Algorithms for Decision Making

Asbjørn Nilsen Riseth

Trinity College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2018

To my family and friends.

Acknowledgements

I would like to thank my supervisors Jeff Dewynne and Chris Farmer for all the time they have devoted to working with me. They have helped me understand other viewpoints of my research and how to communicate new results. I have taken great pleasure in our frequent meetings and the range of technical and everyday discussions we have had.

My work is partially supported by the EPSRC Centre For Doctoral Training in Industrially Focused Mathematical Modelling (EP/L015803/1) in collaboration with dunnhumby Limited. The dunnhumby research team has provided valuable insight into the retail industry and suggested the pricing problems that motivated my work.

Abstract

We investigate algorithms for different steps in the decision making process, focusing on systems where we are uncertain about the outcomes but can quantify how probable they are using random variables. Any decision one makes in such a situation leads to a distribution of outcomes and requires a way to evaluate a decision. The standard approach is to marginalise the distribution of outcomes into a single number that tries in some way to summarise the value of each decision. After selecting a marginalisation approach, mathematicians and decision makers focus their analysis on the marginalised value but ignore the distribution. We argue that we should also be investigating the implications of the chosen mathematical approach for the whole distribution of outcomes.

We illustrate the effect different mathematical formulations have on the distribution with one-stage and sequential decision problems. We show that different ways to marginalise the distributions can result in very similar decisions but each way has a different complexity and computational cost. It is often computationally intractable to approximate optimal decisions to high precision and much research goes into developing algorithms that are suboptimal in the marginalised sense, but work within the computational budget available. If the performance of these algorithms is evaluated they are mainly judged based on the marginalised values, however, comparing the performance using the full distribution provides interesting information: We provide numerical examples from dynamic pricing applications where the suboptimal algorithm results in higher profit than the optimal algorithm in more than half of the realisations, which is paid for with a more significant underperformance in the remaining realisations.

All the problems discussed in this thesis lead to continuous optimisation problems. We develop a new algorithm that can be used on top of existing optimisation algorithms to reduce the cost of approximating solutions. The algorithm is tested on a range of optimisation problems and is shown to be competitive with existing methods.

Contents

1 Introduction **1**

1.1 Revenue management 2

1.1.1 Modelling demand 3

1.1.2 Forecasting and uncertainty 6

1.2 Outline of thesis 7

1.3 Statement of originality 7

2 One-stage optimisation **11**

2.1 Motivating example 12

2.2 Optimisation with random outcomes 14

2.2.1 Expected utilities 15

2.2.2 Mean-deviation problems 17

2.2.3 Nonlinear expectations 18

2.2.4 Argument by paradox 21

2.3 Different approaches lead to the same decision 21

2.3.1 Equivalence for the normal distribution 22

2.3.2 Finding equivalent mean-deviation formulations 23

2.4 Randomness and constraints 26

2.5 Multi-objective optimisation 27

2.5.1 Terminology 27

2.5.2 Approximating the Pareto front 31

2.5.3 Ordering decision points a priori 35

2.5.4 Approximating Pareto fronts with MultiJuMP 36

3 Control strategies for a discrete time pricing problem **39**

3.1 The pricing problem 41

3.1.1 Non-dimensionalisation of the system 43

3.1.2 The Bellman equation 43

3.1.3 A one-product example system 45

3.2 Suboptimal approximations 47

3.2.1 The Certainty Equivalent Control policy 47

3.2.2 Bellman and CEC parameter comparisons 50
3.2.3 Open-Loop Feedback Control policy 52
3.3 Discussion 56

4 Continuous-time pricing with diffusion models 57
4.1 Modelling demand and uncertainty 60
4.1.1 Non-dimensionalised system 62
4.2 Formulation of the decision problem 64
4.2.1 Parameter estimation 67
4.3 Solution to the deterministic system 68
4.3.1 Linear demand function 69
4.3.2 Exponential demand function 72
4.4 Impact of uncertainty 72
4.4.1 Asymptotic analysis 73
4.4.2 Example simulation 78
4.5 Extensions of the pricing problem 81
4.5.1 Other forms of uncertainty 81
4.5.2 Expected utility risk aversion 82
4.6 Discussion 83

5 Objective acceleration for unconstrained optimisation 85
5.1 Acceleration methods overview 86
5.2 Optimisation acceleration with O-ACCEL 88
5.2.1 Algorithm 90
5.2.2 O-ACCEL as a full orthogonalisation method (FOM) 91
5.3 Numerical experiments 93
5.3.1 Test problems from De Sterck 94
5.3.2 Experiment design 98
5.3.3 Performance profiles 99
5.3.4 The tensor optimisation problem from De Sterck 101
5.3.5 CUTEst test problems 102
5.4 Discussion 105
5.5 Tables of numerical results 106

6 Summary and further research 113

Bibliography 114

List of Figures

1.1 Decision making steps 2
1.2 Comparison of demand curves 4
1.3 Exponential demand function 5

2.1 Demand for three products 13
2.2 Probability distribution of profit 14
2.3 Comparison of the exponential and logarithmic utilities 16
2.4 Comparison between two decision maker models 25
2.5 Comparison between two decision maker models 25
2.6 Pareto concepts for two-dimensional problem 28
2.7 A disconnected Pareto front 28
2.8 Two Pareto fronts of competing objects for a retail example 30
2.9 An example Pareto front for two objectives 30
2.10 The Normal Boundary Intersection method 32
2.11 Approximations of a disconnected Pareto front 33
2.12 The NBI extension 34

3.1 The value function and corresponding optimal control function for the pricing problem 46
3.2 Simulations of the pricing system 47
3.3 Comparison of the CEC and Bellman policy functions 49
3.4 Sample profit distributions from following the Bellman and CEC policies 51
3.5 Relative profit L^2 difference between the CEC and Bellman policies . 53
3.6 Sample profit distributions from following the Bellman and OLFC policies 55

4.1 Evolutions of stock for a constant demand forecast 61
4.2 Sample paths of stock following the GBM model 63
4.3 Distribution of the relative error in estimating $G(t)$ 69
4.4 Example optimal, deterministic pricing function 70

4.5 Comparison of the optimal pricing function to the deterministic-case function 74
4.6 A zoomed-in comparison of the Bellman and deterministic-case functions 74
4.7 Numerically computed solutions of asymptotics problem 77
4.8 Statistics of the price paths from Example 4.3 79
4.9 Profit sample distribution comparing the deterministic-case pricing heuristic to the Bellman policy 80

5.1 Performance profiles for Problems A–G 101
5.2 Performance profiles for N-GMRES and O-ACCEL on Problems A–G 102
5.3 Performance profiles from the tensor optimisation test problem 103
5.4 Performance profiles for the CUTEst test problems 104

List of Tables

3.1 Statistics comparing the profits of the Bellman and CEC policies . . 54
3.2 Statistics comparing the profits of the Bellman and OLFC policies . . 55

4.1 Profit statistics comparing the Bellman and deterministic-case policies 80

5.1 Numerical results from the tensor optimisation test problem 103
5.2 Quantiles reporting f evaluations to reach tolerance in Problems A–C 107
5.3 Quantiles reporting f evaluations to reach tolerance in Problem D . . 108
5.4 Quantiles reporting f evaluations to reach tolerance in Problems E–G 109
5.5 Results from the CUTEst problems. 110
5.6 Results from the CUTEst tests. 111

Chapter 1

Introduction

We all want to make good decisions. The decision process requires understanding and prediction of systems that can involve complex interactions fraught with uncertainty. It is, therefore, a challenge for mathematicians to devise good practices to guide decision makers. This thesis contains a collection of topics that are relevant in decision making under uncertainty, primarily motivated by problems in the retail industry. We present algorithms for decision making that address different points in the decision making process. The mathematical approach to this process consists of five general steps, of which each chapter addresses a subset. The steps are summarised as follows.

1. Objective — statement of what the decision maker wants to achieve.
2. Model of the system — mathematical modelling of relevant aspects needed to achieve the objective.
3. Data assimilation — incorporation of available observations into the mathematical model.
4. Risk preference — mathematical modelling of the decision maker's risk preference.
5. Optimisation — algorithms and approximations required to evaluate different decisions.

As we gain more knowledge of the system and the impact of our decisions, the decision maker may refine their objective or we may choose to address some of the steps differently. This interactive process is illustrated in Figure 1.1.

The mathematical decision making process, as we have defined it, is general and can be applied to many decision problems. For a collection of examples and arguments

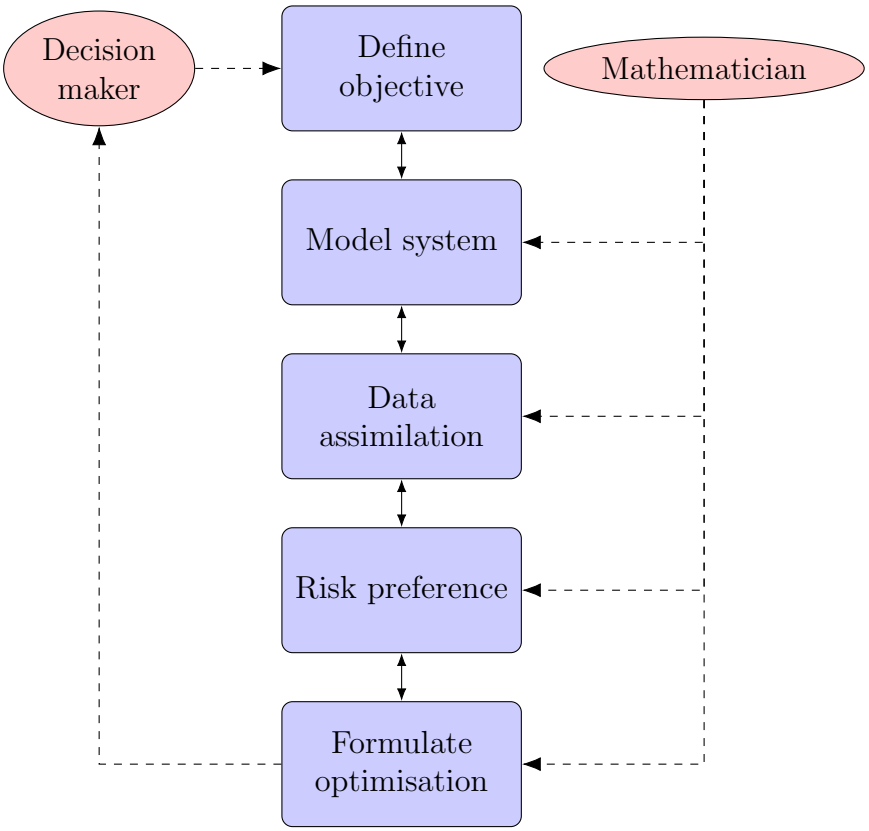


Figure 1.1: The steps of a mathematical approach to decision making. It can be an iterative approach, where the decision maker refines their goals after learning more about the system and the resulting decision impact.

for a mathematical approach to decisions in large organisations, see, for example Keeney and Raiffa (1976) and Farmer (2017). We use examples from the practice of revenue management to motivate the theory and illustrate the algorithmic approach to decision making that we present in this thesis.

1.1 Revenue management

The process of pricing products in order to control demand and maximise revenues has been undertaken for centuries (Smith, 1776, Book 1). In recent decades, data- and model-driven approaches have become increasingly popular in order to advise product managers and automate the process for companies. There are several success stories from early adopters, for example in the airline industry. American Airlines estimated in 1992 that the introduction of revenue management software had, over the preceding three years, contributed $500 million of additional revenue per year, and would continue to do so in the future (Smith et al., 1992). In an example from

Chilean retail, Bitran et al. (1998) report an expected revenue improvement of 7 % to 16 % after implementing model-driven strategies. This range of financial impact due to implementation of pricing and revenue management systems is further supported in other studies, see, for example, Phillips (2005, Ch. 1.2). For retailers with billions of pounds in revenues, a revenue improvement of less than a percent can be worth millions of pounds. In addition unsold items add up to thousands of tonnes of waste per year. So improvements in the control of supply or demand for products would be advantageous for both retailers and the environment.

The theory and practice of pricing and revenue management is multidisciplinary and attracts research from a wide variety of fields. Our focus is on some aspects of the modelling and optimisation procedures from a mathematical point of view. Phillips (2005) and Özer and Phillips (2012) present perspectives on the revenue management process prevalent in the social sciences. A notable resource with more mathematical focus, which cover both theoretical and practical perspectives, is provided by Talluri and van Ryzin (2006). Other mathematical approaches to pricing of retail products can, for example, be found in Caravenna et al. (2014).

The objectives we focus on in this thesis are aspects of sales, revenue, and profit targets that can be influenced by pricing decisions. At the core of such decisions are models connecting price and demand of products. In §1.1.1 we summarise the established approaches to demand modelling in revenue management, for reference in later chapters. The demand models often include unobservable parameters that are inferred from data. These models are used to forecast future demand, sometimes months into the future. The uncertainty in future events, and in estimates of model parameters, is described using random variables.

1.1.1 Modelling demand

We define demand as the quantity of a product that people are willing to buy, per time period, at a particular price. For the purpose of our work, demand may change over time, even if the price is constant. For example, time-dependent aspects of demand include periodic effects, such as seasonality and longer term trends. In addition to pricing decisions, retailers can use advertisements and offers to affect sales. Such demand control mechanisms often have a more significant impact on sales than price changes in their own right.

The relationship between price and demand is often described in a continuum fashion, rather than at discrete levels such as pounds or pence. We are interested

in retailers with large-scale sales and aggregate demand models where treating the demand as a continuum gives a reasonable approximation. In both academia and industry, simple demand curves are often used to model the price-demand relationship of products. See Talluri and van Ryzin (2006, Ch. 7.3) or Phillips (2005) for a discussion of the most popular demand curves. The curves are often chosen in order to simplify the parameter estimation procedure in practice, or to simplify the analysis for academic purposes.

Let $q(a) \geq 0$ denote the demand for a given product at price $a \geq 0$, holding other aspects such as time and competitor product prices constant. The following three demand curve families, where $q^{(1)}$ and $q^{(2)}$ are parameters, are often used in the revenue management literature.

$$q(a) = q^{(1)} - q^{(2)}a, \qquad q^{(1)}, q^{(2)} > 0, \qquad \text{linear demand,} \tag{1.1}$$

$$q(a) = e^{q^{(1)} - q^{(2)}a}, \qquad q^{(1)} \in \mathbb{R}, q^{(2)} > 0, \qquad \text{exponential demand, and} \tag{1.2}$$

$$q(a) = q^{(1)}a^{-q^{(2)}}, \qquad q^{(1)}, q^{(2)} > 0, \qquad \text{power demand.} \tag{1.3}$$

Note that these functions should only be considered as local approximations to demand. That is, we use these models to predict aggregate demand responses in a neighborhood of the current price. Care must be taken especially if one is to use the power demand function: The demand goes to infinity as price goes to zero, and revenues go to infinity as price goes to infinity if $q^{(2)} \in (0, 1)$. Figure 1.2 provides a comparison of the three demand curves which behave similarly close to the current price, $a = 1$.

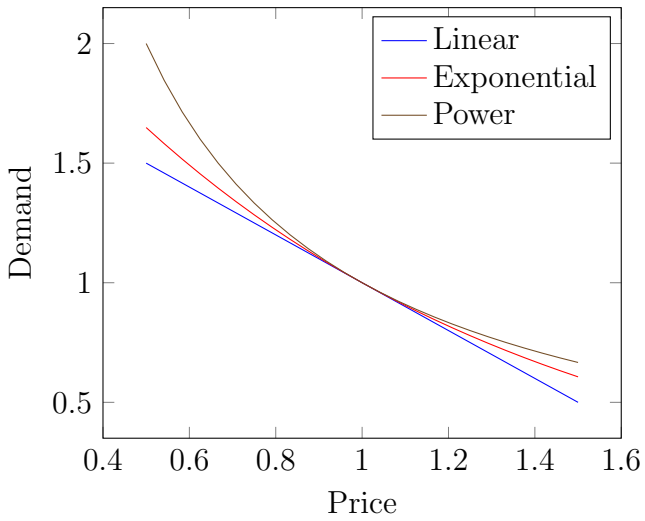


Figure 1.2: Comparison of demand curves. The units of price and time have been re-scaled so that $q(1) = 1$.

In cases where we believe that the price of one product affects the demand of another product, we include cross-effects in the demand functions. For products $j = 1, 2, \ldots, J$, denote their respective prices by a_j and their demands by $q_j(a)$, where $a = (a_1, a_2, \ldots, a_J)$. One way to extend the demand curves in (1.1) to (1.3) to a multi-product case is, according to Talluri and van Ryzin (2006),

$$q_j(a) = q_j^{(1)} - \sum_{l=1}^{J} q_{j,l}^{(2)} a_l, \qquad \text{linear demand,} \tag{1.4}$$

$$q_j(a) = \exp\left(q_j^{(1)} - \sum_{l=1}^{J} q_{j,l}^{(2)} a_l\right), \qquad \text{exponential demand, and} \tag{1.5}$$

$$q_j(a) = q_j^{(1)} \prod_{l=1}^{J} a_l^{-q_{j,l}^{(2)}}, \qquad \text{power demand.} \tag{1.6}$$

The valid domains of the parameters vary. For example, a linear demand model where we believe that increasing the price of product l has a positive effect on the demand of product j constrains $q_{j,l}^{(2)}$ to be negative. Figure 1.3 provides an example of demand for a given product when varying its price and the price of one competing product, using the exponential demand function.

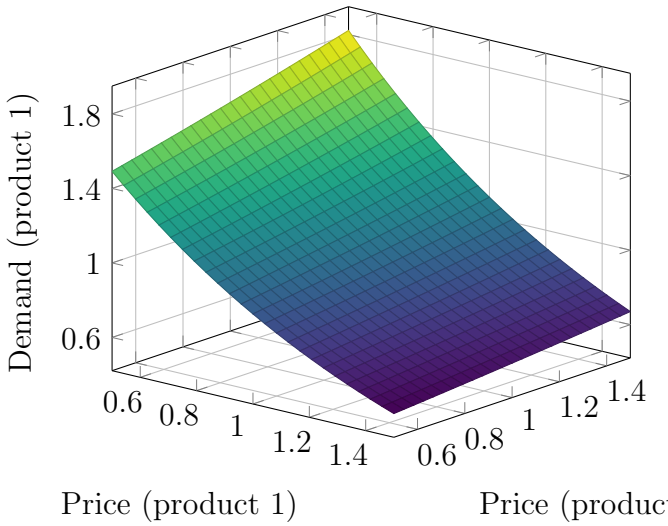


Figure 1.3: Exponential demand function for product 1, with one competing product $j = 2$. Units are rescaled so that $q_1([1, 1]) = 1$.

Remark 1.1. Note that both the exponential and power demand functions are unbounded: The demand of a product j goes to infinity if we take the price of some product $l \neq j$ to infinity, provided $q_{j,l}^{(2)} < 0$. One way that practitioners address this

issue is to define hard constraints on the prices, such as keeping the new price within 20 % of the current price. In Chapter 2 we address this issue in a different way by providing an alternative model for cross-effects.

1.1.2 Forecasting and uncertainty

Demand models, such as those presented above, have parameters that are product specific, and which must be estimated before forecasts of demand can be made. As multi-month forecasts may be required by the retailer, better approximations to future demand are perhaps possible with time-dependencies in the parameters. This estimation procedure is often guided by historic sales data collected by the retailer, using data assimilation techniques (Law et al., 2015, Riseth and Taylor-King, 2017) or time series analysis (Chatfield, 2004).

The models used to forecast future demand at particular prices carry a range of errors and uncertainty. First, estimates of parameter values are affected by uncertainty and numerical errors. Second, the model form may be inappropriate. Third, exogenous events impact demand outcomes in a more general way. A forecast is in essence an extrapolation of an evaluated model to a region of the input domain not yet experienced. We may still have an idea of what form the uncertainty this extrapolation takes, which we model in terms of random variables.

For example, consider a product for which we forecast the sales for two consecutive periods using the exponential demand model,

$$q(t, a) = e^{q^{(1)}(t) - q^{(2)}(t)a}, \qquad t = 1, 2. \tag{1.7}$$

To quantify the uncertainty in our model parameters, we associate with each of $q^{(1)}(t)$, $q^{(2)}(t)$ a probability distribution, not necessarily independent. Let $W(t, a)$ denote a random field that expresses the uncertainty associated with exogenous events and extrapolation outside the observed price region. We forecast the demand in period t to be $Q(t, a) = q(t, a)W(t, a)$, and the total forecasted demand is thus the random variable $Q(1, a_1) + Q(2, a_2)$, where a_t denotes the planned price in period t. Illustrations of the distributions of objectives and example models of $W(t, a)$ are provided in Chapters 2 to 4.

1.2 Outline of thesis

We present the chapters in this thesis by connecting them to the steps for making decisions that are listed on Page 1 and in Figure 1.1. In Chapter 2 we focus on the general discussion of the "objective", "risk preferences", and "optimisation". Chapter 2 contains much of the background material on formulating optimisation problems and can be considered an extension to this introductory chapter. We introduce one-stage optimisation problems with random outcomes and multi-objective optimisation. A comparison of risk preference methods shows that simple and more complicated methods can yield very similar decisions. Therefore, the choice of method should focus on implementational and computational aspects rather than theoretical arguments. In Chapter 3 and Chapter 4 we present aspects of the steps "model of the system", "risk preferences", and "optimisation" for sequential decision problems in discrete and continuous time. Chapters 3 and 4 provide examples of how simplified formulations, that ignore the randomness, result in marginalised outcomes that are practically equal to the values from decisions based on the original formulation. A further investigation of the distribution of outcomes shows that the simplified formulations result in the best outcomes for half of the realisations of the underlying random variables. In the remaining realisations, they result in much worse outcomes, which we interpret as an algorithmic decision that has impacted the modelled risk-attitude.

The first chapters are closely linked to the application of pricing in retail, whilst the penultimate chapter has a mathematical focus independent of particular applications. Chapter 5 presents a new algorithm to improve the optimisation step. The algorithm can be be combined with existing algorithms and we show, with several numerical experiments, that it reduces the computational cost of finding a solution with a prescribed accuracy. The general implications of the findings in this thesis are covered in Chapter 6.

1.3 Statement of originality

Each of Chapters 2 to 5 contains original work. The comparison of decisions from different decision-making models in Chapter 2 has not been conducted in the revenue management community, and the multiobjective optimisation package is written by the author. The dynamic pricing problems in Chapters 3 and 4 use a different combination of demand models, constraints, and profit functions from the existing literature. The investigation of the impact the optimal and suboptimal pricing policies

has on the profit distribution in these chapters is more extensive than the marginalised comparisons that are usually conducted. The solution and the asymptotic analysis of the Hamilton–Jacobi–Bellman (HJB) equations in Chapter 4 are original. The optimisation algorithm in Chapter 5 is an improvement of an existing idea, but the theory and numerical tests are all original.

Much of the work and intuition behind this thesis is a result of journal articles and software that the author has contributed to throughout the D.Phil. research period. The resulting articles and their relevance to the thesis are as follows.

- Riseth et al. (2018): *Erratum to "A risk-averse competitive newsvendor problem under the CVaR criterion"*;
  - accepted in the *International Journal of Production Economics*;
  - part of the work in Chapter 2.
- Riseth et al. (2017): *A comparison of control strategies applied to a pricing problem in retail*;
  - submitted to the *Proceedings of the Royal Society A*;
  - covers the majority of Chapter 3.
- Riseth (2017a): *Dynamic pricing in retail with diffusion process demand*;
  - submitted to the *IMA Journal of Management Mathematics*;
  - covers the majority of Chapter 4.
- Riseth (2017b): *Objective acceleration for unconstrained optimization*;
  - accepted in *Numerical Linear Algebra with Applications*;
  - covers the majority of Chapter 5.
- Mogensen and Riseth (2018): *Optim: A mathematical optimization package for Julia*;
  - published in the *Journal of Open Source Software*;
  - software used in all chapters.

- Riseth and Taylor-King (2017): *Operator Fitting for Parameter Estimation of Stochastic Differential Equations*;
  - submitted to the *Journal of Nonlinear Science*;
  - work on the data assimilation step of the decision process.
- van de Geer et al. (2018): *Dynamic Pricing and Learning with Competition: Insights from the Dynamic Pricing Challenge at the 2017 INFORMS RM & Pricing Conference*;
  - accepted in the *Journal of Revenue and Pricing Management*;
  - initial work to guide further research.

Chapter 2

One-stage optimisation

The aim of this chapter is to discuss approaches to modelling a decision problem in the form of a mathematical optimisation problem that computer algorithms can solve. We focus on one-stage optimisation problems here, which means that we do not consider any temporal structure in the order of decisions. The key concepts for defining an optimisation problem are

- a collection of allowed decisions, and
- an ordering between all decisions induced by their corresponding outcomes.

If there is a deterministic relationship between decisions $x \in \mathcal{X}$ and one, well-defined, objective $f : \mathcal{X} \to \mathbb{R}$, then we can model the decision problem as the optimisation problem[1]

$$\sup_{x \in \mathcal{X}} f(x). \tag{2.1}$$

The set $\mathcal{X}$ may, for example, be a subset of the real line, a finite-dimensional vector space, or a set of functions. If f attains its maximum on in the set $\mathcal{X}$, we can use the notation $\max_{x \in \mathcal{X}} f(x)$. In this chapter, however, we discuss three ways that decision problems with random outcomes may deviate from this setting.

1. If there is a non-deterministic relationship between a decision x and outcomes of the system.
2. If we cannot deterministically determine the allowed decisions $\mathcal{X}$ at the time we make the decision.
3. If there are multiple objectives.

[1]Maximisation problems can equivalently be solved as minimisation problems of the negative objective, $\inf_{x \in \mathcal{X}} -f(x)$. We change between the two depending on the application.

We argue that the second point arises when people try to add randomness to a deterministic model without reconsidering the modelling of the problem. One can view the first point as a decision problem with a possibly infinite number of objectives. In §2.2 to §2.4 we discuss common ways to model decision makers who face non-deterministic outcomes modelled with random variables, and in §2.5 we summarise the theory of multi-objective optimisation.

The different models for ordering decisions that we discuss in §2.2 have various strengths and weaknesses. The models that are appealing from a theoretical point of view may be more expensive to compute or approximate. There are situations where these methods can reach the same decisions as one another, with appropriately adjusted parameters, as we show in an example in §2.3. The message of this chapter is that we should think critically about the complexity of the system we want to control before formulating the mathematical optimisation problem.

2.1 Motivating example

We present a simple problem for pricing three retail products, in order to motivate the three deviations from the deterministic, well-defined setting of (2.1) that we consider in this chapter.

Consider a product manager who is responsible for three products. The manager can change the prices $x = (x_1, x_2, x_3)$ of the products in order to improve the profit or revenue for a given time period. To calculate the profit and revenue, we need to model the product demands as well as the costs associated with selling the products. For simplicity we assume that the quantities we work with have been non-dimensionalised.[2] Assume that the model for product demand $q(x) = (q_1(x), q_2(x), q_3(x))$ for the three products over the time period is deterministic and defined by the functions

$$q_1(x) = e^{-2x_1}(1 - e^{-2x_2/x_1}), \tag{2.2}$$

$$q_2(x) = 0.9e^{-1.8x_2}(1 - e^{-0.8x_1/x_2})(1 - e^{-8x_3/x_2}), \tag{2.3}$$

$$q_3(x) = 1.2e^{-2x_3}(1 - e^{-3x_1/x_3}), \tag{2.4}$$

respectively. The term e^{-2x_1} models the impact of price changes for product-one, keeping product-two's price constant. The term $1 - e^{-2x_2/x_1}$ models the proportion of product-one's demand that product-two takes when their relative price is x_2/x_1. To

[2]We illustrate the modelling and non-dimensionalisation process in more detail in Chapters 3 and 4.

illustrate the behaviour of the demand function, Figure 2.1 shows the total demand $q_1(x) + q_2(x) + q_3(x)$ when $x_3 = 1$.

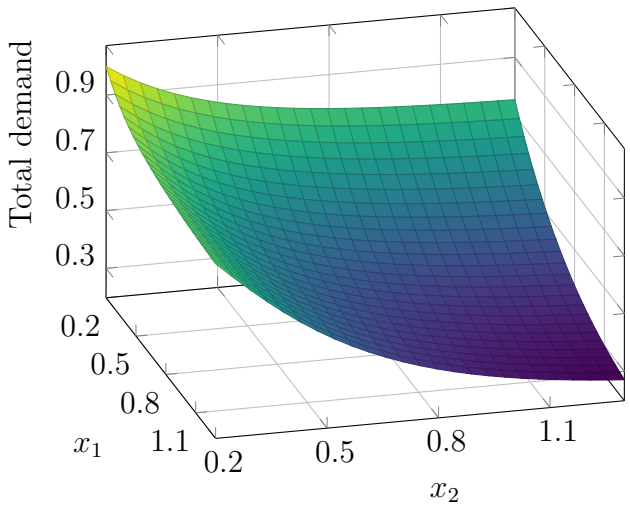


Figure 2.1: The total demand $q_1(x) + q_2(x) + q_3(x)$ shows the demand for products one and two with $x_3 = 1$.

Let us further model the costs associated with sales according to per-unit costs $y = (y_1, y_2, y_3)$. From the demand model we can define functions for the revenue $g(x)$ and profit $f(x, y)$ as

$$g(x) = \langle x, q(x) \rangle, \tag{2.5}$$

$$f(x, y) = \langle x - y, q(x) \rangle, \tag{2.6}$$

where $\langle \cdot, \cdot \rangle$ is the usual inner product on $\mathbb{R}^n$.

Retailers may need to decide the product price before they know the unit costs. We assume that we can model the costs with a random variable $Y \in \mathbb{R}^3$, based on prior knowledge and data. Throughout this chapter, we assume that Y is a log-normally distributed random variable with mean and covariance

$$m_Y = \begin{pmatrix} 0.5 \\ 0.5 \\ 0.65 \end{pmatrix}, \qquad C_Y = \begin{pmatrix} 2.5 & -0.75 & 0 \\ -0.75 & 2.5 & 0 \\ 0 & 0 & 4.2 \end{pmatrix} \times 10^{-3}. \tag{2.7}$$

Figure 2.2 shows the distribution of profit for three combinations of prices. There is no given outcome for each price, and the decision maker must therefore take into account all the outcomes that can follow each decision.

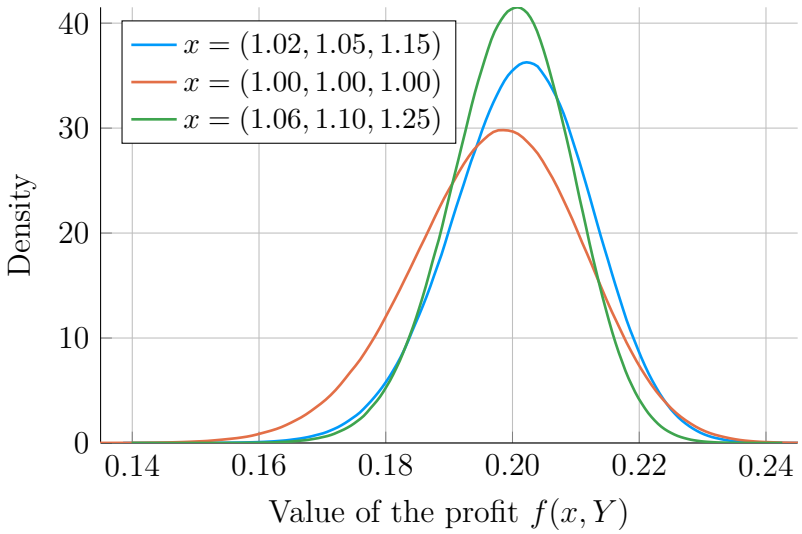


Figure 2.2: The probability distribution of profit $f(x, Y)$ for three sets of prices x. The decision maker should decide which decision is preferable by comparing several properties of the distributions.

The following three decision problems require further modelling of a decision maker's preferences in order to formulate a mathematical optimisation problem.

1. Maximise profit.
2. Maximise revenue and ensure that profit is positive.
3. In some way maximise both revenue and profit.

They are all examples of decision problems with random outcomes. The third problem is a decision problem with multiple objectives. The other examples can also be formulated as multi-objective optimisation problems.

2.2 Optimisation with random outcomes

In this section, we consider situations where there is one well-defined objective that depends on information we do not know with certainty at the time we make the decision. The theory discussed here can be used to formulate an optimisation problem for the example in the previous section, where the product manager wants to maximise the profit, which we do in §2.3.

Define a decision-space $\mathcal{X} \subset \mathbb{R}^n$ and a parameter-space $\mathcal{Y} \subset \mathbb{R}^k$. Let $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a function that is twice continuously differentiable on $\mathcal{X}$. The aim is to choose $x \in \mathcal{X}$ in order to maximise f. In the deterministic setting, we know the parameter $y \in \mathcal{Y}$

and can therefore find the optimal x by optimising $f(\cdot, y)$. Say there is uncertainty in the correct value of the parameter y that we model using a random variable Y, taking values in $\mathcal{Y}$. For a given $x \in \mathcal{X}$, $f(x, Y)$ is now a random variable. Thus, it is no longer straightforward to order two decisions x_1 and x_2 by comparing $f(x_1, Y)$ to $f(x_2, Y)$. Figure 2.2 illustrates the challenge; if one decision provides higher variability in the outcomes than another, we must decide whether the chance of the better outcomes outweigh the chance of the lesser outcomes. We present three classes of optimisation problems that attempt to address the randomness of $f(x, Y)$. They are *expected utilities*, *mean-deviation procedures*, and *nonlinear expectations*.

2.2.1 Expected utilities

One way to compare decisions in $\mathcal{X}$ is to consider a weighted sum, or integral, of all the realisations of the parameter in $\mathcal{Y}$. Expected utilities weigh the different realisations by combining the probability of an outcome with a measure of the benefit of f for that outcome. A family of such measures of benefit are utility functions.

Definition 2.1 (Utility function)**.** A function $u : \mathcal{S} \to \mathbb{R}$ on a set of outcomes $\mathcal{S} \subset \mathbb{R}$ defines a utility if it is non-decreasing.

Definition 2.2 (Convexity)**.** We say that $u(x)$ is *convex* if

$$u(\lambda x + (1-\lambda)x) \leq \lambda u(x) + (1-\lambda)u(x) \qquad \text{for } \lambda \in [0,1] \text{ and } x \in \mathcal{S}. \tag{2.8}$$

If $-u(x)$ is convex, then we say that u is *concave*. The properties of $u(x)$ give rise to different types of decision models. A *risk-seeking* decision model is one where $u(x)$ is strictly convex and a *risk-averse* decision model is one where $u(x)$ is strictly concave. If $u(x)$ is linear we have equality in (2.8) and the decision model is called *risk-neutral*.

The work on utility theory dates back to Cramer (1728) and D. Bernoulli (1738), who both developed a theory to address the St. Petersburg Paradox stated by Nicolaus I Bernoulli in 1713. He proposed a coin flipping lottery for which the expected value is infinite, and the paradox arises from the assumption, at the time, that the expected value is a fair price to pay to participate in a lottery. Both Cramer and D. Bernoulli argued that the marginal benefit of money decreases as the amount of money increases and proposed to model this with increasing but concave functions. An axiomatic approach to utility theory was devised by von Neumann and Morgenstern (1953), and a recent account of the formal theory of preferences using utilities is provided by Föllmer and Schied (2004, Ch. 2).

Example 2.1 (Utility functions)**.** Two popular utility functions are the exponential family $u_1(x) = (1 - e^{-\lambda x})/\lambda$ with $\lambda \in \mathbb{R}$ and the logarithmic utility $u_2(x) = \log(1 + x)$. The exponential utility is defined on $\mathcal{S} = \mathbb{R}$, whilst the logarithmic utility is only valid on $\mathcal{S} = [-1, \infty)$. Figure 2.3 shows the behaviour of the functions near the origin.

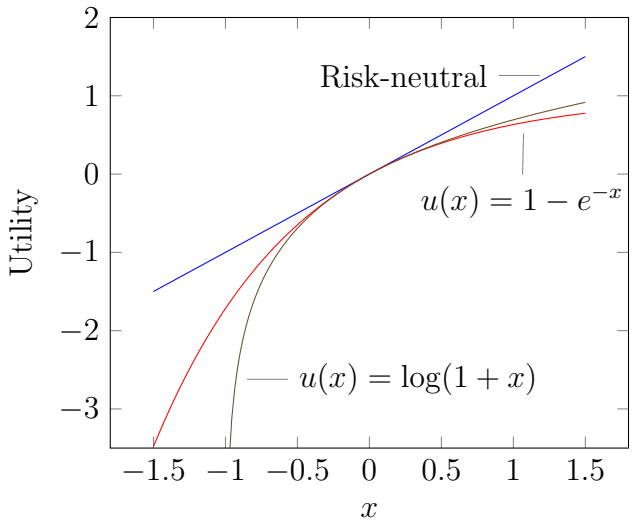


Figure 2.3: Comparison of the exponential and logarithmic utilities. As x approaches -1 from above, the utility to a logarithmic decision maker is modelled to be infinitely bad.

For a given utility function that models a decision maker's measure of benefit for the outcomes of f, one can define an ordering between random variables that decides which of two decisions $x_1, x_2 \in \mathcal{X}$ is best.

Definition 2.3 (Expected utility ordering)**.** A utility function $u(x)$ defines an ordering $\preceq$ between random variables X and Y,

$$X \preceq Y \Leftrightarrow \mathbb{E}[u(X)] \leq \mathbb{E}[u(Y)]. \tag{2.9}$$

The expectation operator $\mathbb{E}[X]$ is taken with respect to the probability distribution of the random variable X.

