College of Engineering

Whiting Data and BP Explosion

Engineering report prepared in accordance to the requirements

for EGR 102 Introduction to Engineering Modeling Section 10

T3amName

Jack Chauvin

Trevis Majtara

Sam DeLoy

Parth Patel

4/27/17

1 Executive Summary

The BP Whiting Refinery Energy Production Project analyzes production, demand, and emissions data in fifteen minute intervals from January 1, 2012 to September 30, 2016. The MATLAB functions and scripts created determined the unmet demand, lost revenue, average emissions, overage in Nitrous Oxide, overage of Carbon Monoxide, and optimum power per unit of fuel burned within the Whiting Plant. The data used for this code was fetched from an Excel spreadsheet and upload to a script in MATLAB, where 22 various variables were presented. Different variables within the data were plotted to create a better visual of trends within the data that can be used by BP. The first function found the Demand and production of the index by fetching data from the equivalent demand and total production values. The unmet demand is the demand minus the production. This value can be used to solve for the Whiting Plants lost revenue, showing that they will lose money if they produce more energy than what is in demand. BP's Whitings Plant power produced is a direct function of the fuel being burned in its surrounding temperature. Matrix multiplication was used to solve for the coefficients used to create the array of power values generated within certain temperature and fuel burn rate vectors. These vectors, when placed in a nested for loop, can determine the optimal power output and December fuel values that solve for the maximum fuel burn rate. The maximum fuel burn rate helps the company establish if they are remaining fuel efficient in accordance to their level of demand. Calculating the emissions produced by one of their turbines determines if specific regulations are being met. If the Whiting Plant surpasses the legal regulations of how much Nitrous Oxide and Carbon Monoxide are produced, the company becomes susceptible to numerous number of fines and sanctions. Functions were created to calculate the average and range values of Nitrous Oxide and and Carbon Monoxide. To find the average emissions, an if else statement with a nested for loop was used. The range of the emissions was found by nesting the average function within a for loop. The corresponding script of these two functions then calculates the overages, or where the emissions exceed governmental regulations. In this case, Nitrous Oxide cannot exceed 3 ppm during its three hour rolling average and Carbon Monoxide cannot exceed 9 ppm at any point during any point during its one hour rolling average.

The results from the BP Whiting Refinery Energy Production Analysis serve to demonstrate where the plant excels and fails. BP can lead as an energy company if they have data that leads to them finding high points and low points where they need to make changes for future success. The project calculated the lost revenue, optimum burn rate, unmet demand, the average and range Nitrous Oxide and Carbon Monoxide emissions, and overage rates. These

results can be used to show where BP Whiting can increase its demand, cut costs, and decrease emissions, allowing them to maximize profits.

2 Introduction

The Whiting Refinery is the 6th largest oil refinery in the US. It processes large amounts of crude oil and also produces steam for electricity production. The BP Whiting plant must keep up with demand and follow government regulations. The data collected by BP Whiting is used to analyze how efficiently demand is met and to see if and when the nitrous oxide and carbon monoxide emissions are greater than government regulations. This is done by using Matlab to give meaning to the large amount of data presented. The main medium of data representation is graphing, both in 2d via line graphs and in 3d via surface graphs. By plotting the data, it made it much easier to see trends in data and find meaning in the numbers given by BP Whiting.

2.1.1 Whiting History:

The BP whiting refinery is an oil refiner located on the Indiana Ship canal and the Southwest Shores of Lake Michigan. The large facility is over 1400 acres and is spread across Whiting Indiana, along with Hammond and East Chicago. This is the largest refinery in the Midwest along with BP's largest refinery. "The facility first opened in 1889, as part of John D. Rockefeller's Standard Oil Company, and for more than 125 years, it has been a key anchor of the northwest Indiana economy. Located about 17 miles southeast of downtown Chicago, Whiting is at the intersection of pipelines and railroads that carry its products to far-flung destinations."[4] Initially the Whiting refinery processed high-sulfur "Sour Crude" from the oilfields of Lima, Ohio. The ideal product of that time was Kerosene for lamp lighting, but by 1910, the automotive industry was booming, resulting in a demand for gasoline as a new product. As time progressed over the 1900's, the Standard of Indiana (Rockefeller's company) ranked as the second-largest American company with annual gross sales of $1.5 billion.

On August 27, 1955, a major catastrophic incident occurred. "Without warning, at about 6:15 a.m., several explosions occurred at the giant hydroformer, and shortly thereafter – as some who were at the refinery that day would later recount – "all hell broke loose." The initial blast tore apart the huge processing unit, hurling 30-foot long chunks of two-inch steel and countless smaller shards in all directions. "I thought the sun had exploded and that this was the end of the world," said one woman, quoted in *The Times* newspaper of northwest Indiana. "There was a terrible noise and a big red flash." The force of the initial blast broke almost every window in a three-mile radius. Smoke could be seen in Chicago and from 30 miles away. Fiery debris from the exploding cat cracker rained down on the refinery grounds and nearby residential areas.

Within the refinery, some of the flying metal and concrete landed on and punctured other oil storage tanks, igniting them in the process, touching off subsequent explosions."[7]

2.1.2 DeepWater Horizon

The DeepWater Horizon was an offshore exploratory drilling rig that operated in the Gulf of Mexico.[8] The purpose of the DeepWater Horizon was to perform exploratory drilling to find new deposits of oil and then cap and seal the well for a more permanent drilling rig to use and extract the oil from the deposit. On April 20, 2010, during the process of capping a well, a natural gas leak caused an explosion of the drilling rig that created a massive oil leak and killed 11 workers and injured 17.[1] The oil leak is the largest environmental disaster in US history.[8]

The causes of this explosion and oil leak boil down to a drive for profit that led to a disregard for safety and poor quality and cheap equipment. Normally when capping a well, an initial cap is placed and then heavy mud is placed to prevent gas leaks while the second cap is placed and then the mud is pumped out.[8] In the case of the DeepWater Horizon, to save time and money, the heavy mud was removed before the second cap was placed. They removed safety features despite a recent history of serious natural gas "kicks", which is when natural gas forces its way up the drilling pipes.[1] Also, they ignored a failed critical pressure test from earlier in the day.[1] Removing the heavy mud became a serious problem because, when the cement for the second cap was curing, it produced a large amount of heat due to the addition of nitrogen which thawed the frozen deposits of natural gas.[8] Nitrogen was added to the cement to reduce the curing time to save time and money which was the cause of the heat from the cement.[1] The newly liberated gas, no longer contained by the heavy mud, forced its way up the well and exploded upwards due to the pressure. This became such a large incident because there was no downwards pressure from the heavy mud and the blowout preventer, which is designed to cut off the pipes in case of an emergency, failed.

