A Note on Optimal Product Pricing

Maximilian Schaller Stephen Boyd

December 4, 2025

Abstract

We consider the problem of choosing prices of a set of products so as to maximize profit, taking into account self-elasticity and cross-elasticity, subject to constraints on the prices. We show that this problem can be formulated as maximizing the sum of a convex and concave function. We compare three methods for finding a locally optimal approximate solution. The first is based on the convex-concave procedure, and involves solving a short sequence of convex problems. Another one uses a custom minorize-maximize method, and involves solving a sequence of quadratic programs. The final method is to use a general purpose nonlinear programming method. In numerical examples all three converge reliably to the same local maximum, independent of the starting prices, leading us to believe that the prices found are likely globally optimal.

Contents

1 Introduction

1.1 Optimal pricing

Already in the mid 19th century, researchers studied the relationship between price and demand for a product, and how to balance marginal revenue and marginal cost (implying profit-maximizing prices). In the 1863 book [Cou63], the producer decides on the production quantity, and the price of the product is determined by the market, as a function of quantity. In early work from the 1930s, instead, the decision variable is the price itself, and *price elasticity of the demand*, the marginal change of demand due to marginal change in price, is used to derive the so-called *Lerner markup rule* [Ler34]. Also in the 1930s, cross-price substitution was first studied [AH34], which laid the foundation for pricing portfolios of products, with product substitutes (where one product may replace another) and complements (where one product is typically sold along with another). Later in the 1950s, this concept was extended to the case where a break-even constraint is imposed [Boi56], and numerical optimization methods including linear and quadratic programming were mentioned in the context of pricing [Sam52, Uza58, Hou60]. It took until the 1990s when frameworks for optimal pricing subject to generic constraints (*e.g.*, on production capacity, inventory, etc.) were widely established [GVR94, Wil93, LT93, §4]. Work from this century has focused on dynamic and personalized pricing [BZ09, FLSL16, BK21] and the use of modern machine learning techniques to improve the demand models used for pricing [HLLBT17, DOBM24]. Further, research from the past two decades has addressed several specialized settings, including joint optimization of prices and production plans in manufacturing [DY06, UU13, BFS16], and optimizing the prices of perishable products, where the demand is a function of price, freshness of the products, and other factors [LYW16, HK17, Dye20, MM23].

There is related work on the theory of multi-product pricing [AV18], which focuses on economic models, but not on numerical optimization methods for choosing prices subject to constraints, as we do. For example, it is common to model demand as the maximizer of the expenditure-adjusted or expenditure-constrained utility of purchasing a portfolio of products, which is sometimes referred to as *Marshallian* demand [AV18, MCWG95, §3.D]. A closely related concept is *Hicksian* demand, which minimizes the total expenditure when purchasing a portfolio of products, subject to a minimum utility [LP09]. Further, multi-product pricing is studied in the context of monopolies [Oi71, BM82, AJPT16], and Cournot oligopolies, where a few firms control a market and maximize their respective profits in a game-theoretic sense [Viv99, JM06, NS18]. In this paper, however, we take a simple model of demand, based on an elasticity matrix or Slutsky substitution matrix, as described in, *e.g.*, [MCWG95, §2.F].

1.2 Our contribution

Optimal pricing problems are nonlinear optimization problems. These can be solved, in the weak sense of possibly finding a locally optimal point, *i.e.*, a feasible point with better objective than nearby points, using generic nonlinear programming methods [LN89, NW06, WB06]. In contrast, *convex optimization problems* can be reliably and efficiently solved, in the strong sense of always finding a point that is feasible and has optimal objective value [BV04]. While many practical problems are convex, many others, including the optimal pricing problem, are not. We will see that the optimal pricing problem can be expressed as an optimization problem that is, roughly speaking, close to convex, which means that methods that exploit this structure can be used to solve it. While this solution is still in the weak sense, we at least get the reliability advantages of convex optimization.

We explore two methods for solving the optimal pricing problem that rely on convex optimization. One method is *minorization maximization* (MM) [SBP16], where in each iteration the non-concave profit function is under-approximated, *i.e.*, *minorized*, by a concave function, which yields a convex optimization problem that can be effectively solved. The other method is a special case of MM, the convex-concave procedure (CCP), where the minorization is obtained by linearizing the convex terms in the objective [LB16, SDGB16]. Numerical experiments show that these methods work well, in the sense of reliably finding locally optimal prices even for problems with thousands of products.

In our numerical experiments we find that the two methods proposed, and a generic non-linear programming solver, always converge to the same prices, independent of initialization. This *suggests* that the prices found by these methods may be global, *i.e.*, the ones that truly give the highest profit subject to the pricing constraints. But we have not shown this.

Code and data to reproduce the results of this paper, as well as to solve general optimal pricing problems, is available at

`https://github.com/cvxgrp/optimal-pricing`.

1.3 Outline

In §2 we introduce a generic product pricing problem (PPP) for maximizing profit generated by selling multiple products, subject to general convex constraints. In §3, we give concrete, practical examples for such constraints, before we describe three solution methods for the PPP in §4. In §5 we assess the convergence properties and numerical performance of the three solution methods on numerical examples.

