Feedback-based Optimization of Tiered Pricing Systems*

Kabir Vats

Advisor: Dr. Jorge Poveda

Jacobs School of Engineering, University of California, San Diego

March 29, 2025

Abstract

This paper presents a framework for feedback-based optimization of tiered pricing systems, targeting monopolistic sellers offering multiple service levels to a heterogeneous customer base. To this end, a parametric model is utilized that captures customer utility using a diminishing returns function of service cost and a continuous valuation distribution. The pricing optimization problem is formulated as a smooth, non-convex objective, and solved using gradient-based methods.

To handle real-world constraints where internal customer preferences are unobservable, a closed-loop controller-estimator system is utilized. The controller uses stochastic gradient descent (Adam) to iteratively improve pricing based on estimated gradients of expected profit, while the estimator applies a particle-based Sequential Monte Carlo (SMC) method to infer hidden population parameters from observed customer choices.

Empirical results show that despite the non-convexity of the objective, gradient-based optimization performs robustly due to the low dimensionality of the price space and the smoothing effect of continuous valuation distributions. Furthermore, the feedback controller converges reliably to near-optimal prices using only sparse, noisy observations.

1 Introduction

When selling a singular product, a seller must calculate the price that maximizes their profits. This is the classic economics problem where a monopolist will choose a price that maximizes marginal utility. [1] However, when a seller is marketing multiple products that compete with each other (such as in the example of most subscription services, or a coffee shop that sells multiple sizes) the seller needs to price their products where low-end products are purchased by more frugal customers, and high-end products are still valued such that customers with more purchasing power will choose to buy them. [2]

*Code and figures available at: `https://github.com/kabir-vats/Tiered-Dynamic-Pricing`

My goal is to consider a monopolistic seller offering multiple levels of a service who seeks to maximize their profit, selling to a heterogeneous customer base, with the ability to vary their prices over time.

2 Literature Review

In existing literature for tiered dynamic pricing, there are notable application-specific explorations in fields of retail and ISP pricing. Additionally, there are notable and relevant examples of optimizing an efficiency or profit objective in tolling, recommender systems, and other fields. [3] [4] [5]

The main benefit of tiered pricing is the ability to market different versions of a product to different customers. There exist many ways to model heterogenous customers. In work optimizing ISP tiered pricing, Lee, Jeong and Seo use a truncated normal distribution to model their heterogeneous population base [6]. Work by Ren et. al in the same area uses a uniform valuation parameter distribution [7]. In Schoengold and Zilerman's work, they use a generalized probability distribution function [8]. In this case, I chose to model customer heterogeneity with uniform and gaussian distributions, but this paper will only cover the guassian distribution.

Where my work fundamentally differs is the restriction to using modeled customer interactions to gain knowledge of the internal system parameters during optimization. Additionally, rather than just searching for the optimal prices, the goal of this system is to find the optimal prices using a small amount of customer interactions, allowing the system to adapt quickly to customer behavioral changes.

To use the limited information to optimize, a projected gradient-descent approach similar to that in Chandrasekaran et al's work optimizing recommender systems.[4] Instead of using a Kalman Filter and an ANN to estimate the static population parameters, I opted for Sequential Monte Carlo approximation.

3 Formulation

From the viewpoints of both the seller and the consumer, a formulation must optimize a larger objective function from the producer, using the preferences and decisions from a consumer with a smaller-scale utility maximization function.

With heterogenous consumers, a common method of modeling this is through a valuation parameter defined by a certain probability distribution. [6] [7] [8]

To formulate this optimization problem, a nonlinear optimization problem is defined with the following notation:

Let:

- n: number of product tiers
- c_i: producer cost for tier i, with $c_0 = 0$
- p_i: price set for tier i, with $p_0 = 0$

- v_j: valuation parameter for customer j
- λ: diminishing returns coefficient
- u_{ij}: utility of customer j for tier i
- b_i: benefit of tier i
- $d_{ij} \in \{0, 1\}$: indicator if customer j chooses tier i, where $\sum_{i=0}^{n} d_{ij} = 1$
- $q_i = \sum_j d_{ij}$: quantity sold at tier i

