Jeremy Yee
34980 PhD Thesis: Mathematics

Subgradient and duality methods for optimal stochastic control

With applications to real options valuation

Doctor of Philosophy
University of Technology Sydney
Faculty of Science
October 2018

Certificate of original authorship

I, Jeremy Yee, declare that this thesis, is submitted in fulfilment of the requirements for the award of the Doctor of Philosophy, in the School of Mathematical and Physical Sciences at the University of Technology Sydney. This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This thesis is the result of a research candidature conducted with CSIRO as part of a collaborative Doctoral degree. This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

Production Note:
Signature removed prior to publication.

Signature:

Date: October 17, 2018

Acknowledgements

I would like to express my utmost gratitude towards my principal PhD supervisor, Juri Hinz. I would also like to extend my gratitude to my industry supervisor, Tanya Tarnopolskaya, for my time at CSIRO. I would like to acknowledge all the people I have met during my time at CSIRO, the University of Technology Sydney, the ATN Industry Doctrate Program, and all the conferences I have attended. They are too numerous to list here but their influence is appreciated. Last but not least, I would like to thank my family and friends.

Abstract

This thesis presents a subgradient and duality approach towards solving an important class of Markov decision processes. The key assumptions lie in the linear state dynamics and in the convexity of the functions in the Bellman recursion. Approximations of the value functions are then constructed using convex piecewise linear functions formed using operations on tangents. This approach can be efficiently implemented with most of the computational effort reduced to simple matrix additions and multiplications. The quality of the approximations can then be gauged using pathwise duality methods which return confidence intervals for the true unknown value. This thesis will then explore the use of nearest neighbour algorithms in reducing the computational effort. Numerical experiments demonstrate that the subgradient and duality approach returns tight confidence intervals indicating good quality results. These methods are tested on a wide range of applications such as financial option pricing, optimal liquidation problems, battery control, and natural resource extraction. Extension to uncontrolled hidden Markov models is also considered. These methods have all been implemented in a *R* package with scripts posted online to reproduce most of the numerical results within this thesis.

Contents

Chapter 1

Introduction

1.1 Literature

The term *real options* was first introduced by Myers (1977) to frame corporate assets as a type of financial call option. Since then, researchers have readily adopted this term to refer to the use of option valuation techniques for captial budgeting decisions. The literature devoted to this area is well developed (see Trigeorgis (1996) and the references contained within) and addresses issues faced by more traditional capital budgeting tools such as *discounted cash flow* which ignores the dynamic flexibility in some captial projects. For example, the classical paper by McDonald and Siegel (1985) used this approach to value firms when they have the flexibility to suspend business during unfavourable conditions. Other applications can be found in petroleum exploration and production (Dias, 2004), mining investments (Cortazar et al, 2001), and other commodity projects (Devalkar et al, 2011). More broadly, real options analysis falls under the umbrella of *optimal stochastic control* in which there are many tools available. This thesis will restrict its attention to cases where the underlying stochastic process is a Markov process. In a continuous time setting, stochastic differential equations e.g. the Hamilton-Jacobi-Bellman (HJB) approach or the backward stochastic differential equations (BSDE) are often used due its well established theory (Dokuchaev and Zhou, 1999; Oksendal and Sulem, 2005; Pham, 2009; Bender and Dokuchaev, 2014). However, obtaining theoretical solutions is often tedious. While the associated partial differential equations may be solved using numerical methods (e.g. finite differences), they tend to suffer from the *curse of dimensionality*. In a discrete time setting, *Markov decision processes* (Bauerle and Rieder, 2011; Puterman, 1994) provides us with a very powerful framework and they can often be solved intuitively using the Bellman equations or dynamic programming principle.

With the advent of cheap powerful computers, some authors (Powell, 2007) have suggested the use of approximate solutions to address the curse of dimensionality. Following this vein, numerous Monte Carlo simulation approaches have appeared e.g. simulated trees (Broadie and Glasserman, 1997), stochastic mesh (Broadie and Glasserman, 2004), and regression based methods (Carriere, 1996; Tsitsiklis and Van Roy, 2001). The review paper by Fu et al (2001) gives a comparison of some of these methods on financial options. The book by Glasserman (2003) provides a comprehensive overview of Monte Carlo methods in financial engineering. The least squares Monte Carlo approach (Carriere, 1996; Tsitsiklis and Van Roy, 1999, 2001; Longstaff and Schwartz, 2001) has enjoyed widespread popularity. This method approximates the conditional expectation in the Bellman recursion as a linear combination of selected basis functions using statistical regression. Convergence properties are addressed in Clement et al (2002) and later generalized in Stentoft (2004); Egloff (2005); Egloff et al (2007). Extensions to multiple exercise rights are considered in Carmona and Touzi (2008), and studied in Belomestny et al (2010) where the connections to statistical learning theory and the theory of empirical processes are emphasized. The use of advanced regression methods such as kernel methods (Ormoneit and Sen, 2002; Ormoneit and Glynn, 2002), local polynomial regression (Fan and Gijbels, 1996), and neural networks (Bertsekas and Tsitsiklis, 1996) have also been studied. An overview on the applications of least squares Monte Carlo methods for energy real options is given by Nadarajah et al (2017).

One of the main advantages of the least squares Monte Carlo methodology is that it reduces computations to simple algebraic operations. Its theoretical justification relies on convergence results stating that if both the number of basis functions and the number of simulated paths grow, the true unknown solution can be approached. However, their growth rates should not be chosen independently of each other. Under appropriate assumptions, Stentoft (2004), Glasserman and Yu (2004), and Belomestny et al (2010) provide conditions on the relative growth rates between the number of paths and the size of the regression basis. In practice, least squares Monte Carlo encounters two major problems: 1) an appropriate choice of the regression basis is often difficult; and 2) increasing the size of the basis may cause unwanted oscillations if the sample size is too small. In applications, the last issue is critical since high dimensional state space may require high basis dimension. The above Monte Carlo methods are often justified by asymptotic arguments and this may be of little practical use. However, duality methods studied by authors such as Rogers (2002); Haugh and Kogan (2004); Andersen and Broadie (2004); Rogers (2007); Chen and Glasserman (2007); Brown et al (2010) offers a way to gauge the quality of the numerical solutions. In this approach, the value of the decision process is characterized as

an infimum over martingales and so one is left with a practical upper bound. It turns out that the minimising martingale is directly related to the true value functions. Therefore, if one were to construct the martingale from the value function approximations instead, the upper bound exhibits an interesting *self-tuning* property. The closer the approximations are to their true counterparts, the lower the upper bound. This thesis will exploit this to gauge the quality of our numerical results.

1.2 Contribution

Numerical approaches to Markov decision processes with uncountable state spaces typically use either a finite discretization of the state space (Pages et al, 2004; Gray and Neuhoff, 2006) or using a finite dimensional approximation of the target functions (Tsitsiklis and Van Roy, 2001). This thesis will focus on the latter approach for problems containing only a finite number of actions and assuming the functions in the Bellman recursion satisfy certain convexity assumptions. Convexity assumptions are often used in practice because it affords many numerical benefits and the theory is well developed (Rockafellar, 1970). For example, Hannah and Dunson (2013) found success in exploiting convexity for the least squares Monte Carlo algorithm. The work done by Hernandez-Lerma and Runggaldier (1994); Hernandez-Lerma et al (1995) combines convexity assumptions with an appropriate discretization of the driving random variable to construct monotone increasing and decreasing bounding function approximations for the true value.

This thesis has been largely motivated by Hinz (2014), in which the author approximates the value functions under linear state dynamics using convex piecewise linear functions formed by matrix operations on tangents. The work done by Hinz (2014) will be presented in Chapter 2 of this thesis for completeness sake. In the following, the novel contributions of this thesis are outlined clearly.

- **Chapter 3**:
  - This chapter extends the use of nearest neighbour algorithms to reduce the computational effort in the subgradient approach as proposed by Hinz and Yap (2015). More specifically, Hinz and Yap (2015) considered the use of the nearest neighbour while we will generalize this to the use of the $N \geq 1$ nearest neighbours.
  - Nearest neighbour algorithms are used to reduce the computational effort in the computation of the additive duals in the solution diagnostics. For the case where the driving

random disturbance takes on only a finite number of possible values, the computational saving is drastic.

- **Chapter 4**:
  - Results from the subgradient and duality approach are compared to published least squares Monte Carlo results on Bermudan put, swing option, and Bermudan max-call options.
  - The subgradient and duality approach is applied to optimal portfolio liquidation problems. This work has already resulted in our publication in Hinz and Yee (2017a).
  - The subgradient and duality approach is applied to optimal battery control for electricity retailers. This work has already resulted in our publication in Hinz and Yee (2017b).
  - The subgradient and duality approach is applied to natural resource extraction. This work has resulted in our paper Hinz et al (In Press), currently in press.

- **Chapter 5**:
  - This chapter studies the extension of the problem setting to cases involving hidden Markov models as proposed by Hinz (2016). The novelty in this chapter lies in the extension of the duality approach to this setting as well. This approach is then demonstrated on a numerical example (Point & Figure charting) from technical analysis with excellent results. This work has already resulted in our publication Hinz and Yee (2017c).

- **Chapter 6**:
  - This chapter presents the first implementation of our algorithms in the programming language *R*. This software package provides a standardized framework for users to solve a very wide range of problems and was used to generate all numerical results in this thesis. An implementation in the language *Julia* is also publicly available. This chapter is based on our accepted paper Hinz and Yee (In Press).

As a final remark, the value function approximation method presented in this thesis has been extended to non-linear state dynamics and more general convex function approximation schemes in Yee (Preprinta). In Yee (Preprinta), the author also presents conditions to construct monotone sequences of lower bounding and upper bounding functions in the spirit of Hernandez-Lerma and Runggaldier (1994). Further, these results have been extended by Yee (Preprintb) for the infinite horizon contracting case. For the brevity of this thesis, these extensions have been omitted from this thesis.

References

Andersen L, Broadie M (2004) Primal-dual simulation algorithm for pricing multidimensional american options. Management Science 50(9):1222–1234

Bauerle N, Rieder U (2011) Markov Decision Processes with applications to finance. Springer, Heidelberg, DOI 10.1007/978-3-642-18324-9

Belomestny D, Kolodko A, Schoenmakers J (2010) Regression methods for stochastic control problems and their convergence analysis. SIAM Journal on Control and Optimization 48(5):3562–3588, DOI 10.1137/090752651

Bender C, Dokuchaev N (2014) A first-order bspde for swing option pricing. Mathematical Finance 26(3):461–491, DOI 10.1111/mafi.12067

Bertsekas D, Tsitsiklis J (1996) Neuro-Dynamic Programming. Athena Scientific

Broadie M, Glasserman P (1997) Pricing american-style securities using simulation. Journal of Economic Dynamics and Control 21(8):1323 – 1352, DOI 10.1016/S0165-1889(97)00029-8

Broadie M, Glasserman P (2004) A stochastic mesh method for pricing high-dimensional American options. Journal of Computational Finance 7(4):35 – 72, DOI 10.21314/JCF.2004.117

Brown D, Smith J, Sun P (2010) Information relaxations and duality in stochastic dynamic programs. Operations Research 58(4):785–801

Carmona R, Touzi N (2008) Optimal multiple stopping and valuation of swing options. Mathematical Finance 18:239 – 268

Carriere J (1996) Valuation of the early-exercise price for options using simulations and nonparametric regression. Insurance: Mathematics and Economics 19:19–30, DOI 10.1016/S0167-6687(96)00004-2

Chen N, Glasserman P (2007) Additive and multiplicative duals for american option pricing. Finance and Stochastics 11(2):153–179

Clement E, Lamberton D, Protter P (2002) An analysis of the Longstaff-Schwartz algorithm for American option pricing. Finance and Stochastics 6(4):449–471

Cortazar G, Schwartz E, Casassus J (2001) Optimal exploration investments under price and geological-technical uncertainty: A real options model. R&D Management 31(2):181–189, DOI 10.1111/1467-9310.00208

Devalkar S, Anupindi R, Sinha A (2011) Integrated optimization of procurement, processing, and trade of commodities. Operations Research 59(6):1369–1381, DOI 10.1287/opre.1110.0959

Dias M (2004) Valuation of exploration and production assets: An overview of real options models. Journal of Petroleum Science and Engineering 44:93–114

Dokuchaev N, Zhou X (1999) Stochastic controls with terminal contingent conditions. Journal of Mathematical Analysis and Applications 238(1):143 – 165, DOI 10.1006/jmaa.1999.6515

Egloff D (2005) Monte Carlo algorithms for optimal stopping and statistical learning. The Annals of Applied Probability 15:1396–1432

Egloff D, Kohler M, Todorovic N (2007) A dynamic look-ahead Monte Carlo algorithm for pricing Bermuda options. The Annals of Applied Probability 17:1138–1171

Fan J, Gijbels I (1996) Local Polynomial Modelling and Its Applications. Chapman and Hall

Fu M, Laprise S, Madan D, Su Y, Wu R (2001) Pricing american options: A comparison of monte carlo simulation approaches. Journal of Computational Finance 4(3):39 – 88

Glasserman P (2003) Monte Carlo Methods in Financial Engineering. Springer

Glasserman P, Yu B (2004) Number of paths versus number of basis functions in american option pricing. Annals of Applied Probability 14:2090–2119, DOI 10.1214/105051604000000846

Gray R, Neuhoff D (2006) Quantization. IEEE Trans Inf Theor 44(6):2325–2383, DOI 10.1109/18.720541

Hannah L, Dunson D (2013) Multivariate convex regression with adaptive partitioning. Journal of Machine Learning Research 14:3261–3294

Haugh M, Kogan L (2004) Pricing American options: A duality approach. Operations Research 52(2):258–270

Hernandez-Lerma O, Runggaldier W (1994) Monotone approximations for convex stochastic control problems. Journal of Mathematical Systems, Estimation, and Control 4(1):99–140

Hernandez-Lerma O, Piovesan C, Runggaldier W (1995) Numerical aspects of monotone approximations in convex stochastic control problems. Annals of Operations Research 56(1):135–156, DOI 10.1007/BF02031704

Hinz J (2014) Optimal stochastic switching under convexity assumptions. SIAM Journal on Control and Optimization 52(1):164–188, DOI 10.1137/13091333X

Hinz J (2016) Using convex switching techniques for partially observable decision processes. IEEE Transactions on Automatic Control 61(9):2727–2732

Hinz J, Yap N (2015) Algorithms for optimal control of stochastic switching systems. Theory of Probability and its Applications 60(4):770–800

Hinz J, Yee J (2017a) An algorithmic approach to optimal asset liquidation problems. Asia-Pacific Financial Markets 24(2):109–129, DOI 10.1007/s10690-017-9226-1

Hinz J, Yee J (2017b) Optimal forward trading and battery control under renewable electricity generation. Journal of Banking & Finance InPress, DOI 10.1016/j.jbankfin.2017.06.006

Hinz J, Yee J (2017c) Stochastic switching for partially observable dynamics and optimal asset allocation. International Journal of Control 90(3):553–565

Hinz J, Yee J (In Press) rcss: R package for optimal convex stochastic switching. The R Journal

Hinz J, Tarnopolskaya T, Yee J (In Press) Efficient algorithms of pathwise dynamic programming for decision optimization in mining operations. Annals of Operations Research

Longstaff F, Schwartz E (2001) Valuing American options by simulation: a simple least-squares approach. Review of Financial Studies 14(1):113–147, DOI 10.1093/rfs/14.1.113

McDonald R, Siegel D (1985) Investment and the valuation of firms when there is an option to shut down. International Economic Review 26(2):331–349

Myers S (1977) Determinants of corporate borrowing. Journal of Financial Economics 5(2):147 – 175, DOI 10.1016/0304-405X(77)90015-0

Nadarajah S, Margot F, Secomandi N (2017) Comparison of least squares Monte Carlo methods with applications to energy real options. European Journal of Operational Research 256(1):196 – 204, DOI 10.1016/j.ejor.2016.06.020

Oksendal B, Sulem A (2005) Applied Stochastic Control of Jump Diffusions. Springer Verlag, Berlin Heidelberg New-York

Ormoneit D, Glynn P (2002) Kernel-based reinforcement learning in average-cost problems. IEEE Transactions in Automatic Control 47:1624–1636

Ormoneit D, Sen S (2002) Kernel-based reinforcement learning. Machine Learning 49:161–178

Pages G, Pham H, Printems J (2004) Optimal Quantization Methods and Applications to Numerical Problems in Finance, Birkhäuser Boston, pp 253–297. DOI 10.1007/978-0-8176-8180-7_7

Pham H (2009) Continuous-time stochastic control and optimization with financial applications, vol 61. Springer Science & Business Media

Powell W (2007) Approximate dynamic programming: Solving the curses of dimensionality. Wiley, Hoboken, New Jersey, DOI 10.1002/9781118029176

Puterman M (1994) Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York, DOI 10.1002/9780470316887

Rockafellar R (1970) Convex Analysis. Princeton landmarks in mathematics and physics, Princeton University Press

Rogers L (2002) Monte carlo valuation of american options. Mathematical Finance 12(3):271–286

Rogers L (2007) Pathwise stochastic optimal control. SIAM J Control Optimisation 46(3):1116–1132

Stentoft L (2004) Convergence of the least squares Monte Carlo approach to American option valuation. Management Science 50(9):576–611, DOI 10.1287/mnsc.1030.0155

Trigeorgis L (1996) Real options: Managerial flexibility and strategy in resource allocation. MIT press

Tsitsiklis J, Van Roy B (1999) Optimal stopping of Markov processes: Hilbert space, theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. IEEE Transactions on Automatic Control 44(10):1840–1851, DOI 10.1109/9.793723

Tsitsiklis J, Van Roy B (2001) Regression methods for pricing complex American-style options. IEEE Transactions on Neural Networks 12(4):694–703, DOI 10.1109/72.935083

Yee J (Preprinta) Convex function approximations for Markov decision processes. arXiv:171200970

Yee J (Preprintb) Value iteration for approximate dynamic programming under convexity. arXiv:180207243

Chapter 2

Subgradient and duality

2.1 Problem setting

The following gives the problem setting considered by Hinz (2014). Given a finite time horizon $t = 0, \ldots, T$, consider a controlled Markov process $(X_t)_{t=0}^T := (P_t, Z_t)_{t=0}^T$ consisting of two parts. The controlled discrete component $(P_t)_{t=0}^T$ describes the evolution of a finite-state controlled Markov chain which takes values in a finite set $\mathbf{P}$. Suppose that at any time $t = 0, \ldots, T-1$ the controller takes an action $a \in \mathbf{A}$ from a finite set $\mathbf{A}$ of all admissible actions in order to cause the one-step transition from the mode $p \in \mathbf{P}$ to the mode $p' \in \mathbf{P}$ with probability $\alpha_{p,p'}^a$, where $(\alpha_{p,p'}^a)_{p,p' \in \mathbf{P}}$ are pre-specified transition probabilities for all $a \in \mathbf{A}$. Now the uncontrolled component $(Z_t)_{t=0}^T$ takes values in an open convex set $\mathbf{Z} \subseteq \mathbb{R}^d$ and is governed via

$$Z_{t+1} = W_{t+1} Z_t, \qquad t = 0, \ldots, T-1$$

by independent random *disturbance matrices* $(W_t)_{t=1}^T$. These random $d \times d$ matrices are assumed to be integrable i.e. the matrix entries are integrable. The transition operator $\mathcal{K}_t^a$ associated with the transition kernels K_t^a governing the evolution of the Markov process $(X_t)_{t=0}^T := (P_t, Z_t)_{t=0}^T$ from time t to $t+1$ is given for each $a \in \mathbf{A}$ by

$$\mathcal{K}_t^a v(p, z) = \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \mathbb{E}[v(p', W_{t+1} z)], \quad p \in \mathbf{P}, \quad z \in \mathbf{Z}, \quad t = 0, \ldots, T-1$$

which acts on function $v : \mathbf{P} \times \mathbf{Z} \to \mathbb{R}$ whenever the above expectations are well-defined.

If the system is in state (p, z), the costs of applying action $a \in \mathbf{A}$ at time $t = 0, \ldots, T-1$ are expressed by function $r_t(p, z, a)$. Having arrived at time $t = T$ in state (p, z), a final *scrap value* $r_T(p, z)$ is collected. The reward and scrap functions

$$r_t : \mathbf{P} \times \mathbf{Z} \times \mathbf{A} \to \mathbb{R}, \quad r_T : \mathbf{P} \times \mathbf{Z} \to \mathbb{R}$$

are exogenously given for $t = 0, \ldots, T-1$. At each time $t = 0, \ldots, T$, the *decision rule* π_t is given by a mapping $\pi_t : \mathbf{P} \times \mathbf{Z} \to \mathbf{A}$, prescribing at time t an action $\pi_t(p, z) \in \mathbf{A}$ for a given state $(p, z) \in \mathbf{P} \times \mathbf{Z}$. A sequence $\pi = (\pi_t)_{t=0}^{T-1}$ of decision rules is called a *policy*. For each policy $\pi = (\pi_t)_{t=0}^{T-1}$, the so-called *policy value* $v_0^\pi(p_0, z_0)$ is defined as the total expected reward

$$v_0^\pi(p_0, z_0) = \mathbb{E}^{(p_0,z_0),\pi} \left[\sum_{t=0}^{T} r_t(P_t, Z_t, \pi_t(P_t, Z_t)) + r_t(P_t, Z_t) \right].$$

Here, $\mathbb{E}^{(p_0,z_0),\pi}$ stands for the expectation with respect to the probability distribution of $(X_t)_{t=0}^T := (P_t, Z_t)_{t=0}^T$ defined by Markov transitions from (P_t, Z_t) to (P_{t+1}, Z_{t+1}), which are induced by the kernels $K_t^{\pi_t(P_t,Z_t)}$ for $t = 0, \ldots, T-1$, started at the initial point $(P_0, Z_0) = (p_0, z_0)$. A policy $\pi^* = (\pi_t^*)_{t=0}^{T-1}$ is called optimal if it maximizes the total expected reward over all policies $\pi \mapsto v_0^\pi(p, z)$. To obtain such policy, one introduces for $t = 0, \ldots, T-1$, the so-called *Bellman operator*

$$\mathcal{T}_t v(p, z) = \max_{a \in \mathbf{A}} \left[r_t(p, z, a) + \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \mathbb{E}[v(p', W_{t+1} z)] \right]$$

for $(p, z) \in \mathbf{P} \times \mathbf{Z}$, acting on all functions v wherever the stochastic kernel is defined. It is clear that the Bellman operator inherits monotonicity from the expectations i.e. if for $p \in \mathbf{P}$ and $z \in \mathbf{Z}$ that $v'(p, z) \leq v''(p, z)$ then $\mathcal{T}_t v'(p', z') \leq \mathcal{T}_t v''(p', z')$ for $p' \in \mathbf{P}$, $z' \in \mathbf{Z}$, and $t = 0, \ldots, T-1$.

Now consider the *Bellman recursion*

$$v_T(p, z) = r_T(p, z), \quad v_t(p, z) = \mathcal{T}_t v_{t+1}(p, z) \tag{2.1.1}$$

for $p \in \mathbf{P}$, $z \in \mathbf{Z}$, and $t = T-1, \ldots, 0$. If it exists, a recursive solution $(v_t^*)_{t=0}^T$ to the above is called the *value functions* and they determine an optimal policy $\pi^* = (\pi_t^*)_{t=0}^T$ via

$$\pi_t^*(p, z) = \arg\max_{a \in \mathbf{A}} \left[r_t(p, z, a) + \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \mathbb{E}[v_{t+1}^*(p', W_{t+1} z)] \right],$$

for $t = T-1, \ldots, 0$. The existence of a solution $(v_t^*)_{t=0}^T$ depends on the functions and expectations in the Bellman recursion being well defined. This occurs, for example, when the reward and scrap functions are globally Lipschitz continuous in z (see Bauerle and Rieder (2011) for more diverse conditions). In this thesis, global Lipschitz continuity is simply referred to as Lipschitz

continuity for shorthand. The choice of the norm $\|\cdot\|$ below is not important since all norms are equivalent on a finite dimensional vector space.

Assumption 1 *The reward $r_t(p,z,a)$ and scrap $r_T(p,z)$ are convex and Lipschitz continuous in z for $p \in \mathbf{P}$, $a \in \mathbf{A}$, and $t = 0,\ldots,T-1$*

With the above assumption, Hinz (2014) (see Lemma 5.1 in Hinz (2014)) proved the existence of the value functions by deriving an upper bounding function for the Markov decision process. However, the following proof offers a more direct approach.

Theorem 2.1. *There exists a recursive solution $(v_t^*(p,z))_{t=0}^T$ to the Bellman recursion. Furthermore, these value functions are Lipschitz continuous in z for all $p \in \mathbf{P}$.*

Proof. Note that $v_T^*(p,W_T z) = r_T(p,W_T z)$ is integrable from the Lipschitz continuity of $r_T(p,z)$ in z and the integrability of W_T. Therefore, $\mathbb{E}[v_T^*(p,W_T z)]$ exists for all $p \in \mathbf{P}$ and $z \in \mathbf{Z}$. Moreover, $\mathbb{E}[v_T^*(p,W_T z)]$ is Lipschitz in z. To see this, observe that for $z', z'' \in \mathbf{Z}$, the Lipschitz continuity of $v_T^*(p,z)$ in z yields

$$\left|\mathbb{E}[v_T^*(p,W_T z')] - \mathbb{E}[v_T^*(p,W_T z'')]\right| \leq c\,\mathbb{E}[||W_T||]\;||z'-z''||$$

for $p \in \mathbf{P}$ and some constant c. Now since the reward functions are Lipshitz in z for all $p \in \mathbf{P}$ and $a \in \mathbf{A}$, the resulting sum and maximization $v_{T-1}^*(p,z)$ exists and is also Lipschitz in z since $\mathbf{A}$ and $\mathbf{P}$ are finite. Proceeding inductively for $t = T-2,\ldots,0$ proves the theorem. □

The next theorem guarantees the convexity of the value functions and the expected vale functions in z. This feature is vital for the subgradient approach presented in the next section.

Theorem 2.2. *The functions $(v_t^*(p,z))_{t=0}^T$ and $(\mathbb{E}[v_{t+1}^*(p,W_{t+1}z)])_{t=0}^{T-1}$ are convex in z for all $p \in \mathbf{P}$.*

Proof. At $t = T$, the result obviously holds due to the assumptions placed on the scrap function r_T. For any realization w of W_T, the value function $v_T^*(p,wz) = r_T(p,wz)$ is convex in z. To see this, recall that the matrix w represents a linear mapping and the function composition of a convex function after a linear mapping yields a convex function. Therefore, $v_T^*(p,W_T z)$ is convex in z with probability one and so

$$\lambda v_T^*(p,W_T z') + (1-\lambda) v_T^*(p,W_T z') - v_T^*(p,W_T(\lambda z' + (1-\lambda)z'')) \geq 0$$

holds almost surely for all $\lambda \in [0,1]$ and $z', z'' \in \mathbf{Z}$. Taking expectations of the above reveals that $\mathbb{E}[v_T^*(p, W_T z)]$ is also convex in z. Now since the reward functions are convex in z for all $p \in \mathbf{P}$ and $a \in \mathbf{A}$, the resulting sum and maximization $v_{T-1}^*(p,z)$ is also convex in z for all $p \in \mathbf{P}$ since the sum of a finite number of convex functions is convex and the pointwise maximum of a finite number of convex functions is also convex. Proceeding inductively for $t = T-2, \ldots, 0$ proves the above theorem. □

2.2 Subgradient approach

The first step in obtaining a numerical solution to the backward induction (2.1.1) is an appropriate discretization of the transition operator to

$$\mathcal{K}_t^{a,(n)} v(p,z) = \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \sum_{k=1}^{n} \rho_{t+1}^{(n)}(k) v(p', W_{t+1}^{(n)}(k) z)$$

where the non-negative weights $(\rho_{t+1}^{(n)}(k))_{k=1}^n$ correspond to sampling $(W_{t+1}^{(n)}(k))_{k=1}^n$ of each disturbance W_{t+1}. In the resulting modified backward induction governed by $v_t^{(n)} = \mathcal{T}_t^{(n)} v_{t+1}^{(n)}$, the modified functions $(v_t^{(n)})_{t=0}^T$ need to be described by algorithmically tractable objects. Since the reward and scrap functions are convex in the continuous variable, these modified value functions are also convex and can be approximated by piecewise linear and convex functions. Introduce the so-called subgradient envelope $\mathcal{S}_{\mathbf{G}^{(m)}} f$ of a convex function $f : \mathbf{Z} \to \mathbb{R}$ on a grid $\mathbf{G}^{(m)} \subset \mathbf{Z}$ with m points by

$$\mathcal{S}_{\mathbf{G}^{(m)}} f = \vee_{g \in \mathbf{G}^{(m)}} (\nabla_g f)$$

which takes the maximum of the tangents $\nabla_g f$ of f on all grid points $g \in \mathbf{G}^{(m)}$.

Definition 2.1. The function $\nabla_g f(z) := f(g) + c^T(z-g)$ for some $c \in \mathbb{R}^d$ is a tangent to convex function $f : \mathbf{Z} \to \mathbb{R}$ at $g \in \mathbf{Z}$ if $f(z) \geq \nabla_g f(z)$ for all $z \in \mathbf{Z}$.

Recall that if a function is convex on $\mathbf{Z}$, then it is also continuous on $\mathbf{Z}$ since $\mathbf{Z}$ is an open convex set. The convex function $f : \mathbf{Z} \to \mathbb{R}$ has tangents on every point in $\mathbf{Z}$ (Theorem 23.4 in Rockafellar (1970)). Note that the tangents may not be unique at some point e.g. when there is a kink in the function. For this case, we just choose any one of the possible tangents. This thesis assumes that $\mathbf{G}^{(m)} \subset \mathbf{G}^{(m+1)}$ and $\cup_{m \in \mathbb{N}} \mathbf{G}^{(m)}$ is dense in $\mathbf{Z}$. With this, the subgradient envelope has the following desirable properties:

- $\mathcal{S}_{\mathbf{G}^{(m)}} f$ converges uniformly to f on compact sets as $m \to \infty$,

- $\mathcal{S}_{\mathbf{G}^{(m)}} f$ is a convex and Lipschitz continuous function,
- $\mathcal{S}_{\mathbf{G}^{(m)}} f(z) \leq \mathcal{S}_{\mathbf{G}^{(m+1)}} f(z) \leq f(z)$ for all $z \in \mathbf{Z}$ and $m \in \mathbb{N}$, and
- $\mathcal{S}_{\mathbf{G}^{(m)}} f'(z) \leq \mathcal{S}_{\mathbf{G}^{(m)}} f''(z)$ for $z \in \mathbf{Z}$, $m \in \mathbb{N}$, and for all convex functions satisfying $f'(z') \leq f''(z')$ for $z' \in \mathbf{Z}$.

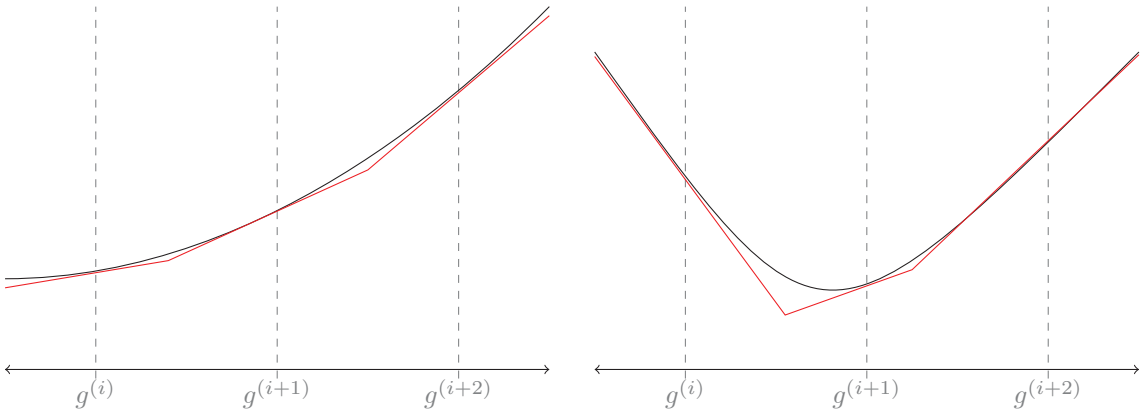

Fig. 2.2.1: Approximating the target functions (black) using the subgradient envelope (red).

An appropriate choice of the grid $\mathbf{G}^{(m)}$ reduces the approximation error associated with the subgradient envelope. Since the expected value functions in the Bellman recursion are typically unknown, the optimal choice of $\mathbf{G}^{(m)}$ for some m is also unknown. However, in some of our applications, an evenly spaced grid return good quality results. For grid selection in high dimensional state spaces, the use of a so-called *stochastic grid* may be recommended. That is, simulate a set of paths and perform k-means clustering (or similar techniques) on all states visited to obtain an appropriate grid e.g. set

$$\mathbf{G}^{(k)} = \text{k-means(all states visited in simulation runs)}$$

where the function k-means() returns the centers of each cluster after applying the k-means clustering algorithm to the point cloud. This approach will be demonstrated in some of our numerical applications. Finally, please keep in mind that the grid above can be made time dependent without affecting the convergence results given in the next section. To reduce the notation clutter, this is not done. Now define the double-modified Bellman operator

$$\mathcal{T}_t^{(m,n)} v(p,z) = \max_{a \in \mathbf{A}} \left(\mathcal{S}_{\mathbf{G}^{(m)}} r_t(p,z,a) + \mathcal{S}_{\mathbf{G}^{(m)}} \mathcal{K}_t^{a,(n)} v(p,z) \right)$$

and the corresponding backward induction

$$v_{T-1}^{(m,n)}(p,z) = \mathcal{T}_{T-1}^{(m,n)} \mathcal{S}_{\mathbf{G}^{(m)}} r_T(p,z,a), \qquad v_t^{(m,n)} = \mathcal{T}_t^{(m,n)} v_{t+1}^{(m,n)}(p,z) \tag{2.2.1}$$

for $p \in \mathbf{P}$, $z \in \mathbf{Z}$, $a \in \mathbf{A}$, and $t = T-2, \dots 0$ which yields the so-called *double-modified value functions* $(v_t^{(m,n)})_{t=0}^T$. Please note that in the above, the subgradient envelope acts on the functions through the continuous variable z. It is not hard to see that if for $p \in \mathbf{P}$ and $z \in \mathbf{Z}$ that $v'(p,z) \leq v''(p,z)$ then $\mathcal{T}_t^{(m,n)} v'(p',z') \leq \mathcal{T}_t^{(m,n)} v''(p',z')$ for all $p' \in \mathbf{P}$, $z' \in \mathbf{Z}$, $m,n \in \mathbb{N}$ and $t = 0, \dots, T-1$. That is, the double modified Bellman operator is also monotone. Also note that the double modified value functions $(v_t^{(m,n)}(p,z))_{t=0}^T$ are Lipschitz continuous in z for all $m,n \in \mathbb{N}$ and $p \in \mathbf{P}$ due to the application of $\mathcal{S}_{\mathbf{G}^{(m)}}$.

