Numerical Methods in Issues of Sustainability

by

Shefali Ramakrishna

May 2022

Submitted to the Faculty of Bryn Mawr College
in partial fulfillment of the requirements for
the degree of Master of Arts in the
Department of Mathematics

Abstract

This thesis broadly focuses on mathematical modeling approaches to solving various issues in sustainability, broken up into two parts, each dealing with specific, but distinct, sustainability issues.

The first part of this thesis is expository, focusing on the modeling of groundwater flow and water table heights given limited information. We begin with the background of the subject, examining how groundwater flow works in the physical world (Section 1.1). We then delve into a derivation of Darcy's Law for two-dimensional flow of groundwater from base principles (Section 1.2). From there we derive the two-dimensional Laplace equation for flow (Section 1.5). We then develop a central difference approximation to solve for intermediate water table heights between two given points, and intermediate water table heights given boundary conditions by solving a system of linear equations (Section 1.6). We then examine two methods of solving this system of linear equations, and compare rates of convergence for both (Section 1.8). We also implement both of these methods into solvers in MATLAB (Appendix 4.1, 4.2, 4.3).

The second part of this thesis is case-based problem solving in a different area of mathematics. We use queuing theory and simulation principles to determine the number of electric car chargers needed to support an electric vehicle fleet on the Bryn Mawr campus, based on current usage numbers for Bryn Mawr vehicles. The model takes in the number and type of electric vehicles, along with their daily use. We then run a simulation to determine whether the current charger level can support the car use with minimal waiting in queues for the charger, depending on the number of available chargers.

Acknowledgements

To my professors at Bryn Mawr College and at Haverford College, thank you for being the best educators the world has to offer. Among my professors, I want to thank Professor Leslie Cheng, who has been an incredible mentor, helped cultivate my interest in analysis, and been such a great support over my time at Bryn Mawr, and Professor Robert Manning, who first introduced me to the fascinating fields of optimization and computational mathematics in MATHH210 and MATHH222. I would not have discovered my passions and gained confidence in my abilities without you.

To my research advisors and mentors over the years, thank you for opening my eyes to the world after college and helping me discover the various corners of the large world of mathematics. Thank you to Dr. Dave Benson, who helped me realize the joys of working in the field of operations research and opened my eyes to the potential of graduate study in the field, and to Professor Ralph Morrison, who was a fantastic research advisor over Summer 2021 and gave me a new appreciation for problems on graphs.

To my friends and family, thank you for putting up with me all this time. Thank you to my parents, Srinivas Ramakrishna and Neeraja Sivaramayya, for their ardent passion for mathematics, which they instilled in me, and for their support of my education in every way. Thank you to my brother, Sam, and pets, Coffee, Becky, Parker, and Bowie, for providing me with love and laughs. Thank you to Charlie, Sbaniel, Megan, and Stella, for being beacons of light and joy in my life.

And of course, thank you to Professor Victor Donnay (who counts as a Bryn Mawr professor *and* research advisor) for your guidance from Multivariable Calculus to Summer 2019 research to this thesis. It was through you I discovered the power of abstract mathematics in bringing about tangible environmental benefits, and I could not imagine where I would be in this thesis without your help. I have learned so much from working with you – you have shaped my interests and passions and I hope to be able to continue this work at the intersection of math and sustainability far into the future.

Contents

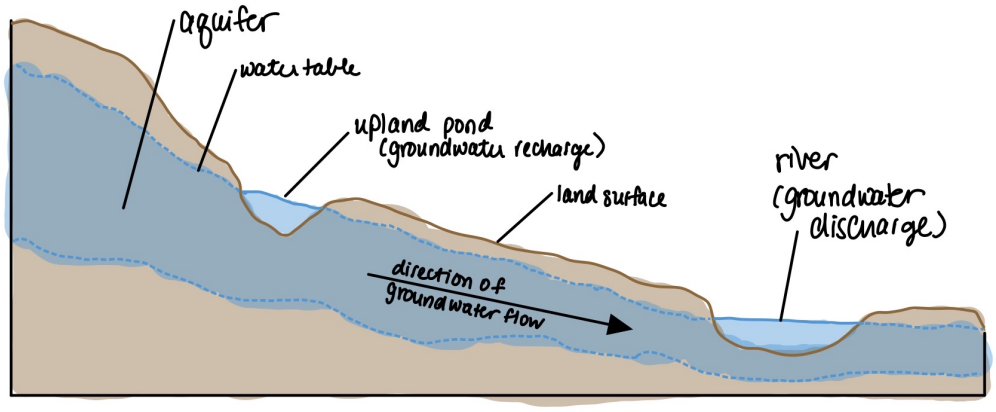

Figure 1: Illustration of Aquifer

1 Part I: Groundwater Flow

1.1 Background

In this section, we follow the presentation of Math Modeling in the EnvironmentHadlock [1998]. We shall begin with an explanation of the basic mechanics of groundwater flow and the motivation for research into this phenomenon.

Below the surface of the Earth lie several layers of soil and rock. The spaces between these subsurface soil particles are filled with groundwater. Groundwater flows through the spaces between soil particles, flowing in the direction of the steepest decline of the land. Groundwater does not flow quickly, but it is always flowing. Water on the surface of the Earth becomes groundwater either through precipitation seeping into the ground or through surface bodies of water seeping into the ground. These sources fill up open spaces in the subsoil up till a certain height. This height is called the water table. Because the underground cannot be directly observed, we need to make models to understand how groundwater is behaving given only a few data points. We can understand the direction and rate of flow of ground water, along with the locations at which contamination may have been added to the groundwater.

Surface bodies of water are in fact portions of the water table that extend above the land. Water below the water table is groundwater. The portion of the underground through which groundwater flows without obstruction is called the aquifer. The water table aquifer is the portion of the subsurface just below the water table.

In Figure 1, the upland pond serves as a source of water into the water table aquifer. We can alternatively refer to this as a source of groundwater recharge. This is not the only possible source for groundwater recharge. A large portion of precipitation on the surface of the earth also seeps into the ground, while the rest evaporates or runs directly into surface water bodies.

In Figure 1, the river serves as a discharge zone for groundwater, as the groundwater from the aquifer can return to the surface via the river by entering the stream bank below the river's surface.

This is where the large-scale impact of groundwater becomes clearer. While groundwater does not move quickly, it moves steadily, and pollution of groundwater has a cumulative effect. For example, if groundwater flows along a number of contaminated site, at the end of these sites, the groundwater will be highly contaminated, as all the contaminants will have entered the groundwater. This groundwater can enter discharge zones and will contaminate the surface water steadily.

Aquifers generally consist of two types of material – soil and bedrock. In order of decreasing particle size, some common materials making up soil include gravel, sand, silt, and clay. Bedrock can be made of a variety of types of solid rock. While groundwater can pass through spaces between soil particles, groundwater cannot do so in bedrock, but some bedrock, like sandstone, is porous, while other bedrock, like limestone, has interconnected fracture networks. Thus water can flow through both soil and bedrock.

These different materials can be classified based on how much they resist water flow. Aquicludes are materials which block water flow entirely. Aquitards are materials which significantly resist water flow. Some examples of aquitards for soil include clays and fine silts. Bedrock aquitards include salt as well as unfractured formations of shale or granite.

The water table most often follows the topography of the land, but there can also be distinct aquifers of different depths, flowing independently and separated from one another by aquicludes. Additionally, deeper aquifers may have flow patterns more complex than simply the topography of the surface due to the distance between recharge zones, the effect of fault zones, as well as pressure from the rock above.

We shall now discuss the motivation for study of groundwater flow, and quantitative questions that can be addressed using our study. Take the following scenario:

Consider the scenario illustrated in Figure 2, where the dotted lines show contours of constant water table heights. Suppose we have a "small leak" from a tank stored underground at the service station. If residents live close to the service station and rely on drinking groundwater from an area close to the station, we may be concerned about contamination of this water.

Suppose a leak occurs from corrosion in an gasoline storage tank. Then the surrounding soil would be contaminated, and so rainwater seeping through the soil, along with the natural forces of gravity would carry the contaminant down to the aquifer, from where the contaminant would be constantly carried in the direction of general groundwater movement (indicated by red arrows).

This leads to the following crucial questions: We want to understand how much

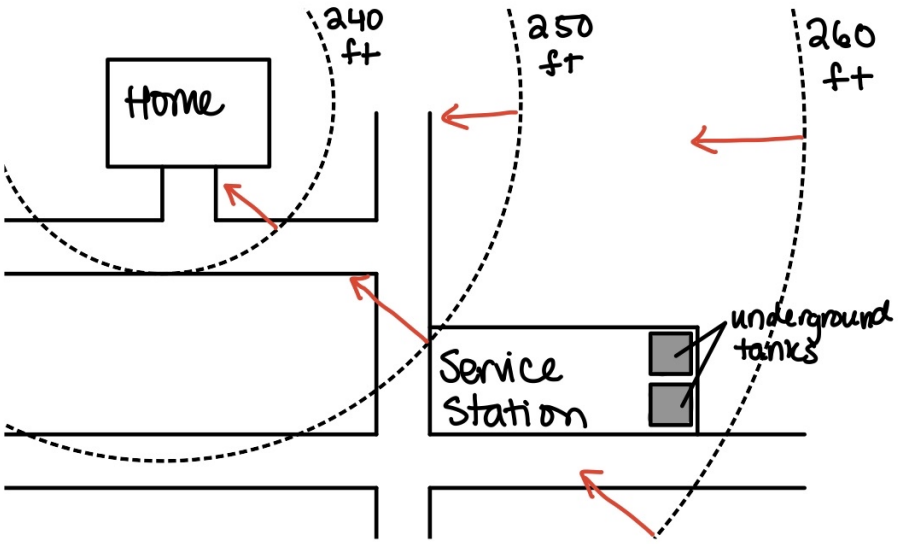

Figure 2: A Scenario of Contaminants Spread By Groundwater

gasoline leaked, for how long it has been leaking, as well as the direction and speed of groundwater flow.

A series of steps can be taken to understand and answer our questions, beginning with underground sampling using test and monitoring wells. These are holes sunk through one or more layers of subsurface soil, lined with pipe so that we may gather water from our point of interest. However, these wells are disruptive to the area and expensive. We want to use math so that we can answer our questions while using fewer wells.

The questions we are asking can be broken down into two broad questions:

I. How much groundwater is flowing through a portion of an aquifer? This is useful because if we can figure out the rate at which contaminant is seeping into the groundwater, we can estimate contaminant concentration in the groundwater. This question can be answered by Darcy's Law, which we will be deriving in Section 1.2.

II. What is the rate at which the groundwater is flowing? Answering this allows us to understand how far contamination might have spread since it was first introduced, as well as the amount of time left to keep it from spreading much farther. This question can be answered by the interstitial velocity equation, which will be derived in Section 1.2.

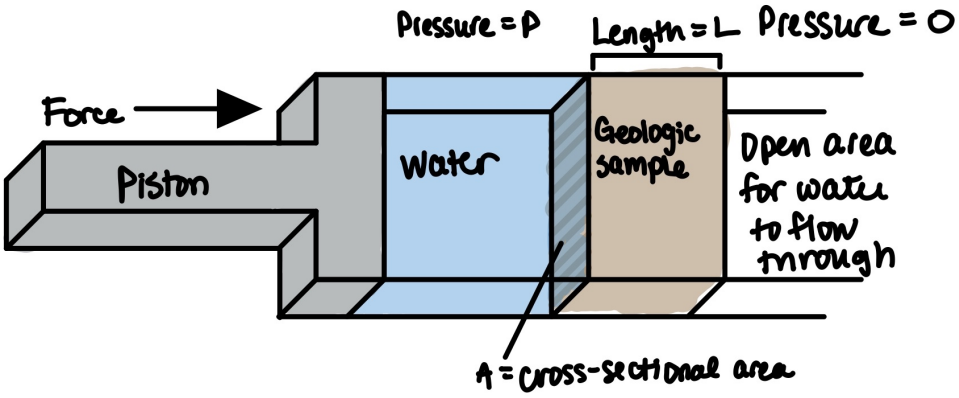

Figure 3: Baseline Experiment

1.2 Derivation of Darcy's Law and Interstitial Velocity Equation

We shall begin by deriving Darcy's Law from first principles. The intuition we shall use is backed up by Darcy's experiments. We wish to better understand how groundwater flows through different geologic mediums. We can begin with a "baseline experiment" of sorts in Figure 3 in which we understand flow rates through different size sand filters.

This baseline consists of a piston pushing water forward with pressure P through a sand filter of length L and cross-sectional area A. There is a pressure gradient across the sand filter, and on the other side of the sand filter, the pressure is 0, so the water can freely flow with rate Q through the sand filter.

First, let us try to understand how pressure might affect flow rate. It is clear through intuition that doubling pressure, P, should double the flow rate, Q, and so we see that $P \propto Q$.

Now let us say that we double the length of our geologic sample/sand filter. Then since the pressure uniformly decreases along the length of our sand filter, it follows that with the same pressure P and an doubled length of sand filter L, the flow rate Q will be halved. This means that $\frac{1}{L} \propto Q$, or that flow rate and sample length are inversely proportional.

Finally, let us consider the cross-sectional area of our sample/sand filter. If we double the area, A, keeping the pressure, P, the same, we can flow through twice as much water, Q, and so $A \propto Q$.

Now, to better relate this to groundwater flow specifically, let us alter our baseline experiment. Instead of using a piston as our source of pressure for water, we can use the weight of a column of water to drive water pressure, as seen in Figure 4.

The pressure caused by the column of water is due to the height difference in

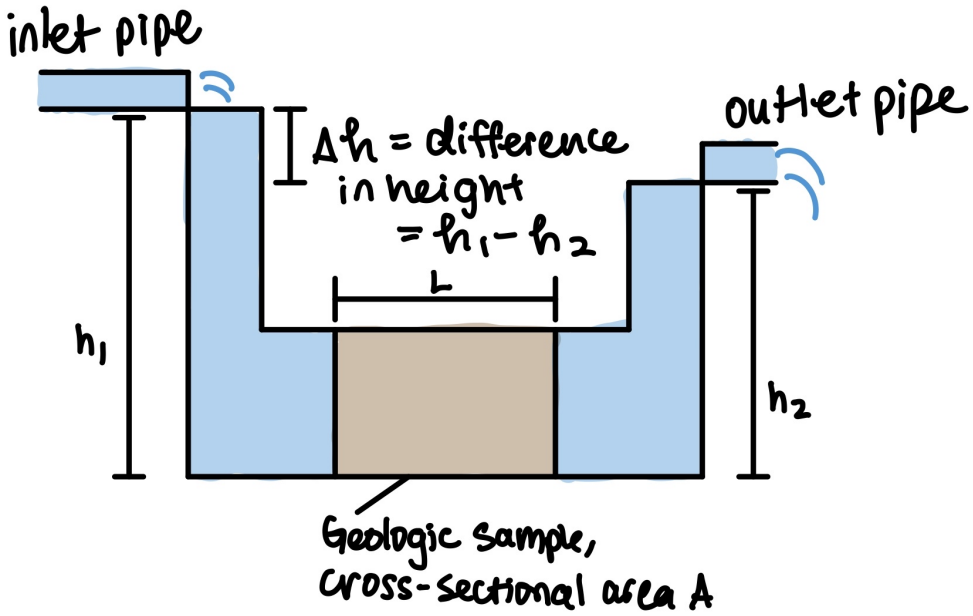

Figure 4: Darcy's Experiment for Groundwater Flow

the water table. Numerically, this is the difference in heights, or $\Delta h = h_1 - h_2$. If $h_1 = h_2$, then the water pressure is equal on both sides so no flow would occur, as the system would be at equilibrium. However, if $h_1 > h_2$, then pressure on the left side would be greater than the pressure on the right side, resulting in a pressure gradient. Then there would be flow from the left to the right until an equilibrium is reached.

To summarize, we have that:

1. The flow rate is proportional to the net driving pressure, Δh.
2. The flow rate is inversely proportional to the length L of our geologic sample.
3. The flow rate is proportional to the cross-sectional area A of our flow pathway.
4. From intuition, the constant of proportionality for the above relationships depends on the geologic medium through which the water is passing.

Combining these ideas, and replacing P with Δh, we get Darcy's Law, which states that

$$\frac{Q}{A} = \frac{k\Delta h}{L},$$

where k is our aforementioned constant of proportionality, depending on the geologic medium. Let's simplify this equation a bit, by separating flow rate Q on one side of the equation and combining some terms.

Let us have

$$i = \frac{\Delta h}{L},$$

and call i the hydraulic gradient, to represent the combined effect of the pressure difference on either side of the geologic medium and the length of the flow through that medium. If flow rate and pathway length are both doubled, the hydraulic gradient does not change.

Thus our finalized Darcy's Law reads

$$Q = kiA. \tag{1}$$

Now we shall derive the Interstitial Velocity Equation, which tells us the rate at which groundwater is flowing through a medium. In order to derive this, let us begin by assuming that water is flowing through an aquifer according to the principle of Darcy's Law.

We shall then focus on a single square foot of this aquifer's cross section, and assume this channel is an "open channel," meaning it provides no resistance to the flow of water whatsoever. Let's now assume that we know that 10 cubic feet of water is flowing out of the end of this cross section of the channel per minute. We can try to derive the flow rate of water in order for this velocity to be possible. Recall that we are looking at a cross-sectional area of 1 square foot. Thus

$$\frac{\text{velocity of water ft}}{\text{min}} \cdot 1 \text{ ft}^2 \text{ area} = \frac{10\text{ft}^3 \text{ water exiting channel}}{\text{min}}$$
$$\frac{\text{velocity of water ft}^3}{\text{min}} = \frac{10\text{ft}^3 \text{ water exiting channel}}{\text{min}}$$

Thus we find that the water must be flowing at a rate of 10 feet per minute. Another way of conceptualizing this problem is by viewing it as a volume problem – in order for x cubic feet of water to come out from a square foot area per minute, what length must the water have traveled in a minute? In mathematical terms, this is

$$\text{volume} = \text{area} \cdot \text{length},$$

where length is synonymous with velocity in this case, as seen in Figure 5.