The failure of equipment and removal of safety equipment allowed for oil to leak out at an unprecedented rate. An estimated 4,900,000 barrels of oil leaked out of the well.[8] This caused a myriad of environmental problems including; contaminated beaches, marshes and estuaries and impacting thousands of birds, mammal and sea turtles.[3] The contaminated beaches and estuaries also affected tourism in coastal states and businesses that rely on tourism.[3] This disaster also put many fisherman out of business because of the contamination of water and destruction of the fish populations.[3] BP was also hurt by this disaster. They have spent million of dollars on cleanup and their reputation was decimated, their attempt at saving money

and maximizing profit led a serious expense and drop in business. After the spill, the economy in areas both directly and indirectly affected by the contamination was drastically impacted. Fishing companies saw a 20% decrease in production. Tourism came to a standstill in those areas because no one wanted to travel to the Gulf.

To help to fix this environmental catastrophe, DWH NRDA (Deepwater Horizon Natural Resource Damage Assessment) intends to restore the nation's natural resources caused by the spill after areas that were once full of plant life eroded away into open water. Also, on April 4, 2016 District Judge Carl Barbier approved a historic $20.8 billion global settlement with BP, the responsible party for the oil spill catastrophe. The settlement also includes a $5.5 billion in Clean Water Act penalties. More settlements throughout the years were placed.
"The U.S. Geological Survey (USGS) as part of the U.S. Department of the Interior (DOI) is responding to the Deepwater Horizon Oil Spill by establishing baseline conditions in water chemistry, bottom sediments, and aquatic invertebrates prior to landfall of the oil spill."[9]

2.2 Cogeneration

Cogeneration is the production of energy and useful heat simultaneously.[6] This is done by BP Whiting by using steam to power a turbine to create energy and by selling the steam as useful heat. This is shown in Figure 1 as the two gas turbines produce the steam for the steam turbine to turn into electricity. The steam used to create electricity is also sold for further use, creating cogeneration.[5] Even when the energy demand is met, the gas turbines still create steam to sell.[5] This allows for efficient and clean use of resources.

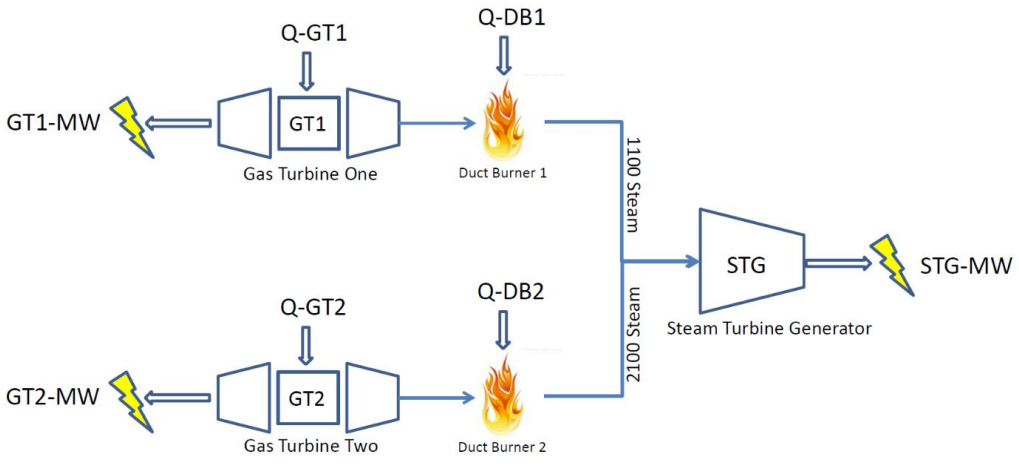

Figure 1: Diagram of whiting Clean Energy Plant's combined cycle gas cogeneration facility

3 Methods

Using Matlab, the data was imported, analyzed, and used to plot and find answers to questions about demand, productivity and emissions. The first thing done was to create a main script that was able to run all of the functions and scripts. This script was called t3amNameProject. The next step was to create the functions and scripts that will be run from the main script.

3.1 Importing Data-Module 3

The importing of data was done in the script t3amNameLoadData. The general point of this script was to load the data provided by BP which would then be analyzed later on in the project. The data from BP was provided in a "data.csv" file. The file was loaded using 'csvread()'. The file contained 161676 rows and 22 columns. Each of the 22 column was a variable and each row was data collected every 15 min over a span of 4 years from January 1st, 2012 to September 30th, 2016.

To make the data more manageable to navigate each column was made into its own vector and label according to the data that it represented. To further make the data more manageable, the datetime() function was used to make the date variable presentable in a Day-Month-Year format. This way, the data would be able to be referenced via specific dates instead of 15min step sizes. Once the t3amNameLoadData Script was completed it was implemented into the main script, t3amNameProject.

