MathWorks Math Modeling Challenge 2020

Troy High School

Team # 13638 Fullerton, California

Coach: Don Allen

Students: Soham Bose, Ho Lyun Jeong, Jimmy Li, Rahil Shah

M3 Challenge Technical Computing Award RUNNER UP—$2,000 Team Prize

JUDGE COMMENTS

Specifically for Team # 13638:
This paper used multiple technical computing methods - curve fitting, numerically ODE solvers, optimization, and a greedy maximal coverage algorithm. The choices were pragmatic and effective, e.g., finding optimal parameters or optimizing a function they wrote themselves. Overall, the explanations were not as strong as the top paper and, in some places, incomplete. The paper did make use of effectively visualizations generated with code. The team had opportunities to leverage their code for further exploration which they did not take.

Overall Judging Perspective for Technical Computing Submissions:
The use of technical computing in the final papers was judged on its effectiveness in advancing a papers' modeling, its creativity, and how it was communicated. We rewarded papers where technical computing was used in an essential way, and did not simply replace functionality which could have been implemented in a spreadsheet or on a calculator. We also rewarded clear explanations, even if the underlying algorithm was relatively simple. This year we noticed that technical computing was used in many papers to enhance presentation: some beautiful plots were used to effectively communicate student ideas and final solutions. Such uses of technical computing were also rewarded. Finally, one of the benefits of implementing a model in code is that it is very easy to change modeling choices or parameters to see how these choices effect the final problem solution. Teams which took advantage of this (through sensitivity analysis, testing multiple models, etc.) were rewarded.

Keep on Trucking: U.S. Big Rigs Turnover from Diesel to Electric

Team 13638

1 Executive Summary

Electric vehicles are the way of the future. There has been a concerted effort in recent history to transition from natural gas, gasoline, and diesel fuel to more sustainable and Eco-friendly energy types. The problem we were presented with included understanding how the transition to electric would occur and using models in order to make the transition as seamless as it could be.

The first problem, we were tasked with tackling was deducing the percentage of semi-trucks that will be electric in the future. To model this, we created a model derived from the SIR model which is commonly used to model the spread of an epidemic since the spread of information about and interest in electric semi-trucks can be assumed to behave similarly to an epidemic. By using a number of modeling and curve-fitting techniques such as using least square curve fitting, binary search, and a system of ordinary differential equations. Solving the system of ordinary differential equations, we arrived at the prediction of that 2.74% of semi-trucks will be electric in 5 years, 11.55% in 10 years, and 88.52% in 20 years from 2020.

For second problem, we were tasked with determining the location and amount of charging stations assuming that all current trucks became electric trucks. To tackle this problem effectively, we utilized the data provided by the M3 Challenge concerning battery charging information, traffic information, and various other factors. With these factors in mind, we modelled the traffic to follow a wave function and used a model based on the maximum location coverage problem. This provided us a robust way to model the amount of traffic and from that, the amount of charging stations and their location. In addition, the problem also tasked us with finding the amount of chargers that each charging station would need. Similar to the location of the chargers, we used the traffic modeled by a wave function to determine the amount of people at each station. From this, we were able to factor in charging information and various other factors to determine the amount of charging stations needed at each charging location.

For the third problem, we were tasked with ranking which areas would be most beneficial to transition to electric trucks first. We used root mean squared of the Z scores after normalizing the data of our four indexes. The four indexes

were: Community benefit index, community wealth index, community environmentalist index, utilitarian index. Throughout this, we realized that LA to SF was the best corridor.

Contents

2 Introduction

As the electric vehicle industry has come into full focus with the emergence of Tesla, every niche has begun to see the benefits. Electric cars have become far more common because of their efficiency and their Eco-friendly nature. Unfortunately, the trucking industry has not been able to make the same strides as the commercial car industry. Class 8 trucks in general average between 5 and 8 mpg[6], which is far worse than UPS says about new EVs. They project their new electric vehicles to be about 52 mpg efficient equivalent.[10] It is extremely likely that soon, all trucks on the road will be electric. Making that transition as smooth and logical is possible is key in helping perfect the industry, which will be both good for the economy and great for the environment. Strategically placing the charging stations and chargers will minimize the up-front cost of transitioning between diesel and electric, which makes it more lucrative for companies and individuals to transfer. Tesla has already announced its own electric semi truck that is already in production and expected to release late this year. The release of the Tesla semi truck signifies the beginning of an era of more Eco-friendly society. However, not everyone can afford the changeover instantly, so it is important to know which geographic areas are more suited to the transition to electric vehicles.

3 Part I

3.1 Problem Definition

Electric trucks represent a path toward the future. In this section, we will outline a model for predicting the transition rate from diesel to electric semis. The results will show the percentage of semis that will have completed the transition by 2025, 2030, and 2040.

3.2 Assumptions and Justifications

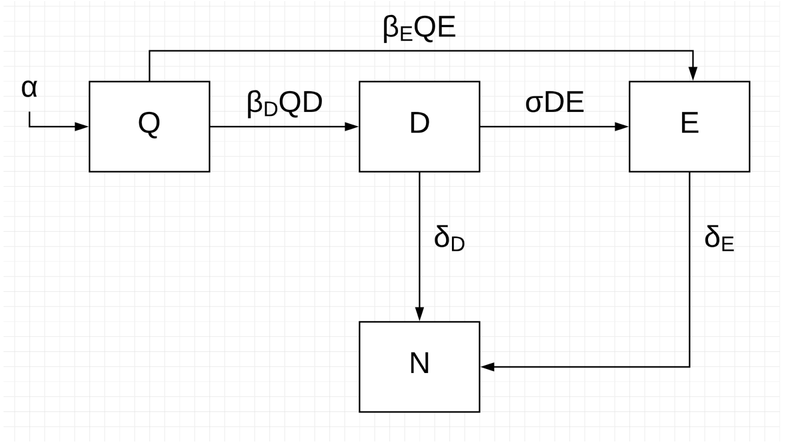
1. The main assumption we made is that the spread of information is similar to the spread of epidemics. We reason that spread of information can be likened to an infection of the mind, the comparison is fairly accurate. There are multiple research papers that support this claim [2][4][5][8].