2 Optimal pricing

2.1 Prices

We are to choose positive prices $p_1, \ldots, p_n$ for n different products or services. Each product has a positive nominal price p_i^{nom}, which is typically the price at which the product (or a similar reference product) has been sold in the past. We denote by

$$\pi_i = \log(p_i/p_i^{\text{nom}}) = \log p_i - \log p_i^{\text{nom}}, \quad i = 1, \ldots, n,$$

the (logarithmic) fractional price change with respect to the nominal price. For example, if $\pi_i = 0$, the price of product i is the nominal price. If $\pi_i = -0.2$, the product price is a factor $\exp(-0.2) \approx 0.819$ compared to the nominal price, *i.e.*, 18.1% lower. We let $\pi \in \mathbf{R}^n$ denote the vector of price changes. Our goal is to choose the prices, or equivalently, the price change vector π.

Price constraints. We are given a set of constraints that the prices must satisfy, which we express in terms of the price changes as $\pi \in \mathcal{P}$, where $\mathcal{P} \subset \mathbf{R}^n$ is the set of allowed price changes. At the very least this will include lower and upper limits on the price changes. It can also specify relationships among the prices, such as that one product price must be at least 10% higher than another. We will describe many other constraints later, in §3.

We will assume that $\mathcal{P}$ is polyhedral, *i.e.*, is described by a set of linear equality and inequality constraints, as

$$\mathcal{P} = \{\pi \mid A\pi = b, \ F\pi \leq g\},$$

where $A \in \mathbf{R}^{k\times n}$, $b \in \mathbf{R}^k$, $F \in \mathbf{R}^{l\times n}$, and $g \in \mathbf{R}^l$. We assume that $\mathcal{P}$ is nonempty and bounded.

2.2 Demand

We denote the (positive) demand for the ith product as d_i. Each product has a positive nominal demand d_i^{nom}, which is the demand for product i at its nominal price p_i^{nom}. We denote by

$$\delta_i = \log(d_i/d_i^{\mathrm{nom}}) = \log d_i - \log d_i^{\mathrm{nom}}, \quad i = 1, \ldots, n,$$

the (logarithmic) fractional demand change, with respect to the nominal demand. We let $\delta \in \mathbf{R}^n$ denote the vector of demand changes. (It follows that the revenue for product i is $p_i d_i$, and the nominal revenue for product i is $p_i^{\mathrm{nom}} d_i^{\mathrm{nom}}$, denoted by r_i^{nom}.)

Price elasticity of the demand. We model the price elasticity of the demand as in [MCWG95, §2.F],

$$\delta = E\pi, \tag{1}$$

where $E \in \mathbf{R}^{n\times n}$ is the *elasticity matrix*, also referred to as the *Slutsky substitution matrix*. The entry E_{ij} is the elasticity of the demand for product i with respect to the price of product j. When $i = j$, this is called a *self-elasticity*. When $i \neq j$, this is called a *cross-elasticity*. This model is basically the linearization of δ as a function of π, around $\pi = 0$ (*i.e.*, nominal prices) [Per09, Var14].

We make no assumptions about the elasticity matrix, but we mention here some typical attributes; see, *e.g.*, [Var92, MCWG95] for more on elasticity matrices. In almost all practical cases, the self-elasticities E_{ii} are negative, which means that an increase in price results in a decrease in demand for that product. Cross-elasticities can be positive or negative. For example, when products i and j are substitutes for each other, E_{ij} and E_{ji} will be positive, as a higher price of one will result in higher demand for the other product (which will be bought as a substitute). When two products are complements (*e.g.*, printer and ink), then their cross-elasticity will be negative. Using the example of printer and ink, increasing the price for ink will decrease the demand for ink, as well as the demand for printers. While the order of magnitude of the self-elasticities is typically -1, the cross-elasticities are typically smaller in magnitude. The elasticity matrix is typically sparse, *e.g.*, block diagonal, with the blocks representing similar or related products.

Utility-based demand. We mention here a different but related demand model, based on a utility function. A utility-based demand model has the form

$$d = \mathcal{D}(p) = \underset{\tilde{d}}{\operatorname{argmax}}(U(\tilde{d}) - p^T \tilde{d}), \tag{2}$$

where $U : \mathbf{R}^n_+ \to \mathbf{R}$ is a strictly concave and increasing utility function; see, *e.g.*, [AV18]. This means that the demand maximizes the utility of purchasing a portfolio of products, minus the total expenditure.

The utility-based demand (2) does not in general have the form (1), but we can form a local approximation that does. We first form the first order Taylor approximation of $\mathcal{D}$ at the nominal price and demand, to obtain

$$d \approx d^{\text{nom}} + D\mathcal{D}(p^{\text{nom}})(p - p^{\text{nom}}),$$

where $D\mathcal{D}$ is the derivative of $\mathcal{D}$, which can be shown to have the form

$$D\mathcal{D}(p^{\text{nom}}) = \left(\nabla^2 U(p^{\text{nom}})\right)^{-1},$$

assuming U is twice differentiable. From this we obtain the first order elasticity approximation $\delta \approx E^{\text{util}}\pi$, with

$$E^{\text{util}} = \mathbf{diag}(d^{\text{nom}})^{-1} \left(\nabla^2 U(p^{\text{nom}})\right)^{-1} \mathbf{diag}(p^{\text{nom}}),$$

where we use the first order approximations $\delta_i \approx (d_i - d_i^{\text{nom}})/d_i^{\text{nom}}$ and $\pi_i \approx (p_i - p_i^{\text{nom}})/p_i^{\text{nom}}$. Since $\left(\nabla^2 U(p^{\text{nom}})\right)^{-1}$ is symmetric and negative definite, it follows that if the elasticity matrix E comes from linearization of a utility-based demand model, then $\mathbf{diag}(r^{\text{nom}})E$, *i.e.*, E with its rows scaled by the nominal revenues, is symmetric and negative definite. This implies, for example, that $E_{ii} < 0$.

