provided at the end of this document. You are allowed to come up with your own topics. Each must be also substantially different from your Homework Assignments. Your project proposal should be no more than 3 pages and should include the following:

1. Name of the project
2. Full names, *cse* usernames and e-mails of group members
3. A description of the problem to be addressed (e.g., How this program will be useful and helpful to the users who are going to use it; how it will impact your discipline)
4. A description of the solution approach (e.g., a GUI that will receive X and generate Y, using algorithms A1, A2, A3) and the implementation design (e.g., .m files needed to be implemented, functions needed to be designed)
5. A table that clearly outlines the task allocation: the responsibilities of each group member.

It is recommended that you pick a project should have sufficient complexity that it is substantial and still do-able within the project duration. (***Important Note***: It is expected that the project is significantly more complex and demanding than a programming assignment/project since 3-4 students will be working on it.)

This proposal should be emailed to us (e-mail addresses: lksoh@cse.unl.edu, yang@cse.unl.edu, vsunkara@cse.unl.edu, zschell@cse.unl.edu ). (***Important Note***: It must be a Microsoft Word document so that we can make changes and insert comments, which will serve as a reference for us to when grading your Final Project.) You will receive an e-mail within 48 hours of the proposal submission deadline that will indicate whether your project has been accepted or it needs more modifications. If your proposal is accepted, that means that your project has

sufficient complexity and clarity. If your proposal is not accepted, you will be provided with comments and suggestions for you to modify and resubmit your proposal accordingly before **April 09, 2018**. In either case, you are strongly encouraged to start working on your project—implementing your solutions as soon as possible after you have submitted your proposal the first time. Failure to get an acceptance before April 09, 2018 will leave your group with less than 3 weeks to implement the program.

To help you develop your proposal, we have also attached a proposal sample to this document.

## Project Requirements

Your project solution should consist of at least 3 main components. Also, your solution should make use of the concepts and techniques covered in the course. Here we provide a list of the minimum requirements of your solution:

1. Your solution should include a reasonable use of ***standard I/O*** or ***GUI I/O*** in Matlab. It must either accept command line arguments specified by the user such as file name(s) and/or inputs or have a GUI that solicits inputs from the user.
2. At least ***three*** (3) ***different conditional and control statements*** should be used (e.g., if, for, while, and switch.).
3. There should be ***File I/O***. Your program must both read input from and write output to at least one file.
4. Your solution should include at least TWO of the following features:
   a) the use of a 'function' function (Chapter 10) (a function that takes another function handle as an input argument)
   b) the use of a function with variable input arguments or variable output arguments (Chapter 10)
   c) the use of an advanced plotting technique (Chapter 11)
   d) the implementation of sorting or searching (Chapter 13 and lecture notes)
5. Your solution should use ***cell arrays or structures*** in Matlab.
6. Your solution should generate at least ***one*** (1) ***informative, useful graph***.
7. Your solution should show ***an effective error handling*** by predicting the user's behavior and possible errors.
8. Your program modules must have ***sufficient and meaningful comments*** embedded in the source code so that someone reading the program can tell what is going on.
9. Your program submission should also include a **README** file that describes specifically how to run and test your program.
10. Your **TEST** file should consist of ***at least five (5) input/output pairs*** which cover common cases, special cases and handling invalid input. If your program uses file I/O then you need to provide the corresponding input/output files. The correctness of your program will be judged based on these input/output values provided.

***Important Note:*** One of the objectives of this project is to mold your ability to work in a team with other programmers; collaborate, learn, and think together. Therefore, the instructor and the

TA will provide help that is not necessarily topic-specific. Instead, they will help clarify some programmatic basics and questions that will help the team move forward and accomplish its project.

**Submission Procedure (Due: Monday, April 25, 2018)**

This project is due **on April 25, 2018 (8:00 AM).** It is highly recommended that you read the grading policy and grading guidelines on the course website for a complete explanation of how this project will be graded. Remember, your program should follow a good programming style, include plenty of comments – both inline documentation and *Matlabdoc* documentation, and perform all of the functionality outlined above.

There are **two** submission steps:

(1) Your ***group*** must "handin" the following files **on-line**:

- All source files
- **VERY IMPORTANT**: A concise, informative Readme file: **README** (See Requirement #9 above.)
- A complete testing file: **TEST**. This test file should show the different case of inputs and error handling done by your program. (See Requirement #10 above.)
- A MS Word documentation **<ProjectName>.docx** that shows how did you think and implement your program, including the justification for the choice of the function type, and different control statements. Also, different errors that you expect a user can perform, and how could you handle it.
- The original reviewed and commented **Proposal**.

(2) You ***group*** must submit your **source** files, the **README** file, **documentation** and the **TEST** file.

(3) Each individual member must submit the contributions form on cse.unl.edu/~cse155n/contributions/ webpage. On this form, you will evaluate the contribution of all other group members in your group.

All these three steps must be done before the due date.