2. We assumed that semi-trucks are either diesel or electric because diesel semis make up 97% of the semis on the road right now. Furthermore, there is an extremely low likelihood of any other gasoline based fuel-types being used in the future because of the poor environmental effects and poor efficiency when compared to modern diesel.[7]

3. We assume the difference in lifespan of an electric semi and a diesel semi is negligible. This assumption comes from extensive research supported by UPS and Forbes.[3][10]

3.3 Model

The model we used to find the percentage of electric semis in the future is based on a SIR model which is commonly used to model the spread of an epidemic. Since the number of electric trucks is directly affected by the spread of information about and interest in electric trucks, according to assumption 1, it is reasonable to predict the percentage of electric semis using an SIR model. Below is a flowchart detailing our theoretical model.



In this model, Q represents the production of semis, D and E represent the number of diesel and electric semis, respectively, and N represents the number of semis which have reached the end of their life span. There is a continuous influx of newly produced semis, and those are then categorized to either diesel- or electricity-powered. We also (observed) that some of the diesel semi drivers or owners may decide to switch to an electric one, of which the rate is proportional to both the number of diesel and electric semis (according to assumption 1).

Due to the difficulty of implementing this model, we took a number of alternative routes and began with a baseline much simpler than the one mentioned above. First, according to assumption 2, since all semis are either diesel or electric, the production of diesel semis is considered to be the total production of semis minus the production of electric semis.

$$\frac{dD}{dt} = \alpha - \frac{dE}{dt} \tag{1}$$

Then, to establish a baseline for $\frac{dE}{dt}$, we created a logistic curve based on NAFTA's projection of their electric truck production as a percentage of total truck production [11]. The ODE version of the formula for a logistic curve is as follows:

$$\frac{d}{dt}\left[\frac{E}{P}\right] = \lambda\frac{E}{P}(M - \frac{E}{P}) \tag{2}$$

Here $\frac{E}{P}$ represents the proportion of the production of electric semis to the total production of semis, λ represents the rate of growth, and M represents the carrying capacity. In our baseline for our model, the carrying capacity is simply the maximum the proportion can be, *i.e.* 1. Integrating the differential equation and plugging in 1 for M, we have:

$$\frac{E}{P} = \frac{E_0/P_0}{E_0/P_0 + (1 - E_0/P_0) \cdot e^{-\lambda t}} \tag{3}$$

Since there is only one unknown, λ, this function is easy to perform curve fitting on. We wrote a simple snippet of code to perform a binary search that finds the value for λ that minimizes the root-mean-square error. We used least square curve fitting in order to acquire the data for future production of electric vehicles. Least square curve fitting is a common method in data analysis in which the root-mean-square error is minimized in order to create a more accurate estimation for the curve. The equation for root-mean-square error is:

$$E(f) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2} \tag{4}$$

Since the square root of an increasing function is still increasing, we can improve our efficiency by skipping the evaluation of the square root and compare the sum of mean squared instead, which would still give us the same answer. The new error function is:

$$E^2(f) = \sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{5}$$

After running the program, we arrived at a λ value of 0.25 for our baseline. To convert this baseline logistic curve to our theoretical model, we then considered the following differential equation:

$$\frac{dE}{dt} = \beta_E E \tag{6}$$

We algebraically manipulated our baseline logistic curve:

$$\frac{d}{dt}[\frac{E}{P}] = \frac{\frac{dE}{dt}P - \frac{dP}{dt}E}{P^2} = \lambda\frac{E}{P}(1 - \frac{E}{P}) = \lambda\frac{E}{P} - \lambda\frac{E^2}{P} \tag{7}$$

$$\frac{dE}{dt} = \lambda E(1 - \frac{E}{P} - \frac{dP}{dt} \cdot \frac{1}{P} \cdot \frac{1}{\lambda E}) \tag{8}$$

The last term is multiplied to a factor of $\frac{1}{P}$, 1 over the total number of semis, therefore it is small enough to be negligible. We then arrived at:

$$\beta_E = \lambda\left(1 - \frac{E}{P}\right) \tag{9}$$

However, since E is time dependent, β_E also becomes time dependent. To estimate a time independent value of β_E, we borrowed the concept of a predictor in Improved Euler Method. We plugged the initial values of E and P for β_E

as a predictor. The initial value for P is found from the American Trucking Associations website [9], and the initial value for E is found by multiplying the production rate of electric trucks in 2019 [11] to P_0.

$$\beta_E = \lambda\left(1 - \frac{E_0}{P_0}\right) \tag{10}$$

The differential equation was then integrated for a predictor $\hat{E}$, and the average value of E was then estimated using the average function value formula:

$$\bar{E} \approx \frac{1}{b-a}\int_a^b \hat{E}dt \tag{11}$$

The new β_E value was evaluated:

$$\beta_E = \lambda\left(1 - \frac{\bar{E}}{P}\right) \tag{12}$$

Finally, we adjusted the value of β_E by dividing it by the average production of semis per year $\bar{\alpha}$ so that it is compatible with our theoretical model.

Another component of our theoretical model was α, the total annual production of semis. We again used least square curve fitting in order to acquire the data for future production of electric vehicles. Using open source least square curve fitting libraries, we predicted the future production of electric vehicles through a sine function:

$$\alpha = \sin(0.8909t + 219.3202) \cdot 45609.0704 + 143645.8527 \tag{13}$$

Completing the last part of our baseline model, we let the production of diesel semis be the total production of semis minus the production of electric semis:

$$\frac{dD}{dt} = \alpha - \frac{dE}{dt} \tag{14}$$

While the baseline model only accounts for the production of semi-trucks, there are other factors such as lifespan and spread of interest in electric semis that can affect the numbers of the two types of semis. According to assumption 3, both diesel and electric semis have a lifespan of around 16 years. Therefore the rate at which semis reach the end of their lifespan was considered to be equal to the rate of production 16 years prior:

$$\delta_D = D(t-16) \tag{15}$$

$$\delta_E = E(t-16) \tag{16}$$

Since the production of electric semis was virtually 0 before 2020, δ_E stays at 0 until 2036.

The final component of our theoretical model is the rate of conversion from a diesel semi driver or owner into an electric semi driver or owner. We found this rate by parameter fitting and determined that the chance of a diesel semi driver or owner converting to using an electric semi when the driver or owner encounters an electric semi driver or owner, σ, is 0.00000002. This value was verified by the reasonable prediction it produced.