Here we are simply observing that if the elasticity matrix comes from a utility-based demand, then it has this specific form; in the sequel, however, we make no assumptions about the structure of E.

2.3 Profit

The revenue for product i is $d_i p_i$; the total revenue is $\sum_{i=1}^n d_i p_i$. Let c_i be the (positive) cost to provide or produce one unit of product i, so the cost of providing product i is $d_i c_i$. The profit for product i is $d_i(p_i - c_i)$, and the total profit is

$$P = \sum_{i=1}^n d_i(p_i - c_i).$$

We can express P in terms of the fractional values as

$$P = \sum_{i=1}^{n} d_i^{\text{nom}} e^{\delta_i} (p_i^{\text{nom}} e^{\pi_i} - c_i).$$

We denote nominal revenue by $r_i^{\text{nom}} = d_i^{\text{nom}} p_i^{\text{nom}}$ and cost for providing product i at nominal demand by $\kappa_i^{\text{nom}} = d_i^{\text{nom}} c_i$. Together with the demand model (1), the profit is

$$P = \sum_{i=1}^{n} \left(r_i^{\text{nom}} e^{\delta_i + \pi_i} - \kappa_i^{\text{nom}} e^{\delta_i} \right).$$

This is readily interpreted. When increasing the price for the ith product, *i.e.*, $\pi_i > 0$, we observe two effects. First, revenue changes by the factor $\exp(\delta_i + \pi_i)$, where the price increase enters via δ_i in terms of changed demand, and additionally via π_i, as each unit of product i is sold at the increased price. The second effect we observe is that the total cost for product i changes by the factor $\exp(\delta_i)$.

2.4 Optimal pricing problem

Our goal is to choose the prices, subject to the constraints, so as to maximize profit. This can be expressed as the *product pricing problem* (PPP)

$$\begin{array}{ll} \text{maximize} & P \\ \text{subject to} & \pi \in \mathcal{P}, \end{array}$$

with variable π. It can be written explicitly as

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} \left(r_i^{\text{nom}} e^{\delta_i + \pi_i} - \kappa_i^{\text{nom}} e^{\delta_i} \right) \\ \text{subject to} & \delta = E\pi, \quad A\pi = b, \quad F\pi \leq g, \end{array} \tag{3}$$

with variables π and δ. The data for the PPP are the vectors of nominal revenue and cost r^{nom}, κ^{nom}; the elasticity matrix E; and the matrices and vectors A, b, F, g, which define the price constraints.

PPP is not a convex optimization problem, since the objective is not concave [BV04]. It can, however, be effectively solved using several methods that rely on convex optimization, as described in §4.

Independent prices and demands. When E is diagonal, *i.e.*, all cross-elasticities are zero, and the set of allowable prices is a set of lower and upper limits on the individual price changes, $\mathcal{P} = \{\pi \mid \pi^{\min} \leq \pi \leq \pi^{\max}\}$, the PPP is readily solved analytically, by maximizing the profit associated with each product separately. When $E_{ii} < -1$, the solution is

$$\pi_i^\star = \textbf{clip} \left(\log \left(\frac{\kappa_i^{\text{nom}}}{r_i^{\text{nom}}} \frac{E_{ii}}{E_{ii} + 1} \right), \pi_i^{\min}, \pi_i^{\max} \right),$$

where

$$\mathbf{clip}(x,l,u) = \begin{cases} l & x < l \\ x & l \le x \le u \\ u & x > u. \end{cases}$$

When $E_{ii} \ge -1$, the solution is $\pi_i^\star = \pi_i^{\max}$.

3 Price constraints

In this section we briefly describe some practical constraints that can be imposed on the prices, by incorporating them into $\mathcal{P}$. They can be combined with one another, and with other polyhedral constraints, and assembled into $\mathcal{P}$.

3.1 Simple constraints

Price limits. We can impose price limits, of the form

$$p_i^{\min} \le p_i \le p_i^{\max}, \quad i = 1, \ldots, n,$$

where $p_i^{\min}$ and $p_i^{\max}$ are given limits. (A minimum price also sets a minimum profit, or if negative, a maximum loss, for each product.) We express these in terms of the price changes as

$$\pi_i^{\min} \le \pi_i \le \pi_i^{\max}, \quad i = 1, \ldots, n,$$

where $\pi_i^{\min} = \log(p_i^{\min}/p_i^{\mathrm{nom}})$, and similarly for $\pi_i^{\max}$. Limits on price changes can also be used to specify a maximum change in price from nominal. For example, to restrict prices to be within ±20% of nominal, we take $\pi_i^{\min} = \log(0.8)$ and $\pi_i^{\max} = \log(1.2)$ for $i = 1, \ldots, n$.

