LP-based Subgradient Algorithm for Joint Pricing and Inventory Control Problems

by

Tingting Rao


Submitted to the School of Engineering in partial fulfillment of the requirements for the degree of Master of Science in Computation for Design and Optimization at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY September 2008

© Massachusetts Institute of Technology 2008. All rights reserved.

Author .................................................

School of Engineering
August 11, 2008

Certified by ......................... .................................

Retsef Levi
Robert N. Noyce Career Development Professor
Assistant Professor of Management
Thesis Supervisor

Certified by .................................................

Georgia Perakis
Associate Professor of Operations Research
Thesis Supervisor

Accepted by .................................

Jaime Peraire
Professor of Aeronautics and Astronautics
Codirector, Computations for Design and Optimization
LP-based Subgradient Algorithm for Joint Pricing and Inventory Control Problems

by

Tingting Rao

Submitted to the School of Engineering on August 11, 2008, in partial fulfillment of the requirements for the degree of Master of Science in Computation for Design and Optimization

Abstract

It is important for companies to manage their revenues and reduce their costs efficiently. These goals can be achieved through effective pricing and inventory control strategies. This thesis studies a joint multi-period pricing and inventory control problem for a make-to-stock manufacturing system. Multiple products are produced under shared production capacity over a finite time horizon. The demand for each product is a function of the prices and no back orders are allowed. Inventory and production costs are linear functions of the levels of inventory and production, respectively.

In this thesis, we introduce an iterative gradient-based algorithm. A key idea is that given a demand realization, the cost minimization part of the problem becomes a linear transportation problem. Given this idea, if we knew the optimal demand, we could solve the production problem efficiently. At each iteration of the algorithm, given a demand vector we solve a linear transportation problem and use its dual variables in order to solve a quadratic optimization problem that optimizes the revenue part and generates a new pricing policy. We illustrate computationally that this algorithm obtains the optimal production and pricing policy over the finite time horizon efficiently.

The computational experiments in this thesis use a wide range of simulated data. The results show that the algorithm we study in this thesis indeed computes the optimal solution for the joint pricing and inventory control problem and is efficient as compared to solving a reformulation of the problem directly using commercial software. The algorithm proposed in this thesis solves large scale problems and can handle a wide range of nonlinear demand functions.

Thesis Supervisor: Retsef Levi
Title: Robert N. Noyce Career Development Professor
Assistant Professor of Management
Thesis Supervisor: Georgia Perakis
Title: Associate Professor of Operations Research
Acknowledgments

With the deepest gratitude I would like to thank my supervisors Professor Retsef Levi and Professor Georgia Perakis for their insightful inspiration and continuous guidance along the way. Without their dedicated and prompt help, it would have been impossible for me to complete this thesis. They have enlightened me and set great examples for me in many ways.

I am really grateful for Brian Lim for his generous help on the use of AMPL and support whenever I encountered problems in research. I would also like to thank my friends in the CDO program for the ups and downs we have been through together.

I want to express my gratitude to my family. It is their unconditional love that keeps me going all the way.

I dedicate this thesis to them.
# Contents

1 Introduction
   1.1 Motivations and Contributions .................................. 13
   1.2 Literature Review .................................................. 16
   1.3 Thesis Outline ...................................................... 21

2 Model Description .................................................. 23
   2.1 Problem Formulation and Assumptions ............................. 23
   2.2 Deterministic Multiproduct Multiperiod Demand Functions ....... 31
      2.2.1 Linear Demand Function ...................................... 31
      2.2.2 Log-Linear (Exponential) Demand ............................ 34
      2.2.3 Constant-Elasticity Demand .................................. 37

3 Algorithm ............................................................. 41
   3.1 Gradient-based Methods .......................................... 41
      3.1.1 Unconstrained Optimization .................................. 42
      3.1.2 Constrained Optimization .................................... 44
   3.2 Algorithm Description and Discussion ............................ 45
   3.3 Procedures and Flowchart ........................................ 49

4 Simulations and Discussions ...................................... 53
   4.1 Motivations and Computer Treatment .............................. 54
      4.1.1 Motivations and Framework .................................. 54
      4.1.2 Computational Treatment .................................... 56
4.2 Linear Demand Cases ........................................ 57
  4.2.1 Instance Generation ........................................ 58
  4.2.2 Lipschitz Constant ........................................ 59
  4.2.3 Performance for Various Scales, Starting points, and Levels of Dominance ........................................ 59
  4.2.4 Performance for Different Capacity Constraints ........ 62
  4.2.5 Performance for Small Number of Time Periods ........ 64
4.3 Nonlinear Demand (Exponential) Cases ....................... 65
  4.3.1 Instance Generation ........................................ 66
  4.3.2 Lipschitz Constant ........................................ 68
  4.3.3 Performance for Various Scales, Starting points, and Levels of Dominance ........................................ 68
4.4 Inter-period Linear Demand Cases ............................ 73
  4.4.1 Demand Models Based on Past and Current Prices .... 73
  4.4.2 Demand Models Based on Past, Current, and Future Prices 76
  4.4.3 Lipschitz Continuity ....................................... 77
  4.4.4 Conclusions ............................................... 79

5 Conclusions .......................................................... 81

A Tables ................................................................. 85
List of Figures

3-1 Flowchart ................................................. 51

4-1 Comparison of the time required by different approaches .......... 60
4-2 Comparison of the time required by different approaches .......... 61
4-3 Comparison of the time required by different approaches .......... 62
4-4 Comparison of the time required by different approaches .......... 62
4-5 Comparison of the time required by different approaches for low capacity 63
4-6 Comparison of the time required by different approaches for medium capacity ........................................ 64
4-7 Comparison of the time required by different approaches for high capacity 64
4-8 Comparison of the time required by different approaches for small number of time periods ........................................ 65
4-9 Comparison of the time required by different approaches for nonlinear demand function ........................................ 69
4-10 Comparison of the time required by different approaches for nonlinear demand function ....................................... 70
4-11 Comparison of the time required by different approaches for nonlinear demand function ........................................ 70
4-12 Comparison of the time required by different approaches for nonlinear demand function ........................................ 71
4-13 Comparison of the time required by different approaches for nonlinear demand function ........................................ 72
4-14 Comparison of the time required by different approaches for nonlinear demand function ........................................... 72
4-15 Comparison of the time required by different approaches with relations between past and current periods ........................................... 78
4-16 Comparison of the time required by different approaches with relations among periods ........................................... 78
List of Tables

A.1 A sample block of matrix $B^{-1}$ for nonlinear demand case .......................... 86
A.2 A sample of vector $a$ for nonlinear demand case ............................................. 86
A.3 A sample block $B_1$ of matrix $B^{-1}$ for inter-period linear demand case 86
A.4 A sample block $A_1$ of matrix $B^{-1}$ for inter-period linear demand case 87
A.5 Original model with linear demand-price relationship using CPLEX .................. 87
A.6 Original model with linear demand-price relationship using LOQO ................. 87
A.7 Optimal results for linear demand with small initial points through iterative gradient-based algorithm .................................................. 88
A.8 Optimal results for linear demand with medium initial points through iterative gradient-based algorithm .................................................. 88
A.9 Optimal results for linear demand with large initial points through iterative gradient-based algorithm .................................................. 88
A.10 Original model with inter-temporal linear demand-price relationship using CPLEX ................................................................. 89
A.11 Optimal results for inter-temporal linear demand with small initial points through iterative gradient-based algorithm ............................ 89
A.12 Optimal results for inter-temporal linear demand with medium initial points through iterative gradient-based algorithm .................... 89
A.13 Optimal results for inter-temporal linear demand with large initial points through iterative gradient-based algorithm .................... 90
A.14 Optimal results for nonlinear demand with small initial points through iterative gradient-based algorithm ........................................ 90
A.15 Optimal results for nonlinear demand with medium initial points through iterative gradient-based algorithm .................. 90
A.16 Optimal results for nonlinear demand with large initial points through iterative gradient-based algorithm .................. 91
A.17 Solving time (Seconds) for linear demand with small number of time periods through iterative gradient-based algorithm .............. 91
Chapter 1

Introduction

1.1 Motivations and Contributions

Fluctuations in demand arise from a variety of factors including promotions and other pricing strategies as well as customer needs and preferences that change over time. These fluctuations in demand make it difficult to plan production efficiently. This is further enhanced by the fact that production and marketing planning are segregated in their respective company divisions. These problems can be of very large scale given the large number of products sold by the various company divisions as well as the potentially long time horizon. A study [6] by *McKinsey Quarterly* discusses how companies should respond to the rising complexity in today’s marketing environment using a common system with pricing across brands, channels, and segments. The pursuit of higher levels of efficiency through coordination has led to a growing interest in joint decision modeling of inventory control and pricing. Traditional models based on Economic Order Quantity (EOQ) theory that are modeled using constant rate exogenous demand are inadequate for industries such as airline, hotel, and car rental. These industries are unable to lock in their product demands over long periods or experience seasonality and variability in demands over time. Hence, there is a strong motivation to model joint pricing and inventory problems as optimal control problems when the demand is dynamic.
There is an extensive literature on how to solve joint pricing and inventory control problems. However, as the discussion in the next section illustrates, most of the existing computational work focuses on relatively simple and small instances. If the demand is a linear function of prices, a common approach would be to formulate the problem as a quadratic optimization problem with linear constraints. This can be handled using commercial software such as CPLEX, which solves large-scale linear and quadratic optimization problems very efficiently. However, modeling demand as a linear function of prices may not be realistic in many scenarios. If the demand is a nonlinear function of the prices, then under some conditions, the problem becomes a general concave maximization problem which is not quadratic and cannot be solved using CPLEX. Alternative approaches involve using general purpose concave maximization algorithms such as interior point and Newton-based methods. Nevertheless, these methods involve Hessian computations and can be numerically unstable and relatively slow, especially for large scale instances.

Another challenge lies in how to model and solve multi-period problems. The goal of modeling is to illustrate the price-demand relation in terms of how past prices can affect the current demand. Most models assume no inter-period relation. Few assume relation among periods but no dependence between different products. However, in reality, the demand can be affected by both past prices, for example, through a reference price formation mechanisms as in [16], and future prices based on consumer expectations of price changes. Furthermore, prices of related goods, substitutes or complement, also influence how much of a product people buy.

Our objective is thus to propose an efficient solution method that is capable of solving more practical large scale joint pricing and inventory control problems. In our approach, we consider a setting where multiple products share capacitated production capabilities. The presence of a joint capacity poses another challenge. We assume that the inventory and the production costs are linear functions of the inventory and the production levels, respectively. Demand for a given product at a specific time
period is either a linear or a nonlinear function that depends on past and current prices for all products. No back-orders are allowed, i.e., all demands must be satisfied on time. This constraint can be easily relaxed to allow linear backlogging costs. The demand price relation is modeled through an invertible function, which allows us to reformulate the revenue as a function of the demand levels rather than the prices. It effectively enables us to work in the demand space instead of using prices as decision variables. This transformation of the problem is quite common and can be done in many models under relatively mild conditions. We shall discuss the specific conditions in the subsequent chapters. Note that the objective function may not even be differentiable (see the discussions in Chapter 3).

By observing that if the demand levels are set, the optimal production problem becomes a transportation problem that can be solved via linear programming. We devise an iterative gradient-based algorithm. The objective function is rewritten as the revenue part minus the cost part, which is the optimal transportation cost given the demand levels. This reformulation helps employing a gradient-based approach by efficiently computing a subgradient in each iteration of the algorithm. Specifically, the gradient of the revenue part of the objective is computed directly and the subgradient of the cost part is computed via the dual solution of the transportation linear programming problem. The procedure is repeated iteratively.

From iteration to iteration, the linear transportation subproblem differs only in terms of the right-hand side (that is, in terms of the demand values). This observation allows us to re-optimize the cost efficiently from one iteration to another starting with the previous optimal dual solution. Since an initial dual feasible solution is available, we use the dual simplex method whereby dual feasibility is maintained while we drive toward primal feasibility to obtain the new optimal solution. In particular, considering the special structure of this transportation problem, the dual network simplex method is applied. Chapter 3 will explain this method in detail.
Since the instantaneous revenue can be a nonlinear function of the prices (or equivalently of the demand), our goal is to devise efficient algorithms that can handle a nonlinear (and potentially non-differentiable) objective function. The performance of our algorithm is compared with that of the original formulation (see 2.1 in Chapter 2) solved directly through LOQO (see Chapter 4 for more details). Instead of solving a general concave maximization problem, in each iteration, we need to solve only a linear optimization problem and subsequently a quadratic problem. In addition, over the multi-period horizon, we consider the price-demand relationships among periods.

We illustrate the efficiency of our algorithm via several computational experiments for linear demand functions, with scales up to 60 products within 60 periods as 3600 decision variables and 18120 constraints, where the current demand depends on past and current prices, with the potential to include expectations of future prices. For linear demand models, our performance is comparable to CPLEX in many instances.

1.2 Literature Review

There is a growing research literature on joint pricing and inventory control problems. This section focuses on deterministic demand models. Pekelman [15], Feichtinger and Hartl [8], Thompson, Sethi and Teng [15], Gilbert [9], Kunreuther and Schrage [12], Van den Heuvel and Wagelmans [20], Thomas [18], Deng and Yano [7], and Lee [13] consider the problem of determining simultaneously the price and production schedule of a single product over a finite time horizon. Popescu and Wu [16], Ahn et al. [2] extend the boundary of price-demand dependency from a single time period to multiple time periods. Gilbert [10] and Adida and Perakis [1] address the problem of jointly determining prices and production schedules for multiple products.

Pekelman [15] considers the problem of determining simultaneously the price and production schedule of a single product over a finite time horizon with no backorder. He models the demand as a linear function of the current period price, while inventory
costs are linear and production costs are strictly convex. He solves this problem by using control theory. In particular, he uses the necessary and sufficient conditions for a state constraint in the form of a nonnegative constraint on the inventory level.

Feichtinger and Hartl [8] extend [15] by considering general nonlinear demand functions and backorders, with both piecewise linear and strictly convex inventory costs for a single product. The optimal solution paths are derived by using optimal control theory. The treatment of linear non-smooth cost functions requires the use of the generalized maximum principle. The authors show that the equilibrium is approached within finite time for non-smooth piecewise linear cost function.

Another extension based on [15] is introduced by Thompson, Sethi and Teng in [19], which treats both linear and non-linear (strictly convex) production cost cases. Upper and lower bounds are imposed on state and control variables. The authors solve the problem by using the Lagrangian form of the maximum principle. The production rate and the level of inventory of a single product are bounded. They decompose the whole problem into a set of smaller problems, which can be solved separately, and their solutions combine to form a complete solution to the problem. They derive a forward branch and bound algorithm to solve the problem and illustrate through a simple example with ten time periods.