3.3.1 Variables

Variable	Definition	Value
Q	Production of Semis	Time Dependent
D	Number of Diesel Semis	Time Dependent
E	Number of Electric Semis	Time Dependent
N	Number of Semis Produced Since 2004 That Are Past Their Lifespan	Time Dependent
α	Rate of Production of Semis	Time Dependent
β_D	Rate of Production of Diesel Semis	N/A
β_E	Rate of Production of Electric Semis	0.00000138095
t	Number of Years Since 2020	Time Dependent
σ	Rate of Conversion From Diesel Semis to Electric Semis	0.00000002
δ_D	Rate of Diesel Semis Reaching End of Lifespan	Time Dependent
δ_E	Rate of Electric Semis Reaching End of Lifespan	Time Dependent

3.3.2 Equations

$$\frac{dQ}{dt} = \alpha - \beta_E QE - \beta_D QD \tag{17}$$

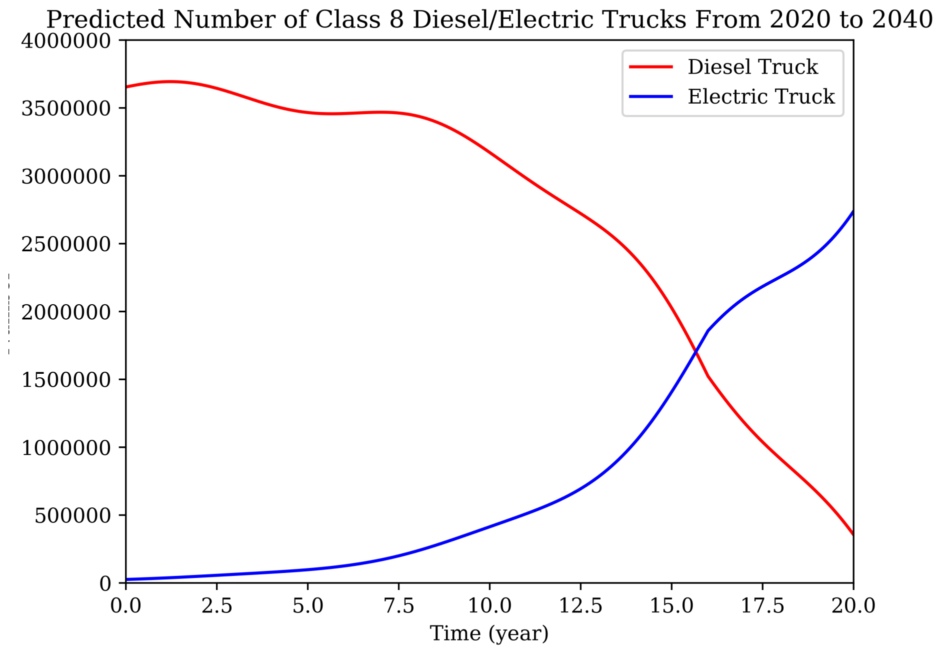
$$\frac{dD}{dt} = \beta_D QD - \sigma DE - \delta_D D \tag{18}$$

$$\frac{dE}{dt} = \beta_E QE + \sigma DE - \delta_E E \tag{19}$$

$$\frac{dN}{dt} = \delta D + \delta E \tag{20}$$

3.4 Results

Our model predicted that 2.74% of semi-trucks will be electric in 5 years, 11.55% in 10 years, and 88.52% in 20 years from 2020. The following figure shows the predicted number of diesel and electric semis from 2020 to 2040:



3.5 Strengths and Weaknesses

The strength of our model lies in the SIR model as its foundation. The SIR model from which our model is derived takes into consideration the spread of the infection of the mind with the benefits of purchasing an electric vehicle.

The weaknesses of our model lie in the alternative routes that we took to implement our theoretical model. Being only an estimate of our theoretical model, our results may have lost accuracy during the process of simplifying the implementation of our theoretical model. Moreover, there are factors which our model could have considered but did not, such as laws and regulations, environmental awareness, and company decisions. However, the parameter fitting for evaluating the rate of conversion from a diesel semi driver or owner to an electric semi driver or owner was able to partially simulate the effect of those factors.

4 Part II

4.1 Problem Definition

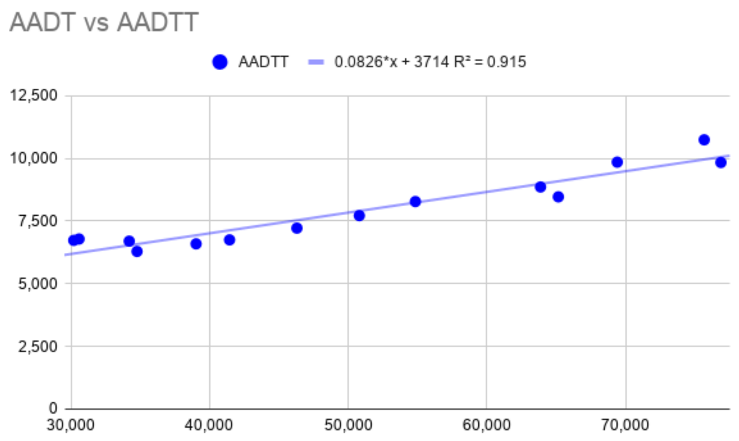
One of the largest issues in transitioning from diesel trucking to electric is the vast amount of infrastructure that must be made. In order to effectively transition, the proper amount of charging stations and chargers are needed to maintain the current efficiency of semis. In this section, we will model the amount of stations and chargers per station that will be necessary for an all electric trucking industry,

4.2 Assumptions and Justifications

1. There is no additional cost in creating a station. The chargers that we are considering will be added to existing infrastructure. Therefore, the initial cost for the chargers will be the only additional cost for maintaining and creating the station.
2. The average maximum range of a electric truck is 300 miles or 480 km.
3. There is a 25% range anxiety for the average driver. Therefore, drivers are only willing to go 75% of their maximum range. From the above assumption, the maximum distance the average driver is willing to is 225 miles or 360 kilometers.
4. There is two-way traffic at each of the stations. For each of the corridors, there will be trucks coming from each direction.
5. The maximum wait time for a given station should be
6. Since all of the routes that are simulated are major trucking routes, all of the chargers used will be Level 3 or DCFC. This is because this maximizes the efficiency of the trucks travelling along the route by minimizing the time that they spend charging their trucks along the route.
7. The corridor between two locations is assumed to be approximately a Euclidean distance and thus straight paths. This is due to the inherent nature of highways and freeways due to the fact that they were constructed in a manner which did not deviate from a straight path due to their purpose for travelling long distances.

