INTERACTIVE CONTROL ENGINEERING
Principles and Applications

Volume 2 Practical PID Control

CHAPTER 9

Surge Pricing as Feedback on a Ride-Hailing Market With a Throughput Fold

Pages 224–246

Edited by John D. Hedengren

How to cite this chapter. "Surge Pricing as Feedback on a Ride-Hailing Market With a Throughput Fold," in *Practical PID Control*, John D. Hedengren, Ed., vol. 2 of *Interactive Control Engineering: Principles and Applications*. Advanced Process Solutions, LLC, 2026, ch. 9, pp. 224–246.

The page numbers in this file are those of the printed volume.

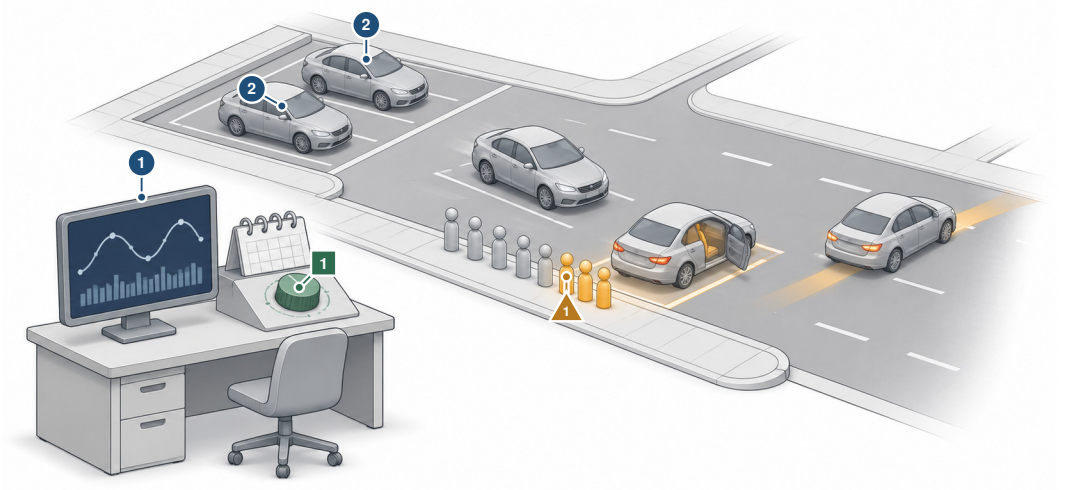
The app for this chapter
https://apmonitor.com/apps/surge-pricing/

The volume online: contents, other chapters and citation records
https://apmonitor.com/apps/books/v02-practical-pid/#surge-pricing

Copyright © 2026 Advanced Process Solutions, LLC. All rights reserved. Produced by Advanced Process Solutions, LLC.

CHAPTER 9

Surge Pricing as Feedback on a Ride-Hailing Market With a Throughput Fold



The controller moves 1 the price multiplier, changed only at a pricing decision

The controller reads 1 the time a car needs to reach a rider 2 the count of free cars

What disturbs it 1 a sudden crowd of ride requests, such as a stadium letting out

Notice the three kinds of car, parked and free, driving to a rider and carrying one: the price has to keep enough cars of the first kind. Illustration, not to scale.

IN PLAIN TERMS A Price That Keeps Rides Moving

The main idea. When many people want a ride at once, a ride app can raise its price. Fewer people then ask for rides, more drivers come to work, and enough cars stay free to reach riders quickly.

The challenge. If too few cars are free, each driver spends a long time driving empty to a far rider, so the same cars finish fewer trips. Riders answer a price at once but drivers come slowly, so a price that rises late or is held down cannot stop this.

Why it matters. The pricing rule decides whether a crowd leaving a stadium gets rides in minutes or gives up waiting.

THREE WAYS TO READ THIS CHAPTER

Explore	11 min	the first page, the fact on trips in one city (Section 9.1), the history of the moving price (Section 9.2), the try-it segments of cases 2 and 3 and the fact on the rain case (Section 9.6), and the key points
Study	50 min	the whole chapter with its three pause points and the twelve exercises at its end
Practice	20 min	the first page, the results (Section 9.6), Technology in Practice (Section 9.7), Limitations (Section 9.8) and the open challenges

The control question A surge multiplier reaches riders at once and drivers only through their earnings, and past the fold of its throughput map a zone completes fewer trips as demand grows. When does a feedback price keep the zone on the near side of that fold, and when do a slow decision calendar, a price cap or the controller's own logic let it cross?

Learning objectives

- Derive the pickup time, the abandoned share and the fixed-fleet throughput map from the three-pool model, and locate the fold.
- Explain the two speeds and the two static gains of a price, and compute the settled fleet as a power of the multiplier.
- Determine the action of the pickup-time PI from the sign of the process gain in the convention of Chapter 1.
- Predict how the decision calendar and the controller gain change the response of the PI to a stadium release.
- Compare a lookup table, the PI and a receding-horizon optimizer with and without event preview on service, abandonment and price.

Prerequisites Sections 1.4.2 and 1.6.2 (velocity form), 1.4.3 (controller action), 1.4.9 (back-calculation), 1.4.11 (sample time) and 1.4.13 (a gain that moves with the operating point). Chapter 8 holds a road below its fold. Section 1.6.3 is needed only for the optimizer.

The app https://apmonitor.com/apps/surge-pricing/

See also Volume 7, Chapter 4 for a process gain that reverses at the peak of a friction curve and Volume 7, Chapter 3 for a plant that the setpoint chooses. Volume 9, Chapter 6 for a price that coordinates consumers and Volume 9, Chapter 5 for releasing ahead of announced demand.

9.1 Introduction

A stadium that empties into a city zone asks for rides faster than any fleet can answer them, and a platform that keeps its fare unchanged answers by sending its last idle cars across the zone. In the simulator of this chapter, a release that quadruples potential requests for 40 minutes, with the fare held at its base, drains the idle pool from 432 to 20 drivers within 9 minutes. The pickup time then reaches 10.2 min, two of every five drivers online are driving empty toward a rider, and up to 2,584 of the 4,800 requests an hour are abandoned. From 12 minutes after the release to its end the zone completes 1,932 to 2,152 trips an hour while the same online fleet, allocated with a fuller idle pool, could sustain 317 to 359 more. A price that rises in time keeps the idle pool and the throughput, and a price that is capped or late does not.

Pricing a perishable capacity over time has long been posed as a control problem. The sale of an inventory that loses its value at a deadline was formulated as intensity control, with structural results for the optimal price as a function of the stock and the time remaining [1]. Ride-sharing platforms were then studied with a queueing-theoretic economic model that captures the incentives of both sides and the high temporal resolution at which they price [2], with stylized models of self-scheduling providers [3], and with networks of locations among which drivers choose where to serve [4], a literature since organized into demand and pricing, supply and incentives, platform operations, and competition, impacts and regulation [5]. Fleets of shared vehicles have been written as fluid models whose rebalancing policy comes from a linear program [6] and coordinated by model predictive control (MPC) with a Lyapunov stability proof [7]. The plant is a set of driver stocks whose flows depend on a price, the actuator is the multiplier, and the controller is a rule, a loop or an optimizer, so the zone is a control problem in the plain sense of the term.

✦ DID YOU KNOW? Nearly 159 Million Trips a Year

In New York City more than 80,000 drivers worked in 2018 for the four largest app-dispatched for-hire services, whose trips grew from roughly 42 million in 2015 to nearly 159 million in 2017 [8].

The belief this plant tests is that a higher price can only cost trips. On the idle-rich side of the market the belief holds, and at the reference demand a multiplier of 2 lowers the completed trips at steady state from 1,185 to 840 an hour. At three times that demand the same model completes 2,563 trips an hour at the base fare and 3,200 at 1.2 times it, because the price buys back the idle drivers whose absence was sending the fleet on long pickups. This failure was anticipated in the theory of dispatched taxis [9] and later modeled and established in platform data as the wild goose chase [10]. The shape is general. Traffic on a crowded road obeys a functional relation between flow and concentration [11], which Chapter 8 draws with a maximum at the road's capacity and defends with a ramp signal, and a centrifugal compressor throttled toward low flow passes the peak of its pressure-rise characteristic, which Volume 4, Chapter 2 guards with a recycle valve. In each the steady map from load to output has a maximum, and past it more load yields less output. This chapter defends such a maximum with a price whose two effects arrive at two speeds.

However, the economic models of the market are solved for equilibria and for optimal static, spatial or contractual prices [2–4, 10], while the fleet controllers move vehicles that the platform owns [6, 7], so neither gives a student one market on which a lookup table, a feedback loop and an optimizer each move a quantized price on a calendar while riders answer at once and drivers later. This chapter works with a market of this kind. It determines when a feedback price keeps a ride-hailing zone on the near side of its throughput fold, and when a decision calendar, a price cap or a controller's own logic lets the zone cross it. The pooled market, the square-root pickup law, the PI loop and receding-horizon control are all established. Every result is measured on one deterministic simulator or derived from its own equations, by linearization and by a map of steady states. Each defect found in the application during this work was corrected there and is reported with its effect before and after.

The chapter is organized as follows. Section 9.2 traces the domain from the regulated taxi to the surge multiplier. Section 9.3 states the zone with every equation, constant and limit, a market whose driver count closes to 5.3×10^{-11} drivers in every case. Section 9.4 derives its two speeds, its two gains and the fold. Section 9.5 describes the pricing rules, among them an optimizer that rolls 44 candidate price paths through the nonlinear model at each decision. Section 9.6 reports nine cases, a constant-demand reference and a 24-hour challenge, each scored on service, abandonment and price together. Section 9.7 describes the devices and platform of a real pricing loop, Section 9.8 states the limitations, Section 9.9 summarizes the results, and the exercises follow in Section 9.10.

9.2 From the Dispatched Taxi to the Surge Multiplier

The price of a taxi ride was long a constant rather than an actuator. In most cities the taxi industry is highly regulated, with restricted entry [12], and a regulated cab charges a metered fare that does not move with demand [13]. A study of 43 cities and counties in the United States and Canada found that such entry controls act differently on the two markets a cab serves. Without them the cab-stand and street-hail market draws an oversupply of cabs and a deterioration of vehicles and drivers, while in the telephone-order, or dispatch, market entry restrictions often leave taxis unavailable [14]. With the fare fixed, only the wait could adjust, and the dispatch market is the one the simulator models.

The geometry that makes waiting fall as supply grows is older than the platforms. For response units scattered over a region as a two-dimensional Poisson process of intensity γ, the mean straight-line distance from a call to the nearest unit is $\frac{1}{2}\gamma^{-1/2}$ [15], so the drive to the nearest idle car shortens only with the square root of the density of idle cars. A model of a dispatched taxi market concluded, in its own title, that taxi travel should be subsidized [9], and a later analysis of the platforms traces the conclusion to the increasing returns of a denser fleet, gives its pickup time as the inverse square

root of the density of idle taxis, and credits it with first anticipating the failure that the platforms would meet [10]. Pickup time in the simulator follows the same law in the size of its idle pool.