Define:

$$b_i = c_1 \left(\frac{c_i}{c_1} \right)^{\lambda}$$

$$u_{ij} = v_j b_i - p_i$$

$$d_{ij} = \begin{cases} 1 & \text{if } i = \arg\max_{k=0,\ldots,n} u_{kj} \\ 0 & \text{otherwise} \end{cases}$$

The monopolist's profit is:

$$F(p) = \sum_{i=1}^{n} (p_i - c_i) \sum_j d_{ij}$$

Optimization Problem

$$\max_{p_1,\ldots,p_n} \quad F(p) = \sum_{i=1}^{n} (p_i - c_i) \sum_j d_{ij}$$

subject to:

$$d_{ij} = 1 \quad \text{only if} \quad u_{ij} \geq u_{kj} \quad \forall k \in \{0, \ldots, n\}, \forall j$$

$$p_i \geq 0 \quad \forall i = 1, \ldots, n$$

We care about reaching the global optimum, and an analytical solution seems implausible. So, gradient descent seems like an idea approach, provided we can find a way to avoid local optima.

3.1 Infinite-Horizon Pricing Optimization via Expected Profit

Gradient calculation in the finite-form expression is difficult.

To extend the model to an infinite stream of customers, we assume that each customer arrives with a valuation parameter $v \in R_{\geq 0}$, drawn from a known probability distribution D. Let $f(v)$ denote the probability density function (PDF) of D, satisfying:

$$\int_0^{\infty} f(v)\, dv = 1$$

Two common choices for D are:

- Uniform distribution: $f(v) = \frac{1}{v_{\max}}$ for $v \in [0, v_{\max}]$
- Truncated normal distribution: $f(v) \propto \exp\left(-\frac{(v-\mu)^2}{2\sigma^2}\right)$ for $v \in [v_{\min}, v_{\max}]$

Each tier $i \in \{1, \ldots, n\}$ is associated with a production cost c_i and price p_i. The utility that a customer with valuation v assigns to tier i is given by:

$$u_i(v) = vb_i - p_i, \quad \text{where } b_i = c_1 \left(\frac{c_i}{c_1}\right)^{\lambda}$$

Customers select the tier i that maximizes their utility, i.e., $i = \arg\max_k u_k(v)$. However, not all tiers are necessarily chosen by any segment of the market. Some tiers may be **dominated**, meaning that there exists another tier with both lower price and higher utility for all v. These tiers should be **excluded from consideration**

Computing Optimal Intervals

To identify valid customer segments and compute valuation intervals for each non-dominated tier, we use the interval construction procedure defined in **Algorithm 1**.

At a high level, the algorithm:

- Sorts tiers by increasing benefit-to-price ratio
- Iteratively computes pairwise indifference points (utility intersections) between tiers
- Prunes dominated tiers by checking and adjusting threshold crossings to maintain monotonic intervals
- Returns a list of valuation intervals $\mathcal{I}_i = [v_i^{\min}, v_i^{\max})$, one for each active tier

The algorithm ensures that each valuation v is assigned to at most one tier, and that only non-dominated tiers are assigned non-empty intervals.

Expected Profit Formulation

Let $\mathcal{I}_i = [v_i^{\min}, v_i^{\max})$ denote the valuation interval over which tier i is chosen, as returned by Algorithm 1. Then the expected profit is:

$$F(p) = \sum_{i=1}^{n} (p_i - c_i) \int_{v_i^{\min}}^{v_i^{\max}} f(v)\, dv$$

Optimization Problem

The pricing optimization problem becomes:

$$\max_{p_1, \ldots, p_n} \quad F(p) = \sum_{i=1}^{n} (p_i - c_i) \int_{v_i^{\min}(p)}^{v_i^{\max}(p)} f(v)\, dv$$

Algorithm 1 Calculate Optimal Intervals for Pricing Tiers

Require: b (benefit vector), p (vector of prices)
Ensure: Ordered list of intervals where each tier is optimal