One thing to note about this concept is that this relationship holds regardless of porosity of the geologic medium. With a porous material, it is simply the case that the path through which the water flows will not be straight, but the net fluid flow velocity in the flow direction will be the same as though a portion had been entirely closed off instead, as illustrated in Figure 6. Thus the porosity, η, represents the percentage of the geologic medium through which water can flow.

This net velocity along the axis of predominant flow is called the interstitial velocity, or the velocity of groundwater. This answers our question of how far a dissolved contaminant has traveled from the source over a period of time.

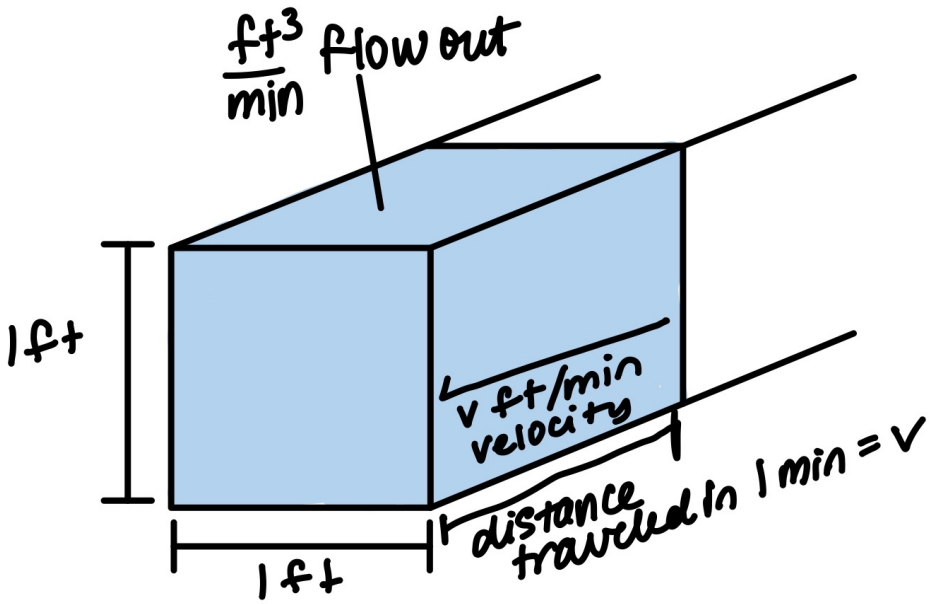

Figure 5: Illustration of Volume and Velocity Relationship

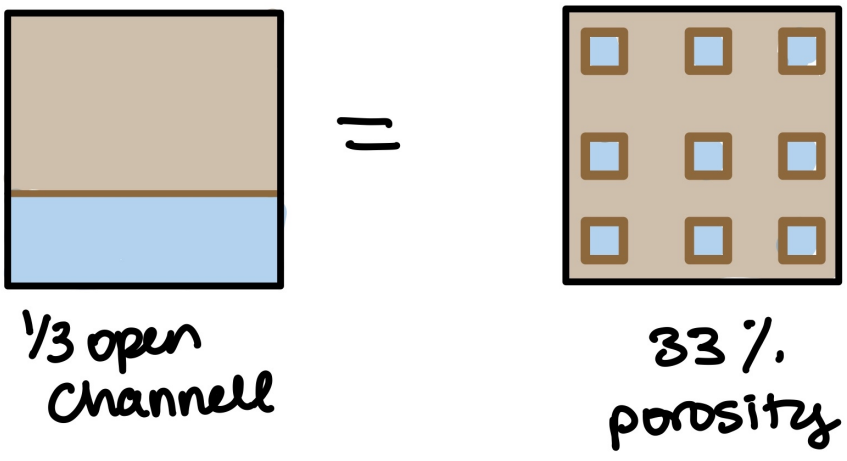

Figure 6: Porous flow vs. Closed-off channel

In order to determine fluid velocity in an aquifer, we need the volumetric fluid flow through a unit cross-sectional analysis, then divide this by the porosity of the aquifer.

Let q be the volumetric fluid flow through a unit cross-sectional area of an aquifer (we can do this by using Darcy's Law and setting $A = 1$). Then we have for Darcy's Law that

$$Q = kiA,$$

and so

$$q = ki(1) = ki.$$

Therefore we have for our velocity v that

$$\begin{aligned} v &= \frac{q}{\eta} \\ \Rightarrow v &= \frac{ki}{\eta}, \end{aligned}$$

the Interstitial Velocity Equation.

It follows that after calculating velocity v, we can find the amount of time t it takes the groundwater to travel a certain distance d in direction of flow, since

$$v = \frac{d}{t} \Rightarrow t = \frac{d}{v}.$$

It is important to note that certain contaminants do degrade over time or enter into chemical reactions with the surrounding rock or soil matrix, reducing their mobility. However contaminant flow is still a ticking time bomb that can travel for long periods of time without detection. When detected, the extent of the problem and amount of contaminant may still be huge and result in costly or even infeasible remedial action.

Now that we have derived equations to answer our most pressing mathematical questions about groundwater flow, we shall review the parameters in each of these equations.

Recall that Darcy's Law states that

$$Q = \frac{kA\Delta h}{L} = kiA,$$

and that the Interstitial Velocity Equation states that

$$v = \frac{q}{\eta} = \frac{ki}{\eta}.$$

The parameters for each of these equations, as well as their units, are stated below.

k: hydraulic conductivity (length/time)
i: hydraulic gradient (no units, the "slope" of water flow)
A: cross-sectional area of flow pathway, or the portion of aquifer under consideration (area, length^2)
η: porosity (no units, represents the fraction of open space in aquifer material)
Q: volumetric flow rate (volume/time, length^3/time)
q: flux, the volumetric flow rate through a cross-sectional area of one unit (length-/time)
v: velocity of fluid (length/time)
Δh: hydraulic head, the height of the water level above a given reference point (length)
L: length of flow pathway under consideration (length)

There are a few remarks to be made about these parameters. First, we notice that the hydraulic conductivity k, the flux q, and the velocity v of our fluid are all in the same units – length/time. However, they all represent vastly different ideas. The hydraulic conductivity is a constant of proportionality that depends on the geologic medium through which groundwater is flowing – it broadly represents how easily water flows through, and its units are length/time to correspond with the units for Darcy's Law. On the other hand, the flux is the volumetric flow rate through a unit cross sectional area, so can be thought of as length^3/time $\cdot$ 1/area, which equals length^3/time $\cdot$ 1/length^2, which in turn equals length/time after cancelling out like units. Finally, velocity is distance traveled per unit time, and so its units are length/time.

The second remark is that porosity and hydraulic conductivity are not related. A more porous material may still be an aquitard or aquiclude. For instance, clay is porous but its pores are far too small for water to pass through.

1.3 Construction of Contour Maps from Test Wells

We can use test wells to determine the depth of the water table at certain points, and from these head values can construct a contour map of the water table.

This contour map, like any other, shows us the locations of constant head values. It is particularly useful to know the the topography of the land, because the water table heights generally mirror the slope of the land above. Furthermore, groundwater usually flows in the direction of steepest descent – the gradient. Thus groundwater will flow perpendicular to the head contour line at that point. The reason that this is not always true of groundwater is because flow can still be deflected by the shape of fractures in bedrock, or by the orientation of pore spaces, but we shall ignore this case in the problems we study.

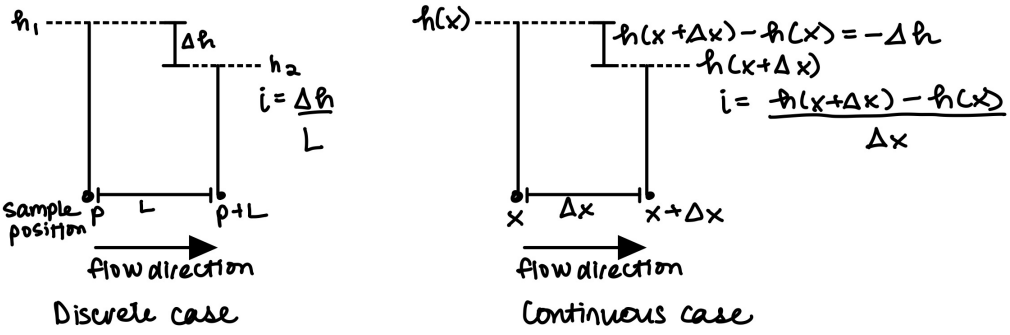

Figure 7: Continuous vs. Discrete i

1.4 Continuous form of Darcy's Law for One-Dimensional Flow

Now we shall use our work in prior sections to derive the continuous version of Darcy's Law. Recall Darcy's Law, which states that

$$Q = kiA \text{ and } v = \frac{ki}{\eta}, \text{ with}$$

$$i = \frac{\Delta h}{L}.$$

We had called the hydraulic gradient i the "slope" of water flow, and will now treat it as a derivative of the hydraulic head function, a continuous function taking a point in space to its hydraulic head height.

For this section, we are treating flows as being sufficiently horizontal. In other words, we shall assume that the head value does not depend on the depth of the point at which we sample the aquifer – it only depends on where we are on the axis of horizontal flow.

We will now start considering Δh to be the loss in head rather than change in head, as seen in Figure 7. So, to get the hydraulic gradient i at a specific point x, we have

$$\begin{aligned} i &= \lim_{\Delta x \to 0} \frac{-[h(x+\Delta x) - h(x)]}{\Delta x} \\ &= -\lim_{\Delta x \to 0} \frac{h(x+\Delta x) - h(x)}{\Delta x} \\ \Rightarrow i &= -\frac{dh}{dx}. \end{aligned} \tag{2}$$

Therefore we find that

$$Q = kiA = -k\frac{dh}{dx}A.$$

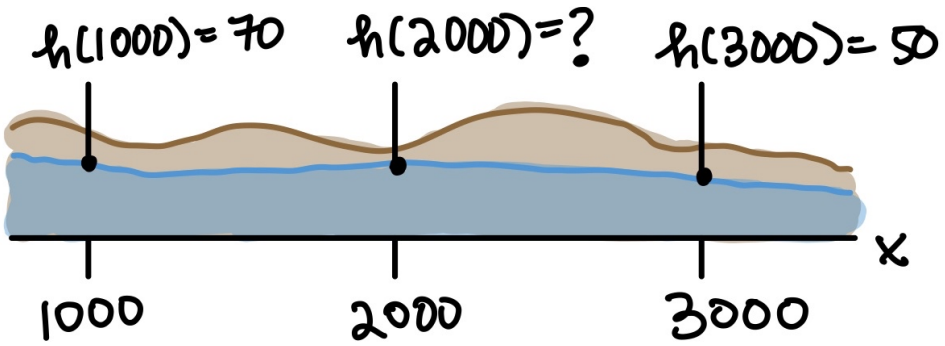

Figure 8: Example One-Dimensional Flow Problem

It then follows that

$$v = \frac{q}{\eta} = \frac{-k\frac{dh}{dx}}{\eta}.$$

Now let us look at an application of the continuous form of Darcy's Law to clearly understand implications of this equation. Suppose we have the following head values for points at $x = 1000$ and $x = 3000$, where $h(1000) = 70$ and $h(3000) = 50$, as illustrated in Figure 8.

Due to the conservation of matter, the flow entering the left face of a geologic sample must equal the flow exiting the right face of a geologic sample.

It then follows that $q_{\text{left}} = q_{\text{right}}$, and so

$$Q_{\text{left}} = -ki_{\text{left}}A = -ki_{\text{right}}A = Q_{\text{right}}.$$

Therefore $i_{\text{left}} = i_{\text{left}}$ since the geologic medium and cross-sectional area must remain constant. Using equation 2, we find that

$$\begin{aligned}
\frac{h(3000) - h(2000)}{3000 - 2000} &= \frac{h(2000) - h(1000)}{2000 - 1000} \\
\Rightarrow \frac{50 - h(2000)}{1000} &= \frac{h(2000) - 70)}{1000} \\
\Rightarrow 50 - h(2000) &= h(2000) - 70 \\
\Rightarrow h(2000) &= 60.
\end{aligned}$$

We can make a couple of observations from this application.

1. We must place strict constraints on any function representing hydraulic head distribution, especially if hydraulic conductivity k is assumed to be constant.

2. There is an interplay between assumptions that a hydrologist might make about hydraulic conductivity and the predictions that might result concerning

hydraulic head values, as our equations directly depend on hydraulic conductivity, k.

In fact, under constant hydraulic conductivity, the hydraulic head function h must satisfy Laplace's Equation:

$$\frac{d^2h}{dt^2} = 0.$$

It follows that if the second derivative of h is 0, then the first derivative of h must be a constant, meaning that h is linear and so is of the form $h(x) = mx + b$ for $m, b \in \mathbb{R}$. Then

$$\begin{aligned} h_x &= m \text{ and} \\ h_{xx} &= 0. \end{aligned}$$

Since water is incompressible, the net amount of fluid entering a fixed incremental volume at any moment must be 0, due to the laws of conservation of mass.

1.5 Laplace Equation in 1 and 2 Dimensions

We shall now prove that groundwater flow must satisfy Laplace's Equation. First we will prove this for one dimension. Since $Q = kiA$ by Darcy's Law, and we just showed that

$$\begin{aligned} &\text{flow through left face} - \text{flow through left face} = 0 \\ &\Rightarrow k(-h_x(x))A - k(-h_x(x + \Delta x))A = 0 \end{aligned}$$

Since we presume that the geologic medium is not an aquiclude, and the cross-sectional area we are examining is nonzero, we can divide both sides by $-kA$, getting

$$\begin{aligned} h_x(x) - h_x(x + \Delta x) &= 0 \\ \Rightarrow \frac{h_x(x) - h_x(x + \Delta x)}{\Delta x} &= 0. \end{aligned}$$

We can do this because we are assuming Δx is a nonzero value. We can now try to take the limit of both sides as Δx approaches 0, getting

$$\begin{aligned} \lim_{\Delta x \to 0} \frac{h_x(x)) - h_x(x + \Delta x)}{\Delta x} &= \lim_{\Delta x \to 0} 0 \\ \Rightarrow \lim_{\Delta x \to 0} \frac{h_x(x)) - h_x(x + \Delta x)}{\Delta x} &= 0 \\ \Rightarrow \frac{dh_x}{dt} &= 0 \\ \Rightarrow h_{xx}(x) &= 0 \end{aligned}$$

As we took arbitrary x initially, it follows that $h_{xx} = 0$ for all, $x \in \mathbb{R}$, and so we have shown Laplace's equation holds for 1 dimension.

Now we shall derive Laplace's Equation for increased dimensions. In one dimension, we had $h_{xx} = 0$, meaning that we could only have linear functions for solutions. However, for two or more dimensions, Laplace's Equation gets a bit more complicated, and the space of "allowed" head functions exams. For instance, Laplace's Equation in two dimensions states that

$$h_{xx} + h_{yy} = 0,$$

and Laplace's Equation in three dimensions states that

$$h_{xx} + h_{yy} + h_{zz} = 0.$$

It must be noted that these equations apply strictly to the case of isotropic aquifers (the geologic medium remains the same under the content being considered) with constant hydraulic conductivity k.

We can rewrite the two- and three-dimensional versions of Laplace's Equation in terms of partial derivatives as follows:

$$\text{For two dimensions: } \frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} = 0$$
$$\text{For three dimensions: } \frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} + \frac{\partial^2 h}{\partial z^2} = 0.$$

Note that these are partial differential equations. Therefore, in order to solve them, we must make sure our solution fits a set of boundary conditions. This is analogous to what we did in our example in Section 1.4.

We shall now derive Laplace's Equation for flow in horizontal x and y directions, again assuming vertical flow is negligent.

Think of Figure 9 as a three-dimensional situation, where z is perpendicular to the drawing. The flow in the z-direction is assumed to be negligible, but the aquifer does have a thickness, and z characterizes the position through that thickness. We are also assuming our aquifer, as before, is isotropic and uniform, so k will characterize our hydraulic conductivity throughout the medium.

