MICHIGAN STATE
UNIVERSITY

College of Engineering

Rockefeller

Engineering report prepared in accordance to the requirements
for EGR 102 Introduction to Engineering Modelling

Group: Rockefeller
Konrad Rauscher
Bradley Luzenski
Nick Wawee

4/27/2017

Executive Summary

The overall purpose of the project was to perform data analysis on plant data provided by the BP Whiting Refinery, located near downtown Chicago. Production was evaluated in order to evaluate whether the plant was meeting demand or not, a rolling average of emissions output for CO and NOx was calculated to compare emissions to legal caps, and power generation as well as CO emissions and NOx emissions were optimized to improve temperature and fuel burn rate.

Data was provided by BP Whiting that consisted of multiple measurable aspects of the plant. The data that was given contained production and consumption data of fuel, steam, CO emissions, NOx emissions, energy as well as the sale prices of the items sold by BP whiting. All the data was transferred from a .csv format to Matlab for calculation and analytical purposes.

The data was then employed to calculate the deficit between total production and total demand of the plant. Lost revenue was calculated by multiplying the sale price of the energy by how much energy was apart of the unmet demand. Plots were produced to aid the analysis of the demanded energy. The figures contained the total demanded energy over specific days as well as total production.

Emissions data of carbon monoxide (CO) and nitrous oxide derivatives (NOx) of one of the turbines was surveyed to compared to regulations. A set rolling average values for both species were produced for July 2015. The rolling averages were then plotted as well as any overages in emissions as a visual aid. Overages occurred for both gasses
(Sec 6)

Introduction

Background on BP Whiting Refinery/Cogeneration

BP's Whiting Refinery has been a fixture of the northwestern Indiana area since 1889. Founded by entrepreneur John D. Rockefeller, the refinery has been a major provider of oil products to the Midwestern United States for years. The refinery is currently operated by the company British Petroleum, otherwise known as BP. Whiting is only a small fraction of BP's total industrial holdings: BP operates in 72 countries and has 74,500 employees across the world [3]. Although its name (British Petroleum) suggests that BP may be focused only on oil products, BP owns numerous wind and biofuel plants, showing that it is exploring alternative energy as a component of its business model.

Whiting is situated in the middle of the large industrial area of metropolitan Chicago, so its products can be used at nearby locations or shipped across America in pipelines or on trains. Much of the plant's output is products of crude oil such as gasoline, diesel fuel, and asphalt. However, it also produces a large amount of electricity using steam turbines.

The Whiting Clean Energy Facility, located on the grounds of the refinery, is a natural gas plant which utilizes the principle of cogeneration in order to best utilize the energy it produces. Cogeneration is a method of electricity generation in which leftover heat from electricity generation is used for other functions, such as generation of steam [5]. The Whiting Clean Energy Facility provides its excess steam to the oil refinery, and the steam is used as a consistent, reliable source for the usages of steam in refinery processes [7].

Description/Motivation of Project

The purpose of this project is to analyze the data given by the BP Whiting refinery by focusing on select subsets of data and evaluating various values in those subsets. Given the large amount of data present, it is more convenient for analysis to take place at a more specified level, and the analysis is intended to use the results of focusing on a small set of data to draw conclusions regarding that data. Specifically, the energy was analyzed with respect to time on various days, the emissions of carbon monoxide and nitrogen oxides in various months, and the relationship between power generation and various other factors of plant production. The usage of real-world data from BP Whiting, data which may be analyzed by real engineers, shows that the project is motivated to be a project showing students a real-world engineering experience, allowing them to make better future decisions regarding engineering majors and careers. The project also allows students to further develop coding skills: the quantity of MATLAB code required means that every student is likely to work on some code, meaning that members of the EGR 102 class will have equal opportunities to test their coding ability by completing one of the modules. In addition to writing code, the assignment promotes practice in writing clear, technical prose, allowing students the opportunity to further develop technical writing skills learned in EGR 100 and move towards applying those skills to a future class or future career. Thus, the project is primarily motivated as a learning experience, and working on the project appears to be a significant step in learning how to better apply computer code to engineering situations.

Deepwater Horizon Event

On April 20th, 2010, a massive explosion occurred on the Deepwater Horizon oil well in the Gulf of Mexico. Eleven workers stationed on the rig were killed in the explosion and resulting fire, and 17 were injured. Though the immediate human casualties were tragic, the damage of the explosion had just begun. After the well exploded and sank, it began leaking oil into the Gulf of Mexico. The leakage continued for 87 days, depositing approximately 3.19 million barrels of oil into the ocean [8]. This was the second largest oil spill in history, and the largest in US history [10].

A. Key Stakeholders

The most significant stakeholder in this event was the BP company, British Petroleum. BP operated the Deepwater Horizon oil well, and it thus bore the primary responsibility for the oil spill. Not only did BP lose equipment and oil sales revenue due to the spill, but it also suffered a loss in credibility from which it is still struggling to recover today. Since it was the primary cause of the accident, BP created a $20 billion compensation fund to distribute money to other stakeholders negatively affected by the spill [16].

An additional significant stakeholder in the Deepwater Horizon oil spill was the tourism industry in the Gulf Coast. As the oil from the spill spread across the Gulf, hundreds of kilometers of beaches were contaminated with oil, destroying the natural beauty of the waterfronts and ensuring that swimming in the Gulf was both undesirable and dangerous. This resulted in a general decrease in tourists visiting the area, hurting local businesses and decreasing tourism activity in the Gulf area as a whole.

In addition to the tourism industry, the fishing industry was harmed. The oil spill brought fishing in the Gulf to a virtual standstill: most fish regularly consumed by humans were contaminated with oil, so economic activity due to selling fish came to a grinding halt. Though fishing in the Gulf has since essentially returned to normal, the fishing industry was greatly affected by the Deepwater Horizon spill and its cleanup methods, so it was a major stakeholder.

The fourth large stakeholder to be discussed in this report is the government. Since the spill occurred in US waters, the main responsibility of organizing and facilitating the cleanup of the spill rested on the United States Government. Several government agencies, such as the Coast Guard, EPA, and NOAA, participated in cleanup efforts [11]. They had to devote considerable resources to mapping and cleaning up the spill, in addition to providing information to the public about the spill, so they are a significant stakeholder.

B. Causes

BP officially took full responsibility for the Deepwater Horizon event in a statement from Chief Executive Tony Hayward in 2010 [11], and later investigation concluded that BP was the primary cause of the spill [12]. However, some responsibility was put upon Halliburton, the company in charge of the well's cement, as well as Transocean, the owner of the rig.

Mechanically, the explosion was directly caused by pressurized natural gas from the bottom of the well making its way to the top of the rig and beginning to combust. The rapid gas movement was caused by defects in the cement at the bottom of the well, which usually keeps oil and natural gas contained near the ocean floor. Investigations said that the issues with the concrete were exacerbated by poor decisions made by teams working on the rig: "poor risk management, last-minute changes to plans, failure to observe and respond to critical indicators, inadequate well control response and insufficient emergency bridge response training by companies and individuals responsible for drilling at the Macondo well and for the operation of the Deepwater Horizon" were some causes named in a September 2011 report from the Bureau of Ocean Energy Management, Regulation, and Enforcement and the Coast Guard [12].

C. Effects

Over 130 million gallons of oil were leaked into the Gulf of Mexico by the Deepwater Horizon spill [11]. The oil went several places throughout the ocean: some rose to the ocean surface to create oil slicks, some sunk to the ocean floor, and some floated in the mid-ocean in giant oil plumes [8].

The ecosystem of the Gulf of Mexico was greatly harmed by the oil spill. Salt marshes near the coast suffered extensively from erosion after the spill, leaving "irreversible" ecological damage [15]. Seabirds lost the ability to fly when their feathers became coated with oil. Sightings of deformed fish became commonplace after the spill, and dolphin deaths drastically increased. As previously discussed, the fishing industry was greatly harmed by the spill, and the tourism industry in the Gulf Coast area also suffered.

The BP company was also negatively affected in several ways: it lost millions of dollars directly due to fines and compensation, and its image was permanently damaged as a result of the spill. The total direct cost of the Deepwater Horizon spill was calculated by BP as $61.6 billion [16]. This cost includes civil penalties, money for ecological rehabilitation, and settlements towards those negatively affected by the oil spill. BP suffered from more than this direct cost: their stock value plummeted from $57.07 to $28.88 immediately after the spill, and it still has not recovered seven years later [13]. BP's public image is now permanently scarred; many members of the American Public now see BP is a villainous corporation dedicated to exploiting the environment.

D. Responses

The massive oil spill that was the Deepwater Horizon event required similarly massive cleanup efforts. One effort used was floating booms, which create a physical obstacle on the ocean surface, stopping the oil spill from spreading further [14]. After isolating the oil, workers needed to actually clean it up. Two main types of methods are used to clean up oil: physical methods and chemical methods. Physical methods remove the oil from the water physically. They include the usage of boats called skimmers, which skim oil off the surface of the water, and sorbents, which absorb oil from the surface and are then collected and disposed of. Chemical methods break down oil in chemical reactions into less harmful products. The BP spill clean-up process consisted of using approximately 1.8 million gallons of chemical dispersants, which react with oil and allow the products to dissolve in water, reducing harm caused to organisms [14]. The products of the dispersion reactions are usually broken down by deep-sea microbes, although they can enter the oceanic food chain and cause harm to some organisms.

