NTNU

Norwegian University of
Science and Technology

Hydroelectric Real Options

A Structural Estimation Approach

Marius Øverland Foss
Alexander Høst

Industrial Economics and Technology Management
Submission date: June 2011
Supervisor: Stein-Erik Fleten, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Preface

This master thesis was conducted at the Norwegian University of Science and Technology (NTNU), Department of Industrial Economics and Technology Management. Our work falls within the Group of Financial Engineering.

First and foremost, we would like to thank our supervisor, Professor Stein-Erik Fleten for helpful assistance and valuable discussions, and for the speed with which he has responded to our various requests. We would also like to thank Ph.D. candidate Daniel Haugstvedt for his guidance and comments, and Matteo Tesser for a long and fruitful discussion.

Trondheim, June 3, 2011

Alexander Høst

Marius Øverland Foss

Hydroelectric real options
A structural estimation approach

Marius Øverland Foss & Alexander Høst

June 3, 2011

Abstract

Structural estimation is an important technique in analyzing economic data. Unfortunately, it is often computationally expensive to implement the most powerful and efficient statistical methods. One such method is the Nested Fixed Point (NFXP) algorithm. In this thesis, we develop methodology and techniques that allow us to apply NFXP to real options models of hydropower production. In particular, we develop a way to regard hydropower planning and scheduling as a stationary problem. Further, we create a numerical method for solving specific types of equation systems with sparse matrices of a specific structure, an approach that significantly increases the speed with which we can compute Fréchet and partial derivatives of contraction mappings for large state spaces.

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we examine the potential of applying *structural estimation* techniques to real options models of hydropower production. The focus of the thesis is *methodology*, in particular how one can model hydropower production to apply these techniques, along with algorithmic modifications that makes solving such problems tractable. To illustrate, we use the widely acclaimed Nested Fixed Point (NFXP) algorithm developed by Rust (1988).

With estimation in mind, Rust (1988) developed NFXP by observing the following: if one assumes that an observed set of states and decisions is governed by a stochastic decision process generated by the Bellman equation, then one should be able to solve the underlying stochastic control problem inversely using maximum likelihood estimation techniques. Plainly; if one observes the actions of an agent, and assumes that the agent is maximizing utility, then one should be able to induce values for utility function parameters, thereby discovering the mathematical objective function of the agent. In Rust (1987), he proceeds to show how this can be done in practice.

Since the development of NFXP, several applications have shown the method to yield good results, e.g. Rust (1987), Muehlenbachs (2009) and Kellogg (2010). At the same time, some problems remain difficult to solve. Among these are problems with large state spaces - problems that inherently lead to the *curse of dimensionality*. To deal with such issues, several other approaches have been suggested[1]. Among these are parametric methods that approximate the value function in the Bellman equation using Chebyshev

[1]For an excellent exposition, see Benítez-Silva et al. (2000).

and B-Spline polynomials (see Gamba and Tesser, 2009) or neural networks, and algorithms that break the curse of dimensionality using random versions of successive approximations (see Rust, 1997).

Hydropower scheduling has traditionally been considered to be a finite horizon problem (see e.g. Nandalal and Bogárdi, 2007). However, treating a problem as infinite-horizon has several advantages, in particular with respect to asymptotic properties of statistical estimators. Infinite-horizon formulations require stationarity in the problem structure - a trait which is not obvious in the case of hydropower production. However, we show that it is possible to leverage the structure of the problem and the assumptions of Rust (1988) to induce stationarity, and that it is particularly useful in the case of inverse stochastic control. Expanding upon the assumptions of Rust (1988), we argue that deviations from stationarity are temporary and captured by an extreme value distribution.

The largest obstacle associated with structural estimation of hydropower production is that of tractability. To our knowledge, the state and decision space we consider is significantly larger than what has been examined in the literature till now. To break the inherent curse of dimensionality, we have thus been forced to devise means of speeding up computation. In particular, we create several numerical techniques that allows us to compute fixed points of contraction mappings and solve large equation systems in a fraction of the time.

Being the most computationally demanding part of NFXP, our efforts have been focused on finding ways to speed up the solution of the fixed point (dynamic programming) part of the algorithm. We identify two beneficial properties in the structure of the problem - properties that let us use methods from linear algebra, and concurrent execution techniques, in a way that significantly speeds up computation:

1. The sparsity structure of the matrix representing the Fréchet derivative of a contraction mapping suggests an iterative method of solving a large system of linear equations quickly, making the associated Newton-Kantorovich method tractable.

2. The structure of the stationary problem makes it a good candidate for parallel computing. By separating the most computationally intensive parts of the algorithm, time consumption is reduced by a factor of close to 52 (where the number 52 is due to a discretization of time into weeks).

Determination of optimal operating strategies for hydropower generation assets has long been the focus of research interest, and there is a rich literature on the subject[2]. Investigation of the actual decisions made by hydropower producers seems to be non-existent. Taking the positive perspective, we examine the applicability of structural estimation as a technique for understanding the key drivers behind hydropower producer decisions. Using the NFXP algorithm as our vehicle of exposition, we hope that the ideas and methodology we develop within can be built upon to develop faster, more efficient and more precise estimation techniques for uncovering hydropower producer preferences.

In chapter 2, we discuss structural estimation and the NFXP algorithm in general. In chapter 3, we construct a real-options model of hydropower production, and derive the equations needed by NFXP. Chapter 4 discusses implementation, algorithmic details and simulation, and chapter 5 concludes. Code may be found at `http://folk.ntnu.no/host/structural/`.[3]

[2]See e.g. Thompson et al. (2004); Fleten and Kristoffersen (2008); Philpott et al. (2000).

[3]Start program with the file `simulation.m`. Use `settings.m` to adjust parameters.

Chapter 2

Structural Estimation of Markov Decision Processes

This chapter focuses on structural estimation of Markov decision processes (MDPs) using the Nested Fixed Point (NFXP) algorithm developed by Rust (1988). Much of what we discuss here can be found in Rust (1994), but we have adapted and shortened it somewhat to fit our specific context. Also, we have tried to simplify the mathematical notation a bit, in an attempt to make it more accessible to the reader. Still, the NFXP algorithm is a complex numerical method that uses advanced concepts from both functional analysis and statistics; concepts that we cannot shy away from without severely inhibiting our ability to explain how and why we arrive at certain results.

First we give a short review of MDPs and finite- and infinite-horizon *dynamic programming* (DP). Readers familiar with these topics can skip ahead.

2.1 Markov Decision Processes

MDPs have been used extensively as a framework for modelling sequential decision making under uncertainty (Rust, 1994). In addition to providing a normative theory on how rational agents should behave, they also provide a good empirical framework of how real-world decision-makers ac-

tually behave[1]. MDPs have two types of variables: *state variables* $\boldsymbol{s}_t^\alpha$ and *decision variables* d_t^α, where t denotes time and α denotes agent or decision-maker[2]. An agent can be represented by a set of primitives (u, p, β) where $u(\boldsymbol{s}_t, d_t)$ is a utility function representing the agent's preference at time t, $p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, d_t)$ is a Markov transition probability representing the agent's beliefs about uncertain future states, and $\beta \in (0, 1)$ is the rate at which the agent discounts future utility. Agents are assumed to be rational: they behave according to an optimal decision rule $d_t = \delta(\boldsymbol{s}_t)$ that solves $V_0^T(\boldsymbol{s}) = \max_\delta \mathbb{E}_\delta \left\{ \sum_{t=0}^T \beta^t u(\boldsymbol{s}_t, d_t) | \boldsymbol{s}_0 = \boldsymbol{s} \right\}$ where $\mathbb{E}_\delta$ denotes expectation with respect to the controlled stochastic process $u(\boldsymbol{s}_t, d_t)$ induced by the decision rule δ.

Consider a finite horizon, discrete-time MDP with the following properties:

- A time index $t \in \{0, 1, 2, \ldots, T\}$
- A state space $\boldsymbol{S}$
- A decision space D
- A family of transition probabilities $\{p_{t+1}(\cdot|\boldsymbol{s}_t, d_t) : \mathcal{B}(\boldsymbol{S}) \to [0, 1])\}$, where $\mathcal{B}(\boldsymbol{S})$ is the Borel σ-algebra of measurable subsets of $\boldsymbol{S}$
- A family of discount functions $\{\beta_t(\boldsymbol{s}_t, d_t) \geq 0\}$ and single-period utility functions $\{u_t(\boldsymbol{s}_t, d_t)\}$ such that the sum of all future utility discounted at the appropriate discount function may be given by

$$U(\boldsymbol{s}, \boldsymbol{d}) = \sum_{t=0}^{T} \left[\prod_{j=0}^{t-1} \beta_j(\boldsymbol{s}_j, d_j) \right] u_t(\boldsymbol{s}_t, d_t),$$

  where we for $t = 0$ define

$$\prod_{j=0}^{-1} \beta_j(\boldsymbol{s}_j, d_j) = 1.$$

The agent's maximization problem is thus to choose some optimal decision rule $\delta^* = (\delta_0, \ldots, \delta_T)$ to solve

$$\max_{\delta=(\delta_0,\ldots,\delta_T)} \mathbb{E}_\delta \left\{ U(\boldsymbol{s}, \boldsymbol{d}) \right\}. \tag{2.1}$$

[1] Rust (1994).

[2] For notational simplicity and ease of exposition, we drop the agent index; properties and results generalize naturally to the multi-agent case.

The expected value of discounted utility over the remaining horizon is called the *value function*, and assumes that an optimal policy δ is followed in the future. The method of dynamic programming exploits the additive separability of the utility functions and the Markovian structure of the model, and calculates the value function and the optimal policy as follows. In the terminal period, the value function V_T and decision δ_T are defined by

$$\begin{aligned} V_T(\boldsymbol{s}_T) &= \max_{d_T \in D_T(\boldsymbol{s}_T)} u_T(\boldsymbol{s}_T, d_T) \\ \delta_T(\boldsymbol{s}_T) &= \arg\max_{d_T \in D_T(\boldsymbol{s}_T)} u_T(\boldsymbol{s}_T, d_T). \end{aligned}$$

In periods $t = 0, 1, \ldots, T-1$, V_t and δ_t are recursively defined by

$$V_t(\boldsymbol{s}_t) = \max_{d_t \in D_t(\boldsymbol{s}_t)} \left\{ u_t(\boldsymbol{s}_t, d_t) + \beta_{t-1}(\boldsymbol{s}_{t-1}, d_{t-1}) \int V_{t+1}(\boldsymbol{s}_{t+1})\, p_{t+1}(d\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, d_t) \right\} \tag{2.2}$$

$$\delta_t(\boldsymbol{s}_t) = \arg\max_{d_t \in D_t(\boldsymbol{s}_t)} \left\{ u_t(\boldsymbol{s}_t, d_t) + \beta_{t-1}(\boldsymbol{s}_{t-1}, d_{t-1}) \int V_{t+1}(\boldsymbol{s}_{t+1})\, p_{t+1}(d\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, d_t) \right\}. \tag{2.3}$$

Solving these recursions in reverse, i.e. finding δ_T, then δ_{T-1} and so on, the optimal decision rule $\delta^* = (\delta_0, \ldots, \delta_T)$ is generated. It then follows that

$$V_0(\boldsymbol{s}) = \max_{\delta} \mathbb{E}_\delta \left\{ U(\boldsymbol{s}, \boldsymbol{d}) | \boldsymbol{s}_0 = \boldsymbol{s} \right\},$$

which is what we wanted to solve in equation (2.1). For a more formalized version of these results, see e.g. Gihman and Skorohod (1979) and Rust (1994).

2.1.1 Infinite horizon dynamic programming and Bellman's equation

If we are willing to assume that the MDP is stationary, then transition probabilities are the same for all t and the discount functions are set equal to some constant $\beta \in [0, 1)$. This assumption fails to yield any useful simplifications for the finite-horizon case, but for the infinite-horizon case, stationarity implies that the future looks exactly the same in state $\boldsymbol{s}_t$ at time t and $\boldsymbol{s}_{t+k}$ at time $t + k$ provided that $\boldsymbol{s}_t = \boldsymbol{s}_{t+k}$. This suggests that the optimal decision rule and corresponding value function are time invariant.

Then, equations (2.2) and (2.3) may be recursively defined by[3]

$$V(\boldsymbol{s}) = \max_{d \in D(\boldsymbol{s})} \left\{ u(\boldsymbol{s}, d) + \beta \int V\left(\boldsymbol{s}'\right) p(d\boldsymbol{s}' | \boldsymbol{s}, d) \right\} \tag{2.4}$$

$$\delta(\boldsymbol{s}) = \arg\max_{d \in D(\boldsymbol{s})} \left\{ u(\boldsymbol{s}, d) + \beta \int V\left(\boldsymbol{s}'\right) p(d\boldsymbol{s}' | \boldsymbol{s}, d) \right\} \tag{2.5}$$

Equation (2.4) is known as *Bellman's equation*.

2.2 Structural estimation

In contrast to so-called *reduced-form* methods, which generally provide evidence about partial equilibria in regression frameworks, structural estimation is a technique for estimating deep *structural* parameters of theoretical models.

Consider the set of primitives (u, p, β) introduced at the start of section 2.1. We will focus on structural estimation of MDPs under the hypothesis that a series of observations of states and decisions, denoted by $\{\boldsymbol{s}_t, d_t\}$, is a realization of a controlled stochastic process. In addition to uncovering the form of this stochastic process, structural estimation methods attempt to uncover (i.e. estimate) the primitives (u, p, β) that generated it, that is, the mathematical objective function of the agent. Rust (1988) calls this the problem of *inverse stochastic control*.

Our ability to determine agents' preferences and beliefs are contingent upon our willingness to impose prior restrictions on (u, p, β). If we, for instance, are not willing to assume anything beyond basic measurability and regularity conditions on u and p, then Rust (1994) argues that it is impossible to consistently estimate (u, p, β), i.e. *the class of all MDPs is nonparametrically unidentified*. On the other hand, if we are willing to restrict u and p to a finite-dimensional parametric family, say

$$\left\{ u = u_{\boldsymbol{\theta}}, p = p_{\boldsymbol{\theta}} | \boldsymbol{\theta} \in \Theta \subseteq \mathbb{R}^K \right\},$$

for some positive K, then the primitives (u, p, β) are identified. If we are willing to impose even stronger prior restrictions, stationarity and *rational*

[3]The usage of $d\boldsymbol{s}'$ in $p(d\boldsymbol{s}'|\boldsymbol{s}, d)$ may seem ambiguous. It should be read as *the probability that the state changes to* $\boldsymbol{s}'$ *given that the current state and action is* $\boldsymbol{s}$ *and* d*, respectively.*

expectations (RE), then we only need parametric restrictions on u in order to identify (u, p, β). So, say that we now bring to bear prior knowledge in the form of a parametric representation for (u, p, β). Then, the problem of structural estimation reduces to the technical issue of estimating a parameter vector $\boldsymbol{\theta} \in \Theta$ where Θ is a compact subset of $\mathbb{R}^K$.

The appropriate econometric method for estimating $\boldsymbol{\theta}$ depends critically on whether the control variable d_t is continuous or discrete. In the case of continuous d_t, the MDP is known as a *continuous decision process* (CDP), whereas if d_t is discrete and countable, the term *discrete decision process* (DDP) is used. For the former type of process, the predominant estimation method is the *generalized method of moments* (GMM) (see Hansen and Singleton, 1982). For the latter, on the other hand, the optimal decision rule is obtained by solving a dynamic programming problem rather than as a zero to a first order condition. Thus, most structural estimation methods for DDPs has to combine dynamic programming with a method for fitting the data.

2.3 Econometric specification

Conceptually, we may regard our econometric method as a form of nonlinear regression, where we search for an estimate $\hat{\boldsymbol{\theta}}$ whose implied decision rule $d_t = \delta(\boldsymbol{s}_t, \hat{\boldsymbol{\theta}})$ best fits the data $\{\boldsymbol{s}_t, d_t\}$. Unfortunately, there are several issues that make direct application of nonlinear regression methods infeasible. Using the language of statistical regression,

1. the *dependent variable* d_t is discrete rather than continuous
2. the functional form of δ is generally not known a priori, but must rather be derived as a solution to the stochastic control problem
3. the *error term* ϵ_t in the regression function δ is typically multidimensional and enters in a non-additive, non-separable fashion, $d_t = \delta(\boldsymbol{s}_t, \epsilon_t, \theta)$.

Since nonlinear regression methods are infeasible, a different approach must be used. Consider uncovering the agent's preferences and expectations by

letting $\hat{\boldsymbol{\theta}}$ be the parameter vector that maximizes the likelihood function

$$\mathcal{L}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T, d_1, \ldots, d_T | \boldsymbol{\theta}) = \prod_{t=2}^{T} P(d_t | \boldsymbol{s}_t, \boldsymbol{\theta}) p(\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, d_{t-1}, \boldsymbol{\theta}). \tag{2.6}$$

Let's dwell at this expression for a moment. $p(\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, d_{t-1}, \boldsymbol{\theta})$ is the familiar Markov transition probability. It expresses the probability of moving from one state to another, given that one undertakes action d. $P(d_t | \boldsymbol{s}_t, \boldsymbol{\theta})$ is called the *conditional choice probability*, and expresses the probability of selecting decision d_t given state $\boldsymbol{s}_t$. Maximizing a likelihood function is essentially a fitting process, where one adjusts parameter vector $\boldsymbol{\theta}$ so that the joint probability distribution under consideration, best fits the data.

The basic motivation for including an error term ϵ_t in the model should be quite obvious. As shown in section 2.1, the optimal decision rule δ is a deterministic function of the state $\boldsymbol{s}$. Thus, trying to fit actual data to any such model would lead to degeneracy without an error term. There are several possible interpretations of the error term, but Rust (1994) argues that the most natural interpretation of ϵ is that it is an *unobserved state variable*. Indeed, it is unlikely that any survey could record all information that is relevant for the agent's decision-making process. ϵ thus provides a natural way to "rationalize" discrepancies between observed behavior and the predictions of the DDP. At this point, the reader may question why the probabilities $P(d_t | \boldsymbol{s}_t, \boldsymbol{\theta})$ do not depend on unobserved states ϵ_t - the reason for this will become apparent shortly.

The conditional choice probabilities, or simply the *choice probabilities*, are more subtle than they may seem. The decisions made now will obviously affect future states, but what is perhaps not so obvious, is that since what is optimal to do now is affected by what is optimal to do in the future, the choice probabilities must in some way depend on the value function. This intuition can be formalized mathematically, but requires some results from the theory of static discrete choice. We find it to be easiest to show this in a bottom-up manner. The remainder of this section will thus be concerned with developing these concepts, which will ultimately lead us to an expression for the choice probabilities.

Let's begin by expressing the choice probabilities in an intuitive way. Recall that we denote the optimal decision by δ. Imagine now that δ is known.

Then, obviously,

$$P(d|\boldsymbol{s},\epsilon,\boldsymbol{\theta}) = \begin{cases} 1 & \text{if } d = \delta(\boldsymbol{s},\epsilon,\boldsymbol{\theta}) \\ 0 & \text{otherwise,} \end{cases}$$

but, since δ is a function of ϵ, the best we can do is to write

$$P(d|\boldsymbol{s},\boldsymbol{\theta}) = \int I\left\{d = \delta\left(\boldsymbol{s},\epsilon,\boldsymbol{\theta}\right)\right\} q(d\epsilon|\boldsymbol{s}),$$

where $I(\cdot)$ is the *Kronecker delta*[4] and $q(d\epsilon|\boldsymbol{s})$ is the conditional distribution of ϵ given $\boldsymbol{s}$. This expression tells us that we need to do at least two things:

1. find an expression for the optimal decision in a state, and
2. find a conditional probability distribution for ϵ.

Rust (1994) argues that it is important that the choice probability has unbounded support, that is,

$$d \in D(\boldsymbol{s}) \iff P(d|\boldsymbol{s},\epsilon,\boldsymbol{\theta}) > 0. \tag{2.7}$$

We say that a specification is *saturated* if statement (2.7) holds for all possible values of $\boldsymbol{\theta}$. The problem with an unsaturated specification is that it is conceivable that the DDP may be contradicted in a sufficiently large data set, i.e. one may encounter observations $(\boldsymbol{s}_t, d_t)$ that cannot be rationalized by any value of ϵ or $\boldsymbol{\theta}$. This would lead to practical difficulties, in that it would cause the log-likelihood function to diverge to negative infinity when it encounters a "zero probability" observation.

Borrowing from McFadden (1981), we now present two central assumptions, which are sufficient to generate a saturated specification for $P(d|\boldsymbol{s},\epsilon,\boldsymbol{\theta})$.

Assumption **AS - Additive separability** *The utility function has the additively separable decomposition*

$$u(s,d) = u(\boldsymbol{s},d,\boldsymbol{\theta}) + \epsilon(d)$$

where $\epsilon(d)$ is the d'th component of the vector ϵ.

[4]The Kronecker delta is a function of two variables, and is 1 if they are equal, and 0 otherwise.

Assumption **CI - Conditional independence** *The transition density for the controlled Markov process* $\{\boldsymbol{s}, \epsilon\}$ *factors as*

$$\pi(d\boldsymbol{s}_{t+1}, d\epsilon_{t+1}|\boldsymbol{s}_t, \epsilon_t, d_t, \boldsymbol{\theta}) = q(d\epsilon_{t+1}|\boldsymbol{s}_{t+1})p(d\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, d_t, \boldsymbol{\theta}).$$

Assumption **CI** implies the following:

1. $\boldsymbol{s}_{t+1}$ is a sufficient statistic for ϵ_{t+1}, i.e. any and all serial dependence between ϵ_t and ϵ_{t+1} is transmitted entirely through $\boldsymbol{s}_{t+1}$.
2. The probability density of $\boldsymbol{s}_{t+1}$ depends entirely on $\boldsymbol{s}_t$ and not ϵ_t.

