ECE 280L Fall 2024

Laboratory 2:
Introduction to Simulink

Contents

1 Introduction

Many engineering problems can be easily modeled using block diagrams. The creators of MATLAB® understand that block diagrams are used to visually depict complicated systems and also that the mathematical characteristics of those systems, once defined, can be used to determine the values for the various signals throughout the system. For that reason, they developed the Simulink® package as an add-on to MATLAB. Simulink allows you to draw your system as if you were drawing a block diagram, and then to simulate the system using a variety of computational methods.

Throughout this course, we will be using block diagrams to represent systems, illustrating the flow of signals from input to output and the operations performed on those signals. In subsequent laboratory exercises, we will use Simulink to design systems (e.g. a touch-tone telephone system), simulate those systems to ensure proper functionality, and then compile them such that they can be implemented in real-time using a digital signal processing (DSP) toolkit in MATLAB.

2 Objectives

The objectives of this laboratory exercise are to:

- Learn how to launch and navigate through Simulink
- Understand the basic blocks and how to make connections between blocks
- Learn how to create arbitrary functions, transfer data between a Simulink model and the MATLAB workspace, and print models
- Become proficient at generating a variety of signals, implementing basic mathematical operations, combining such operations to create a system, visualizing and saving data, and publishing your model
- Build and simulate a simple system to analyze stock price data

3 Instructions

3.1 Exercise #1: Starting Simulink

1. Open MATLAB. Change to the folder in which you plan to save files for this lab. Start Simulink by typing `simulink` at the MATLAB prompt. You will see a new window, the **Simulink Start Page**, pop up. This window has tabs allowing you to create new items and explore examples. For now, create a new blank model. To do this, hover over the **Blank Model** image in the **Simulink** group in the **New** tab (the image has a blue triangle and a red square in it) and then click the **Create Model** box that appears.
2. Once the blank model opens, click the **Library Browser** button. This will bring up the **Simulink Library Browser** window. This window lists the predefined tools and functions, known as *blocks*, which can be used to create a system. Explore the variety

of blocks available by clicking on different titles in the left-hand navigation bar.

3. Now that you have a blank model to work with, the next steps are to learn about the basic blocks, how to insert a block into your model, and how to connect the blocks to each other to construct your system.

3.2 Exercise #2: Basic Blocks and Connections

In this exercise, you will build a simple system that generates a sinusoidal input signal, scales that signal, and both displays the resulting signal on a 'scope' and saves the signal to a variable in the MATLAB workspace. The Simulink library is divided into several categories, and each category contains several building blocks for modeling signals and systems. To look at the contents of a particular category, just select its name in the library window.

1. Expand the **Simulink** category in library window.
   (a) Select **Sources** to display blocks used to generate various types of signals.
   (b) Add **Signal Generator** block by right-clicking and choosing "add block" *or* dragging it into the canvas.
2. To examine and change the parameters of a block, simply double-click on it in your model window. The Signal Generator, for example, allows you to specify a *wave form* (sine, square, sawtooth, random), *time* (you will want to use Simulation Time), the *amplitude* of the signal, the *frequency*, and the *units* of frequency. Change the parameters so that the Signal Generator produces the signal: $x(t) = 3\ \sin(8t)$. When you have made the necessary changes, click the **OK** button to accept them.
3. At this point, you should save your model (if you have not done so already). Click the arrow to the right of Save in the model window then use "Save as" to save the file to your folder in a file called `lab2demo.slx`.
4. Processing Signals: Now that you have generated an input signal, you may want to process it. There are many different ways to process a signal in Simulink, ranging from basic mathematics to advanced image processing and frequency analysis. We will start with a simple operation: scaling.
   (a) Select the **Math Operations** category (a subset of the general **Simulink** category) in the original Library Browser window. Among the different operations you can perform is the **Gain** block, which acts like a multiplier. Drag this block into your existing model, so that you have both a **Signal Generator** and a **Gain** block.
   (b) Double-click the **Gain** block in your model to view its parameters. There are a surprising number of options for such a simple operation. Generally, however, you will only need to change the gain itself. Go ahead and set the gain to 2 and click the **Apply** button. Notice that the number inside the block itself (located in the Model window) changes as a result. The number in the **Gain** block will reflect the actual gain if there is room for it - otherwise the **Gain** block will have the letter K in it. For example, if you put in 0.0001 for the gain and hit **Apply**

again, the block will just have a K in it. Change the gain back to 2 and hit **Apply**, then **OK**.