Ride-sharing services that connect passengers and drivers over the mobile internet joined dispatch to a price that moves. The history box below follows that price through its first measurements, and the technology boxes of Section 9.7 describe how it is measured and set. The practical problems of designing such a pricing system, and the way it coordinates workers who earn on a flexible schedule, were set out from the platform's side at the 2016 conference on economics and computation [16]. The simulator's multiplier has a floor of one, a quantum of 0.1 and a demand elasticity of 0.5.

> ⧖ HISTORY **Fifty Million Quotes at a Moving Price, 2015**
>
> Ride-sharing services that connect passengers and drivers over the mobile internet began to compete with regulated taxis. In most cities with data, the drivers of one service's core product spent a higher fraction of their time, and drove a higher share of their miles, with a passenger in the car than taxi drivers did [12]. Its price moved. In the first 24 weeks of 2015, around 21 percent of that product's sessions in its four largest United States markets saw a multiplier above 1. Because a continuous internal value was rounded to discrete steps, nearly identical market conditions on either side of a step met different prices. A regression discontinuity design on nearly 50 million sessions gave price elasticities of demand mostly between -0.4 and -0.6, and -0.55 for the whole sample [13].

The theory that followed explains why the price has to move. Optimal pricing in such a two-sided platform needs both an economic model of the two sides and a stochastic model of drivers and passengers over time [2]. With providers who decide for themselves how often to work, surge pricing is not the optimal contract but generally earns nearly the optimal profit, and as labor becomes more expensive both providers and consumers fare better under it [3]. Across a network of locations, the balance of the demand pattern captures a network's profit potential [4]. The wild goose chase is the subject of this chapter. An overburdened platform depleted of idle drivers sends cars on long pickups that occupy them, reduce the trips served and the earnings, and so take drivers off the road, and its authors established the failure in platform data and showed that surge pricing, and to a lesser extent other market design interventions, keep it from crippling the market [10].

Control engineers reached the same fleets through the vehicles. A fluid model of customers and shared vehicles at stations gives a rebalancing policy, by linear programming, under which every station reaches an equilibrium with excess vehicles and no waiting customers [6], and a discrete-time model of an autonomous fleet supports an MPC that solves a mixed integer linear program for scheduling and routing in real time [7]. Those controllers move cars. The controllers in this simulator cannot move a single car, and they reach the fleet only through a price that riders see at once and drivers feel through their earnings.

9.3 A Zone of Three Pools

The simulated zone is one aggregate market, drawn in Figure 9.1 with the riders entering at the left, the three pools of drivers in the middle and the pricing desk above. Time is in hours, pools in drivers, flows in drivers or requests an hour, the price as a multiple m of the base fare f, and pickup times in minutes where the application displays them. Table 9.1 classifies the signals.

9.3.1 Pools and Flows

Drivers move among the three pools, and only the logging on and off of drivers changes their total, as shown in Equation (9.1).

$$\begin{aligned} \dot{O} &= c - a + u_s, \qquad & \dot{E} &= a - E/\tau_p, \\ \dot{T} &= E/\tau_p - c, \qquad & c &= T/\tau_t, \qquad N = O + E + T \end{aligned} \tag{9.1}$$

where c is the rate of completed trips, a the rate of accepted dispatches, u_s the net rate at which drivers log on, $\tau_t = 0.25\,\mathrm{h}$ the passenger trip and τ_p the pickup time. A dispatch takes a driver out

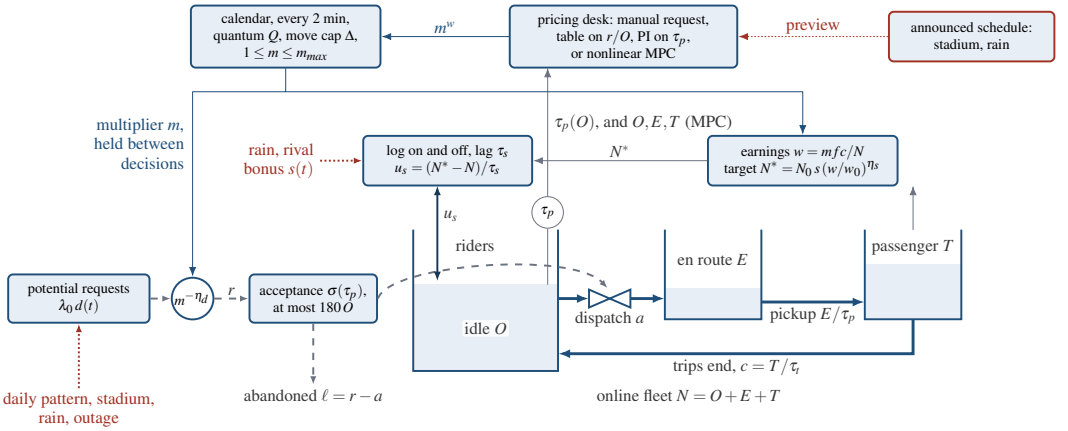


Figure 9.1 One ride-hailing zone as three pools of drivers, not to scale. Potential requests $\lambda_0 d(t)$ pass the price gate $m^{-\eta_d}$ and an acceptance test on the predicted pickup time τ_p, bounded by one dispatch per idle driver every 20 s, and the rest are abandoned. A dispatch a moves a driver from the idle pool O to the en-route pool E, the pickup moves the driver into the passenger pool T, and the end of the trip returns the driver to O. Drivers log on and off at u_s toward a target N^* set by their earnings w and the supply factor $s(t)$. The pricing desk reads τ_p (r/O for the table, and the three pools and the logged requests for the optimizer), and its request m^w becomes the multiplier m only at a calendar decision, rounded to the quantum Q and held within the move cap Δ and the bounds. Riders move along dashed gray arrows. Dotted red arrows mark the disturbances and the preview of the announced schedule, which event preview passes to the optimizer.

Table 9.1 Signals of the Zone by Role

Symbol	Role	Unit
τ_p	CV, pickup time, SP τ^* (3 to 10)	min
m	MV, multiplier in force, 1 to m_{max}	—
m^w	request of the desk or a controller	—
$d(t)$	DV, potential-demand factor	—
$s(t)$	DV, available-supply factor	—
O, E, T	states, idle, en-route and passenger drivers	drivers
N	output, online fleet $O+E+T$	drivers
r, a, ℓ	outputs, requests, dispatches, abandonments	/h
c	output, completed trips	/h
w, N^*	outputs, earnings per driver, supply target	\$/h, drivers
O^*, q^*	fold at the current N, idle pool and trips	drivers, /h

of the idle pool at once, the pickup moves the driver into a trip, and the end of the trip returns the driver to the idle pool. The quantity $N - N(0) - \int u_s \, \mathrm{d}t$ is zero in exact arithmetic and is logged as a conservation residual. The pickup time grows as the idle pool shrinks, as shown in Equation (9.2).

$$\tau_p(O) = \tau_{min} + \frac{a_p}{\sqrt{\max(0.25,\, O)}} \tag{9.2}$$

where $\tau_{min} = 2\,\text{min}$ and $a_p = 0.5$ h drivers$^{1/2}$, so 400 idle drivers give 3.5 min, 100 give 5 min, 25 give 8 min, and the floor of 0.25 drivers bounds the pickup at 62 min. The second term is the nearest-neighbor law of Section 9.2 with the zone's area and speed lumped into a_p.

Table 9.2 Events of the Market and Their Factors

Event	Effect
stadium	$\delta = 4$ for 40 min
rain	$\delta = 1.5$ and $\varsigma = 0.8$ for 2 h (40 min in case 8)
rival bonus	$\varsigma = 0.7$ for 2 h
app outage	$\delta = 0$ for 10 min, then $\delta = 3$ for 10 min

9.3.2 Riders

Requests answer the price at once, and riders decline a predicted pickup that is too long, as shown in Equation (9.3).

$$\begin{aligned} r &= \lambda_0\, d(t)\, m^{-\eta_d}, \qquad \sigma(\tau_p) = \left[1 + \exp \frac{\tau_p - \tau_{ab}}{1.5\,\text{min}}\right]^{-1} \\ a &= \min\big(\sigma(\tau_p)\, r,\ 180\, O\big), \qquad \ell = r - a \end{aligned} \tag{9.3}$$

where $\lambda_0 = 1200/\text{h}$ is the reference potential demand, $\eta_d = 0.5$, the abandonment midpoint is $\tau_{ab} = 10\,\text{min}$ and the exponent is clipped to ± 50. A predicted pickup of 5 min loses 3.4 % of requests, 8 min 20.9 % and 10 min half. The second bound lets each idle driver take one dispatch per 20-second matching cycle. Rejected and unavailable dispatches both count as abandoned, and a rider who never requests because of the price is not counted at all.

9.3.3 Drivers and Their Earnings

The fleet relaxes toward a level set by the hourly earnings of a driver, as shown in Equation (9.4).

$$\begin{aligned} w &= \frac{m f c}{N}, \qquad N^* = N_0\, s(t) \left(\frac{\max(w, 0.1)}{w_0}\right)^{\eta_s} \\ u_s &= \max\big(-6\, O,\ (N^* - N)/\tau_s\big), \qquad w_0 = f \lambda_0 / N_0 \end{aligned} \tag{9.4}$$

where $f = \$12$, $N_0 = 800$ drivers, so $w_0 = \$18$ an hour, $\eta_s = 0.5$ and $\tau_s = 0.5\,\text{h}$. Drivers leave only from the idle pool and at no more than six times its size an hour. The multiplier in force pays every trip that ends, whatever price the trip was accepted at, so w moves the instant m does. Gross fares $\int m f c \,\mathrm{d}t$ are logged and are neither the platform's profit nor the riders' welfare.

9.3.4 Demand, Supply and Events

The two disturbance factors combine a slider, a daily pattern and the events of Table 9.2, as shown in Equation (9.5).

$$\begin{aligned} d(t) &= D\, p(h) \prod_i \delta_i(t), \qquad s(t) = S \prod_i \varsigma_i(t) \\ p(h) &= 0.8 + \mathrm{e}^{-((h-18)/1.8)^2} + 0.3\,\mathrm{e}^{-((h-8)/1.2)^2} \end{aligned} \tag{9.5}$$

where D (0.2 to 4) and S (0.3 to 1.5) are sliders at 1, h is the hour of the day from the case's starting hour, $p(h)$ is replaced by 1 unless the daily pattern is on, and δ_i and ς_i are the factors of the events in force. The pattern is 1.80 at 18:00, 1.10 at 08:00 and 0.80 through the night. A scripted event is announced in the case's schedule, and an event fired from the panel joins the same list at the moment it is fired, after which a forecast can see the rest of it. Nothing in the model is random.

