NTNU
Norwegian University of
Science and Technology

The Academic Version of a Fundamental Market Model

Vegard Paulsen Særen

Master of Energy and Environmental Engineering
Submission date: June 2018
Supervisor: Hossein Farahmand, IEL
Co-supervisor: Martin Hjelmeland, IEL

Norwegian University of Science and Technology
Department of Electric Power Engineering

NTNU – Trondheim
Norwegian University of
Science and Technology

Academic version of a fundamental market model

Vegard Paulsen Særen

June 2018

MASTER THESIS

Faculty of Information Technology and Electrical Engineering

Norwegian University of Science and Technology

Supervisor 1: Associate Professor Hossein Farahmand

Supervisor 2: Martin N. Hjelmeland

Preface

This master thesis is a part of the two-year master program *"Energy Use and Energy Planning"* at NTNU in Trondheim. The thesis was carried out in the spring semester of 2018. The project is done with the collaboration of the Faculty of Information Technology and Electrical Engineering and the Department of Electrical Power Engineering at NTNU. The master thesis is a part of SINTEF Energy and HYDROCEN.

I would like to thank the following persons for their great help during the master thesis work this semester. Hossein Farahmand for guidance and continuous supervision. Martin N. Hjelmeland for helping me understand the Python and Pyomo, and the problem. I would also like to thank my co-students for great discussions and advise when needed.

Trondheim, 2018-06-11

[illegible]

Summary

In this master thesis, the making of a fundamental market model will take place. To understand the model, there is a chapter that describes the theory behind the calculations in the model. The focus of the model is to calculate water values, to be analyzed further. In this thesis, there are 4 main models that is described. The first model is a deterministic model, calculating the water values of fixed input data. The second model is a stochastic model that calculates water values from one area, using 15 price nodes and 5 inflow nodes. The third model is a two-area model that is interconnected with a loss-free line. The fourth main model is the fundamental market model, including 3 areas that are interconnected with losses on the lines. The master thesis also contains a sensitivity analysis of including wind energy into the mix.

To analyze the results, all the models have been tested and produced water values. The water values are further analyzed and have shown the difference in all the models. In this thesis, one can see the impact of including the option of buying and selling energy, and the impact of losses on the lines. Furthermore, the sensitivity analysis shows that wind energy can reduce the water values, and become a contributor for decommissioning thermal energy.

The conclusion of the thesis relay on the contribution of stochastic variables. The analysis shows that the accuracy of the calculations can be improved by increasing the reservoir levels and the price- and inflow nodes. This is because the model has simplifications and assumptions that makes the solution time lesser. By increasing the nodes and discrete reservoir levels one increases the solution time, but also the accuracy in the water values.

Sammendrag

I denne masteroppgaven vil det bli gjennomført en grunnleggende markedsmodell. For å forstå modellen, er det et kapittel som beskriver teorien bak beregningene i modellen. Modellens fokus er å beregne vannverdier, som skal analyseres videre. I denne oppgaven er det 4 hovedmodeller som er beskrevet. Den første modellen er en deterministisk modell, som beregner vannverdiene for faste inngangsdata. Den andre modellen er en stokastisk modell som beregner vannverdier fra ett område, ved hjelp av 15 prisnoder og 5 nedbørsnoder. Den tredje modellen er en to-områdesmodell som er sammenkoblet med en tapsløs linje. Den fjerde hovedmodellen er den grunnleggende markedsmodellen, inkludert 3 områder som er sammenkoblet med tap på linjene. Mastergradsoppgaven inneholder også en følsomhetsanalyse av blant annet vindkraft i blandingen.

For å analysere resultatene har alle modellene blitt testet og produsert vannverdier. Vannverdiene analyseres videre og viser forskjellen i alle modellene. I denne oppgaven kan man se effekten av å inkludere muligheten til å kjøpe og selge energi, og effekten av tap på linjene. Videre viser følsomhetsanalysen at vindenergi kan redusere vannverdiene, og bli en bidragsyter for nedstengning av termisk energi.

Konklusjonen av masteroppgaven, bygger på bidrag fra stokastiske variabler. Analysen viser at nøyaktigheten av beregningene kan forbedres ved å øke reservoarnivåene og pris- og nedbørsnoderne. Dette skyldes at modellen har forenklinger og forutsetninger som gjør oppløsningstiden mindre. Ved å øke noder og diskrete reservoarnivåer øker man oppløsningstiden, men også nøyaktigheten i vannverdiene.

List of Symbols

Symbol	Description
α	Total expected future profit
$\mathbb{L}$	Operational depended cost of a decision
$\mathbb{P}$	Probability
$\mathbb{E}$	Expected profit/cost
D	Demand
C	Cost of thermal production
P	Production
s	Stochastic variable
n	Node
t	Time stage
x	Decision variable
V	Discrete reservoir level
j	Option
I	Inflow
L	Lagrange
λ	Lagrangian multiplier
μ	Lagrangian multiplier
δ	Partial derivative
min	Minimization
max	Maximization

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we would like to develop the academic version of a long-term hydro power scheduling tool. The market model consists of an hydro power station that provides flexibility to the grid and a thermal power plant to provide power when needed. We are interested in calculating the water values of the reservoir, based on the cost of thermal production. In this thesis we also include several areas with transmission losses to evaluate the impact on the water values.

We approach the problem by making a model based on the principals of stochastic dynamic programming (SDP). The usage of SDP is necessary because the problem is a multistage stochastic problem which includes several inflow and price scenarios.

The thesis reflect on the contribution of wind energy in a sensitivity analysis. The analysis shows that the wind energy is increasing the flexibility of the hydro power, resulting in a reduction in the water values. The case studies shows the impact of each extension to the model.

We will make use of a program called Python. We use Python because it is a open-source program, which means it is available, for free, everywhere. This will contribute to the fact that this model will have easy access for power companies and case studies. Furthermore we make use of an additional software in Python called Pyomo. Pyomo is an optimization tool that simplifies the way of constructing optimization problems.

1.1 Background motivation

The European Union (EU) have several long-term goals to reduce greenhouse gases in the future, which will impact the Nordic grid and restructuring is needed. Though the decommission of coal and gas generation will be replaced by new renewable energy, the changes will lead to less flexibility and more unregulated power production. In 2015, a total of about 27.3 GW of new power generation capacity were connected in the EU and 18.2 GW were decommissioned,resulting in 9.1 GW of new net capacity (Fig. 1.1). Renewable energy sources (RES) accounted for 20.6 GW or 75.6% of all new power generation capacity [14].