Sort indices based on b: sorted_indices $\leftarrow$ argsort(b)
Reorder b and p using sorted_indices
Initialize thresholds: thresholds $\leftarrow [-\infty]$
for $i = 0$ to tiers $- 1$ **do**
    if $i = 0$ **then**
        thresholds.append $\left(\frac{p_0}{b_0}\right)$
    else
        intersection $\leftarrow \frac{p_i - p_{i-1}}{b_i - b_{i-1}}$
        $j \leftarrow 0$
        while intersection $<$ thresholds$[i - j]$ **do**
            $j \leftarrow j + 1$
            if $i = j$ **then**
                intersection $\leftarrow \frac{p_i}{b_i}$
                break
            else
                intersection $\leftarrow \frac{p_i - p_{i-j-1}}{b_i - b_{i-j-1}}$
            end if
        end while
        thresholds.append(intersection)
    end if
end for
for $i = 0$ to tiers $- 1$ **do**
    if thresholds[tiers $- i$] $<$ thresholds[tiers $- i - 1$] **then**
        thresholds[tiers $- i - 1$] $\leftarrow$ thresholds[tiers $- i$]
    end if
end for
intervals $\leftarrow [(\text{thresholds}[i], \text{thresholds}[i + 1])$ for i in $[0, \text{tiers} - 1]]$
intervals.append((thresholds$[-1], \infty$))
intervals_ordered $\leftarrow$ [None] $\times$ |intervals|
intervals_ordered[0] $\leftarrow$ intervals[0]
for i, idx in **enumerate**(sorted_indices) **do**
    intervals_ordered[idx $+ 1$] $\leftarrow$ intervals$[i + 1]$
end for
return intervals_ordered

subject to:

$$p_i \geq 0 \quad \forall i \in \{1, \ldots, n\}$$

Remarks

- The intervals $[v_i^{\min}, v_i^{\max})$ are not fixed—they depend on prices p_i and are computed dynamically using Algorithm 1.
- This formulation avoids summing over discrete customers by taking the expectation over a continuous distribution of valuations.
- The resulting optimization is smooth and tractable when $f(v)$ is chosen to allow closed-form or numerically stable integration.

With these thresholds, expected profit can now be expressed like this:
$E[F] = \sum_{i=0}^{N}(p_i - c_i)Pr\{t_i < v < t_{i+1}\}$

4 Static Optimization

Gradient-Based Optimization in a Non-Convex Landscape

The profit function $F(p)$ is not globally convex, as can be seen in **Figure 1**, but experimental results suggest that gradient descent converges reliably to global optima in practice.

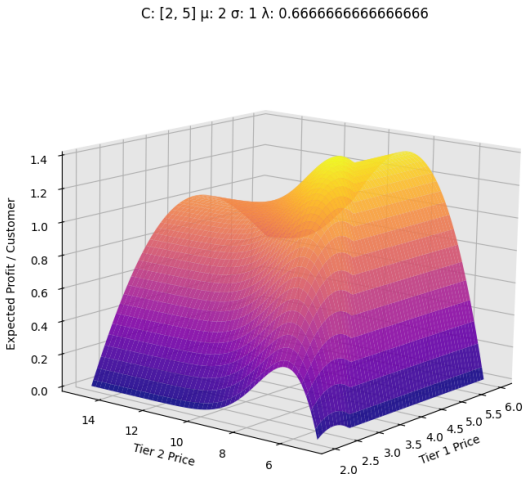

Figure 1: This plot shows expected profit as a function of prices for a system with two pricing tiers. The function is not convex, and has local maxima extending at straight lines that are *nearly* colinear with the global maximum (but are not! If they were, this would be trivial to optimize). The local maxima can be interpreted as areas where a certain tier's price is so high that no customers are purchasing it, making the profit unaware of any increases or decreases in the profit.

Due to the use-cases of tiered pricing optimization, the dimensionality of the price search space is relatively small (between 2-10 tiers), meaning there are few local optimal prices to trap the descent. The use of continuous valuation distributions $f(v)$ smooths the objective, making it nearly differentiable almost everywhere. Additionally, because thresholds are functions of prices and $f(v)$ is smooth, the overall objective responds gradually to price updates, avoiding erratic gradient behavior. The trade-off between price and demand often results in unimodal structures for each tier's contribution to profit, encouraging global convergence.

This structure enables gradient-based optimization to perform well in practice, even without theoretical convexity guarantees.

The gradient function uses similar logic to the interval calculating function, and the procedure to calculate gradients is described in **Algorithm 2**.

Next, it was crucial to run gradient descent on this formulation to prove that a gradient descent-like online feedback optimizer could be used as the price controller. Initially, I tried a naive gradient descent that can be expressed as $p_t = p_{t-1} + \eta \nabla J(p_t)$ where η is a constant learning rate.