9.3.5 The Fold

At a steady state every dispatched driver spends τ_p en route and τ_t in a trip, so $E = c\tau_p$ and $T = c\tau_t$, and a fleet of N drivers with O of them idle can sustain the trip rate of Equation (9.6).

$$q(O; N) = \frac{N - O}{\tau_t + \tau_p(O)} \tag{9.6}$$

Table 9.3 Teaching Values and Where Each Is Set (P a Model-Dialog Parameter, C a Card Control, A an Advanced Control, S the Source)

Symbol	Meaning	Default	Range		Symbol	Meaning	Default	Range	
N_0	baseline drivers	800	200–1600	P	τ^*	pickup target (min)	3.5	3–10	C
λ_0	baseline requests (/h)	1200	300–2400	P	K_c, τ_I	PI gain (/min), integral time (h)	0.45, 0.25	0–2, 0–2	C
η_d, η_s	demand, supply elasticity	0.5, 0.5	0.2–1.5, 0–1.2	P	h	surge-off hysteresis (min)	0.3	0–1.5	C
τ_s	supply lag (h)	0.5	0.1–2	P	m_{max}	price cap, off as shipped	2.5	1–6	C
τ_{min}, a_p	minimum pickup (min), geometry (h drivers$^{1/2}$)	2, 0.5	1–5, 0.1–1.5	P	—	decision calendar	2 min	10 min, 30 s	C
τ_t	passenger trip (h)	0.25	0.1–0.5	P	Q, Δ	quantum, largest move per decision	0.1, 1	0–0.5, 0.1–3	A
τ_{ab}	abandonment midpoint (min)	10	5–30	P	H	forecast horizon (h)	1.5	0.5–2	A
f	base fare ($/trip)	12	5–30	P	w_p, w_a	wait, abandonment weights	1, 2	0–5, 0–10	C
D, S	demand, supply sliders	1, 1	0.2–4, 0.3–1.5	C	w_u, w_c	price-move, trips weights	0.1, 0.4	0–5, 0–5	A
—	abandonment scale (min), dispatch cycle (s)	1.5, 20	—	S	w_e, w_r	earnings, gross-fare weights	0, 0	0–10, 0–10	A
—	exit limit (/h), earnings floor ($/h)	6O, 0.1	—	S	—	price-level weight, in the MPC	0.22	—	S
—	late pickup, challenge (min)	8	—	S	Δt, Δt_c, log	physics, control (s), log (min)	10, 30, 1	—	S

The map vanishes when every driver is idle and falls again as $O \to 0$, because the pickups lengthen, so it has an interior maximum q^* at an idle pool O^*, the fold. The same accounting of idle, en-route and driving drivers underlies the analysis of [10]. The application finds the maximum at the current fleet by ternary search, draws the curve and the live point in its Goose chase view and raises the WILD GOOSE CHASE status whenever $O < O^*$. The live point is transient and need not lie on the curve.

9.3.6 Start, Integration and Checks

Every case starts at the equilibrium of Equations (9.1) to (9.4) at $m = 1$ and constant demand and supply, found by bisection on the idle pool and then relaxed through the complete model, with $O = 432.41$, $E = 68.00$, $T = 296.26$ and $N = 796.66$ drivers and a pickup of 3.443 min. There 1.25 % of the 1,200 requests are abandoned, so earnings are $17.85 an hour, a little below w_0, and the fleet settles 3.34 drivers below N_0. The framework's fourth-order Runge–Kutta (Section 1.6.1) advances the state on a step of $\Delta t = 10$ s (1/360 h), the controller executes every 30 s and the historian logs every simulated minute. At the fastest speed, three simulated hours a second, the four hours of case 2 run in under two seconds. Over the full duration of every case the conservation residual stays below 5.3×10^{-11} drivers and no pool falls below 9.31 drivers (the idle pool of case 8), halving the step changes case 8's peak pickup by 0.0007 min and its 3,338 abandonments by 0.01, and the reference holds its equilibrium for three hours. The application passes its own 12 checks and the framework's 17 shared ones.

9.3.7 Assumptions

One zone, no streets. The zone is one aggregate market, so drivers never relocate, demand has no destinations and the pickup law averages over the zone. Spatial imbalance between locations, which shapes optimal prices on a network [4], is outside the model.

Pools, not agents. Drivers are fluid pools that answer current earnings without delay, strategy or memory, and the first idle driver is always dispatched, with no limit on the dispatch radius. The City view's dots illustrate the pools and are not simulated individually. The fluid model is least credible when the idle pool falls to the 13 and 9 drivers reached in cases 2 and 8.

Riders without a queue. A rider accepts or declines the predicted pickup at the moment of request, so there is no age-structured queue of unmatched riders, and riders the price deters are not counted. Requests are expected values and need not be whole numbers.

Teaching values. Every elasticity, lag, geometry coefficient, threshold and event size is chosen to expose a mechanism, and the model is not fit to any city. The demand elasticity lies inside the range estimated from platform data [13], the pickup law follows the nearest-neighbor result [15], and the supply elasticity and lag are not taken from a published estimate.

9.4 Two Speeds and a Fold

Near the reference the zone is well damped. The Jacobian of Equations (9.1) to (9.4) in (O, E, T) at $m = 1$, by central differences of the application's own flow function, has eigenvalues of −17.72/h, the en-route pool draining at the pickup rate in 3.4 min, and a pair at $-3.36 \pm 0.36j$ per hour, an 18 min mode that couples the trips with the fleet's supply response. The input column is $(1389, -593, 0)$ drivers an hour per unit of m. The price enters the idle pool through the dispatches it prevents and through the drivers its earnings attract, and it reaches the passenger pool only through the en-route pool.

9.4.1 Two Gains

The price acts on the two sides with very different static gains. At the reference demand a multiplier of 2 held for six hours removes 29.3 % of the requests (1,200 to 848.5 an hour) but adds only 12.3 % to the fleet (796.66 to 894.81 drivers), and the idle pool grows from 432 to 640, the pickup falls from 3.44 to 3.19 min, the completed trips fall from 1,185 to 840 an hour and the earnings rise from \$17.85 to \$22.52. With little abandonment, $c \approx r$, so $w \propto m^{1-\eta_d}/N$ and $N \propto w^{\eta_s}$, and the fleet follows $m^{(1-\eta_d)\eta_s/(1+\eta_s)}$, the sixth root of m at the defaults. A multiplier can therefore buy far fewer drivers than it removes riders, and in steady state its main effect on the idle pool comes from the demand side.

9.4.2 Two Speeds

The two sides also answer at different speeds, as shown in Figure 9.2(b). With the reference run in MANUAL and the multiplier request raised from 1 to 2 at hour 0.5, the requests fall by the factor $2^{-1/2}$ at the decision itself. The earnings double at the same instant, from \$17.85 to \$35.70 an hour, because the trips then ending are paid at the new price, so the supply target jumps from 796.7 to 1,126.6 drivers and the fleet starts to grow at once. Timed from the decision, it covers half of its eventual rise in 5.3 min, 63 % in 7.3 min and 95 % in 13.8 min, then overshoots to 913.2 drivers 33 min after the move, while the trip pool empties and the earnings fall back, and settles at 894.8. The lag τ_s is therefore a first-order relaxation and not a dead time. Paying each trip at the multiplier in force when it was accepted would hold the earnings signal back by a pickup and a trip, about 18 min here, but would move every run in which the price moves, so the application keeps the jump and its texts now state it.

9.4.3 The Fold and a Gain That Changes Sign

The fold belongs to the allocation of a fleet and not to any controller. At the fixed fleet $N_0 = 800$, which the model reaches with $\eta_s = 0$, the map $q(O; 800)$ peaks at 2,128.3 trips an hour with 65.66 idle drivers and a pickup of 5.70 min, as shown in Figure 9.2(a). The steady states at the base fare for potential demand from 0.5 to 5 times the reference lie on that curve and walk left along it. At 1.5 times the reference the zone keeps 241.9 idle drivers and completes 1,769 trips an hour, the steady state crosses the maximum at 1.875 times, and beyond it every increase completes fewer trips, 2,069 at twice the reference, 1,897 at three times and 1,839 at four times, with pickups of 7.2, 9.8 and 10.7 min and 13.8 %, 47.3 % and 61.7 % of requests abandoned. These states are reached by integrating the model, so they are stable, and abandonment is the likely reason, because a longer predicted pickup turns more riders away and spares the idle pool.

With the fleet free to answer its earnings, the fold moves with the fleet, but the consequence for a controller is the same, a plant gain that changes by more than an order of magnitude and a trip response that changes sign. Section 1.4.13 shows what such a move does to the loop gain $K_c K_p$ of a