Now, under assumptions **AS** and **CI**, Bellman's equation becomes

$$V_{\boldsymbol{\theta}}(\boldsymbol{s}, \epsilon) = \max_{d \in D(\boldsymbol{s})} \{v_{\boldsymbol{\theta}}(\boldsymbol{s}, d) + \epsilon(d)\} \tag{2.8}$$

where

$$v_{\boldsymbol{\theta}}(\boldsymbol{s}, d) = u(\boldsymbol{s}, d, \boldsymbol{\theta}) + \beta \int V_{\boldsymbol{\theta}}(\boldsymbol{s}', \epsilon')q(d\epsilon'|\boldsymbol{s}')p(d\boldsymbol{s}'|\boldsymbol{s}, d, \boldsymbol{\theta}). \tag{2.9}$$

Assuming ϵ is continuously distributed with unbounded support, it follows that regardless of the values of $v_{\boldsymbol{\theta}}(\boldsymbol{s}, d)$, $V_{\boldsymbol{\theta}}(\boldsymbol{s}, \epsilon)$ is positive for each $d \in D(\boldsymbol{s})$. We later show that $P(d|\boldsymbol{s}, \epsilon, \boldsymbol{\theta})$ depends on $V_{\boldsymbol{\theta}}(\boldsymbol{s}, \epsilon)$ in such a way that $P(d|\boldsymbol{s}, \epsilon, \boldsymbol{\theta})$ is always guaranteed to be positive. The specification for the choice probabilities is thus saturated.

Let's focus now on finding an expression for the choice probabilities. To do this, we introduce what McFadden (1981) calls the *social surplus function*:

$$G\left[\{u\left(\boldsymbol{s}, d\right), d \in D(\boldsymbol{s})\}\,|\boldsymbol{s}, \boldsymbol{\theta}\right] = \int \max_{d \in D(\boldsymbol{s})} \{u(\boldsymbol{s}, d, \boldsymbol{\theta}) + \epsilon(d)\}\, q(d\epsilon|\boldsymbol{s}). \tag{2.10}$$

This function can be thought of as a function that maximizes one-period utility[5]. Now, if $q(d\epsilon|\boldsymbol{s})$ has finite first moments we can show that

$$\frac{\partial G\left[\{u\left(\boldsymbol{s}, d, \boldsymbol{\theta}\right), d \in D(\boldsymbol{s})\}\,|\boldsymbol{s}, \boldsymbol{\theta}\right]}{\partial u\left(\boldsymbol{s}, d, \boldsymbol{\theta}\right)} = P(d|\boldsymbol{s}, \boldsymbol{\theta}).$$

We prove the above in appendix C.1. Going from the static case to the dynamic case, that is, finding a social surplus function that works not only

[5]Notice how the **AS** assumption allows ϵ to be integrated out. This explains why the choice probabilities do not depend on ϵ in the likelihood function (2.6).

for one-period utility but for multi-period utility, requires three regularity conditions. Following Rust (1994), we call these conditions **BU**, **WC** and **BE**, respectively, and give them in appendix B. We are now ready to present two central results. Proofs can be found in Rust (1988).

Theorem 1. *If $\{\boldsymbol{s}_t, \epsilon_t, d_t\}$ is a DDP satisfying* **AS**, **CI** *and regularity conditions* **BU**, **WC** *and* **BE**, *then the optimal decision rule δ is given by*

$$\delta(\boldsymbol{s}, \epsilon) = \arg\max_{d \in D(\boldsymbol{s})} \left[v_\theta(\boldsymbol{s}, d) + \epsilon(d) \right],$$

where v is the unique fixed point to the contraction mapping $T : B \to B$ defined by

$$T(v_\theta)(\boldsymbol{s}, d) = u(\boldsymbol{s}, d, \boldsymbol{\theta}) + \beta \int G\left[\left\{ u\left(\boldsymbol{s}', d', \boldsymbol{\theta}\right), d' \in D(\boldsymbol{s}') \right\} | \boldsymbol{s}' \right] \pi(d\boldsymbol{s}' | \boldsymbol{s}, d) \tag{2.11}$$

Theorem 2. *If $\{\boldsymbol{s}_t, \epsilon_t, d_t\}$ is a DDP satisfying* **AS**, **CI** *and regularity conditions* **BU**, **WC** *and* **BE**, *then the controlled process is Markovian with transition probability*

$$\mathbb{P}\left(d\boldsymbol{s}_{t+1}, d_{t+1} | \boldsymbol{s}_t, d_t, \boldsymbol{\theta}\right) = P(d_{t+1} | \boldsymbol{s}_{t+1}, \boldsymbol{\theta}) p(d\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, d_t, \boldsymbol{\theta}),$$

where the conditional choice probability $P(d|\boldsymbol{s}, \boldsymbol{\theta})$ is given by

$$P(d|\boldsymbol{s}, \boldsymbol{\theta}) = \frac{\partial G\left[\left\{ v_{\boldsymbol{\theta}}\left(\boldsymbol{s}, d\right), d \in D(\boldsymbol{s}) \right\} | \boldsymbol{s}, \boldsymbol{\theta} \right]}{\partial v_{\boldsymbol{\theta}}\left(\boldsymbol{s}, d\right)}.$$

G is the social surplus function and v is the unique fixed point to the contraction mapping defined in equation (2.11).

Theorems 1 and 2 state that the dynamic programming problem can be solved by computing the fixed point v_θ to the contraction mapping in equation (2.11), and that this fixed point can be used to find the choice probabilities. We will now, for convenience, define the function EV_θ, which describes the expected value of the value function.

$$EV_\theta(\boldsymbol{s}, d) = \int V_\theta(\boldsymbol{s}', \boldsymbol{\epsilon}') q(d\boldsymbol{\epsilon}' | \boldsymbol{s}') p(d\boldsymbol{s}' | \boldsymbol{s}, d, \boldsymbol{\theta}) \tag{2.12}$$

This in turn allows us to simplify equation (2.8) and (2.9) to

$$V_\theta(\boldsymbol{s}, \boldsymbol{\epsilon}) = \max_{d \in D(\boldsymbol{s})} \left\{ u(\boldsymbol{s}, d, \boldsymbol{\theta}) + \boldsymbol{\epsilon}(d) + \beta EV_\theta(\boldsymbol{s}, d) \right\}.$$

By selecting various distributions for ϵ we can now obtain expressions for the choice probabilities. In particular, if $q(d\boldsymbol{\epsilon}|\boldsymbol{s})$ is assumed to be a (Gumbel) *multivariate extreme value distribution*, we show (in appendix C.2) that the choice probabilities are given by the multinomial logit formula

$$P(d_t|\boldsymbol{s}_t,\boldsymbol{\theta}) = \frac{\exp\left\{\dfrac{u(\boldsymbol{s}_t,d_t,\boldsymbol{\theta})+\beta EV_\theta(\boldsymbol{s}_t,d_t)}{\sigma}\right\}}{\sum_{d'\in D(\boldsymbol{s}_t)}\exp\left\{\dfrac{u(\boldsymbol{s}_t,d',\boldsymbol{\theta})+\beta EV_\theta(\boldsymbol{s}_t,d')}{\sigma}\right\}} \tag{2.13}$$

where σ is the scale parameter of the extreme value distribution. As can be seen, choosing this particular probability distribution allows us to obtain an explicit expression for the choice probabilities, instead of one that requires (potentially very expensive) numerical integration, as in equation (2.10). The advantages of not having to use numerical integration to obtain the choice probabilities cannot be overstated. In the algorithm we present later, this would have had to be performed so many times that the problem would essentially have become intractable.

Theorem 1 stated that the dynamic programming problem could be solved by computing the fixed point v_θ to the contraction mapping in equation (2.11). In fact, we can show that when we assume $q(d\boldsymbol{\epsilon}|\boldsymbol{s})$ to be extreme value distributed, the expected value function EV_θ itself can be given as a unique fixed point to a contraction mapping $EV_\theta = T_\theta(EV_\theta)$, which we prove in appendix C.3. The fixed point of the contraction mapping is defined by

$$\begin{aligned} EV_\theta(\boldsymbol{s},d) &= T_\theta(EV_\theta)(\boldsymbol{s},d) \\ &\equiv \int \sigma \log\left[\sum_{d'\in D(\boldsymbol{s}')}\exp\left\{\frac{u(\boldsymbol{s}',d',\boldsymbol{\theta})+\beta EV_\theta(\boldsymbol{s}',d')}{\sigma}\right\}\right] p(d\boldsymbol{s}'|\boldsymbol{s},d,\boldsymbol{\theta}). \end{aligned} \tag{2.14}$$

For convenience, we now define the following function:

$$\psi(\boldsymbol{s},d,\boldsymbol{\theta}) \overset{def}{=} \exp\left\{\frac{u(\boldsymbol{s},d,\boldsymbol{\theta})+\beta EV_\theta(\boldsymbol{s},d)}{\sigma}\right\}.$$

This expression will be used frequently. Equation (2.13) and (2.14) can then be simplified to

$$P(d|\boldsymbol{s},\boldsymbol{\theta}) = \frac{\psi(\boldsymbol{s},d,\boldsymbol{\theta})}{\sum_{d'\in D(\boldsymbol{s})}\psi(\boldsymbol{s},d',\boldsymbol{\theta})}$$

and

$$T_\theta(EV_\theta)(\boldsymbol{s}, d) = \int \sigma \log \left[\sum_{d' \in D(\boldsymbol{s}')} \psi(\boldsymbol{s}', d', \boldsymbol{\theta}) \right] p(d\boldsymbol{s}' | \boldsymbol{s}, d, \boldsymbol{\theta}). \tag{2.15}$$

Following Rust (1994) we normalize σ to 1, eliminating it from all equations from hereon out.

2.4 The Nested Fixed Point algorithm - NFXP

In the previous section, we showed that the difficulty in maximizing the likelihood function

$$\mathcal{L}(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T, d_1, \ldots, d_T | \boldsymbol{\theta}) = \prod_{t=2}^{T} P(d_t | \boldsymbol{s}_t, \boldsymbol{\theta}) p(\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, d_{t-1}, \boldsymbol{\theta}). \tag{2.16}$$

is that $\mathcal{L}$ does not have an a priori known functional form: the conditional choice probability $P(d_t | \boldsymbol{s}_t, \boldsymbol{\theta})$ depends on the expected value function EV_θ, which can only be found as the fixed point of the contraction mapping $T_\theta(EV_\theta)$. Rust (1988) observed that this suggests a *nested fixed point* algorithm: an "inner" contraction fixed-point algorithm that computes EV_θ, and an "outer" hill-climbing algorithm that searches for the values of $\boldsymbol{\theta}$ that maximizes $\mathcal{L}$. Rust then proceeded to develop such an algorithm, and illustrated it in his now-famous paper; Rust (1987).

The main idea behind NFXP is to repeatedly switch between the inner and outer parts of the algorithm until it converges. Figure 4.1 in Chapter 4 displays a detailed flowchart of the components of the algorithm.

2.4.1 Description of the algorithm

Consider the flowchart in figure 2.1. Initially, we guess values for the parameter vector $\boldsymbol{\theta}$. We then solve the dynamic programming problem to find the fixed point of EV_θ by using a contraction procedure. Having obtained the expected value of being in any state, we can calculate the choice probabilities and derivatives of the likelihood function. This allows us to perform *one* iteration of a hillclimbing routine on the likelihood-function, after which we return to the inner loop and resolve the DP problem. Switching between

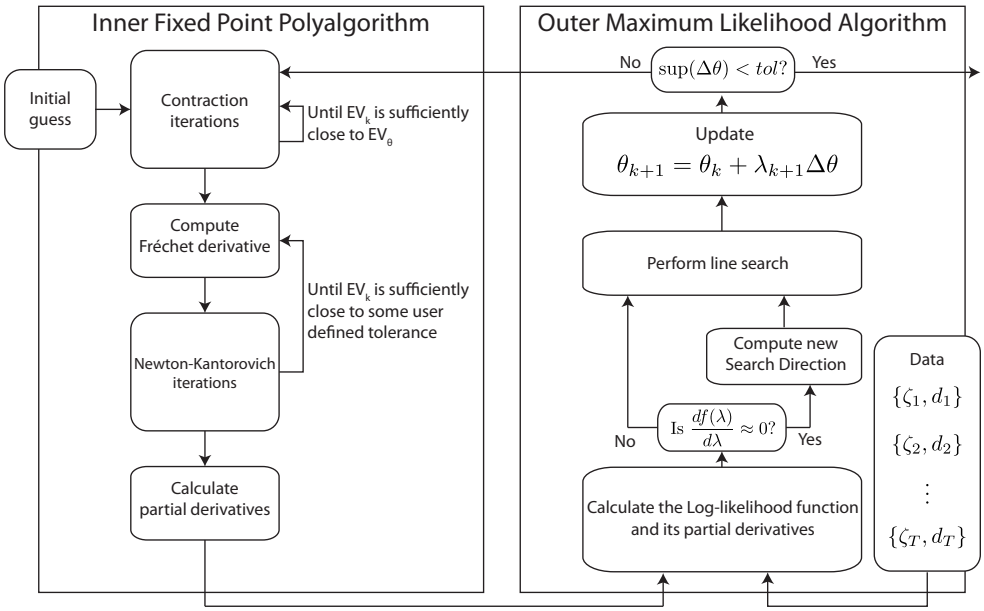

Figure 2.1: The main components of our implementation of NFXP.

the inner and outer loops, we eventually reach a maximum of the likelihood function.

It is surprisingly easy to misunderstand how the maximum of the likelihood function is found. Therefore, we draw analogies to more well-known hillclimbing methods. We urge the reader to read the following carefully. Standard hillclimbing algorithms typically proceed by finding a good search direction in a point, then perform a line search to find a good distance to move, and then move that distance in the search direction to a new point. At this new point, a new search direction is found, and the process is repeated. NFXP does things a bit differently. Because of the structure of the problem, NFXP combines the line search and the overall maximization into one process. Instead of line-searching to a maximum in the search direction, NFXP actually moves to each point evaluated by the line search, thus finding new values for $\boldsymbol{\theta}$. Doing otherwise would be useless, because the values of EV_θ are invalid at any new point in the likelihood function (and so are the derivatives). Thus, NFXP has to perform an iteration of its inner loop *every* time a new point is evaluated. In other words, NFXP not only maximizes the value of a function; it changes the function it is maximizing *while the maximization is being performed.*

NFXP also allows us to estimate the contraction parameter[6] β. We have decided to include it in $\boldsymbol{\theta}$, so whenever we write $\boldsymbol{\theta}$ we are actually referring to the vector $\{\beta, \theta_1, \ldots, \theta_n\}$. It is also worth mentioning that the transition probability may be dependent on $\boldsymbol{\theta}$ as well as the current state $\boldsymbol{s}$ and decision d; this should be clear from equation (2.16). The reason we here say *may* is because it is not always necessary or desirable to estimate the transition probabilities as a part of the NFXP algorithm. We will address this issue in Chapter 3.

2.4.2 Discretization

For continuous state spaces, the fixed point solution of $T_\theta(EV_\theta)$ is an infinite-dimensional object lying in the *Banach space*[7] of all bounded, measurable functions of $\boldsymbol{s}$ under the supremum norm. In order to compute EV_θ numerically, it becomes necessary to discretize the state space so that $\boldsymbol{s}$ only can take on a finite set of values.[8] Once the state space has been discretized, the function EV_θ becomes a finite-dimensional vector with dimension n equal to the number of unique states $\boldsymbol{s}$ in the total set of states $\boldsymbol{S}$. The discretization procedure's sole purpose is to approximate the contraction mapping T_θ on an infinite-dimensional Banach space B by a finite contraction mapping T_θ^n on a high-dimensional euclidean space $\mathbb{R}^n$. That is, we find a fixed point solution to a finite, discrete-state contraction mapping, which approximates the solution to the continuous state contraction mapping T_θ.

The discrete version of equation (2.15) is[9]

$$T_\theta^n(EV_\theta)(\boldsymbol{s}, d) = \sum_{\forall \Delta \boldsymbol{s}'} \left\{ p(\Delta \boldsymbol{s}' | \boldsymbol{s}, d, \boldsymbol{\theta}) \times \log \left[\sum_{d' \in D(\boldsymbol{s}')} \psi(\boldsymbol{s}', d', \boldsymbol{\theta}) \right] \right\}. \quad (2.17)$$

Both $T_\theta^n(EV_\theta)(\boldsymbol{s}, d)$ and $\log \left[\sum_{d' \in D(\boldsymbol{s}')} \psi(\boldsymbol{s}', d', \boldsymbol{\theta}) \right]$ are vectors. It should be clear to the reader that multiplying the latter expression with a transition

[6] Sometimes called the *Lipschitz constant*.

[7] A *Banach* space is a *complete*, *normed* vector space. A norm, often denoted by $||\bullet||$, is a function which defines the length of a vector in the space. To be complete, the limit of every *Cauchy sequence* in the space must also lie in the space. For instance, the familiar Euclidian spaces K^n are Banach spaces, since all convergent sequences (i.e. Cauchy sequences) have their limit in the same space.

[8] An alternative approach, suggested by Rust (1997), does not require discretization of the state space, but uses a random multigrid algorithm.

[9] The usage of $p(\Delta \boldsymbol{s}' | \boldsymbol{s}, d, \boldsymbol{\theta})$ is completely analogous to what we discussed in footnote 3 on page 7.

probability and then summing over all possible transitions, is equivalent to premultiplying the vector $\log\left[\sum_{d'\in D(\boldsymbol{s})}\psi(\boldsymbol{s},d',\boldsymbol{\theta})\right]$ with a Markov transition matrix $\mathcal{P}$. We may thus express equation (2.17) as

$$T_\theta^n(EV_\theta)(\boldsymbol{s},d) = \mathcal{P}\times\log\left[\sum_{d'\in D(\boldsymbol{s})}\psi(\boldsymbol{s},d',\boldsymbol{\theta})\right] \tag{2.18}$$

We omit the n in T_θ^n from here on.

2.4.3 The inner fixed point polyalgorithm

The main goal of this inner loop of NFXP is to approximate the solution to equation (2.12), which we restate here for convenience:

$$\begin{aligned} EV_\theta(\boldsymbol{s},\boldsymbol{\epsilon},d) &= \int V_\theta(\boldsymbol{s}',\boldsymbol{\epsilon}')\pi(d\boldsymbol{s}',d\boldsymbol{\epsilon}'|\boldsymbol{s},\boldsymbol{\epsilon},d,\boldsymbol{\theta}) \\ &= \iint V_\theta(\boldsymbol{s}',\boldsymbol{\epsilon}')q(d\boldsymbol{\epsilon}'|\boldsymbol{s}',\boldsymbol{\theta})p(d\boldsymbol{s}'|\boldsymbol{s},d,\boldsymbol{\theta}). \end{aligned}$$

The inner loop consists of two algorithms (c.f. figure 2.1): contraction iteration and Newton-Kantorovich iteration. The polyalgorithm guarantees convergence to a fixed point solution of an infinite horizon discrete-state Bellman equation.

Contraction iteration

The inner loop starts off with contraction iteration on the contraction mapping displayed in equation (2.18). The transition probabilities $p(\Delta\boldsymbol{s}'|\boldsymbol{s},d,\boldsymbol{\theta})$ are the elements in the $n\times n$ Markov transition matrix $\mathcal{P}$, where element (i,j) equals the probability of moving from state $\boldsymbol{s}^i$ to state $\boldsymbol{s}^j$. Contraction iteration is thus simply a matter of matrix multiplication of the transition matrix with the vector $\log\left[\sum_{d'\in D(\boldsymbol{s})}\psi(\boldsymbol{s},d',\boldsymbol{\theta})\right]$. The contraction iteration formula becomes

$$EV_{k+1}(\boldsymbol{s},d) = \mathcal{P}\times\log\left[\sum_{d'\in D(\boldsymbol{s})}\exp\left\{u(\boldsymbol{s},d',\boldsymbol{\theta})+\beta EV_k(\boldsymbol{s},d')\right\}\right],$$

or in compact notation

$$EV_{k+1}(\boldsymbol{s}, d) = \mathcal{P} \times \log \left[\sum_{d' \in D(\boldsymbol{s})} \psi_k(\boldsymbol{s}, d', \boldsymbol{\theta}) \right]. \tag{2.19}$$

There is a major benefit to being able to use matrix multiplication. We avoid time-consuming summations, which greatly simplifies how we deal with state changes. By the use of Markov transition matrices, we achieve the same as in equation (2.17), while avoiding for-loops in implementation. Changes that are a direct result of a decision, in our case, *reduction in reservoir level due to production*, need to be handled separatly.

Contraction iterations are well behaved in the sense that they always guarantee convergence to the fixed point solution. Contraction iteration is therefore a natural choice of algorithm to begin searching for the fixed point with. However, as the contractions close in on the solution, they tend to slow down and can only guarantee convergence at a linear rate. This is why the inner fixed point polyalgorithm consists of two parts, the second being the faster *Newton-Kantorovich* method.