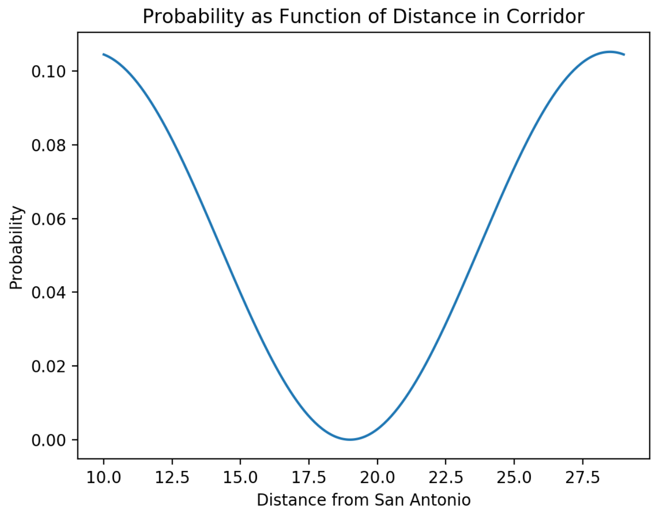
4.3 Model

The corridor dataset provided by the Mathworks Math Modeling Challenge was mainly used for this problem[1]. This dataset provides the incremental distance from one side of the corridor to the other as well as the corresponding Annual Average Daily Truck Traffic(AADTT) values at each of these distances. These AADTT values help describe the amount of traffic present between an interval of the corridor. Some AADTT values were unavailable, but generally, the relationship between AADT and AADTT values presented in the data sets was linear, so we extrapolated the missing AADTT values with linear regression. A separate regression line was made for each geographical area because different geographical areas will inherently have different amounts of AADTT relative to their AADT. For example, areas with more factories will have a higher AADTT rate than other areas. The average R^2 value for the graphs after eliminating outliers was .72. Shown below is an example graph of regression for the Boston to Harrisburg path.



4.3.1 Model Training

To model the distribution of the cars along the corridor at a random time, we had to interpolate the points that were present at each of the given points in the dataset. For example, given an AADTT value at two locations along the corridor, we determined that there should be a probability distribution for the discrete points between the upper and lower bound of two adjacent points. Given the assumption that there is two-way traffic, it is more likely for cars to be concentrated at the points near charging stations. From this assumption, it is also less likely for the cars to be present in the middle. This can be modeled by a modified wavefunction, which plots the probability of being at a certain location at any time. This is seen by this graph:



Formula for the wavefunction:$(\sqrt{\frac{2}{L}}sin(\frac{n\pi x}{L})^2$

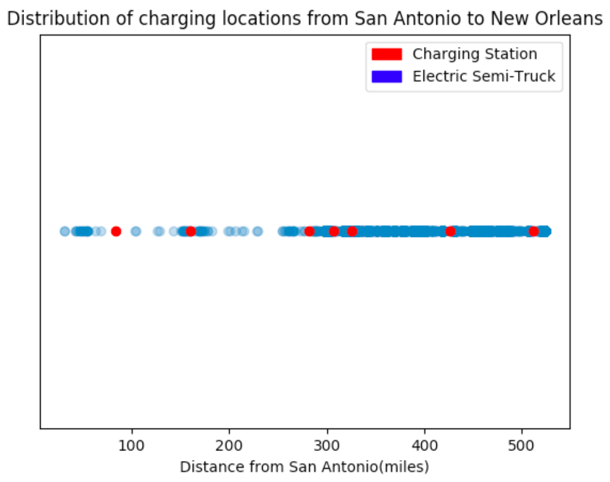
The reasoning behind the usage of this modified wavefunction is to accurately reflect the previous observations that we had about the expected distribution of cars between two points as well as have a probability of a car being present at each discrete point in the interval.

Based on the probabilities generated from the wavefunction, we simulated the state of each corridor at an instantaneous moment. This gave us an accurate representation of the density of truckers along the corridor. As our model was based on covering the maximum amount of truckers in a certain radius, our model closely aligned with the maximum coverage location problem. Specifically, this problem applies to situations wherein there is a condition, in our case truckers, who should be covered by stations, in our case charging stations. In addition, each charging station is deemed to have a radius of coverage or range. In our model, we chose to use a 60 mile radius which is optimized for electric trucking as it satisfies the range anxiety problem and provides ample charging stations for them not to be overcrowded. With these factors, the maximum location coverage problem closely aligns as we face the problem of creating a model for placing charging stations in a way such that the cost is optimized and the truckers are satisfied. The maximum coverage location problem also has an interesting property of not overlapping or double-counting truckers who will be serviced by another station. This allows the algorithm to deduce the optimal amount of stations to have along each corridor along with their location. Due to the benefits and accuracy the maximum location coverage problem would bring to this problem, we decided to use it as the model for this problem. Due to the nature of such a problem being NP-complete, we chose to use a computer program and apply greedy properties in order to generate an approximate yet accurate solution to this problem. Utilizing the probabilities of existence created by the wavefunction simulation, we could apply algorithm to realistic data and generate the points which we deemed should be charging stations.