Remark 2.1. Utility orderings are preserved by positive affine transformations. That is, the ordering defined by $u(x)$ is equivalent to the ordering defined by $x \mapsto au(x) + b$ for $a > 0$ and $b \in \mathbb{R}$.

We can, therefore, formulate a well-defined optimisation problem by modelling the decision maker's attitude to risk with utility functions.

Optimisation problem 2.1 (Expected utility)**.** An optimal choice to maximise a given utility of f is the solution of

$$x^u = \arg\max_{x \in \mathcal{X}} \mathbb{E}[u(f(x, Y))]. \tag{2.10}$$

2.2.2 Mean-deviation problems

The expected utility approach considers the mean outcome of a choice. After one makes a decision $x \in \mathcal{X}$, the realisation of the parameter in $\mathcal{Y}$ only occurs once. If there is a large uncertainty in $f(x, Y)$, or, if the distribution is multimodal, the average value may not be a good representation for the realised value of f. It can, therefore, be useful to take into account some measure of deviation from the mean of f when x is chosen. The mean-deviation approach aims to balance the expected value of f with uncertainty represented by a deviation measure.

Example 2.2 (Mean-variance optimisation)**.** Markowitz (1952) proposed to optimise $f(x, Y)$ in a prescribed way that balances the expected value and variance of an outcome. We can create a single objective function that combines the expected value with a variance-penalty, weighted by some parameter $\lambda > 0$. The optimal decision in $\mathcal{X}$ is then defined as

$$x^\lambda = \arg\max_{x \in \mathcal{X}} \{\mathbb{E}[f(x, Y)] - \lambda \mathrm{Var}[f(x, Y)]\}. \tag{2.11}$$

This implicitly defines an ordering of random variables based on the functional $X \mapsto \mathbb{E}[X] - \lambda \mathrm{Var}[X]$. If f is nonlinear, then x^λ can be unstable to small changes in λ, an effect illustrated in Example 2.14 of §2.5. Small errors in determining the best λ to represent a decision maker can cause large changes in either of the objectives. Marler and Arora (2004) cover more robust methods.

There are other measures of deviation than variance and other ways to weigh the expectation and deviation than a linear combination. We can therefore generalise the mean-variance optimisation (2.11) to what we call a mean-deviation framework.

Definition 2.4 (Deviation measure)**.** Let $\mathbb{D}$ be a functional on a linear space of random variables which is closed under taking absolute values and contains the real numbers. Then $\mathbb{D}$ is a deviation measure if

1. $\mathbb{D}[C] = 0$ for any real number C, and
2. $\mathbb{D}[X] > 0$ for any non-constant random variable X.

Deviation measures on L^2 random variables were introduced by Rockafellar et al. (2006), however, the definition used in this thesis is less strict on the properties of $\mathbb{D}$.

Example 2.3 (Deviation measures)**.** The classic measure of deviation is the standard deviation. Define the L^p norm $\|X\|_p := (\mathbb{E}[|X|^p])^{1/p}$. The standard deviation is the particular case $p = 2$ of the deviation measures defined by the L^p-norms,

$$X \mapsto \|X - \mathbb{E}[X]\|_p. \qquad (L^p\text{-deviations}) \tag{2.12}$$

In some maximisation settings, one may worry about realisations of f that are larger than its mean, especially if the random variable is not symmetric. The lower semi-deviation measure only investigates worse-than-expected outcomes,

$$X \mapsto \mathbb{E}[\min(X - \mathbb{E}[X], 0)]. \qquad \text{(Lower semi-deviation)} \tag{2.13}$$

One can also combine two deviation measures in order to capture different aspects of a random variable. If $\mathbb{D}_1, \ldots, \mathbb{D}_k$ are k deviation measures and $\lambda_i > 0$, then the following is also a deviation measure,

$$\mathbb{D}[X] = \sum_{i=1}^{k} \lambda_i \mathbb{D}_i[X]. \qquad \text{(Weighted deviations)} \tag{2.14}$$

Optimisation problem 2.2 (Mean-deviation)**.** An optimal decision that maximises the expected value of f, whilst minimising the deviation $\mathbb{D}$ of f, is a solution to the bi-objective optimisation problem

$$\max_{x \in \mathcal{X}} \{(\mathbb{E}[f(x, Y)],\ -\mathbb{D}[f(x, Y)])\}. \tag{2.15}$$

For a given deviation measure that models the decision maker's need for stability, we can formulate a bi-objective optimisation problem. We can approach the bi-objective problem from a multi-objective optimisation theory point of view and do so in §2.5. The review by Marler and Arora (2004) covers some of the theory, discusses what optimality means, and introduces approaches to solve such problems.

2.2.3 Nonlinear expectations

A general framework that covers most of the remaining models in optimisation under uncertainty makes use of a class of functionals called nonlinear expectations. The term nonlinear expectation is used by Peng (2010) to generalise stochastic calculus

to distributional uncertainty. The theory was inspired by work in finance for the study of financial positions using risk measures, see, for example, Föllmer and Schied (2004, Ch. 4). Risk measures are used in a minimisation setting related to losses, but we use the term nonlinear expectations in the maximisation setting. The examples of nonlinear expectations $\mathbb{U}$ considered in this section give rise to well-known risk measures ρ by using the mapping $\rho(X) = -\mathbb{U}[X]$.

Definition 2.5 (Nonlinear expectation)**.** Let $\mathbb{U}$ be a functional on a linear space of random variables which is closed under taking absolute values and contains the real numbers. Then $\mathbb{U}$ is a nonlinear expectation if

1. $\mathbb{U}[C] = C$ for any real number C, and
2. $\mathbb{U}[X] \leq \mathbb{U}[Y]$ whenever $X \leq Y$ almost surely.[3]

Note that, contrary to the usual interpretation of the English prefix "non", the linear expectation $\mathbb{E}$ also satisfies the properties of a nonlinear expectation.

A more restricted class of functionals called superlinear expectations relate to the coherent risk measures from mathematical finance.

Definition 2.6 (Superlinear expectation)**.** A nonlinear expectation $\mathbb{U}$ is superlinear if

3. $\mathbb{U}[X + Y] \geq \mathbb{U}[X] + \mathbb{U}[Y]$ for each X, Y, and
4. $\mathbb{U}[\lambda X] = \lambda \mathbb{U}[X]$ for $\lambda \geq 0$.

If $\mathbb{U}$ is a nonlinear expectation with $\mathbb{U}[X] < \mathbb{E}[X]$ for all non-constant X, then one can construct a deviation measure with the functional

$$\mathbb{D}[X] = \mathbb{E}[X] - \mathbb{U}[X]. \tag{2.16}$$

Further connections between nonlinear expectations and deviation measures are discussed by Rockafellar and Uryasev (2013).

Example 2.4 (Nonlinear expectations)**.** With $\lambda > 0$, the functional

$$\mathbb{U}_\lambda[X] = -\frac{1}{\lambda} \log \mathbb{E}[e^{-\lambda X}] \tag{2.17}$$

[3] $X \leq Y$ almost surely if $\mathbb{P}(\{\omega \mid X(\omega) \leq Y(\omega)\}) = 0$.

is called an entropic nonlinear expectation. It is connected to the exponential utility function $u_\lambda(x) = (1 - e^{-\lambda x})/\lambda$, since

$$\mathbb{U}_\lambda[X] = u_\lambda^{-1}(\mathbb{E}[u_\lambda(X)]). \tag{2.18}$$

The worst-case nonlinear expectation on the sample space Ω, defined by

$$X \mapsto \inf_{\omega \in \Omega} X(\omega), \tag{2.19}$$

is superlinear and connects optimisation of nonlinear expectations to robust optimisation (Ben-Tal et al., 2009). If there is an attached probability space on Ω, we use the essential infimum

$$\operatorname*{ess\,inf}_{\omega \in \Omega} X(\omega) = \sup\{x \mid \mathbb{P}(X < x) = 0\}. \tag{2.20}$$

The quantile operator

$$q_\gamma[X] = \inf\{x \mid \mathbb{P}(X \leq x) \geq \gamma\}, \tag{2.21}$$

with $\gamma \in [0, 1]$, satisfies the properties of a nonlinear expectation. Another name for the quantile operator is Value at Risk (Föllmer and Schied, 2004). The operator is not necessarily concave, which has implications for the resulting optimisation problem. A concave lower bound on the quantile operator is the lower super-quantile,

$$\mathbb{U}_\gamma[X] = \frac{1}{\gamma} \int_0^\gamma q_t[X]\, dt, \tag{2.22}$$

which defines a superlinear expectation for $\gamma \in (0, 1]$. The risk measures related to the lower super-quantile operator are also referred to as the Expected Shortfall, Average Value at Risk, and Conditional Value at Risk (Artzner et al., 1999, Rockafellar and Uryasev, 2002, Föllmer and Schied, 2004, Rockafellar and Uryasev, 2013). Their definitions all coincide with (2.22) when the cumulative distribution function of X is continuous, however, in the general case they handle discontinuities differently.

Optimisation problem 2.3 (Nonlinear expectation)**.** An optimal choice to maximise a nonlinear expectation of f is given by

$$x^{\mathbb{U}} = \arg\max_{x \in \mathcal{X}} \mathbb{U}[f(x, Y)]. \tag{2.23}$$

Example 2.5 (Lower super-quantile)**.** Following Ben-Tal and Teboulle (2007), we can reformulate the lower super-quantile optimisation problem arising from (2.22) by

introducing an auxiliary variable $\eta \in \mathbb{R}$.

$$x^{\gamma}_{sq} = \underset{x \in \mathcal{X}, \eta \in \mathbb{R}}{\arg\max} \left\{ \eta - \frac{1}{\gamma} \mathbb{E}[\max(\eta - f(x, Y), 0)] \right\}. \tag{2.24}$$

Note that the objective is not necessarily smooth, due to the $\max(\eta - f(x, Y), 0)$ term. The theory of non-smooth optimisation can address this but may require sophisticated mathematical techniques, as exemplified by Kouri and Surowiec (2016).

2.2.4 Argument by paradox

The use of paradoxes has been a popular way to motivate different decision making models. We have mentioned the St. Petersburg paradox as a motivation to move from decisions purely made by expected values to using expected utilities. Some proponents of expected utility theory point out that mean-variance formulations for a particular portfolio allocation problem cannot describe the behaviour of a "rational" decision maker, exemplified with the so-called Borch paradox. A suggested remedy looks at modelling the mean-variance approach as a quadratic utility function. Two arguments against expected utility theory are the Allais and Ellsberg paradoxes that point out ways that people do not behave according to the axioms of utility theory.

We do not state the four paradoxes here but refer the reader to Ellsberg (1961), Föllmer and Schied (2004), and Johnstone and Lindley (2013). Our point is to note that the paradoxes pick particular cases that highlight when a particular decision model fails. Instead of arguing for a universal model of how humans (ought to) behave, we argue that we should assess which models are suitable for each type of decision problem we address.

2.3 Different approaches lead to the same decision

Can the mean-deviation formulation give rise to the same decision as the expected utility or nonlinear expectation methods? In this section we argue that there are cases where they do, focusing on the mean and standard deviation, exponential utility, and lower super-quantile. Our purpose is to highlight that one should investigate the structure of a problem and choose among the computationally tractable options based on their practical implications instead of solely focusing on theoretical properties of the different formulations.

The exponential utility and lower super-quantile are often more costly to compute or approximate than the mean and variance, and require a full description of the distribution of Y. Say a given nonlinear expectation $\mathbb{U}$, or utility u, implies that the decision x^* is optimal. Further, assume that we can get a decision close to x^* from a bi-objective formulation using mean and standard deviation. The mean-deviation formulation may then be preferable in practical settings. This is, for example, illustrated in the articles by Kouri and Surowiec (2016) and Alexanderian et al. (2017), who address optimisation problems where each value $y \in \mathcal{Y}$ requires the solution to partial differential equations. The lower super-quantile approach of Kouri and Surowiec (2016) requires the authors to draw samples from $\mathcal{Y}$ and find ways to smooth the objective, whilst Alexanderian et al. (2017) approximate mean and variance with Taylor series and do not need to sample from $\mathcal{Y}$.

The use of super-quantiles in revenue management has become popular over the last eleven years (Wu et al., 2014, Xue et al., 2015, Zhou et al., 2008, Ahmed et al., 2007). All of these articles argue for super-quantiles and dismiss optimisation formulations that use the mean and standard deviation on the grounds that they penalise better-than-expected outcomes. Distributions of profit in revenue management do in practice have a more significant lower tail than upper tail, due to inventory restrictions and fixed costs, which alleviates such problems. None of these references discusses to what extent super-quantiles change the optimal decisions for their presented problems compared to a mean-deviation approach. Choi et al. (2011) dismiss the mean-deviation problem by formulating it in terms of the weighted sum of mean and variance, and then argue that the units of these objects do not coincide.

2.3.1 Equivalence for the normal distribution

Fix the distribution of a random variable Y. If $f(x, Y)$ is normally distributed for each value of $x \in \mathcal{X}$, then the properties of the random variable are fully described by the mean and variance functions $x \mapsto \mathbb{E}[f(x, Y)]$ and $x \mapsto \mathrm{Var}[f(x, Y)]$. This indicates that the approaches above implicitly define a choice of ordering for the bi-objective mean–standard-deviation optimisation problem.

Example 2.6 (Lower super-quantile)**.** Let $X = \mathbb{E}[X] + \sqrt{\mathrm{Var}[X]}Z$ for $Z \sim \mathcal{N}(0, 1)$. Then the γ-quantile of X is $q_\gamma[X] = \mathbb{E}[X] + \sqrt{\mathrm{Var}[X]}q_\gamma[Z]$. Thus, the lower super-quantile $\mathbb{U}_\gamma$ of X depends only on its mean and variance, with

$$\mathbb{U}_\gamma[X] = \mathbb{E}[X] + \sqrt{\mathrm{Var}[X]}\mathbb{U}_\gamma[Z]. \tag{2.25}$$

Note that $\mathbb{U}_\gamma[Z] \leq 0$ for each $\gamma \in (0, 1]$, with equality only when $\gamma = 1$. The lower super-quantile optimisation problem is, therefore, a special case of the mean-deviation optimisation problem with $\mathbb{D}[X] = \sqrt{\text{Var}[X]}$, using the weighted sum ordering from (2.11) and $\lambda = -\mathbb{U}_\gamma[Z]$.

Example 2.7 (Exponential utility)**.** Let $X = \mathbb{E}[X] + \mathbb{D}[X]Z$, as in Example 2.6. Then its exponential utility u with parameter μ is

$$u(X) = 1 - e^{-\mu(\mathbb{E}[X]+\mathbb{D}[X]Z)}. \tag{2.26}$$

If $f(x, Y)$ is normally distributed, the expected utility optimisation for u is equivalent to solving

$$\min_{x\in\mathcal{X}} e^{-\mu\mathbb{E}[f(x,Y)]}\mathbb{E}\left[e^{-\mu\mathbb{D}[f(x,Y)]Z}\right]. \tag{2.27}$$

In the context of multi-objective optimisation, this arises from a particular choice of ordering of the mean-standard deviation problem (Marler and Arora, 2004).

The review by Nadarajah et al. (2014) on estimation methods for lower super-quantiles illustrates how people use the first two moments of a random variable to estimate nonlinear expectations for more generic distributions.

2.3.2 Finding equivalent mean-deviation formulations

Consider the decisions that arise from the following optimisation problems of mean-deviation, exponential utility, and lower super-quantile $\mathbb{U}_\gamma$.

$$x^\lambda_{md} = \arg\max_{x\in\mathcal{X}}\{\mathbb{E}[f(x, Y)] - \lambda\sqrt{\text{Var}[f(x, Y)]}\} \qquad \lambda \geq 0, \tag{2.28}$$

$$x^\mu_{eu} = \arg\max_{x\in\mathcal{X}}\{\mathbb{E}[1 - e^{-\mu f(x,Y)}]\} \qquad \mu > 0, \tag{2.29}$$

$$x^\gamma_{sq} = \arg\max_{x\in\mathcal{X}}\{\mathbb{U}_\gamma[f(x, Y)]\} \qquad \gamma \in (0, 1]. \tag{2.30}$$

We want to understand whether there are triples (λ, μ, γ) that give rise to the same decisions, $x_{md} = x_{eu} = x_{sq}$, and investigate this in the following way. For a given value $\mu > 0$ or $\gamma \in (0, 1]$, find the minimisers $\lambda(\mu)$ and $\lambda(\gamma)$ of

$$\min_{\lambda\geq 0}\left\{\frac{\|x^\lambda_{md} - x^\mu_{eu}\|_2}{\|x^\mu_{eu}\|_2}\right\} \quad \text{and} \quad \min_{\lambda\geq 0}\left\{\frac{\|x^\lambda_{md} - x^\gamma_{sq}\|_2}{\|x^\gamma_{sq}\|_2}\right\}. \tag{2.31}$$

If the denominators are zero, we minimise the absolute differences instead. If the minimum is close to zero, it is an indication that either method can be used to model the risk-preferences of a particular decision maker. Figure 2.4 shows the result of such an investigation, and for the particular example considered the differences are small. In general, it is likely to be examples where the choice of optimisation formulation matters for the decision. We remind the reader that the purpose of this section is to highlight that one should investigate the structure of a problem and choose among the computationally tractable options based on their practical implications instead of solely focusing on theoretical properties of the different formulations.

Example 2.8 (Pricing problem)**.** Let us consider the first decision problem in the retail example in §2.1, where the product manager wants to maximise profit $f(x, Y)$, but only knows the distribution of Y. The mean m_Y and covariance matrix C_Y of Y are defined in (2.7).

Note that the mean and standard deviation of the profit are

$$\mathbb{E}[f(x, Y)] = \langle x - m_Y, q(x) \rangle, \qquad \text{Var}[f(x, Y)] = \langle q(x), C_Y q(x) \rangle. \tag{2.32}$$

Thus, the mean-deviation method with $\mathbb{D}[X] = \sqrt{\text{Var}[X]}$ has a closed-form objective and one only needs information about the two first moments of Y to solve the mean-deviation problem (2.28). The solution to the expected utility and lower super-quantile problems (2.29) and (2.30) require that we approximate the objectives, for example, with Monte Carlo.[4] The super-quantile method also needs an extra smoothing step to handle the $\max(\eta - f(x, Y), 0)$ term of (2.24), introducing larger costs and implementational complexity.

We solve the optimisation problems in (2.31) with uniformly spaced values $\mu \in [0, 104]$ and $\gamma \in [1/100, 1]$. The numerical approximations to these problems have been implemented using the Julia programming language (Bezanson et al., 2017). For the inner optimisation problems (2.28) to (2.30) we use the mathematical programming modelling package JuMP (Dunning et al., 2017) with the Ipopt solver (Wächter and Biegler, 2006). For the outer problems in (2.31) we use the package Optim (Mogensen and Riseth, 2018) with the O-ACCEL algorithm that we present in Chapter 5.

Figures 2.4 and 2.5 show the corresponding values $\lambda(\mu)$ and $\lambda(\gamma)$, together with the difference in the objectives. The values of λ are stable to small changes in μ and γ, and the reported price differences are unlikely to have an effect when the products are priced in-store.

[4] For a review of Monte Carlo methods, see (Caflisch, 1998).

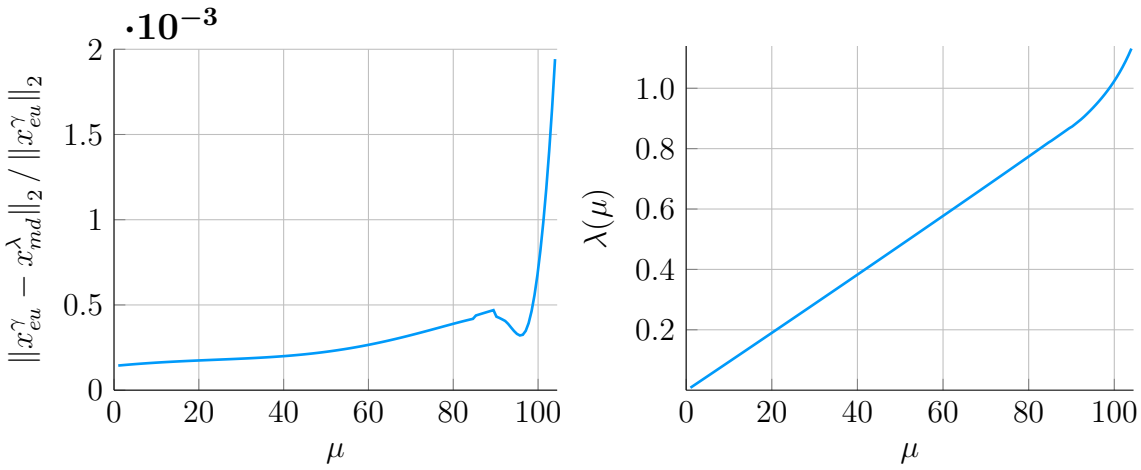


Figure 2.4: The left plot shows the relative difference between the decision made by the mean-deviation and exponential utility methods defined in (2.28) to (2.30), using the corresponding (μ, λ)-pairs from the right plot. The values are obtained by solving (2.31). The drop in the value of the left plot relates is due to projecting the three-dimensional difference vector in $\mathcal{X}$ down to one dimension.

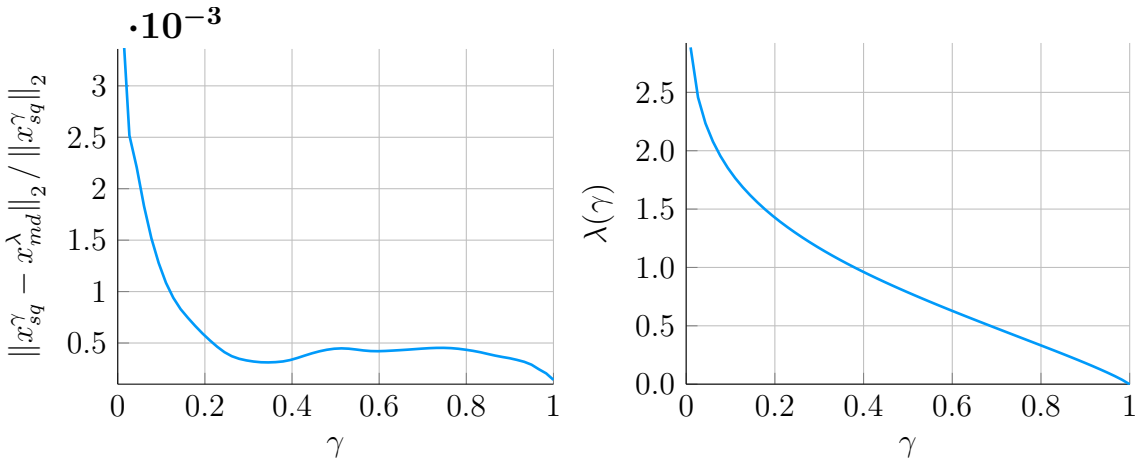


Figure 2.5: The left plot shows the relative difference between the decision made from the mean-deviation and super-quantile methods defined in (2.28) to (2.30), using the corresponding (γ, λ)-pairs from the right plot. The values are obtained by solving (2.31).

2.4 Randomness and constraints

We briefly discuss the situation arising from the second decision problem posed in §2.1, where the decision maker wants to maximise revenue and ensure that the profit is positive.

Introduce a deterministic objective $g : \mathcal{X} \to \mathbb{R}$. In the deterministic setting, with a known parameter $y \in \mathcal{Y}$, one often considers problems of the form

$$\min_{x \in \mathcal{X}} \{g(x) \mid f(x, y) \geq f_{\min}\}, \quad \text{for some } f_{\min} \in \mathbb{R}. \tag{2.33}$$

Then $f \geq f_{\min}$ is considered a constraint, which restricts the decision space to $\{x \in \mathcal{X} \mid f(x, y) \geq f_{\min}\}$. When there is uncertainty about the correct parameter value, this set is not defined at the time of the decision but depends on future knowledge of the parameter.

In the engineering community, it is popular to reinterpret such non-deterministic "constraints" as deterministic via expected utilities, mean-deviation methods or nonlinear expectations (Rockafellar, 2007, Rockafellar and Royset, 2015). Such a reinterpretation defines an acceptance set $\mathcal{A}$ which represents what a decision maker deems to be feasible in a risk-adjusted way and results in the optimisation problem $\min_{x \in \mathcal{A}} g(x)$. The acceptance set approach was one of the motivations for a formal framework for risk measures introduced by Artzner et al. (1999).

Example 2.9 (Acceptance sets)**.** For Y a random variable taking values in $\mathcal{Y}$, popular choices of acceptance sets for the constraint $f(x, Y) \geq f_{\min}$ are

$$\mathcal{A} = \{x \in \mathcal{X} \mid \mathbb{E}[f(x, Y)] \geq f_{\min}\}, \qquad \text{(Expectation)} \tag{2.34}$$

$$\mathcal{A} = \{x \in \mathcal{X} \mid \operatorname*{ess\,inf}_{\omega \in \Omega} f(x, Y(\omega)) \geq f_{\min}\}, \qquad \text{(Worst-case)} \tag{2.35}$$

$$\mathcal{A} = \{x \in \mathcal{X} \mid \mathbb{P}(f(x, Y) \geq f_{\min}) \geq \epsilon\}, \text{ given } \epsilon \in (0, 1). \qquad \text{(Quantile)} \tag{2.36}$$

The acceptance set approach treats what was previously considered a hard constraint in terms of marginalised values that allow decisions $x \in \mathcal{X}$ that may violate the originally intended constraint for some values of the random variable Y. We believe that a better approach is to address this properly by modelling the trade-off between $g(x)$ and the outcomes of $f(x, Y)$, which is a multi-objective optimisation approach.

2.5 Multi-objective optimisation

Competing objectives often appear in a decision process, and the chapter has already made multiple references to multi-objective optimisation. Assume there are m competing objectives $f_1, \ldots, f_m : \mathcal{X} \to \mathbb{R}$ that we wish to minimise[5] over the decision space $\mathcal{X}$. The two main challenges that arise are how to understand the trade-offs between the objectives, and to define an ordering between decisions $x \in \mathcal{X}$. One approach to the first challenge is to find the Pareto front, indifference curve, or efficient frontier, of the objectives, which we cover in §2.5.2. The second challenge is more difficult, as it involves modelling the decision maker's preferences. The concept of optimality that we present here is due to Edgeworth (1881) and Pareto (1906). Keeney and Raiffa (1976) present multi-objective theory focusing on decision making in business from a utility theoretical point of view. For a mathematical discussion of multi-objective optimisation, see, for example, Jahn (2011).

Relevant concepts for multi-objective optimisation are defined in §2.5.1. The main contribution from this section is a software package named MultiJuMP that lets users formulate multi-objective optimisation problems. An example in §2.5.4 shows how it is used to create the data for the figures in this section. The package approximates the Pareto front with three methods: Weighted sums, objective constraints, and the Normal Boundary Intersection method covered in §2.5.2. A broader range of approaches to multi-objective optimisation can be found in the survey by Marler and Arora (2004). The multi-objective ordering problem is discussed in §2.5.3, where it is argued that an approximation of a decision maker's preferences for multiple objectives can be made by combining the objectives' utilities.

2.5.1 Terminology

A key concept for multi-objective optimisation is the partial ordering of outcomes known as Pareto dominance. If a decision leads to an outcome that is not dominated by any other feasible outcome, it is optimal in the Pareto sense. Figure 2.6 and Figure 2.7 provide visual representations of the multi-objective concepts for two-dimensional problems $F : \mathcal{X} \to \mathbb{R}^2$.

[5]We choose to work within a minimisation-framework for the multi-objective theory in order to follow the standards in optimisation literature. Any objective f_i that is to be maximised can be redefined by $f_i \leftarrow -f_i$.

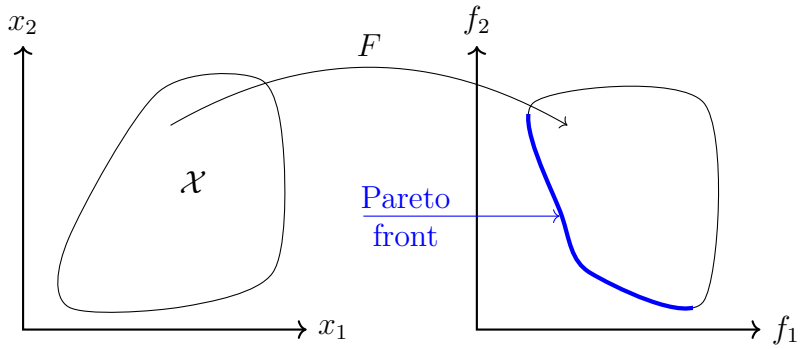


Figure 2.6: Pareto concepts for a mapping $F(x) = (f_1(x), f_2(x))$. The Pareto front is visualised by the bold line segment on the image $F(\mathcal{X})$.

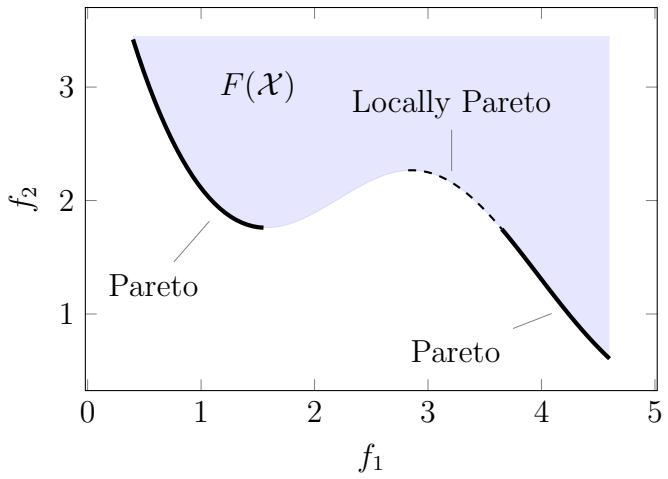


Figure 2.7: A disconnected Pareto front of a continuous function F that also admits locally Pareto optimal points.

Definition 2.7 (Pareto optimality)**.** A point $x \in \mathcal{X}$ is Pareto optimal for the problem

$$\min_{x \in \mathcal{X}} \{f_1(x), \ldots, f_m(x)\} \tag{2.37}$$

if there is no $x^* \in \mathcal{X}$ such that

$$f_i(x^*) \leq f_i(x) \qquad \text{for all } i = 1, \ldots, m, \text{ and} \tag{2.38}$$

$$f_j(x^*) < f_j(x) \qquad \text{for at least one } j \in \{1, \ldots, m\}. \tag{2.39}$$

Pareto optimality, therefore, means that we cannot decrease the value of one objective without increasing another. The point x is locally Pareto optimal if it is Pareto optimal in some neighbourhood around x. As such, a Pareto point is also a local Pareto point, but the converse is not necessarily true. Figure 2.7 shows an example of both Pareto and locally Pareto optimal points.

Definition 2.8 (Pareto front)**.** Define $F(x) = (f_1(x), \ldots, f_m(x))$. The Pareto front, or efficient frontier $\mathcal{P} \subset \mathbb{R}^m$, is the set of Pareto optimal objective values,

$$\mathcal{P} = \{F(x) \mid x \text{ is Pareto optimal}\}. \tag{2.40}$$

Example 2.10 (Retail application)**.** Consider the model from the retail problem described in §2.1. The Pareto front can be used in the retail setting in order to understand the trade-offs between different objectives. We consider two situations, both illustrated in Figure 2.8. First, say the manager wants to understand the balance between maximising expected profit and minimising the profit standard deviation. Second, the Pareto front can tell how much an increase in revenue decreases the expected profit.

Definition 2.9 (Convexity)**.** The Pareto-front is said to be convex if the following set is convex.

$$\overline{\mathcal{P}} = \{t \in \mathbb{R}^m \mid F(x) \leq t \text{ for some feasible } x\}. \tag{2.41}$$

Example 2.11 (Convexity)**.** The pricing problems in Example 2.10 represent convex Pareto fronts. An example of a non-convex Pareto front is shown in Figure 2.7.

Algorithms for approximating the Pareto front often make use of particular points describing extremal values of the objectives.

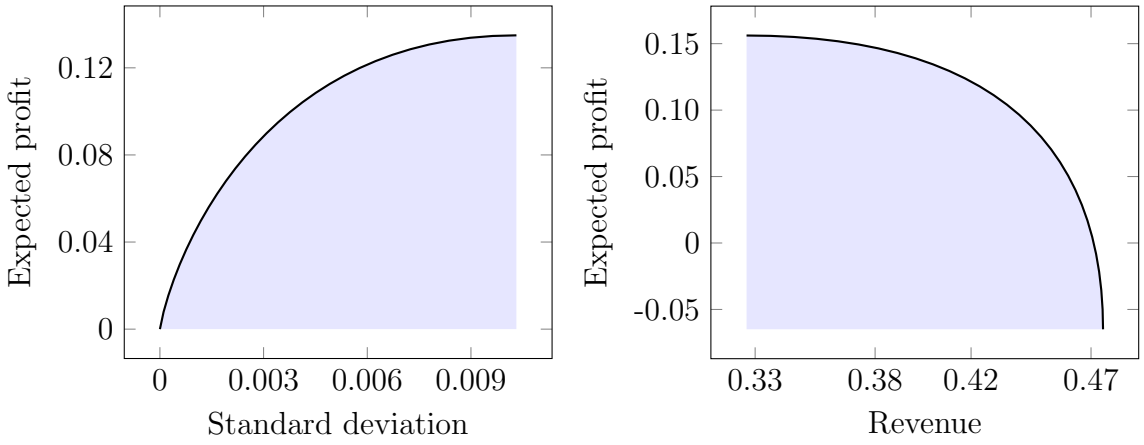


Figure 2.8: The two plots show the Pareto front between two competing objectives. The left figure compares the expected profit of three products, compared to the standard deviation of profit. On the right, one can see the trade-offs that are made between an expected profit maximisation and a revenue maximisation. The code used to generate the left Pareto front is given in Listing 1 later in the chapter.

Definition 2.10. Denote by $\underline{x_i}$ an individual minimiser of f_i, and set $\underline{f_i} = f_i(\underline{x_i})$. The point $\underline{F} = (\underline{f_1}, \ldots, \underline{f_m})$ is the vector containing all individual minima. Define the worst outcome for f_i on the Pareto front by $\overline{f_i} = \max_{1 \leq j \leq m}\{f_i(\underline{x_j})\}$. The point $\overline{F} = (\overline{f_1}, \ldots, \overline{f_m})$ is the vector containing all the worst outcomes.

Example 2.12 (Two-dimensional extremal points)**.** Figure 2.9 illustrates an example of the points $\underline{F}$ and $\overline{F}$ for a two-dimensional problem $F : \mathcal{X} \to \mathbb{R}^2$.

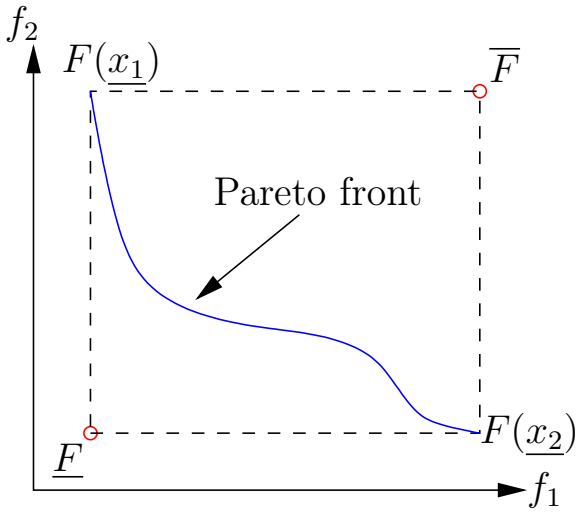


Figure 2.9: An example Pareto front $\mathcal{P}$ of two objectives. The points $\underline{F}$ and $\overline{F}$ represent the lower and upper bound of what values f_1 and f_2 can take on $\mathcal{P}$.

2.5.2 Approximating the Pareto front

The Pareto front contains all the information about the trade-offs between the objectives. A decision x to the multi-objective problem should be a Pareto optimal point, so the generic optimisation problem becomes one of searching over the set $\mathcal{P}$ of co-dimension one. If $\mathcal{X}$ or the objectives are non-convex, $\mathcal{P}$ can have non-convex regions. For some problems, $\mathcal{P}$ can be disconnected, as is illustrated in Figure 2.7. There are several ways to approximate the Pareto front. The simplest methods use a weighted sum or vary constraints on the objectives.

Definition 2.11 (Weighted sums)**.** For $\lambda \in [0,1]^m$ with $\sum_{j=1}^m \lambda_j = 1$, solve

$$\min_{x \in \mathcal{X}} \sum_{j=1}^{m} \lambda_i f_i(x). \tag{2.42}$$

The weighted sum method explores the Pareto front by solving a series of sub-minimisation problems for the function $\sum_{i=1}^m \lambda_i f_i$. The λ_i represent the relative weighting of each objective. By varying λ we may find new points on the Pareto front. This approach often works better if the objectives are normalised by the mapping $f_i \leftarrow (f_i - \underline{f_i})/(\overline{f_i} - \underline{f_i})$.

There are two major problems with the weighted sum approach. First, it cannot find Pareto points on non-convex parts of the Pareto front (Messac et al., 2000). Second, when the objectives are nonlinear, the solution to the optimisation problem reacts nonlinearly to changes in λ. The Pareto front approximation can, therefore, miss out important information, and will have a non-uniform spread of points, as illustrated in Figure 2.11c.