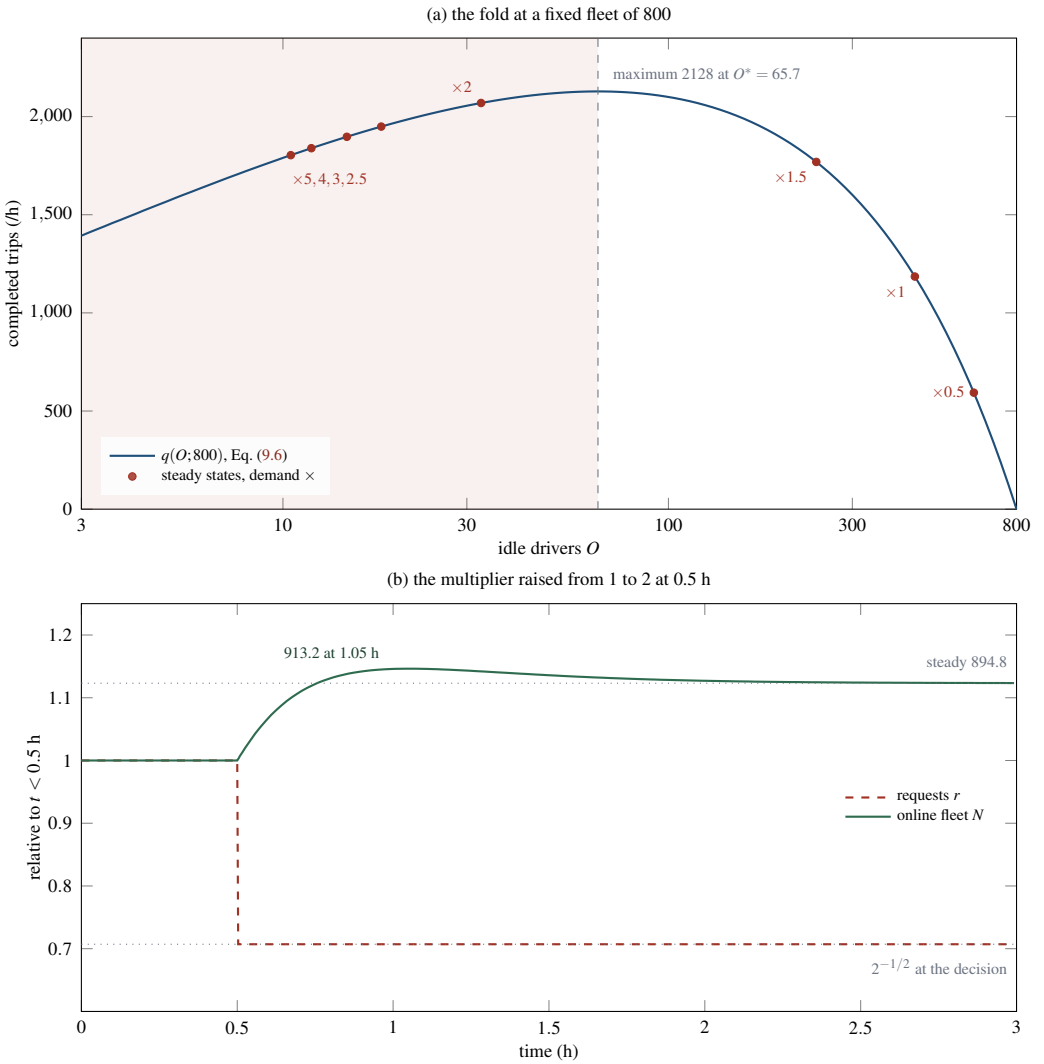


Figure 9.2 Two characteristics of the zone from the application's own functions and engine. (a) The fixed-fleet throughput map of Equation (9.6) at $N_0 = 800$ and the steady states that the full model reaches at the base fare, with the fleet held at N_0, for potential demand from 0.5 to 5 times the reference, drawn as dots on the solid curve. The shaded side is the wild goose chase. (b) The reference with the multiplier request raised from 1 to 2 at hour 0.5, requests (dashed) and online fleet (solid) each divided by its value before the move.

fixed tuning. The steady-state sensitivity of the pickup time to the multiplier, taken between 1 and 1.1, is –0.49 min per unit at the reference demand, −1.27 at 1.8 times, −6.96 at 2.5 times and −10.0 at three times. At three times the reference a multiplier of 1.2 instead of 1 shortens the steady pickup from 8.64 to 4.59 min and raises the completed trips from 2,563 to 3,200 an hour, and at four times a multiplier of 1.5 raises them from 2,373 to 3,460. On the idle-rich side a higher price costs trips, and beyond the fold it buys them, which is the sense in which surge pricing solves the wild goose chase [10]. Volume 7, Chapter 4 meets the same reversal of the process gain at the friction peak of a braking tire.

🕮 **Pause.** Try Exercise 9.2 before reading on.

9.5 Pricing on a Calendar

The controller executes every 30 s, $\Delta t_c = 1/120$ h, and an execution becomes a pricing decision only when the calendar says so. At a decision the selected rule forms a request m^w, and the request is shaped into the multiplier that then holds until the next decision. The manual desk and every automatic rule share the shaping.

9.5.1 The Calendar and the Shape of a Price

Both pieces come from the framework module `core/cadence.js`. The calendar decides every 2 minutes as shipped, every 10 minutes, or at every 30 s execution, and each instant is taken by the first execution that reaches it, beginning at $t = 0$. Both intervals are whole numbers of executions, so the realized period is the nominal one. A decision turns the request into the multiplier in force by Equation (9.7).

$$\begin{aligned} \tilde{m} &= Q\,\mathrm{round}(m^w/Q), \qquad \Delta_Q = Q\max\left(1, \lfloor \Delta/Q \rfloor\right) \\ m_k &= \Pi_{[1,m_{max}]}\left(m_{k-1} + \mathrm{sat}_{\Delta_Q}(\tilde{m} - m_{k-1})\right) \end{aligned} \tag{9.7}$$

where $Q = 0.1$, the move cap is $\Delta = 1$ per decision, m_{max} is the price cap when it is on and 6 otherwise, ties round away from zero, the bounds are moved onto the grid first and a request that is not finite leaves the multiplier unchanged. The application's check over the 150 decisions of case 7 finds every decision on a 2-minute instant, every multiplier on the 0.1 grid and no move larger than the cap, and consecutive decisions there are exactly 120 s apart.

9.5.2 The Desk and the Table

In MANUAL the learner's slider, 1 to 6 in steps of 0.1, sets the request, which the next decision shapes like any other, and a helper button returns it to the base fare. The lookup table reads the requests per idle driver at the price in force, $\rho = r/\max(1, O)$, and asks for 1 when $\rho < 4$, 1.5 when $\rho < 8$, 2.5 when $\rho < 16$ and 4 above. Because r carries the price in force, the price enters the table's own input with no delay.

9.5.3 A PI Loop on Pickup Time

The PI loop is the framework's velocity-form PID with no derivative, executed at decisions only, as written in Equation (9.8).

$$\begin{aligned} e_k &= \tau_{p,k} - \tau^*, \qquad \Sigma_k = \Sigma_{k-1} + K_c e_k - P_{k-1} \\ \hat{m}_k &= \Sigma_k + \mathscr{I}_{k-1}, \qquad m_k^w = \Pi_{[1,m_{max}]}(\hat{m}_k) \\ \mathscr{I}_k &= \mathscr{I}_{k-1} + \Delta t_k \left[K_c e_k/\tau_I + (m_k^w - \hat{m}_k)/T_t\right] \end{aligned} \tag{9.8}$$

where $K_c = 0.45$ per minute of pickup, $\tau_I = 0.25$ h, $T_t = 0.1$ h and Δt_k is the interval back to the last decision. These are Equations (1.23) to (1.26) of Section 1.6.2 with $\tau_D = 0$, in the symbols $\Sigma_k = S_k$, $\hat{m}_k = v_k$, $m_k^w = u_k$ and $\mathscr{I}_k = I_k$, with the bias held in the integral. P_{k-1} is the proportional term $K_c e_{k-1}$ formed at the last decision with the gain then in force, so the running sum is never clipped and a gain change acts on the whole error at once. Back-calculation acts on the integral $\mathscr{I}_k$ only [17]. An integral time of zero removes the integral and its back-calculation.

The error $e_k = \tau_{p,k} - \tau^*$ is the framework's $a(\mathrm{SP} - \mathrm{PV})$ with $a = -1$. A higher price shortens the pickup, so the process gain is negative, and the controller must raise its output as the PV rises. A long pickup raises the price. In the convention of Section 1.4.3 this is a direct-acting controller, and the loop gain aK_cK_p is positive. The app's briefing for case 4 also calls it direct-acting. Named by its push against the pickup error, the same loop would be called reverse-acting, so the error equation, not the name, fixes the sign.

🕮 **Pause.** Try Exercise 9.1 before reading on.

When the shaped multiplier differs from the request by more than one quantum, because the move cap held it, the integral is corrected again by $(m_k - m_k^w)\min(1, \Delta t_k/0.1\,\mathrm{h})$. In MANUAL, or while

the table or the optimizer owns the price, the loop tracks the multiplier in force, which makes every switch between rules or modes bumpless (Section 1.4.10).

The surge-off hysteresis $h = 0.3$ min acts on the request after the PI. A latch turns on when $\tau_p > \tau^* + h$ or $m_k^w \geq 1.3$ and off when $\tau_p < \tau^* - h$, and while it is off any request below 1.3 is replaced by 1. A band of zero removes the latch and this floor. As first built, only the pickup could set the latch, and Section 9.6 measures both versions.

9.5.4 A Receding-Horizon Price

The optimizer plans six blocks of price over a horizon of $H = 1.5$ h by direct shooting on the simulator's own flow function, evaluated with the declared values of Table 9.3 and the base fare in force. It is a nonlinear model predictive controller [18] in that a nonlinear model predicts each candidate and only the first move is applied, but it is solved by a bounded search rather than a gradient method. It differs from the linear controller of Section 1.6.3, which solves a quadratic program on a linear model. The horizon is cut into 90 steps of one minute and the blocks into 15 minutes each, and every candidate path is rolled forward from the present (O, E, T) by six midpoint substeps per minute with the pools clipped at zero. The cost of Equation (9.9) is summed along the roll-out.

$$J = \sum_{j=0}^{89} \frac{1}{60}\Big[w_p(\tau_{p,j} - \tau^*)_+^2 + \frac{12 w_a \ell_j}{\lambda_0} + 0.22(m_j - 1)^2 - \frac{w_c c_j}{\lambda_0} - \frac{w_e w_j}{20} - \frac{w_r m_j c_j}{\lambda_0}\Big] + w_u \sum_{b=0}^{5} (m_b - m_{b-1})^2 \tag{9.9}$$

where τ_p is in minutes, the stage terms are evaluated at the start of each minute, m_{-1} is the multiplier in force, and the weights default to $w_p = 1$, $w_a = 2$, $w_c = 0.4$, $w_u = 0.1$ and $w_e = w_r = 0$. The level term $0.22(m-1)^2$ is fixed in the source and is what keeps a permanent high price from looking free. With event preview on, the forecast at minute j is the market of Equation (9.5) at $t + j/60$ with D and S at their declared value of 1, including the daily pattern, the announced schedule and any event already fired, and its demand is scaled by the requests logged at the price in force over those the declared market gives at that price, a ratio of exactly 1 in every case reported here. With preview off, the present market so formed is held over the horizon.

The search starts from the previous plan, tries six constant paths at 1, 1.5, 2, 3, 4 and m_{max}, then moves each block by ± 0.8, ± 0.3 and ± 0.1 in turn, keeping any improvement, 43 evaluations in all and a 44th to draw the prediction. Every candidate is projected so that the first block lies within Δ of the multiplier in force, each later block within $n_b \Delta$ of the one before, and all within $[1, m_{max}]$, where n_b is the number of decisions in a block, 7.5 at the 2-minute calendar, 1.5 at the 10-minute one and 30 at every execution. The first block is the request, rounded at the decision by Equation (9.7), and the plan is solved again at the next decision. One decision therefore costs 51,480 evaluations of the flow function, and the search is local, with no claim to a global optimum. As first built, the projection allowed only Δ between blocks, so with $\Delta = 0.1$ the preview case began its pre-surge at hour 0.73 instead of 1.17 and peaked at 4.37 min instead of 4.11, with 77.8 instead of 75.0 abandonments at a mean of 1.454 instead of 1.398.

9.5.5 Zone, Predictions and Literature

The simulation model is Equations (9.1) to (9.5), integrated with every clip. The table and the PI hold no model at all. The optimizer's internal model is the same flow function at the declared values of Table 9.3, integrated by a different method, with pools clipped at zero, the declared market either previewed or frozen and scaled by the logged requests, the block-to-block move limit above and the three pools read from the platform's books without noise. An edited coefficient or supply factor is therefore a model mismatch, as on a real plant (Section 1.6.3), that it meets only through these measurements, and the PI and the table read only the logged pickup time, requests and idle drivers.