Demand limits. We can also put limits on (predicted) demand, as

$$\delta_i^{\min} \le \delta_i \le \delta_i^{\max}, \quad i = 1, \ldots, n.$$

This can be done for several reasons. We might limit the predicted demand changes to not exceed 20%, using $\delta_i^{\min} = \log(0.8)$ and $\delta_i^{\max} = \log(1.2)$, because we do not trust the demand model when it predicts larger demand changes. It can also be used to limit demand to not exceed our capacity to provide the product, or some fraction of the total available market.

Partial pricing. To determine the prices of only a subset of products, we simply impose $\pi_i = 0$ when the product i is not to be changed. Here the PPP still takes into account the change in demand for such products, induced by the changes in price of other products.

Inter-price inequalities. We can impose inequality relations between prices, such as the price of product i must be at least 10% higher than the price of product j, as

$$\pi_i - \pi_j \ge \log(1.1) + \log(p_j^{\mathrm{nom}}/p_i^{\mathrm{nom}}).$$

3.2 Pricing policy

The constraints described above directly constrain the price changes. Here we describe another setting where there are constraints among the price changes, induced by imposing a *pricing policy*, which is a simple formula that determines the price of each product based on some attributes of a product. Attributes can be Boolean, categorical, ordinal, or numerical. We illustrate this with the example of a hiking jacket. A Boolean attribute could be whether it is waterproof or not. A categorical attribute could be the color of the jacket. An ordinal attribute might be the thermal protection, with more protection deemed higher. A numerical attribute might be the weight of the jacket (measured in grams), or its size. We denote the attributes for product i as $a_i \in \mathcal{A}$.

We will consider a simple additive form of a pricing policy, based on the values of m attributes of a product, given by

$$\pi_i = \sum_{j=1}^{m} \theta_j \phi_j(a_{ij}), \quad i = 1, \ldots, n, \tag{4}$$

where a_{ij} is the value of attribute j for product i, and $\theta = (\theta_1, \ldots, \theta_m)$ is a vector of parameters that specify the policy. The functions $\phi_1, \ldots, \phi_m$ map the attributes to numerical values. We denote the set of allowable parameters as $\theta \in \Theta$, with Θ polyhedral. In the context of machine learning, a prediction which is a sum of functions of a set of features or attributes is called a *generalized additive model* (GAM) [HT86, Has17]. So we call the pricing policy (4) a *generalized additive pricing policy*.

The constraint that we follow a pricing policy can be directly written as

$$\pi = C\theta, \qquad \theta \in \Theta,$$

where θ is an additional variable to be determined. Together with the demand model $\delta = E\pi$, we can write $\delta = EC\theta$, giving rise to the following interpretation. As E gives the *price* elasticity of the demand, EC gives the *parameter* elasticity of the demand (with respect to the parameters in the affine pricing policy).

Affine policy. Perhaps the simplest policy uses $\phi_j(u) = u$, and we take one attribute to be the constant one. The policy has the simple form $\pi_i = \theta^T a_i$. We interpret θ_j as the amount by which we fractionally increase the price for one unit of increase in attribute j. For small price changes, we have $\pi_i = \log(p_i/p_i^{\text{nom}}) \approx p_i/p_i^{\text{nom}} - 1$, and we can interpret the model as

$$p_i/p_i^{\text{nom}} \approx 1 + \theta^T a_i.$$

Such pricing policies are sometimes referred to as *hedonic* [SMZ05], as the price is broken down into the individual values of the constituent characteristics or attributes of the product.

Value-based policy. When $\phi_j(u) = \log(u)$, then the pricing policy can be interpreted as

$$p_i/p_i^{\text{nom}} = \prod_{j=1}^{m} a_{ij}^{\theta_j}.$$

Here, the parameters θ_j are elasticities of prices with respect to product attributes. This model is analogous to the Cobb-Douglas production function [ZKD66], and can be interpreted as the value that can be created from a product. Therefore, we might call this a *value-based* pricing policy.

Cost-based policy. Here is an interesting special case, where $m = 2$. Suppose we have $a_{i1} = p_i^{\text{nom}}/c_i$ (the nominal *markup factor*) with $\phi_1(u) = -\log(u)$ and $\theta_1 = 1$ imposed by Θ. Also, suppose that $\phi_2(u) = 1$. Then, $\pi_i = -\log(a_{i1}) + \theta_2$ and the resulting markup factor is

$$p_i/c_i = e^{\theta_2}.$$

We call this a *cost-based* pricing policy [GA18]. Clearly, there are simpler ways to represent such a policy, but we use this to demonstrate the expressiveness of equation (4).

4 Solution methods

We present three methods for effectively solving the non-convex problem (3). The first two methods exploit the fact that the objective function is a sum of convex and concave exponentials. These methods linearize the convex exponentials to make the objective concave, and solve the modified (convex) problem repeatedly. We can also approximate the concave exponentials by quadratic functions, in which case the problem solved in each iteration is a quadratic program (QP), for which specialized solvers have been developed [SBG+20]. We call this method *quadratic minorization-maximization*. The third method views the problem as a general nonlinear programming problem, which can be (approximately) solved using generic techniques [LN89, NW06, WB06].

4.1 Convex-concave procedure

Problem (3) is a so-called *difference of convex program*, since the objective is a difference of convex functions [LB16] (in this case convex exponentials). We can solve it with the convex-concave procedure (CCP) [LB16, SDGB16].