3.2 Plotting and Data Analysis-Module 4

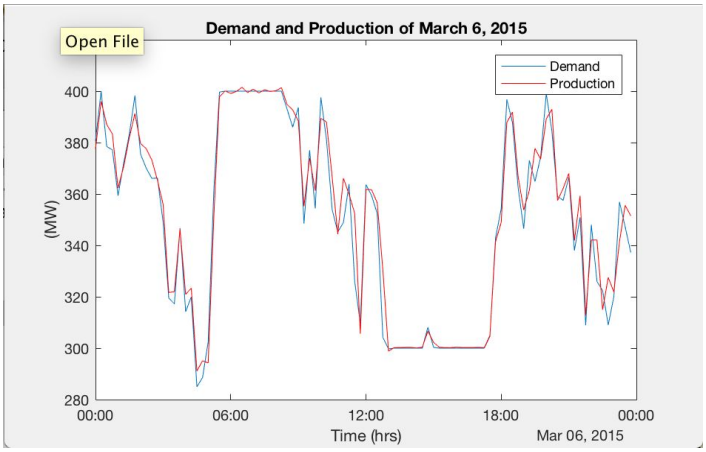

Figure 2: Plot of the Demand and Production of power in MegaWatts on March 6, 2015

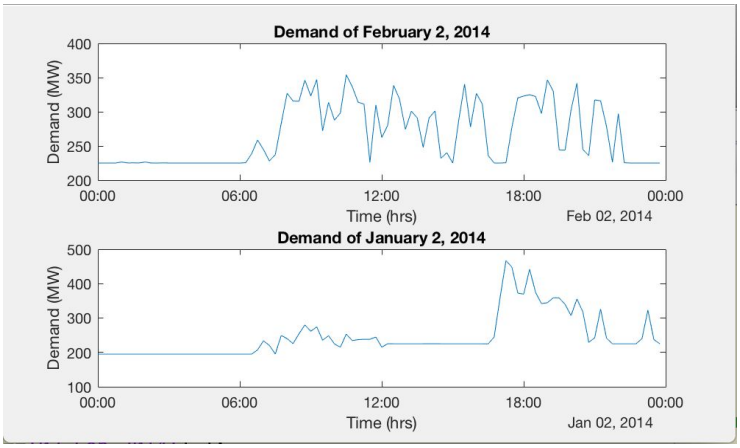

Figure 3: Plot of power demands in MegaWatts for January 2, 2014, and February 2, 2014

With all the data loaded and organized, the information could start to be understood. The production and demand from BP was the first thing on the agenda to be analyzed. Two separate figures were created in order to create a visual representation of the Production and Demand. One figure displays an overlay of the Production and the Demand for the entire 24 hours of March 6th, 2015. Production and Demand are both measured in units of MegaWatts on the y-axis and time on the x-axis. The second figure displays a 2-by-1 sub-plot. The top graph on the sub-plot displays the Demand (in MegaWatts) on February 2nd, 2014. The bottom graph on the sub-plot displays the Demand (in MegaWatts) on January 2nd, 2014.

In order to create the figures displayed the Script t3amNamePlot and the Function t3amNameUnmetDemand were created. The Function t3amNameUnmetDemand used a loop to calculate unmet demand over the specified period of time. The inputs to the function were a StartIndex, EndIndex, demand and production. The output of the function was a single vector that had an UnmetDemand for each element. The StartIndex and EndIndex were used to isolate the specific sections of data from the Demand vector and the Production vector. This way, the data on just date or time period needed could be selected. The next section of the code used the shortened vectors of Demand and Production to calculate unmet demand.

An 'if statement' embedded in a 'for loop' was used to continuously subtract the Production vector from the Demand vector. Thus creating a vector that had the unmet demand for each data point. If the Unmet Demand was negative for a certain Data point then a 0 would be put in its place in the UnmetDemand vector. The length of the 'for loop' was from i=1 to the length of the shortened Demand vector.

The next step was using t3amNamePlot Script to generate the Plots of Demand and Production on the days of February 2nd 2014, January 2nd 2014, and March 6th 2015. First the Figure that just held the sub-plots of February 2nd 2014 & January 2nd 2014 would be generated. A 'febdate vector' (All data points in February 2nd 2014) would be the x-axis and the y-axis would be a shortened 'newDemand' vector that would only contain the Feb-2-2014 Demand data points. The 'febdate' vector and 'newDemand' vector were then plotted on the sub-plot thus showing the demand every 15min on Feb-2-2014. The exact same process was repeated for plotting the January 2nd 2014 data, except the data points corresponding to Jan-2-1014 were implemented.

For creating Figure 2, which displayed an overlay of the Demand and Production on March 6th, 2015, the initial steps taken were identical to the steps taken for the previous figures. An initial startIndex was found (the first data point of Mar-6-2015), and a final endIndex was found (the last data point of Mar-6-2015). A MarchDemand (the March-6-2015 demand) vector that only included the data points that correlated with the data points in-between and including the startIndex and endIndex would was created. A MarchProd (vector holding the Production for March-6-2015) was created in the same method. The vectors were then plotted and thus showing a comparison between Production and Demand on March-6-2015.

In order to calculate the total lost revenue on March-6-2015 the function t3amUnmetDemand was called into the script. The inputs for the function had already been created. The output would be the vector for unmetDemand for Mar-6-2015 and it would be in MegaWatts. Next a March Sale Price vector was created (MarchSalePrice) that encompassed all the sale prices for Mar-6-2015, its units were in Dollars per (MegaWatts*Hour). The MarchSalePrice vector would also need to be multiplied by a scalar of 0.25 because the data point collected are in increments of 15 minutes while the units for SalePrice are per hour.

3.3 Emissions-Module 5

The purpose of this section was to check if Whiting follows the government regulations for emissions and see the levels of carbon monoxide and nitrous oxide on a rolling average and graph the data. The rolling average can be calculated at any given moment over the previous x-hours. The rolling Average for NOx in ppm will be done on a 3-hour basis while the rolling Average for CO in ppm will be done on a 1-hour basis.

The first step to graphing the emission data was to find the rolling average by creating the t3amNameEmissionsAvg function. The purpose of the function was to calculate the 3-hour rolling average for NOx and the 1-hour rolling average for CO. The inputs for

t3amNameEmissionsAvg were index (the time period over which the rolling averages were being calculated), dates (the data points), NOx (NOx emissions in parts per million), CO (CO emissions is parts per million). The outputs of the function were to be Average NOx emissions and Average CO emissions.