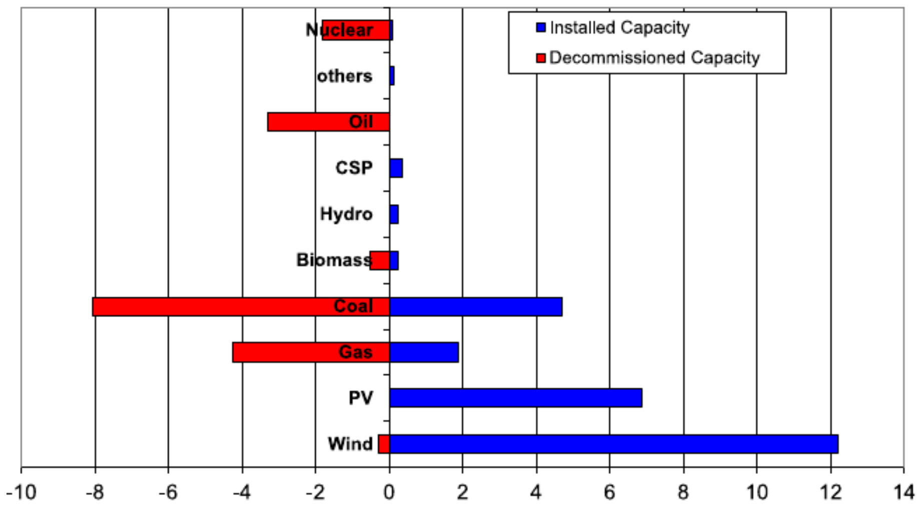


Figure 1.1: New installed or decommissioned capacity in 2015 [GW]

Goals for 2020-2030

As we approach 2020, the energy policy in Europe are drastically changing. The first goals for the European energy policy was the 20/20/20 targets which are 20% cut in greenhouse gases from the 1990 levels, 20% increase of renewable energy production in Europe and 20% improved energy efficiency by the year 2020. Current energy and climate policies are delivering substantial progress towards these 20/20/20 targets [4]:

• Greenhouse gas emissions in 2012 decreased by 18% relative to emissions in 1990 and are ex-pected to reduce further to levels 24% and 32% lower than in 1990 by 2020 and 2030 respectively on the basis of current policies.

• The share of renewable energy has increased to 13% in 2012 as a proportion of final energy consumed and is expected to rise further to 21% in 2020 and 24% in 2030.

• The EU had installed about 44% of the world's renewable electricity (excluding hydro) at the end of 2012.

• The energy intensity of the EU economy has reduced by 24% between 1995 and 2011 whilst the improvement by industry was about 30%.

• The carbon intensity of the EU economy fell by 28% between 1995 and 2010.

After analyzing the progress of the ongoing goals, changes have been made to improve the goals of 2030. The new goals that are set for 2030 is a cut in greenhouse gasses by 40%, an increase in renewable energy production with 27% and improved energy efficiency by 27%.

1.2 Scope

The problems of the model is to make an academic version of a fundamental market model. This includes extending a model from deterministic to stochastic and then including more areas. This is carried out in the research by making use of python. The goal of the thesis is to build a functional model and to conduct market simulations for analysis.

While working with the thesis, the author has been facilitated by the supervisors through regular meetings.

1.3 Contribution

The contribution from the author to this research have been:

A : Extending the deterministic model to stochastic.

B : Extending the model by multiple areas.

C : Extending the model with wind energy to conduct sensitivity analysis.

D : Conducting simulations for analysis.

Chapter 2

Literature survey

Hydro power scheduling problems have been a field of work for a long time. Some of the first work on the field is found in [15] and partly in [8], that made use of the SDP method. In 1961, Stage[**?**] and Lindqvist[8] wanted to find the incremental cost of water power. The incremental cost of hydro power production is the marginal cost of increasing the production by one unit of power. They also made use of the iteration process, by updating the water values with the values at the final stage. They also introduced the SDP method for calculating this. SDP is the stochastic dynamic programming that takes into account the uncertainty in the variables.

Linqvist[8], the year after, wrote a paper on the hydroelectric model of calculating the incremental cost of operation. He actually made a graph, describing the incremental cost of water, with respect to storage capacity in the reservoir. He shows, that the cost of producing decreases as the reservoir storage increases, which today, is basic knowledge. After some years the stochastic dual dynamic programming (SDDP) was taken into account in [9, 10, 11]. As for Lindquist and Stage/Larsson , their only focus was on the aggregated model, a calculation method for several reservoirs was needed. This was the beauty of SDDP, which allows the state variables to not be discretisized. Although it had its problems with facilitating non-convexities. Some of the latest research, that have been used to model scheduling of energy and reserve markets is represented by [1, 2].

For fundamental market models there are some research papers that touch upon the subject.

For example, [12] touches upon introducing reserve capacity and wind generation in a hydro dominated system. This study shows the impact of including wind to a power system that can export and import energy when needed, which was highly relevant to this thesis. For multi-area systems, the EMPS model is often used by the Nordic participants. The model is described in [13]

In addition to this, the curriculum of some courses at NTNU offers an detailed description of calculation of water values, and SDP. These curriculums can be found as compendiums in [6, 5] and the teaching book designed for the course "*Power Markets*" is found in [17].

Chapter 3

Background theory

Section 3.3-3.4 is taken from the project report.

3.1 Trends in European Share of Genaration

To understand how the European grid will change among the years it is important to look at the change in the trends for the future. These changes will impact the rehabilitation and strategies for building a sufficient and secure Nordic power grid.

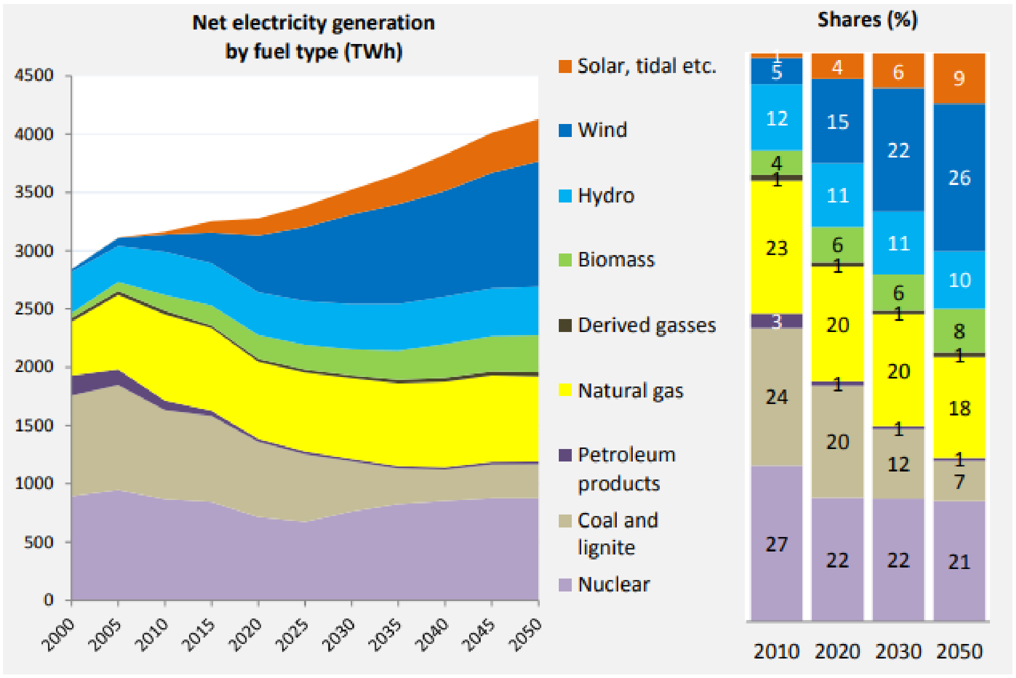


Figure 3.1: Trends in electricity generation 2000-2050[3]

If one look at the trends in electricity generation in figure 3.1, one can see that the fossil generation decreases and renewable energy increases. It is important to understand that the coal and other fossil energy sources works as base load and that the decommission of a share in base load with unregulated renewable energy will make the grid unstable to deliver secure and sufficient.