First, we initialize $\hat{\pi} = \hat{\delta} = 0$, corresponding to no price changes. Then, we linearize the convex revenue terms in the objective of (3), and solve the convex problem

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^{n} (r_i^{\text{nom}} e^{\hat{\delta}_i + \hat{\pi}_i} (\delta_i + \pi_i) - \kappa_i^{\text{nom}} e^{\delta_i}) \\ \text{subject to} & \delta = E\pi, \quad A\pi = b, \quad F\pi \leq g, \end{array} \tag{5}$$

where we have dropped constant terms from the objective. We assign $\pi^\star$ and $\delta^\star$ from the solution of (5) to $\hat{\pi}$ and $\hat{\delta}$, respectively, re-solve (5), and repeat until the profit converges.

The profit increases in each iteration, and so converges. We cannot in general claim that it converges to the global maximum of the PPP, but we suspect that in practical cases it almost always does.

The convex problem (5) can be solved using a generic method for convex optimization, or by expressing the problem as a cone program, using the exponential cone [GC24].

Specification using DCP. It is particularly useful to pose problem (5) using disciplined convex programming (DCP). The DCP rules allow the user to model convex optimization problems with instructions that are very close to the mathematical problem description [AVDB18]. The modeling language CVXPY [DB16] uses DCP to verify convexity and to translate the problem to a form accepted by standard convex optimization solvers. With DCP, the code that declares the problem is human-readable and it is easy to modify the problem, *e.g.*, to add constraints. We give an example in §5.

4.2 Quadratic minorization-maximization

In minorization-maximization (MM) [SBP16], a minorizer to the objective is maximized at each iteration. The minorized objective increases each iteration, and so the actual objective does as well. It follows that the objective increases each iteration and therefore converges. The previously described CCP is a special case of MM, where the minorizer is obtained by linearizing the convex part of the objective.

In addition to linearizing the convex part of the objective, we can also approximate the concave exponentials by concave quadratics, to obtain a concave approximation of the objective that is a minorizer of the actual objective. We call this quadratic minorization-majorization (QMM). With this method the problems solved each iteration are QPs, for which specialized solvers have been developed [SBG+20].

Quadratic minorizer. With prices bounded as $\pi^{\min} \leq \pi \leq \pi^{\max}$, we deduce that the demand is bounded as

$$\delta \leq \delta^{\max} = (E)_+\pi^{\max} - (E)_-\pi^{\min},$$

where $(x)_+ = \max\{x, 0\}$ and $(x)_- = \max\{-x, 0\}$ (elementwise). With that, we can construct a minorizer to $-\exp(\delta_i)$, or, equivalently, a majorizer to $\exp(\delta_i)$. We take the second-order Taylor approximation of $\exp(\delta_i)$ around $\delta_i = \hat{\delta}_i$, and scale the quadratic term by $2\beta_i > 0$ as

$$e^{\hat{\delta}_i} + e^{\hat{\delta}_i}(\delta_i - \hat{\delta}_i) + \beta_i e^{\hat{\delta}_i}(\delta_i - \hat{\delta}_i)^2.$$

We require this to be a majorizer to $\exp(\delta_i)$ with smallest possible β_i, to minimize approximation error. In other words, we require $\exp(\delta_i)$ and its quadratic majorizer to intersect at $\delta_i = \delta_i^{\max}$. Abbreviating $b_i = \delta_i^{\max} - \hat{\delta}_i$, this can be written as

$$e^{\hat{\delta}_i + b_i} = e^{\hat{\delta}_i}(1 + b_i) + \beta_i e^{\hat{\delta}_i} b_i^2,$$

which we solve for β_i as

$$\beta_i = (e^{b_i} - b_i - 1)/b_i^2. \qquad (6)$$

Algorithm. First, we initialize $\hat{\pi} = \hat{\delta} = 0$ and $\beta_i = (\exp(\delta_i^{\max}) - \delta_i^{\max} - 1)/(\delta_i^{\max})^2$, for $i = 1, \ldots, n$. Then, we linearize the convex revenue terms, replace the exponential cost terms with their quadratic minorizers, and solve the quadratic program

$$\begin{array}{ll} \text{maximize} & \sum_{i=1}^n (r_i^{\text{nom}} e^{\hat{\delta}_i + \hat{\pi}_i}(\delta_i + \pi_i) - \kappa_i^{\text{nom}} e^{\hat{\delta}_i}(\delta_i + \beta_i(\delta_i - \hat{\delta}_i)^2)) \\ \text{subject to} & \delta = E\pi, \quad A\pi = b, \quad F\pi \leq g, \end{array} \tag{7}$$

where we dropped constant terms from the objective. We assign $\pi^\star$ and $\delta^\star$ from the solution of (7) to $\hat{\pi}$ and $\hat{\delta}$, respectively, update all β_i according to (6), re-solve (7), and repeat until the profit converges.

As with CCP, we cannot claim that the profit always converges to the global maximum, but we suspect that in practical cases it almost always does.

4.3 Nonlinear programming

One can view the objective function of problem (3) as an instance of general nonlinear (and twice differentiable) functions, and apply local nonlinear programming (NLP) methods that use the local gradient or Hessian (approximation) at every iteration [Ber97, LN89, NW06, KT13]. Well-known NLP solver implementations are the open-source IPOPT [WB06] and the proprietary KNITRO [BNW06].