5. Viewing and Saving Data: Now that you have a signal source and a simple operation, you will want to add the ability to view and/or save the signals/data in your system.
   (a) Select the **Sinks** category (subset of the general **Simulink** category) in the original library browser window. Each of the blocks listed in this category provide a way to display information on the screen, save it to the workspace, or send it to a file. More than one sink can be used in any system.
   (b) For your system, you are going to need to view and analyze the data, so drag both a **Scope** and a **To Workspace** block to your model and place them to the right of the **Gain** block.
   (c) If you double-click the **Scope** in your model window, an oscilloscope window will appear (rather than the parameter options you have seen for previous blocks). The **Scope** is useful for viewing the signals throughout a simulation in real time. Generally, you will add these during the construction of a model, but once you have completed the model, you will want a more permanent record of the data obtained. This is where the **To Workspace** block comes in.
   (d) The **To Workspace** block will record the signal values sent to it as a variable in the MATLAB workspace. If you double-click on the block, note that you can change the variable name - go ahead and make this `y`. The Save format that is most readily used is Array, so go ahead and make that change as well. Once done, click **Apply** and **OK** and notice that the block is now labeled with name including the variable (`out.y`).

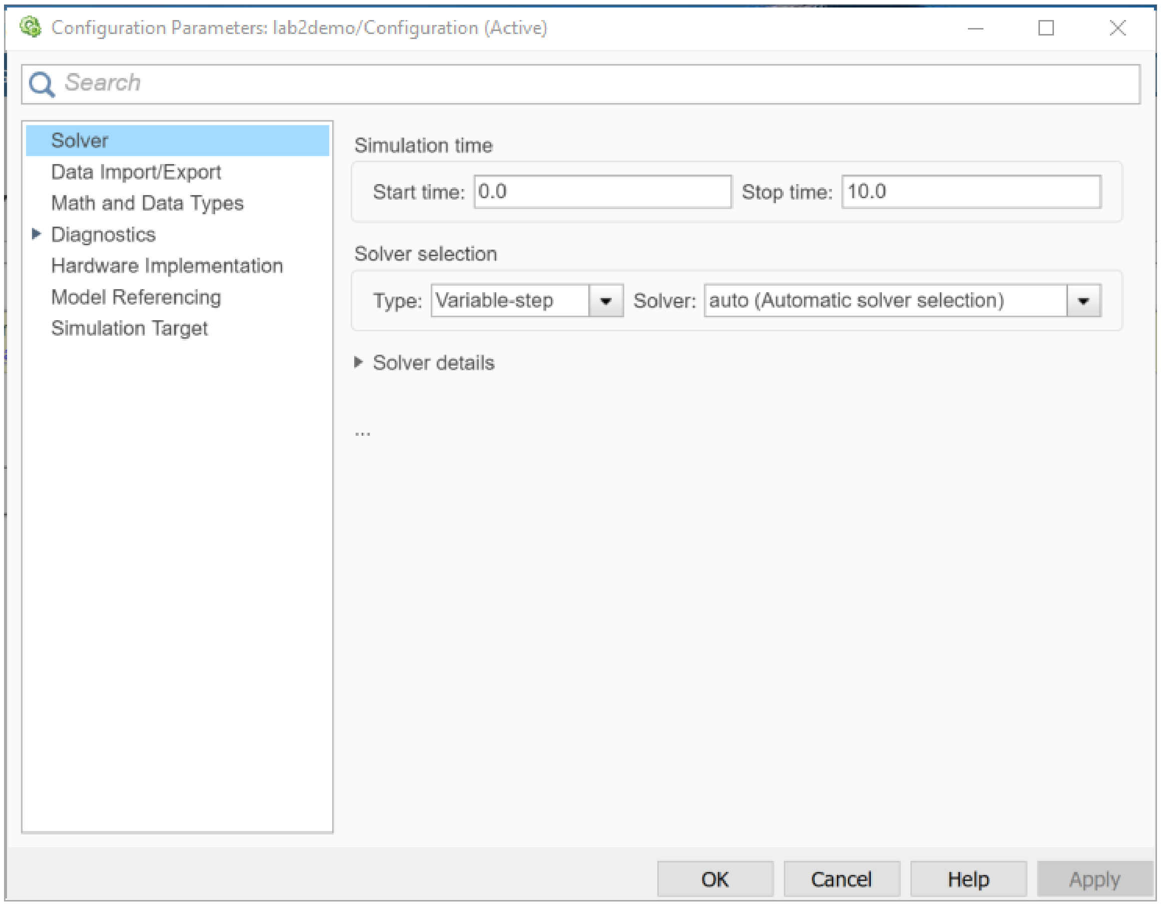
6. Making Connections: Now that the blocks are in place, you need to connect them to construct the system. If you run the cursor over the small port symbol (the > of a block), notice that the cursor changes from an arrow to crosshairs.
   (a) Connect the **Signal Generator** to the **Gain** block by putting the cursor on top of the output port of the **Signal Generator**, holding down the left mouse button, and dragging until the cursor is over the input port of the **Gain** block. You will notice that the cursor changes to double crosshairs when you have found a valid terminus for your wire.
   (b) Connect the output of the **Gain** block to the **Scope**.
   (c) If you also want to connect the **Gain** block to the **To Workspace** block, you will need to first click on the input of the **To Workspace** then drag until you hit the wire connecting the **Scope** to the **Gain** block. You cannot go back to the **Gain** block's output port, because every input or output port can only be used once. You will get the double crosshairs when you are over a wire such that a connection can be made and you will notice a node symbol (small black dot) at the connection after you release the mouse button. If you make a mistake and drop the wire before it is connected, you will get a dashed red wire with a port

