IOE 574 Report

A study on Bus Transportation Systems

Team Members: Anirudh Chatty, Krishna Rao, Harsh Hegde, Shuangwei Yu

TABLE OF CONTENTS

Executive Summary

In a study conducted and published by the United States Environmental Protection Agency (EPA), It was shown that Global emissions due to burning of fossil fuels have increased almost exponentially since 1900. From 1970, there has been more than a 90% increase in emissions and greenhouse gases, with the emissions from burning fossil fuels and fossil fuel based combustion contributing to about 78% of the total greenhouse gas emissions[1].

A natural step for city planners to combat this issue is to explore ways to reduce the dependency on transportation such as personal cars and other motorized vehicles. One way that this has been implemented is with the introduction of bike lanes to push the public into using bikes for intra-city travels. While this may be effective in specific cities and during certain times of the year, more needs to be done in order to allow people to move from one place to another. For example, during winter months in Michigan, where people may not want to cycle as much, may prefer to take the bus to get to their destination. This provides the city planners an opportunity to reduce the emission of greenhouse gases by switching the bus system to either a hybrid or an all-electric fleet. A report published by The National Resources Defence Council (NRDC) in 2015 has shown that using either hybrid or electric systems for transportation systems has the potential to dramatically reduce the lifetime emissions of greenhouse gases by as much as 77%, which is equivalent to reducing emissions by 1700 million metric tons relative to 2015 levels.

The aim of this project is to simulate the University bus (or M-Bus) system for the city of Ann Arbor using 4 different types of buses - conventional buses (liquid fuel based), all electric buses, and hybrid buses. We aim to aid the city planner in making a decision on which type of bus would be the most feasible for the given situation. In order to make this decision, we compare several metrics associated with the operational feasibility of different alternatives through sensitivity analysis on performance based measures, number of buses in the fleet and the level of fuel to be maintained at the start of each day. The simulation model proposed in the paper consists of two parts - a recharging/refilling station (which is modeled in the form a route) and the actual transportation model itself. The transportation network is in the form of a network flow and utilizes a start and end node which represent a bus depot from which the buses are assigned to different routes. The routes are cyclic which makes the re-deployment of a bus to another route possible.

We hope that the simulation model will give us (and the decision makers) insights about the level of complexity involved in the decision making process as well as a starting point as to what to look into, and consider, in the event that the planners are looking to switch to a hybrid or even a fully electric system.

1. Introduction

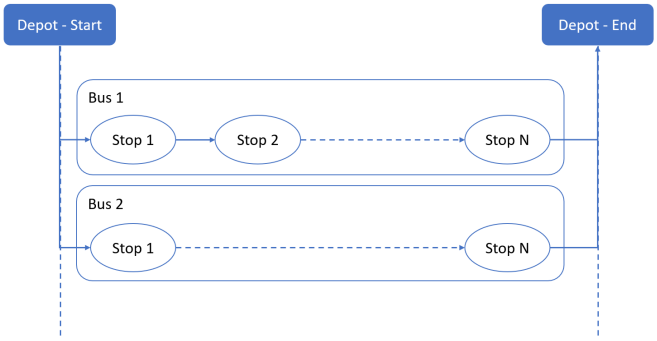


Figure 1: Schematic representation of the routes in the proposed system

As a part of smart urban construction, the first ever hybrid M-bus was introduced to Ann Arbor in 2012, as part of the school's commitment to sustainability. Allowing for better fuel mileage and lower emissions, it is predicted that in the future, the service of bus transportation will be supported fully by electric vehicles.

In this paper we aim to simulate the transportation network of Blue bus routes based on time varying demands throughout the day. The model will provide us insights on bus planning strategy, help us understand the challenges associated with switching to hybrid vehicles compared to conventional vehicles in a high capacity system, and help us understand associated costs for running the system.

With a pre-designated number of buses that are available to be deployed, we view the problem from a system's perspective and investigate the sensitivity of different arrangements of buses. Here the objective is actually threefold where we're trying to get insight on financial, environmental and time aspects of different alternatives which are major factors that decision makers focus on. With the data generated from simulating both the transportation and refueling systems, we estimate the total cost of operation of all bus types in a system and thus give financial advice on possible bus systems transitioning.

The rest of the paper is organized as follows. The next section introduces the relevant background and literature review. In Section 4, we formally describe the problem and how we approach it through simulation modelling. Computational results are reported and analyzed in Section 5 and Section 6; summary and concluding remarks are given in Section 7.

2. Background

3.1 Literature Review

Emission mitigation is one of the more important topics in today's times. The transportation sector contributes immensely to the burning of fossil fuels with several nations on their way to significantly reduce this consumption with the introduction of electric vehicles. Commercial fleets such as public buses are seen as a starting point for this transformation. However, the reduced operational performance of electric vehicles is still seen as a barrier in this transition. Our aim is to present a model for the comparison of public bus fleets using different fuel sources in order to assist city planners and management in making such decisions.

In the study by Nyman, J. et al. 2017 [14], data on route specifications, timetables, and other local factors are used to examine e-buses and charging systems from a total cost viewpoint. The authors have also created a user-friendly tool that allows users to examine and quantify trade-offs between EV battery capacity, charging infrastructure costs, and car fleet running expenses. However, all routes are simulated as straight lines with no variances or time delays taken into account which makes the model not as suitable for real time analysis.

Pelletier et al. [2] provides a comprehensive assessment focusing on electric cars. The methodologies mentioned differ in terms of the vehicle types studied, homogeneous or heterogeneous electric car fleets, with and without conventional combustion engine vehicles, and the mechanism for dealing with charging events. Goeke et al. [16] and Lebeau et al.[17] conducted considerable research on the routing of mixed fleets of conventional and electric cars. They stressed the importance of taking into account vehicle types' distinctive energy consumptions, particularly for cars of varied weights, which drove the energy consumption simulation in this study. Van Duin et al. [18] did not consider battery charging, however, Gonçalves et al. [19] defined charging as being possible at any customer's location.

When conducting bus fleet simulations, a city bus transport optimization study, by Tiechert et al. [20], focuses on an artificial urban bus driving cycle, ignoring influential elements such as city traffic congestion, frequent stops, and so on. According to the authors Xylia et al.[21], cutting the cost of gasoline for electric buses can offset the high investment costs associated with charging infrastructure while also resulting in considerable reductions in polluting emissions. To that end, extensive techno-economic evaluations comparing the Total Cost of Ownership (TCO) of traditional and electric fleets should be carried out.

De Filippo et al. [10] simulated the battery electric bus system over six transit corridors serving Ohio State University main campus. They’ve pointed out in their result that the frequency of service might be harmed if they substitute the current fleet with fully-electrified vehicles. However, they’ve also mentioned that additional infrastructure, like charging stations or high-performance batteries, could be utilized to maintain or even improve the performance of

the system, which might add significant financial burden on the implementation of Battery Electric Buses (BEB).

Perrotta et al. [12] investigated the relation between route characteristics and energy consumption. By simulating on three major bus routes in the city of Oporto, they have argued that curvy routes and short travel distance between stops are the most energy demanding. Kontou and Miles [11] have also analyzed the operational measures of a battery electric bus project at Milton Keynes, they conclude that the actual performance of the buses and charging system is reasonably comparable to and consistent with the theoretical one, considering the energy consumption and the system's efficiency.

Rogge et al. [13] generated a full transit network simulation with fast opportunity-based charging and briefly discussed the impacts on the power grid. However, unlike our simulation model, they have charging stations distributed across the entire transit network. Also, their electric buses have been characterized into three main types: flash, overnight and opportunity with each type representing a distinct profile as it relates to operational feasibility and grid impact. The authors argue that the operation of different BEBs will vary significantly, as will the associated impacts on the distribution power grid.

The existing literature is frequently focused on a specific aspect of electric bus design or operation. The scope of the used models and solution techniques is limited, for example, in terms of considering heterogeneous fleets, cost optimization, and infrastructure needs. Because charging infrastructure is primarily viewed as an input to the problem, charger refilling is not taken into account.

The current study fills this void by operational perspective and the convergence of delays to understand associated costs while performing the comparison of types of bus fleets on the basis of delays, total costs of operations and the stops, traffic and delays. To the best of the authors' knowledge, there have not been studies dealing with simulations for different bus fleets based on real-life driving cycles and the related sensitivity analyses have not been considered in literature this far. Hence, we have created a unique simulation for the comparison of university bus fleets that takes into account time varying demands and real routes all while also incorporating recharging and refueling cycles at designated filling stations. Our model provides decision makers with a comprehensive analysis of all aspects of a bus system in a given geographical area.

3. Model

As mentioned in the introduction, the team plans to assist the decision makers in comparing and choosing a particular type of system with the help of a simulation model. The model proposed in the paper has a fixed simulation time of 12 hours for each replication. This is to ensure that the team is able to capture a wide variety of demands and scenarios as it would capture a typical cycle of demands seen in college towns, such as Ann Arbor. The team also performed 30 replications to emulate a typical month. This would also inform the team on the number of additional replications that may be needed for convergence of the model. The data that is used as an input to the model is as described in the 'Data' section below, and contains the travel times between stops for 4 different routes. For the purpose of the simulation, all of the routes chosen start and end at the Central Campus Transit Center (CCTC) in Ann Arbor, which is considered to be the 'bus depot'.

Keeping in mind that the main task of any bus route is to ensure that its customers reach on time to their respective destinations, the focus of the model is placed on understanding and tracking the bus delays experienced in the system, and recording the extra replications needed to run the model in order to ensure that there is a convergence of the delay value, with a half-width of 1.5 minutes. The half-width was chosen based on what the team had experienced when using the current bus system of Ann Arbor. We also track several other metrics, focusing on both bus performance metrics as well as cost metrics, such as operational and fuel costs, that would help city planners and other stakeholders better understand how they could use the simulation system to make better decisions. The team has also conducted a sensitivity analysis that displays the performance of the system under a variety of scenarios, mainly by varying 4 primary parameters - (1) the performance of the bus itself, which is the consumption of fuel when the bus is running and when it is stationary, (2) the number of buses in the entire system, and (3) the mean level of fuel that is present in each of the buses at the start of the day.

3.1. Data

We have acquired real time data from 4 bus routes on the University of Michigan Blue Bus system. The data was collected by physically taking 5 samples on each route, data available on Google Maps and gathered through General Transit Feed Specification from the University of Michigan Logistics, Transportation and Parking. The routes are as follows:

1. Bursley Baits
2. Northwood
3. Diag to Diag Express
4. Oxford Shuttle

We calculated the interarrival times and the standard deviations for the interarrival times for each stop on the route. The data is attached in the appendix. Since the team had only collected 5 data points for each stop in each route, there may have been some bias introduced in the calculations of the mean and standard variations for the inter-arrival/travel times between

the stops. For future methods and simulations, it would be helpful to take more readings to ensure that the mean and variance are representative of the routes across all conditions.

3.2. Assumptions

For the purpose of simulating the model we have taken the following assumptions into consideration and incorporated them in the model:

1. Passenger demand is translated as an integer demand related to the number of buses required every hour for a specified route, which takes care of the total route demand.
2. All buses start and end at the same point, that is the routes are cyclic. For that purpose we have taken CCTC as the Start depot and buses return back to the same depot at the end of the route.
3. The maximum number of buses that can be deployed across all bus routes is 5. This is based on the assumption that in real life, buses run every 12-15 minutes on a particular route. Hence, we take the upper limit on the number of buses deployed on a route.
4. The fueling/recharging of the buses can only be done at the start and stop nodes and not between the transit of a route. A bus cannot break the flow of a route in the sense that it cannot go for a refuel or recharge in the middle of a route.
5. The buses cannot be deployed onto a route if the fuel level falls below a predetermined value to ensure that the bus is capable of completing a route and is also a safety measure that has been incorporated.
6. The model prioritizes the refueling or recharging of the buses over satisfying the demand. We have also assumed a total of two refueling/recharging stations, with the vehicle needing to travel to the stations in order to fill the tank/charge the battery.
7. The refueling/charging process for the different buses is also treated as a route, where the route has a service time that would serve as the refueling or recharging time. The route would also have a travel time, which, along with the service time, will give the total amount of time taken for a bus to re-fuel or recharge.
8. We have assumed the hybrid vehicle to be fuel based, with the battery only assisting the performance of the bus by reducing its fuel consumption.
9. There is no unmet demand, all demands from the routes have to be served. Since the public transportation system is an essential part of Ann Arbor, we have assumed that no demand goes un-met even if the demand is delayed.
10. There is no upper limit to the delay of meeting a demand, and demands are served based on precedence of demand time. In continuation with the aforementioned point, we have assumed that there should be no upper limit to the delay of an unmet demand and any available bus will be deployed to satisfy the demands based on the precedence of demands. That is, we follow the FIFO (First In First Out) rule to serve route demands.
11. Routes have been given precedence in bus deployment in ascending order. That is, lower index routes eg. ‘route_1’ has higher precedence than ‘route_2’ for the same time of demand, and so on.
12. Demands for the different routes are deterministic and follow a specific sequence. The demand for the hour is generated through the function specified below, and the

individual demand for the hour is equally divided through the hour starting at t. For example, if we have a demand of 2 for 1 hour at t = 0, this would imply that there is a demand at t = 0 and the next demand would be at t = 30. The demand function used for the model is as follows -

$$\text{Demand} = int(a * sin(\frac{t+c}{d}) + b) + 1)$$

where, a, b, c and d are constants, and t is the time of demand

13. Though the model is capable of taking in a distribution for the mean level of fuel, we have taken the fuel level at the start of the day to be a constant, that is fixed the standard deviation to be zero. A benefit of this is that it helps in the variance reduction aspect of the model as well.
14. The performance of the vehicle is directly impacted by the fuel consumption of the bus while running or at standstill. We assume that the running and idle/service consumption goes higher for a lower performing bus, and are les for a better performing bus.

3.3. Model Structure

As discussed in the Model introduction, we have chosen 4 routes for our simulation and we assume that the refueling/ charging cycle of a bus is also a route. The buses are individual entities which can be deployed to any of the routes based on a certain criteria. Buses can only be deployed to a route if they satisfy the minimum demand level for a route, and they are sent to refuel if they get below the minimum limit for deployment. The routes have definite demands throughout the day which are generated and kept in a list with their associated route. A route demand is only satisfied if a bus is available for deployment to that route and will stay until a bus is deployed. Buses can have 3 states: deployed, refueling and standstill - and these are updated based on deployment, refueling or return from a route.

The first priority of the program is to update demand and send low-fuel buses to refuel. Once these two things are checked the main switch cases take over and bus deployment to routes takes precedence over the next bus event, which can be an arrival to a stop or the service of a stop. The model runs till all the demands are met and all the buses have returned to the depot. The flow of the simulation is depicted in the section ‘Flow Diagram’ which follows.