Under the assumption that the firm must commit to a single price for the entire planning horizon, Kunreuther and Schrage [12] develop an algorithm for determining the pricing and ordering decision for a single product with a deterministic demand curve that differs from period to period. Their approach does not guarantee optimality but does provide upper and lower bounds on the optimal price. They propose a heuristic algorithm to solve this problem.

Van den Heuvel and Wagelmans [20] thereafter extend [12] and derive an exact algorithm and show that their algorithm boils down to solving a number of lot-sizing
problems that is quadratic in the number of periods, i.e., the problem can be solved in polynomial time.

Gilbert [9] follows upon [12] and addresses the problem of jointly determining a single price and production schedule for a product with seasonal demand. Using a slightly less general model of demand and requiring that the setup and holding costs be time-invariant, he develops a dynamic programming approach that guarantees optimality by enumerating the number of possible setups and maximizing the profit for that fixed number of setups. The author illustrates this algorithm through a simple linear inverse demand function and considers six time periods. The model guarantees an optimal solution in $O(T^3)$ time.

In [18], Thomas provides a forward algorithm for determining the optimal pricing and production decisions of a single product over a discrete time horizon where demand is a deterministic function of the price and backorders are not allowed. In other words, all demands must be satisfied on time.

Deng and Yano [7] consider the problem of setting prices and choosing production quantities for a single product over a finite horizon for a capacity-constrained manufacturer facing price-sensitive demands. They show that, counter-intuitively, optimal prices may increase as the capacity increases, even when capacity is constant over the horizon. The optimal solution consists of a sequence of regeneration intervals (RI), i.e., a set of consecutive time periods where the inventory level starts and ends at zero and is positive in intermediate time periods. They use a shortest-path method to determine the best sequence of RI’s over the time horizon.

In [13], Lee presents the use of geometric programming (GP) to solve a profit maximizing problem to determine the optimal selling price and order quantity for a single product. Demand is modeled as a nonlinear function of price with constant elasticity $D = kp^{-\alpha}, \alpha > 1$. He then makes use of a posynomial transformation to solve
the problem to optimality for both no-quantity discounts and continuous quantity discount cases. By applying readily available theories in GP, he solves the problem when the profit function is non-concave and obtains lower and upper bounds on the optimal profit level and sensitivity results, which are further utilized to provide additional managerial implications for pricing and lot-sizing policies. However, it would be a challenging task to extend this model to the multi-product case where the non-linear interactions among related products with respect to their demands are taken into account. In this case, multiple revenue terms in the profit (signomial) function would make such posynomial transformation impossible.

In [16], Popescu and Wu consider the dynamic problem of a monopolist firm in a market with repeated interactions, where demand is sensitive to the firm's pricing history. Consumers have memory and are prone to human decision-making biases and cognitive limitations. As the firm manipulates prices, consumers form a reference price that adjusts as an anchoring standard based on price perceptions. Purchase decisions are made by assessing prices as discounts or surcharges relative to the reference price, in the spirit of prospect theory. They prove that optimal pricing policies induce a perception of monotonic prices, whereby consumers always perceive a discount or surcharge relative to their expectation.

In [2], Ahn et al. extend the boundary of price-demand dependency from a single time period to multiple time periods. They consider a single-product, discrete-time, capacitated joint production and pricing problem, accounting for that portion of demand realized in each period that is induced by the interaction of pricing decisions in the current period and in previous periods. They divide demand in each period into current demand that is generated by customers who enter the system at that period and residual demand for customers who entered the system in previous periods but have not yet made a purchase.

The papers we discussed above all focus on the single product case, while our work
considers a multi-product with capacitated production capabilities. In [17], Talluri and Van Ryzin present a general review of common multiproduct-demand functions under the assumption that the revenue function is concave. We expand this discussion and derive specific conditions to ensure concavity of the revenue for various demand models in Section 2.3 of this thesis.

In [10], Gilbert addresses a similar problem of jointly determining prices and production schedules for a set of products that share a production capacity. However, he assumes that the demand is seasonal and dependent on the prices that the firm commits for the entire planning horizon. Hence, each demand decision level variable, $D_j$, has a seasonal coefficient $\beta_{jt}$ such that the demand of each product $j$ at time $t$ is $\beta_{jt}D_j$. The algorithm proposed decomposes the problem into two subproblems for each iteration, a linear programming (LP) subproblem and a nonlinear programming (NLP) subproblem. In the LP subproblem, one determines the production and inventory levels for given demand values with the goal to minimize the total cost over the horizon for the particular demand level. In the NLP subproblem on the other hand, one determines the demand at each period for each product to maximize the total profit. The model presented is similar to the model and solution approach in this thesis. However, in this thesis, demands are not subjected to seasonality effects. Moreover, we describe an efficient solution approach to solve the nonlinear demand function cases.

In [1], Adida and Perakis generalize Pekelman’s work [15] to multiple products. They introduce a continuous time optimal control model for a dynamic pricing and inventory management problem with no backorders. Their model assumes that demand is linear and dependent only on the product’s own price and the multi-product production facility is capacitated in continuous time. The solution approach consists of using a heuristic algorithm that computes the adjoint variable and dual multipliers iteratively to obtain the optimal solution. Their model is more general in terms of utilizing continuous time instead of discrete time and assuming strictly convex
production costs instead of linear production costs. Key differences in the model presented in this thesis is the use of a general demand function whereas [1] is restricted to linear demand. The algorithm proposed here is provably optimal and is not a heuristic.

1.3 Thesis Outline

Chapter 2 defines a set of assumptions and definitions which will be used throughout this thesis. This chapter analyzed and discussed properties of linear and nonlinear demand functions. With all the basics in mind, we formulate the original model addressing this joint multi-period pricing and inventory control problem. Considering the special structure of the model, we further reformulate it via a linear transportation problem.

Chapter 3 presents a detailed discussion of this gradient-based algorithm. A flowchart illustrates its iterative process.

Chapter 4 presents simulations and discussions for three sets of deterministic demand models of pricing for multiple products. The first case studies the situation when the demand is a linear function and the performance of our algorithm is compared with that of the original formulation (see 2.1 in Chapter 2) solved through LOQO and CPLEX. The second case discusses the nonlinear demand model and the original formulation becomes a non-quadratic problem. The performance of our algorithm is compared with that of the original formulation solved through LOQO. The third case deals with obtaining the optimal solution to the joint problem when each product’s demand is a linear function of current prices with an extension to include past prices and expectations of future prices.

Finally, Chapter 5 concludes this thesis with a summary of the proposed algorithm
and its contributions and with suggestions for future research directions.
Chapter 2

Model Description

This chapter describes a multi-period joint pricing and inventory control problem. We consider a setting where multiple products share capacitated production capabilities that are potentially different in each period. Section 2.1 introduces the original model and explain our reformulation as a linear transportation problem when the demand levels are set. Section 2.2 discusses several widely-used linear and nonlinear demand models.

2.1 Problem Formulation and Assumptions

We consider a finite horizon model, with T discrete time periods. Furthermore, there are N products that share the same capacitated production capabilities. For any demand vector $D$ and price vector $P \in \mathbb{R}^{NT}_+$, we use $d_{it}$ and $p_{it}$ respectively to indicate their components for product $i$ at time period $t$, $i = 1, 2, \ldots, N$ and $t = 1, 2, \ldots, T$. We write $D = (d_{11}, \ldots, d_{N1}, \ldots, d_{1T}, \ldots, d_{NT})$ and $P = (p_{11}, \ldots, p_{N1}, \ldots, p_{1T}, \ldots, p_{NT})$, respectively. For each $i$ and $t$, let $D_i^t(P)$ be the demand of item $i$ in period $t$ as a function of price $P$. Let $u^t_i$ denote the production decision on how many units of product $i$ are produced in period $t$. Shared production capacity, production per unit production cost, and per unit holding cost can vary from period to period, and are denoted by $C_t$, $c_i$, and $h_i^t$, respectively.
In this thesis, we study a joint multi-period pricing and inventory control problem for a make-to-stock manufacturing system. The objective is to maximize the total profit, which is the difference between the total revenue over the time horizon for all products (i.e., the price multiplied by demand for every product and every period) and the costs of production and inventory over the planning horizon for all products, i.e.,

\[
\text{Profit} = \text{Revenue} - \text{Costs (Production and Inventory)}
\]

We assume that the inventory and the production costs are linear functions of the inventory and the production levels, respectively. The demand for a product at a time period is either a linear or a nonlinear function that depends on past and current prices for all products. No back-orders are allowed, i.e., all demands must be satisfied on time. This constraint can be easily relaxed to allow linear backlogging costs.

Throughout this thesis, the inner product of vectors \(x\) and \(y\) is defined as \(x^Ty = \sum_i x_i y_i\). Unless otherwise stated, we use \(||x||\) to denote the L-2 norm, which is \(||x|| = \sqrt{x^T x}\). The L-1 norm is also used occasionally, i.e., \(||x||_1 = \sum_i |x_i|\).

Using the notations we defined above, the proposed problem can be formulated as a nonlinear optimization problem as follows. Note that the decision variable in this model is the price vector \(P\).
\[
\begin{align*}
\text{max} & \quad \sum_{i=1}^{N} \sum_{t=1}^{T} \left( D_i^t(P) p_{it} - c_i^t u_i^t - h_i^t I_i^t \right) \\
\text{s.t.} & \quad I_i^t = I_i^{t-1} + u_i^t - D_i^t(P), \quad \forall t = 1, \ldots, T, \quad i = 1, \ldots, N \\
& \quad I_i^t \geq 0, \quad \forall t = 1, \ldots, T, \quad i = 1, \ldots, N \\
& \quad D_i^t(P) \geq 0, \quad \forall t = 1, \ldots, T, \quad i = 1, \ldots, N \\
& \quad p_{it} \geq 0, \quad \forall t = 1, \ldots, T, \quad i = 1, \ldots, N \\
& \quad u_i^t \geq 0, \quad \forall t = 1, \ldots, T, \quad i = 1, \ldots, N \\
& \quad \sum_{i=1}^{N} u_i^t \leq C_t, \quad \forall t = 1, \ldots, T \\
& \quad I_i(0) = I_i^0, \quad i = 1, \ldots, N.
\end{align*}
\]

Constraint (2.2) guarantees the conservation of inventory. It describes the inventory level for each product \(i\), at the end of period \(t\) as the inventory remaining from the previous period \((t - 1)\) for product \(i\), in addition to the production for that product in period \(t\) and subtracting the demand for that product in period \(t\) (i.e. how much of product \(i\) was sold in period \(t\)). Constraint (2.3) indicates the non-negativity of inventory level which prohibits backorders. Constraints (2.4), (2.5), and (2.6) ensure that the demands, prices, and production quantities are non-negative, respectively. Constraint (2.7) guarantees that the total production for all products does not exceed the shared capacity in that time period. More generally, with this framework, we can incorporate other constraints on the demands and prices.

We make an assumption on the invertibility and general price-demand relationship as follows, which is very common in practice (see [10]).

**Assumption 1**

- The demand function \(D(P) : \mathbb{R}_+^{NT} \mapsto \mathbb{R}_+^{NT}\) where \(D(P) = (d_{11}(P), \ldots, d_{NT}(P))\)
is invertible. That is \( P(D) : \mathbb{R}_+^{NT} \rightarrow \mathbb{R}_+^{NT} \) where \( P(D) = (p_{i1}(D), \ldots, p_{iN}(D)) \) is well defined.

- For product \( i \) at time \( t \), we will assume that the price function \( P_i^t(D) \) is a strictly decreasing function of \( d_{it} \) on \( \mathbb{R}^+ \) where all other demand levels are fixed.

The assumption on the inevitability of \( D \) allows us to rewrite the joint pricing and inventory control problem as follows.

\[
\begin{align*}
\text{max} & \sum_{i=1}^N \sum_{t=1}^T \left( P_i^t(D) d_{it} - c_i^t u_i^t - h_i^t I_i^t \right) \\
\text{s.t.} & I_i^t = I_i^{t-1} + u_i^t - d_{it}, \ \forall t = 1, \ldots, T, \ i = 1, \ldots, N \\
& I_i^t \geq 0, \ \forall t = 1, \ldots, T, \ i = 1, \ldots, N \\
& P_i^t(D) \geq 0, \ \forall t = 1, \ldots, T, \ i = 1, \ldots, N \\
& d_{it} \geq 0, \ \forall t = 1, \ldots, T, \ i = 1, \ldots, N \\
& u_i^t \geq 0, \ \forall t = 1, \ldots, T, \ i = 1, \ldots, N \\
& \sum_{i=1}^N u_i^t \leq C_i, \ \forall t = 1, \ldots, T \\
& I_i(0) = I_i^0, \ i = 1, \ldots, N.
\end{align*}
\]

This effectively transforms the decision variable from the price space to the demand space. Now observe that if the demand levels are set, then the cost can be minimized by solving a linear program. Moreover, we can reformulate this as a transportation cost minimization problem \( g(D) \) which can be solved efficiently (see [5] and [3]).

Our objective is to transport the goods from the supply nodes to the demand nodes at minimum cost, which is known as a network flow problem. We assume no initial inventory, i.e., \( I_i(0) = 0 \). In this network, we define

- Supply node \( s \) for each period \( s \) with supply flow \( C_s \), where \( s = 1, \ldots, T \).
Demand node \((i, t)\) for each demand point with demand flow \(d_{it}\), where \(i = 1, \ldots, N\) and \(t = s, \ldots, T\).

Arc \((s, i, t)\) from supply nodes \(s\) to demand nodes \((i, t)\) with cost \(b_{st}^i\) for every \(s \leq t\), i.e., \(i = 1, \ldots, N\), \(s = 1, \ldots, T\), and \(t = s, \ldots, T\).

Therefore,

\[
g(D) = \min_x \quad b^\prime x \\
\quad \text{such that} \quad \sum_{i,t: \ s \leq t \leq T} x_{st}^i \leq C_s, \ s = 1, \ldots, T \\
\quad \sum_{s=1}^t x_{st}^i \geq d_{it}, \ t = 1, \ldots, T, \ i = 1, \ldots, N \\
\quad x_{st}^i \geq 0, \ s = 1, \ldots, T, \ t = s, \ldots, T, \ i = 1, \ldots, N.
\]

At each time period \(s = 1, \ldots, T\), we supply the quantity of \(C_s\), equal to the shared production capacity in that period, to a supply node of that period. Each supply node \(s\) is also connected to demand nodes \((i, t)\), where \(i = 1, \ldots, N\) and \(t = s, \ldots, T\), with demand \(d_{it}\). The arc flow, denoted by \(x_{st}^i\), from supply node \(s\) to demand node \((i, t)\), is the amount of production at time \(s\) used to satisfy demand for product \(i\) at time \(t = s, \ldots, T\). The arc cost per unit of flow is \(b_{st}^i = c_s^i + \sum_{t \leq v < t} h_i^v\), for each supply node. Furthermore, we assign \(N\) additional supply nodes with supply equal to the initial inventory \(I_i^0\), for each product \(i = 1, \ldots, N\), and the cost \(b_{st}^i = \sum_{t \leq v < t} h_i^v\).