symbol at the end of it. You can remove this wire by left clicking on it (thereby selecting it) and hitting Delete.

4 Exercise #3: Running Simulations

Now that you have a block diagram in which the input signal is multiplied by 2 and then displayed on a scope and saved to a variable in the MATLAB workspace, you are ready to simulate the system. In this exercise, you will learn how to run the simulation and access the outputs of the system.

1. Select the **MODELING** tab of your model window, click the arrow by the **Model Settings**, and then select **Model Settings**. You will discover that Simulink has an extensive array of parameters that can be set (see Figure 1).


Configuration Parameters: lab2demo/Configuration (Active)

Simulation time

Start time: 0.0 Stop time: 10.0

Solver selection

Type: Variable-step Solver: auto (Automatic solver selection)

Figure 1. Configuration Parameter window for Simulink Simulation.

2. For now, the critical parameters are in Simulation time. These parameters tell Simulink when to start and stop the simulation. The default setting starts at 0 seconds and stops the simulation after 10 seconds. Hit **OK** to accept these settings. The window will close.

3. To run the simulation, go to the **SIMULATION** tab and click the **Run** button.

4. If it is not already open, double-click on the **Scope** block to display the scope. The scope will show a plot of the values of the output, and when the simulation is finished, MATLAB will beep.

5. If you type `whos` in MATLAB, you will discover Simulink created a variable called `out`. If you type `out` in the command window, you will see that not only did the simulation store an array called `y` but also that it stored an array called `tout` for you. This latter array contains the times at which the output signal was calculated.

6. Now you can run the simulation and, when finished, issue the command `plot(out.tout, out.y)` in MATLAB's command window to plot the signal. When you do this, you will notice something very disturbing - the output signal does not look like a sine wave at 8 radians per second! This is because Simulink tries to simulate/solve your model as quickly as possible by taking as large a time step as it believes will maintain the accuracy of the system. For this particular model, allowing it to do that has produced a somewhat disastrous result - the locations at which data points were taken, if connected using lines, do not accurately represent the signal.

   To fix this, you can go back into the **MODELING** tab's **Model Settings** and make some changes. Specifically, open the **Solver** details section of the Solver and set the 'Max step size' to a value such that enough samples are taken in each period to give you an accurate picture of the signal – perhaps 0.01 seconds. (Later in the course, when we talk about sampling and reconstruction, we'll discuss how to calculate a reasonable sampling rate based on the frequency of the signal.) Apply the new settings, go back to the SIMULATION tab, and re-run the simulation. The scope should look much smoother. Then re-plot your results. The figure will look better, too.