Although this problem is fundamentally NP-complete, we were able to develop an approximate algorithm to identify where the charging stations should be located. Due to the nature of the problem, the algorithm followed the greedy principle. The algorithm was developed to be as follows. **(Greedily select the subset of usubk which selects the most elements from U in its radius such that element e is not apart of any other subset)**

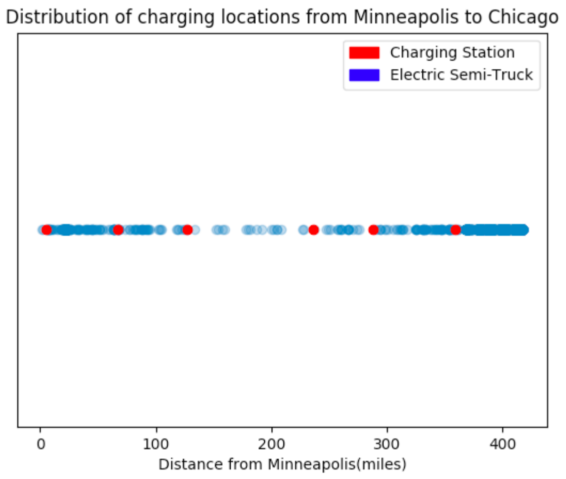
The second part of the problem challenged us to find the number of charging stations at each charging station. In order to calculate this number, we once again referenced the wavefunction simulation in order to approximate the amount of people at the station at an instantaneous moment. Using the same random seed as the simulation from which we deduced the position of charging stations, we calculated the amount of truckers who could possibly arrive at each charging station. This was deduced by taking the distance of each trucker from each charging station and if this distance was less than the radius we selected previously, then that trucker was deemed as a candidate for refueling at that station. This gave us the total possible truckers who could arrive at each station. From the total possible amount of truckers, we decided to account for the worst possible situation and the extreme that all the truckers would arrive at that situation. This would leave the station able to handle maximum capacity. In order to find the number of charging stations needed, we divided the maximum number of truckers at that station by 48. This is due to the fact that there are 48 charging cycles which represents the AADTT and thus gives us the amount of people at the station in an instantaneous moment. This is deduced because there are 30 minute charging increments, meaning 48 charging cycles. From our model, we are thus able to find accurate charging locations which optimize the cost and provide safety so that even in the extreme case every trucker needs to refill, there should be sufficient capacity.

4.4 Results



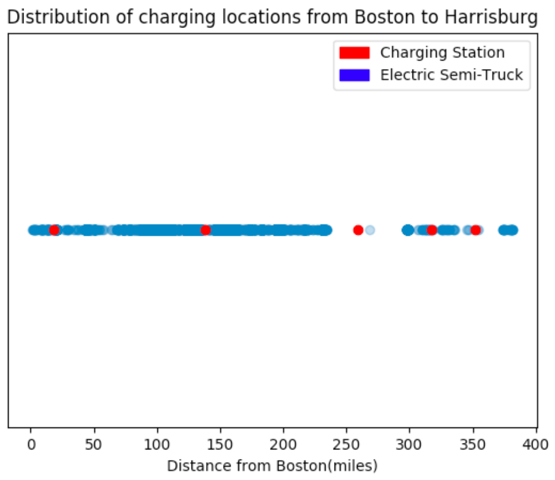
San Antonio to New Orleans

Charging Station Distance from Start (miles)	Number of Possible Trucks Needing Charge	Required Chargers
84	7	1
160	37	1
282	2509	52
307	2782	58
326	3236	67
426	5865	122
512	5056	105



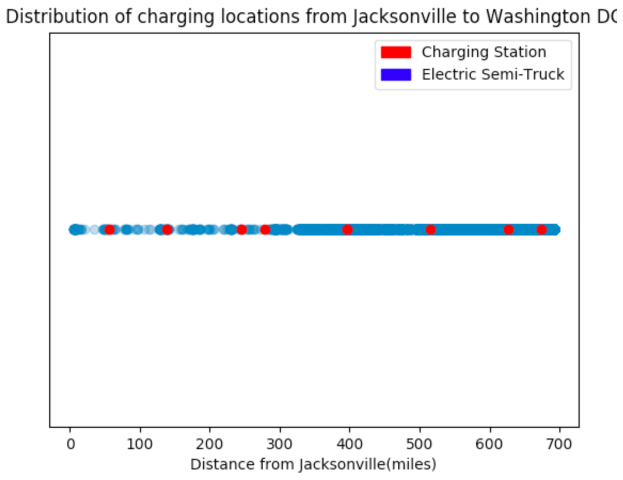
Minneapolis to Chicago

Charging Station Distance from Start (miles)	Number of Possible Trucks Needing Charge	Required Chargers
5	185	4
67	223	5
127	52	1
236	35	1
288	72	2
359	1386	29



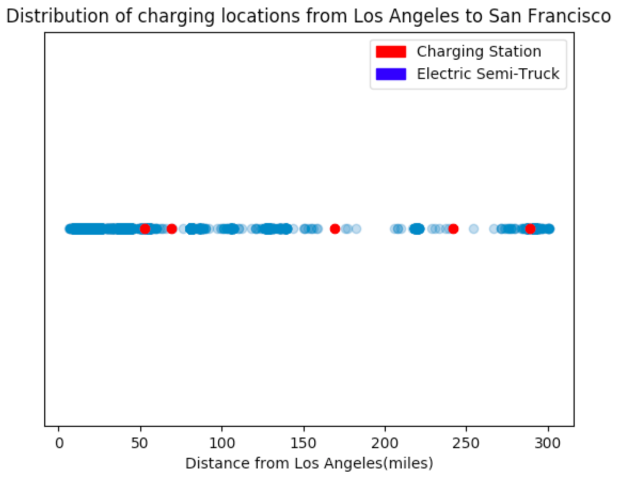
Boston to Harrisburg

Charging Station Distance from Start (miles)	Number of Possible Trucks Needing Charge	Required Chargers
18	443	9
138	3016	63
259	390	8
317	202	4
359	219	5



Jacksonville to Washington DC

Charging Station Distance from Start (miles)	Number of Possible Trucks Needing Charge	Required Chargers
56	87	2
139	97	2
246	127	3
279	246	5
397	2738	57
515	3002	63
626	8161	170
674	4687	98



Los Angeles to San Francisco

Charging Station Distance from Start (miles)	Number of Possible Trucks Needing Charge	Required Chargers
7	2491	52
69	2413	50
156	237	5
259	302	6
280	276	6

4.4.1 Sensitivity Analysis

We conducted sensitivity analysis on our algorithm by changing the radius and confirming that our algorithm was robust. As seen from this graph, the output with the radius increased by 10 is very similar to the radius we used for our results. This proves that our model is robust and is accurate.