However, the shape of the objective functions made learning rate rather unreliable without higher order optimization methods. As a result, I used the most commonly used gradient descent momentum estimator: Adam (Adaptive Moment Estimation) [9]. With the adam optimizer, I was able to get pretty reliable convergence to the global maximum.

The gradient-based optimization then became:

$$v_t = \beta_1 * v_{t-1} - (1 - \beta_1) * \nabla J(p_t), s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * \nabla J^2(p_t)$$

$$p_t = p_{t-1} + \eta \frac{v_t}{\sqrt{s_t + \epsilon}} * \nabla J(p_t)$$

During this paper, v_0, s_0 were 0, and β_1 was 0.9, and $beta_2$ was 0.999.

To guarantee an initial condition on a sloped area, initial prices are always set equal to costs.

In **figure 2**, gradient descent is illustrated for multiple different learning rates in the two tier system. **Figure 3** shows the gradient descent in three tiers, but due to the constraints of living in a 3-d world it is not shown with a nice surface plot.

5 Feedback Optimization

After implementing gradient descent, we now optimize in a real-world setting. The controller in this situation is now completely blind to customer valuation parameters, meaning it needs to use *only* customer purchase decisions to update its beliefs about the population and optimize. The customers are randomly sampled from the true parameter distribution, so their choices give the estimator information on the true parameters.

This controller now only starts with knowledge of tier costs c_i and must estimate μ, σ, and λ to calculate gradients at each step.

Algorithm 2 Gradient Computation for Profit Function

Require: b (benefit vector), c (costs vector), p (prices vector)
Ensure: Gradient vector $\nabla J(p)$

Sort indices based on b: sorted_indices $\leftarrow$ argsort(b)
Reorder b, p, and c using sorted_indices
Initialize thresholds: thresholds $\leftarrow [-\infty]$
Initialize gradient storage: t_grads $\leftarrow []$
for $i = 0$ to tiers $- 1$ **do**
    if $i = 0$ **then**
        thresholds.append $\left(\frac{p_0}{b_0}\right)$
        t_grads.append($\{1 : 1/b_0\}$)
    else
        Compute intersection: intersection $\leftarrow \frac{p_i - p_{i-1}}{b_i - b_{i-1}}$
        t_grad $\leftarrow \{i : -1/(b_i - b_{i-1}), i+1 : 1/(b_i - b_{i-1})\}$
        $j \leftarrow 0$
        while intersection $<$ thresholds$[i - j]$ **do**
            $j \leftarrow j + 1$
            if $i = j$ **then**
                intersection $\leftarrow \frac{p_i}{b_i}$
                t_grad $\leftarrow \{i + 1 : 1/b_i\}$
                break
            else
                Update intersection: intersection $\leftarrow \frac{p_i - p_{i-j-1}}{b_i - b_{i-j-1}}$
                Update gradient: t_grad $\leftarrow \{i - j : -1/(b_i - b_{i-j-1}), i+1 : 1/(b_i - b_{i-j-1})\}$
            end if
        end while
        thresholds.append(intersection)
        t_grads.append(t_grad)
    end if
end for
For each tier, if thresholds$[i]$ $<$ thresholds$[i - 1]$, update the thresholds and gradients.
Compute probability-based gradients for Gaussian distribution
for $i = 0$ to tiers **do**
    Compute gradient for each tier based on Gaussian PDF
    prb_grads $\leftarrow$ Gaussian PDF based gradients
    Compute gradient: grad$[i]$ $\leftarrow \sum ((p - c) \times$ prb_grads$) +$ Gaussian normalization factor
end for
Reorder gradient $\nabla J(p)$ to match the original price order
return $\nabla J(p)$

