KalmanQueue:
An Adaptive Approach to Virtual Queuing

February 10, 2004

Abstract

QuickPass is a virtual queuing system that allows some theme park customers to significantly cut down their waiting time by scheduling their ride in advance. The purpose of this paper is to propose innovative QuickPass systems that maximize customer enjoyment. We propose that only a small portion of customers can effectively use QuickPass, and a good system will maximize this group subject to the constraints that (1) regular users are not significantly affected and (2) maximum waiting time for QuickPass users is small. To build a solid foundation, we define and test a simple model for single line formation before addressing QuickPass. We then develop two QuickPass systems, GhostQueue and KalmanQueue. GhostQueue is an intuitive and simple system, but we find it would be far from optimal in practice. We then propose that the best model is one that adapts to its environment rather than one that tries to enforce rigid parameters. We implement KalmanQueue. KalmanQueue is a highly adaptive system that uses a Kalman filter to adjust the number of QuickPasses given today based on the maximum length of the QuickPass line yesterday, while filtering out random noise. We simulate the KalmanQueue system with a C++ program and randomized input from our line formation model. We find this system quickly converges to a nearly optimal solution subject to our constraints. It is, however, sensitive to some parameters. We discuss the effects of these findings on the expected effectiveness of the system in a real environment. We conclude that KalmanQueue is a good solution.

Contents

1 Introduction

Millions of people visit amusement parks every year. Waiting in long lines is one of the major reported complaints among large amusement parks, such as Disneyland. Recently, a new concept has been developed and tested in many major theme parks to reduce the waiting time for some visitors. The underlying idea is fairly simple: rather than wait in line for a popular ride, visitors purchase special tickets that tell them when to come back, and when they return at that time, they expect to wait in a very short line before going on the ride.

Many such systems have been tested and implemented in theme parks around the world, including QuickPass, FastPass, Freeway, Q-lo, Ticket-To-Ride, and others. Some have failed, to be replaced by augmented versions, and others have thrived. The appeal of these systems to amusement parks is two-fold: they increase people's enjoyment of the park, and they allow people to invest more money into the park while they are not waiting in line.

There have been several implementations of the idea. All of the systems we researched assumed the number of QuickPass users was small, and accounted for it either by restricting the total number of QuickPasses or having a certain number per hour during peak times. Also, joining the two lines has been an issue, some parks choosing to blend the two lines at some point before the attraction, and some actually at the ride. The system we develop will address these issues.

2 Plan of Attack

Our goal is to design a good QuickPass system, one that would maximize customer enjoyment. This is a very broad problem, and it is not immediately clear what "customer enjoyment" means. We must restate the problem mathematically by narrowing our focus and defining our goals in order to obtain a good model, as well as discuss what a good model should be.

- **Define Terms:** We must state a definition of "guest enjoyment" and explain what affects it, why, and how we will model it.
- **State Assumptions:** Discussion of our assumptions will enhance our understanding of QuickPass systems and allow us to restate the problem in a mathematical way.
- **Describe a Good Model:** An effective QuickPass system will have certain desirable characteristics. Describing these will steer our model in the right direction.

Once we have established our definitions, assumptions, and goals, we will present our models.

- **Line Formation Model:** Understanding line formation helps us describe general amusement park behavior. The QuickPass system is at its core

a line manipulation system, and we cannot hope to design it without understanding line formation first.

- **GhostQueue: A Simple Model:** We describe a simple approach to the QuickPass system. While it is an optimal solution in an error-free ideal world, if implemented its capacity is limited.
- **KalmanQueue: An Adaptive Algorithm:** We propose, model, and test an adaptive algorithm as a solution to the QuickPass system. We provide a simple implementation using a Kalman filter, and test it using randomly generated input. We then discuss the strengths and weaknesses of our specific implementation and of the model in general.

3 Increasing Enjoyment

- **Key Points:** Enjoyment - Fairness - Priority

Guest enjoyment is a subjective concept. However, if we keep our ideas about guest enjoyment simple, we can make useful assumptions just from common sense. Once we have described how guest enjoyment varies, we can introduce a qualitative measure of how to utilize it in our model, and justify our choice.

1. **People enjoy wandering freely more than they enjoy waiting in a line.**

   This is the basic assumption that makes virtual queuing potentially useful for increasing guest enjoyment. When not in line for a specific ride, guests are able to enjoy more of the park's attractions, including other rides, food-courts, and shopping areas.

   - **We assume enjoyment of the park increases as overall waiting time decreases.**

2. **All people must perceive that they are being treated fairly.**

   A QuickPass system must operate logically and be comprehensible, at least in function, to the customers. A system that is perceived as random may cause discontent even if it minimizes waiting times. We saw an example of this given in the problem, where unexplained changes in scheduled times between adjacent tickets caused complaints.

3. **QuickPass must not significantly deter from the enjoyment of those not using it.**

   Here we assume that the population of QuickPass users is small compared to the total amusement park population. We justify this assumption when discussing our models. Given this idea, it is clear that we cannot compromise the enjoyment of the majority in favor of the QuickPass users, since this will not benefit amusement park goers overall.

This immediately leads to useful conclusions. For example, regardless of QuickPass implementation, rides must continue to operate at capacity as long as there is demand, otherwise the general population is affected and upset.

Stating these assumptions shows us the characteristics of a good enjoyment measure in our model. We use these assumptions as constraints in any model we propose. The QuickPass system is more enjoyable to all who use it, and does not significantly affect those who do not use it. Thus to maximize park enjoyment, we should maximize QuickPass use, subject to the constraints defined above.