Though oil on the ocean surface has been cleaned up to a large extent, the products of dispersion reactions and the sub-surface oil remaining are still present in the Gulf. Only time and investigation will tell what the ecological impact of these will be in the future.

Methods

Module 3 - Importing Data

Module 3 loads the provided data.csv file and creates separate vectors for each set of data. The data was provided in a .csv format file. In order to load it, the csvread() command was used, starting on row 2 in order to avoid the text in row 1. From there, each column of data was separated and named according to the text header in row 1 of data.csv, a list of which is provided in appendix 3.1. An example of this column separation can be seen in appendix 3.2. In order to reformat the date vector into a Matlab compatible format, the datetime() function was used to convert the date from Excel format, shown in appendix 3.3.

Module 4 - Plotting and Demand

Upon completion of the importation of the data, it was then utilized in plotting and then the creation of a function. The function that was created was called RockefellerUnmetDemand and its purpose was to calculate unmet demand of energy (MW) given a specific time period, the total demand (MW), and total production (MW). This was done by creating a range vector by calling upon the starting and ending time index. A for-loop was then ran starting from 1 to the length of the range vector. For each iteration of the loop, the unmet demand at the time interval was calculated by taking the deficit between the total demand and total production at the time interval. An if statement was incorporated to make the unmet demand equal zero if the unmet demand for the time interval was less than zero. In other words, if the unmet demand had a negative value, then the function would output a zero instead of the negative value. The function ultimately produces a vector of unmet demand in quarter hour intervals.

A script was then created to generate multiple figures and to calculate lost revenue for a specific time interval. The first figure that was generated had two separate plots that plot total demand (MW) for February 2nd, 2014 and January 2nd, 2014 respectively. Both plots were generated on the same figure by using the subplot command and the data called upon in each plot was done in the same way. The find function was then used to define the starting index at the beginning of each day and the ending index at the end of each day. The demand of the specific index was then called upon by using the definitions and was plotted as the dependant variable. The independent variable was plotted by calling upon the dates in the dates vector by using the starting and ending index. The second figure that was generated was an overlay plot that displays the total production and total demand for March 6th, 2015. It was generated by the use of the same method stated for the other two plots. Conclusion of the script includes a calculation of the total lost revenue for March 6th, 2015. Unmet demand, the sale price vector, and a conversion factor were multiplied to create the lost revenue for each time period. The aggregate was taken of the lost revenue vector was taken using the sum command and calculates the total lost revenue ($) .

Module 5 - Emissions

The purpose of this portion was to calculate rolling averages of emissions of NOx and CO from Turbine One. Coding for this portion consisted of two functions and a script named RockefellerEmissionsAvg, RockefellerEmissionsRange, and RockefellerEmissionsPlot. The three components were used in conjunction to generate data for July 2015 which was then analyzed.

RockefellerEmissionsAvg calculated the average emissions (ppm) of NOx and CO over three hours and one hour respectively. Inputs of the function consisted of a starting index, the date vector, the CO emissions vector, and the NOx emissions vector. Outputs of the function were vectors of rolling averages of CO and NOx that were calculated per indice. A while loop was incorporated to obtain a measurement for each indice by using a tolerance of 02:59:00 and 00:59:00 for NOx and CO respectively. The while loop created a vector of values within the tolerance and an average was taken from that vector. The average that was taken is outputted as a scalar as avgCO and avg NOx. Specifics of this process are shown in Appendix 5.1.

RockefellerEmissionsRange calculated a range of averages with the use of RockefellerEmissionsAvg. The inputs were a starting index of dates, an ending index, the date vector, and the vector for CO and NOx emissions (ppm). RockefellerEmissionsRange preallocated vectors for the avgCO and avgNOx outputs from RockefellerEmissionsAvg. The function calculated the ranges of values by evaluating the RockefellerEmissionsAvg function in a for loop for the interval between the starting index and the ending index. Each evaluation was then transferred into rangeNOx and rangeCO vectors which represented the range of emission values (ppm). The values that were calculated by both functions were the desired one hour rolling average and three hour rolling average with the utilization between both functions. The excerpts of the code for this function is can be viewed in Appendix 5.1.2.

RockefellerEmissionsPlot, utilized the preceding dependent functions to calculate and graph the rolling average emissions for NOx and CO (ppm). The purpose of the calculations was to determine whether the emissions surpassed EPA calculations. The begining of the script ran the RockefellerLoadData portion of the code. It then set up the startIndex and endIndex variables, finding the start and end indices of data analysis by searching for the dates in the date array using the find function. It ran the RockefellerEmissionsRange function, assigning its results to the rangeNOx and rangeCO arrays. It then iterated over each of those arrays using a for loop. If the value at a specific index of the array exceeded the EPA emissions limit, (3 ppm for NOx, 9 ppm for CO) the value was placed in the overNOx or overCO array and the corresponding date was added to the dateNOx or dateCO array. The overNOx array and the overCO arrays were the arrays of emissions exceeding EPA limits for NOx and CO. That process is shown in Appendix 5.1.3. Next, the data was plotted using the plot command. Using if statements as shown in Appendix 5.1.4, the script plotted only rangeNOx/rangeCO if there were no overages and plotted rangeNOx/rangeCO and overNOx/overCO if there were overages. The rangeNOx/overNOx and rangeCO/overCO were on separate figures, one figure for NOx and a different figure for CO. The x-labels of the figures were the date, and the y-axis showed the concentration, in ppm. The rangeNOx/rangeCO vectors were plotted in blue, and the overNOx/overCO vectors were plotted as red asterisks.

Module 6 - Regression and Data Fitting

Module 6 consists of 2 files, a function, RockefellerRegression.m, and a script, RockefellerFit.m. these two files found regression constants for the turbine data, and then used these constants in order to analyze NO_x and CO_2 emmissions, along with power generation rates at varied temperatures and fuel consumption rates.

The goal of RockefellerRegression was to take starting dates, startIndex, and ending dates, endIndex, for data, and use zValues, fuel, temp, and gasDensity to calculate regression constants, a, for the given data. To start, the function uses startIndex and endIndex to find the relevant data in gasDensity, fuel, temp, and zValues. The fuel values were given in units of hSCF/hr, so they needed to be converted to MBTU in order to agree with the regression formula. This process can be seen in appendix 6.1. From there, the provided regression system, found in appendix 6.2, was used to create the A and b matrices in order to solve for the regression constants of the zValues. This system of equations, using matrix A and b, was solved using the backslash command, outputting the fitted regression constants, a. The code for this can be found in appendix 6.3.

The script RockefellerFit used the function ReckefellerRegression in order to find the regression coefficients for turbine2Prod, turbine2Nox, and turbine2CO over the month of February 2013. To start, the script used a for loop to find each index where turbine2Fuel equaled 0, and set turbine2CO, turbine2Nox, turbine2Prod, and turbine2Fuel for these indices to the value NaN, which stands for Not a Number. This is done in order to obtain a more accurate regression, as the zero values will not contribute to the regression, and can be seen in appendix 6.4. From there, the RockefellerRegression.m function was used to find regression coefficients aProdFeb13, aNoxFeb13, and aCOFeb13, for turbine2Prod, turbine2Nox, and turbine2CO respectively.

After finding the regression constants, they were used to find data on power, NOx, and CO_2 production over a range of temp and fuel values. First, the vector tempRange was created, a vector from 0 to 100 ^oF, with a step size of 0.1 ^oF. The vector fuelRange was also created, a vector ranging from 0 to 2500 MBTU, with a step size of 10 MBTU. The code creating these vectors can be seen in appendix 6.5. From there, nested for loops, used to create an array of all possible tempRange and fuelRange combinations, was implemented. Inside the nested for loops, the arrays powerRange, the power generated (MW), NoxRange, the NOx emissions (ppm), and and CORange, the CO emissions (ppm), were generated, solving for each value at each combination of temperature and fuel consumption. The formulas for finding powerRange, NoxRange, and CORange can be found in appendix 6.6. After finding each array, powerRange, CORange, and NOxRange were graphed.

After finding powerRange, CORange, and NoxRange, the next step was to find all values of CORange and NoxRange that were over the emissions cap. First, new arrays were created, CORangeCap and NoxRangeCap. This was to preserve the data in CORange and NoxRange. To find values over the cap, the length of each array was found by using the size()

command, and then multiplying the resulting dimensions to get the total length of each array. From there, a for loop was used to each each index of CORange and NoxRange to see if it was within the range of 5ppm and 2.5ppm, respectively. If the value was less than or equal to the cap, the value was set to 0. This process can be seen in appendix 6.7. From there, the values for CORangeCap and NoxRangeCap were summed into the vector NoxCORangeCap. Logically, if the value of NoxCORangeCap was greater than 0, one of the CORangeCap or NoxRangeCap values was greater than 0, meaning it was over the emissions cap. Using the find() function, all indices of powerRangeCap where NoxCORangeCap was greater than 0 were set to NaN, so they would not be graphed. The summing process can be seen in appendix 6.8. powerRangeCap was then graphed using the mesh() command.