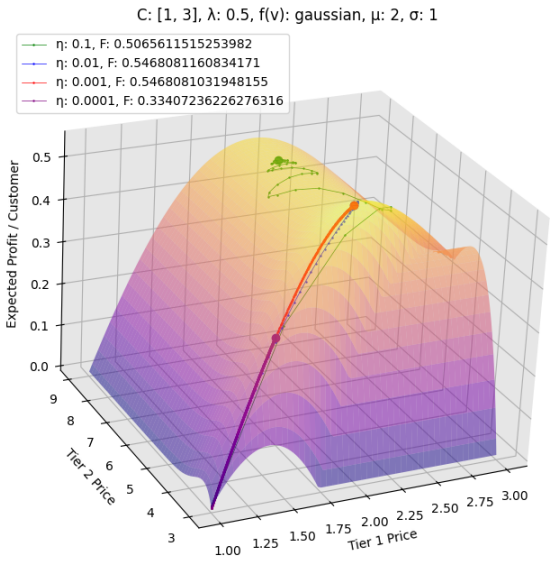

Figure 2: This plot shows many gradient descents (fully system-aware) with different learning rates, plotted against the price-profit function. The line for learning rate 0.1 becomes trapped in a local maximum where tier 2's price doesn't affect profit, and learning rates of 0.01 and 0.001 both reach the optimal profit in a timely manner.

5.1 Stochastic Gradient Descent

The system implemented works with a stochastic gradient descent controller which samples one customer's purchase each iteration. The purchase is used to update beliefs about the population, and the surrogate system with those beliefs is then used to generate a gradient estimate.

Following these gradient estimates leads the system to the optimal prices through gradient descent, while simultaneously gaining more and more accurate gradients.

At the start of a simulation, a descent is likely using extremely faulty gradients, but because prices are initialized to cost, the gradient is still in the direction of the global optimum (in a positive direction) regardless of the current parameter estimates. Then, as the system gains more information from more purchases, gradients become more accurate.

The high-level procedure for the stochastic gradient descent is described in **algorithm 3**.

This SGD uses a learning rate hyperparameter, but it is difficult to choose an optimal learning rate, because as shown in **Figure 11**, the optimal learning rate is influenced by choices for $\mu, \sigma and \lambda$ which are hidden parameters. Future improvements could involve finding the optimal learning rate dynamically.

5.2 Sequential Monte Carlo Parameter Estimation

A gaussian distribution of valuation parameters can be represented with the mean (μ) and standard deviation (σ).

To estimate these parameters, reasonable prior bounds are used to generate a large amount

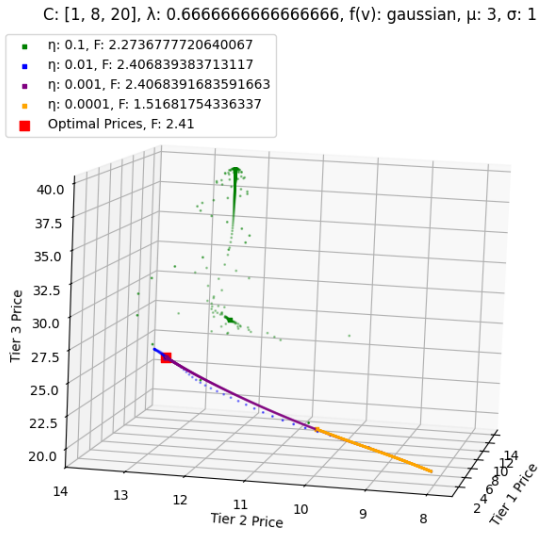

Figure 3: This plot shows many gradient descents (fully system-aware) with different learning rates in 3-price space. Learning rate 0.1 is once again trapped in a local maximum, and learning rates 0.01 and 0.001 both reach optimal profit (but a closer look shows that 0.01's descent initially overshot optimal profit before correcting).

of parameter estimates via the Sobol sampling method. [10] Each customer purchase step, these estimates are then reweighed using a Sequential Monte Carlo that uses resampling and jittering, and is detailed in "Sequential Monte Carlo Methods in Practice" by Doucet et al. [11]. The adaptation of this algorithm is written in **Algorithm 4**.

To increase exploration and avoid only taking samples along the descent path, prices can be jittered from the current price by a standard deviation of 10%, which increases the reliability of reaching the maximum as shown in **figure 15**.

6 Conclusion

This paper demonstrates the design and implementation of an algorithm that can optimize prices across multiple tiers to maximize the profit of a monopolist selling to a heterogeneous population. By using stochastic gradient descent on a surrogate system built through population parameter approximations, I was able to achieve high accuracy and reliability in reaching optimal profits within few (less than 500) sales.

Without Optimizing tiered pricing in the presence of customer heterogeneity blends behavioral economics with algorithmic strategy, and requires decisions under uncertainty to be made in real time. Through this project, I developed and implemented a feedback-based optimization system that is capable of learning both customer preferences and optimal pricing simultaneously and without direct access to customer utility or valuation parameters.