The first value that the function computed was the 1-hour rolling average for CO in ppm. This was done by creating a 'while loop' that checked to make sure that the current "i" value fell within the accepted duration of 59 minutes. The duration was set to 59 minutes because the data given to us by BP was not perfect because the data points were not collected at exactly 15 minute intervals but slightly less. In doing so, the information generated from the 'while loop' to create a vector of CO emissions in ppm (lengthCO vector) that fell within the calculated duration could be gathered. The average CO emissions was then collected by using the mean(lengthCO)=Average CO function.

The second value that the function computed was the 3-hour rolling average for NOx in ppm. This was done by creating a 'while loop' that checked to make sure that the current "j" value fell within the accepted duration of 2 hours and 59 minutes. In doing so, the information generated from the 'while loop' to create a vector of NOx emissions in ppm (lengthNOx vector) that fell within the calculated duration could be gathered. The average NOx emissions was then able to be collected by using the mean(lengthNOx)=Average NOx function. The Average NOx and Average CO were the outputs to be generated by the t3amNameEmissionsAvg function.

The next step in checking the regulations for the emissions of NOx and CO was to calculate the average emissions over a specific index (the index it the time period). This was done by creating the function t3amEmissionsRange. The inputs to the function were startIndex, endIndex, Dates, NOx and CO. The outputs of the function would be the range of NOx emissions in ppm and the range of CO emissions in ppm for the specified time interval. In order to complete this action, a 'for loop' was made. The 'for loop' was made to run through every point in the index (the index was the data points between and including the startIndex and the endIndex). The 'for loop' then had the function t3amNameEmissionsAvg embedded into it. By running every point in the index through the function t3amNameEmissionsAvg two vectors would be able to be outputted. The RangeNOx vector and the RangeCO vector. These vectors would be the average emissions over a specified time.

The next step in plotting the emissions for NOx and CO was to create a script called t3amNameEmissionsPlot. The purpose of the Script was to plot the average emissions of NOx and CO in parts per million throughout the entire month of July 2015. First the time index, July

2015 was created by using the find() function to find the first data point of July 2015 and then the last data point of July 2015. Next, two vectors were created, one for the total emissions of NOx from turbine 1 and the other for the total emissions of CO from turbine 1. These vectors were then run through the t3amNameEmissionsRange function to get the RangeNOx vector and the RangeCO vector for the month of July 2015.

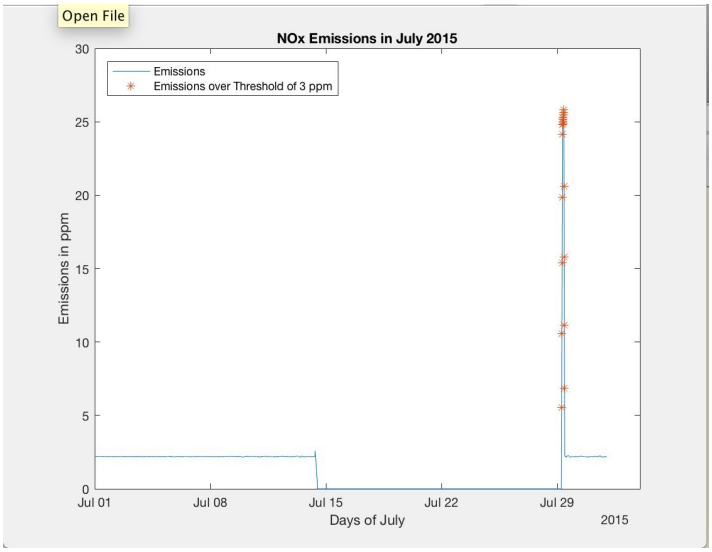

Figure 4: Plot of nitrous oxide emissions in ppm for July 2015

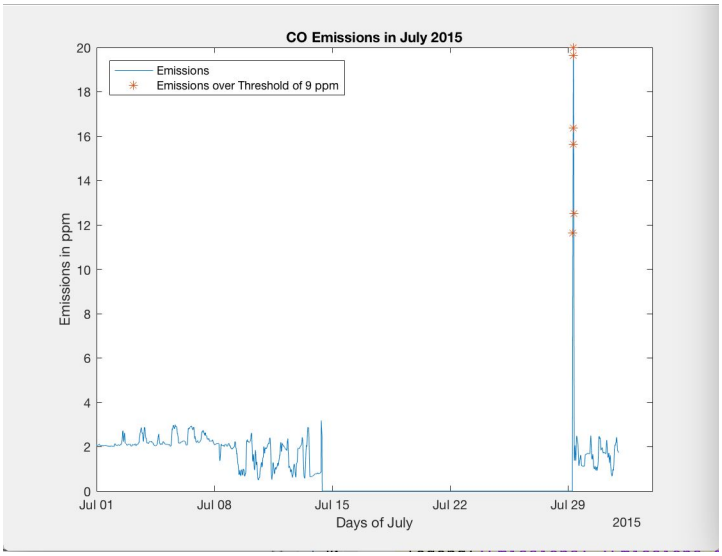

Figure 5: Plot of carbon monoxide in ppm for July 2015

In figure 4 the Range of NOx emissions are plotted for the entire month of July 2015 in ppm. This is seen with the blue line. Government regulations do not allow for NOx emissions to surpass the threshold of 3 ppm so it was necessary to find the data points that breached the threshold. This was done by using the find() function to find where the RangeNOx was greater than 3 ppm. Next the 'hold on' function was used to allow for a separate plot of NOx emissions greater than 3 ppm during the month of July 2015 to be plotted as an overlay using orange asterisk.

In figure 5 the Range of CO emissions are plotted for the entire month of July 2015 in ppm. This is seen with the blue line. Government regulations do not allow for CO emissions to surpass the threshold of 9 ppm so it was necessary to find the data points that breached the threshold. This was done by using the find() function to find where the RangeCO was greater than 9 ppm. Next the 'hold on' function was used to allow for a separate plot of CO emissions greater than 9 ppm during the month of July 2015 to be plotted as an overlay using orange asterisk.