The net flow through the rectangular prism of incremental volume in Figure 10 should be zero, as stated before, due to the conservation of mass. The fluid flux q is a vector with components q_1 in the x-direction and q_2 in the y-direction. Because we are assuming negligible vertical flow in the z-direction, we may assume that fluid can be entering through four of six faces – i.e. all the faces that are not the top or bottom face. We never need to consider z-coordinates for any points, because

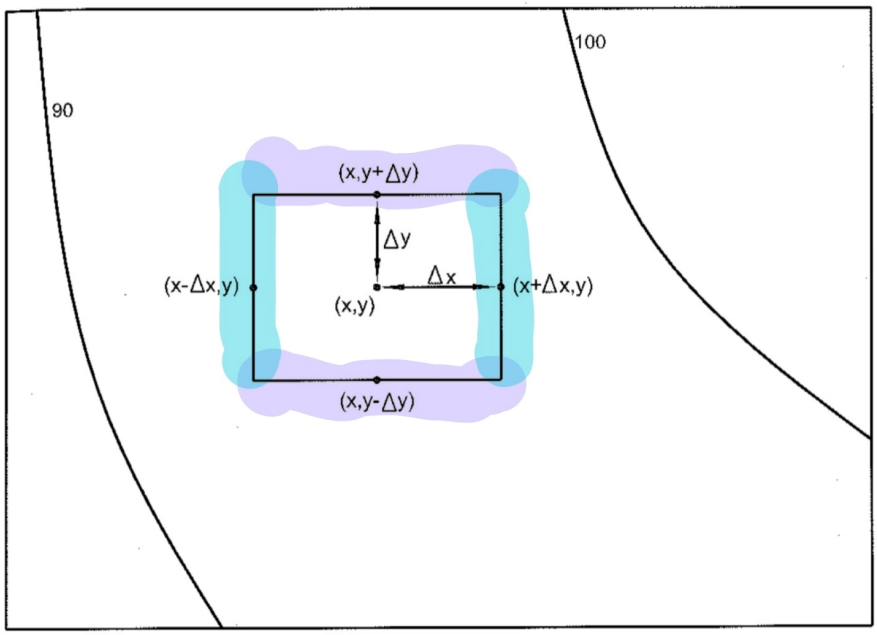

Figure 9: Volume Increment in a Groundwater Flow Field Exhibiting 2D Flow

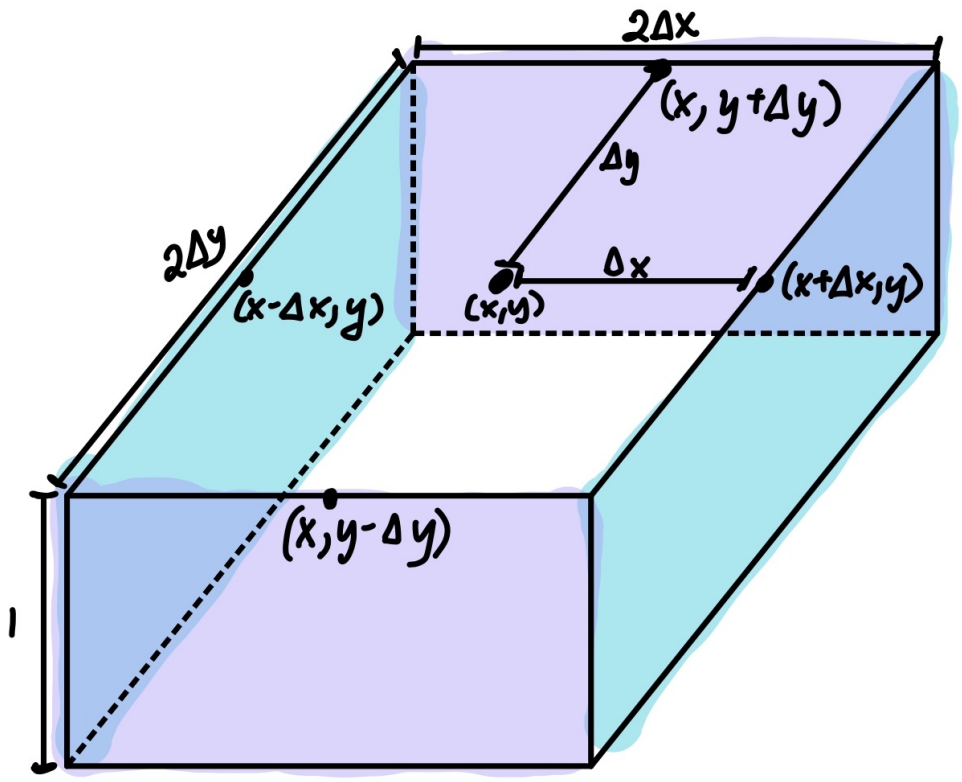

Figure 10: Rectangular Prism of Incremental Volume

everything is assumed to be constant in the z-direction.

Along the left face, the inward fluid flux (which may change with y) can be represented by the one specific fluid flux at the center of the left face, i.e. $q_1(x - \Delta x, y)$. This is the first component of the flux vector q, and is the only one relevant to the left face, because we do not care about y-component flow when looking exactly at the left face.

We may make similar assumptions along the other three faces. From this, we get

$$0 = \text{left flow in} + \text{right flow in} + \text{up flow in} + \text{down flow in}.$$

Using our equations we derived for each of these sides from Figure 10, we get

$$\begin{aligned} 0 &= q_1(x - \Delta x, y)(2\Delta y \cdot 1) - q_1(x + \Delta x, y)(2\Delta y \cdot 1) \\ &\quad + q_2(x, y - \Delta y)(2\Delta x \cdot 1) - q_2(x, y + \Delta y)(2\Delta x \cdot 1). \end{aligned}$$

From here, we may derive the two-dimensional Laplace Equation using our continuous flux equation.

$$\begin{aligned} 0 &= q_1(x - \Delta x, y)(2\Delta y \cdot 1) - q_1(x + \Delta x, y)(2\Delta y \cdot 1) \\ &\quad + q_2(x, y - \Delta y)(2\Delta x \cdot 1) - q_2(x, y + \Delta y)(2\Delta x \cdot 1) \\ \Rightarrow 0 &= -h(x - \Delta x, y)k(2\Delta y) + h(x + \Delta x, y)k(2\Delta y) \\ &\quad - h(x, y - \Delta y)k(2\Delta x) + h(x, y + \Delta y)k(2\Delta x) \\ \Rightarrow 0 &= -2h(x - \Delta x, y)\Delta y + 2h(x + \Delta x, y)\Delta y \\ &\quad - 2h(x, y - \Delta y)\Delta x + 2h(x, y + \Delta y)\Delta x. \end{aligned}$$

Now let us divide both sides by $2\Delta x \Delta y$, both of which takes nonzero values only. Doing so, we get

$$\begin{aligned} 0 &= \frac{h(x + \Delta x, y) - h(x - \Delta x, y)}{2\Delta x} + \frac{h(x, y + \Delta y) - h(x, y - \Delta y)}{2\Delta y} \\ \Rightarrow \lim_{\Delta x \to 0} 0 &= \lim_{\Delta x \to 0} \left[\frac{h(x + \Delta x, y) - h(x - \Delta x, y)}{2\Delta x} + \frac{h(x, y + \Delta y) - h(x, y - \Delta y)}{2\Delta y} \right] \\ \Rightarrow 0 &= \lim_{\Delta x \to 0} \frac{h(x + \Delta x, y) - h(x - \Delta x, y)}{2\Delta x} + \frac{h(x, y + \Delta y) - h(x, y - \Delta y)}{2\Delta y} \\ \Rightarrow 0 &= h_{xx}(x) + \frac{h(x, y + \Delta y) - h(x, y - \Delta y)}{2\Delta y} \\ \Rightarrow \lim_{\Delta y \to 0} 0 &= \lim_{\Delta y \to 0} \left[h_{xx}(x) + \frac{h(x, y + \Delta y) - h(x, y - \Delta y)}{2\Delta y} \right] \\ \Rightarrow 0 &= h_{xx}(x) + h_{yy}(y). \end{aligned}$$

Thus we have derived Laplace's Equation for two dimensions. Functions satisfying

this property are called harmonic functions.

There are two key characteristics to situations satisfying Laplace's Equation:

1. The flow of quantity is at a rate proportional to the gradient of some "potential" (pressure difference from varying hydraulic heads, in this situation).

2. Conservation condition (conservation of mass and volume here) requiring that through the flow regime, no material spontaneously appears or disappears.

We will now begin solving boundary problems using linear approximations in the next section.

1.6 Numerical Methods for Solving Head Value Boundary Problem

Our broad strategy to finding numerical solutions for head values is to find a system of linear equations that are a reasonable approximation to our partial differential equations. We then shall solve that system of equations.

We know that the definition of a derivative is

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

Thus, for small Δx, it follows that

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x}.$$

In fact, we see that if our function f does not have a high degree of curvature, the slope of the secant line is a reasonable approximation to the slope of the tangent line.

Three different approximations can be seen from Figure 11. They are the forward difference approximation

$$f'(x) \approx \frac{f(x + \Delta x) - f(x)}{\Delta x},$$

the backward difference approximation,

$$f'(x) \approx \frac{f(x) - f(x - \Delta x)}{\Delta x},$$

and the central difference approximation

$$f'(x) \approx \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}.$$

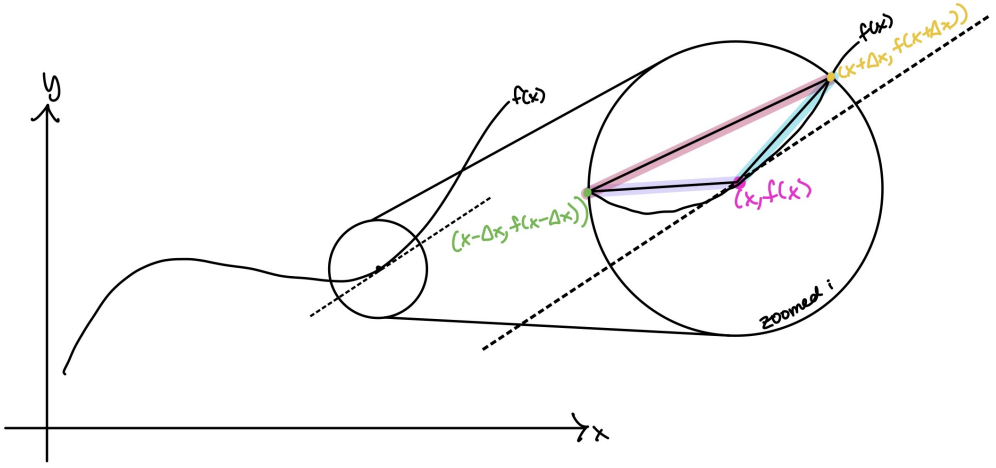

Figure 11: Examples of Linear Approximations of Function Values

We can use the central difference approximation at the second derivative as follows:

$$\begin{aligned} f''(x) &\approx \frac{f'(x+\frac{\Delta x}{2}) - f'(x-\frac{\Delta x}{2})}{\Delta x} \\ \Rightarrow f''(x) &\approx \frac{\frac{f(x+\Delta x)-f(x)}{\Delta x} - \frac{f(x)-f(x-\Delta x)}{\Delta x}}{\Delta x} \\ \Rightarrow f''(x) &\approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}. \end{aligned}$$

Plugging this in to our hydraulic head functions, we get the following approximations:

$$\begin{aligned} h_{xx}(x,y) &\approx \frac{h(x+\Delta x, y) - 2h(x,y) + h(x-\Delta x, y)}{\Delta x^2} \\ h_{yy}(x,y) &\approx \frac{h(x, y+\Delta y) - 2h(x,y) + h(x, y-\Delta y)}{\Delta y^2}. \end{aligned}$$

Recall that we derived the two-dimensional Laplace Equation, which states that

$$h_{xx} + h_{yy} = 0,$$

so plugging in these approximations, we get that

$$\frac{h(x+\Delta x, y) - 2h(x,y) + h(x-\Delta x, y)}{\Delta x^2} + \frac{h(x, y+\Delta y) - 2h(x,y) + h(x, y-\Delta y)}{\Delta y^2} = 0.$$

Suppose we have a site where the distance between wells in the x direction is d, and the distance between wells in the y direction is also d. Then our equation becomes

$$\frac{h(x+d, y) - 2h(x,y) + h(x-d, y)}{d^2} + \frac{h(x, y+d) - 2h(x,y) + h(x, y-d)}{d^2} = 0.$$

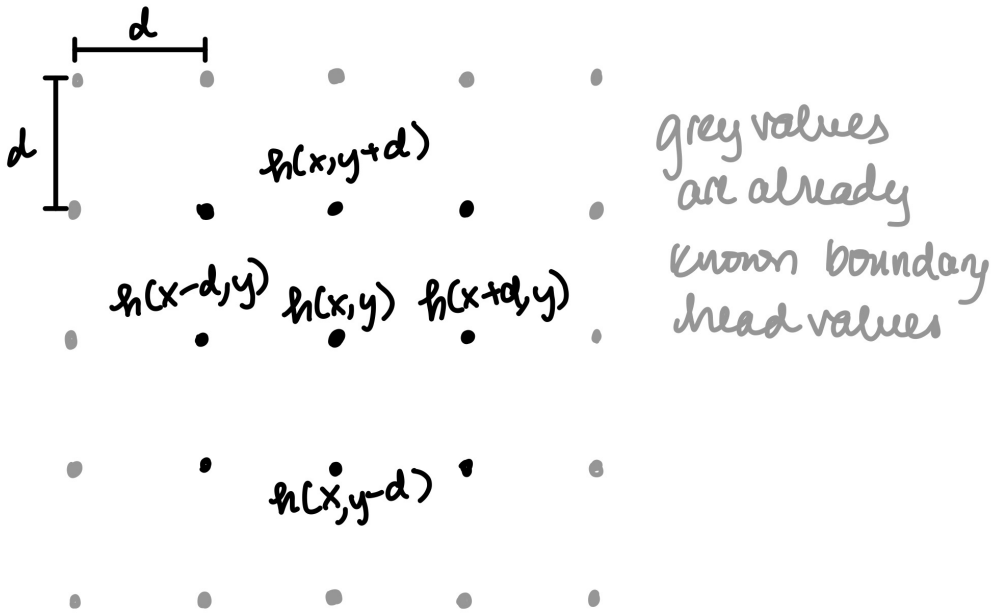

Figure 12: Example of Boundary Value Head Problem

We can multiply both sides by d^2, and get

$$h(x+d,y) - 2h(x,y) + h(x-d,y) + h(x,y+d) - 2h(x,y) + h(x,y-d) = 0.$$

Simplifying this further, we see that

$$4h(x,y) = h(x+d,y) + h(x-d,y) + h(x,y+d) + h(x,y-d),$$

and dividing both sides by 4, we have

$$h(x,y) = \frac{1}{4}(h(x+d,y) + h(x-d,y) + h(x,y+d) + h(x,y-d)).$$

As we can see in Figure 12, we find that, using this linear approximation, the water table height at a point (x,y) is simply the average of the water table heights in the x- and y- directions a distance of d away.

Let us now look at this approximation applied to the following example:

Suppose we want to find the height of h_{25}, located at point (5000,1000). Using our approximation of the Laplace equation, with $\Delta x = \Delta y = 1000$, we get that

$$\begin{aligned} 0 &= \frac{340 - 2h_{25} + h_{20}}{1000^2} + \frac{h_{24} - 2h_{25} + 300}{1000^2} \\ \Rightarrow 0 &= 340 + h_{20} + h_{24} + 300 - 4h_{25} \\ \Rightarrow h_{25} &= \frac{340 + h_{20} + h_{24} + 30}{4} \end{aligned}$$

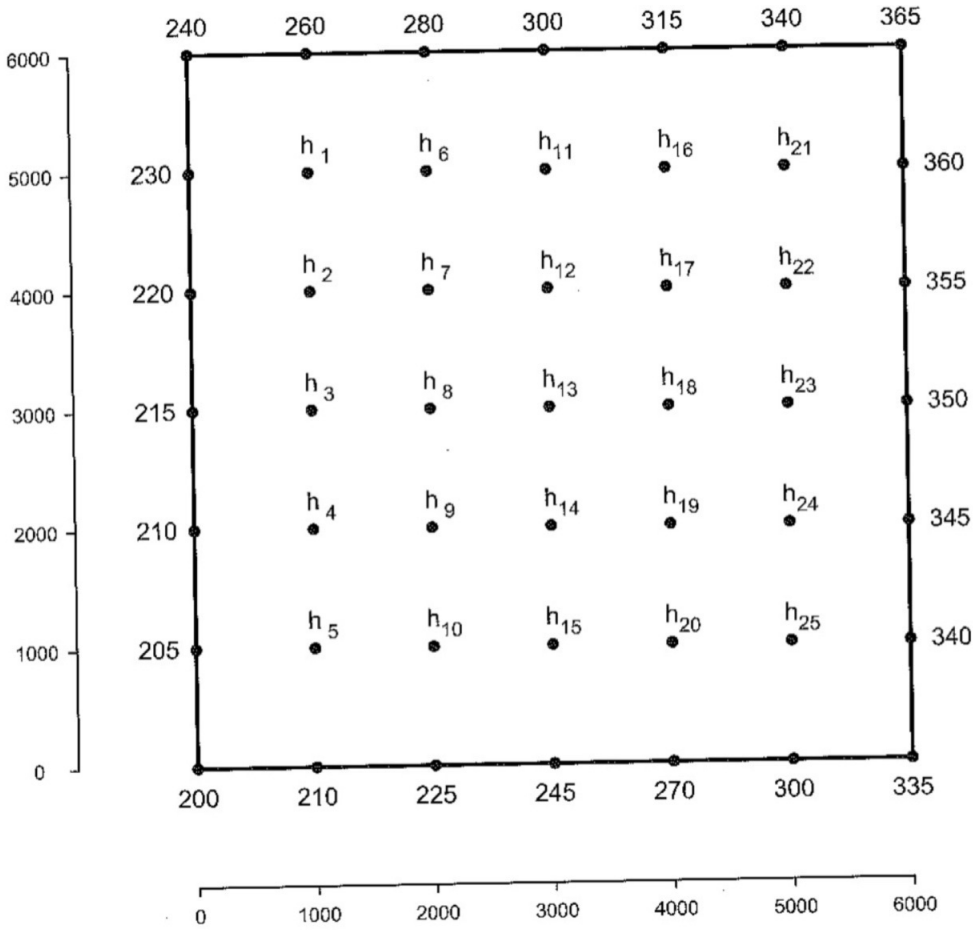

Figure 13: Example Boundary Value Problem for 25 Wells

In other words, the head height of h_{25} is equal to the average of the four heads surrounding its cardinal directions.

This leads us to the last portion of our work in water table height derivations. Now, given a set of boundary values of hydraulic head heights, we can solve a system of linear equations to find the hydraulic head heights at the interior points.

1.7 Constructing a System of Linear Equations for Water Table Heights

In order to understand how to construct our system of linear equations, let us take a look at an example problem. Suppose we want to find the water table heights a_i in a 3×3 square, where the boundary values b_{Pi} are provided (where P is their position, like top, bottom, left, or right, and i is their index in that position. For instance, b_{L3} is the third entry in the left boundary). The set up looks something like this:

b_{TL}	b_{T1}	b_{T2}	b_{T3}	b_{TR}
b_{L1}	a_1	a_2	a_3	b_{R1}
b_{L2}	a_4	a_5	a_6	b_{R2}
b_{L3}	a_7	a_8	a_9	b_{R3}
b_{BL}	b_{B1}	b_{B2}	b_{B3}	b_{BR}

Let us call the side length of our square of unknown values r. In this case, $r = 3$. So we have to solve a r^2, or 9-variable, system of 9 linear equations. We have found that the value of a hydraulic head height is the average of the four heights surrounding it – thus, we can view each of our linear equations as corresponding to the average of one of our unknowns.