Table 9.4 The Cases and the Challenge

#	Change from the defaults	Teaching point
1	MANUAL at $m = 1$, daily pattern from 15:00, 6 h	two speeds by hand
2	MANUAL at $m = 1$, stadium at hour 1, 4 h	the fold
3	table, every 10 min, daily pattern from 15:00, 6 h	a relay
4	PI, daily pattern from 15:00, 6 h	a band and a quantum
5	MPC, preview off, $\Delta = 0.1$, stadium at hour 2, 4 h	no foresight
6	case 5 with preview on	pre-surge
7	PI, daily pattern from 16:00, rain at hour 1, 5 h	both sides move
8	MPC, cap 1, stadium and 40 min of rain at hour 1, 4 h	infeasible
9	MPC, $w_e = 3$, stadium at hour 2, 4 h	whose objective
ch	PI, $\tau^* = 5$, cap 2, preview off, daily from 00:00, rain at 17:00, stadium at 23:00, 24 h	a city day
ref	MANUAL at $m = 1$, 3 h	the balance

Table 9.5 Measured Cases Over Their Full Duration

Case	peak τ_p (min)	least O	aband.	completed	mean m	moves
1	3.76	289.6	130.2	9423	1.000	0
2	10.23	13.3	1420.8	5779	1.000	0
3, every 10 min	3.56	368.6	112.5	8902	1.125	16
3, every 2 min	3.54	381.9	112.2	8894	1.125	68
4, $h = 0.3$	3.71	308.1	117.2	9037	1.078	8
4, $h = 0$	3.60	353.2	117.0	9043	1.074	4
5	5.22	86.7	98.7	5994	1.287	27
6	4.11	202.1	75.0	5797	1.398	24
7, PI	4.15	195.3	116.6	8346	1.496	18
7, MPC	3.79	282.0	108.3	8284	1.497	18
7, $m = 1$	9.33	16.7	1925.0	8528	1.000	0
8, cap 1	11.83	9.3	3338.1	5462	1.000	0
8, cap 2.5	9.02	18.2	656.2	5874	1.613	7
9, $w_e = 3$	3.54	377.6	48.1	4522	2.470	42
9, $w_e = 0$	4.15	195.0	75.2	5804	1.396	23
ref	3.44	432.4	44.9	3555	1.000	0

Abandoned and completed are totals over the case, and every run conserves drivers to 5.3×10^{-11}. Variants are measured through Reset.

The models these stand for are the steady-state accounting and calibrated market of [10], the queueing model of [2] and the dispatch model of [9], whose coefficients are not copied here. Nothing in the application is fit to data.

9.6 Eleven Runs of One Zone

Table 9.4 lists the cases. Each starts from the defaults of Table 9.3 and changes only what its row lists. The app lists the challenge as case 10 and the reference as case 11. The values below were measured headless on the current engine. A setting chosen before Run is included in the priming decision at $t = 0$, exactly as the same setting carried through Reset, which keeps the reader's settings (Section 1.6.1). Every variant was checked on both startup paths. Table 9.5 gathers the main values, and Figure 9.3 draws three of its columns. Each case but the challenge is credited once it has run for its full duration and the reader records the comparison of pickup, price and driver pools that it is built around. The challenge is judged on the case as given, so a changed model parameter or disturbance voids the attempt.

9.6.1 Rush Hour by Hand and a Stadium at the Base Fare

The first case runs the evening rush from 15:00 at the base fare. Requests grow from 1,309 an hour at 16:00 to 2,160 at 18:00, the pickup peaks at 3.76 min at 17:59 with 289.6 drivers idle, and 130.2 of

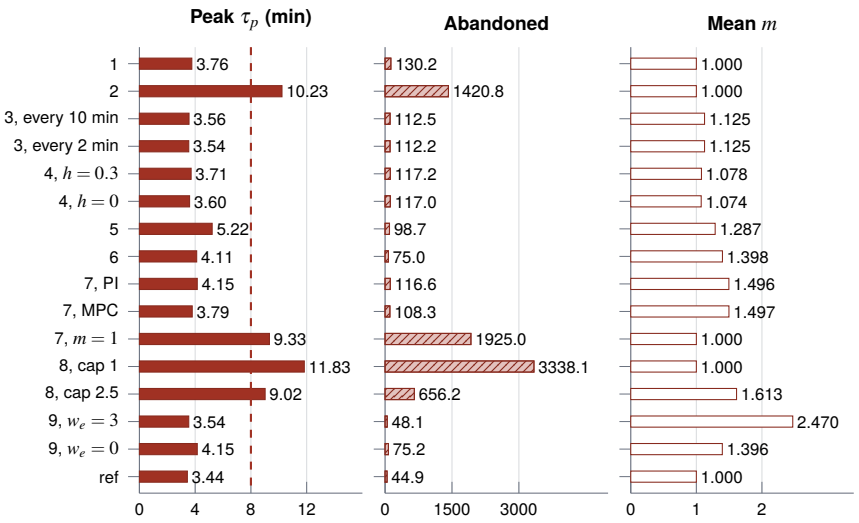


Figure 9.3 Three columns of Table 9.5, each case over its full duration: the peak pickup time (solid bars) with the 8 min limit dashed, the abandoned requests (hatched) and the mean multiplier (open). Notice that every run whose peak crosses the limit, case 2, case 7 at $m = 1$ and both runs of case 8, abandons more requests than any run that stays below it, and that case 9 with $w_e = 3$ abandons the fewest requests of the stadium runs at the highest mean multiplier of the table.

9,518 requests are abandoned, so the rush alone never approaches the fold. A request of 1.5 made at 16:30 takes effect at that decision, cutting the requests from 1,559 to 1,273 an hour, and the fleet grows from 814.6 to 928.0 drivers by 17:00 and 1,020.5 by 19:00. The pickup never exceeds 3.54 min, the abandonments fall to 95.3, and the six hours complete 8,115 trips instead of 9,423 for $135,470 of gross fares instead of $113,073.

The second case holds the base fare while a stadium quadruples the potential requests for 40 minutes from hour 1. The idle pool falls below the fold 7.2 min after the release, the pickup passes 8 min at 8.3 min and peaks at 10.23 min with 13.28 drivers idle, and between 40.7 and 41.5 % of the fleet drives empty to a rider from hour 1.2 to the end of the event (both ends included), against 8.5 % before it. The fleet grows to 965.7 drivers by hour 1.93 because the earnings rise, but over the same window the throughput stays 317 to 359 trips an hour below the maximum of the current fleet. The state at hour 1.5 is shown in Figure 9.4. The zone is in the wild goose chase from hour 1.119 to 1.711, and when the crowd has gone the refilled idle pool lets it complete 2,423 trips an hour at hour 1.78, more than at any moment of the event. The four hours abandon 1,420.8 requests and keep the pickup below 8 min 86.6 % of the time.

> **TRY IT IN THE APP** **2 · No Surge and the Capacity Fold**
>
> Load the case and press Run. Watch the status line and the live point of the Goose chase view after the stadium empties at hour 1. The status turns to WILD GOOSE CHASE, the point slides onto the shaded side of the curve, and the pickup climbs past 8 minutes to about 10.2 min while the price stays at 1.

Under any automatic rule with the shipped settings the same stadium never reaches the fold, although the PI passes within 7.5 idle drivers of it. The PI peaks at 5.36 min, the table at 3.71 and the optimizer at 3.73 without preview and 4.15 with it. The calendar decides whether the PI keeps up. Deciding every 30 s it peaks at 5.10 min with 45 moves, every 2 minutes at 5.36 with 17, and every 10 minutes the zone spends 0.28 h in the goose chase, the idle pool falls to 13.4 drivers, the pickup reaches 10.19 min and 232.8 requests are abandoned. Its gain matters as much. With K_c of

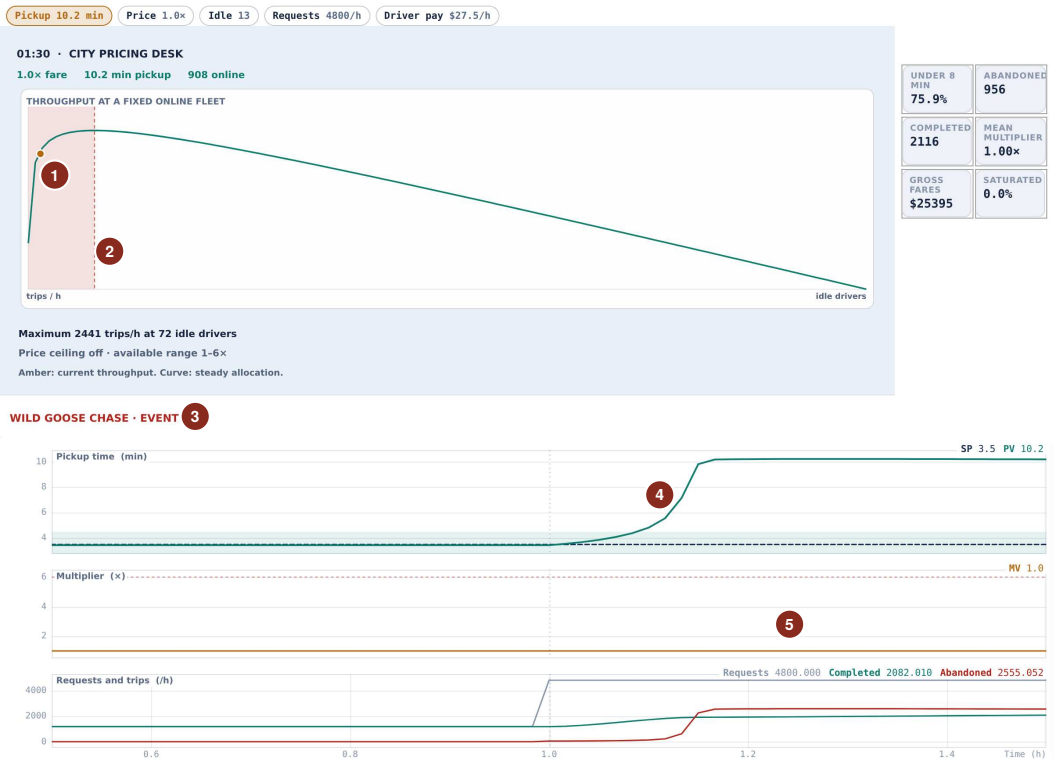


Figure 9.4 Case 2, the stadium at the base fare, at $t = 1.50$ h in the Goose chase view, in MANUAL as loaded, with a one-hour trend window. The figure is a composite of the screenshot: the plant view above the trends, with six tiles of the Learn panel framed at its right and the header and Control panel cut. The clock reads 01:30, although the time, a running sum of steps, is $1.5 - 4.7 \times 10^{-15}$ h, because it counts whole minutes with a tolerance. The badges read pickup 10.2 min, price 1.0, idle 13, requests 4,800 an hour and driver pay $27.5 an hour, and the status line (3) reads WILD GOOSE CHASE and EVENT. The plot holds the online fleet at its present 908 drivers, with a maximum of 2,441 trips an hour at 72 idle drivers, and the live point (1) sits on the shaded side, at 13.4 idle drivers and 2,082 trips an hour by the run's record, left of the dashed line at the maximum (2). In the trends the pickup (4) climbs from just below its target of 3.5 min, 3.44 min by the run's record, after the release at hour 1, the multiplier (5) stays at 1, and the zone completes 2,082 trips an hour while 2,555 of the 4,800 requests an hour are abandoned. The tiles read 75.9 % of the time below 8 min, 956 abandoned, 2,116 completed and $25,395 of gross fares, and the Saturated tile reads 0.0 %, the share of time the price request has sat at its cap.