**Important Dates:**

| Day | Time | Deadline |
|---|---|---|
| Monday, April 02, 2018 | 8:00 AM | Proposal Submission |
| Monday, April 09, 2018 | 8:00 AM | Proposal Re-Submission (If necessary) * |
| Wednesday, April 25, 2018 | 8:00 AM | Final Project Submission |

* Starting from April 2, 2018 the 3 weeks will start counting down (i.e., you will have less than 3 weeks to complete your project).

## List of Topics

Here is a list of topics, ordered by discipline. These topics might help you decide your final project. You can either select one of them or they might help you come up with ideas of a good project of your choice.

### 1. Chemical Engineering:

### Topic 1. Balancing Chemical Equations

Similar to the mathematical equations, chemical equations must be balanced. However, the chemical equation must be balanced with respect to the number of atoms of each chemical element involved in this reaction, and the electrical charge on both sides of the equation.

Assuming a non-ionic equation (i.e. the electrical charge balancing can be neglected) we become concerned with balancing the chemical equation with respect to the number of atoms.

Example: Consider the following unbalanced equation:

$$H_2 + O_2 \rightarrow H_2O$$

The purpose of this project is to take such an unbalanced chemical reaction from the user as an input, and perform gives the following balanced equation (with 4 atoms of $H$ and 2 of $O$) as an output:

$$H_2 + O_2 \rightarrow 2H_2O$$

### Topic 2. Homework Solver

In this project, you can choose three of four problems from one of your textbook. A GUI solver is to be created that gives the user an option to select the type of the problems to be solved, and then ask the user to input few useful parameters, to get the output.

### 2. Mechanical Engineering:

### Topic 1. Force Analyzer

The force exerted on a body that is located on a surface is affected by so many factors. Some of these factors are, the inclination of the surface, the nature of this surface (smooth or rough), weight of the body, friction, etc. A proper analysis might be helpful to determine the force you need to keep this body on an inclined surface so that not to slide down and hard people, or the amount of power you need to move this body upward the inclined surface. A good program would take input from users regarding these variables, and gives the user back a useful information that might help him decide the tools needed to complete this job.

### Topic 2. Closed Loop Control

Many applications involve the control of actuators which in turn control the speed or motion of certain device. A careful design of this controller is needed in order to avoid errors or unexpected responses that might be harmful. A well-known example is the mass-string-damper example. Using Matlab, show how can you design a control system that takes inputs, and design a control system that maintain the response of the system within the desired requirements of the user.

### Topic 3. Homework Solver

In this project, you can choose three of four problems from one of your textbooks in your discipline or major. A GUI solver is to be created that gives the user an option to select the type of the problems to be solved, and then ask the user to input few useful parameters, to get the output.

## 3. Civil Engineering

### Topic 1. Evapometer

Evaporation is a type of vaporization of a liquid that only occurs on the surface of a liquid. The other type of vaporization is boiling, which, instead, occurs within the entire mass of the liquid. Evaporation is an essential part of the water cycle. The sun (solar energy) drives evaporation of water from oceans, lakes, moisture in the soil, and other sources of water. Evaporation of water occurs when the surface of the liquid is exposed, allowing molecules to escape and form water vapor; this vapor can then rise up and form clouds[1]. Many mathematical models were developed to represent this process. These models vary in the complexity and the number of factors considered while estimating the mass of the evaporated water. Some of these models and equations are: Penman equation and Shuttleworth equations. Evapometer aims to calculate the mass of evaporated water given certain required parameters by the user. Also, the calculator can offer the user different methods of calculation, or to calculate the mass using different methods simultaneously, and present them to the user.

### Topic 2. Atmospheric Dispersion Calculator

Dispersion models are used to mathematically model the dispersed air pollutants from factories. One of the most well-known models is the Complete Equation for Gaussian Dispersion Modeling of Continuous, Buoyant Air Pollution Plumes. This model calculates the concentration of emissions. Since this pollutant concentration depends on the wind speed, and the height of the emission plume, a program that implements this model might be of great use by engineers designing the factories with toxic emissions.

### Topic 3. Homework Solver

In this project, you can choose three of four problems from one of your textbook. A GUI solver is to be created that gives the user an option to select the type of the problems to be solved, and then ask the user to input few useful parameters, to get the output.

[1] The definition and importance of Evaporation is copied from Wikipedia.

**Proposal Sample**

In the following a proposal sample of a project that will be done by a team of four members. You will need to follow the structure of this sample proposal while you are creating yours.

**Important Notes:** You are not allowed to use the same project in the project proposal sample.

CSCE 155N: Matlab

Final Project Proposal

Tuesday, April 1, 2013

"Gravitational Force Between Planets and Asteroids"

**Group Members:**

| Last Name | First Name | CSE Username | CSE Email |
|---|---|---|---|
| **Xyz** | Newton | nxyz | nxyz@cse.unl.edu |
| **Pqr** | Einstein | epqr | epqr@cse.unl.edu |
| **Abc** | Tesla | tabc | tabc@cse.unl.edu |
| **Def** | Galileo | gdef | gdef@cse.unl.edu |

**Problem Description**

In 1680s, Newton formulated what is so called *Newton's law of universal gravitation.* This law simply states that every pair of masses in the universe has an attraction force in between that is proportional to masses of the pair and inversely proportional to the square of the distance between them [see Figure 1].