In the formulation of \(g(D)\), \(x_{st}^i\) and \(b_{st}^i\) represent the production quantity and the cost respectively, corresponding to a specific product \(i\), produced at time \(s\) and sold at time \(t\). Incorporating the non-negativity constraint, \(D\) is the set of feasible demand vectors \(D\) for this linear transportation problem. That is all the vector \(D\) such that there exists a feasible solution to the transportation problem.

This gives rise to the reformulation of the joint pricing and inventory control problem.
as

\[
\max_{D \in D \subseteq \mathbb{R}^{NT+}} \tau(D) = P(D)'D - g(D).
\]

\[D = \{D|d_{it} \geq 0; \sum_{i=1}^{N} \sum_{t=1}^{r} d_{it} \leq \sum_{s=1}^{r} C_{s}; r = 1, \ldots, T\}.
\]

For the remainder of this thesis, our goal is to solve this problem, \(\tau(D)\), efficiently and accurately.

Next we discuss several assumptions that guarantee the tractability of \(g(D)\) and \(r(D)\).

**Proposition 1** \(g(D)\) is a piecewise linear convex function on \(\mathbb{R}^{NT+}\).

**Proof:** For any feasible \(D\), the optimum of \(g(D)\), by strong duality, is equal to the optimal value of its dual objective. Consider the dual problem of \(g(D)\) as below

\[
\max_{y,z} \sum_{i,t} d_{it} z_{it} - \sum_{s} C_{s} y_{s}
\]

such that

\[
z_{it} - y_{s} \leq b_{it}^{s}, \quad s = 1, \ldots, T, \quad t = s, \ldots, T, \quad i = 1, \ldots, N
\]

\[y_{s} \geq 0, \quad s = 1, \ldots, T\]

\[z_{it} \geq 0, \quad t = 1, \ldots, T, \quad i = 1, \ldots, N.
\]

from Corollary 2.1 in [5], we know that for the dual problem \(g(D)\) with a finite number of linear inequality constraints, we have a finite number of basic feasible solutions of \((z_{it}^{j}, y_{s}^{j})\), \(j = 1, \ldots, m\), where \(m\) is the number of basic feasible solutions.

Since the optimum of the dual must be attained at an extreme point, \(g(D)\) is equivalent to the maximum of a finite collection of linear functions over all the basic feasible solutions \((z_{it}, y_{s})\):

\[
g(D) = \max_{j=1, \ldots, m} \sum_{i,t} d_{it} z_{it}^{j} - \sum_{s} C_{s} y_{s}^{j}.
\]
It is therefore a piecewise linear function. This follows from sensitivity analysis with respect to the right hand side in linear optimization. Q.E.D.

Now we proceed to make the concavity assumption.

**Definition 1** The Hessian matrix of a twice differentiable function \( f(x_1, x_2, \ldots, x_n) \) is an \( n \times n \) matrix whose \( ij^{th} \) entry is

\[
\frac{\partial^2 f}{\partial x_i \partial x_j}(x_1, \ldots, x_n).
\]

**Definition 2** Let \( F \) be a concave function defined on a convex set \( S \). Let \( b^* \) be an element of \( S \). We say that a vector \( p \) is a subgradient of \( F \) at \( b^* \) if

\[
F(b^*) + p'(b - b^*) \geq F(b), \quad \forall b \in S.
\]

**Definition 3** Consider a function \( f: S \subseteq \mathbb{R}^n \rightarrow \mathbb{R} \). Then \( f \) is a strictly concave function if for all \( a, b \in S \) with \( a \neq b \), for all \( \lambda \in (0, 1) \), the following inequality holds,

\[
f[(1 - \lambda)a + \lambda b] < (1 - \lambda)f(a) + \lambda f(b). \quad (2.8)
\]

The function is concave if we also allow equality in the above expression.

**Proposition 2** Letting \( f \) be a function of multiple variables with continuous partial derivatives of first and second order on the convex open set \( S \) and denoting the Hessian of \( f \) at the point \( x \) by \( H(x) \). Then

1. \( f \) is concave if and only if \( H(x) \) is negative semidefinite for all \( x \in S \). If \( H(x) \) is negative definite for all \( x \in S \) then \( f \) is strictly concave.

2. \( f \) is convex if and only if \( H(x) \) is positive semidefinite for all \( x \in S \). If \( H(x) \) is positive definite for all \( x \in S \) then \( f \) is strictly convex.
Therefore, for a function to be strictly concave, its Hessian matrix has to be negative definite.

**Definition 4** Let matrix $B = [\beta_{ij}]$. The matrix $B$ is said to be strictly diagonally dominant if $|\beta_{ii}| > \sum_{j \neq i} |\beta_{ij}|$.

To ensure that the function is concave, one sufficient condition is that its Hessian matrix satisfies the strictly diagonally dominant condition. This condition also guarantees that the Hessian matrix is invertible (See [11]).

**Definition 5** A twice differentiable function $G(D)$ is strongly concave if the minimum over all nonnegative vectors $D$ of the minimum eigenvalue of minus the Hessian matrix of $G(D)$ is strictly positive (i.e., $\min_D \lambda_{\min}(-H(G(D))) \equiv \lambda > 0$, where $H(G)$ denotes the Hessian matrix of function $G$).

In this thesis, function $G$ corresponds to the total revenue $P(D)'D$ for all products over the entire time horizon. Since $G(D)$ is concave, this implies that the profit function is strongly concave. Notice that alternatively, if the price function $P(D)$ is not differentiable, the definition above follows if we require that for any two demand vectors $D$ and $\tilde{D}$,

$$(-\text{subgradient}\{P(D)'D\} + \text{subgradient}\{P(\tilde{D})'\tilde{D}\})'(D - \tilde{D})$$

$$\geq \lambda \|D - \tilde{D}\|^2, \text{ for some } \lambda > 0.$$  

This property is also referred to as strong monotonicity of function $-\text{subgradient}\{P(D)'D\}$.

**Definition 6** A function $G(D)$ is Lipschitz continuous with constant $L_1 > 0$ if

$$\|G(D) - G(\tilde{D})\| \leq L_1 \|D - \tilde{D}\|, \quad \forall D, \tilde{D}.$$
Assumption 2  The function $P(D)'D$ that describes the total revenue for all products over the time horizon, $P(D)'D$, is:
1. a strongly concave function in $D \subseteq \mathbb{R}^{NT}$.
2. A subgradient of $-P(D)'D$ is Lipschitz continuous with constant $L_1 > 0$.

2.2  Deterministic Multiproduct Multiperiod Demand Functions

In the case where there are multiple products, the demand function of a product can depend on its current or past price as well as the prices of its substitutes or complementary products. Here, we discuss some of the most common demand functions and the specific conditions required for them to satisfy Assumption 1 and Assumption 2. Note that in this section, we denote the demand and price vectors as $D = (d_i,...,d_n)$ and $P = (p_1,...,p_n)$, $i = 1,...,n$, $n = N \times T$, where $N$ is the number of products and $T$ is the number of time periods.

2.2.1  Linear Demand Function

The linear demand model is popular because of its simple functional form and the parameters characterizing the linear price demand relation can be estimated through past data using linear regression. In the multiproduct case, the demand function is:

$$D(P) = a - BP.$$ 

where $a = (a_1,a_2,...,a_n)$ is a vector of coefficients, $a \geq 0$. The value of $a$ tells us the demand when the price is equal to zero.
is a matrix of price sensitivity coefficients with $\beta_{ii} \geq 0$ for all $i = 1, \ldots, n$ and the sign of $\beta_{ij}, i \neq j$ depends on whether the products are complements ($\beta_{ij} \leq 0$), helping complete one another in some way, or substitutes ($\beta_{ij} \geq 0$), satisfying the same set of goals or preferences.

The revenue function, as the product of price and demand, can be written as follows:

$$f(P) = P^T D(P) = \sum_{i=1}^{n} P_i (a_i - \beta_{ii} P_i + \sum_{j \neq i} \beta_{ij} P_j).$$

The Hessian matrix of the revenue function with respect to price $P$ is

$$H[P] = \frac{\partial^2 f}{\partial p_i \partial p_j} = -2 \begin{bmatrix} \beta_{11} & -\beta_{12} & \cdots & -\beta_{1n} \\ -\beta_{21} & \beta_{22} & \cdots & -\beta_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\beta_{n1} & -\beta_{n2} & \cdots & \beta_{nn} \end{bmatrix} = -2B.$$

To ensure that the revenue function is concave, one sufficient condition is that the row coefficients satisfy the strictly diagonally dominant condition

$$|H[P]_{ii}| > \sum_{j \neq i} |H[P]_{ij}|, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n.$$ 

This condition guarantees that $H$ is invertible as well (See [11]).

Similarly, if $B$ is strictly diagonal dominant and hence nonsingular, the inverse-
demand function exists and is given by

\[ P(D) = B^{-1}(a - D), \]

where \( B^{-1} = [b_{ij}] \).

This transformation enables us to work in the demand space instead of the price space.

\[ f(D) = P(D)^T D = B^{-1}(a - D) \cdot D = \sum_{i=1}^{n} \left( \sum_{j=1}^{n} b_{ij}(a_j - d_j) \right) d_i. \]  

(2.9)

(2.10)

(2.11)

The gradient of \( f \) can be written as

\[ \nabla f(D) = \left( \sum_{i=1}^{n} b_{1i}(a_i - d_i) - \sum_{j=1}^{n} b_{i1}d_j, \sum_{i=1}^{n} b_{2i}(a_i - d_i) - \sum_{j=1}^{n} b_{i2}d_j, \ldots, \sum_{i=1}^{n} b_{ni}(a_i - d_i) - \sum_{j=1}^{n} b_{jn}d_j \right). \]

where the \( m^{th} \) term is:

\[ [\nabla f(D)]_m = \sum_{i=1}^{n} b_{mi}(a_i - d_i) - \sum_{j=1}^{n} b_{jm}d_j. \]

The Hessian matrix of the revenue function with respect to demand \( D \) is

\[ H[D] = \frac{\partial^2 f}{\partial d_i \partial d_j} = \begin{bmatrix} -2b_{11} & -b_{12} & \cdots & -b_{1n} & -b_{n1} \\ -b_{12} & -2b_{22} & \cdots & -b_{2n} & -b_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ -b_{1n} & -b_{n1} & -b_{2n} & \cdots & -2b_{nn} \end{bmatrix} = -2B^{-1}. \]  

(2.12)

Refer to the Assumption 1 and Assumption 2, we need to ensure that the revenue function with respect to demand \( D \) is concave, one sufficient condition is
2.2.2 Log-Linear (Exponential) Demand

The log-linear, or exponential, demand function is popular in econometric studies (see [14]) and has several desirable theoretical and practical properties. Notice that the log-linear demand function is always non-negative and its logarithm is a linear function of the prices. Thus the log-linear model can be estimated easily from data using linear regression.

\[ D(P)_j = e^{a_j - B_j^T P}, \quad j = 1, \ldots, n. \]

where \( a_j \) is a scalar coefficient and \( B_j = [-\beta_{j1}, -\beta_{j2}, \ldots, \beta_{jj}, \ldots, -\beta_{jn}] \) is a vector of price sensitivity coefficients.

Let \( a = (a_1, \ldots, a_n) \) and

\[
B = \begin{bmatrix}
\beta_{11} & -\beta_{12} & \cdots & -\beta_{1n} \\
-\beta_{21} & \beta_{22} & \cdots & -\beta_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
-\beta_{n1} & -\beta_{n2} & \cdots & \beta_{nn}
\end{bmatrix}
\]

where \( \beta_{ij} \) is the price sensitivity coefficients with \( \beta_{ii} \geq 0 \) for all \( i \) and the sign of \( \beta_{ij}, i \neq j \) depends on whether the products are complements (\( \beta_{ij} \leq 0 \)) or substitutes (\( \beta_{ij} \geq 0 \)).

The revenue function can be calculated as:
\[ f(P) = P^T D(P) \]  
(2.13)  
\[ = \sum_{i=1}^{n} p_i \exp(a_i - \beta_{ii}p_i + \sum_{j \neq i} \beta_{ij}p_j). \]  
(2.14)  

The Hessian matrix \( H = [H_{ij}] \) with respect to price \( P \) can be written as

\[
H[P]_{ij} = \frac{\partial f}{\partial p_i \partial p_j} 
= \beta_{ij}(1 - p_i \beta_{ii}) \exp(a_i - \beta_{ii}p_i + \sum_{m \neq i} \beta_{im}P_m) 
+ \beta_{ij}(1 - p_j \beta_{jj}) \exp(a_j - \beta_{jj}p_j + \sum_{m \neq j} \beta_{im}P_m) 
+ \sum_{m \neq i, m \neq j} P_m \beta_{mi} \beta_{mj} \exp(a_m - \beta_{mm}P_m + \sum_{q \neq m} \beta_{mq}p_q). 
\]

As in the linear model, to ensure that the revenue function is concave, one sufficient condition is

\[ |H[P]_{ii}| > \sum_{j \neq i} |H[P]_{ij}|, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n, \]

which also guarantees that \( H[P] \) is invertible as well [11].