To find the maxAllowableFR, or max allowable fuel burn rate, for each row of powerRangeCap, a for loop was used. For each iteration, from 1 to the number of rows of powerRangeCap, the index of the maximum value of the row was found. This was possible because NaN is not considered when finding the maximum value, and all unacceptable values of powerRangeCap equaled NaN. This index was then used to find the corresponding index of fuelRange, and that value was added as the i-th value of maxAllowableFR. The aforementioned for loop can be seen in appendix 6.9.

Analysis

Module 3 - Load Data

Upon use of the csvread() function, the data was successfully loaded. Each column of the data was separated into a separate data vector, named according to the text entry in row 1 of each column. Using the datetime() function, the date data was converted from Excel format to Matlab format.

Module 4 - Plotting and Demand

The total demand of January 2nd, 2014 and February 2nd, 2014 display different patterns. The demand for energy throughout the duration of February 2nd, 2014 started to peak consistently at 6:00am and ended around 11:00pm. On January 2nd, the demand peaked minimally around 6:00am but started to consistently peak around 7:00pm. This pattern can be explained because January 2nd is observed as a holiday while February 2nd is not. Furthermore, people are more apt to have a later start in the day thus demanding more energy.

There are four main peaks where the total demand exceeded the total production on March 6th, 2015. Each of the times were at blank blank blank

Module 5 - Emissions

The NOx emissions in February 2015 (Figure 5.5.1) appear to be almost constant around 2.2 ppm, varying slightly sinusoidally. There is a small peak on February 19th and a larger peak on February 28th. This indicates that the processes which emit NOx happened at relatively constant rates throughout February 2015, and these processes may have occurred faster at the aforementioned peak times.

(***Taken from summary***Emissions of NOx and CO were graphed for the month of July 2015. Emissions were relatively constant from approximately July 1 to July 14 and July 29 to the end of July. NOx emissions were approximately 2.2 ppm in those time periods. The CO emissions were close to 2 ppm during those time periods, but they displayed more variance than the NOx emissions, varying from approximately 0.5 ppm to 3 ppm. From approximately July 14 to approximately July 29, the emissions of both CO and NOx were zero. On July 29, a spike of both emissions happened and both emissions went over EPA limits. CO emissions were over the EPA limit of 9 ppm at 6 data points, and their maximum value was 20 ppm. NOx emissions were over the EPA limit of 3 ppm at 17 data points, and their maximum value was 25.8192 ppm. These anomalies in the data can be interpreted as the refinery shutting down from approximately July 15 to July 29, explaining why emissions were zero during that time period. When the refinery reopened on July 29, the emissions were likely much higher because CO and NOx may have built up in the refinery equipment while production was not occurring, and the chemicals were released in large amounts when production initially began again. *****)

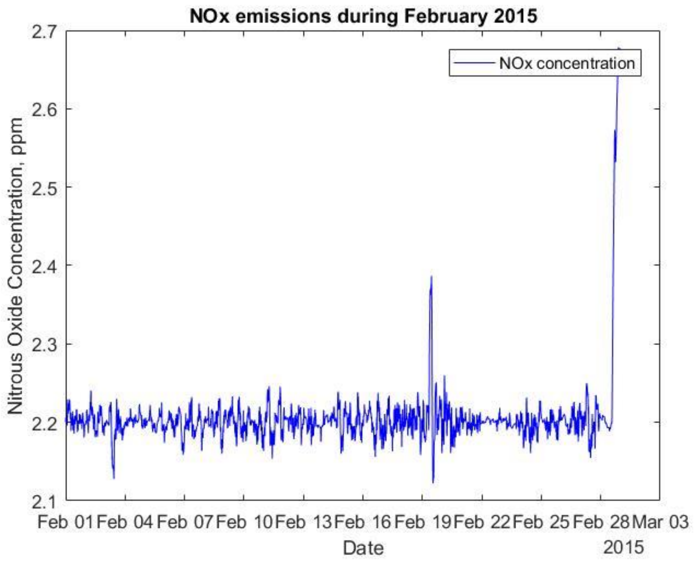


Figure 5.0.1: NOx Emissions in Feb. 2015

The CO emissions (Figure 5.5.2) vary more than the NOx emissions. They oscillate consistently from about 2 to about 2.5 ppm from February 1 to 6, about 1.5 to 2 ppm from February 7 to 13, and about 2.5 to 3.2 ppm from February 14 to 20. After that, there is a peak of approximately 3.6 ppm on February 23rd and a smaller peak of approximately 3 ppm on February 27, followed by a sharp drop in emissions to approximately 0.6 ppm after February 27. The sharp drop in CO emission correlates with a rise in NOx emissions at a similar time. This may indicate that a process emitting CO was used less during that time period, and a process emitting NOx was used instead.

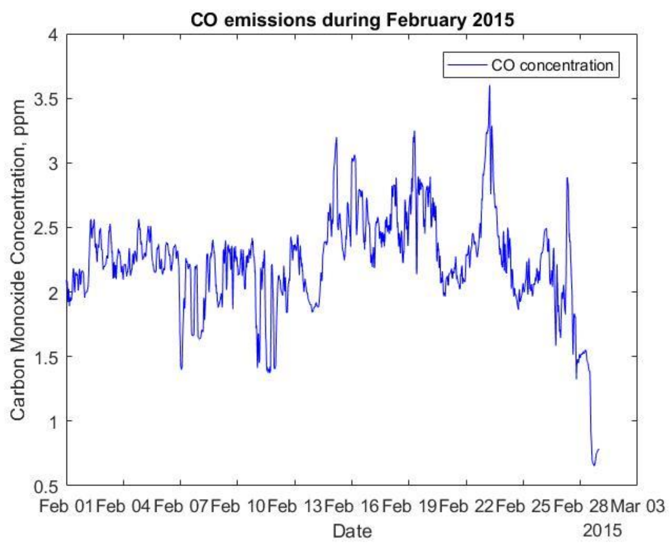


Figure 5.0.2: CO Emissions in Feb. 2015

The emissions for July 2015, which were initially used to run the function, are shown in Appendix 5.2.

Module 6 - Regression and Data Fitting

Module 6 used turbine 2 data in order to evaluate power generation, CO_2 emissions, and NOx emissions over a range of temperatures and fuel burn rates. The resulting curve for power generation showed a powerful correlation between power generated and fuel burned. If more fuel was burned, more power was generated, and this trend was present over all temperature values. However, as seen in figure 0.6.1, power generation had diminishing returns as fuel burn rate increases. This diminishing of returns from the fuel

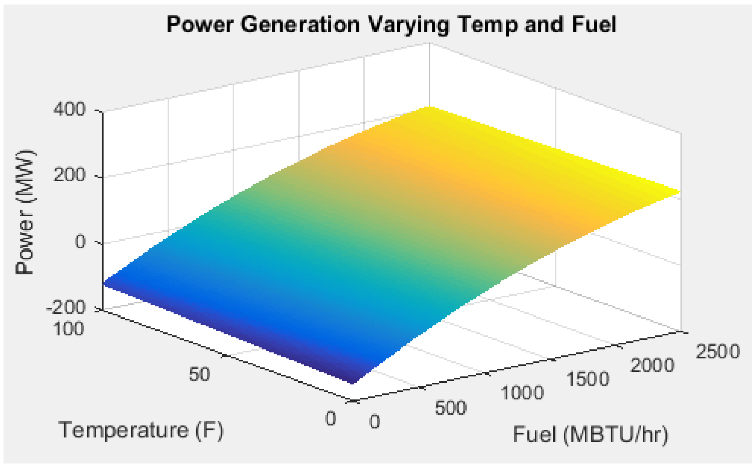


Figure 0.6.1: Graph of power generation (MW) over a range of temperature (F) and fuel burn rate (MBTU/hr) values.

burning is likely to be a result of lost efficiency in the turbine as more fuel is pumped into it per hour, or a sign that the regression equation is not accurate, and could be improved using a different model. Power generation also showed a positive trend in relation to temperature. As temperature increased, power generation increased slightly as well.