- **To maximize park enjoyment, maximize QuickPass use subject to constraints.**

4 Properties of a Good Model

We can write down what we look for in a good model. We are considering QuickPass to be an actual system to be implemented in an amusement park. A good solution should have the following properties:

1. **Solves the problem.**

   Our model should maximize user enjoyment as we have defined it, subject to the constraints we defined. It need not be optimal, but it should be very good.

2. **Ease of implementation.**

   We are intending for this system to be used in an actual theme park. Thus we aim for simplicity of implementation rather than mathematical complexity.

3. **Ease of use.**

   We cannot have unreasonable expectations of people using the system. That is, we do not want a system that only runs smoothly when everybody shows up exactly on time, and degenerates when this is not the case.

4. **Not be sensitive to random events.**

   Park attendance can vary over time, and how people use rides can be modeled by various probability distributions. We want the effectiveness of our QuickPass system to not decrease due to random chance.

5. **Adjustable and adaptive.**

   We do not want a model with a large number of parameters that have to be set and reset every day because of various conditions. We want a model that can easily be adjusted, or adjust itself, based on its environment.

Now that we have defined our terms and set our goals, we discuss our tools and models.

5 Basic Queuing Theory: Is It Useful?

- **Key Points:** Queuing theory assumption - Impossibility of application to old fashioned theme park lines - Motivation for applying to QuickPass lines - Short example from our cafeteria

Queuing theory is a pre-existing and well researched branch of mathematics, with applications ranging from grocery store line models to computer processing event queues. This section will introduce the basic concepts of queuing theory, and discuss how we can apply them to our specific problem. We will see that the concepts are more applicable to our GhostQueue implementation of the QuickPass system than to our model for the regular line largely because of the unavailability of precise data.

The QuickPass system operates during peak hours, when the lines formed for major attractions are not increasing at a significant rate. Because of this observation, we assume that we are in a steady state. This assumption makes sense, because we expect people to stop getting into lines if they grow too large.

Once we are in a steady state, we can make the following key assumptions:

1. **Mean people served per minute, μ, is constant.**
2. **Mean people arriving per minute, λ, is constant.**
3. **On average, more people are served per minute than arrive. That is, $\mu > \lambda$.**

Assumptions 1 and 2 mean that neither services nor arrivals depend on any other factors, most importantly time and preexisting line length. Note that for both of these parameters, only the time averaged input and output rates are being considered, but **the time between any two consecutive arrivals or departures need not be the same**. This randomness leads to nonintuitive conclusions below. Assumption 3 is valid, since if it was not the case, the line would continue to grow.

Given these assumptions, the following results can be quickly derived as in [2].

$$mean\ number\ of\ people\ in\ line = \frac{\lambda^2}{\mu(\mu - \lambda)} \tag{1}$$

$$mean\ waiting\ time\ for\ those\ who\ wait = \frac{1}{\mu - \lambda} \tag{2}$$

$$probability\ of\ having\ to\ wait = \rho = \frac{\lambda}{\mu} \tag{3}$$

For general lines we expect the probability of having to wait to be less than 1. It is often the case that a customer arrives and can be immediately serviced. However, when the service is a popular attraction, from our own experience we know the chance of a line being empty during peak hours is approximately zero. If we examine (1) and (2) **we see that problems arise when $\boldsymbol{\lambda \approx \mu}$**. That

Table 1: Statistics for Ten Popular Roller Coaster Rides at Cedar Point Amusement Park (Complete with somewhat tongue-in-cheek "Thrill Rating") [1].

Thrill Rating	Average Wait Time	Riders/Hr	Name of Ride
3/5	15-30 min	1,400	Blue Streak
3/5	15-30 min	2,000	Iron Dragon
2/5	15 min	1,800	Jr. Gemini
4/5	30-45 min	2,000	Magnum
4.5/5	45 min	1,800	Mantis
5/5	1+ hours	1,600	Millennium Force
4.5/5	45 min	1,800	Raptor
5/5	1-3 hrs	1,000	Top Thrill Dragster
4.5/5	45 min	1,000	Wicked Twister
3.5/5	30 min	1,800	Wild Cat

is, both the mean waiting time and the line grow arbitrarily large when $\boldsymbol{\rho}$ is near one. Now consider a ride with a wait time of one hour, like those in Table 1.

$$\mathbf{60}\boldsymbol{min} = \frac{\mathbf{1}}{\boldsymbol{\mu} - \boldsymbol{\lambda}} \Rightarrow \boldsymbol{\mu} = \boldsymbol{\lambda} - \frac{\mathbf{1}}{\mathbf{60}}$$

We see that in order to accurately predict the waiting time, even on the order of an hour, one must know the parameters $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ to at least two decimal places. This may be possible given accurate statistics collected over a long period of time. However, these figures are not easily found, perhaps due to competitive nature of the theme park business. In short, **we need a new model for long lines.**

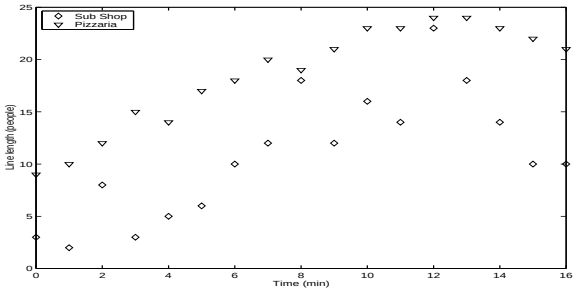
- **A New Model:** It is necessary to have a model that (a) can predict the long wait times shown in Table 1 and (b) is not terribly sensitive to the parameters $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$.

We will pursue this goal in section 6. However, we will present a short example that suggests queuing theory can be used when wait times and line lengths are small.

