KalmanQueue: An Adaptive Approach to Virtual Queueing

Tracy Clark Lovejoy
Aleksandr Yakovlevitch Aravkin
Casey Schneider-Mizell
University of Washington
Seattle, WA

Advisor: James Allen Morrow

Summary

QuickPass (QP) is a virtual queueing system to allow some theme-park guests to cut their waiting time by scheduling a ride in advance. We propose innovative QP systems that maximize guest enjoyment. Only a small portion of guests can effectively use QP, and a good system maximizes this group subject to the constraints that regular users are not significantly affected and maximum waiting time for QP users is small.

We define and test a simple model for single-line formation and then develop two QP systems, GhostQueue and KalmanQueue. GhostQueue is intuitive and simple but would be far from optimal in practice. We then propose that the best model is one that adapts to its environment rather than trying to enforce rigid parameters. We implement KalmanQueue, a highly adaptive system that uses an algorithm inspired by the Kalman filter to adjust the number of QPs given today based on the maximum length of the QP line yesterday, while filtering out random noise. We simulate the KalmanQueue system with a C++ program and randomized input from our line-formation model. This system quickly converges to a nearly optimal solution. It is, however, sensitive to some parameters. We discuss the expected effectiveness of the system in a real environment and conclude that KalmanQueue is a good solution.

The UMAP Journal 25 (3) (2004) 337–352.

Introduction

The underlying idea of how to reduce waiting time for some theme-park guests is simple: Rather than wait in line, guests get tickets that tell them when to come back; when they return, they wait in a shorter line before going on the ride.

Many such systems have been implemented, including QuickPass, FastPass, Freeway, Q-lo, and Ticket-To-Ride; some have failed and others have thrived. The appeal to the parks is twofold: The systems increase guests' enjoyment, and guests spend more money in the park instead of waiting in line.

All systems that we researched assume that the number of QP users is small and manage the system either by restricting the total number of QPs or by charging a fee for them.

Plan of Attack

We seek to maximize guest enjoyment.

- **Define Terms:** We state a definition of "guest enjoyment" and explain what affects it, why, and how we model it.
- **State Assumptions:** We restate the problem in a mathematical way.
- **Describe a Good Model:** An effective QP system has certain desirable characteristics. describing these steers our model in the right direction.

We then present our models:

- **Line-Formation Model:** The QP system is a line-manipulation system, and we cannot hope to design it without first understanding line formation.
- **GhostQueue: A Simple Model:** We describe a simple but limited approach to a QP system.
- **KalmanQueue: An Adaptive Algorithm:** We propose, model, and test an adaptive algorithm as a solution to the QP system. We provide a simple implementation using an adapted 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.

Increasing Enjoyment

- **Guests enjoy wandering freely more than they enjoy waiting in a line.** This is the basic assumption that makes virtual queueing potentially useful for increasing guest enjoyment. When not in line for a specific ride, guests

can enjoy more of the park's attractions, including other rides, food courts, and shopping areas. We assume that enjoyment of the park increases as overall waiting time decreases.

- **All guests must perceive that they are treated fairly.** A QP system must operate logically and be comprehensible, at least in function, to the guests. A system that is perceived as random may cause discontent even if it minimizes waiting times. We see an example of this in the problem statement, where unexplained changes in scheduled times between adjacent tickets causes complaints.

- **QP must not significantly deter from the enjoyment of those not using it.** The population of QP users is small compared to the total park population. Regardless of QP implementation, rides must operate at capacity as long as there is demand, otherwise the general population is affected and upset. The QP system should be more enjoyable to those who use it and not significantly affect those who do not.

Properties of a Good Model

- **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.

- **Ease of implementation.** We intend for this system to be used in an actual theme park. Thus, we aim for simplicity of implementation rather than mathematical complexity.

- **Ease of use.** We do not want a system that runs smoothly only when everybody shows up exactly on time but degenerates when this is not the case.

- **Not be sensitive to random events.** Park attendance varies, and how guests use rides can be modeled by various probability distributions. We want the effectiveness of our QP system not to decrease due to chance.

- **Adjustable and adaptive.** We do not want a model with a large number of parameters that must be re-set every day because of various conditions. We want a model that can easily be adjusted, or adjusts itself, based on its environment.

Basic Queueing Theory: Is It Useful?

Queueing theory is a well-researched branch of mathematics, with applications ranging from grocery-store-line models to computer-processing event queues. We discuss its basic concepts and apply them to our problem.