Since $(v_t^{(m,n)})_{t=0}^T$ are piecewise linear and convex, they can be expressed using matrix representations where each linear functional is represented by a row in the matrix. Let us agree on the following notation: Given a function f and a matrix F, we write $f \sim F$ whenever $f(z) = \max(Fz)$ holds for all $z \in \mathbf{Z}$. Let us emphasize that the subgradient envelope operation $\mathcal{S}_{\mathbf{G}^{(m)}}$ is reflected in terms of a matrix representative by a specific row-rearrangement operator

$$f \sim F \quad \Leftrightarrow \quad \mathcal{S}_{\mathbf{G}^{(m)}} f \sim \Upsilon_{\mathbf{G}^{(m)}}[F]$$

where the row-rearrangement operator $\Upsilon_{\mathbf{G}^{(m)}}$ acts on matrix F with d columns as follows:

$$(\Upsilon_{\mathbf{G}^{(m)}} F)_{i,\cdot} = F_{\operatorname{argmax}(Fg^i),\cdot} \qquad \text{for all } i = 1, \dots, m.$$

For convex piecewise linear functions, the result of maximization, summation, and composition with linear mapping, followed by subgradient envelope can be obtained using their matrix representatives. More precisely, if $f_1 \sim F_1$ and $f_2 \sim F_2$ holds, it follows that

$$\begin{aligned}
\mathcal{S}_{\mathbf{G}^{(m)}}(f_1 + f_2) &\sim \Upsilon_{\mathbf{G}^{(m)}}(F_1) + \Upsilon_{\mathbf{G}^{(m)}}(F_2) \\
\mathcal{S}_{\mathbf{G}^{(m)}}(f_1 \vee f_2) &\sim \Upsilon_{\mathbf{G}^{(m)}}(F_1 \sqcup F_2) \\
\mathcal{S}_{\mathbf{G}^{(m)}}(f_1(W\cdot) &\sim \Upsilon_{\mathbf{G}^{(m)}}(F_1 W)
\end{aligned}$$

where W is an arbitrary $d \times d$ matrix and the operator $\sqcup$ stands for binding matrices by rows. Therefore, the backward induction (2.2.1) can be expressed in terms of the matrix representatives $V_t^{(m,n)}(p)$ of the value functions $v_t^{(m,n)}(p,z)$ for $p \in \mathbf{P}$ and $t = 0, \dots T-1$. This approach is outlined in Algorithm 2.2.1.

Algorithm 2.2.1: Value function approximation

for $p \in \mathbf{P}$ **do**
  $V_T^{(m,n)}(p) \leftarrow R_T^{(m)}(p,\cdot)$
end
for $t \in \{T-1,\ldots,0\}$ **do**
  for $p \in \mathbf{P}$ **do**
    $\widetilde{V}_{t+1}^{(m,n)}(p) \leftarrow \sum_{k=1}^{n} \rho_{t+1}^{(n)}(k) \Upsilon_{\mathbf{G}^{(m)}} V_{t+1}^{(m,n)}(p) W_{t+1}^{(n)}(k)$
  end
  for $p \in \mathbf{P}$ **do**
    $V_t^{(m,n)}(p) \leftarrow \Upsilon_{\mathbf{G}^{(m)}} \sqcup_{a\in\mathbf{A}} \left(R_t^{(m)}(p,\cdot,a) + \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^{a} \widetilde{V}_{t+1}^{(m,n)}(p') \right)$
  end
end

In the following algorithm listing, matrices $R_T^{(m)}(p,\cdot)$ and $R_t^{(m)}(p,\cdot,a)$ are the matrix representatives of $\mathcal{S}_{\mathbf{G}^{(m)}} r_T(p,\cdot)$ and $\mathcal{S}_{\mathbf{G}^{(m)}} r_t(p,.,a)$, respectively. Notice that Algorithm 2.2.1 is written in terms dimension-independent matrix operations. In this sense, the implementation philosophy does not depend on state dimensions. However, the calculation times are dimension-dependent due to size of the processed data fields. Now a candidate for a nearly optimal policy is given by

$$\pi_t^{(m,n)}(p,z) = \arg\max_{a\in\mathbf{A}} \left[r_t(p,z,a) + \max \left(\sum_{p'\in\mathbf{P}} \alpha_{p,p'}^{a} \widetilde{V}_{t+1}^{(m,n)}(p') z \right) \right] \tag{2.2.2}$$

where $\widetilde{V}_{t+1}^{(m,n)}$ is obtained by our value function approximation approach in Algorithm 2.2.1. The next section examines the convergence properties of the double modified value functions.

2.3 Convergence

The following concept is used throughout this section.

Definition 2.2. A sequence of convex and Lipschitz continuous real-valued functions $(h^{(n)})_{n\in\mathbb{N}}$ is called a *ULCC* sequence on $\mathbf{Z}$ if $(h^{(n)})_{n\in\mathbb{N}}$ converges uniformly on all compact sets of $\mathbf{Z}$ and all the functions in the sequence share a common Lipschitz constant.

Now introduce partition $\mathit{\Pi}^{(n)} = \{\mathit{\Pi}^{(n)}(k) \subset \mathbf{W} : k = 1,\ldots,n\}$ such that generated sigma-algebras $\sigma_{t+1}^{(n)} = \sigma(\{W_{t+1} \in \mathit{\Pi}^{(n)}(k)\},\ k = 1,\ldots,n)$ satisfy $\sigma(W_{t+1}) = \sigma(\cup_{n\in\mathbb{N}} \sigma_{t+1}^{(n)})$ for $t = 0,\ldots,T-1$. Assume that $\mathit{\Pi}^{(n+1)}$ is a refinement of $\mathit{\Pi}^{(n)}$ i.e. each component in $\mathit{\Pi}^{(n+1)}$ is

a subset of a component in $\Pi^{(n)}$. To avoid any potential confusion, the conditional expectation $\mathbb{E}[W_{t+1} \mid W_{t+1} \in \Pi^{(n)}(k)]$ below refers to the expectation of W_{t+1} conditioned on the event $W_{t+1} \in \Pi^{(n)}(k)$.

Theorem 2.3. *For $t = 0, \ldots, T-1$, let $(v_{t+1}^{(n)}(p,z))_{n\in\mathbb{N}}$ represent a ULCC sequence of functions in z for $p \in \mathbf{P}$ and assume this sequence converges to v_{t+1}^*. Choose sampling $(W_{t+1}^{(n)}(k))_{k=1}^n$ such that $W_{t+1}^{(n)}(k) = \mathbb{E}[W_{t+1} \mid W_{t+1} \in \Pi^{(n)}(k)]$ with $\rho_{t+1}^{(n)}(k) = \mathbb{P}(W_{t+1} \in \Pi^{(n)}(k))$ for $k = 1, \ldots, n$. For all $p \in \mathbf{P}$, $z \in \mathbf{Z}$, and $a \in \mathbf{A}$, it holds that*

$$\lim_{n\to\infty} \mathcal{K}_t^{a,(n)} v_{t+1}^{(n)}(p,z) = \mathcal{K}_t^a v_{t+1}^*(p,z)$$

and $(\mathcal{K}_t^{a,(n)} v_{t+1}^{(n)}(p,z))_{n\in\mathbb{N}}$ forms a ULCC sequence in z for all $a \in \mathbf{A}$ and $p \in \mathbf{P}$.

Proof. See Section 6 in Hinz (2014). □

The below proves uniform convergence on compact sets of the double modified value functions under the above disturbance sampling scheme. Let $(m_n)_{n\in\mathbb{N}}$ and $(n_m)_{m\in\mathbb{N}}$ represent sequences of natural numbers increasing in n and m, respectively.

Theorem 2.4. *Using the disturbance sampling in Theorem 2.3, it holds that*

$$\lim_{n\to\infty} v_t^{(m_n,n)}(p,z) = \lim_{m\to\infty} v_t^{(m,n_m)}(p,z) = v_t^*(p,z)$$

for $p \in \mathbf{P}, z \in \mathbf{Z}$ and $t = T-1, \ldots, 0$. Both $(v_t^{(m_n,n)}(p,z))_{n\in\mathbb{N}}$ and $(v_t^{(m,n_m)}(p,z))_{m\in\mathbb{N}}$ form ULCC sequences in z for all $p \in \mathbf{P}$ and $t = T-1, \ldots, 0$.

Proof. See Section 7 in Hinz (2014). □

2.4 Solution diagnostics

While Theorem 2.4 offers asymptotic guarantees, this may be of little practical use due to computational constraints. To address this, this section uses the duality methods developed by Rogers (2002); Haugh and Kogan (2004); Andersen and Broadie (2004); Rogers (2007); Chen and Glasserman (2007); Brown et al (2010) to gauge the distance to optimaility of our results as well as providing confidence intervals for the true value functions. To this goal, introduce random variables $\varphi_t(p,z,a)$ for $p \in \mathbf{P}$, $z \in \mathbf{Z}$, $a \in \mathbf{A}$, and $t = 1, \ldots, T$. Suppose further that these random

variables are zero mean, i.e. $\mathbb{E}[\varphi_t(p,z,a)] = 0$. Given $\varphi = (\varphi_t)_{t=1}^{T}$ and a policy $\pi = (\pi_t)_{t=0}^{T-1}$, introduce random functions $\bar{v}_t^{\varphi}$ and $\underline{v}_t^{\pi,\varphi}$ which are defined recursively for $t = T,\ldots,1$ by

$$\begin{aligned}
\bar{v}_T^{\varphi}(p,z) &= r_T(p,z) \\
\bar{v}_t^{\varphi}(p,z) &= \max_{a\in\mathbf{A}} \left(r_t(p,z,a) + \varphi_{t+1}(p,z,a) + \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^{a} \bar{v}_{t+1}^{\varphi}(p', W_{t+1}z) \right)
\end{aligned}$$

and

$$\begin{aligned}
\underline{v}_T^{\pi,\varphi}(p,z) &= r_T(p,z) \\
\underline{v}_t^{\pi,\varphi}(p,z) &= r_t(p,z,\pi_t(p,z)) + \varphi_{t+1}(p,z,\pi_t(p,z)) + \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^{\pi_t(p,z)} \underline{v}_{t+1}^{\pi,\varphi}(p', W_{t+1}z).
\end{aligned}$$

The following theorem establishes the lower bounding and upper bounding behaviour of the expectations of the above.

Theorem 2.5. *At $t = T-1,\ldots,0$,*

$$\mathbb{E}[\underline{v}_t^{\pi,\varphi}(p,z)] \leq v_t^*(p,z) \leq \mathbb{E}[\bar{v}_t^{\varphi}(p,z)]$$

for all $p\in\mathbf{P}$, $z\in\mathbf{Z}$, and any policy π. Moreover, if

$$\varphi_t^*(p,z,a) = \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^{a} \mathbb{E}[v_t^*(p', W_t z)] - v_t^*(p', W_t z)$$

then

$$\mathbb{E}[\underline{v}_t^{\pi^*,\varphi^*}(p,z)] = v_t^*(p,z) = \mathbb{E}[\bar{v}_t^{\varphi^*}(p,z)]$$

for all $p\in\mathbf{P}$, $z\in\mathbf{Z}$ and $t = T-1,\ldots,0$.

Proof. See Section 7 in Hinz and Yap (2015). □

It turns out that the closer that φ_t resembles φ_t^*, the tighter the bounds in Theorem 2.5. Unfortunately, the true value functions v_t^* are used to construct φ_t^* and they are unknown in practice since their knowledge vitiates the need to perform numerical work in the first place. However, the double modified value functions from the subgradient approximation can be used in their place instead. With this use, φ_t acts as a control variate for the lower bounds and the bounds allow us to guage the quality of our numerical work. Many authors (e.g. Haugh and Kogan (2004); Chen and Glasserman (2007)) have referred to the random variables φ_t as the *additive duals*. The following presents a stylized implementation of this technique.

Primal-dual estimation

1. Given value function approximations $(v_t)_{t=0}^T$ and prescribed policy $(\pi_t)_{t=0}^{T-1}$ from (2.2.2), implement control variables $(\varphi_t)_{t=1}^T$ as

$$\varphi_t(p,z,a) = \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^a \left(\frac{1}{I'} \sum_{i'=1}^{I'} v_t(p', W_t^{(i')} z) - v_t(p', W_t z) \right), \tag{2.4.1}$$

   for $p \in \mathbf{P}$, $a \in \mathbf{A}$, $z \in \mathbf{Z}$, where $(W_t^{(1)}, \ldots, W_t^{(I')}, W_t)_{t=1}^T$ are independent and $(W_t^{(1)}, \ldots, W_t^{(I')}, W_t)$ are identically distributed for each $t = 1, \ldots, T$.
2. Chose a number $I \in \mathbb{N}$ of Monte-Carlo trials and obtain for $i = 1, \ldots, I$ independent realizations $(W_t(\omega_i))_{t=1}^T$ of disturbances.
3. Starting at $z_0^i := z_0 \in \mathbb{R}^d$, define for $i = 1, \ldots, I$ trajectories $(z_t^i)_{t=0}^T$ recursively

$$z_{t+1}^i = W_{t+1}(\omega_i) z_t^i, \qquad t = 0, \ldots, T-1$$

   and determine realizations

$$\varphi_t(p, z_{t-1}^i, a)(\omega_i)$$

   for $t = 1, \ldots, T$ and $i = 1, \ldots, I$.
4. For each $i = 1, \ldots, I$ initialize the recursion at $t = T$ as

$$\underline{v}_T^{\pi,\varphi}(p, z_T^i)(\omega_i) = \overline{v}_T^{\pi}(p, z_T^i)(\omega_i) = r_T(p, z_T^i) \qquad \text{for all } p \in \mathbf{P}$$

   and continue for $t = T-1, \ldots, 0$ and $p \in \mathbf{P}$ with $\underline{v}_t^{\pi,\varphi}(p, z_t^i)(\omega_i)$ given by

$$r_t(p, z_t^i, \pi_t(p, z_t^i)) + \varphi_{t+1}(p, z_t^i, \pi_t(p, z_t^i))(\omega_i) + \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^{\pi_t(p,z_t^i)} \underline{v}_{t+1}^{\pi,\varphi}(p', z_{t+1}^i)(\omega_k)$$

   and $\overline{v}_t^{\varphi}(p, z_t^i)(\omega_k)$ given by

$$\max_{a\in\mathbf{A}} \left[r_t(p, z_t^k, a) + \varphi_{t+1}(p, z_t^i, a)(\omega_k) + \sum_{p'\in\mathbf{P}} \alpha_{p,p'}^a \overline{v}_{t+1}^{\varphi}(p', z_{t+1}^k)(\omega_i) \right].$$

5. Calculate sample means $\frac{1}{I}\sum_{k=1}^{I}\underline{v}_0^{\pi,\varphi}(p_0,z_0)(\omega_i)$ and $\frac{1}{I}\sum_{k=1}^{I}\overline{v}_0^{\varphi}(p_0,z_0)(\omega_i)$ to estimate the expectations $\mathbb{E}[\underline{v}_0^{\pi,\varphi}(p_0,z_0)]$, $\mathbb{E}[\overline{v}_0^{\varphi}(p_0,z_0)]$ along with their confidence intervals. These estimates will be referred to as the *primal* and *dual* values, respectively.

The above approach is expressed in a more compact form by Algorithm 2.4.1. Line 5 in Algorithm 2.4.1 computes the additive duals.

Algorithm 2.4.1: Solution Diagnostics

```
1  for i = 1,...,I do
2  |  z_0^i ← z_0
3  |  for t = 0,...,T−1 do
4  |  |  z_{t+1}^i ← W_{t+1}(ω_i) z_t^i
5  |  |  φ_{t+1}^i(p,a) ← Σ_{p'∈P} α_{p,p'}^a ( (1/I') Σ_{i'=1}^{I'} v_{t+1}(p', W_{t+1}^{(i')} z_t^i) − v_{t+1}(p', z_{t+1}^i) )
6  |  end
7  end
8  for i = 1,...,I do
9  |  for p ∈ P do
10 |  |  v̄_T^i(p) ← v̲_T^i(p) ← r_T(p, z_T^i)
11 |  end
12 |  for t = T−1,...,0 do
13 |  |  for p ∈ P do
14 |  |  |  v̄_t^i(p) ← max_{a∈A} [ r_t(p, z_t^i, a) + φ_{t+1}^i(p,a) + Σ_{p'∈P} α_{p,p'}^a v̄_{t+1}^i(p') ]
15 |  |  |  a_t^i ← π_t(p, z_t^i)
16 |  |  |  v̲_t^i(p) ← r_t(p, z_t^i, a_t^i) + φ_{t+1}^i(p, a_t^i) + Σ_{p'∈P} α_{p,p'}^{a_t^i} v̲_{t+1}^i(p')
17 |  |  end
18 |  end
19 end
20 Determine primal and dual estimators by (1/I) Σ_{k=1}^{I} v̄_0^i(p) and (1/I) Σ_{k=1}^{i} v̲_0^i(p).
```

Finally, let us remark on the somewhat related Lagrange type duality as covered in Bismut (1978). The key difference between the the duality considered above and the Lagrange type duality is that the latter requires the construction of stochastic differential equations. In this sense, the duality considered in this thesis is much simpler to implement.

2.5 Conclusion

This chapter describes the work done by Hinz (2014) and Hinz and Yap (2015) to present a unified approach towards Markov decision processes under certain convexity assumptions on the functions in the Bellman recursion. To approximate the value functions and obtain the cor-

responding prescribed policy, a subgradient approach is presented which exploits the tangents from the reward and scrap functions. This subgradient approach enjoys uniform convergence on compact sets. This approach is intuitive and can be implemented efficiently mostly using simple matrix additions and multiplications. Recent pathwise duality techniques are then employed to gauge the quality of our numerical solutions as well as provinding lower and upper bound estimates for the true unknown value. Finally, it is important to note that the subgradient approach can be easily modified to handle disturbances W_{t+1} that are action dependent i.e. W^a_{t+1} where $a \in \mathbf{A}$. However, this adds complications to the solution diagnositics since a reference measure (and the corresponding densities) will then be needed for the sample path generation. In addition, all applications considered in this thesis contain uncontrolled W_{t+1}. Therefore, for simplicity and consistency, we omit this W^a_{t+1} consideration.

It is also important to point out that a different type of disturbance sampling can be employed than the one proved in this chapter (local averages). In fact, our numerical experiments indicate that sampling obtained via Monte Carlo return good quality results. Also note that the Lipschitz assumption on the reward and scrap functions is not required for convergence and this has been addressed in Yee (Preprinta). Extension to infinite horizon contracting settings is also proven in Yee (Preprintb). In the next chapter, nearest neighbour algorithms are used to reduce the computational effort of both the subgradient and duality approach. The following chapter then demonstrates these methods on numerous types of real options with excellent results. In Chapter 5, the setting and numerical methods are then extended to problems containing a hidden Markov model. Finally, the software implementation of the presented subgradient and duality appraoch is described in Chapter 6.

References

Andersen L, Broadie M (2004) Primal-dual simulation algorithm for pricing multidimensional american options. Management Science 50(9):1222–1234

Bauerle N, Rieder U (2011) Markov Decision Processes with applications to finance. Springer, Heidelberg, DOI 10.1007/978-3-642-18324-9

Bismut J (1978) An introductory approach to duality in optimal stochastic control. SIAM Review 20(1):62–78, DOI 10.1137/1020004

Brown D, Smith J, Sun P (2010) Information relaxations and duality in stochastic dynamic programs. Operations Research 58(4):785–801

Chen N, Glasserman P (2007) Additive and multiplicative duals for american option pricing. Finance and Stochastics 11(2):153–179

Haugh M, Kogan L (2004) Pricing American options: A duality approach. Operations Research 52(2):258–270

Hinz J (2014) Optimal stochastic switching under convexity assumptions. SIAM Journal on Control and Optimization 52(1):164–188, DOI 10.1137/13091333X

Hinz J, Yap N (2015) Algorithms for optimal control of stochastic switching systems. Theory of Probability and its Applications 60(4):770–800

Rockafellar R (1970) Convex Analysis. Princeton landmarks in mathematics and physics, Princeton University Press

Rogers L (2002) Monte carlo valuation of american options. Mathematical Finance 12(3):271–286

Rogers L (2007) Pathwise stochastic optimal control. SIAM J Control Optimisation 46(3):1116–1132

Yee J (Preprinta) Convex function approximations for Markov decision processes. arXiv:171200970

Yee J (Preprintb) Value iteration for approximate dynamic programming under convexity. arXiv:180207243

Chapter 3

Nearest neighbours

3.1 Introduction

This chapter examines the use of *nearest neighbour* algorithms in reducing the computational effort of the numerical methods presented before. Nearest neighbour algorithms are popular in many real world applications such as pattern identification and object recognition (Berg et al (2005); Philbin et al (2007)). The literature devoted to this problem is well developed and recent work has found success in high dimensional settings (Muja and Lowe (2014)). The goal of these methods is to find, for every query point, the closest point (or closest number of points) from a set of reference points according to some user defined metric. For the subgradient approach in low dimensional state space settings, the use of nearest neighbours can reduce the effort from $O(m^2)$ to $O(m\log(m))$ for increasing number of grid points m. For the solution diagnostics, the effort can be reduced from $O(m)$ to $O(\log m)$. Let us explain how our contribution differs from that of Hinz and Yap (2015). Hinz and Yap (2015) considered the use of the 1 nearest neighbour while we will consider the use of the $N \geq 1$ nearest neighbours in the subgradient approach. Increasing the number of nearest neighbours gives more accurate results but with a small increase in computational times. In addition, nearest neighbour algorithms will be used to reduce the computational effort in the computation of the additive duals in the solution diagnostics. For the case where the driving random disturbance takes on only a finite (even very large) number of possible values, the computational saving is drastic. This use of nearest neighbours in the solution diagnostics approach is novel and has not been published before.

3.2 Subgradient approach

The following explores the intuition behind the use of nearest neighbours for the value function approximation. The majority of the computational time for Algorithm 2.2.1 is due to the calculation of the expected value functions on Line 6. Recall that the row-rearrangement operator used in Algorithm 2.2.1 determines the maximizing subgradient by comparing all tangents held by the grid points. The expected value function $\widetilde{V}_{t+1}^{(m,n)}(p)$ is determined by finding the maximum of these tangents of $V_{t+1}^{(m.n)}(p)$ on a disturbed grid. Figure 3.2.1 illustrates this. On the disturbed grid points $W_{t+1}g^1$, $W_{t+1}g^2$ and $W_{t+1}g^3$, the original row-rearrangement operator Υ returns the correct maximizing tangents s^2, s^2 and s^3, respectively.

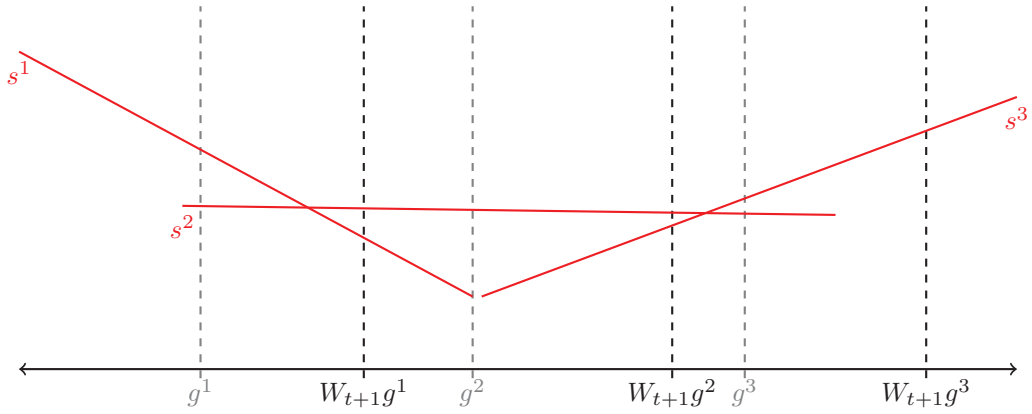

Fig. 3.2.1: Using nearest neigbours for expected value function estimation.

Now suppose we only consider the nearest neighbour. That is, for each disturbed grid point, we choose the tangent active at the point on the original grid closest to the disturbed point. In Figure 3.2.1, this would mean tangents s^2, s^3 and s^3 are returned for the disturbed points $W_{t+1}g^1$, $W_{t+1}g^2$ and $W_{t+1}g^3$, respectively. In this case, the choice for $W_{t+1}g^2$ is incorrect and so there is an approximatione error. However, if we maximise over the 2 nearest neighbours instead, this error disappears since the subgradient s^2 is higher than s^3 on $W_{t+1}g^2$. Note that increasing the number of nearest neighbours considered increases the computational cost and so there is a clear tradeoff between accuracy and computational time. It is also obvious that the density of the grid affects these approximation errors. More specifically, if the grid is sufficiently dense there are no such errors when we use nearest neighbour methods.

Let us now discuss the associated reduction in computational effort. Recall that n and m represents the size of the disturbance sampling and the number of grid points, respectively. To find the maximising tangents for every grid point in the original approach using the row-

rearrangement operator, one needs to compare all m possible tangents for every m grid points and for every n disturbances. Therefore, the effort is around $O(nm^2)$. Now to implement nearest neighbour searching, this thesis will use kd-trees due to their simplicity (see k-d tree (2018) for an overview). The construction of a kd-tree search index is often accepted as an $O(m \log m)$ operation. Searching this tree for the N nearest neighbors for all of the nm possible disturbed points is roughly an $O(Nnm \log m)$ operation. Now finding the maximising tangent amongst the N possible candidates for every possible disturbed point is $O(Nnm)$. Therefore, the approximate total effort is given by $O(Nnm \log m)$. Thus, the significant reduction of effort is from $O(nm^2)$ to $O(Nnm \log m)$ where N is the number of nearest neighbours considered for each disturbed point. Our experiments indicate that the searching component $O(Nnm \log m)$ makes up a vast majority of the time and so very significant time savings may be achieved using another nearest neighbour algorithm that addresses this. However, this is beyond the scope of this thesis and is left for future research. The use of nearest neighbours above lends well to code parrallelization since the nearest neighbours searching can performed efficiently on mutliple threads. Before proceeding, it is well known that kd-tree methods perform poorly in high dimensions and so other algorithms should be used in those settings.

3.3 Expectation matrices

Even with the aid of nearest neighbours, Algorithm 2.2.1 suffers from the repetitive matrix operations on Line 6 and this cost grows with the disturbance sampling size n. Namely,

1. the repetitive rearrangement $\Upsilon_{\mathbf{G}^{(m)}}[VW(k)]$ of large matrices $V \cdot W(k)$ and
2. the summation of $\Upsilon_{\mathbf{G}^{(m)}}[V \cdot W(k)]$ over a large index range $k = 1, \ldots, n$.

It turns out that one can approximate the row-rearrangment operation on Line 6 in Algorithm 2.2.1 with an appropriate matrix multiplication. More precisely, one can construct a matrix $Y(k)$ such that

$$\Upsilon_{\mathbf{G}^{(m)}}[VW(k)] \approx Y(k)VW(k) \tag{3.3.1}$$

for $k = 1, \ldots, n$. Given (3.3.1), the summands on Line 6 of Algorithm 2.2.1 can be approximated by

$$\Upsilon_{\mathbf{G}^{(n)}} V_{t+1}^{(m,n)}(p) W_{t+1}^{(n)}(k) \approx Y_{t+1}^{(n)}(k) V_{t+1}^{(m,n)}(p) W_{t+1}^{(n)}(k).$$

Now suppose each $W_{t+1}^{(n)}(k)$ can be represented by

$$W_{t+1}^{(n)}(k) = \overline{W}_{t+1} + \sum_{j=1}^{J} \epsilon_{t+1}^{j}(k) E_{t+1}(j), \qquad k = 1, \ldots, n.$$

with non-random matrices $\overline{W}_{t+1}$, $(E_{t+1}(j))_{j=1}^{J}$ and where $\epsilon_{t+1}^{j}(k)$ are realizations of random coefficients $(\varepsilon_{t+1}^{j})_{j=1}^{J}$. Utilizing this, one can approximate the conditional expectation on Line 6 in Algorithm 2.2.1 with

$$\left(\sum_{k=1}^{n} \rho_{t+1}^{(n)}(k) Y_{t+1}^{(n)}(k) \right) V_{t+1}^{(m,n)}(p) \overline{W}_{t+1} + \sum_{j=1}^{J} \left(\sum_{k=1}^{n} \rho_{t+1}^{(n)}(k) \epsilon_{t+1}^{j}(k) Y_{t+1}^{(n)}(k) \right) V_{t+1}^{(m,n)}(p) E_{t+1}(j). \tag{3.3.2}$$

If one pre-computes the following matrices

$$D_{t+1}^{(n)}(0) = \sum_{k=1}^{n} \rho_{t+1}^{(n)}(k) Y_{t+1}^{(n)}(k), \quad D_{t+1}^{(n)}(j) = \sum_{k=1}^{n} \rho_{t+1}^{(n)}(k) \epsilon_{t+1}^{j}(k) Y_{t+1}^{(n)}(k) \tag{3.3.3}$$

for $j = 1, \ldots, J$, a significant simplification to (3.3.2) can be obtained as

$$D_{t+1}^{(n)}(0) V_{t+1}^{(m,n)}(p) \overline{W}_{t+1} + \sum_{j=1}^{J} D_{t+1}^{(n)}(j) V_{t+1}^{(m,n)}(p) E_{t+1}(j).$$

Now the final remaining question is an appropriate choice of matrices $Y_{t+1}^{(n)}(k)$ which we will refer to as *expectation matrices* since they approximate the conditional expectation operator in the Bellman recursion. The use of nearest neighbours discussed before reveals many possible choices. The following choice of $Y_{t+1}^{(n)}(k)$ was proposed by Hinz and Yap (2015). Let $Y_{t+1}^{(n)}(k)$ be a matrix of zeros except for entries defined as follows:

- For all $i = 1, \ldots, m$, define $g^*(i)$ to be grid index of the nearest grid point to $W_{t+1}^{(n)}(k) g^i$. Set the $(i, g^*(i))$-th entry of $Y_{t+1}^{(n)}(k)$ to be one.

The choice of $Y_{t+1}^{(n)}(k)$ above is esssential the same as using the tangent held by the nearest point on the grid for each disturbed grid point as discussed in the previous section. Other choices of $Y_{t+1}^{(n)}(k)$ include assigning weights to the different tangents held by the N nearest points instead. The impact of different choices of $Y_{t+1}^{(n)}(k)$ is beyond the scope of this thesis and is left for future research. As a final note, the precalculations of matrices in (3.3.3) are computationally demanding and so a gain in computational performance can only be achieved if not too many of these matrices are generated. For example, if disturbances $(W_t)_{t=1}^{T}$ are identically distributed

accross time, the precalculations need only be done once and so there is a drastic reduction in overall effort.

3.4 Study: Bermudan put

Optimal switching problems are common in financial markets. A simple example is given by the Bermudan put option. This option gives its owner the right but not the obligation to chose a time to exercise the option in order to receive a payment which depends on the price of the underlying asset at the exercise time. The so-called fair price of the Bermudan option is related to the solution of an optimal stopping problem (see Glasserman (2003)). Here, the asset price process $(Z_t)_{t=0}^T$ at time steps $0,\dots,T$ is modelled as a sampled geometric Brownian motion

$$Z_{t+1} = W_{t+1} Z_t, \qquad t = 0,\dots,T-1, \quad Z_0 \in \mathbb{R}_+,$$

where $(W_t)_{t=1}^T$ are independent random variables following a log-normal distribution. The fair price of such option with strike price K, interest rate $\rho \geq 0$ and maturity date T, is given by the solution to the optimal stopping problem

$$\sup\{\mathbb{E}[e^{-\rho\tau}(K - Z_\tau)^+] : \ \tau \text{ is } \{0,1,\dots,T\}\text{-valued stopping time}\}$$

where $(z)^+ := \max\{z, 0\}$.

This switching system is defined by two positions $\mathbf{P} = \{1,2\}$ and two actions $\mathbf{A} = \{1,2\}$. Here, the positions 'exercised' and 'not exercised' are represented by $p = 1$, $p = 2$ respectively, and the actions 'don't exercise' and 'exercise' are denoted by $a = 1$ and $a = 2$ respectively. With this interpretation, the position change is given by deterministic transitions to specified states

$$\alpha^a_{p,p'} = \begin{cases} 1 \text{ if } p' = \alpha(p,a) \\ 0 \text{ else} \end{cases}$$

deterministically determined by the target positions

$$(\alpha(p,a))_{p,a=1}^2 \sim \begin{bmatrix} \alpha(1,1)\ \alpha(1,2) \\ \alpha(2,1)\ \alpha(2,2) \end{bmatrix} = \begin{bmatrix} 1\ 1 \\ 2\ 1 \end{bmatrix},$$

while the rewards at time $t = 0,\dots,T$ and scrap are defined as

$$r_t(p,z,a) = e^{-\rho t}(K-z)^+(p-\alpha(p,a)),$$
$$r_T(p,z) = e^{-\rho T}(K-z)^+(p-\alpha(p,1)),$$

for all $p \in \mathbf{P}$, $a \in \mathbf{A}$, $z \in \mathbb{R}_+$.