For example, the equation corresponding to head value a_7 is:

$$a_7 = \frac{a_2 + a_{12} + a_6 + a_8}{4}$$
$$\Rightarrow 0 = a_2 + a_6 - 4a_7 + a_8 + a_{12}$$

We can classify our unknowns into three categories: corner unknowns, border unknowns, and center unknowns. Corner unknowns, like a_1, a_3, a_7, and a_9, border exactly two known boundary values, as seen above. Border unknowns, like a_2, a_4, a_6, and a_8, border exactly one known boundary value, as seen above. Finally, center unknowns, like a_5, border no known boundary values. As r increases, we find that the majority of unknowns will be center unknowns, but in our current example, only a_5 is a center unknown. We use this classification in order to set up generalizations

for the different linear equations we will see corresponding to the averages of each of these types of unknown.

Let us begin by considering the system of equations for center unknowns. We can see that for any non-corner, non-boundary head value a_i, the corresponding linear equation is of the form:

$$\text{(Center unknowns) } a_i\text{: } a_{i-r} + a_{i-1} - 4a_i + a_{i+1} + a_{i+r} = 0.$$

Now let us consider what happens at corner unknowns. In our specific $r = 3$ example, the entries at the corners are entries a_1, a_3, a_7, and a_9.

In an arbitrary head value problem, the corner unknowns would be the top left corner, a_1, the top right corner, a_r (because it would be the last entry in the first row, and there are r entries per row, the bottom right corner, a_{r^2} (because it would be the last entry and there are a total of r^2 entries), and the bottom left corner, a_{r^2-r+1} (because it would be the first entry in the last row, and there are r entries per row, its index would be last entry−number of entries in row+1). We can verify that this generalization works for our specific example by plugging in $r = 3$, and getting corner entries of a_1, a_3, a_7, and a_9, as desired.

Since the corner unknowns each involve exactly two known boundary values, they have a different set of equations:

$$\begin{aligned}
\text{(Top left corner) } a_1\text{: } & b_{T1} + b_{L1} - 4a_1 + a_{1+1} + a_{1+r} = 0 \\
& \Rightarrow -4a_1 + a_2 + a_{1+r} = -(b_{T1} + b_{L1}) \\
\text{(Top right corner) } a_r\text{: } & b_{Tr} + a_{r-1} - 4a_r + b_{R1} + a_{r+r} = 0 \\
& \Rightarrow a_{r-1} - 4a_r + a_{2r} = -(b_{Tr} + b_{R1}) \\
\text{(Bottom left corner) } a_{r^2-r+1}\text{: } & a_{r^2-r+1-r} + b_{Lr} - 4a_{r^2-r+1} + a_{r^2-r+1+1} + b_{B1} = 0 \\
& \Rightarrow a_{r^2-2r+1} - 4a_{r^2-r+1} + a_{r^2-r+2} = -(b_{Lr} + b_{B1}) \\
\text{(Bottom right corner) } a_{r^2}\text{: } & a_{r^2-r} + a_{r^2-1} - 4a_{r^2} + b_{Br} + b_{Rr} = 0 \\
& \Rightarrow a_{r^2-r} + a_{r^2-1} - 4a_{r^2} = -(b_{Br} + b_{Rr})
\end{aligned}$$

Finally, let us consider the equations corresponding to the border unknowns,

which border exactly one known boundary value each. Their equations are as follows:

$$\begin{aligned}
\text{(Top border unknowns) } a_i\text{: } & b_T + a_{i-1} - 4a_i + a_{i+1} + a_{i+r} = 0 \\
& \Rightarrow a_{i-1} - 4a_i + a_{i+1} + a_{i+r} = -b_T \\
\text{(Left border unknowns) } a_i\text{: } & a_{i-r} + b_L - 4a_i + a_{i+1} + a_{i+r} = 0 \\
& \Rightarrow a_{i-r} - 4a_i + a_{i+1} + a_{i+r} = -b_L \\
\text{(Right border unknowns) } a_i\text{: } & a_{i-r} + a_{i-1} - 4a_i + b_R + a_{i+r} = 0 \\
& \Rightarrow a_{i-r} + a_{i-1} - 4a_i + a_{i+r} = -b_R \\
\text{(Bottom border unknowns) } a_i\text{: } & a_{i-r} + a_{i-1} - 4a_i + a_{i+1} + b_B = 0 \\
& \Rightarrow a_{i-r} + a_{i-1} - 4a_i + a_{i+1} = -b_B
\end{aligned}$$

We can combine these linear equations to get a system of equations of the form $Ax = b$. For our example with $r = 3$, our system of equations comes out to

$$\begin{bmatrix}
-4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\
1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix}
=
\begin{bmatrix} -(b_{T1} + b_{L1}) \\ -b_{T2} \\ -(b_{T3} + b_{R1}) \\ -b_{L2} \\ 0 \\ -b_{R2} \\ -(b_{L3} + b_{B1}) \\ -b_{B2} \\ -(b_{B3} + b_{R3}) \end{bmatrix}$$

Now that we have a method of constructing a system of linear equations, we can explore different ways of quickly solving these systems.

1.8 Iterative Methods

Because solving large systems of equations can be quite time intensive, we will explore two iterative numerical methods of approximating solutions to a system of linear equations: the Jacobi and Gauss-Seidel methods. Iterative methods of solving systems of linear equations are often much faster than Gaussian elimination. The two methods we are focusing on have a computational complexity on the order of n^2, whereas Gaussian elimination has complexity on the order of n^3. These are both iterative numerical methods, meaning that they ideally converge to an exact solution over time. The following analyses of iterative methods, the Jacobi Method, and the Gauss-Seidel method follow work by David M. Strong in 2005 Strong [2005].

Iterative numerical methods can be thought of as a modification of an original

system of equations. The original system is of the form $Ax = b$, where A is our coefficient matrix, x is our vector of unknowns, and b is a column vector of constants.

These two iterative numerical methods take the form

$$\begin{aligned} Mx^{(k+1)} &= Nx^{(k)} + b \\ \Rightarrow x^{(k+1)} &= M^{-1}Nx^{(k)} + M^{-1}b \\ \Rightarrow x^{(k+1)} &= Bx^{(k)} + \widetilde{b}, \end{aligned}$$

where $B = M^{-1}N$ and $\widetilde{b} = M^{-1}b$ for invertible square matrices M and N. We will discuss the significance of M and N shortly. We call B our iteration matrix. We have converged to our exact solution x when $x^{(k+1)} = x^{(k)} = x$, meaning that x is the fixed point of our equation. At x, we see that

$$\begin{aligned} x &= Bx + \widetilde{b} \\ \Rightarrow x &= M^{-1}Nx + M^{-1}b \\ \Rightarrow Mx &= Nx + b \\ \Rightarrow (M - N)x &= b. \end{aligned}$$

Recall that our original system of equations had that $Ax = b$, so it follows that $A = M - N$. However, we must note that choosing M and N such that $A = M - N$ does not alone guarantee convergence. Convergence depends on our iteration matrix $B = M^{-1}N$. We can examine the conditions for convergence to occur by looking at the error $e^{(k)}$ between our $k + 1$st iteration $x^{(k+1)}$ and exact solution x. Recall that

$$x = Bx + \widetilde{b} \text{ and} \tag{3}$$

$$x^{(k+1)} = Bx^{k} + \widetilde{b} \tag{4}$$

Subtracting (4) from (3), we get

$$\begin{aligned} x - x^{(k+1)} &= B(x - x^{(k)}) + \widetilde{b} - \widetilde{b} \\ \Rightarrow e^{(k)} &= Be^{(k-1)} \\ &= B(Be^{(k-2)}) \\ &= B^n e^{(k-n)} \\ \Rightarrow e^{(k)} &= B^k e^{(0)} \end{aligned}$$

We can then take the norm of both sides, getting that

$$\begin{aligned} ||e^{(k)}|| &= ||B^k e^0|| \\ &\leq ||B||^k ||e^{(0)}||. \end{aligned}$$

Thus, it follows that our error $e^{(k)}$ converges to 0 if and only if $||B|| < 1$. We notice that the smaller $||B||$ is, the faster our error will converge to 0, and so the faster our approximation will converge to our exact solution x. However, we must also note that if $||B|| > 1$, our error will grow.

One way we can guarantee that $||B|| < 1$ is by making sure that our original coefficient matrix A is strictly diagonally dominant, meaning that for each row of A, the absolute value of the diagonal element is strictly larger than the sum of the absolute values of the off-diagonal elements. However, there do exist non-diagonally dominant matrices that still converge Strong [2005]. In fact, the system of linear equations for the boundary value problem is a non-diagonally dominant matrix that converges to a solution.

1.9 The Jacobi Method

Let us explore the first iterative method: the Jacobi Method. Recall that our original system of equations is of the form

$$Ax = b,$$

where A is our coefficient matrix, x is our vector of unknowns, and b is a column vector of constants. We can split our matrix A into three components: a lower triangular matrix L (which does not include diagonal entries), a diagonal matrix D, and an upper triangular matrix U (which also does not include diagonal entries), which all sum to A. Then we have Then we have

$$\begin{aligned} Ax &= b \\ \Rightarrow (L + U + D)x &= b \\ \Rightarrow (L + U)x + Dx &= b \\ \Rightarrow Dx &= -(L + U)x + b \\ \Rightarrow x &= D^{-1}[-(L + U)x + b]. \end{aligned}$$

If x is our exact solution, then we can see from above that we must have $x = D^{-1}[-(L + U)x + b]$. Thus we have

$$M = D \text{ and } N = -(L + U),$$

such that

$$M - N = D - (-L - U) = L + U + D = A.$$

Thus our iterative matrix B is

$$B = D^{-1}(-(L + U)),$$

and our $\widetilde{b}$ is

$$\widetilde{b} = D^{-1}b.$$

1.10 The Gauss-Seidel Method

Now let us consider our second iterative method: the Gauss-Seidel Method. Let us similarly decompose our original coefficient matrix A into L, U, and D such that $A = L + U + D$. Then we have

$$\begin{aligned} Ax &= b \\ \Rightarrow (L + D + U)x &= b \\ \Rightarrow (L + D)x + Ux &= b \\ \Rightarrow (L + D)x &= -Ux + b \\ \Rightarrow x &= (L + D)^{-1}[-Ux + b]. \end{aligned}$$

If x is our exact solution, then we can see from above that we must have $x = (L + D)^{-1}[-Ux + b]$. Thus we have

$$M = (L + D) \text{ and } N = -(U),$$

such that

$$M - N = L + D - (-U) = L + U + D = A.$$

Thus our iterative matrix B is

$$B = (L + D)^{-1}(-U),$$

and our $\widetilde{b}$ is

$$\widetilde{b} = (L + D)^{-1}b.$$

1.11 Remarks

We implemented both these solvers in MATLAB (see Appendix 4.1, 4.2, 4.3), and compared rates of convergence for both. We find that it takes approximately twice as many iterations in the Jacobi solver to achieve the same level of accuracy as the

Gauss-Seidel solver. We experimented with error bounds for different powers of 10 and found this to consistently be the case. This is due to $||B|| = \lambda_{\max}$, the largest eigenvalue, being different in each case.In fact, the norm of the iterative matrix for the Jacobi Solver (0.866) is the square root of the norm of the iterative matrix for the Gauss-Seidel matrix (0.75) in our example from Figure 13. Examples of results and eigenvalues for Example 13 for each method are included in Appendix 4.4 and 4.5.

2 Part II: Models for Electric Vehicle Charger Use

The second part of this thesis broadly focuses on modeling the usage of electric car chargers on the Bryn Mawr College campus based on current usage data. We constructed a queuing simulation to get a rough idea of the number of electric chargers that would be needed to adequately support usage if all non-bus College vehicles were switched to their electric counterparts.

We find in Section 2.6 that switching all of Bryn Mawr's non-bus vehicles to their electric counterparts could be supported by just 6 Level 2 chargers, and that the sixth charger would be available quite often, making it a possible candidate for public use. Bryn Mawr College should move towards replacing their vehicle fleet, as the present vehicles age out, with their equivalent electric vehicle alternatives, and support them with an electric behicle charging network of 6 chargers.

2.1 Motivation

Climate change has been a pressing issue since the dawn of the industrial revolution, but the urgency with which we must alter our behavior has been increasing over the past few years. The Intergovernmental Panel on Climate Change (IPCC) published a report in 2019 on the 1.5 °C increase in global temperature since pre-industrial times on Climate Change [2019]. One of the primary methods of mitigation for this climate change is reduction of greenhouse gas emissions. According to the EPA, 27% of greenhouse gas emissions are due to transportation Agency [2020], and of that 27%, about 58% of these emissions are from passenger vehicles Agency [2019]. As a result, a switch to electric vehicles, powered by electric motors rather than burning fuel, is a key component of reducing carbon emissions on a large scale. Bryn Mawr College has set a goal of reaching net zero carbon emissions by 2035 College [2021], and a key part of reducing emissions is switching its fleet to electric vehicles. This section aims to understand the number of chargers that would be needed on campus to support switching the College's non-bus vehicles to their electric counterparts. It is important to note that all college-owned non-bus vehicles could be replaced with electric versions.

2.2 Problem Background and Assumptions

Before we explain the design of our queuing simulation, we shall explain the background of the problem to provide some context for design choices.

We are assuming that each College car drives around a certain amount each day, and depletes their battery each day based on the distance traveled. We assume that once the car has used eighty percent of its usable battery capacity, it returns to a

charger to fully recharge. If all chargers are occupied, the car enters the last spot in the virtual queue and will not be used until it finishes charging.

We are assuming all chargers are Level 2 chargers, which means that they charge cars at a constant rate of 7 kilowatts. For those vehicles for which we have mileage data, we are using data from time since purchase to calculate daily use and charging frequency. For vehicles without mileage data, we are making an estimate of their mileage use by comparing with the data for other similar college-operated vehicles.

There are five categories of college vehicle. The electric counterparts, along with the mileage con [2022] and usable battery capacity, are listed below:

Car Type	Range (miles)	Usable Battery Capacity (kWh)	Mileage (miles/kWh, $\frac{\text{range}}{\text{battery capacity}}$)	Time to charge 80% (hours, $\lceil \frac{\text{mileage}}{7} \rceil$)
8-person Minivan	350	111	3.15	16
Passenger Van	140	80	1.75	12
Cargo Van	126	68	1.85	10
SUV	220	71.4	3.08	11
Pickup Truck	230	98	2.35	14
High Roof	126	68	1.85	10

For each car with usage data, we divided the most recent odometer mileage reading by the number of years since the car had been purchased in order to get yearly use, then derived daily use from there. We divided the range for the model by the daily distance traveled in order to calculate the frequency of charging. The makes of these electric counterparts are listed in Appendix 4.6. Note that every car had an electric alternative. The data for nineteen cars was available, and their models, daily use, department, and charging frequency are listed below, in Table 1. This data is from Bryn Mawr College Transportation con [2022]. For cars whose usage data was not available (cars 20-40), we assumed a charging frequency of 15 days. This is because we based usage off of campus safety vehicle usage, since we assumed facilities vehicles would be moving around with about the same frequency.

We looked at the usage of Level 2 chargers given this set of electric vehicles. We assumed that if a car arrived when all chargers were already being used, they waited in a virtual queue, during which the car was not used and was presumably parked until a charger was available. We also wanted to consider that if a car was left to charge overnight, it would not be retrievable until the morning, so we wanted to introduce a period of "inactivity" for the charger during which vehicles could not be retrieved from chargers and could not begin charging.