The QP system operates during peak hours, when lines for major attractions are not increasing at a significant rate. Thus, we can assume that we are in a steady state. This assumption makes sense, because we expect guests to stop getting into lines if they grow too large.

In a steady state, we can make the following key assumptions:

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

Assumptions 1 and 2 mean that neither services nor arrivals depend on other factors, most importantly time and pre-existing line length. For both of these parameters, only the time-averaged input and output rates are 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 were not the case, the line would continue to grow.

Given these assumptions, the results are [Ruiz-Pala et al. 1967]:

$$\begin{aligned} \text{mean number of guests in line} &= \frac{\lambda^2}{\mu(\mu-\lambda)}, \\ \text{mean waiting time for those who wait} &= \frac{1}{\mu-\lambda}, \\ \text{probability of having to wait} &= \rho = \frac{\lambda}{\mu}. \end{aligned} \tag{1}$$

Problems arise when $\lambda \approx \mu$; both the mean waiting time and the line grow arbitrarily large when ρ is near 1. Consider the case of a ride with a wait time of 1 h, like those in **Table 1**:

$$60 \text{ min} = \frac{1}{\mu-\lambda} \quad \longrightarrow \quad \mu = \lambda + \frac{1}{60}.$$

To predict the waiting time accurately, even on the order of 1 h, one must know λ and μ to at least two decimal places. This may be possible given accurate statistics over a period of time; however, these figures are not easily found, perhaps due to competition in the theme-park business.

In short, we need a new model for long lines that can predict the long wait times shown in **Table 1** and is not terribly sensitive to the parameters μ and λ.

We pursue this goal later; now we present a short example that suggests that queueing theory is useful when wait times and line lengths are small.

Table 1.
Statistics for 10 popular rides at Cedar Point Amusement Park (with somewhat tongue-in-cheek "Thrill rating") [Cedar Point Information 2003].

Thrill rating (out of 5)	Average wait time (min)	Riders/hr	Ride
3	15–30	1,400	Blue Streak
3	15-30	2,000	Iron Dragon
2	15	1,800	Jr. Gemini
4	30–45	2,000	Magnum
4.5	45	1,800	Mantis
5	60+	1,600	Millennium Force
4.5	45	1,800	Raptor
5	60–180	1,000	Top Thrill Dragster
4.5	45	1,000	Wicked Twister
3.5	30	1,800	Wild Cat

Queueing Theory in Our Cafeteria

We collected data during the noon lunch rush in our university's cafeteria (**Figure 1**), from which we find $\lambda = 1$, $\mu_{\text{subs}} = 1.1$ and $\mu_{\text{pizza}} = 4$, and

$$\text{pizza mean wait time } = \frac{1}{4-1} \quad = \quad 0.3 \text{ min},$$
$$\text{sub mean wait time } = \frac{1}{1.1-1} = 10 \quad \text{min}.$$

These figures agree with our experience. If our QP system has the same characteristics as the sub shop and the pizzeria, then we are in great shape.

Long-Line Formation Model with Limited Sensitivity

We need to consider only one line, because we can treat every ride as independent. The number of visitors to a ride who have QPs for another ride is assumed to be small, so we neglect their impact.

The queueing-theory results assume that the average arrivals and the average rate of service are constant. Here we discard these assumptions in favor of a differential-equation approach to modeling a line. Then we selectively add assumptions as necessary to produce a realistic approach.

The rate of change of the length of a line should depend on the number of guests in the park, the probability that they want to join the line, and the constant service rate of the ride. For a line of length L, the rate of change is given the input rate I minus the output rate O:

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

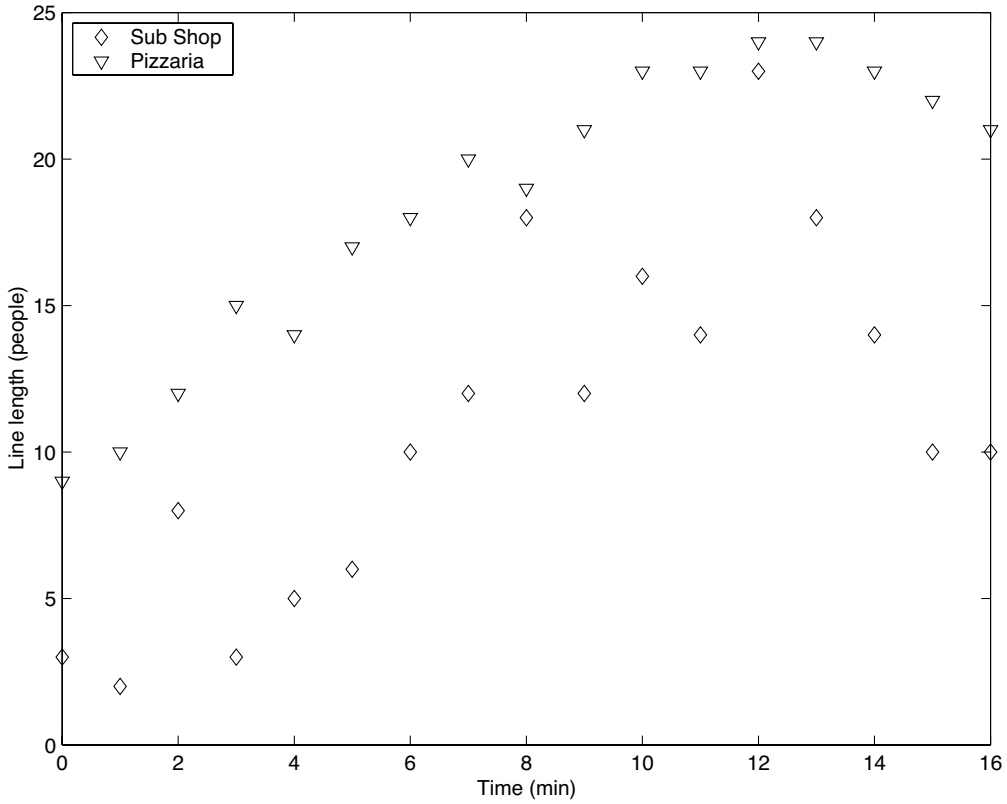

Figure 1. Observed line lengths as a function of time in the cafeteria during the lunch rush.

The input is the number of guests who join the line. This is given by the product of the population P who could get on the ride with the probability α that they are interested in it during one time interval. Hence,

$$\frac{dL(t)}{dt} = \alpha P(t) - O. \tag{2}$$

Here

- α, the probability that someone joins the line, is a function of the current length of the line and the perceived fun of that ride;
- $P(t)$, the number of guests in the park, is a function of time; and
- O is constant, since rides run only as often as the machinery allows.

Let α be constant. Then, for the estimate of park attendance $P(t)$ shown in **Figure 2**, the solution to **(2)** is intuitive: The line has zero length until the park population reaches the O/α line, when it briefly has an increasing slope. Next, the slope is constant until park attendance begins to decrease. Only then does the line reach its maximum as park attendance falls below O/α.

The longest lines occur 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

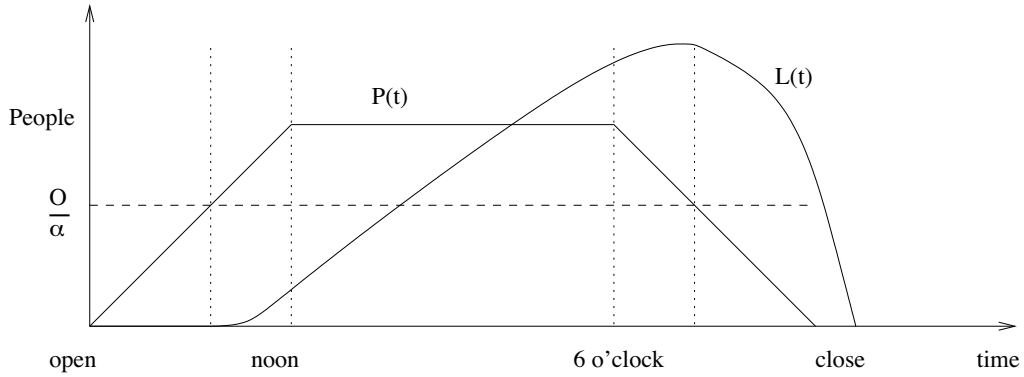

Figure 2. Line length $L(t)$ as predicted by **(2)** for constant α and the park population $P(t)$ shown.

the peak. Since this is exactly when our model should be effective, we assume that the line length does not change greatly over time.

The differential equations assume a lack of variation on the part of guests and ride operators; if both act with clockwork precision, the approach is sufficient. However, queueing theory and common sense tell us that the differential-equation approach is too deterministic; a good model must thus take statistical deviations into account. Even so, the line length predicted in **Figure 2** matches remarkably well with the data in **Figure 1**.

To incorporate statistical deviations, we introduce a computer simulation.

Computer Simulation of Long Line Model

Our computer simulation dequeues (removes from the queue) at a fixed probability, enqueues (adds to the queue) at a probability dependent on time, and both are subject to noise from a random-number generator. We subdivide time into N equal discrete time steps.