The base load is usually govern by heavy industries, households and a tertiary containing mainly IT, leisure and communication appliances. The trend in demand for electricity is showed in figure 3.2, and by comparison one can see the effect on the decommissioned coal and fossil fuels by there share in the total demand.

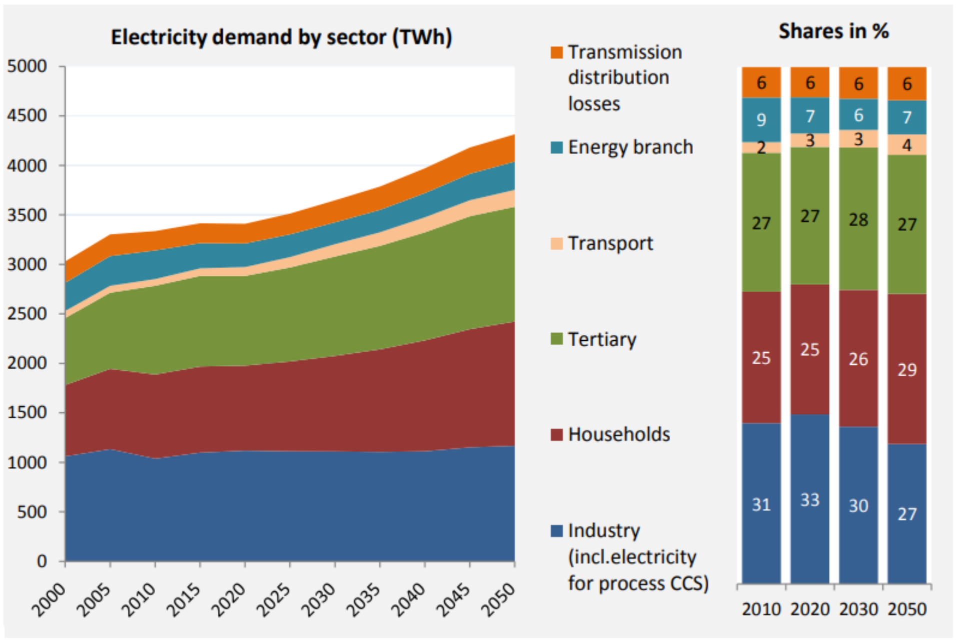


Figure 3.2: Trends in demand 2000-2050[3]

The installed capacity throughout the years will have an increase in renewable energy. A trend estimation is done in figure 3.3, and it shows the decommission of solids and gas. It also correspond to the trend in demand from figure 3.2, which makes sense.

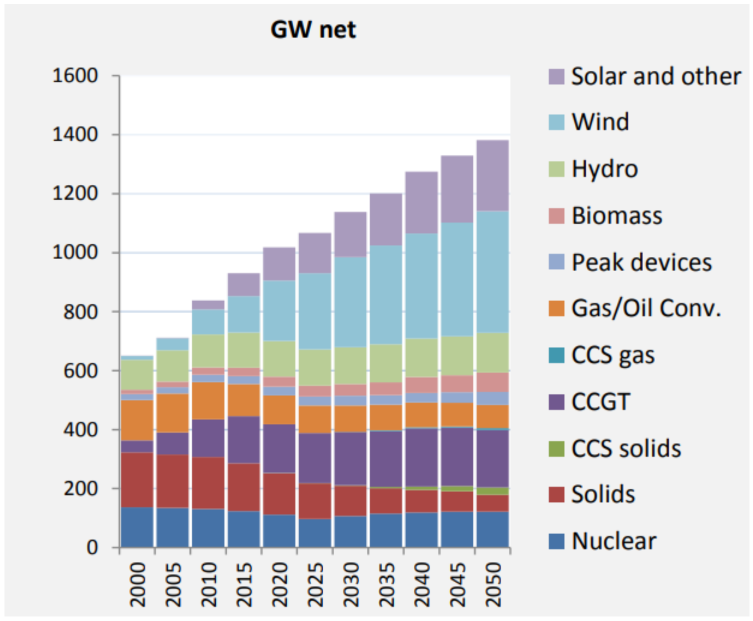


Figure 3.3: Trends in installed capacity 2000-2050[3]

The trends shown above indicates how the European power grid will change in time. This is important to the thesis, because the price of producing wind energy and solar energy is significantly lesser than hydro power. This leads to a change in the market, with an increase in capacity sales and more activity on the regulating market.

3.2 Nord Pool Market

The Nordic power market contains of several markets that the producers can participate in. The Nordic market is controlled by an impartial market operator that is owned by the transmission operators (TSO's) in Norway (Statnett), Sweden (Svenska kraftnät), Denmark (Energinet), Finland (Fingrid), Estonia (Elering), Lithuania (Litgrid) and Latvia (AST), called Nord Pool AS. The Nord Pool market contains of two physical markets in the nordic marked; the Day-Ahead market and the Intra-Day market. The rest of the Nordic market is controlled by the TSO's in each country. The financial aspect of the market is controlled by NASDAQ, which is a stock market. An illustration of the time window of the different markets are shown in figure 3.4.