Car Number	Original Model	Department	Electric Model	Daily Use (miles)	Frequency of Charging (days)	Time to Fully Charge (minutes)
1	Odyssey van	Transportation (Rental)	Minivan	67.48	4.1	960
2	RAV4	Campus Safety	SUV	21.49	8.2	660
3	RAV 4	Lantern Van (Campus Safety)	SUV	18.51	9.5	660
4	E350 pass. van	Athletics	Passenger Van	11.40	9.8	720
5	E350 pass. van	Athletics	Passenger Van	10.25	10.9	720
6	E-350 pass. Van	Athletics	Passenger Van	9.99	11.2	720
7	RAV4	Campus Safety	SUV	11.77	15.0	660
8	Suburban	Geology	SUV	11.18	15.7	660
9	Outback	Biology	SUV	9.15	19.2	660
10	1500 Cargo van	Dining Services (Wyndham)	Cargo Van	4.65	21.7	600
11	Cargo Van	Post Office	Cargo Van	4.27	23.6	600
12	Handicap van	Transportation (Rental)	Minivan	6.39	43.8	960
13	Caravan	Transportation (Rental)	Minivan	5.75	48.7	960
14	Ram Van	Transportation (Rental)	Minivan	5.71	49.0	960
15	Civic - CNG	Transportation (Rental)	SUV	3.57	49.3	660
16	2500 CNG van	LITS (Multimedia)	Cargo Van	1.92	52.6	600
17	Caravan	Transportation (Rental)	Minivan	4.59	61.0	960
18	NVNV High Roof	Dining Services	High-roof	1.16	86.6	660
19	Sodena - 4 dr wagon	Transportation (Bi-Co)	Minivan	1.69	165.5	960
20	F250 Pickup Truck	Facilities	Pickup Truck	N/A	15	840
21	F250 pickup	Facilities (Carpenters)	PICKUP	N/A	15	840
22	E150 cargo van	Facilities (Carpenters)	Cargo Van	N/A	15	600
23	E150 cargo van	Facilities (Electricians)	Cargo Van	N/A	15	600
24	E150 cargo van	Facilities (Electricians)	Cargo Van	N/A	15	600
25	F350 pickup	Facilities (Grounds)	Pickup Truck	N/A	15	840
26	F250 pickup	Facilities (Grounds)	Pickup Truck	N/A	15	840
27	F350 utility body	Facilities (HVAC)	Pickup Truck	N/A	15	840
28	F250 Utility body	Facilities (HVAC)	Pickup Truck	N/A	15	840
29	Transit Conn. Van	Facilities (HVAC)	Cargo Van	N/A	15	600
30	Transit 150 LR cargo van	Facilities (HVAC)	Cargo Van	N/A	15	600
31	Transit	Facilities (HVAC)	Cargo Van	N/A	15	600
32	utility Caravan	Facilities (Locksmith)	Cargo Van	N/A	15	600
33	Closed utilitybody	Facilities (Plumbing)	Pickup Truck	N/A	15	840
34	F350 utility truck	Facilities	Pickup Truck	N/A	15	840
35	E150 cargo van	Facilities	Cargo Van	N/A	15	600
36	Suburban	Geology	SUV	N/A	15	660
37	F-150	Housekeeping	Pickup Truck	N/A	15	840
38	F350 Stake body	Housekeeping	Pickup Truck	N/A	15	840
39	Transit Van	Housekeeping	Cargo Van	N/A	15	600
40	Odyssey van	Transportation (Rental)	Minvan	N/A	15	960

Table 1: Table of Car Data for Bryn Mawr College

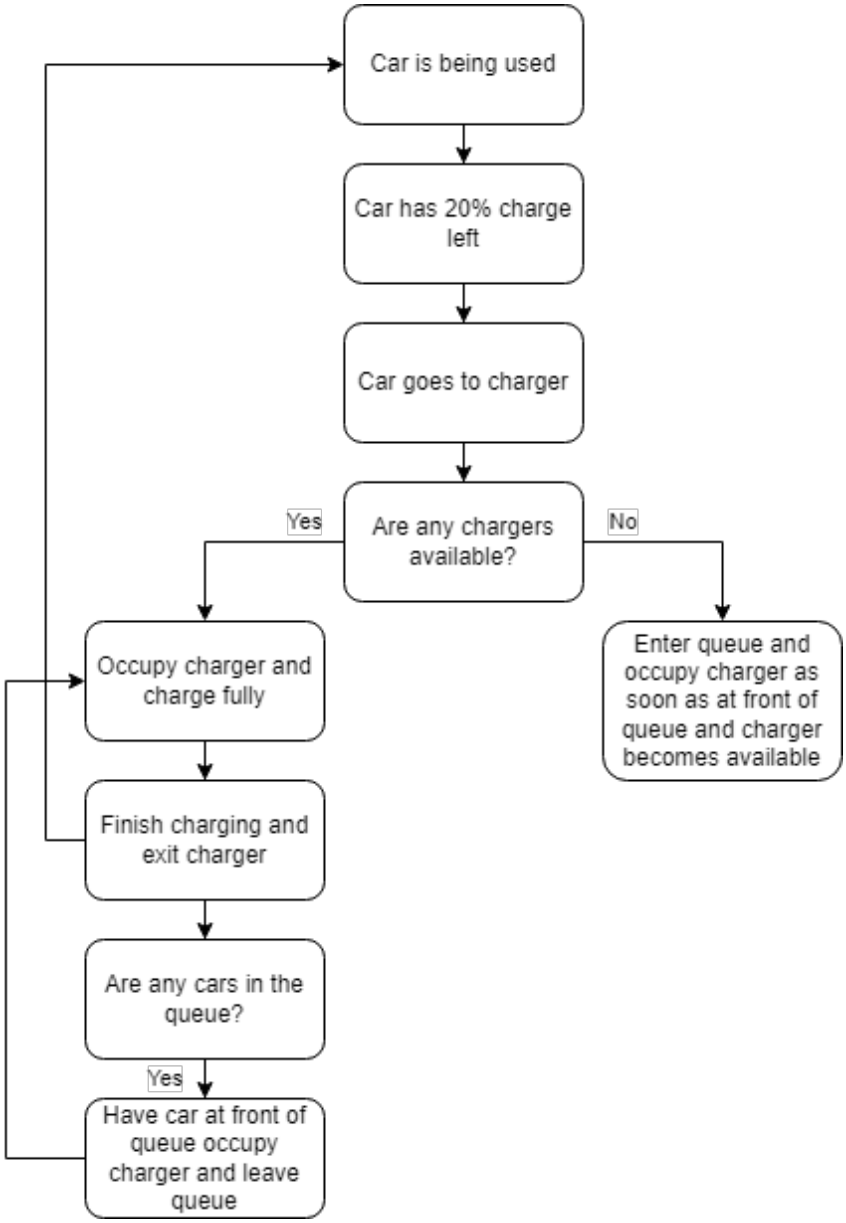

Figure 14: Overall Structure for System Behavior

2.3 Methodology and Development

We wanted our model to follow a basic overall structure. Cars would be used until they reached 20% charge, at which point, they would head to a charging station to charge. When they arrived at the charging station, they would plug in and begin charging if a charger was available. If not, they would enter the queue and only begin charging once they were at the front of the queue and a charger became unoccupied. While in the queue, the car would not be used further. When a car plugs in to charge, it charges fully. Once finished charging, it stops occupying the charger and goes back to being used until it hits 20% charge again. When a car leaves the charger, we plug in any car at the front of the queue immediately. This process is illustrated in Figure 14.

While this system is fairly simple at first glance, there are many moving parts. Of note are three submethods: first, the way in which initial car arrival times at stations are handled (Section 2.3.1), second, the way the queue is handled (Section 2.3.2), and third, the way we can implement rest hours for the chargers (Section 2.3.3). These submethods also tie into the overall structure of the simulation, which deals with how we look at the passage of time in the simulation (Section 2.3.4) and how cars move through the system in the simulation (Section 2.3.5).

2.3.1 Initial Arrival Generation Methods

There were multiple ways to handle the initial time of arrival for cars at the charge station. The first way to generate initial arrival times for cars was to assume that all cars began the simulation fully charged, and have their initial arrival time equal the frequency of charging. However, in our Bryn Mawr case, cars 20-40 all have a frequency of 15 days, meaning that on day 15 of the simulation, there would be a huge overflow of cars. Furthermore, over time, the cars arrival times would space out due to time spent waiting in the queue for a charger. Therefore, we decided to forego waiting for the cars to stagger their arrival times as the simulation ran, and instead staggered arrival times in the first place. The second way of generating initial arrival times was to think of it as cars beginning the simulation with a random amount of charge. That way, we generated initial arrival times randomly for each car, with values ranging from 0 minutes into the simulation up to the frequency of charging for that car. This second method was the one we ended up using for our simulation. This submethod for generating initial arrival times is called the **arrivals** function.

2.3.2 Queue Handling Methods

We also considered how the queuing system worked for this simulation. The first method of queuing assumed that a car arriving at an occupied charging station knew at what time the a charger would be soonest available, and would enter the queue for that charger. Future cars arriving would know the availability time for the next-soonest available charger and would enter the queue for that charger, and so on until every charger had a car in the queue. However, future cars that arrived would have no idea of how long the cars in the queue would take to finish charging, so instead of entering the queue, would return to try to grab a spot some fixed amount of time in the future. While this method worked, we found that it is instead feasible to set up one virtual queue for all the chargers. Under this new system, there would just be one queue for all the chargers, with unlimited capacity. Once all chargers were occupied, if a new car arrived, it would enter the end of the queue and stop being

used. As soon as a charger was available, whoever was at the front of the queue would occupy that charger, leaving the queue and moving the next person in the queue to the front. This system is possible in practice due to virtual queues being in common usage in the world; one can easily set up a system to keep track of a queue and notify a driver as soon as a charging spot opens up. This second method is the queuing method we utilized for our final simulation.

2.3.3 Implementation of Nighttime Hours

Now understanding how the queuing system worked for our simulation, we had to consider the practicality of some of its aspects. Suppose a spot opened up at 3 AM? It is unreasonable to assume that a driver would wake up to go and plug in the car to charge at that hour. Similarly, suppose a car finished charging at 3 AM? Again, it is unreasonable to assume that a driver would wake up to unplug the car from the charger. For this reason, we wished to introduce hours of inactivity, or "nighttime hours," for the simulation. For a stretch of time each day, cars would not be able to arrive at the charger and would not be able to leave the charger due to personnel not being available to perform those tasks during that time. The exact algorithm used to calculate whether or not it was nighttime in the simulation, **isnight**, is discussed in Section 2.4.

2.3.4 Event- vs. Time-Driven Simulations

Moving onto broader, structure-based questions, we consider the question of how the simulation moves through time. In our simulation, we are looking at how cars pass through the system and charge over time. There are two approaches to understand how the simulation handles this. The first approach is called an event-driven simulation, in which our simulation only moves forward when events occur. For example, the first version of our simulation began with this event-driven structure based on the ship harbor queuing model from A First Course in Mathematical Modeling Giordano et al. [2015]. Instead of iterating over time, the simulation iterated over the number of cars charged over the run time of the simulation. A flowchart of this simulation is depicted in Figure 15. The simulation moves forward by considering only the arrival time of the next car – time itself is not tracked on its own. For example, the first car arrival time is generated, and then the arrival time of the next car is generated based on the arrival time of the first car. Then we look at the difference between the arrival time of the second car and the time at which the first car finishes charging, and determine whether the second car needs to wait or not. Once the second car's charge finish time is calculated, we generate the third car's arrival time based on the arrival time of the second car, and repeat until we have run

through the number of cars we wanted to consider in our simulation. At first glance, a number of issues crop up with this event-driven simulation. Rather than looking at the behavior of our system over a period of time, we follow a certain number of cars and see what happens over that time. Furthermore, since arrival times are so closely linked between consecutive cars, we have a much "flatter" simulation, with less room to maneuver with staggered arrival times. Furthermore, as we will discuss in Section 2.3.5, this method makes it impossible to keep track of specific instances of cars after they leave their charger.

For this reason, we switched to a time-driven structure for our final simulation, in which, as the name suggests, the simulation increments over time rather than over car arrivals. In this case, the simulation begins at time zero and runs for a fixed length of time. At each minute, the system checks if a car has arrived or finished charging, and handles the system accordingly, then increases time by 1. At the next time unit, it does the same, and so on until the simulation has run through the desired length of time.

2.3.5 Open- vs. Closed-Network Queuing Simulations

Finally, we wanted to consider how cars moved through the system outlined in Figure 14. There are two ways to approach how cars move through the system. The first method is an open-network queuing system, in which a car is created, moves through the process of entering a queue and charging, and once finished charging, leaves the system and cannot return to charge again. The process illustrated in Figure 15 is an example of an open-network queuing system. Each car is generated, moves through the system, then is effectively destroyed. However, for our specific case we could not have this happen. We intended to explore the usage behavior of a specific 40-car fleet of vehicles, and in order to do so, would need to have cars preserved as they moved through the system. For that reason, we switched to a closed-network queuing system.

In a closed-network queuing system, as the name implies, the cars moving through the system are a specific set of cars. In our case, there were a set of 40 cars moving through the processes of being used and depleting charge, arriving at the chargers, waiting in queue if necessary, then charging and returning back to being used once fully charged.

As a result, the final simulation algorithm was a Time-Driven Closed-Network Queuing Simulation. The final simulation algorithm is included, with comments, in Appendix 4.7. A flowchart of this final simulation is depicted in Figure 17, and Section 2.4 will discuss the full algorithm in more detail.

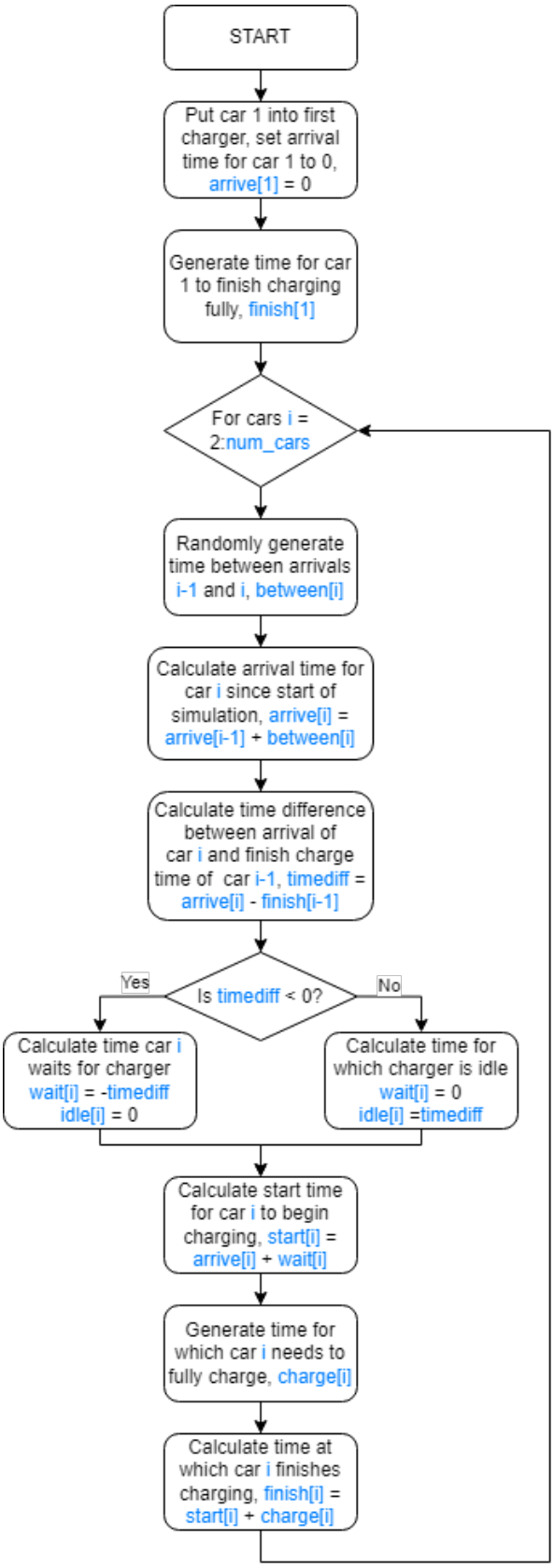

Figure 15: Event-Driven Open Network Queuing Simulation

2.4 In-Depth Explanation of Final Simulation Algorithm

As stated before, our final simulation was a time-driven closed network queuing simulation, referred to as a TDCN queuing simulation. Rather than having time move forward as determined by arrival times, at each unit of time, we check if an event occurs, then increment time by one, hence the term "time-driven". This simulation relied on four nested functions, each of which will be described below:

isnight function: This function informs us if a given time is during the hours of inactivity for the chargers.

Input: t (timestamp in minutes), n (number of hours of inactivity)

Output: 1 if t is within the period of inactivity, 0 if not

Step 1: Calculate nextstart, the timestamp of start of active period for next day (in minutes).

nextstart $= \lceil \frac{t}{24 \cdot 60} \rceil \cdot 24 \cdot 60$

Step 2: Calculate time (in minutes) until start of active period for next day.

timetillnext = nextstart $- t$

Step 3: If the time till the start of the next active period is less than n hours, but more than 0 minutes, we are in the inactive period.

if timetillnext $< n \cdot 60$ **AND** timetillnext > 0:

return 1

Step 4: Else, we are in an active period.

else return 0

generate_car function: This function generates the timestamp (in minutes) at which a given car will leave the charger.

Input: t (timestamp in minutes), car_name (identity of car being charged), n (number of hours of inactivity)

Output: timestamp in minutes at which car is fully charged and can be retrieved from the charger

Step 1: Generate charge finish time for car based on car name, without considering whether car can be retrieved at that time.

chargetime $= t +$ car_profile[car_name]

Step 2: Check if car can be retrieved at that time.

inactive = **isnight**(chargetime, n)

Step 3: If car cannot be retrieved then, change retrieval time to earliest possible time.

if inactive $= 1$:

return $\lceil \frac{t}{24 \cdot 60} \rceil \cdot 24 \cdot 60$

Step 4: If car can be retrieved then, return that time.
else return chargetime

The remaining function is the charger selection function, **select_charger**, which is described in Figure 16. The function first calls the time of arrival of the car. If the car would arrive during nighttime, it is instead supposed to arrive at the beginning of the next day, so the function returns a -1, effectively adding the car to the end of the queue (the queue is checked at the start of each day so that cars that would start charging first thing in the morning can do so). If the car would arrive during daytime, we check each charger to see if it is occupied. The value of the first unoccupied charger is returned. If no charger is unoccupied, the function returns -1, and so the car is added to the end of the queue.