3.4 Data Fitting-Module 6

The purpose of this last section in the code was to generate data for maximum allowable fuel burned at each ambient temperature. This way the relationship between fuel consumption, temperature, and the power produced by the turbine was able to be analyzed. The amount of power produced by a gas turbine is a function of the fuel that it's burning and the ambient temperature. In order to be able to make predictions and perform numerical methods on the data, the function was derived that relates the power produced by the turbines to the the resources it consumes and the ambient temperature. The relationship is shown by Appendix 4. P, (power in MegaWatts) is a function of the fuel burned, F (Million British Thermal units per hour) and the ambient temperature, T (Fahrenheit)

Using partial derivatives, the sum of square of residuals are minimized when the following normal equations are solved in order to solve for the coefficients in equation 1. The below array would need to be created.

$$\begin{bmatrix} n & \sum F & \sum T & \sum TF & \sum F^2 \\ \sum F & \sum F^2 & \sum TF & \sum TF^2 & \sum F^3 \\ \sum T & \sum TF & \sum T^2 & \sum T^2F & \sum TF^2 \\ \sum TF & \sum TF^2 & \sum T^2F & \sum T^2F^2 & \sum TF^3 \\ \sum F^2 & \sum F^3 & \sum TF^2 & \sum TF^3 & \sum F^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} \sum P \\ \sum FP \\ \sum TP \\ \sum TFP \\ \sum F^2P \end{bmatrix}.$$

Figure 6: Matrix used in performing Gaussian elimination to solve for the a coefficient values of each equation

In order to determine the max fuel burn rates, a function that would solve for the coefficients [a_0 a_1 a_2 a_3 a_4] had to be created. The function t3amNameRegression had the inputs startIndex, endIndex, zValues, fuel, temp, gasDensity. The outputs for this function were to be a 5 by 1 vector of the coefficients [a_0 a_1 a_2 a_3 a_4]. The very first step that the function performed was to convert the Fuel vector from units of hSCF/hr to MBTU/hr or else the final coefficients would not be correct. Next a Fuel vector, a Temperature vector, and a zValues vector were created. The length of all 3 of these vectors was based on the indices indicated by the startIndex and the endIndex.

Next, it was necessary to create the 2 known arrays that would be needed in solving for the unknown coefficients. The first matrix created was a 5 by 5 array. *n* was solved for by using the length() function to determine the length from the startIndex to the endIndex. Next the sum() function was used for remaining elements in the vector and the corresponding *fuel, temperature* or *zValues* in order to solve for the remaining elements. The second matrix created was a 5 by 1 array. In order to get the desired values for each element the sum() function was used and the corresponding *fuel, temperature* or *zValues*. Once both matrix's had been created, matrix division was used to in order to solve for the *'a'* vector which would hold the coefficients [a_0 a_1 a_2 a_3 a_4].

The next step in the process of generating the data for maximum allowable fuel burned at each ambient temperature was to create the Script t3amNameFit. The script solves for coefficients of production equations and plots the equations on graphs. The time interval domain which the script would work with was taken from the month of February 2013. The startIndex would be the first data points of Feb-2013 and the endIndex would be the last data point of Feb-2013. Next a 'for loop' was created with an embedded 'if statement'. The 'for loop' was run from i=1: length(index) which was the month of February 2013. The 'if statement'

embedded in the 'for loop' would check for when a certain value in the turbine2Fuel vector equaled 0. If one element in the turbine2Fuel vector equaled 0 then all the corresponding elements in other vectors would be set to NaN in order to increase the accuracy of the outputted coefficients. Below are the vectors that would be set to NaN if turbine2Fuel=0.

None of the elements in turbine2Fuel ever equaled 0 for the set index of February-2013. Therefore the 'for loop' could have also been completely omitted from the code and the coefficient values that were found would still be exactly correct.

The next step in the Script t3amNameFit was to solve for the regression coefficients over the data from the month of February 2013 of gas turbine 2 for Production, CO, and NOx. In order to do this, the function t3amNameRegression was called into the script. The function was run 3 times with 3 different vectors for the zValues. The function t3amNameRegression was run through with the turbine2Production vector in order to solve for the coefficients for production. Next, the function t3amNameRegression was run through with the turbine2CO (which had the emissions of CO from turbine 2) vector in order to solve for the coefficients for CO. Finally, the function t3amNameRegression was run through with the turbine2NOx (which had the emissions of NOx from turbine 2) vector in order to solve for the coefficients for NOx. Below are all the inputs for each time the function was ran.

The coefficients for Prod_a=[-148.22, 0.23, 0.28, 0.00, 0.00]. The coefficients for CO_a=[10.97, -0.01, -0.08, 0.00, 0.00]. The coefficients for NOx_a=[2.52, 0.00, 0.00, 0.00, 0.00].

Once the coefficients for Production had been solved for, the CO and NOx equations below were used to predict potential production outputs for Power (MegaWatts), CO (ppm), and NOx (ppm).

In order to predict potential production outputs a certain set of indices were created. A vector of Temperature was created in Fahrenheit from 0 degrees to 100 degrees with a step-size of 0.1 degrees. A second vector for the fuel burned was created in (MBTU/hr) from 0 to 2500 with a step-size of 10. Once this was completed, a 'for loop' embedded in another 'for loop' was created. The purpose of the embedded 'for loops' was so that all of the possible Temperature and Fuel Burn combinations were tested to solve for predicted production of Power (MW), CO (ppm), and NOx(ppm). Equations 4,5, and 6 from Appendix 7 were implemented into the 'for loop' because each of the corresponding coefficients were solved for and could test each equation at all plausible conditions. In the outer 'for loop', i=1 to the length() of the temperature vector. In the embedded 'for loop', j=1 to the length() of FuelBurn vector.

Once the ‘for loop’ runs all the possible combinations, the next step is to plot the 3-dimensional graphs in order to have a visual representation of each of the predicted production values for Power, CO and NOx at the different Temperatures and FuelBurn points. One figure with 3 sub-plots was used and in it the surf() function was used to graph each of the potential predicted productions. Below are the x,y and z axis for each of the graphs depicting all the possible combinations of FuelBurn and Temp to get different production outputs. Below the each graph is plotted on a sub-plot. The Predicted Power Production was in MegaWatts, Predicted CO production was in ppm, and Predicted NOx production was also in ppm.