5.1 Queuing Theory in Our Cafeteria

While it seems that we can not apply queuing theory results to our long lines, it is possible they are useful for short lines. In order to justify further application of this theory, we collected some primitive data during the noon lunch rush in our university's cafeteria. From this data we find that $\boldsymbol{\lambda} = \mathbf{1}$, $\boldsymbol{\mu_{subs}} = \mathbf{1.1}$ and $\boldsymbol{\mu_{pizza}} = \mathbf{4}$. This gives us the following:

Figure 1: Observed line length as a function of time at the sub shop and pizzeria in our university cafeteria during the 12:20 lunch rush



$$\textbf{\textit{pizzeria mean wait time}} = \frac{\mathbf{1}}{\mathbf{4-1}} = \mathbf{.33}\textbf{\textit{min}} \text{ and}$$

$$\textbf{\textit{sub shop mean wait time}} = \frac{\mathbf{1}}{\mathbf{1.1-1}} = \mathbf{10}\textbf{\textit{min.}}$$

Since we have attended these places a large number of times, we can say from personal experience that these figures seem accurate. It remains to be seen how this can be applied to our amusement park lines. If our QuickPass system has the same characteristics of the sub shop or pizzeria, then we are in great shape.

6 Long Line Formation Model with Limited Sensitivity

- **Key Points:** Line model based on a motivated differential equation - Too stable - Generally agrees with experiment - Computer simulation.

Assumption: We need to consider only one line. We justify this assumption because we can treat every ride at an amusement park as independant. The number of visitors to a ride with QuickPasses for another ride is assumed to be small. Their impact is thus neglected in our model.

The previously cited queuing theory results assume that the average rate of people who arrive and average rate of people who are served are constant. We will discard these assumptions and write down a differential equation approach to modeling a line. Then we will selectively add assumptions as necessary or justifiable to produce a realistic approach.

The rate of change of the length of a line should depend on the number of people in the park, probability that they want to join the line, and the constant number of people the ride is servicing. So,

The rate of change of line length $\boldsymbol{L}$ is given by the input rate $\boldsymbol{I}$ minus the output rate $\boldsymbol{O}$

$$\frac{\boldsymbol{dL(t)}}{\boldsymbol{dt}} = \boldsymbol{I - O.} \tag{4}$$

The input is the number of people who join the line. This is given by the product of the population who could get on the ride, $\boldsymbol{P}$, with the probability that they are interested in the ride during one time interval, $\boldsymbol{\alpha}$.

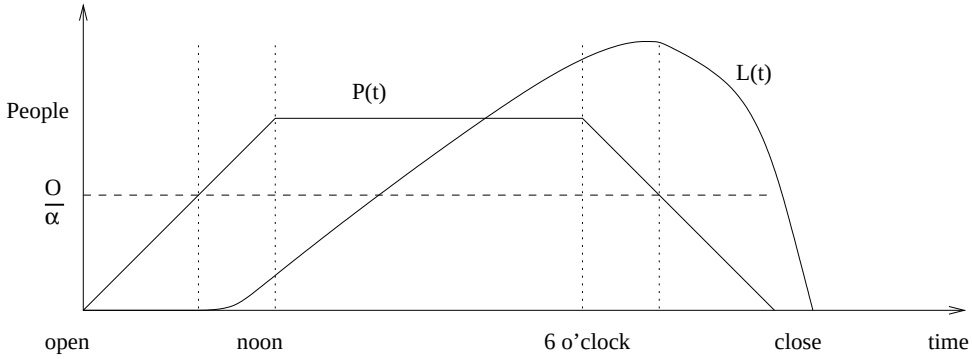
$$\frac{\boldsymbol{dL(t)}}{\boldsymbol{dt}} = \boldsymbol{\alpha P(t) - O} \tag{5}$$

Where

- $\boldsymbol{\alpha}$, the probability that someone joins the line, is potentially a function of the current length of the line, and the perceived fun of that ride.
- $\boldsymbol{P(t)}$, the number of people in the park, is a function of time.
- $\boldsymbol{O}$ is a constant, since rides are run only as often as the machinery allows.

Let us assume that $\boldsymbol{\alpha}$ is a constant. Then, for the estimate of park attendance shown in Figure 2 the solution to (5) is intuitively clear. The line will be zero length until the park population reaches the $\frac{\boldsymbol{O}}{\boldsymbol{\alpha}}$ line, then it will briefly have an increasing slope. Next, the slope is constant until park attendance begins to decrease. Only then will the line reach its maximum as park attendance falls below $\frac{\boldsymbol{O}}{\boldsymbol{\alpha}}$.

Figure 2: Line length L(t) as predicted by (5) given constant $\boldsymbol{\alpha}$ and the park population P(t) shown.



The longest lines are occurring around the peak. This is also the flattest part of the line length curve, varying on the order of $\pm$ 10% during the time span about the peak. Since this is exactly the time period in which our model is to be effective, **we make the assumption that standard line lengths do**

not change greatly over time. This will be useful in our estimates of the effectiveness of the virtual queue. This assumption is further justified by the belief that people will be less likely to get into long lines, thus line length will level off before the peak expected by this simple model, but stay constant due to the increased input when length starts to go down.

By its very nature, the differential equations we write down assume a lack of variation on the part of guests and ride operators. If both agents act with clockwork precision, the above approach would be sufficient, but this is not the case.

- Queuing theory and common sense tell us that the differential equations approach is too deterministic.

The distribution of arrival times has been shown above to be an important factor in understanding line evolution. A good model must thus take these statistical deviations into account. Even so, the line predicted in Figure 2 matches remarkably well with the limited data we have collected as shown in Figure 1. To incorporate statistical deviations into our model we introduce a computer simulation.