4.5 Strengths and Weaknesses

One strength of our model is that the maximum location coverage problem solves not only the location of the charging stations, but also the amount of charging stations. This means that unlike alternative algorithms and models in which these must be deemed separately, our model takes into account inherent characteristics based on the location. This provides a more robust model and should lead to improved results. One weakness of our solution is that the maximum location coverage problem is inherently NP-complete. This meant that for edge cases, particularly areas with low amount of vehicles, our model could perform poorly. However, this can be easily mitigated with human verification.

5 Part III

5.1 Problem Definition

5.2 Assumptions and Justifications

1. We assumed that the average GDP per capita of each of the states traveled in the path was an accurate estimate of the overall economy of the areas because the GDP is the most common estimate of economic state of an area.

2. We assumed that all registered voters have an opinion on environmentalism, which although isn't 100% true, should be a solid estimate. Although not all voters would have an opinion on environmentalist issues, current non-registered voters should make up the difference.

3. We assumed that the corridor data from part 2 is still applicable, and the assumption that the relationship between AADT and AADTT is still linear.

5.3 Model Development

Our model uses four different factors to create our necessity index that determines which of the corridors should be targeted for development first. The four main factors that we use to create this index is the community wealth index, community environmentalist index, community benefit index, and the utilitarian index. We determined a weighted sum that would map to a aggregate index, called the necessity index, which would allow us to determine the corridor which would be most benefited by being targeted with increased development. For each of the parameters, they will be normalized by calculating the mean, standard deviation, and z-score of the 5 values corresponding to each of the different corridors. A root-mean-square of the 4 indices will then be averaged to determine the final necessity index.
The mean is calculated as:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \tag{21}$$

The standard deviation is calculated as:

$$\sigma = \sqrt{\frac{1}{s-1}(\sum_{i=1}^{n} X_i^2 - \frac{1}{n}(\sum_{i=1}^{n} X_i)^2)} \tag{22}$$

The z-score is calculated as:

$$z = \frac{x - \bar{X}}{\sigma} \tag{23}$$

The root mean square is calculated as:

$$\sqrt{\frac{\sum_{i=1}^{n} a_i^2}{n}} \tag{24}$$

Each parameter is described as follows:

1. Community Wealth Index: We utilized the GDP per capita of each region to determine which areas are the most economically sound. This is important because a more economically strong area would be more receptive to the change toward electric trucks. The initial costs of switching towards electric trucks would be an added expense that requires excess wealth. Furthermore, the higher the GDP, the better the overall living conditions generally are in an area which makes it so that the area would be more able to make the transition to electric without anyone in the community suffering.

2. Community Environmentalist Index: The environmentalist index provides an attempt at estimating the environmentalists in the areas affected. Environmentalists would generally be more receptive to the changes and actively support them. In theory, having a higher environmentalist index would mean a lower up front cost because there would be a higher likelihood for donations and support for environmentally friendly actions.

3. Community Benefit Index: A greater amount of trucker traffic would mean a greater benefit for transitioning to electric, and a faster economic benefit. The community benefit index uses the AADT and AADTT values found from part 2 to create an index that seeks to maximize the amount of trucker traffic that is reached. The index itself takes the slope of the linear regression equations found earlier to account for the baseline AADTT and the percentage growth of the AADTT per AADT.

4. Utilitarian Index: The utilitarian index refers to its namesake philosophy, seeking the greatest good for the greatest number. The utilitarian index finds the total number of people who will be affected by the limited amount of emissions and the long term economic benefits of electric vehicles and seeks to maximize that number.

5.4 Results

Index	SA to NOLA	MIN to CHI	BOS to HBURG	JAX to DC	LA to SF
Community Benefit Index	756.1056	128.3416	306.7764	223.9257	311.7852
Community Wealth Index	48856	51687	61408	62977	58619
Community Environmentalist Index	118233332	10150459	20159331	21785976	17641362
Utilitarian Index	31606634	24095317	42875521	53752268	38802500

Root mean squared for Sa to Nola 1.192 Root mean squared for Minn to Chi 1.057 Root mean squared for SA to NOLA 0.579 Root mean squared for Bos to hburg 1.054 Root mean squared for SF to LA 1.375

6 Bibliography

References

[1] Corridor Data. *MathWorks Math Modeling Challenge 2020.* https://m3challenge.siam.org/node/478 —.

[2] Rene Escalante and Marco Odehnal. *A Deterministic Mathematical Model for the Spread of Two Rumors.* Cornell University, 2017.

[3] Forbes. *Charging An Electric Vehicle Is Far Cleaner Than Driving On Gasoline, Everywhere In America.* 2018. URL: `https://www.forbes.com/sites/energyinnovation/2018/03/14/charging-an-electric-vehicle-is-far-cleaner-than-driving-on-gasoline-everywhere-in-america/#463e802471f8`.

[4] Sebastian Funk, Chris Watkins, and Vincent Jansen. *The spread of awareness and its impact on epidemic outbreaks.* 2009.

[5] Raúl Isea and Rafael Mayo-García. *Mathematical analysis of the spreading of a rumor among different subgroups of spreaders.* 2016.

[6] Kristin Jennifer. *The Best Specs & Fuel Mileage for Class 8 Trucks.* 2019. URL: `https://itstillruns.com/specs-mileage-class-8-trucks-7668977.html`.

[7] Fuel Oil News. *Diesel and the Future of Trucking.* 2019. URL: `https://fueloilnews.com/2019/05/22/diesel-and-the-future-of-trucking/`.

[8] Jose Roberto Piqueira. *Rumor Propagation Mode: An Equilibrium Study.* 2010.

[9] trucking.org. *Reports, Trends Statistics.* https://www.trucking.org/News$_and_Information_Reports_Indust$

[10] UPS. *Inside UPS's electric vehicle strategy.* 2018. URL: `https://www.ups.com/us/es/services/knowledge-center/article.page?kid=ac91f520`.

[11] I Wagner. *Heavy-duty electric truck production share in NAFTA 2018-27.* `https://www.statista.com/statistics/814148/nafta-heavy-duty-electric-truck-penetration/`. 2017.