The core insight from this work is that even in the absence of explicit customer data, structured purchase behavior contains enough information to both estimate underlying distributional parame-

Algorithm 3 Stochastic Gradient Descent via Sequential Monte Carlo Surrogate Simulation

Require: c (costs vector),
Ensure: Price vector p

$p \leftarrow c$
for $t = 0$ to max iters **do**
    selling_price$_i \leftarrow \mathcal{N}(p_i, (0.1p_i)^2)$
    Sell 1 item at selling_price and record the customer's choice
    Update parameter weights using the choice
    Calculate $J(p_t)$ using approximated system
    $v_t \leftarrow \beta_1 * v_{t-1} - (1 - \beta_1) * \nabla J(p_t)$
    $s_t \leftarrow \beta_2 * s_{t-1} - (1 - \beta_2) * \nabla J^2(p_t)$
    $p_t \leftarrow p_{t-1} + \eta \frac{v_t}{\sqrt{s_t + \epsilon}} * \nabla J(p_t)$
end for
return p

Algorithm 4 Parameter Update Procedure

Require: c (costs vector), selling_price (selling price vector), customer_choice (tier # customer chose), estimates (list of μ_i, σ_i, λ_i), weights (list of probabilities corresponding to each estimate)
Ensure: $\mu_{est}, \sigma_{est}, \lambda_{est}$, estimates, weights

for $j = 0$ to length(estimates) **do**
    $\text{b}_i \leftarrow c_1(\frac{c_i}{c_1})^{\lambda_j}$ for all nonzero tiers i
    Calculate intervals I using b and p (**Algorithm 1**)
    $\text{prob}_i = \text{CDF}_{[\text{Start of } I_i[0], \text{End of } I_i]} = \Phi(\text{End of } I_i; \mu, \sigma) - \Phi(\text{Start of } I_i; \mu, \sigma)$ for all tiers i
    $\text{weights}_i \leftarrow \text{weights}_i * \sqrt{\text{probs}_{\text{customer\_choice}}}$
end for
$\text{weights}_i \leftarrow \frac{\text{weights}_i}{\sum_j weights_j}$ for all i
if 50% of the weights sum to less than 10% **then** (this happens maybe 10 across 500 iters)
    Resample $\text{estimates}_i \leftarrow$ Resampled estimate according to weights
    Add jitter: $\varepsilon_i \sim \mathcal{N}(0, \Sigma), \quad \Sigma = \text{diag}(0.05^2, 0.05^2, 0.01^2)$
    $estimate_i \leftarrow estimate_i + \varepsilon_i \quad$ for all i
    $\text{weights\_i} \leftarrow \frac{1}{N} \quad$ for all i
end if
$\mu_{est}, \sigma_{est}, \lambda_{est} \leftarrow \sum_i \text{estimates}_i \text{weights}_i$
return $\mu_{est}, \sigma_{est}, \lambda_{est}$, estimates, weights

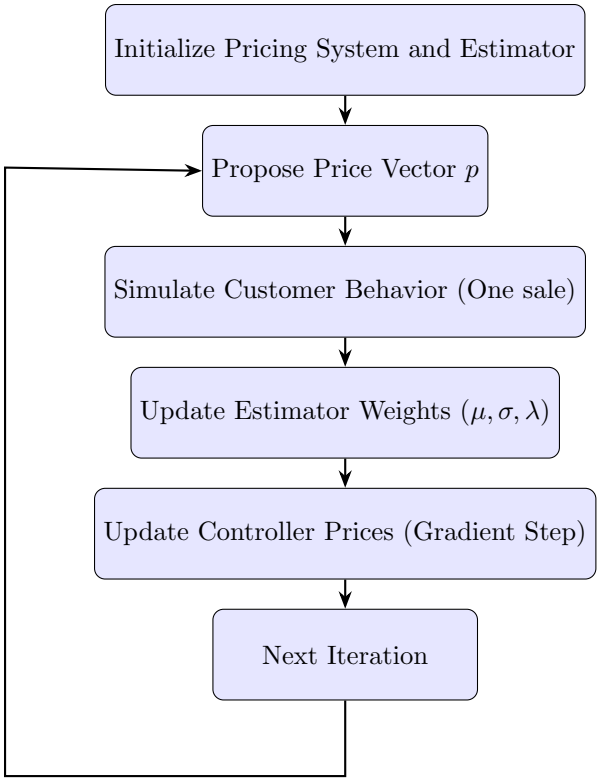

Figure 4: This flowchart shows the high-level steps the feedback controller takes to optimize prices.

ters and guide price optimization. By integrating stochastic gradient-based control with a Sequential Monte Carlo estimator, the system learns to price effectively from sparse feedback alone. This allows the controller to adapt in settings where behavior may drift, making it suitable for real-world deployments where direct data on willingness-to-pay is either unavailable or unreliable.