Figure 3 shows an example of the output for $N = 2{,}000$, an average of 1 guest dequeued per time step, and 0 to 2 guests enqueued per time step. The shape of the figure closely resembles the model for line growth in **Figure 2** and our data in **Figure 1**.

GhostQueue

The GhostQueue process behaves as if the guest has a "ghost" who stands in line instead, calling the guest back only when the ghost reaches the front of the line. We find that GhostQueue works well at very limited capacity; but as capacity grows, it suffers from the same problem as a normal line.

We assume that the wait time for the normal line is known to the system. Many virtual-queueing systems, such as Disneyland's FastPass, display this information at the QP kiosk [O'Brien 2001].

The GhostQueue system works in the following way:

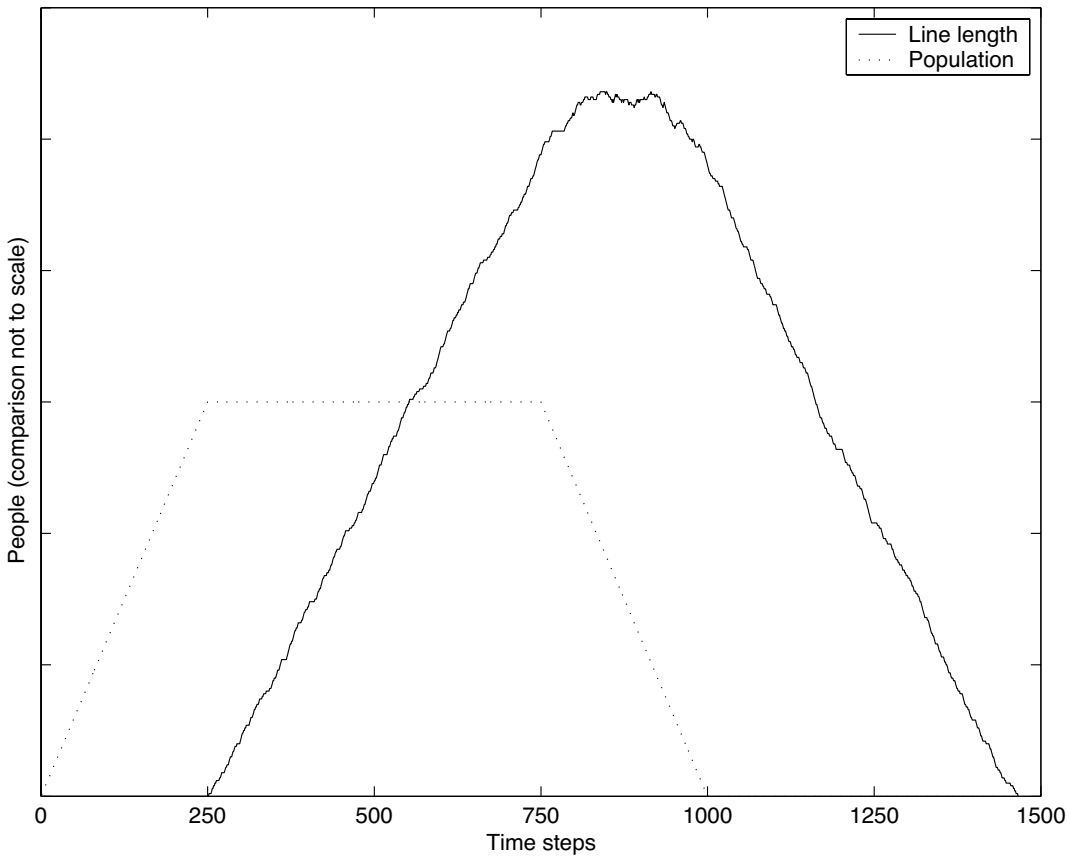

Figure 3. Simulated line length for the population input shown.

- The system checks the length of the normal line, computes the expected time when the guest would enter the ride, and gives the guest a ticket stamped with this time as the beginning of a short time window during which the guest can enter the GhostQueue line.
- The guest is free to roam about the park.
- When the guest's window is about to begin, the guest goes back to the ride.
- The ride takes guests first from the GhostQueue line, then from the normal line.