6.1 Computer Simulation of Long Line Model Plus Noise

- **Key Points:** How our computer simulation works - Data, Diff. Eq. Model, and Simulation all agree.

This section describes how our computer simulation works. As a general overview, it dequeues (removes from the queue) at a fixed rate, enqueues (adds to the queue) at a rate dependant on time, and both are subject to noise.

The rules that define the behavior of our simulated queue are as follows:

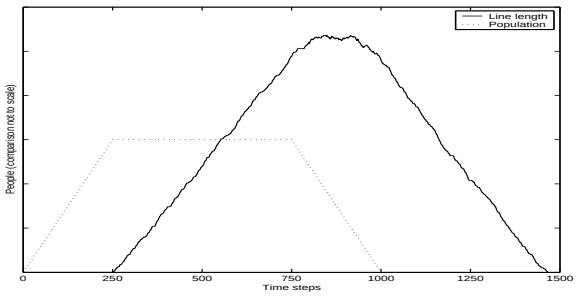
1. **Time is subdivided into N discrete time steps.**
2. **A person is dequeued with a fixed probability, subject to noise from a random number generator.**
3. **A person is enqueued with a variable probability, also subject to noise. The probability of enqueuing is a function of time step.**

We can refer to Figure 3 for an example of what output this simulation gives. In this case, N=2000, an average of one person is dequeued per time step, and zero to two people are enqueued per time step as shown in the figure. Note that this figure closely resembles our model for line growth shown in Figure 2 and our collected data shown in Figure 1.

- **Section 9 will use this same program in a new way to test an adaptive QuickPass system.**

Now that we have developed a line formation model, we can go on to discuss the effect of implementing a QuickPass scheme.

Figure 3: Simulated line length with respect to time step given "population" input shown.



7 Mixing the Two Lines

- **Key Points:** Physically Implementing any QuickPass Scheme at the Point Where the Two Lines Merge

Before we propose detailed QucikPass schemes, we should make precise our notion of how we are going to physically implement these systems. A single, standard, line system at an amusement park is much less complex than a system with a QuickPass. Any QuickPass system necessarily introduces the possibility of four lines:

1. **The standard line (to get onto the ride).**
2. **The QuickPass line (to get onto the ride).**
3. **The Kiosk Line (to get the QuickPass).**
4. **The line to get into the QuickPass line (because customers are not allowed to enter until their alloted time).**

For our purposes we will assume that the Kiosk line will be very fast serving, and that the line will therefore be negligible. The fourth line is a bit of a paradox. The QuickPass system was implemented for those people who do not like waiting in line, so it seems this line should be small. If this proves not to be true in practice, then people are missing the point of the QuickPass system. We will therefore assume that the fourth line is insignificant.

Not all parks are equipped or desire to have two lines meeting exactly at the ride.

- **We must integrate the standard line, and the QuickPass line some time before the ride.**

This also makes sense because time spent arranging people in line is not lost; these people are waiting anyway. Yet, time spent arranging people on the ride

pre-ride platform could possibly cause delays in getting people on the ride. This hurts everyone, so it must be avoided.

We propose to always mix the lines about five to ten minutes before the pre-ride platform. The mixing rate could potentially be controlled by the QuickPass system itself. Whether the rate is constant or variable, a system of red light/green light watched over by an employee could be used to mix the lines without too much trouble.

8 GhostQueue

Key points: A "ghost queuing" system is defined - This common sense ideal system is unstable - Few people can benefit from this system.

This process behaves as if the guest has a "ghost" that stands in line for him, calling him back only when he has reached the front of the line. We will find that GhostQueue works well at very limited capacity, but it becomes subject to the same problem as regular lines as capacity grows.

- **Assumption: The wait time for the normal line is known to the system.** Many current systems of virtual queuing, such as Disneyland's FastPass, provide this information at the QuickPass kiosk [3]. This is provided by the knowledge of the average rate at which a line moves, a statistic available to theme park management. A park employee can, for instance, enter this information into the QuickPass system at frequent intervals.

The GhostQueue system works in the following way:

1. **A guest enters his ticket into the kiosk.**
2. **The kiosk looks at the current length of the normal line and returns a ticket stamped with this time. This is the beginning of a short time window for returning to the GhostQueue line.**
3. **The guest is free to roam about the park.**
4. **When his window is about the begin, he makes his way back to the ride.**
5. **The employee administering the ride first takes people from the GhostQueue line, then brings the ride to capacity from the normal line.**
6. **The person has a great deal of fun on the ride and is no longer in any line.**

In theory nothing has changed for people in the standard line. That line acts exactly as if all guests were still present, even if only their ghost waits. The

average wait time, $\bar{w}$ is given by

$$\bar{w} = \frac{\text{Total Time Waited}}{\text{Total People}} \tag{6}$$

With some people not waiting, yet the total people staying the same, $\bar{w}$ seems goes down with each new person using the ghost queue.

- **The optimal solution thus appears to be (but is not!) assigning everyone a ghost and watching the average wait time drop to zero.**

Why is this not the best approach? People returning at a predetermined time is not a deterministic process, but rather a probabilistic process, so our queuing theory results are applicable. In order to run at capacity and not have a standing line outside, people must arrive at the same rate that the ride is boarding. In the parameters of queueing theory, this means $\boldsymbol{\lambda = \mu}$. However, equation 2 tells us that

$$\textbf{Mean wait time for entry} = \frac{\mathbf{1}}{\boldsymbol{\mu - \lambda}} \overset{\lambda \to \mu}{\longrightarrow} \boldsymbol{\infty}.$$

Thus, as was remarked previously, it is not enough to simply strike a balance between expected number of people coming arriving and expect zero wait time. In fact,

- **If we run the fully ghosted system at capacity, actual wait times would get arbitrarily long!**

Reducing the number of ghost spots is equivalent to reducing $\boldsymbol{\lambda}$. Since we wish to keep the wait time small for users of the ghost queue, we must both

1. **make $\frac{1}{\mu - \lambda}$ small and,**
2. **keep the system stable to variations in $\boldsymbol{\lambda}$.**

The second goal is met when $\frac{d}{d\lambda}\frac{1}{\mu - \lambda} = \frac{1}{(\mu - \lambda)^2}$ is small. The first goal implies that $\boldsymbol{\mu - \lambda}$ be made as large as possible. Both goals thus encourage the same end result. However, if the ride is not always filled by the ghost queue, which, due to random distribution of arrivals, will occur at some times no matter what.