Looking at CO_2 and NOx emissions, seen in figure 0.6.2 and 0.6.3, both show similar relationships between emissions rates and fuel burn rate. Initially, emissions are high, followed by a parabolic decrease and then subsequent increase. Both reach their minimum point at approximately 1500 MBTU/hr. While the curves followed the same pattern in

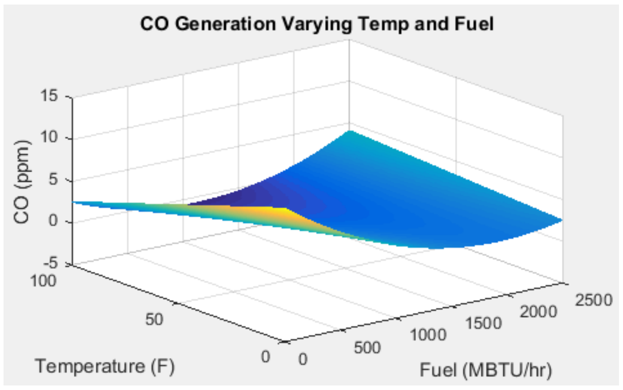


Figure 0.6.2: CO_2 emissions (ppm) over a range of temperature (F) and fuel burn rate (MBTU/hr) values.

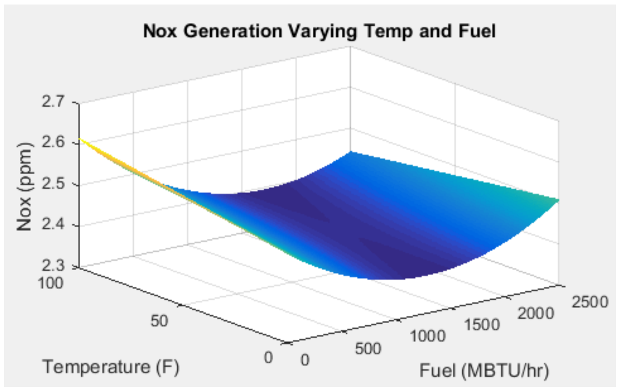


Figure 0.6.3: NOx emissions (ppm) over a range of temperature (F) and fuel burn rate (MBTU/hr) values.

relation to the fuel burn rate, they differ in relation to ambient temperature. As temperature increases, CO_2 emission levels decrease at low fuel consumption levels, and increase at high fuel consumption levels. However, NOx emissions are the exact opposite. At low fuel consumption rates, the emissions increase with temperature, but at high fuel consumption rates, NOx emissions decrease as temperature rises.

It was found that for most ambient temperatures, the fuel burn rate that produced the most power while staying within acceptable emissions limits was 2500 MBTU/hr. This would probably increase if higher ranges of fuel burn rate were tested, however, extrapolation of Figure 0.6.2 and 0.6.3 suggests that there will be a limit where the fuel burn rate creates unacceptable CO_2 and NOx emissions. A graph of all power generation values that meet emissions regulations can be seen in figure 0.6.4.

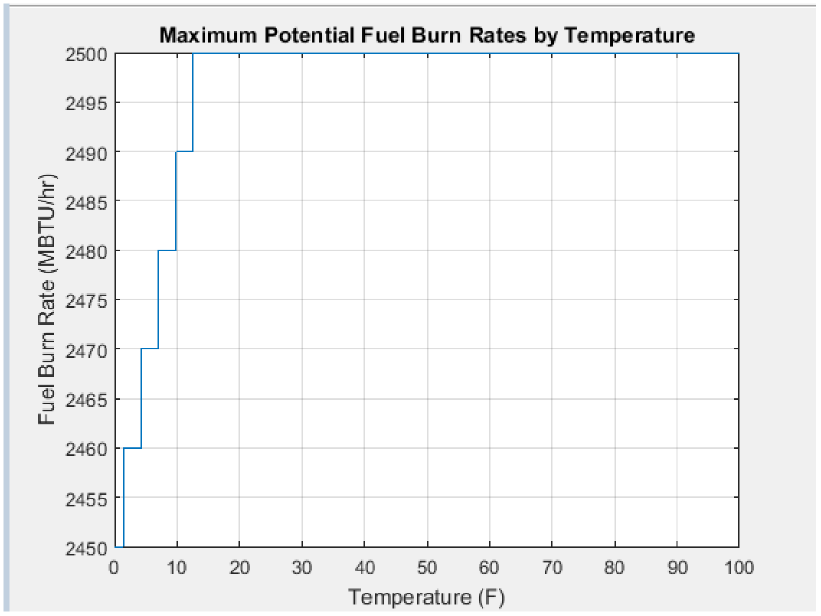


Figure 0.6.4: Graph of the maximum fuel burn rates of turbine 2 at each ambient temperature.

It can be seen from figure 0.6.4 that, over most temperature values (12°F to 100°F) running the turbines with a fuel burn rate of 2500 MBTU/hr will produce the most power while also generating acceptable levels of CO_2 and NOx pollution. A graph of all power production levels with acceptable CO_2 and NOx emissions can be seen in appendix 6.10.

*****Taken from summary****Regression data was used from February 2013, power generation, NOx emissions, and CO_2 emissions were graphed over a temperature range of 0 to 100 °F and a fuel burn range of 0 to 2500 MBTU/hr. Using this data, it was found that power generation varies directly with temperature and fuel burn rate, however with diminishing returns from fuel burn rate. It was also found that the lowest CO_2 and NOx emission rates were near the center of the fuel burn rate, around 1500 MBTU/hr. Only low fuel burn rates exceeded the emissions caps for CO_2 and NOx, however it can be extrapolated that increasing fuel burn rate higher than 2500 MBTU/hr will begin to violate emissions caps. It was also found that the fuel burn rate that produced the most power with acceptable emissions rates was 2500 MBTU/hr, or within 50 MBTU/hr of 2500 MBTU/hr. A graph of this data can be found in figure 0.6.4.******

Conclusion

Module 4 results, keep brief but informative

Emissions of NOx and CO were graphed for the month of July 2015. The graph showed that emissions were zero from approximately July 14 to July 29, and a huge spike in emissions occurred on July 29, during which emissions were over EPA limits for a small range of data points. Outside of those dates, the emissions were relatively constant.

Power generation, NOx emissions, and CO_2 emissions were graphed over a range of temperature and fuel burn data. Using this data, it was found that power generation varies directly with temperature and fuel burn rate, but with diminishing returns. It was also found that the lowest CO_2 and NOx emission rates were near the center of the fuel burn rate, around 1500 MBTU/hr. Mostly low fuel burn rates exceeded the emissions caps for CO_2 and NOx, however it can be extrapolated that increasing fuel burn rate higher than 2500 MBTU/hr will begin to violate emissions caps. It was also found that the fuel burn rate that produced the most power with acceptable emissions rates was 2500, or within 50 MBTU/hr of 2500 MBTU/hr.

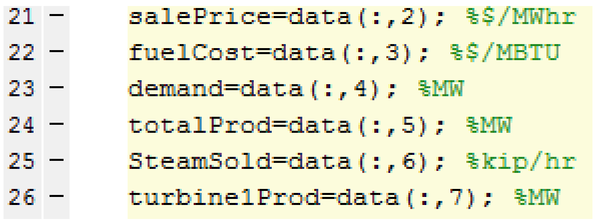
Appendix

Appendix 3.1

Table 3.1: A list of the 22 data sets provided, with units and a description for each.

Name	**Unit**	**Description**
date		Date and time of data collection, Excel standard
salePrice	\$/MWhr	Sale price of energy produced
fuelCost	\$/MBTU	Price of fuel used in running turbines and duct burner
demand	MW	Power demand by MISO
totalProd	MW	Total power produced by the plant.
steamSold	kip/hr	Quantity of steam sold by plant.
turbine1Prod	MW	Power produced, turbine 1
turbine1Fuel	hSCF/hr	Fuel consumed, turbine 1
turbine1CO	ppm	CO_2 produced, turbine 1
turbine1Nox	ppm	NO_x produced, turbine 1
turbine2Prod	MW	Power produced, turbine 2
turbine2Fuel	hSCF/hr	Fuel consumed, turbine 2
turbine2CO	ppm	CO_2 produced, turbine 2
turbine2Nox	ppm	NO_x produced, turbine 2
db1Fuel	hSCF/hr	Fuel consumed, duct burner 1
db1Steam	kip/hr	Steam produced, duct burner 1
steamProd	MW	Power produced, steam turbine
steamFuel	kip/hr	Steam consumed, steam turbine
natGasDensity	BTU/SCF	Energy density of natural gas
temp	^oF	Ambient temperature

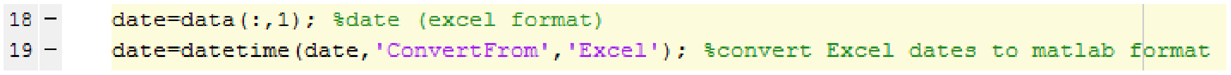
Appendix 3.2



```
salePrice=data(:,2); %$/MWhr
fuelCost=data(:,3); %$/MBTU
demand=data(:,4); %MW
totalProd=data(:,5); %MW
SteamSold=data(:,6); %kip/hr
turbine1Prod=data(:,7); %MW
```