5 Numerical examples

We compare solving PPPs with CCP, QMM, and NLP. We use a relative objective tolerance of 0.001 for all three methods. To solve the convex subproblems of CCP, we use the open-source convex optimization solver SCS [OCPB16]. To solve the quadratic subproblems of QMM, we use the open-source QP solver OSQP [SBG+20]. For NLP, we use the open-source NLP solver IPOPT [WB06]. We interface with all solvers via CVXPY [DB16] and use their respective default settings. We run the experiments on an Apple M1 Pro.

CVXPY specification. Figure 1 shows how the convex-concave procedure outlined in §4.1 is implemented with a few lines of CVXPY code. In lines 4–12, problem (5), with price limits and a pricing policy, is modeled with CVXPY. In line 7, we use a `CallbackParam`, such that the linearization will be updated automatically when the problem is re-solved. In line 11, the ijth entry of `C` stores the attribute a_{ij}. In line 15, the convex-concave procedure is initialized, before iterations are run in lines 16 and 17. The code for QMM and NLP is very similar.

5.1 Data generation

Elasticity, revenue, and cost. We generate random instances of the PPP of various dimensions. We consider a block-diagonal elasticity matrix E with block size 10, representing

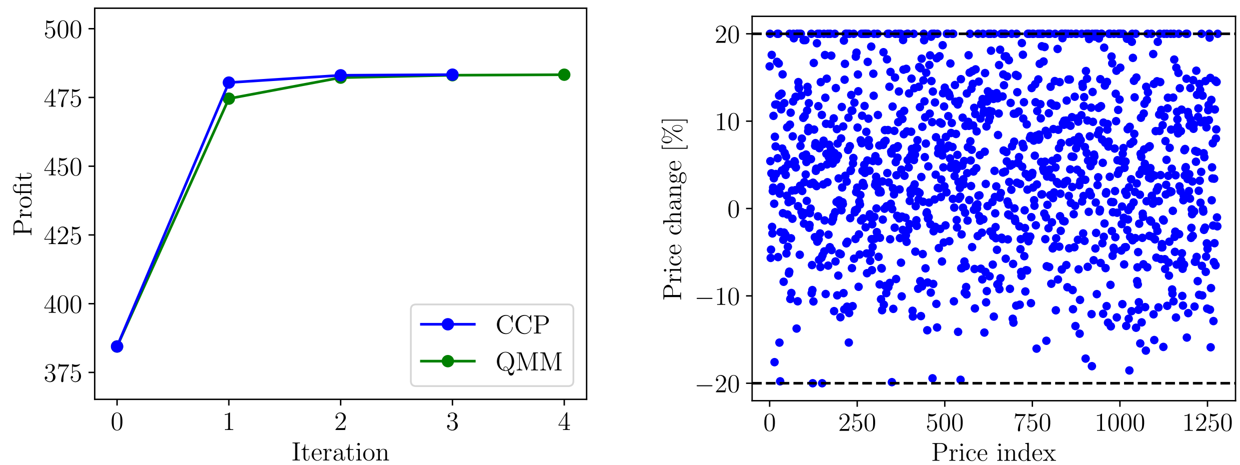
```python
import cvxpy as cp

# variables and parameters
pi = cp.Variable(n, bounds=[pi_min, pi_max])
delta = cp.Variable(n)
theta = cp.Variable(m)
rscaled = cp.CallbackParam(
                    lambda: rnom * np.exp(E @ pi.value + pi.value), n)

# objective and constraints
obj = rscaled @ (delta + pi) - knom @ cp.exp(delta)
con = [delta == E @ pi, pi == C @ theta]
prob = cp.Problem(cp.Maximize(obj), con)

# solve
pi.value = np.zeros(n)
for i in range(5):
    prob.solve()
```

Figure 1: Modeling and solving the PPP with CVXPY. The dimensions `n`, `m` and the data `pi_min`, `pi_max`, `rnom`, `knom`, `E`, `C` are given.



(a) Profit versus CCP and QMM iterations.

(b) Resulting price changes.

Figure 2: Profit and price changes for $n = 1280$ and $m = 256$.

groups of related products that might be substitutes or complements. We sample the self-elasticities E_{ii} between -3.0 and -1.0, and the cross-elasticities E_{ij} (within each block) between -0.05 and 0.05. We set the nominal revenue per product r_i^{nom} between 1.0 and 5.0, and the nominal cost to $\kappa_i^{\mathrm{nom}} = 0.9 r_i^{\mathrm{nom}}$, *i.e.*, a nominal profit margin of 10%.

Constraints. We limit price changes to $\pm 20\%$, and impose an affine pricing policy as described in §3.2, where the attributes a_{ij} are sampled IID from $\mathcal{N}(0, 1)$. We do not restrict the policy parameters directly, *i.e.*, $\Theta = \mathbf{R}^m$.