Definition 2.12 (Constraint method)**.** Choose $\epsilon_i \in (\underline{f_i}, \overline{f_i})$ for $i = 2, \ldots, m$, and solve the minimisation problem

$$\min_x \{f_1(x) \mid f_i(x) \leq \epsilon_i,\ i = 2, \ldots, m\}. \tag{2.43}$$

The constraint method can find Pareto points in non-convex regions of $\mathcal{P}$, and gives us control of the spread of points on the Pareto front along the axes of $f_2, \ldots, f_m$. In general, the points will not be uniformly distributed along the f_1-axis.

The Normal Boundary Intersection (NBI) method by Das and Dennis (1998) can create a more uniform spread of the Pareto points in both convex and non-convex regions of the Pareto front. Its optimisation problem formulation makes use of the convex hull of the individual minima (CHIM) defined below.

Definition 2.13 (Convex hull of individual minima (CHIM))**.** Define a matrix $\Phi \in \mathbb{R}^{m\times m}$ with columns $\Phi_j = F(\underline{x}_j) - \underline{F}$. The CHIM is the set

$$\{\Phi\beta + \underline{F} \mid \beta \in [0,1]^m,\ \textstyle\sum_{j=1}^m \beta_j = 1\}. \tag{2.44}$$

Example 2.13. Figure 2.10 provides an example of the CHIM for a bi-objective problem $F : \mathcal{X} \to \mathbb{R}^2$.

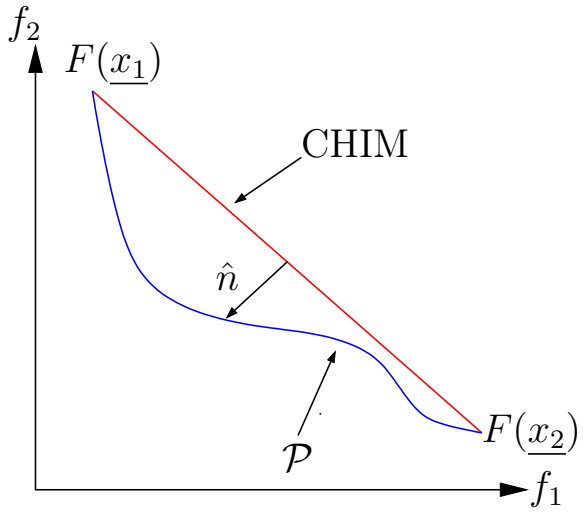


Figure 2.10: The Normal Boundary Intersection method. It traces out the Pareto front by moving from points of the CHIM along the normal direction $\hat{n}$.

Definition 2.14 (Normal Boundary Intersection)**.** Let $\hat{n}$ denote the unit normal to the CHIM pointing in the direction of smaller values in $\mathbb{R}^m$. Given $\beta \in [0,1]^m$ such that $\sum_{j=1}^m \beta_j = 1$, solve

$$\max_{x\in\mathcal{X},\, t\in\mathbb{R}} \{t \mid \Phi\beta + t\hat{n} = F(x) - \underline{F}\}. \tag{2.45}$$

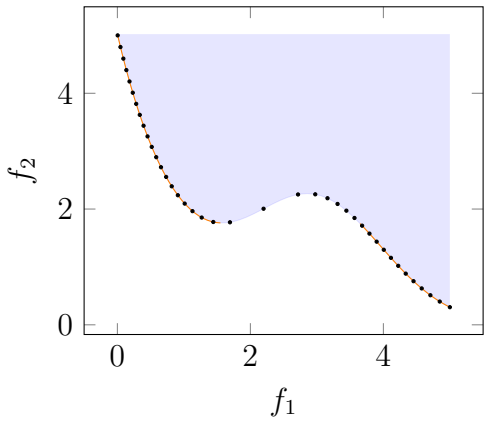
Example 2.14 (Comparison of Pareto front approximations)**.** We now investigate the approximations of the Pareto front on a simple example using the three different methods. Define the function $F : [0,5]^2 \to \mathbb{R}^2$ by $F = (f_1, f_2)$, with $f_i(x) = x_i$. Consider the bi-objective problem

$$\min_{x\in[0,5]^2} \{F(x) \mid 5e^{-x_1} + 2e^{-\frac{1}{2}(x_1-3)^2} \leq x_2\}. \tag{2.46}$$

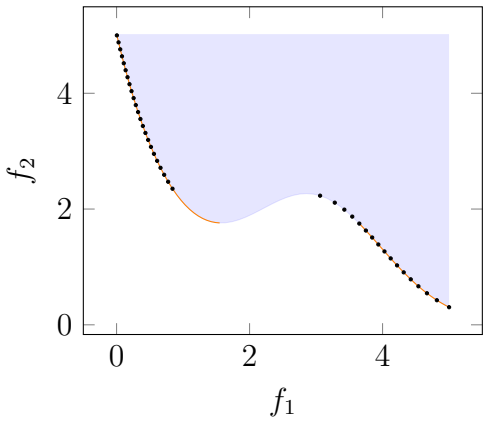
The Pareto front is disconnected and contains points that are locally, but not globally, Pareto optimal. The Pareto front and the image of $F(x)$ are shown in Figure 2.7 for

the restricted domain $\mathcal{X}_\mathrm{r} = \{x \in [0.4, 4.6]^2 \mid 5e^{-x_1} + 2e^{-\frac{1}{2}(x_1-3)^2} \leq x_2\}$.

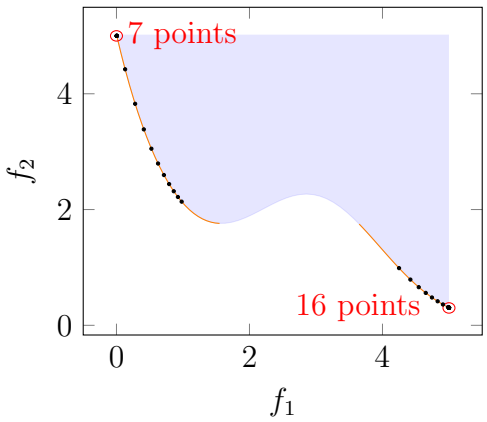
All the three methods, weighted sum, varying constraints, and NBI, define sub-problems parameterised by a value $\lambda \in [0,1]^m$. Without any a priori knowledge or adaptivity, we wish to approximate the Pareto front with 40 uniformly spaced parameter values. In Figure 2.11 one can see how the three methods perform in creating a uniform spread of points along the Pareto front.



(a) NBI



(b) Varying constraint on f_2



(c) Weighted sums

Figure 2.11: Approximations of a disconnected Pareto front, together with the Pareto front in orange. The NBI algorithm finds points that are not Pareto optimal, but is otherwise providing the best approximation of the Pareto front.

The weighted sum method provides the least information about the Pareto front, and many of the points are clustered on top of each other at the two ends of the Pareto front. The varying constraint method misses a portion of the Pareto front, and instead produces some points that are only locally Pareto optimal. This is because the underlying optimisation algorithm for each sub-problem finds local solutions to (2.43).

The NBI method provides a uniform spread that gives us the most information for the same number of sub-problem optimisations, however, it also produces some points that are not locally Pareto optimal.

When the Pareto front is disconnected, Example 2.14 shows that the NBI method can find points that are not locally Pareto optimal. We can modify the method so that the method produces locally optimal points.

Definition 2.15 (NBI extension)**.** Given $\beta \in [0,1]^m$ with $\sum_{j=1}^{m} \beta_j = 1$, solve

$$\max_{x \in \mathcal{X},\, t \in \mathbb{R}} \{t \mid \Phi\beta + t\hat{n} \geq F(x) - \underline{F}\}. \tag{2.47}$$

Example 2.15 (Disconnected Pareto front with NBI extension)**.** Consider the bi-objective optimisation problem from Example 2.14. Figure 2.12 shows 40 points approximating the Pareto front using the NBI extension, which are all locally Pareto optimal.

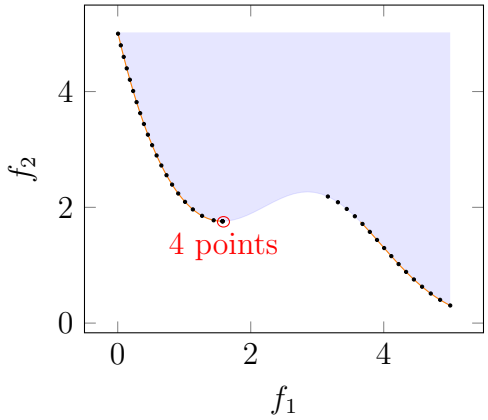


Figure 2.12: The NBI extension creates only locally Pareto optimal points, in contrast to the NBI method, as seen in Figure 2.11a. There are four points lying on top of each other because the current algorithm is not adaptive.

The NBI method is a special case of the scalarization methods for vector-optimisation formulated by Pascoletti and Serafini (1984). The NBI extension fits within the Pascoletti-Serafini framework in such a way that it guarantees local Pareto optimality. Figures 2.11 and 2.12 show issues that arise with a disconnected Pareto front. The NBI extension implemented in the package MultiJuMP, described in §2.5.4, finds the same point multiple times on the border of disconnected components of the Pareto front, as illustrated in Figure 2.12. Eichfelder (2009) has devised an adaptive algorithm based on Pascoletti-Serafini scalarization that can handle non-convex and disconnected Pareto fronts better the algorithms presented here.

2.5.3 Ordering decision points a priori

Up to now we have described a process for making a decision $x \in \mathcal{X}$ that involves approximating the Pareto front. The cost of the approximation stage increases exponentially with the number of objectives. The preferred position on the Pareto front is subjective and prone to change depending on the presentation of $\mathcal{P}$. With more than two objectives, it is difficult to present the Pareto front without losing some information. Given the same Pareto front, presented in k different ways, the variation of decisions $x^1, \ldots, x^k$ is likely to increase with the number of objectives. Ideally one would like to create an ordering between decisions so that the optimal decision is the same every time the same optimisation problem is run.

Simple approaches to creating an ordering include the weighted sum and constrained objectives methods. With weighted sums, the ordering will be based on an ordering defined by the function $u(x) = \sum_{i=1}^{n} \lambda_i f_i(x)$. The challenge is then to decide the relative weights of the objectives. The nonlinear response of λ in the weighted sum method means that the defined ordering can be unstable to small perturbations of λ. One can instead define upper bounds for $f_2, \ldots, f_m$ and minimise f_1. Given the upper bounds $\epsilon_i \in [\underline{f_i}, \overline{f_i}]$, the ordering is only on the value of the first objective. Small changes in ϵ can turn an optimal decision x infeasible. There is no room for the optimisation algorithm to look at trade-offs between objectives, although one could get some understanding of trade-offs from the Lagrange multipliers of the constraints.

To define an ordering of decisions, one should find a way to express the total value $u : \mathbb{R}^m \to \mathbb{R}$ of the multi-objective function F. The survey by Marler and Arora (2004) describes different choices of the function u that have been used in the literature. Messac (1996) attempts to create a total value function with an approach he calls Physical Programming. The decision maker defines what a "desirable", "tolerable", "undesirable" value corresponds to for each objective. The implementation of this method is much more complicated than the other approaches considered but may give us an easier framework to approximate a decision maker's preferences.

The total value function $u(F)$ can be thought of as a multivariate utility function. To ensure that an optimal decision according to u is also Pareto optimal, we impose that u is strictly increasing in all of its arguments. If the decision problem involves randomness, one may already have tried to model the decision maker's risk-aversion for each objective. With the expected utility approach to risk-aversion, that would mean that utility functions u_i are readily available. One can then make a simple approximation of the total value $u(F)$ with the mapping $F \mapsto \sum_{i=1}^{n} u_i(f_i)$. If the u_i are concave, then u defines a multivariate risk-averse utility (Campi and Owen,

2011, Sec. 2.3.2). This could provide a good first approximation to orderings in multi-objective optimisation problems involving randomness.

2.5.4 Approximating Pareto fronts with MultiJuMP

As part of the work on multi-objective optimisation, we have created a software package called MultiJuMP. It is available online at `https://github.com/anriseth/MultiJuMP.jl.git`. MultiJuMP approximates Pareto fronts in a numerical solver-independent environment, using the four methods discussed in §2.5.2. The package is written in the Julia programming language and is based on the mathematical programming modelling package JuMP. It separates the challenge of expressing an optimisation problem from that of choosing an algorithm for solving the problem. Listing 1 shows an example of how to set up a multi-objective optimisation model with the MultiJuMP routines. The code works for any number of objectives, however, visualising a Pareto front for more than three objectives becomes difficult.

Listing 1 Code used to generate one of the figures in Example 2.10. The functions `MultiModel()`, `getMultiData()`, `SingleObjective()` and `solve()` were created as part of the work on MultiJuMP.

```
using JuMP, MultiJuMP
# Define parameters
a  = [1, 0.9, 1.2];
B  = [2 2 0; 0.8 1.8 8; 3 0 2]
my = [0.5, 0.5, 0.65]
Cy = [0.0025 -0.00075 0; -0.00075 0.0025 0; 0 0 0.0042]
x0 = [1, 1, 1.3]
n  = length(x0)

# Set up Multiobjective model
m = MultiModel()

# JuMP commands create functions
@variable(m, x[i=1:n]  0, start=x0[i])
@NLexpressions m begin
    demand[i=1:n],  (a[i]*exp(-B[i,i]*x[i])*
                     prod(1-exp(-B[i,j]*x[j]/x[i])
                          for j=1:n if (j != i) && !iszero(B[i,j])))
    profit[i=1:n],  (x[i]-my[i])*demand[i]
    totalProfit,    sum(profit[i] for i=1:n)
    stdeviation,    (sqrt(sum(Cy[i,j]*demand[i]*demand[j]
                              for j=1:n for i=1:n)))
end

# Define two objectives
md              = getMultiData(m)
obj1            = SingleObjective(stdeviation, sense = :Min)
obj2            = SingleObjective(totalProfit, sense = :Max)
md.objectives   = [obj1, obj2]
md.pointsperdim = 30

# solve calls MultiJuMP to approximate Pareto front
solve(m, method = :NBI)

# The Pareto points are stored in md.paretofront
```

Chapter 3

Control strategies for a discrete time pricing problem

In the previous chapter, we discussed one-stage decision problems. The focus in this chapter is on the temporal structure of the decisions, and the randomness in the objective is marginalised using the expected value. We consider another simple problem for pricing products in retail to motivate the concepts and results.

When sales of a product are affected by randomness in demand, retailers can use dynamic pricing strategies to maximise their profits. In this chapter, the pricing problem is formulated as a discrete-time stochastic optimal control problem, for which the optimal pricing policy satisfies a recursive equation known as the Bellman equation (Bertsekas, 2005), defined in §3.1. A canonical example of a stochastic control problem that has a closed-form solution that can be found with the Bellman approach is the Merton portfolio problem due to Merton (1969) and Samuelson (1969). For realistic retail applications, however, modelling the problem and solving it to optimality in this way is intractable. Thus, practitioners make simplifying assumptions and design suboptimal policies, but a thorough investigation of the relative performance of these policies is lacking. To better understand such assumptions, we simulate the performance of two algorithms on a one-product system and compare them to the Bellman solution.

The aim of this chapter is to highlight the implications that suboptimal policy choices have on the distribution of the relevant objective and not only marginalised quantities such as the expected value. In particular, we consider a one-product pricing problem and investigate the optimal Bellman policy, an often-used suboptimal policy known as the *Certainty Equivalent Control* (CEC) policy, and a compromise between optimality and practicality known as the *Open-Loop Feedback Control* (OLFC) policy.

Rather surprisingly, we find that more than half the time, using the suboptimal CEC policy leads to a higher profit than using the Bellman policy, for a wide range of products. Of course, the Bellman policy performs better on average, but this is due to the CEC policy generating a larger, lower tail on the distribution of profits than the Bellman policy. Heuristically, one could say that the CEC policy has a higher risk associated with it. We wish to emphasise that this could indicate that a choice of approximate control policy is implicitly also a statement of risk attitude. Thus, we propose that it should be addressed at the same level as a description of a decision maker's risk attitude as expressed with utility functions or risk measures.

In practical applications in retail and other domains, it is almost always intractable to solve the control problem to optimality (Farmer, 2017). The modelling of a real system and decision process can become very complicated, and we must create a very large state space in order to make use of the Bellman equation. We often have to take into account unobservable state variables, such as estimated parameters, and constraints that may depend on history. Also, decision-makers may change their mind about the objective over the course of the decision period, which should be incorporated in the modelling as parameters with corresponding, estimated probability distributions. From a software implementation perspective, writing code that can solve the Bellman equation efficiently can be much more complicated than other methods. Finally, the dimensions of the state, policy, and exogenous information spaces can quickly make a numerical solution to the Bellman equation intractable. An increase in the dimension can happen very quickly, even for one-product problems: Bertsimas and Perakis (2001) model a simple one-product demand function with uncertainty in the function parameters, and present an eight-dimensional dynamic programming solution to the problem. Much of the focus in the research community has therefore been on developing tractable algorithms with comparable average performance to the optimal policies, see for example Powell (2011) or Bertsekas (2012) for an overview. One can either create explicit policy functions off-line, at the start of the decision process, or implicitly through an automated search for the best policy on-line for each decision point. An advantage of the pre-calculated, explicit policy functions is the speed at which we can make our decisions in the future.

In the classical pricing problems, the dynamics of the underlying system do not require instant pricing decisions. Thus suboptimal decision rules that are created as they are needed are often used instead (Talluri and van Ryzin, 2006). Estimation of the system and optimisation of prices are normally separated, and constraints are more easily updated at each decision point. There are several proposed approximations in

the literature, although for revenue management applications much of the domain knowledge is kept within respective commercial organisations (Talluri and van Ryzin, 2006, Ch. 9). Many of the suboptimal pricing algorithms are justified on practical grounds (Aviv and Vulcano, 2012), or from asymptotic results (Gallego and Van Ryzin, 1994). Our results add to the literature by highlighting the distributional implications the chosen pricing algorithm can have on the objective.

The chapter is organised as follows. In §3.1 we formulate the pricing problem mathematically and describe an algorithm to solve the problem via the associated Bellman equation. We also give an example of a problem with one product and present the optimal pricing rules. A discussion of the CEC and OLFC policies follows in §3.2, where we compare its performance to that of the optimal policy. Finally, we conclude and propose further work in §3.3.

3.1 The pricing problem

We consider a particular retail pricing problem. Given an initial amount of stock of a product and a fixed future termination time $\hat{T}$, the pricing problem is to set the price of the product dynamically, in order to both maximise the revenue and minimise the cost of unsold stock at the termination time. In this section, we define the optimal control pricing problem, describe the optimality conditions given by the Bellman equation and solve it numerically for an example system.

Consider a system over discrete, equispaced, time points $\hat{t} = 0, 1, \ldots, \hat{T}$, with state $\hat{S}_{\hat{t}}$ and pricing process $\hat{\alpha}$ such that the $\hat{\alpha}_t$ take values in a closed interval $\hat{A} = [\hat{a}_{\min}, \hat{a}_{\max}]$ of prices.[1] The state $\hat{S}_{\hat{t}}$ is the stock of a product at time $\hat{t}$, and $\hat{\alpha}_{\hat{t}}$ the price for the product in the time period from $\hat{t}$ to $\hat{t}+1$. We model the amount of product sold over each time period according to a forecast demand function $\hat{q} : \hat{A} \to \mathbb{R}_+$, which is bounded, continuous and decreasing. Exogenous influences on demand are considered as randomness in the system, and are modelled in a multiplicative fashion by a stochastic process $\hat{W} = (\hat{W}_1, \ldots, \hat{W}_{\hat{T}})$ taking non-negative values. See, for example, Talluri and van Ryzin (2006, Ch. 7) for a discussion of demand models and the modelling of uncertainty.

For a given pricing process $\hat{\alpha}$, the system evolves from some initial state $\hat{S}_0 > 0$,

[1] In practice, policy values may have additional constraints that depend on the current state, in which case we say that $\hat{\alpha}_{\hat{t}}$ must take values in a set $\hat{A}(\hat{S}_{\hat{t}})$.

according to

$$\hat{S}^{\hat{\alpha}}_{\hat{t}+1} = \hat{S}^{\hat{\alpha}}_{\hat{t}} - \min(\hat{S}^{\hat{\alpha}}_{\hat{t}}, \hat{q}(\hat{\alpha}_{\hat{t}})\hat{W}_{\hat{t}+1}), \qquad \hat{t} = 0, \ldots, \hat{T} - 1. \tag{3.1}$$

The function

$$\hat{Q}(\hat{s}, \hat{a}, \hat{w}) = \min(\hat{s}, \hat{q}(\hat{a})\hat{w}) \tag{3.2}$$

denotes the unit sales over a period at price $\hat{a}$, starting with stock $\hat{s}$, and with exogenous influences characterised by $\hat{w}$. The minimum operator is used to ensure that the amount of product sold over the time period does not exceed the current stock level.

The revenue accrued over period $\hat{t} \to \hat{t} + 1$ is $\hat{\alpha}_{\hat{t}}\hat{Q}(\hat{S}^{\hat{\alpha}}_{\hat{t}}, \hat{\alpha}_{\hat{t}}, \hat{W}_{\hat{t}+1})$. The cost of remaining stock at time $\hat{T}$ is modelled by a cost per unit stock $\hat{C} \geq 0$.[2] Let $\hat{\mathcal{A}}$ denote the set of feasible processes $\hat{\alpha}$ that take values in $\hat{A}$. Define the value of having stock $\hat{s}$ at time $\hat{t} \leq \hat{T}$ by the value function $\hat{v}$, such that

$$\hat{v}(\hat{t}, \hat{s}) = \max_{\hat{\alpha} \in \hat{\mathcal{A}}} \hat{J}(\hat{t}, \hat{s}, \hat{\alpha}), \tag{3.3}$$

where

$$\hat{J}(\hat{t}, \hat{s}, \hat{\alpha}) = \mathbb{E}_{\hat{W}} \left[\sum_{\hat{\tau}=\hat{t}}^{\hat{T}-1} \hat{\alpha}_{\hat{\tau}} \hat{Q}(\hat{S}^{\hat{\alpha}}_{\hat{\tau}}, \hat{\alpha}_{\hat{\tau}}, \hat{W}_{\hat{\tau}+1}) - \hat{C}\hat{S}^{\hat{\alpha}}_{\hat{T}} \mid \hat{S}^{\hat{\alpha}}_{\hat{t}} = \hat{s} \right]. \tag{3.4}$$

We use the subscript on $\mathbb{E}_{\hat{W}}$ to emphasise that the expectation is taken with respect to the random variables $\hat{W}_{\hat{\tau}}$. The definition of the value function leads us to the following mathematical formulation of the pricing problem:

Definition 3.1. Given an initial amount of stock $\hat{S}_0 > 0$ and a constant cost per unit unsold stock $\hat{C} \geq 0$, the *pricing problem* is to find a pricing process $\hat{\alpha}^* \in \hat{\mathcal{A}}$ such that

$$\hat{J}(0, \hat{S}_0, \hat{\alpha}^*) = \hat{v}(0, \hat{S}_0), \tag{3.5}$$

that is, $\hat{\alpha}^* = \arg\max_{\hat{\alpha} \in \hat{\mathcal{A}}} \hat{J}(0, \hat{S}_0, \hat{\alpha})$.

We choose to maximise the expected profit over the period, which assumes a risk-neutral decision maker. Although we are maximising expected profit, it is still important to understand the distribution of profits for a given pricing policy $\hat{\alpha}$.

[2] In some situations, unsold items at time $\hat{T}$ may be sold at some "salvage price", in which case we could allow $\hat{C} < 0$. We assume $\hat{C} \geq 0$ in this chapter.

Therefore, we simulate the distribution when we investigate the performance of algorithms in §3.2. If we assume that the random variables $\hat{W}_{\hat{t}}$ are independent, this stochastic optimal control problem can be solved by considering the optimality conditions that arise from the Dynamic Programming principle, also known as the Bellman equation (Bertsekas, 2005).

3.1.1 Non-dimensionalisation of the system

We now consider a non-dimensionalised representation of the system. The units of stock are scaled with respect to the initial stock $\hat{S}_0$, and units of money are scaled with respect to the upper bound on price, $\hat{a}_{\max}$, which we assume is finite. Let the corresponding dimensionless quantities be defined without hats. Then we set

$$s = \frac{\hat{s}}{\hat{S}_0}, \qquad a = \frac{\hat{a}}{\hat{a}_{\max}}, \qquad C = \frac{\hat{C}}{\hat{a}_{\max}}, \qquad t = \hat{t}, \qquad W_t = \hat{W}_{\hat{t}}. \tag{3.6}$$

The dimensionless functions $q(a)$ and $Q(s, a, w)$, for forecasted demand and realised sales respectively, are

$$q(a) = \frac{\hat{q}(a \cdot \hat{a}_{\max})}{\hat{S}_0}, \qquad Q(s, a, w) = \min(s, q(a)w). \tag{3.7}$$

The collection of pricing policies $\mathcal{A}$ contains all the processes $\hat{\alpha}_{\hat{t}}/\hat{a}_{\max}$, where $\hat{\alpha} \in \hat{\mathcal{A}}$. Now we can define the dimensionless value function

$$v(t, s) = \max_{\alpha \in \mathcal{A}} J(t, s, \alpha), \quad \text{where} \tag{3.8}$$

$$J(t, s, \alpha) = \mathbb{E}_W \left[\sum_{\tau=t}^{T-1} \alpha_\tau Q(S_\tau^\alpha, \alpha_\tau, W_{\tau+1}) - C S_T^\alpha \middle| S_t^\alpha = s \right]. \tag{3.9}$$

Finally, the dimensionless optimal control problem is to find $\alpha^* \in \mathcal{A}$, such that $J(0, 1, \alpha^*) = v(0, 1)$. For the remainder of this chapter, we work with the non-dimensionalised system.

3.1.2 The Bellman equation

We choose $\mathcal{A}$ to be the set of Markovian policies for the problem. This means that for each $\alpha \in \mathcal{A}$, there exists a measurable function $a : \{0, \ldots, T-1\} \times \mathbb{R}_+ \to A$, such that for each possible outcome ω, the process α is given by $\alpha_t(\omega) = a(t, S_t^\alpha(\omega))$. An approach to finding the value function above is to use the Dynamic Programming

principle (Bertsekas, 2005). It is also known as Bellman's principle of optimality, due to Bellman (1954);

> "*Principle of Optimality.* An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions."

In the context of our problem, it states that if $\alpha = (\alpha_u)_{u=0}^{T-1}$ is an optimal policy over the time points $u \in \{0, \ldots, T-1\}$, then an optimal policy over the time points $u \in (t, \ldots, T-1)$ is the restriction $(\alpha_u)_{u=t}^{T-1}$ of α. Thus, v can be defined recursively by

$$v(t,s) = \max_{a \in A} \mathbb{E}_{W_{t+1}} \left[aQ(s,a,W_{t+1}) + v\big(t+1, s - Q(s,a,W_{t+1})\big) \right]. \tag{3.10}$$

The value function is the solution to the (backwards-in-time) recursive relation (3.10), with terminal value $v(T,s) = -Cs$, and the optimal policy function $a(t,s)$ is given by

$$a(t,s) = \arg\max_{a \in A} \mathbb{E}_{W_{t+1}} \left[aQ(s,a,W_{t+1}) + v\big(t+1, s - Q(s,a,W_{t+1})\big) \right]. \tag{3.11}$$

The recursion (3.10) is called the *Bellman equation*, and a discussion of its validity can be found, for example, in Bertsekas (2005). Analytical solutions to Bellman equations are only available in rare cases, and numerical approaches are normally needed to approximate the solutions.

We implement the following algorithm to approximately solve the optimal control

problem, using the Bellman equation:

1. Create a grid of equispaced points $0 = s_1 < s_2 < \cdots < s_K = S_0$, and arrays $v^K \in \mathbb{R}^{K\times(T+1)}$, $\alpha^K \in \mathbb{R}^{K\times T}$.
2. Set $I[v^K](s) = -Cs$.
3. Set $v^K[i, T] = I[v^K](s_i)$ for $i = 1, \ldots, K$.
4. For $t = T-1, \ldots, 0$:
   (a) Set $v^K[i,t] = \max_{a\in A} \mathbb{E}_{W_{t+1}} \left[aQ(s_i, a, W_{t+1}) + I[v^K](s_i - Q(s, a, W_{t+1}))\right]$ for $i = 1, \ldots, K$.
   (b) Set $\alpha^K[i,t]$ to the maximiser above.
   (c) Set $I[v^K](s) = \text{Interpolate}(s, (s_i)_i, (v^K[i,t])_i)$.
5. Return v^K, α^K.

The expectation above is approximated using Monte Carlo simulation, for which the *variance* of the approximation error decreases by the reciprocal of the number of samples used (Caflisch, 1998). In this chapter we use 1000 samples for the approximation of the expectation in Step 4(a). We choose to use piecewise linear interpolation for $I[v^K](s)$ in Step 4(c) with a grid of $K = 201$ points. All numerical optimisation procedures in this chapter use the package Optim.jl (Mogensen and Riseth, 2018) from the Julia programming language (Bezanson et al., 2017). The calculation of the Bellman policy is only approximate. Tests, not included here, show that our choice of grid points and Monte Carlo samples in the numerical examples yield high accuracy approximations. We therefore refer to the calculated policy as the solution to the Bellman equation from now on.

3.1.3 A one-product example system

In order to investigate the optimal pricing of a specific system, we choose to look at a family of commonly used demand functions of the form $q(a) = q^{(1)}e^{-q^{(2)}a}$, where $q^{(1)}, q^{(2)} > 0$. For a discussion about their properties and usage in modelling demand, see Talluri and van Ryzin (2006, Ch. 7). All the numerical experiments in this chapter use this family of exponential demand functions, with price constraint interval $A = [0, 1]$. In the current section, and in Figures 3.1 to 3.4, we consider a particular

choice $q^{(1)} = e^2/3$ and $q^{(2)} = 3$ so that the demand function is given by $q(a) = \frac{1}{3}e^{2-3a}$. In Figure 3.5 and Table 3.1 a larger range of values for $q^{(1)}, q^{(2)}$ are considered.

We assume the exogenous disturbance process is a sequence of shifted, independent and identically Beta-distributed random variables with mean 1 and variance γ^2. That is, we define $W_t \sim \frac{1}{2} + X$, where $X \sim \text{Beta}(\mu, \nu)$,[3] and $\mu = \nu = \frac{1}{8\gamma^2} - \frac{1}{2}$. To ensure that X is unimodal, we require that $\gamma^2 < 1/12$.

Set $\gamma = 0.05$, $C = 1$ and $T = 3$. The numerical solution to the Bellman equation, and the corresponding optimal pricing policy, is shown in Figure 3.1. Let us now

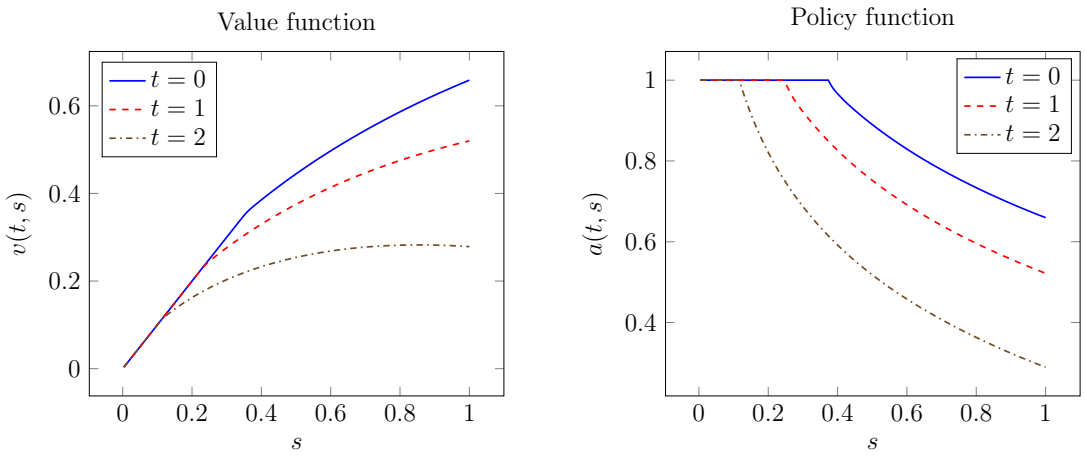


Figure 3.1: The value function and corresponding optimal control function for the pricing problem. The regions where the value function moves from a linear to a nonlinear regime corresponds to the pricing policy moving from the upper bound 1 to less than 1.

investigate the behaviour of the optimal pricing policy α and the outcome of following this policy. Define a random variable $P(\alpha)$, which for each realisation represents the total profit, by

$$P(\alpha) = \sum_{t=0}^{T-1} \alpha_t Q(S_t^\alpha, \alpha_t, W_{t+1}) - C S_T^\alpha. \tag{3.12}$$

By sampling from the stochastic process $W = (W_1, \ldots, W_T)$, we can estimate the random variables α_t and $P(\alpha)$. The plots in Figure 3.2 show the results of simulating the system 10 000 times.[4]

[3] A Beta(μ, ν) random variable has support $[0, 1]$ with probability density function given by $p(x) = \frac{x^{\mu-1}(1-x)^{\nu-1}}{B(\mu,\nu)}$. The function $B(\mu, \nu)$ is the normalising factor.

[4] Preliminary investigations indicated that 1000 simulations is sufficient to obtain an accurate representation of the distributions in this chapter. However, we have chosen to use more when this is computationally convenient.

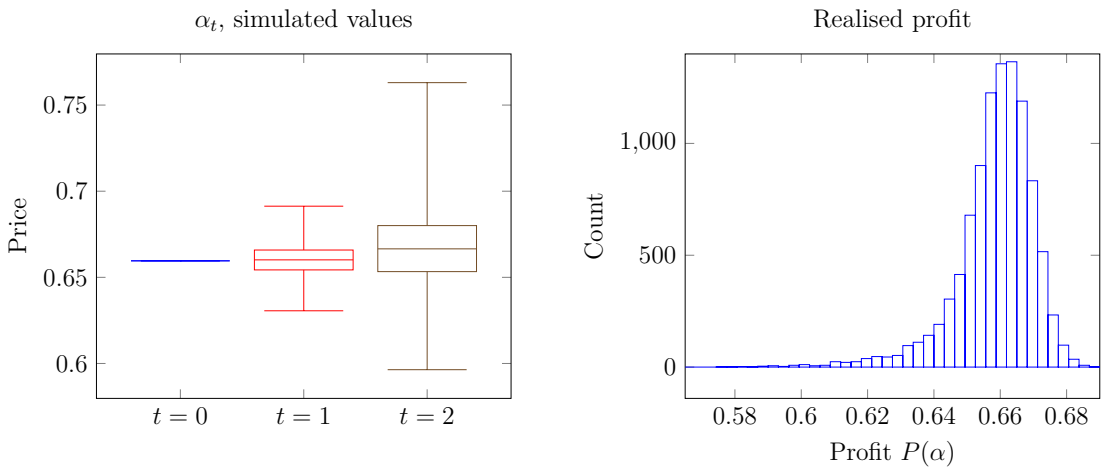


Figure 3.2: Simulations of the pricing system, started at $S_0 = 1$ and controlled by the optimal policy α shown in Figure 3.1. The left figure is a box plot that shows the median, the quartiles and the extremal prices observed in the simulations. As we see, the variance of the prices increases in time, reflecting the wider range of realised remaining stock at these times. The right hand figure is a histogram of the total profit over the pricing period.

3.2 Suboptimal approximations

In this section we look at two suboptimal policies among the class of algorithms that calculate decisions on-line. Whenever a decision must be made, these methods simulate the future behaviour of the system and optimise the current decision based on these simulations. We consider two special cases known as the *Certainty Equivalent Control* (CEC) policy, which uses a point estimate of the system, and the *Open-Loop Feedback Control* (OLFC) policy, which uses more information about the future behaviour of the system (Bertsekas, 2005, Ch. 6). For the example system in this chapter, it turns out that we can find the CEC policy analytically. Numerical comparison experiments between these policies and the optimal Bellman policy are made using the example pricing problem introduced in §3.1.3. We also extend the comparison to cover a much larger range of parameters in the problem, in order to show that the same results hold for a wide range of retail products.

3.2.1 The Certainty Equivalent Control policy

An often-used, tractable, algorithm for approximating the solution of stochastic optimal control problems is the Certainty Equivalent Control policy, as Bertsekas (2005) calls it. It is also known as (deterministic) Model Predictive Control in the engineering

community. The algorithm is particularly practical, because the optimisation problem is reduced to a standard deterministic problem that can be solved with existing solvers which handle very large systems. At each decision time, a deterministic optimisation problem for the remaining decision horizon is solved. Only the decision for the current time step is used, whilst the subsequent decisions are discarded. For each decision time $t = 0, 1 \ldots, T-1$, the CEC policy for the pricing problem calculates the current price using the following steps:

1. Observe the state s.
2. Take a best estimate $w_{t+1}, \ldots, w_T$ of $(W_\tau)_{t+1}^T$.
3. Solve the optimisation problem

$$\max_{\mathbf{a} \in A^{T-t}} \left\{ \sum_{\tau=t}^{T-1} \mathbf{a}_\tau Q(S_\tau^{\mathbf{a}}, \mathbf{a}_\tau, w_{\tau+1}) - C S_T^{\mathbf{a}} \right\}, \quad \text{s.t.} \quad S_t^{\mathbf{a}} = s, \tag{3.13}$$

   where A^{T-t} is the $(T-t)$ times Cartesian product $A \times \cdots \times A$.
4. Set the price corresponding to the element $\mathbf{a}_t \in A$ of a maximiser to (3.13).