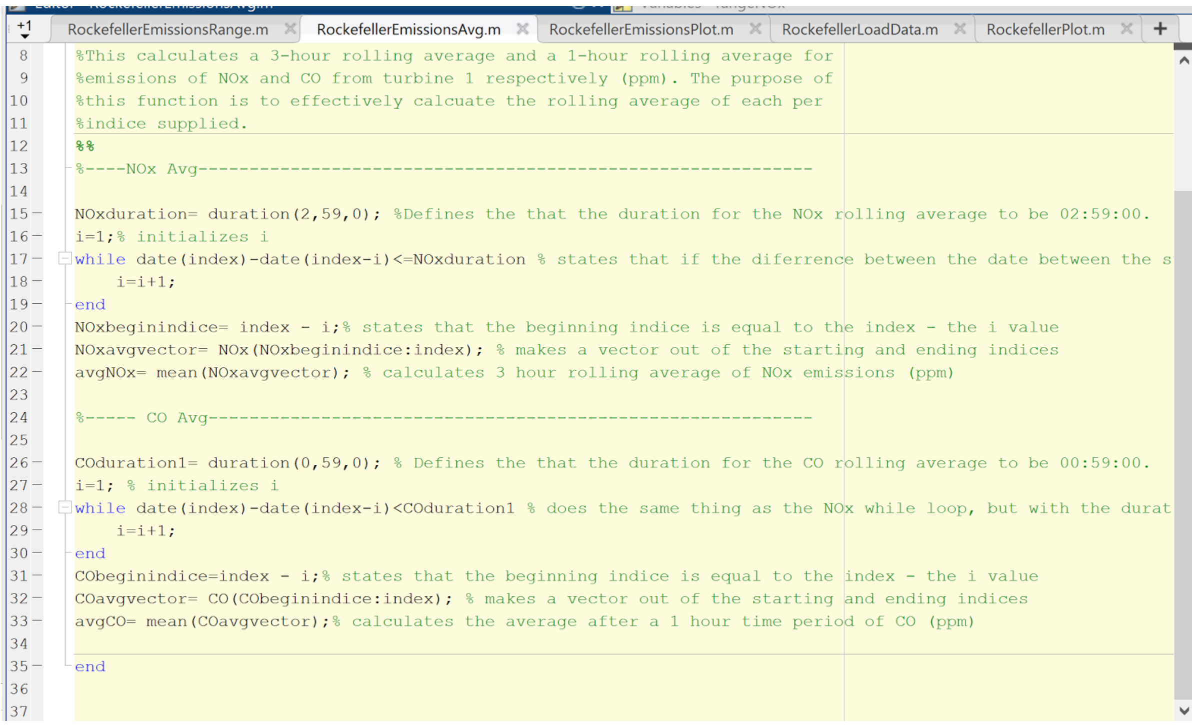
Figure 3.1: Code separating the data.csv matrix into separate columns.

Appendix 3.3



```
date=data(:,1); %date (excel format)
date=datetime(date,'ConvertFrom','Excel'); %convert Excel dates to matlab format
```

Figure 3.2: Code using datetime() to convert date data from Excel standard to Matlab format.

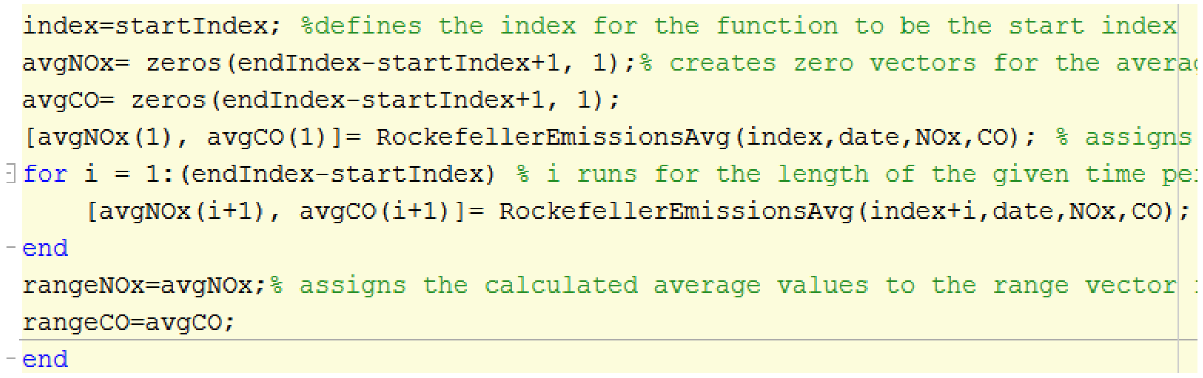

RockefellerEmissionsRange.m
RockefellerEmissionsAvg.m
RockefellerEmissionsPlot.m
RockefellerLoadData.m
RockefellerPlot.m

```
%This calculates a 3-hour rolling average and a 1-hour rolling average for
%emissions of NOx and CO from turbine 1 respectively (ppm). The purpose of
%this function is to effectively calcuate the rolling average of each per
%indice supplied.
%%
%----NOx Avg-------------------------------------------------------------

NOxduration= duration(2,59,0); %Defines the that the duration for the NOx rolling average to be 02:59:00.
i=1;% initializes i
while date(index)-date(index-i)<=NOxduration % states that if the diferrence between the date between the s
    i=i+1;
end
NOxbeginindice= index - i;% states that the beginning indice is equal to the index - the i value
NOxavgvector= NOx(NOxbeginindice:index); % makes a vector out of the starting and ending indices
avgNOx= mean(NOxavgvector); % calculates 3 hour rolling average of NOx emissions (ppm)

%----- CO Avg------------------------------------------------------------

COduration1= duration(0,59,0); % Defines the that the duration for the CO rolling average to be 00:59:00.
i=1; % initializes i
while date(index)-date(index-i)<COduration1 % does the same thing as the NOx while loop, but with the durat
    i=i+1;
end
CObeginindice=index - i;% states that the beginning indice is equal to the index - the i value
COavgvector= CO(CObeginindice:index); % makes a vector out of the starting and ending indices
avgCO= mean(COavgvector);% calculates the average after a 1 hour time period of CO (ppm)

end
```

Appendix 5.1.1

Figure 5.1.1: RockefellerEmissionsAvg function

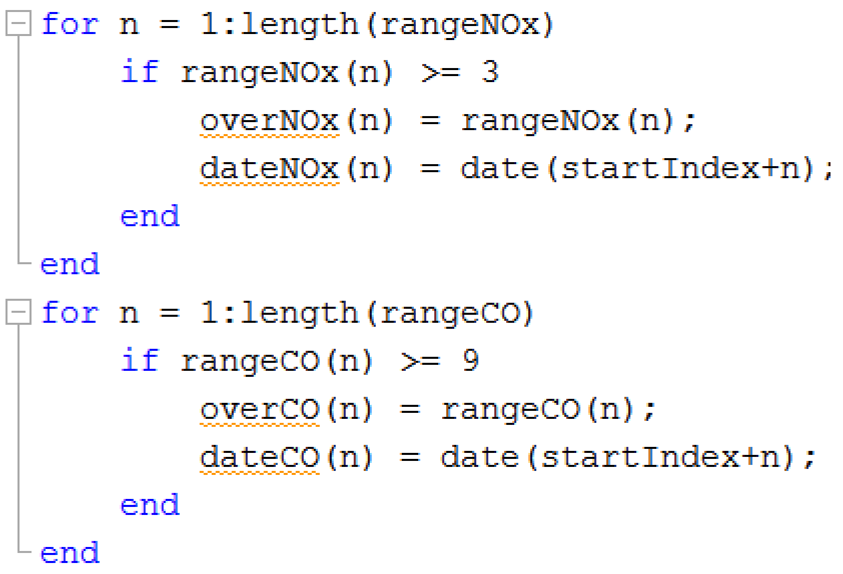
Appendix 5.1.2



```
index=startIndex; %defines the index for the function to be the start index
avgNOx= zeros(endIndex-startIndex+1, 1);% creates zero vectors for the avera
avgCO= zeros(endIndex-startIndex+1, 1);
[avgNOx(1), avgCO(1)]= RockefellerEmissionsAvg(index,date,NOx,CO); % assigns
for i = 1:(endIndex-startIndex) % i runs for the length of the given time pe
    [avgNOx(i+1), avgCO(i+1)]= RockefellerEmissionsAvg(index+i,date,NOx,CO);
end
rangeNOx=avgNOx;% assigns the calculated average values to the range vector
rangeCO=avgCO;
end
```

Figure 5.1.2: Code for the RockefellerEmissionsRange function

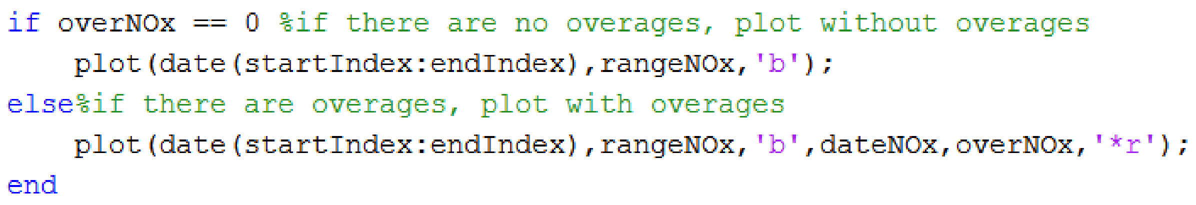
Appendix 5.1.3



```
for n = 1:length(rangeNOx)
    if rangeNOx(n) >= 3
        overNOx(n) = rangeNOx(n);
        dateNOx(n) = date(startIndex+n);
    end
end
for n = 1:length(rangeCO)
    if rangeCO(n) >= 9
        overCO(n) = rangeCO(n);
        dateCO(n) = date(startIndex+n);
    end
end
```