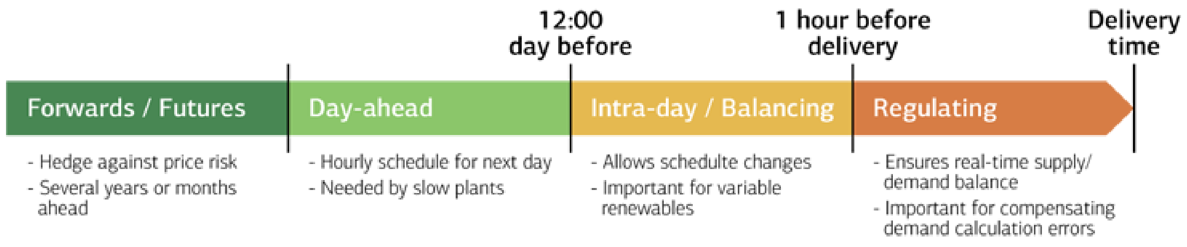

12:00
day before
1 hour before
delivery
Delivery
time
Forwards / Futures
Day-ahead
Intra-day / Balancing
Regulating
- Hedge against price risk
- Several years or months ahead
- Hourly schedule for next day
- Needed by slow plants
- Allows schedulte changes
- Important for variable renewables
- Ensures real-time supply/ demand balance
- Important for compensating demand calculation errors

Figure 3.4: Nord Pool market discription [7]

Day-ahead market

The Day-Ahead market (DAM) is the largest market for European power exchange. This is the market were buyers and sellers makes bids for purchase and selling power for the next day. Nord Pool manages bids from producers and purchasers where they specify the price and the quantity of the power. At 12:00 CET the day before, the market will close and the bids will be sorted out to create demand- and supply-curves. The market equilibrium is found where the demand-curve crosses the supply-curve. The market equilibrium sets the market price which is the same for all participants. The market equilibrium also sets the quantity of the total amount of sold energy.

Intra-day market

Since the DAM sets the market equilibrium one day ahead of delivery, there are many things that could prevent the energy of being produced. To keep a balance between demand and supply, the Intra-Day market (IDM) provides a market where participants can sell and buy power from 14:00 CET until 1 hour before delivery. The IDM is also managed by Noord Pool, and works to keep the DAM in balance.

Because of an increase in unregulated renewable energy generation, the IDM is expected to increase with the years to come. As seen in section 3.1, the increase in wind energy generation will increase the uncertainty of delivery to the DAM and the need for IDM will be higher.

Balancing market

The Balancy market opens one hour before the market closes. This market is run by the TSO's to ensure that unbalances in the grid is taken care of. Producers can regulate the balance in three reserves, primary, secondary and tertiary reserves. In this market, the goal is to keep the frequenze at 50 Hz.

Reserve capacity market

The reserve capacity market, known as the RKOM, is to ensure stability in the nordic system and is also run by the TSO's. This market is needed to keep reserve capacity on stand by if an situation would occur. The market can trade on a weekly basis or seasonal basis.

The rest of the theory is taken from the master project.

3.3 Introduction to stochastic dynamic programming

To understand the basics of stochastic dynamic programming(SDP), one need to understand the concept of dynamic programming, hereafter referred as *DP*, and the usage of it. While DP is a way of optimizing a problem, when taken into account the total cost of all decisions, the SDP includes the opportunity to make use of uncertainty by creating sub-optimal solutions to the problem. Every sub-optimal problem has a probability of occurring, which is multiplied with the sub-optimal solution to find the weighted optimal solution in each state. In a DP-problem, the parameters are deterministic, meaning they are fixed in each stage.

Dynamic programming

In SDP-problems the objective is to minimize future expected cost, or maximize total future profit of an operation, when the decision of moving from one stage to another stage comes with an uncertainty. Since maximizing profits is the same as minimizing the cost of the operation, provided that the load is inelastic, the objective function will be of respect to the costs. Take into account a one-stage linear minimization problem

$$\min \mathbb{E}\left\{\sum CP\right\} \tag{3.1}$$

where

$\mathbb{E}$ is expected value

C is cost of production

P is amount of production

without any further information this problem is simply an equation where the sum of cost of production equals the total value. This formulation can be expanded into a DP-problem, by adding decision variables and multiple stages, as explained in 3.3

$$\alpha_t(x_t) = \min\left\{\sum \mathbb{L}(x_t) + \alpha_{t+1}(x_{t+1})\right\} \tag{3.2}$$

where

α_t is the future expected cost

x_t is the decision variable

$\mathbb{L}$ is the operating dependent cost from decision x_t

In (3.2), t is the stage index of the DP-problem and it is used for time, since the stages represents hours, weeks, or years. The introduction of a decision variable must correspond with an additional total cost of the decisions, hence $\alpha_{t+1}(x_{t+1})$. This part ensures that the decisions will effect the total future expected costs and is referred to as the dynamic part of the problem.

Stochastic dynamic programming

Take into account the functionality of DP, and the way to solve DP-problems. Imagine that the states comes with an uncertainty, or if the cost of transition comes with an uncertainty. The DP- problem becomes even more complex and the way of solving it is even harder. The way of solving it can be with recursion, and is called backwards SDP. Backwards SDP utilizes the future cost of operation, so when you are calculating the optimal in a state, you always know the optimal operational cost of the future. This means that one calculates from the last stage to the first stage of the year. SDP builds upon DP in a way which one or more variable(s) with uncertainty is added. These variables is called *stochastic variables* and represents the uncertainty of making a decision.

To describe the problem one must establish these new variables. The stochastic variables are set for each node, but the transition of getting that variable comes with an uncertainty. How to generate these probabilities are another field, which this report will not touch upon, but the theory behind will be explained shortly in 3.4.

$$\alpha_t(s_t) = \min_{x_t}\left\{\mathbb{L}(x_t, s_t) + \mathbb{E}\left[\alpha_{t+1}(s_{t+1}|s_t)\right]\right\} \tag{3.3}$$

where

s_t is stochastic variables

Each state has several price nodes that indicates an outcome to occur and the price model is shown in figure 3.5.

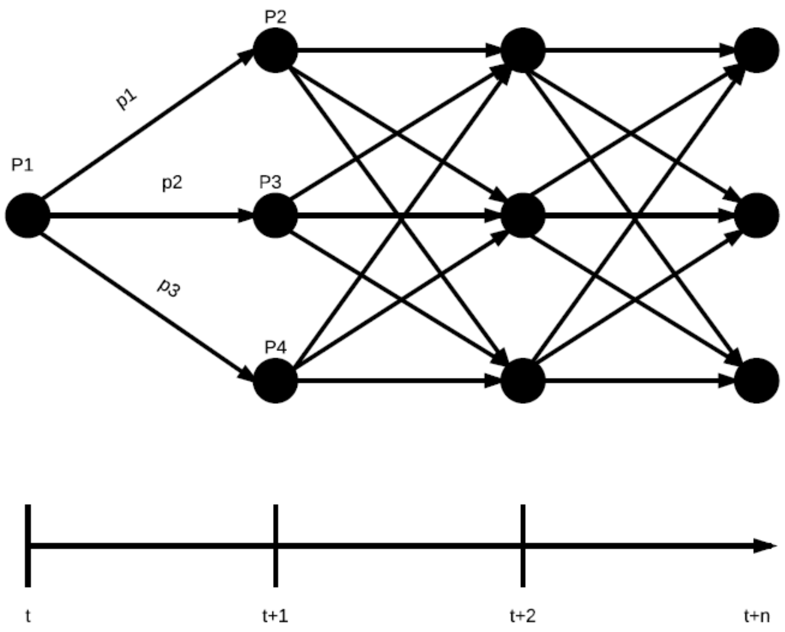


Figure 3.5: Stochastic price model

This gives value to the stochastic variables, in terms om the objective function. The probability from each node will yield 3.4

$$\begin{gathered}\sum_{n=1}^{N} \mathbb{P} = 1.0 \\ P_0 = \sum_{n=1}^{N} \mathbb{P} * P_n\end{gathered} \tag{3.4}$$

where

$\mathbb{P}$ is probability of p_n

N is number of nodes

P_n is price in node n

Thus, the future expected costs in equation 3.3, can be written as

$$\alpha_t(s_t) = \min_{x_t} \left\{ \mathbb{L}(x_t, s_t) + \sum_{s_t \in N} \mathbb{P}(s_{t+1}|s_t) \cdot \alpha_{t+1}(s_{t+1}) \right\} \tag{3.5}$$