The following numerical experiments were run using the author's *R* package (Hinz and Yee, 2017) on a Linux Ubuntu 16.04 machine with Intel i5-5300U CPU @2.30GHz and 16GB of RAM. As a demonstration, let us consider a Bermudan put that has strike price 40, expires in 1 year, and resides in the Black Scholes universe. The put option is on a non-divendend stock and is exercisable at 51 evenly spaced time points in the year, which includes the start and end of the year. The interest rate and the volatility is set at 0.06 p.a. and 0.2 p.a., respectively. We use $m = 301$ equally spaced points between $z = 30$ and $z = 60$ to construct the grid. For the disturbance sampling, we partition $\mathbf{W} = \mathbb{R}_+$ into $n = 1000$ components with equal probability measure and use the local averages on each component.

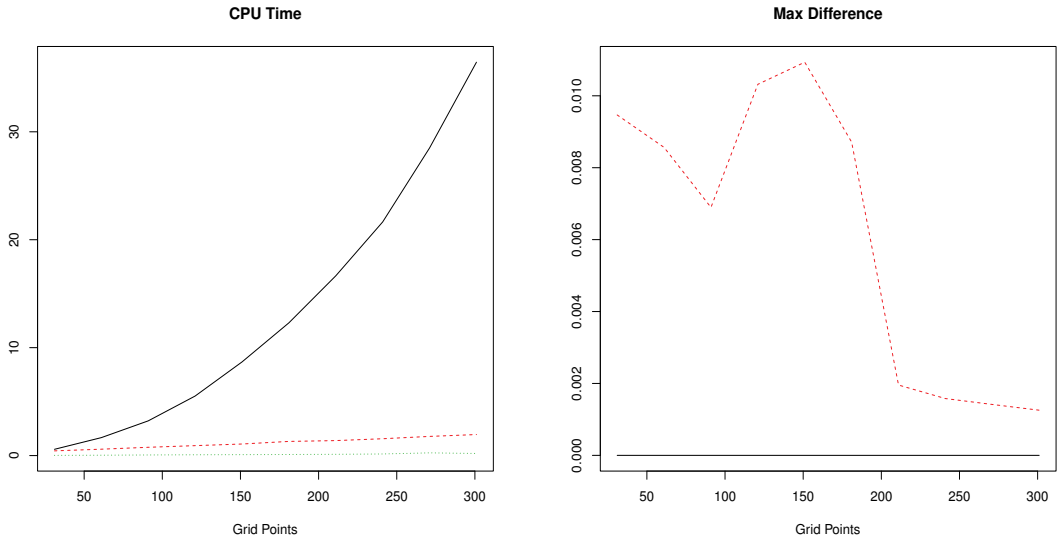

Fig. 3.4.1: The left plot gives a comparison of the cpu times while the right gives the maximum absolute difference between the value function approximations.

In Figure 3.4.1, the left plot compares the computational times for Algorithm 2.2.1 (in black), maximising over 2 nearest neighbours (in red), and the expectation matrices using the 1 nearest neighbour as proposed by Hinz and Yap (2015) (in green) as a function of grid density between $z = 30$ and $z = 60$. A drastic reduction in the times compared to the original approach is clear. A similar observation was made when the size of the disturbance sampling is varied. The right plot gives the maximum difference between the value function approximation from Algorithm

2.2.1 and that from maximising between 2 nearest neighbours (in black), and the maximum difference between Algorithm 2.2.1 with that of expectation matrices using 1 nearest neighbour (in red). The right plot reveals that the approximation error induced by the use of expectation matrices decreases with the density of the grid. The plots below also reveal that the use of 2 nearest neighbours induces no approximation error in the example while only being slightly slower than the use of expectation matrices. For high dimensional state spaces where the grid points are likely to be sparse, the use of $N > 1$ nearest neighbours will likely return superior approximations than that from expectation matrices. Therefore, there is practical value in considering $N > 1$ nearest neighbours instead of $N = 1$ nearest neighbour as proposed by Hinz and Yap (2015).

3.5 Additive duals

The vast majority of the computational time for the solution diagnostics is taken up by the calculation of the additive duals. The following extends the same nearest neighbour approach to significantly reduce the computational effort for the additive duals. This is novel and has not been published before. The main performance hurdle in computing the additive duals in the solution diagnostics is that it involves determining the maximising subgradient for each scenario in the nested simulation at each sample path. Let us make this point clear. The subgradient approach returns value function approximations in the form of matrices. To compute the additive duals presented by (2.4.1) requires finding the maximising matrix row (representing a tangent) which in turn requires a comparison between all rows. For each sample path at each time point, this requires $O(I'm)$ effort where I' is the number of nested simulations and m is the number of grid points. However, one can approximate (2.4.1) using nearest neighbours in a similar vein to Section 3.2.

First define $N_{\mathbf{G}^{(m)}}(g)$ to be the grid point in $\mathbf{G}^{(m)}$ closest to query point g according to some user defined metric. Now instead of (2.4.1) for the additive duals, one can use:

$$\varphi_t(p,z,a) = \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \frac{1}{I'} \sum_{i'=1}^{I'} v^{(m,n)}_{t+1}(p', N_{\mathbf{G}^{(m)}}(W^{(i')}_t z)) - v^{(m,n)}_{t+1}(p', N_{\mathbf{G}^{(m)}}(W_t z)). \tag{3.5.1}$$

In the above, the tangent on the nearest grid point is selected and this amounts to using the corresponding row in the value function matrix instead of comparing all rows. Assuming the kd-tree search index has already been constructed in the subgradient approach, (3.5.1) leads

to roughly an $O(I' \log m)$ effort. The bounds obtained using (3.5.1) are still valid since (3.5.1) is zero mean and should be very similar to (2.4.1) for sufficiently dense grid. This will be demonstrated in the next section.

Now suppose that the disturbance sampling contains all possible realizations of disturbances W_{t+1}. A drastic reduction in computational effort can be further obtained by using the continuation value function approximations. However, it is important to keep in mind that the method used to approximate the continuation value function should be consistent with the manner in which the maximising tangent is approximated for the additive duals. For example, if Algorithm 2.2.1 was used, then

$$\varphi_t(p,z,a) = \mathcal{K}^{(a,n)} v_{t+1}^{(m,n)}(p, N_{\mathbf{G}^{(m)}}(z)) - \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a v_{t+1}^{(m,n)}(p', W_t N_{\mathbf{G}^{(m)}}(z)) \tag{3.5.2}$$

should be used. If the N nearest neighbours approach from Section 3.2 was used to construct the expected value functions, then

$$\varphi_t(p,z,a) = \mathcal{K}^{(a,n)} v_{t+1}^{(m,n)}(p, N_{\mathbf{G}^{(m)}}(z)) - \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \max_{g \in W_t N_{\mathbf{G}^{(m)}}(z,N)} v_{t+1}^{(m,n)}(p', g) \tag{3.5.3}$$

should be used where $N_{\mathbf{G}^{(m)}}(g, N)$ gives the N nearest grid points to g and $W_t N_{\mathbf{G}^{(m)}}(z, N)$ refers to the disturbed N nearest grid points after applying W_t to each of the nearest neighbours. In the above, the manner in which the first term is constructed should coincide with the access method for the second term to ensure that the above are zero mean and yield valid bound estimates. The effort for (3.5.2) and (3.5.3) is roughly $O(m)$ and $O(N \log m)$, respectively, if the kd-tree search index was used in the subgradient approach. They do not depend on I' i.e. there is no nested simulations involved in the calculation of the additive duals and so a drastic amount of computational effort can be saved in this case.

3.6 Computational times

The following reuses the Bermudan put option example from before. Table 3.6.1 below indicates that the use of either (2.4.1) and (3.5.1) give similar tight bounds and standard errors for different grid specifications. The number of paths I and nested simulations I' are both set to be 500 each. The table below shows that the results become identical for both (2.4.1) and (3.5.1) on a sufficiently dense grid. This coincides with our argument above. We now turn our attention

to their relative computational times of all the additive duals. The left plot of Figure 3.6.1 illustrates the computational times for the different methods at different grid densities. The solid black lines depict the times for the original method while the red dashed lines represent the new method using nearest neighbours. The times for (2.4.1) are clearly increasing in the number of grid points. However, the effect of the grid size m for (3.5.1) is much less severe. In this sense, (3.5.1) seems to be better suited than (2.4.1) for problems requiring large grids.

Table 3.6.1: Bound estimates for $Z_0 = 36$ using the different additive dual formulations.

	Equation (2.4.1)		Equation (3.5.1)	
Grid	Lower	Upper	Lower	Upper
91	4.4776 (.0008)	4.4787 (.0008)	4.4775 (.0008)	4.4785 (.0009)
181	4.4777 (.0008)	4.4786 (.0008)	4.4778 (.0008)	4.4787 (.0008)
271	4.4777 (.0008)	4.4785 (.0008)	4.4777 (.0008)	4.4785 (.0008)
361	4.4777 (.0008)	4.4785 (.0008)	4.4777 (.0008)	4.4785 (.0008)

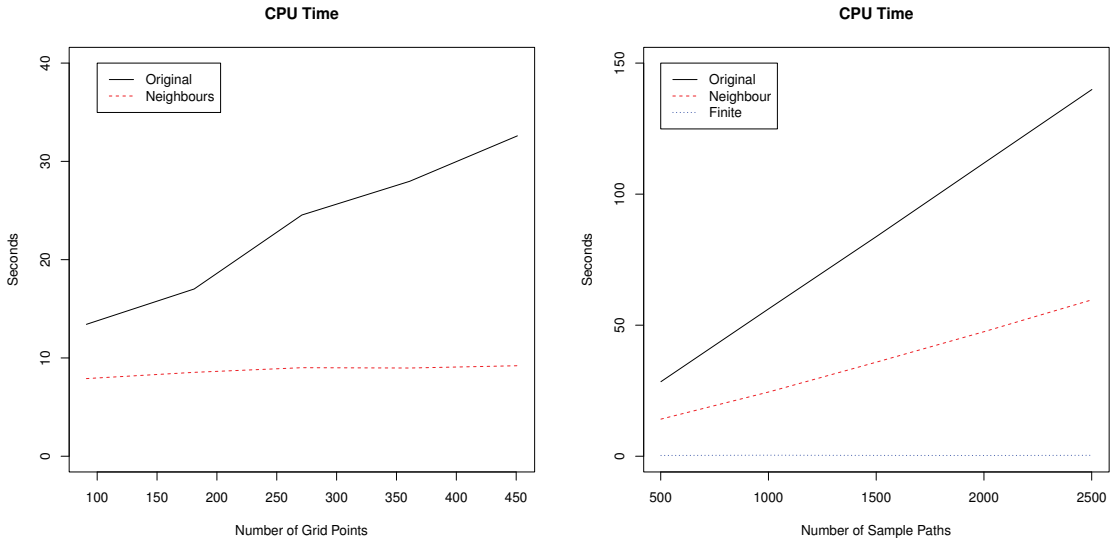

Fig. 3.6.1: Additive duals computation times as a function of grid density under the true distribution (left plot) and as a function of the number of sample paths under the finite distribution (right plot).

The right plot in Figure 3.6.1 examines the finite distribution case where the disturbance W_{t+1} takes only $n = 100000$ possible values. The different curves give the computational times for the additive duals as a function of the number of simulated paths I using the different Equations (2.4.1), (3.5.1) and (3.5.3) where $I' = 1000$ and $N = 1$. The drastic reduction in effort is clear for the finite distribution case when (3.5.3) is used. This has significant practical implications. If one is willing to replace the orginal distribution with a sufficiently dense finite

distribution, then computational times can be reduced dramatically since the nested simulation is removed from the calculation of the additive duals.

3.7 Conclusion

This chapter demonstrates that the use of nearest neighbour algorithms in the subgradient and duality approach can significantly reduce the computational effort with little loss in accuracy. Note that kd-trees were used in this chapter for simplicity and it would be interesting to see the impact of other types of nearest neighbour methods. This is left for future research. In the next chapter, the nearest neighbour approach will be applied to a wide range of real options applications. The expectation matrices approach will be used to approximate the value functions and (3.5.1) will be employed to compute the additive duals for the lower and upper bound estimates.

References

Berg A, Berg T, Malik J (2005) Shape matching and object recognition using low distortion correspondences. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol 1, pp 26–33

Glasserman P (2003) Monte Carlo Methods in Financial Engineering. Springer

Hinz J, Yap N (2015) Algorithms for optimal control of stochastic switching systems. Theory of Probability and its Applications 60(4):770–800

Hinz J, Yee J (2017) rcss: Convex switching systems. URL `https://github.com/YeeJeremy/rcss`, available at https://github.com/YeeJeremy/rcss, R package version 1.5

k-d tree (2018) k-d tree — Wikipedia, the free encyclopedia. URL `https://en.wikipedia.org/wiki/K-d_tree`, [Online; accessed 30-January-2018]

Muja M, Lowe D (2014) Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence 36(11):2227–2240

Philbin J, Chum O, Isard M, Sivic J, Zisserman A (2007) Object retrieval with large vocabularies and fast spatial matching. In: 2007 IEEE Conference on Computer Vision and Pattern Recognition, pp 1–8

Chapter 4

Optimal Switching and Real options

4.1 Introduction

The main goal of this chapter is to demonstrate the subgradient and duality approach on a wide range of optimal switching problems and real options. The term optimal switching is used due to the discrete controlled component in the state process. While computational times are provided, please keep in mind they were obtained using software (Hinz and Yee, 2017) aimed at providing users with a simple interface to solve a standardized problem formulation. Therefore, faster times can be attained if, for example, the computational effort resides entirely in $C++$ and the code is specialised to the problem. The following numerical experiments were run using the author's R package (Hinz and Yee, 2017) on a Linux Ubuntu 16.04 machine with Intel i5-5300U CPU @2.30GHz and 16GB of RAM. All numerical results presented here are reproducible using R scripts posted online at `https://github.com/YeeJeremy/scripts/tree/master/thesis`. The code is multi-threaded and computational times listed in cpu seconds with the real world elapsed time given in paranthesis.

4.2 Option pricing

Let us begin by revisiting the Bermuda put example considered in the previous chapter. This example was also considered by Longstaff and Schwartz (2001) in which they used least squares Monte Carlo to generate their results. Recall that the put option has strike price 40, expires in 1 year, and resides in the Black Scholes universe. The put option is on a non-divendend stock and is exercisable at 51 evenly spaced time points in the year, which includes the start and end of the year. The interest rate and the volatility is set at 0.06 p.a. and 0.2 p.a., respectively. We use $m = 301$ equally spaced points between $z = 30$ and $z = 60$ to construct the grid. For the

disturbance sampling, we partition $\mathbf{W} = \mathbb{R}_+$ into $n = 1000$ components with equal probability measure and use the conditional averages on each component. For the primal-dual values, we use $I' = 1000$ nested simulations and $I = 1000$ paths. It takes roughly 0.2 cpu seconds (0.05 elapsed seconds) to compute the value function approximation using the expectation matrices approach. It takes roughly 25 cpu seconds (10 elapsed seconds) to calculate the primal and dual values for each z_0, with most of the time taken up by the calculation of φ_t control variates. Table 4.2.1 compares the results from the subgradient approach in column 3 with those from Longstaff and Schwartz (2001) listed in Column 2. The fourth and fifth columns give the sample mean of the primal and dual values, respectively. Observe the tight bound estimates and their low standard errors. The similarity of the values contained in Columns $3-5$ indicate good quality value function approximations. Finally, Column 6 gives the values obtained by solving the associated partial differential equations numerically from Longstaff and Schwartz (2001).

Table 4.2.1: Comparision of least squares Monte Carlo (from Longstaff and Schwartz (2001)) and the subgradient approach.

z_0	LSM	Subgradient	Primal	Dua	Finite Differences
36	4.472 (0.010)	4.47689	4.47850 (0.00038)	4.47875 (0.00039)	4.478
38	3.244 (0.009)	3.24898	3.25057 (0.00043)	3.25092 (0.00044)	3.250
40	2.313 (0.009)	2.31287	2.31364 (0.00045)	2.31397 (0.00044)	2.314
42	1.617 (0.007)	1.61582	1.61659 (0.00045)	1.61675 (0.00045)	1.617
44	1.118 (0.007)	1.10874	1.10956 (0.00040)	1.10986 (0.00041)	1.110

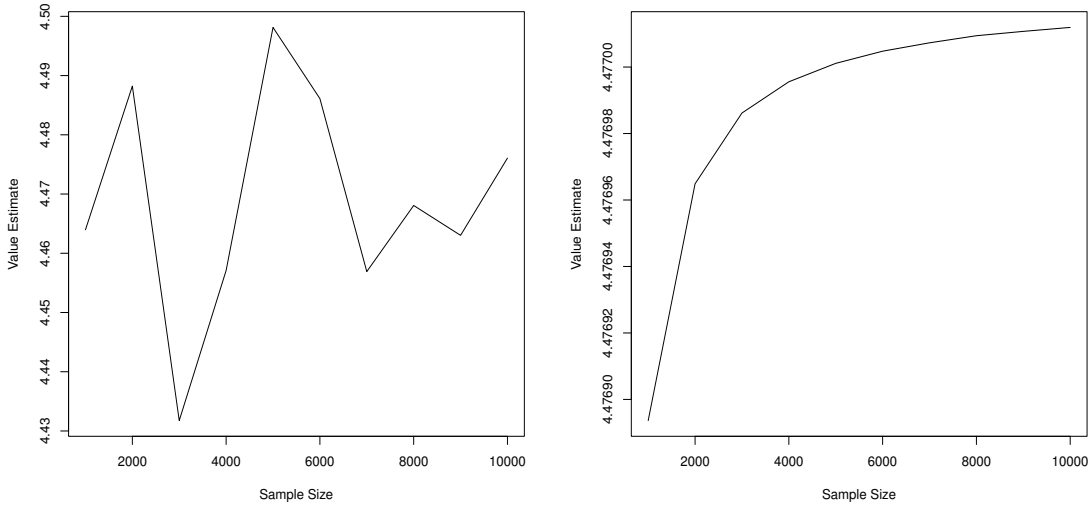

Fig. 4.2.1: Different sampling shcemes: Monte Carlo (left) and discretization (right).

Figure 4.2.1 compares the value of the option with $z_0 = 36$ under two different sampling schemes (Monte Carlo vs. local averages) as the size of n is increased. It is clear that one should avoid the use of Monte Carlo to obtain better results. However, this may not be possible in high dimensional settings. In such cases, Monte Carlo sampling also lead to good results as demonstrated later when multiple stochastic factors are considered in Section 4.2.2.

4.2.1 Swing option

Now consider the swing option which gives the owner the right but not the obligation to obtain a certain commodity (such as gas) at a contracted price and volume at a number of exercise times. For simplicity, attention is restricted to the *unit-time refraction period* swing option where there is a limit to exercise only one right at any time. Given the discounted commodity price $(S_t)_{t=0}^T$, the fair price of the swing option with N rights is given by

$$\sup_{0\le\tau_1<\cdots<\tau_N\le T} \mathbb{E}\Big[\sum_{n=1}^{N}(S_{\tau_n} - Ke^{-\rho\tau_n})^+\Big]$$

over all stopping times $\tau_1,\ldots,\tau_N$ with values in $\{0,\ldots,T\}$. Use the position set $\mathbf{P} = \{1,\ldots,N+1\}$ to describe the number of exercise rights remaining where $p \in \mathbf{P}$ stands for the situation when there are $p-1$ rights remaining. The action set $\mathbf{A} = \{1,2\}$ gives the action to exercise $(a = 1)$ or not to exercise $(a = 2)$. The transition probabilities $(\alpha^a_{p,p'})$ are given by

$$\alpha^1_{p,p'} = \begin{cases} 1 \text{ if } p' = 1 \vee (p-1) \\ 0 \text{ else,} \end{cases} \quad \text{and} \quad \alpha^2_{p,p'} = \begin{cases} 1 \text{ if } p' = p \\ 0 \text{ else} \end{cases}$$

for all $p,p' \in \mathbf{P}$. The deterministic control of the discrete component is easier to describe in terms of the matrix $(\alpha(p,a))_{p\in\mathbf{P},a\in\mathbf{A}}$ where $p' = \alpha(p,a) \in \mathbf{P}$ stands for the discrete component which is reached from $p \in \mathbf{P}$ by the action $a \in \mathbf{A}$. For the case of the swing option this matrix is

$$(\alpha(p,a))_{p\in\mathbf{P},a\in\mathbf{A}} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 2 & 3 \\ \vdots & \vdots \\ N & N+1 \end{bmatrix}.$$

Having modelled the discounted commodity price process as an exponential mean-reverting process with a reversion parameter $\kappa \in [0,1[$, long run mean $\mu > 0$ and volatility $\sigma > 0$, the logarithm of the discounted price process is obtained as

$$\tilde{Z}_{t+1} = (1-\kappa)(\tilde{Z}_t - \mu) + \mu + \sigma\epsilon_{t+1}, \quad \tilde{Z}_0 = \ln(S_0).$$

To attain linear state dynamics, introduce an augmentation with 1 via

$$Z_t = \begin{bmatrix} 1 \\ \tilde{Z}_t \end{bmatrix}, \qquad t = 0, \ldots, T.$$

then it becomes possible to represent the evolution as the linear state dynamics

$$Z_{t+1} = W_{t+1} Z_t, \qquad t = 0, \ldots, T-1$$

with independent and identically distributed random matrices $(W_t)_{t=1}^T$ given by

$$W_{t+1} = \begin{bmatrix} 1 & 0 \\ \kappa\mu + \sigma\epsilon_{t+1} & (1-\kappa) \end{bmatrix}, \quad t = 0, ..., T-1.$$

and the reward for $t = 0, \ldots, T-1$ and scrap values by

$$r_t(p, (z^{(1)}, z^{(2)}), a) = (e^{z^{(2)}} - Ke^{-\rho t})^+ (p - \alpha(p, a))$$
$$r_T(p, (z^{(1)}, z^{(2)}), a) = (e^{z^{(2)}} - Ke^{-\rho T})^+ (p - \alpha(p, 1))$$

respectively for all $p \in \mathbf{P}$ and $a \in \mathbf{A}$. Note that while the reward and scrap functions are convex in z for each $p \in \mathbf{P}$, they are not globally Lipshitz in $z^{(2)}$.

Table 4.2.2 considers the swing option example from Meinshausen and Hambly (2004) starting from $Z_0^{(2)} = 0$ with $N = 100$ rights, $\rho = 0$, $\sigma = 0.5$, $\kappa = 0.9$, $\mu = 0$, $K = 0$ and $T = 1000$. The grid was constructed from 101 equally spaced grid points between $z^{(2)} = -2$ and $z^{(2)} = 2$. A disturbace sampling was constructed using local averages on a $n = 1000$ component partition of $\mathbf{W}$. The expectation matrices approach was used for the value function approximation and takes roughly 30 cpu seconds (20 elapsed seconds) to generate for all positions combined. The primal and dual values were obtained using $I = 100$ paths and $I' = 100$ nested simulations. This process takes around 60 cpu seconds (40 elapsed seconds) for all positions combined. In

Table 4.2.2: Comparision of the subgradient approach and the regression approach from Meinshausen and Hambly (2004).

N Rights	Subgradient	Primal-Dual 99% CI	MH 99% CI
1	4.734	(4.749, 4.817)	(4.773, 4.794)
2	9.000	(9.013, 9.104)	(9.016, 9.091)
3	12.995	(13.006, 13.112)	(12.959, 13.100)
4	16.800	(16.816, 16.933)	(16.773, 16.906)
5	20.461	(20.480, 20.605)	(20.439, 20.580)
10	37.332	(37.341, 37.514)	(37.305, 37.540)
15	52.684	(52.692, 52.899)	(52.670, 53.009)
20	67.056	(67.068, 67.296)	(67.050, 67.525)
30	93.791	(93.815, 94.079)	(93.662, 94.519)
40	118.610	(118.621, 118.923)	(118.353, 119.625)
50	142.029	(142.033, 142.367)	(141.703, 143.360)
60	164.334	(164.336, 164.700)	(163.960, 166.037)
70	185.719	(185.717, 186.099)	(185.335, 187.729)
80	206.319	(206.319, 206.719)	(205.844, 208.702)
90	226.232	(226.247, 226.664)	(225.676, 228.985)
100	245.543	(245.565, 245.999)	(244.910, 248.651)

aggregate, around 1.5 cpu minutes (1 elapsed minute) was required to compute the first three columns in Table 4.2.2. Column 3 in Table 4.2.2 give the 99% confidence intervals for the true value calculated using the primal-dual values and are compared to those from Meinshausen and Hambly (2004) listed in Column 4 in which they use a regression approach. We obtain similar results in a short amount of time. The similarity between the values in Column 2 and 3 indicate good quality function approximations atleast around Z_0. These value function approximations in terms of S_0 are depicted in Figure 4.2.2.

Let us make the following note before proceeding. As shown in Section 2.4, the confidence intervals in the above table represent confidence intervals for the true unknown values and not for the subgradient estimates (from the value function approximation). Therefore, it should not be surprising if the estimate is not within the 99% confidence intervals above.

4.2.2 Bermudan max-call

The previous examples contains only one stochastic factor. Let us return to the Black Scholes universe and consider a Bermudan max-call option on multiple underlying stocks. In this setting, the curse of dimensionality has a bigger impact since each stock essentially represents a stochastic factor. We will see shortly that our approach still performs remarkable well under a relatively high number of stochastic factors. Suppose that each of the N dividend paying stock

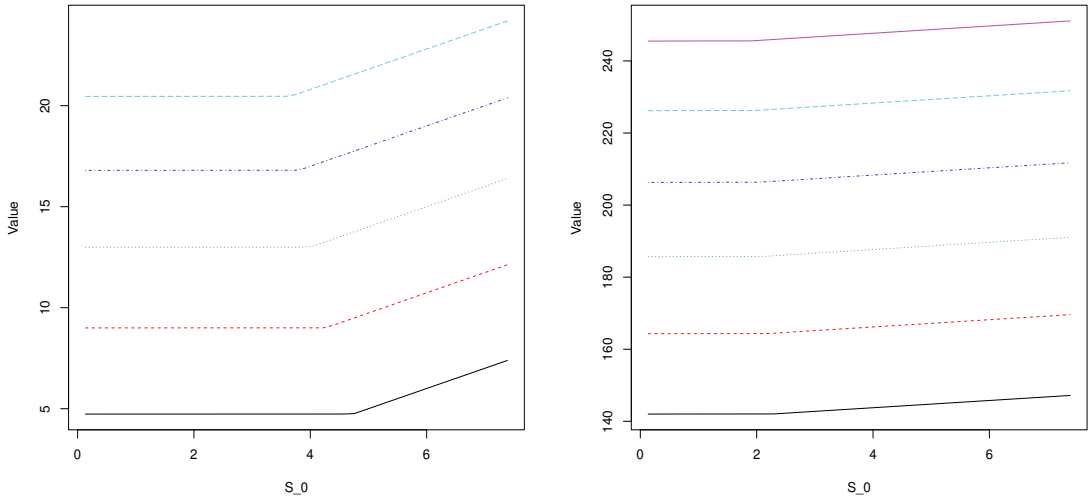

Fig. 4.2.2: Value function approximations. Left plot contain rights $N = 1, 2, 3, 4, 5$ and the right plot gives $N = 60, 70, 80, 90, 100$.

price $Z_t^{(i)}$ for $i = 1, \ldots, N$ evloves as geometric Brownian motion

$$Z_{t+1}^{(i)} = \epsilon_{t+1}^{(i)} Z_t^{(i)}, \quad t = 0, \ldots, T-1$$

where $\{\epsilon_{t+1}^{(i)}\}_{i=1}^N$ are correlated log-normal random variables. With this, linear state dynamics can be induced by

$$Z_{t+1} = \begin{bmatrix} Z_{t+1}^{(1)} \\ \vdots \\ Z_{t+1}^{(N)} \end{bmatrix} = \begin{bmatrix} \epsilon_{t+1}^{(1)} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \epsilon_{t+1}^{(N)} \end{bmatrix} Z_t = W_{t+1} Z_t$$

As before in the Bermuda put case, this problem is defined by two positions $\mathbf{P} = \{1, 2\}$ and two actions $\mathbf{A} = \{1, 2\}$ with the same interpretation i.e. $p = 1$ refers to 'exercised' and $a = 2$ refers to 'exercise the option'. The transition probabilities are given by

$$\alpha_{p,p'}^a = \begin{cases} 1 \text{ if } p' = \alpha(p, a) \\ 0 \text{ else} \end{cases}$$

deterministically determined by the target positions

$$(\alpha(p, a))_{p,a=1}^2 \sim \begin{bmatrix} \alpha(1,1)\ \alpha(1,2) \\ \alpha(2,1)\ \alpha(2,2) \end{bmatrix} = \begin{bmatrix} 1\ 1 \\ 2\ 1 \end{bmatrix}.$$

Now the reward at time $t = 0, \ldots, T-1$ and scrap are defined as

$$r_t(p, (z^{(1)}, \ldots, z^{(N)}), a) = e^{-\rho t}(\max\{z^{(1)}, \ldots, z^{(N)}\} - K)^+(p - \alpha(p, a)),$$
$$r_T(p, (z^{(1)}, \ldots, z^{(N)})) = e^{-\rho T}(\max\{z^{(1)}, \ldots, z^{(N)}\} - K)^+(p - \alpha(p, 1))$$

for all $p \in \mathbf{P}$, $a \in \mathbf{A}$, $z^{(1)}, \ldots, z^{(N)} \in \mathbb{R}_+$ where ρ is the interest rate and K the strike price. Therefore, the fair price of such option is given by the solution to the optimal stopping problem

$$\sup\{\mathbb{E}[e^{-\rho\tau}(\max\{Z_\tau^{(1)}, \cdots, Z_\tau^{(N)}\} - K)^+] : \ \tau \text{ is } \{0, 1, \ldots, T\}\text{-valued stopping time}\}.$$

Note that since $\mathbf{Z}$ is a $N-$dimensional, it is impractical to use a grid containing equally spaced points for the subgradient approach. In this case, one could use generate a so-called stochastic grid obtained by simulating a number of paths and applying an algorithm such as k-means on all states visited to obtain an appropriate grid. This approach is suitable for problems with large N. Another issue is that the discretization of W_{t+1} also becomes impractical in high dimensions and so Monte Carlo sampling becomes the only real choice. Fortunately, both of these approaches performs well as demonstrated by the following results.

Table 4.2.3: Comparision between the subgradient approach and regression approach used by Andersen and Broadie (2004).

N	$Z_0^{(1)}$	Subgradient	Primal-Dual 95% CI	AB 95% CI
2	90	8.015	(8.052, 8.086)	(8.053, 8.082)
	100	13.843	(13.877, 13.915)	(13.892, 13.934)
	110	21.257	(21.300, 21.371)	(21.316, 21.359)
3	90	10.794	(11.233, 11.334)	(11.265, 11.308)
	100	18.174	(18.616, 18.923)	(18.661, 18.728)
	110	26.879	(27.444, 27.871)	(27.512, 27.663)
5	90	12.938	(15.661, 17.067)	(16.602, 16.655)
	100	22.413	(25.142, 26.773)	(26.109, 26.292)
	110	32.302	(35.429, 37.749)	(36.704, 36.832)

Let us consider the Bermuda max-call example studied by Andersen and Broadie (2004). Here, the option expires in 3 years and is exercisable at 10 equally spaced time points (including the start). The stike price is given by $K = 100$, the interest rate by 5% per annum, and assume the stock prices are uncorrelated. The stock has dividend yield of 10% per annum and the volatility of the stock price id 20% per annum. Assume further that $Z_0^{(1)} = \cdots = Z_0^{(N)}$ for simplicity. The results in 4.2.3 compares the results from the the subgradient approach with those published by Andersen and Broadie (2004) in Column 5. For the $N = 2, 3$ case, a stochastic

grid containing 1000 points generated using k-means clustering on 2000 simulated paths was used. A disturbance sampling containing 20000 values were generated via Monte Carlo sampling. For the primal and dual values, $I = 1000$ paths and $I' = 1000$ nested simulations were used. For $N = 2$, the value function approximation takes around 13 cpu seconds (4 elapsed seconds) and around 8 cpu seconds (4 elapsed seconds) to compute the primal-dual values for each starting $Z_0^{(1)}$. For $N = 3$, the corresponding times are around 40 cpu seconds (16 elapsed) and 12 cpu seconds (5 elapsed), respectively. For the $N = 5$ case, a larger stochastic grid of 200 points and a disturbance sampling of size $n = 40000$ via Monte Carlo was used to account for the higher state space dimension. In addition, we also set $I = 1500$ and $I' = 1500$. The computational times for value function approximation and bound estimation is around 220 cpu seconds (65 elapsed) and 70 cpu seconds (22 elapsed), respectively. We obtain good agreement with the results presented in Andersen and Broadie (2004) which was obtained using least squares Monte Carlo. Note that our bounds in Column 4 can be easily improved by increasing grid density m, disturbance sample n or the primal-dual parameters I and I'.