- **Therefore, there must be a standard line to keep the ride full.**

From the perspective of a guest, the length of the visible standard line must be related to its wait time, or else they will view the line as unfair. For example, if there were only 20 people in the standard line, but only 1 person per minute was boarding a roller coaster from that line - the rest coming from the ghost queue - then this would not be an attractive line in which to stand. A balance needs to be created between this perceived fairness and the average wait time.

- **The GhostQueue system is thus feasible only if the number of people who use it is kept low relative to the number using the normal line.**

The above conclusion relies partially on a notion of how fairness affect happiness. However, in the opening sections of this paper we introduced fairness as an absolute requirement. Thus, it is beyond the scope of our model to weight fairness quantitatively.

It should be noted that a similar system, Lo-Q, is in use at six Six Flags amusement parks. The user limit comes from a finite number of devices that must be rented to access the ghost queuing feature. Based on the claim that 750,000 people had used the system by October of 2003 [4], after the end of the warm weather peak season, and a total 2003 attendance across the six parks of approximately 13,000,000 visitors [5], this puts the average utilization around 20%, in agreement with the magnitude of usage predicted above.

9 An Adaptive Algorithm Using a Kalman Filter

- **Key Points:** Idea of a Dynamic Algorithm - Set up Dynamic Algorithm - The Dynamic Algorithm Itself (Kalman Filter) - Testing/Implementing this System

9.1 The Idea of an Adaptive Algorithm

As was remarked in previous sections, line growth is most appropriately modeled with a semi-chaotic, or at least probabilistic, approach. Optimization becomes difficult to define in chaotic system. We therefore introduce an adaptive model that does not try to maximize anything per say, but instead tries to adaptively make today's performance better than yesterday's.

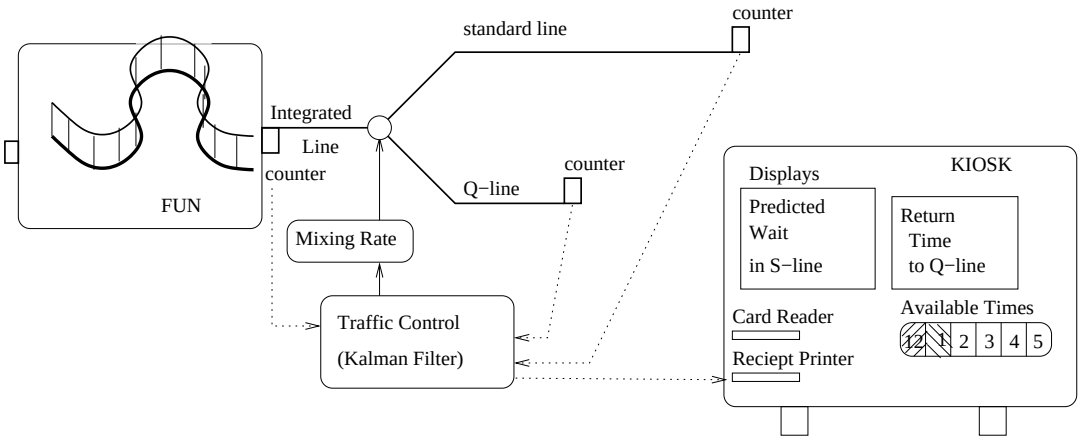
- **Instead of trying to achieve a rigid optimal solution, we try to stay near a good solution by continually adapting.**

A simple, albeit crude, adaptive algorithm might count the number of people who have to wait more than say 10 minutes in the QuickPass line today, and then assign that amount fewer QuickPasses tomorrow. Among the problems with this algorithm are (a) the sensitivity to random variations in attendance, and (b) lumping the whole day into one block of time is too coarse to capture many subtleties of park attendance. We propose then an algorithm that breaks the day into more blocks of time, and is not as sensitive to random fluctuations.

9.2 Set up for an Adaptive Algorithm Implementation

The set up for this dynamic algorithm is described largely in Figure 4. We have a traffic control box which knows how many people are in each line. This traffic

Figure 4: A diagram of the park showing the elements including the Traffic Control Center, the Kiosk, and of course the FUN



control box uses an algorithm to determine how many QuickPass tickets should be available per each hour time slot. This information, as well as the wait times for the standard and QuickPass lines are fed into the Kiosk. The Kiosk displays these waiting times, and gives people an option for which time to return. The noon and one options are blacked out indicating that those time slots are full.

9.3 Assumptions for Adaptive Algorithm

Before we describe our algorithm, we state our assumptions.

1. **There is little day to day variation in the overall distribution of park attendance.**

   We can make this assumption because the QuickPass system will be used during the peak seasons and peak hours. It is reasonable to expect that the pattern of line formation for a specific attraction will change slowly over those times. Thus if we wish to predict demand for QuickPasses for a block of time during day $\boldsymbol{k}$, we should consider the demand during that same block of time during day $\boldsymbol{k-1}$.