Newton-Kantorovich

The Newton-Kantorovich (NK) method may be regarded as a functional extension of Newton's method. When we perform contraction iterations, we search for the fixed point $EV_\theta = T_\theta(EV_\theta)$. With Newton-Kantorovich however, we aim at finding the solution to $[I - T_\theta](EV_\theta) = 0$ by using a first-order Taylor expansion. Expressed in operator jargon, we find the zero solution to the nonlinear operator $[I - T_\theta]$[10]. The logic behind this approach is analogous to that of Newton's method, since the contraction mapping $EV_\theta = T_\theta(EV_\theta)$ holds if and only if $[I - T_\theta](EV_\theta) = 0$.

Rust (1988) showed that the operator $[I - T_\theta]$ has a *Fréchet derivative*[11] $[I - T'_\theta]$ on B with a bounded inverse $[I - T'_\theta]^{-1}$. This invertibility comes in

[10] I is the identity operator on a Banach space B and 0 is the zero function.

[11] The Fréchet derivative is a generalization of the idea of linear approximation from functions of one variable, to functions on Banach spaces. Assume V and W are Banach spaces, and $U \subset V$. A function $f : U \mapsto W$ is called Fréchet differentiable at $x \in U$ if there exists a bounded linear operator $T_x : V \mapsto W$ such that

$$\lim_{h \to 0} \frac{||f(x+h) - f(x) - T_x(h)||_w}{||h||_v} = 0$$

handy when solving the Taylor series expansion of $[I - T_\theta](EV_\theta) = 0$. We get

$$[I - T_\theta](EV_\theta) \approx [I - T_\theta](EV_k) + [I - T'_\theta](EV_{k+1} - EV_k) = 0 \tag{2.20}$$

Solving equation (2.20) for EV_{k+1} yields the Newton-Kantorovich iteration formula

$$EV_{k+1} = EV_k - [I - T'_\theta]^{-1}[I - T_\theta](EV_k)$$

Derivation of the separate parts of this formula related to our specific problem will be discussed in detail in Chapter 4.

The switch between the two methods is adapted dynamically as the algorithm runs, but as Newton-Kantorovich only guarantees convergence when its initial value is in the neighbourhood of the fixed point, we need to make sure that the contraction iterations run a sufficient number of iterations before we make the switch.

2.4.4 The outer maximum likelihood algorithm

To uncover the agent's preference u, we wish to find a parameter vector $\hat{\boldsymbol{\theta}}$ that maximizes the full likelihood function $\mathcal{L}^f$ defined by

$$\mathcal{L}^f(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T, d_1, \ldots, d_T|\boldsymbol{\theta}) = \prod_{t=2}^{T} P(d_t|\boldsymbol{s}_t, \boldsymbol{\theta}) p(\boldsymbol{s}_t, |\boldsymbol{s}_{t-1}, \boldsymbol{\theta}) \tag{2.21}$$

where $p(\boldsymbol{s}_t, |\boldsymbol{s}_{t-1}, \boldsymbol{\theta})$ are state transition probabilities that satisfy the Markov property, and $P(d_t|\boldsymbol{s}_t, \boldsymbol{\theta})$ are the conditional choice probabilities. We also define another function which we name the *partial* likelihood function

$$\mathcal{L}^p(\boldsymbol{s}_1, \ldots, \boldsymbol{s}_T, d_1, \ldots, d_T|\boldsymbol{\theta}) = \prod_{t=1}^{T} P(d_t|\boldsymbol{s}_t, \boldsymbol{\theta}). \tag{2.22}$$

The difference between the two is the inclusion of the transition probabilities. We may choose to estimate the transition probabilities as a part of the algorithm using equation (2.21), or precalculate them. Using precalculation, $p(\boldsymbol{s}_t, |\boldsymbol{s}_{t-1}, \boldsymbol{\theta})$ would no longer be a function of $\boldsymbol{\theta}$, and we could simply maximize the partial likelihood instead.

where $||\bullet||_w$ and $||\bullet||_v$ denotes the norms in W and V respectively. If this limit exists, then T_x is said to be the Fréchet derivative of the function f evaluated at point x. For more details on this, we recommend Griffel (2002).

Maximizing the likelihood function

Since no analytical solution to the likelihood function exists, its maximization requires the use of search algorithms. We employ the search directions provided by the BHHH and BFGS methods. Both BHHH and BFGS are quasi-Newton gradient searches, and we give short expositions of the two below. As our line-search algorithm, we use secant iteration. The following section may be skipped by readers that are familiar with this class of optimization algorithms.

Newton- and Quasi-Newton methods

The well-known gradient search algorithm exploits the fact that a non-zero gradient $\nabla\mathcal{L}(\boldsymbol{\theta}_k)$ yields the locally steepest rate of objective function improvement. Thus, the gradient move direction becomes $\Delta\boldsymbol{\theta}_{k+1} = \pm\nabla\mathcal{L}(\boldsymbol{\theta}_k)$ (+ for maximize, − for minimize) and solutions are iterated by

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda_{k+1}\Delta\boldsymbol{\theta}_{k+1}$$

where λ_{k+1} is a good step size obtained by a line search. We find λ_{k+1} by solving (at least approximately)

$$\max \text{ or } \min\ \mathcal{L}(\boldsymbol{\theta}_k + \lambda_{k+1}\Delta\boldsymbol{\theta}_{k+1})$$

Newton's method can be viewed as pursuing the move direction suggested by the second-order Taylor approximation

$$\mathcal{L}'(\boldsymbol{\theta}_k + \lambda_{k+1}\Delta\boldsymbol{\theta}_{k+1}) \doteq \mathcal{L}(\boldsymbol{\theta}_k) + \lambda_{k+1}\nabla\mathcal{L}(\boldsymbol{\theta}_k)\cdot\Delta\boldsymbol{\theta}_{k+1} + \frac{\lambda_{k+1}^2}{2}\Delta\boldsymbol{\theta}_{k+1}\boldsymbol{H}(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta}_{k+1} \tag{2.23}$$

where $\boldsymbol{H}(\cdot)$ is the Hessian matrix. Equation (2.23) is a quadratic function of $\lambda_{k+1}\Delta\boldsymbol{\theta}_{k+1}$, and will thus have a local optimum. This optimum is easily obtained by fixing $\lambda_{k+1} = 1$, differentiating $\mathcal{L}'$ with respect to the components $\Delta\boldsymbol{\theta}_{k+1}$ and setting equal to zero. We find

$$\begin{aligned}\nabla\mathcal{L}'(\Delta\boldsymbol{\theta}_{k+1}) = \nabla\mathcal{L}(\boldsymbol{\theta}_k) + \boldsymbol{H}(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta}_{k+1} &= 0\\ \boldsymbol{H}(\boldsymbol{\theta}_k)\Delta\boldsymbol{\theta}_{k+1} &= -\nabla\mathcal{L}(\boldsymbol{\theta}_k)\end{aligned} \tag{2.24}$$

By solving the linear equation system (2.24) we produce what is known as the *Newton step*, $\Delta\boldsymbol{\theta}_{k+1}$. Using this as the direction in an improving search is known as *Newton's method*.

Now we enter the realm of the quasi-Newton methods. Assume that the Hessian $\boldsymbol{H}(\cdot)$ is non-singular. We may then premultiply (2.24) by the inverse Hessian to obtain

$$\Delta\boldsymbol{\theta}_{k+1} = -\boldsymbol{H}^{-1}(\boldsymbol{\theta}_k)\nabla\mathcal{L}(\boldsymbol{\theta}_k). \tag{2.25}$$

Now, consider some *deflection matrix* $\boldsymbol{D}_k$ that produces modified search directions

$$\Delta\boldsymbol{\theta}_{k+1} = -\boldsymbol{D}_k\nabla\mathcal{L}(\boldsymbol{\theta}_k). \tag{2.26}$$

For the Newton method, obviously, $\boldsymbol{D}_k = \boldsymbol{H}^{-1}(\boldsymbol{\theta}_k)$. Also, notice that setting $\boldsymbol{D}_k = \pm\boldsymbol{I}$ (the identity matrix) produces the simple gradient search direction.

Quasi-Newton methods work with a deflection matrix that approximates the inverse Hessian, $\boldsymbol{H}^{-1}(\boldsymbol{\theta}_k)$, of Newton's method. Unlike the full Newton's method, however, $\boldsymbol{D}_k$ is built up from prior search results using only first derivatives.

BHHH

Since the likelihood function $\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}, EV_\theta)$, maximization using the Newton method would have required calculation of the Hessian of the expected value function $\boldsymbol{H}(EV_\theta)$, which is computationally difficult. The BHHH method (Berndt et al., 1974) employs a different strategy, and requires only the first derivatives of the log-likelihood function.

The method is based on the *information equality* identity, which states that the expectation of the tensor product[12] of the log-likelihood function's gradient equals the negative of the expectation of the log-likelihood function's Hessian, when both are evaluated at the "true" parameter vector $\boldsymbol{\theta}^*$, that is,

$$\mathbb{E}\left[\nabla\mathcal{L}_l(\boldsymbol{\theta}^*, EV_{\theta^*}) \otimes \nabla\mathcal{L}_l(\boldsymbol{\theta}^*, EV_{\theta^*})\right] = -\mathbb{E}\left[\boldsymbol{H}(\mathcal{L}_l(\boldsymbol{\theta}^*, EV_{\theta^*}))\right],$$

[12]The tensor product of vectors is also known as an *outer product*, and produces a matrix such that if $\boldsymbol{u} = (u_1, u_2, \ldots, u_m)^T$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)^T$ the outer product is

$$\boldsymbol{u} \otimes \boldsymbol{v} = \begin{bmatrix} u_1v_1 & u_1v_2 & \ldots & u_1v_n \\ u_2v_1 & u_2v_2 & \ldots & u_2v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_mv_1 & u_mv_2 & \ldots & u_mv_n \end{bmatrix}$$

where $\mathcal{L}_l$ denotes the log-likelihood.

The BHHH algorithm uses the standard iteration:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda_{k+1}\Delta\boldsymbol{\theta}_{k+1},$$

which we've seen yields the Newton method iteration when using equation (2.25) and $\lambda_{k+1} = 1$, and quasi-Newton methods when using (2.26). Now, let

$$\boldsymbol{A}_k = \nabla\mathcal{L}_l(\boldsymbol{\theta}_k, EV_\theta) \otimes \nabla\mathcal{L}_l(\boldsymbol{\theta}_k, EV_\theta)$$

i.e. $\boldsymbol{A}_k$ is the tensor product of the gradient of the log-likelihood function. Then with the deflection matrix $\boldsymbol{D}_k = -\boldsymbol{A}_t^{-1}$, we arrive at the BHHH iteration

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \lambda_{k+1}\boldsymbol{D}_k\nabla\mathcal{L}(\boldsymbol{\theta}_k).$$

Notice now that the search direction is $\Delta\boldsymbol{\theta}_{k+1} = \boldsymbol{A}_k^{-1}\nabla\mathcal{L}(\boldsymbol{\theta}_k)$.

Some further comments are in order. Firstly, since BHHH only requires first derivatives of the log-likelihood function, one only needs to calculate first derivatives of the expected value function. These derivatives can be calculated analytically using the Fréchet derivative we obtain during Newton-Kantorovich:

$$\nabla EV_\theta = \left[I - T'_\theta\right]^{-1}\nabla T_\theta(EV_\theta) \tag{2.27}$$

We calculate these derivatives during the final Newton-Kantorovich iteration of the inner fixed point polyalgorithm.

BFGS

BFGS[13] is a quasi-Newton search that has proven to be quite efficient. In particular, its rate of convergence is well-known to be higher than that of BHHH, in that it generally provides more accurate estimates of the Hessian of the likelihood function. However, BFGS may fail to converge when the initial estimate of $\hat{\boldsymbol{\theta}}$ is far away from the "true" parameter vector $\boldsymbol{\theta}^*$. Thus, we suggest starting the maximization with BHHH, and then switching to

[13]Named after its four major contributors; C. Broyden, R. Fletcher, D. Goldfarb, and D. Shanno.

BFGS when the former algorithm starts to exhibit linear rates of convergence, indicating that the maximization process has entered a domain of attraction.

The BFGS updating formula is given by

$$\boldsymbol{D}_{k+1} = \boldsymbol{D}_k + \left(\mathbf{1} + \frac{\boldsymbol{g}\boldsymbol{D}_k\boldsymbol{g}}{\boldsymbol{d}\cdot\boldsymbol{g}}\right)\frac{\boldsymbol{d}\boldsymbol{d}^\mathsf{T}}{\boldsymbol{d}\cdot\boldsymbol{g}} - \frac{\boldsymbol{D}_k\boldsymbol{g}\boldsymbol{d}^\mathsf{T} + \boldsymbol{d}\boldsymbol{g}^\mathsf{T}\boldsymbol{D}_k}{\boldsymbol{d}\cdot\boldsymbol{g}}$$

where $\boldsymbol{d} = \boldsymbol{\theta}_{k+1} - \boldsymbol{\theta}_k$ and $\boldsymbol{g} = \nabla\mathcal{L}(\boldsymbol{\theta}_{k+1}) - \nabla\mathcal{L}(\boldsymbol{\theta}_k)$. The initial iteration uses $\boldsymbol{D}_0 = -\boldsymbol{I}$, which results in a first search direction

$$\Delta\boldsymbol{\theta}_1 = -\boldsymbol{D}_0\nabla\mathcal{L}(\boldsymbol{\theta}_0) = I\nabla\mathcal{L}(\boldsymbol{\theta}_0) = \nabla\mathcal{L}(\boldsymbol{\theta}_0),$$

This is simply the gradient search direction. Obviously, $\boldsymbol{\theta}_0$ is the parameter estimate produced in the last iteration of BHHH.

Line search

All that remains now is to specify a line search algorithm to give us successive estimates of the step-size parameter λ. Specifically, we are looking for step-sizes λ^* that maximizes the univariate function $f(\lambda) = \mathcal{L}^f(\boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta})$. One way of doing this is to find λ^* iteratively by solving the equation $\partial f(\lambda)/\partial\lambda = 0$ using Newton's method, which yields iterations on the form

$$\lambda_{k+1} = \lambda_k - \frac{\partial f(\lambda_k)/\partial\lambda}{\partial^2 f(\lambda_k)/\partial\lambda^2}.$$

Unfortunately, using Newton's method directly would require the second derivative of $f(\lambda)$, which in turn would require the second derivative of EV_θ, thereby defeating the purpose of employing BHHH. We suggest the use of *secant iteration* instead. Secant iterations is simply Newton's method with a finite difference approximation of $\partial^2 f(\lambda_k)/\partial\lambda^2$. This approximation is

$$\partial^2 f(\lambda_k)/\partial\lambda^2 \approx \frac{\partial f(\lambda_k)/\partial\lambda - \partial f(\lambda_{k-1})/\partial\lambda}{\lambda_k - \lambda_{k-1}}.$$

Thus, the line search iteration becomes

$$\lambda_{k+1} = \lambda_k - \frac{(\lambda_k - \lambda_{k-1})\partial f(\lambda_k)/\partial\lambda}{\partial f(\lambda_k)/\partial\lambda - \partial f(\lambda_{k-1})/\partial\lambda}.$$

Finally, to terminate the line search, one must employ some stopping criteria. We have experienced good results by simply continuing the line search while the percentage increase in the likelihood function exceeds some user-specified tolerance, subject to constraints on minimum and/or maximum number of iterations.

Chapter 3

NFXP applied to Hydropower

The previous chapter dealt with the NFXP algorithm in a very general manner. However, what should be gleaned for an initial first reading is that NFXP requires that one solves an infinite-horizon dynamic programming problem by finding a fixed point to a contraction mapping of the value function, and that doing so requires *problem stationarity*.

Hydropower production scheduling has traditionally not been treated as a stationary problem. Indeed, most scheduling tools take on a finite-horizon view and solve the scheduling problem either using deterministic or stochastic dynamic programming (for a good summary of such methods, see Nandalal and Bogárdi (2007)). Given that the price and inflow expectations of a hydropower producer are likely to be highly non-stationary, this seems to be a very sensible approach. At the same time, it represents a potential difficulty to applying structural estimation methods such as NFXP. Fortunately, we have been able to develop an approach that allows us to view the scheduling problem as stationary, while still allowing the expectations of the hydropower producer to change with time.

Before proceeding, we would like to clarify that the choice of solution method for the inner loop is a result of two things: 1) we need a model for which we can derive partial derivatives with respect to the model parameters, and 2) additionally, the model must be able to handle the error term associated with the application of NFXP. These two features are not something that

regular hydropower production planning models usually take into consideration, and suggests therefore the use of the contraction mapping formula proposed by Rust (1994). Further, we recommend that readers unfamiliar with basics of hydropower production and scheduling read the cursory introduction given in appendix A.

3.1 Achieving stationarity

Hydropower production scheduling can be considered as a problem of *stochastic control*. Our problem, on the other hand, is a problem of *inverse stochastic control*. With respect to stationarity, this difference creates an advantage in our favor: the existence of an *error term* ϵ_t. Recalling the discussion in section 2.3, Rust (1988) argued that the error terms may be viewed as unobserved state variables. Now, expanding upon that idea, we assume that decisions made by the hydropower producer that deviate from the solution of the underlying stochastic control problem, may under stationarity assumptions be explained by ϵ. What then remains is a way to naturalize the idea of stationarity to the problem of hydropower production.

Consider the plots of weekly inflow over nine years reported by two hydropower producers in Norway in figures E.1 and E.3 in appendix E. Clearly, for the former, seasonality plays an important role. For the latter, however, seasonality is much less pronounced. What is also quite clear, is that time of year seems to be an important factor in explaining inflow. Indeed, running time-series regression with time-varying volatility, we find that week number is a good variable in explaining the first, whereas the latter is approximately a white noise process. For the data available to us, we generally observe that, given week number, we may subtract the mean of the inflow and fit normal or log-normal distributions to the remainder. As we will show in the following subsection, these observations suggest approaching stationarity in a different manner.

3.1.1 The cyclical MDP

Recall the Bellman equation for infinite-horizon stochastic dynamic programming problems:

$$V(\boldsymbol{s}) = \max_{d \in D(\boldsymbol{s})} \left\{ u(\boldsymbol{s}, d) + \beta \int V\left(\boldsymbol{s}'\right) p(d\boldsymbol{s}' | \boldsymbol{s}, d) \right\}$$
$$\delta(\boldsymbol{s}) = \arg\max_{d \in D(\boldsymbol{s})} \left\{ u(\boldsymbol{s}, d) + \beta \int V\left(\boldsymbol{s}'\right) p(d\boldsymbol{s}' | \boldsymbol{s}, d) \right\}.$$

As discussed in section 2.1.1, these equations arise when one is willing to assume stationarity. Specifically, one assumes that the future looks exactly the same in state $\boldsymbol{s}_t$ at time t and $\boldsymbol{s}_{t+k}$ at time $t+k$ provided that $\boldsymbol{s}_t = \boldsymbol{s}_{t+k}$.

Consider now the case of an MDP where this stationarity condition does not hold, that is, one may *not* assume that, for all t, the future looks the same as long as $\boldsymbol{s}_t = \boldsymbol{s}_{t+k}$ for **any** k. This is clearly the case for hydropower production scheduling. However, we have found that, conditional upon week of the year, we are able to sufficiently describe the amount of inflow to reservoirs using simple probability measures. It then seems natural to consider a somewhat weaker form of stationarity. For instance, say we assume that for all t, the future looks the same as long as $\boldsymbol{s}_t = \boldsymbol{s}_{t+k}$ for **some** k, specifically, for all $k = \phi i, i = 0, 1, 2, \ldots$ where we call ϕ the *modulus*. Then, by letting time (week) be a state variable, we can reduce a non-stationary problem to an approximately stationary one - one in which large deviations from the stationary expectations is explained by an extreme value distribution.

To illustrate, consider the case where time is discretized into weeks and $\boldsymbol{s}_t = \boldsymbol{s}_{t+52\times i}$. Then, $\phi = 52$ and the MDP is stationary given the week of the year. We will do exactly this in our problem formulation, and we find it convenient to define a particular set of states here. Let

$$\boldsymbol{\zeta} = \{\boldsymbol{s}, w\},$$

where w denotes the current week of the year and $\boldsymbol{s}$ is a vector of all other states.

With this new state formulation, we now proceed to develop a model for the NFXP algorithm. Since it seems to be the most interesting and natural one, we have chosen to illustrate the algorithm using a utility function for the profit-maximizing agent.

3.2 Model

Consider the following (simplified) utility function:

$$u(x, y, w, d, \boldsymbol{\theta}) = q(x) \sum_{d \in D(x)} (y - \theta_1)$$

We will expand upon this function later, but for now, we keep things simple. This function represents the total profit earned in a single week w. Assume now, that we are at the beginning of that week, our current reservoir level is x, and the expected average market price is y. At reservoir level x, the turbine power is $q(x)$ and the decision space is given by $D(x)$. d represents the number of hours we decide to produce the following week. θ_1 is a cost parameter we want to estimate, and which represents a combination of actual costs (in general quite small for hydropower producers), and loss in option value due to production.

To summarize; price less cost is summed over d hours and multiplied with plant power, q, which result in net profit for week w.

We now proceed to refine and formalize what was just introduced. Following this we are going to derive all the formulas[1] required to run the NFXP algorithm on the given utility function.