Perhaps most importantly, this work deepened my understanding of how inference and optimization can be tightly coupled in dynamic systems. I also gained a greater appreciation for simulation design, because I had to go through many iterations of performance enhancement and system redesign to run all of the simulations on my laptop.

6.1 Further Work

This exploration is somewhat limited in scope, and there are many aspects of real-world pricing systems that are not accounted for in this project. Some of these are:

- **Preference Heterogeneity:** Rather than a valuation that is a factor of cost, some tiers of services appeal more to certain customers than others due to components independent of the cost. For example, a multiple devices plan for streaming services would be valued more by families than by individuals.

- **Irregular Valuation Distributions:** In real-world systems, it may be the case where valu-

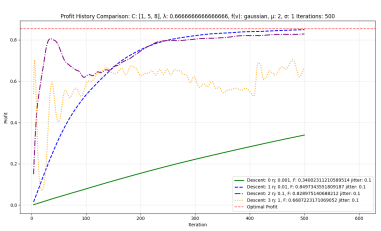

Figure 5: Profit Comparison

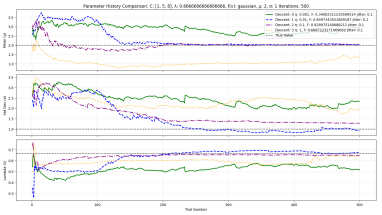

Figure 6: Parameter Estimation Comparison

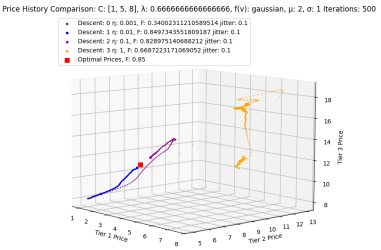

Figure 7: Price Comparison

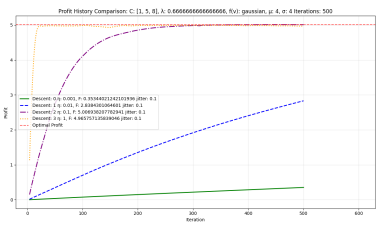

Figure 8: Profit Comparison

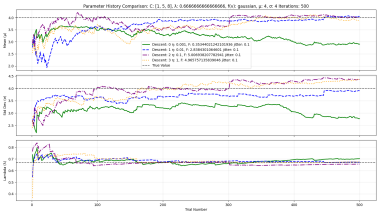

Figure 9: Parameter Estimation Comparison

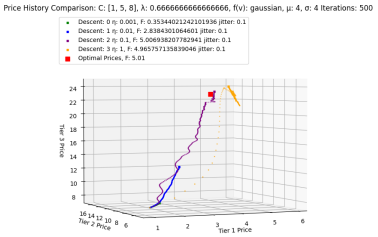

Figure 10: Price Comparison

Figure 11: *Impact of population parameters and learning rate on profit, parameter estimates, and pricing.* In figures 1, 2 and 3, $\mu = 2, \sigma = 1$. The highest learning rate of 1 is too large, so it overshoots the optimal prices and gets stuck in a local minima. Learning rate 0.01 is optimal in this case, smoothly converging on correct prices and parameters. In figures 4, 5 and 6, the highest learning rate of 1 is actually not too large, and quickly reaches the optimal pricing within 100 iterations, and a learning rate 0.01 is too slow.