Taking the logarithm, we have

\[ \ln(D(P)) = \left( \ln(D(P), \ldots, \ln(D(P))) \right) \]

\[ = a - BP. \]
Similarly, if $B$ is strictly diagonal dominant and hence nonsingular, the inverse-demand function exists and is given by

$$P(D) = B^{-1}(a - ln(D)),$$

where $B^{-1} = [b_{ij}]$.

$$f(D) = P(D)^T D$$

$$= B^{-1}(a - ln(D)) \cdot D$$

$$= \sum_{i=1}^{n} \left( \sum_{j=1}^{n} b_{ij}(a_j - ln(d_j)) \right) d_i. \quad (2.16)$$

The $m^{th}$ term of $\nabla f$ is:

$$[\nabla f(D)]_m = \sum_{j=1}^{n} b_{mj}(a_j - ln(d_j)) - \sum_{j=1}^{n} b_{jm} \frac{d_j}{d_m}. \quad (2.17)$$

The Hessian matrix $H = [H_{ij}]$ with respect to demand $D$ can be written as

$$H[D]_{ij} = \frac{\partial^2 f}{\partial d_i \partial d_j} \quad (2.18)$$

$$= - \frac{b_{ij}}{d_j} - \frac{b_{ji}}{d_i}. \quad (2.19)$$

For Assumption 1 and Assumption 2 to hold, we need to ensure that the revenue function with respect to demand $D$ is concave, one sufficient condition is

$$|H[D]_{ii}| > \sum_{j \neq i} |H[D]_{ij}|, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n.$$

### 2.2.3 Constant-Elasticity Demand

Some goods have a constant-elasticity demand curve whose logarithm is linear with respect to the logarithms of the prices. This means that the change in quantity demanded divided by the change in price remains a constant over all price levels.
\[ d_i(P) = a_i p_1^{-\beta_{i1}} p_2^{-\beta_{i2}} \cdots p_n^{-\beta_{in}}, i = 1, \ldots, n. \]

where \( \mathbf{a} = (a_1, \ldots, a_n) \) and matrix \( \mathbf{B} = [\beta_{ij}] \) defines the cross and own price elasticities among the products.

The revenue function is:

\[
f(P) = \mathbf{P}^T \mathbf{D}(P)
\]

\[
= \sum_{i=1}^{n} a_i p_i^{-\beta_{ii} + 1} \prod_{m \neq i}^{n} p_m^{-\beta_{im}}.
\]

The Hessian matrix with respect to \( \mathbf{P} \) can be obtained as

\[
H(\mathbf{P})_{ij} = \frac{\partial^2 f}{\partial p_i \partial p_j}
\]

\[
= a_i (\beta_{ii} - 1) \beta_{ij} p_i^{-\beta_{ii}} p_j^{-\beta_{ij} - 1} \prod_{m \neq i, m \neq j}^{n} p_m^{-\beta_{im}}
\]

\[
+ a_j (\beta_{jj} - 1) \beta_{ij} p_i^{-\beta_{ij}} p_j^{-\beta_{ij} - 1} \prod_{m \neq i, m \neq j}^{n} p_m^{-\beta_{im}}
\]

\[
+ \sum_{m \neq i, m \neq j} a_m \beta_{im} \beta_{jm} p_i^{-\beta_{im} - 1} p_j^{-\beta_{jm} - 1} \prod_{q \neq i, q \neq j, q \neq m}^{n} p_q^{-\beta_{qm}}.
\]

Applying similar sufficient conditions as in the linear case, we require

\[
|H(\mathbf{P})_{ii}| > \sum_{j \neq i} |H(\mathbf{P})_{ij}|, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n
\]

to ensure the strictly concave property for the revenue function.

Similarly, if \( \mathbf{B} \) is strictly diagonal dominant and hence nonsingular, the inverse demand function exists and is given by
\[(\log(d_1(P)), ..., \log(d_n(P))) = a - BP,\]

where \(a = (a_1, ..., a_n), B = [\beta_{ij}]\).

The inverse demand function is:

\[P(D) = B^{-1}[a_1 - \log(d_1(P)), ..., a_n - \log(d_n(P))],\]

where \(B^{-1} = [b_{ij}]\).

\[f(D) = D^TP(D)\]
\[= B^{-1}(a - (\log(d_1(P)), ..., \log(d_n(P))))D\]
\[= \sum_{i=1}^{n} \left( \sum_{j=1}^{n} b_{ij}(a_j - \log(d_j)) \right) d_i.\]

The \(m^{th}\) term of \(\nabla(P(D)^TD)\) is:

\[\nabla f(D)\]_m = \[\sum_{i=1}^{n} b_{mi}(a_i - \log(d_i)) \] - \[\sum_{j=1}^{n} b_{jm}\frac{d_j}{d_i\ln 10}.\]

The Hessian matrix \(H = [H_{ij}]\) with respect to demand \(D\) can be written as

\[H[D]_{ij} = \frac{\partial f}{\partial d_i \partial d_j} = -\frac{b_{ij}}{d_i\ln 10} - \frac{b_{ji}}{d_j\ln 10}.\]

For Assumption 1 and Assumption 2 to hold, we need to ensure that the revenue function with respect to demand \(D\) is concave, one sufficient condition is
\[ |H[D]_{i\|} > \sum_{j \neq i} |H[D]_{ij}|, \quad i = 1, \ldots, n, \quad j = 1, \ldots, n. \]

Moreover, from the discussion of various demand functions, it can be noticed that the condition on concavity assumption is clearer and easier to check in the demand space than that in the price space.
Chapter 3

Algorithm

In Chapter 2, we illustrated that given a demand vector, maximizing profit for the joint multi-period pricing and inventory control problem is equivalent to minimizing the cost through a linear transportation problem. We use this to devise an iterative gradient-based algorithm which will be described in detail in this chapter. We start with a general description of gradient-based optimization methods.

3.1 Gradient-based Methods

Definition 7 Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a real-valued function such that the partial derivative $\frac{\partial f(x)}{\partial x_k}$ exists for $k = 1, 2, ..., n$. The gradient of $f(x)$ with respect to a vector variable $x = (x_1, ..., x_n)$, denoted by $\nabla f(x)$, is the vector

$$\nabla f(x) = \left( \frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2}, ..., \frac{\partial f(x)}{\partial x_n} \right).$$

Consider a general nonlinear programming problem as

$$\min \quad f(x)$$

such that $x \in S$.
where we assume throughout that:

- $S$ is a nonempty and convex subset of $\mathbb{R}^n$.
- The function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is continuously differentiable over $S$.

**Proposition 3** If $S$ is a convex subset of $\mathbb{R}^n$ and $f : S \to \mathbb{R}$ is a convex function, then a local minimum of $f$ over $S$ is a global minimum. If in addition $f$ is strictly convex, then there exists at most one global minimum of $f$ over $S$.

**Proof:** See Proposition 2.1.1 in [4].

### 3.1.1 Unconstrained Optimization

We start by considering the unconstrained case where $S = \mathbb{R}^n$:

Given a vector $x \in \mathbb{R}^n$ with $\nabla f(x) \neq 0$, consider the half line of vectors

$$x_\alpha = x - \alpha \nabla f(x), \quad \forall \alpha \geq 0.$$ 

From the first order Taylor series expansion around $x$, we can write

$$f(x_\alpha) = f(x) + \nabla f(x)'(x_\alpha - x) + o(\|x_\alpha - x\|)$$

$$= f(x) + \nabla f(x)'(x - \alpha \nabla f(x) - x) + o(\|x - \alpha \nabla f(x) - x\|)$$

$$= f(x) - \alpha \|\nabla f(x)\|^2 + o(\alpha \|\nabla f(x)\|),$$

that is

$$f(x_\alpha) = f(x) - \alpha \|\nabla f(x)\|^2 + o(\alpha).$$

The term $\alpha \|\nabla f(x)\|^2$ dominates $o(\alpha)$ for $\alpha$ near zero, so for positive but sufficiently small $\alpha$, $f(x_\alpha)$ is smaller than $f(x)$.

In general, consider that half line of vectors

$$x_\alpha = x + \alpha d, \quad \forall \alpha \geq 0,$$
where the direction vector \( d \in \mathbb{R}^n \) makes an angle with \( \nabla f(x) \) that is greater than 90 degrees, that is
\[
\nabla f(x)'d \leq 0.
\]

A first order Taylor expansion implies that
\[
f(x_\alpha) = f(x) + \alpha \nabla f(x)'d + o(\alpha).
\]

For a near zero, the term \( \alpha \nabla f(x)'d \) dominates \( o(\alpha) \). As a result, for positive but sufficiently small \( \alpha \), \( f(x + \alpha d) \) is smaller than \( f(x) \) as illustrated.

The preceding observations form the gradient-based optimization methods (see [4]).

For step \( k \) when we are at \( x^k \in S \), we move to \( x^{k+1} \) by taking a step in the direction of \( d^k \). If \( \nabla f(x^k) \neq 0 \), the direction \( d^k \) is chosen so that
\[
\nabla f(x^k)'d^k \leq 0,
\]
and the stepsize \( \alpha^k \) is chosen to be positive. If \( \nabla f(x^k) = 0 \), the method stops, i.e., \( x^{k+1} = x^k \). In view of the relation \( \nabla f(x^k)'d^k \leq 0 \) of the direction \( d^k \) and the gradient \( \nabla f(x^k) \), we call algorithms of this type gradient methods.

Generally, gradient methods do not converge in a finite number of steps, so it is necessary to set certain criteria for terminating the iterations with some assurance that the method does converge to a stationary point. In addition, there are many possibilities for choosing the stepsize \( \alpha^k \). For instance, if it is known that the method converges, we might consider \( \alpha^k \to 0 \) and \( \sum_{k=0}^{\infty} \alpha^k = \infty \) before setting the stepsize (see Propositions 1.2.3 and 1.2.4 in [4]).
3.1.2 Constrained Optimization

Similar to the gradient based method we discussed in the previous subsection, we proceed to describe a gradient based method that applies to nonlinear minimization problems with constraints.

**Theorem 1** Let $z$ be a fixed vector in $\mathbb{R}^n$ and consider the problem of finding a vector $x^*$ in a closed convex set $S$, which is at a minimum distance from $z$; that is

$$
\min_{x} f(x) = \|z - x\|^2 \\
\text{such that } x \in S
$$

We refer to this problem as the projection of $z$ on $S$ and denote its solution by $Pr_S(z)$ (see [4]).

Gradient projection methods solve a subproblem at each iteration with quadratic cost. The simplest gradient projection method is a feasible direction method of the form

$$
x^{k+1} = x^k + \alpha^k (\bar{x}^k - x^k),
$$

where

$$
\bar{x}^k = Pr_S(x^k - s^k \nabla f(x^k)).
$$

$\alpha^k$ is a stepsize and $s^k$ is a positive scalar. Thus, to obtain the vector $\bar{x}^k$, we take a step $s^k \nabla f(x^k)$ along the negative gradient, as in the unconstrained minimization case that we discussed above. We then project the result $x^k - s^k \nabla f(x^k)$ on $S$, thereby obtaining the feasible vector $\bar{x}^k$. Finally, we take a step along the feasible direction $\bar{x}^k - x^k$ using the stepsize $\alpha^k$.

In this thesis, we view the scalar $s^k$ as a stepsize as we select $\alpha^k = 1$ for all $k$, in which case $x^{k+1} = \bar{x}^k$ and the method becomes

$$
x^{k+1} = Pr_S(x^k - s^k \nabla f(x^k))
$$

(3.1)
If \( x^k - s^k \nabla f(x^k) \) is feasible, the gradient projection iteration becomes an unconstrained case iteration, in the form of (3.1). Notice that the following result applies in the constrained case:

\[
x^* = Pr_s(x^* - s^k \nabla f(x^*)) \text{ for all } s > 0 \text{ if and only if } x^* \text{ is stationary.}
\]

Thus the method stops if and only if it encounters a stationary point.

There are several stepsize selection procedures in the gradient projection method. For instance, constant stepsize or diminishing stepsizes where \( s^k \to 0, \sum_{k=0}^{\infty} s^k = \infty \). The convergence properties are similar to the ones of their unconstrained counterpart. An important assumption is Lipshcitz continuity (see Definition 6). Although we do not know whether this property is always true for our case, in Chapter 4, we have observed it does hold computationally for the problem instances we study in this thesis.

For a non-differentiable function, the subgradient (see Definition 2) method relates to the gradient methods and the gradient projection methods, but uses subgradients in place of gradients as directions of improvement of the distance to the optimum. There is a version of the method that uses subgradient information in the place of a gradient (see [4] for more details).

### 3.2 Algorithm Description and Discussion

**Proposition 4** The profit function \( r(D) \) is a concave function on \( \mathbb{R}^{NT}_+ \).

**Proof:** Follows from Assumptions 1, 2, Proposition 1, and the definition of function \( r(D) = P(D)/D - g(D) \). Q.E.D.

Our objective is to maximize the profit \( r(D) \) as a concave maximization problem over a convex feasible region. This is equivalent to the convex minimization problem of \( -r(D) \), which can be solved through gradient projection methods.
From the discussion of Chapter 2, the revenue function $P(D)'D$ is a differentiable function of $D$ and the gradient for every $D$ exists. The cost function $g(D)$ is nevertheless a non-differentiable piecewise linear function where subgradients exist.

We have seen that if $P(D)'D$ is defined, finite, and linear in the vicinity of a certain vector $D$, then there is a unique optimal dual solution, equal to the gradient of $P(D)'D$ at that point, which leads to the interpretation of dual optimal solutions as marginal costs. We would like to extend this interpretation so that it remains valid at the breakpoint of $g(D)$.

**Theorem 2** Consider the general linear programming problem of minimizing $c'x$ subject to $Ax = b^*$ and $x \geq 0$ is feasible and that the optimal cost is finite. Then a vector $p$ is an optimal solution to the dual problem if and only if it is a subgradient of the optimal cost function $F$ at point $b^*$.

**Proof:** See Theorem 5.2 in [5] for details.

Therefore, for our case, a vector is a subgradient of the optimal cost function of $g(D)$ at $D$ if and only if it is an optimal solution to the dual problem of $g(D)$. In the above formulation, a subgradient of interest is given by the dual variable $z_{it}$. Furthermore, from the linearity property of the gradient and subgradient, the subgradient of $r(D)$ is obtained as the difference between the gradient of the revenue function, $P(D)'D$, and the subgradient of the cost function $g(D)$, i.e., $\nabla(P(D)'D) - \nabla g(D)$. As we have already observed in Chapter 2, given a demand vector, the optimal production problem becomes a transportation problem that can be solved via linear programming. We use this observation to devise an iterative gradient-based algorithm. The objective function is rewritten as the revenue part minus the cost part, which is the optimal transportation cost given the demand levels. This helps employ a gradient-based approach efficiently in order to compute a subgradient at each iteration of the algorithm. Specifically, the gradient of the revenue part of the objective is computed.
directly and the subgradient of the cost part is computed via the dual solution of the transportation linear programming problem. The procedure is repeated iteratively.

**Theorem 3** An optimal solution of the multi-period joint pricing and inventory control problem is equivalent to the solution of the fixed point problem

\[ D^* = Pr_D(D^* - \alpha h^*), \quad (3.2) \]

where \( h^* \) denotes a subgradient of \(-r(.)\) at \( D^* \), \( D \) is the feasible region of \( D \), and \( \alpha > 0 \) is a sufficiently small constant stepsize.