3.4 SDP in hydro power scheduling problems

When assessing hydro power scheduling problems, the principals of stochastic dynamic programming allows the calculation of a multi-stage problem to be decomposed into single stages of sub-problems as explained previously. This approach is commonly used by participants in the Nordic market, and the outcome is used for simulations to find the optimum scheduling for maximizing of profits.

hydro power scheduling

In Norway, the biggest share of energy production comes from hydro power and naturally the need for maximizing the profits comes with the vast variation of producers. Since hydro power is a renewable energy source, it's dependant on the climate. Therefore, future expected profits will be based on hydrology and economics. If one were to combine these with an optimal way of running the power plant, one gets hydro power scheduling.

Hydro power scheduling can be divided into three main parts: *Long-term scheduling, Seasonal scheduling* and *Short-term scheduling*. Short-term scheduling is used for a more detailed deterministic approach. Seasonal scheduling is used when the changes in seasons are at stake. Long-term scheduling can be divided into two parts, strategy and simulation. In this report the main focus is the strategy part, where the SDP-method is plays a central role. The simulation part calculates the optimal way to schedule several areas of price. The seasonal scheduling uses a more detailed description of the model, where several scenarios are calculated. The short-term scheduling is a deterministic prosses, which uses the prices and inflow calculated in the other two scheduling terms.

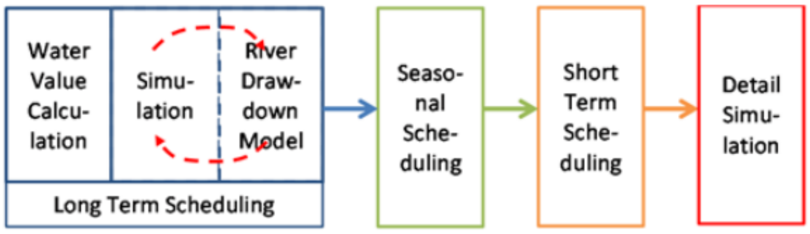


Figure 3.6: Coupling between scheduling and simulation[5]

Further explanation will be on the strategic part of the long-term scheduling, and the mathematics behind the calculation.

Aggregated model

In figure 3.7, a power system model is presented with an aggregated hydro power station. The reason that we aggregate the hydro power station is to simplify the calculations. Often, the hydro power plant has several reservoirs that needs to be accounted for, but the time of convergence will increase with additional reservoirs. Therefore, one must aggregate the model into one reservoir, and one power station.

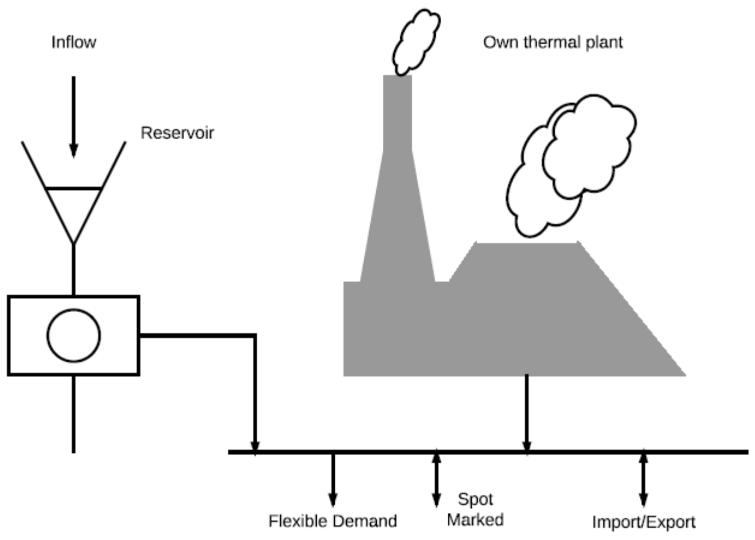


Figure 3.7: Power system model with aggregate hydro power representation

In figure 3.7, there is an additional power supply from a thermal plant. The thermal plant works as a supplementary power supply, that comes with a cost of producing. This cost needs to be minimized in order to maximize profit. The cost of producing power will then result in a marginal price for the water in the reservoir, something that is discussed in 3.4.

Water values

In the long-term scheduling part in figure 3.6, the first strategy is the calculation of water values. This is essentially the main part of this report. In this sub-chapter, the mathematical description behind the water values will be explained, and the calculation will be explained.

As explained in 3.3, one can use recursive calculation to find the optimal way. In this case the optimum is producing energy at the lowest cost. The state variable will be the reservoir level, v_n and the decision variable must be the production of hydro power, x_t. So the future expected operating cost, α_t will be a function of the value of the state variable and the decision variable. This figure represents the decision of the operating problem of minimizing the costs, and it can

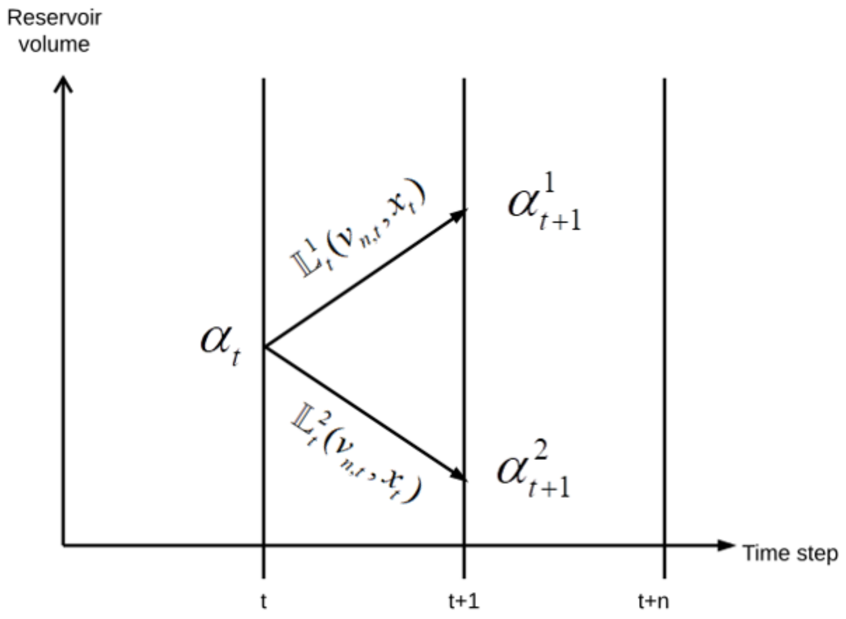


Figure 3.8: Visual representation of the decision problem of minimizing operation cost[5]