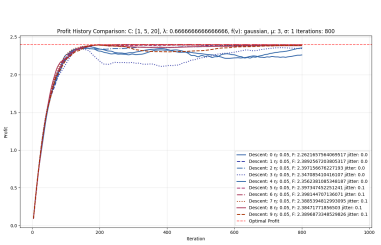

Figure 12: Profit Comparison

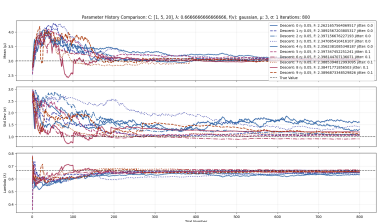

Figure 13: Parameter Estimation Comparison

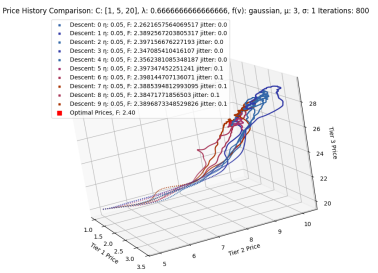

Figure 14: Price Comparison

Figure 15: *Impact of jitter of 0.1 (10% standard deviation) vs no jitter on pricing.* The blue lines display descents without jitter, and the red lines display descents with jitter. In the aggregate, red lines converge to optimal prices faster and more reliably than blue lines. Particularly, figure 10's graph shows that controllers that don't jitter prices reached inaccurate estimates of the population standard deviation.

ation parameters do not follow gaussian *or* uniform distributions.

- **Repeated Interactions:** Customers don't like it when prices increase in a short amount of time, as happens in this paper. [12] It would be interesting to explore the tradeoff between

varying prices and losing customer interest.

- **Dynamic Valuation Parameters:** Customer valuations are not constant, and can be impacted by seasonality, the economy, and even interactions with the brand that may impact the customer's perception of the product value. Modeling this will make this price optimization framework far more applicable to real-world settings.
- **Competition:** This paper examines a monopoly. Most companies do not have monopolies, so it would be interesting to do a study of tiered pricing in a competitive environment.

References

1. Perloff JM. Microeconomics: Theory and Applications with Calculus. 2nd. Retrieved from https://www.sfu.ca/~wainwrig/Econ201/6500/Perloff/11M_Perloff_8008884_02_Micro_C11.pdf. Addison-Wesley, 2011. Chap. 11:4.
2. Mussa M and Rosen S. Monopoly and Product Quality. Journal of Economic Theory 1978;18:301–17.
3. Poveda JI, Brown PN, Marden JR, and Teel AR. A class of distributed adaptive pricing mechanisms for societal systems with limited information. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017:1490–5. DOI: 10.1109/CDC.2017.8263863.
4. Chandrasekaran S, Pasquale GD, Belgioioso G, and Dörfler F. Network-aware Recommender System via Online Feedback Optimization. 2024. arXiv: 2408.16899 [eess.SY]. URL: https://arxiv.org/abs/2408.16899.
5. Moradpour J, Zhang W, Grootendorst P, Anis AH, and Hollis A. Modeling Tiered Pricing Frameworks: A Simulation Approach. Value in Health 2023;26:351–8.
6. Lee SH, Jeong HY, and Seo SW. Optimal pricing and capacity partitioning for tiered access service in virtual networks. Computer Networks 2013;57:3941–56.
7. Ren S, Park J, and Schaar M van der. User subscription dynamics and revenue maximization in communications markets. In: *2011 Proceedings IEEE INFOCOM*. 2011:2696–704. DOI: 10.1109/INFCOM.2011.5935099.
8. Schoengold K and Zilberman D. The economics of tiered pricing and cost functions: Are equity, cost recovery, and economic efficiency compatible goals? Water Resources and Economics 2014;7:1–18.
9. Kingma DP and Ba J. Adam: A Method for Stochastic Optimization. CoRR 2014;abs/1412.6980.
10. Burhenne S and Jacob D. Sampling based on Sobol' sequences for Monte Carlo techniques applied to building simulations. In: 2011:1816–23.
11. Berzuini C and Gilks W. RESAMPLE-MOVE Filtering with Cross-Model Jumps. In: *Sequential Monte Carlo Methods in Practice*. Ed. by Doucet A, Freitas N de, and Gordon N. New York, NY: Springer New York, 2001:117–38. DOI: 10.1007/978-1-4757-3437-9_6. URL: https://doi.org/10.1007/978-1-4757-3437-9_6.

12. Rotemberg JJ. Customer anger at price increases, changes in the frequency of price adjustment and monetary policy. Journal of Monetary Economics 2005;52. SNB:829–52.