The decision maker decides what a "best estimate" in Step 2 means. It can, for example, be derived from a data assimilation procedure, or the mean or mode of the W_τ.

For the one-product pricing problem, we can simplify the optimisation problem and in some cases obtain analytical solutions for the policy function. Consider the expected value estimate $w_\tau = \mathbb{E}[W_\tau] = 1$ for $\tau = t+1, \ldots T$. Then we can rewrite the maximisation problem to find the certainty equivalent value function,

$$\widetilde{v}(t, s) = \max_{\mathbf{a} \in A^{T-t}} \left\{ \sum_{\tau=t}^{T-1} (\mathbf{a}_\tau + C) \min \left(s - \sum_{r=t}^{\tau-1} q(\mathbf{a}_r), q(\mathbf{a}_\tau) \right) - Cs \right\}. \tag{3.14}$$

The optimal choice here is to let $\mathbf{a}_\tau = a^* \in A$ for each τ, such that the same amount of stock is forecast to be sold in each period. The optimality of a constant-in-time price follows from the Hotelling rule, due to Hotelling (1931), which states that the price of an exhaustible resource must grow at a rate equal to the rate of interest. In our case the rate of interest is zero, and thus the price is constant. The optimal price $a^* = a^C(t, s)$ is given by the policy function

$$a^C(t, s) = \arg\max_{a \in A} \left\{ (a + C) \min \left(\frac{s}{T-t}, q(a) \right) \right\}. \tag{3.15}$$

Let $\mathcal{P}_A$ be the projection operator from $\mathbb{R}$ onto the interval A. In the case when $q(a) = q^{(1)}e^{-q^{(2)}a}$, the policy function is

$$a^C(t,s) = \mathcal{P}_A\left[\max\left(\frac{1}{q^{(2)}}\log\left(\frac{q^{(1)}(T-t)}{s}\right), \frac{1}{q^{(2)}} - C\right)\right]. \tag{3.16}$$

We start with investigating policy functions for a single combination of the system parameters, and then provide a comparison between the Bellman policy and CEC policy for a larger range of parameters in §3.2.2.

3.2.1.1 Example system

The example system in §3.1.3 has a demand function of the form $q(a) = q^{(1)}e^{-q^{(2)}a}$, where $q^{(1)} = e^2/3$, $q^{(2)} = 3$, $\gamma = 0.05$, and $C = 1$. We can therefore compare the optimal Bellman policy with the pricing policy that the CEC algorithm above implies. Denote the Bellman policy function by a^B and the CEC policy function by a^C, as in (3.16). The plot in Figure 3.3 shows the functions $a^B(t,\cdot)$, $a^C(t,\cdot)$ for each $t = 0,\ldots,(T-1)$. Both of the policy functions reach the upper bound in A for small values of s, but most of the time, the Bellman policy is pricing the products lower than the CEC policy.

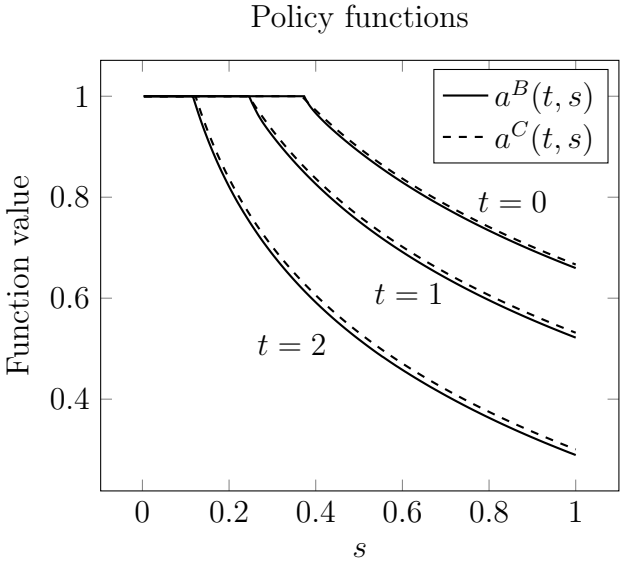


Figure 3.3: Comparison of the CEC and Bellman policy functions a^C and a^B. The CEC function sets higher prices than the Bellman function for most of the values of s.

What is more important than how the policy function works, is how it impacts the distribution of realisable profit. We would like to see how the two policy pro-

cesses α^B and α^C perform. One way to evaluate their performance is to look at the distribution of the profits $P(\alpha^B)$ and $P(\alpha^C)$. We remind the reader that $P(\alpha) = \sum_{t=0}^{T-1} \alpha_t Q(S_t^\alpha, \alpha_t, W_{t+1}) - CS_T^\alpha$. From the optimality of the Bellman policy function, we must have that $\mathbb{E}_W[P(\alpha^B)] \geq \mathbb{E}_W[P(\alpha^C)]$, although in our case violations of this result can happen because of numerical errors in approximating α^B and the expectations. Marginalising a random variable with the expectation operator, however, loses a lot of information which can be of interest. An approximation of the distributions of $P(\alpha^B), P(\alpha^C)$ and their difference $P(\alpha^B) - P(\alpha^C)$, based on 10 000 realisations of the underlying W, can be seen in Figure 3.4. In the experiment, the average value of following the Bellman policy is higher than following the CEC policy, as expected. However, from the bottom figure we see that in more than half of the cases, the CEC policy outperforms the optimal policy α^B. What we can take from this experiment is that, in a colloquial sense of the word, the CEC policy induces a more risk-seeking pricing strategy than α^B: It results in slightly larger profits for a majority of the realisations of W, but at a cost of taking a more significant reduction in profits in the remaining realisations.

3.2.2 Bellman and CEC parameter comparisons

The experiment in the previous section only provides results for a fixed combination of the five parameters termination time T, unsold items cost C, uncertainty γ, and demand function parameters $q^{(1)}$ and $q^{(2)}$. In this section we explore the differences between the Bellman policy and the CEC policy for a larger parameter range. The formula for a^C in (3.16) indicates that the initial price is largely determined by the relationship between $\log(Tq^{(1)})$ and $1/q^{(2)}$, and hence we choose to keep $T = 3$ fixed whilst varying $q^{(1)}$ and $q^{(2)}$. We are interested in the difference between the profit following a Bellman policy α^B and a CEC policy α^C. The policy α^B is computed numerically, and α^C is obtained using the function a^C from the formula in Equation (3.16). In particular, the difference between the two policies is measured using the L^2-norm with respect to the probability distribution induced by the disturbance $(W_1, W_2, \ldots, W_T)$, that is

$$\left\| P(\alpha^B) - P(\alpha^C) \right\|_2^2 = \mathbb{E}_W \Big[\left(P(\alpha^B) - P(\alpha^C) \right)^2 \Big]. \tag{3.17}$$

To reduce the number of combinations of parameters, we choose only four combinations of $(C, \gamma) \in \{(0.5, 0.05), (0.5, 0.1), (1, 0.05), (1, 0.1)\}$. Then, for each combination of (C, γ), we can create a heatmap of the difference between the two policy functions

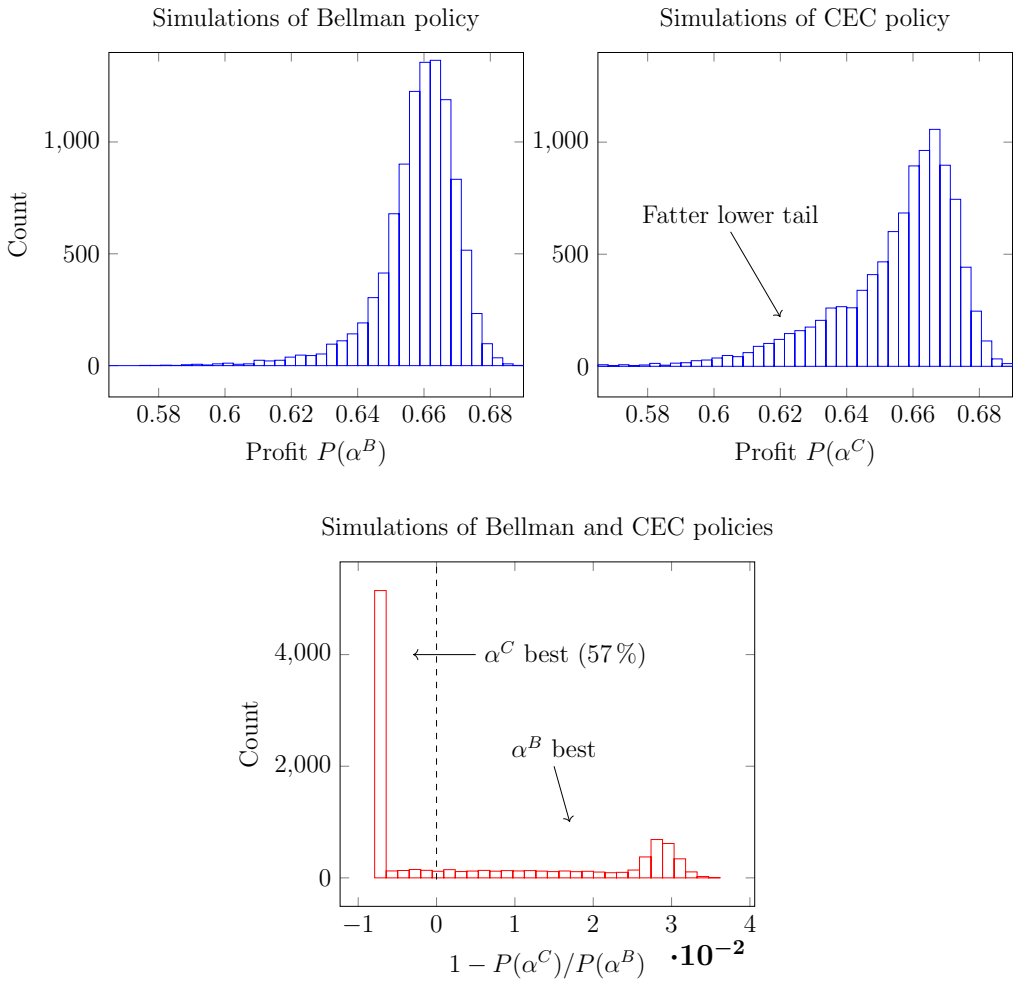


Figure 3.4: This shows the distributions from 10 000 samples of the profits of following the Bellman and CEC policies. The sample mean $\mathbb{E}_W[P(\alpha^B) - P(\alpha^C)] \approx 3.8 \times 10^{-3}$, confirms that Bellman is better on average, as it should be. Importantly, however, in more than half the samples the suboptimal policy outperforms the Bellman policy. The distribution of $1 - P(\alpha^C)/P(\alpha^B)$ appears to be bimodal.

by varying the parameters $q^{(1)}, q^{(2)}$. The relative L^2 distance between the CEC and Bellman policy outcomes is shown in Figure 3.5. The norms were approximated with 1000 samples from $(W_1, W_2, \ldots, W_T)$. Comparing the left and right column, we see that the relative difference doubles as the standard deviation γ is doubled. The cost C plays a large role, both in terms of the shape of the difference surface and its magnitude. The black region at the bottom right of the plots corresponds to popular, low-elasticity products where both policies decide to sell at the maximum price $a = 1$.

The parameters used to generate Figure 3.4 were $C = 1$, $\gamma = 0.05$, $q^{(1)} = e^2/3 \approx 2.5$, and $q^{(2)} = 3$. This corresponds to a point in the region where the relative difference $\|P(\alpha^B) - P(\alpha^C)\|_2 / \|P(\alpha^B)\|_2 \approx 0.016$ — see the white dot in the bottom, left frame in Figure 3.5. This difference is in the middle of that seen for all the combinations of parameters, so the conclusions made in the chapter can be considered as relevant for a wider range of systems. Table 3.1 indicates that the result from §3.2.1.1 is not restricted to particular choices of parameters but holds more generally: The CEC policy outperforms the Bellman policy for the majority of events at the expense of a stronger underperformance for the remainder of the events. Notably, the CEC policy is better more than 50 % of the time for all the parameter combinations considered in this chapter. We can also see from the table that the implicit risk-seeking nature of the CEC policy increases with the relative L^2 distance, as its profit distribution widens compared to the Bellman policy. In particular, the frequency at which the CEC policy outperforms the Bellman policy increases at the expense of a larger underperformance, or tail loss, in the remaining realisations. The values in Table 3.1 were approximated using 10 000 samples from W, and their corresponding parameter combinations are shown as grey dots in Figure 3.5.

3.2.3 Open-Loop Feedback Control policy

We conclude this chapter with the example of the OLFC policy (Bertsekas, 2005, Ch. 6). This is another suboptimal policy, similar to the CEC policy, but which better takes into account the uncertainty in the system. The OLFC policy works as follows: At each decision time, a stochastic optimisation problem for the remaining decision horizon is solved, based on the most recent quantification of the uncertainty in the system. Only the decision for the current time step is used, whilst the subsequent decisions are discarded. For each decision time $t = 0, 1, \ldots, T-1$, the OLFC policy

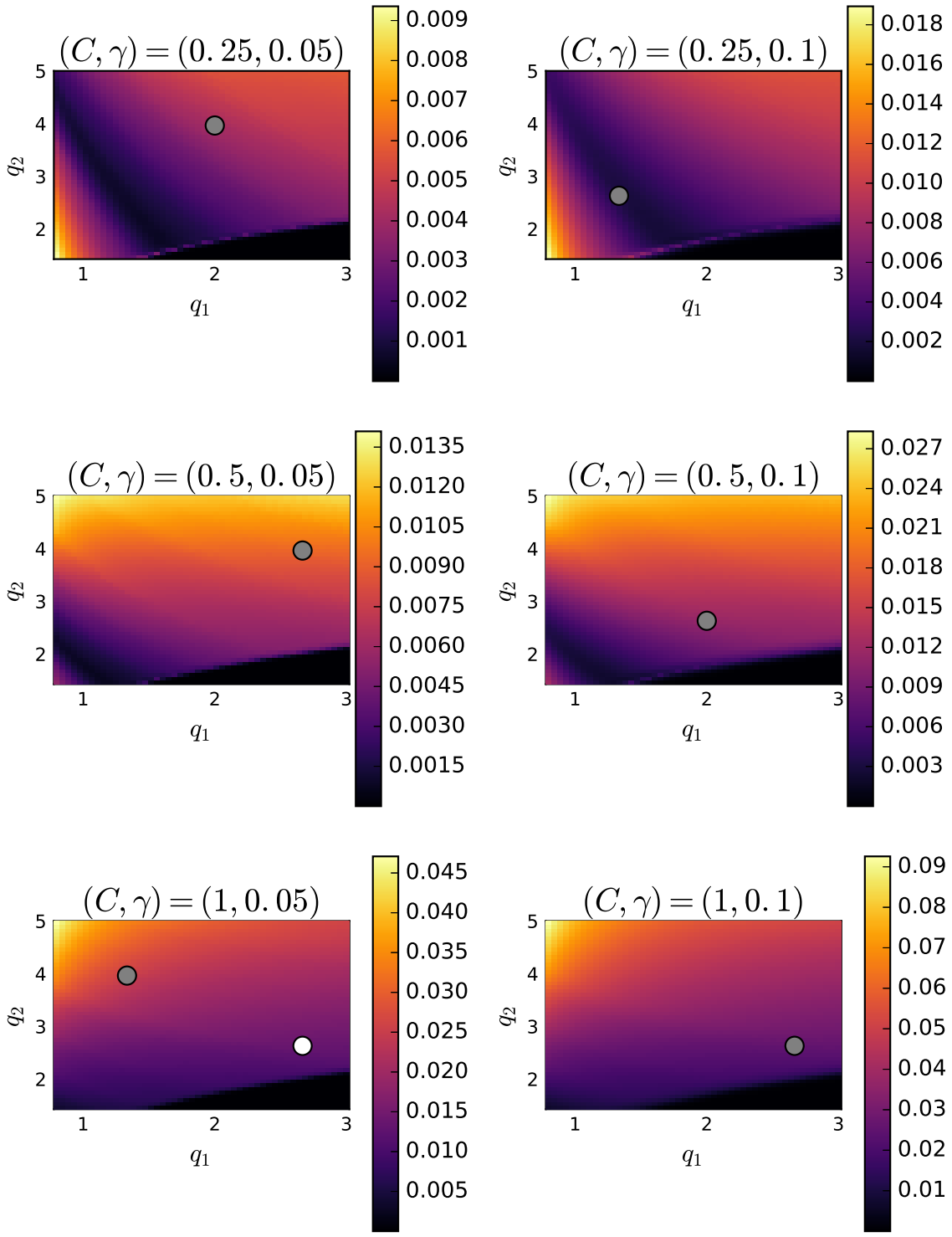


Figure 3.5: The relative L^2 difference $\|P(\alpha^B) - P(\alpha^C)\|_2 / \|P(\alpha^B)\|_2$ representing how different the CEC policy is from the optimal Bellman policy. Each plot defines a combination of (C, γ), whilst the axes vary the parameters $q^{(1)}, q^{(2)}$ of the demand function $q(a) = q^{(1)} e^{-q^{(2)} a}$. The white dot on the bottom, left plot corresponds to the parameters chosen for the experiments in this chapter. The gray dots correspond to the parameters in Table 3.1.

C	γ	$q^{(1)}$	$q^{(2)}$	$\mathcal{Q}_{0.05}$	Median	$\mathcal{Q}_{0.95}$	L^2
0.25	0.05	2.0	4.0	**-0.4**	**-0.3**	**0.6**	**0.4**
0.25	0.1	1.33	2.67	**-0.5**	**-0.0**	**0.6**	**0.3**
0.5	0.05	2.67	4.0	**-0.6**	**-0.6**	**1.9**	**1.1**
0.5	0.1	2.0	2.67	**-0.9**	**-0.6**	**1.9**	**1.2**
1.0	0.05	1.33	4.0	**-1.2**	**-1.1**	**5.3**	**2.7**
1.0	0.1	2.67	2.67	**-1.5**	**-1.3**	**5.8**	**2.9**
				$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-2}$	$\times 10^{-2}$

Table 3.1: Statistics comparing profits $P(\alpha^B)$ and $P(\alpha^C)$ for six different parameter combinations. The columns $\mathcal{Q}_s$ represent the the s^{th} quantile of the relative difference $1 - P(\alpha^C)/P(\alpha^B)$. The values in the column L^2 refers to the relative L^2 difference $\|P(\alpha^B) - P(\alpha^C)\|_2 / \|P(\alpha^B)\|_2$ from Figure 3.5. Each of the parameter combinations correspond to a grey dot in Figure 3.5.

for the pricing problem calculates the current price using the following steps:

1. Observe the state s.

2. Solve the optimisation problem

$$\max_{\mathbf{a} \in A^{T-t}} \mathbb{E}_W \left[\sum_{\tau=t}^{T-1} \mathbf{a}_\tau Q(S_\tau^{\mathbf{a}}, \mathbf{a}_\tau, W_{\tau+1}) - C S_T^{\mathbf{a}} \mid S_t^{\mathbf{a}} = s \right]. \tag{3.18}$$

3. Set the price corresponding to the element $\mathbf{a}_t \in A$ of a maximiser to (3.18).

For this chapter, we approximate the expectation operator with Monte Carlo using 1000 samples, in the same way that the expectation in the Bellman policy is computed. Note that good approximations for the expectation in Step 2 can be prohibitively expensive for real-world systems.

As was done for the CEC policy, we compare the performance of following the OLFC policy α^O with the performance of following the Bellman policy α^B. With the parameters from the example system in §3.1.3, the empirical distribution of $P(\alpha^B) - P(\alpha^O)$ is shown in Figure 3.6. The empirical distribution is generated using 10 000 samples of W. There are two notable differences in this distribution from the distribution of $P(\alpha^B) - P(\alpha^C)$ shown in Figure 3.4. First, the distribution appears to be unimodal and more concentrated around zero. Second, the values are an order of magnitude smaller. This supports the claim that the OLFC policy better approximates the Bellman policy.

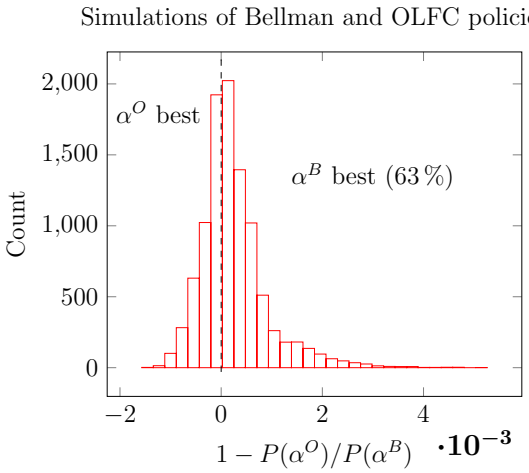


Figure 3.6: This shows the distributions from 10 000 samples of the profits of following the Bellman and OLFC policies. The OLFC policy generates profits an order of magnitude closer to the Bellman policy than the CEC policy (Figure 3.4). This distribution appears to be unimodal.

For completeness, Table 3.2 includes the statistics of the relative difference $1 - P(\alpha^O)/P(\alpha^B)$ for the same parameters that were used to compare the CEC and Bellman policies in Table 3.1. The distributions of the relative difference indicate a more symmetric distribution around the median, with values of an order of magnitude smaller than in the CEC comparison.

C	γ	$q^{(1)}$	$q^{(2)}$	$\mathcal{Q}_{0.05}$	Median	$\mathcal{Q}_{0.95}$	L^2
0.25	0.05	2.0	4.0	**-0.6**	**0.1**	**1.2**	**0.6**
0.25	0.1	1.33	2.67	**-2.5**	**0.4**	**4.2**	**2.1**
0.5	0.05	2.67	4.0	**-0.5**	**0.1**	**1.2**	**0.6**
0.5	0.1	2.0	2.67	**-2.3**	**0.3**	**4.6**	**2.1**
1.0	0.05	1.33	4.0	**-0.8**	**0.0**	**2.8**	**1.1**
1.0	0.1	2.67	2.67	**-2.2**	**0.4**	**5.8**	**2.4**
				$\times 10^{-3}$	$\times 10^{-3}$	$\times 10^{-3}$	$\times 10^{-3}$

Table 3.2: Statistics comparing the relative profits of $P(\alpha^O)$ against $P(\alpha^B)$ for six different parameter combinations. Note that the values are an order of magnitude smaller than in the comparison between Bellman and CEC shown in Table 3.1

Note that the CEC policy is a special case of the OLFC policy, where the expectation (3.18) is approximated with (3.13), a zeroth-order expansion around the estimate $w_{t+1}, \ldots, w_T$. The OLFC policy is thus much more costly to compute than the CEC policy, but will also better approximate the optimal Bellman policy. In

practice, the decision maker can balance the cost-versus-optimality by how accurately they approximate the expectation (3.18).

3.3 Discussion

In this chapter, we have looked at a mathematical formulation of a retail pricing problem for profit maximisation, and have investigated the performance of two algorithms that balance practicality with the degree of suboptimality against an "exact" Bellman solution. The motivation is to better understand how well the suboptimal policies approximate the returns from the optimal policy. Pricing problems are often formulated as an expected value maximisation, but different algorithms may induce different distributions of the profits. Even though the *expected values* of suboptimal policies are not better than the optimal policy, they may have a higher profit for many realisations of the underlying probability distribution. We found in §3.2 that there are a large number of reasonable system parameters for which the suboptimal Certainty Equivalent Control policy resulted in a higher profit than the optimal policy in more than half of the realisations. In some of the remaining realisations, however, the CEC policy resulted in much smaller profits. We interpret these results as an indication that the suboptimal policy is more risk-seeking, in a colloquial sense of the term, than the optimal policy: it puts more weight on better-than-average outcomes and less weight on worse-than-average outcomes. The results in this chapter underscore the importance of looking at the impact of different suboptimal algorithms have on the distribution of an objective, and not only the impact of marginalised statistics such as the expected value; particularly if the actual optimisation only occurs for a single realisation.

The model problem in this chapter is simple, and we propose three specific lines for future research that take this analysis closer to practical models. First, to investigate two- and three-product problems where the demand and availability of one product depend on the other products. Second, to introduce a state dependence in the uncertainty in the system, that is, to allow W_{t+1} to depend explicitly on the value of W_t. Third, to try and formalize the "risk-seeking" formulation of the CEC method. We hope that these two extensions will lead to a better understanding of whether the effects seen in this chapter will be stronger or diluted in real-life problems.

Chapter 4

Continuous-time pricing with diffusion models

In the previous two chapters we considered decisions with no temporal structure and with a discrete, sequential temporal structure. As the frequency of sequential decisions increases there may be an advantage in using a continuum approximation. We consider a continuous-time version of the pricing problem from Chapter 3 to motivate such decision problems.

Consider a monopolist retailer who wants to design a dynamic pricing policy for a product over a given period, in order to maximise their total revenue and minimise the cost associated with handling unsold items at a given terminal time. Two important components in the decision process are the ability to take into account the uncertainty associated with future cost and demand and to optimally adjust for new knowledge as it arrives. We illustrate how to address both components in this chapter, focusing on large-inventory limits and multiplicative demand uncertainty. Assume the retailer sells large volumes of its product at high frequency. In such a setting, it is appropriate to model the sales process in a continuum limit, both for product volume and for time, similar to Kalish (1983).

In the revenue management literature, most efforts to model demand uncertainty in continuous time have focused on discrete Poisson processes. See, for example, the overviews by Bitran and Caldentey (2003) or Aviv and Vulcano (2012). We stress that in other communities, such as financial markets, both diffusion processes and jump-diffusions are common for modelling demand and spot-prices (Benth et al., 2014). The survey by Carmona and Coulon (2014) gives an indication of how flexible more advanced models used for commodity markets can be. With this chapter, we wish to inspire the revenue management community to take advantage of this research

when modelling uncertainty in situations where it may be more relevant than Poisson processes. For the remainder of this section, however, we mainly focus on modelling in the revenue management literature.

A pure Poisson process assumption is not compatible with taking a time-continuum limit for sales volume with demand uncertainty and may be better suited for demand modelling in industries with lower product sales volumes, such as the airline and hotel industries. Our model should therefore scale better to large-inventory problems, which is more relevant to retail chains. Maglaras and Meissner (2006) and Schlosser (2015a,b) propose pricing heuristics similar to those of this chapter, by considering a deterministic model based on the asymptotic continuum limit of Poisson processes. To the author's knowledge, however, few attempts have been made to unify demand uncertainty with the continuum limit. For example, Raman and Chatterjee (1995) and Wu and Wu (2016) model demand uncertainty as increments of a Brownian motion. As we show in this chapter, their approaches lead to demand processes that admit negative sales, with a probability approaching 1/2 over infinitesimally small time periods. We believe this is an important factor for why so little research has been done in this area. This chapter proposes a different approach to modelling demand uncertainty in order to remedy this. In our approach, the parameters of the system are described as diffusion processes that are solutions to stochastic differential equations (SDEs). This enables modelling of the demand volatility, which Poisson processes do not. Our proposed approach can be combined with parameter estimation methods already used by retailers and also extends naturally to multiple products. Modelling the demand over time as a diffusion process has previously been done by Chambers (1992) at the macroeconomic scale with UK national data. His focus was on the data assimilation aspect and was not applied to a setting of optimal control.

The retailer's dynamic pricing policies are stochastic processes that control the SDE which describes the depletion of stock. In this chapter, we seek an optimal pricing policy that maximises the expected value of the profit over a given pricing period. One way to find an optimal pricing policy is to solve an associated nonlinear partial differential equation (PDE), known as the Hamilton-Jacobi-Bellman (HJB) equation (Pham, 2009). We provide closed-form solutions for the HJB equation in the deterministic case, for both linear and exponential demand functions. The solution identifies two pricing regimes, one where the retailer maximises their profits without depleting the inventory and another where the retailer aims to maximise the price and still deplete its inventory. Xu and Hopp (2006) have considered a continuous-time pricing problem where the uncertainty is modelled by a geometric Brownian motion.

Their expected demand function is unbounded, with the result that the optimal pricing strategy ensures that all stock is sold by the terminal time. The demand functions we consider are bounded, and by introducing a penalty on unsold stock we capture the pricing regime change that does not arise in Xu and Hopp (2006).

In financial markets traders face a similar problem to that presented in this chapter, known as the optimal execution, or liquidation, problem. There, a trader tries to sell, or purchase, a particular amount of an asset by a predetermined time. See, for example, Cartea et al. (2015) for an overview of this problem. Instead of controlling the price, the focus of the retailer pricing problem, the trader directly controls how much of the product to sell at a particular time. If the expected demand model is invertible, the retailer's pricing problem can be reformulated to control the expected amount of stock to sell at each time. This reformulation is sometimes chosen in the revenue management community as well, see, for example, Bitran and Caldentey (2003). In this chapter we focus on the formulation that writes the expected demand model in terms of the price.

By investigating the terms in the HJB equation, we identify the cases where the deterministic-case solution is appropriate. Potentially significant changes to the pricing policy for the stochastic system are at the interface between the two pricing regimes — far away from this interface the deterministic pricing policy is near-optimal. For example, the expected price path is decreasing when one takes into account uncertainty, while it is not for the deterministic heuristic. For a risk-neutral decision maker, however, the differences in profit are insignificant for most cases that may be relevant in industry.

The chapter is structured as follows. In §4.1, we describe the modelling of the system and compare the new parameter uncertainty approach to the existing Brownian increments approach. Then, a formulation of the pricing problem and the associated HJB equation is given in §4.2. We also propose a method to estimate the multiplicative factor in our model, in order to implement the pricing policy in practice. The optimal pricing policy in the deterministic limit is covered in §4.3, and the comparison to the stochastic system is shown in §4.4. Extensions to the problem, such as other models for uncertainty, and risk aversion, are discussed in §4.5. Finally, we conclude and suggest avenues for further research in §4.6.

4.1 Modelling demand and uncertainty

For a given, positive amount of initial stock of a product, we are interested in modelling the product sales over some finite time period. Assume that the initial quantity of stock is large, and that there is a substantial volume sold over time periods that are small compared to the total period of interest. These assumptions can apply to many products sold by large retailers, such as lettuce or milk. For example, in the monopoly setting, this leads to a continuum model similar to that of Kalish (1983). For a given product, denote the amount of stock left at time $\hat{t}$ by $\hat{S}(\hat{t})$, and let $\hat{q}(\hat{a})$ represent product demand at price $\hat{a}$, per unit time. For simplicity, we assume $\hat{q}$ does not explicitly depend on time. In the continuum limit, the change in stock at time $\hat{t}$ is thus

$$d\hat{S}(\hat{t}) = \begin{cases} -\hat{q}(\hat{a})\, dt & \text{if } \hat{S}(\hat{t}) > 0, \\ 0 & \text{if } \hat{S}(\hat{t}) \leq 0. \end{cases} \tag{4.1}$$

For a pricing policy $\hat{\alpha}(\hat{t})$, the remaining stock at time $\hat{t}$ is then

$$\hat{S}(\hat{t}) = \hat{S}(0) - \int_0^{\hat{t}} \hat{q}(\hat{\alpha}(\hat{u}))\, d\hat{u}. \tag{4.2}$$

In the remainder of this chapter, we emphasise the dependence of remaining stock on a particular pricing policy $\hat{\alpha}(\hat{t})$ using the superscript $\hat{S}^{\hat{\alpha}}$.

At the start of the prediction period, it is not known exactly what the demand will be at future times. We now discuss how to represent this uncertainty in the model. First, we note that the Brownian noise approach of Raman and Chatterjee (1995) and Wu and Wu (2016) leads to negative sales with probability tending to 0.5 as the time period goes to zero. Then, we propose a method that guarantees non-negative sales over all time periods. Let $\hat{W}(\hat{t})$ denote a Brownian motion (Øksendal, 2000), and let $\hat{\sigma}(\hat{t}, \hat{s}, \hat{a})$ be the volatility in demand as a function of time, stock, and price. Then one may say that the uncertainty in future sales is due to the changes in $\hat{W}(\hat{t})$, in the following sense,

$$d\hat{S}^{\hat{\alpha}}(\hat{t}) = \begin{cases} -\hat{q}(\hat{\alpha}(\hat{t}))\, d\hat{t} + \hat{\sigma}(\hat{t}, \hat{S}(\hat{t}), \hat{\alpha}(\hat{t}))\, d\hat{W}(\hat{t}) & \text{if } \hat{S}(\hat{t}) > 0, \\ 0 & \text{if } \hat{S}(\hat{t}) \leq 0. \end{cases} \tag{4.3}$$

In Figure 4.1 we can see six realisations of $\hat{S}^{\hat{\alpha}}(\hat{t})$ under this model, using $\hat{q}(\hat{a}) = 1$ and $\hat{\sigma}(\hat{t}, \hat{s}, \hat{a}) = 0.05$. As we zoom in on the sales paths, it is obvious that the stock often increases over short time periods, corresponding to negative sales.

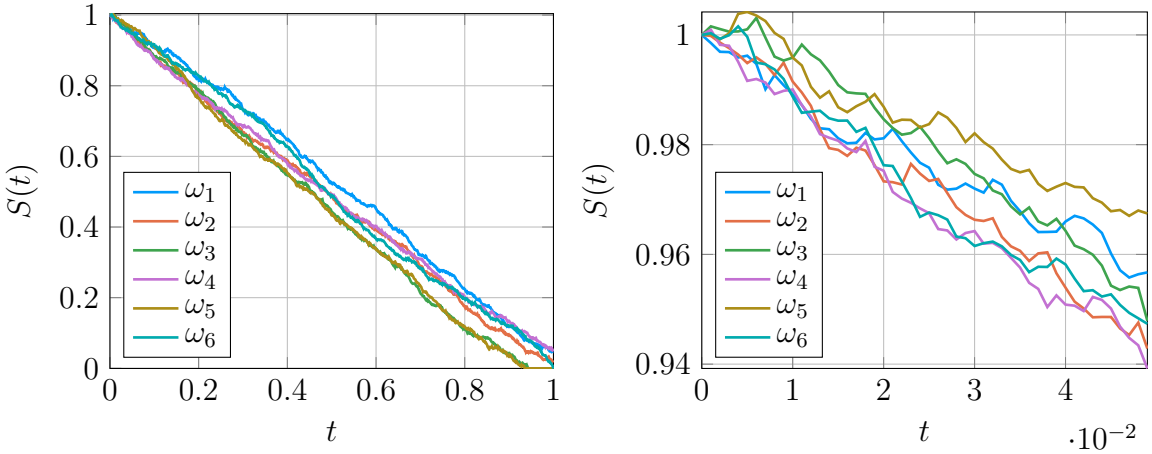


Figure 4.1: Evolutions of stock $\hat{S}^{\hat{\alpha}}(\hat{t})$ for a constant demand forecast $\hat{q}(\hat{a}) = 1$ with different sample paths ω_k of a Brownian motion. The right hand figure shows that over small timescales, sales are often negative. All values are described in dimensionless quantities for simplicity, see §4.1.1.

We now show that for any model with $\hat{\sigma} > 0$, the Brownian noise model (4.3) causes negative sales with high probability. For a given time period $[0, \Delta\hat{t})$, assume the demand and volatility are constant, $\hat{q}(\hat{a}) = \tilde{q} > 0$ and $\hat{\sigma}(\hat{t}, \hat{s}, \hat{a}) = \tilde{\sigma} > 0$. Then $\hat{S}(\Delta\hat{t}) = \hat{S}(0) - \tilde{q}\Delta\hat{t} + \tilde{\sigma}\hat{W}(\Delta\hat{t})$. Sales are negative in this period if $\hat{S}(\Delta\hat{t}) > \hat{S}(0)$, that is when $\tilde{q}\Delta\hat{t} + \tilde{\sigma}\hat{W}(\Delta\hat{t}) < 0$. The distribution of Brownian motion is normally distributed as $\hat{W}(\Delta\hat{t}) \sim \mathcal{N}(0, \Delta\hat{t})$. So $\hat{W}(\Delta\hat{t})$ is equal in distribution to $\sqrt{\Delta\hat{t}}Z$, where $Z \sim \mathcal{N}(0, 1)$, and thus

$$\mathbb{P}(\hat{S}(\Delta\hat{t}) > \hat{S}(0)) = \mathbb{P}(Z \leq -\sqrt{\Delta\hat{t}}\,\tilde{q}/\tilde{\sigma}) \to 0.5^{-}, \qquad \text{as } \Delta\hat{t} \to 0^{+}. \tag{4.4}$$

We therefore argue that it is generally inappropriate to model the uncertainty in future demand with increments of a Brownian motion. Instead, the uncertainty in demand could be modelled more realistically by describing the evolution of parameters in the demand function. This chapter focuses on multiplicative demand uncertainty, which may arise from estimates of seasonality or long-term trends in demand. As such, we introduce a multiplicative parameter $\hat{g} \geq 0$, and consider the demand function given by $(\hat{a}, \hat{g}) \mapsto \hat{q}(\hat{a})\hat{g}$. Assume that the parameter $\hat{g}$ is not directly affected by price, that occurs through $\hat{q}(\hat{a})$, but only represents exogenous information outside the retailer's control. For our purposes, we model the multiplicative parameter as a

geometric Brownian motion (GBM),

$$\hat{G}(\hat{t}) = \exp\left(-\frac{1}{2}\hat{\sigma}^2\hat{t} + \hat{\sigma}\hat{W}(\hat{t})\right), \tag{4.5}$$

with volatility coefficient $\hat{\sigma} \geq 0$, with $\hat{\sigma} = 0$ corresponding to a deterministic model. The GBM approach is also considered by Xu and Hopp (2006) for a similar dynamic pricing problem to that presented in this chapter, but with a different choice of demand function $\hat{q}(\hat{a})$ and unsold stock cost. We restrict ourselves to a GBM with no drift to focus on the impact of the uncertainty, rather than modelling time-dependent behaviour such as seasonality. For any $\Delta\hat{t} \geq 0$, some relevant properties of $\hat{G}(\hat{t})$ are as follows.

$$\hat{G}(0) = 1, \tag{4.6}$$

$$\mathbb{E}[\hat{G}(\hat{t} + \Delta\hat{t}) \mid \hat{G}(\hat{t})] = \hat{G}(\hat{t}), \qquad \text{martingale property,} \tag{4.7}$$

$$\mathrm{Var}[\hat{G}(\hat{t} + \Delta\hat{t}) \mid \hat{G}(\hat{t})] = \hat{G}(\hat{t})^2(e^{\hat{\sigma}^2\Delta\hat{t}} - 1), \qquad \text{increasing variance.} \tag{4.8}$$

With this model, we expect future demand $\hat{q}(\hat{a})\hat{G}(\hat{t})$ at a given price $\hat{a} \geq 0$ to be the current experienced demand, but with decreasing certainty the further ahead we forecast. The SDE governing the system, started at $\hat{S}(0) > 0$, $\hat{G}(0) = 1$, is

$$\begin{aligned} d\hat{G}(\hat{t}) &= \hat{\sigma}\hat{G}(\hat{t})\, d\hat{W}(\hat{t}), \\ d\hat{S}^{\hat{\alpha}}(\hat{t}) &= -\hat{q}(\hat{\alpha}(\hat{t}))\hat{G}(\hat{t})\, d\hat{t}, \qquad \text{stopped at zero.} \end{aligned} \tag{4.9}$$

The sample paths of $\hat{S}^{\hat{\alpha}}(\hat{t})$ following the GBM model, as shown in Figure 4.2, are more regular than of the Brownian noise model.