Figure 5.1.3 Code creating overNOx/dateNOx and overCO/dateCO

Appendix 5.1.4



```
if overNOx == 0 %if there are no overages, plot without overages
    plot(date(startIndex:endIndex),rangeNOx,'b');
else%if there are overages, plot with overages
    plot(date(startIndex:endIndex),rangeNOx,'b',dateNOx,overNOx,'*r');
end
```

Figure 5.1.4 Code determining whether to attempt to plot overage vectors

Appendix 5.2.1

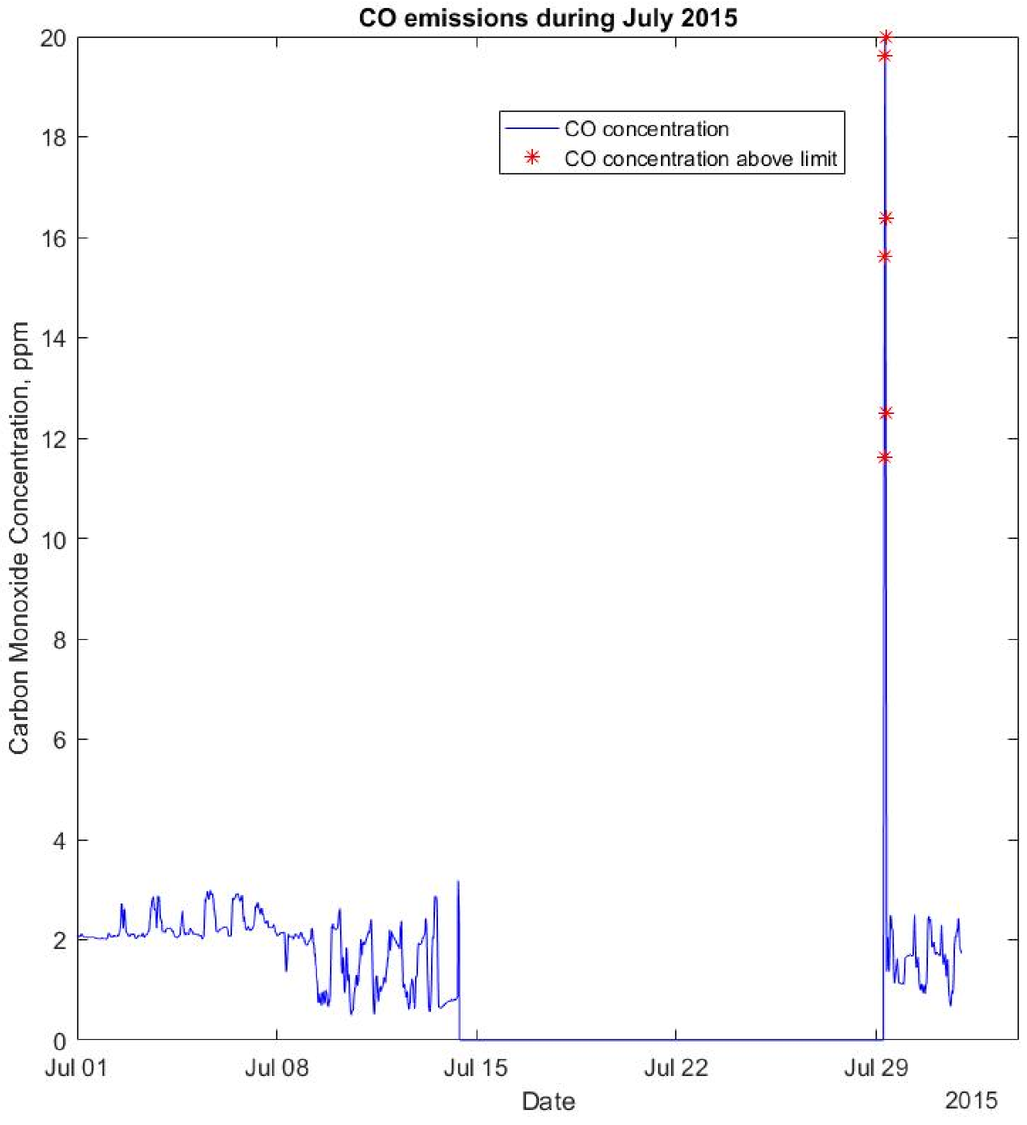


Figure 5.1.1: Emissions of CO during July 2015

Appendix 5.2.2

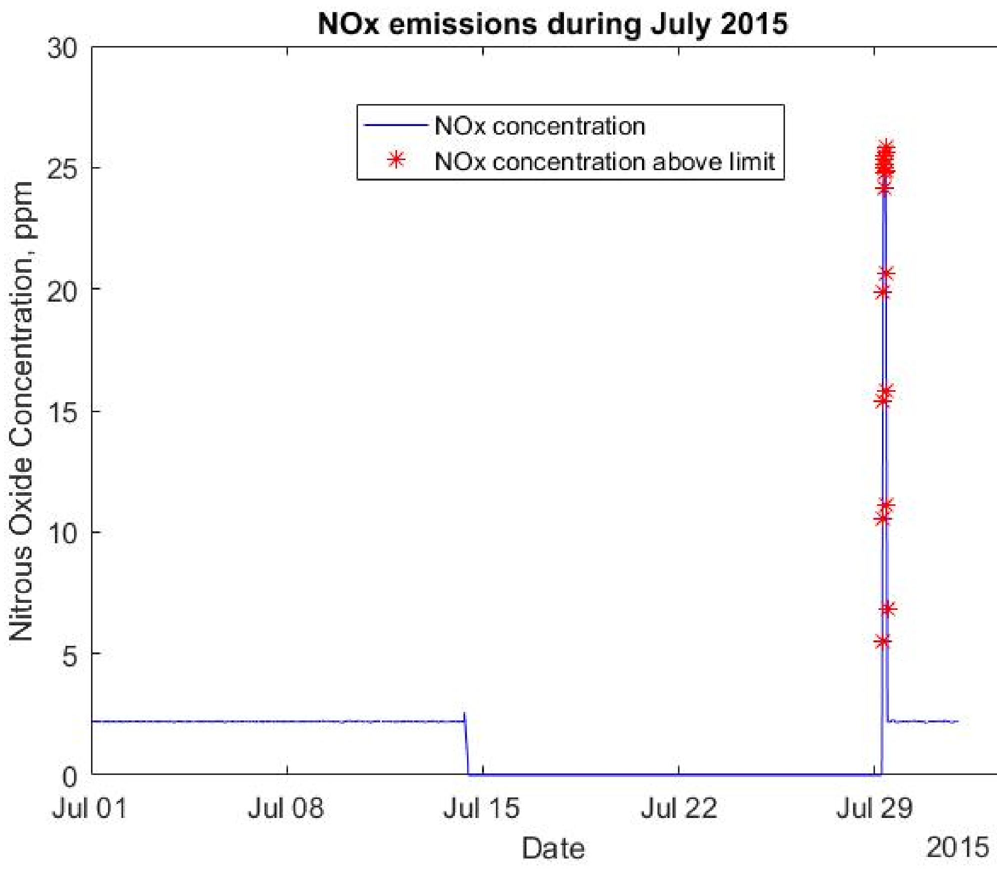
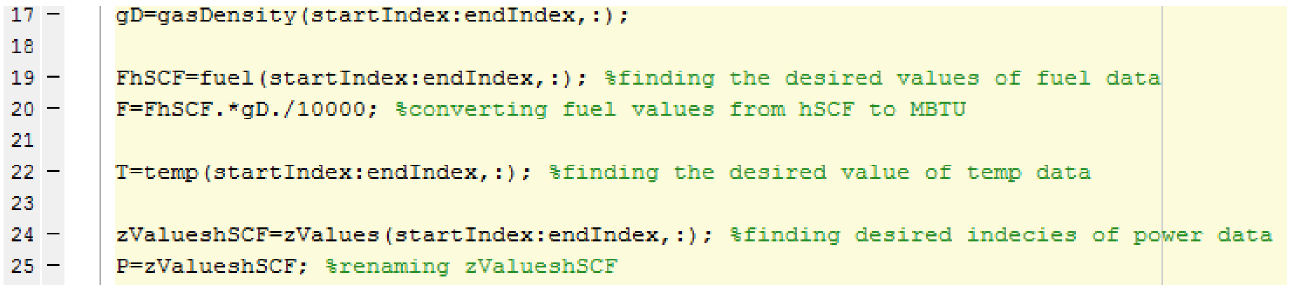


