For the first part of the project, a Student-T distribution with 10000 samples was analyzed and compared with a normal Gaussian distribution. The parameter for a Student-T distribution is the degrees of freedom, v. Theoretically, as the degrees of freedom approaches infinity, the Student-T distribution will become closer to the normal Gaussian distribution. By plotting a histogram of the 10000 sample Student-T distribution and placing the normal Gaussian distribution pdf on top of it, we can visualize the relationship between the two distributions. In Matlab, 10000 samples of the Student-T distribution were generated by using the trnd() function, in which one of the parameters of the function is the degrees of freedom. These samples were stored in a vector so that the histogram() function can plot the samples. In order to view the histogram and the normal Gaussian pdf on the same scale, the histogram() function must be specified to be normalized to pdf, through one of the parameters. In addition, the line “hold on” must be included to view both graphs on the same figure. 5 different Student-T distributions were created with increasing degrees of freedom. As can be seen, as the degrees of freedom increases, the Student-T distribution approaches the normal Gaussian distribution. For small values of degrees of freedom, the Student-T distribution can be characterized to be lower in peak value than the normal Gaussian distribution. Not only that, but the tails of the Student-T distributions seem to be bigger than those of the normal Gaussian distribution. Nonetheless, we can clearly see that at the degrees of freedom value of 100, the Student-T distribution looks more similar to the normal Gaussian distribution than the Student-T distributions of lower degrees of freedom do.

For the second part of the project, a data file was taken, and the data were fitted to try to find the approximate distribution that created that set of data. The three distributions that the data were fitted with are the Log Normal, Gamma, and Normal distributions. The data was loaded into Matlab using the load() function. In order to create those distributions, the parameters of those distributions were calculated using the loaded data. The Log Normal distribution needed the mu and sigma parameters, which were calculated from the loaded data using the fact that the median is equal to exp(mu), and mean = exp(u) \* exp(sigma^2 / 2). Combining these two equations, we see that sigma = √(2 \*ln(mean/median)). The mean and median can be found from using the functions mean() and median() on the loaded data. Therefore, the value of sigma can easily be found using these mean and median values. Likewise, mu can be found by taking the ln(median). The Gamma distribution needed the shape and scale parameters, which were calculated from the loaded data using the fact that scale = variance / mean, and that shape = mean / scale. The variance and mean can be found from using the functions mean() and var() on the loaded data. The Normal distribution needed the mu and sigma parameters, which were calculated from the loaded data using the fact that mu = mean, and that sigma is equal to the square root of the variance. The mean and variance of the loaded data can be found by using the functions mean() and var().Using these parameters, the respective pdf can be created using the lognpdf(), gampdf(), and the normpdf() functions. Each of these pdfs were plotted with the loaded data histogram so that their similarity can be seen. The loaded data was plotted using the histogram() function, and the pdfs were plotted using the plot() function.

For the third part of the project, a system known as a state-space model was simulated. The system is characterized by a combination of normal random variables. Specifically, two random variables are present, and both are normal random variables with a mean of 0. The two normal random variables were created in Matlab by using the functions normrnd(), in which the mean were set to 0, and the variance could be varied. Specifically, 501 normal random variable samples were created for each random variable, so each random variable is a vector of 501 values. The system was simulated with varying variance values. The x-interval of interest was from 1 to 500. Since we are given that x\_0 = 0, we can conclude that x\_t is the cumulative sum of the random variable u. In other words, x\_t = u\_0 + u\_1 + … + u\_t. Therefore, x\_t can be created in Matlab as a vector such that the value at each index t is equal to the cumulative sum of random variable u at index t. Note that the first value of x\_t at index 1 is actually the value for x\_0, since Matlab indexes start at a value of 1. This can be done by using the function cumsum(), in which the parameter would be the vector u. On the other hand, y\_t = 0.5(x\_t)^2 + v. Since this equation operates element wise, we need to specify the multiply operator in Matlab with a period symbol. This way, the vectors, x\_t and v will be multiplied element wise, and no error will occur. The resulting vector y\_t will be a vector of values characterizing the overall system. Plotting this vector with the plot() function will give a single realization. If we put this entire process into a for loop for 50 times, we will get a graph with 50 realizations. It is important to include “hold on” here so that the 50 realizations can be seen in one graph. This was repeated for the x process so that its realizations can be seen too. For the y process, the systems all seem similar in that the densest part of the graph is at the bottom, near the y value of 0. As a matter of fact, the system with variances(u,v) = (0.1,10), seems to be the most densest at the bottom of the graph. For the x process, the systems all look similar in that they all spread outwards in both negative and positive y directions. In particular, the system with variance\_u = 10 and variance\_v = 0.1, has output values that are higher than those of the other systems. For both x and y processes, as the x variance increases, so does the y axis scale.