In theory, nothing changes for guests in the normal line; it acts exactly as if all guests were still present, even though only their ghosts wait. The average wait time $\bar{w}$ is given by

$$\bar{w} = \frac{\text{total time waited}}{\text{total guests}}.$$

With some guests not waiting, yet the total number of guests staying the same, $\bar{w}$ seems to go down with each new guest using GhostQueue. 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? Guests returning at a predetermined time is a probabilistic rather than a deterministic process, so queueing-theory results apply. To run at capacity and not create a line, guests must arrive at the same rate that the ride is boarding. In terms of the parameters of queueing theory, this means $\lambda = \mu$. However, **(2)** tells us that

$$\text{mean wait time for entry } = \frac{1}{\mu - \lambda} \overset{\lambda \to \mu}{\longrightarrow} \infty.$$

Thus, even if there is a balance between arrivals and departures, not only is the expected wait time not zero, but if we run the fully ghosted system at capacity, *actual wait times get arbitrarily long*!

Reducing the number of ghost spots is equivalent to reducing λ. Since we wish to keep the wait time short for users of the ghost queue, we must both

- make $1/(\mu - \lambda)$ small, and
- keep the system stable to variations in λ.

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 we should make $\mu - \lambda$ 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 occurs sometimes because of the random distribution of arrivals), then the ride runs at less than full capacity. Therefore, there must be a normal line to keep the ride full.

From the perspective of guests, the length of the visible normal line must be related to its wait time, or else they will view the line as unfair. For example, if there are only 20 guests in the normal line but only 1 guest per minute is boarding 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 perceived fairness and the average wait time.

Hence, the GhostQueue system is feasible only if the number of guests who use it is kept low relative to the number using the normal line.

In a similar system, Lo-Q at Six Flags amusement parks, the user limit comes from a fixed number of devices that must be rented to access the ghost-queueing feature. Based on the claim that 750,000 guests had used the system by October of 2001 [O'Brien 2001], and that the total 2001 attendance across the six parks was approximately 13 million visitors [O'Brien 2002], the utilization is 6%, in agreement with what we would predict.

An Adaptive Algorithm

The Idea of an Adaptive Algorithm

We introduce an adaptive model that does not try to maximize anything per se but instead adapts to make today's performance better than yesterday's.

A simple adaptive algorithm might count the number of guests who wait more than 10 min in the QP line today and assign that many fewer QPs tomorrow. Problems with this algorithm are the sensitivity to random variations in attendance and that lumping the whole day into one block of time is too coarse to capture many subtleties of park attendance. We propose an algorithm that breaks the day into more blocks and is not as sensitive to random fluctuations.

Setup

The setup is described largely in **Figure 4**. We have a traffic control box that knows how many guests are in each line and determines how many QP tickets to make available in each hour. This information, as well as the wait times for the normal and QP lines, is fed into the kiosk, which displays waiting times and gives guests options for when to return. In the figure, the options noon and 1 P.M. are blacked out, indicating that those time slots are full.

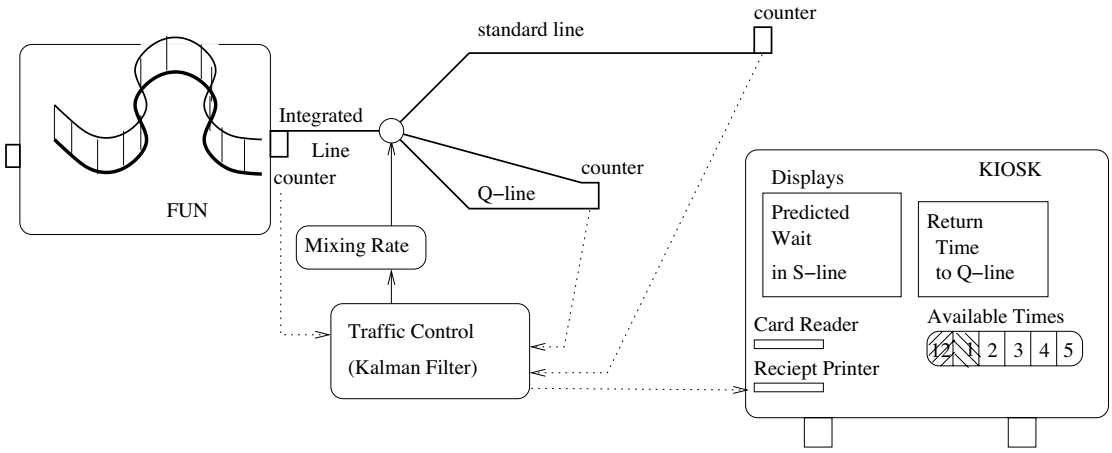

Figure 4. Schematic diagram of the park showing the the traffic control center, the kiosk, and, of course, the fun.