3.3.1. Variables

The variables used in the model are as follows:

1. System State (**SS**): **(F_N, D_T, B_{ND}, B_{NR}, B_{NS})**
    a. Fleet size - **F_N**
    b. Total current demand from routes - **D_T**
    c. Number of Deployed buses - **B_{ND}**
    d. Number of Refuelling buses - **B_{NR}**
    e. Number of Standstill buses - **B_{NS}**
2. Time variables:
    a. **T** - Simulation end time
    b. **t** - Current clock time

3. Route Variables:
    a. Routes Table - **RT**
    b. Route index - $\mathbf{R_I}$ {route where deploying}
    c. Demands Time - $\mathbf{R_T}$
    d. Minimum fuel required for a route - $\mathbf{R_F}$
4. Bus Variables:
    a. Bus index - $\mathbf{B_I}$ {to reference bus in fleet}
    b. Bus state - $\mathbf{B_S}$ {Deployed:1, Refuel:0, Standstill:-1}
    c. Next Event Time - $\mathbf{B_T}$
    d. Next Event Type - $\mathbf{B_E}$ {1: Arrival, 0: Service}
    e. Fuel level - $\mathbf{B_F}$
    f. Refuel characteristics - ‘recharge’ or ‘refill’
    g. Fuel Consumption rates:
        i. Travel consumption - $\mathbf{FC_T}$
        ii. Service Consumption - $\mathbf{FC_S}$
        iii. Refuel rate - $\mathbf{FC_R}$

We have performed sensitivity analysis by varying certain parameters in order to check the effects on the delay times and costs associated with the fleets. The parameters we have varied include

Table 1: Description of variable parameters for simulation

Travel consumption	running_consumption
Service Consumption	service_consumption
Number of Buses Deployed	n_buses
Mean Fuel Level	level_mean

There are fixed parameters which we have defined for analysing the associated costs of running and maintaining the bus infrastructure. The following table provides insights on the definition of these parameters and the variable used to define them in the simulation.

Table 2: Description of fixed parameters for simulation

Parameter	Variable Name	Gasoline	Electric
Standard Deviation of initial tank level	level_std	0	
Refuel flag	refuel	refill	recharge
Average Bus Speed (miles/hr)	average_bus_speed	30	
Number of refuelling stations	refuel_stations	2	
Conversion factor for tank capacity	conversion_factor	100/tank_size	
Hourly rate of bus driver ($)	emp_rate	15	
Fuel Rate ($/lt. or $/kWh)	fuel_rate	0.882	0.1275
Delay rate (losses suffered per min)	delay_rate	2.4	
Maintenance Rate (dollar per min)	maintain_rate	1.67	2.25
Number of routes	n_routes	4	

Simulation Time (T)	SimTime	720

3.3.2. Flow Diagram

Following is a detailed flow diagram of the model which depicts how an event is updated in the model and how decisions are made based on precedence/priority of a decision.

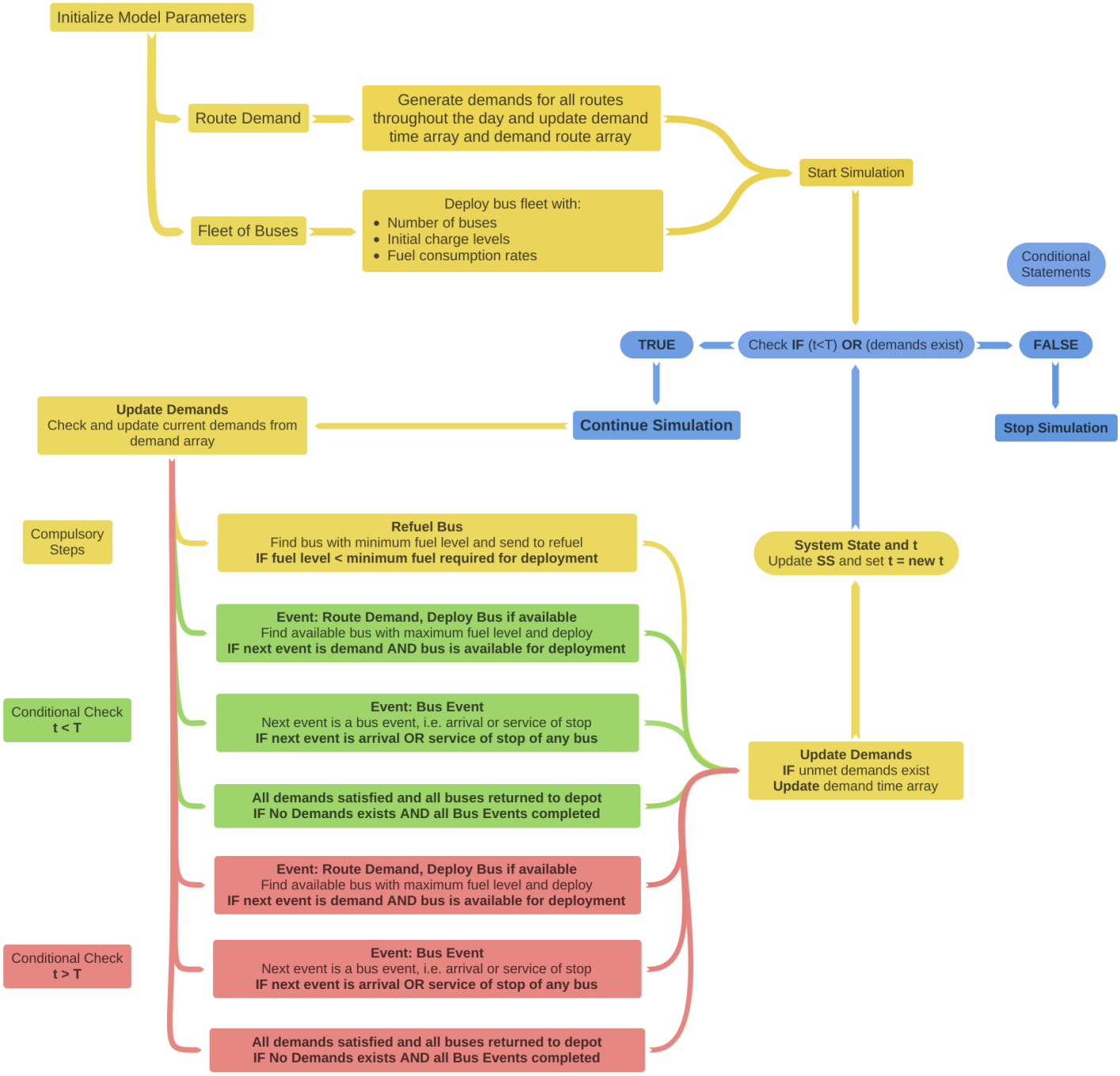


Figure 2: Flow diagram of simulation

3.3.3. Simulation Pseudo-Code

1. Initialize Model Parameters
    a. Set time variables
        i. **t** = 0
        ii. **T** = 720
    b. Route characteristics
        i. Table (**RT**) containing data for generating route times
        ii. $\mathbf{D_T}$ = 0
    c. Bus characteristics
        i. Set fleet size - $\mathbf{F_N}$
        ii. Charge/Fuel levels - $\mathbf{B_F}$
        iii. Refuel characteristic - ‘recharge’ or ‘refill’

2. Generate demands for each route throughout the day
    a. Route demands generated from predetermined distributions and stored as demand time array $\mathbf{R_T}$, and demanding route index array $\mathbf{R_I}$

3. Start simulation with **t** = 0,
   WHILE (**t** < **T**) **OR** ($\mathbf{D_T}$ > 0):

    <u>Compulsory Step</u>:
    a. Check if any demand exists, update $\mathbf{D_T}$ if there is demand, else set to 0.

    <u>Conditional Steps</u>:
    b. Find the bus $\mathbf{B_I}$ with lowest fuel level and check,
    IF $\mathbf{B_F}$ < $\mathbf{R_F}$, fuel level is less than minimum required for deployment
        i. Send bus $\mathbf{B_I}$ to refuel, generate refuel route time from **RT**
        ii. Update $\mathbf{B_T}$ and $\mathbf{B_E}$ based on generated refuel route
        iii. $\mathbf{B_S}$ = 0
        iv. $\mathbf{B_{NR}}$ = $\mathbf{B_{NR}}$ + 1
        v. $\mathbf{B_{NS}}$ = $\mathbf{B_{NS}}$ - 1
        vi. Update **SS**

    c. **ELIF** (**t** < **T**) **AND** ($\mathbf{R_T}$ < $\mathbf{B_T}$) **AND** ($\mathbf{D_T}$ > 0), upcoming event is route deployment
        i. Bus index = $\mathbf{B_I}$
        ii. $\mathbf{B_S}$ = 1
        iii. $\mathbf{B_{ND}}$ = $\mathbf{B_{ND}}$ + 1
        iv. $\mathbf{B_{NS}}$ = $\mathbf{B_{NS}}$ - 1
        v. **t** = $\mathbf{R_T}$
        vi. Update **SS**
        vii. $\mathbf{D_T}$ = $\mathbf{D_T}$ - 1

    d. **ELIF** ((**t** < **T**) **AND** ($\mathbf{B_T}$ < $\mathbf{R_T}$)) **OR**
        ((**t** < **T**) **AND** ($\mathbf{R_T}$ < $\mathbf{B_T}$) **AND** ($\mathbf{D_T}$ > 0) **AND** (No buses to deploy)),
    upcoming event is a bus event, arrival or service
        i. **t** = $\mathbf{B_T}$
        ii. Updating fuel level:
            - IF $\mathbf{B_E}$ = 1, **FC** = $\mathbf{FC_T}$,

ELIF $\mathbf{B_E}$ = 1, $\mathbf{FC} = \mathbf{FC_S}$,
ELIF $\mathbf{B_E}$ = 0 AND Route = 'refuel', $\mathbf{FC} = \mathbf{FC_R}$,
- $\mathbf{B_F} = \mathbf{B_F} - \mathbf{FC} * (\mathbf{t_{new}} - \mathbf{t_{old}})$

iii. Update **SS**
iv. **IF** $\mathbf{B_E}$ is the last arrival event (return to depot):
- Set, $\mathbf{B_T}$ = infinity & $\mathbf{B_E}$ = None
- $\mathbf{B_{ND}} = \mathbf{B_{ND}} - 1$
- $\mathbf{B_{NS}} = \mathbf{B_{NS}} + 1$

e. **ELIF** ($\mathbf{t} < \mathbf{T}$) and ($\mathbf{D_T} = 0$), all demands have been met within time-frame
i. Set $\mathbf{t} = \mathbf{T}$
ii. Update **SS**

f. **ELIF** ($\mathbf{t} > \mathbf{T}$) **AND** ($\mathbf{R_T} < \mathbf{B_T}$) **AND** ($\mathbf{D_T} > 0$), upcoming event is route deployment
i. Bus index = $\mathbf{B_I}$
ii. $\mathbf{B_S} = 1$
iii. $\mathbf{B_{ND}} = \mathbf{B_{ND}} + 1$
iv. $\mathbf{B_{NS}} = \mathbf{B_{NS}} - 1$
v. $\mathbf{t} = \mathbf{R_T}$
vi. Update **SS**
vii. $\mathbf{D_T} = \mathbf{D_T} - 1$

g. **ELIF** (($\mathbf{t} > \mathbf{T}$) **AND** ($\mathbf{B_T} < \mathbf{R_T}$)) **OR**
(($\mathbf{t} > \mathbf{T}$) **AND** ($\mathbf{R_T} < \mathbf{B_T}$) **AND** ($\mathbf{D_T} > 0$) **AND** (No buses to deploy)),
upcoming event is a bus event, arrival or service
i. $\mathbf{t} = \mathbf{B_T}$
ii. Updating fuel level:
- IF $\mathbf{B_E}$ = 1, $\mathbf{FC} = \mathbf{FC_T}$,
ELIF $\mathbf{B_E}$ = 1, $\mathbf{FC} = \mathbf{FC_S}$,
ELIF $\mathbf{B_E}$ = 0 AND Route = 'refuel', $\mathbf{FC} = \mathbf{FC_R}$,
- $\mathbf{B_F} = \mathbf{B_F} - \mathbf{FC} * (\mathbf{t_{new}} - \mathbf{t_{old}})$

iii. Update **SS**
iv. **IF** $\mathbf{B_E}$ is the last arrival event (return to depot):
- Set, $\mathbf{B_T}$ = infinity & $\mathbf{B_E}$ = None
- $\mathbf{B_{ND}} = \mathbf{B_{ND}} - 1$
- $\mathbf{B_{NS}} = \mathbf{B_{NS}} + 1$

h. **ELIF** ($\mathbf{t} > \mathbf{T}$) and ($\mathbf{D_T} = 0$), all demands have been met within time-frame
i. Set $\mathbf{t}$ = infinity
ii. Update **SS**

i. Update demand array

4. Save associated data for **SS**

4. Results

4.1. Base Case

We have run the simulation model for the base case scenarios for different types of fleets. For the base case for Conventional Fuel Buses, we used a running consumption of 0.57 lt./min. and a service consumption of 0.061 lt./min. with 10 buses and 60% fuel level. We found the total number of delays to be 1204 and a total delay of 511.83 minutes and an average delay of 1.08 minutes for the 30 days of simulation. From the cost point of view, the average fuel cost was $2076 and average delay costs were $12273 per day.

For the base case for Hybrid Buses, we used a running consumption of 0.355 lt./min. and a service consumption of 0.055 lt./min. with 10 buses and 60% fuel level. We found the total number of delays to be 1106 and a total delay of 3942.88 minutes and an average delay of 0.89 minutes for the 30 days of simulation. From the cost point of view, the average fuel cost was $1365 and average delay costs were $9463 per day.

For the base case for Electric Buses, we used a running consumption of 0.300 kWh and a service consumption of 0.050 kWh with 10 buses and 60% fuel level. We found the total number of delays to be 1736 and a total delay of 212346.41 minutes and an average delay of 50.92 minutes for the 30 days of simulation. From the cost point of view, the average fuel cost was $180 and average delay costs were $509631 per day.

We can see that the Fuel Buses have the least delay here and the Electric Buses have the least fuel cost but highest Delay Cost.

4.2. Sensitivity Analysis

Table 3: Description of parameters altered for Sensitivity Analysis

		Liquid Fuel			**Hybrid**			**Electric**		
Parameter	**Variable Name**	Min Value	Base Case	Max Value	Min Value	Base Case	Max Value	Min Value	Base Case	Max Value
Running Consumption	running_consumption	0.475	0.570	0.750	0.289	0.355	0.427	0.180	0.300	0.360
Idle/Service Consumption	service_consumption	0.037	0.061	0.068	0.033	0.055	0.061	0.010	0.050	0.070
Refuel Rate at station (taken as negative values)	refuel_consumption	-30	-30	-30	-30	-30	-30	-2	-2	-2
Tank Size/Capacity (lts. of kWh)	tank_size	150	150	150	150	150	150	240	240	240
Number of buses	n_buses	8	10	12	8	10	12	8	10	12
Initial level of tank/charge in %	level_mean	40	60	80	40	60	80	40	60	80

4.2.1. Conventional buses

4.2.1.1. Performance Analysis

Keeping the vehicle number and mean level coherent with the base case, we manipulate only the level of consumption for both running and service to see how the result of the simulated bus system is going to react to this change. In this case, the total number of refills required increases significantly as a response to the plummet in fuel efficiency. However, the average delay of all events doesn't change drastically while maintaining at a level of 1 min.