3.2.1 States

Let $\boldsymbol{\zeta} = \{x, y, w\}$ be the total set of states, where x denotes *reservoir level*, y denotes *average price*, and w denotes *week*. Further, let $\boldsymbol{\zeta}(d)$ denote the state at the beginning of week $w + 1$ after decision d has been made in week w. Implicitly, this means that w is increased by one[2] and x is reduced by an amount corresponding to decision d. Finally, let $\boldsymbol{\zeta}^i$ denote a certain realization of $\{x, y, w\}$ to distinguish it from some other realization $\boldsymbol{\zeta}^j$.

Sets, variables & indexes

X	set of all discretized reservoir levels
Y	set of all discretized price levels
W	set of all weeks $\{1, \ldots, 52\}$

[1] Except for a few which are quite elaborate and therefore regarded as appendix material.

[2] Moving to the next week is *modulus* 52, i.e. the week after week 52 is week 1.

Z	set of all states ($Z = X \cup Y \cup W$)
$\bar{X}$	maximum reservoir level
x_t	reservoir level at time t
y_t	average price at time t
w_t	week number at time t
$\boldsymbol{\zeta}_t$	state at beginning at time t
$\boldsymbol{\zeta}^i$	state realization i
$\boldsymbol{\zeta}(d)$	state at beginning of week $w+1$ after making decision d

The choice of time interval has already been discussed, but as we saw in Chapter 2, we also need to discretize reservoir level and price. Due to the *curse of dimensionality*, choosing a too fine resolution (i.e. too many states) will make the problem computationally difficult, whereas choosing a too coarse resolution may result in loss of vital information or inability to capture essential parts of the decision process. Specifically, the discretization of the reservoir has to be fine enough to allow at least the most common production decisions to result in a drop to a lower discretized state.[3] With price we have more freedom to choose level of discretization, as there are no limitations other than less realistic price scenarios.

3.2.2 Inflow

In addition to being changed by the amount of production, reservoir levels are affected by precipitation and inflow. The spring flood, for instance, can in some cases fill an entire reservoir in a matter of days. Thus, it is natural to expect that the decisions made by hydropower producers are very much affected by their expectations of amount and timing of inflow. These expectations are captured by transition matrices that we will discuss later in this chapter. For now, we introduce some notation.

Variables

dx'_w	change in reservoir level due to inflow in week w (continuous)
$\Delta x'_w$	change in reservoir level due to inflow in week w (discrete)

[3] We may still end up at the same level due to inflow, but that's a matter we will discuss later.

3.2.3 Decisions

One possible way to characterize decisions is by the number of hours produced per week. Each week has $24 \times 7 = 168$ hours, and so a possible decision variable could simply be $d = \{0, \ldots, 168\}$. This, though, is not necessarily realistic nor practical. We can find little or no evidence that hydropower producers make their production decisions at that level of detail. Furthermore, having that many decisions to compare would be a computational nightmare. A more logical approach is to group hours together in blocks, and then letting the decision be whether or not to produce certain blocks. From here on out, we call these blocks *production blocks*.

We define a production block as a set of hours with constant price, for which the producer can choose to produce or not. If he chooses to produce he must produce the whole block, or desist from it completely. We assume that the producer will start with the block he expects to be most profitable, i.e. the one with the highest combined block price, and works his way down. If there are D blocks in total in a week, the producer may decide to produce the d most profitable blocks, where $d \in (0, 1, \ldots, D)$. Notice that the producer may choose to produce no blocks, i.e. $d = 0$.

The block sizes are created dynamically as the algorithm proceeds, according to the following:

- The total number of production blocks is D. For each week, the producer has $|D + 1|$ decisions to choose from (unless the reservoir level is low, in which case the producer can only select a subset of the production blocks).
- All spot prices for a given week are sorted from high to low, and the total price span is divided into D equally sized intervals.
- Each hour is linked to its corresponding price interval and the number of hours in each interval determines the size of the production block.

The use of production blocks reduces the decision space, while capturing the fact that electricity prices can vary significantly through a single day. When running the algorithm, we typically get several small (narrow) blocks first, representing the few hours with abnormally high prices, while the remaining blocks are wider, representing less demand-intensive periods of the week.

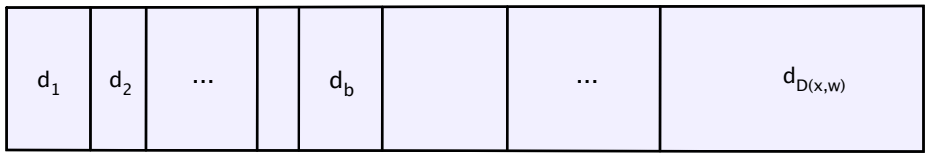

Figure 3.1: Illustration of production blocks

Due to the fact that the block structure varies from week to week and that the number of decisions are limited by the reservoir level, we introduce $D(x, w)$ as a measure of the decision space associated with a reservoir level x in week w. The reasoning behind $D(x, w)$'s dependency on x should be obvious - if the reservoir level is very low, you simply can't produce as much as you may want to. w, on the other hand, is directly related to the dynamic nature of block sizes. As the price distribution within a week varies with the seasons, this must also be true for the production blocks, which explains why $D(x, w)$ is a function of week.

We use b as an index for a specific production block. The coefficient c_{bw}, which we call *block price coefficients*, represent what we must multiply the average week price y with in order to get the correct price for block b. Lastly, h_{bw} is the length in hours of the corresponding block.

Sets, variables & indexes

D	total number of production blocks
$D(x, w)$	number of production blocks that are possible to produce in week w given reservoir level x
d	number of blocks to produce (decision)
b	block number
c_{bw}	block price coefficient for block b in week w
h_{bw}	length of block b in week w

In summary; the producer can choose d production blocks on the interval $[0, D(x, w)]$ in week w. The blocks are of length h_{bw} and yield a price of $c_{bw}y$, respectively.

3.2.4 Utility function

We are now ready to define a complete utility function for the *profit-maximizing hydropower producer*:

$$u(\boldsymbol{\zeta}, d, \boldsymbol{\theta}) = q(x, d, w) \sum_{b=1}^{d} (c_{bw} y - \theta_1)\, h_{bw} \tag{3.1}$$

Recall, $\boldsymbol{\zeta} = \{x, y, w\}$. Comparing this to the preliminary utility-function described at the beginning of this chapter, one can see that there are several differences. Instead of simply a price y, we now use $c_{bw}y$, which is the relevant block price coefficient multiplied with an average week price, thus giving the correct price in block b given week price y. The cost parameter θ_1 is then subtracted from this price and the net is multiplied with the block length h_{bw}, resulting in a value in $€/MW$. Multiplying this with plant power q gives us the total profit in week w in €.

Power function

In equation (3.1) one can see that the power function q is not only dependent on x, but also on decision d and week w. To see why this is so, recall that power q is dependent on turbine head[4]. Since we are making decisions on a weekly basis, we suggest using the average head throughout the week as input to the power function $q(\bar{x})$. To make matters easier, we use the level x instead of head h as input to q. This is completely safe, as there is a perfect deterministic relationship between head and reservoir level, given by the geometry of the reservoir. Average reservoir level in one week is approximated by

$$\bar{x} = x + \frac{dx'_w - \rho(x, d, w)}{2} \tag{3.2}$$

where dx'_w is the expected inflow given by the state transition probability $p(x_t|x_{t-1}, \boldsymbol{\theta})$. The other new variable, $\rho(x, d, w)$, is the total production of week w given reservoir level x and decision d. The dependency on w is due to the nature of the decision discretization, where d blocks in one week isn't necessarily the same number of hours as d blocks in another week.

[4]Turbine head is the vertical distance from the turbine up to the surface of the reservoir, and is an absolute measure of turbine pressure. Higher pressure equals higher efficiency per m^3 of water.

There is a recursive relationship between q and ρ, in the sense that q is a function of $\bar{x}$, which in turn is dependent on ρ, given by

$$\rho = q(x, d, w) \sum_{b=1}^{d} h_{bw}. \tag{3.3}$$

By combining equations (3.2) and (3.3) we get

$$\bar{x} = x + \frac{dx'_w - q(x, d, w) \sum_{b=1}^{d} h_{bw}}{2}, \tag{3.4}$$

thus q depends on x, d and w, as well as itself. Substituting (3.4) into a function $q(\bar{x})$ and solving for q, we can find the appropriate power value. If one assumes a simple power function, it is trivial to find an explicit solution to this problem.[5] The power function varies from power plant to power plant, but as long as we know at least the approximate geometry of the reservoir, we can estimate a good model using regression analysis. Throughout the remainder of this paper, we will use the notation $q\left(x + \frac{dx}{2}, d, w\right)$ to symbolize the power of the power plant given an initial reservoir level x, inflow dx and a production level given by equation (3.3).

3.2.5 Expected value function

In section 2.4 we introduced the contraction mapping version of the expected value function

$$EV_\theta(\boldsymbol{\zeta}, d) = T_\theta(EV_\theta)(\boldsymbol{\zeta}, d) = \int \log \left[\sum_{d' \in D(\boldsymbol{\zeta}')} \psi(\boldsymbol{\zeta}', d', \boldsymbol{\theta}) \right] p(d\boldsymbol{\zeta}' | \boldsymbol{\zeta}, d, \boldsymbol{\theta}) \tag{3.5}$$

where

$$\psi(\boldsymbol{\zeta}', d', \boldsymbol{\theta}) = \exp\left\{ u(\boldsymbol{\zeta}', d', \boldsymbol{\theta}) + \beta EV_\theta(\boldsymbol{\zeta}', d') \right\} \tag{3.6}$$

We will now work our way to a complete formulation of the expected value function for the proposed utility function. In this case, the state vector $\boldsymbol{\zeta}'$ is defined as

$$\boldsymbol{\zeta}' \stackrel{def}{=} \left\{ \min[\bar{X}, x + dx'], y + dy', w \right\}.$$

[5] A linear or quadratic power function is a good approximation in most cases. Appendix C.4 demonstrates the transition from a quadratic power function $q(\bar{x})$ to an explicit expression for $q(x, d, w)$.

This state vector expresses the changes in states that may arise while in week w. The reasoning behind the min-expression is simple; if the inflow, dx', is sufficiently large, there may be spillover, and the new reservoir level will be $\bar{X}$. If not, the new level will be the current level x plus inflow dx'. Average week price will increase by dy' (which may be negative).

The transition probability $p(d\boldsymbol{\zeta}'|\boldsymbol{\zeta}, d, \boldsymbol{\theta})$ in equation (3.5) reflects the probability of a state change from $\{x, y\}$ to $\{x', y'\}$, that is, the probability of inflow dx' and a price change dy' during week w. Both are independent of decision d, as we have assumed that the producers are price-takers, thus

$$p(d\boldsymbol{\zeta}'|\boldsymbol{\zeta}, d, \boldsymbol{\theta}) = p(d\boldsymbol{\zeta}'|\boldsymbol{\zeta}, \boldsymbol{\theta}) \qquad \Longrightarrow \qquad EV_\theta(\boldsymbol{\zeta}, d) = EV_\theta(\boldsymbol{\zeta}). \tag{3.7}$$

Since the transition probabilities are independent of the decision d, this will also be true for the expected value function (3.5). In addition, the transition probability is a joint probability consisting of two independent elements: inflow probability $p(dx'|\boldsymbol{\zeta}, \boldsymbol{\theta})$ and price change probability $p(dy'|\boldsymbol{\zeta}, \boldsymbol{\theta})$. The first is independent of all states except w, and the second is independent of x. This gives us the relationship

$$p(d\boldsymbol{\zeta}'|\boldsymbol{\zeta}, \boldsymbol{\theta}) = p(dx'|w, \boldsymbol{\theta})p(dy'|y, w, \boldsymbol{\theta}). \tag{3.8}$$

The fact that the expected value function is independent of the decision d may seem strange. One might argue that a decision d reduces the reservoir level and thus affects future value. This is naturally correct, but it is captured in a different way. Recall the notation $\boldsymbol{\zeta}(d)$, introduced in section 3.2.1, which expresses the effect decision d will have on the state of week $w + 1$:

$$\boldsymbol{\zeta}(d) \overset{def}{=} \left\{\min\left[\bar{X}, x + dx' - \rho(x, d, w)\right], y + dy', w + 1\right\}.$$

The new reservoir level will be the lesser of $\bar{X}$ and $\{x + dx' - \rho(x, d, w)\}$, where $\rho(x, d, w)$ is the total production throughout the week. As before, average week price will increase by dy' (which still may be negative) and w is incremented by one to reflect the move to the next week.

We are now ready to substitute the utility function into equations (3.5) and (3.6). This yields

$$EV_\theta(\boldsymbol{\zeta}) = \iint \log\left[\sum_{d=0}^{D(x+dx',w)} \psi(\boldsymbol{\zeta}', d, \boldsymbol{\theta})\right] p(dx'|w, \boldsymbol{\theta})p(dy'|y, w, \boldsymbol{\theta}) \tag{3.9}$$

with

$$\psi(\boldsymbol{\zeta}', d, \boldsymbol{\theta}) = \exp\left\{ q(x + \frac{dx'}{2}, d, w) \sum_{b=1}^{d} \left[c_{bw}(y + dy') - \theta_1 \right] h_{bw} + \beta EV_\theta(\boldsymbol{\zeta}'(d)) \right\}. \tag{3.10}$$

Converted to a discretized state space we get

$$EV_\theta(\boldsymbol{\zeta}) = \mathcal{P} \times \log\left[\sum_{d=0}^{D(x,w)} \exp\left\{ q(x, d, w) \sum_{b=1}^{d} \left[c_{bw} y - \theta_1 \right] h_{bw} + \beta EV_\theta(\boldsymbol{\zeta}(d)) \right\} \right], \tag{3.11}$$

where the elements of the Markov transition matrix $\mathcal{P}$ are the joint transition probabilities $p(dx'|w, \boldsymbol{\theta})p(dy'|y, w, \boldsymbol{\theta})$. $EV_\theta(\boldsymbol{\zeta})$ becomes vector in $\mathbb{R}^{|Z|}$, where $|Z|$ is defined as $|Z| = |X| \times |Y| \times |W|$.

Before we discuss the implementation of the NFXP algorithm, there is one more element in equation (3.11) that requires our attention.

3.2.6 The Markov transition matrix

One possible way to determine $\mathcal{P}$ is to include the transition probabilities, i.e. the elements of $\mathcal{P}$, in the parameter vector $\boldsymbol{\theta}$ and estimate them as part of the maximum likelihood routine[6]. Doing this would allow us to calculate the actual *expectations* of the hydropower producers, both with respect to price and inflow. This approach is quite common in the literature, and while it may seem like an excellent idea, there are a few issues associated with it. The applications of NFXP that can be found in the literature are typically to "easier" problems - problems where the stochastic nature of transitions is significantly simpler. Typically, the number of parameters estimated range from about three to ten. In our case, however, that number is much larger.

First of all, we have two random state variables (reservoir level and price) whereas Rust (1987), in his *bus engine replacement* problem, had only one. This instantly doubles the number of transition probability parameters. More important however, is the discretization of time into weeks. As price and inflow clearly fluctuates throughout the year, we need one separate transition matrix for each week, increasing the number of parameters by a factor of 52. To make matters even worse; in order to describe the process in a

[6] See e.g. Rust (1987).

satisfying way, we would require a higher number of transition probabilities per variable per week than Rust (1987) had for his single variable. All this combined leads to a parameter vector $\boldsymbol{\theta}$ with a dimension somewhere between 600 and 2000, depending on the resolution of the state space. Solving this problem for a large state space would clearly be intractable. Thus, we choose to estimate these probabilities in advance based on historical data.

One might argue that making our own assumptions about the distribution of prices and inflow, and then (essentially) imposing these assumptions upon the hydropower producer, will give us biased estimates for the parameters in the utility function. We freely admit that this might be the case, but keep in mind that our goal is to evaluate structural estimation as an approach to the hydropower decision problem, not *precisely* estimate the utility function parameters. Thus, we consider our transition matrices to be sufficient for the purpose of this thesis. For details on calculation of these matrices, see appendix E.

Chapter 4

Implementation

We have used MATLAB to implement NFXP. MATLAB has highly efficient routines for matrix and vector manipulation, making it an ideal programming language for our purposes. This chapter discusses the actual implementation and the adaptations that are necessary to translate the mathematics of chapter 3 into code. We also provide an algorithm that leverages the structure of the Fréchet derivative, thereby letting us quickly solve large equation systems that are needed during Newton-Kantorovich iterations. In particular, we show that the asymptotic running time of this algorithm is $O(XY^2T\kappa)$[1], a significant improvement over, say, naïve Gauss-Jordan or Gauss-Seidel, which have asymptotic running times of $O(X^3Y^3T^3)$ and $O(X^2Y^2T^2\kappa)$ respectively. Finally, we give a simulation-based analysis of the correctness of the algorithm.

4.1 The inner loop

Contraction iterations

The inner loop's main objective is to find the fixed point of the contraction mapping

$$T_\theta(EV_\theta)(\boldsymbol{\zeta}, d) = \mathcal{P} \times \log \left[\sum_{d' \in D(\boldsymbol{s})} \psi(\boldsymbol{\zeta}, d', \boldsymbol{\theta}) \right],$$

[1] κ is a function that has an inverse relationship to the rate of convergence

and compute its partial derivatives w.r.t. the elements of $\boldsymbol{\theta}$. These derivatives are needed to calculate the partial derivatives of the log-likelihood function, which are used in the maximization process of the outer loop.

The key to finding the fixed point is to compute the discretized expected value vector $EV_\theta(\boldsymbol{\zeta})$ by first solving

$$\begin{aligned} EV_{k+1}(\boldsymbol{\zeta}) &= T_\theta(EV_k)(\boldsymbol{\zeta}) \\ &= \mathcal{P} \times \log \left[\sum_{d=0}^{D(x,w)} \exp \left\{ q(x,d,w) \sum_{b=1}^{d} \left[c_{bw} y - \theta_1 \right] h_{bw} + \beta EV_k(\boldsymbol{\zeta}(d)) \right\} \right] \end{aligned}$$

iteratively until we get sufficiently close to the fixed point of T_θ, and then switching to Newton-Kantorovich iterations. For each iteration of $EV_{k+1} = T_\theta(EV_k)$ we must evaluate the value function

$$q(x,d,w) \sum_{b=1}^{d} \left[c_{bw} y - \theta_1 \right] h_{bw} + \beta EV_k(\boldsymbol{\zeta}(d)) \tag{4.1}$$

for all states in Z, which is a computationally expensive. To avoid two for-loops, we contain these expressions within vectors, and utilize MATLAB's considerable efficiency at vector addition and multiplication. Testing shows

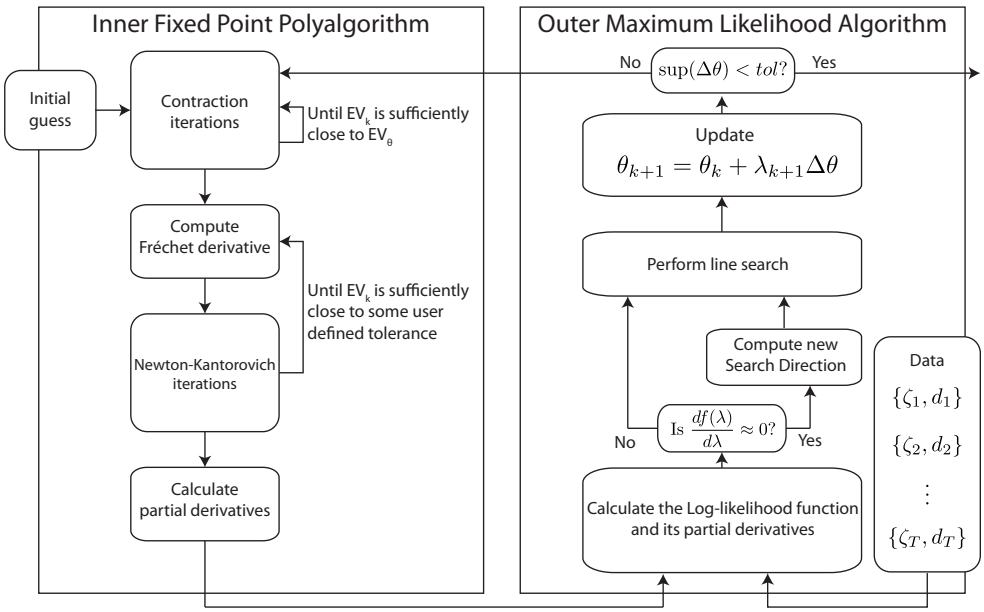

Figure 4.1: The main components of our implementation of NFXP.

that this reduces time spent on each contraction iteration significantly - the larger the state space, the larger the relative improvement

Newton-Kantorovich iterations

The Fréchet derivative T'_θ is required to perform the Newton-Kantorovich iterations:

$$EV_{k+1} = EV_k - \left[I - T'_\theta\right]^{-1} [I - T_\theta](EV_k).$$

For a $|Z|$-dimensional vector EV_k, the Fréchet derivative becomes a $|Z|\times|Z|$ matrix where element (i, j) is given by[2]

$$\frac{\partial T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)}{\partial EV_\theta(\boldsymbol{\zeta}^j)} = \mathcal{P} \times \beta \frac{\sum_{d \in F} \psi(\boldsymbol{\zeta}^i, d, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x,w)} \psi(\boldsymbol{\zeta}^i, d', \boldsymbol{\theta})},$$

where F is the set of all decisions d such that $\boldsymbol{\zeta}^i(d) = \boldsymbol{\zeta}^j$. Translating this expression into computer code and constructing the matrix $[I - T'_\theta]$ is straighforward. The difficulties of Newton-Kantorovich first arise when one tries to invert $[I - T'_\theta]$, which, depending on the size of the state space, can be quite costly[3]. To deal with this, we exploit the structure of $[I - T'_\theta]$ and develop an iterative method for solving large systems of linear equations. First, though, we explore the structure of $EV_\theta(\boldsymbol{\zeta})$, since this affects the structure of $[I - T'_\theta]$.