2. **We set the number and times of QuickPasses at the beginning of the day.**

   Our system clearly displays how many QuickPasses are available for purchase, and for which time slots. People can buy these at any point during the day, even non-peak hours, on a first-come-first-serve basis. The system is thus fair and logical, and will not behave strangely if there is variation in line formation. Furthermore, it maximizes enjoyment of the people who

choose to buy the QuickPasses, without affecting everybody else as long as we keep that number small.

3. **Line speeds at peak hours are nearly constant.**

   As long as the ride is operating without breakdowns, people are using it at a constant rate. Thus there is a certain speed V with which the lines are moving, and this information is available at specific theme parks from previous research.

4. **We allot a portion of space on the ride for QuickPass users.**

   The percentage of ride seats made available per unit time for QuickPass users is the same as the mixing rate of the normal and QuickPass lines. We would like to maximize this mixing rate subject to the constraint that people in the normal line do not notice the slowdown. This maximum mixing rate, $\boldsymbol{M}$, can easily be found from a pilot study, or just taken to be reasonably small, on the order of 5-10%, if a pilot study is not an option.

5. **We declare a target maximum QuickPass queue length.**

   We are trying to maximize use of the QuickPass system, without making the QuickPass line too long.

9.4 Kalman Queue

The Kalman filter is a set of recursive equations that provides a computationally efficient solution to the least squares method [6]. Kalman filters have many applications, most notably in autonomous navigation systems. They are appropriate here because our model is a discrete-time controlled process, and also because Kalman filters should satisfy our requirements for a good model. We will briefly describe general Kalman filters, and then show how we use them in our model.

- A Kalman filter can be used to estimate a state $\boldsymbol{X} \in \Re^{\boldsymbol{n}}$ at time $\boldsymbol{k+1}$ using the state $\boldsymbol{X}$ at time $\boldsymbol{k}$ and an observation $\boldsymbol{Z}$ at time $\boldsymbol{k}$.

Kalman filters are adaptive, yet they filter out random noise to yield a stable system. As the subscripts indicate, we are working with some timestep $\boldsymbol{k}$. The following equations describe a general Kalman filter with no control input.

$$\boldsymbol{X_{k+1}} = \boldsymbol{A} * \boldsymbol{X_k} + \boldsymbol{w_k} \tag{7}$$

$\boldsymbol{A}$ relates the previous state to the next state and $\boldsymbol{w_k}$ is Gaussian process noise.

$$\boldsymbol{Z_k} = \boldsymbol{H} * \boldsymbol{X_k} + \boldsymbol{v_k} \tag{8}$$

$\boldsymbol{Z_k}$ is the observation made at time $\boldsymbol{k}$, and $\boldsymbol{H}$ relates the magnitude of the state to the magnitude of the input. $\boldsymbol{v_k}$ is the Gaussian measurement noise.

Now we consider our model.

- **State:** Number of QuickPasses available for a given block of time.

  We only keep track of this quantity in our state, so $\boldsymbol{n = 1}$.

- **Measurement**: (Target QP queue length − Attained QP queue length.)

  If this simple difference is positive, we should assign more QuickPasses the next day, since the QuickPass system is not running at capacity. If it is negative, the QuickPass line is too long, and we should assign fewer QuickPasses the next day.

- **Finding $\boldsymbol{H}$**

  $\boldsymbol{H}$ is a scalar that relates the magnitudes of state and input. We assume the QuickPass line is moving at speed $\boldsymbol{Mv}$, and so a clear choice is

$$\boldsymbol{H = \frac{1}{Mv}} \tag{9}$$

  We now give our recursive equations for the Kalman filter. As we have no control input, we only need the measurement update equations.

$$\boldsymbol{K_{k+1} = \frac{P_k * H}{H^2 P_k + R}} \tag{10}$$

$$\boldsymbol{X_{k+1} = X_k + K_k(Z_k - H * X_K)} \tag{11}$$

$$\boldsymbol{P_{k+1} = (1 - K_k * H)P_k} \tag{12}$$

  Here, $\boldsymbol{P}$ is the error covariance, $\boldsymbol{R}$ is the measurement noise, and $\boldsymbol{K}$ is a quantity known as the Kalman gain.

The variables $\boldsymbol{X}$, $\boldsymbol{P}$, and $\boldsymbol{K}$ are clearly related, as each adjusts per iteration depending on the other ones. Though the relationship seems complicated, the adjustments are intuitive. The measurement $\boldsymbol{Z}$ is scaled by $\boldsymbol{K}$, the Kalman gain. $\boldsymbol{K}$ is thus a measure of how much we trust the measurement when computing the next state based on previous experience. The Kalman gain varies with $\boldsymbol{R}$, and would simply be $\frac{1}{H}$ if $\boldsymbol{R = 0}$, which means that we would simply add $\boldsymbol{Z}$ to $\boldsymbol{X}$ if $\boldsymbol{Z}$ was completely accurate. Additionally, the condition $\boldsymbol{R = 0}$ would impose the condition $\boldsymbol{P_k = 0}$, which also makes good sense.

9.5 Testing Our Adaptive Algorithm

- **Key Points:** Guessing Parameters - Applying line length simulation - Experimental Results

Our Kalman filter takes as input four parameters: $\boldsymbol{P_0}$, $\boldsymbol{K_0}$, $\boldsymbol{R}$, and $\boldsymbol{H}$. $\boldsymbol{P}$ and $\boldsymbol{K}$ are self-adjusting, and so the filter is not sensitive to those initial conditions. $\boldsymbol{H}$ and $\boldsymbol{R}$, however, strongly affect the convergence of the model. Amusement parks keep their attendance data closely guarded, so we do not have correct values for these parameters to use with real data. As a result, we are forced to guess reasonable values for these parameters and create our own test situation. We emphasize that an actual amusement park can easily calculate these parameters from their statistics and obtain much better initial values and parameters that will ensure lightning-fast convergence. We show here that even with our rough guesses the Kalman filter still settles to equilibrium fairly quickly.