Assumptions for the Adaptive Algorithm

- There is little day-to-day variation in the overall park attendance. The QP system is used during peak seasons and peak hours, when the pattern of line formation for a ride changes slowly.

- We set the number and times of QPs at the beginning of the day. Guests can get these at any point during the day, even nonpeak 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.
- Line speeds at peak hours are nearly constant. As long as the ride does not break down, guests are using it at a constant rate.
- We allot a percentage of ride seats for QP users. The number of such seats per unit time is the same as the rate at which the normal and QP lines can be mixed. The maximum feasible mixing rate M could be found from a pilot study or just taken to be reasonably small (5–10%).
- We declare a target maximum QP queue length.

KalmanQueue

The Kalman filter is a set of recursive equations that provides a computationally efficient solution to the least-squares method [Welch and Bishop 2001]. Kalman filters have many applications, 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 briefly describe general Kalman filters and then adapt them for use in our model.

A Kalman filter estimates a state $X \in \Re^n$ at time $k+1$ from the state X at time k and an observation Z at time k.

Kalman filters are adaptive, yet they filter out random noise to yield a stable system. A general Kalman filter with no control input is described by

$$X_{k+1} = AX_k + w_k,$$
$$Z_k = HX_k + v_k$$

where

- A relates the previous state to the next state,
- w_k is Gaussian process noise,
- Z_k is the observation at time k,
- H relates the magnitude of the state to the magnitude of the observation, and v_k is Gaussian measurement noise.

Now we outline our model:

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

- **Measurement U_k = (Target QP length − Observed QP length):**

  If the difference is positive, we assign more QPs the next day, since the QP system is not running at capacity; if it is negative, the QP line is too long and we assign fewer QPs the next day.

- **Finding H:** The scalar H relates the magnitudes of state and input. Since our state and our observation are on the same scale (guests in line), H is simply 1.

We now give the recursive equations for our adaptive algorithm derived from the Kalman filter. We let X_k be the number of QPs available on day k, while P_k and K_k determine how much we trust our observed data.

$$K_{k+1} = \frac{R}{P_k + R},$$
$$X_{k+1} = X_k + K_k U_k,$$
$$P_{k+1} = (P_k + V_k)\,\frac{P_k}{P_k + R}.$$

Here V_k is a measure of line fluctuation for day k, R is the expected variance (from previous data), and K_k is a scalar by which we weight the observation before we change state.

The variables X, P, and K are related, since each adjusts per iteration depending on the others. The measurement U_k is scaled by K_k, a measure of how much we trust the observation when computing the next state based on previous experience. Naturally, we expect K_k to go to 1 when there are no fluctuations, and this indeed happens; it also should approach 0 when the observed variance is very large, and it does this too. Additionally, if the observed variance is about the same as expected, then we trust the observation with scale factor of approximately 1/2.

Testing the Adaptive Algorithm

Our algorithm takes as input three parameters: P_0, K_0, and R. The first two are self-adjusting, so the filter is not sensitive to their initial values; but R strongly affects convergence of the model. Amusement parks closely guard their attendance data, so we do not have data for these parameters and must guess reasonable values. However, even with rough guesses, our adaptive algorithm settles quickly to equilibrium.

We tested the filter by iterating our computer simulation. Given initial values, the first relevant output from the Kalman algorithm is the number of QPs assigned per hour block of time. We assume that guests arrive uniformly over their assigned block. We graph how many guests arrive as a function of time, exactly the input for our computer simulation.

This test does not capture the true power of our adaptive algorithm, because we do not know the value of the parameter R, and our model for line growth is a fairly rough and simple probability model.

The Test:

1. We assume that the peak time of day is subdivided into 6 equal blocks. We "couple" the blocks by having the final queue length of block i as the initial queue length for block $i + 1$.

2. We use the same model for line formation as in our simulation. Additionally, we employ our algorithm on each block.

3. We guess the parameter R and pick initial values for K_0, P_0, and X_0 (the number of QPs per time block).

4. Subject to noise, we input this X_0 into our line-formation model. Our algorithm measures the deviation of the actual line from the ideal line.

5. The Kalman filter outputs new values for X, K, and P. We now have a new value for the number of QPs for every block.

6. We iterate this process 1,000 times. The input to step 5 is the output of step 4.

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

A pictographic version of the results of a trial is shown in **Figure 5**; a plot of the actual results is shown in **Figure 6**. The system output was programmed to vary around 100 guests/h, but we input an initial value of 120 QPs/h.