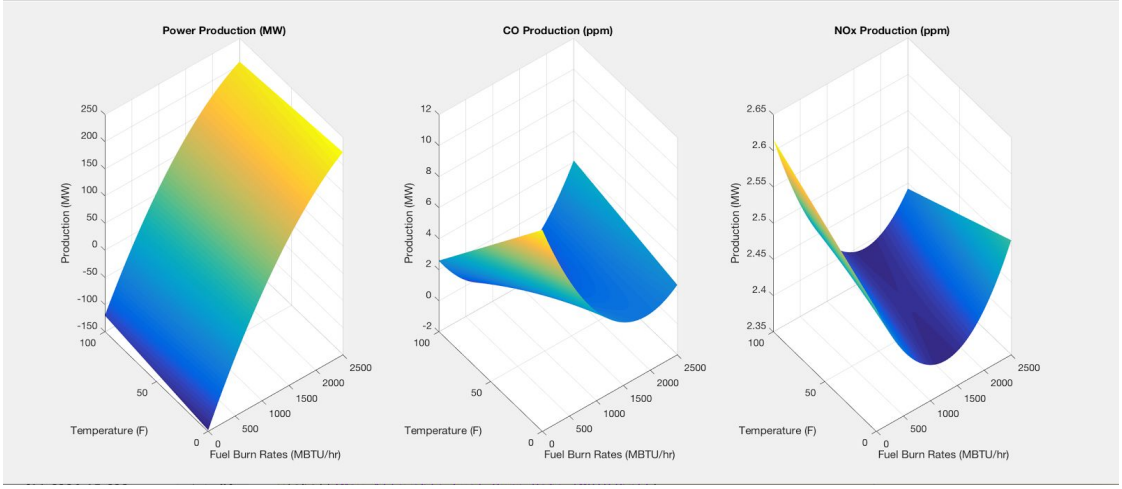

Figure 7: Plot of surfaces of power (MW), carbon monoxide (ppm), and nitrous oxide (ppm) production with FuelBurn and Temp on the x and y axes

The next step in this section was to create a 3D surface plot that showed all of the values of Power Production that fell within EPA government regulations. The regulations set were that CO emissions had to be less than or equal to 5 ppm and the emissions for NOx had to be less than or equal to 2.5 ppm. In order to find the Power Production within these specific parameters an ‘if statement’ was embedded within a ‘for loop’ which was then nested within another ‘for loop’.

The ‘for loop’ would run through Temperature and FuelBurn combinations one more time and check at what values the emissions of CO and NOx were. The ‘if statement’ would conclude that if at a certain Temperature and FuelBurn data point that either the CO or NOx

emissions breached government regulations then the Power Production for that data point would not be counted.

If the CO and NOx emissions both did not breach EPA regulations then the Power production at that certain Temperature/FuelBurn point would be recorded. All the recorded Power outputs that fell within EPA regulations were recorded in a vector named newPower. This newPower vector would be plotted against Temperature and FuelBurn using a surf() function. [*surf(FuelBurn,Temp,newPower)*]. FuelBurn was in units of (MBTU/hr), Temperature was in units of (Fahrenheit) and newPower was in units of (MegaWatts).

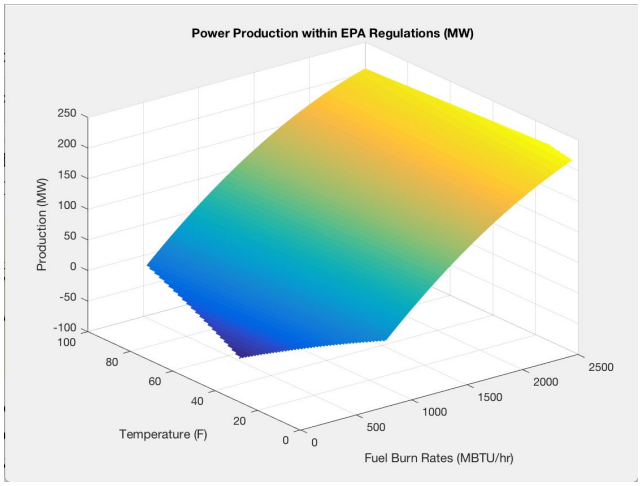

Figure 8: Plot of surface of Power production (MW) within EPA regulation

Figure 8 is the 3D plot of all the predicted Power generation that falls within EPA government regulations for the emissions of CO and NOx. Once the EPA allowed Power Production had been plotted, the next step in the project was to create a plot of the Maximum Allowed Fuel Burn Rates that fell within EPA regulations and to Plot them against Ambient Temperature. This was also to be done in the Script t3amNameFit. Maximum Allowed Fuel Burn Rates were in units of (MBTU/hr) and Ambient Temperature was in units of (Fahrenheit).

The 'for loop' (Appendix 11) used in finding the maxAllowableFuel Burn used an embedded find() function within a max() function that would find the corresponding elements in the FuelBurn vector. In the end, a vector called maxAllowableFR was created that held all the acceptable points found from the 'for loop'. From there, maxAllowableFR was plotted against Ambient Temperature. The graph of this plot is seen in Figure 9.

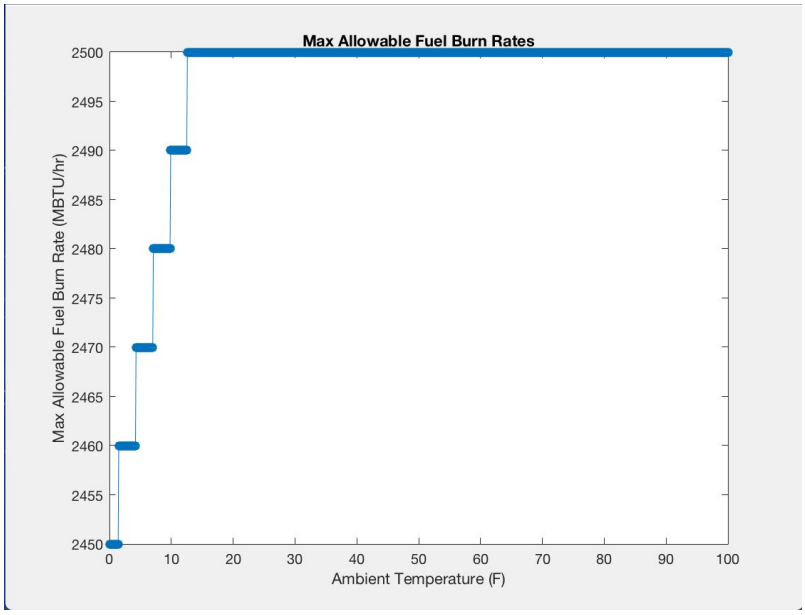

Figure 9: Plot of max allowable fuel burn rate (MTBU/hr) over a range of temperatures (°F)