be described mathematically in 3.6

$$\alpha_t(v_{n,t}) = \min\left\{\sum_{t=1}^{T} \mathbb{L}(v_{n,t}, x_t) + \alpha_{t+1}(v_{n,t+1})\right\} \tag{3.6}$$

which can be described as

$$\alpha_t(v_{n,t}) = \min\left\{\sum_{j=1}^{J} \mathbb{C}_{jt}(p_{gth,jt}) + \alpha_{t+1}(v_{n,t+1})\right\} \tag{3.7}$$

by introducing thermal production p_{gth} in option *j*. Equation 3.7 shows that the operational cost is depended on the production of hydro power[1]. Furthermore, the constraints needs to be taken into account. There are two major constraints, which controls the production and the reservoir level, the *energy balance* and the *reservoir balance*. For the energy balance constraint, on must state that the produced energy is equal to the demand, neglecting losses on transmission or production.

$$\sum_{j=1}^{J} p_{gth,jt} + p_{gh,t} = D_{jt} \tag{3.8}$$

where $p_{gh,t}$ is the hydro power production for week *t* and D_{jt} is the demand in option *j*, week *t*. This states the direct correlation between the thermal- and hydro power plant in figure 3.7, and supports the fact that the cost of operation depends on the hydro production[1]. The reservoir balance states that the the water into the reservoir, must equal the water out of the reservoir.

$$v_{n+1} = v_n + I_t - p_{gh,t} \tag{3.9}$$

where

I_t is the weekly inflow of water in week *t*.

There are also other constraints that will be touched upon, but for this description we only need these two. Even though the inflow comes for free in a hydro power station, the water in the reservoir has a value. To calculate this value, we need to dig deeper into the field of operational research and introduce the *Lagrangian equation* and the *Lagrangian multiplier*.

$$\begin{aligned} L = &-\Big(\sum_{j=1}^{J_{th}} \mathbb{C}_{jt}(p_{gth,jt}) + \alpha_{t+1}(v_{n,t+1})\Big) \\ &+ \lambda_t\Big(\sum_{j=1}^{J_{th}} p_{gth,jt} + p_{gh,t} - D_{jt}\Big) \\ &+ \mu_t\Big(v_n + I_t - p_{gh,t} - v_{n+1}\Big) \end{aligned} \tag{3.10}$$

The Lagrangian multiplier(λ) represents the shadow-price of a constraint, which represents the cost of increasing the right-hand side of the constraint with one unit. We introduce the Karoush

[1]The more hydro power one produces, the less thermal power needs to be produced, and the operational cost will decrease.

Kuhn-Tucker (KKT) first order conditions by deriving the Lagrangian equation with the respect to the hydro power production and the reservoir in the end of the week.

$$\frac{\partial L}{\partial p_{gh,t}} = \lambda_t - \mu_t = 0 \tag{3.11}$$

$$\frac{\partial L}{\partial v_{n,t+1}} = -\frac{\partial \alpha_{t+1}}{\partial v_{n,t+1}} - \mu_t = 0 \tag{3.12}$$

Combine 3.11 and 3.12 we get an expression for the shadow-price λ_t:

$$\lambda_t = -\frac{\partial \alpha_{t+1}}{\partial v_{n,t+1}} \tag{3.13}$$

This is the (marginal) *water value*, because it is the cost of increasing the reservoir level by one unit. This states that the water value is a fraction of the total operation cost with respect to the reservoir level at the end of the week[5].

Stochastic variables

Inflow

Depending on your model, there are some variables that comes with an uncertainty, as discussed in 3.3. The most common variables are price and inflow. Some models use the water value calculations to find the market price since the marginal cost of operation in the optimum in week *t* equals the marginal cost of operation in week *t+1*. This is also equal to the water value[5]. This is a market model, where the price of operation is calculated, and the only stochastic variable is the uncertainty of inflow. For this way of calculating, the inflow will be seen upon as a scenario-tree like 3.5, where the use of backwards SDP is applicable. For every reservoir level, in every time steps there will be multiple scenarios for the inflow, and the iteration of calculating the water values will depend upon the probability for the inflow. Figure 3.9 visually shows the iteration process of stochastic inflow with seven inflow scenarios. The figure also shows the backward SDP with week 52 being time step *n*. The iteration process calculates the water value at the end of $n-52$, and replaces the first initial water values with the calculated values. This iteration continues until the difference between the initial and the calculated is within a convergence limit.

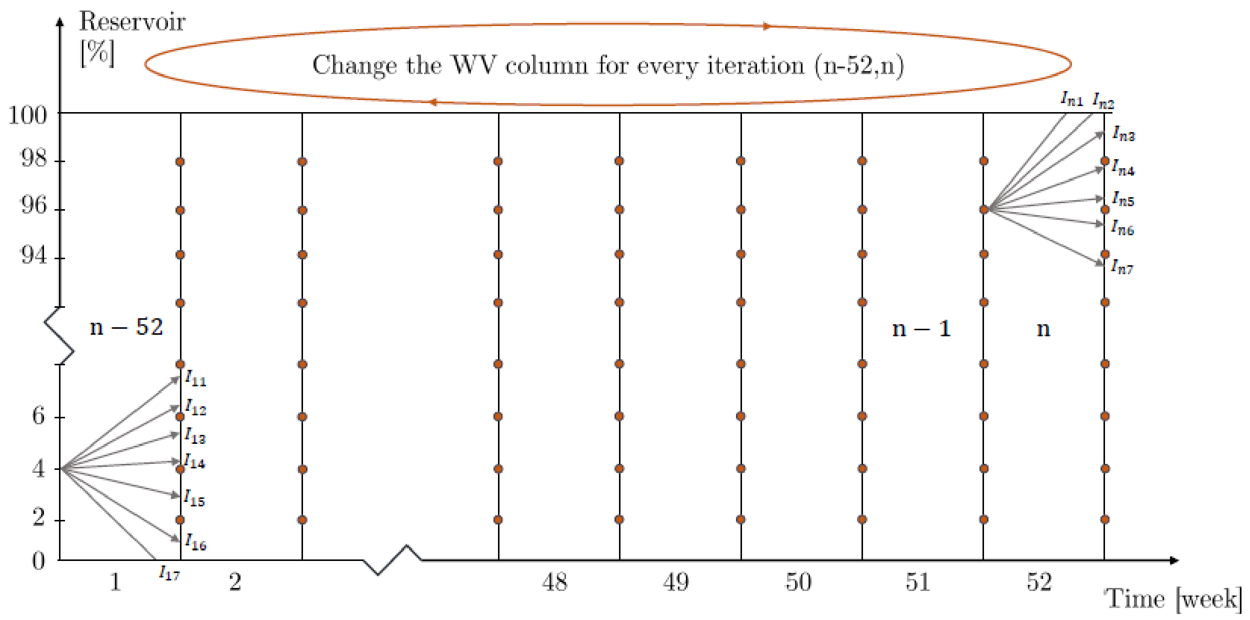


Figure 3.9: Iteration process for water value calculations [6]

In this case, the probability yields

$$\begin{gathered}\sum_{n=1}^{N} \mathbb{P} = 1.0 \\ I_n = \sum_{n=1}^{N} \mathbb{P} * I_{n+1}\end{gathered} \tag{3.14}$$

where

I_n is Inflow in node n.