We can think of each time step as a previous similar day; for a Saturday, we look to last Saturday's data. For the first day, the line grows rapidly, as the ride cannot service the demand. By day 10, the line has visibly deformed. The distribution on day 30 is almost as good as on day 900, so the algorithm converges quickly, even with a bad initial guess, (in actual implementation we can start with very good initial values).

Justification of Uniform Arrival Rate

Guests arrive with some probability distribution throughout their block. However, we should be able to overlap the blocks in such a way that the average arrival is constant. For example, **Figure 7** shows how we overlap the blocks when guests arrive with a normal distribution about the center of their block.

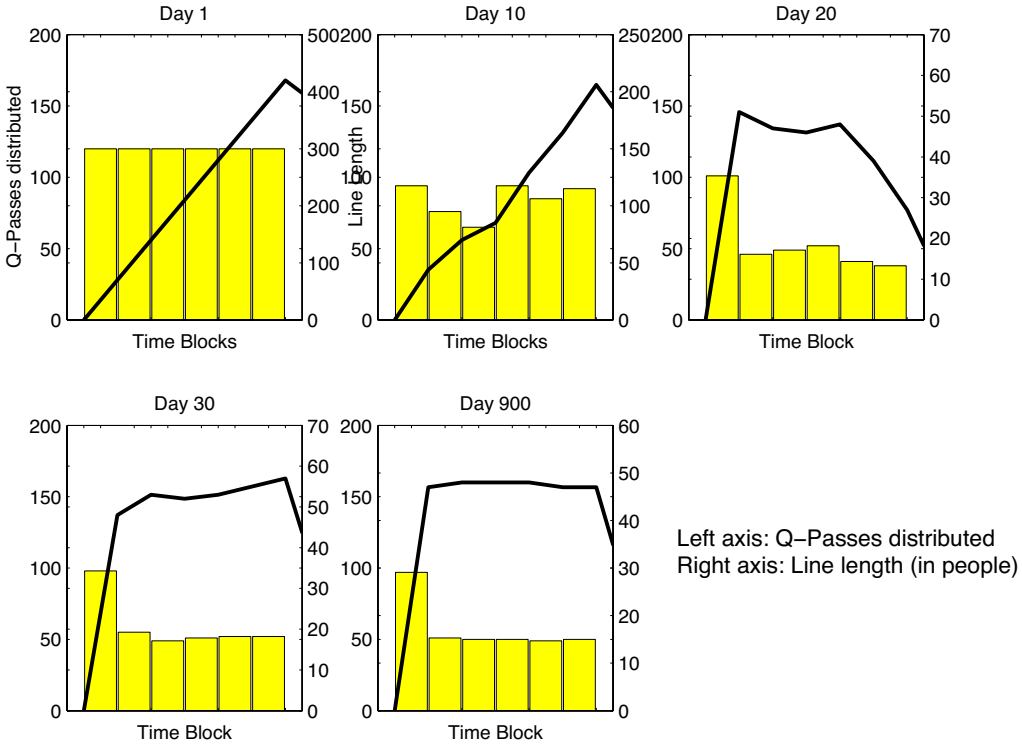

Figure 5. Number of QPs alloted per hour (histogram) and QP line length (black line) for days 1, 10, 20, 30, and 900.