**Proof:** The proof follows from the fact that the joint pricing and inventory control problem is a concave maximization problem over a convex feasible region (see Proposition 4). We then use the equivalent first order optimality conditions of a convex minimization problem and subsequently the reformulation of any convex minimization problem as a fixed point problem through a projection operator (see [4] for more details). Q.E.D.

Furthermore, the projection operator can be transformed to a simple quadratic problem as demonstrated in the following proposition.

**Proposition 5** Let \( D^{k+1} = Pr_D(D^k - \alpha^k h^k) \) for some \( D \in D \).

Then this is equivalent to a quadratic problem

\[ D^{k+1} = \arg \min_{D \in D} 2(h^k)^T D + \frac{1}{\alpha^k} \| D^k - D \|^2, \]

where \( \alpha^k > 0 \) is a sufficiently small constant as defined above.

**Proof:** For

\[ D^{k+1} = \arg \min_{D \in D} 2(h^k)^T D + \frac{1}{\alpha^k} \| D^k - D \|^2, \]
where $h^k$ and $D^k$ are given vectors and $\alpha^k$ is a sufficiently small constant as defined above.

By adding the constant term $-2(h^k)^T D^k + \alpha^k \|h^k\|^2$ to the objective, we can equivalently write this problem as

$$D^{k+1} = \arg\min_{D \in \mathcal{D}} -2(h^k)^T (D^k - D) + \alpha^k \|h^k\|^2 + \frac{1}{\alpha^k} \|D^k - D\|^2$$

$$D^{k+1} = \arg\min_{D \in \mathcal{D}} \frac{1}{\alpha^k} \|D^k - \alpha^k h^k - D\|^2$$

This is equivalent to the projection of the vector $(D^k - \alpha^k h^k)$ onto the convex set $D \in \mathcal{D}$. i.e.

$$D^{k+1} = Pr_D(D^k - \alpha^k h^k).$$

Q.E.D.

From iteration to iteration, the linear transportation subproblem differs only in terms of the right-hand side (that is, in terms of the demand values). This observation allows us to compute the subgradient of $g(D)$ in each iteration by re-optimizing the dual LP of $g(D)$ starting with the demand values from the previous iteration. Since an initial dual feasible solution is available, we use the dual simplex method whereby dual feasibility is maintained while we drive towards primal feasibility to obtain the optimal solution. In particular, we utilize the special structure of this transportation problem and apply the dual network simplex method which is a more computationally efficient method.

**Theorem 4 Convergence for Constant and Diminishing Stepsizes**

Let $x^k$ be a sequence generated by a gradient method

$$x^{k+1} = x^k + \alpha^k d^k,$$
where $d^k$ is gradient related. Assume that for some constant $L \geq 0$, we have

$$
\|\nabla f(x) - \nabla f(y)\| \leq L \|x - y\|, \quad \forall x, y \in \mathbb{R}^n,
$$

where $L$ is the Lipshcitz constant.

Assume that either

* there exists a scalar $\epsilon$ such that for all $k$

$$
0 < \epsilon \leq \alpha^k \leq \frac{(2 - \epsilon) \left| \nabla f(x^k)'d^k \right|}{L \|d^k\|^2}
$$

or

* $\alpha^k \to 0$ and $\sum_{k=0}^{\infty} \alpha^k = \infty$.

Then either $f(x^k) \to -\infty$ or else $f(x^k)$ converges to a finite value and every limit point of $x^k$ is a stationary point.

**Proof:** See Propositions 1.2.3 and 1.2.4 in [4].

### 3.3 Procedures and Flowchart

In what follows, we propose an gradient-based algorithm $A$. $\alpha^k$ is set to be sufficiently small. We also set $\|D^{k+1} - D^k\| \leq \epsilon$ as the stopping condition.

- Start with an initial feasible demand vector, $D^0 \in D \subseteq \mathbb{R}_{+}^{NT}$.

- At step $k + 1$, given a feasible demand vector $D^k$, solve the linear optimization problem $g(D^k)$.

$h^k$ denotes the subgradient of $-r(.)$ at $D^k$ that corresponds to the dual variables $z_u$. If $g(D^k)$ is degenerate, then in the Simplex method we use the lexicographically smallest row to exit the basis (see [5]).
\[ D^{k+1} = Pr_D(D^k - \alpha^k h^k). \]

For \( D \in \mathcal{D} \), this is equivalent to

\[ D^{k+1} = \arg\min_{D \in \mathcal{D}} 2(h^k)^T D + \frac{1}{\alpha^k} ||D^k - D||^2. \]

where \( \alpha^k > 0 \) is a sufficiently small constant as defined above.

To sum up, we present the algorithm in a flowchart as follows.
Initial Feasible Demand \( D^* \in K \)

\[ K = \{ D | D_{it} \geq 0 \} \]

\[ \sum_{i=1}^{N} \sum_{t=1}^{r} D_{it} \leq \sum_{s=1}^{r} C_s, \quad r = 1, \ldots, T \}

\[ g(D) = \min \frac{b}{x} \]

such that

\[ \sum_{t:s \leq T} x_t^s \leq C_s, \quad s = 1, \ldots, T \]

\[ \sum_{i=1}^{N} \sum_{t=1}^{T} x_t^i \geq d_{it}, \quad t = 1, \ldots, T, \quad i = 1, \ldots, N \]

\[ x_t^s \geq 0, \quad s = 1, \ldots, T, \quad t = s, \ldots, T, \quad i = 1, \ldots, N. \]

\[ -f(D) = -P(D)'D + g(D) \]

\[ h^k = -P(D^k) - \nabla P(D^k)D^k + Z^* \]

\[ D^{k+1} = \text{arg min} \frac{1}{\alpha^k} \| D^k - D \|^2 \]

Figure 3-1: Flowchart
Chapter 4

Simulations and Discussions

In the previous chapters, we formulated and analyzed a model for the joint pricing and inventory control problem. We also proposed an iterative gradient-based algorithm to solve this problem efficiently.

This chapter provides a computational study with the goal to demonstrate how the performance of our algorithm is compared with using the original formulation and applying commercial and publically available software packages, specifically, CPLEX and LOQO. In Section 4.1, we describe the motivations as well as the hardware and software environment for the numerical simulations we performed. In Section 4.2 and 4.3, with different scales of varied products and time periods (up to 60 products within 60 periods), we conduct various numerical experiments with regard to different initial points, for nonlinear and linear price-demand relation cases. The performance of our algorithm, in terms of efficiency and accuracy, is compared with those of the original formulation solved through CPLEX and the widely-used nonlinear solver LOQO. In Section 4.4, over the multi-period horizon, we consider a price-demand relation such that the prices from past periods can affect the current demand. We illustrate the efficiency of our algorithm by considering several computational experiments for linear demand functions where the current demand depends on past and current prices, with the potential to include expectations of prices in the future.
4.1 Motivations and Computer Treatment

4.1.1 Motivations and Framework

To evaluate the performance of our algorithm, we create instances of various scales, with respect to time period and product variety, for both linear and nonlinear demand functions. We apply our algorithm and compare the efficiency and accuracy with that of using the original formulation (see 2.1 in Chapter 2) solved by CPLEX and LOQO. In what follows we discuss some of the motivating questions that our computations address.

- Our algorithm uses only first order information instead of Hessian information. Computing the Hessian matrix at each iteration can be time consuming and that for some demand functions computing the Hessian matrix is not possible at all.

How does our algorithm perform when the scale increases compared with CPLEX or LOQO experimentally?

This question is addressed for both linear and nonlinear demand models in Section 4.2 and Section 4.3, respectively. For linear demand models, as the scale increases, our algorithm dominates LOQO and is close to CPLEX. For 60 products within 60 periods, in the best case, our algorithm only costs 76 percent of the time required by CPLEX. For nonlinear demand models, the algorithm also outperforms LOQO in most instances. In the best case, it only costs one-sixth of the time required by LOQO.

- When the demand function is nonlinear, the original formulation is possibly no longer quadratic and thus cannot be solved through CPLEX. One alternative approach is to use LOQO. In our iterative algorithm, instead of computations
involving non-quadratic functions, at each iteration, we solve a linear trans-
portation problem and then subsequently solve a quadratic optimization prob-
lem with fairly simple constraints.

How does our algorithm perform compared with the original formulation solved
through LOQO when the problem is not quadratic?

This question is investigated in Section 4.3. Our algorithm outperforms LOQO
in most instances. In the best case, it only takes one-sixth of the time required
by LOQO.

- We are motivated by our own experience that our purchasing decisions depend
  on past prices, current prices and expectations of future prices.

How does our algorithm perform when we consider a demand function that is
affected not just by current prices but also by past prices?

This question is studied in Section 4.4. For instance, our worst case takes 1.7
times the time required by CPLEX, our best case takes only 95 percent of the
time required by CPLEX.

**Problem Size**

Most of our instances range from 10 products within 10 periods to 60 products within
60 periods. For the 60 products within 60 periods, the original formulation (2.1)
has 3600 decision variables with 18060 constraints. The approach we consider (see
Flowchart 3.3) has 3660 decision variables with 14460 constraints for the linear part
and 3600 decision variables with 3660 constraints for the nonlinear part of the for-
mulation. The size of the matrix $B$ as discussed in Section 4.2.1 becomes $3600 \times 3600$.  

55
We have investigated some cases from 10 products within 10 periods up to 100 products within 100 periods as in the section 4.2.4, where the size of the matrix $B$ becomes $10000 \times 10000$.

### 4.1.2 Computational Treatment

The models in this work were modeled using AMPL and solved with CPLEX 10.1. The hardware used was an IBM Thinkpad with Intel Duo Core CPU 2.0 Ghz and 2.0 Gb of RAM, under the Windows Vista operating system.

AMPL is a comprehensive and powerful algebraic modeling language for linear and nonlinear optimization problems, in discrete or continuous variables. It provides an interactive command environment with batch processing options as well as a broad support for common notation and familiar concepts to formulate optimization models and examine solutions. The author has found useful its flexibility and convenience in rapid prototyping and model development. In general, it allows for easy definitions of a large myriad of complex optimization programs in concise forms and is able to link up to a wide variety of powerful solvers including the CPLEX optimizer.

CPLEX 10.1 is a state-of-the-art solver which utilizes algorithms that implements the dual simplex, primal simplex, network simplex algorithms and barrier methods. It provides flexible, high-performance optimizers used when solving linear programming, quadratic programming, quadratically constrained programming and mixed integer programming problems. Of much relevance and significance to this work is their outstanding performance in solving large scale quadratic problems. The degeneracy control and warm-start options were useful in implementing the iterative based algorithm we propose.

LOQO is a system for solving smooth constrained optimization problems. The problems can be linear or nonlinear, convex or non-convex, constrained or unconstrained.
The only real restriction is that the functions defining the problem must be smooth (at the points evaluated by the algorithm). If the problem is convex, LOQO finds a globally optimal solution. Otherwise, it finds a locally optimal solution near to a given starting point. The author has found its ability in handling small scale problem useful for both linear and nonlinear demand functions.

MATLAB is a high-level language and interactive environment for computationally intensive tasks. In this thesis, it has been used to generate the instances and compute the Lipschitz constant.

4.2 Linear Demand Cases

In this section, we provide computational results for the linear demand model where the original problem is quadratic. Throughout this chapter, unless otherwise stated, \( n = N \times T \), where \( N \) is the number of products and \( T \) is the number of time periods. As discussed in Section 2.3, the revenue function can be written as

\[
P(D)'D = B^{-1}(a - D) \cdot D
\]

\[
= \sum_{i=1}^{n} \left( \sum_{j=1}^{n} b_{ij}(a_j - d_j) \right) d_i,
\]

where \( B^{-1} = [b_{ij}] \).

As discussed in Section 3.3, the stopping condition for this iterative algorithm is set as

\[
\|D^{k+1} - D^k\| \leq 0.05.
\]

With this threshold, we converge to the true optimum exactly for linear demand models.

The original approach to this problem is to solve the problem as in (2.1).
4.2.1 Instance Generation

In this case, the demand is a function of current prices only, indicating no correlation between time periods. Therefore, $B$ is a symmetric block diagonal matrix with zero off-diagonal blocks. There are $T \times T$ blocks with size $N \times N$. For the same time period, we assume that a decrease of the price for product $i$ or an increase of the prices for its substitutes will increase the demand for product $i$. Therefore, each nonzero block in $B$ has positive diagonal entries and negative off-diagonal entries. As the product of $B$ and $B^{-1}$ returns the identity matrix, $B^{-1}$ is a symmetric block diagonal matrix with nonnegative entries.

From the discussion of (2.12) in Section 2.2.1, the Hessian matrix is $H[D] = -2B^{-1}$. We choose $B^{-1}$ to be diagonally dominant. This ensures the diagonal dominance of the Hessian matrix as well. To generate one nonzero block in matrix $B^{-1}$, we create a matrix of size $N \times N$, which is with absolute values of data generated from the standard Normal distribution. For all $j = 1, ..., N$, the diagonal entry in the $j^{th}$ row is then set to be twice the sum of all the values in the $j^{th}$ row. This new matrix forms one nonzero block in $B^{-1}$ and ensures matrix $B^{-1}$ to be diagonal dominant. In addition, the entries of vector $a$ follow the uniform distribution between 1 and 2. By noticing that

$$P(D) = B^{-1}(a - D),$$

we impose the constraint that $D \leq a$ to ensure non-negativity of the prices. The capacity constraints are thereafter set as half of the upper bound of the demand within that period, i.e., $0.5a$.

In addition, the production cost and holding cost, respectively, for each time period are generated from a uniform distribution on the unit interval.
4.2.2 Lipschitz Constant

To determine the Lipschitz constant, from the discussion in Section 2.2.1 with \( f(P) = P'D(P) \) (see (2.9)), the following applies

\[
\nabla f(D) = a - 2B^{-1}D.
\]

Given a feasible demand \( D^1 \), we have

\[
||\nabla f(D) - \nabla f(D^1)||^2 = ||2B^{-1}(D^1 - D)||^2 \\
\leq 4\lambda_{\text{max}}[(B^{-1})^T(B^{-1})]||D^1 - D||^2.
\]

where \( \lambda_{\text{max}}[(B^{-1})^T(B^{-1})] \) refers to the maximal eigenvalue of matrix \((B^{-1})^T(B^{-1})\). From Definition 6, we know that the Lipschitz continuity constant can be written as

\[\sqrt{\lambda_{\text{max}}[(B^{-1})^T(B^{-1})]}]. \tag{4.1}\]

4.2.3 Performance for Various Scales, Starting points, and Levels of Dominance

Performance for Various Scales

We consider the situation where demand depends on its own price and prices of substitute products within the same time period. We compare the performance of our algorithm with different scales of varied products and time periods (up to 60 products within 60 periods) to those of commercial and publically available software packages, specifically, CPLEX and LOQO. Both CPLEX and LOQO converge to the optimal value under these conditions; their results are independent of their initial starting points.