Table 4: Results of Sensitivity Analysis on Performance parameters for Conventional buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	running_consumption	**0.475**	**0.515**	**0.570**	**0.645**	**0.750**
	service_consumption	**0.037**	**0.053**	**0.061**	**0.063**	**0.068**
Half-width in minutes		0.897	0.934	0.899	0.984	1.052
Extra replications to be done for convergence		0	0	0	0	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		1094	1119	1204	1274	1315
Total delay in minutes		4582.37	4983.37	5113.83	5802.54	6475.35
Average of delay events in minutes		4.19	4.45	4.25	4.55	4.92
Average delay (all events) in minutes		1.02	1.09	1.08	1.22	1.31
Std. Deviation of delay (all events) in minutes		2.4	2.5	2.41	2.64	2.82
Maximum delay (all events) in minutes		18.05	17.89	15.18	18.25	20.21
Total number of deployments		4472	4554	4736	4769	4956
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		602	684	866	899	1086
Average running time for all buses per day in minutes		4158.05	4158.25	4164.16	4168.55	4169.7
Average service time for all buses per day in minutes		1505.93	1510.29	1507.46	1509.46	1506.47
Average refuel time for all buses per day in minutes		54.75	61.89	78.44	81.53	98.26
Average employee costs paid per day in dollars		1430	1433	1438	1440	1444
Average fuel costs paid per day in dollars		1449	1638	2076	2157	2600
Average delay costs paid per day in dollars		10998	11960	12273	13926	15541
Average maintenance costs paid per day in dollars		9550	9570	9603	9618	9643

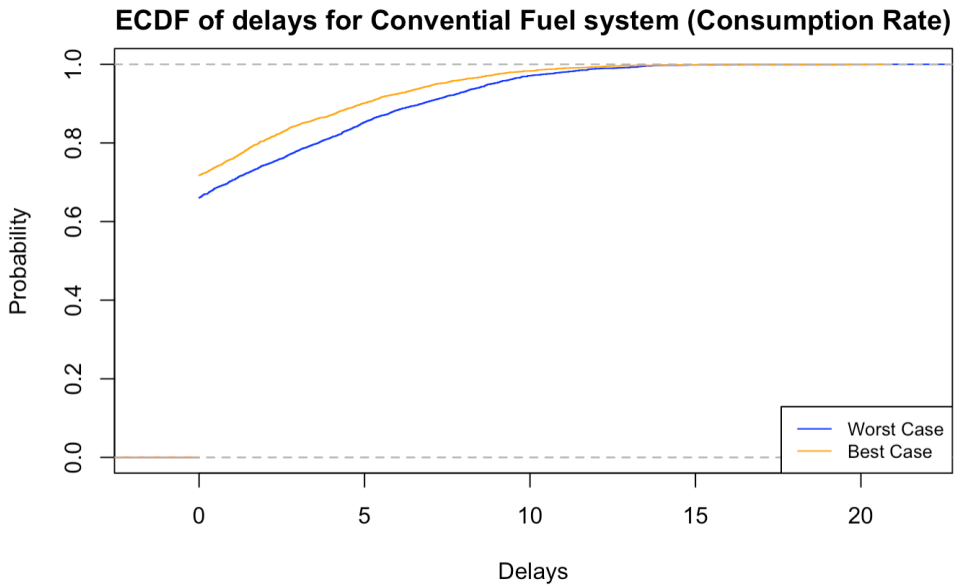


Figure 3: Best and Worst case ECDF graph for the delays seen in consumption rate for conventional fuel

4.2.1.2. Mean Fuel Level

To find out the effect of start-off level of fuel for each bus on efficiency of the simulated system, we alter the mean level of fuel while keeping the other variables constant to the base case. Systems with the least and highest level of fuel slightly outperformed the systems ranging in the middle in the sense of minimizing delay while less money and time expense are required for a system with higher start-off fuel.

Table 5: Results of Sensitivity Analysis on Mean Fuel level for Conventional buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	level_mean	**40**	**50**	**60**	**70**	**80**
Half-width in minutes		0.862	0.942	0.899	0.894	0.902
Extra replications to be done for convergence		0	0	0	0	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		1177	1221	1204	1162	1187
Total delay in minutes		4856.41	5433.35	5113.83	4822.37	4917.91
Average of delay events in minutes		4.13	4.45	4.25	4.15	4.14
Average delay (all events) in minutes		1.02	1.14	1.08	1.04	1.09
Std. Deviation of delay (all events) in minutes		2.31	2.52	2.41	2.4	2.42
Maximum delay (all events) in minutes		15.21	17.78	15.18	17.07	16.52
Total number of deployments		4768	4757	4736	4623	4499
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		898	887	866	753	629
Average running time for all buses per day in minutes		4147.13	4158.92	4164.16	4149.81	4155.65
Average service time for all buses per day in minutes		1513.43	1508.59	1507.46	1509.72	1508.22

Average refuel time for all buses per day in minutes	81.6	80.98	78.44	68.41	57.16
Average employee costs paid per day in dollars	1436	1437	1438	1432	1430
Average fuel costs paid per day in dollars	2159	2143	2076	1810	1512
Average delay costs paid per day in dollars	11655	13040	12273	11574	11803
Average maintenance costs paid per day in dollars	9589	9600	9603	9566	9554

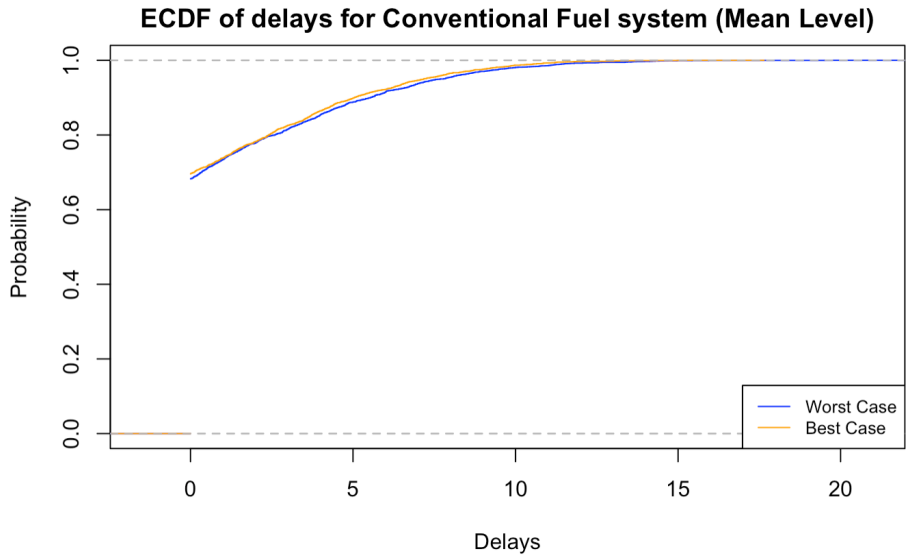


Figure 4: Best and Worst case ECDF graph for the delays seen in mean fuel rate for conventional fuel

4.2.1.3. Number of buses

We experiment with the system having a range of 8 to 12 buses while keeping other variables constant to the base case. We spot drastic changes in all delay associated result parameters by either adding or subtracting bus from the system.

Table 6: Results of Sensitivity Analysis on Number of buses for Conventional buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	n_buses	**8**	**9**	**10**	**11**	**12**
Half-width in minutes		10.395	3.977	0.899	0.267	0.08
Extra replications to be done for convergence		1412	182	0	0	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		2824	2387	1204	321	52
Total delay in minutes		116799.65	41017.29	5113.83	688.08	78.87
Average of delay events in minutes		41.36	17.18	4.25	2.14	1.52

Average delay (all events) in minutes	25.03	8.76	1.08	0.15	0.02
Std. Deviation of delay (all events) in minutes	27.84	10.65	2.41	0.72	0.21
Maximum delay (all events) in minutes	91.26	39.87	15.18	7.96	6.37
Total number of deployments	4667	4680	4736	4634	4579
Total number of route deployments	3870	3870	3870	3870	3870
Total number of refills deployments	797	810	866	764	709
Average running time for all buses per day in minutes	4169.43	4161.13	4164.16	4159.99	4164.8
Average service time for all buses per day in minutes	1499.63	1511.74	1507.46	1503	1500.97
Average refuel time for all buses per day in minutes	72.21	73.51	78.44	69.23	64.47
Average employee costs paid per day in dollars	1435	1437	1438	1433	1433
Average fuel costs paid per day in dollars	1911	1945	2076	1832	1706
Average delay costs paid per day in dollars	280319	98441	12273	1651	189
Average maintenance costs paid per day in dollars	9588	9596	9603	9573	9570

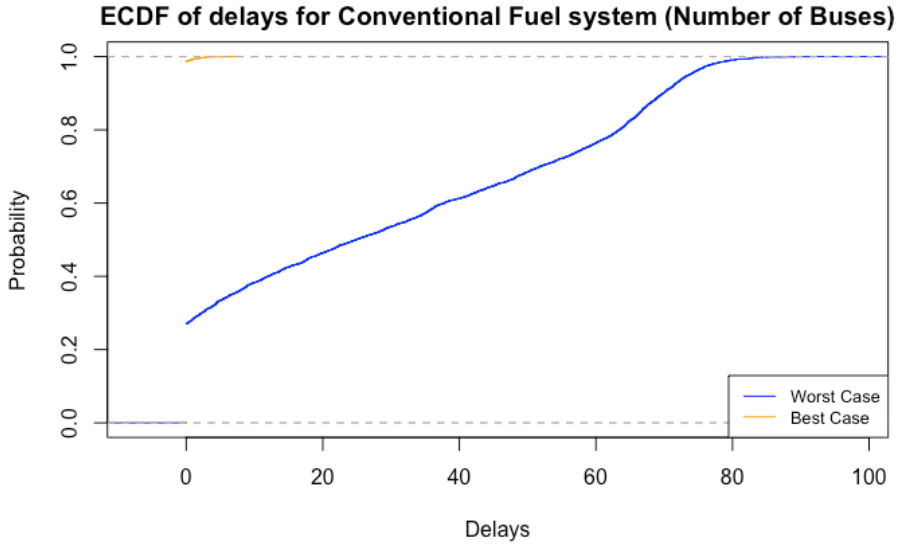


Figure 5: Best and Worst case ECDF graph for the delays seen in number of buses for conventional fuel

4.2.2. Hybrid buses

4.2.2.1. Performance Analysis

In this analysis we have varied the fuel consumption levels and observed the effects on delays and fuel, employee, and delay costs. We can observe a general decrease in delay levels and increase in fuel costs as we increase the consumption.

Table 7: Results of Sensitivity Analysis on Performance parameters for Hybrid buses

Parameter changed:	Percentile	0	25	50	75	100
	running_consumption	0.289	0.315	0.355	0.382	0.427
	service_consumption	0.033	0.048	0.055	0.059	0.061
Half-width in minutes		0.896	0.853	0.769	0.715	0.802
Extra replications to be done for convergence		0	0	0	0	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		1164	1162	1106	1060	1115
Total delay in minutes		4828.04	4531.35	3942.88	3642.26	4249.42
Average of delay events in minutes		4.15	3.9	3.56	3.44	3.81
Average delay (all events) in minutes		1.16	1.07	0.89	0.82	0.95
Std. Deviation of delay (all events) in minutes		2.4	2.28	2.06	1.92	2.19
Maximum delay (all events) in minutes		14.44	16.18	16.2	14.61	15
Total number of deployments		4173	4248	4436	4466	4470
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		303	378	566	596	600
Average running time for all buses per day in minutes		4155.11	4158.06	4150.57	4146.46	4158.5
Average service time for all buses per day in minutes		1508.71	1506.27	1508.59	1500.71	1511.73
Average refuel time for all buses per day in minutes		27.4	34.47	51.57	54.03	54.69
Average employee costs paid per day in dollars		1423	1425	1428	1425	1431
Average fuel costs paid per day in dollars		725	912	1365	1430	1447
Average delay costs paid per day in dollars		11587	10875	9463	8741	10199
Average maintenance costs paid per day in dollars		9504	9517	9537	9521	9561

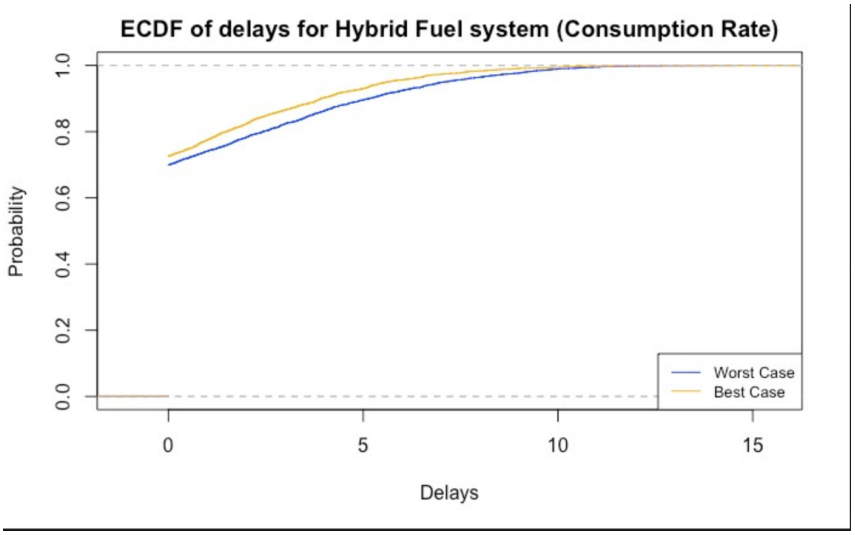


Figure 6: Best and Worst case ECDF graph for the delays seen in consumption rate for hybrid fuel

4.2.2.2. Mean Fuel Level

In this analysis we have varied the mean fuel levels and observed the effects on delays and fuel, employee, and delay costs. We can observe a general decrease in delay levels as we move towards the base case and a significant decrease in average fuel costs as we increase the mean fuel levels.

Table 8: Results of Sensitivity Analysis on Mean Fuel level for Hybrid buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	level_mean	**40**	**50**	**60**	**70**	**80**
Half-width in minutes		0.832	0.782	0.769	0.875 min	0.936
Extra replications to be done for convergence		0	0	0	0	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		1114	1058	1106	1171	1147
Total delay in minutes		4308.45	3988.12	3942.88	4623.8	4900.85
Average of delay events in minutes		3.87	3.77	3.56	3.95	4.27
Average delay (all events) in minutes		0.96	0.89	0.89	1.08	1.18
Std. Deviation of delay (all events) in minutes		2.23	2.1	2.06	2.34	2.51
Maximum delay (all events) in minutes		15.73	13.52	16.2	15.32	16.48
Total number of deployments		4469	4470	4436	4270	4171
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		599	600	566	400	301
Average running time for all buses per day in minutes		4157.55	4150.07	4150.57	4148.66	4143.51
Average service time for all buses per day in minutes		1517.87	1506.37	1508.59	1506.09	1504.54
Average refuel time for all buses per day in minutes		54.43	54.5	51.57	36.56	27.28
Average employee costs paid per day in dollars		1432	1428	1428	1423	1419
Average fuel costs paid per day in dollars		1440	1442	1365	967	722
Average delay costs paid per day in dollars		10340	9571	9463	11097	11762
Average maintenance costs paid per day in dollars		9569	9537	9537	9504	9478

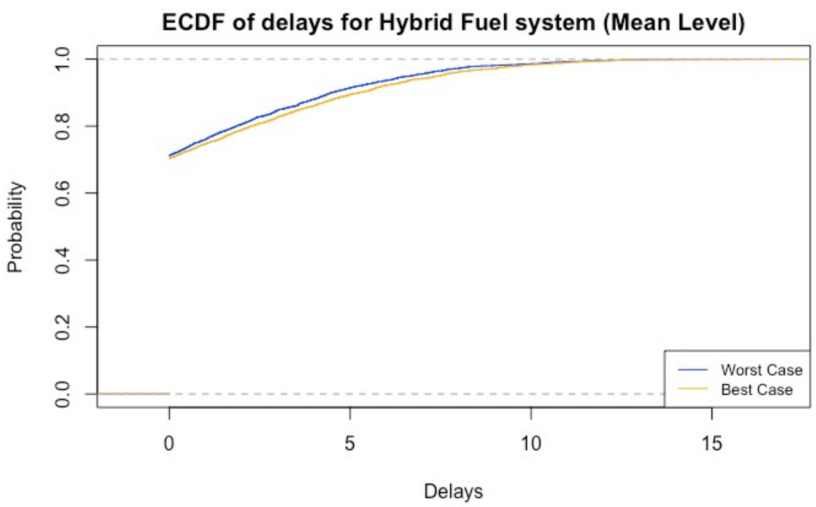


Figure 7: Best and Worst case ECDF graph for the delays seen in mean fuel level for hybrid fuel

4.2.2.3. Number of buses

In this analysis we have varied the number of buses in the system and observed the effects on delays and fuel, employee, and delay costs. We can observe a significant decrease in delay levels and delay costs and an increase in fuel costs as we move towards the base case.