While similar results to Andersen and Broadie (2004) were obtained, our approach is much easier to implement in practice. More specifically, selecting a suitable regression basis for least squares Monte Carlo in high dimensional settings is a notoriously difficult problem. For example, Andersen and Broadie (2004) used:

"a set of 12 functions, consisting of the largest and the second largest asset prices, three polynomials of degree two (e.g., the squares of each asset price and the product of the two), four polynomials of degree three, the value of a European max-call option on the largest two assets, and the square and the cube of this value (and a constant term is also included)"

for their regression basis. In contrast, our approach is essentially automated using Monte Carlo for both grid and disturbance sampling selection. This difference may have significant implications for practical use.

4.3 Portfolio liquidation

Having demonstrated and comparing the subgradient and duality approach with least squares Monte Carlo using classical problems in option pricing, let us now turn to optimal liquidation problems. Optimal asset liquidation is a decision problem of selling a (large) amount of

assets within a given time horizon under minimal costs. With the increased penetration of algorithmic trading, diverse algorithms for optimal liquidation have attracted attention from the financial industry and academia. Their research has focused on the determination of an optimal trade-off between risk and profitability while diminishing the negative price impact of order placement on the asset price by an optimal split into multiple small orders. A crucial aspect of this optimization is the proper choice of order type (limit versus market orders) and an appropriate distribution of orders over the entire time horizon. In this context, the asset liquidation problem is naturally viewed as finding an optimal mixture of active and passive trading. Here, passive trading is about placement of the so-called *limit orders* which provide liquidity and are launched onto the limit order book waiting for an agent to fill them, whereas active trading focuses on submitting *market orders* which consume liquidity and are executed immediately against the opposite limit orders. Over the recent years, versatile variations of the optimal asset liquidation problem have attracted strong research interest and numerous theoretical and practical results have been attained under diverse simplifying assumptions. However, in practice, the optimal liquidation problem still represents a stochastic control problem which is too complex for a comprehensive theoretical analysis and too large for the existing numerical methods. Asset liquidation has been addressed by numerous publications, starting from considerations on execution costs minimization (Almgren and Chriss (2000); Bertsimas and Lo (1998); Obizhaeva and Wang (2013)) and market liquidity and response (Alfonsi and Schied (2010); Alfonsi et al (2010); Bayraktar and Ludkovski (2011)). Diverse aspects of optimal order placement are discussed in Cont et al (2014); Horst and Naujokat (2014); Becherer et al (2016); Kratz and Schoneborn (2016), while the work Cont and Kukanov (2014) addresses the optimal mixture of passive and active strategies, similar to our approach.

4.3.1 Setting

Assume that an electronic broker attempts to sell a number of tradable asset units within a fixed time interval. That is, the only available action is to submit sell orders. However, the time of order submission, the order size, and the order type must be chosen dynamically to achieve an optimal goal. To formulate this problem in our framework, we assume that at each time $t = 0, \dots, T-1$ the trader knows the number $p \in \mathbb{N}$ of asset units held in their own portfolio, observes the bid and the ask prices and decides to submit a number of assets units for sale, either as a market order or as a limit order. Thereby, it is supposed that the limit sell order is always

placed at the ask price and is valid for one time step only. The underlying randomness in the model comes from both the price and the spread evolution. However, due to market efficiency, the price direction can not be predicted with reasonable accuracy, unlike the bid-ask spread. For this reason, our model does not count a potential win/loss from the price, focusing instead on the effect caused by the spread, when deciding whether to place a limit or a market order. While the market sell order is executed with a high probability at the current bid price, the amount liquidated through a limit order is uncertain at the time of order submission. However, selling through limit orders may achieve a higher price. The realized price difference between market and limit orders can be approximately described by the recent bid-ask spread.

Let us now formulate our asset liquidation as *stochastic switching with linear state dynamics* as explained above. Suppose that the discrete component $p \in \mathbf{P}$ describes the number of asset units held in the portfolio and suppose that $a_{\max} \in \mathbb{N}$ stands for the maximum order size. Introduce the action set as

$$\mathbf{A} = \big(\{0,\dots,a_{\max}\} \times \{1\}\big) \cup \big(\{0,\dots,a_{\max}\} \times \{2\}\big), \tag{4.3.1}$$

its elements are typically $a' = (a'_1, 1)$, $a'' = (a''_1, 2)$ which stands for the limit and market order of size $a'_1, a''_1 \in \{0,\dots,a_{\max}\}$, respectively. In this work, we assume for simplicity that the trader has to decide at each time whether to place an order or not and if the order is placed, whether it will be a limit or a market order. The possibility to place both order types at the same time can be modelled in a straight-forward way by introducing appropriate action type for simultaneous placement. However, this is not done here.

Given a market order $a'' = (a''_1, 2)$, the number of asset units withdrawn from the portfolio is given by $a''_1 \wedge p$. That is, we have

$$\alpha_{p,p'}^{(a''_1,2)} = F_2^{p,a''_1}(p-p') - F_2^{p,a''_1}((p-p')-1), \qquad p' \in P,$$

where the F_2^{p,a''_1} is given by the distribution function of the Dirac measure placed at $a''_1 \wedge p$. In the case of a limit order $(a'_1, 1)$, the number of sold asset units is random and follows a distribution on the set $\{0,\dots a'_1 \wedge p\}$. We define

$$\alpha_{p,p'}^{(a'_1,1)} = F_1^{p,a'_1}(p-p') - F_1^{p,a'_1}((p-p')-1), \qquad p' \in P,$$

with

$$F_1^{p,a_1'} = F\mathbb{1}_{]-\infty,p\wedge a_1'[} + \mathbb{1}_{[p\wedge a_1',\infty[},$$

where F is the exogenously pre-specified distribution function of the number of asset units requested through buy market orders within one time step.

Having introduced the controlled dynamics of the discrete component, let us now turn to the autonomous evolution $Z_{t+1} = W_{t+1}Z_t$ of the continuous component that describes the fluctuation of the bid-ask spread around its mean $\mu \in \mathbb{R}$. For the sake of concreteness, we assume that the spread fluctuation follows a univariate auto-regressive model with auto-regression coefficient $\phi \in]0,1[$, driven by a white noise of variance $\sigma \in]0,\infty[$. This process can be represented by the second component $(Z_t^{(2)})_{t\in\mathbb{N}}$ of the linear state space process $(Z_t)_{t\in\mathbb{N}}$ defined by the recursion

$$\underbrace{\begin{bmatrix} Z_{t+1}^{(1)} \\ Z_{t+1}^{(2)} \end{bmatrix}}_{Z_{t+1}} = \underbrace{\begin{bmatrix} 1 & 0 \\ \sigma\epsilon_{t+1} & \phi \end{bmatrix}}_{W_{t+1}} \underbrace{\begin{bmatrix} Z_t^{(1)} \\ Z_t^{(2)} \end{bmatrix}}_{Z_t}, \qquad \begin{bmatrix} Z_0^{(1)} \\ Z_0^{(2)} \end{bmatrix} = \begin{bmatrix} 1 \\ z_0 \end{bmatrix} \tag{4.3.2}$$

where $(\epsilon_t)_{t=1}^{\infty}$ is a sequence of independent and identically distributed random variables. With this choice, the parameter μ does not enter the auto-regression (4.3.2) since this process describes the deviation of the spread from its mean μ, meaning that the spread evolution is given by $(\mu + Z_t)_{t=0}^{T}$. Modelling spread fluctuation around its mean in terms of auto-regression represents a simplification, which we have chosen for illustrative purposes.

Finally, let us turn to the definition of the liquidation costs by introducing the scrap value and the reward functions. Given at time $t = 0,\dots,T-1$ the position size p and the spread state $z = (1, z^{(2)})$, consider the costs of an action $a = (a_1, a_2) \in \mathbf{A}$. Recall that according to the definition (4.3.1) the first entry $a_1 \in \{0,\dots,a_{\max}\}$ of a stands for the order size whereas the second entry $a_2 \in \{1,2\}$ labels the order type, with $a_2 = 1$ representing the limit and $a_2 = 2$ standing for the market order. Defining the scrap and reward functions for $t = 0,\dots,T-1$ by

$$\begin{aligned} r_t(p,z,a) = r_t(p,z,(a_1,a_2)) &= -g_t(b_t - p) - (\mu + z^{(2)})(a_2 - 1) \\ r_T(p,z) &= -g_T(b_T - p), \end{aligned}$$

the term $-(\mu + z^{(2)})(a_2 - 1)$ describes the loss from crossing the spread when placing a market order $a_2 = 2$ (note that this term equals to zero for the limit order $a_2 = 1$). Furthermore, $g_t(b_t - p)$ stands for a penalty on the deviation $b_t - p$ at time $t = 0,\dots,T$ of the current long unliquidated position p from a pre-determined level $b_t \in \mathbb{R}$ targeted at times $t = 0,\dots,T$. We refer to b_t as the *benchmark* at time t. In this setting, the exogenously given functions $(g_t)_{t=0}^{T}$

determine the precision at which the trader shall follow the pre-determined benchmark $b_t \in \mathbb{R}$ for $t = 0, \dots, T$. The above reward functions are convex and globally Lipshitz continuous in z, allowing us to use the subgradient approach presented in the previous chapter.

4.3.2 Case study

To generate illustrative results we assume that the maximal order size is given by $a_{\max} = 3$, the initial number of asset units $p_0 = 50$ and the time horizon $T = 50$. We define the parameters $\phi = 0.9$, $\sigma = 0.5$, and $\mu = 1$. Furthermore, we consider a benchmark $b_t = p_0 - t$ for $t = 0, \dots, T$ and examine the asset liquidation for two different penalty functions: the point target $g_t(u) = \gamma_t |u|$ and the range target $g_t(u) = \gamma_t 1_{]1,\infty[}(|u|)|u|$ for $u \in \mathbb{N}$ and $t = 0, \dots, T$ with three different penalty specifications for γ_t as outlined in Table 4.3.1. For the market and the limit orders, we induce the following probabilities:

$$\text{Limit order:} \quad \alpha^{(a_1,1)}_{p,(p-a_1)\vee 0} = \begin{cases} 0.3 \text{ if } a_1 = 1; \\ 0.2 \text{ if } a_1 = 2; \\ 0.1 \text{ if } a_1 = 3; \end{cases} \text{ and } \alpha^{(a_1,1)}_{p,p} = 1 - \alpha^{(a_1,1)}_{p,(p-a_1)\vee 0}.$$

$$\text{Market order:} \quad \alpha^{(a_1,2)}_{p,(p-a_1)\vee 0} = \begin{cases} 1 \quad \text{if } a_1 = 1; \\ 0.9 \text{ if } a_1 = 2; \\ 0.8 \text{ if } a_1 = 3; \end{cases} \text{ and } \alpha^{(a_1,2)}_{p,p} = 1 - \alpha^{(a_1,2)}_{p,(p-a_1)\vee 0}.$$

Note that in this case study we have generalized the transitional probabilities for the market orders as compared to the previous subsection. For market orders, orders of size greater than one are no longer cleared with certainty.

Table 4.3.1: Point target benchmark vs. range benchmark.

		Point benchmark		Range benchmark	
γ_t	$Z_0^{(2)}$	Subgradient	99% CI	Subgradient	99% CI
1	-1	-27.926	(-27.923, -27.871)	-11.392	(-11.397, -11.345)
	0	-35.653	(-35.653, -35.599)	-16.904	(-16.911, -16.862)
	1	-41.446	(-41.446, -41.394)	-21.134	(-21.141, -21.096)
$\frac{1}{50}t$	-1	-20.169	(-20.171, -20.115)	-9.739	(-9.745, -9.691)
	0	-24.989	(-24.994, -24.942)	-14.505	(-14.514, -14.466)
	1	-28.840	(-28.844, -28.795)	-18.286	(-18.294, -18.250)
$\frac{1}{20}t$	-1	-30.050	(-30.049, -30.001)	-12.498	(-12.502, -12.451)
	0	-35.656	(-35.656, -35.610)	-17.499	(-17.504, -17.459)
	1	-40.245	(-40.244, -40.201)	-21.544	(-21.551, -21.508)

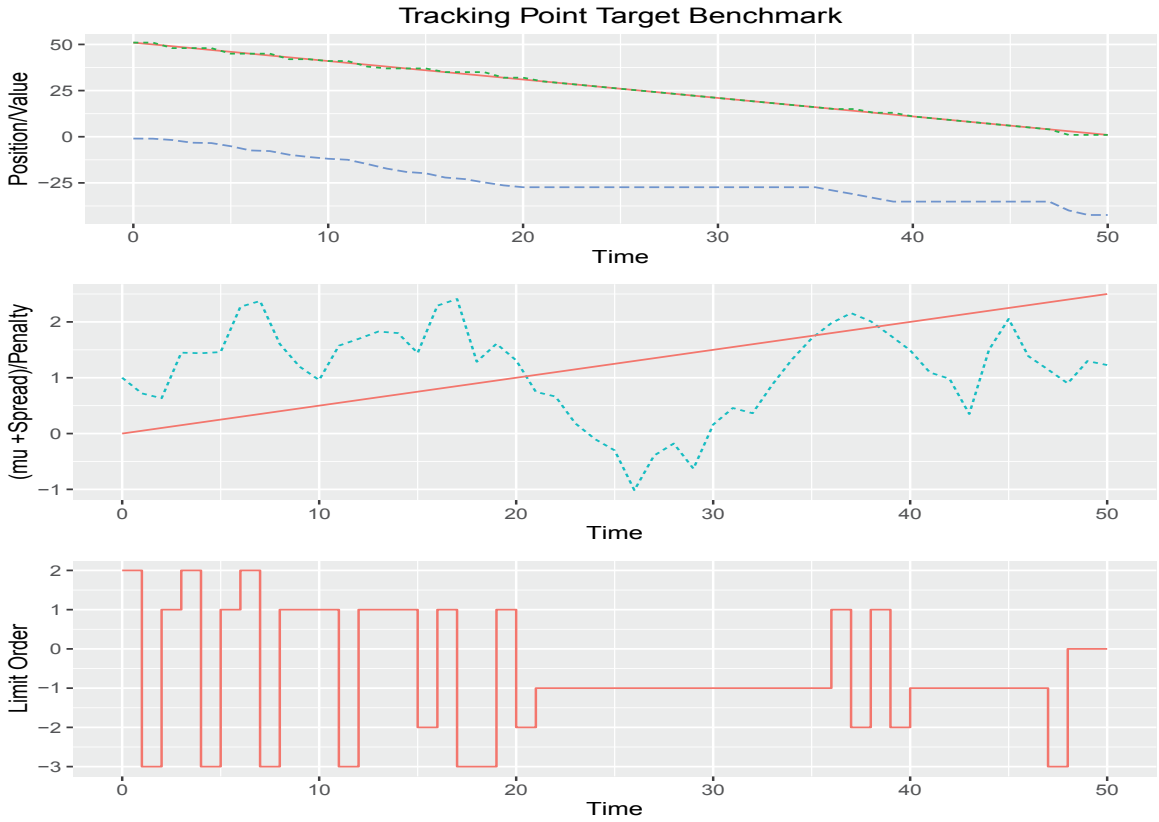

Fig. 4.3.1: Point target benchmark for a sample path (seed = 123).

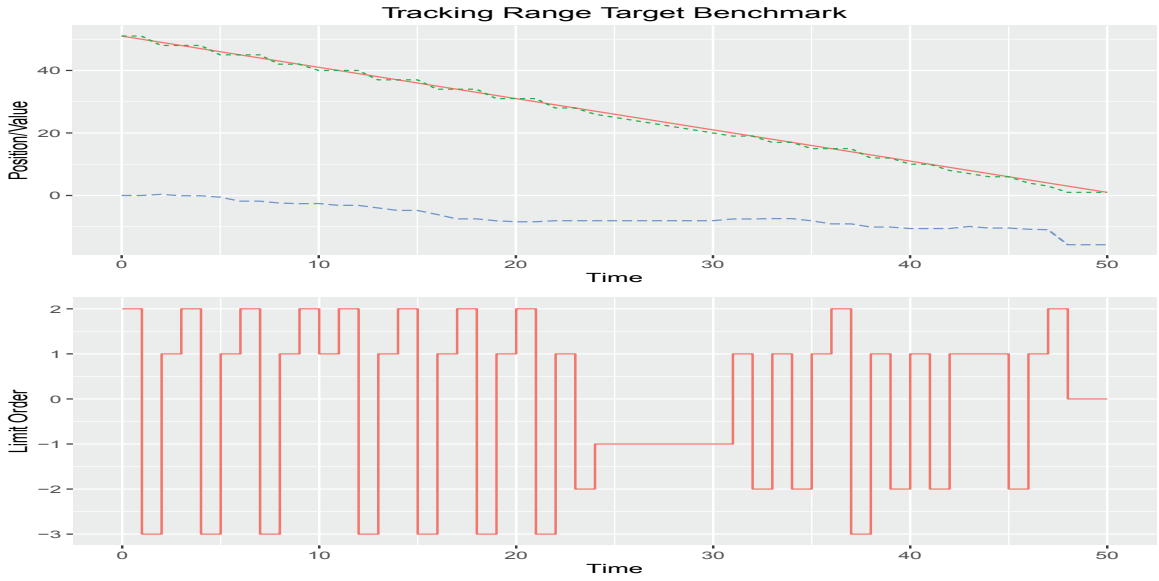

Fig. 4.3.2: Range target benchmark for a sample path (seed = 123).

Table 4.3.1 gives the value estimates from the subgradient approach and the 99% confidence intervals formed using the primal-dual values and their standard errors. To approximate the value functions, we use an equally spaced grid containing 201 points from $z^{(2)} = -5$ and $z^{(2)} = 5$ and a disturbance sampling generated by partitioning $\mathbf{W}$ into 1000 components of equal probability measure. For each γ_t, it takes around 2 cpu seconds (1 elapsed) to obtain the function approximations. To obtain the primal and dual values, $I = 200$ sample paths and $I' = 200$ nested simulations were used. The resulting confidence intervals take around 6 cpu seconds (3 elapsed) to compute for each satrting $Z_0^{(2)}$. The similarity of the subgradient value estimates and the 99% confidence intervals suggest that the value function approximations are of good quality atleast around $Z_0^{(2)}$. Figure 4.3.1 illustrates a test run of the candidate policy for a particular sample path. In the upper plot, the overall asset position (dotted green line), the benchmark (solid red line) and the total accumulated penalties (lower dashed blue line) are given. The middle graph shows the evolution of the price spread in a cyan dotted line and the penalty parameter $(\gamma_t)_t$ in solid red. The bottom shows the actions taken. Here we depict the limit orders by positive numbers and the market orders by negative, with the absolute value representing the order size. Figure 4.3.2 shows the same information (excluding the middle plot) for the benchmark range target on the same sample path.

This section addresses optimal asset liquidation as stochastic switching problems with linear state dynamics. We demonstrate that our methodology yields numerically efficient and precise solutions to stochastic control problems arising in this setting. Further refinements and generalizations to more complex problems of asset liquidation type are possible and can be treated under this framework.

4.4 Battery control

The recent proliferation of renewable energies and technological progress in electric battery storage systems create an increasing demand for sound algorithmic solutions to optimal control problems arising in the dispatch optimization of power supply given a storage facility and uncertainty caused by the market prices or/and weather conditions. Such problems are numerically challenging due to high dimensionality of the state spaces involved. This work suggests a quantitative approach to address a growing need for efficient decentralized electricity dispatch and storage. Let us describe its typical framework. The traditional electricity market players satisfy consumers' energy demand by purchasing electricity in advance, usually taking positions in the

so-called day-ahead market (also called the spot market) such that any energy imbalances must be compensated in real-time as they occur. This real-time balancing can either be achieved through complex over-the-counter trading or, more realistically, by transferring supply from or to electricity grid at the so-called real time grid prices. Figure 4.4.1 provides a simplified illustration of this optimal control problem. However, in the presence of storage and renewable generation facilities, the problem changes. On this new structure, the agent's control problem now requires simultaneously taking optimal positions and setting optimum energy storage levels as shown in Figure 4.4.2. The decision optimization problem becomes significantly more complex due to the uncertainty stemming from the future battery capacity levels, electricity prices, and output of renewable energy.

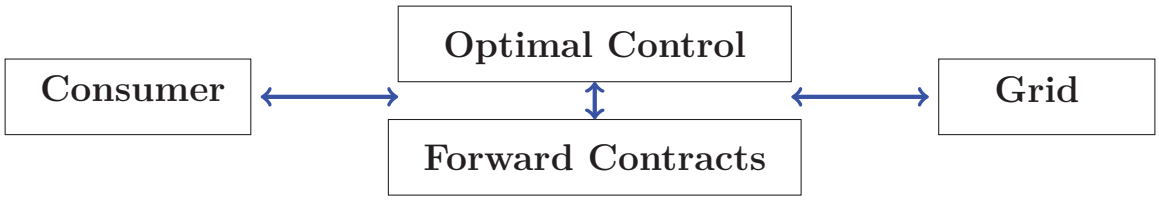

Fig. 4.4.1: Traditional energy dispatch.

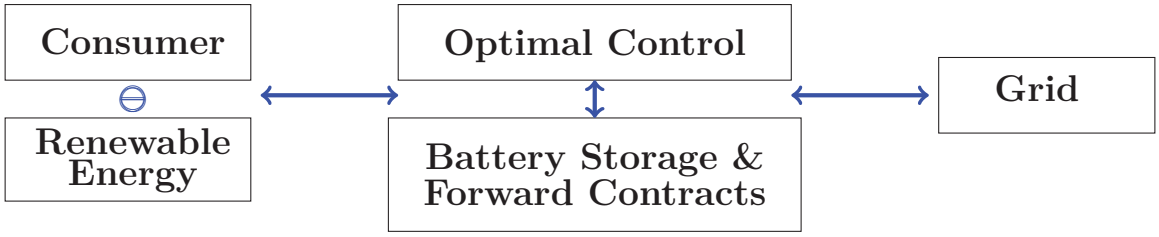

Fig. 4.4.2: Energy dispatch in the presence of renewable energy and battery storage.

Many renewable energy sources such as wind and solar are notoriously intermittent and unreliable. The potential of energy storage devices to address the highly erratic nature of renewable energy generation (Breton and Moe (2009); Dincer (2011)) and energy demand has been discussed extensively in the literature (see Beaudin et al (2010); Díaz-González et al (2012); Evans et al (2012); Kempener and Borden (2015); Yang et al (2014)). Their incorporation into a modern energy grid will encourage more environmentally friendly policies which will also have significant impact on investor atittudes towards firms (Ramiah et al (2013); Chan and Walter (2014); Renneboog et al (2008)). The authors of Lu et al (2014) studied the possible usage of battery storage systems to defer costly modifications to the energy grid by addressing peak loads

in the power grid. An extensive recent review of available energy storage technologies has been given by Luo et al (2015) and future innovation looks bright. While there exist numerous types of energy storage systems, Beaudin et al (2010) found that no single storage system consistently outperforms all the others for all types of renewable energy sources and applications. So for the sake of simplicity, this chapter will assume that the energy retailer pictured in Figure 4.4.2 uses a battery device for storing energy. However, the methods and results contained within this chapter can easily be extended for other types of storage technologies or even to the use of multiple types of storage devices. From a real options analysis point of view, the incorporation of energy storage devices into energy grid also poses interesting investment questions. The work done by Bakke et al (2016); Bradbury et al (2014); Locatelli et al (2016) examined the profitability of investing in energy storage devices. However, Schachter and Mancarella (2016) questions the suitability of the current real options approach, stating that the risk neutrality assumption may not be appropriate for risk averse investors. The introduction of batteries also gives rise to important optimal stochastic control problems. The optimal dynamic storage and discharge of energy from battery devices has been examined in Dokuchaev (2016); Kim and Powell (2011); Oudalov et al (2007); Teleke et al (2010).

Rather than focusing on capacity investment decision, the present contribution focuses on optimal operational management in terms of energy purchase and dispatch optimization, given a storage device of a fixed capacity. Thereby, we suppose that storage facility is only used for the compensation of any imbalance between consumers' demand, renewable energy generation and an existing financial position. This issue is connected to the market behaviour addressed in Cartea and Villaplana (2008) in terms of reducing the risk in the sense of Benth et al (2008), since a storage acts as a safety buffer. This section helps to investigate the effect of battery on forward energy trading.

4.4.1 Model

Within a given time horizon $t = 0, \ldots, T-1$, the net energy demand Q_t within each period is the difference between the consumer's demand and the renewable energy output. Given an existing financial position F_t, the energy imbalance $F_t - Q_t$ will be compensated using energy from the battery storage followed by a possible offset through real-time energy from the power grid. That is, in the case of energy surplus $F_t - Q_t \geq 0$, the electricity is first used to charge the battery up to the maximal level and the remaining energy is then supplied to the grid.

Similarly, if there is an energy shortage $F_t - Q_t < 0$, the required electricity is taken from the battery up to a minimal battery level before the required electricity rest is taken from the grid.

Let us assume that the net demand realization is given by $Q_t = q_t + \varepsilon_t$ with zero-mean random variable ε_t describing the deviation of the net demand from its predicted level q_t and suppose that the financial position is given in terms of $F_t = q_t + l$ where the quantity l describes a safety margin and stands retailer's decision to buy/sell in the spot market an energy amount $q_t + l$ which deviates from the predicted net demand q_t by l. Thereby, we model the decision of the retailer in the choice $F_t = F_t(a)$ of financial positions in terms of the action $a \in \mathbf{A}$ from a finite set $\mathbf{A}$ of all possible actions, each characterized by its specific safety margin $l(a)$. With the assumptions above, given the action $a \in \mathbf{A}$, the realized net energy to be balanced is given by

$$F_t(a) - Q_t = q_t + l(a) - (q_t + \varepsilon_t) = l(a) - \varepsilon_t, \quad a \in \mathbf{A}.$$

That is, the action $a \in \mathbf{A}$ determines a certain distribution $\nu_t(a)$ of the energy volume which must be balanced and is determined as $\nu_t(a) \sim l(a) - \varepsilon_t$ for all $a \in \mathbf{A}$. In order to describe the battery storage control, we suggest discretizing the storage levels by a finite set $\mathbf{P}$. Having chosen the action $a \in \mathbf{A}$, the imbalance energy $F_t(a) - Q_t = l(a) - \varepsilon_t$ follows a distribution $\nu_t(a)$ which determines for each battery storage level $p \in \mathbf{P}$ a probability $\alpha^a_{p,p'}$ that the storage reaches its next-day level $p' \in \mathbf{P}$. Furthermore, the expected energy excess $\underline{e}^a_p$ and shortage $\overline{e}^a_p$ are uniquely determined by the current battery level $p \in \mathbf{P}$ and the action $a \in \mathbf{A}$ through the imbalance distribution $\nu_t(a)$.

Now let us turn to the costs of energy imbalance. For this, we introduce the random variables

$$0 \le \underline{\Pi}_t \le \overline{\Pi}_t, \qquad t = 0, \dots, T$$

which stand for the sell/buy real time grid prices expected at time t, when the financial position $F_t(a)$ is taken. With these definitions, the revenue/costs associated with energy imbalance for the action $a \in \mathbf{A}$ are modeled by $-\overline{e}^a_p \overline{\Pi}_t + \underline{e}^a_p \underline{\Pi}_t$. Finally, let us denote by Π_t the energy price at time $t = 0, \dots, T$. Since we assume that all feasible financial positions are given as $q_t + l(a)$ with $a \in \mathbf{A}$, the position costs for the action $a \in \mathbf{A}$ are

$$(q_t + l(a))\Pi_t = q_t \Pi_t + l(a)\Pi_t.$$

With assumptions and notations as above, the revenue/loss associated with the action $a \in \mathbf{A}$ depends on the current price Π_t, the expected demand q_t, and the recent battery level $p \in \mathbf{P}$ as

$$r_t(p,(q_t,\Pi_t),a) = -q_t\Pi_t - l(a)\Pi_t - \overline{e}_p^a\overline{\Pi}_t + \underline{e}_p^a\underline{\Pi}_t.$$

Observe that the term $-q_t\Pi_t$ neither depends on the action a nor on battery level p. Thus, we agree that the choice $a \in \mathbf{A}$ of the optimal safety margin $l(a)$ will depend only on electricity price and re-define the reward as

$$r_t(p,\Pi_t,a) = -l(a)\Pi_t - \overline{e}_p^a\overline{\Pi}_t + \underline{e}_p^a\underline{\Pi}_t. \tag{4.4.1}$$

Note that we also do not consider other revenues associated with income streams due to delivery commitments at fixed price. On this account, the revenue (4.4.1) serves a vehicle to optimize trading activity in terms of optimal safety margins and does not reflect actual cash flows.

The revenue optimization from battery storage management is a typical sequential decision problem under uncertainty. Having chosen at time $t = 0,\dots,T-1$ an action $a \in \mathbf{A}$ in the data (p,Π_t) a certain revenue/costs $r_t(p,\Pi_t,a)$ is incurred immediately. However, the action $a \in \mathbf{A}$ also changes the probability of transition to the subsequent states (next battery levels) which influences all future revenues and decisions.

4.4.2 Case study

As a demonstration, let us consider a model based on the auto-regressive state dynamics. To cover this process under our framework, we introduce $Z_t = [Z_t^{(1)}, Z_t^{(2)}]^T = [1, Z_t^{(2)}]^T$ where the first component equals to one for $t = 0,\dots,T$ and define the linear state dynamics

$$Z_{t+1} = \begin{bmatrix} 1 \\ Z_{t+1}^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \mu + \sigma N_{t+1} & \phi \end{bmatrix} \begin{bmatrix} 1 \\ Z_t^{(2)} \end{bmatrix} = W_{t+1}Z_t, \quad t = 0,\dots,T-1. \tag{4.4.2}$$

with constants $\mu \in \mathbb{R}$, $\sigma \in \mathbb{R}_+$ and $\phi \in [0,1]$, driven by independent standard normally distributed random variables $(N_t)_{t=1}^T$. Further, we assume that the electricity price $(\Pi_t)_{t=0}^T$ is governed by the function $f : \mathbb{N}_+ \times \mathbb{R} \mapsto \mathbb{R}$ applied to the state process as

$$\Pi_t = f(t, Z_t^{(2)}), \qquad t = 0,\dots,T.$$

In this work, we restrict ourselves to consider deterministic affine linear functions $(f(t,\cdot))_{t=0}^{T}$ to appropriately describe any seasonal pattern of the electricity price, frequently observed in practice.

To model the the consumer's energy demand $Q_t = q_t + \varepsilon_t$ realized at time t, we suppose that conditioned on the information at time t, the deviation ε_t of the realized demand from its predicted value q_t follows a centered normal distribution with a given variance $\varsigma^2 \in \mathbb{R}_+$. To describe the evolution of the battery storage levels, let us assume that a finite set $\mathbf{P}$ describes the storage levels of the battery which are equidistantly spaced between the minimal $\underline{p} = \min \mathbf{P}$ and the maximal $\overline{p} = \max \mathbf{P}$ level with a step size $\Delta > 0$. Furthermore, consider a finite set $\mathbf{A}$ of actions along with the function $l : \mathbf{A} \to \mathbb{R}$ prescribing the safety margin $l(a)$ chosen by the retailer's action $a \in \mathbf{A}$. According to our assumptions, let us agree that having chosen the action $a \in \mathbf{A}$ at the current battery level P_t, the next level P_{t+1} is modeled as

$$P_{t+1} = \arg\min_{p \in \mathbf{P}} |p - (P_t + l(a) - \varepsilon_t)|,$$

from which the transition probabilities in storage levels are induced by the action a are

$$\alpha^a_{p,p'} = \begin{cases} \mathcal{N}(p + l(a), \varsigma)([p' - \Delta/2, p' + \Delta/2]) & \text{if } \underline{p} < p' < \overline{p}, \\ \mathcal{N}(p + l(a), \varsigma)(]-\infty, \underline{p} + \Delta/2]) & \text{if } p' = \underline{p}, \\ \mathcal{N}(p + l(a), \varsigma)([\overline{p} - \Delta/2, +\infty[) & \text{if } p' = \overline{p} \end{cases}$$

where $\mathcal{N}(p + l(a), \varsigma)$ stands for the probability measure associated with the normal distribution with mean $p + l(a)$ and variance ς^2. In a similar manner, the expected excess $\underline{e}^a_p$ and shortage $\overline{e}^a_p$ of the imbalance energy can be written as

$$\begin{aligned} \underline{e}^a_p &= \textstyle\int_{\overline{p}+\Delta/2}^{\infty} (x - \overline{p}) \mathcal{N}(p + l(a), \varsigma)(dx), \\ \overline{e}^a_p &= \textstyle\int_{-\infty}^{\underline{p}-\Delta/2} (\underline{p} - x) \mathcal{N}(p + l(a), \varsigma)(dx). \end{aligned}$$

With these definitions, the reward functions are given as above in (4.4.1). More specifically, having introduced the rewards

$$(p, \Pi_t, a) \mapsto -l(a)\Pi_t - \overline{e}^a_p \overline{\Pi}_t + \underline{e}^a_p \underline{\Pi}_t, \qquad p \in \mathbf{P}, a \in \mathbf{A}, t = 0, \ldots, T-1$$

and assuming that at maturity date T the entire energy from storage capacity can be sold at the forward market, the scrap value is given by

$$(p, \mathit{\Pi}_T) \mapsto p\mathit{\Pi}_T, \quad p \in \mathbf{P}.$$

Finally, let us assume that the buy/sell grid prices are constant and deterministic

$$\underline{\mathit{\Pi}}_t = \underline{\mathit{\Pi}}, \quad \overline{\mathit{\Pi}}_t = \overline{\mathit{\Pi}}, \quad t = 0, \dots, T, \quad \text{with } 0 < \underline{\mathit{\Pi}} < \overline{\mathit{\Pi}}$$

to define the reward functions by

$$r_t(p, (z^{(1)}, z^{(2)}), a) = -l(a) f(t, z^{(2)}) - \overline{e}^a_p \overline{\mathit{\Pi}} + \underline{e}^a_p \underline{\mathit{\Pi}}, \quad t = 0, \dots, T-1, \tag{4.4.3}$$

for all $a \in \mathbf{A}$, $p \in \mathbf{P}$ and $(z^{(1)}, z^{(2)}) \in \mathbb{R}^2$. Finally, introduce the scrap value by

$$r_T(p, (z^{(1)}, z^{(2)})) = p f(T, z^{(2)}), \quad p \in \mathbf{P}, \quad (z^{(1)}, z^{(2)}) \in \mathbb{R}^2. \tag{4.4.4}$$

Note that with the definitions (4.4.3), (4.4.4) and (4.4.2) our problem of battery storage control is uniquely determined, whose numerical solution is demonstrated in the next section.