7 Appendix

7.1 Estimating data using Least Square Curve Fitting

```
import xlrd
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

from scipy.optimize import leastsq

df = pd.read_excel('semi_production_and_use.xlsx', sheet_name='production', skip

data = df['Class␣8␣total']

#initial parameters
guess_freq = 1
guess_amplitude = 3*np.std(data)/(2**0.5)
guess_phase = 0
guess_offset = np.mean(data)

#define the sin function and tuple
p0=[guess_freq, guess_amplitude,
            guess_phase, guess_offset]
def my_sin(x, freq, amplitude, phase, offset):
        return np.sin(x * freq + phase) * amplitude + offset



fit = curve_fit(my_sin, df['Year'], data, p0=p0)

#first guess
data_g1 = my_sin(df['Year'], *p0)

#optimize the curve fitting
data_fit = my_sin(df['Year'], *fit[0])

fig, ax = plt.subplots()

#plot all data
ax.plot(data, '.')
ax.plot(data_fit, label='after␣fitting')
ax.plot(data_g1, label='first␣guess')
ax.legend()
```

```
#save the plot to a png
fig.savefig('sine.png', dpi=500)
```

7.2 Utilizing a greedy algorithm to solve the maximum location coverage problem and predict charging station distributions

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from random import seed, uniform, randint
from scipy.spatial import distance_matrix
import matplotlib.patches as mpatches

seed(0)

# MCLP Implementation referenced from Duke University
# Uses integer programming and assumes it is a straight line from location 1 to

def genCandidateSites(minX, maxX, numOfCandidates):
    sites = []

    for i in range(numOfCandidates):
        randX = randint(minX, maxX)
        sites.append([randX, 0])

    return sites

def mclp(points, candidateSites, radius, maxSites):
    distMat = distance_matrix(candidateSites, points)
    returnedSites = []
    stringSites = []

    for i in range(maxSites):
        print(distMat)
        currHighestPoints = -1
        currCandidate = None

        for j in range(len(candidateSites) - 1):
            # For each proposed site, we want to calculate its distance and see
            currSum = 0

            for k in range(len(points) - 1):

                if(distMat[j][k] != -1 and distMat[j][k] <= radius):
                    currSum += 1
            if(currSum > currHighestPoints):
                # Current site becomes highest candidate
                currHighestPoints = currSum
                currCandidate = j

        # After site is selected, set all points in its radius = to -1

        for j in range(len(points) - 1):
            if(distMat[currCandidate][j] <= radius):
                for k in range(len(candidateSites) - 1):
                    distMat[k][j] = -1

        print('Found candidate at: ' + str(currCandidate))
        stringSites.append('Found candidate: ' + str(currCandidate) + ' at: ' +
        returnedSites.append(currCandidate)
        if(currCandidate == 0 and distMat[0][0] == -1):
            print(stringSites)
            return returnedSites

    return returnedSites

df = pd.read_excel('fixed_corridor_data.xlsx', sheet_name='LA_SF', skiprows=[0,1

startLoc = 'Los Angeles'
endLoc = 'San Francisco'

distances = df[df.columns[0]]
aadtt = df[df.columns[3]]

points = []

# Use of wave function to determine probability of existing at certain given AAD
dataDict = dict(zip(distances, aadtt))
dictKeys = list(dataDict.keys())

L = 20
n = 1
def psi_squared(x,L,n):
    return np.square(np.sqrt(2/L)*np.sin(n*np.pi*x/L))

psi_val_dict = {}
total_psi_val = []
x_list_dict = {}
total_x_list = []
for i in range(len(dataDict)-1):
    x_list_dict[i] = np.linspace(dictKeys[i],dictKeys[i+1],int(dataDict[dictKeys
    total_x_list.extend(x_list_dict[i])
    psi_val = []
    for x in x_list_dict[i]:
        psi_val.append(psi_squared(x,dictKeys[i+1]-dictKeys[i],n))
        psi_val_dict[i] = psi_val
    total_psi_val.extend(psi_val)

print(len(total_x_list))
print(len(total_psi_val))
print(type(total_x_list))
print(type(total_x_list))

probability_of_existance = dict(zip(total_x_list, total_psi_val))

for item in probability_of_existance.items():
    position, probability = item
    random_num = uniform(0, 1)
    if(random_num < probability):
        points.append([int(position), 0])

low_x = int(points[0][0])
high_x = int(points[len(points) - 1][0])
print(str(low_x) + ' ' + str(high_x))
candidateSites = genCandidateSites(low_x, high_x, 400)

chosen_stations =  mclp(points, candidateSites, 70, 30)

plt.title('Distribution of charging locations from ' + startLoc + ' to ' + endLo

point_x = []
point_y = []
for point in points:
    point_x.append(point[0])
    point_y.append(point[1])

station_locations_x = []
station_locations_y = []
for station in chosen_stations:
    station_locations_x.append(candidateSites[station][0])
    station_locations_y.append(candidateSites[station][1])

plt.scatter(point_x, point_y, alpha=0.3)
plt.scatter(station_locations_x, station_locations_y, color='r')

plt.xlabel('Distance from ' + startLoc + '(miles)')
plt.yticks([])

red_patch = mpatches.Patch(color='red', label='Charging_Station')
blue_patch = mpatches.Patch(color='blue', label='Electric_Semi-Truck')

plt.legend(handles=[red_patch, blue_patch])