Table 9: Results of Sensitivity Analysis on Number of buses for Hybrid buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	n_buses	**8**	**9**	**10**	**11**	**12**
Half-width in minutes		10.097	3.601	0.769	0.212	0.049
Extra replications to be done for convergence		1331	144	0	0	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		2814	2467	1106	251	39
Total delay in minutes		113251.21	36778.15	3942.88	473.55	40.99
Average of delay events in minutes		40.25	14.91	3.56	1.89	1.05
Average delay (all events) in minutes		26.04	8.34	0.89	0.11	0.01
Std. Deviation of delay (all events) in minutes		27.04	9.64	2.06	0.57	0.13
Maximum delay (all events) in minutes		87.25	40.49	16.2	8.03	3.32
Total number of deployments		4350	4409	4436	4338	4230
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		480	539	566	468	360
Average running time for all buses per day in minutes		4150.22	4140.37	4150.57	4150.51	4151.29
Average service time for all buses per day in minutes		1511.33	1508.23	1508.59	1509.62	1518.01
Average refuel time for all buses per day in minutes		43.43	48.92	51.57	42.39	32.61
Average employee costs paid per day in dollars		1426	1424	1428	1426	1425
Average fuel costs paid per day in dollars		1149	1294	1365	1123	864
Average delay costs paid per day in dollars		271803	88268	9463	1137	98
Average maintenance costs paid per day in dollars		9527	9515	9537	9523	9522

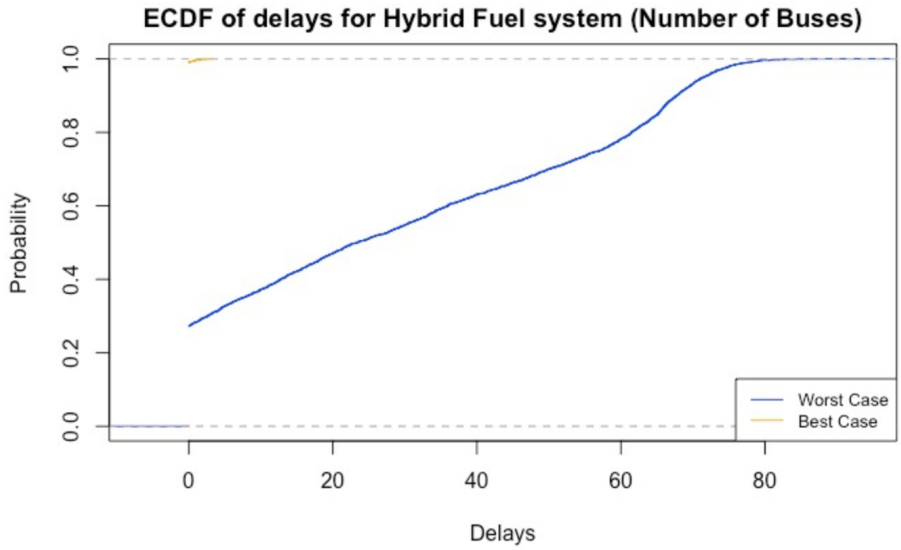


Figure 8: Best and Worst case ECDF graph for the delays seen in number of buses for hybrid fuel

4.2.3. Electric buses

4.2.3.1. Performance Analysis

Below are the simulation results of electric buses undergoing different combinations of running and servicing consumptions. In general, the electric buses system doesn't perform well under all conditions given that the average delay in all events is over 40 minutes for every circumstance that have been addressed in this chart which is extremely inefficient given the same parameters being under 2 mins for other two vehicles.

Table 10: Results of Sensitivity Analysis on Performance parameters for Electric buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	running_consumption	**0.240**	**0.270**	**0.300**	**0.330**	**0.360**
	service_consumption	**0.030**	**0.040**	**0.050**	**0.060**	**0.070**
Half-width in minutes		17.479	23.587	28.328	29.955	31.788
Extra replications to be done for convergence		5746	7389	10671	11935	13445
Total number of replications (days)		30	30	30	30	30
Total number of delays		1444	1652	1736	1882	1959
Total delay in minutes		82373.67	145522.22	212346.41	253690.62	297531.51
Average of delay events in minutes		57.05	88.09	122.32	134.8	151.88
Average delay (all events) in minutes		20.04	34.9	50.92	60.84	71.35
Std. Deviation of delay (all events) in minutes		46.81	63.17	75.86	80.22	85.13
Maximum delay (all events) in minutes		178.04	190.33	201.08	202.19	206.11
Total number of deployments		4110	4170	4170	4170	4170
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		240	300	300	300	300

Average running time for all buses per day in minutes	4148.79	4141.91	4142.01	4158.73	4154.26
Average service time for all buses per day in minutes	1503.78	1513.49	1507.85	1502.04	1507.58
Average refuel time for all buses per day in minutes	564.46	705.13	704.57	704.95	705.23
Average employee costs paid per day in dollars	1554	1590	1589	1591	1592
Average fuel costs paid per day in dollars	144	180	180	180	180
Average delay costs paid per day in dollars	197697	349253	509631	608857	714076
Average maintenance costs paid per day in dollars	13988	14311	14298	14323	14326

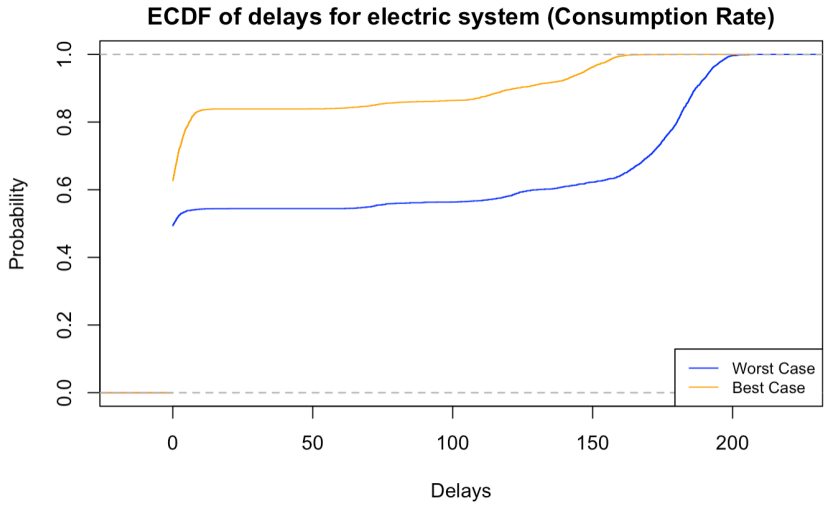


Figure 9: Best and Worst case ECDF graph for the delays seen in consumption rate for electric charge

4.2.3.2. Mean Fuel Level

Start-off battery level is crucial for the performance of electric buses since the charging time is extensively larger than the refuel time for liquid fuel buses. Here below in the chart, we change the initial battery level from 40 percent to 80 percent. The resulting parameters are even worse for situations where we are starting off with less than 80 percent of battery level for each bus, but the performance of the system is optimized suddenly when we change the level_mean to 80 percent. We'll have our explanation on this addressed in the next section.

Table 11: Results of Sensitivity Analysis on Mean Charge level for Electric buses

Parameter changed:	**Percentile**	**0**	**25**	**50**	**75**	**100**
	level_mean	**40**	**50**	**60**	**70**	**80**
Half-width in minutes		28.529	32.059	28.328	16.515	0.681
Extra replications to be done for convergence		10824	13675	10671	3608	0
Total number of replications (days)		30	30	30	30	30
Total number of delays		2699	2118	1736	1454	924
Total delay in minutes		416693.71	337435.94	212346.41	74883.29	2970.28
Average of delay events in minutes		154.39	159.32	122.32	51.5	3.21
Average delay (all events) in minutes		99.93	80.94	50.92	18.22	0.77

Std. Deviation of delay (all events) in minutes	76.4	85.86	75.86	44.23	1.82
Maximum delay (all events) in minutes	186.49	211.15	201.08	164.59	11.91
Total number of deployments	4170	4169	4170	4110	3875
Total number of route deployments	3870	3870	3870	3870	3870
Total number of refills deployments	300	299	300	240	5
Average running time for all buses per day in minutes	4140.04	4139.83	4142.01	4142.53	4143.11
Average service time for all buses per day in minutes	1506.44	1506.19	1507.85	1510.81	1504.51
Average refuel time for all buses per day in minutes	704.21	703.71	704.57	565.93	11.47
Average employee costs paid per day in dollars	1588	1587	1589	1555	1415
Average fuel costs paid per day in dollars	180	179	180	144	3
Average delay costs paid per day in dollars	1000065	809846	509631	179720	7129
Average maintenance costs paid per day in dollars	14289	14287	14298	13993	12733

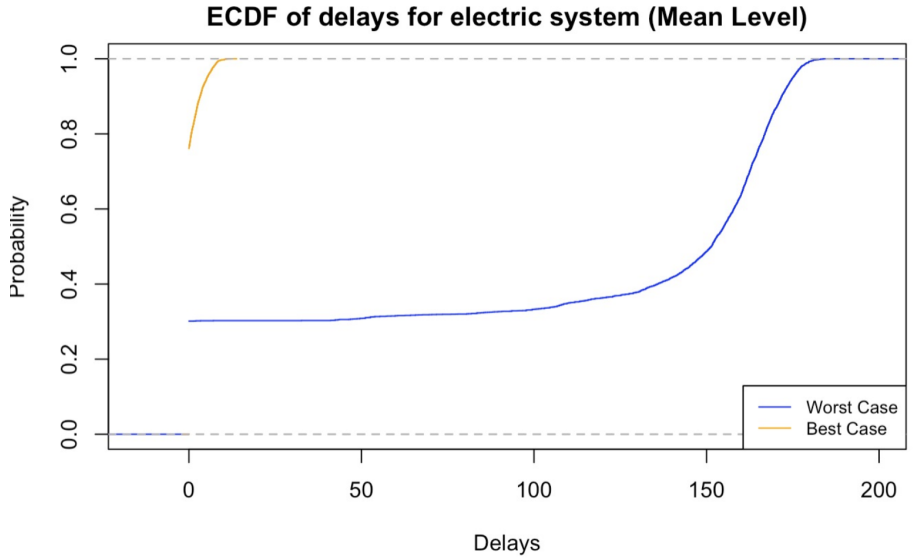


Figure 10: Best and Worst case ECDF graph for the delays seen in mean fuel level for electric charge

4.2.3.3. Number of buses

Unlike what is the case for liquid fuel buses and hybrid buses, the result from electrical bus simulation doesn't respond as much if we add or subtract additional vehicles to the fleet. But still, the delay related results are improving which might imply that more electric buses are needed so as to make the electric bus system favorable among these three bus transit systems.

Table 11: Results of Sensitivity Analysis on Number of buses for Electric buses

Parameter changed:	Percentile	0	25	50	75	100
	n_buses	8	9	10	11	12
Half-width in minutes		36.639	32.05	28.328	23.367	19
Extra replications to be done for convergence		17870	13667	10671	7431	3832
Total number of replications (days)		30	30	30	30	30
Total number of delays		2741	2339	1736	1169	792
Total delay in minutes		374387.64	281174.29	212346.41	145497.79	97656.29
Average of delay events in minutes		136.59	120.21	122.32	124.46	123.3
Average delay (all events) in minutes		91.09	67.92	50.92	34.88	23.52
Std. Deviation of delay (all events) in minutes		98.12	85.83	75.86	62.58	50.81
Maximum delay (all events) in minutes		240.78	211.39	201.08	178.61	167.81
Total number of deployments		4110	4140	4170	4171	4152
Total number of route deployments		3870	3870	3870	3870	3870
Total number of refills deployments		240	270	300	301	282
Average running time for all buses per day in minutes		4141.19	4148.19	4142.01	4144	4150.74
Average service time for all buses per day in minutes		1505.68	1508.45	1507.85	1506.03	1512.84
Average refuel time for all buses per day in minutes		563.89	635.15	704.57	707.71	663.06
Average employee costs paid per day in dollars		1553	1573	1589	1589	1582
Average fuel costs paid per day in dollars		144	162	180	180	169
Average delay costs paid per day in dollars		898530	674818	509631	349195	234375
Average maintenance costs paid per day in dollars		13974	14157	14298	14305	14235

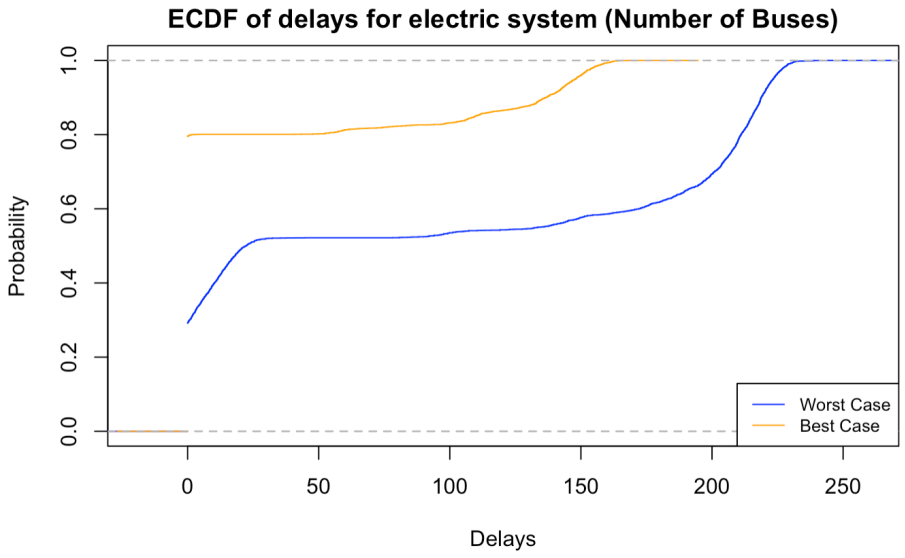


Figure 11: Best and Worst case ECDF graph for the delays seen in number of buses for electric charge

4.3. Variance Reduction

For variance reduction, we have used the concept of Common Random Numbers (CRN) to compare the three different alternatives, i.e. Conventional, Hybrid and Electrical buses. We have taken one base case for each alternative and using that base case we do a comparison study on the other two cases. This is achieved by storing all interarrival and service times for the base cases and injecting the same interarrival and service times for the associated bus deployments in the other two cases. The results are discussed in the following section.