4.1.1 Non-dimensionalised system

In order to capture similarities between different pricing decisions, irrespective of units such as a particular currency, it is helpful to work in a dimensionless system. We thus non-dimensionalise the model, which also helps to reduce the number of parameters in the decision problem. The units at play in our system are the time and the product price, for example measured in weeks and £. If $\hat{s}, \hat{a}, \hat{t}$ denote unscaled quantities of stock, price, and time respectively, we rescale them with dimensionless quantities

$$s = \frac{\hat{s}}{\hat{S}(0)}, \qquad t = \frac{\hat{t}}{\hat{T}}, \qquad a = \frac{\hat{a}}{\bar{a}}. \tag{4.10}$$

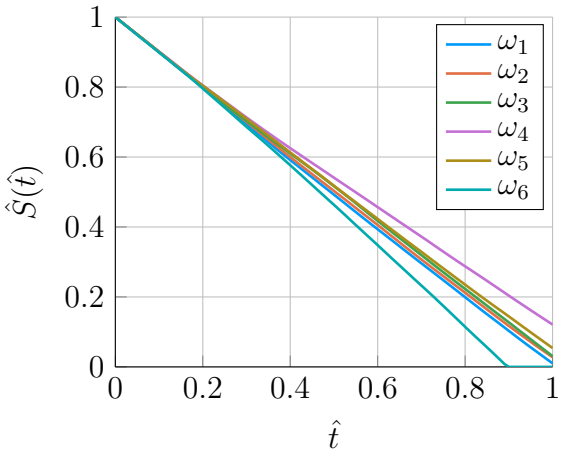


Figure 4.2: Sample paths of $\hat{S}^{\hat{\alpha}}(\hat{t})$ following the GBM model (4.9), with $\hat{q}(\hat{\alpha}(\hat{t})) = 1$, and GBM volatility $\hat{\sigma} = 0.1$. The paths are more regular than that of the Brownian noise model shown in Figure 4.1.

Here, $\hat{S}(0)$ is the initial quantity of stock, $\hat{T}$ is the time-horizon for the pricing problem, and $\bar{a}$ is some reference price chosen to make typical prices a continuous and of order one. In the following sections we choose $\bar{a}$ based on the demand function. This scaling means that we work solely with stock and time on the unit interval, $s, t \in [0, 1]$. For typographical reasons we define the upper limit of the time horizon as T, which means $T = 1$ and that

$$t \in [0, T]. \tag{4.11}$$

In order to write down a dimensionless formulation of the system's SDE (4.9), we need to work with the expected demand function, volatility parameter, and Brownian motion defined as

$$q(a) = \frac{\hat{T}}{\hat{S}(0)}\hat{q}(a\bar{a}), \qquad \sigma = \sqrt{\hat{T}}\hat{\sigma}, \qquad W(t) = \frac{\hat{W}(\hat{t})}{\sqrt{\hat{T}}}. \tag{4.12}$$

It follows that the multiplicative term is given by

$$\hat{G}(\hat{t}) = \exp\left(-\tfrac{1}{2}\hat{\sigma}^2\hat{T}t + \hat{\sigma}\sqrt{\hat{T}}\,W(t)\right) \tag{4.13}$$

$$= \exp\left(-\tfrac{1}{2}\sigma^2 t + \sigma W(t)\right). \tag{4.14}$$

So we define dimensionless versions of the process $\hat{G}(\hat{t})$ using

$$G(t) = \hat{G}(\hat{T}t), \quad g = \hat{g}. \tag{4.15}$$

Thus, for a given pricing policy $\alpha(t)$, the system starts at $S(0), G(0) = 1$, and evolves according to

$$\begin{aligned} dG(t) &= \sigma G(t)\, dW(t), \\ dS^{\alpha}(t) &= -q(\alpha(t))G(t)\, dt, \qquad \text{stopped at zero.} \end{aligned} \tag{4.16}$$

4.2 Formulation of the decision problem

We restate the retailer's objective which guides the choice of pricing policy: Over some decision horizon, continuously adjust the price of a given product in order to maximise the total profit generated from sales revenue minus the cost of handling unsold items at the terminal time. We formulate this mathematically as a continuous-time stochastic optimal control problem, for which the optimal pricing policy can be found by solving the associated HJB equation. In §4.2.1 we address the real-world restrictions of the continuous-time assumption.

Let $C > 0$ denote the handling cost per unit of stock at the terminal time. Define T_h to be the hitting time $T_h = \min\{T, T_0\}$, where $T_0 = \inf\{t \geq 0 \mid S^{\alpha}(t) = 0\}$. The total profit accrued from a pricing policy $\alpha(t)$ is then

$$P(\alpha) = \int_0^{T_h} \alpha(u) q(\alpha(u)) G(u)\, du - C S^{\alpha}(T). \tag{4.17}$$

Note that we focus on time horizons where we believe discounting future cash is negligible. This profit is a random variable that depends on the event ω or, equivalently, the path of the Brownian motion $W(t)$. In the context of this chapter, we assume a retailer that wants to maximise the expected profit $\mathbb{E}[P(\alpha)]$.

Further, we restrict the product price to be in some closed interval $A \subset \mathbb{R}_{\geq 0}$. In addition, we only look for Markovian pricing policies in an admissible set $\mathcal{A}$. We say that $\alpha(t)$ is Markovian if there is a function of the form $a(t, s, g)$ such that for each event ω,

$$\alpha(t)(\omega) = a(t, S^{\alpha}(t)(\omega), G(t)(\omega)) \in A. \tag{4.18}$$

Thus, we seek pricing policies that set the price at time t, based on the knowledge of the state at that time. We also assume that $\mathcal{A}$ only contains pricing policies such that the integral in (4.17) exists and $P(\alpha)$ is integrable. We can now state the mathematical problem we seek to solve in the remainder of the chapter.

Definition 4.1 (Pricing problem)**.** In order to maximise the retailer's expected profit, find a solution to the stochastic optimal control problem

$$\max_{\alpha \in \mathcal{A}} \mathbb{E}[P(\alpha)]. \tag{4.19}$$

Our chosen strategy for finding the corresponding pricing function $a^*(t, s, g)$ of a maximiser $\alpha^* \in \mathcal{A}$ is to solve the associated HJB equation for the pricing problem. For a detailed explanation of the theory behind stochastic optimal control and HJB equations, see, for example, Pham (2009). The HJB equation is a nonlinear PDE, where the solution describes the value of being in a particular state. We assume that there exists a solution to the HJB equation but do not worry about uniqueness in this chapter. From the value function defined below, one can calculate the optimal pricing function $a^*(t, s, g)$.

If at time t, we know the value of $S(t)$ and $G(t)$, then the value function represents the expected value of applying the optimal pricing policy for the remainder of the pricing period. We define the value function $v(t, s, g)$ for $t \leq T$ and $s, g \geq 0$ by

$$v(t, s, g) = \max_{\alpha \in \mathcal{A}} \mathbb{E}_t \left[\int_t^{T_h} \alpha(u) q(\alpha(u)) G(u)\, du - C S^\alpha(T) \right], \tag{4.20}$$

$$v(t, 0, g) = 0, \tag{4.21}$$

$$v(t, s, 0) = -Cs. \tag{4.22}$$

The subscript on the expectation denotes that we condition on $S^\alpha(t) = s$ and $G(t) = g$. In the limit $t \to T$, we see from (4.20) that $v(T, s, g) = -Cs$.

We derive the HJB equation for v in a non-rigorous manner using a continuous-time analogue of the Dynamic Programming principle stated in §3.1.2. For a rigorous proof, see, for example, Pham (2009, Ch. 3). Let $\Delta t > 0$ and, for brevity, denote

$$v^{(\Delta t)} = v(t + \Delta t, S^\alpha(t + \Delta t), G(t + \Delta t)) \tag{4.23}$$

Then, as in (3.10) from §3.1.2, we can split (4.20) into two terms

$$v(t, s, g) = \max_{\alpha \in \mathcal{A}} \mathbb{E}_t \left[\int_t^{t+\Delta t} \alpha(u) q(\alpha(u)) G(u)\, du + v^{(\Delta t)} \right] \tag{4.24}$$

using the tower rule $\mathbb{E}_t[\cdot] = \mathbb{E}_t[\mathbb{E}_{t+\Delta t}[\cdot]]$. It follows that

$$\mathbb{E}_t\left[\int_t^{t+\Delta t} \alpha(u)q(\alpha(u))G(u)\,du + v^{(\Delta t)} - v(t,s,g)\right] \leq 0 \quad \forall \alpha \in \mathcal{A}, \tag{4.25}$$

with equality achieved for a maximiser $\alpha^* \in \mathcal{A}$. Consider an arbitrary $\alpha \in \mathcal{A}$ and let $a = \alpha(t) \in A$ denote its value at time t. We have the following Taylor-expansion of the expectation of the integral of $\alpha(u)q(\alpha(u))G(u)$.

$$\mathbb{E}_t\left[\int_t^{t+\Delta t} \alpha(u)q(\alpha(u))G(u)\,du\right] = \Delta t\, aq(a)g + \mathcal{O}(\Delta t^2) \tag{4.26}$$

By Itô's lemma (Pham, 2009, Ch. 1), we know that

$$\mathbb{E}_t[v^{(\Delta t)}] - v(t,s,g) = \mathbb{E}_t \int_t^{t+\Delta t} (v_t + \tfrac{\sigma^2 g^2}{2} v_{gg} - q(\alpha(u))gv_s)(u, S^\alpha(u), G(u))\,du \tag{4.27}$$

$$= \Delta t \left(v_t + \tfrac{\sigma^2 g^2}{2} v_{gg} - q(a)gv_s\right)(t,s,g) + \mathcal{O}(\Delta t^2). \tag{4.28}$$

The subscripts on v denote partial derivatives with respect to the given argument. Introducing the Taylor expansion into (4.25) yields

$$v_t + \tfrac{\sigma^2 g^2}{2} v_{gg} + g(a - v_s)q(a) + \mathcal{O}(\Delta t) \leq 0, \qquad \forall a \in A. \tag{4.29}$$

The upper bound is reached for a maximiser $a^* = \alpha^*(t)$, and thus taking $\Delta t \to 0$ gives us the the local representation

$$v_t + \frac{\sigma^2 g^2}{2} v_{gg} + g(a^* - v_s)q(a^*) = 0, \tag{4.30}$$

valid for $t < T$ and $s, g > 0$. Together with the boundary and terminal conditions on v, this constitutes the HJB equation for the pricing problem,

$$v_t(t,s,g) + \frac{\sigma^2}{2} g^2 v_{gg}(t,s,g) + g \max_{a \in A}\{(a - v_s(t,s,g))q(a)\} = 0, \tag{4.31}$$

$$v(t,0,g) = 0, \tag{4.32}$$

$$v(t,s,0) = -Cs, \tag{4.33}$$

$$v(T,s,g) = -Cs. \tag{4.34}$$

If we know v, the optimal pricing policy $\alpha^*(t) = a(t, S^\alpha(t), G(t))$ can be calculated

from the univariate optimisation problem

$$a^*(t,s,g) = \arg\max_{a\in A}\{(a - v_s(t,s,g))q(a)\}. \tag{4.35}$$

In §4.3 and §4.4, solutions for the pricing problem are found via the HJB equations with linear and exponential demand functions.

Remark 4.1. The value function is not necessarily sufficiently smooth to satisfy the HJB equation (4.31) in the classical sense. This is indeed the case for some examples in this chapter when $\sigma = 0$, and we must therefore consider the solutions in the viscosity sense. See Pham (2009) for a description of viscosity solutions to HJB equations.

4.2.1 Parameter estimation

In a real retail application we cannot update the price in continuous time, and it is not possible to infer $G(t)$ exactly. To represent real world conditions, we assume that the price is piecewise constant and updated frequently at fixed time points $t_0 < t_1 < \cdots < T_h$. Further, we assume that stock levels are only observed at these time points. When computing an optimal pricing strategy we still consider the continuous-time function (4.35) that can be computed from the HJB equation as the optimal pricing function for the problem. Such a continuous-time approximation to inherently discrete systems is common, for example in pricing of options using the Black-Scholes equations (Black and Scholes, 1973). In order to use a pricing function $a(t_k, s, g)$ we must estimate $G(t_k)$. Due to the Markov properties of $G(t)$ and $S^\alpha(t)$, information about the process for $t < t_{k-1}$ is not needed, and we can estimate $G(t_k)$ based on $\alpha(t_{k-1})$, $S^\alpha(t_{k-1})$, and $S^\alpha(t_k)$. For brevity, let us leave out the superscript α of $S^\alpha(t)$ for the remainder of this section. By assumption, the price has been constant, $\alpha(u) = a_{k-1}$, for the time period $u \in [t_{k-1}, t_k)$. Say $S^\alpha(t_k) > 0$, and that we wish to update the price at time t_k. From (4.16), the SDE describing our system, we have

$$S(t_k) = S(t_{k-1}) - q(a_{k-1}) \int_{t_{k-1}}^{t_k} G(u)\, du. \tag{4.36}$$

In the numerical examples in this chapter, we estimate $G(t_k)$ with $\hat{G}(t_k) = \frac{S(t_{k-1}) - S(t_k)}{q(a_{k-1})(t_k - t_{k-1})}$. We now discuss the derivation and properties of this estimator.

Define $B(t) = \exp(-\sigma^2 t/2 + \sigma\sqrt{t}Z_{k-1})$, where $Z_{k-1} \sim \mathcal{N}(0,1)$. The evolution of $G(t)$ is known in closed form, which gives $G(t_k) = G(t_{k-1})B(\Delta t)$, with $\Delta t = t_k - t_{k-1}$.

From (4.36) it follows that

$$G(t_k) = \frac{S(t_{k-1}) - S(t_k)}{q(a_{k-1})} \frac{B(\Delta t)}{\int_0^{\Delta t} B(u)\,du}. \tag{4.37}$$

So long as $\Delta t = t_k - t_{k-1}$, and the variance of $B(u)$, are sufficiently small, we may use the approximation $\int_0^{\Delta t} B(u)\,du \approx \frac{\Delta t}{2}(B(0) + B(\Delta t)) = \frac{\Delta t}{2}(1 + B(\Delta t))$. Hence, the conditional distribution of $G(t_k)$ is approximated as

$$G(t_k) \mid (a_{k-1}, S(t_{k-1}), S(t_k)) \approx \frac{S(t_{k-1}) - S(t_k)}{q(a_{k-1})} \frac{2}{\Delta t} \frac{B(\Delta t)}{1 + B(\Delta t)} \tag{4.38}$$

$$\approx \frac{S(t_{k-1}) - S(t_k)}{q(a_{k-1})} \frac{1 + B(\Delta t)}{2\Delta t}. \tag{4.39}$$

The second approximate equality comes from the Taylor expansion $\frac{x}{1+x} \approx \frac{1}{4}(1 + x)$ about $x = 1$. This gives us the following expressions for the first two moments of $G(t_k)$:

$$\mathbb{E}[G(t_k) \mid a_{k-1}, S(t_{k-1}), S(t_k)] \approx \frac{S(t_{k-1}) - S(t_k)}{q(a_{k-1})\Delta t}, \tag{4.40}$$

$$\mathrm{Var}[G(t_k) \mid a_{k-1}, S(t_{k-1}), S(t_k)] \approx \frac{1}{4}\left(\frac{S(t_{k-1}) - S(t_k)}{q(a_{k-1})\Delta t}\right)^2 \left(e^{\sigma^2 \Delta t} - 1\right). \tag{4.41}$$

The conditional expectation in (4.40) is equal to our estimator $\hat{G}(t)$. Figure 4.3 shows the distribution of the relative difference $1 - \hat{G}(t)/G(t)$ between the estimate and the true $G(t)$, for the parameters used in §4.4. The relative estimator error when $\sigma = 0.1$ and $\Delta t = 0.01$ is typically within 1 %.

4.3 Solution to the deterministic system

In this section, we provide solutions to the pricing problem in the deterministic case, $\sigma = 0$, for families of linear and exponential demand functions $q_l(a)$ and $q_e(a)$ respectively.

$$q_l(a) = q^{(1)} - q^{(2)} a, \qquad \text{for } q^{(1)}, q^{(2)} > 0, \tag{4.42}$$

$$q_e(a) = q^{(1)} e^{-q^{(2)} a}, \qquad \text{for } q^{(1)}, q^{(2)} > 0. \tag{4.43}$$

These demand functions are often used in the literature. For a discussion about their properties and usage in modelling demand, see Talluri and van Ryzin (2006, Ch. 7).

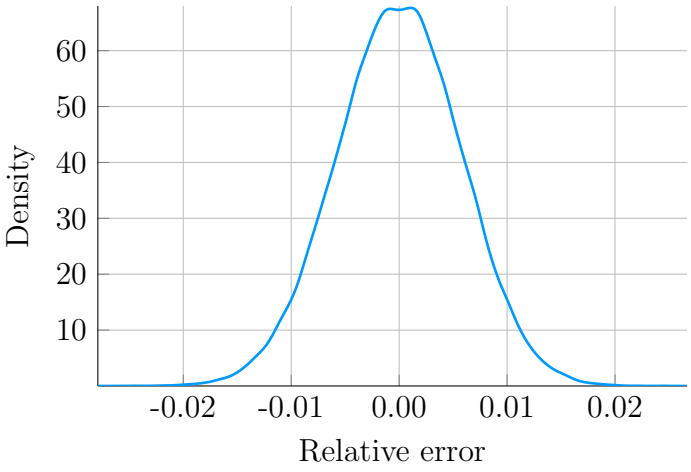


Figure 4.3: Distribution of the relative error in estimating $G(t)$ with $\sigma = 0.1$ and $\Delta t = 0.01$, based on 100 000 samples.

With both demand functions, the optimal pricing function policy is to hold the price constant over the pricing horizon. They translate to the following intuition: For small amounts of stock, relative to demand, sell at the highest price so that all stock is depleted by the terminal time. As the amount of stock increases, the retailer should decrease the price until it arrives at the price that maximises the profits balancing revenue and cost of unsold stock at the terminal time.

4.3.1 Linear demand function

When $q(a) = q^{(1)} - q^{(2)}a$ for $q^{(1)}, q^{(2)} > 0$, the maximisation in the HJB equation can be solved in closed form. To reduce the number of parameters, we can rescale the price per product with $q^{(2)}$ by setting $q^{(2)}a$ to a. Then $q(a) = q^{(1)} - a$, and we set the pricing interval to $A = [0, q^{(1)}]$ so that the demand function is non-negative. Now, use the ansatz that $a^D(t, s, g)$ sets the price so that $\alpha^D(t)$ is constant for the remainder of the pricing period. From the expression of the value function in (4.20), the ansatz pricing policy must also be given by

$$a^D(t,s,g) = \arg\max_{a\in A} \left\{ \int_t^T aq(a)g\,du - C\left(s - \int_t^T q(a)g\,du\right) \mid s \geq \int_t^T q(a)g\,du \right\} \tag{4.44}$$

$$= \arg\max_{a\in A} \left\{ (T-t)(a+C)q(a)g - Cs \mid s \geq (T-t)q(a)g \right\}. \tag{4.45}$$

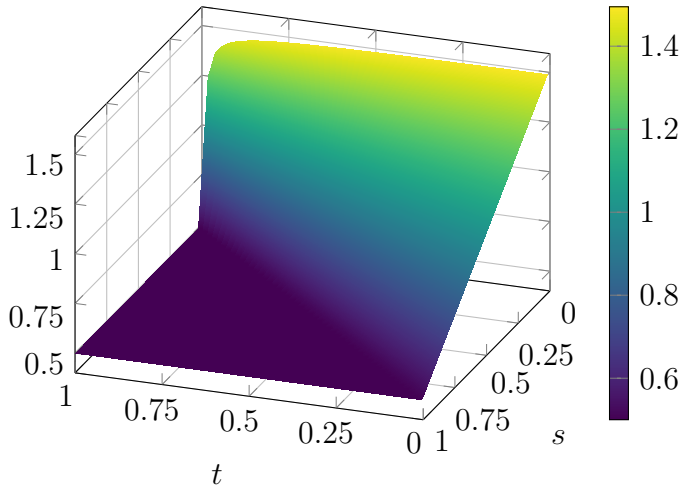
The maximiser therefore satisfies the equality constraints that s equals $(T-t)gq(a)$, is zero, or is in the interior of the feasible set, given by A. For $q(a) = q^{(1)} - a$, we can

verify that this implies

$$a^D(t,s,g) = \begin{cases} q^{(1)} - \frac{s}{(T-t)g}, & \text{if } 0 \le s \le (T-t)g \min\{q^{(1)}, \frac{1}{2}(q^{(1)}+C)\}, \\ \max(0, q^{(1)} - C)/2, & \text{otherwise.} \end{cases} \tag{4.46}$$

It follows that $T_h = T$ when following the optimal pricing strategy. This is an intuitive result, because if $T_h < T$, one can increase the price until $T_h = T$ and $S^\alpha(T) = 0$, which earns extra revenue at no extra cost. The pricing policy suggested by the deterministic assumption provides the following, obvious, heuristics: First, find the price that maximises profits, ignoring inventory constraints. Second, if the sales forecast suggests that you will deplete stock before the end of the time horizon at this price, increase it accordingly. This is consistent with the heuristic proposed by Schlosser (2015a,b), which he finds by considering a deterministic continuum approximation to a Poisson process demand model.

Example 4.1. To demonstrate what form the pricing function may take, Figure 4.4 shows a plot of $a^D(t,s,1)$ for a given set of parameters. We have chosen the parameters so that the pricing problem starts at the most interesting point: at the kink separating the regimes where all the stock is sold out and where it is not. This corresponds to a combination of parameters such that $\min(q^{(1)}, \frac{1}{2}(q^{(1)}+C)) = 1$.



$a^D(t,s,1)$

Figure 4.4: Example optimal, deterministic pricing function $a^D(t,s,1)$ from (4.46), as a function of time and stock. Notice how sensitive $a^D(t,s,1)$ is to changes in s, for large t. The parameters used are $q^{(1)} = 3/2$, and $C = 1/2$. In Figure 4.5 we compare this function to the optimal price when $\sigma = 0.1$.

To verify that $a^D(t,s,g)$ given by (4.46) is the solution to the deterministic pricing problem, we show that it satisfies the HJB equation. On the interior of the (t,s,g)-domain, $a^D(t,s,g)$ must solve

$$\max_{a\in A}\{(a - v_s(t,s,g))(q^{(1)} - a)\}. \tag{4.47}$$

Let $\mathcal{P}_A$ denote the projection of the real line to A. The objective is concave, and hence the maximiser is

$$a^D(t,s,g) = \mathcal{P}_A\left[\frac{q^{(1)} + v_s}{2}\right]. \tag{4.48}$$

Define Γ to be the boundary of the terminal time problem, that is when $t = T$, $s = 0$, or $g = 0$. Let us assume that $C < q^{(1)}$, then the deterministic-case HJB equation for v is

$$v_t(t,s,g) + \frac{g}{4}(q^{(1)} - v_s(t,s,g))^2 = 0, \tag{4.49}$$

$$v(t,s,g) = -Cs, \qquad (t,s,g) \in \Gamma. \tag{4.50}$$

If $C \geq q^{(1)}$, then there are regions where $a^D(t,s,g) = 0$. In particular, this means that for t, s, g such that $s > (T-t)gq^{(1)}$, v must satisfy

$$v_t(t,s,g) - gq^{(1)}v_s(t,s,g) = 0. \tag{4.51}$$

From the two expressions for $a^D(t,s,g)$ in (4.46) and (4.48), the ansatz implies that the value function must satisfy

$$v^D(t,s,g) = \begin{cases} q^{(1)}s - \frac{s^2}{(T-t)g}, & \text{if } 0 \leq s \leq (T-t)g\min\{q^{(1)}, \frac{1}{2}(q^{(1)}+C)\}, \\ -Cs + V(C)(T-t)g, & \text{otherwise.} \end{cases} \tag{4.52}$$

$$V(C) = \begin{cases} \left[\frac{1}{2}(q^{(1)}+C)\right]^2, & C < q^{(1)}, \\ q^{(1)}C, & C \geq q^{(1)}. \end{cases} \tag{4.53}$$

This v does indeed satisfy the deterministic HJB equation given by (4.49)-(4.51), and hence we can conclude that $a^D(t,s,g)$ is an optimal pricing function. Note that v is not smooth for all parameter combinations, and is therefore considered a solution in the viscosity sense, as noted in Remark 4.1.

4.3.2 Exponential demand function

With the same ansatz that was used for the linear demand function, we can also find the optimal, deterministic, pricing function for exponential demand $q(a) = q^{(1)}e^{-q^{(2)}a}$. Indeed, this is true for any demand function for which a closed form solution exists for $\max_{a\in A}\{(a+C)q(a))\}$ and $(T-t)gq(a) = s$. As with the linear demand, we can eliminate the parameter $q^{(2)}$ so that $q(a) = q^{(1)}e^{-a}$, given by replacing $q^{(2)}a$ by a. In the exponential demand case, with $A = [0, \infty)$, the ansatz gives us the optimal pricing function

$$a^D(t,s,g) = \begin{cases} \log \frac{q^{(1)}g(T-t)}{s}, & \text{if } 0 \leq \frac{s}{q^{(1)}g(T-t)} \leq e^{C-1}, \\ \max\left(0, 1-C\right), & \text{otherwise.} \end{cases} \tag{4.54}$$

For completeness, we state the HJB equation for the exponential demand case when $C < 1$, and provide the solution so that the optimality of (4.54) can be verified. The maximiser of $\max_{a\in A}\{(a-v_s)q(a)\}$ in the HJB equation is $a^D = 1 + v_s$. Thus, the value function must satisfy the HJB equation

$$v_t(t,s,g) + gq^{(1)}e^{-1-v_s(t,s,g)} = 0, \tag{4.55}$$

$$v(t,s,g) = -Cs, \qquad (t,s,g) \in \Gamma. \tag{4.56}$$

The viscosity solution of the HJB equation, acquired from the ansatz $a^D(t,s,g)$ in (4.54), is

$$v^D(t,s,g) = \begin{cases} s\log \frac{q^{(1)}g(T-t)}{s}, & \text{if } 0 \leq \frac{s}{q^{(1)}g(T-t)} \leq e^{C-1}, \\ -Cs + V(C)(T-t)g, & \text{otherwise,} \end{cases} \tag{4.57}$$

$$V(C) = q^{(1)}e^{C-1}. \tag{4.58}$$

4.4 Impact of uncertainty

We now discuss to what degree multiplicative uncertainty changes our policy. In this section, we solve the HJB equation numerically with the linear demand function $q(a) = q^{(1)} - a$ defined on $A = [0, q^{(1)}]$, and compare the resulting pricing policy to the deterministic-system policy from the previous section. With the diffusion term in the HJB equation, one can expect the kink in the deterministic pricing function to smooth out. It turns out that the difference between an optimal policy and a heuristic policy based on the solution to the deterministic system is at most $\mathcal{O}(\sigma\sqrt{T-t})$. Further,

numerical tests indicate that the closed-form pricing functions found in the previous section perform sufficiently well in most situations.

We assume in the following that $(q^{(1)} + v_s) \in [0, 2]$ for $t \in [0, T)$, $s, g > 0$, so that the pricing function satisfies $a(t, s, g) = (q^{(1)} + v_s(t, s, g))/2$ as given by (4.48). Using (4.31) and (4.49) it then follows that v should satisfy

$$v_t(t, s, g) + \frac{\sigma^2}{2} g^2 v_{gg}(t, s, g) + \frac{g}{4}(q^{(1)} - v_s)^2 = 0, \tag{4.59}$$

$$v(t, s, g) = -Cs, \qquad (t, s, g) \in \Gamma. \tag{4.60}$$

The numerical solution to the HJB equation is solved with the following procedure:

1. Reformulate the PDE with the similarity transformation $\xi = s/g$ and $v = g\phi$.
2. Truncate the boundary for $\xi \to \infty$ and set an asymptotic Dirichlet boundary condition based on the deterministic-system solution.
3. Approximate the PDE for $\phi(t, \xi)$ with central finite differences and the `Tsit5` time stepping procedure in `DifferentialEquations.jl` (Rackauckas and Nie, 2017), implemented in the Julia programming language (Bezanson et al., 2017).

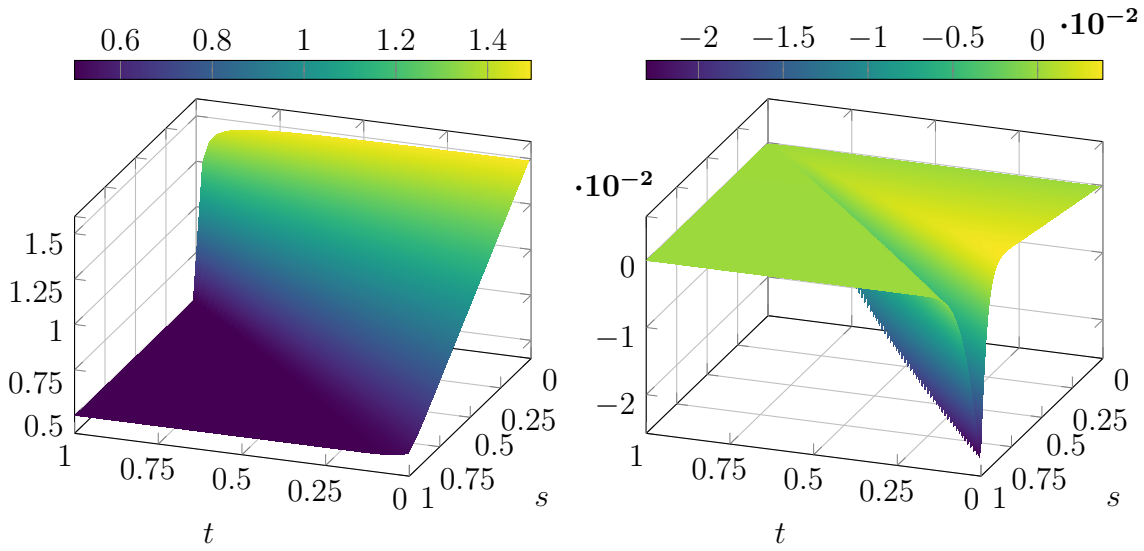
We denote the computed pricing function and pricing policy by $a^B(t, s, g)$ and $\alpha^B(t)$ respectively.

Example 4.2. Let us consider the particular example system used in Example 4.1. That is, a linear demand function $q(a) = q^{(1)} - a$, with $q^{(1)} = 3/2$ and $C = 1/2$. We set the volatility level of $G(t)$ to $\sigma = 0.1$, which corresponds to a true demand near the terminal time within 20 % of the expected demand $q(a)$, with probability 0.95. Figure 4.5 shows the optimal pricing function for $g = 1$, and a plot of the difference $a^B(t, s, g) - a^D(t, s, g)$. The only visible difference is along the kink line, $s = g(T - t)$, where a^B smooths out the transition between the two regions, and hence sells the product at a slightly higher price.

4.4.1 Asymptotic analysis

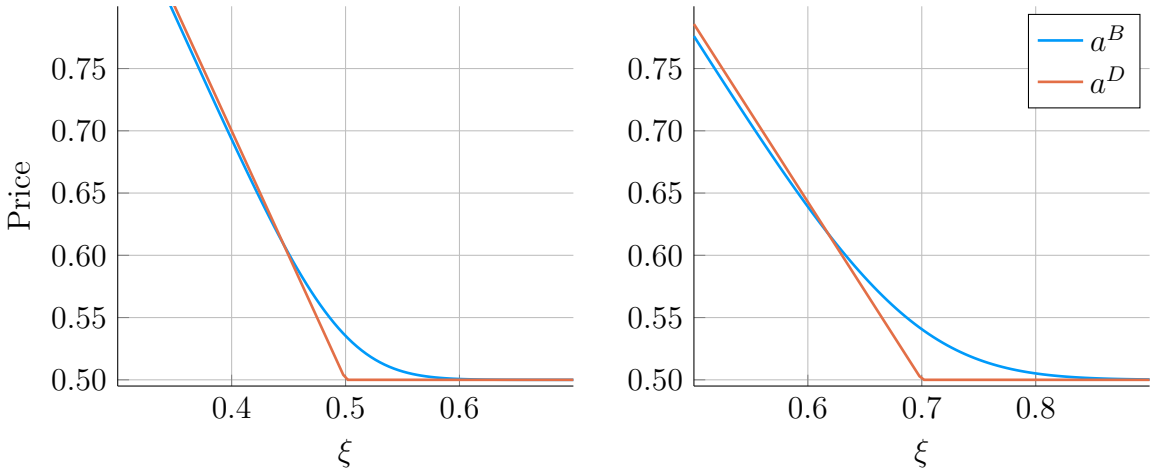
Figure 4.5 indicates that one can do an asymptotic analysis of the impact of $0 < \sigma \ll 1$ on the pricing function. Figure 4.6 provides another visualisation of the differences between a^B and a^D to aid the analysis.

We summarise the results before showing the details of the asymptotic expansions. Define $\beta = \min(q^{(1)}, (q^{(1)} + C)/2)$. There is an inner layer around the surface $\frac{s}{g} =$



(a) $a^B(t, s, 1)$ (b) $(a^D - a^B)(t, s, 1)$

Figure 4.5: The optimal pricing function ($a^B(t, s, g)$, left) smooths out the kink as compared to the deterministic heuristic ($a^D(t, s, g)$, Figure 4.4). The right plot shows the impact of uncertainty on the optimal pricing function: (i) When we do not expect to sell out of the product, the prices are the same. (ii) When we expect to sell out of the product, the deterministic heuristic takes a slightly larger price. (iii) In the transition between the two regions, uncertainty increases the optimal price.



(a) $t = 0.5$ (b) $t = 0.3$

Figure 4.6: A zoomed in comparison of $a^B(t, s, g)$ and $a^D(t, s, g)$ in the transformed coordinate $\xi = s/g$. The difference between a^B and a^D is of order $\sigma\sqrt{T-t}$ at the kink, and $-\sigma^2\xi/2$ as we move to the left in the plots. This example uses $q^{(1)} = 3/2$, $C = 1/2$, and $\sigma = 0.2$.

$(T-t)\beta$ which smooths out the kink of the solution $a^D(t,s,g)$ that arises when $\sigma = 0$. At the kink, the first-order correction in the pricing function is of order $\mathcal{O}(\sigma\sqrt{T-t})$. The width of the layer is of the order $\mathcal{O}(\sigma(T-t)^{3/2})$, which connects the two pricing regimes identified in the deterministic case. As we move from the inner layer to larger values of $\xi := s/g > (T-t)\beta$, the solution tends to the value $\delta = \max(0, q^{(1)} - C)/2$, which coincides with a^D. As we move from the inner layer to smaller values of ξ, the leading order solution of the inner layer coincides with a^D. Further, there is a second order correction in the region $\xi < (T-t)\beta$ equal to $-\sigma^2\xi/2$.

The first and second order corrections reflect the insights one can arrive at when taking into account the deviations of actual future demand from expected demand. At the interface where we expect to sell all the inventory at the optimal lower-bound, "infinite-inventory" price δ, taking into account the possibility of higher future demand means that we can increase the price. When the inventory is so low that we expect to sell it all at some price higher than δ, taking into account the possibility of lower future demand means that we should price the product lower than in the deterministic-case in order to reduce the probability of having excess inventory at the terminal time.

We carry out the asymptotic analysis of the pricing function that arises from the linear demand HJB equation (4.59) under the assumptions of §4.4. Further, we assume that the value and pricing functions are sufficiently differentiable to carry out the operations below. The smoothing effect of the diffusion in the PDEs when $\sigma > 0$ justifies this assumption.