We tested the filter by iterating the computer simulation described in section 6.1. Given initial values, the first relevant output from the Kalman filter is the number of QuickPasses assigned per hour block of time. We assume people arrive uniformly over their assigned block, which is justified in section 9.6. Given this, and the number of QuickPasses assigned per block, we can make a graph of how many people are arriving as a function of time. This is exactly the input for our computer simulation previously developed.

- We can think of the distribution of people into blocks as a new special population in the park and feed this into our simulation for line growth to determine the length of QuickPass line.
- This is a simulation.

This test does not capture the true power of our adaptive algorithm because we do not know the parameters $\boldsymbol{H}$ and $\boldsymbol{R}$, and our model for line growth is a fairly rough and simple probability model. In actuality the data which the Kalman filter uses as input is smoothed by the large number of people in a real system with known parameters. Kalman filters have been extensively tested in real world situations and found very effective for similar processes, so we expect it to be effective for this one. However, we should still test our implementation and see how well the Kalman filter can model it.

The Test:

1. We assume the peak time of day is subdivided into six equal blocks. A real amusement park might subdivide further, if behavior varies too much within the blocks. We "couple" the blocks by having the final queue length of block $\boldsymbol{i}$ as an initial queue length for block $\boldsymbol{i+1}$.
2. We use the same model for line formation as in our simulation in section 6.1. Additionally, we attach a Kalman filter to each block. We emphasize that the blocks are coupled, so the behavior of each block depends on the previous blocks.

3. We guess parameters ($\boldsymbol{H}$, $\boldsymbol{R}$) and far fetched intial values ($\boldsymbol{K}$, $\boldsymbol{P}$, $\boldsymbol{Qpb}$: QuickPasses Per Time Block) for the initial Kalman filter input.

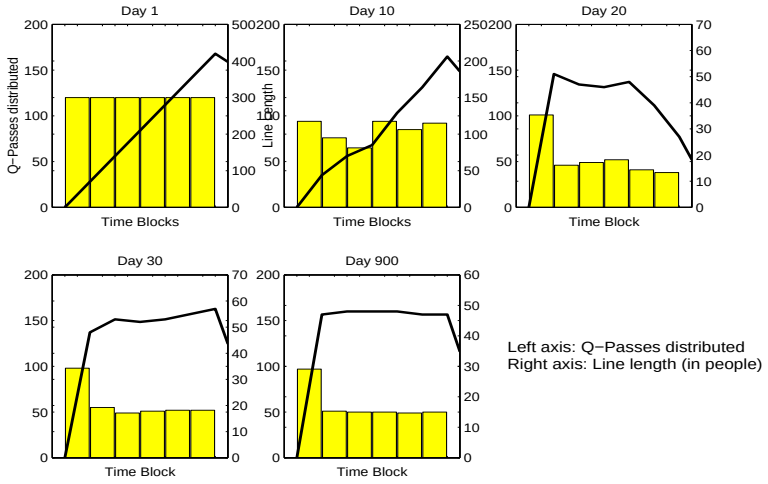
4. Subject to noise, we input this $\boldsymbol{Qpb}$ into our line growth model. The Kalman filter measures the deviation of the actual line from the ideal line.

5. The Kalman filter outputs a new value for QPB, K, and P. We now have a new value for QuickPass for every block.

6. We iterate this process 1000 times. The output of step **5** is the output of step **4**.

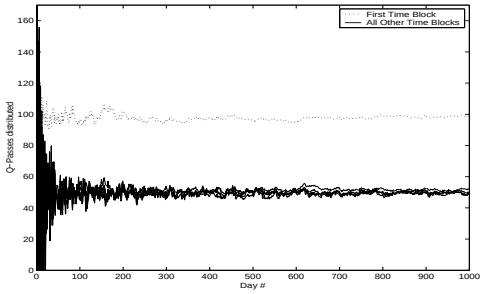
7. We confirm visually that regardless of initial values, the Kalman filter converges to a steady, optimal QuickPass number per block that results in a stable and optimal QuickPass queue length over time.

Figure 5: Number of QuickPasses alloted per hour as determined by Kalman Filter and QuickPass line length as determined by line growth model for days 1, 10, 20, 30 and 900.



A pictographic version of the results of a trial are shown in Figure 5. An actual plot of the results is shown in Figure 6 The system output was programmed to vary around 100 people per hour, but we input the initial value of 120 QuickPassers per hour. We can think of each timestep as previous similar day. For instance, Saturday is more popular than Wednesday, thus when using this system on a Saturday, we look to last Saturday's data. Clearly, for the first day the the line grows rapidly as the ride can not service the demand. If we think of each iteration as a day, then the figure shown is day 10. By day 10 the line has visibly deformed. On days 20, 30 and, finally, 900 the QuickPasses

Figure 6: A plot of the QuickPasses alloted day by day given wildly wrong initial values. Figure shows how the Kalman filter stabilizes over time even without good data to begin with.



distributed per hour result in a nearly constant and optimal QueuePass queue length.