4.1 Creating Arbitrary Inputs

There are several ways to create different inputs signals. The **Signal Generator** is the easiest way to create a basic sinusoid, square, or sawtooth wave. For other signals, you can use a combination of sources and basic math to construct the desired signal or the **Signal Editor** block. If your signal already exists in the MATLAB workspace, you can use the **From Workspace** block to import the signal. Likewise, you can import data from a .mat file using the **From File** block. During this assignment and the semester, you will become more familiar with the different ways that you can create arbitrary input functions using different blocks.

4.2 Transferring Data Between Simulink and MATLAB

As mentioned earlier, you will often want to transfer data from the Simulink simulation to the MATLAB workspace. The **To Workspace** block is the easiest way to get a signal's information to the workspace. Just make sure that the variable name is unique for each of the blocks and that you have used the **Array** method for storing the data. You can also save a signal to a .mat file using the **To File** block.

4.3 Saving and Printing Models

To save a Simulink model, just go to the **SIMULATION** tab's **Save** menu and select either **Save** or **Save As**. Simulink models should end with `.slx`. To print your model, you can use the **Print** function in the **Print** menu.

4.4 Other Commands: Using the Help files

You may find the Help files quite useful as you learn to use Simulink. Within the **Library Browser** window, go to the **Help** menu (question mark in a bubble) and select **Simulink** in the navigation panel at the left. The sections within Get Started with Simulink and Simulation will be particularly useful.

You can also get help on individual blocks using the Help files. Within the Library Browser window, go to the **Help** menu and select **Simulink**, then click the **Blocks** tab in the right panel. You will find the following blocks particularly useful:

1. Continuous: Derivative, Integrator, State-space, Transfer Fcn
2. Discontinuities: Saturation
3. Discrete: Difference, Discrete Derivative, Discrete State-Space, Discrete Transfer Fcn, Discrete-Time Integrator, Integer Delay, Unit Delay
4. Math Operations: All of them (OK, not all of them)
5. Sinks: All of them (IBID)
6. Sources: All of them (IBID)
7. User-Defined Functions: Fcn, MATLAB Fcn

4.5 Exercise #4: Simulate a Signal Mixer

1. Start a new blank model and then save it as `lab2ex4.slx`. Use Simulink to create a model for the input signals $x(t)$ and $y(t)$ as defined below:

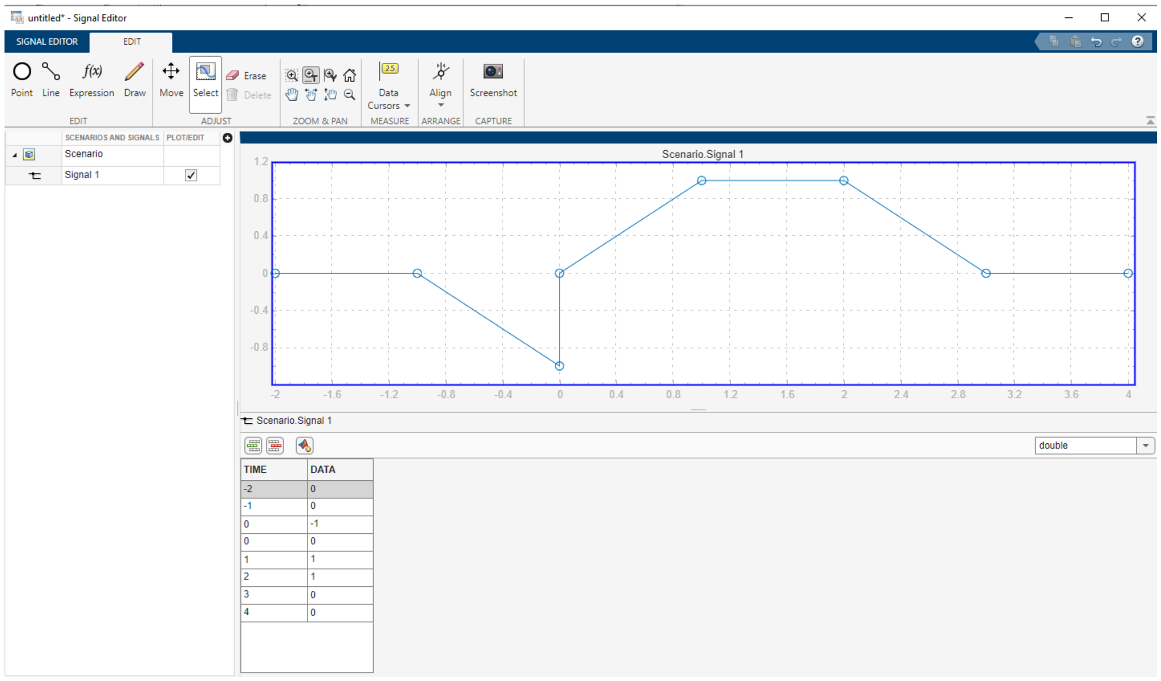
$$x(t) = \begin{cases} 0, & t < -1 \\ -t-1, & -1 \le t \le 0 \\ t, & 0 < t \le 1 \\ 1, & 1 < t \le 2 \\ -t+3, & 2 < t \le 3 \\ 0 & t > 3 \end{cases} \qquad y(t) = \begin{cases} 0, & t < -2 \\ 1, & -2 \le t \le -1 \\ -1, & -1 < t \le 0 \\ t-1, & 0 < t \le 1 \\ 1, & 1 < t \le 2 \\ 0, & t > 2 \end{cases}$$