Let us first reduce the dimension of the problem with a similarity transform working in reverse time, and then present a PDE satisfied by the pricing function. Consider the transformations

$$\xi(s,g) = s/g, \tag{4.61}$$

$$\tau(t) = T - t, \tag{4.62}$$

$$v(t,s,g) = g\,\phi(\tau(t), \xi(s,g)), \tag{4.63}$$

which require $\phi(\tau,\xi)$ to satisfy

$$\phi_\tau = \frac{1}{2}\sigma^2\xi^2\phi_{\xi\xi} + \frac{1}{4}(q^{(1)} - \phi_\xi)^2 \tag{4.64}$$

$$\phi(0,\xi) = -C\xi \tag{4.65}$$

$$\phi(\tau,0) = 0, \tag{4.66}$$

$$\phi_\xi(\tau,\infty) = -C. \tag{4.67}$$

In the new coordinates, define the pricing function in terms of the function $\psi(\tau, \xi)$ so that $a^B(t, s, g) = \psi(\tau(t), \xi(s, g))$. Then $\psi = (q^{(1)} + \phi_\xi)/2$ satisfies the following PDE that arises from differentiating (4.64) with respect to ξ.

$$\psi_\tau = \frac{1}{2}\sigma^2\xi^2\psi_{\xi\xi} + (\sigma^2\xi - q^{(1)} + \psi)\psi_\xi, \tag{4.68}$$

$$\psi(\tau, 0) = q^{(1)}, \tag{4.69}$$

$$\psi(\tau, \infty) = \delta := \max(0, (q^{(1)} - C)/2). \tag{4.70}$$

When $\sigma = 0$, we know from §4.3 that the viscosity solution $\psi^{(0)}$ is

$$\psi^{(0)}(\tau, \xi) = \begin{cases} q^{(1)} - \xi/\tau, & \text{if } 0 \le \xi \le \beta\tau, \\ \delta, & \text{if } \quad \xi > \beta\tau, \end{cases} \qquad \text{where } \beta := \min(q^{(1)}, (q^{(1)} + C)/2). \tag{4.71}$$

This leads us to consider an inner layer asymptotic analysis near the kink $\xi = \beta\tau$. Zoom in near the kink using the coordinates

$$x(\tau, \xi) = (\xi - \beta\tau)/\sigma, \tag{4.72}$$

$$\psi(\tau, \xi) = \sigma u(\tau, x(\tau, \xi)) + \delta. \tag{4.73}$$

In the new coordinates (4.68) becomes

$$u_\tau = \frac{1}{2}(\sigma x + \beta\tau)^2 u_{xx} + \sigma(\sigma x + \beta\tau)u_x + uu_x, \qquad \tau > 0,\ x > -\beta\tau/\sigma, \tag{4.74}$$

with matching conditions

$$\lim_{x\to-\infty} u_x(\tau, x) = -1/\tau \qquad \text{and} \qquad \lim_{x\to\infty} u(\tau, x) = 0. \tag{4.75}$$

The leading order equation for $u = u^{(0)} + \sigma u^{(1)} + \sigma^2 u^{(2)} + \cdots$ is therefore

$$u^{(0)}_\tau = \frac{1}{2}(\beta\tau)^2 u^{(0)}_{xx} + u^{(0)}u^{(0)}_x. \tag{4.76}$$

This equation has a similarity solution using the transformations

$$\eta(\tau, x) = x/\tau^{3/2}, \tag{4.77}$$

$$u^{(0)}(\tau, x) = \tau^{1/2} f(\eta(\tau, x)). \tag{4.78}$$

The function $f(\eta)$ must satisfy the boundary-value ODE

$$\beta^2 f'' + (3\eta + 2f)f' - f = 0, \tag{4.79}$$

$$f \to 0 \qquad \text{as } \eta \to \infty, \tag{4.80}$$

$$f' \to -1 \qquad \text{as } \eta \to -\infty. \tag{4.81}$$

Figure 4.7 shows numerically computed solutions to the ODE for different values of β.

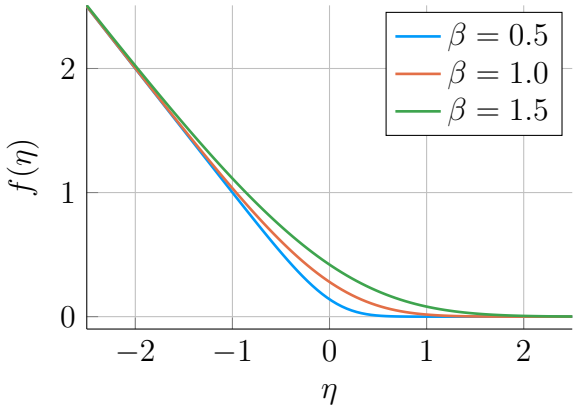


Figure 4.7: Numerically computed solutions of (4.79). Note the similarity to the optimal pricing function curves zoomed in near the kink in Figure 4.6.

We briefly note that we can find a closed form expression for the second-order correction in the outer region $\xi < \beta t$. Let $\psi = \psi^{(0)} + \sigma^2\psi^{(1)} + \sigma^4\psi^{(2)} + \cdots$, where $\psi^{(0)}$ is given in (4.71) and we wish to find $\psi^{(1)}$. By substituting this expansion into (4.68) and matching the σ^2-terms we get the equation

$$\psi^{(1)}_\tau = \psi^{(0)}_\xi(\psi^{(1)} + \xi) + (\psi^{(0)} - q^{(1)})\psi^{(1)}_\xi, \tag{4.82}$$

$$\psi^{(1)}(\tau, 0) = 0. \tag{4.83}$$

When $\xi < \beta$ we have

$$\psi^{(0)}(\tau, \xi) = q^{(1)} - \xi/\tau, \tag{4.84}$$

$$\psi^{(0)}_\xi(\tau, \xi) = -1/\tau. \tag{4.85}$$

The PDE for $\psi^{(1)}$ is therefore

$$-\tau\psi^{(1)}_\tau = \psi^{(1)} + \xi + \xi\psi^{(1)}_\xi. \tag{4.86}$$

The stationary solution to this equation is

$$\psi^{(1)}(\tau, \xi) = -\xi/2, \tag{4.87}$$

which also satisfies the boundary condition. Thus the optimal price in the region $\xi < \beta\tau$ that takes into account the uncertainty σ reduces the price from the deterministic-case solution,

$$\psi(\tau, \xi) \approx \psi^{(0)}(\tau, \xi) - \sigma^2\xi/2. \tag{4.88}$$

This explains the slightly lower prices observed from the comparison between a^D and the numerical approximation to a^B in the plots of Figure 4.6.

4.4.2 Example simulation

The asymptotic results together with Figure 4.5 indicate that one can expect a price decrease over time when following the optimal pricing policy $\alpha^B(t)$. The numerical investigation in the next example verifies this, but highlights that there are negligible gains in total profit from pricing according to $\alpha^B(t)$ rather than the deterministic-case pricing policy $\alpha^D(t)$.

Example 4.3. Let us simulate the system from Example 4.2 with the numerical approximation to $\alpha^B(t)$ and compare it to $\alpha^D(t)$. We draw 100 000 sample paths from the underlying Brownian motion $W(t)$, and set the price to be constant on intervals of size $\Delta t = 0.01$ using a policy $\alpha(t)$. We use the estimator described in §4.2.1 as an approximation to $G(t)$. The simulations are run with both $\alpha^B(t)$ and $\alpha^D(t)$, and statistics of their paths are shown in Figure 4.8. The prices $\alpha^B(t)$ start slightly higher than $\alpha^D(t)$, but will then over time decrease, on average, towards the lower bound $\delta = 1/2$. We also see that the deterministic heuristic is less anticipative, and will begin increasing the prices compared to the optimal policy after $t = 0.2$.

The measure of interest, however, is how much profit the different policies make. Recall that the profit of following a policy $\alpha(t)$ is the random variable

$$P(\alpha) = \int_0^{T_h} q(\alpha(u))\alpha(u)G(u)\,du - CS^\alpha(T). \tag{4.89}$$

From simulations we estimate the distribution of $P(\alpha)$ for the optimal and deterministic policies, and compare their performance. The improvement is negligible, as we see

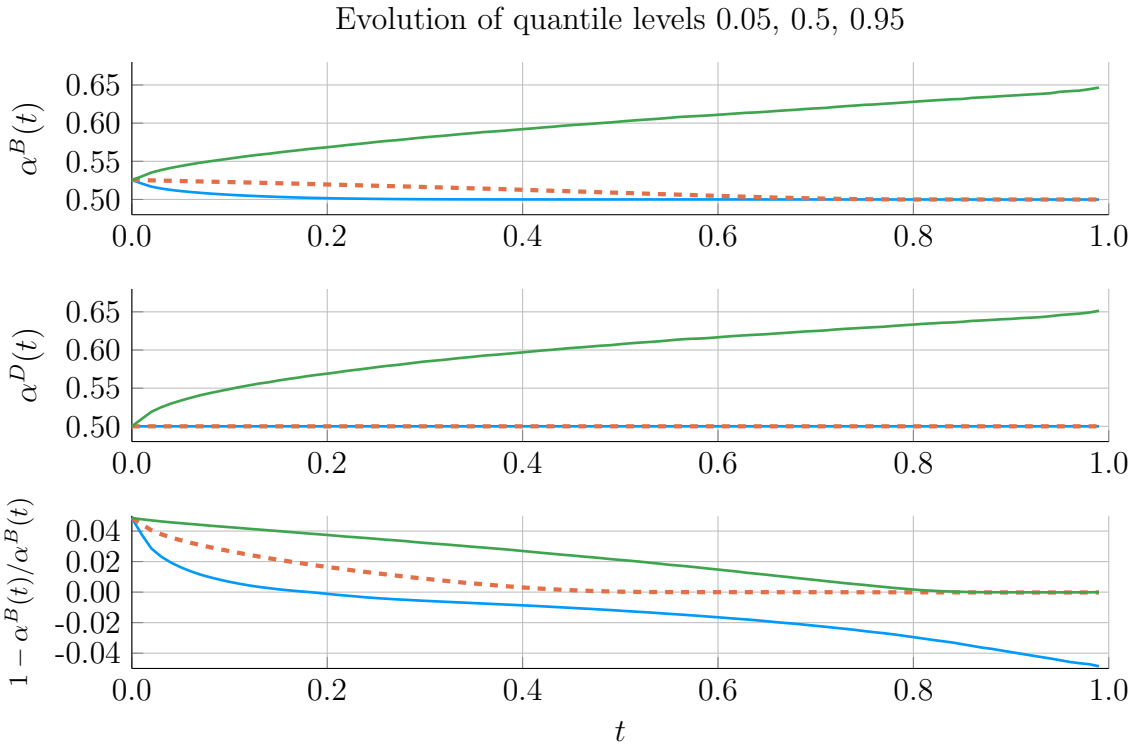


Figure 4.8: Statistics of the price paths from Example 4.3, showing the $0.05, 0.5$, and 0.95 quantiles of $\alpha(t)$. From top to bottom, the optimal policy, the deterministic heuristic, and their relative difference. The optimal policy starts slightly higher, and decreases over time. Note that the 0.05 and 0.5 quantiles lie on top of each other for $\alpha^D(t)$, so prices are likely to stay constant at $1/2$.

from the relative statistics

$$\mathbb{E}\left[1 - P(\alpha^D)/P(\alpha^B)\right] \approx 2 \times 10^{-4}, \tag{4.90}$$

$$\text{std}\left[1 - P(\alpha^D)/P(\alpha^B)\right] \approx 6 \times 10^{-4}, \tag{4.91}$$

$$\text{Median}\left[1 - P(\alpha^D)/P(\alpha^B)\right] \approx -1 \times 10^{-4}. \tag{4.92}$$

The calculated optimal pricing policy results in 0.02 % higher profits than the heuristic on average. It even results in lower profits than the heuristic for more than 50 % of the realisations. Figure 4.9 shows a histogram that approximates the distribution of the relative loss from using $\alpha^D(t)$ over the optimal pricing policy. The differences between the two are small, but the distribution is non-symmetric: The heuristic $\alpha^D(t)$ results in slightly larger profit for more than half of the realisations, at the expense of performing worse for the remaining realisations.

In Example 4.3 and Figure 4.9 it appears that the relative improvement in profits

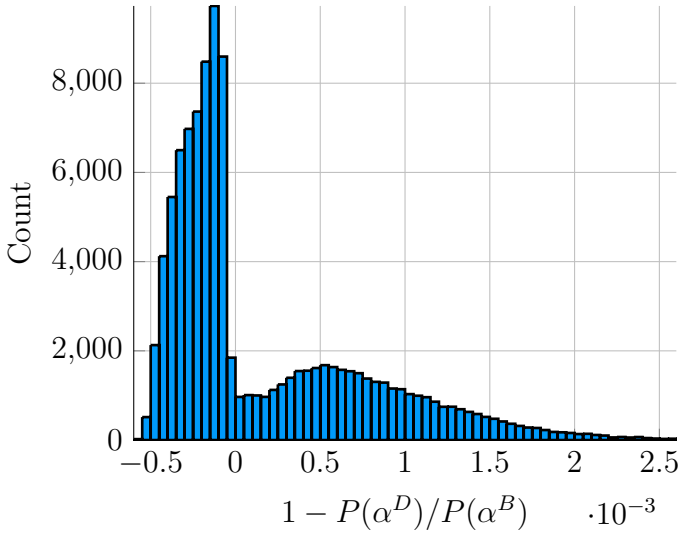


Figure 4.9: The histogram shows the distribution of the relative profit by using the deterministic pricing heuristic $\alpha^D(t)$ to using the optimal policy $\alpha^B(t)$, as described in Example 4.3. Positive values correspond to realisations of $W(t)$ where $\alpha^B(t)$ is better. The shape of the distribution is similar to the discrete-time pricing problem in Chapter 3.

by pricing according to $\alpha^B(t)$, rather than the heuristic policy $\alpha^D(t)$, is negligible. Further numerical experiments for other values of σ strengthen these results. See Table 4.1 for summary statistics of the relative difference in profits $1 - P(\alpha^D)/P(\alpha^B)$. The relative improvement of following the strategy $\alpha^B(t)$ increases with σ, however the standard deviations are all on the order of 0.01 % to 0.1 %.

σ	mean	std	$\mathcal{Q}_{0.05}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.95}$
0.05	0.0	0.4	-0.5	-0.2	0.9
0.1	0.2	0.6	-0.4	-0.1	1.4
0.2	0.3	0.7	-0.3	0.0	1.8
0.4	0.4	1.2	-0.3	0.1	1.8
	$\times 10^{-3}$	$\times 10^{-3}$	$\times 10^{-3}$	$\times 10^{-3}$	$\times 10^{-3}$

Table 4.1: Summary statistics of the relative profit difference $1 - P(\alpha^D)/P(\alpha^B)$ from the model in Example 4.3 with different levels of uncertainty σ. The headings $\mathcal{Q}_z$ denote the z-quantile of the distribution.

4.5 Extensions of the pricing problem

For our one-product system with multiplicative parameter dynamics with uncertainty, two natural extensions to the pricing problem are

1. to incorporate other forms of uncertainty, and
2. to formulate the problem for risk-averse decision makers.

For example, the costs of handling unsold stock may not be known a priori, and risk aversion may be modelled from an expected utility viewpoint. These extensions increase the input dimensions of the corresponding HJB equations, which we present without further discussion.

4.5.1 Other forms of uncertainty

In the prior sections, the source of randomness in the system has come from the multiplicative term $G(t)$, modelled as a geometric Brownian motion martingale. A different non-negative stochastic process may be more appropriate, and the choice of dynamics can be guided by existing sales data. More generally, the demand function $q(a)$ depends on multiple parameters that exhibit different levels of uncertainty and dynamics in time. Another parameter in the pricing problem is the unit cost C, where its value at the terminal time may depend on factors unknown at times $t < T$. Let $\theta(t) \in \mathbb{R}^n$ denote the vector of parameters that are relevant to the problem, and say the demand function and the unit cost depend explicitly on θ. We write $q(a;\theta)$ and $C(\theta)$ for this dependence. Within the diffusion-based stochastic framework, we can model the dynamics of $\theta(t)$ with functions $b(t,\theta) \in \mathbb{R}^n$, $\sigma(t,\theta) \in \mathbb{R}^{n\times p}$, and a vector-valued, uncorrelated, Brownian motion $W(t) \in \mathbb{R}^p$. We assume that $\theta(t)$ does not depend on the pricing policy $\alpha(t)$ or the remaining stock $S^\alpha(t)$. Thus, the system for the pricing problem is described by the SDE

$$d\theta(t) = b(t,\theta(t))\,dt + \sigma(t,\theta(t))\,dW(t), \tag{4.93}$$

$$dS^\alpha(t) = -q(\alpha(t);\theta(t)). \tag{4.94}$$

Let $\Theta \in \mathbb{R}^n$ denote the state space for $\theta(t)$. Let $\nabla_\theta v$ and $H_\theta v$ denote the gradient and Hessian of $v(t,s,\theta)$ with respect to θ. We denote the transpose operator with a superscript $\intercal$, and introduce the volatility matrix $\Sigma(t,\theta) = \sigma(t,\theta)\sigma(t,\theta)^\intercal$. The HJB

approach for the pricing problem is then to find $v : [0,1]^2 \times \Theta \to \mathbb{R}$ which satisfies

$$v_t + b(t,\theta)^\intercal \nabla_\theta v + \tfrac{1}{2}\mathrm{tr}\left(\Sigma(t,\theta) H_\theta v\right) + \max_a \{q(a;\theta)(a - v_s)\} = 0, \tag{4.95}$$

$$v(T,s,\theta) = -C(\theta)s. \tag{4.96}$$

Additional boundary conditions may be necessary, depending on $\theta(t)$. For example, the boundary condition for $g = 0$ in our multiplicative model from the previous sections. The value function and the pricing function now depend explicitly on each element in θ, thus increasing the dimension of the corresponding HJB equation. Efficient algorithms for solving high-dimensional PDEs of this form exist. See, for example, the multigrid preconditioning approach by Reisinger and Arto (2017), or the splitting into a sequence of lower-dimensional PDEs by Reisinger and Wissmann (2018). In higher dimensions it is of even greater importance to critically balance computational cost with the suboptimality of approximations. As §4.4 shows, approximate policies can perform well. This underscores the importance of assessing whether new sources of randomness in the model have significant impact on the objective and the optimal policy.

4.5.2 Expected utility risk aversion

One may argue that for a retailer as a whole, an assumption of a risk-neutral decision maker is valid. For individual product managers, whose performance is evaluated over shorter time horizons, a degree of risk aversion can be preferable from their point of view. Formulations and investigations of the impact of risk aversion on pricing policies is also noted as an interesting area of research by Bitran and Caldentey (2003). For investigations of the expected utility problem with Poisson process demand, see, for example, Lim and Shanthikumar (2007) or Feng and Xiao (2008).

In this section, we assume that the decision maker is evaluated based on the performance of the total profit from selling a product over the time interval $[0,T]$. Then, the pricing decision at time t may also depend on how much revenue has been accrued at that time. So we introduce a state variable $R^\alpha(t)$ representing the accrued revenue at time t, with dynamics $dR^\alpha(t) = \alpha(t)q(\alpha(t);\theta(t))\,dt$. We consider risk aversion based on an expected utility-maximising decision maker. Given a utility function $U(x)$, the pricing problem is then to find a pricing policy α that maximises the expected utility $\mathbb{E}[U(R^\alpha(T) - C(\theta(T))S^\alpha(T))]$. For this utility function, the value

function is defined as

$$v(t,s,\theta,r) = \max_{\alpha\in\mathcal{A}} \mathbb{E}_t \left[U\Big(R^\alpha(T) - C\big(\theta(T)\big) S^\alpha(T) \Big) \right]. \tag{4.97}$$

The corresponding HJB problem is then to find $v : [0,T] \times [0,1] \times \Theta \times [0,\infty) \to \mathbb{R}$ that solves

$$v_t + b(t,\theta)^\intercal \nabla_\theta v + \tfrac{1}{2}\mathrm{tr}\left(\Sigma(t,\theta) H_\theta v\right) + \max_a \{q(a;\theta)(a v_r - v_s)\} = 0, \tag{4.98}$$

$$v(T,s,\theta,r) = U(r - C(\theta)s), \tag{4.99}$$

plus additional boundary conditions. The impact of the risk aversion on pricing decisions appears in the maximisation term in the HJB equation,

$$\max_a \{q(a;\theta)(a v_r - v_s)\}. \tag{4.100}$$

The v_r term represents the relative importance of accruing more revenue to the utility of selling more stock, as represented by the v_s term. The risk-neutral case $U(x) = x$ that we have considered in the previous sections give $v_r = 1$.

4.6 Discussion

This chapter focuses on continuum approximations for dynamic pricing problems under uncertainty. Most of the existing literature on continuous time dynamic pricing for revenue management is based on Poisson processes. This is suitable for many applications, however, for large retailers approximating the number of sales and stock as a continuum can simplify calculation of pricing rules. We present an approach for modelling the sales as a continuous time dynamical system, where the uncertainty in demand arises from stochastic processes. An advantage over the Poisson process model is that this approach allows us to directly model the demand volatility. Under this model, we consider a pricing problem where the retailer aims to deplete inventory of a product at maximum profit. By formulating the problem as a stochastic optimal control problem, we can express the optimal pricing policy in terms of the solution to a nonlinear PDE. For linear and exponential demand functions, we find closed-form expressions for the pricing policy when the system is deterministic. It turns out that, for a risk-neutral decision maker, the deterministic pricing policy is a near-optimal heuristic for systems with demand uncertainty. Numerical errors in calculating the

optimal pricing policy may, in fact, result in lower profits on average than with the heuristic pricing policy.

There are two topics of particular interest for future study. The first is to understand why demand uncertainty has such a small effect on the optimal pricing policy for risk-neutral decision makers, and whether constraints such as requiring monotone-in-time pricing policies may increase this effect. Second, a case study of the continuum model framework for multiple products and time-dependent demand is needed, in order to understand how well this approach can scale to revenue management implementations for retailers.

Chapter 5

Objective acceleration for unconstrained optimisation

We now move from modelling decision making processes to developing an algorithm that a computer can use to approximate the solution to an optimisation problem

$$\min_{x \in \mathbb{R}^n} f(x). \tag{5.1}$$

A local minimum x^* of f is one such that $f(x^*) \leq f(x)$ for all x in a neighbourhood of x^*. When f is sufficiently smooth, the gradient at x^* is zero, $\nabla f(x^*) = 0$. Derivative based algorithms for minimising f are mainly iterative procedures that look for roots of ∇f. Note that $\nabla f(x) = 0$ at local maximisers or saddle points x as well. To find a local minimum, the algorithms also attempt to move in directions that decrease the value of f. Two common examples of such algorithms are the steepest descent and Newton methods (Nocedal and Wright, 2006). The gradient represents the direction where f increases the most, so the negative gradient is the direction of steepest descent. Newton's method minimises a second-order Taylor approximation to the objective. Let the Hessian of f at x be denoted by $H(x)$. Given a current guess x of a local minimiser of f, the two algorithms propose a new guess x^P according to

$$x^P(\lambda) = x - \lambda \nabla f(x) \qquad \text{(steepest descent)}, \tag{5.2}$$

$$x^P(\lambda) = x - \lambda H(x)^{-1} \nabla f(x) \qquad \text{(Newton's method)}, \tag{5.3}$$

where $\lambda > 0$ is determined using a line search algorithm. Line search algorithms decide the step length for each iteration by approximately solving the one-dimensional

optimisation problem

$$\min_{\lambda>0} f(x^P(\lambda)). \tag{5.4}$$

Newton's method has superior convergence properties to the steepest descent algorithm, but the cost of calculating the Hessian and solving the linear system may be prohibitive in some applications. To find a middle ground one can also try to approximate the inverse Hessian, known as quasi-Newton methods, or systematically cover the search space, as nonlinear conjugate gradient methods do. Further information on iterative optimisation algorithms and line search procedures is covered by Nocedal and Wright (2006). In this chapter, we focus on a class of algorithms that are used in combination with other algorithms to, somehow, accelerate the guess x^P towards a local minimiser. We refer to them as acceleration methods. In most of the work on these algorithms, such as the nonlinear generalised minimal residual method (N-GMRES), acceleration is based on minimising the ℓ_2 norm of some target subspaces of $\mathbb{R}^n$. We propose a natural modification to N-GMRES, which significantly improves the performance in a testing environment originally used to advocate N-GMRES. Our proposed approach, which we refer to as O-ACCEL (Objective acceleration), is novel in that it minimises an approximation to the objective function on subspaces of $\mathbb{R}^n$.

The N-GMRES and O-ACCEL algorithms have been implemented in Optim (Mogensen and Riseth, 2018) as part of the thesis work. In this chapter we discuss the algorithm in the context of unconstrained optimisation. The optimisation problems carried out in Chapters 2 to 4 are all constrained so that the decision variable lies in a compact box in $\mathbb{R}^n$. We can still use the algorithm proposed here for these problems as an inner optimiser of the box-constrained optimisation procedure `Fminbox` in Optim. The work which this chapter is based on originated from an investigation of N-GMRES for nonlinear solvers in reservoir simulation (Riseth, 2015).

5.1 Acceleration methods overview

Gradient based optimisation algorithms normally iterate based on tractable approximations to the objective function at a particular point. Acceleration algorithms aim to combine the strengths of existing solvers with information from previous iterates. We propose an acceleration scheme that can be used on top of existing optimisation algorithms, which generates a subspace from previous iterates, over which it aims to optimise the objective function.

Our idea closely resembles the work of De Sterck (2013), which introduced the preconditioned N-GMRES algorithm for optimisation. By using a more appropriate target to accelerate the optimisation than N-GMRES does, we show, with numerical examples, how O-ACCEL more efficiently accelerates the steepest descent algorithm. When optimising an objective f, N-GMRES is used as an accelerator from the point of view of solving the nonlinear system $\nabla f(x) = 0$ which arises from the first-order condition of optimality. It uses the idea of Krylov subspace acceleration from Washio and Oosterlee (1997) and Oosterlee and Washio (2000) for solving nonlinear equations that arise from discretisations of partial differential equations. The name N-GMRES arises from the fact that steepest descent preconditioned N-GMRES is equivalent to the standard GMRES procedure for linear systems of equations (Washio and Oosterlee, 1997, De Sterck, 2013). A similar idea, also arising from nonlinear equations, was described by Anderson (1965). See Walker and Ni (2011) for a note on the similarities of the methods, and Fang and Saad (2009) which puts Anderson acceleration in the context of a Broyden-type approximation of the inverse Jacobian. Brune et al. (2015) show, with many numerical examples, that N-GMRES and Anderson acceleration can greatly improve convergence on nonlinear systems, when combined with an appropriate preconditioner (nonlinear solver), by reducing the number of function and Jacobian evaluations required to reach a given tolerance. In the setting of optimisation, De Sterck (2012) and De Sterck and Howse (2016) show large improvements in convergence by applying N-GMRES acceleration to the computation of tensor decompositions.

More recently, Scieur et al. (2016) have developed another acceleration method for convex optimisation denoted regularised nonlinear acceleration (RNA), which Cartis and Geleta (2017) have extended to the nonconvex case. Acceleration techniques differ from one another in several ways, but, for convex quadratic objectives, the Anderson, N-GMRES and Scieur et al. algorithms all coincide (Cartis and Geleta, 2017). These methods all minimise the ℓ_2 norm of some objective in $\mathbb{R}^n$, the space of the decision variable. The proposed algorithm in this chapter instead aims to minimise the objective function over a subspace of $\mathbb{R}^n$. We believe this is a natural target to accelerate against, especially when the optimisation procedure is seeking descent directions. For convex, quadratic functions we prove that O-ACCEL with a steepest descent preconditioner reduces to the full orthogonalisation method (FOM, Saad (2003)), a Krylov subspace procedure for solving linear systems. This differentiates our method from the other acceleration techniques, which are related to the GMRES algorithm for linear systems.

Due to the close similarity with the proposed algorithm and N-GMRES, this chapter focuses on numerical comparisons to N-GMRES under the same testing conditions as used by De Sterck (2013). On the test set from De Sterck (2013), our acceleration scheme compares favourably to N-GMRES, as well as implementations of the nonlinear conjugate gradient (N-CG) and limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) methods (Nocedal and Wright, 2006). Further tests on the CUTEst test problem set (Gould et al., 2015) show that L-BFGS is more applicable to these problems, however, O-ACCEL again performs better than N-GMRES.

The chapter is organized as follows. Motivation for the algorithm, and discussion around it, is covered in §5.2. Numerical tests that show the efficiency of our proposed acceleration procedure applied to steepest descent are presented in §5.3. We conclude and discuss further potential work in §5.4.

5.2 Optimisation acceleration with O-ACCEL

To fix notation, consider a twice continuously differentiable function $f \in C^2(\mathbb{R}^n)$ that is bounded below and has at least one minimiser. We aim to find a local minimum of the optimisation problem

$$\min_{x \in \mathbb{R}^n} f(x). \tag{5.5}$$

Let $\mathcal{M}(f, x)$ denote an optimisation procedure for f with initial guess $x \in \mathbb{R}^n$. This optimisation procedure can, for example, be the application of one steepest descent, or Newton, step (5.2, 5.3). We refer to $\mathcal{M}$ as the preconditioner, because it is applied in the same fashion as a right preconditioner for iterative procedures of linear systems (Brune et al., 2015). Given a sequence of previously explored iterates $x^{(1)}, \ldots, x^{(k)}$, and a proposed new guess $x^P = \mathcal{M}(f, x^{(k)})$, we try to accelerate the next iterate $x^{(k+1)}$ towards a minimiser. Define

$$\mathcal{K}_k^O(x^P) = \text{span}\{x^{(1)} - x^P, \ldots, x^{(k)} - x^P\}. \tag{5.6}$$

The acceleration step aims to minimise f over the subset $x^P + \mathcal{K}_k^O(x^P)$, which can be interpreted as a generalisation from a line search to a hyperplane search. Let $\alpha \in \mathbb{R}^k$, and set

$$x^A(\alpha) = x^P + \sum_{j=1}^{k} \alpha_j (x^{(j)} - x^P). \tag{5.7}$$

Note that, when $k = 1$, minimising f over $\mathcal{K}_k^O(x^P)$ is equivalent to the standard line search problem of minimising $\lambda \mapsto f(x^{(1)} + \lambda(x^P - x^{(1)}))$. The first-order condition for α to be a minimiser of the function $\alpha \mapsto f(x^A(\alpha))$ is

$$\nabla_\alpha f(x^P + \textstyle\sum_{j=1}^k \alpha_j (x^{(j)} - x^P)) = 0. \tag{5.8}$$

Define the gradient $g(x) = \nabla_x f(x)$. For $l = 1, \ldots, k$,

$$\frac{\partial}{\partial \alpha_l} f(x^P + \textstyle\sum_{j=1}^k \alpha_j (x^{(j)} - x^P)) = g\left(x^P + \textstyle\sum_{j=1}^k \alpha_j (x^{(j)} - x^P)\right)^\intercal (x^{(l)} - x^P), \tag{5.9}$$

where superscript $\intercal$ denotes the transpose. The O-ACCEL algorithm aims to linearise the first-order condition $\nabla_\alpha f(x^A(\alpha)) = 0$ in the following way. Let $H(x)$ denote the Hessian of f at x. By linearising $\alpha \mapsto g(x^A(\alpha))$, we get

$$\begin{aligned} g\left(x^P + \textstyle\sum_{j=1}^k \alpha_j (x^{(j)} - x^P)\right) &\approx g(x^P) + H(x^P) \sum_{j=1}^k \alpha_j (x^{(j)} - x^P) \\ &= g(x^P) + H(x^P)(\boldsymbol{X} - \boldsymbol{X}^P)\alpha, \end{aligned} \tag{5.10}$$

where we use the matrices $\boldsymbol{X} = \left[x^{(1)}, \ldots, x^{(k)}\right] \in \mathbb{R}^{n \times k}$ and $\boldsymbol{X}^P = \left[x^P, \ldots, x^P\right] \in \mathbb{R}^{n \times k}$. Given this linearisation we aim to find an $\alpha \in \mathbb{R}^k$ that approximately satisfies the first-order condition. We can do this by combining (5.9) and (5.10), and then look for an $\alpha \in \mathbb{R}^k$ that solves

$$\alpha^\intercal (\boldsymbol{X} - \boldsymbol{X}^P)^\intercal H(x^P)(x^{(l)} - x^P) = -g(x^P)^\intercal (x^{(l)} - x^P), \qquad l = 1, \ldots, k. \tag{5.11}$$

In matrix form, the system of equations becomes

$$\left(\boldsymbol{X} - \boldsymbol{X}^P\right)^\intercal H(x^P) \left(\boldsymbol{X} - \boldsymbol{X}^P\right) \alpha = -(\boldsymbol{X} - \boldsymbol{X}^P)^\intercal g(x^P). \tag{5.12}$$

There are cases where we may not wish to compute the Hessian of f explicitly, for example, if $\mathcal{M}$ does not use it. We can instead use an approximation $\tilde{H}(x^P)$ of the Hessian $H(x^P)$, or its action on vectors in $\mathcal{K}_k^O(x^P)$. The iterative Hessian approximation algorithms that are used in quasi-Newton methods can provide one avenue of research. In the numerical experiments provided in this chapter, we instead focus on approximating the action of the Hessian on $\mathcal{K}_k^O(x^P)$ to first order by

$$H(x^P)(x^{(l)} - x^P) \approx g(x^{(l)}) - g(x^P). \tag{5.13}$$

Let $g(\boldsymbol{X}) = \big[g(x^{(1)}), \ldots, g(x^{(k)})\big]$, and define $g(\boldsymbol{X}^P)$ similarly. This gives a second approximation to the first-order conditions,

$$(\boldsymbol{X} - \boldsymbol{X}^P)^\intercal \big(g(\boldsymbol{X}) - g(\boldsymbol{X}^P)\big)\, \alpha = -(\boldsymbol{X} - \boldsymbol{X}^P)^\intercal g(x^P). \tag{5.14}$$

In this chapter, we investigate the performance of the objective-based acceleration using (5.14).

To contrast our work with the N-GMRES optimisation algorithm in De Sterck (2013), minimising the ℓ_2 norm of the approximation of $g(x^A)$ established from (5.10) and (5.13) results in the linear least squares problem

$$\min_{\alpha \in \mathbb{R}^k} \Big\| g(x^P) + \sum_{j=l}^{k} \alpha_l (g(x^{(l)}) - g(x^P)) \Big\|_2. \tag{5.15}$$

Its solution can be found from the normal equation

$$\big(g(\boldsymbol{X}) - g(\boldsymbol{X}^P)\big)^\intercal \big(g(\boldsymbol{X}) - g(\boldsymbol{X}^P)\big)\, \alpha = -\big(g(\boldsymbol{X}) - g(\boldsymbol{X}^P)\big)^\intercal g(x^P). \tag{5.16}$$

We argue that the O-ACCEL algorithm is more appropriate for an optimisation problem than N-GMRES. When we are restricted to subsets of the decision space, reduction in the value of the objective is a better indicator of moving towards a minimiser than reduction in the gradient norm. In effect, N-GMRES ignores the extra information provided by f. This is better illustrated in the case when $k = 1$, where it is standard to perform a line search on the objective rather than the gradient norm.

5.2.1 Algorithm

The proposed acceleration procedure is described in Algorithm 1. The number of stored previous iterates w denotes the history size. Setting an upper bound on the history size can be necessary due to storage constraints, or to prevent the local approximations of (5.10) and (5.13) from using iterates far away from x^P. If the direction from x^P to the accelerated step x^A of (5.7) is not a descent direction, it indicates that the linearised approximation around x^P is bad for the currently stored iterates. For simplicity, we therefore choose to reset the history size to $w = 1$ when we encounter such cases.

To prevent re-computation of $g(x^{(j)})$ for $j = 1, \ldots, w$ in each application of the procedure, we store these vectors for later use. The computational cost of the algorithm is approximately the same as w-history L-BFGS with two-loop recursion (De Sterck,

Algorithm 1 The O-ACCEL algorithm

procedure OACCEL($x^{(1)}, \ldots, x^{(w)}$)
  $x^P \leftarrow \mathcal{M}(f, x^{(w)})$
  Approximate $H(x^P) \approx \tilde{H}$, or its action
  $A \leftarrow \left(\boldsymbol{X} - \boldsymbol{X}^P\right)^\intercal \tilde{H} \left(\boldsymbol{X} - \boldsymbol{X}^P\right)$ In §5.3: $A \leftarrow (\boldsymbol{X} - \boldsymbol{X}^P)^\intercal \left(g(\boldsymbol{X}) - g(\boldsymbol{X}^P)\right)$
  $b \leftarrow -(\boldsymbol{X} - \boldsymbol{X}^P)^\intercal g(x^P)$
  Solve $A\alpha = b$
  $x^A \leftarrow x^P + \sum_{j=1}^{w} \alpha_j (x^{(j)} - x^P)$
  if $x^A - x^P$ is a descent direction **then**
    $x^{(w+1)} \leftarrow$ linesearch$(x^P + \lambda(x^A - x^P))$
    `reset` $\leftarrow$ `false`
  else
    $x^{(w+1)} \leftarrow x^P$
    `reset` $\leftarrow$ `true`
  return $(x^{(w+1)}$, `reset`$)$

2013). In terms of storage, O-ACCEL and L-BFGS both store $2w$ vectors of size n. In addition, our implementation of O-ACCEL, as described in Algorithm 2 below, reduces the number of flops required by storing a $w \times w$ matrix of previously calculated values. For the numerical experiments we have used $w = 20$, in accordance with De Sterck (2013). It was, however, shown by De Sterck (2013) that N-GMRES can already provide good results with $w = 3$. Tests using O-ACCEL with $w = 5$, although not included here, provide almost as good results as reported in §5.3. Note that, if the Hessian is sparse, it may be more storage efficient to find α from the linear system in (5.12) than using a large w.