$EV_\theta(\boldsymbol{\zeta})$ is a vector representing the expected value for all different states. We now introduce the notation $EV_\theta(y, x, w)$, which is the expected present value of observing price y and reservoir level x in week w. Further, we describe y and x using indexes $\boldsymbol{y} = \{1, 2, ..., Y\}$ and $\boldsymbol{x} = \{1, 2, ..., X\}$ instead of actual prices and reservoir levels. This is just to simplify the exposition of EV_θ.

[2] We derive this in appendix C.5.

[3] When we use (relatively) realistic discretizations, $[I - T'_\theta]$ is approximately a $30'000 \times 30'000$ matrix.

$EV_\theta(\boldsymbol{\zeta})$ then looks like this:

$$\left.\begin{array}{l}\left.\begin{array}{l}EV_\theta(1,1,1)\\EV_\theta(2,1,1)\\\quad\vdots\\EV_\theta(Y,1,1)\end{array}\right\}\text{price block}\\\\EV_\theta(1,2,1)\\\quad\vdots\\EV_\theta(Y,2,1)\\\quad\vdots\\EV_\theta(Y,X,1)\end{array}\right\}\text{level block}\\\\EV_\theta(1,1,2)\\\quad\vdots\\EV_\theta(Y,X,2)\\\quad\vdots\\EV_\theta(Y,X,T)\end{array}\right\}EV_\theta(\boldsymbol{\zeta})\tag{4.2}$$

The full set of price states is nested within each reservoir level. The full set of reservoir levels is in turn nested within each week, thus, for all weeks, creating a vector that covers all possible states. The result is a block structure, with *price blocks* and larger *level blocks* within the full vector $EV_\theta(\boldsymbol{\zeta})$.

4.1.1 Sparse matrix solver

The Newton-Kantorovich iteration

$$EV_{k+1} = EV_k - \left[I - T'_\theta\right]^{-1}[I - T_\theta](EV_k)$$

requires inversion of $[I - T'_\theta]$. Now, imagine we instead try to solve the equivalent problem

$$\left[I - T'_\theta\right]\boldsymbol{u} = \boldsymbol{b} \tag{4.3}$$

for $\boldsymbol{u}$, where $\boldsymbol{u} = EV_{k+1} - EV_k$ and $\boldsymbol{b} = [I - T_\theta](EV_k)$. We would never actually invert $[I - T'_\theta]$, but rather use some of MATLAB's excellent built in solvers to solve the system of equations. However, due to the size of

$[I - T'_\theta]$, we find that even these procedures spend an unreasonable amount of time to find the solution.

Consider now the matrix $[I - T'_\theta]$. I is the identity matrix, and T'_θ is the Fréchet derivative of $T_\theta(EV_\theta)$ with respect to EV_θ. With EV_θ as described above, $[I - T'_\theta]$ looks like this:

$$\begin{bmatrix} I & \begin{bmatrix} -T_\theta^1 \end{bmatrix} & 0 & \dots & 0 \\ 0 & I & \begin{bmatrix} -T_\theta^2 \end{bmatrix} & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & 0 \\ 0 & \vdots & \ddots & I & \begin{bmatrix} -T_\theta^{51} \end{bmatrix} \\ \begin{bmatrix} -T_\theta^{52} \end{bmatrix} & 0 & \dots & 0 & I \end{bmatrix}$$

where the submatrix

$$\begin{bmatrix} -T_\theta^w \end{bmatrix}$$

is the Fréchet derivative for all realizations of x and y while holding w constant. To fully understand this structure, recall that each element $\{i, j\}$ of T'_θ represents

$$\frac{\partial T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)}{\partial EV_\theta(\boldsymbol{\zeta}^j)},$$

which is zero for (at least) all $\{i, j\}$ that do not satisfy $w^i + 1 = w^j$. This contributes to a shift of the submatrices of one level block to the right. A more thorough explanation of this property can be found within the derivation of the Fréchet derivative in appendix C.5.

As can be seen above, $[I - T'_\theta]$ has a highly sparse structure which we can exploit in order to speed up computation. We define $\boldsymbol{u}_w$ and $\boldsymbol{b}_w$ as the

elements of $\boldsymbol{u}$ and $\boldsymbol{b}$ which corresponds to week w. Using this structure, we obtain

$$\begin{aligned}
\boldsymbol{u}_1^{k+1} &= T_\theta^1 \boldsymbol{u}_2^k + \boldsymbol{b}_1, \\
\boldsymbol{u}_2^{k+1} &= T_\theta^2 \boldsymbol{u}_3^k + \boldsymbol{b}_2, \\
&\vdots \\
\boldsymbol{u}_{52}^{k+1} &= T_\theta^{52} \boldsymbol{u}_1^k + \boldsymbol{b}_{52}.
\end{aligned}$$

These equations are solved iteratively while measuring the change in $\boldsymbol{u}$ from one iteration to the next. The procedure continues until the change is smaller than some user defined tolerance, and an approximation to the solution of equation (4.3) is found.

Additionally, the submatrix T_θ^w has a rather distinct structure itself. It is a sparse, banded diagonal matrix with all zeros in the upper triangle. Thus, rather than using matrix multiplication when computing $T_\theta^w \boldsymbol{u}_{w+1}^k$, we make use of this property and solve the equations using two nested for-loops that run through the full set Y (which is small), but only a minor subset of X. The matrix multiplication of $T_\theta^w \boldsymbol{u}_{w+1}^k$ can therefore be performed in XY^2 time. Addition with $\boldsymbol{b}_w$ is an XY time operation, which is dominated (asymptotically) by the former operation. Since each recursion is performed T times, the asymptotic running time becomes $O(XY^2T)$ for each iteration. This is a significant improvement upon using Gauss-Seidel directly, which is $O(X^2Y^2T^2)$ per iteration.

We prove the convergence of this algorithm in appendix C.7.

4.1.2 Partial derivatives

The partial derivatives of EV_θ with respect to the elements of $\boldsymbol{\theta}$ are given by

$$\nabla EV_\theta = \left[I - T_\theta'\right]^{-1} \nabla T_\theta(EV_\theta). \tag{4.4}$$

This set of equations is solved by the use of the sparse matrix solution algorithm presented in the previous section, and the reason to why this is done within the inner loop should now be obvious. Solving equation (4.4)

requires the Fréchet derivative, and it is therefore natural to do this in the last Newton-Kantorovich iteration of the inner loop.

The gradient ∇ is in our case

$$\nabla = \begin{bmatrix} \partial/\partial\beta \\ \partial/\partial\theta_1 \end{bmatrix},$$

which leaves us with the task of calculating $\nabla T_\theta(EV_\theta)$. These are given by[4]

$$\begin{aligned} \frac{\partial T_\theta(EV_\theta)\,(\boldsymbol{\zeta})}{\partial\beta} &= \mathcal{P} \times \frac{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}, d, \boldsymbol{\theta}) EV_\theta(\boldsymbol{\zeta}(d))}{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}, d, \boldsymbol{\theta})} \\ \frac{\partial T_\theta(EV_\theta)\,(\boldsymbol{\zeta})}{\partial\theta_1} &= -\mathcal{P} \times \frac{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}, d, \boldsymbol{\theta}) q(x, d, w) \sum_{b=0}^{d} h_{bw}}{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}, d, \boldsymbol{\theta})} \end{aligned} \tag{4.5}$$

4.2 The outer loop

Recall the full and partial likelihood functions (2.21) and (2.22), which we restate here for convenience:

$$\begin{aligned} \mathcal{L}^f(\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_T, d_1, \ldots, d_T) &= \prod_{t=2}^{T} P(d_t|\boldsymbol{\zeta}_t, \boldsymbol{\theta}) p(\boldsymbol{\zeta}_t, |\boldsymbol{\zeta}_{t-1}), \\ \mathcal{L}^p(\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_T, d_1, \ldots, d_T) &= \prod_{t=1}^{T} P(d_t|\boldsymbol{\zeta}_t, \boldsymbol{\theta}). \end{aligned} \tag{4.6}$$

Dropping function arguments for simplicity, consider now the following *log*-likelihood functions, given by

$$\mathcal{L}_l^f = \log \mathcal{L}^f = \sum_{t=2}^{T} \log P(d_t|\boldsymbol{\zeta}_t, \boldsymbol{\theta}) + \log p(\boldsymbol{\zeta}_t, |\boldsymbol{\zeta}_{t-1}), \tag{4.7}$$

$$\mathcal{L}_l^p = \log \mathcal{L}^p = \sum_{t=1}^{T} \log P(d_t|\boldsymbol{\zeta}_t, \boldsymbol{\theta}). \tag{4.8}$$

[4]The derivations are quite easy and may be done in pretty much the same fashion as the Fréchet derivative.

Maximization of the log-likelihood function (4.7) and maximization of the likelihood function (4.6) itself yields the same optimal solution $\hat{\boldsymbol{\theta}}$, that is,

$$\underset{\boldsymbol{\theta}\in R^2}{\operatorname{argmax}}\ \mathcal{L}^f = \underset{\boldsymbol{\theta}\in R^2}{\operatorname{argmax}}\ \mathcal{L}_l^f.$$

The Markov transition probabilities $p(\boldsymbol{\zeta}_t, |\boldsymbol{\zeta}_{t-1})$ are independent of $\boldsymbol{\theta}$, and are therefore superfluous in the estimation procedure:

$$\underset{\boldsymbol{\theta}\in R^2}{\operatorname{argmax}}\ \mathcal{L}^f = \underset{\boldsymbol{\theta}\in R^2}{\operatorname{argmax}}\ \mathcal{L}_l^f = \underset{\boldsymbol{\theta}\in R^2}{\operatorname{argmax}}\ \mathcal{L}_l^p.$$

Thus, we need only maximize the *partial* log-likelihood function (4.8) to arrive at the optimal $\hat{\boldsymbol{\theta}}$. This objective function is

$$\mathcal{L}_l^p = \sum_{t=1}^{T} \log \left[\frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right]$$

Maxmization using the BHHH algorithm requires the gradient of the objective function (see section 2.4.4). These are[5]

$$\frac{\partial \mathcal{L}_l^p}{\partial \beta} = \sum_{t=1}^{T} \left[\phi_\beta(d_t) - \frac{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta}) \phi_\beta(d')}{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right]$$

$$\frac{\partial \mathcal{L}_l^p}{\partial \theta_1} = \sum_{t=1}^{T} \left[\phi_{\theta_1}(d_t) - \frac{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta}) \phi_{\theta_1}(d')}{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right]$$

where

$$\phi_\beta(d) = \beta \frac{\partial EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)}{\partial \beta} + EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)$$

$$\phi_{\theta_1}(d) = \beta \frac{\partial EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)}{\partial \theta_1} - q(x_t, d, w_t) \sum_{b=0}^{d} h_{bw_t}$$

[5]Derived in appendix C.6.

This gives us the gradient

$$\nabla \mathcal{L}_l^p = \begin{bmatrix} \dfrac{\partial \mathcal{L}_l^p}{\partial \beta} \\ \dfrac{\partial \mathcal{L}_l^p}{\partial \theta_1} \end{bmatrix}$$

and the deflection matrix

$$\boldsymbol{D} = -\left[\nabla \mathcal{L}_l^p \otimes \nabla \mathcal{L}_l^p\right]^{-1} = -\begin{bmatrix} \dfrac{\partial \mathcal{L}_l^p}{\partial \beta} \times \dfrac{\partial \mathcal{L}_l^p}{\partial \beta} & \dfrac{\partial \mathcal{L}_l^p}{\partial \beta} \times \dfrac{\partial \mathcal{L}_l^p}{\partial \theta_1} \\ \dfrac{\partial \mathcal{L}_l^p}{\partial \theta_1} \times \dfrac{\partial \mathcal{L}_l^p}{\partial \beta} & \dfrac{\partial \mathcal{L}_l^p}{\partial \theta_1} \times \dfrac{\partial \mathcal{L}_l^p}{\partial \theta_1} \end{bmatrix}^{-1}.$$

The BHHH iterations become

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \lambda_{k+1}\Delta\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \lambda_{k+1}\boldsymbol{D}_k \nabla \mathcal{L}_{l\,k}^p$$

.

4.2.1 Line search

Recall from section 2.4.4 that we suggested the secant method as a suitable line search. In the secant method, λ_{k+1} is given by

$$\lambda_{k+1} = \lambda_k - \frac{(\lambda_k - \lambda_{k-1}) df(\lambda_k)/d\lambda}{df(\lambda_k)/d\lambda - df(\lambda_{k-1})/d\lambda}.$$

Defining the univariate function

$$f(\lambda) = \mathcal{L}_l^p\left(\boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta}\right),$$

it is clear that we require $\frac{\partial f(\lambda)}{\partial \lambda}$ in order to find this optimal λ. Now,

$$f(\lambda) = \mathcal{L}_l^p\left(\boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta}\right) = \sum_{t=1}^{T} \log P\left(d_t | \boldsymbol{\zeta}_t, \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta}\right)$$

$$= \sum_{t=1}^{T} \log \left[\frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta})}{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta})} \right]$$

where $\boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta} = \begin{bmatrix} \beta \\ \theta_1 \end{bmatrix} + \lambda \begin{bmatrix} \Delta\beta \\ \Delta\theta_1 \end{bmatrix}$ and $\psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta})$ is given by

$$\begin{aligned}
&\psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta}) \\
&\quad = \exp\left\{ q(x_t, d, w_t) \sum_{b=0}^{d} \left(c_{bw_t} y_t - (\theta_1 + \lambda\Delta\theta_1)\right) h_{bw_t} + (\beta + \lambda\Delta\beta) EV_\theta\left(\boldsymbol{\zeta}_t(d)\right) \right\} \\
&\quad = \exp\left\{ q(x_t, d, w_t) \sum_{b=0}^{d} \left(c_{bw_t} y_t - \theta_1 - \lambda\Delta\theta_1\right) h_{bw_t} + (\beta + \lambda\Delta\beta) EV_\theta\left(\boldsymbol{\zeta}_t(d)\right) \right\}
\end{aligned}$$

The derivative of $f(\lambda)$ becomes[6]

$$\frac{df(\lambda)}{d\lambda} = \sum_{t=1}^{T} \left[\phi_\lambda(d_t) - \frac{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta}) \phi_\lambda(d')}{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta} + \lambda\Delta\boldsymbol{\theta})} \right]$$

where

$$\phi_\lambda(d) = \Delta\beta EV_\theta\left(\boldsymbol{\zeta}_t(d)\right) - \Delta\theta_1 q(x_t, d, w_t) \sum_{b=0}^{d} h_{bw_t}.$$

4.3 Simulation

We use simulation to analyze the model's correctness. By applying NFXP to a data set for which we know the true parameter values that generated it, we are able to directly evaluate the quality of the results. This helps to improve confidence in the model. We use the inner loop to generate an optimal decision policy, thus creating a fictitious data set for a "known" $\boldsymbol{\theta}$, which we in turn approximate using NFXP.

To generate the data set we use a β of 0.7 and a θ_1 equal to 5. A low beta is chosen to increase convergence of the inner loop[7]. With initial guesses of $\beta = 0.8$ and $\theta_1 = 0$, the algorithm solved the problem in approximately

[6] The derivation of $\frac{fd(\lambda)}{d\lambda}$ is quite similar to those of the partial derivatives of the Log-likelihood function.

[7] See section 5.3 for more on this.

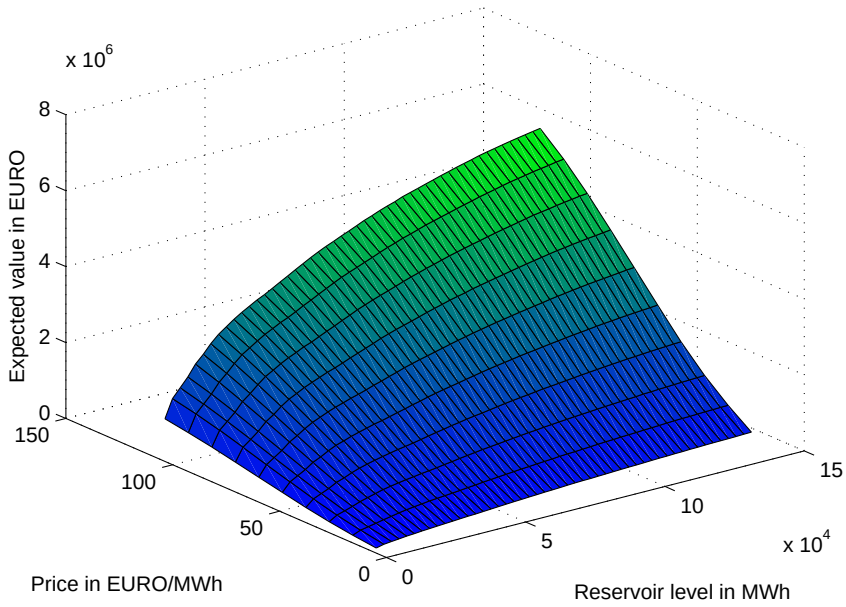

Figure 4.2: Expected value function versus reservoir level and price for week 1.

43 hours. The estimated parameter values after this run was $\hat{\beta} = 0.7021$ and $\hat{\theta}_1 = 5.0037$. The information matrix provided standard deviations of 0.0013 and 0.000126, respectively.

As a step in the NFXP algorithm we find the solution to the expected value function in the infinite horizon Bellman equation

$$EV_\theta(\boldsymbol{\zeta}) = \int V_\theta(\boldsymbol{\zeta}')p(d\boldsymbol{\zeta}'|\boldsymbol{\zeta}, d, \boldsymbol{\theta}).$$

Figure 4.2 and 4.3 show these values for all reservoir levels and prices in week 1 and 20, respectively. We have chosen these two weeks because they clearly demonstrate the effect of the anticipated spring flood. Examining figure 4.2, we see that if price is high, there is a greater probability of high prices in the immediate future, thus resulting in a higher expected value. The reader may question why current price affects expected value to such a high degree. The reason is directly related to the low β, which makes future

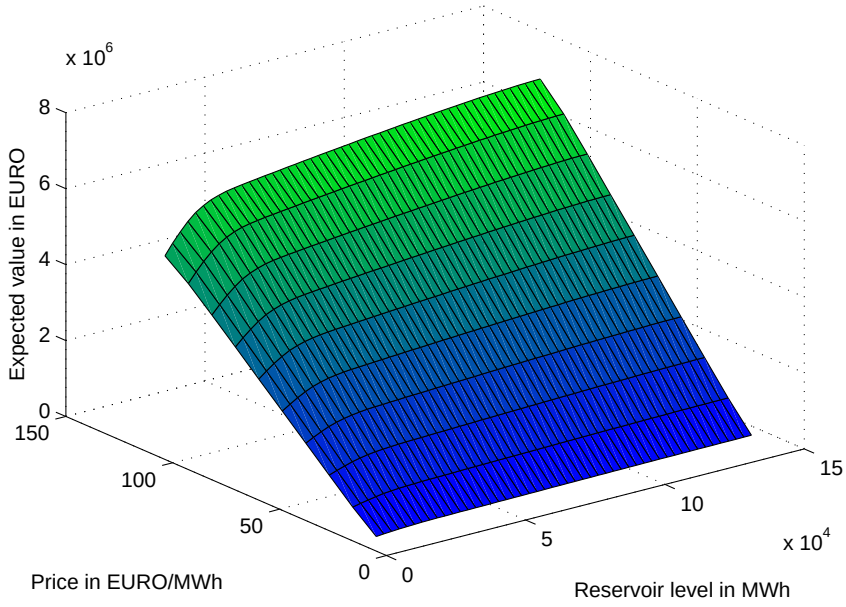

Figure 4.3: Expected value function versus reservoir level and price for week 20.

utility much less important, thus making a high price today relatively more valuable than it would have been with a realistic β. Using a beta of 0.7 on a weekly basis implies very heavy discounting. With a larger beta (close to 1), these plots become much less dependent on price.

Notice further that a full reservoir in week 1 is very valuable. In figure 4.3, on the other hand, we see that the expected value is primarily dependent on price. This is due to the fact that the spring flood, which is expected to occur within just a matter of weeks, or even days, is likely to fill the entire reservoir, making the current level of less importance. Figure 4.4 shows the difference between the fixed point calculated based on the true parameters and the ones estimated by NFXP.