   Use the **Signal Editor** as a source and use **To Workspace** blocks to obtain vectors for `x` and `y` in the MATLAB workspace.

   1. Move the Signal Editor block into the Simulink model.
   2. Open the Signal Editor block and press the icon next to ”To create and edit all scenarios, launch Signal Editor user interface” to open the signal editor user

interface.

3. Expand the Scenario 1 dropdown.
4. Check "Plot/Edit" next to Signal 1.
5. Add the points in for your piecewise signal at the bottom of the pane. Note that Simulink extrapolates from the first and last segments for times before and after the signal is defined, so you need to explicitly have a flat section at 0 at the beginning and end.
6. Save your changes and close the user interface when you're done.
7. In the Signal Editor block, check the "AApply signal properties to all scenarios" and "Apply signal properties to all signals" boxes.
8. Check the "Interpolate data" and the "Enable zero-crossing detection" boxes. Apply the changes. As an example, the **Signal Editor** for `x(t)` should be:



TIME	DATA
-2	0
-1	0
0	-1
0	0
1	1
2	1
3	0
4	0

Figure 2. Piecewise Signal for Exercise #4.

2. In your model, produce output signals for `x`, `y`, and a signal called `z` that is the product of `x` and `y`. Make sure the **To Workspace** blocks create arrays.
3. Change the Simulation time to run from -3 to 5 (using **MODELING** tab, **Model Setting**, **Solver**, **Simulation time**).
4. Use the following code to print the results:

```
plot(out.tout, out.x, 'k-', ...
     out.tout, out.y, 'k--', ...
```

```
    out.tout, out.z, 'k:')
xlabel('Time (s)');
ylabel('Amplitude');
legend('x', 'y', 'x*y')
title('Two Input Signals and Their Product (NetID)');
```

where `NetID` is your login ID (for identification purposes), and `out.tout` is the automatically generated time vector.

5. Save your Simulink model and the MATLAB figure. Print both your model and your plot. You will submit these with your final report.

4.6 Exercise #5: Analysis of Stock Price Data

In this exercise, you will implement an N-point *moving average filter*, which computes an output by averaging N input data points. This type of filter is commonly used to smooth out noisy signals. In this particular exercise, you will apply it to stock price data. In general, an N-point moving average filter is described by the following difference equation:

$$y[n] = \frac{1}{N}\left(x[n] + x[n-1] + x[n-2] + \cdots + x[n-N+1]\right)$$

where $x[n]$ is the input and $y[n]$ is the output of the filter.

1. You will be using actual stock price data for this exercise. To obtain this data, do the following:

   (a) Go to the website http://finance.yahoo.com

   (b) Enter the symbol for the stock you would like to analyze (or `IXIC`, for the NASDAQ composite).

   (c) When the quote and chart appear, select the **Historical Data** tab.

   (d) We will be using a 60-day set of data, but it will be easier to grab 3 months and then slice out the data we need. Choose a 3 month period by editing the **Time Period** and click **Apply**. (This will provide enough data for interesting results, but not too much data to plot.)

   (e) Underneath **Apply**, you will see an option to **Download**. Click on this. Your data will be saved to a Microsoft Excel `.csv` file (e.g., `IXIC.csv`). Find that file, copy it to your MATLAB folder, and call it `table.csv`.