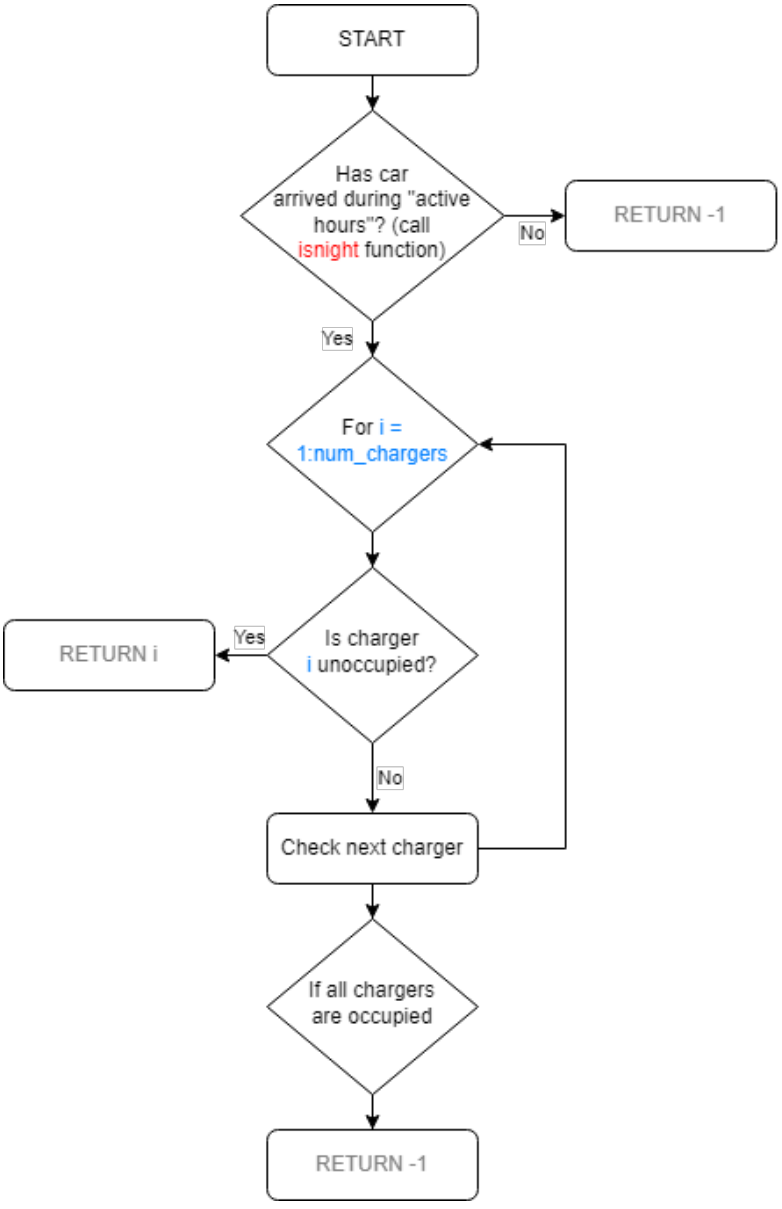

Figure 16: Charger Selection Function for TDCN Queuing Simulation

These functions combine to form the overall simulation. The initial arrival times of all cars are generated (**arrivals**), and we find the soonest arrival time, *time_pointer*, of a car by taking the min of all arrival times. Then we set the time to equal 0 and run the following loop while time is less than timespan, the length of time for which the simulation runs. At each time unit, four processes are run in order.

1. (Orange) We check whether it is the start of a new day. If it is and any cars would have arrived the previous night, they instead arrive at this point. Cars that arrive at night go into the queue based on their order of arrival during the night. We check the queue to see which cars arrived the previous night and occupy all chargers possible (**select_charger**, **generate_car**), removing these cars from the queue. As soon as no chargers are available, we move to the next process.

2. (Green) We check all chargers to see if any cars just finished charging. If any have, we remove them from the charger and generate their next arrival time based on their charging frequency. Once this charger is unoccupied, we check if there are any cars in the queue. If there are, the frontmost car occupies this charger and its finish time is generated based on which model it is (**generate_car**). Note that the finish time of a car factors in whether it finishes charging at night or during the day. This car is removed from the queue. After we have checked to see if all fully-charged cars have been dealt with, we move on to the next process.

3. (Purple) Finally, we check if any cars have arrived by checking if the current time is greater than *time_pointer*, which was the min of all arrival times. If the time is greater than *time_pointer*, we check which car is arriving and select a charger or enter the queue (**select_charger**) based on availability and time of day. If we occupy a charger, we generate finish charge time (**generate_car**). As soon as a car arrives, its next arrival time is set to the simulation time length + 1 so that it is not the minimum arrival time value. This is so that we avoid a case where a car is slated to arrive while it is still charging; generally, setting next arrival time to any very large value would work. At this point, we change *time_pointer* to equal the next soonest arrival time, determined by taking the min of all arrival times.

4. Increase time by 1.

We run Steps 1-4 until the simulation has run through the desired length of time. This process is detailed in Figure 17.

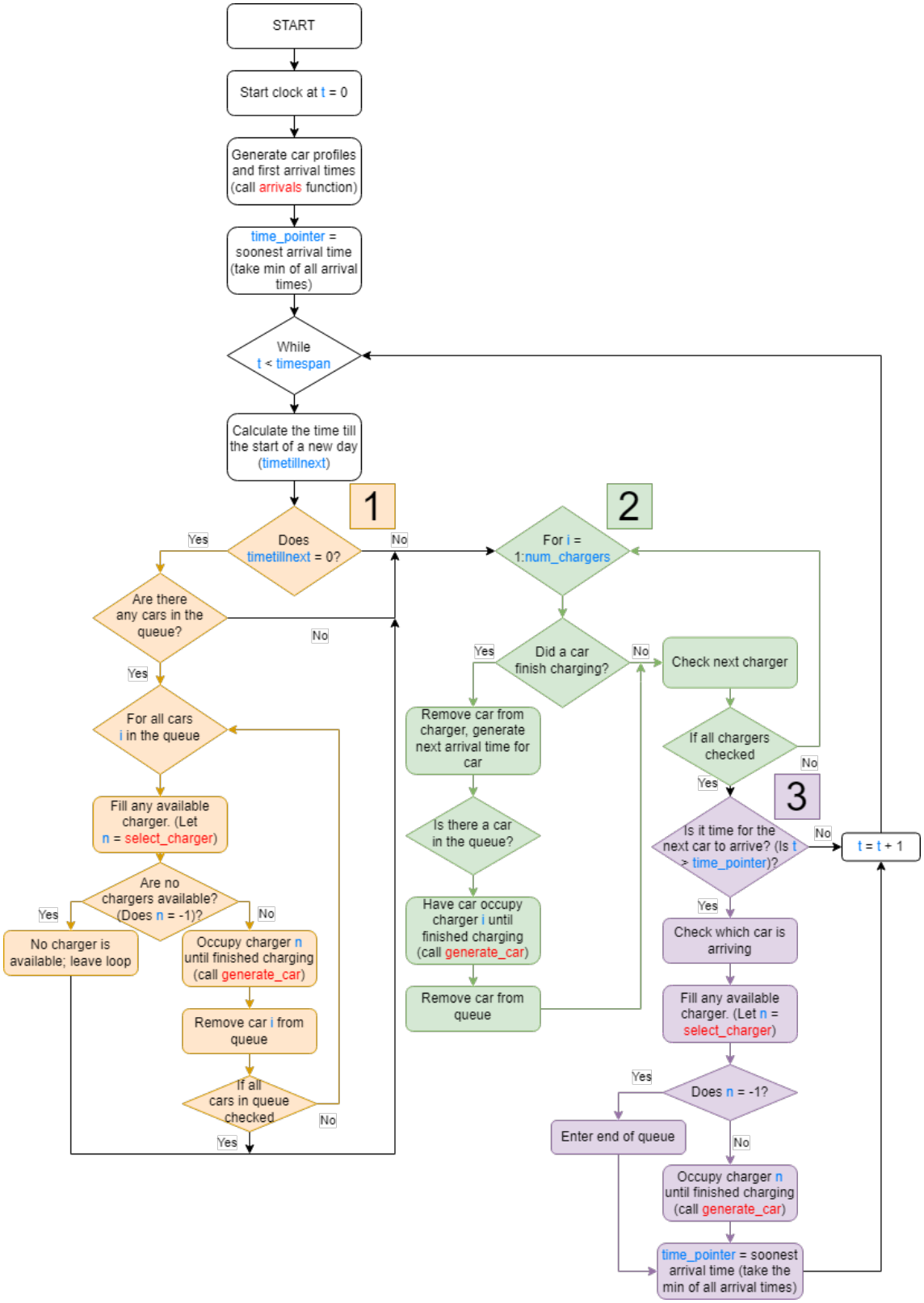

Figure 17: Time-Driven Closed-Network Queuing Simulation Flowchart

2.5 Debugging

We used three visual representations of our results to debug the simulation.

Our first visual, Figure 18, displayed activity at each of the chargers over time. At each minute in the simulation, we plotted whether the charger was occupied, and if it was, which car was occupying the charger. That way we were able to check when each car began and finished charging. When we incorporated inactivity periods for the chargers, we made sure that no car finished charging during the periods of inactivity. We were also able to verify that no car was charging in two chargers at once, and were able to verify that if a car was charging, the charger was not also marked as being unoccupied. In Figure 18,the blue lines represent whether it is nighttime or not. If the blue line is nonzero, it is nighttime, and if the blue line is zero, it is daytime. If the other lines are positive, their values correspond to which car is charging at that time. For example, if a line has value X with $X > 0$, car X is being charged. If their values are negative, then that charger is unoccupied. For example, if a line has value Y with $Y < 0$, charger Y is unoccupied.

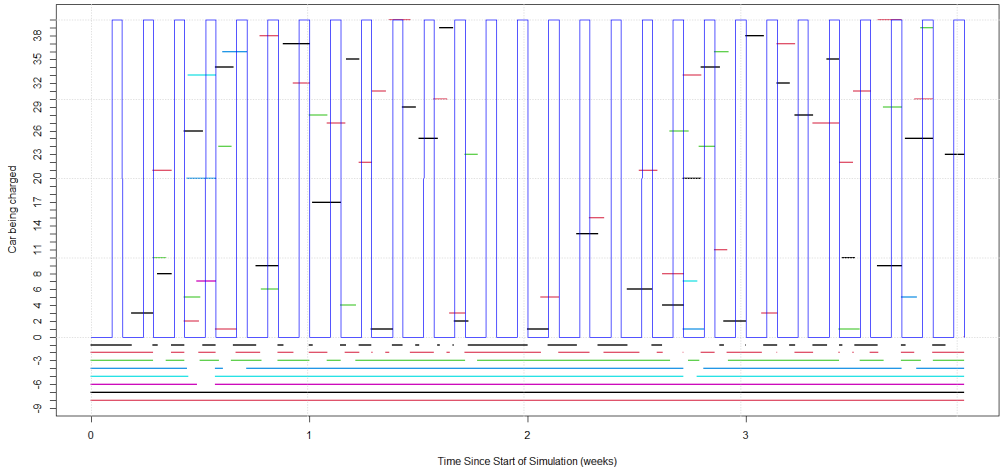

Figure 18: Plot of Charger Activity Over Time

Our second visual, Figure 19, graphed, for each car in the queue, the time spent waiting in the queue versus the time since the start of the simulation. We plotted this superimposed with the length of the queue at each minute. This allowed us to verify that the queue portion of our algorithm was working correctly. We can use this visual to note that for these parameters, all waiting time occurred during the nighttime, indicating that there was no charger back up outside of night hours.

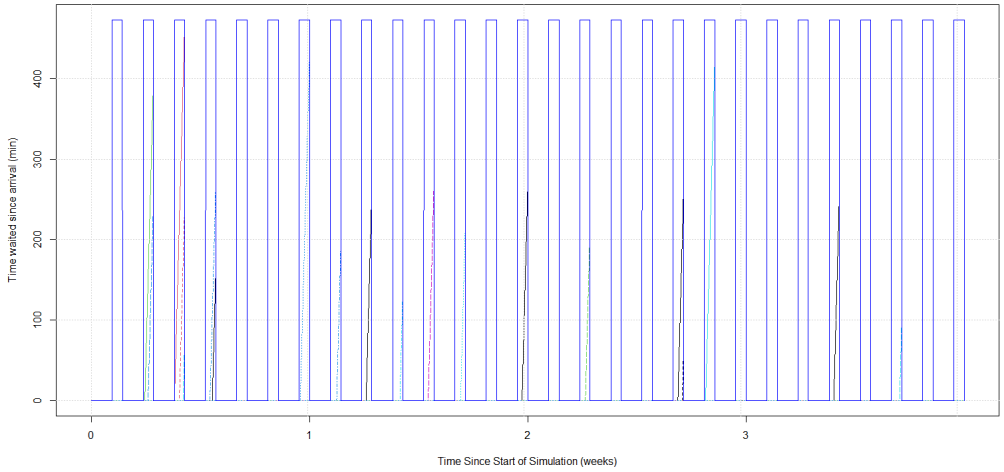

Figure 19: Plot of Minutes Waited Since Arrival

Our third visual, Figure 20, graphed, for each charger, the amount of time left until the charger would be unoccupied. This allowed us to make sure that cars finished charging fully before they left, and that charger selection prioritization was working systematically, as intended.

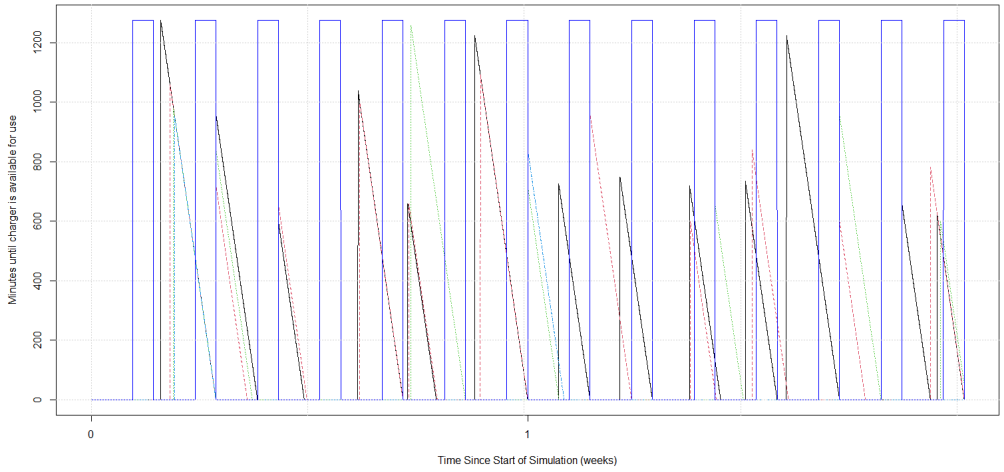

Figure 20: Plot of Minutes Left Until Charger Unoccupied

The main issues we ran into while debugging had to do with very small coding errors. For example, defining something based on a variable created within a function, or missing a set of parentheses when defining a range of values. These tools allowed us to catch those errors very quickly.

2.6 Results

We now examine our 40-car fleet's usage patterns (Section 2.2), based on the usage detailed in Table 1. We compare average and maximum wait times (Table 2), along with usage amounts by charger (Table 3), for 10, 6, 5, and 4 charger systems

with nighttime lengths of 0, 8, and 12 hours. We run each simulation over six months. Number of cars charged over this length of time does not vary much with changing parameters, so we did not post these results. Because we are using randomly-generated initial arrival times, we posted the results for each combination of parameters averaged over 20 runs.

	0 hours of nighttime	**8 hours of nighttime**	**12 hours of nighttime**
10 chargers	(0,0)	(36.8, 473.7)	(65.3, 679.5)
6 chargers	(0.2, 37.1)	(49.1, 483.7)	(72.0, 700.9)
5 chargers	(0.9, 186.5)	(48.5, 559.4)	(71.6, 797.0)
4 chargers	(5.4, 346.5)	(52.3, 676.3)	(89.2, 1018.8)

Table 2: Average Wait Time, Maximum Wait Time (minutes)
Over 20 Six-Month Runs of TDCN Queuing Simulation

	0 hours of nighttime	**8 hours of nighttime**	**12 hours of nighttime**
10 chargers	55.75%	60.60%	67.76%
	37.70%	42.58%	48.35%
	20.66%	24.85%	29.61%
	8.52%	11.20%	15.67%
	2.73%	4.01%	6.60%
	0.64%	0.88%	2.07%
	0.14%	0.14%	0.68%
	0.00%	0.02%	0.16%
	0.00%	0.00%	0.04%
	0.00%	0.00%	0.00%
6 chargers	56.48%	59.05%	68.22%
	37.76%	41.97%	47.92%
	20.51%	24.53%	30.24%
	8.16%	11.58%	15.03%
	2.47%	4.24%	6.50%
	0.89%	1.27%	2.15%
5 chargers	56.56%	58.88%	68.05%
	37.95%	42.40%	48.63%
	20.73%	25.33%	30.98%
	8.09%	11.66%	15.83%
	2.48%	4.14%	6.37%
4 chargers	57.42%	61.25%	69.03%
	38.56%	44.13%	50.31%
	21.42%	25.68%	32.45%
	8.75%	11.43%	17.37%

Table 3: Average Percent of Time Each Charger Was Used Over 20 Six-Month Runs of TDCN Queuing Simulation

For 10 chargers, we find that for all tested nighttime lengths, only 6 out of 10 of the chargers are used for more than 1 percent of the total time. As a result, we explored how wait times and usage amounts looked for 6, 5, and 4 chargers. We find that there is a tipping point between 5 and 4 chargers, where the average wait time for a 0-hour nighttime increases from 0.9 minutes to 5.4 minutes, blowing up by a factor of six. This is indicative of a major pileup problem between 5 and 4 chargers. Furthermore, looking at the 12-hour case for 5 and 4 chargers, we see that the maximum wait time increases from 797.0 minutes to 1018.8 minutes, an increase of around 3.5 hours, when previous wait times increased by only around 1.5 hours. This tells us that 4 chargers is far too few to support the College's needs. However, we see relatively small differences between 6 and 5 chargers in the 8 and 12 hour nighttime length scenarios, meaning that around 5 or 6 chargers may be suitable to meet the College's needs. We also want to note that for 10 and 6 charger systems, the maximum wait time closely matches up with the rest length, and so the wait time is likely due to that. We would want to discuss more with the drivers of College vehicles to get a better idea of what wait times are acceptable.

2.7 Questions for Further Exploration

There are some further refinements we would like to make to the simulation, outlined below:

1. This model could be refined further by taking into account consistency of usage. For example, an athletics rental vehicle would be used sparsely, but would be used for very long trips, whereas the post office van would be used consistently for small distances daily.

2. Introduce level 3 chargers, which charge rapidly from 20% to 80% of battery capacity, as another charger option. Also introduce a method, perhaps linked with priority, of choosing which charger to use when both are available. This way highly-used campus safety vans, for example, have priority to use faster chargers so that they may be used for long stretches of each day.