5.2 Results

Convergence and price changes. We solve the problem for $n = 1280$ and $m = 256$ with CCP (in 2.4 seconds), QMM (in 1.5 seconds), and NLP (in 2.7 seconds). Figure 2a shows the profit versus iterations of CCP and QMM. As expected, the profit increases at each iteration and converges after an increase from about 384 to about 483. It takes QMM one more iteration to converge, due to its initial approximation errors of the concave exponentials. Still, the overall solve time is smaller with QMM, since each QP can be solved fast. Figure 2b presents the ultimate price changes. We observe that almost all prices are changed, and the $\pm 20\%$ limit takes effect for a number of prices.

Scaling. To explore how the methods scale with problem size, we generate PPP instances with dimensions $n = 20, 40, 80, 160, 320, 640, 1280, 2560$, using $m = n/5$ parameters in the pricing policy. Figure 3 shows the solve times for each value of n, for each of the three methods CCP, QMM, and NLP. Overall, the solve times are comparable between the three solution methods. QMM solves the problems fastest for almost all sizes. The positive

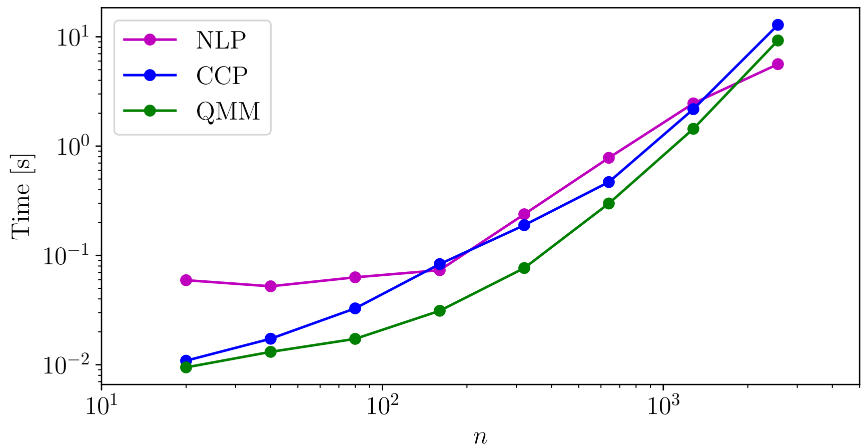


Figure 3: Solve times for different problem sizes, with CCP, QMM, and NLP.

effect of dealing with a quadratic program at each iteration of QMM (and being able to use a specialized solver) appears to outweigh the effect of larger approximation errors. In fact, QMM took around 3–5 iterations for all problem sizes, just slightly more than the 3–4 iterations required by CCP. These scaling results were insensitive to the seed used for generating the data.

Effect of initialization. In our final experiments, we explore the effect of the starting point on the final prices found. We randomly initialize π between $\pi^{\min}$ and $\pi^{\max}$, solve the PPP with all three methods, repeating this 1000 times. In all such cases, the method converged to the same prices as our all-zeros initialization, with all final objectives within our objective tolerance. We cannot claim that the prices found are globally optimal, but these experiments suggest that they might be.

Acknowledgments

The authors thank Garrett van Ryzin for his insightful discussions and valuable suggestions.

References

[AH34] R. Allen and J. Hicks. A reconsideration of the theory of value, II. *Economica*, 1(2):196–219, 1934.

[AJPT16] R. Amir, J. Jin, G. Pech, and M. Tröge. Prices and deadweight loss in multiproduct monopoly. *Journal of Public Economic Theory*, 18(3):346–362, 2016.

[AV18] M. Armstrong and J. Vickers. Multiproduct pricing made simple. *Journal of Political Economy*, 126(4):1444–1471, 2018.

[AVDB18] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[Ber97] D. Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3):334–334, 1997.

[BFS16] N. Bajwa, B. Fontem, and C. Sox. Optimal product pricing and lot sizing decisions for multiple products with nonlinear demands. *Journal of Management Analytics*, 3(1):43–58, 2016.

[BK21] G. Ban and N. Keskin. Personalized dynamic pricing with machine learning: High-dimensional features and heterogeneous elasticity. *Management Science*, 67(9):5549–5568, 2021.

[BM82] D. Baron and R. Myerson. Regulating a monopolist with unknown costs. *Econometrica: Journal of the Econometric Society*, pages 911–930, 1982.

[BNW06] R. Byrd, J. Nocedal, and R. Waltz. KNITRO: An integrated package for nonlinear optimization. In *Large-scale nonlinear optimization*, pages 35–59. Springer, 2006.

[Boi56] M. Boiteux. Sur la gestion des monopoles publics astreints à l'équilibre budgétaire. *Econometrica, Journal of the Econometric Society*, pages 22–40, 1956.

[BV04] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BZ09] O. Besbes and A. Zeevi. Dynamic pricing without knowing the demand function: Risk bounds and near-optimal algorithms. *Operations research*, 57(6):1407–1420, 2009.

[Cou63] A. Cournot. *Principes de la Théorie des Richesses*. Hachette, 1863.

[DB16] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[DOBM24] K. Douaioui, R. Oucheikh, O. Benmoussa, and C. Mabrouki. Machine learning and deep learning models for demand forecasting in supply chain management: A critical review. *Applied System Innovation (ASI)*, 7(5), 2024.

[DY06] S. Deng and C. Yano. Joint production and pricing decisions with setup costs and capacity constraints. *Management Science*, 52(5):741–756, 2006.

[Dye20] C. Dye. Optimal joint dynamic pricing, advertising and inventory control model for perishable items with psychic stock effect. *European Journal of Operational Research*, 283(2):576–587, 2020.