Let us suppose that the battery level are equidistantly discretized with $\underline{p} = 0$ MWh, $\overline{p} = 100$ MWh, and $\mathit{\Delta} = 5$ MWh. Furthermore, we assume that energy retailer chooses actions $a \in \mathbf{A} = \{1, 2, \dots, 11\}$ with corresponding safety margins $l(a) = 5(a-1)$ MWh for all $a \in \mathbf{A}$. Further, consider the time horizon of a week at half-hourly frequency i.e $t = 0, 1, \dots, 335 = T$ and define the auto-regressive state dynamics as above, with by $\mu = 0$, $\sigma = 0.5$ and $\phi = 0.9$. To describe seasonality, we assume that the affine linear functions are given by

$$f(t, z^{(2)}) = u_t + v_t z^{(2)}, \qquad t = 0, \dots, T, \quad z^{(2)} \in \mathbb{R} \tag{4.4.5}$$

with deterministic coefficients $u_t = 10 + \cos(\frac{2\pi}{48} t + \frac{3\pi}{2})$, $v_t = 1 + \sin(\frac{2\pi}{48} t + \frac{3\pi}{2})/2$ for $t = 0, \dots, T$. Figure 4.4.3 depicts trajectories of the corresponding price evolution having started at $Z^{(2)}_0 = z^{(2)}_0 = 0$.

Assume that the standard deviation of the consumer's demand prediction error is $\varsigma = 10$ and define grid prices by $\overline{\mathit{\Pi}} = 20$ and $\underline{\mathit{\Pi}} = 0$, respectively. With these quantities, we apply the subgradient approach on a state space grid G containing 501 points equally distributed on the line connecting the points $(1, -15)$ and $(1, 15)$. Furthermore, we discretize the disturbance random variable by partitioning the disturbance space into 1000 components of equal probability measure and then using the conditional expectations on each of the components. Using the expectation matrices approach, it takes around 13 cpu seconds (7 elapsed) to compute the

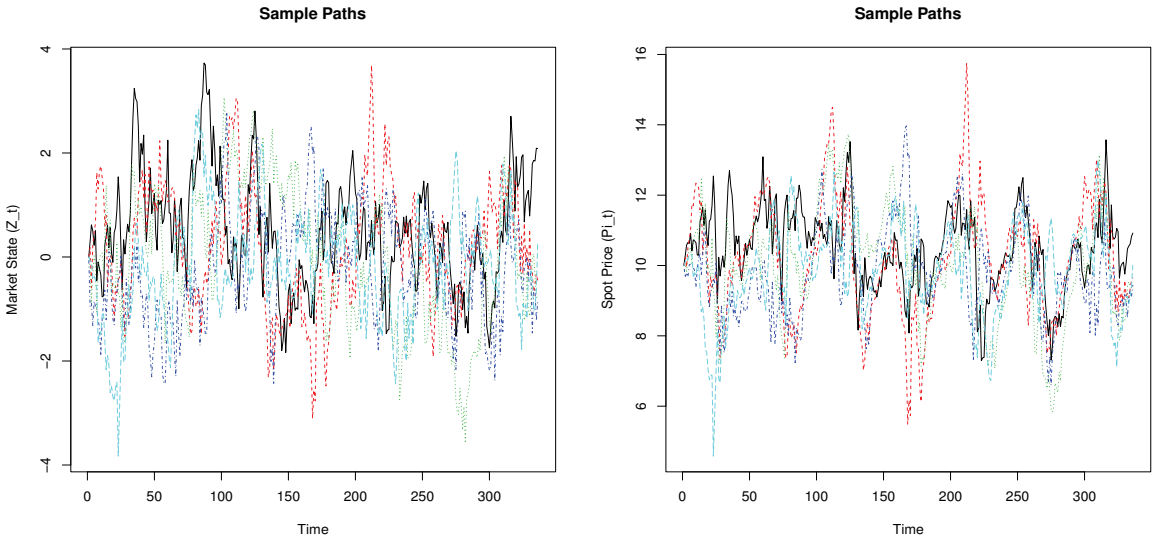

Fig. 4.4.3: Paths the states $(Z_t^{(2)})_{t=0}^T$ (left) and the prices $(\Pi_t = f(t, Z_t^{(2)}))_{t=0}^T$, (right).

Table 4.4.1: Subgradient and duality approach using $Z_0^{(2)} = 0$.

Start Level (MWh)	Subgradient Estimate	99% CI
0	-1679.859	(-1679.862, -1679.612)
5	-1629.859	(-1629.862, -1629.612)
10	-1579.859	(-1579.862, -1579.612)
15	-1529.859	(-1529.862, -1529.612)
20	-1480.168	(-1480.171, -1479.922)
25	-1433.574	(-1433.577, -1433.328)
30	-1389.685	(-1389.688, -1389.441)
35	-1348.509	(-1348.512, -1348.266)
40	-1310.129	(-1310.131, -1309.887)
45	-1274.601	(-1274.604, -1274.362)
50	-1241.952	(-1241.955, -1241.714)
55	-1212.186	(-1212.188, -1211.949)
60	-1185.294	(-1185.297, -1185.059)
65	-1161.261	(-1161.263, -1161.027)
70	-1140.063	(-1140.066, -1139.831)
75	-1121.677	(-1121.680, -1121.446)
80	-1106.080	(-1106.083, -1105.850)
85	-1093.250	(-1093.253, -1093.021)
90	-1083.161	(-1083.164, -1082.932)
95	-1075.727	(-1075.730, -1075.499)
100	-1070.728	(-1070.731, -1070.500)

value functions represented by Column 2. The confidence intervals in Column 3 were computed using the primal and dual values generated using $I = 100$ paths and $I' = 100$ nested simulations. This process took around 16 cpu seconds (8 elapsed seconds). In all, it took slight over 30 cpu seconds to generate the whole table above. The close resemblence of the subgradient estimates and the confidence intervals indicate excellent quality value function approximation atleast around $Z_0^{(2)} = 0$.

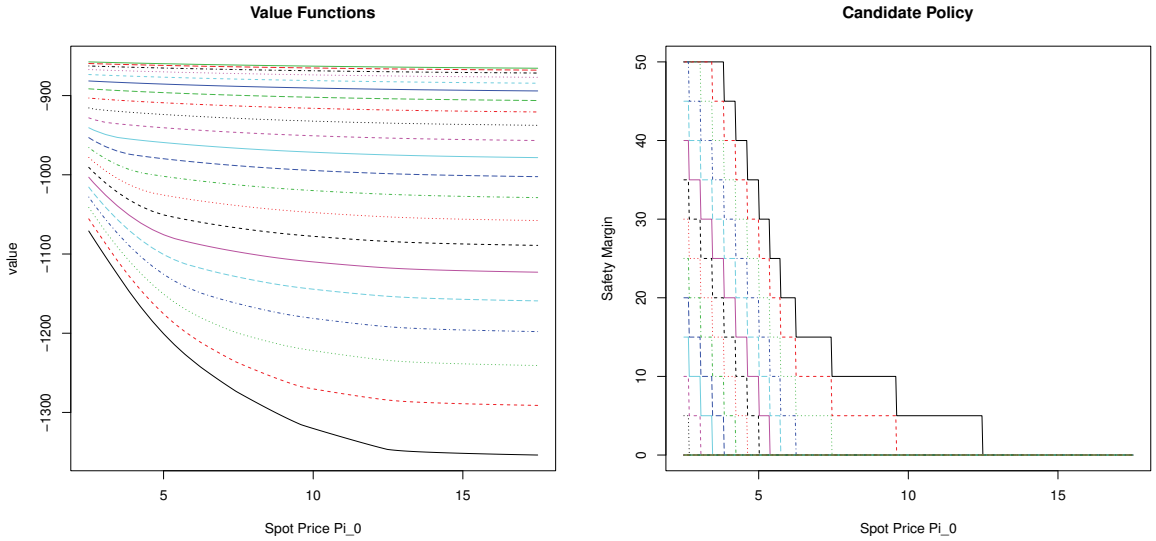

Fig. 4.4.4: Value functions and optimal policy at $t = 0$. Different colours for different positions. At each price, the value increases with the current storage level.

Finally, let us discuss a typical economic application. Having solved the problem of optimal operational management, the questions of investment and capacity allocation can be addressed. Here, the first step is an estimation of the random revenue profile, followed in the second step by a detailed risk analysis from potential side effects of operational flexibility on the entire portfolio of existing physical and financial assets. Thereby, the hedging value of flexibility (see Doege et al (2009)) is essential. In our study, we illustrate this first step using Monte-Carlo simulations. Having calculated the revenue of our approximately optimal strategy on 10000 randomly generated path scenarios, its density histogram is plotted in the left plot of Figure 4.4.5 for different starting storage levels. In line with our expectations, we observe in this figure that a higher initial battery level yields a higher cumulated reward. This is also seen from the left plot of the Figure 4.4.4 which shows that the value function at any price is increasing in storage level (which correspond to different curves). The right graph of Figure 4.4.4 indicates that a higher initial storage level also yields a lower safety margin in the optimal strategy at the

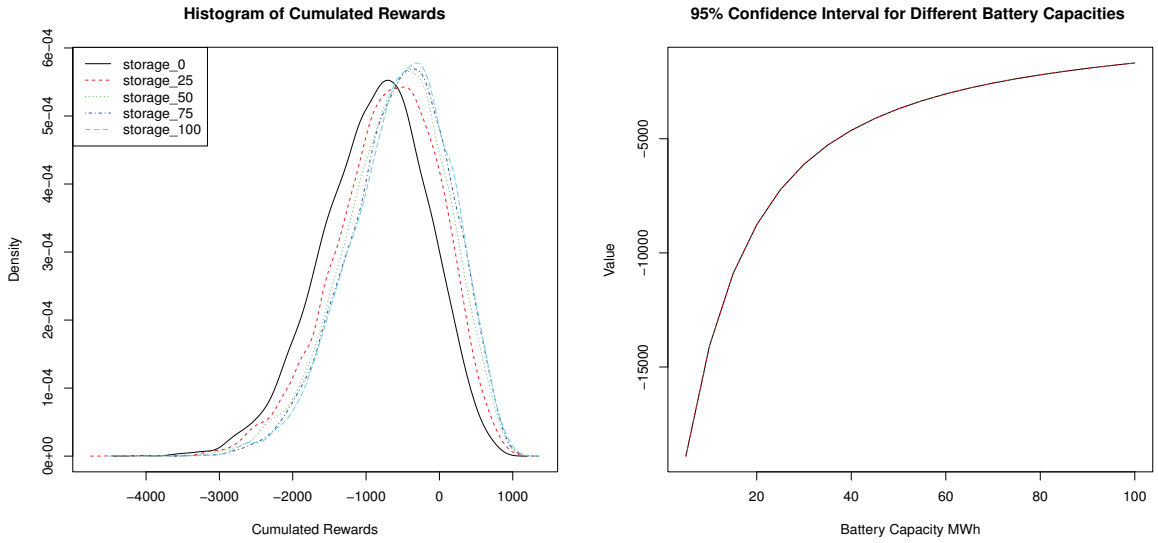

Fig. 4.4.5: The left plot shows the histogram of the cumulated rewards on 10000 sample paths for different starting levels using the prescribed policy. The right depicts two curves, enclosing the 95% confidence region for the cumulated reward, depending on storage capacity.

initial time. The right plot in Figure 4.4.5 shows the dependence of the cumulated rewards on the battery size. In this graph, we gradually increase the capacity from 5 MWh to 100MWh and determine the value function for an initially empty storage (for $\phi = 0.9$, $z_0^{(2)} = 0$). The concave curve depicted in this graph shows that the value grows with the capacity at a rate which is steadily decreasing. Such insight may be very valuable in practice. For instance, the optimal size for battery deployment would typically result from equating the marginal value of the storage to its marginal cost. Notice however, that the true practical value of such optimization usually stems from risk hedging effects within agent's energy generation portfolio, whose analysis now becomes possible, using a reliable strategy optimization provided by our concepts.

4.4.3 Trading and storage

In this section, we discuss the impact of storage facility on optimal energy trading. Recall that in our model, the agent indirectly controls the storage level in terms of energy trading. The point is that the storage absorbs some (if not all) of any unexpected demand which adds some flexibility in the price choice when energy is purchased. To this end, we investigate the impact of battery size, comparing a small (5MWh) against large (100MWh) storage.

Table 4.4.2: Bounds on cumulated rewards estimation for different parameters.

	Small Battery 5 MWh		Large Battery 100 MWh	
ϕ	Lower	Upper	Lower	Upper
0.9	-18904.06 (0.151)	-18904.06 (0.151)	-1679.759 (0.042)	-1679.756 (0.042)
0.6	-19004.19 (0.073)	-19004.19 (0.073)	-1682.616 (0.037)	-1682.609 (0.037)
0.3	-19017.53 (0.060)	-19017.52 (0.059)	-1679.807 (0.038)	-1679.799 (0.039)
0.1	-19019.21 (0.057)	-19019.20 (0.057)	-1676.744 (0.042)	-1676.732 (0.042)

In our state dynamics (4.4.2), the parameter ϕ controls the speed of mean reversion for the energy price. Thereby, the lower levels of ϕ lead to stronger mean reversion with more frequent return to the seasonal price component of the price. Table 4.4.2 compares the expected cumulated rewards for a small battery with 5 MWh capacity against a large battery of size 100 MWh under the assumptions that both batteries are initially empty. This table shows that there is a very substantial benefit of extra storage capacity for all levels of mean reversion. However, mean reversion seems to have very little impact on the expected cumulated rewards. Further, Figure 4.4.6 illustrates the role of storage capacity for energy purchase. Here, we depict the difference between the safety margins optimally entered at time $t = 0$ for empty storage. We observe that having a large battery capacity allows to buy more energy in advance. Note also that this effect decreases with increasing price. Thereby, the impact of mean reversion parameter is low again. Finally, we examine in Figure 4.4.7 the averaged behavior (on 10000 scenarios) of safety margins in dependence on storage capacity for different mean reversion parameters. In line with the previous observations, the speed of mean reversion seems to have only little impact. Remarkably, there is a clear seasonal pattern in the difference of optimal safety margins. This is caused by the seasonal nature of prices as shown in Figure 4.4.3. Namely, for large storage capacity, the optimal trading follows price seasonality stronger than if the storage is small. This issue is also obvious from Figure 4.4.8 which shows that for large capacity it is optimal to keep a certain intermediate level whereas small storage must be filled right at the beginning to hedge against unexpected demand fluctuations. We also observe that close to maturity there is an attempt to fill the storage in order to benefit from price at the end of the time period.

In our finite horizon setting, three phases are observable from Figure 4.4.8. After an initial charge to an "optimal intermediate level", the battery is used to absorb unexpected demand/-supply fluctuations while remaining close to this level. However, closer to maturity, the storage is further filled to take advantage of the price at the end of the time horizon. On this account, it is important to compare the portion of the cumulative reward which results from intermediate balancing to that earned from selling energy at maturity. To investigate this problem, alter the scrap value definition (4.4.4) to

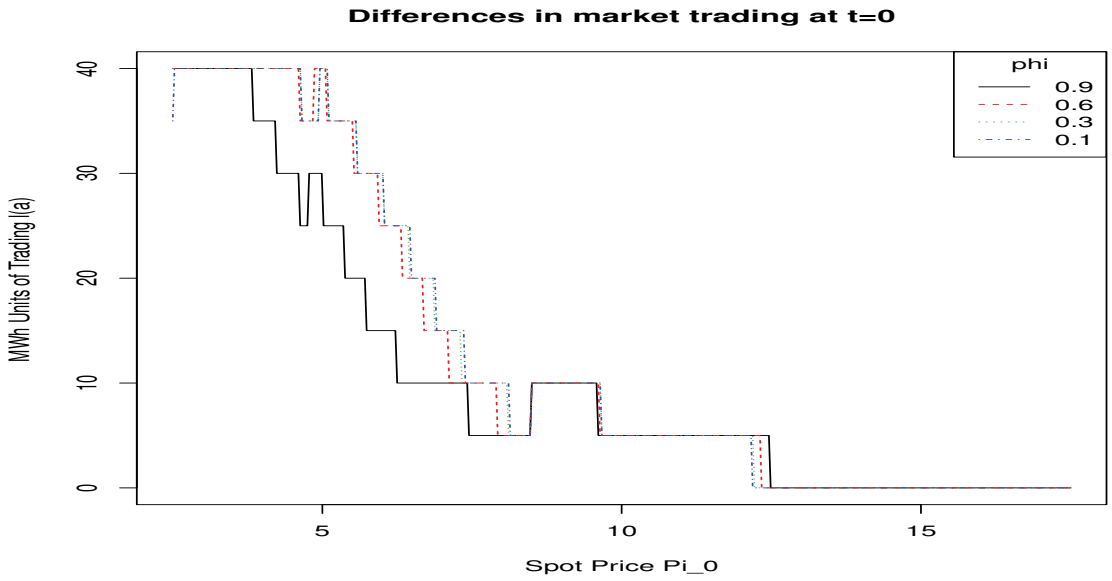

Fig. 4.4.6: Differences in optimal trading at $t = 0$. Both batteries intially empty.

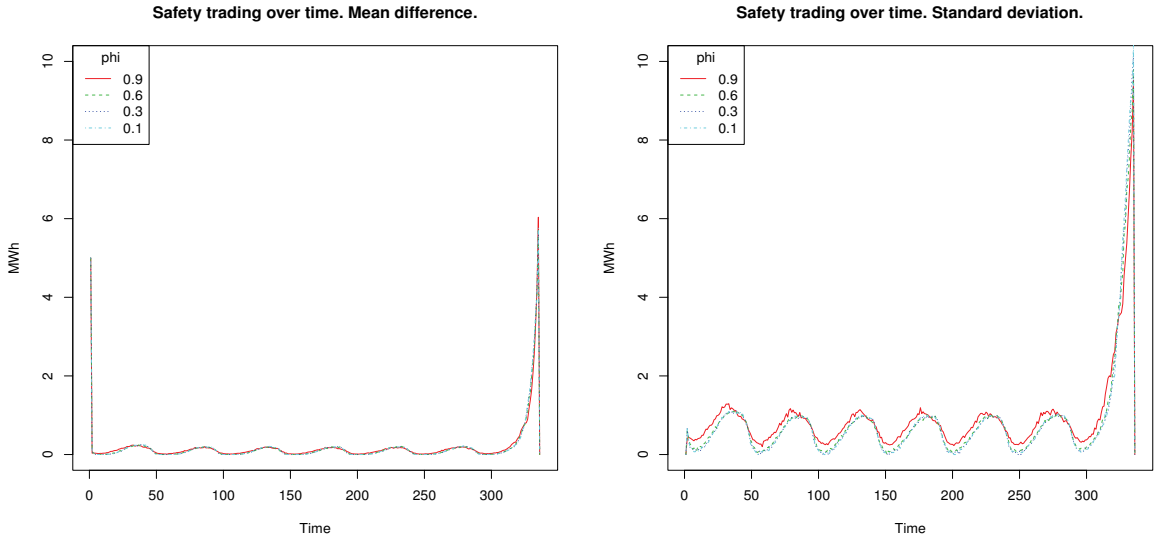

Fig. 4.4.7: Averaged difference (10000 scenarios) in safety margin between the large and small battery (both initially empty) over time for different mean reversions. The empirical mean is on the left and the empirical standard deviation is on the right.

$$r_T(p, (z^{(1)}, z^{(2)})) = 0 \quad p \in \mathbf{P}, \quad (z^{(1)}, z^{(2)}) \in \mathbb{R}^2. \tag{4.4.6}$$

With this change, any energy remaining at the end is worthless. Table 4.4.3 compares the expected cumulative rewards of the original (4.4.4) problem to that with (4.4.6) for different

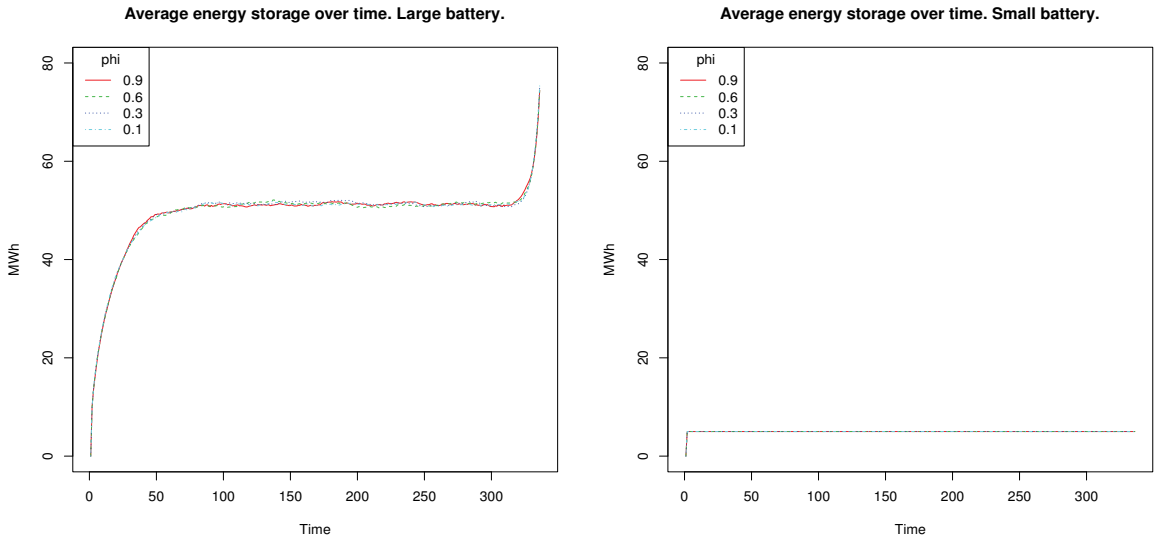

Fig. 4.4.8: The averaged evolution (10000 scenarios) of the energy storage level for the large battery, on the left, and for small battery on the right.

Table 4.4.3: Bounds on cumulated rewards under $\phi = 0.9$ and $z^{(2)} = 0$.

Capa-	Non-zero scrap value		Zero scrap value	
city	Lower	Upper	Lower	Upper
5	-18904.061 (0.151)	-18904.059 (0.151)	-18929.872 (0.151)	-18929.871 (0.151)
10	-14068.958 (0.115)	-14068.957 (0.115)	-14124.612 (0.115)	-14124.611 (0.115)
15	-10902.280 (0.093)	-10902.279 (0.093)	-10987.056 (0.093)	-10987.055 (0.093)
20	-8762.276 (0.078)	-8762.275 (0.077)	-8879.116 (0.078)	-8879.115 (0.078)
25	-7229.580 (0.061)	-7229.579 (0.061)	-7377.276 (0.061)	-7377.275 (0.061)
30	-6114.388 (0.049)	-6114.388 (0.049)	-6292.050 (0.049)	-6292.049 (0.049)
35	-5278.425 (0.042)	-5278.424 (0.042)	-5485.823 (0.043)	-5485.822 (0.043)
40	-4629.497 (0.039)	-4629.496 (0.039)	-4866.371 (0.039)	-4866.370 (0.039)
45	-4110.807 (0.036)	-4110.806 (0.036)	-4376.629 (0.036)	-4376.628 (0.036)
50	-3685.724 (0.033)	-3685.723 (0.033)	-3980.018 (0.033)	-3980.017 (0.033)
55	-3332.049 (0.032)	-3332.048 (0.032)	-3654.509 (0.031)	-3654.508 (0.031)
60	-3033.977 (0.030)	-3033.977 (0.030)	-3384.379 (0.029)	-3384.379 (0.029)
65	-2779.557 (0.028)	-2779.557 (0.028)	-3157.764 (0.027)	-3157.763 (0.027)
70	-2559.781 (0.028)	-2559.781 (0.028)	-2965.728 (0.027)	-2965.728 (0.027)
75	-2367.843 (0.029)	-2367.842 (0.029)	-2801.529 (0.026)	-2801.528 (0.026)
80	-2198.558 (0.031)	-2198.557 (0.031)	-2660.035 (0.027)	-2660.034 (0.027)
85	-2047.928 (0.033)	-2047.926 (0.033)	-2537.283 (0.028)	-2537.282 (0.028)
90	-1912.817 (0.035)	-1912.815 (0.035)	-2430.169 (0.029)	-2430.168 (0.029)
95	-1790.759 (0.038)	-1790.755 (0.038)	-2336.235 (0.031)	-2336.234 (0.031)
100	-1679.759 (0.042)	-1679.756 (0.042)	-2253.495 (0.033)	-2253.493 (0.033)

storage capacities. It is not surprising that a larger battery allows exploiting the remaining energy to a greater extent. However comparing both columns in this table, we observe that a very significant part of the battery value results from the energy balancing.

4.4.4 Final thoughts

Electrical storages have the potential to essentially change the nature of electricity trading and may have profound impact on energy price dynamics. This section provides quantitative concepts to better understand and analyze this development. We demonstrate that using our algorithmic approach to battery storage management, a detailed and accurate strategy optimization is possible. Further details, such as modelling uncertainties in grid prices, costs of deep discharge affecting battery's life time, and stochastic futures price dynamics can be incorporated. The authors will address these exciting topics in future research.

4.5 Natural resource extraction

This section examines the management of a commodity resource with a finite amount of commodity as studied by Brennan and Schwartz (1985). In this setting, the decision maker aims to maximize the total expected profit using controls whose costs are given in Table 4.5.1. The controller dynamically switches the operational mode of the resource between {Opened, Closed, Abandoned}. When opened, commodity is extracted, sold at the spot market and a revenue is realized based on the commodity price. While closed, commodity is preserved but a maintenance cost is incurred. The abandoned mode incurs no cost and provides no revenue. Switching to the abandoned mode causes the mode to remain permanently there. There is a switching cost between modes. The original problem Brennan and Schwartz (1985) was considered in a continuous time setting and solved using partial differential equations. However, this paper will consider the problem in discrete time and a finite time horizon. The rates, costs and taxes above will be adjusted accordingly.

Table 4.5.1: Hypothetical copper mine (see Brennan and Schwartz (1985)).

Output rate: 10 million pounds/year	Inventory level: 150 million pounds
Costs of production: \$0.50/pound	Opening/closing costs: \$200,000
Maintenance costs: \$500,000/year	Inflation rate: 8%/year
Convenience yield: 1%/year	Price variance: 8%/year
Real estate tax: 2%/year	Income tax: 50%
Royalty tax: 0%	Interest rate: l0%/year

In this section, we use the discrete component $(P_t)_{t=0}^{T}$ to convey information regarding the remaining level of commodity and the current operational mode. More precisely, set

$\mathbf{P} = \{0,1,\ldots,I\} \times \{1,2\}$ to capture all possible positions with the interpretation that the first component $p^{(1)}$ of the position $(p^{(1)}, p^{(2)}) \in \mathbf{P}$ describes the remaining commodity level such that $p^{(1)} = 0$ represent an exhausted or abandoned resource. The second component $p^{(2)}$ indicates whether the current operational mode is closed $p^{(2)} = 1$ or opened $p^{(2)} = 2$. To introduce the mode control, we set $\mathbf{A} = \{0,1,2\}$ where $a = 0,1,2$ stands for the action to abandon, to close, and to open the asset, respectively. The second component $(Z_t)_{t=0}^T$ will be used to capture the market dynamics of the the commodity price. The exact form of the disturbance matrices $(W_t)_{t=1}^T$ depends on the particular choice of the price process.

Let us now turn to the control costs. If the system is in the state (p,z), the costs of applying action $a \in \mathbf{A}$ at time $t = 0,\ldots,T-1$ are expressed through $r_t(p,z,a)$. Having arrived at time $t = T$ in the state (p,z), a final *scrap value* $r_T(p,z)$ is collected. Thereby the reward and scrap functions

$$r_t : \mathbf{P} \times \mathbf{Z} \times \mathbf{A} \to \mathbb{R}, \quad r_T : \mathbf{P} \times \mathbf{Z} \to \mathbb{R}$$

are exogenously given for $t = 0,\ldots,T-1$. If the asset is abandoned, there are neither costs nor revenue

$$r_t((0,p^{(2)},z),a) = 0, \qquad a \in \mathbf{A}, \quad z \in \mathbf{Z}.$$

For the case where $p^{(1)} > 0$, we define

$$r_t((p^{(1)},p^{(2)},z),a) = h_t(z)1_{\{2\}}(a) - m_t 1_{\{1\}}(a) - c_t 1_{\{1,2\}}(a)|p^{(2)} - a|$$

with the following interpretation: If the resource is opened, a revenue is collected which depends on continuous state component z through a pre-specified function h_t. If the resource is closed, then non-stochastic maintenance fee m_t is to be paid. Finally, a deterministic switching fee c_t is incurred whenever the operational mode transitions from opened to closed or vice versa.

4.5.1 Brennan and Schwartz (1985)

In the original formulation of (Brennan and Schwartz, 1985), a continuous time setting was assumed and that the price of the commodity Z_t evolved as a geometric Brownian motion i.e.

$$\mathrm{dZ_t} = (\mathrm{r} - \delta)\mathrm{Z_t dt} + \sigma \mathrm{Z_t dB_t}$$

where r is the interest rate, δ the convenience yield, and B_t is a standard Brownian motion. Since the subgradient approach perform backward induction, the parameters used by Brennan and Schwartz (1985) in Table 4.5.1 are discretized in the following manner. Let us denote the choice of the time step by Δ and time horizon by $\bar{T}$. Suppose the first entry of the discrete component takes values $p^{(1)} \in \left\{0, 1, ..., \frac{\bar{p}}{\Delta}\right\}$ where $\bar{p}$ stands for the minimum number of years it takes to deplete the resource when the mine is opened continuously. The number of decision epochs is given by $T = 1 + \frac{\bar{T}}{\Delta} + 1$, the maintenance costs $m_t = m_0 \Delta e^{(\rho - r - \zeta)t\Delta}$ and the switching costs $c_t = c_0 e^{(\rho - r - \zeta)t\Delta}$ with coefficients $r = 0.1$, $\rho = 0.08$, $\zeta = 0.02$, $m_0 = 0.5$ and $c_0 = 0.2$ standing for the interest rate, rate of inflation, real estate tax, initial maintenance cost, and initial switching cost, respectively.

The spot price is given by the state component $(Z_t)_{t=0}^T$ which follows linear state dynamics:

$$Z_{t+1} = \exp\left(\left(r - \delta - \frac{\sigma^2}{2}\right)\Delta + \sigma\sqrt{\Delta} N_{t+1}\right) Z_t \tag{4.5.1}$$

with $\delta = 0.01$, $\sigma^2 = 0.08$ and independent identically standard normally distributed $(N_t)_{t=1}^T$. The transition probabilities of the discrete component, given $(p^{(1)}, p^{(2)}) \in \mathbf{P}$, are uniquely determined by

$$\begin{aligned}
&\alpha^{\text{Open}}_{(p^{(1)},p^{(2)}),(\max\{p^{(1)}-1,0\},\text{Opened})} = 1,\\
&\alpha^{\text{Close}}_{(p^{(1)},p^{(2)}),(p^{(1)},\text{Closed})} = 1,\\
&\alpha^{\text{Abandon}}_{(p^{(1)},p^{(2)}),(0,p^{(2)})} = 1.
\end{aligned}$$

That is, the operation mode is controlled deterministically under the above specfication. Now let us define the revenue of the opened resource by

$$h_t(z) = 5\Delta z e^{-(r+\zeta)t\Delta} - 2.5\Delta e^{(\rho - r - \zeta)t\Delta}$$

where the variable z represents the commodity price. The scrap is given by $r_T = 0$ for all states.

Assume $\Delta = 0.25$ years, $\bar{p} = 15$, and $\bar{T} = 30$ years. Using 2001 equally spaced grid points from $z = 0$ to $z = 10$ and a disturbance sampling obtained using 1000 local averages from partitioning the possible realizations of the lognormal random variable in (4.5.1), Table 4.5.2 contains the value estimates from the subgradient approach as well as the confidence intervals for the true value obtained using $I = 500$ sample paths and $I' = 500$ nested simulations. The function approximations took 56 cpu seconds (16 elasped) and the confidence intervals took 80

cpu seconds (50 elapsed) to compute for each starting Z_0. The results below indicate that our value function approximations are of excellent quaility around Z_0. For comparison, Columns 4 and 7 give the values obtained by numerically solving the partial differential equations as shown in (Brennan and Schwartz, 1985).

Table 4.5.2: Valuation under geometric Brownian motion.