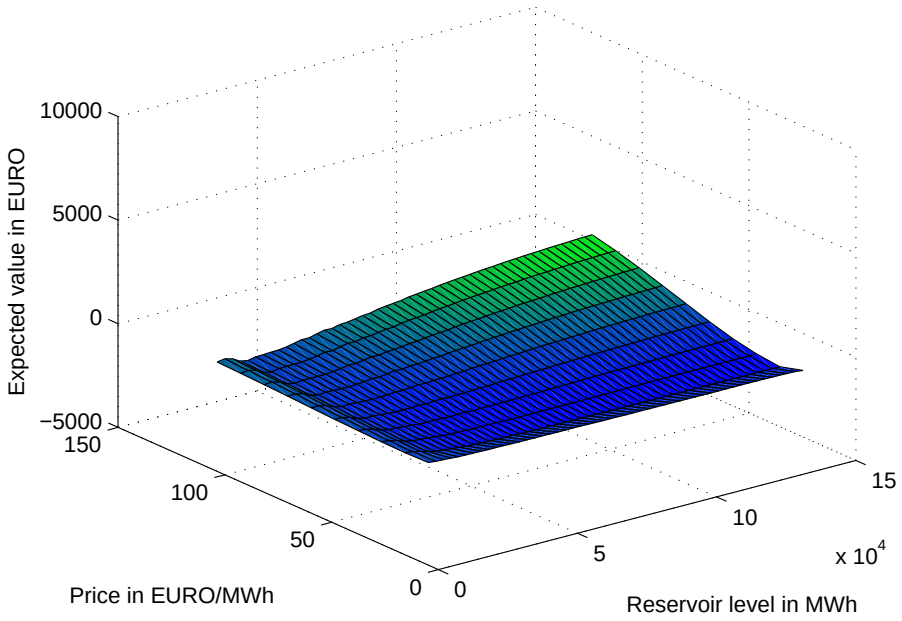

Figure 4.4: Difference between expected value for true $\boldsymbol{\theta}$ and estimate $\hat{\boldsymbol{\theta}}$ for week 1.

Chapter 5

Conclusions

In the previous chapters we have extensively discussed an application of one particular structural estimation method. In this chapter, we zoom out a bit, and look at our methodology in a larger perspective.

There is considerable disagreement in the literature about the most efficient algorithms to solve high-dimensional DP problems. The debate is roughly divided between whether it is better to solve DP problems by *discrete approximation* or by *parametric approximation*. The former approach is the one which has been applied in this thesis. Evaluating the effectiveness (or lack thereof) of parametric methods is beyond the scope of this work. What we can do, however, is suggest paths for future work. Now, we suspect that, regardless of choice of method, several *ad hoc* methods would have to be devised to speed up computation. Despite this, some approaches seem to be more promising with respect to hydropower production than others. We will now proceed to discuss these, and suggest how the experiences we have gained in our work might be leveraged to yield more useful results in the future.

5.1 The JS approach

Judd and Su (2006) argue that there are several difficulties associated with NFXP as an estimation method:

1. it is often computationally difficult to solve NFXP on large state spaces
2. one repeatedly solves a DP problem for parameter values far from the optimal solution
3. NFXP is costly to implement (formulation and programming)
4. the degree of algorithmic and mathematical complexity can be daunting (e.g. BHHH, functional analysis, Newton-Kantorovich etc.).

Using the assumptions of Rust, Judd and Su argue that NFXP in fact may be viewed as a constrained optimization problem:

$$\max_{\boldsymbol{\theta}, EV_\theta} \quad L(\boldsymbol{\theta}) = \prod_{t=2}^{T} P(d_t|\boldsymbol{s}_t, \boldsymbol{\theta}) p(\boldsymbol{s}_t|\boldsymbol{s}_{t-1}, d_{t-1}, \boldsymbol{\theta})$$

$$\text{s.t.} \qquad EV_\theta(\boldsymbol{s}, d) = T_\theta(EV_\theta(\boldsymbol{s}, d))$$

which may be submitted to any *state-of-the-art* constrained optimization program. $EV_\theta(\boldsymbol{s}, d)$ is still a vector describing expected present value for all states $\boldsymbol{s}$.

Our general impression is that this approach shows promise. In particular, the absence of an "inner loop" seems to be an attractive feature, since this is a particularly burdensome part of our MATLAB implementation. We have done some initial programming using the MOSEL language for Xpress, and believe that with some work, this will outperform our implementation. The formulation as a regular constrained optimization problem has appealing intuition, but is not without difficulty. In particular, there are several algorithmic details that we are uncertain whether can be directly handled by the optimizer. Now, Xpress (and other optimization packages) have the ability to interface with other languages, such as MATLAB. This allows for conventional programming techniques and *ad hoc* methods to be merged with the optimization routines of the optimizer. We suspect that large portions of our code (downloadable from `http://folk.ntnu.no/host/structural/`) may be used directly with an optimizer capable of interfacing with it, though it will likely be necessary to make several adaptations to it.

5.2 Parametric approximation methods

The intuitive appeal of parametric approximation methods is that a potentially infinite-dimensional problem (e.g finding the solution V to the Bellman equation) is reduced to a finite-dimensional problem with a relatively small number K of unknown parameters. To illustrate, suppose we are interested in approximating the value function $V(s)$, where

$$V(s) = T(V(s)) \equiv \max_{d \in D} \left[u(s,d) + \beta \int V(s') p(ds'|s,d) \right].$$

Suppose we assume that V can be approximated by a linear combination of "basis functions" $\{\rho_1(s), \ldots, \rho_k(s)\}$,

$$V_\theta(s) = \sum_{k=1}^{K} \theta_k \rho_k(s).$$

If the basis that is chosen is "good", i.e. a linear combination of these basis functions indeed provides a good approximation, then optimization methods can be used to minimize a *residual function*, $\Phi(V_\theta) = V_\theta - T(V_\theta)$. Benítez-Silva et al. (2000) argue that minimizing such a function potentially could be much faster than a discrete approximation of V. However, they also note that, as far as they know, there exists no formal proofs that parametric approximation methods succeed in breaking the curse of dimensionality. In fact, they argue that "*. . . in the absence of some sort of "special structure", the number of basis functions required to perform a uniform approximation to a smooth function of d variables increases exponentially in d.*" Further, they examine the literature on applications using parametric methods, and find that applications have generally yielded mixed results. In some cases, the nonlinear optimization problem can be solved quickly and reliably, but others have been plagued my problems of multiple optima and have experienced considerable difficulty in getting the minimization of $\Phi(V_\theta)$ to converge, especially when the underlying function being approximated has kinks and discontinuities. The observations of Benítez-Silva et al. (2000) lead us to consider parametric approximation methods to be the least attractive candidate (of those we have considered) for future work.

5.3 Computational performance

NFXP is quite demanding to implement, both with respect to algorithmic complexity and sheer amount of code. We estimate that we, in total, spent about 800 hours in the implementation phase. Our experience with NFXP is that translating the mathematics into code is far more difficult than one might expect. Sections of the algorithm that we at an initial glance had thought to be trivial to implement, very often proved to be quite challenging. Moreover, the obvious implementations were often much too slow, requiring specialized numerical tweaks to perform at a satisfying speed. This is not meant as critique of the algorithm, but rather a mere warning to those who want to pursue NFXP as a solution-method to their problem. We would also argue that implementation complexity is highly related to the complexity of the problem.

However, it is not the difficulties associated with implementation that ultimately restricts NFXP's applicability to the problem of hydropower production scheduling. In the end, the *running time* of the algorithm is the most restrictive issue. For most problems that we have seen in the literature, NFXP performs very well. In our case, however, the state space is simply too large. The outer loop runs quite fast, especially as BFGS takes over as the provider of the search direction, but the need to calculate a new fixed point for every iteration step of the outer loop results in slow convergence. Even with the speed-ups discussed in Chapter 4, the time it takes to perform one contraction iteration, with a realistic discretization of the state space, is about three seconds. If the change in $\boldsymbol{\theta}$ induced by the line search is large, the number of these iterations can be quite significant.

Table 5.1: Time to evaluate fixed point with a 2.4 GHz dual core processor

β	0.7	0.8	0.9	0.99
Seconds	212	301	514	5433

Discretization of state space: $|X|= 50$ and $|Y|= 10$
Discretization of decision space: $|D|= 6$

Another issue is that of the contraction parameter β. The rate at which the contraction iterations close in on the fixed point is linearly proportional to $log(\beta^{-1})$. Thus, when β is close to 1 (say, $\beta > 0.99$), the convergence rate

of the contraction iterations is very poor. Further, for β to be realistic in a weekly perspective, it would have to be even larger than 0.99.[1] Table 5.1 show running times for different values of β with a realistic resolution of the state- and decision spaces.

5.3.1 Parallel computing

Clearly, the most arduous part of the algorithm is to achieve the fixed point for every step of the outer loop. With this in mind, we implemented the inner loop in a way that allows for the use of parallel computing. Both the contraction iterations and the computation of the Fréchet derivative can be separated into 52 subproblems for each level block (see equation (4.2)), which in turn can be computed independently. Thus, the inner loop may be sped up significantly by using parallel computing. The improvement for the contraction iterations would be by approximately a factor of 52, thereby letting the inner loop converge quicker to the point where Newton-Kantorovich takes over. Since the calculation of the Fréchet derivative is performed once for each iteration of NK, this would also be computed a lot faster.

5.4 Uncovering preference

Despite the rapid growth in computing power and developments in the literature on numerical dynamic programming, multi-dimensional infinite-horizon continuous-state dynamic programming problems are still quite challenging to solve. Since very few of these problems have analytic solutions, discretization of the state space is required, and so the *curse of dimensionality* becomes an issue. Recent developments suggest that, under certain conditions, the curse of dimensionality may be broken (see Rust, 1997; Rust et al., 2002), but for most problems, *ad hoc* solutions must be applied. Indeed, this is precisely what has been necessary in our case. Luckily, it seems that there is an intrinsic structure in the inverse stochastic

[1] If we, for the sake of the argument, use a discount rate of, say, 20%, then

$$\beta = \frac{1}{1 + 0.2/52} \approx 0.996$$

.

control problem of hydropower production that may be exploited in several ways.

Undeniably, the problem of optimal hydropower scheduling and planning is a difficult one. That the inverse problem, that is, quantitatively assessing the decisions of hydropower producers, is paralleled in its complexity comes as no surprise. Still, we hope that the ideas and methodology we have developed within, can be built upon to develop faster, more efficient and more precise estimation techniques for uncovering hydropower producer preference.

Bibliography

Benítez-Silva, H., Hall, G., Hitsch, G., Pauletto, G., Brook, S., and Rust, J. (2000). A comparison of discrete and parametric approximation methods for continuous-state dynamic programming problems. *Working paper*.

Berndt, E., Hall, B., Hall, R., and Hausman, J. (1974). Estimation and inference in nonlinear structural models. *Annals of Economic and Social Measurement*, 3(4):103–106.

Fleten, S. and Kristoffersen, T. (2008). Short-term hydropower production planning by stochastic programming. *Computers & Operations Research*, 35(8):2656–2671.

Gamba, A. and Tesser, M. (2009). Structural estimation of real options models. *Journal of Economic Dynamics and Control*, 33(4):798–816.

Gihman, I. and Skorohod, A. (1979). *Controlled Stocastic Processes*. Springer.

Griffel, D. (2002). *Applied functional analysis*. Dover publications.

Hansen, L. and Singleton, K. (1982). Generalized instrumental variables estimation of nonlinear rational expectations models. *Econometrica*, 50:1269–1286.

Judd, K. and Su, C. (2006). A new optimization approach to maximum likelihood estimation of structural models. *Computing in Economics and Finance 2006*.

Kellogg, R. (2010). The Effect of Uncertainty in Investment: Evidence from Texas Oil Drilling. *National Bureau of Economic Research*, (16541).

Kolsrud, C. and Prokosch, M. (2009). Dynamics of hydropower scheduling: Producers' response to forward prices and other uncertainties. *Project Thesis, NTNU*.

Kolsrud, C. and Prokosch, M. (2010). Reservoir Hydropwer: The value of flexibility. *M.Sc Thesis, NTNU*.

McFadden, D. (1981). Econometric models of probabilistic choice, in ‘‘Structural Analysis of Discrete Data with Econometric Applications,’’(CF Manski and D. McFadden, Eds.).

Muehlenbachs, L. (2009). Idle Oil Wells: Half Empty or Half Full? *na*.

Nandalal, K. and Bogárdi, J. (2007). *Dynamic programming based operation of reservoirs: applicability and limits*. Cambridge Univ Pr.

Philpott, A., Craddock, M., and Waterer, H. (2000). Hydro-electric unit commitment subject to uncertain demand. *European Journal of Operational Research*, 125(2):410–424.

Rust, J. (1987). Optimal replacement of GMC bus engines: An empirical model of Harold Zurcher. *Econometrica: Journal of the Econometric Society*, 55(5):999–1033.

Rust, J. (1988). Maximum likelihood estimation of discrete control processes. *SIAM Journal on Control and Optimization*, 26:1006.

Rust, J. (1994). Structural estimation of Markov decision processes. *Handbook of Econometrics*, 4:3081–3143.

Rust, J. (1997). Using randomization to break the curse of dimensionality. *Econometrica: Journal of the Econometric Society*, pages 487–516.

Rust, J., Traub, J., and Wozniakowski, H. (2002). Is there a curse of dimensionality for contraction fixed points in the worst case? *Econometrica*, 70(1):285–329.

Thompson, M., Davison, M., and Rasmussen, H. (2004). Valuation and optimal operation of electric power plants in competitive markets. *Operations Research*, 52(4):546–562.

Appendix A Hydropower production

Hydropower plants usually have one or more reservoirs where water is stored. These reservoirs are subject to continuous, but highly variable inflows of water from their surroundings. In general, but especially in the colder regions of the globe, the magnitude and timing of these inflows are highly dependent on weather and seasons. The availability of water for power production may thus vary greatly throughout the year. When hydropower is a part of the base load supply of electricity, good production planning becomes important not only in the financial sense, but also in the economic sense (Nandalal and Bogárdi, 2007).

The production flexibility of the producer depends on the size of the reservoir, the amount of inflow to the reservoir, and the production capacity (Kolsrud and Prokosch, 2010). If the size of the reservoir is small relative to annual inflow, the producer will be less flexible, as it has to produce more often to avoid spillover. Furthermore, flexibility of production is often limited by maximum water flow rates and permissible upper and lower reservoir levels set by governing authorities. Reservoir inflow is dependent on the amount and timing of precipitation, the amount of snow stored in the reservoirs water system, and the timing of the spring flood. Typically, reservoir levels are quite low just before the spring flood, as inflow is scarce during winter and producers seem to produce in such a manner that their reservoirs can accommodate as much of the spring flood inflow as possible.

The amount of electricity generated is determined by the energy coefficient of the particular plant, which depends non-linearly on the head (the difference in height between reservoir level and turbine), and the flow-rate through the turbine. The higher the energy coefficient, the more power per volume of water is generated. Consider two reservoirs with same volume of water but significantly different geometry, say one that is deep but narrow and one that is shallow and wide. The former reservoir may then yield large changes in efficiency as the reservoir level changes, whereas the latter reservoir may yield approximately the same efficiency for all levels.

The decision process of power producers may be divided into two stages.

Every day, power producers submit price-dependent bids to the spot market on Nord Pool for hourly production the following day. These decisions are thus made under limited information on the day-ahead market prices. After the bidding process is finished, demand and supply bids are aggregated to determine an equilibrium price for each hour the following day, called the system price. The production decision itself, that is, the decision of whether to produce power or not in a certain period, is deferred until information has been fully disclosed (Fleten and Kristoffersen, 2008). From a normative perspective, the obvious issue for the rational profit-maximizing hydropower producer is then to determine what constitutes an optimal production strategy with respect to prevailing prices, current reservoir levels, and price and inflow expectations.

Appendix B Assumptions

Assumption **BU** *For each $d \in D(\boldsymbol{\zeta})$, $u(\boldsymbol{\zeta}, d)$ is an upper semicontinuous function of x with bounded expectation*

$$\begin{aligned} R(\boldsymbol{\zeta}) &\equiv \sum_{t=1}^{\infty} \beta^t R_t(\boldsymbol{\zeta}) < \infty, \\ R_{t+1}(\boldsymbol{\zeta}) &= \max_{d \in D(\boldsymbol{\zeta})} \int R_t(\boldsymbol{\zeta}') \pi(d\boldsymbol{\zeta}' | \boldsymbol{\zeta}, d), \\ R_1(\boldsymbol{\zeta}) &= \max_{d \in D(\boldsymbol{\zeta})} \iint \max_{d' \in D(\boldsymbol{\zeta}')} |u(\boldsymbol{\zeta}', d') + \epsilon_{d'}| q(d\epsilon | \boldsymbol{\zeta}') \pi(d\boldsymbol{\zeta}' | \boldsymbol{\zeta}, d). \end{aligned}$$

Assumption **WC** *$\pi(d\boldsymbol{\zeta}' | \boldsymbol{\zeta}, d)$ is a weakly continuous function of $(\boldsymbol{\zeta}, d)$: for each bounded continuous function $h : \boldsymbol{Z} \to \mathbb{R}$, $\int h(\boldsymbol{\zeta}') \pi(d\boldsymbol{\zeta}' | \boldsymbol{\zeta}, d)$ is a continuous function of $\boldsymbol{\zeta}$ for each $d \in D(\boldsymbol{\zeta})$*

Assumption **BE** *Let B be the Banach space of bounded, Borel measurable functions $h : Z \to \mathbb{R}$ under the essential supremum norm. Then $u \in B$ and for each $h \in B, Eh \in B$, where Eh is defined by*

$$Eh(\boldsymbol{\zeta}, d) \equiv \int G\left[\left\{h(\boldsymbol{\zeta}', d), d \in D(\boldsymbol{\zeta}')\right\}\right] \pi(d\boldsymbol{\zeta}' | \boldsymbol{\zeta}, d)$$

Appendix C Derivations

C.1 Social surplus function

Recall that the social surplus function and conditional choice probabilities are defined by

$$G\left[\left\{u\left(\boldsymbol{s},d\right),d\in D(\boldsymbol{s})\right\}|\boldsymbol{s}\right]=\int_{R^{|D|}}\max_{d\in D(\boldsymbol{s})}\left\{u(\boldsymbol{s},d)+\epsilon(d)\right\}q(d\epsilon|\boldsymbol{s}).$$

and

$$P(d|\boldsymbol{s})=\int I\left\{d=\delta\left(\boldsymbol{s},\epsilon\right)\right\}q(d\epsilon|\boldsymbol{s})$$

respectively. Then,

$$\begin{aligned}\frac{\partial G\left[\left\{u\left(\boldsymbol{s},d\right),d\in D(\boldsymbol{s})\right\}|\boldsymbol{s}\right]}{\partial u\left(\boldsymbol{s},d\right)}&=\int\left(\frac{\partial\left\{\max_{d\in D(\boldsymbol{s})}\left[u(\boldsymbol{s},d)+\epsilon(d)\right]\right\}}{\partial u(\boldsymbol{s},d)}\right)q(d\epsilon|\boldsymbol{s})\\&=\int I\left\{d=\arg\max_{d'\in D(\boldsymbol{s})}\left[u(\boldsymbol{s},d')+\epsilon(d')\right]\right\}q(d\epsilon|\boldsymbol{s})\\&=P(d|\boldsymbol{s})\end{aligned}$$

■

C.2 Conditional choice probability

Under the assumptions AS and CI, Bellman's equation has the form

$$V(\boldsymbol{s},\epsilon)=\max_{d\in D(\boldsymbol{s})}v(\boldsymbol{s},d)+\epsilon(d)\tag{C.1}$$

where

$$v(\boldsymbol{s},d)=u(\boldsymbol{s},d)+\beta\int V(\boldsymbol{s}',\epsilon)q(d\epsilon|\boldsymbol{s}')\pi(d\boldsymbol{s}'|\boldsymbol{s},d)$$

and we let $\boldsymbol{s}$ denote all state variables and d denote decisions. Assume $\epsilon(d)$ follows a multivariate extreme-value distribution (Gumbel),

$$q(\epsilon|\boldsymbol{s}) = \prod_{d \in D(\boldsymbol{s})} f_{\epsilon(d)} = \prod_{d \in D(\boldsymbol{s})} \frac{1}{\sigma} e^{-\frac{\epsilon(d)-\gamma}{\sigma}} e^{-e^{-\frac{\epsilon(d)-\gamma}{\sigma}}}$$

where $f_{\epsilon(d)}$ is the probability density in the univariate case, and σ and γ are known as the scale and shape parameters, respectively. Setting γ equal to Euler's constant (0.577) shifts the extreme value distribution so it has unconditional mean zero.