   (f) Import your stock price data into MATLAB using the commands:

```
Days = 60
StockInfo = readtable('QQQ.csv')
StockPrices = StockInfo.AdjClose((end-Days+1):end)
```

   This will isolate the "Adjusted Closing" stock values for the data and then slice out the last 60 days worth of information. `StockPrices` will be a 60 row, 1 column array with the last 60 days' of adjusted close prices.

(g) Create an index variable from 0 to one less than the number of days – Simulink will need this when you import your data:

```
n = [1:Days]'-1;
```

(h) Finally, concatenate n and StockPrices to have a 60 by 2 array with n in the first column and StockPrices in the second column

2. Use Simulink to build a system that implements a 5-point moving average filter. The system should use your stock price data (in the MATLAB workspace) as the input signal and produce a single output vector, `out.ma`, which is saved to the MATLAB workspace. Note that you will need to ensure that the input array (used in the **From Workspace** block) contains both the time (`n`) and data (`StockPrices`) vectors. Use only Simulink blocks found in the Simulink library toolbox such as gains, delays, and summing point. This system can be built using 5 or fewer different types of Simulink blocks (including the input and output blocks). *Make sure to set the sample time of the **From Workspace** and other blocks in this model to 1.*

3. In the **MODELING** tab Mo**del Settings**, set the start time to 0, the end time to 65, the **Solver selection** type to Fixed-step, and the **Solver selection solver** to discrete. Run the simulation.

4. Create a plot showing both the input data (`StockPrices`) and the filtered output data (`out.ma`). It is useful to shorten both data sets to show only valid data points. You will need to think about why this is and comment on this in your lab report. For now, note that the following will display all the valid data:

```
figure(1); clf
plot(0:(Days-1), StockPrices(1:Days), 'k.-');
hold on
plot(4:(Days-1), out.ma(5:Days), 'r.--');
legend('Price[n]', 'ma[n]')
xlabel('Day (n)')
ylabel('Stock Price ($)');
title('Stock Price and Moving Average vs. Day (NetID)')
```

5. For your report, answer the following questions:

(a) Why did the plot command for `StockPrices` in Exercise #5, part 4. use `out.ma(5:Days)` and not `out.ma`?

(b) What is the length of the output (`out.ma`) vector? What do you observe about the length of the output (`out.ma`) relative to the length of the input (`StockPrices`)? Why is this the case?

(c) What do you observe about the values of the output (`out.ma`) relative to the input (`StockPrices`)? Comment both on the initial values of `out.ma` and on the overall pattern in the output relative to the input. (You will probably want to zoom in on the plot to see some detail.) Explain these results in terms of the system that you implemented.

(d) What properties does this system have (Memoryless, Causal, Stable, Time Invariant, and/or Linear)? Justify your answers.

6. Save your Simulink model and the MATLAB figure. Print both your model and your plots. Submit these with your final report.

5 Lab Report

5.1 Objectives (6)

Summarize the stated objectives in your own words.

5.2 Results and Discussion (82)

- Completion of Exercises #1-3 (10 points)
  - Simulink Model
  - Plot of Simulink output after setting max step size to 0.01.
- Submit your Simulink model and MATLAB plot from Exercise #4: Simulate a Signal Mixer (16 points)
- Submit your Simulink model and MATLAB plot(s) from Exercise #5: Analysis of Stock Price Data (22 points)
- Answer the questions from Exercise #5
  1. Why did the plot command for StockPrices in Exercise #5, part 4. use only part of ma and not all of ma? (3 points)
  2. What is the length of the output (ma) vector? (2 points) What do you observe about the length of the output (ma) relative to the length of the input (StockPrices)? (2 points) Why is this the case? (3 points)
  3. What do you observe about the values of the output (ma) relative to the input (StockPrices)? Comment both on the initial values of ma and on the overall pattern in the output relative to the input. (You will probably want to look at the first and last several data points in each array.) Explain these results in terms of the system that you implemented. (12 points)
  4. What properties does this system have (Memoryless, Causal, Stable, Time Invariant, and/or Linear)? Justify your answers. (12 points)

5.3 Conclusions (12)

Summarize what you have learned through this laboratory exercise.

Include the statement "I have adhered to the Duke Community Standard in completing this assignment", along with your signature, on the cover page and properly cite any sources you reference in your written report. Each person should turn in one report.

6 References

7 Updates

- June 2024: Turned into a standalone document
- August 2021: Converted to LaTeX.