4 Analysis

4.1 Importing Data

Downloading the data from the excel spreadsheet and into Matlab made it accessible to work on. Importing and separating the data into vectors made the data much more usable and accessible. By changing the date from numbers of days since 1900 to a day, month, year format made the data much easier to find and allowed for start and end indexes to function more easily. Overall, this section allows for the data to be efficiently accessed and worked with.

4.2 Plotting and Data Analysis

Figure 3 showing the Demand on January-2-2014 and on February-2-2014 can be used to interpret different energy usage trends on different days. For example on January 2nd most schools would be on winter vacation and many people may not be at work. The energy demand remains at about 200 (MW) all the way from midnight to about 5pm, with small influxes from 7am-10am (around the time when most people wake up). There is an uptick in energy demand from 5pm-9pm, this could be when most people have returned home. While on the contrary February 2nd pretty much all schools are back in session and adults are back to work. February 2nd represents a regular working day while January 2nd does not and this would be a reason why the energy demands are vastly different.

Figure 2 showed Production and Demand for March 6th 2015. It is important for a company such as BP to know how much energy they are producing and how much energy that they are selling, that way the company can make adjustments in order to not be wasteful. Even optimization of the smallest processes that may seem miniscule can have big effects on savings and profits for the company. Essentially the closer that the blue demand line and the red production line match up with one another then the better optimized the production at the BP plant was. To numerically measure how well BP optimized its production can be deduced simply by the magnitude of the Total Lost Revenue. The smaller the Total Lost Revenue would show better optimization of energy production. Less money lost means more money saved. Thus the closer that Production can match the actual demand without overshooting or undershooting then the greater that the overall profits will be.

4.3 Emissions

The EPA puts regulations on the emissions of certain gases that can be released into the air. This is to protect the environment and the general public from harmful consequences. The

government is then able to hold companies such as BP accountable for their actions in case EPA emissions regulations have been breached. By plotting the emissions for NOx and CO on graphs along with when they breach emission regulations then BP has a starting point on where to look for solving problems that caused the breaches. By taking these steps there is now a visual representation of the NOx & CO emissions in ppm for the entire month of July 2015. There is also a strong visual representation of when BP breaches the government regulations for NOx and CO.

For Turbine 1, from July 1st to July 15th both NOx emissions and CO emissions were steady and safely under government regulations. There was no concern. On July 15th-July 29th the were no NOx nor CO emissions from Turbine 1. The best assumption to be made is that Turbine 1 during this time was not being operated at all, hence the rolling average emissions were 0 ppm. On July 29th it seems that Turbine 1 was once again in operation and turned on. This is also exactly when NOx and CO emissions greatly breached EPA regulations. The best assumption for this over breach is that once Turbine 1 was returned to operation that it still was not correctly calibrated to keep the NOx and CO emissions within EPA regulations.

4.4 Data Fitting

The variation of Fuel Burn Rates (MBTU/hr) against variations of the temperature (Fahrenheit) that production was occurring at was an application of Chatelier's Principle. By changing the different combinations of Temperature and Fuel Burn Rates, BP can find the most cost efficient method to produce Power (MegaWatts) that falls within EPA government regulations. Figure 7 displays all the possible Power Production, CO emissions, and NOx emissions. So while one combination of Temperature and Fuel Burn Rate have optimized Power Production it may have caused CO emissions to breach EPA regulations and thus that combination of Temperature and Fuel Burn Rate cannot be used.

Figure 8 depicts Power Production at all combinations of Temperature and Fuel Burn with restrictions being set on NOx and CO to comply with EPA regulations. BP is then able to pick from any of these combinations of Temperature and Fuel Burn in order to find the optimal combination for Power Production. When the Ambient Temperature is paired correctly with the Fuel Burn then the overall cost of production will be lower and the company is able to save large amounts of money over a short period of time.

5 Conclusion

There are many things the BP can do to improve overall as a company. One very flagrant example is to not ignore negative pressure value tests when drilling as this type of reckless behavior is what caused the Deepwater Horizon Oil spill and essentially one of the worst man-made ecological disasters. Another improvement that BP should focus on is minimization and eradication of the release of pollutants into the atmosphere such as Nitrous Oxide, Nitric Oxide, and Carbon Monoxide gases. These gases are harmful to the planet and all organisms on the planet that inhale this air into their lungs.

BP has taken a step towards reducing the emissions of their plant by using cogeneration. The plant produces steam and uses that steam to drive their two turbines. These two turbines produce energy which is sold. The steam also gives of heat which is then captured as energy and thus that is sold also. By taking recorded data from the BP Whiting Refinery every 15-minutes covering a span of four years from January 2012 to September 2014 analysis of different aspects of the plants overall production became possible and recommendations could be made.

The aspects analyzed in this project included; The demand of energy in MegaWatts from the plant and the energy produced by the Plant on March 6th 2015. The purpose of this analysis being to determine how well the Plant was at Matching energy demand with energy production and seeing how well the Plant optimized the process.

Another aspect analyzed was the demand of energy on January 2nd 2014 compared to the demand of energy on February 2nd 2014. This allowed for the different patterns in energy consumption to be determined. The most obvious being that different days in the year where things may be out of the ordinary, such as January 2nd compared to February 2nd energy demands from customers will also be different. This knowledge allows BP to produce the right amount of energy per day depending on demand.