3. Make car day-to-day usage have a partially random element, and occur in a range of values rather than a fixed value.

3 Final Recommendation

This work clearly shows us that under current usage assumptions, switching all of Bryn Mawr's non-bus vehicles to their electric counterparts could be supported by just 6 Level 2 chargers, and that the sixth charger would be available quite often, making it a possible candidate for public use.

Bryn Mawr College should move towards replacing their vehicle fleet, as the present vehicles age out, with their equivalent electric vehicle alternatives, and support them with an electric behicle charging network of 6 chargers.

4 Appendix

4.1 Coefficient Matrix for System of Equations

```
# finding r and initializing coefficient and constant
   vectors
r = length(bT);
A = zeros(r^2,r^2);
b = zeros(r^2,1);

  # "center" head equation generation
  for i = 0:(r-3)
    for j = ((i*r)+(r+2)):((i*r)+(2*r-1))
      A(j,j-r) = -1; # top
      A(j,j-1) = -1; # left
      A(j,j) = 4; # self
      A(j,j+1) = -1; # right
      A(j,j+r) = -1; # bottom
    endfor
  endfor

  # top boundary head equation generation, excluding
     corners
  for j = 2:(r-1)
    b(j) = bT(j);
    A(j,j-1) = -1; # left
      A(j,j) = 4; # self
      A(j,j+1) = -1; # right
      A(j,j+r) = -1; # bottom
  endfor

  # bottom boundary head equation generation, excluding
     corners
  for j = (r^2-r+2):(r^2-1)
    b(j) = bB(j-(r^2-r+2)+2);
    A(j,j-r) = -1; # top
    A(j,j-1) = -1; # left
    A(j,j) = 4; # self
    A(j,j+1) = -1; # right
```

```
endfor

# left boundary head equation generation, excluding
   corners
for j = (1+r):r:(r^2-2*r+1)
  b(j) = bL(((j-1)/r)+1);
  A(j,j-r) = -1; # top
  A(j,j) = 4; # self
  A(j,j+1) = -1; # right
  A(j,j+r) = -1; # bottom
endfor

# right boundary head equation generation, excluding
   corners
for j = (2*r):r:(r^2-r)
  b(j) = bR(j/r);
  A(j,j-r) = -1; # top
  A(j,j-1) = -1; # left
  A(j,j) = 4; # self
  A(j,j+r) = -1; # bottom
endfor

# top left corner head equation generation
b(1) = bT(1)+bL(1);
A(1,1) = 4; # self
A(1,2) = -1; # right
A(1,1+r) = -1; # bottom

# top right corner head equation generation
b(r) = bT(r)+bR(1);
A(r,r) = 4; # self
A(r,r-1) = -1; # left
A(r,2*r) = -1; # bottom

# bottom left corner head equation generation
b(r^2-r+1) = bL(r)+bB(1);
A(r^2-r+1,r^2-r+1) = 4; # self
A(r^2-r+1,r^2-r+1-r) = -1; # top
A(r^2-r+1,r^2-r+2) = -1; # right
```

```
# bottom left corner head equation generation
b(r^2) = bR(r)+bB(r);
A(r^2,r^2) = 4; # self
A(r^2,r^2-r) = -1; # top
A(r^2,r^2-1) = -1; # left
```

4.2 Solver and Analysis for Jacobi Method

```
clc
format short

function heads = JacobiVals(bT,bB,bL,bR,errbound)
  # computing exact values
  A;
  b;
  headval = A\b;

  # calculating iteration matrix for Jacobi B
  # & displaying its eigenvalues
  B = inv(diag(diag(A)))*-1*(tril(A,-1)+triu(A,1));
  eigenB = eig(B);
  maxeigenB = max(abs(eigenB));

  # for loop for examining convergence of Jacobi method
  btilde = inv(diag(diag(A)))*b;
  errval = 1;
  erratio = [];
  errvec = [errval];
  iterations = 0;
  approxhead = zeros(r^2,1);
  while errval > errbound
    approxhead = B*approxhead + btilde;
    iterations = iterations + 1;
    errval = norm(approxhead-headval);
    errvec = [errvec errval];
    erratio = [erratio errval/errvec(iterations)];
  endwhile

  # compare ratio of errors vs max magnitude eigenvalue
     of B
  erratio = erratio(2:end);
  hold on
  plot(1:length(erratio),erratio)
  plot(1:length(erratio), maxeigenB*ones(1,length(erratio
     )))
```

```
  legend('Ratio of error over time','Maximum magnitude
     eigenvalue of B')
  hold off


  # OPTIONAL: displaying exact head values
  # (can be replaced w/ approximate head values) in a
     grid
  heads = zeros(r,r);

  for i = 1:r
    for j = 1:r
      plusval = mod(j,r);
      if plusval == 0
        plusval = 5;
      endif
      heads(i,j) = headval(((i-1)*r)+plusval);
    endfor
  endfor
  disp(["The total number of iterations needed to get to
     an error of ",
  num2str(errbound)," from an initial guess of all zeros
     was ",
  num2str(iterations),"."])
endfunction

bound_top = [260 280 300 315 340];
bound_bottom = [210 225 245 270 300];
bound_left = [230 220 215 210 205];
bound_right = [360 355 350 345 340];

actualval = JacobiVals(bound_top,bound_bottom,bound_left,
   bound_right,.001)
```

4.3 Solver and Analysis for Gauss-Seidel Method

```
clc
format short

function heads = GaussSeidelVals(bT,bB,bL,bR,errbound)
  # computing exact values
  A;
  b;
  headval = A\b;

  # calculating iteration matrix for Gauss-Seidel B &
     displaying its eigenvalues
  B = inv((tril(A,-1)+diag(diag(A))))*-1*triu(A,1);
  eigenB = eig(B);
  maxeigenB = max(abs(eigenB));
  eig(B);
  B
  # for loop for examining convergence of Gauss-Seidel
     method
  btilde = inv((tril(A,-1)+diag(diag(A))))*b;
  errval = 1;
  erratio = [];
  errvec = [errval];
  iterations = 0;
  approxhead = zeros(r^2,1);
  while errval > errbound
    approxhead = B*approxhead + btilde;
    iterations = iterations + 1;
    errval = norm(approxhead-headval);
    errvec = [errvec errval];
    erratio = [erratio errval/errvec(iterations)];
  endwhile

  # compare ratio of errors vs max magnitude eigenvalue
     of B
  erratio = erratio(2:end);
  hold on
  plot(1:length(erratio),erratio)
```

```
  plot(1:length(erratio), maxeigenB*ones(1,length(erratio
     )))
  legend('Ratio␣of␣error␣over␣time','Maximum␣magnitude␣
     eigenvalue␣of␣B')
  hold off


  # OPTIONAL: displaying exact head values
  # (can be replaced w/ approximate head values) in a
     grid
  heads = zeros(r,r);

  for i = 1:r
    for j = 1:r
      plusval = mod(j,r);
      if plusval == 0
        plusval = 5;
      endif
      heads(i,j) = headval(((i-1)*r)+plusval);
    endfor
  endfor
  disp(["The␣total␣number␣of␣iterations␣needed␣to␣get␣to␣
     an␣error␣of␣",
  num2str(errbound),"␣from␣an␣initial␣guess␣of␣all␣zeros␣
     was␣",
  num2str(iterations),"."])
endfunction

bound_top = [260 280 300 315 340];
bound_bottom = [210 225 245 270 300];
bound_left = [230 220 215 210 205];
bound_right = [360 355 350 345 340];

actualval = GaussSeidelVals(bound_top,bound_bottom,
bound_left,bound_right,.01)
```

4.4 Results for Jacobi Solver for 25-well Example

```
head_values =
   251.21   272.03   292.62   312.78   335.92
   242.82   264.29   285.67   307.57   330.88
   235.77   256.63   278.22   300.96   325.05
   228.62   248.26   269.61   293.00   318.35
   220.45   238.17   258.96   283.08   310.36
max Btilde eigenvalue =
0.8660
Number of iterations to get to .01 total error from initial guess of
all zeros =
82
```

4.5 Results for Gauss-Seidel Solver for 25-well Example

```
head_values =
   251.21   272.03   292.62   312.78   335.92
   242.82   264.29   285.67   307.57   330.88
   235.77   256.63   278.22   300.96   325.05
   228.62   248.26   269.61   293.00   318.35
   220.45   238.17   258.96   283.08   310.36
max Btilde eigenvalue =
0.75
Number of iterations to get to .01 total error from initial guess of
all zeros =
42
```

4.6 Electric Car Alternatives for Bryn Mawr Vehicles

Car Type	**Alternative Model**	**Link to Specs**
8-person Minivan	Volkswagen ID Buzz	topelectricsuv.com/news/volkswagen/vw-id-buzz-update/
Passenger Van	Lightning Electric Passenger Van	lightningemotors.com/lightningelectric-ford-transit-shuttle/
Cargo Van	Ford 2022 Cargo Van	ford.com/commercial-trucks/e-transit/models/cargo-van
SUV	Subaru Solterra	topelectricsuv.com/news/subaru/subaru-electric-car-fresh-details
Pickup Truck	Ford F150 Lightning	ford.com/trucks/f150/f150-lightning/
High Roof	Ford 2022 Cargo Van (different height from Cargo Van)	ford.com/commercial-trucks/e-transit/models/cargo-van

4.7 Time-Driven Closed Network Queuing Simulation Code

```
# SIMULATION PARAMETERS
timespan =  6*28*24*60 # length of time for which
   simulation runs (minutes)
num_chargers = 10 # number of chargers
restlength = 8 # number of hours chargers are "inactive"
num_cars = 40
weekvec= seq(1,(timespan/(24*60)),by=7)

# NESTED FUNCTIONS
isnight_function = function(ct,n) {
  ns = ceiling(ct/(24*60))*24*60
  ttn = ns - ct
  if (ttn < n*60 && ttn > 0){
    return(1)
  } else{
    return(0)
  }
}


arrivalsvector = function() { # generate vector of car
   arrivals
  carnames = c(1:num_cars) # number/'name' of car
  frequency_charge = c(ENTER VALUES FROM TABLE) #
     frequency of charge (minutes)
  charging_length = c(ENTER VALUES FROM TABLE) # time
     needed to fully charge (minutes)
  for (i in 1:num_cars){
    arrival_timestamp[i] = runif(1,0,frequency_charge[i])
       # randomly generate first arrival time
  }
  # arrival_timestamp = frequency_charge # time of first
     arrival from start of clock (in minutes) -- this
     changes as the simulation runs
  arrivals = array(c(carnames,frequency_charge,charging_
     length,arrival_timestamp),dim=c(num_cars,4)) #
     compile into one array
```

```
  return(arrivals)
}


generate_car = function(car_name,arrvec,time) { #
   generate car profile/car charge finish time
  charge_time = time + arrvec[car_name,3] # pull time to
     fully charge from vector of charging times by car
  # check if in active hours -- if not, adjust finish
     time
  nextstart = ceiling(charge_time/(24*60))*24*60 #
     calculate time of start of next day (when finished
     charging)
  timetillnext = nextstart - charge_time # calculate time
      until start of next day
  if (timetillnext <= restlength*60-1) { # if less than
     restlength hours left (i.e. if chargers are inactive
     )
    return(nextstart) # then we add the car to the queue
  }
  return(charge_time)
}


select_charger = function(chargvec,time) { # select which
    charger for a car to use, if available

  # check if in arrival hours -- if not, add to queue
  nextstart = ceiling(time/(24*60))*24*60 # calculate
     time of start of next day
  timetillnext = nextstart - time # calculate time until
     start of next day
  if (timetillnext <= restlength*60-1 && timetillnext>0)
     { # if less than restlength hours left (i.e. if
     chargers are inactive)
    return(-1) # then we add the car to the queue
  }

  # else if we are within arrival hours, check for
```

```
    available chargers
  for (i in 1:dim(chargvec)[1]) { # go through list of
    chargers
    if (chargvec[i,1] == 0) { # if unoccupied
      return(i) # select charger
    }
  }
  return(-1) # else return -1 for no chargers available
}
# END OF FUNCTIONS

# START CLOCK
t = 0 # start clock at time = 0

# GENERATE ARRIVAL TIMES
arrivals = arrivalsvector() # generate car arrival times
# arrivals = arrivalscopy

# INITIALIZE POINTERS and QUEUES
present_car_arrival = arrivals[which.min(arrivals[,4]),1]
    # initialize name of current car that has arrived
arrivals_time_pointer = arrivals[present_car_arrival,4] #
    initialize time pointer for car arrivals (time after
   which to check for arrivals)
arrival_number = present_car_arrival # initialize "
   arrival index pointer" = which arrival (in terms of
   ordering) we are considering next

chargers = array(c(numeric(num_chargers),numeric(num_
   chargers)),dim=c(num_chargers,2)) # initialize charger
    availability (and which car is at which charger)

charger_queue = array(c(numeric(1000),numeric(1000)),dim=
   c(1000,2)) # initialize queue for chargers
charger_queue_pointer = 1 # initialize queue pointer at 1
   st entry in queue
charger_queue_length = 0 # initialize value of number of
   cars waiting in queue
first_car_in_queue = 0 # initialize name of first car in
```

```
   queue

isnight = numeric(timespan) # initialize night switch

nextstart = 0
timetillnext = 0

# INITIALIZE OUTPUTS
CHARGETIME = numeric(num_chargers) # total time chargers
   are in use
cars_served = 0 # total number of cars charged
wait_time = 0 # average wait time per car
carplot = array(numeric(timespan),dim=c(timespan,num_
   chargers)) # array to store which car is in which
   charger at time t
queueplot = numeric(timespan) # array to store number of
   cars in queue at time t
waitplot = array(numeric(timespan),dim=c(timespan,num_
   cars)) # array to store how long cars are waiting
chargerplot = array(numeric(timespan),dim=c(timespan,num_
   chargers)) # array to store how long until chargers
   are free next
queuestartplot = numeric(timespan)

# SIMULATION START

while (t < timespan) { # while simulation is running
  nextstart = ceiling(t/(24*60))*24*60 # calculate time
     of start of next day
  timetillnext = nextstart - t # calculate time until
     start of next day
  if (timetillnext <= restlength*60-1 && timetillnext>0)
     { # if less than restlength hours left (i.e. if
     chargers are inactive)
    isnight[t] = 1 # then we are in a period of
       inactivity
  }

  if (timetillnext == 0) { # if it is the start of a new
```

```
    day
  if (charger_queue_length > 0) { # if there is a queue
    for (i in charger_queue_pointer:(charger_queue_
       pointer+charger_queue_length-1)) { # for each
       car in the queue
      available_charger = select_charger(chargers,t)
      if (available_charger != -1) { # if a charger is
         available
        chargers[available_charger,1] = generate_car(
           charger_queue[i,1],arrivals,t) # occupy the
           free charger
        chargers[available_charger,2] = charger_queue[i
           ,1] # note which car is occupying the
           charger
        wait_time = wait_time + t - charger_queue[i,2]
           # calculate wait time based on current time
           minus time car started charging
        charger_queue_pointer = charger_queue_pointer +
            1 # consider next spot in queue and later (
           i.e. "remove" from queue)
        charger_queue_length = charger_queue_length - 1
            # reduce queue length by 1
        CHARGETIME[available_charger] = CHARGETIME[
           available_charger] + chargers[available_
           charger,1] - t # add to total time charger
           is in use
      }
    }
  }
}

for (i in 1:num_chargers) { # check if any cars are
    done charging
  if (chargers[i,1] != 0) { # among chargers that are
     occupied
    if (t >= chargers[i,1]){ # if any cars are done
       charging
      present_car_arrival = chargers[i,2] # focus on
         car that just finished charging
```

```
      chargers[i,1] = 0 # make charger available
      arrivals[present_car_arrival,4] = t + arrivals[
         present_car_arrival,2] # calculate time of
         next needed charge
      chargers[i,2] = 0 # clear out car in that charger
          slot
      arrival_number = arrivals[which.min(arrivals[,4])
         ,1] # move arrival index pointer to next car
      arrivals_time_pointer = arrivals[arrival_number
         ,4] # move arrival timestamp pointer to next
         car's arrival
      cars_served = cars_served + 1

      if (charger_queue_length > 0) { # if there is a
         queue
        first_car_in_queue = charger_queue[charger_
           queue_pointer,1] # note down name of first
           car in queue
        chargers[i,1] = generate_car(first_car_in_queue
           ,arrivals,t) # occupy charger until specific
            time t, based on name of current car
        chargers[i,2] = first_car_in_queue # note which
            car is in this charger
        wait_time = wait_time + t - charger_queue[
           charger_queue_pointer,2] # since a car
           begins charging from the queue, add to total
            wait time by taking difference of arrival
           time and current time (time waiting in line
           before charging)
        charger_queue_pointer = charger_queue_pointer +
            1 # consider next spot in queue and later (
           i.e. "remove" from queue)
        charger_queue_length = charger_queue_length - 1
            # reduce queue length by 1
        CHARGETIME[i] = CHARGETIME[i] + chargers[i,1] -
            t # add to total time charger is in use
      }
    }
  }
```

```
}



if (t >= arrivals_time_pointer) { # check if a car is
   arriving
  present_car_arrival = arrival_number # set current
     car being considered to the car that just arrived
  available_charger = select_charger(chargers,t) #
     select a charger for the car
  if (available_charger != -1) { # if a charger is
     available
    chargers[available_charger,1] = generate_car(
       present_car_arrival,arrivals,t) # generate time
       of finishing charging (based on car that will
       begin charging), from start of clock, occupy
       available charger
    chargers[available_charger,2] = present_car_arrival
    CHARGETIME[available_charger] = CHARGETIME[
       available_charger] + chargers[available_charger]
        - t # add to total time charger is in use
  }
  if (available_charger == -1) { # if no chargers are
     available
    charger_queue[charger_queue_pointer+charger_queue_
       length,1] = present_car_arrival # add to end of
       queue the name of car that just arrived
    charger_queue[charger_queue_pointer+charger_queue_
       length,2] = t # note arrival time of car in
       queue
    charger_queue_length = charger_queue_length + 1
  }
  arrivals[present_car_arrival,4] = timespan + 1 #
     while charging, remove this car from consideration
      for arrival for charging
  arrival_number = arrivals[which.min(arrivals[,4]),1]
     # initialize name of current car that has arrived
  arrivals_time_pointer = arrivals[arrival_number,4] #
```

```
      now that  car has arrived and been sorted, move
      arrival timestamp pointer to next car's arrival
  }
  if (charger_queue_length > 0){
    for (i in charger_queue_pointer:(charger_queue_
      pointer+charger_queue_length-1)) { # for each car
      in the queue
      waitplot[t,charger_queue[i,1]]=t-charger_queue[i,2]
          # plot time waited so far
    }
  }
  queuestartplot[t] = charger_queue_pointer

  for (i in 1:num_chargers) {
    carplot[t,i] = chargers[i,2] # for all chargers note
      which car is charging
    if (chargers[i,2] == 0) { # if a charger isnt being
      used
      carplot[t,i] = -i # note that it isnt being used
    }
    chargerplot[t,i] = max(0,chargers[i,1]-t)
  }
  queueplot[t] = charger_queue_length
  t = t+1 # a minute passes
}


for (i in 1:num_chargers) { # once clock is done running,
    calculate % of time each charger spent in use
  CHARGETIME[i] = CHARGETIME[i]/timespan
}
wait_time = wait_time/cars_served


print(CHARGETIME)
print(cars_served)
print(wait_time)
print(max(waitplot))
```