As shown in Figure 4-1, the dashed lines denote the solving time of the original for-
mulation through LOQO and CPLEX, respectively. For small scale problems, LOQO solves this problem very fast. But as the scale of the problem increases, the time required by LOQO soars dramatically. CPLEX solves the problem consistently fast as it takes only one-sixth of the time required by LOQO for 60 products within 60 periods. The performance of our algorithm remains close to that of CPLEX. This will be clearer as we discuss the performance for various starting points as follows.

**Performance for Various Starting Points**

For the starting points with different L-1 norms, our algorithm converges to the true optimum exactly with slight differences in timing. When the scale increases, the solving time is close to or even shorter than the results of CPLEX. Except for the worst case starting from the upper bound, our algorithm takes 94 percent and 76 percent, for two different starting points respectively, of the time required by CPLEX to solve the case of 60 products within 60 periods. This can be observed when we zoom in and notice the difference in Figure 4-2.

![Figure 4-1: Comparison of the time required by different approaches](image)

Figure 4-1: Comparison of the time required by different approaches
Performance for Different Levels of Dominance

Varying the level of dominance of the B matrix, as discussed in Chapter 2, the results shown in Figure 4-3 and Figure 4-4 resemble what we obtain for a less dominant matrix B. Our algorithm continues to dominate LOQO and is close to or faster than CPLEX, depending on the choice of the initial points. For 60 products within 60 periods, in the best case, our algorithm takes 83 percent of the time required by CPLEX; in the worst case, our algorithm takes 1.12 times the time required by CPLEX.
4.2.4 Performance for Different Capacity Constraints

In this section, we apply our algorithm in different capacity settings and compare its performances with those of CPLEX. These experiments are conducted on pcl in the
Varying the capacity constraints, with scales up to 100 products within 100 periods, our algorithm converges exactly to the true optimum but with great differences in timing. When none of the capacity constraints are tight, the solving time for our algorithm is one-third of that required by CPLEX. When some of the capacity constraints are tight, our best case takes 62 percent of the time required by CPLEX while our worst case takes 1.5 times the time required by CPLEX. When all the capacity constraints are tight, we take 33 times the time required by CPLEX in the worst case. Take 100 products within 100 periods for instance, if the constraints are not tight, it takes 48 iterations to reach optimum in the dual network simplex step; if some of the constraints are tight, 132 iterations are required to reach the optimum. We believe that when the capacity is tight the reoptimization of $g(D)$ in each iteration takes a few steps, while when the capacity is high the optimal basis of the dual stays the same regardless of the values of $D$. These results are illustrated in Figure 4-5, 4-6, and 4-7 as follows.

4.2.5 Performance for Small Number of Time Periods

In this section, we investigate the case when the number of time periods is small, say 5 time periods. These experiments are conducted on pcl in the MIT Operations Research Center (Linux orc-pcl 2.6.15-51-amd64-xeon). For 100 products, on average, our algorithm converges to 97 percent of the true optimum but takes only 11 percent of the time required by that of CPLEX. We believe that this might result from the reduction of the problem size for smaller number of time periods. These results are illustrated in Figure 4-8.
Figure 4-5: Comparison of the time required by different approaches for low capacity

Figure 4-6: Comparison of the time required by different approaches for medium capacity
In this section, we provide computational results for nonlinear demand models. In particular, our approach is important when the original problem formulation is no longer quadratic and as a result, we cannot employ CPLEX to solve it. From the discussion (see 2.15) in Chapter 2, we know that the revenue function can be written as

\[ P(D)^T D = B^{-1} (a - \ln(D)) \cdot D \]

\[ = \sum_{i=1}^{n} \left( \sum_{j=1}^{n} b_{ij}(a_j - \ln(d_j)) \right) d_i, \]

where \( B^{-1} = [b_{ij}] \).

As discussed in Section 3.3, the stopping condition for this iterative algorithm is set as

\[ ||D^{k+1} - D^k|| \leq 0.05. \]

The original approach to this is to solve the problem as in (2.1).
4.3.1 Instance Generation

In this case, the demand is a function of current prices only, indicating no correlation between time periods. Therefore, $B$ is a symmetric block diagonal matrix with zero off-diagonal blocks. There are $T \times T$ blocks with size $N \times N$. For the same time period, we assume that a decrease of the price for product $i$ or an increase of the prices for its substitutes will increase the demand for product $i$. Therefore, each nonzero block in $B$ has positive diagonal entries and negative off-diagonal entries. As the product of $B$ and $B^{-1}$ returns the identity matrix, $B^{-1}$ is a symmetric block diagonal matrix with nonnegative entries.

We assume $B^{-1}$ is a symmetric matrix and the demand $D_j \geq 1, j = 1, \ldots, n$. From the discussion of (2.18) in Section 2.2.2, the Hessian matrix has elements $H_{ij} = -\left(\frac{b_j}{D_j} + \frac{b_k}{D_k}\right)$. One sufficient condition to ensure diagonal dominance is $|H_{ii}| > \sum_{j \neq i} |H_{ij}|$. The entries of vector $a = (a_1, a_2, \ldots, a_n)$ follow the uniform distribution between 1 and 2. Therefore, we generate instances satisfying:
1. \( b_{ij} \geq e^{a_0} \sum_{i \neq j} b_{ij}, \ i = 1, \ldots, n; \ j = 1, \ldots, n \)

2. \( 1 \leq D_j \leq e^{a_j} \)

These conditions ensure that

\[
|H_{ii}| \geq b_{ii} \frac{2}{e^{a_0}} \geq \sum_{i \neq j} 2b_{ij} \geq \sum_{i \neq j} 2b_{ij} \left( \frac{1}{D_j} + \frac{1}{D_i} \right) = \sum_{j \neq i} |H_{ij}|,
\]

where \( a_0 \) is the maximal entry for parameter vector \( a, i = 1, \ldots, n, j = 1, \ldots, n \).

Therefore, to generate one nonzero block in matrix \( B^{-1} \), we create a matrix of size \( N \times N \), with absolute values of data generated from the standard Normal distribution. For all \( j = 1, \ldots, N \), the diagonal entry in the \( j^{th} \) row is then set to be \( e^{a_0} \) times the sum of all the values in the \( j^{th} \) row. This new matrix forms one nonzero block in \( B^{-1} \) and ensures \( B^{-1} \) to be A diagonal dominant matrix. An example of a block of matrix \( B^{-1} \) and an example of a vector \( a \) are attached in the appendix. By noticing that

\[
P(D) = B^{-1}(a - \ln(D)),
\]

we impose the constraint \( \ln(D) \leq a \) to ensure the non-negativity of the price. The capacity constraints are thereafter set as half of the upper bound of the demand in that period, i.e., \( 0.5e^a \).

In addition, the production cost and holding cost, respectively, for each time period are generated from a uniform distribution on the unit interval.
4.3.2 Lipschitz Constant

To determine the Lipschitz constant, we let $D^q \in \mathcal{D}$, where $D^q_p$ refers to the $p^{th}$ entry of vector $D^q$. From the discussion in Section 2.2.2 with $f(P) = P'D(P)$ (see (2.13)), we have that

$$[\nabla f(D^1)]_m = [\nabla f(D^2)]_m + (D^1 - D^2) \cdot [\nabla f(\bar{D})]'$$

where $\bar{D} \in [D^1, D^2]$

$$[\nabla f(D^1)]_m - [\nabla f(D^2)]_m = -\sum_{j=1}^{n} \left( \frac{b_{mj}}{\bar{D}_j} + \frac{b_{jm}}{\bar{D}_m} \right) \cdot (D^1_j - D^2_j)$$

$$([\nabla f(D^1)]_m - [\nabla f(D^2)]_m)^2 = \left( \sum_{j=1}^{n} \left( \frac{b_{mj}}{\bar{D}_j} + \frac{b_{jm}}{\bar{D}_m} \right) \cdot (D^1_j - D^2_j) \right)^2$$

$$\leq \sum_{j=1}^{n} \left( \frac{b_{mj}}{\bar{D}_j} + \frac{b_{jm}}{\bar{D}_m} \right)^2 \sum_{j=1}^{n} (D^1_j - D^2_j)^2$$

$$\sum_{m=1}^{n} ([\nabla f(D^1)]_m - [\nabla f(D^2)]_m)^2 \leq \sum_{m=1}^{n} \left[ \sum_{j=1}^{n} \left( \frac{b_{mj}}{\bar{D}_j} + \frac{b_{jm}}{\bar{D}_m} \right)^2 \cdot \sum_{j=1}^{n} (D^1_j - D^2_j)^2 \right]$$

Since $D_j \geq 1$, from Definition 6, the Lipschitz constant is then set to be

$$L = 2 \sqrt{\sum_{j=1}^{n} \sum_{i=1}^{n} (b_{ij})^2}.$$  

4.3.3 Performance for Various Scales, Starting points, and Levels of Dominance

Performance for Various Scales

We consider the situation where demand depends on its own price and prices of substitute products within the same time period. We compare the performance of our algorithm with different scales of varied products and time periods (up to 60 products within 60 periods) to that of LOQO. As the scale of the problem increases and due to the choices of tolerance parameters, our algorithm converges to an optimal that is slightly less than that obtained through LOQO. For various starting points
(Small, Medium, and Large) with respect to the values of L-1 norm, the difference between the optimum obtained through our algorithm and the true optimum remains within 0.01 percent of the true optimum for all instances. This level of discrepancy is negligible. In general, our algorithm outperforms LOQO in most cases, especially when the scale is large. These results are clearer when we discuss the various starting points as follows.

**Performance for Various Starting Points**

Figures 4-9, 4-10, and 4-11 show the comparison of solving time between our algorithm and LOQO for different starting points.

![Graph showing comparison of solving time](image)

Figure 4-9: Comparison of the time required by different approaches for nonlinear demand function

As shown in these figures, in spite of the difference between various starting points, the solving time through LOQO remains consistently high. On the contrary, even in the worst case where the starting point is the upper bound, our performance is marginally different, within 20 percent, from that of LOQO. Except for the worst
Figure 4-10: Comparison of the time required by different approaches for nonlinear demand function

Figure 4-11: Comparison of the time required by different approaches for nonlinear demand function
case, our algorithm outperforms LOQO to a great extent. As shown in Figure 4-10, LOQO costs more than six times the time required by our algorithm.

**Performance for Different Levels of Dominance**

By increasing the level of dominance of the matrix B, where the diagonal elements are much larger than the sum of the absolute value of the off-diagonal entries, we obtain results shown in Figures 4-12, 4-13, and 4-14, which exhibit a similar trend. Our performance is close to that of LOQO for small scale problems and dominates LOQO as the scale increases. For 60 products within 60 periods, in the best case, our algorithm takes 35 percent of the time required by LOQO; in the worst case, our algorithm takes 77 percent of the time required by LOQO.

![Figure 4-12: Comparison of the time required by different approaches for nonlinear demand function](image)

Figure 4-12: Comparison of the time required by different approaches for nonlinear demand function
Figure 4-13: Comparison of the time required by different approaches for nonlinear demand function

Figure 4-14: Comparison of the time required by different approaches for nonlinear demand function
4.4 Inter-period Linear Demand Cases

Over multi-period horizon, we consider the price-demand relationship among periods. $B$ is no longer a block diagonal matrix but is still diagonal dominant. From Assumption 1, we further assume that the demand for a product at a time period decreases as

- the price for the same product at that time period increases;
- the price for the substitutes at that time period decreases;
- the prices for the same product at the previous time periods decrease;
- the prices for the substitutes at the previous time periods increase;
- the prices expectation for the same product at the future time periods decrease;
- the prices expectation for the substitutes at the future time periods increase.

4.4.1 Demand Models Based on Past and Current Prices

We first consider the situation that the demands depend on past and current prices only. We assume that the shorter the difference between time periods is, the stronger the correlation between products is. In our simulations, we make the entries in the block matrices which are on the left of the diagonal to be much smaller than those in the diagonal block matrices.

In the multiproduct case, the demand function is:

$$D(P) = a - BP.$$
where \( \mathbf{a} = (a_1, a_2, \ldots, a_n) \) is a vector of coefficients, \( \mathbf{a} \geq 0 \). The value of \( \mathbf{a} \) tells us the demand when the price is equal to zero.

\[
B = \begin{bmatrix}
B_1 & 0 & \cdots & 0 \\
A_2 & B_2 & 0 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & A_T & B_T
\end{bmatrix},
\]

\( A_i \) and \( B_i \) are block matrices with size \( N \times N \), for all \( i = 1, \ldots, T \). The absolute values of the entries in \( A_i \) are much smaller than those in \( B_i \).

As discussed in Section 3.3, the stopping condition for this iterative algorithm is set as

\[
\|D^{k+1} - D^k\| \leq 0.05.
\]

With this threshold, we converge to the true optimum exactly for linear demand models.

**Instance Generation and Lipschitz Constant**

Following the discussion in Section 4.2 for linear demand models, to generate one nonzero block \( B_i \) in matrix \( B^{-1} \), we create a matrix of size \( N \times N \), which is filled with absolute values of data generated from the standard Normal distribution. For all \( j = 1, \ldots, N \), the diagonal entry in the \( j^{th} \) row is then set to be twice the sum of all the values in the \( j^{th} \) row. This new matrix \( B_i \) forms one nonzero diagonal block in \( B^{-1} \). Each entry in the off-diagonal matrix \( A_i \) is then set to be 0.005 times the respective entry in \( B_i \). This ensures the diagonal dominance of matrix \( B^{-1} \). An example of \( A_i \) and an example of \( B_i \) are attached in the appendix. In addition, the entries of vector \( \mathbf{a} \) are set to be 3.

By noticing that

\[
P(D) = B^{-1}(\mathbf{a} - D),
\]

where \( B^{-1} = [b_{ij}] \),
we impose the constraint that $D \leq a$ to ensure the non-negativity of the price. In addition, the production cost and holding cost, respectively, for each time period are generated from a uniform distribution on the unit interval. The Lipschitz constant is generated similarly to (4.1).

Performance for Various Scales

We compare the performance of our algorithm with different scales of varied products and time periods (up to 60 products within 60 periods) to that of CPLEX. Figure 4-15 addresses situations when the demand depends on past and current prices. Both our algorithm and the original formulation solved through CPLEX converge to the correct optimum exactly. For 60 products within 60 periods, in the best case, our algorithm takes 94 percent of the time required by CPLEX. These results are clearer when we discuss the various starting points as follows.