	Opened Resource			Closed Resource		
Z_0	Estimate	99% CI	PDEs	Estimate	99% CI	PDEs
0.3	1.209	(1.204, 1.233)	1.25	1.409	(1.404, 1.433)	1.45
0.4	4.100	(4.096, 4.138)	4.15	4.300	(4.296, 4.338)	4.35
0.5	7.895	(7.893, 7.937)	7.95	8.068	(8.066, 8.110)	8.11
0.6	12.506	(12.501, 12.547)	12.52	12.472	(12.468, 12.516)	12.49
0.7	17.580	(17.573, 17.623)	17.56	17.380	(17.373, 17.426)	17.38
0.8	22.941	(22.931, 22.984)	22.88	22.741	(22.731, 22.784)	22.68
0.9	28.487	(28.476, 28.532)	28.38	28.287	(28.276, 28.332)	28.18
1.0	34.159	(34.148, 34.206)	34.01	33.959	(33.948, 34.006)	33.81

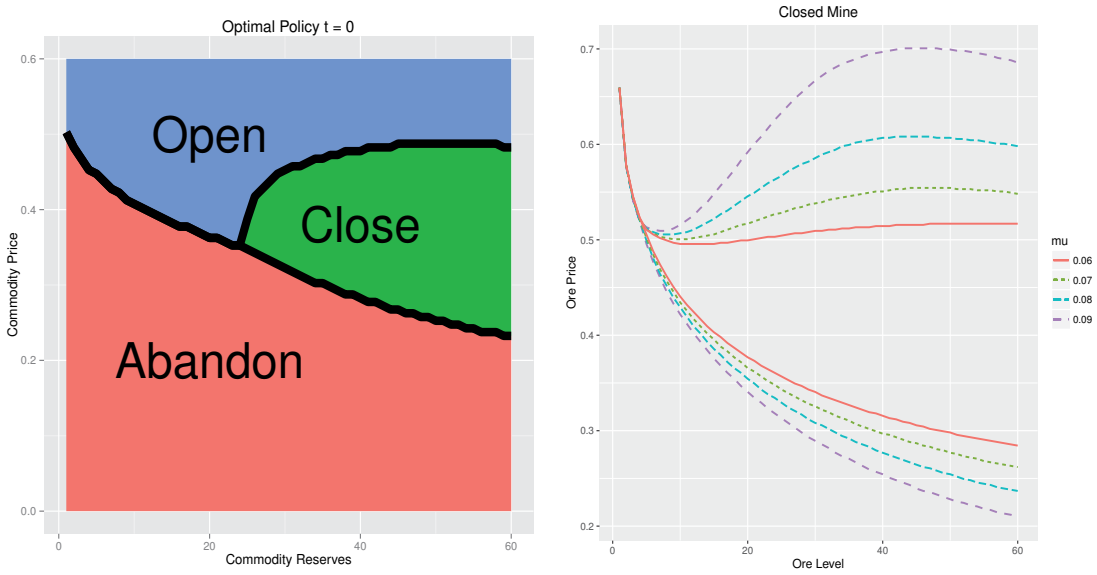

Fig. 4.5.1: The optimal policy at $t = 0$. The horizontal axis represents remaining reserves while the vertical represents the commodity price. The right plot displays policy for a commodity resource that begins closed for $\mu = 0.09, 0.08, 0.07, 0.06$.

In this work we illustrate control policies in the form of curves as depicted in Figure 4.5.1. Namely, we have observed that for all situations examined in this work, the optimal policies share the same structure: It turns out that it is optimal to operate an open resource when the commodity price is sufficiently high, while below some price level the resource should be shut

down, either to the closed or to the abandoned mode. Typically, we have a curve bifurcation and the area between both curve branches represents the price range where switching to the closed mode is optimal. In the right plot of Figure 4.5.1, the optimal policies for a closed resource is shown for different values of μ. Recall that there is a strong element of scarcity in our problem. That is, selling a unit of limited commodity now means that the same unit of commodity cannot be sold at a future date. Thus, we observe from Figure 4.5.1, that the decision maker is more willing to preserve the commodity for future use if the drift term μ is high. In our subsequent studies, we will assume drift $\mu = r - d = 0.09$ (unless otherwise stated) in order to stay in line with Brennan and Schwartz (1985).

4.5.2 Mean reversion

Commodity prices often mimic the business cycle in the real-world market (see Paschke and Propkopczuk (2010); Schwartz (1997)). Prices with significant mean reversion often lead to different optimal behavior compared to those following geometric Brownian motions (Sarkar (2003); Schwartz (1997); Tsekrekos (2010)). To examine the impact of mean reversion, the logarithm of the commodity price $(\log \tilde{Z}_t)_{t=0}^T$ is assumed to follow an AR(1) process, an auto-regression of order one. With this choice, define the linear state evolution $(Z_t)_{t=0}^T$ in the required form $Z_{t+1} = W_{t+1} Z_t$ as

$$Z_{t+1} := \begin{bmatrix} 1 \\ \log \tilde{Z}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \left(r - \delta - \frac{\sigma^2}{2}\right)\Delta + \sigma\sqrt{\Delta} N_{t+1} & \phi \end{bmatrix} \begin{bmatrix} 1 \\ \log \tilde{Z}_t \end{bmatrix},$$

for $t = 0, \ldots, T-1$, with initial value $\log \tilde{Z}_0 \in \mathbb{R}$ and $r - \delta = 0.09$, $\sigma^2 = 0.08$. Again, $(N_t)_{t=1}^T$ are independent standard normally distributed random variables and the parameter $\phi \in [0,1]$ determines the speed of mean reversion, where $\phi = 1$ gives a geometric Brownian motion. Using the same parameters as before, we merely adjust the cash flow function accordingly for $t = 0, \ldots, T-1$ to

$$h_t(z) = 5\Delta \exp\left(z^{(2)}\right) \exp(-(r+\zeta)t\Delta) - 2.5\Delta \exp((\rho - r - \zeta)t\Delta)$$

for $z = (z^{(1)}, z^{(2)}) \in \mathbb{R}^2$ since now the variable $z^{(2)}$ captures the logarithmic commodity price. With this, the reward functions are not globally Lipschitz in $z^{(2)}$. However, as methioned before, global Lipschitz continuity was used as one of the sufficient condtions for the existence of the

value functions and is not a neccessary condition for the subgradient approach to work. This is demonstrated by the following numerical results.

Table 4.5.3: Valuation under mean reverting prices.

		Opened Resource		Closed Resource	
ϕ	$\tilde{Z}_0$	Estimate	99% CI	Estimate	99% CI
1.0	0.3	1.215	(1.209, 1.253)	1.415	(1.409, 1.453)
	0.4	4.077	(4.100, 4.144)	4.277	(4.300, 4.344)
	0.5	7.901	(7.895, 7.938)	8.073	(8.068, 8.112)
	0.6	12.448	(12.504, 12.548)	12.416	(12.470, 12.514)
	0.7	17.541	(17.578, 17.622)	17.341	(17.378, 17.422)
0.8	0.3	6.320	(6.317, 6.321)	6.303	(6.301, 6.305)
	0.4	7.268	(7.274, 7.277)	7.076	(7.081, 7.084)
	0.5	8.116	(8.114, 8.118)	7.916	(7.914, 7.918)
	0.6	8.868	(8.876, 8.880)	8.668	(8.676, 8.680)
	0.7	9.576	(9.581, 9.584)	9.376	(9.381, 9.384)
0.6	0.3	7.674	(7.673, 7.674)	7.604	(7.603, 7.605)
	0.4	8.148	(8.151, 8.153)	7.951	(7.953, 7.955)
	0.5	8.575	(8.575, 8.577)	8.375	(8.375, 8.377)
	0.6	8.958	(8.962, 8.964)	8.758	(8.762, 8.764)
	0.7	9.320	(9.323, 9.325)	9.120	(9.123, 9.125)

In Table 4.5.3, the value function approximations were computed using 2000 equally spaced grid points ranging from -5 to 5 for $\log(\tilde{Z}_t)$ and a disturbance sampling constructed from the local averages after partitioning the set of possible values for N_{t+1} into 1000 components of equal probability measure. The computional time under these parameters is 54 cpu seconds (17 elapsed seconds). The 99% confidence intervals were obtained using the primal and dual values computed using $I = 500$ sample paths and $I' = 500$ nested simulations. Table 4.5.3 contains results for different strengths of mean reversion $\phi = 1, 0.8, 0.6$. Note that when $\phi = 1$, the geometric Brownian motion case in the previous subsection is recovered. The computational time for computing the confidence intervals is 77 cpu seconds (50 elapsed) for each starting Z_0, with the majority of the time taken up by the calculation of the control variates. Table 4.5.3 suggests that the function approximations are of good quality as evidenced by the tight confidence intervals.

Figure 4.5.2 provides an interesting insight. We observe that under mean reversion $|\phi| < 1$, each additional unit of commodity reserve exhibits diminishing marginal value which directly contrasts the behaviour under geometric Brownian motion. This phenomenon was also observed for the closed commodity resource.

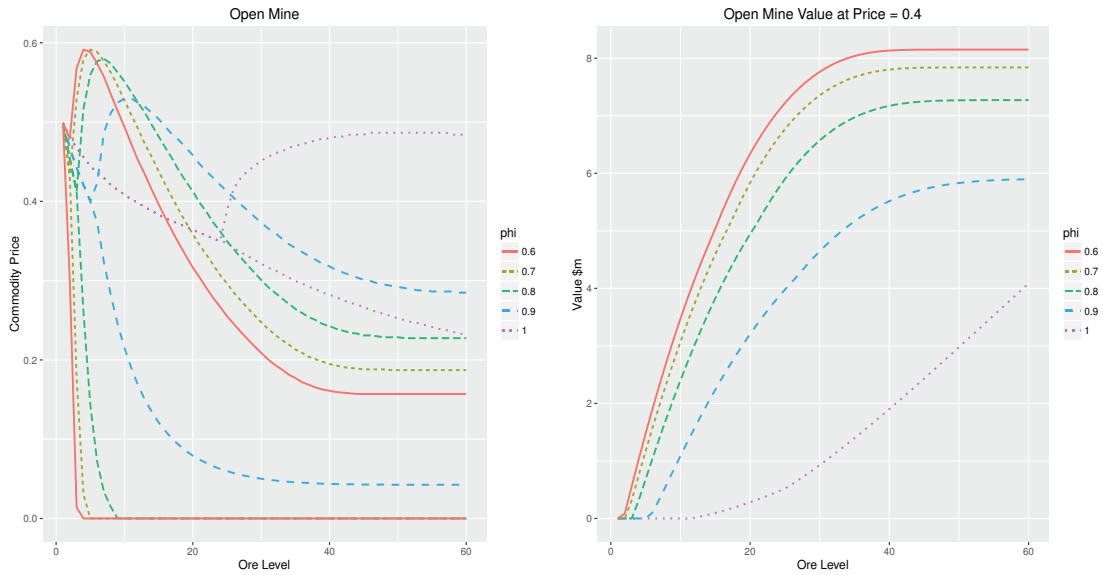

Fig. 4.5.2: Left plot shows policy for an opened commodity resource under $\phi = 1, 0.9, 0.8, 0.7, 0.6$ while the right plot displays the value of the resource as a function of commodity reserves for $\tilde{Z}_0 = 0.4$.

4.5.3 Stochastic volatility

Let us now consider prices with stochastic volatility. Such processes are popular in modeling the effects of time-changing fluctuation intensity, the so-called volatility. However, the original GARCH definition includes a non-linear recursion and is not covered by our approach which requires a linear state dynamics. For this reason, we suggest a simple modification in order to retain this characteristic feature. In what follows, we consider a GARCH(1,1)-like model. We define the process $(\log \tilde{Z}_t)_{t=0}^T$ recursively using independent standard normally distributed random variables $(N_t)_{t=1}^T$ as

$$\begin{aligned}
\sigma_{t+1}^2 &= \sigma^2\beta + \beta_1\sigma_t^2 + \beta_2 Y_t^2 \\
Y_{t+1}^2 &= \sigma_{t+1}^2 N_t^2 \\
\log(\tilde{Z}_{t+1}) &= \kappa\Delta + \phi\log(\tilde{Z}_t) + \sigma_{t+1}^2\sqrt{\Delta}N_{t+1}
\end{aligned}$$

for $t = 0, \ldots, T-1$, with initial values $\sigma_0^2 = \sigma^2 = \sqrt{0.08}$ $Y_0^2, \tilde{Z}_0 \in \mathbb{R}_+$ and parameters $\kappa = \mu - \sigma^4/2 = 0.05$, $\beta_1, \beta_2 \in \mathbb{R}_+$ with $\beta_1 + \beta_2 \in [0,1]$ such that $\beta = 1 - \beta_1 - \beta_2$. With this definition, $(\sigma_t^2)_{t=0}^T$ follows the same recursion as the conditional variances the original GARCH(1,1) process and can be interpreted as the volatility proxy of the commodity price $(\tilde{Z}_t)_{t=0}^T$, since $\sigma_{t+1}^2\sqrt{\Delta}$ is the conditional standard deviation of the increment $\log \tilde{Z}_{t+1} - \log \tilde{Z}_t$ for $t = 0, \ldots T-1$. Positive

values of β_1, β_2 induce volatility clustering and a mean reversion is induced when $\phi \in]0,1[$. With this choice, we define the linear evolution $(Z_t)_{t=0}^T$ for the state variables

$$Z_t = [\,1,\, \sigma_t^2,\, Y_t^2,\, \log \tilde{Z}_t,]^\top, \qquad t = 0, \dots T,$$

in the required form $Z_{t+1} = W_{t+1} Z_t$ as

$$Z_{t+1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \sigma^2\beta & \beta_1 & \beta_2 & 0 \\ \sigma^2\beta N_{t+1}^2 & \beta_1 N_{t+1}^2 & \beta_2 N_{t+1}^2 & 0 \\ \kappa\Delta + \sigma^2\beta\sqrt{\Delta}N_{t+1} & \beta_1\sqrt{\Delta}N_{t+1} & \beta_2\sqrt{\Delta}N_{t+1} & \phi \end{bmatrix} \begin{bmatrix} 1 \\ \sigma_t^2 \\ Y_t^2 \\ \log \tilde{Z}_t \end{bmatrix}$$

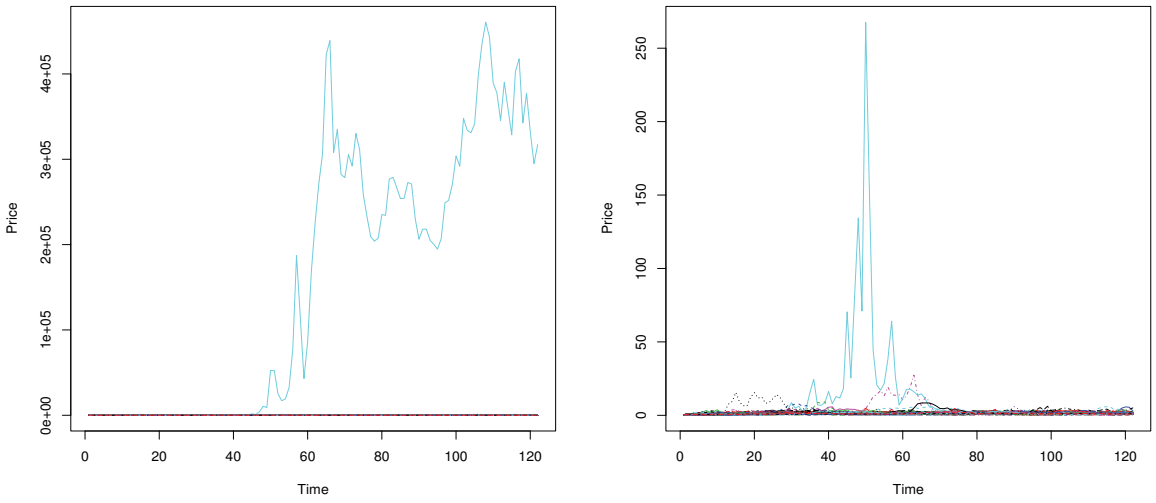

Fig. 4.5.3: Sample paths for the spot price under no mean reversion $\phi = 1$ (left) and mean reversion $\phi = 0.9$ (right)

Figure 4.5.3 depicts the sample paths for the commodity price $(\tilde{Z}_t)_{t=0}^T$ for parameters: $\tilde{Z}_0 = 0.4$, $\phi = 1$ and 0.6, $\kappa = 0.05$, $\beta_1 = 0.8$, $\beta_2 = 0.1$, $\sigma^2 = \sigma_0^2 = \sqrt{0.08}$ and $Y_0^2 = 1$. The left and the right plots show the sample paths with and without mean reversion, for $\phi = 1$ and $\phi = 0.9$, respectively. The behaviour of the prices are clearly very different. Since the logarithmic commodity price is now contained in the fourth component $z^{(4)}$ of the state vector $z = (z^{(i)})_{i=1}^4$, we adjust the cash flow function to

$$h_t(z) = 5\Delta \exp\left(z^{(4)}\right) \exp(-(r+\zeta)t\Delta) - 2.5\Delta \exp((\rho - r - \zeta)t\Delta).$$

Unlike before, for this four dimensional state space we generate the so-called stochastic grid using a clustering of the point cloud generated by Monte Carlo sample paths simulations. Such procedure is appropriate for problems involving high dimensional state spaces as mentioned before. The value function approximations in Table 4.5.4 were calculated using value function approximations obtained from a 3000 point stochastic grid generated using 1000 sample paths starting from $(1, \sqrt{0.08}, 1, \log(0.5))$ and a 10000 disturbance sampling using local averages after partitioning the set of possible values of N_{t+1} into components of equal probability measure. The computational time for the function approximations using expectation matrices is 350 cpu seconds (100 elapsed seconds). Table 4.5.4 contain confidence intervals for the value of the commodity resource for $\beta_1 = 0.8$, $\beta_2 = 0.1$ and $Y_0^2 = 1$ using $I = 500$ paths and $I' = 500$ nested simulations. The computational time needed to compute the confidence intervals for each starting $\tilde{Z}_0$ is 139 cpu seconds (85 elapsed seconds). Table 4.5.4 reveals that the subgradient approach performs poorly when the stochastic process are extremely volatile $\phi = 1$. The difference in the volatility of the sample paths can be seen in Figure 4.5.3. In principle, the quality of the approximations can be improved by selecting a wider and more dense grid or a larger disturbance sampling.

Table 4.5.4: Resource valuation under stochastic volatility where $\kappa = 0.05$, $\beta_1 = 0.8$, $\beta_2 = 0.1$, $\sigma_0^2 = \sqrt{0.08}$ and $Y_0^2 = 1$.

		Opened Resource		Closed Resource	
ϕ	$\tilde{Z}_0$	Estimate	99% CI	Estimate	99% CI
1.0	0.3	2.772	(3.232, 6.458)	2.972	(3.432, 6.658)
	0.4	6.340	(7.286, 11.280)	6.540	(7.486, 11.480)
	0.5	10.646	(11.794, 17.608)	10.846	(11.994, 17.808)
	0.6	15.581	(16.793, 22.954)	15.663	(17.002, 23.086)
	0.7	20.990	(22.362, 28.420)	20.900	(22.389, 28.421)
0.9	0.3	4.418	(4.542, 4.599)	4.564	(4.688, 4.744)
	0.4	6.161	(6.250, 6.302)	6.021	(6.110, 6.163)
	0.5	7.712	(7.789, 7.839)	7.512	(7.589, 7.639)
	0.6	9.116	(9.192, 9.241)	8.916	(8.992, 9.041)
	0.7	10.417	(10.492, 10.542)	10.217	(10.292, 10.342)
0.8	0.3	6.410	(6.493, 6.519)	6.396	(6.493, 6.521)
	0.4	7.412	(7.456, 7.480)	7.225	(7.271, 7.297)
	0.5	8.270	(8.304, 8.327)	8.070	(8.104, 8.128)
	0.6	9.038	(9.071, 9.095)	8.838	(8.871, 8.895)
	0.7	9.749	(9.780, 9.804)	9.549	(9.580, 9.604)

4.5.4 Digital mining

Finally, let us emphasize a potential application of our methods in a novel domain, the "mining" of crypto-currencies. At first glance, the analogy between commodity extraction and digital tokens processing appears as a pun with the word "mining". However, planing and optimizing operations in both areas requires using stochastic control methods. The authors believe that in the field of crypto currencies, diverse crucial questions of network stability (resilience to the so-called *double spending attacks*) must be addressed in terms of sound and precise solutions to complex stochastic switching problems. Let us briefly share our observations. In the context of BitCoin mining (Nakomoto (2008); Rosenfeld (2014); Gruenspan and Perez-Marco (2017)), it is supposed that a merchant waits for $y \in \mathbb{N}$ confirming blocks after the paying transaction of the attacker, before a product/service is provided. While the network is mining these y blocks, the attacker tries building a secret private branch which includes an invalidation. The analysis by Rosenfeld (2014) provides an estimate of attacker's success probability depending on his computational power and the number y of confirming blocks. However, this investigation is based on a number of simplifying assumptions, neglecting the relation between mining costs and potential revenues, the possibility to cancel the secret mining at any time, and supposing that the attacker attempts to fork the block chain right before the paying transaction. Particularly the last assumption is crucial and is justifiable only if the merchant accepts nothing but an immediate payment. However, in reality the attacker usually can choose the payment moment. Doing so, it is possible to work on a the secret branch long before the paying transaction is placed. In order to investigate the double-spending problem under such realistic assumptions, the success probability and the costs of sufficiently more sophisticated attacks must be determined, which requires numerical solutions to complex optimal stochastic switching problems. The authors will address these exciting questions in future research.

References

Alfonsi A, Schied A (2010) Optimal trade execution and absence of price manipulations in limit order book models. SIAM J Financial Math 1(1):490–522

Alfonsi A, Schied A, Fruth A (2010) Optimal execution strategies in limit order books with general shape functions. Quantitative Finance 10(2):143–157

Almgren R, Chriss N (2000) Optimal execution of portfolio transactions. Journal of Risk 3(2):5–39

Andersen L, Broadie M (2004) Primal-dual simulation algorithm for pricing multidimensional american options. Management Science 50(9):1222–1234

Bakke I, Fleten S, Hagfors L, Hagspiel V, Norheim B, Wogrin S (2016) Investment in electric energy storage under uncertainty: a real options approach. Computational Management Science 13(3):483–500

Bayraktar E, Ludkovski M (2011) Optimal trade execution in illiquid markets. Mathematical Finance 21(4):681–701

Beaudin M, Zareipour H, Schellenberglabe A, Rosehart W (2010) Energy storage for mitigating the variability of renewable electricity sources: An updated review. Energy for Sustainable Development 14(4):302 – 314

Becherer D, Bilarev T, Frentrup P (2016) Optimal asset liquidation with multiplicative transient price impact. arXiv:150101892v2

Benth F, Cartea A, Kiesel R (2008) Pricing forward contracts in power markets by the certainty equivalence principle: Explaining the sign of the market risk premium. Journal of Banking & Finance 32(10):2006 – 2021

Bertsimas D, Lo AW (1998) Optimal control of execution costs. J Financial Markets 1(1):1–50

Bradbury K, Pratson L, Patiño-Echeverri D (2014) Economic viability of energy storage systems based on price arbitrage potential in real-time u.s. electricity markets. Applied Energy 114:512 – 519

Brennan M, Schwartz E (1985) Evaluating natural resource investments. The Journal of Business 58(2):135–57

Breton S, Moe G (2009) Status, plans and technologies for offshore wind turbines in europe and north america. Renewable Energy 34(3):646 – 654

Cartea A, Villaplana P (2008) Spot price modeling and the valuation of electricity forward contracts: The role of demand and capacity. Journal of Banking & Finance 32(12):2502 – 2519

Chan P, Walter T (2014) Investment performance of "environmentally-friendly" firms and their initial public offers and seasoned equity offers. Journal of Banking & Finance 44:177 – 188

Cont R, Kukanov A (2014) Optimal order placement in limit order markets. arXiv:12101625v4

Cont R, Kukanov A, Stoikov S (2014) The price impact of order book events. Journal of Financial Econometrics 12(1):47

Dincer F (2011) The analysis on photovoltaic electricity generation status, potential and policies of the leading countries in solar energy. Renewable and Sustainable Energy Reviews 15(1):713 – 720

Doege J, Fehr M, Hinz J, Luthi H, Wilhelm M (2009) Risk management in power markets: The hedging value of production flexibility. European Journal of Operational Research 199(3):936 – 943

Dokuchaev N (2016) Optimal energy storing and selling in continuous time stochastic multi-battery setting. arXiv:1511.01365

Díaz-González F, Sumper A, Gomis-Bellmunt O, Villafáfila-Robles R (2012) A review of energy storage technologies for wind power applications. Renewable and Sustainable Energy Reviews 16(4):2154 – 2171

Evans A, Strezov V, Evans T (2012) Assessment of utility energy storage options for increased renewable energy penetration. Renewable and Sustainable Energy Reviews 16(6):4141 – 4147

Gruenspan C, Perez-Marco R (2017) Double spend races. Working paper

Hinz J, Yee J (2017) rcss: Convex switching systems. URL https://github.com/YeeJeremy/rcss, available at https://github.com/YeeJeremy/rcss, R package version 1.5

Horst H, Naujokat F (2014) When to cross the spread? trading in two-sided limit order books. SIAM J Financial Math 5(1):278–315

Kempener R, Borden E (2015) Battery storage for renewables: Market status and technology outlook

Kim J, Powell W (2011) Optimal energy commitments with storage and intermittent supply. Operations Research 59(6):1347 – 1360

Kratz P, Schoneborn T (2016) Optimal liquidation and adverse selection in dark pools. Mathematical Finance pp n/a–n/a, DOI 10.1111/mafi.12126

Locatelli G, Invernizzi D, Mancini M (2016) Investment and risk appraisal in energy storage systems: A real options approach. Energies 104:114 – 131

Longstaff F, Schwartz E (2001) Valuing American options by simulation: a simple least-squares approach. Review of Financial Studies 14(1):113–147, DOI 10.1093/rfs/14.1.113

Lu C, Xu H, Pan X, Song J (2014) Optimal sizing and control of battery energy storage system for peak load shaving. Energies 7(12):8396–8410

Luo X, Wang J, Dooner M, Clarke J (2015) Overview of current development in electrical energy storage technologies and the application potential in power system operation. Applied Energy 137:511 – 536

Meinshausen N, Hambly B (2004) Monte carlo methods for the valuation of multiple-exercise options. Mathematical Finance 14(4):557–583

Nakomoto S (2008) A peer-to-peer electronic cash system. Working paper

Obizhaeva A, Wang J (2013) Optimal trading strategy and supply/demand dynamics. J Financial Markets 16(1):1–32

Oudalov A, Chartouni D, Ohler C (2007) Optimizing a battery energy storage system for primary frequency control. IEEE Transactions on Power Systems 22(3):1259–1266

Paschke R, Propkopczuk M (2010) Commodity derivatives valuation with autoregressive and moving average components in the price dynamics. Journal of Banking and Finance 34(1):2742–2752

Ramiah V, Martin B, Moosa I (2013) How does the stock market react to the announcement of green policies? Journal of Banking & Finance 37(5):1747 – 1758

Renneboog L, Horst J, Zhang C (2008) Socially responsible investments: Institutional aspects, performance, and investor behavior. Journal of Banking & Finance 32(9):1723 – 1742

Rosenfeld M (2014) Analysis of hashrate-based double spending. Working paper

Sarkar S (2003) The effect of mean reversion on investment under uncertainty. Journal of Economic Dynamics and Control 28(2):377–396

Schachter J, Mancarella P (2016) A critical review of real options thinking for valuing investment flexibility in smart grids and low carbon energy systems. Renewable and Sustainable Energy Reviews 56:261 – 271

Schwartz E (1997) The stochastic behavior of commodity prices: Implications for valuation and hedging. The Journal of Finance 52(3):923–973

Teleke S, Baran M, Bhattacharya S, Huang A (2010) Optimal control of battery energy storage for wind farm dispatching. IEEE Transactions on Energy Conversion 25(3):787–794

Tsekrekos A (2010) The effect of mean reversion on entry and exit decisions under uncertainty. Journal of Economic Dynamics and Control 34(4):725–742

Yang Y, Li H, Aichhorn A, Zheng J, Greenleaf M (2014) Sizing strategy of distributed battery storage system with high penetration of photovoltaic for voltage regulation and peak load shaving. IEEE Transactions on Smart Grid 5(2):982 – 991

Chapter 5

Hidden Markov model

5.1 Setting

Partially observable Markov decision processes (Monahan, 1982; Hauskrecht, 2000) or POMDPs appear frequently in decision making under uncertainty since they allow for a convenient way to model states that the controller cannot observe directly without noise. This framework has been addressed by many authors such as Rhenius (1974); Yushkevich (1976); Hernandez-Lerma (1989); Feinberg et al (2016); Bauerle and Rieder (2011). In the following, we consider a decision problem subject to a hidden Markov model. Suppose a stochastic process $(Y_t)_{t=0}^T$ is driven solely by a background device that operates in different regimes. Assume that the operating regime is not directly observed and evolves as a Markov chain $(E_t)_{t=0}^T$ on a finite space which is identified, for simplicity, with the set $\{e_1, \dots, e_d\}$ of unit vectors in $\mathbb{R}^d$. In some applications, the hidden process $(E_t)_{t=0}^T$ describes the evolution of latent variables. It turns out that it is possible to roughly trace the evolution of the hidden states indirectly based on the observation of $(Y_t)_{t=0}^T$ by using efficient recursive schemes for calculation of the so-called *hidden state estimate* given by

$$\widehat{E}_t := \mathbb{E}[E_t \mid Y_j, j \leq t] \qquad t = 0, \dots T.$$

At each time $t = 0, \dots, T-1$, the probability vector $\widehat{E}_t$ describes the distribution of E_t conditioned on the past observations $(Y_j)_{j=0}^t$. Although $(Y_t)_{t=0}^T$ may not be Markovian in general, it turns out (see Yushkevich (1976)) that the observations $(Y_t)_{t=0}^T$ equipped with latent variables $(\widehat{E}_t)_{t=0}^T$ form a two-component process where the evolution $(\widehat{E}_t, Y_t)_{t=0}^T$ is Markovian. This will be shown in the next section. The next section demonstrates how the above control problem involving a hidden Markov model can be solved using the subgradient approach under linear state dynamics. This is surprising since the dynamics under partial observation involves a Bayesian information update, which introduces a non-linearity by renormalization.

5.2 Subgradient approach

Recall that the stochastic process $(E_t)_{t=0}^T$ evolves on the set $\mathbf{E} = \{e_1, \ldots, e_d\}$ representing unit vectors in $\mathbb{R}^d$. The controller also fully observes a signal process $(Y_t)_{t=0}^T$ which moves in some space $\mathbf{Y}$. In our formulation, the signal Y_{t+1} reveals some information on the location of E_t but not E_{t+1}. As before, the agent is subjected to the discrete process $(P_t)_{t=0}^T$ which involves as a Markov chain taking only a finite number of possible values in $\mathbf{P}$ with transition probabilities $(\alpha^a_{p,p'})_{a\in\mathbf{A},p,p'\in\mathbf{P}}$. It is assumed that joint process $(P_t, E_t, Y_t)_{t=0}^T$ follows Markov stochastic transition kernels $\mathcal{Q}_t^a$ for $t = 0, \ldots, T-1$ and action $a \in \mathbf{A}$. Now the transition operator Q_t^a associated with this Markov transition kernel acts on functions $\phi : \mathbf{P} \times \mathbf{E} \times \mathbf{Y} \to \mathbb{R}$ by

$$\begin{aligned} Q_t^a \phi(p, e, y) &= \int \phi(p', e', y') \mathcal{Q}_t^a(\mathrm{d}(p', e', y') | (p, e, y)) \\ &= \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \sum_{e' \in \mathbf{E}} \int_{\mathbf{Y}} \phi(p', e', y') \Gamma_{e,e'} \mu_e(\mathrm{d}y'), \end{aligned} \tag{5.2.1}$$

where the transition probability $\Gamma = (\Gamma_{e,e'})_{e,e'\in\mathbf{E}}$ describes the transition from $E_t = e$ to $E_{t+1} = e'$, $\alpha^a_{p,p'}$ represents the transition probability for P_{t+1} from p to p', and μ_e gives the distribution of Y_{t+1} conditioned on $E_t = e$. Assume that the distribution μ_e is absolutely continuous with respect to some measure μ on $\mathbf{Y}$, i.e. there exist densities

$$\nu_e(y) = \frac{\mathrm{d}\mu_e}{\mathrm{d}\mu}(y), \quad y \in \mathbf{Y}, \ \ e \in \mathbf{E}. \tag{5.2.2}$$

Now introduce the *hidden state estimate process* $(\widehat{E}_t)_{t=0}^T$:

$$\widehat{E}_t := \mathbb{E}[E_t \mid Y_i, i \le t] \qquad t = 0, \ldots T, \tag{5.2.3}$$

which takes values in the set $\widehat{\mathbf{E}}$ representing all convex combinations of the unit vectors $\{e_1, \ldots, e_d\}$. It has been shown by Yushkevich (1976) that the joint process $(P_t, \widehat{E}_t, Y_t)_{t=0}^T$ also follows Markov transition kernels which we will denote by $\widehat{\mathcal{Q}}_t^a$ for $t = 0, \ldots, T-1$ and action $a \in \mathbf{A}$. Let us explore the intuition of this important concept further. Given an action a at time t, we wish to show that $(P_{t+1}, \widehat{E}_{t+1}, Y_{t+1})$ depends only on $(P_t, \widehat{E}_t, Y_t)$ and no other past history. By definition, given action a at time t, P_{t+1} depends only on P_t. Since $(\widehat{E}_t, Y_t)_{t=0}^T$ do not depend on the action, we are left to show that $(\widehat{E}_t, Y_t)_{t=0}^T$ is a Markovian process. Now by definition in (5.2.3), $(\widehat{E}_t)_{t=0}^T$ is Markovian since the conditional expectation in (5.2.3) basically represents an accumulation all observable history. Finally, recall that random variable Y_{t+1} is

soley driven by unobservable E_{t+1} and $\widehat{E}_t$ accumulates all past oberservable information to estimate E_{t+1}. Thus,

$$\mathbb{P}(Y_{t+1} \in \mathrm{d}y | \widehat{E}_0, Y_0, \ldots, \widehat{E}_t, Y_t) = \mathbb{P}(Y_{t+1} \in \mathrm{d}y | \widehat{E}_t).$$

Combining all of the above shows that $(P_t, \widehat{E}_t, Y_t)_{t=0}^T$ follows Markov evolution given the actions.