0.1, 0.2, 0.45, 1 and 2 the peak pickup is 8.80, 7.14, 5.36, 4.50 and 4.08 min, because the stadium is a disturbance that the price can meet only if it moves within minutes.

9.6.2 A Table That Chatters

The third case prices the evening rush with the lookup table every 10 minutes. From hour 1.50 to 4.17 the table alternates between 1 and 1.5, moving at 16 of its 17 decisions. At the decision at hour 1.5 the ratio ρ is 4.11 at $m = 1$ and would be 3.36 at $m = 1.5$, on either side of the threshold of 4, so each move reverses the one before. Deciding every 2 minutes gives 68 moves and every 30 s 264, while the mean multiplier stays 1.125, the abandonments 112.5, 112.2 and 112.1 and the peak pickup 3.56, 3.54 and 3.53 min. The table is a relay closed through the price's direct effect on its own input,

so every move reverses the last on any calendar, and only the number of price changes riders see depends on the calendar. The table has no band around its thresholds, the usual remedy for a relay that chatters.

> **TRY IT IN THE APP 3 · A Supply–Demand Lookup Table**
>
> Load the case and run it. From hour 1.5 the multiplier trend steps between 1 and 1.5 at almost every 10-minute decision, while the pickup never passes 3.56 min. Then set the decision calendar to every 2 minutes, press Reset and run again. The price moves 68 times instead of 16, and the pickup hardly changes.

9.6.3 A Band, a Floor and a Quantum

The fourth case prices the same rush with the PI. As first built, only a pickup above 3.8 min could set the latch, which a case peaking at 3.709 min never does, so every request below 1.3 became 1 while the PI asked for 1.291 to 1.311, and the price alternated between 1.0 and 1.3, 30 moves from hour 2.43 to 4.03. With the latch also set by a request of 1.3, the price jumps from 1 to 1.3 at the decision at hour 2.43, when the PI's request reaches 1.31, and the latch stays on for the rest of the case because the pickup never falls below 3.2 min (its least value after the latch is 3.39 min). Five moves between 1.2 and 1.3 follow from hour 3.00 to 3.23, while the PI asks for 1.247 to 1.251, on the boundary between two quanta, and the price returns to 1.1 at hour 4.17 and to 1 at 4.73, 8 moves in all, at a mean of 1.078 with 117.2 abandonments. With no hysteresis the price moves four times, to 1.1 at hour 1.63, 1.2 at 2.43, 1.1 at 3.90 and 1 at 4.63, at a mean of 1.074 with 117.0 abandonments. The band delays the first surge by 48 minutes, makes it three quanta tall and cuts the time the pickup spends above its target from 2.26 to 1.56 h. Bands from 0.3 to 1.5 min give the same run, 0.2 gives 10 moves and 0.1 gives four, and the reversals that remain come from the quantum, not from the band. Gain makes its own switching, because with no hysteresis and $K_c = 2$ the price moves 20 times with 15 reversals.

9.6.4 Foresight Against a Rate Limit

Cases 5 and 6 release the stadium at hour 2 to the optimizer with a move cap of 0.1 per decision. Without preview it sets 1.1 at hour 0.17 and holds it until the crowd arrives, then climbs 0.1 at every decision to 2.4 at hour 2.40, and the pickup peaks at 5.22 min at hour 2.27 with 86.7 drivers idle. With preview it begins to raise the price 0.83 h before the event and reaches 2.2 when the crowd arrives, with 907.5 drivers online and 637.0 idle at the release against 809.7 and 461.6 without preview. Its peak pickup is 4.11 min and comes at the end of the event, after the optimizer, knowing the crowd will leave, has already lowered the price to 1.5. Preview cuts the abandonments from 98.7 to 75.0 and raises the mean multiplier from 1.287 to 1.398, and it also suppresses 234 requests and completes 197 fewer trips (5,797 against 5,994), so foresight buys service with price and not with capacity.

The advantage belongs to the rate limit. With the shipped move cap of 1 per decision the blind optimizer reaches its price in two decisions, peaks at 3.73 min and abandons 75.3 requests at a mean of 1.3025, while preview peaks at 4.15 and abandons 75.2 at 1.396. With a move cap of 0.3 the two abandon 78.0 and 75.2. With a move cap of 0.1 and decisions every 10 minutes the blind optimizer falls into the goose chase (9.29 min, 695.6 abandoned) and preview holds 4.01 min and 75.2. Each pair runs the same through Reset as before Run. A forecast substitutes for actuator speed, and where the actuator is fast enough it adds little. Volume 9, Chapter 5 uses announced demand in the same way, releasing water one travel time ahead of published orders.

Pause. Try Exercise 9.3 before reading on.

9.6.5 Rain on Both Sides

The seventh case runs the evening rush from 16:00 and adds rain from 17:00 to 19:00, which raises the requests by half and cuts the supply target by a fifth at the peak of the rush. At the base fare the zone falls into the wild goose chase from 17:17 to 19:03, the pickup reaches 9.33 min, only 67.2 %

of the time stays below 8 min and 1,925.0 requests are abandoned. The PI first raises the price at 17:04, after the rain has begun, reaches 2.0 at 18:02 and peaks at 4.15 min with 116.6 abandonments at a mean of 1.496. The optimizer, whose preview includes the daily pattern and the announced rain, raises the price at the priming decision at 16:00, reaches 2.0 at 17:22 and peaks at 3.79 min with 108.3 abandonments at a mean of 1.497, and the table peaks at 3.76 min with 118.8 and 58 moves. As first built, the case ran at the reference demand, where even the base fare peaked at only 4.04 min.

> ✦ DID YOU KNOW? **More Trips, Far More Abandoned**
>
> From the simulator: in the rain case the base fare completes 8528 trips and the PI 8346, yet the base fare abandons 1925.0 requests against 116.6 (Table 9.5). The PI's price turns riders away before they request, and the model does not count them as abandoned.

9.6.6 What a Cap Removes

The eighth case adds 40 minutes of rain to the stadium and caps the price at 1. Every legal multiplier is then 1, so the PI, the table and the optimizer produce the same run, a peak pickup of 11.83 min at hour 1.10 with 9.31 drivers idle, 3,338.1 abandonments and 84.8 % of the time below 8 min. With the cap raised to 2.5, as the briefing suggests, the optimizer with preview starts to surge at hour 0.37, reaches the cap at 0.40 and still peaks at 9.02 min with 656.2 abandonments, the PI peaks at 10.02 with 1,123.1 and the table at 9.27 with 884.3, and all three spend 0.45 to 0.625 h in the goose chase. With no cap the optimizer climbs to 3.7 and holds 5.23 min with 88.1 abandonments, while the PI still peaks at 10.02 min, because it reacts after the crowd has drained the idle pool. A cap of 2.5 therefore puts the 8 min limit out of reach of every rule here, although no rule in this case comes near the 15 min that the design plan asked the cap case to exceed.

9.6.7 Whose Objective

The ninth case gives the stadium to the optimizer with a weight of 3 on driver earnings, for a mean multiplier of 2.470, earnings of $33.56 an hour averaged over the four hours and 4,522 trips, against 1.396, $25.72 and 5,804 trips with that weight at zero. A weight of 3 on gross fares instead drives the price to the ceiling of 6 for 0.47 h, a mean of 3.483, $192,540 of fares and 4,074 trips, and a weight of 3 on completed trips gives a mean of 1.191 and 6,194 trips for $91,772. The run that pays drivers most, $40.25 an hour under the fares weight, also grosses the most and completes the fewest trips, and the run that charges riders least completes the most, so each weight buys its own outcome.

9.6.8 A City Day

The challenge runs 24 hours from midnight with the daily pattern, rain from 17:00 to 19:00 at the peak of the rush and a stadium at 23:00, decisions every 2 minutes and a quantum of 0.1. It asks for a pickup below 8 min at least 95 % of the day, at most 420 abandonments and a mean multiplier of at most 1.3. Table 9.6 compares ten configurations through Reset, as the briefing prescribes. The same settings chosen before Run give identical trajectories, including constant prices applied at the priming decision. Every automatic run keeps the pickup below 8 min all day. The shipped PI with a 5 min target and a cap of 2 fails on abandonment with 538.3. Its cap never binds, and the run is identical with the cap off. Its target is the problem, because the PI first raises the price at 17:16, inside the rain, and 222.6 of its abandonments come before 17:00 and 178.3 during the rain, against 70.4 during the rain with a 3.5 min target, which passes.

The suggested optimizer passes with 356.6 abandonments, 63.4 inside the limit, at a mean of 1.198, but the blind optimizer with the cap in place passes as well, with 354.1 at 1.194 and the cap binding for 0.20 h, so the event calendar is not what wins the day, and the table passes too. As first built, the mean limit was 1.4, and a constant 1.4 passed with 398.3 abandonments. At 1.3 a constant 1.4 fails on the mean and a constant 1.3 on abandonment (608.8), so no constant price passes, which the application now checks, and every automatic run that passed still passes. Every passing run

Table 9.6 The 24-Hour Challenge in Ten Configurations

Setting	below 8 (%)	aband.	mean m	peak τ_p (min)	
PI, $\tau^* = 5$, cap 2 (shipped)	100.0	538.3	1.036	5.62	fail
PI, $\tau^* = 5$, no cap	100.0	538.3	1.036	5.62	fail
PI, $\tau^* = 3.5$, cap 2	100.0	384.3	1.130	4.60	pass
table, cap 2	100.0	380.2	1.094	3.84	pass
MPC, no preview, cap 2	100.0	354.1	1.194	3.77	pass
MPC, preview, no cap (suggested)	100.0	356.6	1.198	4.18	pass
MPC, preview, no cap, $\tau^* = 3.5$	100.0	348.1	1.212	3.97	pass
MANUAL, $m = 1$	91.3	2878.7	1.000	9.43	fail
MANUAL, $m = 1.3$	100.0	608.8	1.300	6.88	fail
MANUAL, $m = 1.4$	100.0	398.3	1.400	5.00	fail

abandons between 1.25 and 1.33 % of its requests, against 1.25 % in the reference at the base fare, so the limit of 420 is met by holding pickups near the reference and partly by the requests the price removes.

9.7 Technology in Practice

The loop of Section 9.5 reads the pickup time of Equation (9.2) exactly and acts through a multiplier that both sides feel at the decision. Figure 9.5 draws it with the parts of a real platform in place of the schematic's symbols.