plt.show()
```

7.3 Calculating the amount of people charging at each station

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from random import seed, uniform, randint
from scipy.spatial import distance_matrix
import matplotlib.patches as mpatches

seed(0)

# MCLP Implementation referenced from Duke University
# Uses integer programming and assumes it is a straight line from location 1 to

def genCandidateSites(minX, maxX, numOfCandidates):
    sites = []

    for i in range(numOfCandidates):
        randX = randint(minX, maxX)
        sites.append([randX, 0])

    return sites

def mclp(points, candidateSites, radius, maxSites):
    distMat = distance_matrix(candidateSites, points)
    returnedSites = []
    stringSites = []

    for i in range(maxSites):
        print(distMat)
        currHighestPoints = -1
        currCandidate = None

        for j in range(len(candidateSites) - 1):
            # For each proposed site, we want to calculate its distance and see
            currSum = 0
            for k in range(len(points) - 1):

                if(distMat[j][k] != -1 and distMat[j][k] <= radius):
                    currSum += 1

            if(currSum > currHighestPoints):
                # Current site becomes highest candidate
                currHighestPoints = currSum
                currCandidate = j

        # After site is selected, set all points in its radius = to -1

        for j in range(len(points) - 1):
            if(distMat[currCandidate][j] <= radius):
                for k in range(len(candidateSites) - 1):
                    distMat[k][j] = -1

        print('Found candidate at: ' + str(currCandidate))
        stringSites.append('Found candidate: ' + str(currCandidate) + ' at: ' +
        returnedSites.append(currCandidate)
        if(currCandidate == 0 and distMat[0][0] == -1):
            print(stringSites)
            return returnedSites

    return returnedSites

df = pd.read_excel('fixed_corridor_data.xlsx', sheet_name='LA_SF', skiprows=[0,1

startLoc = 'Los Angeles'
endLoc = 'San Francisco'

distances = df[df.columns[0]]
aadtt = df[df.columns[3]]

points = []

# Use of wave function to determine probability of existing at certain given AAD
dataDict = dict(zip(distances, aadtt))
dictKeys = list(dataDict.keys())

L = 20
n = 1
def psi_squared(x,L,n):
    return np.square(np.sqrt(2/L)*np.sin(n*np.pi*x/L))
```

```
psi_val_dict = {}
total_psi_val = []
x_list_dict = {}
total_x_list = []
for i in range(len(dataDict)-1):
    x_list_dict[i] = np.linspace(dictKeys[i],dictKeys[i+1],int(dataDict[dictKeys
    total_x_list.extend(x_list_dict[i])
    psi_val = []
    for x in x_list_dict[i]:
        psi_val.append(psi_squared(x,dictKeys[i+1]-dictKeys[i],n))
        psi_val_dict[i] = psi_val
    total_psi_val.extend(psi_val)

print(len(total_x_list))
print(len(total_psi_val))
print(type(total_x_list))
print(type(total_x_list))

probability_of_existance = dict(zip(total_x_list, total_psi_val))

for item in probability_of_existance.items():
    position, probability = item
    random_num = uniform(0, 1)
    if(random_num < probability):
        points.append([int(position), 0])

low_x = int(points[0][0])
high_x = int(points[len(points) - 1][0])
print(str(low_x) + '␣' + str(high_x))
candidateSites = genCandidateSites(low_x, high_x, 400)

chosen_stations =  mclp(points, candidateSites, 70, 30)

plt.title('Distribution␣of␣charging␣locations␣from␣' + startLoc + '␣to␣' + endLo

point_x = []
point_y = []
for point in points:
    point_x.append(point[0])
    point_y.append(point[1])

station_locations_x = []
station_locations_y = []
for station in chosen_stations:
    station_locations_x.append(candidateSites[station][0])
    station_locations_y.append(candidateSites[station][1])
```

```
plt.scatter(point_x, point_y, alpha=0.3)
plt.scatter(station_locations_x, station_locations_y, color='r')

plt.xlabel('Distance from ' + startLoc + '(miles)')
plt.yticks([])

red_patch = mpatches.Patch(color='red', label='Charging Station')
blue_patch = mpatches.Patch(color='blue', label='Electric Semi-Truck')

plt.legend(handles=[red_patch, blue_patch])

plt.show()
```

7.4 Prediction of number of class 8 diesel and electric trucks from 2020 to 2040

```
from scipy.integrate import odeint
import matplotlib.pyplot as plt
import numpy as np
plt.rcParams['font.family'] = 'serif'
plt.rcParams['font.serif'] = ['Times New Roman'] + plt.rcParams['font.serif']

# number of diesel and electric truck at the beginning of 2020
y0 = 3654240, 25760
xMin, xMax = 0, 20
yMin, yMax = 0, 4000000
resolution = 20
t = np.linspace(xMin, xMax, (xMax - xMin) * resolution + 1)

# rate of class 8 electric truck production
betaE = 0.25 * (1 - 760012 / 3680000) / 143645.85273649602
sigma = 0.01 * 0.01 * 0.01 * 0.02

# returns production of class 8 trucks in year (2004 + t)
def production(t):
  return np.sin(t * 0.8909006172269278 + 219.32023815530073) * 45609.07035271879

# a naive derivative to be integrated as a predictor
def predictorDeriv(E, t):
  return betaE * production(t + 16) * E

# deltaE[t] = predicted production of class 8 electric trucks in year (2004 + t
deltaE = odeint(predictorDeriv, 25760, t).T[0]

# ODE system
```

```
def deriv(y, t):
  D, E = y
  dEdt = betaE * production(t + 16) * E - (betaE * production(t) * deltaE[min(in
  dDdt = production(t + 16) - dEdt - production(t) - sigma * D * E
  return dDdt, dEdt

D, E = odeint(deriv, y0, t).T

fig, ax = plt.subplots()
ax.axis([xMin, xMax, yMin, yMax])
d, = ax.plot(t, D, 'r', label = 'Diesel Truck')
e, = ax.plot(t, E, 'b', label = 'Electric Truck')
ax.grid()
ax.set(xlabel = 'Time (year)', ylabel = 'Number', title='Predicted Number of Cla
plt.legend(loc = 'upper right', handles = [d, e])
ax.grid()
fig.savefig('graph.png', dpi = 500)

# print proportion of electric trucks in all class 8 trucks 5 years, 10 years, a
print(E[5 * 20] / (E[5 * 20] + D[5 * 20]))
print(E[10 * 20] / (E[10 * 20] + D[10 * 20]))
print(E[20 * 20] / (E[20 * 20] + D[20 * 20]))
```

7.5 Binary search curve fitting for logistic curve

```
from math import exp

data = [0.018, 0.035, 0.045, 0.06, 0.07, 0.08, 0.09, 0.10]

# logistic growth function
def function(lambda_, t):
  return data[0] / (data[0] + (1 - data[0]) * exp(-lambda_ * t))

# loss function
def square_diff(lambda_):
  sum = 0
  for t in range(len(data)):
    sum = sum + (function(lambda_, t) - data[t]) ** 2
  return sum

min_a, min_b, min_loss = 0, 0, 1e10

# binary search for lambda
l = 0
r = 1
while r - l > 0.000001:
  m = (l + r) / 2
  if square_diff(l) < square_diff(r):
      r = m
  else:
      l = m

# print results
print(l)
```