Using (5.2.1), (5.2.2) and (5.2.3), one can represent the transition operator $\widehat{Q}_t^a$ associated with the Markov kernels $\widehat{\mathcal{Q}}_t^a$ for the process $(P_t, \widehat{E}_t, Y_t)_{t=0}^{T-1}$ and acting on functions $\phi : \mathbf{P} \times \widehat{\mathbf{E}} \times \mathbf{Y} \to \mathbb{R}$ by

$$\begin{aligned}\widehat{Q}_t^a \phi(p, \widehat{e}, y) &= \int_{\widehat{\mathbf{E}} \times \mathbf{Y}} \phi(p', \widehat{e}', y') \widehat{\mathcal{Q}}_t^a(\mathrm{d}(p', \widehat{e}', y') | (p, \widehat{e}, y)) \\ &= \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \int_{\mathbf{Y}} \phi\left(p', \frac{\Gamma^\top \mathcal{V}(y') \widehat{e}}{\|\mathcal{V}(y') \widehat{e}\|}, y'\right) \|\mathcal{V}(y') \widehat{e}\| \mu(\mathrm{d}y'),\end{aligned}$$

where $\mathcal{V}(y)$ defines the diagonal matrix whose diagonal elements are given by $(\nu_e(y))_{e \in \mathbf{E}}$ for $y \in \mathbf{Y}$ and $\|\cdot\|$ now gives the L^1 norm.

In the following, the reward and scrap will not explicitly depend on the unobservable state. Suppose that the reward and scrap functions are given by

$$r_t : \mathbf{P} \times \widehat{\mathbf{E}} \times \mathbf{A} \to \mathbb{R}, \qquad r_T : \mathbf{P} \times \widehat{\mathbf{E}} \to \mathbb{R}, \quad t = 0, \ldots, T-1. \tag{5.2.4}$$

Here, the reward and scrap values do not explicitly depend on the signal y. However, the signal $(Y_t)_{t=0}^T$ indirectly influences the expectation of the next-step reward through its impact on $(\widehat{E}_t)_{t=0}^T$ via (5.2.3). With this, the resulting Bellman recursion is given by

$$\begin{aligned}v_T^*(p, \widehat{e}) &= r_T(p, \widehat{e}), \\ v_t^*(p, \widehat{e}) &= \max_{a \in \mathbf{A}} \left(r_t(p, \widehat{e}, a) + \widehat{Q}_t^a v_{t+1}^*(p, \widehat{e}) \right),\end{aligned}$$

for $p \in \mathbf{P}$ and $\widehat{e} \in \widehat{\mathbf{E}}$ where the kernel $\widehat{Q}_t^a$ acts on function $\phi : \mathbf{P} \times \widehat{\mathbf{E}} \times \mathbf{Y} \to \mathbb{R}$ by

$$\widehat{Q}_t^a \phi(p, \widehat{e}, y) = \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \int_{\mathbf{Y}} \phi\left(p', \frac{\Gamma^\top \mathcal{V}(y') \widehat{e}}{\|\mathcal{V}(y') \widehat{e}\|}, y'\right) \|\mathcal{V}(y') \widehat{e}\| \mu(\mathrm{d}y'). \tag{5.2.5}$$

Before presenting the following lemma and theorem, let us first introduce the concept of *function extension*. Given functions $r_t : \mathbf{P} \times \widehat{\mathbf{E}} \times \mathbf{A} \to \mathbb{R}$ and $\phi : \mathbf{P} \times \widehat{\mathbf{E}} \times \mathbf{Y} \to \mathbb{R}$, define their

positive-homogeneous extensions $\widetilde{r}_t : \mathbf{P} \times \mathbb{R}^d_+ \times \mathbf{A} \to \mathbb{R}$ and $\widetilde{\phi} : \mathbf{P} \times \mathbb{R}^d_+ \to \mathbb{R}$ by

$$\widetilde{r}_t(p,z,a) := \|z\| r_t\left(p, \frac{z}{\|z\|}, a\right) \quad \text{and} \quad \widetilde{\phi}(p,z) := \|z\| \phi\left(p, \frac{z}{\|z\|}, y\right)$$

for $p \in \mathbf{P}$, $z \in \mathbb{R}^d_+$, $a \in \mathbf{A}$, and $y \in \mathbf{Y}$.

Lemma 5.1. *For $t = 0, \dots, T-1$, define the transition operator K^a_t by*

$$K^a_t \widetilde{\phi}(p,z) = \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \mathbb{E}[\widetilde{\phi}(p', W_{t+1} z)], \qquad p \in \mathbf{P}, \quad z \in \mathbb{R}^d_+, \quad a \in \mathbf{A},$$

where the random matrix W_{t+1} is defined as $W_{t+1} = \Gamma^\top \mathcal{V}(Y_{t+1})$. It holds that $K^a_t \widetilde{\phi}$ is a positive-homogenous extension of $\widehat{Q}^a_t \phi$ for $a \in \mathbf{A}$, and $t = 0, \dots, T-1$.

Proof. This has been proven by Lemma 1 in Hinz and Yee (2017). However, for completeness, the following presents the proof. From (5.2.5), taking the positive-homogenous extension of $\widehat{Q}^a_t \phi$ gives us

$$\begin{aligned}
\|z\| \widehat{Q}^a_t \phi\left(p, \frac{z}{\|z\|}, y\right) &= \|z\| \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \int_{\mathbf{Y}} \phi\left(p', \frac{\Gamma^\top \mathcal{V}(y') z}{\|\mathcal{V}(y') z\|}, y'\right) \frac{\|\mathcal{V}(y') z\|}{\|z\|} \mu(\mathrm{d}y') \\
&= \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \int_{\mathbf{Y}} \widetilde{\phi}\left(p', \Gamma^\top \mathcal{V}(y') z\right) \|\Gamma^\top \mathcal{V}(y') z\|^{-1} \|\mathcal{V}(y') z\| \mu(\mathrm{d}y') \\
&= \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \int_{\mathbf{Y}} \widetilde{\phi}\left(p', \Gamma^\top \mathcal{V}(y') z\right) \mu(\mathrm{d}y') \\
&= \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \mathbb{E}[\widetilde{\phi}(p', W_{t+1} z)] = K^a_t \widetilde{\phi}(p,z)
\end{aligned}$$

for $z \in \mathbb{R}^d_+$, $p \in \mathbf{P}$, $a \in \mathbf{A}$, $t = 0, \dots, T-1$. □

The positive-homogenous extension above allows us to induce linear state dynamics into our decision problem.

Theorem 5.1. *Consider the extended value functions $(\widetilde{v}^*_t)_{t=0}^T$ obtained recursively by*

$$\begin{aligned}
\widetilde{v}^*_T(p,z) &= \widetilde{r}_T(p,z), \\
\widetilde{v}^*_t(p,z) &= \max_{a \in \mathbf{A}} \left(\widetilde{r}_t(p,z,a) + K^a t \widetilde{v}^*_{t+1}(p,z)\right),
\end{aligned}$$

for $p \in \mathbf{P}$, $z \in \mathbb{R}^d_+$, and $t = T-1, \dots, 0$. It holds that $\widetilde{v}^*_t$ is the positive-homogeneous extension of v^*_t for all $t = 0, \dots, T$.

Proof. This has been proven by Proposition 1 in Hinz and Yee (2017). However, for completeness, the following presents the proof. We prove this inductively. At $t = T$, the statement obviously holds. At $t = T-1$, we know from Lemma 5.1 that $K_t^a \widetilde{v}_T$ is the positive-homogeneous extension of $\widehat{Q}_t^a v_T$ for each $a \in \mathbf{A}$. Now it is not hard to see that the summations and maximizations of positive-homogenous extensions also results in a positive-homogenous extension. Therefore, $\widetilde{v}_t^*$ is the positive-homogeneous extension of v_t^*. Procededing inductively for $t = T-2, \ldots, 0$ gives the desired result. □

Finally, to employ the subgradient approach to approximate the positive-homogeneous extended value functions in Theorem 5.1, one requires the following assumption:

$$\begin{gathered}\text{positive-homogeneous extenstions } (\widetilde{r}_t(p,z,a))_{t=0}^{T-1} \text{ and } \widetilde{r}_T(p,z) \text{ for} \\ p \in \mathbf{P}\ a \in \mathbf{A} \text{ are convex and globally Lipschitz continuous in } z.\end{gathered}$$

With the above, the subgradient approach can now be applied in a straightforward manner.

5.3 Solution diagnostics

In what follows, the solution diagnostics approach presented in Chapter 2 is adapted to our new problem setting. Like in Chapter 2, introduce zero mean random variables $\varphi_t(p,z,a)$ for $p \in \mathbf{P}$, $z \in \mathbb{R}_+^d$, $a \in \mathbf{A}$, and $t = 1, \ldots, T$. Given $\varphi = (\varphi_t)_{t=1}^T$ and a policy $\pi = (\pi_t)_{t=0}^{T-1}$, introduce random functions $\overline{\widetilde{v}}_t^{\varphi}$ and $\underline{\widetilde{v}}_t^{\pi,\varphi}$ which are defined recursively for $t = T, \ldots, 1$ by

$$\begin{aligned}\overline{v}_T^{\varphi}(p,z) &= \widetilde{r}_T(p,z) \\ \overline{v}_t^{\varphi}(p,z) &= \max_{a \in \mathbf{A}} \left(\widetilde{r}_t(p,z,a) + \varphi_{t+1}(p,z,a) + \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \overline{v}_{t+1}^{\varphi}(p', W_{t+1} z) \right)\end{aligned}$$

and

$$\begin{aligned}\underline{v}_T^{\pi,\varphi}(p,z) &= \widetilde{r}_T(p,z) \\ \underline{v}_t^{\pi,\varphi}(p,z) &= \widetilde{r}_t(p,z,\pi_t(p,z)) + \varphi_{t+1}(p,z,\pi_t(p,z)) + \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^{\pi_t(p,z)} \underline{v}_{t+1}^{\pi,\varphi}(p', W_{t+1} z).\end{aligned}$$

Theorem 5.2. *At $t = T-1, \ldots, 0$,*

$$\mathbb{E}[\underline{v}_t^{\pi,\varphi}(p,z)] \leq \widetilde{v}_t^*(p,z) \leq \mathbb{E}[\overline{v}_t^{\varphi}(p,z)]$$

for all $p \in \mathbf{P}$ *and* $z \in \mathbb{R}^d_+$. *Moreover, if*

$$\varphi_t^*(p,z,a) = \sum_{p' \in \mathbf{P}} \alpha^a_{p,p'} \mathbb{E}[\widetilde{v}_t^*(p', W_t z)] - \widetilde{v}_t^*(p', W_t z)$$

then

$$\mathbb{E}[\underline{v}_t^{\pi^*,\varphi^*}(p,z)] = \widetilde{v}_t^*(p,z) = \mathbb{E}[\overline{v}_t^{\varphi^*}(p,z)]$$

for $p \in \mathbf{P}$, $z \in \mathbb{R}^d_+$ *and* $t = T-1, \ldots, 0$.

Proof. This is proven in the same manner as in Theorem 2.5. □

The below outlines a stylized implementation of the above theorem similar to Chapter 1. There is a slight modification to account for the generation of the signal process using the reference probability measure in Step 1 below. Note that the reference probability measure used to generate the signals coincides with (5.2.2).

Primal-dual estimation

Suppose we are given extended value function approximations $(\widetilde{v}_t)_{t=0}^T$ and candidate policy $(\pi_t)_{t=0}^{T-1}$ from (2.2.2).

1) Chose a number $I \in \mathbb{N}$ of Monte-Carlo trials and obtain for $k = 1, \ldots, I$ independent realizations $(Y_t(\omega_i))_{t=1}^T$.
2) Starting at $z_0^i := \widehat{e}_0 \in \widehat{\mathbf{E}}$, define for $i = 1, \ldots, I$ the trajectories $(z_t^i)_{t=0}^T$ recursively

$$z_{t+1}^i = \Gamma^\top \mathcal{V}(Y_{t+1}(\omega_i)) z_t^i, \qquad t = 0, \ldots, T-1$$

4) Determine the realizations

$$\varphi_{t+1}(p, z_t^i, a)(\omega_i), \qquad t = 0, \ldots, T-1, \quad i = 1, \ldots, I.$$

of random variables

$$\sum_{p \in \mathbf{P}} \alpha^a_{p,p'} \frac{1}{I'} \sum_{i'=1}^{I'} \widetilde{v}_{t+1}\left(p', \Gamma^\top \mathcal{V}(Y_{t+1}^{(i')}) z\right) - \widetilde{v}_{t+1}\left(p', \Gamma^\top \mathcal{V}(Y_{t+1}) z\right).$$

with $I' \in \mathbb{N}$, $p \in \mathbf{P}$, $z \in \mathbb{R}^d_+$, and where $(Y_{t+1}^{(i')})_{i'=1}^{I'}$ are independent and identically distributed copies of Y_{t+1} obtained in Step 1.

5) For each $i = 1, \dots, I$ initialize the recursion at $t = T$ as

$$\overline{v}_T^{\varphi}(p, z_T^i) = \widetilde{r}_T(p, z_T^i)$$
$$\underline{v}_T^{\pi,\varphi}(p, z_T^i) = \widetilde{r}_T(p, z_T^i)$$

for all $p \in \mathbf{P}$ and continue for $t = T-1, \dots, 0$ by

$$\overline{v}_t^{\varphi}(p, z_t^i) = \max_{a \in \mathbf{A}} \left(\widetilde{r}_t(p, z_t^i, a) + \varphi_{t+1}(p, z_t^i, a)(\omega_i) + \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^a \overline{v}_{t+1}^{\varphi}(p', z_{t+1}^i) \right)$$
$$\underline{v}_t^{\pi,\varphi}(p, z_t^i) = \widetilde{r}_t(p, z_t^i, \pi_t(p, z_t^i)) + \varphi_{t+1}(p, z_t^i, \pi_t(p, z_t^i))(\omega_k) + \sum_{p' \in \mathbf{P}} \alpha_{p,p'}^{\pi_t(p,z_t^i)} \underline{v}_{t+1}^{\varphi}(p', z_{t+1}^i).$$

5) Determine the sample means $\frac{1}{I}\sum_{i=1}^I \overline{v}_0^{\varphi}(p, z_0^i)$, $\frac{1}{I}\sum_{i=1}^I \underline{v}_0^{\varphi}(p, z_0^i)$ to estimate $v_0^*(p, z_0)$ from above and below possibly using confidence bounds.

5.4 Point & Figure

This section considers the optimization of an investment strategy for a single risky asset under the assumption that the increments of the sampled asset price process follows a hidden Markov dynamics (see Elliott and Hinz (2002)). To obtain such price evolution, we introduce a random time sampling of the *continuous* price process $(S_t)_{t \geq 0}$, which is inspired by the so-called *Point&Figure Chart* technique. Suppose trading shall occur only at times where a notable price change may require a position re-balancing. Thereby, the price evolution is sampled as follows: Having fixed a price change step $\Delta > 0$ and starting the observations at the initial time $\tau_0 = 0$, one writes into a *Point&Figure Chart* one of the symbols **x** or **o** at the first time τ_1 where the asset price leaves the interval $[S_{\tau_0} - \Delta, S_{\tau_0} + \Delta]$. If the price increases to the upper bound $S_{\tau_0} + \Delta$ one writes the symbol **x**, otherwise the symbol **o** is written. Repeating the same procedure with the next interval $[S_{\tau_1} - \Delta, S_{\tau_1} + \Delta]$ and proceeding further, a sequence of stopping times $(\tau_k)_{k \in \mathbb{N}}$ is determined, with the symbols **x** or **o** at each time, which are arranged in a diagram as shown in Figure 5.4.1.

Assume that the trading occurs only at $(\tau_k)_{k \in \mathbb{N}}$, and that at each time τ_k the trading decision is based only on the observation of the sampled price history $S_{\tau_0}, \dots, S_{\tau_k}$. Since the price process $(S_t)_{t \geq 0}$ is continuous, the stochastic driver of our model is given by the binary increment process

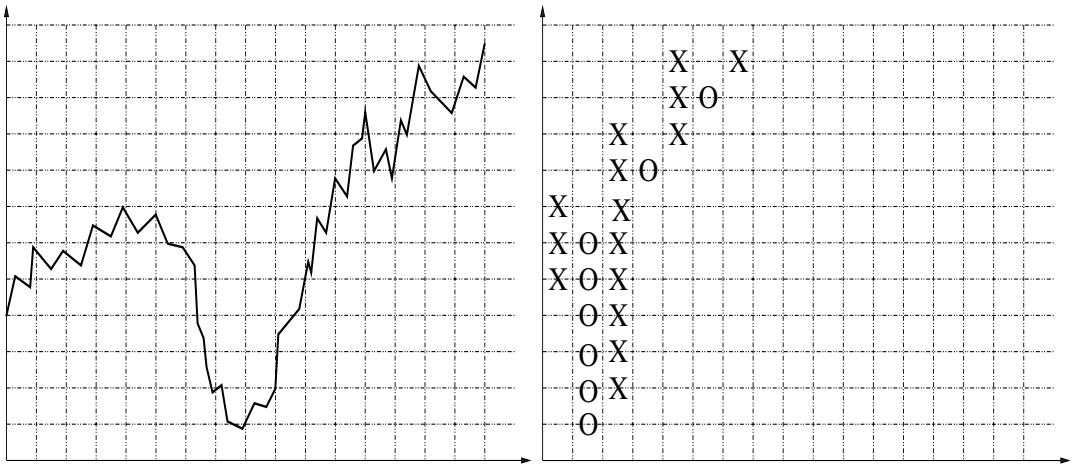

Fig. 5.4.1: Asset price and its Point & Figure chart.

$$Y_t = S_{\tau_t} - S_{\tau_{t-1}}, \qquad t = 1, \dots, T,$$

which takes values in the set $\mathbf{Y} = \{-\Delta, \Delta\}$. This process is modeled as the observable part $(Y_t)_{t=1}^T$ of a hidden Markov dynamics $(E_t, Y_t)_{t=0}^T$. For the sake of concreteness, we suppose that the hidden regimes $\mathbf{E}$ can be identified with some background market situations. As a simple illustration, we consider a two-state $\mathbf{E} = \{e_1, e_2\}$ regime switching with transition matrix

$$\Gamma = \begin{bmatrix} p_1 & (1-p_1) \\ (1-p_2) & p_2 \end{bmatrix}$$

and assume that if the market is in the state $E_t = e_1$ then the next price increment Y_{t+1} takes values in $\mathbf{Y} = \{-\Delta, \Delta\}$ with probabilities q_1 and $(1-q_1)$ respectively. Similarly, conditioned on the current state $E_t = e_2$ we have the probabilities $(1-q_2)$ and q_2 for the observation Y_{t+1} of the next price move. Choosing the reference measure μ as the uniform distribution on $\mathbf{Y}$ by $\mu(\{\Delta\}) = \mu(\{-\Delta\}) = 1/2$, we obtain the following diagonal density matrices

$$\mathcal{V}(-\Delta) = 2 \begin{bmatrix} q_1 & 0 \\ 0 & 1-q_1 \end{bmatrix}, \qquad \mathcal{V}(\Delta) = 2 \begin{bmatrix} 1-q_2 & 0 \\ 0 & q_2 \end{bmatrix},$$

which gives merely two disturbance matrix realizations $\Gamma^\top \mathcal{V}(-\Delta)$. According to Theorem 5.1, we define the disturbance matrices by $(W_t = \Gamma^\top \mathcal{V}(Y_t))_{t=1}^T$, using independent identically distributed random variables $(Y_t)_{t=1}^T$, whose distribution is the reference measure μ.

Now, we introduce the position control for our dynamic asset allocation problem. Consider a situation where the asset position can either be short, neutral, or long, labeled by the numbers $p = 1,2,3$ respectively. At each time $t = 0,\dots,T$, the controller must make a decision whether the next position shall be short, neutral, or long. Given the set $\mathbf{P} = \{1,2,3\}$ of all possible positions, we introduce the action set as $\mathbf{A} = \{1,2,3\}$ where a stands for the targeted position after re-allocation. The transition probabilities for P_t is therefore given by

$$\alpha^a_{p,p'} = 1 \qquad \text{if } a = p'$$

or more conveniently as the position control function α determined by the following matrix:

$$\begin{bmatrix} \alpha(1,1)\ \alpha(1,2)\ \alpha(1,3) \\ \alpha(2,1)\ \alpha(2,2)\ \alpha(2,3) \\ \alpha(3,1)\ \alpha(3,2)\ \alpha(3,3) \end{bmatrix} = \begin{bmatrix} 1\ 2\ 3 \\ 1\ 2\ 3 \\ 1\ 2\ 3 \end{bmatrix}$$

with the interpretation that $\alpha(p,a)$ stands for the next position after action a is chosen at position p. Finally, let us turn to the definition of the reward and the scrap functions. In this example, we model the payoff in terms of affine linear reward function

$$r_t(p,\widehat{e},a) = r(p,\widehat{e},a) = (p-1)\rho^\top \widehat{e} - c|p-\alpha(p,a)|, \quad t = 0,\dots T-1 \tag{5.4.1}$$

for all $e \in \widehat{\mathbf{E}}$, $a \in \mathbf{A}$, and where $\rho = \Delta[1-2q_1, 2q_2-1]^\top$. Each component in the vector ρ describes the next expected price change depending on a market state. Here, $c(p-\alpha(p,a))$ represents the proportional transaction costs determined by a parameter $c > 0$ and the term $(p-1)\rho^\top \widehat{e}$ stands for the expected revenue from holding position p from time t to $t+1$, if the distribution of the market state is described by the probability vector $\widehat{e} \in \widehat{\mathbf{E}}$. Assuming that at the end $t = T$, all asset positions must be closed, we define the scrap value as

$$r_T(p,\widehat{e}) = r(p,\widehat{e},2) \quad \text{for } t = 0,\dots,T-1,\ p \in \mathbf{P},\ \widehat{e} \in \widehat{\mathbf{E}}. \tag{5.4.2}$$

Note that since all entries of the probability vector $\widehat{e} \in \widehat{\mathbf{E}}$ sum up to one i.e. $\mathbf{1}^\top \widehat{e} = 1$, the constant transaction cost term in (5.4.1) can be included for $t = 0,\dots,T-1$, $p \in \mathbf{P}$, $\widehat{e} \in \widehat{\mathbf{Z}}$ as

$$r_t(p,\widehat{e},a) = R(p,a)\widehat{e}, \quad r_T(p,\widehat{e}) = R(p,1)\widehat{e} \tag{5.4.3}$$

with

$$R(p,a) = ((p-2)\rho - c|p-\alpha(p,a)|\mathbf{1})^\top$$

for $t = 0,\dots,T-1,\ p \in \mathbf{P},\ a \in \mathbf{A}$. Because of this linearity, we observe that the positive-homogeneous extensions are obtained by the same formula

$$\widetilde{r}_t(p,z,a) = R(p,a)z, \quad \widetilde{r}_T(p,z) = R(p,2)z$$

for all $z \in \mathbb{R}^2_+$, $t = 0,\dots,T-1$, $p \in \mathbf{P}$, $a \in \mathbf{A}$.

Let us consider a hidden Markov model with parameters $p_1 = p_2 = 0.8$ and $q_1 = q_2 = 0.9$ generates increments of size $\Delta = 1$ of the asset price for $t = 0,\dots,10$. Having supposed transaction costs $c = 0.5$, we now turn our attention to the specification of our grid $\mathbf{G}^{(m)}$. Since the positive-homogenous extensions of the reward and value functions are defined on $\mathbb{R}^2_+$, an appropriate grid needs to be chosen in order to obtain accurate results. Again, we use the so called *stochastic grid*. First, let us introduce an equally spaced seed grid $\mathbf{G}^{(0)}$ of size 6 by

$$\mathbf{G}^{(0)} = \left\{ \frac{k}{100}e_1 + (1-\frac{k}{100})e_2 : k = 0,\dots,5 \right\} \subset \widehat{\mathbf{E}}.$$

The stochastic grid $G^{(m)}$ is generated as

$$\mathbf{G}^{(m)} = \left\{ \left(\Gamma^\top \mathcal{V}(-\Delta)\right)^i \left(\Gamma^\top \mathcal{V}(-\Delta)\right)^j \hat{e} : i,j = 0,1,...,n, \hat{e} \in \mathbf{G}^{(0)} \right\}$$

where $\left(\Gamma^\top \mathcal{V}(-\Delta)\right)^i$ is the i-th matrix power of $\Gamma^\top \mathcal{V}(-\Delta)$. For the following numerical illustration, we set our grid $\mathbf{G}^{(m)} = \mathbf{G}^{(8)}$. This stochastic grid is shown in the left plot of Figure 5.4.2.

As an illustration we apply this bounds estimation to the adaptive investment strategy. Recall that $(Y_t)_{t=0}^T$ takes on values in the set $\mathcal{Y} = \{-\Delta, \Delta\}$. Let us specify the reference measure μ on $\mathcal{Y}$ by parameters $\mu_1 := \mu(\{-\Delta\})$, $\mu_2 := \mu(\Delta)$ which gives the diagonal matrices are

$$\mathcal{V}(-\Delta) = \frac{1}{\mu_1}\begin{bmatrix} q_1 & 0 \\ 0 & 1-q_1 \end{bmatrix}, \qquad \mathcal{V}(\Delta) = \frac{1}{(1-\mu_1)}\begin{bmatrix} 1-q_2 & 0 \\ 0 & q_2 \end{bmatrix}.$$

Note that the Step 2 of bound estimation requires a sampling with respect to the reference measure μ. Thus $(Y_t)_{t=0}^T$ is constructed from independent identically binary distributed random variables, each taking value $-\Delta$ and Δ with probabilities μ_1 and μ_2 respectively. Because each

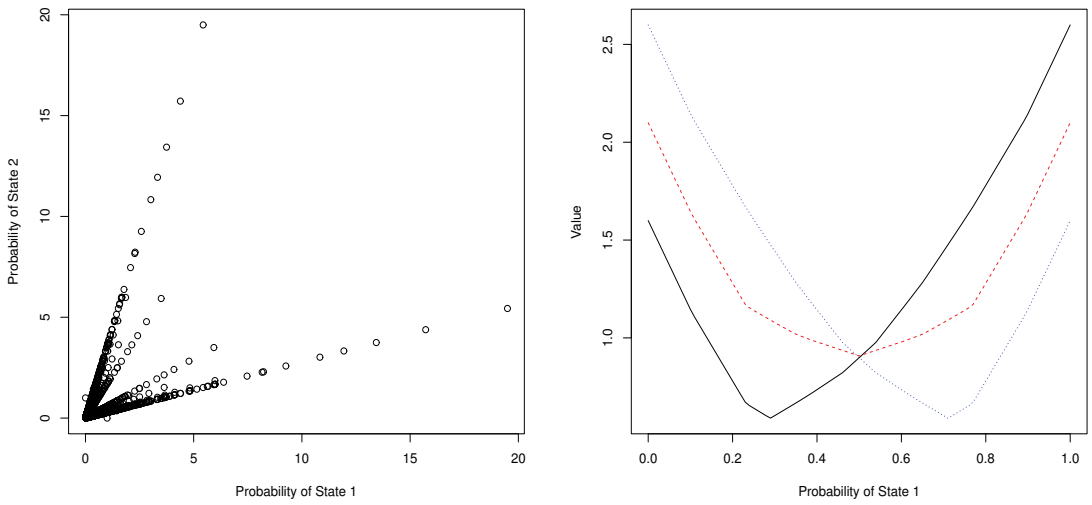

Fig. 5.4.2: Grid (left plot) and value function approximation (right) for $T = 10$.

Y_t takes realizations in a finite set $\{-\Delta, \Delta\}$, the generation of φ_t in Step 4 of the primal-dual estimaion can be obtained more efficiently by the exact calculation of the mean

$$\mu_1 \widetilde{v}_{t+1}(p', \Gamma^\top \mathcal{V}(-\Delta)z) + (1-\mu_1)\widetilde{v}_{t+1}(p', \Gamma^\top \mathcal{V}(\Delta)z)$$

instead of the sample mean

$$\frac{1}{I'}\sum_{i'=1}^{I'} \widetilde{v}_{t+1}\left(p', \Gamma^\top \mathcal{V}(Y_{t+1}^{(i')})z\right).$$

In Table 5.4.2, the first column lists the conditional probability of the hidden state being in the first mode. The second column lists the position number and the third column returns the value given by the value function approximations from the subgradient approach. Using the expectation matrices approach, the value function approximation took around 1 cpu second (1 elapsed) and these value functions for $t = 0$ is given in the right plot of Figure 5.4.2. The fourth column gives the sample mean for the testing of the resulting prescribed policy on 50000 simulated paths. Finally, the last column gives the 99% confidence intervals for the true value given by the prima-dual estimation. The calculation of the primal andf dual values took roughly 2 cpu seconds (1 elapsed). The results in Table 5.4.2 indicate excellent value function approximations.

Table 5.4.1: Subgradient and duality approach for $T = 10$.

State = 1	p	Subgradient	Test	99% CI
0.0	1	1.59950	1.59634 (.02010)	(1.59959, 1.59964)
0.0	2	2.09950	2.09634 (.02010)	(2.09959, 2.09964)
0.0	3	2.59950	2.59634 (.02010)	(2.59959, 2.59964)
0.2	1	0.77984	0.77521 (.01616)	(0.77984, 0.77984)
0.2	2	1.27984	1.27521 (.01616)	(1.27984, 1.27984)
0.2	3	1.77984	1.77521 (.01616)	(1.77984, 1.77984)
0.4	1	0.73544	0.73115 (.01304)	(0.73545, 0.73545)
0.4	2	0.97941	0.97571 (.01285)	(0.97942, 0.97942)
0.4	3	1.14315	1.14020 (.01334)	(1.14315, 1.14316)
0.6	1	1.14315	1.14020 (.01334)	(1.14315, 1.14316)
0.6	2	0.97941	0.97571 (.01285)	(0.97942, 0.97942)
0.6	3	0.73544	0.73115 (.01304)	(0.73545, 0.73545)
0.8	1	1.77984	1.77521 (.01616)	(1.77984, 1.77984)
0.8	2	1.27984	1.27521 (.01616)	(1.27984, 1.27984)
0.8	3	0.77984	0.77521 (.01616)	(0.77984, 0.77984)
1.0	1	2.59950	2.59634 (.02010)	(2.59959, 2.59964)
1.0	2	2.09950	2.09634 (.02010)	(2.09959, 2.09964)
1.0	3	1.59950	1.59634 (.02010)	(1.59959, 1.59964)

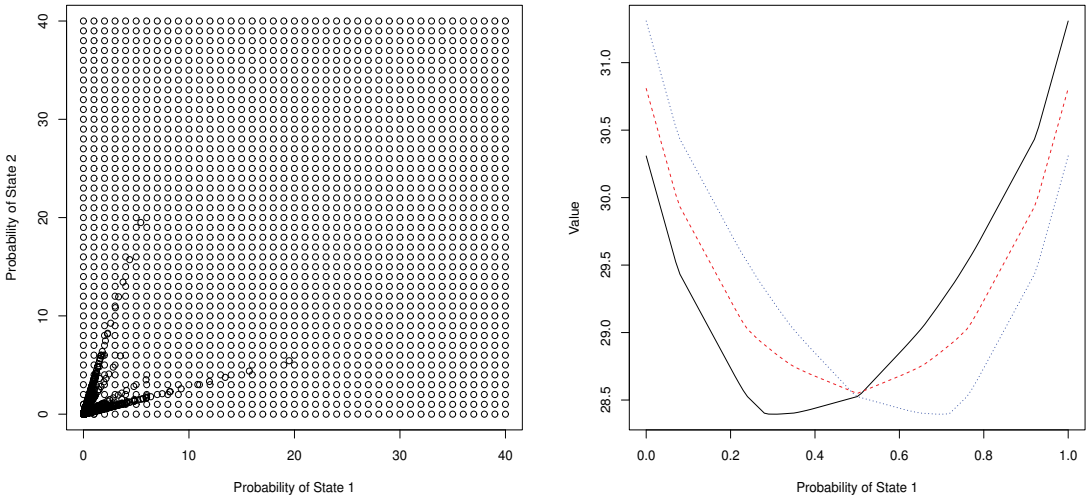

Fig. 5.4.3: Grid (left plot) and value function approximation (right) for $T = 100$.

Let us now increase the time horizon to $T = 100$. To account for the longer time frame, we superimpose the original stochastic grid with a lattice as shown in the left plot of Figure 5.4.3. Using the same number if sample paths for the testing of the prescribed policy and for the primal-dual estimation, Table 5.4.2 indicates that relatively good results are still attainable. In this case, the value function approximation using expectation matrices took around 23 cpu seconds (21 elapsed) and the primal-dual estimation took around 19 cpu seconds (8 elapsed). The resulting value function approximations are illustrated in the right plot of Figure 5.4.3. As a final remark, one could in principle solve this problem using brute force by considering all possible states. That is, for starting z_0, one could trace all possible states it may visit since there are only 2 possible disturbances. For each $z_0 \in \mathbf{Z}$, there are $\frac{1(1-2^{T+1})}{-1}$ possible states $(Z_t)_{t=0}^{T}$ may visit and it is not hard to see that this number becomes impractical for large T. For example, when $T = 100$, one needs to track 2.535301×10^{30} states. In such cases, approximate dynamic programming methods such as ours becomes the only real practical alternative.

Table 5.4.2: Subgradient and duality approach for $T = 100$.