Performance for Various Starting Points

Taking the 60 product within 60 time periods as an example, our worst case, starting from the upper bound of the feasible region, takes 1.4 times the time required by CPLEX while our best case takes 94 percent of the time required by CPLEX.
4.4.2 Demand Models Based on Past, Current, and Future Prices

Similarly, we consider the situation that the demand depends on past, current, and future prices. In our simulations, we make the entries in the block matrices which are on the left and on the right of the diagonal to be much smaller than those in the diagonal block matrices, i.e.,

\[
B = \begin{bmatrix}
B_1 & C_1 & \cdots & 0 \\
A_2 & B_2 & C_2 & \cdots & 0 \\
\vdots & \ddots & \ddots & \ddots & \vdots \\
0 & \cdots & A_T & B_T
\end{bmatrix},
\]

where \(A_i\) and \(B_i\) are block matrices with size \(N \times N\), for all \(i = 1, \cdots, T\). The absolute values of entries in \(A_i\) and \(C_i\) are much smaller than those in \(B_i\).

As discussed in Section 3.3, the stopping condition for this iterative algorithm is set as

\[
\|D^{k+1} - D^k\| \leq 0.05.
\]

With this threshold, we converge to the true optimum exactly for linear demand models.

Instance Generation and Lipschitz Constant

Following the discussion in Section 4.2 for linear demand models, to generate one nonzero block \(B_i\) in matrix \(B^{-1}\), we create a matrix of size \(N \times N\), which is filled with absolute values of data generated from the standard Normal distribution. For all \(j = 1, \cdots, N\), the diagonal entry in the \(j^{th}\) row is then set to be twice the sum of all the values in the \(j^{th}\) row. This new matrix \(B_i\) forms one nonzero diagonal block in \(B^{-1}\). Each entry in the off-diagonal matrix \(A_i\) and \(C_i\) is then set to be 0.005 times the respective entry in \(B_i\). This ensures the diagonal dominance of matrix \(B^{-1}\). An example of \(A_i\) and an example of \(B_i\) are attached in the appendix. In addition, the entries of vector \(a\) are set to be 3.
By noticing that

\[ P(D) = B^{-1}(a - D), \]

where \( B^{-1} = [b_{ij}] \),

we impose the constraint that \( D \leq a \) to ensure the non-negativity of the price. In addition, the production cost and holding cost, respectively, for each time period are generated from a uniform distribution on the unit interval. The Lipschitz constant is generated similarly to (4.1).

**Performance for Various Scales**

We compare the performance of our algorithm with different scales of varied products and time periods (up to 60 products within 60 periods) to that of CPLEX. Figure 4-16 addresses situations when the demand depends on past and current prices. Both our algorithm and the original formulation solved through CPLEX converge to the correct optimum exactly. For 60 products within 60 periods, in the best case, our algorithm takes 95 percent of the time required by CPLEX.

**Performance for Various Starting Points**

As shown in Figure 4-16, although our worst case takes 1.7 times the time required by CPLEX, our best case takes only 95 percent of the time required by CPLEX.

**4.4.3 Lipschitz Continuity**

In the simulations, we note that the difference of the subgradients \( g(D) \) between consecutive iterations drops to zero very quickly. Essentially this implies the subgradient is Lipschitz continuous and hence the result in Chapter 3 applies. In addition, although our algorithm converges to the optimum pretty fast, the time in calculating the Lipschitz continuity constant through finding the eigenvalue of a large matrix (see Section 2.2) is not negligible. A future research direction of interest is to devise an
Figure 4-15: Comparison of the time required by different approaches with relations between past and current periods.

Figure 4-16: Comparison of the time required by different approaches with relations among periods.
algorithm which can return the eigenvalue of a large scale matrix fast.

4.4.4 Conclusions

In conclusion, we have devised efficient algorithms that can handle linear and non-linear price-demand relations. We compare the performance of our algorithm, with scales up to 60 products within 60 periods, with those of the original formulation solved directly through LOQO and CPLEX. For most of the large scale instances, our algorithm outperforms LOQO and is comparable to CPLEX. In addition, over the multi-period horizon, we consider the price-demand relationship between periods. We illustrate the efficiency of our algorithm by considering several computational experiments for linear demand functions where the current demand depends on past and current prices, with the potential to include expectations of future prices. Our performance is much better than LOQO and is comparable to CPLEX in many instances. For future work, we can further explore the far-reaching benefits in the iterative gradient-based scheme, such as the nonlinear demand-price relation among periods in joint pricing and inventory control problems.
Chapter 5

Conclusions

In this thesis, we have investigated a joint pricing and inventory problem as an optimal control problem. In our approach, we consider a setting where multiple products share capacitated production capabilities. We assume that the inventory and the production costs are linear functions of the inventory and the production levels, respectively. Demand for a given product at a specific time period is either a linear or a nonlinear function that depends on past and current prices for all products. No backorders are allowed, i.e., all demands must be satisfied on time. The demand price relation is modeled through an invertible function, which allows us to reformulate the revenue as a function of the demand levels rather than the prices. It effectively enables us to work in the demand space instead of using prices as decision variables. Note that the objective function may not even be differentiable.

We utilize the observation that given a demand vector, the optimal production problem becomes a transportation problem that can be solved via linear programming, and we devise an iterative gradient-based algorithm. The objective function is rewritten as the revenue part of computation minus the cost part, which is the optimal transportation cost given the demand levels. This reformulation helps employ a gradient-based approach efficiently to compute a subgradient in each iteration of the algorithm. Specifically, the gradient of the revenue part of the objective is computed directly and the subgradient of the cost part is computed via the dual solution of the trans-
portation linear programming problem. The procedure is repeated iteratively.

From iteration to iteration, the linear transportation subproblem differs only in terms of the right-hand side (that is, in terms of the demand values). This observation allows us to re-optimize the cost efficiently from one iteration to another, starting with the previous optimal dual solution. Since an initial dual feasible solution is available, we use the dual simplex method whereby dual feasibility is maintained while we drive toward primal feasibility to obtain the new optimal solution. In particular, considering the special structure of this transportation problem, we apply the dual network simplex method.

We have devised efficient algorithms that can handle both linear and nonlinear price-demand relations. We compare the performance of our algorithm, with scales of up to 100 products within 100 periods, with that of the original formulation solved directly through LOQO and CPLEX. Our algorithm uses only first order information instead of Hessian information. For linear price-demand relationship, in the best case, our algorithm only costs 76 percent of the time required by CPLEX. If the number of time periods is small, we only takes 11 percent of the time required by CPLEX. For nonlinear price-demand relationship, in each iteration, we need to solve only a linear optimization problem and subsequently a quadratic problem with fairly simple constraints, instead of handling a non-quadratic problem. For most of the large scale instances, our algorithm outperforms LOQO. In the best case, it only takes one-sixth of the time required by LOQO. In addition, over the multi-period horizon, we consider the price-demand relationship among periods. We illustrate the efficiency of our algorithm by considering several computational experiments for linear demand functions where the current demand depends on past and current prices, with the potential to include expectations of future prices. Our performance is comparable to CPLEX in many instances and our best case takes only 95 percent of the time required by CPLEX.
In conclusion, we believe that the iterative gradient-based algorithm holds promise as a more robust and tractable technique to solve large scale joint pricing and inventory control problems. Its applicability for handling the general demand-price relation is particularly attractive in the non-quadratic nonlinear problem. For future work, we can further explore the far-reaching benefits in the iterative gradient-based scheme, such as the nonlinear demand-price relation among periods in joint pricing and inventory control problems.
Appendix A

Tables

Unless otherwise stated, $N$ denotes number of products and $T$ denotes number of periods. The L-1 norm will be used to measure the value of demand $D$ and starting points (small, medium, and large). Some of the data are attached in this appendix.
Table A.1: A sample block of matrix $B^{-1}$ for nonlinear demand case

<table>
<thead>
<tr>
<th></th>
<th>40.4281</th>
<th>0.4230</th>
<th>0.3593</th>
<th>0.4705</th>
<th>0.9592</th>
<th>0.9875</th>
<th>0.3825</th>
<th>0.5690</th>
<th>0.6736</th>
<th>0.1532</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4230</td>
<td>32.4745</td>
<td>0.9912</td>
<td>0.0130</td>
<td>0.4147</td>
<td>0.5163</td>
<td>0.2128</td>
<td>0.6140</td>
<td>0.2939</td>
<td>0.6673</td>
<td></td>
</tr>
<tr>
<td>0.3593</td>
<td>0.9912</td>
<td>43.9690</td>
<td>0.9390</td>
<td>0.9172</td>
<td>0.9458</td>
<td>0.5086</td>
<td>0.1500</td>
<td>0.1421</td>
<td>0.7816</td>
<td></td>
</tr>
<tr>
<td>0.4705</td>
<td>0.0130</td>
<td>0.9390</td>
<td>46.6245</td>
<td>0.5358</td>
<td>0.2170</td>
<td>0.5200</td>
<td>0.3071</td>
<td>0.9633</td>
<td>0.9012</td>
<td></td>
</tr>
<tr>
<td>0.9592</td>
<td>0.4147</td>
<td>0.9172</td>
<td>0.5358</td>
<td>42.6145</td>
<td>0.6867</td>
<td>0.9190</td>
<td>0.7262</td>
<td>0.5705</td>
<td>0.5092</td>
<td></td>
</tr>
<tr>
<td>0.9875</td>
<td>0.5163</td>
<td>0.9458</td>
<td>0.2170</td>
<td>0.6867</td>
<td>38.4129</td>
<td>0.9452</td>
<td>0.5641</td>
<td>0.2049</td>
<td>0.3836</td>
<td></td>
</tr>
<tr>
<td>0.3825</td>
<td>0.2128</td>
<td>0.5086</td>
<td>0.5200</td>
<td>0.9190</td>
<td>0.9452</td>
<td>40.9943</td>
<td>0.1457</td>
<td>0.2638</td>
<td>0.7518</td>
<td></td>
</tr>
<tr>
<td>0.5690</td>
<td>0.6140</td>
<td>0.1500</td>
<td>0.3071</td>
<td>0.7262</td>
<td>0.5641</td>
<td>0.1457</td>
<td>41.1260</td>
<td>0.0908</td>
<td>0.4533</td>
<td></td>
</tr>
<tr>
<td>0.6736</td>
<td>0.2939</td>
<td>0.1421</td>
<td>0.9633</td>
<td>0.5705</td>
<td>0.2049</td>
<td>0.2638</td>
<td>0.0908</td>
<td>46.7936</td>
<td>0.5704</td>
<td></td>
</tr>
<tr>
<td>0.1532</td>
<td>0.6673</td>
<td>0.7816</td>
<td>0.9012</td>
<td>0.5092</td>
<td>0.3836</td>
<td>0.7518</td>
<td>0.4533</td>
<td>0.5704</td>
<td>36.3805</td>
<td></td>
</tr>
</tbody>
</table>

Table A.2: A sample of vector $\mathbf{a}$ for nonlinear demand case

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.7557</td>
</tr>
<tr>
<td>1.2556</td>
</tr>
<tr>
<td>1.6816</td>
</tr>
<tr>
<td>1.0406</td>
</tr>
<tr>
<td>1.4541</td>
</tr>
<tr>
<td>1.7970</td>
</tr>
<tr>
<td>1.5658</td>
</tr>
<tr>
<td>1.6074</td>
</tr>
<tr>
<td>1.7519</td>
</tr>
<tr>
<td>1.6743</td>
</tr>
</tbody>
</table>

Table A.3: A sample block $B_1$ of matrix $B^{-1}$ for inter-period linear demand case

<table>
<thead>
<tr>
<th></th>
<th>9.2079</th>
<th>0.0661</th>
<th>0.2509</th>
<th>0.6181</th>
<th>0.2554</th>
<th>0.7246</th>
<th>0.1152</th>
<th>0.0696</th>
<th>0.0292</th>
<th>0.9834</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0661</td>
<td>9.9645</td>
<td>0.3576</td>
<td>0.1926</td>
<td>0.1006</td>
<td>0.3469</td>
<td>0.3105</td>
<td>0.9289</td>
<td>0.6930</td>
<td>0.0484</td>
<td></td>
</tr>
<tr>
<td>0.2509</td>
<td>0.3576</td>
<td>12.8974</td>
<td>0.6867</td>
<td>0.5977</td>
<td>0.7348</td>
<td>0.1903</td>
<td>0.7764</td>
<td>0.2528</td>
<td>0.4431</td>
<td></td>
</tr>
<tr>
<td>0.6181</td>
<td>0.1926</td>
<td>0.6867</td>
<td>11.9565</td>
<td>0.7251</td>
<td>0.3834</td>
<td>0.0920</td>
<td>0.4565</td>
<td>0.0430</td>
<td>0.5704</td>
<td></td>
</tr>
<tr>
<td>0.2554</td>
<td>0.1006</td>
<td>0.5977</td>
<td>0.7251</td>
<td>12.9616</td>
<td>0.4161</td>
<td>0.0536</td>
<td>0.6692</td>
<td>0.2957</td>
<td>0.6150</td>
<td></td>
</tr>
<tr>
<td>0.7246</td>
<td>0.3469</td>
<td>0.7348</td>
<td>0.3834</td>
<td>0.4161</td>
<td>10.1314</td>
<td>0.4455</td>
<td>0.3057</td>
<td>0.0932</td>
<td>0.3443</td>
<td></td>
</tr>
<tr>
<td>0.1152</td>
<td>0.3105</td>
<td>0.1903</td>
<td>0.0920</td>
<td>0.0536</td>
<td>0.4455</td>
<td>12.7737</td>
<td>0.6025</td>
<td>0.6979</td>
<td>0.9101</td>
<td></td>
</tr>
<tr>
<td>0.0696</td>
<td>0.9289</td>
<td>0.7764</td>
<td>0.4565</td>
<td>0.6692</td>
<td>0.3057</td>
<td>0.6025</td>
<td>10.9852</td>
<td>0.2905</td>
<td>0.1804</td>
<td></td>
</tr>
<tr>
<td>0.0292</td>
<td>0.6930</td>
<td>0.2528</td>
<td>0.0430</td>
<td>0.2957</td>
<td>0.0932</td>
<td>0.6979</td>
<td>0.2905</td>
<td>13.4140</td>
<td>0.3042</td>
<td></td>
</tr>
<tr>
<td>0.9834</td>
<td>0.0484</td>
<td>0.4431</td>
<td>0.5704</td>
<td>0.6150</td>
<td>0.3443</td>
<td>0.9101</td>
<td>0.1804</td>
<td>0.3042</td>
<td>10.5376</td>
<td></td>
</tr>
</tbody>
</table>
Table A.4: A sample block $A_i$ of matrix $B^{-1}$ for inter-period linear demand case

\[
\begin{array}{cccccccccccc}
0.0460 & 0.0003 & 0.0013 & 0.0031 & 0.0013 & 0.0036 & 0.0006 & 0.0003 & 0.0001 & 0.0049 \\
0.0003 & 0.0498 & 0.0018 & 0.0010 & 0.0005 & 0.0017 & 0.0016 & 0.0046 & 0.0035 & 0.0002 \\
0.0013 & 0.0018 & 0.0645 & 0.0034 & 0.0030 & 0.0037 & 0.0010 & 0.0039 & 0.0013 & 0.0022 \\
0.0031 & 0.0010 & 0.0034 & 0.0598 & 0.0036 & 0.0019 & 0.0005 & 0.0023 & 0.0002 & 0.0029 \\
0.0013 & 0.0005 & 0.0030 & 0.0036 & 0.0648 & 0.0021 & 0.0003 & 0.0033 & 0.0015 & 0.0031 \\
0.0036 & 0.0017 & 0.0037 & 0.0019 & 0.0021 & 0.0507 & 0.0022 & 0.0015 & 0.0005 & 0.0017 \\
0.0006 & 0.0016 & 0.0010 & 0.0005 & 0.0003 & 0.0022 & 0.0639 & 0.0030 & 0.0035 & 0.0046 \\
0.0003 & 0.0046 & 0.0039 & 0.0023 & 0.0033 & 0.0015 & 0.0030 & 0.0549 & 0.0015 & 0.0009 \\
0.0001 & 0.0035 & 0.0013 & 0.0002 & 0.0015 & 0.0005 & 0.0035 & 0.0015 & 0.0671 & 0.0015 \\
0.0049 & 0.0002 & 0.0022 & 0.0029 & 0.0031 & 0.0017 & 0.0046 & 0.0009 & 0.0015 & 0.0527 \\
\end{array}
\]