The cumulative distribution in the univariate case is given by

$$\begin{aligned} F_\epsilon(\alpha) &= P\{\epsilon < \alpha\} \\ &= \frac{1}{\sigma} \int_{-\infty}^{\alpha} e^{-\frac{\epsilon(d)-\gamma}{\sigma}} e^{-e^{-\frac{\epsilon(d)-\gamma}{\sigma}}} d\epsilon \end{aligned}$$

Substituting $z = e^{-\frac{\epsilon(d)-\gamma}{\sigma}}$ one finds $d\epsilon = (-\sigma/z)dz$ and so

$$\begin{aligned} F_\epsilon(\alpha) &= \frac{1}{\sigma} \int_{-\infty}^{*} z e^{-z} (-\frac{\sigma}{z}) dz = \int_{-\infty}^{*} -e^{-z} dz \\ &= \left[e^{-z}\right]_{-\infty}^{*} = \left[e^{-e^{-\frac{\epsilon(d)-\gamma}{\sigma}}}\right]_{-\infty}^{\alpha} \\ &= e^{-e^{-\frac{\alpha-\gamma}{\sigma}}}. \end{aligned}$$

Now, let $P(d|\boldsymbol{s})$ denote the probability of choosing decision d when in state $\boldsymbol{s}$. A utility-maximizing agent will choose d so as to satisfy equation (C.1). Thus, $P(d|\boldsymbol{s})$ simply denotes the probability that decision d maximizes utility. This may be stated by

$$\begin{aligned} P(d_i|\boldsymbol{s}) &= P\{v(\boldsymbol{s}, d_i) + \epsilon(d_i) > v(\boldsymbol{s}, d_j) + \epsilon(d_j)\} \quad \forall j \neq i \\ &= P\{\epsilon(d_j) < v(\boldsymbol{s}, d_i) + \epsilon(d_i) - v(\boldsymbol{s}, d_j)\} \quad \forall j \neq i \\ &= P\{\epsilon(d_j) < \phi_j\} \quad \forall j \neq i \end{aligned}$$

with $\phi_j = \epsilon(d_i) + v(\boldsymbol{s}, d_i) - v(\boldsymbol{s}, d_j)$. Now assume that $\epsilon(d_i)$ is known. Then, the conditional probability

$$P(d_i|\boldsymbol{s}, \epsilon(d_i)) = P\left\{\epsilon(d_1) < \phi_1, \ldots, \epsilon(d_{i-1}) < \phi_{i-1}, \epsilon(d_{i+1}) < \phi_{i+1}, \ldots \epsilon(d_n) < \phi_n\right\}$$

$$= \prod_{j \neq i} P\left\{\epsilon(d_j) < \phi_j\right\}, \quad \textit{by independence}$$
$$= \prod_{j \neq i} e^{-e^{-\frac{\phi_j - \gamma}{\sigma}}}.$$

We integrate out $\epsilon(d_i)$,

$$P(d_i|\boldsymbol{s}) = \int_{-\infty}^{\infty} P(d_i|\boldsymbol{s}, \epsilon(d_i)) f_{\epsilon(d_i)} d\epsilon(d_i)$$

$$= \int_{-\infty}^{\infty} \left(\prod_{j \neq i} e^{-e^{-\frac{\phi_j - \gamma}{\sigma}}} \right) \frac{1}{\sigma} e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}} e^{-e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}}} d\epsilon(d_i)$$

$$= \int_{-\infty}^{\infty} \left(\prod_{j} e^{-e^{-\frac{\phi_j - \gamma}{\sigma}}} \right) \frac{1}{\sigma} e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}} e^{-e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}}} e^{e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}}} d\epsilon(d_i)$$

Since $\phi_j = \epsilon(d_i) + v(\boldsymbol{s}, d_i) - v(\boldsymbol{s}, d_j)$, we have $\phi_i = \epsilon(d_i) + v(\boldsymbol{s}, d_i) - v(\boldsymbol{s}, d_i) = \epsilon(d_i)$, and so

$$P(d_i|\boldsymbol{s}) = \frac{1}{\sigma} \int_{-\infty}^{\infty} \left(\prod_{j} e^{-e^{-\frac{\phi_j - \gamma}{\sigma}}} \right) e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}} e^{-e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}}} e^{e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}}} d\epsilon(d_i)$$

$$= \frac{1}{\sigma} \int_{-\infty}^{\infty} \left(\prod_{j} e^{-e^{-\frac{\phi_j - \gamma}{\sigma}}} \right) e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}} d\epsilon(d_i)$$

$$= \frac{1}{\sigma} \int_{-\infty}^{\infty} \left(e^{-\sum_j e^{-\frac{\phi_j - \gamma}{\sigma}}} \right) e^{-\frac{\epsilon(d_i) - \gamma}{\sigma}} d\epsilon(d_i)$$

$$= \frac{1}{\sigma} \int_{-\infty}^{\infty} \left(e^{-\sum_j e^{-\frac{\epsilon(d_i)+v(\boldsymbol{s},d_i)-v(\boldsymbol{s},d_j)-\gamma}{\sigma}}} \right) e^{-\frac{\epsilon(d_i)-\gamma}{\sigma}} d\epsilon(d_i)$$

$$= \frac{1}{\sigma} \int_{-\infty}^{\infty} \left(e^{-e^{-\frac{\epsilon(d_i)-\gamma}{\sigma}} \sum_j e^{-\frac{v(\boldsymbol{s},d_i)-v(\boldsymbol{s},d_j)}{\sigma}}} \right) e^{-\frac{\epsilon(d_i)-\gamma}{\sigma}} d\epsilon(d_i)$$

We set $K = \sum_j e^{-\frac{v(\boldsymbol{s},d_i)-v(\boldsymbol{s},d_j)}{\sigma}}$ so

$$P(d_i|\boldsymbol{s}) = \frac{1}{\sigma} \int_{-\infty}^{\infty} \left(e^{-Ke^{-\frac{\epsilon(d_i)-\gamma}{\sigma}}} \right) e^{-\frac{\epsilon(d_i)-\gamma}{\sigma}} d\epsilon(d_i)$$

We may now perform the same substitution as above, namely $z = e^{-\frac{\epsilon(d_i)-\gamma}{\sigma}}$. Then

$$P(d_i|\boldsymbol{s}) = \int_0^{\infty} e^{-zK} dz = \frac{1}{K}$$

$$= \frac{1}{\sum_j e^{-\frac{v(\boldsymbol{s},d_i)-v(\boldsymbol{s},d_j)}{\sigma}}}$$

$$= \frac{1}{e^{-\frac{v(\boldsymbol{s},d_i)}{\sigma}} \sum_j e^{\frac{v(\boldsymbol{s},d_j)}{\sigma}}}$$

$$P(d_i|\boldsymbol{s}) = \frac{e^{\frac{v(\boldsymbol{s},d_i)}{\sigma}}}{\sum_j e^{\frac{v(\boldsymbol{s},d_j)}{\sigma}}} = \frac{\exp\left\{\frac{v(\boldsymbol{s},d_i)}{\sigma}\right\}}{\sum_j \exp\left\{\frac{v(\boldsymbol{s},d_j)}{\sigma}\right\}}$$

Thus if the choice set is $D(\boldsymbol{s})$ then

$$P(d|\boldsymbol{s}) = \frac{\exp\left\{\frac{v(\boldsymbol{s},d)}{\sigma}\right\}}{\sum_{d' \in D(\boldsymbol{s})} \exp\left\{\frac{v(\boldsymbol{s},d')}{\sigma}\right\}}$$

which is known as the *multinomial logit* formula.

C.3 Expected value function

We simplify notation by omitting dependency on $\boldsymbol{\theta}$ from these calculations. This will not affect the results. We want to show that

$$EV(\boldsymbol{s},d) = \iint V(\boldsymbol{s}',\boldsymbol{\epsilon}')q(d\boldsymbol{\epsilon}'|\boldsymbol{s}')p(d\boldsymbol{s}'|\boldsymbol{s},d)$$

$$= \int \sigma \log \left[\sum_{d' \in D(\boldsymbol{s}')} \exp \left\{ \frac{u(\boldsymbol{s}',d') + \beta EV(\boldsymbol{s}',d')}{\sigma} \right\} \right] p(d\boldsymbol{s}'|\boldsymbol{s},d) \tag{C.2}$$

where

$$V(\boldsymbol{s},\boldsymbol{\epsilon}) = \max_{d \in D(\boldsymbol{s})} \{u(\boldsymbol{s},d) + \boldsymbol{\epsilon}(d) + \beta EV(\boldsymbol{s},d)\} \tag{C.3}$$

If we insert (C.3) into equation (C.2), we see that what we essentially have to prove is that

$$\int \max_{d \in D(\boldsymbol{s})} \{v(\boldsymbol{s},d) + \boldsymbol{\epsilon}(d)\} \, q(d\boldsymbol{\epsilon}|\boldsymbol{s}) = \sigma \log \left[\sum_{d' \in D(\boldsymbol{s})} \exp \left\{ \frac{v(\boldsymbol{s},d')}{\sigma} \right\} \right]$$

where we for simplicity use

$$v(\boldsymbol{s},d) = u(\boldsymbol{s},d) + \beta EV(\boldsymbol{s},d)$$

as the value function seen by the econometrician. In addition, we need the definition of the conditional choice probability from appendix C.1

$$P(d|\boldsymbol{s}) = \int I\{d = \delta(\boldsymbol{s},\boldsymbol{\epsilon})\} q(d\boldsymbol{\epsilon}|\boldsymbol{s})$$

where $q(d\boldsymbol{\epsilon}|\boldsymbol{s})$ is the conditional probability distribution of $\boldsymbol{\epsilon}$ given $\boldsymbol{s}$. We are now ready to show the derivation of the relationship in equation (C.2).

$$\int \left[\max_{d \in D(\boldsymbol{s})} [v(\boldsymbol{s},d) + \boldsymbol{\epsilon}(d)] \right] q(d\boldsymbol{\epsilon}|\boldsymbol{s})$$

$$= \int \left[\int \frac{\partial \left\{ \max_{d \in D(\boldsymbol{s})} \left[v(\boldsymbol{s}, d) + \boldsymbol{\epsilon}(d) \right] \right\}}{\partial v(\boldsymbol{s}, d)} dv(\boldsymbol{s}, d) \right] q(d\boldsymbol{\epsilon}|\boldsymbol{s})$$

$$= \int \left[\int I \left\{ d = \arg \max_{d' \in D(\boldsymbol{s})} \left[v(\boldsymbol{s}, d) + \boldsymbol{\epsilon}(d) \right] \right\} dv(\boldsymbol{s}, d) \right] q(d\boldsymbol{\epsilon}|\boldsymbol{s})$$

Switch the order of integration,

$$= \int \left[\int I \left\{ d = \arg \max_{d' \in D(\boldsymbol{s})} \left[v(\boldsymbol{s}, d) + \boldsymbol{\epsilon}(d) \right] \right\} q(d\boldsymbol{\epsilon}|\boldsymbol{s}) \right] dv(\boldsymbol{s}, d)$$

use the definition of the conditional choice probability,

$$= \int P(d|\boldsymbol{s}) dv(\boldsymbol{s}, d)$$

$$= \int \frac{\exp \left\{ \frac{v(\boldsymbol{s}, d)}{\sigma} \right\}}{\sum_{d' \in D(\boldsymbol{s})} \exp \left\{ \frac{v(\boldsymbol{s}, d')}{\sigma} \right\}} dv(\boldsymbol{s}, d)$$

and integrate

$$= \log \left[\sum_{d' \in D(\boldsymbol{s})} \exp \left\{ \frac{v(\boldsymbol{s}, d')}{\sigma} \right\} \right]$$

■

C.4 Effect function

Lets assume that q is a quadratic function of $\bar{x}$ where

$$\bar{x} = x + \frac{dx}{2} - \frac{q\sum_{b=1}^{d} h_{bw}}{2}$$

$$q(\bar{x}) = c_1\bar{x}^2 + c_2\bar{x} + c_3$$

and the c_i's are coefficients from a regression based on a power plants reservoir geometry and head efficiency. If we substitute the expression for $\bar{x}$ into this effect function we get

$$q = c_1\left(x + \frac{dx}{2} - \frac{q\sum_{b=1}^{d} h_{bw}}{2}\right)^2 + c_2\left(x + \frac{dx}{2} - \frac{q\sum_{b=1}^{d} h_{bw}}{2}\right) + c_3$$

which is a standard second order equation of q. Solving this for q gives

$$q\left(x + \frac{dx}{2}, d, w\right) = \frac{1}{c_1\tilde{h}^2}\left(\left[2c_1x + c_1dx + c_2\right]\tilde{h} + 2 + \sqrt{\left[c_2^2 - 4c_1c_3\right]\tilde{h}^2 + 4\left[2c_1x + c_1dx + c_2\right]\tilde{h} + 4}\right) \tag{C.4}$$

where

$$\tilde{h} = \sum_{b=1}^{d} h_{bw}$$

For the case of no inflow equation (C.4) reduces to

$$q(x, d, w) = \frac{1}{c_1\tilde{h}^2}\left(\left[2c_1x + c_2\right]\tilde{h} + 2 + \sqrt{\left[c_2^2 - 4c_1c_3\right]\tilde{h}^2 + 4\left[2c_1x + c_2\right]\tilde{h} + 4}\right)$$

C.5 Fréchet derivative

We need to prove that for each pair $\{i, j\} \in Z$

$$\frac{\partial T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)}{\partial EV_\theta(\boldsymbol{\zeta}^j)} = \mathcal{P} \times \beta \frac{\sum_{d \in F} \psi(\boldsymbol{\zeta}^i, d, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x,w)} \psi(\boldsymbol{\zeta}^i, d', \boldsymbol{\theta})}$$

where F is the set of all decisions d such that $\boldsymbol{\zeta}^i(d) = \boldsymbol{\zeta}^j$. To do this we first derive the analogous for the continuous state space. We start with the contraction mapping formula

$$EV_\theta(\boldsymbol{\zeta}) = T_\theta(EV_\theta)(\boldsymbol{\zeta}) = \iint \log \left[\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}', d, \boldsymbol{\theta}) \right] p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta})$$

where

$$\psi(\boldsymbol{\zeta}', d, \boldsymbol{\theta}) = \exp \left\{ q(x + \frac{dx'}{2}, d, w) \sum_{b=1}^{d} \left[c_{bw}(y + dy') - \theta \right] h_{bw}(x + dx') + \beta EV_\theta(\boldsymbol{\zeta}'(d)) \right\}$$

For a pair $\{i, j\} \in Z$ the Fréchet derivative is defined as the derivative of $T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)$ with respect to $EV_\theta(\boldsymbol{\zeta}^j)$

$$\frac{\partial T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)}{\partial EV_\theta(\boldsymbol{\zeta}^j)}$$

$$= \frac{\partial}{\partial EV_\theta(\boldsymbol{\zeta}^j)} \iint \log \left[\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta}) \right] p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta})$$

$$= \iint \frac{\partial}{\partial EV_\theta(\boldsymbol{\zeta}^j)} \log \left[\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta}) \right] p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta})$$

$$= \iint \left[\frac{1}{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})} \frac{\partial \sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})}{\partial EV_\theta(\boldsymbol{\zeta}^j)} \right] p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta})$$

$$= \iint \left[\frac{1}{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})} \sum_{d=0}^{D(x,w)} \frac{\partial \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})}{\partial EV_\theta(\boldsymbol{\zeta}^j)} \right] p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta}) \tag{C.5}$$

We will for simplicity solve $\partial\psi/\partial EV$ separately

$$\frac{\partial \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})}{\partial EV_\theta(\boldsymbol{\zeta}^j)}$$

$$= \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta}) \frac{\partial}{\partial EV_\theta(\boldsymbol{\zeta}^j)} \left(q(x + \frac{dx'}{2}, d, w) \sum_{b=1}^{d} \left[c_{bw}(y + dy') - \theta \right] h_{bw}(x + dx') + \beta EV_\theta(\boldsymbol{\zeta}'^i(d)) \right)$$

$$= \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta}) \beta \frac{\partial EV_\theta(\boldsymbol{\zeta}'^i(d))}{\partial EV_\theta(\boldsymbol{\zeta}^j)}$$

and

$$\frac{\partial EV_\theta(\boldsymbol{\zeta}'^i(d))}{\partial EV_\theta(\boldsymbol{\zeta}^j)} = \begin{cases} 1 & \text{if } \boldsymbol{\zeta}'^i(d) = \boldsymbol{\zeta}^j \\ 0 & \text{otherwise} \end{cases}$$

If we use these findings in equation (C.5) we get

$$\frac{\partial T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)}{\partial EV_\theta(\boldsymbol{\zeta}^j)}$$

$$= \begin{cases} \iint \left[\frac{\psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})}{\sum_{d=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})} \beta \right] p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta}) & \text{if } \boldsymbol{\zeta}'^i(d) = \boldsymbol{\zeta}^j \\ 0 & \text{otherwise} \end{cases}$$

$$= \begin{cases} \iint \beta \frac{\psi(\boldsymbol{\zeta}'^i, d, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x,w)} \psi(\boldsymbol{\zeta}'^i, d', \boldsymbol{\theta})} p(dx'|w, \boldsymbol{\theta}) p(dy'|y, w, \boldsymbol{\theta}) & \text{if } \boldsymbol{\zeta}'^i(d) = \boldsymbol{\zeta}^j \\ 0 & \text{otherwise} \end{cases}$$

which is the continuous version of the Fréchet derivative. The complication when moving to a discrete state space is that there might be several decisions that will satisfy the restriction $\boldsymbol{\zeta}'^i(d) = \boldsymbol{\zeta}^j$, and this is where the set F comes in. Thus we need to verify for each decision d whether it satisfies $\boldsymbol{\zeta}'^i(d) = \boldsymbol{\zeta}^j$, and if it does include it in the set d.

As before when changing to the discrete case we interchange the integration with multiplication with a transition matrix, and change $\boldsymbol{\zeta}'^i$ to $\boldsymbol{\zeta}^i$. The result is

$$\frac{\partial T_\theta(EV_\theta)\left(\boldsymbol{\zeta}^i\right)}{\partial EV_\theta(\boldsymbol{\zeta}^j)} = \mathcal{P} \times \beta \frac{\sum_{d \in F} \psi(\boldsymbol{\zeta}^i, d, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x,w)} \psi(\boldsymbol{\zeta}^i, d', \boldsymbol{\theta})}$$

where F is the set of all decisions d such that $\boldsymbol{\zeta}^i(d) \equiv \boldsymbol{\zeta}^j$.

■

C.6 Partial derivatives of Log-likelihood

We will only derive $\partial \mathcal{L}_l^p / \partial \beta$ as the approach for $\partial \mathcal{L}_l^p / \partial \theta$ is almost identical.

We begin with

$$\mathcal{L}_l^p = \sum_{t=1}^{T} \log \left[\frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x_t,w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right]$$

with

$$\psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta}) = \exp \left\{ q(x_t, d, w_t) \sum_{b=0}^{d} \left(c_{bw_t} y_t - \theta \right) h_{bw_t} + \beta EV_\theta \left(\boldsymbol{\zeta}_t(d) \right) \right\}$$

In the process of calculating $\partial \mathcal{L}_l^p / \partial \beta$ we are going to need the derivatives of $\psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta}) / \partial \beta$, thus it is natural to begin with

$$\frac{\partial \psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta})}{\partial \beta} = \psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta}) \left(\beta \frac{\partial EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)}{\partial \beta} + EV_\theta\left(\boldsymbol{\zeta}_t(d)\right) \right)$$

If we now differentiate $\mathcal{L}_l^p$ we get

$$\begin{aligned}
&\frac{\partial \mathcal{L}_l^p}{\partial \beta} \\
&= \sum_{t=1}^{T} \left[\frac{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})}{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})} \frac{\partial}{\partial \beta} \left(\frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})}{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right) \right] \\
&= \sum_{t=1}^{T} \left[\frac{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})}{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})} \left(\frac{\frac{\partial \psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})}{\partial \beta} \sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})}{\left(\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta}) \right)^2} \right. \right. \\
&\qquad \left. \left. - \frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta}) \sum_{d'=0}^{D(x_t, w_t)} \frac{\partial \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})}{\partial \beta}}{\left(\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta}) \right)^2} \right) \right]
\end{aligned}$$

We now introduce the derivative of ψ with respect to β derived above

$$\frac{\partial \psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta})}{\partial \beta} = \psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta}) \left(\beta \frac{\partial EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)}{\partial \beta} + EV_\theta\left(\boldsymbol{\zeta}_t(d)\right) \right)$$

and define

$$\phi_\beta(d) = \beta \frac{\partial EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)}{\partial \beta} + EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)$$

Using this new notation, we can simplify to

$$= \sum_{t=1}^{T} \left[\frac{1}{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})} \left(\frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta}) \phi_\beta(d_t) \sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})}{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} - \frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta}) \sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta}) \phi_\beta(d')}{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right) \right]$$

$$= \sum_{t=1}^{T} \left[\frac{1}{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta})} \left(\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta}) \phi_\beta(d_t) - \frac{\psi(\boldsymbol{\zeta}_t, d_t, \boldsymbol{\theta}) \sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d, \boldsymbol{\theta}) \phi_\beta(d')}{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta})} \right) \right]$$

$$= \sum_{t=1}^{T} \left[\phi_\beta(d_t) - \frac{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta}) \phi_\beta(d')}{\sum_{d'=0}^{D(x_t, w_t)} \psi(\boldsymbol{\zeta}_t, d', \boldsymbol{\theta\theta})} \right]$$

where

$$\phi_\beta(d) = \beta \frac{\partial EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)}{\partial \beta} + EV_\theta\left(\boldsymbol{\zeta}_t(d)\right)$$

■

C.7 Sparse solve convergence

The algorithm converges if and only if

$$\lim_{k\to\infty} \frac{||\boldsymbol{\epsilon}_w^{k+1}||}{||\boldsymbol{\epsilon}_w^{k}||} < 1,$$

thus we need an expression for the error of the k-th iteration $\boldsymbol{\epsilon}_w^k$. Recall that

$$\boldsymbol{u}_w^{k+1} = T_\theta^w \boldsymbol{u}_{w+1}^k + \boldsymbol{b}_w$$