[FLSL16] K. Ferreira, B. Lee, and D. Simchi-Levi. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & service operations management*, 18(1):69–88, 2016.

[GA18] R. Guerreiro and J. Amaral. Cost-based price and value-based price: Are they conflicting approaches? *Journal of Business & Industrial Marketing*, 33(3):390–404, 2018.

[GC24] P. Goulart and Y. Chen. Clarabel: An interior-point solver for conic programs with quadratic objectives. *arXiv preprint arXiv:2405.12762*, 2024.

[GVR94] G. Gallego and G. Van Ryzin. Optimal dynamic pricing of inventories with stochastic demand over finite horizons. *Management science*, 40(8):999–1020, 1994.

[Has17] T. Hastie. Generalized additive models. *Statistical models in S*, pages 249–307, 2017.

[HK17] A. Herbon and E. Khmelnitsky. Optimal dynamic pricing and ordering of a perishable product under additive effects of price and time on demand. *European Journal of Operational Research*, 260(2):546–556, 2017.

[HLLBT17] J. Hartford, G. Lewis, K. Leyton-Brown, and M. Taddy. Deep IV: A flexible approach for counterfactual prediction. In *International Conference on Machine Learning*, pages 1414–1423. PMLR, 2017.

[Hou60] H. Houthakker. The capacity method of quadratic programming. *Econometrica, Journal of the Econometric Society*, pages 62–87, 1960.

[HT86] T. Hastie and R. Tibshirani. Generalized additive models. *Statistical science*, 1(3):297–310, 1986.

[JM06] J. Johnson and D. Myatt. Multiproduct Cournot oligopoly. *The RAND Journal of Economics*, 37(3):583–601, 2006.

[KT13] H. Kuhn and A. Tucker. *Nonlinear Programming*. Springer, 2013.

[LB16] T. Lipp and S. Boyd. Variations and extension of the convex–concave procedure. *Optimization and Engineering*, 17(2):263–287, 2016.

[Ler34] A. Lerner. The concept of monopoly and the measurement of monopoly power. *The review of economic studies*, 1(3):157–175, 1934.

[LN89] D. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[LP09] A. Lewbel and K. Pendakur. Tricks with Hicks: The EASI demand system. *American Economic Review*, 99(3):827–863, 2009.

[LT93] J. Laffont and J. Tirole. *A Theory of Incentives in Procurement and Regulation*. MIT press, 1993.

[LYW16] Q. Li, P. Yu, and X. Wu. Managing perishable inventories in retailing: Replenishment, clearance sales, and segregation. *Operations Research*, 64(6):1270–1284, 2016.

[MCWG95] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*, volume 1. Oxford University Press New York, 1995.

[MM23] C. Mahato and G. Mahata. Optimal pricing and inventory decisions for perishable products with multivariate demand function under trade credit. *Operations Research Forum*, 4(2): Article 24, 2023.

[NS18] V. Nocke and N. Schutz. Multiproduct-firm oligopoly: An aggregative games approach. *Econometrica*, 86(2):523–557, 2018.

[NW06] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2006.

[OCPB16] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

[Oi71] W. Oi. A Disneyland dilemma: Two-part tariffs for a Mickey Mouse monopoly. *The Quarterly Journal of Economics*, 85(1):77–96, 1971.

[Per09] J. Perloff. *Microeconomics*. Pearson Education, 2009.

[Sam52] P. Samuelson. Spatial price equilibrium and linear programming. *The American economic review*, 42(3):283–303, 1952.

[SBG+20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[SBP16] Y. Sun, P. Babu, and D. Palomar. Majorization-minimization algorithms in signal processing, communications, and machine learning. *IEEE Transactions on Signal Processing*, 65(3):794–816, 2016.

[SDGB16] X. Shen, S. Diamond, Y. Gu, and S. Boyd. Disciplined convex-concave programming. In *2016 IEEE 55th conference on decision and control (CDC)*, pages 1009–1014. IEEE, 2016.

[SMZ05] S. Sirmans, D. Macpherson, and E. Zietz. The composition of hedonic pricing models. *Journal of real estate literature*, 13(1):1–44, 2005.

[UU13] A. Upasani and R. Uzsoy. Integrated production planning and pricing decisions in congestion-prone capacitated production systems. In *Essays in Production, Project Planning and Scheduling: A Festschrift in Honor of Salah Elmaghraby*, pages 29–68. Springer, 2013.

[Uza58] H. Uzawa. Iterative methods for concave programming. *Studies in linear and nonlinear programming*, 6:154–165, 1958.

[Var92] H. Varian. *Microeconomic Analysis*, volume 3. Norton New York, 1992.

[Var14] H. Varian. *Intermediate Microeconomics with Calculus: A Modern Approach*. WW Norton & Company, 2014.

[Viv99] X. Vives. *Oligopoly Pricing: Old Ideas and New Tools*. MIT Press (MA), 1999.

[WB06] A. Wächter and L. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[Wil93] R. Wilson. *Nonlinear Pricing*. Oxford University Press, 1993.

[ZKD66] A. Zellner, J. Kmenta, and J. Dreze. Specification and estimation of Cobb-Douglas production function models. *Econometrica: Journal of the Econometric Society*, pages 784–795, 1966.