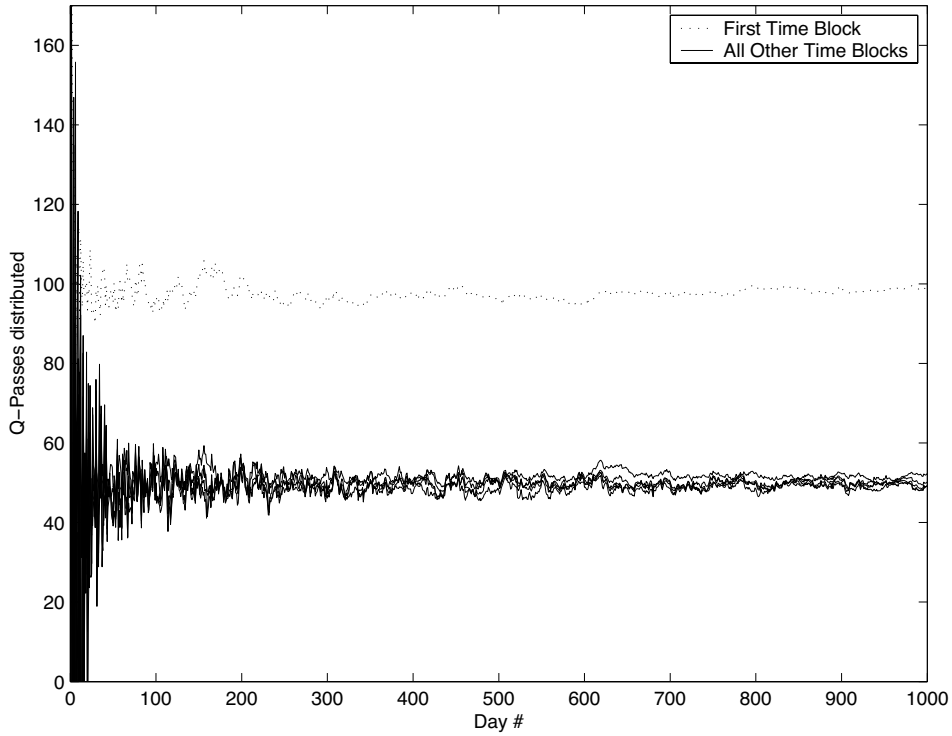

Figure 6. A plot of QPs alloted day by day, given wildly wrong initial values. Nevertheless, the algorithm stabilizes.

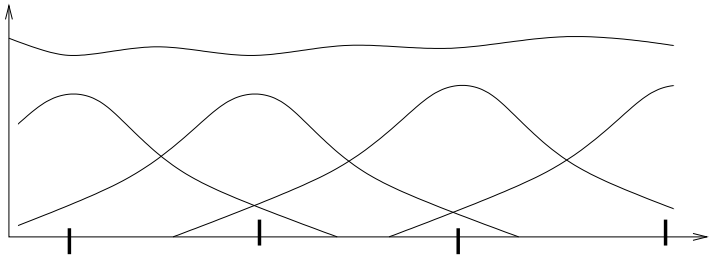

Figure 7. How to add normal distributions to achieve constant arrival.

Conclusions

The GhostQueue decreases wait times and increases happiness, but in a way hard to optimize, and fairness is difficult to define and test. It is a good solution only if an external way (such as selling access) keeps utilization low.

The KalmanQueue process meets the criteria for a good model, despite some sensitivity in its parameters. The dynamic optimization approach is also easy to use, since the system adjusts itself to meet an ideal line condition. Hence, we recommend use of a KalmanQueue system for rides with long lines.

Strengths & Weaknesses

General Model

- **Strengths**
  - Our line-formation model agrees with our rough data, and our computer model agrees with both.
  - 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 guest is more likely to join a short line than a long one.
  - We ignore the effect of virtually-queued guests joining normal lines, thus increasing the normal line wait times.
  - Our model is valid only during peak hours.
  - Our model lacks a rigorous definition of optimality.

GhostQueue

- **Strengths**
  - The return-time calculation is simple and comprehensible to the guest.

  - No guest waits longer than without the GhostQueue.

- **Weaknesses**
  - Utilization must be kept low for GhostQueue to be beneficial.
  - Fairness is a dominant factor in optimal utilization; but our assumptions do not quantify fairness, so optimizing is beyond the scope of the model.

KalmanQueue

- **Strengths**
  - The primary input is the desired behavior of the QP line, and the model adjusts itself accordingly.
  - The KalmanQueue process satisfies all six properties of a good model.
  - The Kalman filter is highly adaptive to changes in the queueing process (e.g., time-varying output rates), so the core framework of our Kalman-based algorithm is valid for a wide variety of situations.

- **Weaknesses**
  - Our model assumes a constant mixing rate. An extension of the model (a two-dimensional Kalman filter) would allow for the determination of the mixing rate based on the relative lengths of the normal and QP lines.
  - R, the random variance of the number of guests in line, must be determined accurately for the model to be useful.
  - The Kalman filter tries to decrease the *maximum* QP line length. A future model should try to decrease the *average* QP line length.

References

Cedar Point Information. 2003. The Point Online. `www.thepointol.com` .

Ruiz-Pala, Ernesto, Carlos Abila-Beloso, and William W. Hines. 1967. *Waiting-Line Models: An Introduction to Their Theory and Application.* New York: Reinhold Publishing Co.

Mouse Planet Inc. 2003. Disneyland Information Guide—What is Fastpass? `http://www.mouseplanet.com/al/docs/fast.htm` .

O'Brien, Tim. 2001. Six Flags debuts queue management. *Amusement Business* (5 March 2001). `http://www.lo-q.com/Press/Amusement%20Business%20March%202001.htm` .

_____. 2002. North American parks down slightly from 2002. *Amusement Business* 115 (51) 9.

Welch, Greg, and Gary Bishop. 2002. An introduction to the Kalman filter. `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html` .