4.4. Comparison of Alternatives

As mentioned in section 5.3, we have simulated the alternatives with variance reduction to compare the three alternatives on the same grounds. The inherent difference, which is the difference in the fuel systems for the buses, in these alternatives produce different results, which is evident from the difference in mean and variance of delays as discussed in the previous section. The table below shows the results obtained from the simulations done with the three base cases and their associated variance reduced alternatives.

Table 12: Results of Variance Reduction on Comparison of Alternatives

Parameter changed:	Base Case: Conventional			Base Case: Hybrid			Base Case: Electric		
	Conventional	Hybrid	Electric	Conventional	Hybrid	Electric	Conventional	Hybrid	Electric
n_buses	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
level_mean	**60**	**60**	**60**	**60**	**60**	**60**	**60**	**60**	**60**
running_consumption	**0.570**	**0.355**	**0.300**	**0.570**	**0.355**	**0.300**	**0.570**	**0.355**	**0.300**
service_consumption	**0.061**	**0.055**	**0.050**	**0.061**	**0.055**	**0.050**	**0.061**	**0.055**	**0.050**
Half-width in minutes	0.899	0.724	28.367	0.862	0.684	27.874	2.59	2.385	28.328
Extra replications to be done for convergence	0	0	10701	0	0	10331	61	47	10671
Total number of replications (days)	30	30	30	30	30	30	30	30	30
Total number of delays	1204	1049	1751	1165	1063	1753	1793	1724	1736
Total delay in minutes	5113.83	3601.55	212197.9	4917.68	3495.74	206956.04	18913.74	16125.05	212346.41
Average of delay events in minutes	4.25	3.43	121.19	4.22	3.29	118.06	10.55	9.35	122.32
Average delay (all events) in minutes	1.08	0.81	50.89	1.04	0.79	49.63	4.09	3.7	50.92
Std. Deviation of delay (all events) in minutes	2.41	1.94	75.97	2.31	1.83	74.65	6.94	6.39	75.86
Maximum delay (all events) in minutes	15.18	14.45	197.92	15.38	13.71	197.75	34.88	31.99	201.08
Total number of deployments	4736	4441	4170	4733	4445	4170	4629	4357	4170
Total number of route deployments	3870	3870	3870	3870	3870	3870	3870	3870	3870
Total number of refills deployments	866	571	300	863	575	300	759	487	300
Average running time for all buses per day in minutes	4164	4157	4149	4161	4154	4147	3860	3853	4142
Average service time for all buses per day in minutes	1507	1507	1507	1506	1506	1506	2096	2096	1508
Average refuel time for all buses per day in minutes	78	52	705	78	52	706	69	44	705
Average employee costs paid per day in dollars	1438	1429	1590	1436	1428	1590	1506	1498	1589
Average fuel costs paid per day in dollars	2076	1385	180	2072	1375	199	1821	1171	180
Average delay costs paid per day in dollars	12273	8644	509275	11802	8390	496694	45393	38700	509631
Average maintenance costs paid per day in dollars	9603	9546	14314	9595	9539	12852	10061	10009	14298

5. Discussion of results/recommendations

5.1. Sensitivity Analysis on Conventional Buses

Once the team was able to model, verify and validate the base cases with the help of a structured walkthrough, we switched to analyzing all three different systems under various scenarios to understand how the model would perform and if it would pass the intuition test, i.e., if the model would behave the way we expect it to when we increase the parameters that serve as inputs. The main parameters under consideration are the running consumption, which is the amount of fuel used by a bus when moving, the service consumption, which is the amount of fuel used when the bus is stationary, the number of buses in the system as a whole, and finally the mean level of fuel/charge that is present in the bus at the start of the day. A point to note about the consumption levels is in practice, we usually see a correlation between an increase in running consumption and idle consumption as this is usually due to factors such as a decrease in engine performance or a degradation in some other system which affects the bus as a whole and not just any one system in isolation.

For conventional fuel buses, we see that a systematic increase in the running and idle consumption levels translates to a steady increase in metrics such as the number of delays which increases from 4500 minutes (best case scenario in terms of fuel consumption) to 6475 minutes (worst scenario for fuel consumption), the total time of delays, and the average delays. This result does make sense intuitively, as we expect an increase in the fuel consumption rates to lead to more refueling, which in turn could lead to more delays as the simulation prioritizes the refueling cycle as opposed to satisfying demand. In terms of costs, while there is an increase in costs as the bus performance worsens, it is not as significant a change as is with the delay time.

The mean level of fuel in each of the buses at the start of the day had a range from 40% to 80% in increments of 10%. Interestingly, changing this parameter did not yield any significant trends, as the total number of delays and the total delays in minutes remained relatively constant.

The final parameter analyzed was the number of buses present in the system, which was varied from 8 buses to 12 buses. As expected, the number of buses had a huge impact on the total number of delays (ranging from 2824 for 8 buses to 50 delays for 12 buses) and on the total time of delays in minutes. This also affected the costs associated with delays as expected, but the fuel and employee costs interestingly did not see any significant changes. The maximum delay did also see a significant improvement, reducing by almost 95% in the process.

5.2. Sensitivity Analysis on Hybrid Buses

We need to examine the robustness of our model and the extent to which results are affected by changes in methods, models, values of unmeasured variables, or assumptions. Some of the changes we have considered are changes in consumption levels, number of buses and mean level.

For hybrid buses, when we increased the consumption levels, there was a decrease in the delays caused on the routes from a total delay of 4828.04 minutes for 0.289 lpm consumption to a delay of 3642.26 for 0.382 lpm consumption. We can observe a monotonic decrease in all delay related parameters with an increase in consumption levels. However, as expected the fuel and maintenance costs increase with an increase in fuel consumption. Concerning the mean fuel level to start with, we have the lowest delay at 50% gas which is our base case with an increasing delay levels as the gas level changes.However, the average fuel costs linearly decreased with an increased fuel level which is expected as we would need less refuels when there is a high initial fuel level.

Next we perform sensitivity analysis by varying the number of buses in the system. There is a significant decrease in delay from 113251 minutes to 0.99 minutes as we increase the number of buses 8 to 12 and this would also translate to a decrease in delay costs. We can clearly see that 12 buses is optimal for our system. However, we observe that the average fuel costs increase as buses increase to 10 and decrease thereafter, which can be linked to the initial mean fuel level present in the buses and the distance they have to travel.

5.3. Sensitivity Analysis on Electric Buses

By analyzing the performance and cost associated result of the electric bus with different set of running consumption and service consumption parameters ranging from 0.24 /0.03 kwh per min to 0.36 / 007 kwh per min, we could spot a monotonic trend of increasing in all of the delay affiliated parameters such as total number of delay, average length of delay and also average delay costs paid per day in dollar by increasing the running and service consumption parameter. Besides this trend that we've spotted, we should notice that the average delay of all events is over 40 minutes for every circumstance that has been addressed in this chart which is extremely inefficient given the same parameters being under 10 mins for the other two vehicles type.

The resulting parameters are even worse for situations where we are starting off with less than 80 percent of battery level for each bus, but the performance of the system is optimized suddenly when we change the level_mean to 80 percent. When we started the electric bus with 80 percent of battery capacity, we ended up with a simulation performance similar to the base case of other two types of buses with an average delay around 1 min. However, different between the two systems, the simulation result shows that electric buses merely need a recharge during their shift of the day as the recharge deployment counts and dollar spent on recharge is almost zero for both cases.

5.4. Comparison Study

Considering only the base case scenario for all three types of buses, There is an obvious distinction with respect to the performance between electric bus and fuel & hybrid bus given the strength of fuel and hybrid buses being able to be refilled in a rather short time period. The average fuel time for fuel-intrinsic buses to be reconsidered as deployable is around 50 minutes whereas the same parameter is 10 times larger for electric buses.

The total number of delays doesn't vary much for each type of bus. However, the average of delay events for electrical vehicles is 122 minutes which converts to 2 hours. This means that for all the delay that happened for the electrical bus system, the average wait time for the passengers to get on the bus is 2 hours. This is an extremely long time especially considering the same outcome being 4,25 and 3.43 accordingly which are reasonable wait time for a transit system. This parameter will deteriorate the efficiency of the electric bus system under the base case assumption. As addressed before in the sensitivity analysis, this issue could be solved by allowing more electric storage beforehand or by generously increasing the amount of electrical buses that are dispatch-able.

By only comparing the result statistics between fuel bus and hybrid bus, we can conclude that hybrid bus outperforms liquid fuel bus because of the less delay incidents, average delay time and also the less cost indexes for running servicing and refueling. However, the difference between these two systems are, as we've said before, indistinguishable and can be easily reverted given minor changes in the base case.

6. Summary and Conclusions

With the ever increasing push from communities and governments to reduce reliance on the burning of fossil fuels, exploring alternative sources of energy for daily transportation has increasingly become a topic of interest across the world. One area where alternative sources of fuels and technologies can be used in the movement of people with the help of buses within cities and regions. This report aims to address this by providing city planners and other stakeholders a way to compare and contrast different technologies and fuel sources and to ultimately help inform and aid them in their decision making processes. As a part of the project, the team has worked together to model a bus system in the city of Ann Arbor that contains a subset of the routes found in real life. The team was able to successfully incorporate several features such as time varying demand, bus performance, real life driving cycles, travel times, service times and recharging and refueling cycles in order to provide a comprehensive solution to aid the different stakeholders. While the focus of the simulation was on tracking delays throughout the system across various scenarios and ensuring that there is convergence on the

delay value with a chosen half-width of 1.5 minutes, cost and performance metrics were also measured to provide a holistic review of the entire system. To further develop the model, more attention could be given to data collection for the travel times between the different stops as well as in researching more efficient recharging and refueling methods which would help reduce the overall delay found in the system. Efforts could also be focused on developing better technologies that would aid in faster charging of electric vehicles, which in turn would address the delays caused in the purely electric system. Finally, understanding the needs and wants of the people in the community could go a long way in ensuring that they remain satisfied and incentivised to step away from personal, conventional fuel vehicles, which would help the government meet their goals in reducing their dependence on fossil fuels.

7. References

1. https://www.epa.gov/ghgemissions/global-greenhouse-gas-emissions-data#Sector (Used in the opening paragraph for the exec summary)
2. https://data.bloomberglp.com/professional/sites/24/2018/05/Electric-Buses-in-Cities-Report-BNEF-C40-Citi.pdf (Running Consumption for electric bus)
3. https://reader.elsevier.com/reader/sd/pii/S0968090X1930868X?token=C3B3248EB4C162EB3684F1AC5F62BD62EBD78894370EECE24460412E8FC57B76179FCE4457AE4721F83A4412CD59A34D&originRegion=us-east-1&originCreation=20211206221214(Electric demand cost)
4. https://www.oneshift.com/new-cars/car-fuel-consumption/5545/yutong-zk6958hq-bus/(Running Consumption for fuel bus)
5. https://www.energy.gov/eere/vehicles/fact-861-february-23-2015-idle-fuel-consumption-selected-gasoline-and-diesel-vehicles(idle consumption for fuel bus)
6. https://hal.archives-ouvertes.fr/hal-02169856/document(idle consumption for electric bus)
7. https://escholarship.org/content/qt0fn8s2jh/qt0fn8s2jh.pdf
8. https://link.springer.com/content/pdf/10.1007/s12469-014-0086-z.pdf
9. https://calstart.org/wp-content/uploads/2018/10/Peak-Demand-Charges-and-Electric-Transit-Buses.pdf
10. Giovanni De Filippo, Vincenzo Marano, Ramteen Sioshansi, Simulation of an electric transportation system at The Ohio State University, Applied Energy,Volume 113,2014,Pages1686-1691,ISSN0306-2619,https://doi.org/10.1016/j.apenergy.2013.09.011
11. A. Kontou, J. Miles, Electric buses: lessons to be learnt from the Milton Keynes demonstration project, defining the future of sustainability and resilience indesign, Eng. Constr. 118 (2015) 1137–1144.
12. D. Perrotta, J.L. Macedo, R.J.F. Rossetti, J.F. de Sousa, Z. Kokkinogenis, B.Ribeiro, J.L. Afonso, Route planning for electric buses: a case study in Oporto,transportation: can we do more with less resources? Porto 2013, in: 16thMeeting of the Euro Working Group on Transportation, 111, 2014, pp.1004–1014.

13. M. Rogge, S. Wollny, D.U. Sauer, Fast charging battery buses for the electrification of urban public transport—a feasibility study focusing on charging infrastructure and energy storage requirements Energies 8 (2015)4587–4606.
14. 1.Nyman, J.; Olsson, O.; Grauers, A.; Östling, J.; Ohlin, G.; Pettersson, S. A user-friendly method to analyze cost effectiveness of different electric bus systems. In Proceedings of the 30th International Electric Vehicle Symposium & Exhibition, Stuttgart, Germany, 9−11 October 2017. [**Google Scholar**]
15. S. Pelletier, O. Jabali, G. Laporte: 50th anniversary invited article—goods distribution with electric vehicles: review and research perspectives Transport Sci, 50 (1) (2016), pp. 3-22
16. D. Goeke, M. Schneider: Routing a mixed fleet of electric and conventional vehicles Eur J Oper Res, 245 (1) (2015), pp. 81-99
17. P. Lebeau, C. de Cauwer, J. van Mierlo, C. Macharis, W. Verbeke, T. Coosemans Conventional, hybrid, or electric vehicles: which technology for an urban distribution centre?
18. J.H.R. van Duin, L.A. Tavasszy, H.J. Quak: Towards E(lectric)- urban freight: first promising steps in the electric vehicle revolution Eur Transport Trasporti Europei, 54 (2013), p. 9
19. Gonçalves F, Sónia R Cardoso, Susana Relvas, Barbosa-Póvoa APFD. Optimization of a distribution network using electric vehicles: A VRP problem. In: the Proceedings of the IO2011-15 Congresso da associação Portuguesa de Investigação Operacional, Coimbra, Portugal 2011:18–20
20. Tiechert, O.; Chang, F.; Ongel, A.; Lienkamp, M. Joint Optimization of Vehicle Battery Pack Capacity and Charging Infrastructure for Electrified Public Bus Systems. *IEEE Trans. Transp. Electrif.* **2019**, *5*, 672–682.
21. Xylia, M.; Leduc, S.; Patrizio, P.; Kraxner, F.; Silveira, S. Locating charging infrastructure for electric buses in Stockholm. *Transp. Res. Part C* **2017**, *78*, 183–200.