Figure 5.1.2: Emissions of NOx during July 2015

Appendix 6.1



```
17 -   gD=gasDensity(startIndex:endIndex,:);
18
19 -   FhSCF=fuel(startIndex:endIndex,:); %finding the desired values of fuel data
20 -   F=FhSCF.*gD./10000; %converting fuel values from hSCF to MBTU
21
22 -   T=temp(startIndex:endIndex,:); %finding the desired value of temp data
23
24 -   zValueshSCF=zValues(startIndex:endIndex,:); %finding desired indecies of power data
25 -   P=zValueshSCF; %renaming zValueshSCF
```

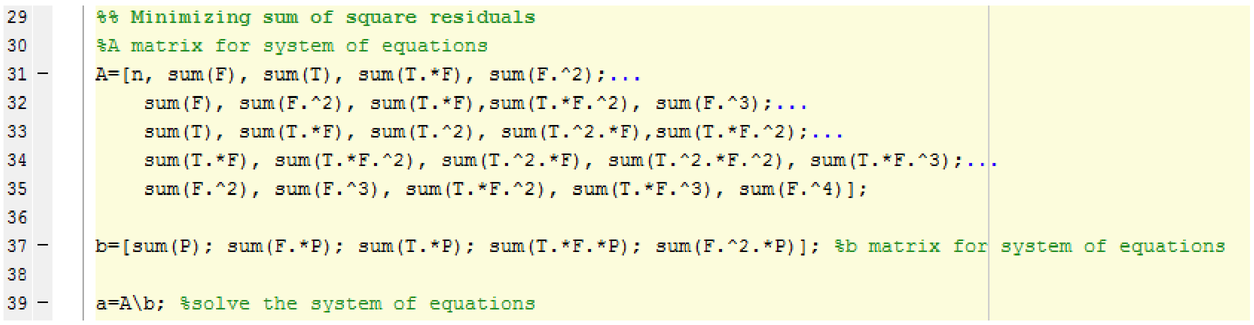
Figure 6.1: Finding relevant data for gasDensity, fuel, temp, and zValues using startIndex and endIndex. The fuel value is also converted from hSCF to MBTU in line 20.

Appendix 6.2

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

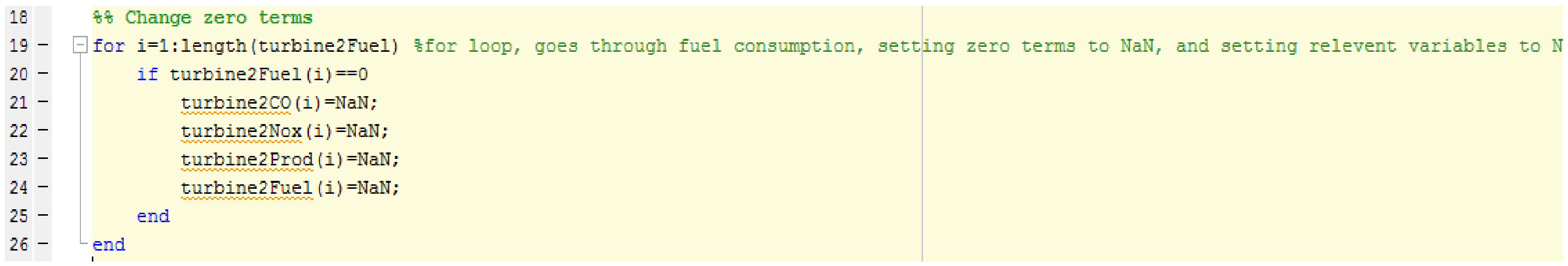
Figure 6.2: The system of equations provided to fit a function to the zValues data, solving for the regression constants a_0 thru a_4.

Appendix 6.3



```
29      %% Minimizing sum of square residuals
30      %A matrix for system of equations
31 -    A=[n, sum(F), sum(T), sum(T.*F), sum(F.^2);...
32          sum(F), sum(F.^2), sum(T.*F),sum(T.*F.^2), sum(F.^3);...
33          sum(T), sum(T.*F), sum(T.^2), sum(T.^2.*F),sum(T.*F.^2);...
34          sum(T.*F), sum(T.*F.^2), sum(T.^2.*F), sum(T.^2.*F.^2), sum(T.*F.^3);...
35          sum(F.^2), sum(F.^3), sum(T.*F.^2), sum(T.*F.^3), sum(F.^4)];
36
37 -    b=[sum(P); sum(F.*P); sum(T.*P); sum(T.*F.*P); sum(F.^2.*P)]; %b matrix for system of equations
38
39 -    a=A\b; %solve the system of equations
```

Figure 6.3: Code calculating the A and b matrices for the regression, given by the system of equations in appendix 6.2. The regression constants, a, are then calculated in line 39.

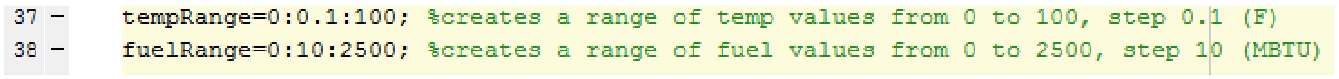


```
%% Change zero terms
for i=1:length(turbine2Fuel) %for loop, goes through fuel consumption, setting zero terms to NaN, and setting relevent variables to N
    if turbine2Fuel(i)==0
        turbine2CO(i)=NaN;
        turbine2Nox(i)=NaN;
        turbine2Prod(i)=NaN;
        turbine2Fuel(i)=NaN;
    end
end
```

Appendix 6.4

Figure 6.4: Code using a for loop to find indices of turbine2Fuel that equal 0, and changing all relevant data at that index to NaN in order to create a better regression.

Appendix 6.5



```
tempRange=0:0.1:100; %creates a range of temp values from 0 to 100, step 0.1 (F)
fuelRange=0:10:2500; %creates a range of fuel values from 0 to 2500, step 10 (MBTU)
```

Figure 6.5: Creating two vectors, tempRange, ranging from 0 to 100 °F with a step size of 0.1, and fuelRange, ranging from 1 to 2500 MBTU with a step of 10.

Appendix 6.6

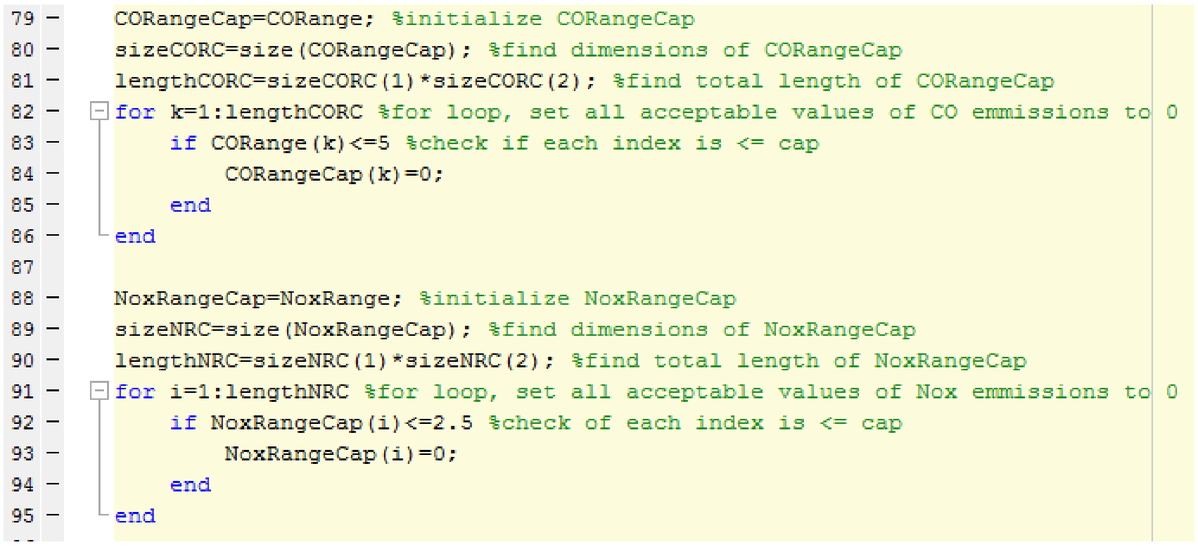
$$P(F,T) = a_0 + a_1F + a_2T + a_3FT + a_4F^2;$$

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

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

Figure 6.6: Formulas to solve for power (MW), NOx emissions (ppm), and CO emissions (ppm) at a given temperature T (°F) and given fuel consumption F (MBTU). The coefficients a_0 thru a_4 are regression coefficients calculated by the function RockefellerRegression.m.