What Measures It

No instrument on a street measures a pickup time. The platform estimates it for each request from the positions that drivers' phones report and from a routing engine that adds up the traversal times of the road segments on the best path, a naive estimate that a learned model can refine [19]. The positions are good to a few meters under open sky and degrade among tall buildings [20]. In the model the pickup time is the zone average of the nearest-neighbor law of Equation (9.2), free of noise and of rounding, and the counts that the table and the optimizer read come from the platform's books, which Section 9.3 treats as exact.

> **TECHNOLOGY** **Sensor: A Pickup Estimate From Phone Positions**
>
> The pickup time a platform reads is an estimate. A routing engine divides the road network into short segments with traversal times, finds the best path from the driver to the rider and adds the times. A learned model can then correct that naive estimate for the route actually taken and the time lost near the ends of the trip [19]. The driver's position comes from the phone's satellite receiver, typically accurate to within 4.9 m under open sky. Near buildings, bridges and trees, blocked and reflected signals degrade it and can show a car in the wrong place [20], so the estimate can move while the car does not. One platform's wait estimate was continuous but shown to riders rounded to whole minutes [13].

What Moves It

The final element is the multiplier printed in the quote and carried into the driver's pay. The quantum Q and the move cap Δ of Equation (9.7) stand for the discrete steps riders see and for the business rules that hold back sharp moves, and the floor of 1 is the base fare. The model pays every trip that ends at the multiplier in force, the proportional pay that the survey cited below takes as the common setting.

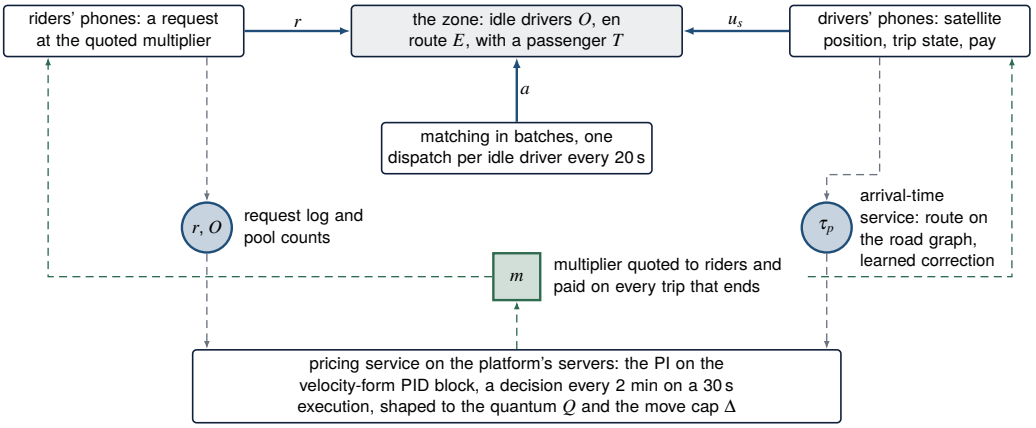


Figure 9.5 The pricing loop with the parts of a real platform in place of the schematic's symbols. The drivers' phones report positions from which an arrival-time service estimates the pickup time τ_p, the platform's books give the requests r and the idle pool O, and a pricing service sets the multiplier m that riders see in their quote and drivers in their pay. Notice that the controller never moves a car: it acts only through the price, at once on the riders and through the pay on the drivers.

> **TECHNOLOGY** **Final Element: The Multiplier in the Quote**
>
> The actuator is a number in the rider's quote. One platform's surge algorithm monitored rider demand and available drivers and placed a multiplier on the base price when demand outstripped supply at the base price, never lowering the price below the base fare. It computed a continuous value, riders saw discrete steps with 1.2 the lowest, and business rules overrode the algorithm in some cases, mainly to keep riders and drivers from very sharp price changes [13]. Driver pay is commonly taken as roughly proportional to the fare, so one price acts on demand and on supply together [21].

Where the Controller Runs

The controller runs as software on the platform's servers, not in a device in the field. A typical ride-sharing system has five modules: pricing, matching, repositioning, pooling and routing. Pricing sits upstream of the others as a macro-level lever on the balance of supply and demand, adjusting prices in real time, while matching usually batches open requests and available drivers within windows of fixed length and pairs them at predefined review times [21]. The sources read here give no period for either module. In the simulator the matching cycle is 20 s, the controller executes every 30 s, and the PI of Equation (9.8) is the framework's velocity-form PID block, which acts only at a decision of the calendar, every 2 minutes as shipped.

> **IN PRACTICE** **A Pay Floor Divided by Utilization**
>
> New York City's taxi regulator adopted a minimum pay rule for app-dispatched drivers in December 2018. Each trip must pay at least \$0.631 a mile and \$0.287 a minute, divided by the company's utilization rate: the time its drivers spend carrying passengers over the time they are available for dispatch. At the industry average of 58 percent, a 7.5-mile, 30-minute trip pays at least \$23.00. A company whose drivers wait longer for a dispatch pays more per trip, and the regulator assesses each company's utilization on a regular basis and publishes the adjusted rates [8].

What Is Modernizing It

Three developments touch this loop. The first is the learned arrival-time estimate: a residual network that corrects the routing engine's estimate lowered the mean and median absolute errors in offline

experiments and online tests, and it was built for low-latency, high-throughput serving [19]. The second is pricing optimized jointly with matching or with the repositioning of idle cars, for which reinforcement learning is a promising direction because the interaction of price with supply and demand is hard to model explicitly. The third is the matching window itself, now chosen by learning methods that trade the wait before a match against the pickup after it [21]. The optimizer of Section 9.5 is the model-based counterpart of the second: it plans the price on the simulator's own flow function and leaves matching fixed.

9.8 Limitations

The zone is one aggregate market of fluid pools without streets, strategy, noise or a rider queue, and every coefficient is a teaching value. Drivers are paid the multiplier in force on every trip that ends, so the earnings respond to a price at once and the fleet's lag starts at the decision. The abandonment count leaves out riders whom the price deters, which flatters high prices, and the optimizer reads the platform's pools without noise, plans with the declared coefficients and searches locally.

> **OPEN CHALLENGE** **Elasticities No Simulator Knows**
>
> A pricing loop is only as good as its picture of how riders and drivers answer a price. A survey of learning methods for ride-sharing found that no public simulator supported pricing decisions and that the simulators of pricing papers make strong assumptions on passenger and driver price elasticities. The supply response shapes every sequence of price decisions, yet it was not yet widely considered in learning approaches to pricing, and no public simulator was faithful enough for a policy trained in it to go straight to production [21]. This chapter's supply elasticity and lag are likewise not taken from a published estimate. Progress would be rider and driver behavior learned from data.

The challenge does not reward foresight over a blind optimizer, and its cap never binds for the shipped PI. Choosing a setting before Run includes it in the priming decision, so startup comparisons agree with Reset.

> **OPEN CHALLENGE** **A Price or a Queue at the Peak**
>
> A rising price is not the only answer to a peak. A platform can also queue requests when no vacant car is near, and in some markets the queue is perceived as more socially acceptable than surge pricing. A queue raises its own open questions: how large an area each queue should cover, whether that area should change over time, and when incoming requests should start to queue [21]. A model that answers them would let a price and a queue be compared on the same zone, the comparison this chapter's scorecard makes among pricing rules only.

9.9 Conclusion

This chapter's simulator writes surge pricing as a feedback loop on a market whose throughput map folds. A multiplier of 2 removes 29.3 % of the requests at the decision and adds 12.3 % to the fleet, the sixth root of the price, and the fleet's response starts at once and overshoots to 913.2 drivers. At a fixed fleet of 800 the throughput peaks at 2,128 trips an hour with 65.7 idle drivers, and past that point three times the reference demand completes fewer trips than twice it, while with the fleet free a multiplier of 1.2 raises the trips at three times from 2,563 to 3,200. A stadium at the base fare puts 41 % of the fleet on empty pickups and the pickup at 10.2 min, a PI deciding every 2 minutes holds it to 5.36 min and one deciding every 10 minutes does not, and preview helps only when the price is slow, cutting the abandonments from 98.7 to 75.0 at 0.1 per decision and from 75.3 only to 75.2 at 1. In the capped case a cap of 2.5 keeps every rule above 8 min.

The surge-off latch, the optimizer's move limit and the challenge's mean limit were corrected in the application during this work, and the rain case moved onto the rush. A fuller model would add zones with relocation, a rider queue, earnings paid at the accepted price and a challenge that

rewards foresight. The simulator makes the price that keeps a ride-hailing zone working a measurable property of its fold.

KEY POINTS

- A price reaches riders at once and drivers only through their pay, and at the defaults the fleet follows only the sixth root of the multiplier.
- Past the fold a zone completes fewer trips as demand grows, and there a higher price buys trips instead of costing them.
- A higher price shortens the pickup, so the process gain is negative and the direct-acting PI raises the price as the pickup rises.
- A PI deciding every 2 minutes holds the stadium to 5.36 min, and one deciding every 10 minutes lets the zone into the goose chase.

Where else this applies. Chapter 8 holds a freeway merge short of the same kind of fold with a ramp signal.

References

[1] G. Gallego and G. van Ryzin, "Optimal dynamic pricing of inventories with stochastic demand over finite horizons," *Management Science*, vol. 40, no. 8, pp. 999–1020, 1994.

[2] S. Banerjee, R. Johari, and C. Riquelme, "Pricing in ride-sharing platforms: A queueing-theoretic approach," in *Proceedings of the Sixteenth ACM Conference on Economics and Computation*. New York, NY, USA: ACM, 2015, p. 639.

[3] G. P. Cachon, K. M. Daniels, and R. Lobel, "The role of surge pricing on a service platform with self-scheduling capacity," *Manufacturing & Service Operations Management*, vol. 19, no. 3, pp. 368–384, 2017.

[4] K. Bimpikis, O. Candogan, and D. Saban, "Spatial pricing in ride-sharing networks," *Operations Research*, vol. 67, no. 3, pp. 744–769, 2019.

[5] H. Wang and H. Yang, "Ridesourcing systems: A framework and review," *Transportation Research Part B: Methodological*, vol. 129, pp. 122–155, 2019.

[6] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, "Robotic load balancing for mobility-on-demand systems," *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.

[7] R. Zhang, F. Rossi, and M. Pavone, "Model predictive control of autonomous mobility-on-demand systems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1382–1389.

[8] New York City Taxi and Limousine Commission, "Notice of promulgation: Rules amending provisions regarding driver income and vehicle lease transparency," Adopted December 4, 2018, 2018, accessed 2026-11-20. [Online]. Available: https://www.nyc.gov/assets/tlc/downloads/pdf/driver_income_rules_12_04_2018.pdf

[9] R. Arnott, "Taxi travel should be subsidized," *Journal of Urban Economics*, vol. 40, no. 3, pp. 316–333, 1996.