Another aspect analyzed was the NOx and CO emissions for the month of July 2015. The graph of this data showed when the BP plant managed to stay under government regulations and when they breached government regulations. This information will allow the government to punish BP for the infractions and to help BP to find and fix the problems with emissions.

The final aspect analyzed was determining the efficient fuel burn rates. This would display the rates that would be optimal for burning the fuel in order to generate the most power possible while at the same time remaining within government regulations. This way the company would be able to maximize its profits.

References

1. Eley, Tom. "What Caused the Explosion on the Deepwater Horizon?" WSWS, 14 May 2010. Web. 25 Apr. 2017.
2. "Deepwater Horizon Response & Restoration." *Response & Restoration*. U.S. Department of the Interior, 19 Oct. 2016. Web. 25 Apr. 2017.
3. Brennan, Katelyn. "A Stakeholder Analysis of the BP Oil Spill and the Compensation Mechanisms Used to Minimize Damage." (n.d.): n. pag. Web.
4. "EGR 102 Final Project." *D2L*. Michigan State University, n.d. Web. 25 Apr. 2017.
5. "Site Map." *What Is Cogeneration?* N.p., n.d. Web. 25 Apr. 2017.
6. J.D. "Inferno at Whiting." *The Pop History Dig*. N.p., n.d. Web. 25 Apr. 2017.
7. Pallardy, Richard. "Deepwater Horizon Oil Spill of 2010." *Encyclopædia Britannica*. Encyclopædia Britannica, Inc., 31 Mar. 2017. Web. 25 Apr. 2017.
8. Communications, Office Of. "USGS Oil Spill Response." N.p., n.d. Web. 25 Apr. 2017.

Appendix

```
for i=1:length(Temp)
    for j=1:length(FuelBurn)
        if CO(i,j)>5 || NOx(i,j)>2.5
            newPower(i,j)=NaN;
        else
            newPower(i,j)=Power(i,j);
        end
    end
end
```

Appendix 1: For loops used to solve for the power production that is within the EPA regulations calling for carbon monoxide and nitrous oxide emissions to be less than 5 ppm and 2.5 ppm

unmetDemand=Demand(i)-Production(i)

If Demand(i)-Production(i)<0 then unmetDemand=0

Appendix 2: Equation and restriction used for unmet demand

*Lost_Revenue Vector=MarchSalePrice*unmetDemand*0.25*

Total Lost Revenue on March 6th 2015=sum(Lost_Revenue Vector)

The Total Lost Revenue on March-6-2015 would total to be $1638.02

Appendix 3: Equation of lost revenue and total lost revenue of March 6, 2015

1) $P(F,T)=a0+a1F+a2T+a3FT+a4F2$

Appendix 4: Equation 1 used to solve for power production (MW). Fuel usage (MBTU/hr) and temperature (^{o}F) are inputs

f=fuel(startIndex:endIndex)

t=temp(startIndex:endIndex)

p=zValues(startIndex:endIndex)

Appendix 5: Code used for shortening the fuel, temp, and "zValues" vectors to the specified domains

```
for i=1:length(startIndex:endIndex)
If  turbine2Fuel(i)==0
turbine2Prod(i)=NaN;
turbine2Fuel(i)=NaN;
temp(i)=NaN;
natGasDensity(i)=NaN;
turbine2CO(i)=NaN;
turbine2NOx(i)=NaN;
end
```

Appendix 6: For loop: when turbine2Fuel=0, the values of turbine2Prod, turbine2CO, turbine2NOx, temp, and natGasDensity are changed to "NaN"

$$4)\ P(F,T)=a_0+a_1F+a_2T+a_3FT+a_4F^2$$

$$5)\ NOx(F,T)=a_0+a_1F+a_2T+a_3FT+a_4F^2$$

$$6)\ CO(F,T)=a_0+a_1F+a_2T+a_3FT+a_4F^2$$

Appendix 7: Equations 4,5, and 6. They solve for the power (MW), nitrous oxide (ppm), and carbon monoxide (ppm) production. The a coefficient values vary for each equation and are solved for using the functions displayed in Appendix 8

```
[Prod_a]=t3amNameRegression(startIndex,endIndex,turbine2Prod,turbine2Fuel,temp,natGasDensity)
[CO_a]=t3amNameRegression(startIndex,endIndex,turbine2CO,turbine2Fuel,temp,natGasDensity)
[NOx_a]=t3amNameRegression(startIndex,endIndex,turbine2NOx,turbine2Fuel,temp,natGasDensity)
```

Appendix 8: Functions used to solve for Prod_a, CO_a, and NOx_a: the a coefficients for each production equation

```
Power(i,j)=Prod_a(1)+Prod_a(2).*FuelBurn(j)+Prod_a(3).*Temp(i)+Prod_a(4).*Temp(i).*Fu
elBurn(j)+Prod_a(5).*(FuelBurn(j)).^2;
```

```
CO(i,j)=CO_a(1)+CO_a(2).*FuelBurn(j)+CO_a(3).*Temp(i)+CO_a(4).*Temp(i).*FuelBurn(j)+
CO_a(5).*(FuelBurn(j)).^2;
```

*NOx(i,j)=NOx_a(1)+NOx_a(2).*FuelBurn(j)+NOx_a(3).*Temp(i)+NOx_a(4).*Temp(i).*FuelBurn(j)+NOx_a(5).*(FuelBurn(j)).^2;*

Appendix 9: Code used to solve for various power (MW), carbon monoxide (ppm), and nitrous oxide (ppm) with different combinations of temperature (°F) and fuel usage (MBTU/hr) as inputs

surf(FuelBurn,Temp,Power)

surf(FuelBurn,Temp,CO)

surf(FuelBurn,Temp,NOx)

Appendix 10: Code used to plot surface of Power (MW), carbon monoxide (ppm), and nitrous oxide (ppm) production with FuelBurn and Temp as x and y inputs

```
for i=1:length(Temp)
maxAllowableFR(i)=FuelBurn(max(find(newPower(i,:)>0)));
end
```

Appendix 11: For loop that solves for the max allowable Fuel burn rate (MTBU/hr) for each temperature value (°F) tested