For our test we made wild guesses about QuickPass distribution initially and tried a few values for $\boldsymbol{H}$ before chosing one that worked reasonably. Figure 7 shows the dependance on $\boldsymbol{H}$. An increase of a factor of 8 in $\boldsymbol{H}$ only corresponded to a difference of about 15%, indicating good stability in this parameter. The largest effect is, as expected, the rate of convergence. Higher $\boldsymbol{H}$ values make it converge much faster and stay more stable, at the cost of having lower queue lengths. The purpose of this test was to show that Kalman filter is highly adaptive. In an actual implementation of this scheme the initial values of QuickPass distribution and the parameter $\boldsymbol{H}$ can be determined very precisely from previously collected data. **Thus our algorithm is likely better than this test indicates.**

- The initial variation in QuickPass distribution would be much less in an actual implementation.

Figure 7: The dependance of our model on the parameter H. Note that should be determined from accurate data collected at the park rather than guessed as we did.

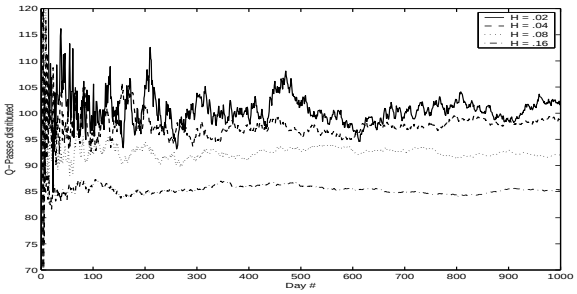
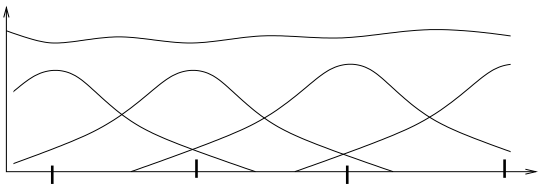


Figure 8: Sketch of how to add normal distributions to achieve constant arrival.



It was necessary for the purpose of testing to make the Kalman filter make big adjustments because our line growth model is not as variable as actual line growth.

9.6 Uniform Arrival Rate Justification

- **Key Points:** It is reasonable to say people arrive uniformly over their block of time.

People either arrive uniformly over their block of time, or they arrive with some distribution throughout their block. Assuming they arrive with a fixed distribution over the course of their block, we should be able to overlap the blocks in such a way that the average arrival is still a constant. For example, Figure 8 shows a rough idea for how was can overlap the blocks assuming people arrive with a normal distribution about the center of their block. This complexity is well suited to being added in after a model is running sufficiently well, as it adds complexity while not significantly increasing usefulness.

10 Conclusions

The GhostQueue system decreases wait times and thus increases happiness. However, the manner in which this is done is hard to optimize. Fairness considerations would be difficult to define and test. We suggest GhostQueue as a good solution only if an external way of keeping GhostQueue utilization low, such as selling access to it, is applied.

The KalmanQueue process was designed in accordance with our defined criteria for a good model. It has been shown above to meet these criteria, though it has some sensitivity in its parameters. Because these parameters can be determined from detailed observation of a ride, we state this is not a sufficient reason to doubt this model. The dynamical optimization approach is also easy for the park to use, since the system adjusts itself to meet an ideal line condition. **We recommend the use of a KalmanQueue system for rides with large line lengths.**

11 Strengths & Weaknesses

General Model

- **Strengths**
  - We developed a theoretical line formation model which agrees with our rough data. Our computer model agrees with both despite working on different principles, implying it behaves as we want.
  - Our line model incorporates the natural randomness of human behavior.
- **Weaknesses**
  - Current line length is not taken into account by the line formation model. In real life, a person will be more likely to join a short line than a long one, which our model does not predict.
  - We ignore the effect of the virtually queued people being able to join standard lines, thus increasing the standard line wait time.
  - Our model is only valid during peak hours.
  - No claims are able to be made about the idealness of our solutions.

GhostQueue

- **Strengths**
  - The return time calculation is simple and intuitive for the guest.
  - No guest waits longer than if he would have without the GhostQueue, thus making the system seem fair.
- **Weaknesses**
  - The utilization must be kept low for GhostQueue to be beneficial.
  - Fairness becomes a dominant factor in optimal utilization. Our assumptions do not quantify fairness, thus optimizing it is beyond the scope of the model.

KalmanQueue

- **Strengths**
  - The primary input is the desired behavior of the QuickQueue line and the model adjusts itself accordingly. Thus an administrative decision is all that is needed to make the system work properly.
  - The KalmanQueue process satisfies all six properties of a good model, as defined in section 4.

  - The Kalman filter is highly adaptive to subsequent changes in the queuing process (e.g. time varying output rates), thus the core framework is valid for a wide variety of alterations.

- **Weaknesses**
  - Our model assumes a constant line mixing rate, which does not seem likely to be ideal.
  - $\boldsymbol{H}$, the adjustment parameter, and $\boldsymbol{R}$, random variance of the number of Q-Passes distributed, must be determined accurately for the model to be useful.
  - The Kalman filter currently tries to decrease the *maximum* QuickPass queue length. A future model should try to decrease the *average* QuickPass length.

References

[1] www.thepointol.com, Accessed Feb. 7th, 2003.

[2] Ruiz-Pala et al, Waiting-Line Models, New York: Reinhold Publishing Co., 1967

[3] http://www.mouseplanet.com/al/docs/fast.htm, Accessed Feb. 7th, 2004.

[4] http://www.lo-q.com/Press/Amusement%20Business%20March%202001.htm, Accessed Feb 8th, 2004.

[5] Tim O'Brien, *North American parks down slightly from 2002*. Amusement Business. Vol. 115, Issue 51, p. 9.

[6] Welch & Bishop, *An Introduction to the Kalman Filter*. Department of Computer Science. University of North Carolina at Chapel Hill. http://www.cs.unc.edu/ welch/kalman/kalmanIntro.html, Accessed Feb. 8th, 2004.