[10] J. C. Castillo, D. Knoepfle, and G. Weyl, "Surge pricing solves the wild goose chase," in *Proceedings of the 2017 ACM Conference on Economics and Computation*. New York, NY, USA: ACM, 2017, pp. 241–242.

[11] M. J. Lighthill and G. B. Whitham, "On kinematic waves II. A theory of traffic flow on long crowded roads," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 229, no. 1178, pp. 317–345, 1955.

[12] J. Cramer and A. B. Krueger, "Disruptive change in the taxi business: The case of Uber," *American Economic Review*, vol. 106, no. 5, pp. 177–182, 2016.

[13] P. Cohen, R. Hahn, J. Hall, S. Levitt, and R. Metcalfe, "Using big data to estimate consumer surplus: The case of Uber," National Bureau of Economic Research, Cambridge, MA, NBER Working Paper 22627, 2016.

[14] B. Schaller, "Entry controls in taxi regulation: Implications of US and Canadian experience for taxi regulation and deregulation," *Transport Policy*, vol. 14, no. 6, pp. 490–506, 2007.

[15] R. C. Larson and A. R. Odoni, *Urban Operations Research*. Englewood Cliffs, NJ: Prentice-Hall, 1981.

[16] M. K. Chen, "Dynamic pricing in a labor market: Surge pricing and flexible work on the Uber platform," in *Proceedings of the 2016 ACM Conference on Economics and Computation*. New York, NY, USA: ACM, 2016, p. 455.

[17] K. J. Åström and T. Hägglund, *Advanced PID Control*. Research Triangle Park, NC: ISA – The Instrumentation, Systems, and Automation Society, 2006.

[18] J. D. Hedengren, R. Asgharzadeh Shishavan, K. M. Powell, and T. F. Edgar, "Nonlinear modeling, estimation and predictive control in APMonitor," *Computers & Chemical Engineering*, vol. 70, pp. 133–148, 2014.

[19] X. Hu, T. Binaykiya, E. Frank, and O. Cirit, "DeeprETA: An ETA post-processing system at scale," arXiv:2206.02127, 2022.

[20] GPS.gov, National Coordination Office for Space-Based Positioning, Navigation, and Timing, "GPS accuracy," 2026, accessed 2026-11-20. [Online]. Available: https://www.gps.gov/gps-accuracy

[21] Z. Qin, H. Zhu, and J. Ye, "Reinforcement learning for ridesharing: An extended survey," arXiv:2105.01099; published in Transportation Research Part C 144 (2022) 103852, 2021.

9.10 Exercises

9.1 Concept check · 2 min

The pickup-time loop of Equation (9.8) forms its error as $e_k = \tau_{p,k} - \tau^*$. In the terms of Section 1.4.3, which statement is right?

(a) Reverse acting with $a = +1$, because a long pickup makes the controller push the price up

(b) Direct acting with $a = -1$, because the process gain is negative and the price must rise with the PV

(c) Reverse acting with $a = +1$, because near the reference a higher price lengthens the pickup time

(d) Direct acting with $a = -1$, because the completed trips fall whenever the multiplier rises

9.2 Concept check · 2 min

At three times the reference demand, raising the multiplier from 1 to 1.2 raises the steady completed trips from 2563 to 3200 an hour. Which explanation is right?

(a) The price attracts about a fifth more drivers, enough to carry the extra trips

(b) The price shortens each passenger trip, so every driver completes more trips an hour

(c) Fewer requests leave a larger idle pool, so pickups shorten and fewer drivers drive empty

(d) The price lowers the abandonment midpoint, so riders accept longer predicted pickups

9.3 Concept check · 2 min

With a largest price move of 0.1 per decision, event preview cuts the abandonments of the stadium case from 98.7 to 75.0. With the shipped move of 1 per decision it cuts them only from 75.3 to 75.2. What do the four runs show?

(a) Preview is worth little here, because the optimizer's horizon already sees the stadium coming

(b) The blind optimizer with the fast move is better tuned, so its weights leave preview nothing to add

(c) A large move cap makes the price chatter, and the chatter cancels what preview would gain

(d) A forecast substitutes for actuator speed, and a price that can move fast needs little of it

9.4 Theory and calculation · 5 min

Use Equations (9.2) and (9.3) with their defaults. The minimum pickup is $\tau_{min} = 2\,\text{min}$, the geometry coefficient is $a_p = 0.5\,\text{h drivers}^{1/2}$, which is $30\,\text{min drivers}^{1/2}$, the abandonment midpoint is $\tau_{ab} = 10\,\text{min}$ and its scale is 1.5 min.

(a) Find the idle pool at which the pickup reaches the 8 min limit of the challenge. *Checkpoint:* 25 drivers.

(b) Find the share of requests that riders abandon at that pickup. *Checkpoint:* 20.9 %.

(c) During the stadium release 4800 requests an hour reach the zone at the base fare. With 25 drivers idle, say which bound of Equation (9.3) limits the dispatches, and find the abandonments an hour. *Checkpoint:* The riders' acceptance binds, about 1001 an hour.

9.5 Theory and calculation · 7 min

At steady state with little abandonment, the completed trips equal the requests, $c \approx r = \lambda_0 m^{-\eta_d}$, and the online fleet equals its target N^* of Equation (9.4).

(a) Show that N is proportional to $m^{(1-\eta_d)\eta_s/(1+\eta_s)}$, and evaluate the exponent at the defaults $\eta_d = \eta_s = 0.5$. *Checkpoint:* 1/6.

(b) The reference holds 796.66 drivers and 1200 requests an hour at $m = 1$. Predict the requests and the settled fleet at $m = 2$, and compare with the 848.5 and 894.81 that the chapter reports. *Checkpoint:* 848.5 an hour and 894.2 drivers.

(c) Find the exponent and the factor at $m = 2$ for a supply elasticity of 1, and the exponent for a demand elasticity of 1 with $\eta_s = 0.5$. *Checkpoint:* 1/4 and 1.189, then 0.

9.6 Theory and calculation · 7 min

The fixed-fleet throughput map of Equation (9.6) with the pickup law of Equation (9.2) has its fold where $\partial q/\partial O = 0$. Work in minutes, with $\tau_t = 15\,\text{min}$, $\tau_{min} = 2\,\text{min}$ and $a_p = 30\,\text{min drivers}^{1/2}$.

(a) Show that for $O > 0.25$ the fold satisfies $\tau_t + \tau_{min} + a_p O^{-1/2} = (N - O)\,\frac{1}{2} a_p O^{-3/2}$. *Checkpoint:* Differentiate the quotient.

(b) For $N = 800$, show that $O = 65.66$ satisfies the condition, and find the largest trip rate and the pickup there. *Checkpoint:* Both sides 20.70 min, 2128 trips an hour, 5.70 min.

(c) The reference completes 1185 trips an hour with 432 idle drivers and a fleet near 800. Find the share of the largest trip rate it uses, and say on which side of the fold it sits. *Checkpoint:* 55.7 %, the idle-rich side.

9.7 App challenge · 8 min

Load scenario 11, Reference · constant demand, which starts in MANUAL at the base fare. Before Run, set Multiplier request to 2.0. Run the full 3 h at the default seed and export the CSV file.

(a) In the first row of the file, read `requested`, `earnings` and `target`. *Checkpoint:* 848.5 an hour, \$35.70 an hour and 1126.6 drivers, within 0.1.

(b) Find the largest value of the online fleet `N` and the time at which it occurs. *Checkpoint:* 913.2 drivers at 0.55 h, within 0.2 drivers and 0.02 h.

(c) Read `N` in the last row, divide it by 796.66 and compare the ratio with the sixth root of the multiplier. *Checkpoint:* 894.8 drivers, a ratio of 1.123 against 1.122.

9.8 App challenge · 8 min

Load scenario 2, No surge and the capacity fold, and switch the header to AUTO before Run. The controller is the PI on pickup time with the shipped tuning. Set Decision calendar to Every 30 seconds and run the full 4 h at the default seed. Then set Every 2 minutes, press Reset and run again, and do the same with Every 10 minutes.

(a) For each run, read the Abandoned tile and find the largest `pickup` in the exported file. *Checkpoint:* 5.10 min and 97, 5.36 min and 99, 10.19 min and 233, within 0.02 min and 1.

(b) For the run on the 10-minute calendar, count the rows in which the idle pool `O` is below the fold's idle pool `foldO`, and convert the count to hours. *Checkpoint:* 16 rows, about 0.27 h.

Explain why the slowest calendar fails when the other two succeed.

9.9 App challenge · 9 min

Load scenario 10, Challenge · a complete city day, and run the full 24 h as shipped at the default seed.

(a) Read the tiles Under 8 min, Abandoned, Mean multiplier and Challenge. *Checkpoint:* 100.0 %, 538, 1.04 and not met.

(b) In the exported file, read `totalAbandoned` at 17:00 and at 19:00, and split the abandonments into those before the rain and those during it. *Checkpoint:* 222.6 before and 178.3 during, within 0.5.

(c) Set Pickup target to 3.5 min, press Reset and run the day again. Repeat (a), and find the abandonments during the rain. *Checkpoint:* 100.0 %, 384, 1.13, met. 70.4 during the rain.

9.10 Application area · 3 min

At the reference demand a multiplier of 2 held to steady state cuts the requests from 1200 to 848.5 an hour and the pickup from 3.44 to 3.19 min.

(a) Use Equation (9.3) to find the abandonments an hour before and after the move. *Checkpoint:* 15.0 and 9.0 an hour.

(b) Find the riders an hour whom the price turns away and the Abandoned tile never counts, and say what this means for ranking pricing rules by that tile. *Checkpoint:* 351.5 an hour.

9.11 Application area · 4 min

Scenario 8 caps the price at 1 during a stadium release with rain. Under the optimizer with preview, Table 9.5 gives a peak pickup of 11.83 min with 3338.1 abandonments at the cap of 1, and 9.02 min with 656.2 at a cap of 2.5.

(a) Explain why the PI, the table and the optimizer produce the same run at the cap of 1. *Checkpoint:* Every legal price is 1.

(b) A regulator weighs a cap of 2.5 against a cap of 1 for such events. Find what the higher cap saves in abandonments, and say why the 8 min limit stays out of reach. *Checkpoint:* 2682 abandonments, about 80 %.

9.12 Review the agent · 5 min

An AI assistant is asked to make the pickup-time PI safer and replies as follows. “The steady-state gain from multiplier to pickup is −0.49 min per unit at the reference demand and −10.0 min at three times it, twenty times larger. With $K_c = 0.45$ fixed, the loop gain grows from 0.22 to 4.5, far beyond any gain margin. Schedule the gain on measured demand as $K_c = 0.45 \times 0.49/|K_p|$, which gives 0.022 at three times the demand and holds the loop gain at 0.22, so the PI stays well damped through a stadium release.” Find the flaw in the proposal and say how it could be checked.