Price

In section 3.3, price was given as an example of SDP, and for hydro power scheduling the calculations of water values can be calculated in a similar approach. For this approach, inflow is deterministic and the only stochastic variable is the price. In addition to that, we have a given quantity of energy to produce. The nodes on the scenario-tree, will each contain an expected future price of selling. By calculating each route, one can use the probability to calculate the expected water value of each node. The future water values, is then used to find out when and where to sell. Its optimal to sell when the water value is below the expected future price of selling. This is because the water value represents the marginal price of the water and in the Nordic market, the participants bid into the market at marginal prices. If the water value is beneath the market price then the income will be the quantity of production times the difference between the market price and the bidding price[17].

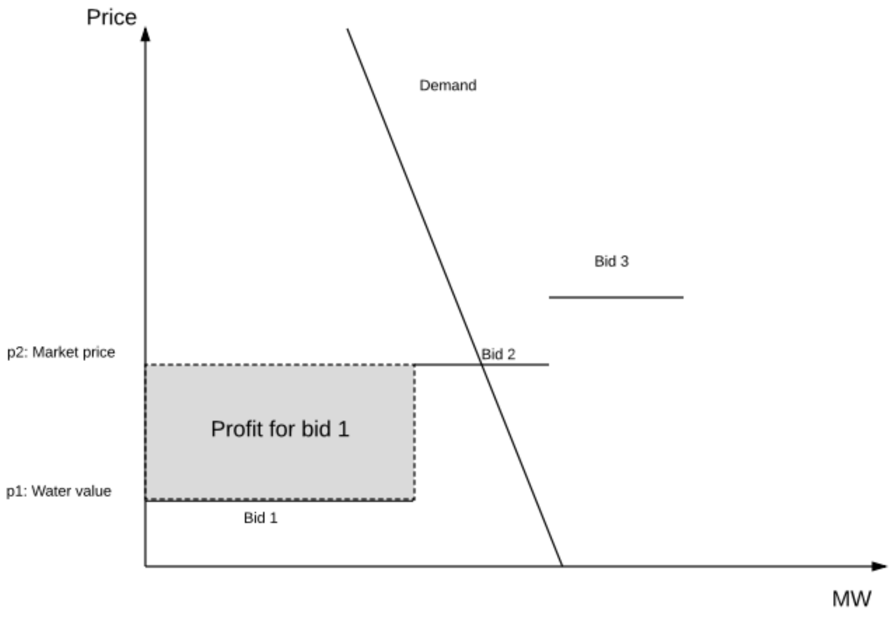


Figure 3.10: Market description of one area

Inflow and Price

As described previously, when dealing with inflow as the stochastic variable, the objective function minimizes the cost of operation. When price is the stochastic variable, the objective function maximizes the total revenue of the operation. If both inflow and price are at the same time stochastic variables, the objective function will also be to maximize future profit. The water value will then be the fraction of the profit, divided by reservoir volume. The visualization of the calculation will the become a three dimensional graph, with profit, reservoir level and time as the axes. For every calculation, both inflow and price will determine the future profit of the decision. In figure 3.11, the graph shows the combination between two stochastic variables, inflow and price. The total profit is then found by backward SDP for both variables.

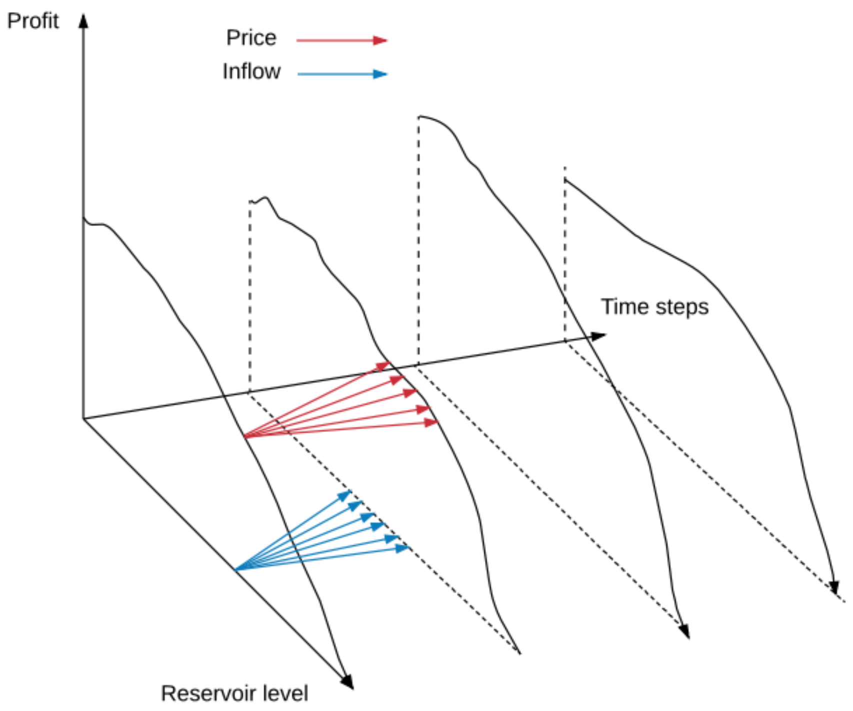


Figure 3.11: Stochastic descripion of water values

Usage of Markov chains in SDP

In similarity to the DP optimization technique, one must discretize the reservoir into several equal states, with a maximum and a minimum operating state (100% and 0%). Depending on the computation time and the accuracy, the amount of states describing the reservoir can be chosen freely. The time stages can be weeks or months, resulting in a year that is one cycle. As for the DP method, one must also calculate recursive, because of the usage of the future expected cost.