State = 1	p	Subgradient	Backtest	99% CI
0.0	1	30.30855	27.52483 (3.55539)	(30.13080, 32.51606)
0.0	2	30.80855	28.02483 (3.55539)	(30.63080, 33.01606)
0.0	3	31.30855	28.52483 (3.55539)	(31.13080, 33.51606)
0.2	1	28.73752	25.93118 (3.00981)	(28.62106, 30.71315)
0.2	2	29.23752	26.43118 (3.00981)	(29.12106, 31.21315)
0.2	3	29.73752	26.93118 (3.00981)	(29.62106, 31.71315)
0.4	1	28.43930	25.63065 (2.69582)	(28.36940, 30.26692)
0.4	2	28.67700	25.86857 (2.69581)	(28.60712, 30.50454)
0.4	3	28.84834	26.04016 (2.69583)	(28.77844, 30.67605)
0.6	1	28.84834	26.04016 (2.69583)	(28.81039, 30.64637)
0.6	2	28.67700	25.86857 (2.69581)	(28.63906, 30.47493)
0.6	3	28.43930	25.63065 (2.69582)	(28.40135, 30.23737)
0.8	1	29.73752	26.93118 (3.00981)	(29.70739, 31.63284)
0.8	2	29.23752	26.43118 (3.00981)	(29.20739, 31.13284)
0.8	3	28.73752	25.93118 (3.00981)	(28.70739, 30.63284)
1.0	1	31.30855	28.52483 (3.55539)	(31.25620, 33.40328)
1.0	2	30.80855	28.02483 (3.55539)	(30.75620, 32.90328)
1.0	3	30.30855	27.52483 (3.55539)	(30.25620, 32.40328)

5.5 Conclusion

In this chapter, the subgradient and duality approach is extended to decision making under partial information. We demonstrate that the applications of our approach cover a broad range

of decision-making situations and illustrate this aspect on a detailed case study of an asset allocation problem with excellent results.

References

Bauerle N, Rieder U (2011) Markov Decision Processes with applications to finance. Springer, Heidelberg, DOI 10.1007/978-3-642-18324-9

Elliott R, Hinz J (2002) Portfolio optimization, hidden Markov models, and technical analysis of p&f-charts. International Journal of Theoretical and Applied Finance 5(4):1–15

Feinberg E, Kasyanov P, Zgurovsky M (2016) Partially observable total-cost markov decision processes with weakly continuous transition probabilities. Mathematics of Operations Research 41(2):656–681, DOI 10.1287/moor.2015.0746

Hauskrecht M (2000) Value-function approximations for partially observable Markov decision processes. Journal of Artificial Intelligence Research 13:33–94

Hernandez-Lerma O (1989) Adaptive Markov Control Processes. Springer

Hinz J, Yee J (2017) Stochastic switching for partially observable dynamics and optimal asset allocation. International Journal of Control 90(3):553–565

Monahan G (1982) A survey of partially observable Markov decision processes: Theory, models, and algorithms. Management Science 28:1–16

Rhenius D (1974) Incomplete information in markovian decision models. The Annals of Statistics 2(6):1327–1334

Yushkevich A (1976) Reduction of a controlled markov model with incomplete data to a problem with complete information in the case of borel state and control spaces. Theory of Probability and its Applications 21:153–158

Chapter 6

Software

6.1 R package

This chapter concludes this thesis by briefly describing the software used to generate all the numerical results in this thesis. The subgradient and duality approach has been implemented in Hinz and Yee (2017) and the latest version can be found at `https://github.com/YeeJeremy/rcss`. Note that a comprehensive tutorial of the software implementation is beyond the scope of this thesis and we instead point interested users to the software manual that can be found at `https://github.com/YeeJeremy/RPackageManuals/blob/master/rcss-manual.pdf`. The main aim of the software is to give users a quick and easy way to numerically solve the general problem setting presented in Chapter 2 using the subgradient and duality approach. The software Hinz and Yee (2017) is written in a combination of the languages *R* (see R Core Team (2013)) and *C++* via Eddelbuettel and Francois (2011). For large scale applications, the user is strongly recommended to write their own code. This is for two main reasons. First, significant computation effort can be obtained by exploiting specific features in their problems. Secondly, *R*'s memory management issues for large problems are notorious. However, one of the major benefits of implementing these methods in *R* is that the results can be analysed using the vast number of statistical tools avaliable in this language. In this sense, Hinz and Yee (2017) serves as an excellent prototyping tool for large problems.

6.2 Bermudan put

As a demonstration, let us consider a Bermuda put option with strike price 40 that expires in 1 year. The put option is exercisable at 51 evenly spaced time points in the year, which includes the start and end of the year. The following code approximates the value functions

in the Bellman recursion. On a Linux Ubuntu 16.04 with Intel i5-5300U CPU @2.30GHz and 16GB of RAM, the following code takes around 0.2 cpu second and around 0.05 real world seconds.

Listing 6.1: Value function approximation

```
library(rcss)
rate <- 0.06 ## Interest rate
step <- 0.02 ## Time step between decision epochs
vol <- 0.2 ## Volatility of stock price process
n_dec <- 51 ## Number of decision epochs
strike <- 40 ## Strike price
control <- matrix(c(c(1, 1), c(2, 1)), nrow = 2, byrow = TRUE) ## Control
grid <- as.matrix(cbind(rep(1, 301), seq(30, 60, length = 301))) ## Grid
## Disturbance sampling
u <- (rate - 0.5 * vol^2) * step
sigma <- vol * sqrt(step)
condExpected <- function(a, b){
    aa <- (log(a) - (u + sigma^2)) / sigma
    bb <- (log(b) - (u + sigma^2)) / sigma
    return(exp(u + sigma^2 / 2) * (pnorm(bb) - pnorm(aa)))
}
weight <- rep(1 / 1000, 1000)
disturb <- array(0, dim = c(2, 2, 1000))
disturb[1,1,] <- 1
part <- qlnorm(seq(0, 1, length = 1000 + 1), u, sigma)
for (i in 1:1000) {
    disturb[2,2,i] <- condExpected(part[i], part[i+1]) / (plnorm(part[i+1], u, sigma) -
        plnorm(part[i], u, sigma))
}
## Subgradient representation of reward
in_money <- grid[,2] <= strike
reward <- array(0, dim = c(301, 2, 2, 2, n_dec - 1))
reward[in_money,1,2,2,] <- strike
reward[in_money,2,2,2,] <- -1
for (tt in 1:n_dec - 1){
    reward[,,,,tt] <- exp(-rate * step * (tt - 1)) * reward[,,,,tt]
}
## Subgrad representation of scrap
scrap <- array(data = 0, dim = c(301, 2, 2))
scrap[in_money,1,2] <- strike
scrap[in_money,2,2] <- -1
scrap <- exp(-rate * step * (n_dec - 1)) * scrap
## Bellman
r_index <- matrix(c(2, 2), ncol = 2)
bellman <- FastBellman(grid, reward, scrap, control, disturb, weight, r_index)
```

The matrix `grid` represents our choice of grid points where each row represents a point. The 3-dimensional array `disturb` represents our sampling of the disturbances where `disturb[,,i]` gives the i-th sample. Here, we use local averages on a 1000 component partition of the disturbance space. The 5-dimensional array `reward` represents the subgradient approximation with `reward[,,a,p,t]` representing $\mathcal{S}_{\mathbf{G}^{(m)}} r_t(p,.,a)$. The object `bellman` is a list containing the approximations of the value functions and expected value functions for all positions and decision epochs. Please refer to the package manual for the format of the inputs and outputs. To obtain the value function of the Bermuda put option, simply run the plot command below.

Listing 6.2: Option value function

```
plot(grid[,2], rowSums(bellman$value[,,2,1] * grid), type = "l", xlab = "Stock Price", ylab =
    "Option Value")
```

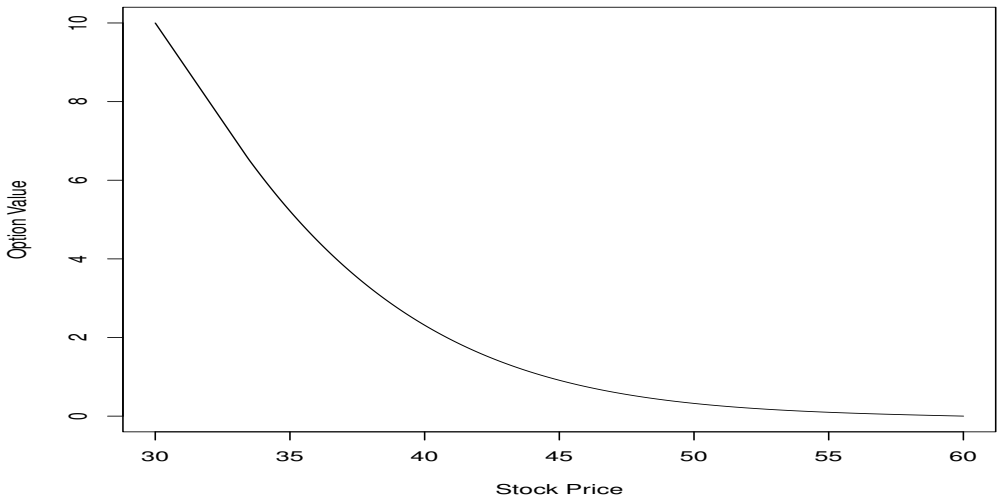

Fig. 6.2.1: Bermuda put value function.

The following code then computes the lower and upper bound estimates for the value of the option when $Z_0 = 36$. On our machine, the following takes around 10 cpu seconds and around 5 real world seconds to run.

Listing 6.3: Lower and upper bounds

```
## Reward function
RewardFunc <- function(state, time) {
    output <- array(data = 0, dim = c(nrow(state), 2, 2))
    output[,2,2] <- exp(-rate * step * (time - 1)) * pmax(40 - state[,2], 0)
    return(output)
}
## Scrap function
ScrapFunc <- function(state) {
    output <- array(data = 0, dim = c(nrow(state), 2))
    output[,2] <- exp(-rate * step * (n_dec - 1)) * pmax(40 - state[,2], 0)
    return(output)
}
## Get primal-dual bounds
start <- c(1, 36)
## Path disturbances
set.seed(12345)
n_path <- 500
path_disturb <- array(0, dim = c(2, 2, n_path, n_dec - 1))
path_disturb[1, 1,,] <- 1
rand1 <- rnorm(n_path * (n_dec - 1) / 2)
rand1 <- as.vector(rbind(rand1, -rand1))
path_disturb[2, 2,,] <- exp((rate - 0.5 * vol^2) * step + vol * sqrt(step) * rand1)
path <- PathDisturb(start, path_disturb)
policy <- FastPathPolicy(path, grid, control, RewardFunc, bellman$expected)
## Subsim disturbances
n_subsim <- 500
subsim <- array(0, dim = c(2, 2, n_subsim, n_path, (n_dec - 1)))
subsim[1,1,,,] <- 1
rand2 <- rnorm(n_subsim * n_path * (n_dec - 1) / 2)
rand2 <- as.vector(rbind(rand2, -rand2))
subsim[2,2,,,] <- exp((rate - 0.5 * vol^2) * step + vol * sqrt(step) * rand2)
subsim_weight <- rep(1 / n_subsim, n_subsim)
mart <- FastAddDual(path, subsim, subsim_weight, grid, bellman$value, ScrapFunc)
bounds <- AddDualBounds(path, control, RewardFunc, ScrapFunc, mart, policy)
```

The above code takes the exact reward and scrap functions as inputs. The function `FastPathPolicy` computes the candidate optimal policy. The object `bounds` is a list containing the primals and duals for each sample path i and each position p at each decision time t. Again, please refer to the package manual for the format of the inputs and outputs. If the price of the underlying asset is 36, the 99% confidence interval for the option price is given by the following.

Listing 6.4: 99% confidence interval

```
> print(GetBounds(bounds, 0.01, 2))
[1] 4.475802 4.480533
```

The package *'rcss'* also allows the user to test the prescribed policy from the Bellman recursion on any supplied set of sample paths. The resulting ouput can then be further studied with time series analysis or other statistical work. In the following code, we will use the previously generated 500 sample paths to backtest our policy and generate histograms.

Listing 6.5: Backtesting Policy

```
test <- FullTestPolicy(2, path, control, RewardFunc, ScrapFunc, policy)
## Histogram of cumulated rewards
hist(test$value, xlab = "Cumulated Rewards", main = "")
## Exercise times
ex <- apply(test$position == 1, 1, function(x) min(which(x)))
ex[ex == Inf] <- 51
ex <- ex - 1
hist(ex, xlab = "Exercise Times", main = "")
```

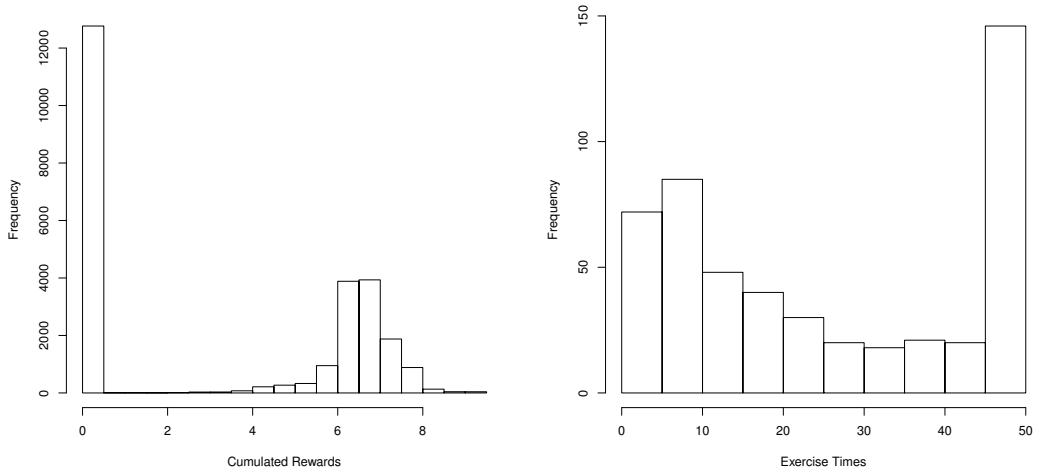

Fig. 6.2.2: Distribution of cumulated rewards and exercise times.

Figure 6.2.2 contains the histograms for the cumulated rewards and exercise times. Let us emphasise the usefulness of such scenario generation. Given an approximately optimal policy

and backtesting, one can perform further statistical analysis on the backtested values to obtain practical insights.

6.3 Swing option

In this subsection, consider a swing option example considered before with 5 rights exercisable on 101 time points. As before, we begin by performing the value function approximation. On our machine, the following code takes around 0.4 cpu seconds or around 0.15 real world seconds to run.

Listing 6.6: Value function approximation

```
library(rcss)
## Parameters
rho <- 0
kappa <- 0.9
mu <- 0
sigma <- 0.5
K <- 0
n_dec <- 101 ## number of time epochs
N <- 5 ## number of rights
n_pos <- N + 1 ## number of positions
grid <- cbind(rep(1, 101), seq(-2, 2, length = 101)) ## Grid
## Control matrix
control <- cbind(c(1, 1:N), 1:(N + 1))
## Reward subgradient representation
reward <- array(0, dim = c(101, 2, 2, nrow(control), n_dec - 1))
slope <- exp(grid[, 2])
for (tt in 1:(n_dec - 1)) {
    discount <- exp(-rho * (tt - 1))
    for (pp in 2:n_pos) {
        intercept <- (exp(grid[,2]) - K * discount) - slope * grid[, 2]
        reward[, 1, 1, pp, tt] <- intercept
        reward[, 2, 1, pp, tt] <- slope
    }
}
## Scrap subgradient representation
scrap <- array(0, dim = c(101, 2, nrow(control)))
discount <- exp(-rho * (n_dec - 1))
for (pp in 2:n_pos) {
    intercept <- (exp(grid[,2]) - K * discount) - slope * grid[, 2]
    scrap[, 1, pp] <- intercept
    scrap[, 2, pp] <- slope
}
## Disturbance sampling
weight <- rep(1/1000, 1000)
disturb <- array(0, dim = c(2, 2, 1000))
disturb[1, 1,] <- 1
disturb[2, 2,] <- 1 - kappa
CondExpected <- function(a, b){
    return(1/sqrt(2 * pi) * (exp(-a^2/2)- exp(-b^2/2)))
}
part <- qnorm(seq(0, 1, length = 1000 + 1))
for (i in 1:1000) {
    disturb[2,1,i] <- kappa * mu + sigma * (CondExpected(part[i], part[i+1]) /
        (pnorm(part[i+1]) - pnorm(part[i])))
}
## Bellman recursion
r_index <- matrix(c(2, 1), ncol = 2)
bellman <- FastBellman(grid, reward, scrap, control, disturb, weight, r_index)
```

After obtaining these function approximations, the following code computes the 99% confidence intervals for the value of a swing option with 5 remaining rights. The code below takes approximately 20 cpu seconds or 10 real world seconds to run.

Listing 6.7: Lower and upper bounds

```
## Exact reward function
RewardFunc <- function(state, time) {
    output <- array(0, dim = c(nrow(state), 2, nrow(control)))
    discount <- exp(-rho * (time - 1))
    for (i in 2:nrow(control)) {
        output[, 1, i] <- pmax(exp(state[, 2]) - K * discount, 0)
    }
    return(output)
}
## Exact scrap function
ScrapFunc <- function(state) {
    output <- array(0, dim = c(nrow(state), nrow(control)))
    discount <- exp(-rho * (n_dec - 1))
    for (i in 2:nrow(control)) {
        output[, i] <- pmax(exp(state[, 2]) - K * discount, 0)
    }
    return(output)
}
## Generate paths
set.seed(12345)
n_path <- 500
path_disturb <- array(0, dim = c(2, 2, n_path, n_dec - 1))
path_disturb[1, 1,,] <- 1
path_disturb[2, 2,,] <- 1 - kappa
rand1 <- rnorm(n_path * (n_dec - 1) / 2)
rand1 <- as.vector(rbind(rand1, -rand1))
path_disturb[2, 1,,] <- kappa * mu + sigma * rand1
start <- c(1, 0)
path <- PathDisturb(start, path_disturb)
policy <- FastPathPolicy(path, grid, control, RewardFunc, bellman$expected)
## Set subsimulation disturbances
n_subsim <- 500
subsim <- array(0, dim = c(2, 2, n_subsim, n_path, n_dec - 1))
subsim[1, 1,,,] <- 1
subsim[2, 2,,,] <- 1 - kappa
rand2 <- rnorm(n_subsim * n_path * (n_dec - 1) / 2)
rand2 <- as.vector(rbind(rand2, -rand2))
subsim[2, 1,,,] <- kappa * mu + sigma * rand2
subsim_weight <- rep(1 / n_subsim, n_subsim)
## Primal-dual
mart <- FastAddDual(path, subsim, subsim_weight, grid, bellman$value, ScrapFunc)
bounds <- AddDualBounds(path, control, RewardFunc, ScrapFunc, mart, policy)
```

Listing 6.8: 99% confidence interval

```
> print(GetBounds(bounds, 0.01, 6))
[1] 13.42159 13.44162
```

6.4 Battery

Let us revisit the battery control problem in Section 4.4.2. The following code listing generates the results in Table 4.4.1. The code below begins by specifying the grid, the battery specifications, the action set, and the effect of the action on the battery level.

Listing 6.9: Value function approximation

```
library(rcss)
## Grid
n_grid <- 501 ## number of grid points
grid_start <- -15 ## lowest state
grid_end <- 15 ## highest state
grid <- cbind(rep(1, n_grid), seq(grid_start, grid_end, length = n_grid))
## Battery specification
step <- 5 ## step between battery levels
pos_start <- 0
pos_end <- 100
position <- seq(pos_start, pos_end, by = step) ## battery levels
n_pos <- length(position)
## Standard deviation for the consumer demand
std <- 10
## Actions
n_action <- 11 ## number of safety margins
safety_start <- 0
safety_end <- 50
safety <- seq(safety_start, safety_end, length = n_action) ## safety margins
## Control array
control <- array(data = 0, dim = c(n_pos, n_action, n_pos))
for (p in 1:n_pos) {
    for (a in 1:n_action) {
        temp <- position[p] + safety[a] ## center of normal distribution
        control[p,a,1] <- pnorm(pos_start + step/2, temp, std)
        control[p,a,n_pos] <- 1 - pnorm(pos_end - step/2, temp, std)
        for (pp in 2:(n_pos-1)) {
            control[p,a,pp] <- pnorm(position[pp] + step/2, temp, std) -
                pnorm(position[pp] - step/2, temp, std)
        }
    }
}
```

Next the code computes the expected excess and shortages in the control problem.

Listing 6.10: Value function approximation

```
## Functions to calculate expected excess and shortage energy demand
erf <- function(x){ ## error function
    return(2 * pnorm(x * sqrt(2)) - 1)
}
Excess <- function(pos, act) {
    temp1 <- pos_end + step/2
    temp2 <- pos + act
    result <- std/sqrt(2*pi) * exp(-(temp1-temp2)^2/(2*std^2)) +
        (temp2 - pos_end)/2 * (1 - erf(1/sqrt(2*std^2) * (temp1 - temp2)))
    return(result)
}
Shortage <- function(pos, act) {
    temp1 <- pos_start - step/2
    temp2 <- pos + act
    result <- std/sqrt(2*pi) * exp(-(temp1-temp2)^2/(2*std^2)) +
        (pos_start - temp2)/2 * (erf(1/sqrt(2*std^2) * (temp1 - temp2)) + 1)
    return(result)
}
## Expected excess and shortage energy demand
excess <- matrix(data = NA, nrow = n_pos, ncol = n_action)
shortage <- matrix(data = NA, nrow = n_pos, ncol = n_action)
for (p in 1:n_pos) {
    for (a in 1:n_action) {
        excess[p,a] <- Excess(position[p], safety[a])
        shortage[p,a] <- Shortage(position[p], safety[a])
    }
}
```

The subgradient envelope of the reward and scrap functions are then specified below.

Listing 6.11: Value function approximation

```
## Subgradient representation of reward functions
```

```
n_dec <- 48 * 7 ## number of decision epochs
u_t <- 10 + cos((0:(n_dec-1)) * 2*pi/48 + 3*pi/2)
v_t <- 1 + (sin((0:(n_dec-1)) * 2*pi/48 + 3*pi/2))/2
buy <- 20 ## price to buy from grid
sell <- 0 ## price to sell to grid
reward <- array(0, dim = c(n_grid, 2, n_action, n_pos, n_dec - 1))
for (p in 1:n_pos) {
    for (a in 1:n_action) {
        for (t in 1:(n_dec-1)) {
            reward[,1,a,p,t] <- -safety[a] * u_t[t] - shortage[p, a] * buy + excess[p, a] *
                sell
            reward[,2,a,p,t] <- -safety[a] * v_t[t]
        }
    }
}
scrap <- array(0, dim = c(n_grid, 2, n_pos))
for (p in 1:n_pos) {
    scrap[,1,p] <- position[p] * u_t[n_dec]
    scrap[,2,p] <- position[p] * v_t[n_dec]
}
```

Finally, the disturbance sampling and the resulting double modified value functions are computed using the subgradient approach.

Listing 6.12: Value function approximation

```
## Parameters for AR(1) process (Z_t)
mu <- 0
sigma <- 0.5
phi <- 0.9
## Disturbance sampling
n_disturb <- 1000 ## size of sampling
disturb_weight <- rep(1/n_disturb, n_disturb) ## probability weights
disturb <- array(matrix(c(1, 0, 0, phi), ncol = 2, byrow = TRUE), dim = c(2, 2, n_disturb))
CondExpected <- function(a, b){
    return(1/sqrt(2 * pi) * (exp(-a^2/2)- exp(-b^2/2)))
}
part <- qnorm(seq(0, 1, length = n_disturb + 1))
for (i in 1:n_disturb) {
    disturb[2,1,i] <- mu + sigma * (CondExpected(part[i], part[i+1]) / (pnorm(part[i+1]) -
        pnorm(part[i])))
}
r_index <- matrix(c(2, 1), ncol = 2) ## randomness index
## Fast bellman recursion
bellman <- FastBellman(grid, reward, scrap, control, disturb, disturb_weight, r_index)
```

The following code listing performs the solution diagnostics and then outputs the results found in Table 4.4.1.

Listing 6.13: Solution diagnostics

```
## Exact reward function
RewardFunc <- function(state, time) {
    output <- array(0, dim = c(nrow(state), n_action, n_pos))
    for (p in 1:n_pos) {
        for (a in 1:n_action) {
            output[,a,p] <- -safety[a] * (u_t[time] + v_t[time] * state[,2]) - shortage[p,a] *
                buy + excess[p,a] * sell
        }
    }
    return(output)
}
## Scrap function
ScrapFunc <- function(state) {
    output <- array(0, dim = c(nrow(state), n_pos))
    for (p in 1:n_pos) {
        output[,p] <- position[p] * (u_t[n_dec] + v_t[n_dec] * state[,2])
    }
    return(output)
```

```
}
## Generate sample path disturbances
set.seed(12345)
n_path <- 100
path_disturb <- array(matrix(c(1, 0, 0, phi), ncol = 2, byrow = TRUE),
                      dim = c(2, 2, n_path, n_dec - 1))
rand <- rnorm(n_path * (n_dec - 1) / 2)
rand <- as.vector(rbind(rand, -rand))
path_disturb[2,1,,] <- mu + sigma * rand
start <- c(1, 0) ## z_0
path <- PathDisturb(start, path_disturb)
policy <- FastPathPolicy(path, grid, control, RewardFunc, bellman$expected)
## Specifying subsimulation disturbance matrices
n_subsim <- 100
subsim_weight <- rep(1/n_subsim, n_subsim)
subsim <- array(matrix(c(1, 0, 0, phi), ncol = 2, byrow = TRUE), dim = c(2, 2, n_subsim,
    n_path, n_dec - 1))
rand <- rnorm(n_subsim * n_path * (n_dec - 1)/2)
rand <- as.vector(rbind(rand, -rand))
subsim[2,1,,,] <- mu + sigma * rand
## Compute primal and dual values
mart <- FastAddDual(path, subsim, subsim_weight, grid, bellman$value, ScrapFunc)
bounds <- AddDualBounds(path, control, RewardFunc, ScrapFunc, mart, policy)
## Storing the results
results <- matrix(data = NA, nrow = n_pos, ncol = 3)
alpha <- 0.01
for (p in 1:n_pos) {
    results[p,1] <- sum(bellman$value[251,,p,1] * grid[251,])
    results[p,2:3] <- GetBounds(bounds, alpha, p)
}
print(results)
```

6.5 Mining

This section considers the optimal extraction problem in Section 4.5.1 and generates the results found in Table 4.5.2. As before, the code below begins by specifying the grid, disturbance sampling, and the impact of the actions on the operational mode of the mine.

Listing 6.14: Value function approximation

```
library(rcss)
## Parametrs
n_grid <- 2001
grid <- as.matrix(cbind(rep(1, n_grid), seq(0, 10, length = n_grid)))
rate <- 0.1
delta <- 0.01
vol <- sqrt(0.08)
step <- 0.25
## Disturbance sampling
n_disturb <- 1000
disturb_weight <- rep(1/n_disturb, n_disturb)
disturb <- array(0, dim = c(2, 2, n_disturb))
disturb[1, 1,] <- 1
quantile <- rep(NA, n_disturb)
u <- (rate - delta - 0.5 * vol^2) * step
sigma <- vol * sqrt(step)
part <- qlnorm(seq(0, 1, length = n_disturb + 1), u, sigma)
condExpected <- function(a, b){ ##[a,b]
    aa <- (log(a) - (u+sigma^2))/sigma
    bb <- (log(b) - (u+sigma^2))/sigma
    vv <- exp(u + sigma^2/2) * (pnorm(bb) - pnorm(aa))
    return(vv)
}
for (i in 1:n_disturb) {
```

```
    quantile[i] <- condExpected(part[i], part[i+1]) / (plnorm(part[i+1], u, sigma) -
        plnorm(part[i], u, sigma))
}
disturb[2, 2,] <- quantile
r_index <- matrix(c(2, 2), ncol = 2)
## Control matrix, a = 1 (close), 2 (abandon), 3 (open)
## p = 1 (exhausted), levels + 1 (full open), 2 * levels + 1 (full closed)
levels <- 15 / step
control <- matrix(data = 1, nrow = (2 * levels + 1), ncol = 3)
control[2:(2 * levels + 1), 1] <- (levels + 2):(2 * levels + 1)
control[2:(2 * levels + 1), 3] <- 1:levels
```

The following then specifies the subgradient envelope of the reward and scrap functions. The double modified value functions are then computed on Line 71.

Listing 6.15: Value function approximation

```
## Subgrad rep of rewards
H1 <- 5 * step
H2 <- -2.5 * step
maint <- 0.5 * step
switch <- 0.2
inflation <- 0.08
tax_property <- 0.02
n_dec <- 1 + 30 / step + 1
discount <- exp(-(rate + tax_property) * step)
n_pos <- nrow(control)
n_action <- ncol(control)
n_dim <- ncol(grid)
reward <- array(data = 0, dim = c(n_grid, n_dim, n_action, n_pos, n_dec - 1))
for (p in 2:n_pos) {
    for (t in 1:(n_dec - 1)) {
        pi <- exp(inflation * (t-1) * step)
        adjust <- discount ^ (t-1)
        ## Close the asset
        if (p > (levels + 1)) { ## Closed
            ## Close
            reward[, 1, 1, p, t] <- adjust * -maint * pi
            ## Open
            reward[, 1, 3, p, t] <- (H2 - switch) * pi * adjust
            reward[, 2, 3, p, t] <- H1 * adjust
        } else if (p <= (levels + 1)) { ## Opened
            ## Close
            reward[, 1, 1, p, t] <- -(maint + switch) * pi * adjust
            ## Open
            reward[, 1, 3, p, t] <- H2 * pi * adjust
            reward[, 2, 3, p, t] <- H1 * adjust
        }
    }
}
## Subgrad rep of scrap
scrap <- array(data = 0, dim = c(n_grid, n_dim, n_pos))
## Performing fast bellman recursion
bellman <- FastBellman(grid, reward, scrap, control, disturb, disturb_weight, r_index)
```

The following code then performs the solution diagnostics and prints the results found in Table 4.5.2.

Listing 6.16: Solution diagnostics

```
## Reward function
RewardFunc <- function(state, time) {
    Fpi <- exp(inflation * (time-1) * step)
    Fdiscount <- exp(-(rate + tax_property) * (time-1) * step)
    FH1 <- 5 * step * Fdiscount
    FH2 <- -2.5 * step * Fdiscount * Fpi
    Fmaint <- 0.5 * step * Fdiscount * Fpi
    Fswitch <- 0.2 * Fdiscount * Fpi
    output <- array(0, dim = c(nrow(state), n_action, n_pos))
    output[,1,(2:(levels+1))] <- -(Fmaint + Fswitch)
```

```
    output[,3,(2:(levels+1))] <- FH2 + FH1 * state[,2]
    output[,1,(levels+2):n_pos] <- -Fmaint
    output[,3,(levels+2):n_pos] <- FH2 - Fswitch + FH1 * state[,2]
    return(output)
}
## Scrap function
ScrapFunc <- function(state) {
    output <- array(data = 0, dim = c(nrow(state), n_pos))
    return(output)
}
### Perform the solution diagnostics
## Path disturbances
set.seed(12345)
n_path <- 500
path_disturb <- array(0, dim = c(2, 2, n_path, n_dec - 1))
path_disturb[1,1,,] <- 1
rand1 <- rnorm(n_path * (n_dec - 1) / 2)
rand1 <- as.vector(rbind(rand1, -rand1))
path_disturb[2,2,,] <- exp((rate - delta - 0.5 * vol^2) * step + vol * sqrt(step) * rand1)
## Subsimulation disturbance
n_subsim <- 500
subsim_weight <- rep(1 / n_subsim, n_subsim)
subsim <- array(0, dim = c(2, 2, n_subsim, n_path, n_dec - 1))
subsim[1,1,,,] <- 1
rand2 <- rnorm(n_subsim * n_path * (n_dec - 1) / 2)
rand2 <- as.vector(rbind(rand2, -rand2))
subsim[2,2,,,] <- exp((rate - delta - 0.5 * vol^2) * step + vol * sqrt(step) * rand2)
## Looping
startIndex <- cbind(rep(1,8), seq(0.3, 1, by = 0.1))
pp <- levels + 1
results <- matrix(NA, nrow=nrow(startIndex), ncol = 6)
for (i in 1:nrow(startIndex)) {
    ## Paths
    start <- startIndex[i,]
    path <- PathDisturb(start, path_disturb)
    policy <- FastPathPolicy(path, grid, control, RewardFunc, bellman$expected)
    ## Primal and dual values
    mart <- FastAddDual(path, subsim, subsim_weight, grid, bellman$value, ScrapFunc)
    bounds <- AddDualBounds(path, control, RewardFunc, ScrapFunc, mart, policy)
    ## Store
    ind <- rflann::FastKDNeighbour(matrix(start, nrow = 1), grid, 1)
    results[i,1] <- sum(bellman$value[ind, , pp, 1] * grid[ind,])
    results[i,4] <- sum(bellman$value[ind, , n_pos, 1] * grid[ind,])
    results[i,2:3] <- GetBounds(bounds, 0.01, pp)
    results[i,5:6] <- GetBounds(bounds, 0.01, n_pos)
}
## Print results
print(round(results,3))
```

6.6 Conclusion

This chapter gives a demonstration of the R package *rcss* in solving optimal switching problems. The problem setting discussed in this thesis is broad and can be used to model a wide range of problems. Using nearest neighbour algorithms, the package *rcss* is able to solve some real world problems in an accurate and quick manner. We have also implemented these numerical methods in the *Julia* programming language available at: `https://github.com/YeeJeremy/ConvexSwitch.jl`.

References

Eddelbuettel D, Francois R (2011) Rcpp: Seamless R and C++ integration. Journal of Statistical Software 40(8):1–18, DOI 10.18637/jss.v040.i08

Hinz J, Yee J (2017) rcss: Convex switching systems. URL `https://github.com/YeeJeremy/rcss`, available at https://github.com/YeeJeremy/rcss, R package version 1.5

R Core Team (2013) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, URL `http://www.R-project.org/`, ISBN 3-900051-07-0