If we start with $\boldsymbol{u}_w^0 = \mathbf{0}$, then $\boldsymbol{\epsilon}_w^0 = \boldsymbol{u}_w^*$ where $\boldsymbol{u}_w^*$ is the final solution. Let's write out the first iterations:

$$\begin{aligned}
\boldsymbol{u}_w^1 &= \boldsymbol{b}_w \\
\boldsymbol{u}_w^2 &= \boldsymbol{b}_w + T_\theta^w \boldsymbol{b}_{w+1} \\
\boldsymbol{u}_w^3 &= \boldsymbol{b}_w + T_\theta^w (\boldsymbol{b}_{w+1} + T_\theta^{w+1} \boldsymbol{b}_{w+2}) \\
&= \boldsymbol{b}_w + T_\theta^w \boldsymbol{b}_{w+1} + T_\theta^w T_\theta^{w+1} \boldsymbol{b}_{w+2} \\
\boldsymbol{u}_w^4 &= \boldsymbol{b}_w + T_\theta^w (\boldsymbol{b}_{w+2} + T_\theta^{w+1} (\boldsymbol{b}_{w+1} + T_\theta^{w+2} \boldsymbol{b}_{w+3})) \\
&= \boldsymbol{b}_w + T_\theta^w \boldsymbol{b}_{w+2} + T_\theta^w T_\theta^{w+1} \boldsymbol{b}_{w+1} + T_\theta^w T_\theta^{w+1} T_\theta^{w+2} \boldsymbol{b}_{w+3} \\
&\vdots \\
\boldsymbol{u}_w^k &= \boldsymbol{b}_w + \sum_{i=1}^{k-1} \left[\prod_{j=0}^{i-1} T_\theta^{w+j} \right] \boldsymbol{b}_{i+w} \\
&\vdots \\
\boldsymbol{u}_w^* &= \boldsymbol{b}_w + \lim_{k\to\infty} \sum_{i=1}^{k-1} \left[\prod_{j=0}^{i-1} T_\theta^{w+j} \right] \boldsymbol{b}_{i+w}
\end{aligned}$$

then

$$\begin{aligned}
\boldsymbol{\epsilon}_w^k &= \boldsymbol{\epsilon}_w^0 - \boldsymbol{b}_w - \sum_{i=1}^{k-1} \left[\prod_{j=0}^{i-1} T_\theta^{w+j} \right] \boldsymbol{b}_{i+w} \\
&= \lim_{s\to\infty} \sum_{i=1}^{s-1} \left[\prod_{j=0}^{i-1} T_\theta^{w+j} \right] \boldsymbol{b}_{i+w} - \sum_{i=1}^{k-1} \left[\prod_{j=0}^{i-1} T_\theta^{w+j} \right] \boldsymbol{b}_{i+w}
\end{aligned}$$

and

$$\frac{||\boldsymbol{\epsilon}_w^{k+1}||}{||\boldsymbol{\epsilon}_w^{k}||} = \frac{\left|\lim_{s\to\infty}\sum_{i=1}^{s-1}\left[\prod_{j=0}^{i-1}T_\theta^{w+j}\right]\boldsymbol{b}_{i+w} - \sum_{i=1}^{k}\left[\prod_{j=0}^{i-1}T_\theta^{w+j}\right]\boldsymbol{b}_{i+w}\right|}{\left|\lim_{s\to\infty}\sum_{i=1}^{s-1}\left[\prod_{j=0}^{i-1}T_\theta^{w+j}\right]\boldsymbol{b}_{i+w} - \sum_{i=1}^{k-1}\left[\prod_{j=0}^{i-1}T_\theta^{w+j}\right]\boldsymbol{b}_{i+w}\right|}.$$

Since

$$\left|\sum_{i=1}^{k-1}\left[\prod_{j=0}^{i-1}T_\theta^{w+j}\right]\boldsymbol{b}_{i+w}\right| < \left|\sum_{i=1}^{k}\left[\prod_{j=0}^{i-1}T_\theta^{w+j}\right]\boldsymbol{b}_{i+w}\right|,$$

then

$$\lim_{k\to\infty}\frac{||\boldsymbol{\epsilon}_w^{k+1}||}{||\boldsymbol{\epsilon}_w^{k}||} < 1.$$

■

C.8 Reduction of exponentials

In addition to running time issues, we were faced with another problem when we implemented the model. Reduction of exponentials is a concept we used extensively throughout the algorithm, and has to do with numerical limitations when exponentiating large numbers on a computer. We will use the conditional choice probability

$$P(d|\boldsymbol{\zeta}) = \frac{\exp\left\{u(\boldsymbol{\zeta},d)+\beta EV(\boldsymbol{\zeta},d)\right\}}{\sum_{d'\in D(\boldsymbol{\zeta})}\exp\left\{u(\boldsymbol{\zeta},d')+\beta EV(\boldsymbol{\zeta},d')\right\}}$$

to demonstrate, where have omitted all irrelevant variables and parameters for simplicity.

In the choice probabilities, we exponentiate the value of $u(\boldsymbol{\zeta},d)+\beta EV(\boldsymbol{\zeta},d)$ in both the numerator and the denominator. This value represents the sum of all utility from today and to infinity, and will in general become very large. When exponentiating values of this size in MATLAB the result will be infinity (the limit is e^{709}), and we therefore need a little trick.

Observe that

$$\begin{aligned} P(d|\boldsymbol{\zeta}) &= \frac{\exp\left\{u(\boldsymbol{\zeta},d)+\beta EV(\boldsymbol{\zeta},d)\right\}}{\sum_{d'\in D(\boldsymbol{\zeta})}\exp\left\{u(\boldsymbol{\zeta},d')+\beta EV(\boldsymbol{\zeta},d')\right\}} \\ &= \frac{\exp\left\{u(\boldsymbol{\zeta},d)+\beta EV(\boldsymbol{\zeta},d)\right\}}{\sum_{d'\in D(\boldsymbol{\zeta})}\exp\left\{u(\boldsymbol{\zeta},d')+\beta EV(\boldsymbol{\zeta},d')\right\}}\frac{\exp\{-K\}}{\exp\{-K\}} \\ &= \frac{\exp\left\{u(\boldsymbol{\zeta},d)+\beta EV(\boldsymbol{\zeta},d)-K\right\}}{\sum_{d'\in D(\boldsymbol{\zeta})}\exp\left\{u(\boldsymbol{\zeta},d')+\beta EV(\boldsymbol{\zeta},d')-K\right\}} \end{aligned}$$

for some value K. If we now set

$$K = \max_{d'}\left\{u(\boldsymbol{\zeta},d')+\beta EV(\boldsymbol{\zeta},d')\right\}.$$

Then all values $\{u(\boldsymbol{\zeta},d')+\beta EV(\boldsymbol{\zeta},d')-K\}$ will be zero or less, and can safely be exponentiated without affecting the validity of the results. This method must be used whenever we calculate the exponential of the value function.

Appendix D Data

D.1 Partition of weeks

Year	First day of year	Leap year	Period (52 weeks)	Event
2000	Saturday	1	3/1-2000 - 31/12-2000	3/1 = Monday
2001	Monday	0	1/1-2001 - 30/12-2001	
2002	Tuesday	0	31/12-2001 - 29/12-2002	
2003	Wednesday	0	30/12-2002 - 28/12-2003	
2004	Thursday	1	29/12-2003 - 26/12-2004	Delete week 53
2005	Saturday	0	3/1-2005 - 1/1-2006	
2006	Sunday	0	2/1-2006 - 31/12-2006	
2007	Monday	0	1/1-2007 - 30/12-2007	
2008	Tuesday	1	31/12-2007 - 28/12-2008	

Table D.1: Partitioning of weeks.

D.2 Power producer data

We originally had data from 14 producers available, from which we selected two. Selecting only two of the 14 producers might seem strange, but for reasons of exposition, it has been necessary to restrict ourselves to power stations with a very simple structure. By "simple structure", we mean the following:

- one reservoir
- one turbine
- available data for *inflow, reservoir level* and *production*
- no pump-storage

One would think that including a power station with, say, two turbines instead of one, or several reservoirs instead of one, should be a relatively simple task. The fact of the matter is that this is simply not the case. We discuss this in section 5.3, but for now, suffice it to say that switching to, say, two turbines, would require reprogramming huge portions of the algorithm. We discuss the data and our selection criteria below.

Our data selection criteria are similar to those used by Kolsrud and Prokosch (2009), albeit for different reasons. The criteria are:

Producer is a price taker: This criteria is necessary to avoid the problem of simultaneity. A situation of simultaneity arises when the production decision of the producer affects price at the same time as price affects the production decision. Using data from a non-price taker would necessitate a much more complicated model, since the production decision of the producer would affect the probability distribution of prices in the transition matrices.

Independent power station: The production decision of the power station is independent of the decisions made by other power stations. In other words, power stations that are coupled with other power stations are excluded.

One reservoir: A complicated topology (combination of reservoirs) has a confounding effect on the recorded production decision, since the producer may control the timing of inflow to lower reservoirs.

One turbine: For each turbine one includes, one increases the decision space of the producer. This has a dramatic impact on the time and space required by algorithm. The curse of dimensionality entails that even an algorithm with polynomial time- and space-complexity will suffer significant increases in time to convergence. Thus, we expect that, say, doubling the decision space would *at least* double the space and time requirements, if not *significantly more*. True, the production decision for separate turbines are quite positively correlated (we typically see correlations lying in the range 0.4-0.6), but we consider this to be too low to allow decisions to apply to several turbines.

Both power stations that we have considered fulfill the criteria above.

We have *hourly* production data from each producer for the period January 1, 2000 to December 31, 2008. Since we needed presicely 52 weeks per

year, we had to partition the production data in a very specific way. This partitioning can be seen in appendix D.1. The production data is used directly as input to the algorithm.[1]. Further, we have daily inflow and reservoir level data. The time interval of state transitions is in weeks, so we have simply summed the inflow and used the reservoir level at the end of each week as input. Plots of inflow, production and reservoir levels can be seen in the plots below.

[1]Recall, we group decisions together in a very specific manner, as described in section 3.2.3

Appendix E Plots

When calculating transition matrices, we essentially fit probability distributions for each variable and for each week to the real historical data of inflow and spot prices. Using these distributions, we generate a joint probability distribution that expresses the probability of moving from state $\boldsymbol{\zeta}^i$ to $\boldsymbol{\zeta}^j$. By continuing this process for all states in all weeks, we end up with the elements $\{i, j\}$ of the full Markov transition matrix $\mathcal{P}$.

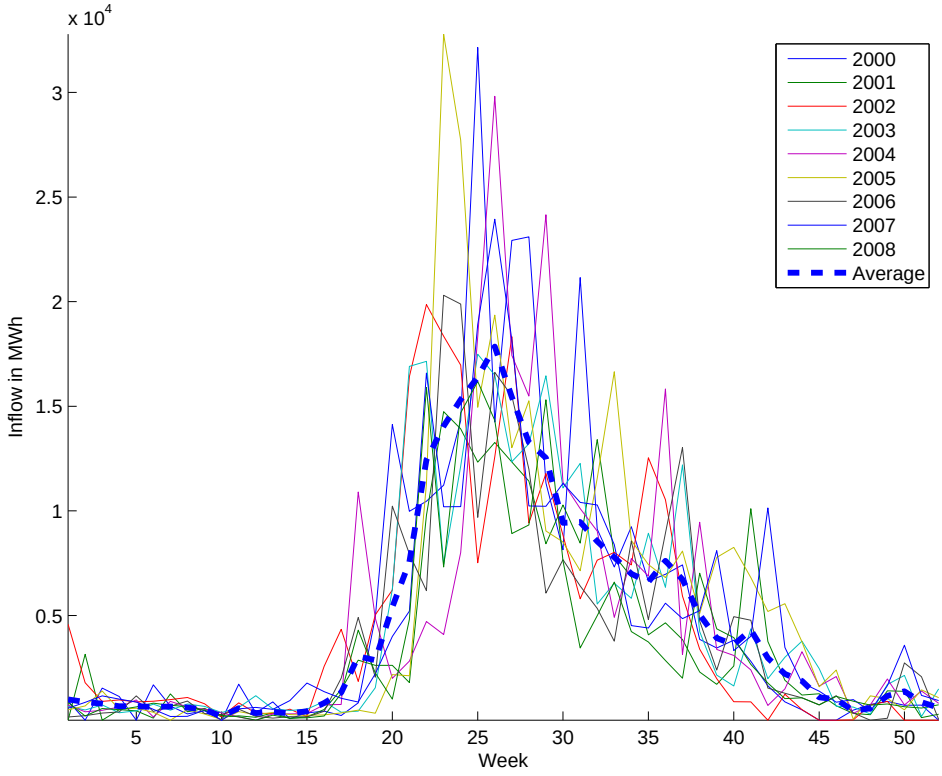

Figure E.1: Inflow, hydropower producer I

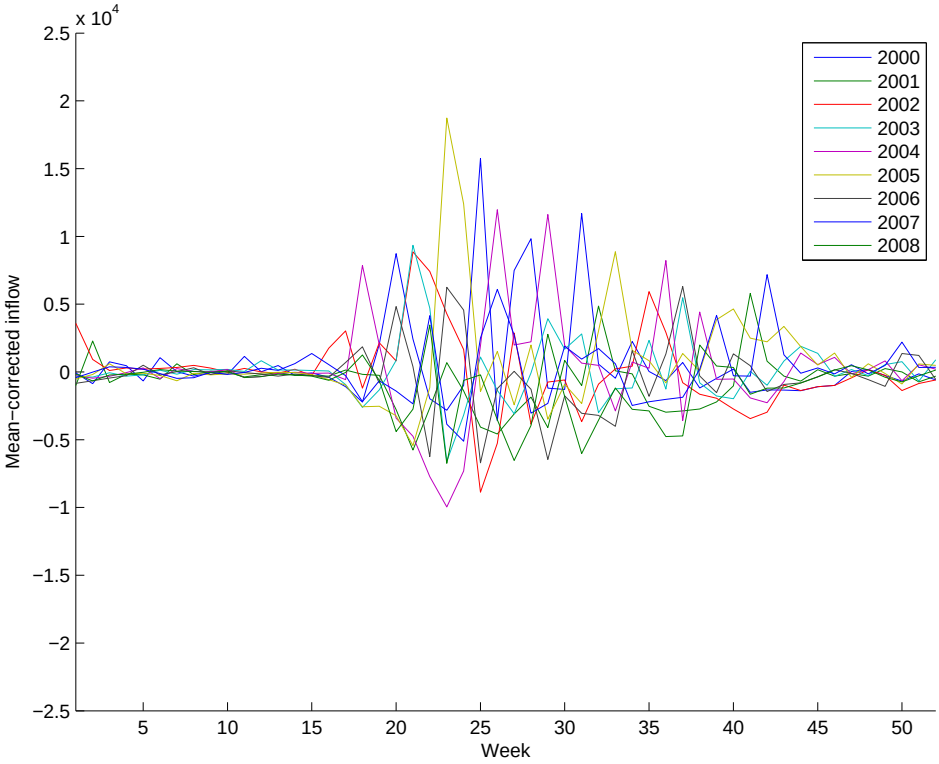

Figure E.2: Mean-corrected inflow, hydropower producer I

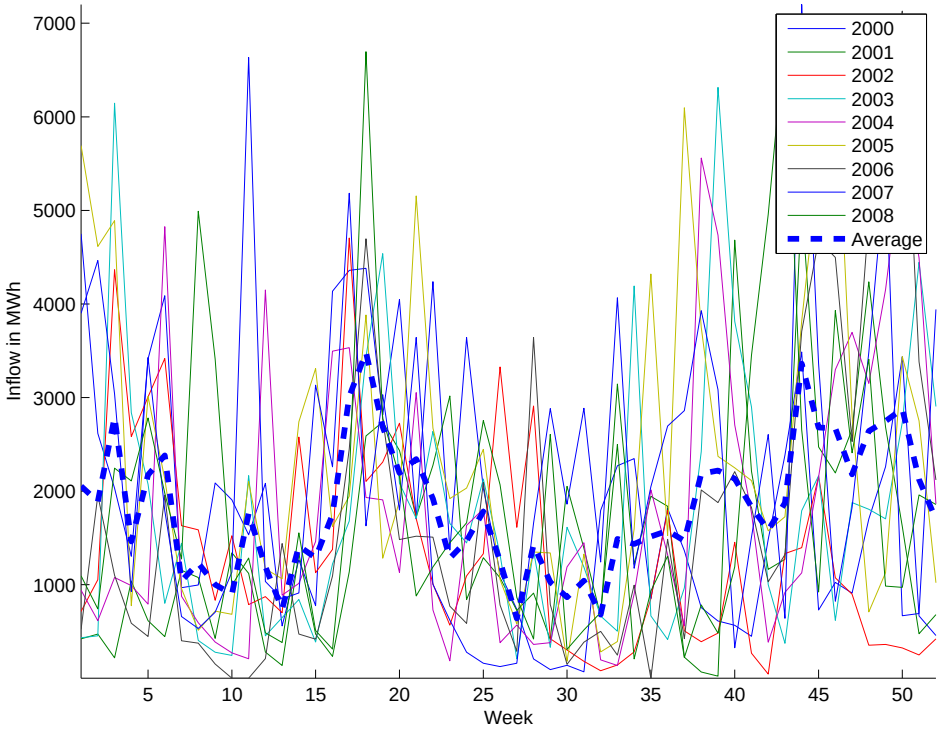

Figure E.3: Inflow, hydropower producer II

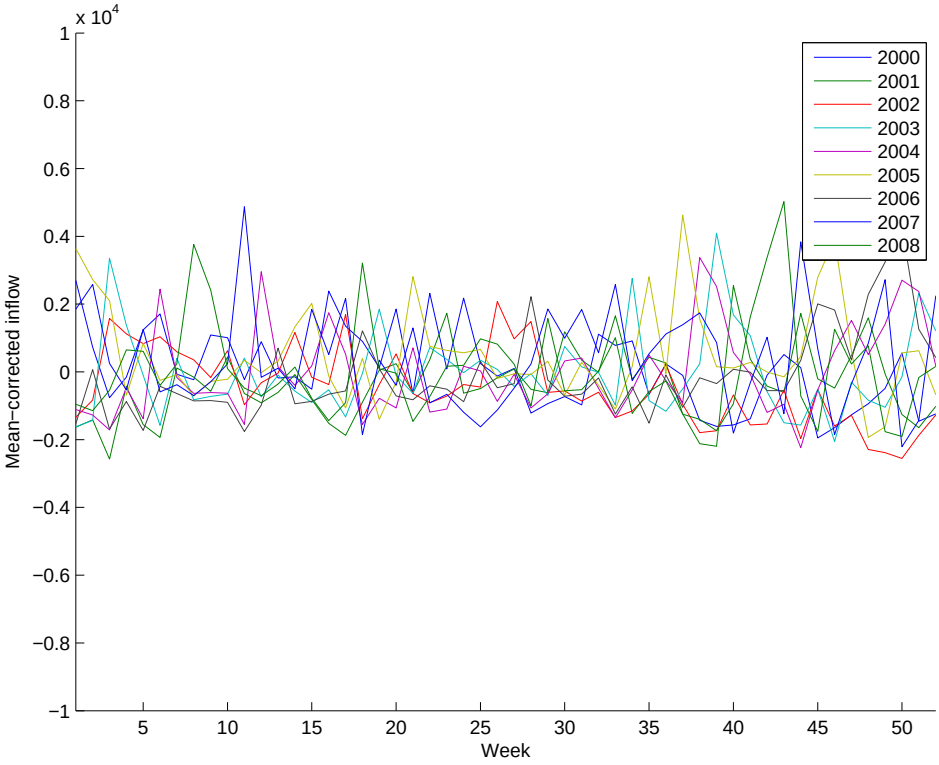

Figure E.4: Mean-corrected inflow, hydropower producer II

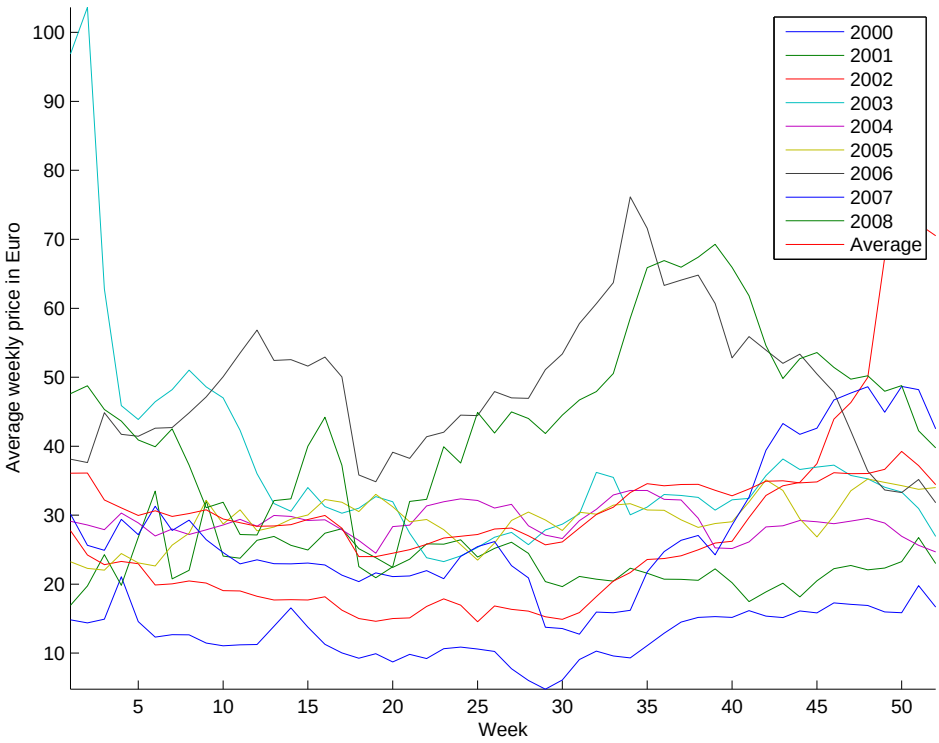

Figure E.5: Average weekly prices, 2000-2008