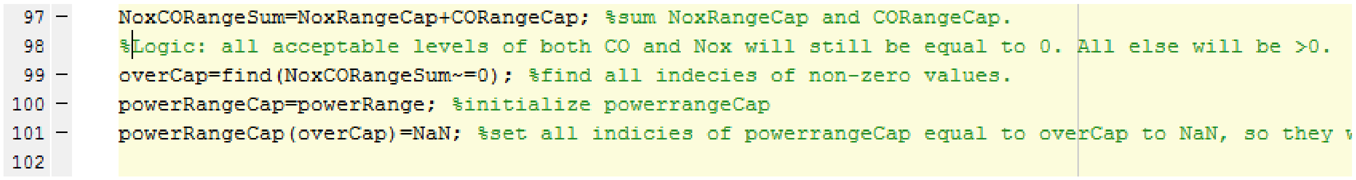
Appendix 6.7



```
CORangeCap=CORange; %initialize CORangeCap
sizeCORC=size(CORangeCap); %find dimensions of CORangeCap
lengthCORC=sizeCORC(1)*sizeCORC(2); %find total length of CORangeCap
for k=1:lengthCORC %for loop, set all acceptable values of CO emmissions to 0
    if CORange(k)<=5 %check if each index is <= cap
        CORangeCap(k)=0;
    end
end

NoxRangeCap=NoxRange; %initialize NoxRangeCap
sizeNRC=size(NoxRangeCap); %find dimensions of NoxRangeCap
lengthNRC=sizeNRC(1)*sizeNRC(2); %find total length of NoxRangeCap
for i=1:lengthNRC %for loop, set all acceptable values of Nox emmissions to 0
    if NoxRangeCap(i)<=2.5 %check of each index is <= cap
        NoxRangeCap(i)=0;
    end
end
```

Figure 6.7: Code initializing CORangeCap and NoxRangeCap, and evaluating each against emissions caps for CO_2 and NOx. If the value was under the limits of 5ppm and 2.5ppm, respectively, the value of that indice was set to 0.

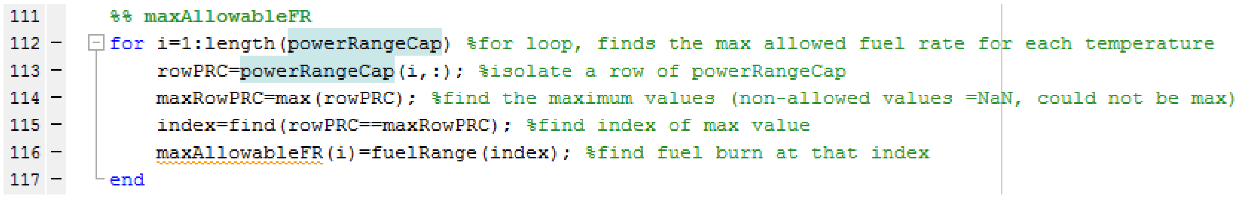
Appendix 6.8



```
NoxCORangeSum=NoxRangeCap+CORangeCap; %sum NoxRangeCap and CORangeCap.
%Logic: all acceptable levels of both CO and Nox will still be equal to 0. All else will be >0.
overCap=find(NoxCORangeSum~=0); %find all indecies of non-zero values.
powerRangeCap=powerRange; %initialize powerrangeCap
powerRangeCap(overCap)=NaN; %set all indicies of powerrangeCap equal to overCap to NaN, so they
```

Figure 6.8: Code where CORangeCap and NoxRangeCap are summed. Because of the process explained in appendix 6.7, of the sum is greater than 0, the index is set to NaN for powerRangeCap, as that index is over either the NOx or CO_2 emissions cap.

Appendix 6.9



```
%% maxAllowableFR
for i=1:length(powerRangeCap) %for loop, finds the max allowed fuel rate for each temperature
    rowPRC=powerRangeCap(i,:); %isolate a row of powerRangeCap
    maxRowPRC=max(rowPRC); %find the maximum values (non-allowed values =NaN, could not be max)
    index=find(rowPRC==maxRowPRC); %find index of max value
    maxAllowableFR(i)=fuelRange(index); %find fuel burn at that index
end
```

Figure 6.9: Code for finding the highest allowed power generation at each ambient temperature (F), and the method for finding the corresponding fuel burn rate (MBTU/hr).

Appendix 6.10

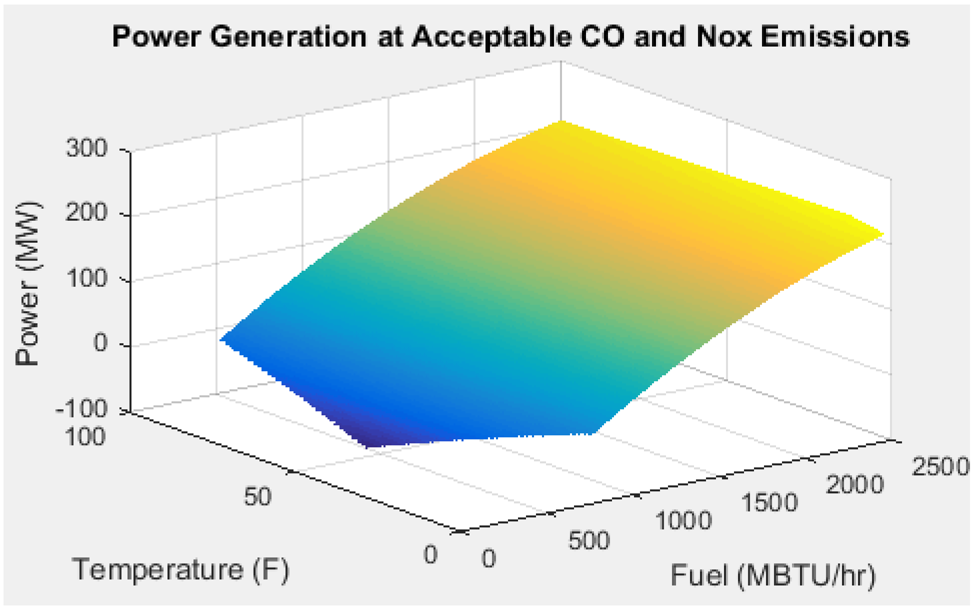


Figure 6.10: Graph displaying power generation only at acceptable CO_2 and NOx emissions rates.

References

[1]"BP Indiana", *BP*, 2017. [Online]. Available: http://www.bp.com/en_us/bp-us/where-we-operate/bp-indiana.html. [Accessed: 18- Apr- 2017].

[2]"Whiting Refinery: Facility Fact Sheet", *BP*, 2017. [Online]. Available: http://www.bp.com/content/dam/bp-country/en_us/PDF/Asphalt/Whiting_Facility_Fact_Sheet.pdf. [Accessed: 18- Apr- 2017].

[3]"Who We Are", *BP*, 2017. [Online]. Available: http://www.bp.com/en_us/bp-us/who-we-are.html. [Accessed: 18- Apr- 2017].

[4]"OSHA Technical Manual (OTM) | Section IV: Chapter 2 - Petroleum Refining Process | Occupational Safety and Health Administration", *Osha.gov*, 2017. [Online]. Available: https://www.osha.gov/dts/osta/otm/otm_iv/otm_iv_2.html. [Accessed: 18- Apr- 2017].

[5]"What is cogeneration?", *Centrax Gas Turbines*, 2017. [Online]. Available: http://www.centraxgt.com/about-us/what-is-cogeneration. [Accessed: 20- Apr- 2017].

[6]"BP At a Glance", *BP*, 2017. [Online]. Available: http://www.bp.com/en/global/corporate/about-bp/bp-at-a-glance.html. [Accessed: 18- Apr- 2017].

[7]NWI Times, "BP acquires Whiting Clean Energy", 2008.

[8]"Gulf Oil Spill", *Ocean Portal | Smithsonian*, 2017. [Online]. Available: http://ocean.si.edu/gulf-oil-spill. [Accessed: 23- Apr- 2017].

[9]"Deepwater Horizon Oil Spill", *Response.restoration.noaa.gov*, 2017. [Online]. Available: http://response.restoration.noaa.gov/oil-and-chemical-spills/significant-incidents/deepwater-horizon-oil-spill. [Accessed: 23- Apr- 2017].

[10]C. Ezoezue, S. Maouyo, J. Moline, R. Turlington and E. Umeike, "Policy Brief: Deepwater Horizon Oil Spill", MIT OCW, 2013.

[11]K. Brennan, "A Stakeholder Analysis of the BP Oil Spill and the Compensation Mechanisms Used to Minimize Damage", 2013.

[12]"Report Regarding the Causes of the April 20, 2010 Macondo Well Blowout", The Bureau of Ocean Energy Management, Regulation, and Enforcement, 2011.

[13]"USA TODAY: Stock Info (BP PLC)", *USA TODAY*, 2017. [Online]. Available: https://www.usatoday.com/money/lookup/stocks/BP/. [Accessed: 23- Apr- 2017].

[14]"Deepwater Horizon oil spill of 2010", *Encyclopædia Britannica*. Encyclopædia Britannica, inc., 2017.

[15]The Washington Post, "The Deepwater Horizon spill may have caused 'irreversible' damage to Gulf Coast marshes", 2016.

[16]USA Today, "BP's Deepwater Horizon costs total $62B", 2016.