Appendix

Bursley Baits Route

Route	Interarrival	STD
CCTC	0	0
Stockwell Hall	1.31	1.04
Cardiovascular Center	0.69	0.65
Glen Catherine	3	0.48
Mitchell Field	5.11	0.47
Pierpont Commons	1.69	0.36
Bursley	2.31	0.70
Baits II	1	0.62
Baits 1	1.2	0.21
Baits II	1.2	0.19
Bursley	1	0.61
Pierpont Commons	2.31	0.92
Mitchell Field	1.69	0.36
Glen/Catherine	5.11	0.47
Rackham	1.79	0.48
CCTC	2.13	1.09

Northwood Route

CCTC	0	0.00
Glen + Catherine	1.3	0.88
Fuller + Cedar Bend	2.7	0.85
Pierpont Commons Murfin	2.78	0.30
Northwood III Outbound	1.22	0.38
Northwood I Outbound	0.3	0.12
Cram Circle Outbound	0.3	0.16
Northwood III Outbound	1.3	0.19
Fire Station Outbound	2.1	0.34
Plymouth Road Crosswalk	2	0.79
Northwood IV	0.7	0.27
Northwood Community Center	1.3	0.29
Hayward.Hubbard Outbound	1	0.61
NCAC Hubbard Outbound	1	0.41
Northwood V(1)	0.49	0.27

Northwood V(2)	1.51	0.17
NCAC Hubbard Inbound	0.49	0.32
Hubbard + Huron Parkway	1	0.23
Hayward/Hubbard Inbound	1	0.35
FXB Inbound	3.81	0.56
Cooley Lab Inbound	2.19	0.94
Pierpont Commons	1.61	1.76
Fuller Road at Mitchell Field	4.39	0.70
Zina Pitcher	4.42	1.37
Lloyd Observatory	1.58	0.41
Stockwell Hall Inbound	1.5	0.84
CCTC	1.5	0.92

Diag to Diag Route

Geddes + CCTC	0	0
East Quad Church	3.76	0.57
Henderson House	1.8	0.73
Oxford Housing	3.84	0.27
Trotter House	1.75	0.45
CCTC	2.72	0.42
Power Center	1.42	0.40
Glen/Catherine	2.37	0.45
Fuller+ Cedar	4.22	0.40
Ford Presidential Library	1.28	0.13
Cooley Lab Inbound	0.46	0.24
Pierpont Commons	0.5	0.02
Fuller Road at Mitchell Field	2.76	0.84
Glen/Catherine Inbound	3.12	0.54
Rackham Building	1.3	0.36
Geddes + CCTC	1.42	0.34

Oxford Shuttle

Geddes + CCTC	0	0.00
East Quad	1.85	0.57
Henderson House	1.01	0.73
Oxford House	2.13	0.27
Trotter House	1.67	0.45
Shapiro Library	2.75	0.51
S University + State	0.98	0.32
State + SU	0.4	0.86
Michigan League	2.53	0.71
CCTC	1.67	0.24

System State

SS = (n_buses, curr_dem, buses_deployed, buses_recharge, buses_standstill)

Monitoring System State

Time	System State	Bus	Charge	Route	State	Event	Process Time	Demand-Current	Demand-Actual	Demand-Charge
0	(1,1,1,0,0)	1	50	'1'	1	0	0	0	0	15
10.3	(1,0,1,0,0)	1	48	'1'	1	1	10.3	-	-	-
12.3	(1,0,1,0,0)	1	47.5	'1'	1	0	2.0	-	-	-

Monitoring Bus Deployments

Time	Demand-Current	Demand-Actual	Demand-Charge	Bus	Charge	Route	Event Array	Time Array
0	0	0	15	1	50	'1'	[1, 0, 1]	[12, 35, 41]
40	41	40	20	1	35	'2'	[1, 0, 1]	[48, 50, 55]

In [1]:

```
import pandas as pd
import numpy as np
import numpy.random as random
from route_functions import *
import matplotlib.pyplot as plt
import scipy.stats as st
import statsmodels.api as sm

# to ignore warning on calculations
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
path = r"C:\Users\krishrao\Desktop\Laptop Backup\Krishna\Fall'21\IOE 574\Term Project\IOE574_Project\da
```

In [3]:

```
# intializing route table
routes = pd.read_excel('Model_Parameters.xlsx', 'Routes')

n_buses = 10                            # number of buses

# For liquid fuel buses
running_consumption = 0.355             # fuel(lts.)/kWh per minute
service_consumption = 0.055             # fuel(lts.)/kWh per minute
refuel_consumption = -30                # -30 for fuel, -2 for electric
tank_size = 150                         # 150 lts. / 240 kWh
level_mean = 80                         # normal distribution.. can be changed
level_std = 0                           # currently constant at level_mean
refuel = 'refill'                       # 'refill', recharge'
fuel_rate = 0.882                       # dollar 0.882 per lt. / 0.1275 per kWhr
maintain_rate = 1.67                    # dollar per min 1.67 for fuel, 2.25 for elctric

'''
# For electric buses
running_consumption = 0.30              # fuel(lts.)/kWh per minute
service_consumption = 0.05              # fuel(lts.)/kWh per minute
refuel_consumption = -2                 # -30 for fuel, -2 for electric
tank_size = 240                         # 150 lts. / 240 kWh
level_mean = 50                         # normal distribution.. can be changed
level_std = 0                           # currently constant at level_mean
refuel = 'recharge'                     # 'refill', recharge'
fuel_rate = 0.1275                      # dollar 0.882 per lt. / 0.1275 per kWhr
maintain_rate = 2.25                    # dollar per min 1.67 for fuel, 2.25 for elctric
'''

#-----
```

```
# customizable parameters
#average_bus_speed = 30 miles /hr
refuel_stations = 2
conversion_factor = 100/tank_size

# cost rates
emp_rate = 15                           # per hour basis
delay_rate = 2.4                        # dollar per min ** reason


# setting up initial values for simulation
n_routes = 4                            # number of routes
SimTime = 720

SS_cols = ['Time', 'System_State', 'Bus', 'Charge', 'Route',
           'State', 'Event', 'Process_Time', 'Demand_Current',
           'Demand_Actual', 'Demand_Charge']
BD_cols = ['Time', 'Demand_Current', 'Demand_Actual', 'Demand_Charge',
           'Bus', 'Charge', 'Route', 'Event_Array', 'Time_Array']
```

In [4]:

```
replicates = 30
ss_table = pd.DataFrame(columns=SS_cols+['Replication'])
bd_table = pd.DataFrame(columns=BD_cols+['Replication'])

for i in range(replicates):
    print('-----\nRunning buses on Day ', i+1)                    # intializing required variabes
    buses = [bus(level_mean, level_std) for i in range(n_buses)]  # fleet of buses
    t = 0                                                         # start time of simulation
    T = SimTime
    demand_at, demand_r, demand_c = gen_demands(n_routes, T)      # updating demands
    demand_ct = demand_at.copy()                                  # demand current time and demand act
    dct_flag = 0                                                  # demand at current time
    ss_tab, bd_tab = fleet_simulation(t, T, routes, buses, refuel, running_consumption,
                                      service_consumption, refuel_stations, refuel_consumption, convers
                                      demand_at, demand_ct, demand_r, demand_c, dct_flag, SS_cols, BD_c
    ss_tab['Replication'] = i+1
    bd_tab['Replication'] = i+1
    ss_table = ss_table.append(ss_tab, ignore_index=True)
    bd_table = bd_table.append(bd_tab, ignore_index=True)
```

```
-----
Running buses on Day  1
-----
Running buses on Day  2
-----
Running buses on Day  3
-----
Running buses on Day  4
Stuck in a loop!
Updating next bus event
        Time - 240.0
        Time Check - 240.0
        Demands - [255.0, 255.0, 270.0, 270.0, 270.0, 285.0] [255.0, 255.0, 270.0, 270.0, 270.0, 285.0]
['1', '3', '1', '2', '3', '1'] [25.0, 25.0, 25.0, 25.0, 25.0, 25.0]
        dct_flag - 0
-----
Running buses on Day  5
-----
Running buses on Day  6
Stuck in a loop!
Updating next bus event
        Time - 660.0
        Time Check - 660.0
        Demands - [675.0, 680.0, 680.0, 690.0, 700.0, 700.0] [675.0, 680.0, 680.0, 690.0, 700.0, 700.0]
['1', '2', '4', '1', '2', '4'] [25.0, 25.0, 25.0, 25.0, 25.0, 25.0]
        dct_flag - 0
-----
Running buses on Day  7
-----
Running buses on Day  8
-----
Running buses on Day  9
```

```
-----
Running buses on Day  10
-----
Running buses on Day  11
-----
Running buses on Day  12
-----
Running buses on Day  13
Stuck in a loop!
Updating next bus event
        Time - 420.0
        Time Check - 420.0
        Demands - [435.0, 435.0, 440.0, 440.0, 450.0, 450.0] [435.0, 435.0, 440.0, 440.0, 450.0, 450.0]
['1', '3', '2', '4', '1', '3'] [25.0, 25.0, 25.0, 25.0, 25.0, 25.0]
        dct_flag - 0
-----
Running buses on Day  14
-----
Running buses on Day  15
-----
Running buses on Day  16
-----
Running buses on Day  17
-----
Running buses on Day  18
-----
Running buses on Day  19
-----
Running buses on Day  20
-----
Running buses on Day  21
-----
Running buses on Day  22
-----
Running buses on Day  23
-----
Running buses on Day  24
-----
Running buses on Day  25
-----
Running buses on Day  26
-----
Running buses on Day  27
-----
Running buses on Day  28
-----
Running buses on Day  29
-----
Running buses on Day  30
```

ss_table

bd_table

Converging the model on delays

In [5]:

```
# delay analysis on deployments
num_delay = sum(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])>0)
total_delay = sum(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])))
avg_delay = round(np.mean(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual']
std_delay = round(np.std(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])
max_delay = max(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])))

# cost analysis
run_time, ser_time, ref_time, emp_cost, fuel_cost, delay_cost = cost_analysis(n_buses, replicates, refu
                                                                bd_table, emp_rate, fuel_
                                                                refuel_consumption)


# convergence on delays
beta = 1.5
alpha = 0.05
```

```
ci_n = 1-alpha/2
n_beta = (st.t.ppf(ci_n, df=replicates-1)*std_delay
         /np.sqrt(replicates))
'''
print('---\nCurrent half-width - ', round(n_beta, 3), 'min')
print('Current replications -', replicates)
print('Number of bus deployments -', bd_table.shape[0])

while n_beta > beta:
    s2_n =  (std_delay)**2
    t = st.t.ppf(ci_n, df=replicates-1)
    n_rep = int((t*np.sqrt(s2_n)/beta)**2) + 1
    n_rep = int(n_rep*replicates/replicates) + 1
    print('The extra replications to be done - ', n_rep-replicates)

    for h in range(replicates, n_rep):
        print('-----\nRunning buses on Day ', h+1)                      # intializing required variabes
        buses = [bus(level_mean, level_std) for i in range(n_buses)]   # fleet of buses
        t = 0                                                          # start time of simulation
        T = SimTime
        demand_at, demand_r, demand_c = gen_demands(n_routes, T)     # updating demands
        demand_ct = demand_at.copy()                                 # demand current time and demand a
        dct_flag = 0                                                 # demand at current time
        ss_tab, bd_tab = fleet_simulation(t, T, routes, buses, refuel, running_consumption,
                                          service_consumption, refuel_stations, refuel_consumption, con
                                          demand_at, demand_ct, demand_r, demand_c, dct_flag, SS_cols,
        ss_tab['Replication'] = h + 1
        bd_tab['Replication'] = h + 1
        ss_table = ss_table.append(ss_tab, ignore_index=True)
        bd_table = bd_table.append(bd_tab, ignore_index=True)

    replicates = n_rep
    n_beta = (st.t.ppf(ci_n, df=replicates-1)*std_delay/np.sqrt(replicates))
    print('---\nThe new half-width', round(n_beta, 3), 'min')
    print('Current replications -', replicates)
    print('Number of bus deployments -', bd_table.shape[0])
'''
print('---')
```

Model Analysis

In [6]:

```
# half-width analysis
print('---\nCurrent half-width - ', round(n_beta, 3), 'min')
print('Current replications -', replicates)
print('Number of bus deployments -', bd_table.shape[0])
if n_beta > beta:
    s2_n =  (std_delay)**2
    t = st.t.ppf(ci_n, df=replicates-1)
    n_rep = int((t*np.sqrt(s2_n)/beta)**2) + 1
    n_rep = int(n_rep*replicates/replicates) + 1
    print('The extra replications to be done - ', n_rep-replicates)


# delay analysis on deployments
num_delay = sum(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])>0)
total_delay = sum(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])))
avg_delay = round(np.mean(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual']
std_delay = round(np.std(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])
max_delay = max(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])))

# cost analysis
run_time, ser_time, ref_time, emp_cost, fuel_cost, delay_cost = cost_analysis(n_buses, replicates, refu
                                                                              bd_table, emp_rate, fuel_
                                                                              refuel_consumption)


print('---\nDelay Analyis')
print('\tTotal number of replications (days) -', replicates)
print('\tTotal number of delays -', num_delay)
```

```
print('\tTotal delay in minutes -', round(total_delay, 2))
print('\tAverage of delay events in minutes -', round(total_delay/num_delay, 2))

print('\n\tAverage delay (all events) in minutes -', round(avg_delay, 2))
print('\tStd. Deviation of delay (all events) in minutes -', round(std_delay, 2))
print('\tMaximum delay (all events) in minutes -', round(max_delay, 2))


print('\n\tTotal number of deployments -', bd_table.shape[0])
print('\tTotal number of route deployments -', sum(bd_table['Route']!=refuel))
print('\tTotal number of refills deployments -', sum(bd_table['Route']==refuel))

print('---\n\nCost Analysis')
print('\tAverage running time for all buses per day in minutes -', run_time)
print('\tAverage service time for all buses per day in minutes -', ser_time)
print('\tAverage refuel time for all buses per day in minutes -', ref_time)

print('\n\tAverage employeee costs paid per day in dollars -', emp_cost)
print('\tAverage fuel costs paid per day in dollars -', fuel_cost)
print('\tAverage delay costs paid per day in dollars -', delay_cost)
print('\tAverage maintenance costs paid per day in dollars -', round(maintain_rate*(run_time+ser_time+r
```

```
---
Current half-width -  0.936 min
Current replications - 30
Number of bus deployments - 4171
---
Delay Analyis
        Total number of replications (days) - 30
        Total number of delays - 1147
        Total delay in minutes - 4900.85
        Average of delay events in minutes - 4.27

        Average delay (all events) in minutes - 1.18
        Std. Deviation of delay (all events) in minutes - 2.51
        Maximum delay (all events) in minutes - 16.48

        Total number of deployments - 4171
        Total number of route deployments - 3870
        Total number of refills deployments - 301
---

Cost Analysis
        Average running time for all buses per day in minutes - 4143.51
        Average service time for all buses per day in minutes - 1504.54
        Average refuel time for all buses per day in minutes - 27.28

        Average employeee costs paid per day in dollars - 1419
        Average fuel costs paid per day in dollars - 722
        Average delay costs paid per day in dollars - 11762
        Average maintenance costs paid per day in dollars - 9478
```

In [7]:

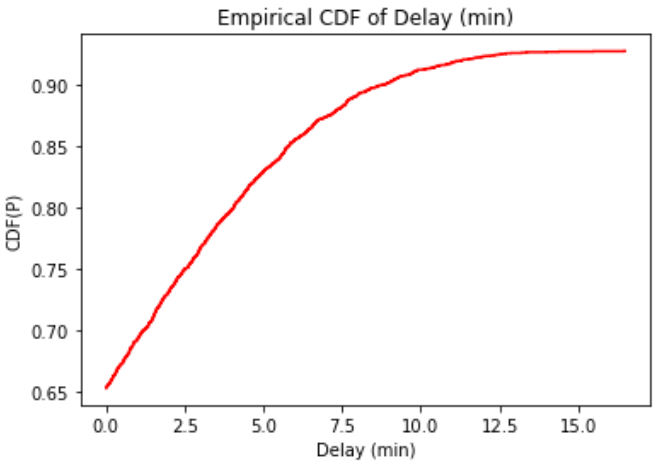
```
# Charge distribution
ss_table['Charge'].min(), ss_table['Charge'].max()
```