Remark 5.1. The RNA approach suggested by Scieur et al. (2016) is called separately from the iterations by the optimiser, when judged appropriate. This contrasts with Algorithm 1, where the acceleration happens at each iteration. The O-ACCEL acceleration can be applied separately in the same fashion as in Scieur et al. (2016), however, this is not considered in this chapter.

5.2.2 O-ACCEL as a full orthogonalisation method (FOM)

The optimality condition (5.9) for the function $\alpha \mapsto f(x^A(\alpha))$ is $g(x^A)^\intercal(x^{(l)} - x^P) = 0$, for $l = 1, \ldots, k$. Hence, we look for $x^A \in x^P + \mathcal{K}_k^O(x^P)$ so that $g(x^A) \perp \mathcal{K}_k^O(x^P)$. This condition reduces to FOM (Saad, 2003) when $g(x)$ is linear and $\mathcal{M}(f, x)$ is a steepest descent algorithm. When the Hessian is symmetric positive-definite, FOM is mathematically equivalent to the conjugate gradient method. We can therefore think

of O-ACCEL as a N-CG method that approximates the orthogonality condition with a larger history size.

The FOM is an iterative procedure for solving a linear system $Ax = b$. With initial guess $x^{(1)}$ and residual $r^{(1)} = b - Ax^{(1)}$, define the Krylov subspace

$$\mathcal{K}_k(A, r^{(1)}) = \text{span}\{r^{(1)}, Ar^{(1)}, \ldots, A^{k-1}r^{(1)}\}. \tag{5.17}$$

The FOM iterate $x^{(k+1)}$ is an element in $x^{(1)} + \mathcal{K}_k(A, r^{(1)})$ such that the residual is perpendicular to the Krylov subspace,

$$b - Ax^{(k+1)} \perp \mathcal{K}_k(A, r^{(1)}). \tag{5.18}$$

For convex, quadratic objectives $f(x) = \frac{1}{2}x^\intercal Ax - x^\intercal b$, the gradient $g(x) = Ax - b$ is linear and the optimum must satisfy the equation $Ax = b$. The residuals $r^{(k)} = b - Ax^{(k)}$ are equal to the negative gradient $-g(x^{(k)})$. Therefore, O-ACCEL with a steepest descent preconditioner yields $x^P = \mathcal{M}(f, x^{(k)}) = x^{(k)} + \lambda^{(k)}r^{(k)}$ for some $\lambda^{(k)} > 0$.

Theorem 5.1. Let $\mathcal{M}$ be a steepest descent preconditioner and $f(x) = \frac{1}{2}x^\intercal Ax - x^\intercal b$. Let the O-ACCEL algorithm take the step $x^{(w+1)} = x^A$ in Line 9 of Algorithm 1. Then the iterates of the O-ACCEL algorithm form the FOM sequence of the linear system $Ax = b$.

We shall shortly prove the theorem after deriving new expressions for $\mathcal{K}_k(A, r^{(1)})$. First, note that for any x, a reordering of terms can show that

$$\mathcal{K}_k^O(x) = \text{span}\{x - x^{(k)}, x^{(k)} - x^{(k-1)}, \ldots, x^{(2)} - x^{(1)}\}, \tag{5.19}$$

$$x + \mathcal{K}_k^O(x) = x^{(1)} + \text{span}\{x - x^{(k)}, x^{(k)} - x^{(k-1)}, \ldots, x^{(2)} - x^{(1)}\}. \tag{5.20}$$

This motivates the next lemma, which connects the space on the right hand side of (5.19) to $\mathcal{K}_{k+1}(A, r^{(1)})$.

Lemma 5.1. Let $x^{(1)}, \ldots, x^{(k)}$ be a given sequence of FOM iterates for a linear system $Ax = b$. Assume that span$x^{(k)} - x^{(k-1)}, \ldots, x^{(2)} - x^{(1)} = \mathcal{K}_k(A, r^{(1)})$, and let $x^P = x^{(k)} + \lambda r^{(k)}$ for some $\lambda > 0$. Then,

$$\text{span}\{x^P - x^{(k)}, x^{(k)} - x^{(k-1)}, \ldots, x^{(2)} - x^{(1)}\} = \mathcal{K}_{k+1}(A, r^{(1)}). \tag{5.21}$$

Proof. By definition of x^P and the properties of the FOM sequence,

$$x^P - x^{(k)} = \lambda r^{(k)} \perp \mathcal{K}_k(A, r^{(1)}). \tag{5.22}$$

As $x^{(k)} \in x^{(1)} + \mathcal{K}_k(A, r^{(1)})$, we have $r^{(k)} \in \mathcal{K}_{k+1}(A, r^{(1)})$ because

$$r^{(k)} \in b - A(x^{(1)} + \mathcal{K}_k(A, r^{(1)})) = r^{(1)} - A\mathcal{K}_k(A, r^{(1)}) \in \mathcal{K}_{k+1}(A, r^{(1)}). \tag{5.23}$$

Therefore, $\text{span}\{r^{(k)}, \mathcal{K}_k(A, r^{(1)})\} = \mathcal{K}_{k+1}(A, r^{(1)})$. This equality yields the result by replacing $r^{(k)}$ and $\mathcal{K}_k(A, r^{(1)})$ with (5.22) and $\text{span}\{x^{(k)} - x^{(k-1)}, \ldots, x^{(2)} - x^{(1)}\}$. □

Proof of Theorem 5.1. We prove the result by induction on the sequence $x^{(1)}, \ldots, x^{(k)}$ arising from the O-ACCEL algorithm. Let $k = 2$, then

$$x^{(2)} = x^P + \alpha^{(1)}(x^{(1)} - x^P) \tag{5.24}$$

$$= x^{(1)} + \lambda^{(1)} r^{(1)} - \alpha^{(1)}\lambda^{(1)} r^{(1)} \in x^{(1)} + \mathcal{K}_1(A, r^{(1)}). \tag{5.25}$$

and so $\text{span}\{x^{(2)} - x^{(1)}\} = \mathcal{K}_1(A, r^{(1)})$. From (5.9) the residual $b - Ax^{(2)} \perp x^P - x^{(1)} = \lambda^{(k)} r^{(1)}$, and thus $x^{(2)}$ is the second FOM iterate. This establishes the base case for the induction proof.

The inductive step follows from Lemma 5.1 together with (5.19) and (5.20), and hence proves that the O-ACCEL iterates are the FOM iterates for $Ax = b$. □

Remark 5.2. The connection to the FOM differentiates O-ACCEL from N-GMRES, Anderson acceleration, and RNA, which reduce to GMRES for quadratic objectives.

5.3 Numerical experiments

In order to investigate the performance of the proposed algorithm, we implement it with two preconditioners $\mathcal{M}$. The first is steepest descent with line search, and the second is steepest descent with a fixed step length. They are compared to the N-GMRES algorithm with the same preconditioners, and implementations of the N-CG variant with the Polak-Ribière update formula and the two-loop recursion version of the L-BFGS method (Nocedal and Wright, 2006). The test problems considered in §5.3.1 to §5.3.4 are the same eight problems that were used in De Sterck (2013) to advocate N-GMRES. We also include experiments from 33 CUTEst problems to further test the applicability of the algorithms. The results are presented in the form of performance profiles, as introduced by Dolan and Moré (2002), based on the number of function/gradient evaluations.

The main focus of this chapter is to compare the performance of the proposed algorithm to the N-GMRES algorithm. To this end, we have used the MATLAB

implementation of this algorithm, available online.[1] The O-ACCEL implementation, and the rest of the code required to generate the test result data, is also made available by the author.[2] Our implementation of O-ACCEL follows the exact same steps, only replacing the calculations needed to solve the N-GMRES system in (5.16) with those of the linear system in (5.14). The implementation is detailed in Algorithm 2. It closely follows the instructions from Washio and Oosterlee (1997), including a regularisation for the linear system.

The regularisation is used to prevent the direct linear solver for finding α from crashing when A is ill-conditioned or singular, which can happen if the vectors $g(x^{(k)}) - g(x^P)$ are linearly dependent. Let $A \in \mathbb{R}^{w \times w}$ denote the system matrix $(\boldsymbol{X} - \boldsymbol{X}^P)^\intercal \left(g(\boldsymbol{X}) - g(\boldsymbol{X}^P)\right)$. Then, for some tolerance $\epsilon_0 > 0$, set $\epsilon = \epsilon_0 \cdot \max\{A_{ii}\}_{i=1}^{w}$. The max term is used to scale the regularisation in accordance with the optimisation problem. With $I \in \mathbb{R}^{w \times w}$ the identity matrix, we solve the linear problem

$$(A + \epsilon I)\alpha = b, \tag{5.26}$$

rather than the linear problem $A\alpha = b$ as defined in Algorithm 1. This is a Tikhonov type regularisation (Neumaier, 1998), often employed to regularise ill-conditioned problems. Washio and Oosterlee (1997) shows that the error in the resulting α is negligible for the N-GMRES problem (5.16) provided ϵ is much smaller than the smallest non-zero eigenvalue of the system matrix. The error for the O-ACCEL system can be analysed within a general Tikhonov regularisation framework, see, for example, Neumaier (1998). We do not investigate the impact of the regularisation parameter further in this chapter, and use the value $\epsilon_0 = 10^{-14}$ that was used in the N-GMRES code by De Sterck (2013).

For the remainder of the section, we present the test problems, provide details for the parameter choices, and discuss the test results.

5.3.1 Test problems from De Sterck

We describe the seven test problems from De Sterck (2013). All the functions are defined as $f : \mathbb{R}^n \to \mathbb{R}$, and the matrices mentioned are all in $\mathbb{R}^{n \times n}$.

Problem A. Quadratic objective function with symmetric, positive definite diagonal

[1] hansdesterck.net/Publications-by-topic/nonlinear-preconditioning-for-nonlinear-optimization
[2] `https://github.com/anriseth/objective_accel_code`

Algorithm 2 Implementation of O-ACCEL algorithm. Indentation and curly brackets denote scope.

Input: f, g, $\mathcal{M}$, x, $w_{\max}$, ϵ_0, tolerance description
Output: x satisfying tolerance description

while Not reached tolerance **do**
  $\mathtt{x}_1 \leftarrow x$; $\mathtt{r}_1 \leftarrow g(x)$; $\mathtt{q}_{11} \leftarrow x^\intercal \mathtt{r}_1$
  $w \leftarrow 1$; $k \leftarrow 0$; $\mathtt{reset} \leftarrow \mathtt{false}$
  while `reset` is `false` **do**
    $k \leftarrow k + 1$
    $x \leftarrow \mathcal{M}(f, x)$; $r \leftarrow g(x)$
    if reached tolerance **then**
      break
    $\eta \leftarrow x^\intercal r$
    for $i = 1, \ldots, w$ { $\xi_i^{(1)} \leftarrow \mathtt{x}_i^\intercal r$; $\xi_i^{(2)} \leftarrow x^\intercal \mathtt{r}_i$; $\mathtt{b}_i \leftarrow \eta - \xi_i^{(1)}$ }
    for $i = 1, \ldots, w$ { **for** $j = 1, \ldots, w$ { $\mathtt{A}_{ij} \leftarrow \mathtt{q}_{ij} - \xi_i^{(1)} - \xi_j^{(2)} + \eta$ } }
    $\epsilon \leftarrow \epsilon_0 \cdot \max\{\mathtt{A}_{11,} \ldots, \mathtt{A}_{ww}\}$
    Solve $\begin{pmatrix} \mathtt{A}_{11} + \epsilon & \cdots & \mathtt{A}_{1w} \\ \vdots & \ddots & \vdots \\ \mathtt{A}_{w1} & \cdots & \mathtt{A}_{ww} + \epsilon \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_w \end{pmatrix} = \begin{pmatrix} \mathtt{b}_1 \\ \vdots \\ \mathtt{b}_w \end{pmatrix}$
    $x^A \leftarrow x + \sum_{i=1}^{w} \alpha_i (\mathtt{x}_i - x)$
    $d \leftarrow x^A - x$
    if $d^\intercal r \geq 0$ **then**
      $\mathtt{reset} \leftarrow \mathtt{true}$
    else
      $x \leftarrow \text{linesearch}(x + \lambda d)$
      $w \leftarrow \min(w + 1, w_{\max})$
      $j \leftarrow (k \mod w_{\max}) + 1$
      $\mathtt{x}_j \leftarrow x$
      $\mathtt{r}_j \leftarrow g(x)$
      for $i = 1, \ldots, w$ { $\mathtt{q}_{ij} \leftarrow \mathtt{x}_i^\intercal \mathtt{r}_j$; $\mathtt{q}_{ji} \leftarrow \mathtt{x}_j^\intercal \mathtt{r}_i$ }

matrix D,

$$\begin{aligned} f(x) &= \tfrac{1}{2}(x - x^*)^\intercal D (x - x^*), \text{ where} \\ D &= \text{diag}(1, 2, \ldots, n), \text{ and} \\ x^* &= [1, \ldots, 1]\,. \end{aligned} \tag{5.27}$$

The minimiser x^* of Problem A is unique, with $f(x^*) = 0$. The gradient is given by $g(x) = D(x - x^*)$.

Problem B. Problem A with paraboloid coordinate transformation,

$$\begin{aligned} f(x) &= \tfrac{1}{2} y(x-x^*)^\intercal D y(x-x^*), \text{ where} \\ D &= \operatorname{diag}(1,2,\ldots,n), \\ x^* &= [1,\ldots,1]\,, \text{ and} \\ y_1(z) &= z_1 \text{ and } y_j(z) = z_j - 10 z_1^2 \quad (i=2,\ldots,n). \end{aligned} \tag{5.28}$$

The minimiser is again x^*, with $f(x^*) = 0$. The gradient is $g(x) = Dy(x-x^*) - 20(x_1 - x_1^*) \times \left(\sum_{j=2}^n \left(Dy(x-x^*)\right)_j\right) [1,0,\ldots,0]^\intercal$.

Problem C. Problem B with a random nondiagonal matrix T with condition number n,

$$\begin{aligned} f(x) &= \tfrac{1}{2} y(x-x^*)^\intercal T y(x-x^*), \text{ where} \\ x^* &= [1,\ldots,1]\,, \\ y_1(z) &= z_1 \text{ and } y_j(z) = z_j - 10 z_1^2 \quad (i=2,\ldots,n), \text{ and} \\ T &= Q \operatorname{diag}(1,2,\ldots,n)\, Q^\intercal, \end{aligned} \tag{5.29}$$

where Q is a random orthogonal matrix. As in Problems A and B, the minimiser is x^* with $f(x^*) = 0$. The gradient is $g(x) = Ty(x-x^*) - 20(x_1 - x_1^*) \times \left(\sum_{j=2}^n \left(Ty(x-x^*)\right)_j\right) [1,0,\ldots,0]^\intercal$.

Problem D. Extended Rosenbrock function, Problem (21) from Moré et al. (1981),

$$\begin{aligned} f(x) &= \tfrac{1}{2} \sum_{j=1}^n t_j(x)^2, \text{ where } n \text{ is even,} \\ t_j &= 10(x_{j+1} - x_j^2) \qquad (j \text{ odd}), \text{ and} \\ t_j &= 1 - x_{j-1} \qquad\quad (j \text{ even}). \end{aligned} \tag{5.30}$$

The unique minimum $f(x^*) = 0$ is attained at $x^* = [1,\ldots,1]$. The derivative can be computed using $g_k(x) = \sum_{j=1}^n t_j \frac{\partial t_j}{\partial x_k}$, $(k = 1,\ldots,n)$. Gradients for Problems E-G can be computed in similar fashion.

Problem E. Extended Powell singular function, Problem (22) from Moré et al. (1981),

$$\begin{aligned} f(x) &= \tfrac{1}{2}\sum_{j=1}^{n} t_j(x)^2, \text{ where } n \text{ is a multiple of 4,} \\ t_{4j-3} &= x_{4j-3} + 10x_{4j-2}, \\ t_{4j-2} &= \sqrt{5}(x_{4j-1} - x_{4j}), \\ t_{4j-1} &= (x_{4j-2} - 2x_{4j-1})^2, \\ t_{4j} &= \sqrt{10}(x_{4j-3} - x_{4j})^2 \qquad \text{for } j = 1, \ldots, n/4. \end{aligned} \tag{5.31}$$

The unique minimum $f(x^*) = 0$ is attained at $x^* = 0$.

Problem F. The Trigonometric function, Problem (26) from Moré et al. (1981),

$$\begin{aligned} f(x) &= \tfrac{1}{2}\sum_{j=1}^{n} t_j(x)^2, \text{ where} \\ t_j &= n + j(1 - \cos x_j) - \sin x_j - \sum_{i=1}^{n} \cos(x_i). \end{aligned} \tag{5.32}$$

The unique minimum $f(x^*) = 0$ is attained at $x^* = 0$. Note that in De Sterck (2013), a minus sign is used in front of $j(1 - \cos x_j)$. We follow the original formulation of Moré et al. (1981).

Problem G. Penalty function I, Problem (23) from Moré et al. (1981),

$$\begin{aligned} f(x) &= \tfrac{1}{2}\Big(t_0(x)^2 + \sum_{j=1}^{n} t_j(x)^2\Big), \text{ where} \\ t_0 &= -0.25 + \sum_{j=1}^{n} x_j^2, \text{ and} \\ t_j &= \sqrt{10^{-5}}(x_j - 1) \qquad (j = 1, \ldots, n). \end{aligned} \tag{5.33}$$

The minimum is not known explicitly for Problem G, and depends on the value of n.

5.3.2 Experiment design

We test the N-GMRES and O-ACCEL algorithms with two steepest descent preconditioners

$$\mathcal{M}_{\mathrm{Z}}(f, x) = x - \lambda_{\mathrm{Z}} \frac{g(x)}{\|g(x)\|_2}, \qquad \text{with Z = A, B, and}$$

$$\lambda_{\mathrm{A}} = \text{determined by line search,} \tag{5.34}$$

$$\lambda_{\mathrm{B}} = \min(\delta, \|g(x)\|_2). \tag{5.35}$$

Thus, the two preconditioners only differ in the choice of step length. Option A employs a globalizing strategy with a chosen line search, whilst option B takes a predetermined step length. By choosing a short, predetermined step length $\delta > 0$, we expand the subspace to search for α and stay close to the previous iterate $x^{(k)}$, hopefully improving the linearisations in (5.10) and (5.13). For the experiments, we use the line search algorithm by Moré and Thuente (1994), which satisfies the Wolfe conditions (Nocedal and Wright, 2006). It is both employed for $\mathcal{M}_{\mathrm{A}}$, and in the line search $x^P + \lambda(x^A - x^P)$ between the preconditioned step x^P and the accelerated step x^A of the N-GMRES and O-ACCEL routines.

To closely follow the testing conditions of De Sterck (2013), we use the N-CG, L-BFGS and Moré-Thuente line search implementations from the Poblano toolbox by Dunlavy et al. (2010). These may not be state of the art implementations, but the main focus of this chapter is to investigate the performance of the N-GMRES and O-ACCEL algorithms.

All optimisation procedures employ the Moré-Thuente line search with the following options: decrease tolerance $c_1 = 10^{-4}$ and curvature tolerance $c_2 = 0.1$ for the Wolfe conditions, starting step length $\lambda = 1$, and a maximum of 20 f/g evaluations. The N-GMRES and O-ACCEL history lengths are set to $w_{\max} = 20$, and the regularisation parameter is set to $\epsilon_0 = 10^{-12}$. For $\mathcal{M}_{\mathrm{B}}$, the fixed step length is set to $\delta = 10^{-4}$. The L-BFGS history size is set to 5. Larger history sizes were found by De Sterck (2013) to be harmful for the L-BFGS performance on this test set.

Note that our choice of curvature tolerance $c_2 = 0.1$ is different from De Sterck (2013), where $c_2 = 0.01$ was used. There are two reasons for this. First, our choice is often used in practice, see Nocedal and Wright (2006, Ch. 3.1), and it reduces the number of function evaluations for all the solvers considered. Second, we are interested in comparing the outer solvers, however, smaller values of c_2 moves work from the outer solvers to the line search algorithms.

We test Problem A-C for both problem sizes $n = 100$ and $n = 200$. Problem D is tested with $n = 500, 1000, 50\,000, 100\,000$. Problem E with $n = 100, 200, 50\,000, 100\,000$. Problem F is called with $n = 200, 500$, and finally, Problem G with $n = 100, 200$. Each combination of problem and problem size is run 1000 times, with the components of the initial guess drawn uniformly random from the interval $[0, 1]$. For Problem C, each instance of the problem generates a new, random, orthogonal matrix Q. This results in 18 000 individual tests for the comparison. To evaluate performance, we count the number of objective evaluations required for the algorithms to reach an iterate x such that $f(x) - f^* < 10^{-10}(f(x^{(0)}) - f^*)$. A solver run is labelled as failed if it does not reach tolerance within 1500 iterations. The minimum value f^* is known for Problems A-F, however for Problem G we estimate f^* using the lowest value attained across all the optimisation procedures. The results on the collection of 18 000 test instances are discussed in §5.3.3, whilst §5.5 provides tables of results on the individual problems and problem sizes.

Note that our reporting of the numerical experiments differs from that of De Sterck (2013) in two ways: First, we run each problem combination 1000 times, instead of 10 times. Second, we evaluate the results based on performance profiles and tables of quantiles, instead of solely reporting the average number of evaluations to reach tolerance. We believe the high number of test runs is important for more consistent values of the statistics reported in §5.5 across computers, further stabilised by using quantiles rather than averages.

5.3.3 Performance profiles

In order to evaluate the performance of optimisers on test sets with problems of varying size and difficulty, Dolan and Moré (2002) proposed the use of performance profiles. For completeness, we first define the performance profile for our chosen metric of objective evaluations. Let $\mathcal{P}$ denote the test set of the $n_p = 18\,000$ problems, and n_s the number of solvers. For each problem $p \in \mathcal{P}$, and solver s, define

$$t_{p,s} = \text{number of } f \text{ evaluations required to reach tolerance.} \tag{5.36}$$

In the numerical tests we say that the solver has reached tolerance for the problem when the relative decrease in the objective value is at least 10^{-10}, that is

$$t_{p,s} = \min\{k \geq 1 \mid f(x^{(k)}) - f^* < 10^{-10}(f(x^{(0)}) - f^*)\}. \tag{5.37}$$

Remark 5.3. Note that the numbers of objective and gradient calls are the same for each of the optimisers considered in this chapter. This is due to the use of the Moré-Thuente line search algorithm.

Let $\underline{t}_p$ denote the lowest number of f evaluations needed to reach tolerance for problem p across all the solvers,

$$\underline{t}_p = \min\{t_{p,s} \mid 1 \leq s \leq n_s\}. \tag{5.38}$$

The performance ratio measures the performance on problem p by solver s, as defined by

$$\rho_{p,s} = t_{p,s}/\underline{t}_p. \tag{5.39}$$

The value is bounded below by 1, and $\rho_{p,s} = 1$ for at least one solver s. If solver s does not solve problem p, then we set $\rho_{p,s} = \infty$. We define the performance profile $p_s : [1, \infty) \to [0, 1]$, for solver s, by

$$p_s(\tau) = \frac{1}{n_p}\text{size}\left\{p \in \mathcal{P} \mid \rho_{p,s} \leq \tau\right\}. \tag{5.40}$$

The performance profile for a solver s can be viewed as an empirical, cumulative "distribution" function representing the probability of the solver s reaching tolerance within a ratio τ of the fastest solver for each problem. In particular, $p_s(1)$ gives the proportion of problems for which solver s performed best. For large values of τ, the performance profile $p_s(\tau)$ indicates robustness, that is, what proportion of all the test problems were solved by the solver.

Figure 5.1 plots the performance profile of the $n_s = 6$ solvers considered: N-CG, L-BFGS, and N-GMRES and O-ACCEL with steepest descent preconditioning using both a line search (A) and a fixed step size (B). It is clear that O-ACCEL-B and L-BFGS are the best performers across the test set. For 44 % of the test problems they reach tolerance in the fewest f evaluations, and they also solve the largest proportion of problems within higher factors τ of the best performance ratio. There is also a region where N-CG does particularly well, solving the largest proportion of problems within two to three times the highest performing solver. The worst performers are N-GMRES-A and O-ACCEL-A, mainly due to the high amount of work that the line search must do to satisfy the Wolfe conditions along the steepest descent directions.

It is notable that O-ACCEL-B is competitive with L-BFGS on the test set. Tests, not presented in this work, indicate that the L-BFGS performance improves by using a line search with Wolfe curvature condition parameter $c_2 = 0.9$, rather than $c_2 = 0.1$

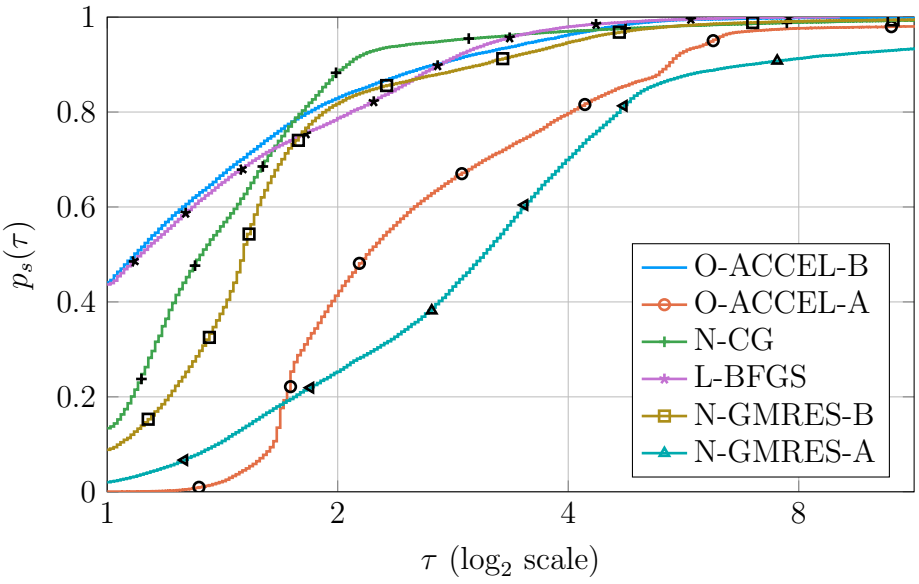


Figure 5.1: Performance profiles, defined in (5.40), for Problems A–G. O-ACCEL preconditioned with a fixed-step steepest descent (B) and L-BFGS mostly outperform the rest, except for higher factors of τ. They are also more robust, solving the largest proportion of the problems when the computational budget is large.

as used in this chapter. The main focus of this chapter is, however, to investigate the potential improvement of minimising the objective rather than an ℓ_2 norm of the gradient. Thus, we are more interested in the comparison between N-GMRES and O-ACCEL. The two plots in Figure 5.2 show the performance profiles comparing N-GMRES and O-ACCEL, and in both cases show a significant improvement by minimising the objective. In fact, O-ACCEL reaches tolerance first on 63 % to 71 % of the test problems. The instances where N-GMRES does better is primarily in Problems E, F, and G, as can be seen from Table 5.4 in §5.5. One of the findings of De Sterck (2013) was that N-GMRES with line search-steepest descent often stagnated or converged very slowly. From the left plot of Figure 5.2, we see that this issue is reduced with the O-ACCEL acceleration. It also turns out that O-ACCEL-A has a larger success rate over the test set than N-GMRES-A.

5.3.4 The tensor optimisation problem from De Sterck

The original motivation for N-GMRES was to improve convergence for a tensor optimisation problem (De Sterck, 2012). De Sterck (2012, 2013) shows that using N-GMRES with a domain-specific ALS preconditioner is better than generic optimisers such as L-BFGS and N-CG. De Sterck (2013) states that

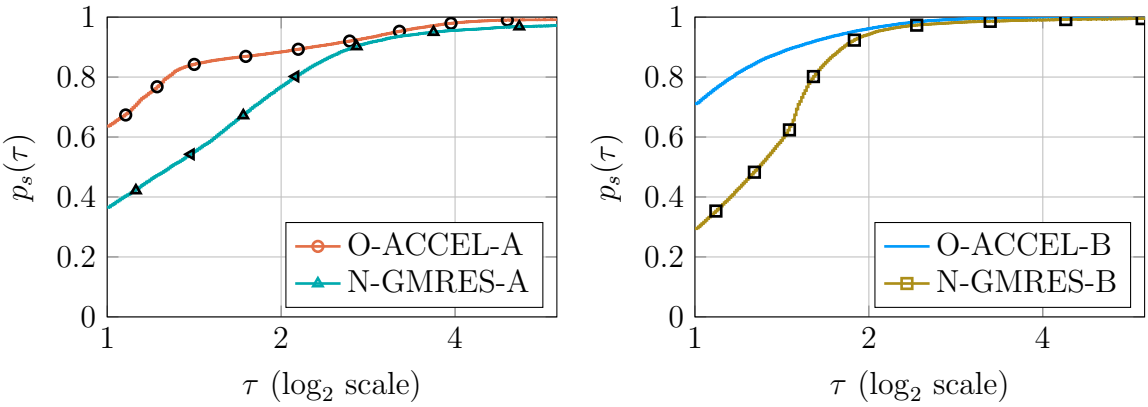


Figure 5.2: Performance profiles comparing N-GMRES and O-ACCEL with steepest descent with line search (A, left) and without (B, right). O-ACCEL outperforms N-GMRES in both cases on our test set. Note that the lines of O-ACCEL-A and N-GMRES-A cross in Figure 5.1, but not in the left figure here, because the performance profiles change depending on the set of solvers considered.

> "In this problem, a rank-three canonical tensor approximation (with 450 variables) is sought for a three-way data tensor of size $50 \times 50 \times 50$. The data tensor is generated starting from a canonical tensor with specified rank and random factor matrices that are modified to have prespecified column colinearity, and noise is added. This is a standard canonical tensor decomposition test problem (Acar et al., 2011)."

For this chapter, we run the 1000 realisations of the test problem using the code provided by De Sterck (2013) with the parameter values described in §5.3.2. The algorithms tested for this problem are vanilla ALS, N-GMRES-ALS, O-ACCEL-ALS, N-CG, and L-BFGS. Figure 5.3 and Table 5.1 show the performance profiles and quantiles for the number of f evaluations required to reach tolerance. We see that O-ACCEL-ALS and N-GMRES-ALS perform better than the other algorithms, which underscores the advantage of applying these acceleration methods to domain-specific algorithms.

5.3.5 CUTEst test problems

The test problems we have considered so far were taken from De Sterck (2013) and originally used to promote N-GMRES. We finish by presenting results from a numerical experiment using problems from the CUTEst problem set (Gould et al., 2015). For this experiment, we compare the solvers O-ACCEL-B, L-BFGS, and N-GMRES-B, with the parameter values described in §5.3.2. The minima are not known for many of the CUTEst problems, and so we change the tolerance criterion to be defined in

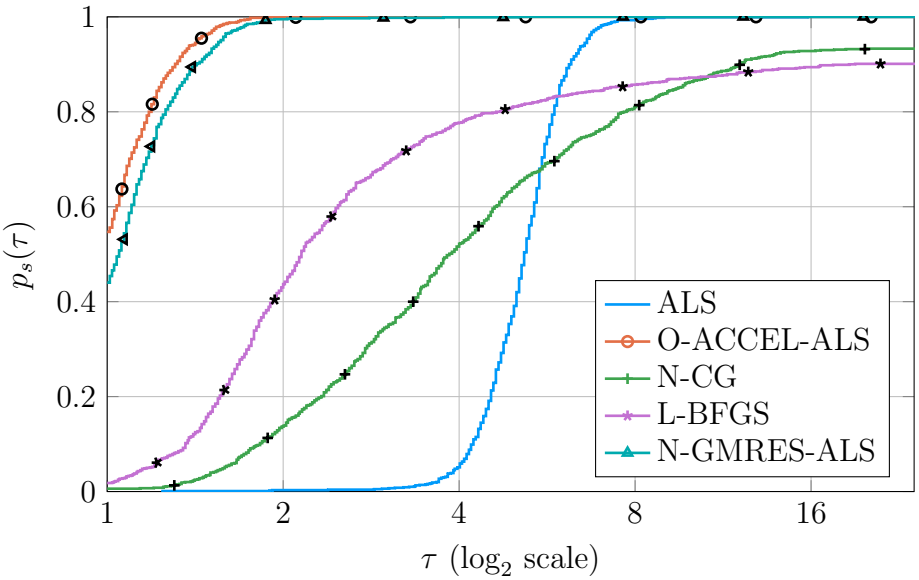


Figure 5.3: Performance profiles from the tensor optimisation test problem. O-ACCEL and N-GMRES perform significantly better than the other solvers.

Algorithm	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
ALS	878	1107	1212
O-ACCEL-ALS	170	227	297
N-CG	405	807	2450
L-BFGS	302	438	3037
N-GMRES-ALS	169	234	312

Table 5.1: Numerical results from the tensor optimisation test problem showing statistics on number of f evaluations. O-ACCEL and N-GMRES perform significantly better than the other solvers.

terms of the relative decrease of the gradient norm.[3] The performance measure used for this experiment is

$$t_{p,s} = \min\{k \geq 1 \mid \|g(x^{(k)})\|_\infty \leq 10^{-8}\|g(x^{(0)})\|_\infty\}. \tag{5.41}$$

A solver run is labelled as failed if it does not reach tolerance within 2000 iterations.

We run the experiment using implementations of the solvers from the package Optim. To be sure, we have also verified that the Optim code yields the same results as the MATLAB code for Problems A–G.

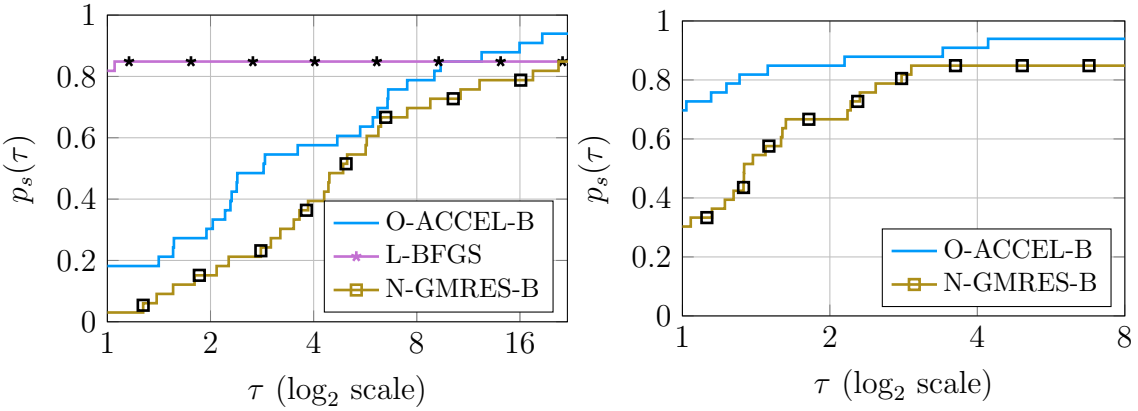


Figure 5.4: Performance profiles for the CUTEst test problems from Tables 5.5 and 5.6. L-BFGS is the highest performing most of the time, however, O-ACCEL-B reaches tolerance for more problems.

The 33 problems we consider are listed in Tables 5.5 and 5.6 of §5.5 together with the results of the numerical experiment. We selected the problems with dimension $n = 50$ to 10 000 that satisfy the two criteria (i) the objective type is in the category "other" (ii) at least one of the solvers succeed in reaching tolerance. Figure 5.4 shows performance profiles from the experiment. L-BFGS reaches tolerance first for most of the problems, however, O-ACCEL reaches tolerance within 2000 iterations for more of the test problems. In the problems where L-BFGS does not reach tolerance it stops because it fails prematurely, whilst N-GMRES-B only fails due to reaching 2000 iterations. We believe the poorer performance of the acceleration algorithms for the CUTEst problems, compared to the previous experiments, is due to the poor performance of the steepest descent preconditioner on these problems. Again, O-ACCEL-B performs better than N-GMRES-B, which underscores our claim that

[3]Using the gradient norm as a tolerance criterion should, if anything, be advantageous to N-GMRES because it minimises gradient norm whilst O-ACCEL minimises the objective.

accelerating based on the objective function is better than accelerating based on the gradient norm.

5.4 Discussion

We have proposed a simple acceleration algorithm for optimisation, based on the N-GMRES algorithm by Washio and Oosterlee (1997), De Sterck (2013). N-GMRES for optimisation aims to accelerate a solver step when solving the nonlinear system $\nabla f(x) = 0$ by minimising the residual in the ℓ_2 norm over a subspace from previous iterates. The acceleration step consists of solving a small linear system that arises from a linearisation of the gradient.

We propose to take advantage of the structure of the optimisation problem and instead accelerate based on the objective value $f(x)$. This new approach, labelled O-ACCEL, shows a significant improvement to the original N-GMRES algorithm in numerical tests when accelerating a steepest descent solver. The first test problems are taken from De Sterck (2013) and run under the same conditions that proved to be beneficial for N-GMRES. Further tests on a selection of CUTEst problems strengthen the conclusion that O-ACCEL outperforms N-GMRES. Another strength of these acceleration algorithms is that they can be combined with many types of optimisers. We have seen O-ACCEL's efficiency with steepest descent, and accelerating quasi-Newton, Newton methods, and domain-specific methods have potential to reduce costs for more expensive algorithms. For example, in De Sterck (2013) it is shown that N-GMRES significantly accelerates the alternating least squares algorithm (ALS), which already without acceleration performs much better than L-BFGS and N-CG on a standard canonical tensor decomposition problem. Our numerical tests show that O-ACCEL further improves the ALS convergence for this problem.