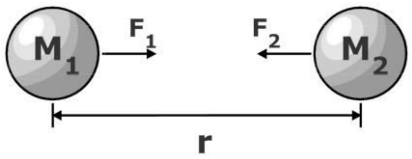


**Figure 1: Gravitational force between two masses $M_1$ and $M_2$**

In mathematical form, the Newton's law of universal gravitation expresses the force $(F)$ between two masses $M_1$ and $M_2$ by:

$$F = G\frac{M_1 * M_2}{r^2} \qquad (1)$$

where:

$F$: is the force between the two masses (note that $F_1 = F_2$) measured in newtons (N)

$G$: is the gravitational constant which is approximately equal to $6.674 * 10^{-11}\ Nm^2kg^{-2}$

$M_1 and\ M_2$: are the masses of the first and the second objects, respectively, in kilograms (kg)

$r$: is the distance between the **centers** of the masses in meters.

The following table shows some of the parameters of the Sun and each planet in our solar system.

| Planet | Mass ($10^{24}$ kg) | Diameter (km) | Distance from the Sun (AU) |
|---|---|---|---|
| **Sun** | $1.989*10^6$ | 1390000 | 0.000 |
| **Mercury** | 0.330 | 4879 | 0.390 |
| **Venus** | 4.87 | 12104 | 0.723 |
| **Earth** | 5.97 | 12756 | 1.000 |
| **Mars** | 0.642 | 6792 | 1.523 |
| **Jupiter** | 1899 | 142984 | 5.203 |
| **Saturn** | 568 | 120536 | 9.539 |
| **Uranus** | 86.8 | 51118 | 19.180 |
| **Neptune** | 102 | 49528 | 30.060 |
| **Pluto (a dwarf planet)** | 0.0125 | 2390 | 39.530 |

Distances in the solar system are very large. Therefore, using miles or kilometers to express the distances is not practical because it will need more analysis to interpret and compare planet's information. As a result, in astronomy, astronomical unit (AU) is usually used to represent the distances. AU represents the distances normalized with respect to the Earth's distance from the sun; hence, the distance of the earth from the sun is 1 AU. In other words, we know that the average distance from the earth to the sun is approximately 149,600,000 km, hence:

$$1\ AU = 149{,}600{,}000\ km \qquad (2)$$

If we know that the average distance from the sun to Mars is $227.9 * 10^6$ km, then the distance of Mars in AU units is:

$$Distance\ of\ Mars\ from\ the\ sun\ in\ AU\ units = \frac{227.9 * 10^6}{149.6 * 10^6} = 1.523$$

The physics and astronomy department can make use of this program. This program can help the students easily find the information, and gravitational forces between different planets and sun in our solar system.

**Solution Approach**

First, we will input the data of the solar system in a format of a data structure so that we can easily address different objects in the solar system. Our program will provide the user with many functionalities. The functions of our program are listed below:

1. The program will be able to take two objects as an input from the user and calculate the gravitational force between them, and then store the output data of these two planets into an output file that holds the name of these two objects.
2. In case of the wrong input by the user, the program will suggest a corrected name in the form of "Did you mean: " question. This will be done by comparing two aspects of the user's input: initial letter and length.

3. Given a certain planet, our program will calculate the gravitational force excreted on this planet from all other objects in the solar system. An efficient sorting algorithm will be used to sort these forces in a descending order. Also, a bar chart will be displayed that represents the forces in the descending order, normalized with respect to the smallest force.
4. In case of the existence of an asteroid in between two planets, the user will input the names of these two planets. Then, our program will provide the user with the safe distance at which the asteroid can safely move without the danger of being attracted to any of these two planets. This will be done by finding the point in between the two planets at which the force excreted on the asteroid is equivalent. This will include all planets to the left and to the right of this asteroid.
5. The last step in our project is to create a GUI that is user friendly in order to take inputs and provide the user with the proper output in a readable manner.

**Task and Responsibilities Allocation**

| Last Name | First Name | CSE Username | Task |
|---|---|---|---|
| **Xyz** | Newton | nxyz | Responsible of creating the gravitational force function, and the script that performs smart suggestions to the user to match his input in case of non-GUI form of the algorithm is used. |
| **Pqr** | Einstein | epqr | Responsible of designing the sorting algorithm, and implementing it. Also, will be responsible of creating the graph to be displayed, ensuring its ease of readability by the user. |
| **Abc** | Tesla | tabc | Responsible of creating the function that will detect the safe distance of the asteroid. |
| **Def** | Galileo | gdef | Responsible of creating the GUI of the program. |