Table A.5: Original model with linear demand-price relationship using CPLEX

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Profit</th>
<th>$D^*$</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>229.973</td>
<td>28.5117</td>
<td>0.0468003</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>2170.53</td>
<td>119.937</td>
<td>0.109201</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>18221.7</td>
<td>489.833</td>
<td>0.218401</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>65676.1</td>
<td>1153.81</td>
<td>0.577204</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>152777</td>
<td>2045.4</td>
<td>1.49761</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>289487</td>
<td>3116</td>
<td>3.32282</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>515924</td>
<td>4545.72</td>
<td>6.89524</td>
</tr>
</tbody>
</table>

Table A.6: Original model with linear demand-price relationship using LOQO

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Profit</th>
<th>$D^*$</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>229.973</td>
<td>28.5117</td>
<td>0.0156001</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>2170.53</td>
<td>119.937</td>
<td>0.0468003</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>18221.7</td>
<td>489.833</td>
<td>0.514803</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>65676.1</td>
<td>1153.81</td>
<td>1.57561</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>152777</td>
<td>2045.4</td>
<td>4.71123</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>289487</td>
<td>3116</td>
<td>23.2909</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>515924</td>
<td>4545.72</td>
<td>36.707</td>
</tr>
</tbody>
</table>

87
Table A.7: Optimal results for linear demand with small initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>D₀</th>
<th>Iterations</th>
<th>Profit</th>
<th>D*</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>25</td>
<td>5</td>
<td>229.973</td>
<td>28.5065</td>
<td>0.405603</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>100</td>
<td>5</td>
<td>2170.5</td>
<td>119.788</td>
<td>0.452403</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>400</td>
<td>5</td>
<td>18221.7</td>
<td>489.808</td>
<td>0.499203</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>900</td>
<td>6</td>
<td>65676</td>
<td>1153.67</td>
<td>1.10761</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>1600</td>
<td>6</td>
<td>152777</td>
<td>2045.28</td>
<td>2.21521</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>2500</td>
<td>7</td>
<td>289487</td>
<td>3115.99</td>
<td>3.49442</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>3600</td>
<td>7</td>
<td>515924</td>
<td>4545.67</td>
<td>6.48964</td>
</tr>
</tbody>
</table>

Table A.8: Optimal results for linear demand with medium initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Dᵣ</th>
<th>Iterations</th>
<th>Profit</th>
<th>D*</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>45.064</td>
<td>5</td>
<td>229.972</td>
<td>28.5043</td>
<td>0.405603</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>184.803</td>
<td>5</td>
<td>2170.52</td>
<td>119.912</td>
<td>0.436803</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>699.11</td>
<td>5</td>
<td>18221.6</td>
<td>489.894</td>
<td>0.577204</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>1568.56</td>
<td>6</td>
<td>65676</td>
<td>1153.88</td>
<td>1.07641</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>2830.73</td>
<td>7</td>
<td>152777</td>
<td>2045.42</td>
<td>2.12161</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>4443.89</td>
<td>8</td>
<td>289487</td>
<td>3116.09</td>
<td>3.21362</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>6387.79</td>
<td>8</td>
<td>515924</td>
<td>4545.78</td>
<td>5.27283</td>
</tr>
</tbody>
</table>

Table A.9: Optimal results for linear demand with large initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Dᵢ</th>
<th>Iterations</th>
<th>Profit</th>
<th>D*</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>65.1265</td>
<td>5</td>
<td>229.972</td>
<td>28.506</td>
<td>0.390003</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>269.603</td>
<td>5</td>
<td>2170.44</td>
<td>119.732</td>
<td>0.421203</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>699.11</td>
<td>5</td>
<td>18221.6</td>
<td>489.439</td>
<td>0.780005</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>1568.56</td>
<td>6</td>
<td>65676</td>
<td>1153.2</td>
<td>1.99681</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>2830.73</td>
<td>7</td>
<td>152777</td>
<td>2044.76</td>
<td>3.19483</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>4443.89</td>
<td>8</td>
<td>289487</td>
<td>3115.52</td>
<td>3.72842</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>6387.79</td>
<td>9</td>
<td>515924</td>
<td>4545.36</td>
<td>7.61285</td>
</tr>
</tbody>
</table>
Table A.10: Original model with inter-temporal linear demand-price relationship using CPLEX

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Profit</th>
<th>D*</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>2267.42</td>
<td>53.1426</td>
<td>0.095606</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>19312.8</td>
<td>245.357</td>
<td>0.218401</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>58562</td>
<td>531.278</td>
<td>0.936006</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>144701</td>
<td>952.786</td>
<td>2.54282</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>283371</td>
<td>1505.53</td>
<td>7.33205</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>475938</td>
<td>2147.73</td>
<td>11.5129</td>
</tr>
</tbody>
</table>

Table A.11: Optimal results for inter-temporal linear demand with small initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Iterations</th>
<th>Profit</th>
<th>D*</th>
<th>Time (Seconds)</th>
<th>L calculation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>7</td>
<td>2436.11</td>
<td>69.347</td>
<td>0.764405</td>
<td>0.0027</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>18492.9</td>
<td>242.472</td>
<td>1.21681</td>
<td>0.0824</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>12</td>
<td>61423.7</td>
<td>532.41</td>
<td>2.54282</td>
<td>0.833</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>14</td>
<td>139040</td>
<td>962.63</td>
<td>5.25723</td>
<td>4.6466</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>276899</td>
<td>1517.01</td>
<td>7.44125</td>
<td>16.2412</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>17</td>
<td>479313</td>
<td>2183.98</td>
<td>13.1353</td>
<td>47.3354</td>
</tr>
</tbody>
</table>

Table A.12: Optimal results for inter-temporal linear demand with medium initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Profit</th>
<th>D*</th>
<th>Time (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>2436.11</td>
<td>69.347</td>
<td>0.0624004</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>18492.9</td>
<td>242.472</td>
<td>0.234002</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>61423.7</td>
<td>532.41</td>
<td>0.936006</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>139040</td>
<td>962.63</td>
<td>2.58962</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>276899</td>
<td>1517.01</td>
<td>5.72524</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>479313</td>
<td>2183.98</td>
<td>10.7953</td>
</tr>
</tbody>
</table>

89
Table A.13: Optimal results for inter-temporal linear demand with large initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>Iterations</th>
<th>Profit</th>
<th>$D^*$</th>
<th>Time (Seconds)</th>
<th>L calculation time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>10</td>
<td>7</td>
<td>2267.42</td>
<td>53.1426</td>
<td>0.733205</td>
<td>0.0029</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>10</td>
<td>19312.8</td>
<td>245.357</td>
<td>1.23241</td>
<td>0.0823</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>12</td>
<td>58562</td>
<td>531.278</td>
<td>2.34001</td>
<td>0.7572</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>13</td>
<td>144701</td>
<td>952.786</td>
<td>4.63323</td>
<td>4.2676</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>15</td>
<td>283371</td>
<td>1505.54</td>
<td>10.3429</td>
<td>15.9747</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>17</td>
<td>475938</td>
<td>2147.73</td>
<td>19.9213</td>
<td>46.815</td>
</tr>
</tbody>
</table>

Table A.14: Optimal results for nonlinear demand with small initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>$D_s$</th>
<th>Profit</th>
<th>$D^*$</th>
<th>Time (Seconds)</th>
<th>Ori Profit</th>
<th>Ori Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>25</td>
<td>2418.07</td>
<td>41.2315</td>
<td>0.842405</td>
<td>2418.09</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>100</td>
<td>21941.7</td>
<td>182.4</td>
<td>0.748805</td>
<td>21944.4</td>
<td>0.0780005</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>400</td>
<td>164041</td>
<td>731.504</td>
<td>1.71601</td>
<td>164049</td>
<td>0.764405</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>900</td>
<td>516780</td>
<td>1542.46</td>
<td>3.32282</td>
<td>516813</td>
<td>2.34001</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>1600</td>
<td>2136960</td>
<td>2836.17</td>
<td>7.86245</td>
<td>2137050</td>
<td>7.64405</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>2500</td>
<td>3813610</td>
<td>4047.11</td>
<td>11.2321</td>
<td>3813710</td>
<td>25.6147</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>3600</td>
<td>7610150</td>
<td>6673.16</td>
<td>27.9866</td>
<td>7610320</td>
<td>37.6118</td>
</tr>
</tbody>
</table>

Table A.15: Optimal results for nonlinear demand with medium initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>$D_m$</th>
<th>Profit</th>
<th>$D^*$</th>
<th>Time (Seconds)</th>
<th>Ori Profit</th>
<th>Ori Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>62.5</td>
<td>2417.91</td>
<td>41.4182</td>
<td>0.733205</td>
<td>2418.09</td>
<td>0.0312002</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>220</td>
<td>21944.4</td>
<td>180.879</td>
<td>0.717605</td>
<td>21944.4</td>
<td>0.0624004</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1000</td>
<td>164048</td>
<td>727.878</td>
<td>0.904806</td>
<td>164049</td>
<td>0.670804</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>2070</td>
<td>516808</td>
<td>1534.61</td>
<td>1.60681</td>
<td>516813</td>
<td>2.27761</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>4000</td>
<td>2137050</td>
<td>2818.41</td>
<td>2.74562</td>
<td>2137050</td>
<td>10.3429</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>6250</td>
<td>3813700</td>
<td>4027.7</td>
<td>4.64883</td>
<td>3813710</td>
<td>17.0041</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>9000</td>
<td>7610320</td>
<td>6632.96</td>
<td>7.73765</td>
<td>7610320</td>
<td>31.325</td>
</tr>
</tbody>
</table>
Table A.16: Optimal results for nonlinear demand with large initial points through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>T</th>
<th>D_i</th>
<th>Profit</th>
<th>D^*</th>
<th>Time (Seconds)</th>
<th>Ori Profit</th>
<th>Ori Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5</td>
<td>87.7585</td>
<td>2417.9</td>
<td>41.469</td>
<td>0.686404</td>
<td>2418.09</td>
<td>0.0156001</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>394.008</td>
<td>21943.6</td>
<td>181.649</td>
<td>0.920406</td>
<td>21944.4</td>
<td>0.0624004</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>1579.55</td>
<td>164047</td>
<td>729.491</td>
<td>1.35721</td>
<td>164049</td>
<td>0.561604</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>3270.35</td>
<td>516799</td>
<td>1538.07</td>
<td>2.32441</td>
<td>516813</td>
<td>2.49602</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>6065.74</td>
<td>2136960</td>
<td>2832.21</td>
<td>5.81884</td>
<td>2137050</td>
<td>8.31485</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>8457.09</td>
<td>3813580</td>
<td>4043.6</td>
<td>9.18846</td>
<td>3813710</td>
<td>16.5517</td>
</tr>
<tr>
<td>60</td>
<td>60</td>
<td>14450.6</td>
<td>7609990</td>
<td>6679.16</td>
<td>31.5278</td>
<td>7610330</td>
<td>25.865</td>
</tr>
</tbody>
</table>

Table A.17: Solving time (Seconds) for linear demand with small number of time periods through iterative gradient-based algorithm

<table>
<thead>
<tr>
<th>N</th>
<th>Time-Small</th>
<th>Time-Medium</th>
<th>Time-Large</th>
<th>Time CPLEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0000</td>
<td>0.0560</td>
<td>0.0680</td>
<td>0.0560</td>
<td>0.0160</td>
</tr>
<tr>
<td>20.0000</td>
<td>0.0520</td>
<td>0.0600</td>
<td>0.0880</td>
<td>0.0400</td>
</tr>
<tr>
<td>30.0000</td>
<td>0.0800</td>
<td>0.0960</td>
<td>0.0680</td>
<td>0.0840</td>
</tr>
<tr>
<td>40.0000</td>
<td>0.0880</td>
<td>0.1280</td>
<td>0.0800</td>
<td>0.1600</td>
</tr>
<tr>
<td>50.0000</td>
<td>0.1160</td>
<td>0.1160</td>
<td>0.1040</td>
<td>0.2800</td>
</tr>
<tr>
<td>60.0000</td>
<td>0.1280</td>
<td>0.1320</td>
<td>0.1120</td>
<td>0.4560</td>
</tr>
<tr>
<td>80.0000</td>
<td>0.1400</td>
<td>0.1520</td>
<td>0.1480</td>
<td>0.7921</td>
</tr>
<tr>
<td>100.0000</td>
<td>0.1600</td>
<td>0.1880</td>
<td>0.1880</td>
<td>1.5081</td>
</tr>
</tbody>
</table>
Bibliography