There are two particular paths of interest to improve the proposed acceleration scheme. The first is to reduce the cost by not using a line search between the proposed steps by the solver and O-ACCEL. One can instead rely on heuristics along the lines of those proposed by Washio and Oosterlee (1997). The second is to find better heuristics for choosing previous iterates to use in the acceleration step. Currently, no choices are made, other than discarding all iterates when problems appear. Better guidelines for the number of previous iterates to store is another topic of interest, especially when memory storage is limited.

We would like to investigate connections between the proposed O-ACCEL acceleration step and other optimisation procedures, in the same fashion that Fang and

Saad (2009) put Anderson acceleration in the context of a family of Broyden-type approximations of the inverse Jacobian (Hessian). The preliminary analysis presented in this chapter shows that, for convex quadratic objectives, O-ACCEL with a gradient descent preconditioner is equivalent to FOM for linear systems. As FOM is equivalent to CG for symmetric positive definite systems, we can view O-ACCEL in the context of N-CG methods using a larger history size than usual. There are many new ideas for improving step directions based on previous iterates, such as the acceleration scheme by Scieur et al. (2016), and Block BFGS by Gao and Goldfarb (2016). A better understanding of the overlaps between these and more classical optimisation procedures can provide useful guidance for further research.

Further work is needed to test O-ACCEL on a wider range of problems, with comparisons to other state-of-the-art implementations of solvers and accelerators, in order to provide guidance as to when a method is appropriate. For example, on Problems A–G, O-ACCEL accelerating steepest descent is superior to N-CG and slightly better than L-BFGS. These results may, however, be due to implementations from De Sterck (2013) and test problems favouring the acceleration algorithms. They are still indicative of the power of objective value based optimisation, a research track that is worth pursuing further.

Data access. The MATLAB code used in producing this chapter is available at `https://github.com/anriseth/objective_accel_code`. It includes implementations of the N-GMRES and O-ACCEL algorithms, and code to run Problems A–G as well as the tensor optimisation problem.

5.5 Tables of numerical results

To supplement the performance profiles in the chapter, we include tables that present statistics of the solver performances for the individual test problems.

Tables 5.2 to 5.4 show the results from test problems A–G. Each of the problems was tested with different sizes n, and for each value n the problems were run 1000 times in order to create statistics. The tables report the 0.1, 0.5, and 0.9 quantiles of f evaluations to reach the objective value reduction in (5.37), denoted by $\mathcal{Q}_{0.1}$, $\mathcal{Q}_{0.5}$, and $\mathcal{Q}_{0.9}$ respectively. Table 5.2 provides results for Problems A–C, Table 5.3 for Problem D, and Table 5.4 for the remaining Problems E–G.

Tables 5.5 and 5.6 show the results from the CUTEst problems, where "Fail" means failure to reach the gradient value reduction in (5.41) within 2000 iterations. The norm used for the gradient values in the tables is the infinity norm.

Algorithm	A, $n = 100$			A, $n = 200$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	75	79	81	103	107	111
O-ACCEL-A	131	136	140	171	179	184
N-CG	87	93	99	113	131	145
L-BFGS	75	79	81	103	107	111
N-GMRES-B	111	117	122	158	169	192
N-GMRES-A	166	246	336	307	414	510

Algorithm	B, $n = 100$			B, $n = 200$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	183	267	416	262	365	595
O-ACCEL-A	258	389	546	377	478	800
N-CG	134	211	560	221	359	1598
L-BFGS	76	100	169	99	127	292
N-GMRES-B	215	315	542	317	433	840
N-GMRES-A	272	648	1516	452	809	2204

Algorithm	C, $n = 100$			C, $n = 200$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	113	136	178	151	176	215
O-ACCEL-A	188	208	259	264	292	324
N-CG	165	187	215	259	298	344
L-BFGS	104	114	125	148	160	177
N-GMRES-B	142	164	208	219	254	304
N-GMRES-A	264	333	459	508	620	854

Table 5.2: Quantiles reporting f evaluations to reach tolerance for each solver on test problems A–C from §5.3.1. Grey rows highlight the solver with the best 0.5 quantile. L-BFGS performs best for these easier problems. In Problem A, the L-BFGS and O-ACCEL performance measures are so similar that the quantiles are the same.

Algorithm	D, $n = 500$			D, $n = 1000$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	93	105	123	91	98	116
O-ACCEL-A	193	233	277	192	233	280
N-CG	158	188	196	162	190	197
L-BFGS	128	155	194	129	153	189
N-GMRES-B	141	163	193	142	167	193
N-GMRES-A	284	349	508	290	349	471
Algorithm	D, $n = 50000$			D, $n = 100000$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	101	117	132	122	126	135
O-ACCEL-A	195	226	276	196	225	271
N-CG	159	187	196	159	188	196
L-BFGS	131	156	190	130	156	191
N-GMRES-B	154	178	215	163	190	231
N-GMRES-A	312	378	525	320	394	562

Table 5.3: Quantiles reporting f evaluations to reach tolerance for each solver on test problem D from §5.3.1. Grey rows highlight the solver with the best 0.5 quantile. O-ACCEL-B handles Problem D best.

Algorithm	E, $n = 100$			E, $n = 200$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	190	222	265	198	228	274
O-ACCEL-A	301	349	625	312	371	781
N-CG	205	238	283	213	245	290
L-BFGS	463	627	965	480	639	1036
N-GMRES-B	232	267	330	235	268	338
N-GMRES-A	280	332	395	284	335	401
Algorithm	E, $n = 50000$			E, $n = 100000$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	368	487	689	400	536	798
O-ACCEL-A	745	1157	1356	764	1207	1417
N-CG	297	360	461	321	384	491
L-BFGS	599	703	852	626	725	879
N-GMRES-B	275	335	738	258	318	848
N-GMRES-A	310	391	562	318	402	609

Algorithm	F, $n = 200$			F, $n = 500$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	53	71	118	44	55	97
O-ACCEL-A	81	93	110	84	102	121
N-CG	34	46	60	33	47	69
L-BFGS	41	48	56	34	44	51
N-GMRES-B	48	59	110	43	51	89
N-GMRES-A	76	87	99	78	92	107

Algorithm	G, $n = 100$			G, $n = 200$		
	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$	$\mathcal{Q}_{0.1}$	$\mathcal{Q}_{0.5}$	$\mathcal{Q}_{0.9}$
O-ACCEL-B	148	212	296	196	224	258
O-ACCEL-A	302	940	1078	220	815	957
N-CG	76	191	201	53	165	174
L-BFGS	66	173	180	53	150	156
N-GMRES-B	161	216	266	167	210	245
N-GMRES-A	528	764	4518	203	720	4526

Table 5.4: Quantiles reporting f evaluations to reach tolerance for each solver on test problems E–G from §5.3.1. Grey rows highlight the solver with the best 0.5 quantile. N-GMRES performs best at the median range for two problems, however it is less robust as can be seen from the upper quantile.

Problem		Solver	Iter	f-calls	$f_{\min}$	$\|g_{\min}\|$	Fail
	$n = 5000$	O-ACCEL-B	9	20	0.0	2.3×10^{-6}	
ARWHEAD	$f_0 = 1.5\times10^{4}$	L-BFGS	6	21	0.0	4.8×10^{-7}	
	$\|g_0\| = 4.0\times10^{4}$	N-GMRES-B	6	20	2.6×10^{-9}	2.1×10^{-5}	
	$n = 10\,000$	O-ACCEL-B	19	86	-1.9×10^{3}	2.0×10^{-10}	
BOX	$f_0 = 0.0$	L-BFGS	7	14	-1.9×10^{3}	4.0×10^{-9}	
	$\|g_0\| = 5.0\times10^{-1}$	N-GMRES-B	30	105	-1.9×10^{3}	2.2×10^{-9}	
	$n = 10\,000$	O-ACCEL-B	52	272	-1.0×10^{4}	7.1×10^{-9}	
COSINE	$f_0 = 8.8\times10^{3}$	L-BFGS	12	22	-1.0×10^{4}	3.1×10^{-9}	
	$\|g_0\| = 9.6\times10^{-1}$	N-GMRES-B	29	80	-1.0×10^{4}	2.0×10^{-9}	
	$n = 5000$	O-ACCEL-B	157	478	1.7×10^{3}	4.7×10^{-5}	
CRAGGLVY	$f_0 = 2.7\times10^{6}$	L-BFGS	57	133	1.7×10^{3}	4.0×10^{-5}	
	$\|g_0\| = 5.6\times10^{3}$	N-GMRES-B	229	760	1.7×10^{3}	3.1×10^{-5}	
	$n = 3000$	O-ACCEL-B	48	223	1.0	6.1×10^{-10}	
DIXMAANA	$f_0 = 2.9\times10^{4}$	L-BFGS	6	12	1.0	9.9×10^{-16}	
	$\|g_0\| = 2.8\times10^{1}$	N-GMRES-B	13	53	1.0	1.5×10^{-10}	
	$n = 3000$	O-ACCEL-B	31	99	1.0	3.2×10^{-8}	
DIXMAANB	$f_0 = 4.7\times10^{4}$	L-BFGS	6	11	1.0	2.3×10^{-7}	
	$\|g_0\| = 4.0\times10^{1}$	N-GMRES-B	35	228	1.0	3.8×10^{-10}	
	$n = 3000$	O-ACCEL-B	30	105	1.0	4.1×10^{-8}	
DIXMAANC	$f_0 = 8.2\times10^{4}$	L-BFGS	7	14	1.0	2.3×10^{-8}	
	$\|g_0\| = 7.6\times10^{1}$	N-GMRES-B	13	49	1.0	4.4×10^{-8}	
	$n = 3000$	O-ACCEL-B	23	75	1.0	1.5×10^{-6}	
DIXMAAND	$f_0 = 1.6\times10^{5}$	L-BFGS	8	16	1.0	2.6×10^{-7}	
	$\|g_0\| = 1.5\times10^{2}$	N-GMRES-B	16	78	1.0	4.9×10^{-7}	
	$n = 3000$	O-ACCEL-B	394	1109	1.0	2.6×10^{-7}	
DIXMAANE	$f_0 = 2.2\times10^{4}$	L-BFGS	241	485	1.0	2.7×10^{-7}	
	$\|g_0\| = 2.7\times10^{1}$	N-GMRES-B	885	2751	1.0	2.6×10^{-7}	
	$n = 3000$	O-ACCEL-B	282	765	1.0	3.5×10^{-7}	
DIXMAANF	$f_0 = 4.1\times10^{4}$	L-BFGS	195	393	1.0	3.7×10^{-7}	
	$\|g_0\| = 3.9\times10^{1}$	N-GMRES-B	567	1687	1.0	3.9×10^{-7}	
	$n = 3000$	O-ACCEL-B	277	770	1.0	6.9×10^{-7}	
DIXMAANG	$f_0 = 7.6\times10^{4}$	L-BFGS	165	334	1.0	6.9×10^{-7}	
	$\|g_0\| = 7.5\times10^{1}$	N-GMRES-B	570	1673	1.0	7.2×10^{-7}	
	$n = 3000$	O-ACCEL-B	236	655	1.0	1.5×10^{-6}	
DIXMAANH	$f_0 = 1.5\times10^{5}$	L-BFGS	146	297	1.0	1.5×10^{-6}	
	$\|g_0\| = 1.5\times10^{2}$	N-GMRES-B	620	1835	1.0	1.4×10^{-6}	
	$n = 3000$	O-ACCEL-B	862	1877	1.0	7.1×10^{-7}	
DIXMAANK	$f_0 = 7.4\times10^{4}$	L-BFGS	392	787	1.0	7.1×10^{-7}	
	$\|g_0\| = 7.4\times10^{1}$	N-GMRES-B	556	1640	1.0	7.2×10^{-7}	
	$n = 3000$	O-ACCEL-B	267	685	1.0	1.4×10^{-6}	
DIXMAANL	$f_0 = 1.5\times10^{5}$	L-BFGS	240	485	1.0	1.5×10^{-6}	
	$\|g_0\| = 1.5\times10^{2}$	N-GMRES-B	378	1096	1.0	1.4×10^{-6}	
	$n = 3000$	O-ACCEL-B	1426	3149	1.0	1.3×10^{-6}	
DIXMAANP	$f_0 = 7.1\times10^{4}$	L-BFGS	549	1102	1.0	1.3×10^{-6}	
	$\|g_0\| = 1.3\times10^{2}$	N-GMRES-B	1037	3091	1.0	1.1×10^{-6}	
	$n = 2000$	O-ACCEL-B	75	246	1.2×10^{4}	1.8×10^{-5}	
EDENSCH	$f_0 = 7.4\times10^{6}$	L-BFGS	21	45	1.2×10^{4}	2.0×10^{-5}	
	$\|g_0\| = 2.2\times10^{3}$	N-GMRES-B	54	200	1.2×10^{4}	7.1×10^{-6}	

Table 5.5: Results from the CUTEst problems.

Problem		Solver	Iter	f-calls	$f_{\min}$	$\|g_{\min}\|$	Fail
	$n = 1000$	O-ACCEL-B	6	14	-10.0×10^{2}	1.7×10^{-6}	
EG2	$f_0 = -8.4\times10^{2}$	L-BFGS	3	9	-10.0×10^{2}	4.1×10^{-7}	
	$\|g_0\| = 5.4\times10^{2}$	N-GMRES-B	6	14	-10.0×10^{2}	3.6×10^{-6}	
	$n = 5000$	O-ACCEL-B	47	250	5.6×10^{3}	8.5×10^{-7}	
ENGVAL1	$f_0 = 2.9\times10^{5}$	L-BFGS	33	366	5.6×10^{3}	2.8×10^{-6}	×
	$\|g_0\| = 1.2\times10^{2}$	N-GMRES-B	65	318	5.6×10^{3}	7.6×10^{-7}	
	$n = 5000$	O-ACCEL-B	143	989	-2.5×10^{5}	4.8×10^{-9}	
FLETBV3M	$f_0 = 2.0\times10^{2}$	L-BFGS	24	62	-2.5×10^{5}	5.1×10^{-10}	
	$\|g_0\| = 7.1\times10^{-1}$	N-GMRES-B	92	756	-2.5×10^{5}	5.0×10^{-9}	
	$n = 5625$	O-ACCEL-B	1510	4384	1.0	9.6×10^{-9}	×
FMINSRF2	$f_0 = 2.8\times10^{1}$	L-BFGS	1537	3367	1.0	2.2×10^{-10}	
	$\|g_0\| = 2.4\times10^{-2}$	N-GMRES-B	2000	4661	1.5	9.1×10^{-3}	×
	$n = 5625$	O-ACCEL-B	1741	5007	1.0	9.7×10^{-9}	×
FMINSURF	$f_0 = 2.9\times10^{1}$	L-BFGS	781	1672	1.0	2.2×10^{-10}	
	$\|g_0\| = 2.3\times10^{-2}$	N-GMRES-B	2000	4440	1.6	9.2×10^{-3}	×
	$n = 5010$	O-ACCEL-B	526	1610	-1.1×10^{3}	3.1×10^{-8}	
NCB20	$f_0 = 1.0\times10^{4}$	L-BFGS	467	1089	-1.2×10^{3}	1.3×10^{-6}	×
	$\|g_0\| = 4.0$	N-GMRES-B	2000	4093	-1.1×10^{3}	6.8×10^{-1}	×
	$n = 5000$	O-ACCEL-B	1178	4121	7.4×10^{3}	3.3×10^{-8}	
NCB20B	$f_0 = 1.0\times10^{4}$	L-BFGS	1467	4173	7.4×10^{3}	3.0×10^{-5}	×
	$\|g_0\| = 4.0$	N-GMRES-B	2000	3013	7.4×10^{3}	2.7×10^{-5}	×
	$n = 5000$	O-ACCEL-B	363	1044	1.9×10^{-4}	1.7×10^{-4}	
NONDQUAR	$f_0 = 5.0\times10^{3}$	L-BFGS	208	436	9.3×10^{-5}	1.8×10^{-4}	
	$\|g_0\| = 2.0\times10^{4}$	N-GMRES-B	480	1394	3.8×10^{-4}	1.7×10^{-4}	
	$n = 200$	O-ACCEL-B	283	1798	1.0×10^{-3}	1.5×10^{-3}	
PENALTY3	$f_0 = 1.6\times10^{9}$	L-BFGS	3	15	1.6×10^{9}	1.6×10^{5}	×
	$\|g_0\| = 1.6\times10^{5}$	N-GMRES-B	2000	2165	2.4×10^{172}	2.4×10^{172}	×
	$n = 5000$	O-ACCEL-B	115	460	3.3×10^{-6}	1.2×10^{-6}	
POWELLSG	$f_0 = 2.7\times10^{5}$	L-BFGS	20	49	4.2×10^{-10}	5.5×10^{-7}	
	$\|g_0\| = 3.1\times10^{2}$	N-GMRES-B	105	308	3.2×10^{-9}	1.4×10^{-8}	
	$n = 10\,000$	O-ACCEL-B	186	637	4.0×10^{4}	1.5×10^{4}	
POWER	$f_0 = 2.5\times10^{15}$	L-BFGS	38	107	4.9×10^{4}	1.5×10^{4}	
	$\|g_0\| = 2.0\times10^{12}$	N-GMRES-B	607	1868	7.5×10^{4}	1.4×10^{4}	
	$n = 5000$	O-ACCEL-B	74	208	-1.5×10^{4}	1.0×10^{-8}	
SCHMVETT	$f_0 = -1.4\times10^{4}$	L-BFGS	55	133	-1.5×10^{4}	9.2×10^{-9}	
	$\|g_0\| = 1.1$	N-GMRES-B	78	239	-1.5×10^{4}	9.3×10^{-9}	
	$n = 5000$	O-ACCEL-B	78	297	-6.8×10^{6}	4.9×10^{-5}	
SINQUAD	$f_0 = 6.6\times10^{-1}$	L-BFGS	12	45	-6.8×10^{6}	1.4×10^{-5}	
	$\|g_0\| = 5.0\times10^{3}$	N-GMRES-B	95	483	-6.8×10^{6}	7.1×10^{-6}	
	$n = 10\,000$	O-ACCEL-B	134	598	1.7×10^{-7}	4.4×10^{-6}	
SPARSQUR	$f_0 = 1.4\times10^{7}$	L-BFGS	26	91	1.4×10^{-6}	2.3×10^{-4}	
	$\|g_0\| = 3.2\times10^{4}$	N-GMRES-B	214	797	1.1×10^{-6}	1.8×10^{-5}	
	$n = 50$	O-ACCEL-B	201	538	1.4×10^{3}	1.3×10^{-6}	
TOINTGOR	$f_0 = 5.1\times10^{3}$	L-BFGS	126	265	1.4×10^{3}	9.8×10^{-7}	
	$\|g_0\| = 1.5\times10^{2}$	N-GMRES-B	287	795	1.4×10^{3}	1.3×10^{-6}	
	$n = 50$	O-ACCEL-B	277	963	2.3×10^{2}	6.3×10^{-8}	
TOINTPSP	$f_0 = 1.8\times10^{3}$	L-BFGS	127	334	2.3×10^{2}	2.9×10^{-7}	
	$\|g_0\| = 3.0\times10^{1}$	N-GMRES-B	423	1287	2.3×10^{2}	2.5×10^{-7}	
	$n = 200$	O-ACCEL-B	8	28	6.9×10^{-2}	1.2×10^{2}	
VARDIM	$f_0 = 3.3\times10^{16}$	L-BFGS	1	20	3.3×10^{16}	1.9×10^{15}	×
	$\|g_0\| = 1.9\times10^{15}$	N-GMRES-B	4	39	5.3×10^{3}	5.0×10^{5}	

Table 5.6: Results from the CUTEst tests.

Chapter 6

Summary and further research

In this thesis, we discuss aspects of the decision process, in particular, the mathematical formulation of optimisation problems and algorithms used by numerical solvers to approximate solutions to such problems. When mathematicians and practitioners use models with randomness, they often marginalise over random variables and then mostly ignore the full distribution of the outcomes. The argument is often that the marginalised value is a good representation of what the decision maker cares about, or should care about. It is our intention to highlight that the distribution of the outcomes for relevant decisions can both guide the choice of optimisation formulation and provide information regarding the impact that algorithm decisions have on modelled risk-attitudes. The examples we provide to motivate the work are mainly simple problems related to pricing of retail products, however, the considerations apply equally to other decision processes with random outcomes.

Chapter 2 illustrates that when the distributions are uni-modal, or approximately normal, then formulating the optimisation problem using mean and standard deviation can yield the same decision as using super-quantiles. This is advantageous from both an implementational and a computational point of view. Chapters 3 and 4 provide further examples of how simplified formulations, that ignore the randomness, result in marginalised outcomes that are practically equal to the values from decisions based on the original formulation. A further investigation of the distribution of outcomes shows that the simplified formulations result in the best outcomes for half of the realisations of the underlying random variables. In the remaining realisations, they result in much worse outcomes, which we interpret as an algorithmic decision that has impacted the modelled risk-attitude. The optimisation problems that we formulate in this thesis need to be solved numerically using some optimisation routine using gradient information. In Chapter 5 we propose a new algorithm that can be combined

with existing algorithms and we show, with several numerical experiments, that it reduces the computational cost of finding a solution with a prescribed accuracy.

The examples that we use to illustrate the impact of decisions and distributions of outcomes are simplified problems of retail applications. They provide a template for types of investigations that can help the development of decision making infrastructure in real-world applications. In our examples, these investigations conclude that simplified formulations reduce computational cost with little impact on the objective, however, more complicated systems may behave differently. A natural avenue for further research is to consider a real-world system where the demand models and the distributional information are obtained from a retailer. Within this system one could then:

1. Investigate the typical distributions of objectives that arise based on this data;
2. propose two to four optimisation formulations; and
3. compare the computational cost, decisions, and distribution of outcomes that arise from the different formulations.

The presence of competitors or external decisions that we do not directly control is not discussed in this thesis. Competition is believed to have a significant impact on the objectives by decision makers in business, and work to better understand such game theoretic matters is important. A difficulty with theoretical investigations of such situations is that the researchers have too much control over the problem — they decide a priori both how the underlying system behaves and what policies the competitors can choose from. We have contributed to an experiment that runs a pricing and modelling competition between several competitors (van de Geer et al., 2018). Each competitor writes an algorithm that, only based on data, dynamically prices a product over 1000 periods to maximise revenue without knowing a priori what algorithms the competitors use. We believe this provides a good way to test how the algorithms for decision-making behave in unexpected environments, and recommend that similar competitions are run within the retailer before a new decision-making system goes "live".

Bibliography

E. Acar, D. M. Dunlavy, and T. G. Kolda. A scalable optimization approach for fitting canonical tensor decompositions. *Journal of Chemometrics*, 25(2):67–86, 2011.

S. Ahmed, U. Çakmak, and A. Shapiro. Coherent risk measures in inventory problems. *European Journal of Operational Research*, 182(1):226–238, 2007.

A. Alexanderian, N. Petra, G. Stadler, and O. Ghattas. Mean-variance risk-averse optimal control of systems governed by PDEs with random parameter fields using quadratic approximations. *SIAM/ASA Journal on Uncertainty Quantification*, 5 (1):1166–1192, 2017.

D. G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, 1965.

P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath. Coherent measures of risk. *Mathematical Finance*, 9(3):203–228, 1999.

Y. Aviv and G. Vulcano. Dynamic list pricing. In Ö. Özer and R. Phillips, editors, *The Oxford handbook of pricing management*, chapter 23. Oxford University Press, 2012.

R. Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

A. Ben-Tal and M. Teboulle. An old-new concept of convex risk measures: The Optimized Certainty Equivalent. *Mathematical Finance*, 17(3):449–476, 2007.

A. Ben-Tal, L. El Ghaoui, and A. Nemirovski. *Robust optimization*. Princeton University Press, 2009.

F. E. Benth, S. Koekebakker, and C. M. I. C. Taib. Stochastic dynamical modelling of spot freight rates. *IMA Journal of Management Mathematics*, 26(3):273–297, 2014.

D. Bernoulli. Specimen theoriae novae de mensura sortis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, pages 175–192, 1738. English translation by L. Sommer: Exposition of a new theory on the measurement of risk. *Econometrica* 22:23–36, 1954.

D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, third edition, 2005.

D. P. Bertsekas. *Dynamic programming and optimal control: Approximate Dynamic Programming*, volume 2. Athena Scientific Belmont, MA, fourth edition, 2012.

D. J. Bertsimas and G. Perakis. Dynamic pricing: A learning approach. Technical report, Massachusetts Institute of Technology, Operations Research Center, 2001. `http://hdl.handle.net/1721.1/5314`.

J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A fresh approach to numerical computing. *SIAM Review*, 59(1):65–98, 2017.

G. Bitran and R. Caldentey. An overview of pricing models for revenue management. *Manufacturing & Service Operations Management*, 5(3):203–229, 2003.

G. Bitran, R. Caldentey, and S. Mondschein. Coordinating clearance markdown sales of seasonal products in retail chains. *Operations Research*, 46(5):609–624, 1998.

F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

P. R. Brune, M. G. Knepley, B. F. Smith, and X. Tu. Composing scalable nonlinear algebraic solvers. *SIAM Review*, 57(4):535–565, 2015.

R. E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7:1–49, 1998.

L. Campi and M. P. Owen. Multivariate utility maximization with proportional transaction costs. *Finance and Stochastics*, 15(3):461–499, 2011.

L. Caravenna, J. Dewynne, C. Farmer, O. Jones, C. O. Lund, S. Melnik, B. Pawlowska, and E. Wilson. Customer focused price optimisation. 100th European Study Group with Industry, Oxford, 2014. URL `http://www.maths-in-industry.org/miis/673`.

R. Carmona and M. Coulon. A survey of commodity markets and structural models for electricity prices. In *Quantitative Energy Finance*, pages 41–83. Springer, 2014.

Á. Cartea, S. Jaimungal, and J. Penalva. *Algorithmic and high-frequency trading*. Cambridge University Press, 2015.

C. Cartis and M. Geleta. Accelerating nonlinear optimization algorithms. Numerical analysis technical report, University of Oxford, 2017.

M. J. Chambers. Estimation of a continuous-time dynamic demand system. *Journal of Applied Econometrics*, 7(1):53–64, 1992.

C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall/CRC, 6 edition, 2004.

S. Choi, A. Ruszczynski, and Y. Zhao. A multiproduct risk-averse newsvendor with law-invariant coherent measures of risk. *Operations Research*, 59(2):346–364, 2011.

G. Cramer, 1728. Letter to Nicolaus I Bernoulli, quoted in D. Bernoulli (1738).

I. Das and J. E. Dennis. Normal-Boundary Intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

H. De Sterck. A nonlinear GMRES optimization algorithm for canonical tensor decomposition. *SIAM Journal on Scientific Computing*, 34(3):A1351–A1379, 2012.

H. De Sterck. Steepest descent preconditioning for nonlinear GMRES optimization. *Numerical Linear Algebra with Applications*, 20(3):453–471, 2013.

H. De Sterck and A. Howse. Nonlinearly preconditioned optimization on grassmann manifolds for computing approximate tucker tensor decompositions. *SIAM Journal on Scientific Computing*, 38(2):A997–A1018, 2016.

E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002.

D. M. Dunlavy, T. G. Kolda, and E. Acar. Poblano v1.0: A MATLAB toolbox for gradient-based optimization. Technical Report SAND2010-1422, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Mar. 2010.

I. Dunning, J. Huchette, and M. Lubin. JuMP: A modeling language for mathematical optimization. *SIAM Review*, 59(2):295–320, 2017.

F. Y. Edgeworth. *Mathematical psychics: An essay on the application of mathematics to the moral sciences*. Kegan Paul & Co, London, 1881.

G. Eichfelder. An adaptive scalarization method in multiobjective optimization. *SIAM Journal on Optimization*, 19(4):1694–1718, 2009.

D. Ellsberg. Risk, ambiguity, and the savage axioms. *The Quarterly Journal of Economics*, pages 643–669, 1961.

H.-r. Fang and Y. Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

C. L. Farmer. Uncertainty quantification and optimal decisions. *Proceedings of the Royal Society A*, 473:20170115, 2017.

Y. Feng and B. Xiao. A risk-sensitive model for managing perishable products. *Operations Research*, 56(5):1305–1311, 2008.

H. Föllmer and A. Schied. *Stochastic Finance: An Introduction in Discrete Time*. Walter de Gruyter, 2004.

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

W. Gao and D. Goldfarb. Block BFGS methods. *arXiv preprint*, arXiv:1609.00318, 2016.

N. I. M. Gould, D. Orban, and P. L. Toint. CUTEst: A constrained and unconstrained testing environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015.

H. Hotelling. The economics of exhaustible resources. *Journal of Political Economy*, 39(2):137–175, 1931.

J. Jahn. *Vector Optimization: Theory, Applications, and Extensions*. Springer, second edition, 2011.

D. Johnstone and D. Lindley. Mean–variance and expected utility: The Borch paradox. *Statistical Science*, 28(2):223–237, 2013.

S. Kalish. Monopolist pricing with dynamic demand and production cost. *Marketing Science*, 2(2):135–159, 1983.

R. L. Keeney and H. Raiffa. *Decisions with multiple objectives: preferences and value tradeoffs*. Wiley, 1976.

D. P. Kouri and T. M. Surowiec. Risk-averse PDE-constrained optimization using the Conditional Value-At-Risk. *SIAM Journal on Optimization*, 26(1):365–396, 2016.

K. Law, A. Stuart, and K. Zygalakis. *Data assimilation: A mathematical introduction*, volume 62. Springer, 2015.

A. E. Lim and J. G. Shanthikumar. Relative entropy, exponential utility, and robust dynamic pricing. *Operations Research*, 55(2):198–214, 2007.

C. Maglaras and J. Meissner. Dynamic pricing strategies for multiproduct revenue management problems. *Manufacturing & Service Operations Management*, 8(2): 136–148, 2006.

H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26(6):369–395, 2004.

R. C. Merton. Lifetime portfolio selection under uncertainty: The continuous-time case. *The Review of Economics and Statistics*, 51(3):247–257, 1969.

A. Messac. Physical programming-effective optimization for computational design. *AIAA Journal*, 34(1):149–158, 1996.

A. Messac, C. Puemi-Sukam, and E. Melachrinoudis. Aggregate objective functions and Pareto frontiers: required relationships and practical implications. *Optimization and Engineering*, 1(2):171–188, 2000.

P. K. Mogensen and A. N. Riseth. Optim: A mathematical optimization package for Julia. *Journal of Open Source Software*, 3(24):615, 2018. doi: 10.21105/joss.00615.

J. J. Moré and D. J. Thuente. Line search algorithms with guaranteed sufficient decrease. *ACM Transactions on Mathematical Software (TOMS)*, 20(3):286–307, 1994.

J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software (TOMS)*, 7(1):17–41, 1981.

S. Nadarajah, B. Zhang, and S. Chan. Estimation methods for expected shortfall. *Quantitative Finance*, 14(2):271–291, 2014.

A. Neumaier. Solving ill-conditioned and singular linear systems: A tutorial on regularization. *SIAM Review*, 40(3):636–666, 1998.

J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006.

B. Øksendal. *Stochastic Differential Equations: An introduction with applications*. Springer Verlag, 5, corrected printing edition, 2000.

C. W. Oosterlee and T. Washio. Krylov subspace acceleration of nonlinear multigrid with application to recirculating flows. *SIAM Journal on Scientific Computing*, 21 (5):1670–1690, 2000.

Ö. Özer and R. Phillips. *The Oxford handbook of pricing management*. Oxford University Press, 2012.

V. Pareto. *Manuale di Economia Politica con una Introduzione alla Scienza Sociale*. Societa Editrice Libraria, Milano, Italy, 1906. English translation by A.S. Schwier: Manual of political economy. Macmillan, 1971.

A. Pascoletti and P. Serafini. Scalarizing vector optimization problems. *Journal of Optimization Theory and Applications*, 42(4):499–524, 1984.

S. Peng. Nonlinear expectations and stochastic calculus under uncertainty. *arXiv preprint arXiv:1002.4546*, 2010.

H. Pham. *Continuous-time stochastic control and optimization with financial applications*, volume 61. Springer, 2009.

R. L. Phillips. *Pricing and revenue optimization*. Stanford University Press, 2005.

W. B. Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, second edition, 2011.

C. Rackauckas and Q. Nie. Differentialequations.jl – a performant and feature-rich ecosystem for solving differential equations in Julia. *Journal of Open Research Software*, 5(1), 2017.

K. Raman and R. Chatterjee. Optimal monopolist pricing under demand uncertainty in dynamic markets. *Management Science*, 41(1):144–162, 1995.

C. Reisinger and J. R. Arto. Boundary treatment and multigrid preconditioning for Semi-Lagrangian schemes applied to Hamilton-Jacobi-Bellman equations. *Journal of Scientific Computing*, 72(1):198–230, 2017.

C. Reisinger and R. Wissmann. Finite difference methods for medium-and high-dimensional derivative pricing PDEs. In M. A. H. Dempster, J. Kanniainen, J. Keane, and E. Vynckier, editors, *High-Performance Computing in Finance: Problems, Methods, and Solutions*. Chapman and Hall/CRC, London, 2018.

A. N. Riseth. Nonlinear solver techniques in reservoir management. Technical report, University of Oxford, 2015. URL `http://www.pefarrell.org/wp-content/uploads/2015/09/riseth2015.pdf`.

A. N. Riseth. Dynamic pricing in retail with diffusion process demand. *ArXiv e-prints*, Sept. 2017a.

A. N. Riseth. Objective acceleration for unconstrained optimization. *ArXiv e-prints*, Oct. 2017b.

A. N. Riseth and J. P. Taylor-King. Operator fitting for parameter estimation of stochastic differential equations. *ArXiv e-prints*, Sept. 2017.

A. N. Riseth, J. N. Dewynne, and C. L. Farmer. A comparison of control strategies applied to a pricing problem in retail. *ArXiv e-prints*, Oct. 2017.

A. N. Riseth, M. Wu, and S. X. Zhu. Erratum to "a risk-averse competitive newsvendor problem under the CVaR criterion". *International Journal of Production Economics*, 2018. Accepted.

R. T. Rockafellar. Coherent approaches to risk in optimization under uncertainty. *Tutorials in Operations Research*, 3:38–61, 2007.

R. T. Rockafellar and J. O. Royset. Engineering decisions under risk averseness. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 1(2):04015003, 2015.

R. T. Rockafellar and S. Uryasev. Conditional value-at-risk for general loss distributions. *Journal of Banking & Finance*, 26(7):1443–1471, 2002.

R. T. Rockafellar and S. Uryasev. The fundamental risk quadrangle in risk management, optimization and statistical estimation. *Surveys in Operations Research and Management Science*, 18(1-2):33–53, 2013.

R. T. Rockafellar, S. Uryasev, and M. Zabarankin. Generalized deviations in risk analysis. *Finance and Stochastics*, 10(1):51–74, 2006.

Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2nd edition, 2003. ISBN 0898715342.

P. A. Samuelson. Lifetime portfolio selection by dynamic stochastic programming. *The Review of Economics and Statistics*, 51(3):239–246, 1969.

R. Schlosser. Dynamic pricing with time-dependent elasticities. *Journal of Revenue and Pricing Management*, 14(5):365–383, 2015a.

R. Schlosser. Dynamic pricing and advertising of perishable products with inventory holding costs. *Journal of Economic Dynamics and Control*, 57:163–181, 2015b.

D. Scieur, A. d'Aspremont, and F. Bach. Regularized nonlinear acceleration. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, volume 29, pages 712–720. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6267-regularized-nonlinear-acceleration.pdf`.

A. Smith. *An inquiry into the nature and causes of the wealth of nations*, volume 1. Printed for W. Strahan; and T. Cadell, London., 1776.

B. C. Smith, J. F. Leimkuhler, and R. M. Darrow. Yield management at American Airlines. *Interfaces*, 22(1):8–31, 1992.

K. T. Talluri and G. J. van Ryzin. *The theory and practice of revenue management*, volume 68. Springer, 2006.

R. van de Geer, A. V. den Boer, C. Bayliss, C. Currie, A. Ellina, M. Esders, A. Haensel, X. Lei, K. D. S. Maclean, A. Martinez-Sykora, A. N. Riseth, F. Ødegaard, and S. Zachariades. Dynamic pricing and learning with competition: Insights from the dynamic pricing challenge at the 2017 informs rm & pricing conference. *ArXiv e-prints*, Mar. 2018.

J. von Neumann and O. Morgenstern. *Theory of games and economic behavior (commemorative edition)*. Princeton university press, third edition, 1953.

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

H. F. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

T. Washio and C. W. Oosterlee. Krylov subspace acceleration for nonlinear multigrid schemes. *Electronic Transactions on Numerical Analysis*, 6:271–290, 1997.

L.-L. B. Wu and D. Wu. Dynamic pricing and risk analytics under competition and stochastic reference price effects. *IEEE Transactions on Industrial Informatics*, 12 (3):1282–1293, 2016.

M. Wu, S. X. Zhu, and R. H. Teunter. A risk-averse competitive newsvendor problem under the CVaR criterion. *International Journal of Production Economics*, 156: 13–23, 2014.

X. Xu and W. J. Hopp. A monopolistic and oligopolistic stochastic flow revenue management model. *Operations Research*, 54(6):1098–1109, 2006.

W. Xue, L. Ma, and H. Shen. Optimal inventory and hedging decisions with CVaR consideration. *International Journal of Production Economics*, 162:70–82, 2015.

Y.-j. Zhou, X.-h. Chen, and Z.-r. Wang. Optimal ordering quantities for multi-products with stochastic demand: Return-CVaR model. *International Journal of Production Economics*, 112(2):782–795, 2008.