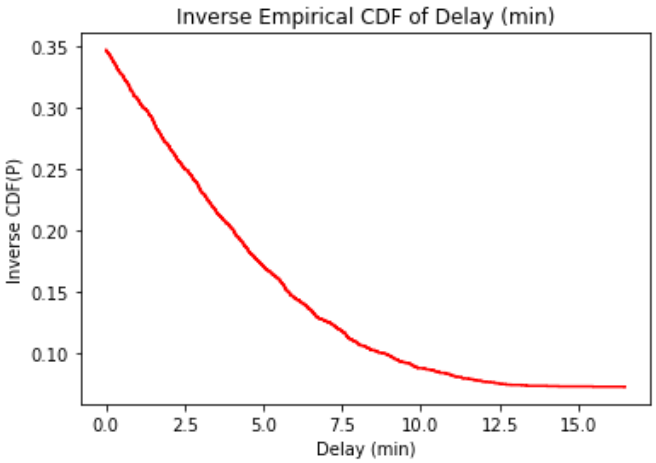
Out[7]:

```
(12.286, 87.66)
```

In [8]:

```
delay_arr = np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual'])
ecdf(delay_arr, 'Delay (min)')
inv_ecdf(delay_arr, 'Delay (min)')
```





In [9]:

```
ss_table.to_parquet(path+'ss_table_'+refuel+'_'+np.str(n_buses)+'_'+
                    np.str(level_mean)+'_'+np.str(running_consumption)+'_'+
                    np.str(service_consumption)+'.parquet')
bd_table.to_parquet(path+'bd_table_'+refuel+'_'+np.str(n_buses)+'_'+
                    np.str(level_mean)+'_'+np.str(running_consumption)+'_'+
                    np.str(service_consumption)+'.parquet')
```

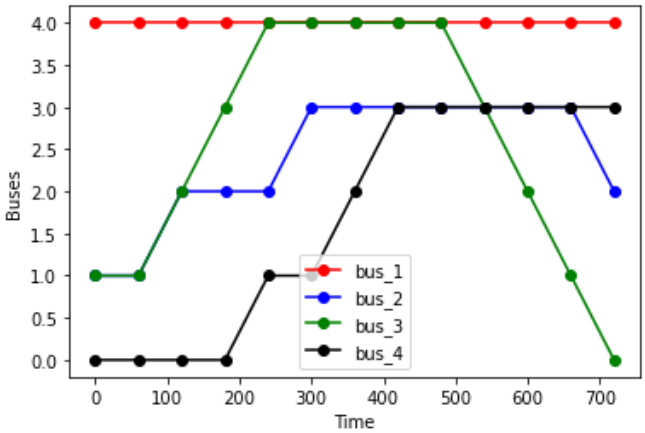
Testing Area

Testing for route demands

In [10]:

```
# route demand functions
array_t = np.array(range(0, 12*60+1, 60))

# https://ltp.umich.edu/wp-content/uploads/bursley_baits.pdf
a, b, c, d = 0, 4, 0, 1
plt.plot(array_t, np.ceil(a*np.sin((array_t+c)/d) + b),
         color='r', label='bus_1', marker='o')

# https://ltp.umich.edu/wp-content/uploads/northwood.pdf
a, b, c, d = 1.5, 1.5, -180, 180
plt.plot(array_t, np.ceil(a*np.sin((array_t+c)/d) + b),
         color='b', label='bus_2', marker='o')

# https://ltp.umich.edu/wp-content/uploads/diag_diag.pdf
a, b, c, d = 3, 1, -60, 180
plt.plot(array_t, np.ceil(a*np.sin((array_t+c)/d) + b),
         color='g', label='bus_3', marker='o')

# https://ltp.umich.edu/wp-content/uploads/oxford_shuttle.pdf
a, b, c, d = 2, 1, -320, 180
plt.plot(array_t, np.ceil(a*np.sin((array_t+c)/d) + b),
         color='black', label='bus_4', marker='o')
```

```
plt.xlabel('Time')
plt.ylabel('Buses')
plt.legend()
plt.show()
```



In [ ]:

```
import pandas as pd
import numpy as  np
import string as str
import numpy.random as rand
import matplotlib.pyplot as plt
import scipy.stats as st
import statsmodels.api as sm
```

In [ ]:

```
def route_time(routes, route_c, t):
    time_ar = []
    if route_c.isnumeric():
        search_r = 'route_' + route_c
    else:
        search_r = route_c
    sm_data = routes[routes.columns[routes.columns.str.match(search_r)]]
    sm_data.dropna(inplace=True)
    for i in range(sm_data.shape[0]):
        if sm_data[search_r+'_index'][i]==0:
            t_arr = rand.gamma(sm_data[search_r+'_mean'][i],
                               sm_data[search_r+'_std'][i])         # change distributions
            t = t + t_arr
        elif sm_data[search_r+'_index'][i]==1:
            t_ser = rand.gamma(sm_data[search_r+'_mean'][i],
                               sm_data[search_r+'_std'][i])         # change distributions
            t = t + t_ser
        time_ar.append(round(t, 2))
    stops = np.array(sm_data[search_r+'_index'])
    return list(time_ar), list(stops)
```

In [ ]:

```
class bus:
    def __init__(self, charge, charge_std):            # Class initialization
        self.charge = round(rand.normal(charge, charge_std), 3)
        self.state = -1                                # deployed = 1, refill = 0, standstill = -1
        self.route = None                              # route in string, eg. '1', 'refill', 'recharge'
        self.time_arr = list()                         # array containing travel or stop service times
        self.event_arr = list()                        # array to denote travel or stop service state

    def assign_route(self, routes, route_c, t):        # assigning a specific route to the bus
        self.time_arr, self.event_arr = route_time(routes, route_c, t)
        self.route = route_c
        if (route_c=='refill')or(route_c=='recharge'): # deployed = 1, refill = 0, standstill = -1
            self.state = 0
        elif route_c.isnumeric():
            self.state = 1

    def assign_route_varred(self, e_array, t_array, route_c, t):  # assigning a specific route to the b
        self.time_arr, self.event_arr = list(t_array), list(e_array)
        self.route = route_c
        if (route_c=='refill')or(route_c=='recharge'):  # deployed = 1, refill = 0, standstill = -1
            self.state = 0
        elif route_c.isnumeric():
            self.state = 1

    def next_t(self):                      # passing the next event time
        if len(self.time_arr)==0:
            return np.inf
        else:
            return self.time_arr[0]

    def next_e(self):                      # passing the next event type
        if len(self.event_arr)==0:
            return np.inf
        else:
            return self.event_arr[0]

    def last_t(self):                      # passing the last event time for a route
        if len(self.event_arr)==0:
            return np.inf
        else:
```

```
            return self.time_arr[len(self.event_arr)-1]

    def info(self):                         # all details passed as a dictionary for table insertion
        bus_dict = {'charge': self.charge,
                    'state' : self.state,
                    'route' : self.route,
                    'event' : self.event_arr[0]}
        return bus_dict
```

In [ ]:

```
def gen_demands(routes, T):
    dem = pd.read_excel('Model_Parameters.xlsx', 'Demands')
    t_arr = np.array(range(0, T, 60))
    demand_r, demand_t, demand_c = [], [], []
    for row in dem.itertuples(index=False):
        if row[0]<=routes:
        # generating demands
            route, a, b, c, d, charge = np.str_(row[0]), row[1], row[2], row[3], row[4], row[5]
            demand = np.ceil(a*np.sin((t_arr+c)/d) + b)        # a distribution can also be used
        # generating times wrt demands
            for i, t in enumerate(t_arr):
                if demand[i]>0:
                    for t_d in range(0, 60, int(60/demand[i])):
                        demand_t.append(round(t + t_d, 2))
                        demand_r.append(route)
                        demand_c.append(charge)

    demand_r.append(None)
    demand_c.append(np.inf)
    demand_t.append(np.inf)
    all_dt = pd.DataFrame(columns=['routes', 'times', 'charge'])
    all_dt['routes'] = demand_r
    all_dt['times'] = demand_t
    all_dt['charge'] = demand_c
    all_dt = all_dt.sort_values(['times','routes'], ignore_index=True)
    demand_r = np.array(all_dt['routes'])
    demand_c = np.array(all_dt['charge'])
    demand_t = np.array(all_dt['times'])
    return list(demand_t), list(demand_r), list(demand_c)
```

In [ ]:

```
def next_bus_e(buses):
    min_t = np.inf
    index = None
    for i in range(len(buses)):
        if (buses[i].next_t()<min_t):
            min_t = buses[i].next_t()
            index = i
    return min_t, buses[i].next_e(), index

def available_bus(buses, dem_charge):
    b_charges = [buses[i].charge for i in range(len(buses)) if buses[i].state==-1]
    b_index = [i for i in range(len(buses)) if buses[i].state==-1]
    index = -1
    if len(b_charges)>0:
        if (max(b_charges)>dem_charge):
            for i in range(len(b_charges)):
                if max(b_charges)==b_charges[i]:
                    index = b_index[i]
    return index

def unavailable_bus(buses, min_charge):
    b_charges = [buses[i].charge for i in range(len(buses)) if buses[i].state==-1]
    b_index = [i for i in range(len(buses)) if buses[i].state==-1]
    index = -1
    if (len(b_charges)>0)and(min_charge!=np.inf):
        if (min(b_charges)<min_charge):
            for i in range(len(b_charges)):
                if min(b_charges)==b_charges[i]:
                    index = b_index[i]
    return index

def buses_status(buses):
```

```
    n_dep = sum([1 for i in buses if i.state==1])
    n_ref = sum([1 for i in buses if i.state==0])
    n_stds = sum([1 for i in buses if i.state==-1])
    return n_dep, n_ref, n_stds
```

In [ ]:

```
def SS_update(SS_table, t, buses, dct, bus_e, t_updt, index=-1, dem_ct=0, dem_at=0, dem_c=0):
    ss_d = {}
    ss_d['Time'] = t_updt
    ss_arr = [len(buses), dct]
    ss_dep, ss_re, ss_ss = buses_status(buses)
    ss_arr.append(ss_dep)
    ss_arr.append(ss_re)
    ss_arr.append(ss_ss)
    ss_d['System_State'] = ss_arr
    if index!=-1:
        ss_d['Bus'] = index + 1
        ss_d['Charge'] = buses[index].charge
        ss_d['Route'] = buses[index].route
        ss_d['State'] = buses[index].state
    else:
        ss_d['Bus'] = np.nan
        ss_d['Charge'] = np.nan
        ss_d['Route'] = None
        ss_d['State'] = np.nan
    ss_d['Event'] = bus_e
    ss_d['Process_Time'] = t_updt - t
    if dem_c==0:
        ss_d['Demand_Current'] = np.nan
        ss_d['Demand_Actual'] = np.nan
        ss_d['Demand_Charge'] = np.nan
    else:
        ss_d['Demand_Current'] = dem_ct
        ss_d['Demand_Actual'] = dem_at
        ss_d['Demand_Charge'] = dem_c
    SS_table = SS_table.append(ss_d, ignore_index=True)
    return SS_table

def BD_update(BD_table, t_updt, dem_ct, dem_at, dem_c, buses, index):
    bd_d = {}
    bd_d['Time'] = t_updt
    bd_d['Demand_Current'] = dem_ct
    bd_d['Demand_Actual'] = dem_at
    bd_d['Demand_Charge'] = dem_c
    bd_d['Bus'] = index + 1
    bd_d['Charge'] = buses[index].charge
    bd_d['Route'] = buses[index].route
    bd_d['State'] = buses[index].state
    bd_d['Event_Array'] = np.array(buses[index].event_arr)
    bd_d['Time_Array'] = np.array(buses[index].time_arr)
    BD_table = BD_table.append(bd_d, ignore_index=True)
    return BD_table
```