4.8 Results With Figures for Various Parameters

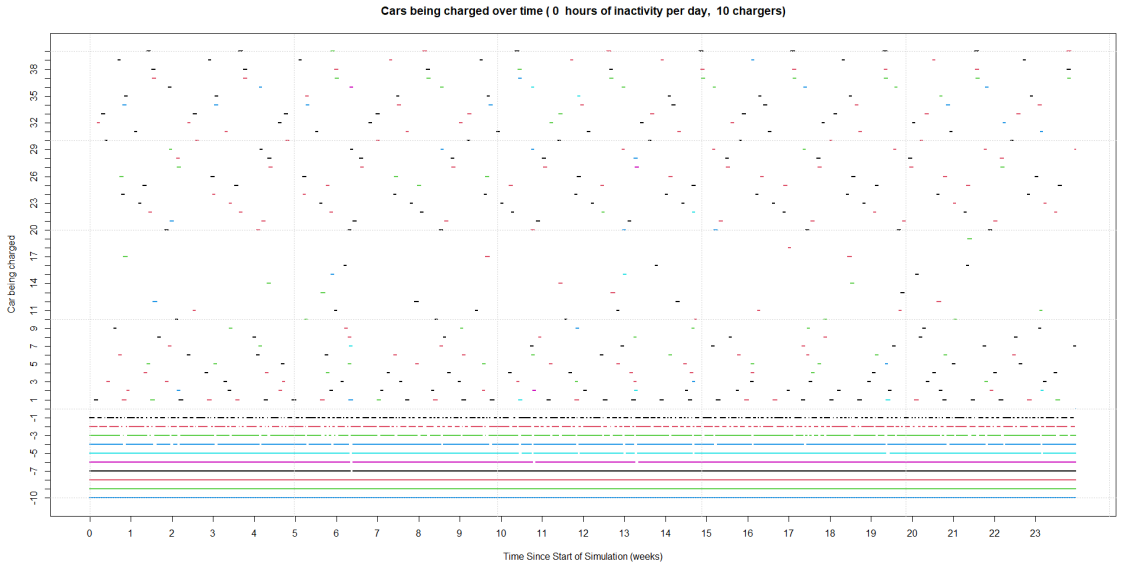

Figure 21: Usage Pattern for 10 Chargers with 0 Hours of Inactivity Daily

Percent of time charger was in use	55.28% 39.31% 18.20% 7.71% 2.65% 0.09% 0.00% 0.00% 0.00% 0.00%
Total number of cars charged	404
Average wait time per car (minutes)	0
Maximum wait time (minutes)	0

Table 4: Usage Data for 10 Chargers with 8 Hours of Inactivity Daily

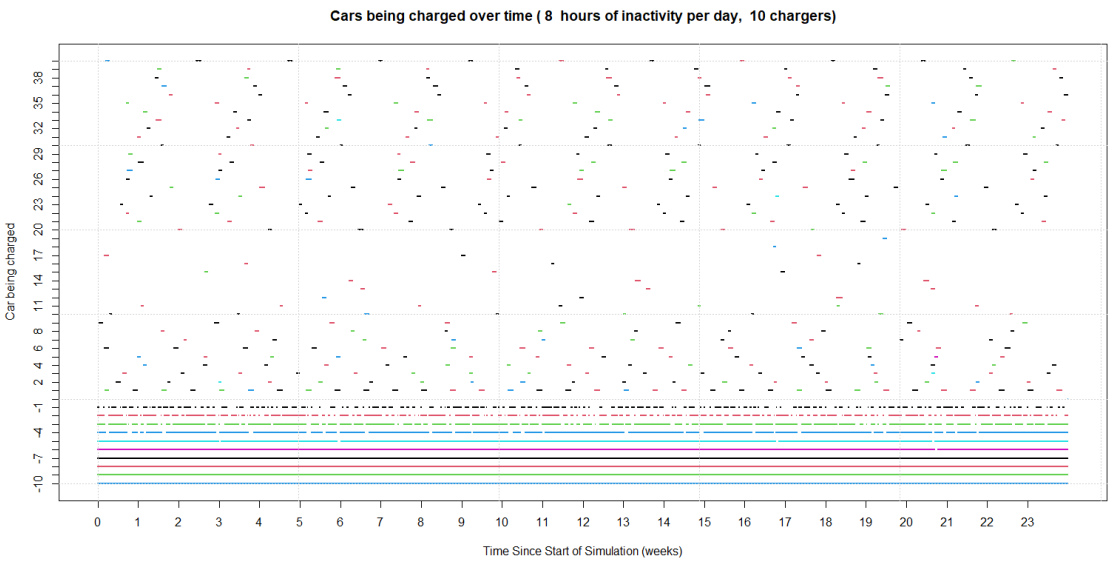

Figure 22: Usage Pattern for 10 Chargers with 8 Hours of Inactivity Daily

Percent of time charger was in use	63.19% 44.81% 25.33% 11.11% 01.26% 0.00% 0.00% 0.00% 0.00% 0.00%
Total number of cars charged	405
Average wait time per car (minutes)	22.6
Maximum wait time (minutes)	476

Table 5: Usage Data for 10 Chargers with 8 Hours of Inactivity Daily

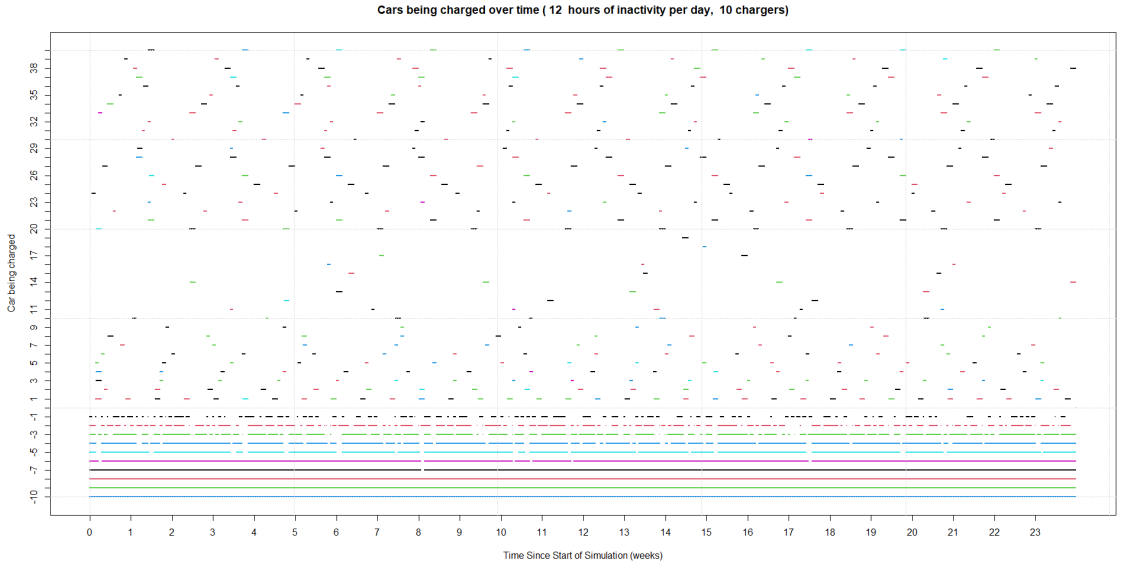

Figure 23: Usage Pattern for 10 Chargers with 12 Hours of Inactivity Daily

Percent of time charger was in use	68.28% 47.31% 29.35% 14.49% 8.16% 1.86% 0.00% 0.00% 0.00% 0.00%
Total number of cars charged	400
Average wait time per car (minutes)	65.55
Maximum wait time (minutes)	706

Table 6: Usage Data for 10 Chargers with 12 Hours of Inactivity Daily

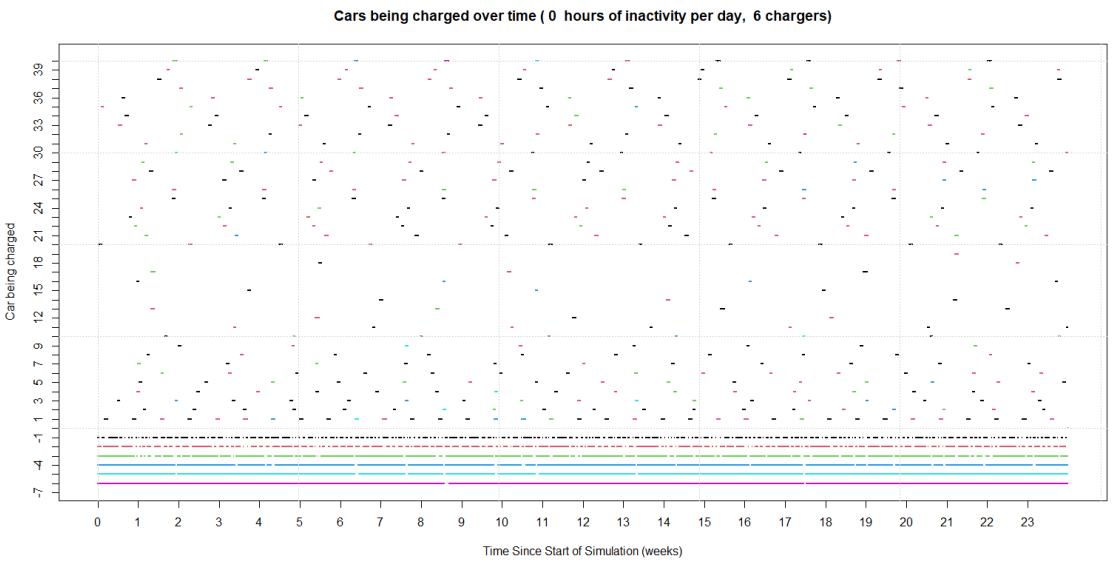

Figure 24: Usage Pattern for 6 Chargers with 0 Hours of Inactivity Daily

Percent of time charger was in use	58.90% 40.38% 16.27% 6.47% 2.41% 0.00%
Total number of cars charged	407
Average wait time per car (minutes)	0
Maximum wait time (minutes)	0

Table 7: Usage Data for 6 Chargers with 0 Hours of Inactivity Daily

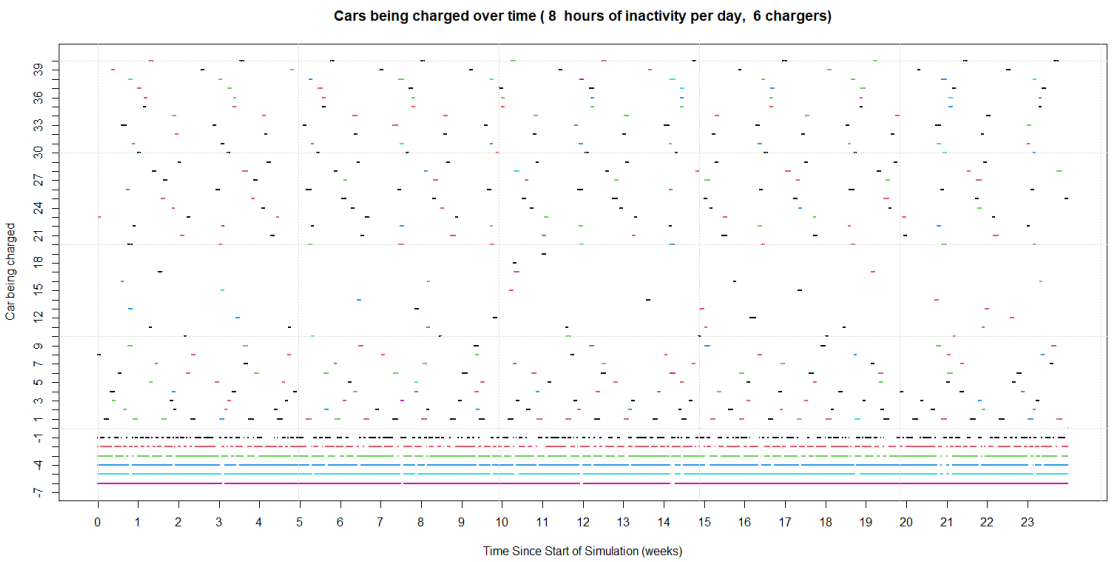

Figure 25: Usage Pattern for 6 Chargers with 8 Hours of Inactivity Daily

Percent of time charger was in use	62.98% 41.08% 23.46% 10.91% 5.47% 1.38%
Total number of cars charged	406
Average wait time per car (minutes)	26.8
Maximum wait time (minutes)	467

Table 8: Usage Data for 6 Chargers with 8 Hours of Inactivity Daily

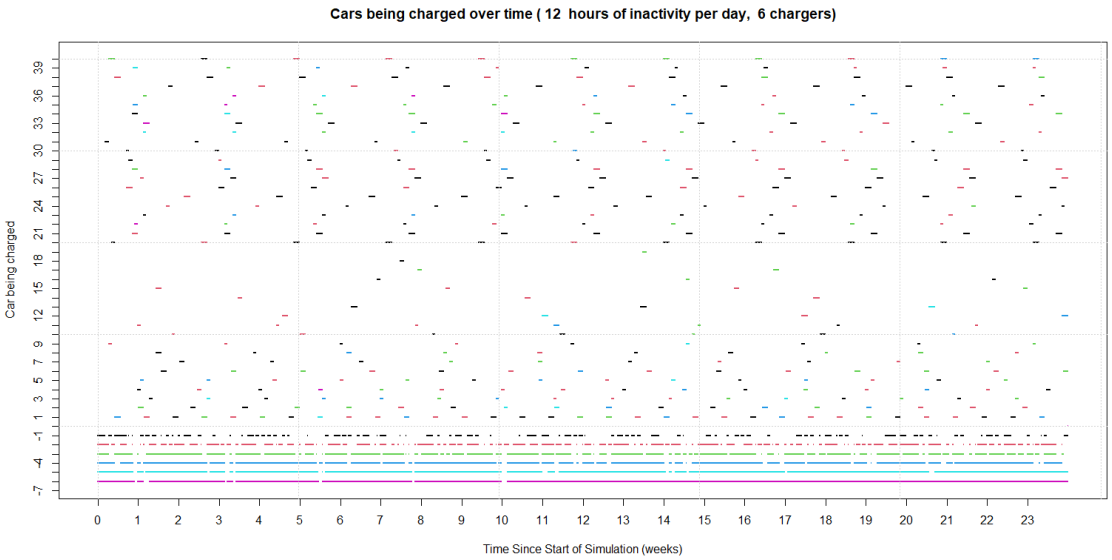

Figure 26: Usage Pattern for 6 Chargers with 12 Hours of Inactivity Daily

Percent of time charger was in use	70.91% 49.30% 27.75% 15.67% 6.53% 2.84%
Total number of cars charged	400
Average wait time per car (minutes)	51.78
Maximum wait time (minutes)	718

Table 9: Usage Data for 6 Chargers with 12 Hours of Inactivity Daily

References

Personal communication with Steve Green, Director of Transportation at Bryn Mawr College, 2022.

United States Environmental Protection Agency. Fast facts on transportation greenhouse gas emissions, 2019. URL https://www.epa.gov/greenvehicles/fast-facts-transportation-greenhouse-gas-emissions. (Last Accessed 04/24/2022).

United States Environmental Protection Agency. Sources of greenhouse gas emissions, 2020. URL https://www.epa.gov/ghgemissions/sources-greenhouse-gas-emissions. (Last Accessed 04/24/2022).

Bryn Mawr College. College announces plans to be carbon neutral by 2035, 2021. URL https://www.brynmawr.edu/news/college-announces-plans-be-carbon-neutral-2035. (Last Accessed 04/24/2022).

Frank R. Giordano et al. *"Appendix B": A First Course in Mathematical Modeling*. Cengage Learning, 2015.

Charles R. Hadlock. *Mathematical Modeling In The Environment*. MAA, 1998.

Intergovernmental Panel on Climate Change. Global warming of 1.5°c, 2019. URL https://www.ipcc.ch/site/assets/uploads/sites/2/2019/06/SR15_Full_Report_High_Res.pdf. (Last Accessed 04/24/2022).

David M. Strong. Iterative methods for solving ax = b - analysis of Jacobi and Gauss-Seidel methods, 2005. URL https://www.maa.org/press/periodicals/loci/joma/iterative-methods-for-solving-ax-b-introduction-to-the-module. (Last Accessed 04/12/2022).