In [ ]:

```
def fleet_simulation(t, T, routes, buses, refuel, running_consumption, service_consumption,
                     refuel_stations, refuel_consumption, conversion_factor,
                     demand_at, demand_ct, demand_r, demand_c, dct_flag, SS_cols, BD_cols):
    ss_table = pd.DataFrame(columns=SS_cols)
    bd_table = pd.DataFrame(columns=BD_cols)

    # initial SS update
    time_check = np.inf
    ss_table = SS_update(ss_table, t, buses, dct_flag, np.nan, np.nan)

    # previous times for fleet
    prev_time = [np.nan]*len(buses)


    t_check = []
    # Simulate! Simulate! Simulate!
    while (t<T)or(time_check!=np.inf):
        #print('---\nNew Event')
        time_check = min(next_bus_e(buses)[0], demand_ct[0])
```

```
    # -----
    # Priority One: Updating demands
    if ((demand_ct[0]!=np.inf) and
        (demand_ct[0]==time_check)):
        new_demand = [1 for i in range(len(demand_ct)) if demand_ct[i]==time_check]
        dct_flag = sum(new_demand)
        # testing
        t = time_check

    #----------
    # Main Switch Statements
    bus_chk = available_bus(buses, demand_c[0])

    #-----
    # Case 1: Sending low-fuel buses to refuel
    refuel_index = unavailable_bus(buses, min(demand_c))
    if ((time_check!=np.inf) and ((t<T) or (dct_flag>0)) and
        (refuel_index!=-1) and
        (buses_status(buses)[1]<refuel_stations)):
        # and(refuel_index!=np.inf)):- might create complications
        dem_ct, dem_at, dem_c = np.nan, np.nan, np.nan
        #print('\tSending Bus for Refuel')
        msg = 'Sending Bus for Refuel'
        buses[refuel_index].assign_route(routes, refuel, t)
        prev_time[refuel_index] = t
        bus_e = 0
        t_updt = t

        ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt, refuel_index)
        bd_table = BD_update(bd_table, t_updt, dem_ct, dem_at, dem_c, buses, refuel_index)

        t = t_updt
        refuel_index = -1


    #-----
    # Case 2: Checking bus availability and deploying buses
    elif ((t<T) and
          (demand_ct[0]==time_check) and
          (demand_ct[0]!=np.inf) and
          (bus_chk!=-1)):

        #print('\tDeploying Bus')
        msg = 'Deploying Bus'
        dem_ct, dem_at = demand_ct.pop(0), demand_at.pop(0)
        dem_c, dem_r = demand_c.pop(0), demand_r.pop(0)
        index = available_bus(buses, dem_c)
        t_updt = dem_ct
        buses[index].assign_route(routes, dem_r, t_updt)
        prev_time[index] = t_updt
        bus_e = 0

        ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt, index, dem_ct, dem_at, de
        bd_table = BD_update(bd_table, t_updt, dem_ct, dem_at, dem_c, buses, index)
        dct_flag -= 1
        t = t_updt

        dem_ct, dem_at, dem_c, dem_r = np.nan, np.nan, np.nan, np.nan
        t_updt = np.nan
        index = np.nan


    #-----
    # Case 3: Checking next bus event and updating SS
    elif (((t<T) and
          (next_bus_e(buses)[0]==time_check) and
          (next_bus_e(buses)[0]!=np.inf) and
          ((buses_status(buses)[0]>0) or (buses_status(buses)[1]>0))) or
          (((t<T) and
          (demand_ct[0]==time_check) and
          (demand_ct[0]!=np.inf) and
          (bus_chk==-1)))):
```

```
        #print('\tUpdating next bus event')
        msg = 'Updating next bus event'
        index = next_bus_e(buses)[2]
        t_updt = buses[index].time_arr.pop(0)
        bus_e = buses[index].event_arr.pop(0)
        diff_t = prev_time[index]

        if (bus_e==1):
            mul_c = running_consumption
        elif (bus_e==0)and(buses[index].state==1):
            mul_c = service_consumption
        elif (bus_e==0)and(buses[index].state==0):
            mul_c = refuel_consumption
        buses[index].charge = round(buses[index].charge - (t_updt - diff_t)*mul_c*conversion_factor

        ss_table = SS_update(ss_table, diff_t, buses, dct_flag, bus_e, t_updt, index)
        t = t_updt
        prev_time[index] = t

        # if its the last event for the bus, update bus parameters to standstill
        if buses[index].next_e()==np.inf:
            buses[index].state = -1
            buses[index].route = None
            buses[index].time_arr = list()
            buses[index].event_arr = list()
            prev_time[index] = np.nan
        index = np.nan
        t_updt = np.nan
        bus_e = np.nan


    #-----
    # Case 4: All demands are completed within the timeframe
    elif ((t<T) and
          (time_check==np.inf)):
        #print('\tJumping to EOD')
        msg = 'Jumping to EOD'
        t_updt = T
        ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt)
        t = t_updt


    #-----
    # Case 5: Demands still exist beyond timeframe and need to be met
    elif ((t>=T) and
          (demand_ct[0]==time_check) and
          (demand_ct[0]!=np.inf) and
          (bus_chk!=-1)):
        #print('\tDeploying Bus beyond timeframe')
        msg = 'Deploying Bus beyond timeframe'
        dem_ct, dem_at = demand_ct.pop(0), demand_at.pop(0)
        dem_c, dem_r = demand_c.pop(0), demand_r.pop(0)
        index = available_bus(buses, dem_c)
        t_updt = dem_ct
        buses[index].assign_route(routes, dem_r, t_updt)
        prev_time[index] = t_updt
        bus_e = 0
        ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt, index, dem_ct, dem_at, de
        bd_table = BD_update(bd_table, t_updt, dem_ct, dem_at, dem_c, buses, index)
        dct_flag -= 1
        t = t_updt
        dem_ct, dem_at, dem_c, dem_r = np.nan, np.nan, np.nan, np.nan
        t_updt = np.nan
        index = np.nan


    #-----
    # Case 6: Going beyond timeframe, checking deployed buses and updating SS
    elif (((t>=T) and
          (next_bus_e(buses)[0]==time_check) and
          (next_bus_e(buses)[0]!=np.inf)) or
          ((t>=T) and
          (demand_ct[0]==time_check) and
          (demand_ct[0]!=np.inf) and
```

```
              (bus_chk==-1))):
            #print('\tUpdating next bus event beyond timeframe')
            msg = 'Updating next bus event beyond timeframe'
            index = next_bus_e(buses)[2]
            t_updt = buses[index].time_arr.pop(0)
            bus_e = buses[index].event_arr.pop(0)
            diff_t = prev_time[index]
            if (bus_e==1):
                mul_c = running_consumption
            elif (bus_e==0)and(buses[index].state==1):
                mul_c = service_consumption
            elif (bus_e==0)and(buses[index].state==0):                    # different for 'refill' a
                mul_c = refuel_consumption                               # difference from 90 for r
            buses[index].charge = buses[index].charge - (t_updt - diff_t)*mul_c*conversion_factor
            ss_table = SS_update(ss_table, diff_t, buses, dct_flag, bus_e, t_updt, index)
            t = t_updt
            prev_time[index] = t

            # if its the last event for the bus, update bus parameters to standstill
            if buses[index].next_e()==np.inf:
                buses[index].state = -1
                buses[index].route = None
                buses[index].time_arr = list()
                buses[index].event_arr = list()
                prev_time[index] = np.nan
            index = np.nan
            t_updt = np.nan
            bus_e = np.nan


        #-----
        # Case 7: All demands are completed outside the timeframe
        elif ((t>=T) and
              (time_check==np.inf)):
            #print('\tJumping to EOD')
            msg = 'Jumping to EOD'
            t_updt = np.inf
            ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt)
            t = t_updt

        # updating the current demand - if demand exists it'll automatically be updated
        demand_ct = list(np.sort(demand_ct))
        for k in range(dct_flag):
            demand_ct[k] = t

        # code to check if the program is running in loops
        if len(t_check)<5:
            t_check.append(t)
        else:
            t_check.pop(0)
            t_check.append(t)
        if (len(t_check)==5)and(t_check[0]==max(t_check)):
            print('Stuck in a loop!')
            print(msg)
            ss_table.to_parquet('ss_check.parquet')
            bd_table.to_parquet('bd_check.parquet')
            print('\tTime -', t)
            print('\tTime Check -', time_check)
            print('\tDemands -', demand_ct[:6], demand_at[:6], demand_r[:6], demand_c[:6])
            print('\tdct_flag -', dct_flag)

    return ss_table, bd_table
```

In [ ]:

```
def fleet_simulation_varred(t, T, routes, event_array, time_array, init_time, buses, refuel, running_co
                    service_consumption, refuel_stations, refuel_consumption, conversion_factor,
                    demand_at, demand_ct, demand_r, demand_c, dct_flag, SS_cols, BD_cols):

    ss_table = pd.DataFrame(columns=SS_cols)
    bd_table = pd.DataFrame(columns=BD_cols)

    # Initial SS update
    time_check = np.inf
```

```
ss_table = SS_update(ss_table, t, buses, dct_flag, np.nan, np.nan)

# previous times for fleet
prev_time = [np.nan]*len(buses)

# Simulate! Simulate! Simulate!
while (t<T)or(time_check!=np.inf):
    #print('---\nNew Event')
    time_check = min(next_bus_e(buses)[0], demand_ct[0])

    # -----
    # Priority One: Updating demands
    if ((demand_ct[0]!=np.inf) and
        (demand_ct[0]==time_check)):
        new_demand = [1 for i in range(len(demand_ct)) if demand_ct[i]==time_check]
        dct_flag = sum(new_demand)
        t = time_check

    #----------
    # Main Simulation
    bus_chk = available_bus(buses, demand_c[0])

    #-----
    # Case 1: Sending low-fuel buses to refuel
    refuel_index = unavailable_bus(buses, min(demand_c))
    if (((t<T) or(dct_flag>0)) and  # or (time_check!=np.inf)
        (refuel_index!=-1) and
        (buses_status(buses)[1]<refuel_stations)):
        dem_ct, dem_at, dem_c = np.nan, np.nan, np.nan
        #print('\tSending Bus for Refuel')
        buses[refuel_index].assign_route(routes, refuel, t)
        prev_time[refuel_index] = t
        bus_e = 0
        t_updt = t

        ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt, refuel_index)
        bd_table = BD_update(bd_table, t_updt, dem_ct, dem_at, dem_c, buses, refuel_index)

        t = t_updt
        refuel_index = -1


    #-----
    # Case 2: Checking bus availability and deploying buses
    elif ((t<T) and
          (demand_ct[0]==time_check) and
          (demand_ct[0]!=np.inf) and
          (bus_chk!=-1)):

        #print('\tDeploying Bus')
        dem_ct, dem_at = demand_ct.pop(0), demand_at.pop(0)
        dem_c, dem_r = demand_c.pop(0), demand_r.pop(0)
        index = available_bus(buses, dem_c)
        t_updt = dem_ct
        be_arr = event_array.pop(0)
        bt_arr = np.array(time_array.pop(0))
        bit_arr = dem_ct - np.array(init_time.pop(0))
        buses[index].assign_route_varred(be_arr, bt_arr+bit_arr, dem_r, t_updt)
        prev_time[index] = t_updt
        bus_e = 0

        ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt, index, dem_ct, dem_at, de
        bd_table = BD_update(bd_table, t_updt, dem_ct, dem_at, dem_c, buses, index)
        dct_flag -= 1
        t = t_updt

        dem_ct, dem_at, dem_c, dem_r = np.nan, np.nan, np.nan, np.nan
        t_updt = np.nan
        index = np.nan


    #-----
    # Case 3: Checking next bus event and updating SS
    elif (((t<T) and
```

```
      (next_bus_e(buses)[0]==time_check) and
      (next_bus_e(buses)[0]!=np.inf) and
      ((buses_status(buses)[0]>0) or (buses_status(buses)[1]>0))) or
      (((t<T) and
      (demand_ct[0]==time_check) and
      (demand_ct[0]!=np.inf) and
      (bus_chk==-1)))):

    #print('\tUpdating next bus event')
    index = next_bus_e(buses)[2]
    t_updt = buses[index].time_arr.pop(0)
    bus_e = buses[index].event_arr.pop(0)
    diff_t = prev_time[index]

    if (bus_e==1):
        mul_c = running_consumption
    elif (bus_e==0)and(buses[index].state==1):
        mul_c = service_consumption
    elif (bus_e==0)and(buses[index].state==0):                    # different for 'refill' a
        mul_c = refuel_consumption                                # difference from 90 for r
    buses[index].charge = round(buses[index].charge - (t_updt - diff_t)*mul_c*conversion_factor

    ss_table = SS_update(ss_table, diff_t, buses, dct_flag, bus_e, t_updt, index)
    t = t_updt
    prev_time[index] = t

    # if its the last event for the bus, update bus parameters to standstill
    if buses[index].next_e()==np.inf:
        buses[index].state = -1
        buses[index].route = None
        buses[index].time_arr = list()
        buses[index].event_arr = list()
        prev_time[index] = np.nan
    index = np.nan
    t_updt = np.nan
    bus_e = np.nan


#-----
# Case 4: All demands are completed within the timeframe
elif ((t<T) and
      (time_check==np.inf)):

    #print('\tJumping to EOD')
    t_updt = T

    ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt)
    t = t_updt


#-----
# Case 5: Demands still exist beyond timeframe and need to be met
elif ((t>=T) and
      (demand_ct[0]==time_check) and
      (demand_ct[0]!=np.inf) and
      (bus_chk!=-1)):

    #print('\tDeploying Bus beyond timeframe')
    dem_ct, dem_at = demand_ct.pop(0), demand_at.pop(0)
    dem_c, dem_r = demand_c.pop(0), demand_r.pop(0)
    index = available_bus(buses, dem_c)
    t_updt = dem_ct
    be_arr = event_array.pop(0)
    bt_arr = np.array(time_array.pop(0))
    bit_arr = dem_ct - np.array(init_time.pop(0))
    buses[index].assign_route_varred(be_arr, bt_arr+bit_arr, dem_r, t_updt)
    prev_time[index] = t_updt
    bus_e = 0

    ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt, index, dem_ct, dem_at, de
    bd_table = BD_update(bd_table, t_updt, dem_ct, dem_at, dem_c, buses, index)
    dct_flag -= 1
    t = t_updt
```

```
            dem_ct, dem_at, dem_c, dem_r = np.nan, np.nan, np.nan, np.nan
            t_updt = np.nan
            index = np.nan


        #-----
        # Case 6: Going beyond timeframe, checking deployed buses and updating SS
        elif (((t>=T) and
               (next_bus_e(buses)[0]==time_check) and
               (next_bus_e(buses)[0]!=np.inf)) or
               ((t>=T) and
               (demand_ct[0]==time_check) and
               (demand_ct[0]!=np.inf) and
               (bus_chk==-1))):

            #print('\tUpdating next bus event beyond timeframe')
            index = next_bus_e(buses)[2]
            t_updt = buses[index].time_arr.pop(0)
            bus_e = buses[index].event_arr.pop(0)
            diff_t = prev_time[index]

            if (bus_e==1):
                mul_c = running_consumption
            elif (bus_e==0)and(buses[index].state==1):
                mul_c = service_consumption
            elif (bus_e==0)and(buses[index].state==0):                          # different for 'refill' a
                mul_c = refuel_consumption                                     # difference from 90 for r
            buses[index].charge = buses[index].charge - (t_updt - diff_t)*mul_c*conversion_factor

            ss_table = SS_update(ss_table, diff_t, buses, dct_flag, bus_e, t_updt, index)
            t = t_updt
            prev_time[index] = t

            # if its the last event for the bus, update bus parameters to standstill
            if buses[index].next_e()==np.inf:
                buses[index].state = -1
                buses[index].route = None
                buses[index].time_arr = list()
                buses[index].event_arr = list()
                prev_time[index] = np.nan
            index = np.nan
            t_updt = np.nan
            bus_e = np.nan

        #-----
        # Case 7: Jumping to next demand
        elif ((t>=T) and (bus_chk==-1) and
              (demand_ct[0]==time_check) and
              (demand_ct[0]!=np.inf)):
            #print('\tJumping to next demand')
            msg = 'Jumping to next demand'
            t = next_bus_e(buses)[0]
            #dct_flag += 1

        #-----
        # Case 8: All demands are completed outside the timeframe
        elif ((t>=T) and
              (time_check==np.inf)):

            #print('\tJumping to EOD')
            msg = 'Jumping to EOD'
            t_updt = np.inf
            ss_table = SS_update(ss_table, t, buses, dct_flag, bus_e, t_updt)
            t = t_updt

        # updating all current demand times
        demand_ct = list(np.sort(demand_ct))
        for k in range(dct_flag):
            demand_ct[k] = t

    return ss_table, bd_table
```

In [ ]:

```
def cost_analysis(n_buses, replicates, refuel, ss_table, bd_table, emp_rate, fuel_rate, delay_rate, ref
    ss_table['Process_Time'].replace({np.inf: 0}, inplace=True)
    ss_table['Demand_Charge'].replace({np.nan: -1}, inplace=True)
    cost_ss = ss_table[(ss_table['Demand_Charge'])==-1][['Bus', 'Route', 'Event', 'Process_Time']].drop

    runcost_ss = cost_ss[cost_ss['Event']==1]
    sercost_ss = cost_ss[(cost_ss['Event']!=1)&(cost_ss['Route']!=refuel)]
    refcost_ss = cost_ss[(cost_ss['Event']!=1)&(cost_ss['Route']==refuel)]
    runcost_ss = runcost_ss[['Bus', 'Process_Time']].groupby('Bus').sum()/replicates
    sercost_ss = sercost_ss[['Bus', 'Process_Time']].groupby('Bus').sum()/replicates
    refcost_ss = refcost_ss[['Bus', 'Process_Time']].groupby('Bus').sum()/replicates

    run_time = round(np.sum(runcost_ss['Process_Time']), 2)
    ser_time = round(np.sum(sercost_ss['Process_Time']), 2)
    ref_time = round(np.sum(refcost_ss['Process_Time']), 2)
    emp_cost = round((run_time + ser_time + ref_time)*emp_rate/60)
    fuel_cost = round(ref_time*refuel_consumption*-1*fuel_rate)
    tot_delay = round(sum(np.nan_to_num(np.array(bd_table['Demand_Current'] - bd_table['Demand_Actual']
    delay_cost = round(tot_delay* delay_rate)
    return run_time, ser_time, ref_time, emp_cost, fuel_cost, delay_cost
```

In [ ]:

```
def ecdf(target, title):
    numbs = np.array(target)
    ecdf = sm.distributions.ECDF(numbs)
    x = np.linspace(min(numbs), max(numbs), len(target))
    y = ecdf(x)
    plt.step(x, y, color='r')
    plt.xlabel(title)
    plt.ylabel('CDF(P)')
    plt.title('Empirical CDF of ' + title)
    plt.show()
    return

def inv_ecdf(target, title):
    numbs = np.array(target)
    ecdf = sm.distributions.ECDF(numbs)
    x = np.linspace(min(numbs), max(numbs), len(target))
    y = 1-ecdf(x)
    plt.step(x, y, color='r')
    plt.xlabel(title)
    plt.ylabel('Inverse CDF(P)')
    plt.title('Inverse Empirical CDF of ' + title)
    #plt.gca().invert_yaxis()
    plt.show()
    return
```