In SDP, we introduce a new state variable, that is the inflow to the reservoir. This means the SDP recursive operational formulation will consist of a two-dimensional sate variable with the storage volume and the inflow to the reservoir. With this in consideration one must now address the inflow characteristic as a variable of uncertainty. To describe the inflow, one can look at historic inflows for the modelled area, and find an inflow of four to seven inflow scenarios. These scenarios are usually made of a maximum and minimum inflow, and then discretized into equal size. Since the optimum is to minimize the future cost of operation, the recursive calculations start from the last stage(week), where there is no future cost. The objective function is described by the cost of operation at the state, and the future expected cost, including the inflow scenarios.

$$\alpha_t(s_t^p, s_t^e) = \max_{x_t} \left\{ J_t(x_t, s_t^p, s_t^e) + \sum_{s_{t+1}^e \in N_E}^{N} \mathbb{P}(s_{t+1}^e | s_t^e) \cdot \alpha_{t+1}(s_{t+1}^p, s_{t+1}^e) \right\} \qquad \forall t \tag{3.15}$$

This equation states that the future expected operational cost of one state (s) in stage (t) is the maximization of the profit of state (st), state transition (st+1) and inflow. One must also take into the consideration of the inflow transition probability and the future cost. This function uses the final storage as decision variable. The transition probability is a Markov chain, where the probability of a transition between inflow scenario j in stage (t+1) given the inflow in (t) is i. This will be discussed further in the summary.

The idea of the SDP recursive function is to produce the future expected cost function to calculate the incremental cost of hydro production. The objective is to minimize the immediate cost of operation in (t) and the total expected future cost of operation. Since the problem now contains of different inflow scenarios, one must calculate every sub-problem for the different stages. By doing this, one can interpolate against the cost function to find the future expected cost for each sub-problem, and then calculate the new state optimal cost of operation by multiplying the probability of the scenarios.

The transition probability needs to be generated as a Markov chain, that is, a matrix of probabilities where each probability represents the transition to another inflow scenario, given an inflow to the present week. It is natural to understand how the Markov chain operates, to fully understand how the SDP problem works. The stochastic part of the recursive function makes use of the probability of the scenarios. These transitions have their own probability matrix for each stage. They do not correlate between states in the same stage but between stages. The matrix will be made from inflow scenarios and the transition is the probability of an occurring inflow (j) in stage (t+1) given the inflow (i) in stage (t):

The probabilities do not move between rows, and the sum of one row is always 1.0. These inflow

$$\begin{bmatrix} p_{ij} & p_{ij} & p_{ij} & p_{ij} \\ p_{ij} & p_{ij} & p_{ij} & p_{ij} \\ p_{ij} & p_{ij} & p_{ij} & p_{ij} \\ p_{ij} & p_{ij} & p_{ij} & p_{ij} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \\ p_{41} & p_{42} & p_{43} & p_{44} \end{bmatrix}$$

Figure 3.12: Transition probability matrix

scenarios are computed with historical data, where one look to discretize the inflow in equal intervals. Then, by analysing the historical data, one can find out the probability of a streamflow occurring in stage (t) by the amount of occurrence the streamflow enters the given discretized interval. Then one can compute the transition probability by cross-checking the number of occurrences of going from one interval in stage (t) to another interval in stage (t+1).

As one can imagine, the historical data should be of a large number of years, because of the occurrence number being reliable. As one also can imagine, some of the inflow transition occurrences may never happen, and because of that, one will remain with a Markov chain that is containing some zero-elements. This is not preferable, as we want to move in a more ergodic matrix. One can call a matrix ergodic when all the members in a chain can form a single recurrent chain. In other words, the final state of an ergodic chain is independent of the initial state. This is important in the SDP, as it is one of the two main criteria to converge. The second criteria for convergence is the same as for the DP recursive algorithm; the incremental operation value function must be stationary or within the approved limit.

Chapter 4

Model discriptions

4.1 Deterministic model

One of the goals of this thesis is to create a model to calculate water values. In this chapter, a deterministic model, created by Martin N. Hjelmeland will be presented, as well as the algorithm in it. This model is created in Python and the optimization is calculated with Pyomo. A case study will then be presented, to analyze and verify the model.

Construction

Calculation process

The deterministic model is based on a deterministic input data series that is fixed for every stage. The deterministic model is defined by stage, state and decision variables. It also includes an objective function subjected to physical constraints and thereby iterated through a recursive equation. The optimum value is obtained by iterating over each state in each stage until the convergence is successful.

Discretization

The reservoir needs to be handled by discretization. The active storage in the reservoir is divided into a number of segments that represents the percentage of the total reservoir level. The upper and lower limits are often determined by the maximum operating capacity, or the highest volume regulated (HRV), and the lower boundary is often zero. These boundaries set the feasible limit of the optimal solution. If the operating point extend these limits it will lead to spillage or a dry reservoir.

Stage

In a recursive formulated problem, the time periods are described as stages. The number of stages is described by the way the system is divided into. A stage can be a week or a month, respectively 52 or 12 stages in a cycle.

State and decision variables

In a dynamic program, it is essential to establish the variables that describes the state and the decision made at every stage. The reservoir volume describes the state of the system. Since the system is discretized there will be different states of the system in every stage. The decision to be made every stage is how much hydro power to produce, hence the release through the turbine. Since this is a linear dynamic program there will be only one state variable and one decision variable. This is because of the deterministic input data which states the inflow, price and demand at every stage. In a stochastic dynamic program, there are two state variables, that is reservoir level and inflow/price. This is because of the need for a Markovian nature of the inflow, which are incorporated by describing it as a state variable.

State transformation equation

The state of the system is described above, and that is the available reservoir volume. For every optimal linear problem (LP) solution there is a state transformation. This transformation depends on the inflow, hydro production, spillage and the current state. The state equation describes the state at the start of next stage. The state at the start of the next stage is used to calculate the future expected costs, by interpolation. In this deterministic model, the state transformation equation is based on the continuity of the water resources in the reservoir.

$$V_{t+1} = V_t + I_t - R_t - Sp_t \qquad \forall t \tag{4.1}$$

Where V_t is initial volume in reservoir at the beginning of stage *t*, I_t is the inflow to the reservoir through the whole stage *t*, R_t is the release from the reservoir during stage *t*, Sp_t is the spillage from reservoir during stage *t* and *t* is the stages within the cycle.

Physical constraints

In this optimization problem, the handling of the reservoir comes with physical constraints. There are also other physical constrains that will be mentioned. The reservoir is handled by describing the physical limitations of the reservoir. This contains the volume of the reservoir which has a maximum capacity and a minimum capacity.

$$V_{min,t} < V_t < V_{max,t} \qquad \forall t \tag{4.2}$$

In this model, the reservoir volume [V] is described as the amount of energy [Mm^3] it contains. Further more the reservoir is has a upper limit witch describes the amount of water the reservoir could have before spillage. The hydro power station is limited by the production capacity which has an upper limit and a lower limit. This also applies to the thermal power plant which also has an upper and lower limit.

$$Ph_{min,t} < Ph_t < Ph_{max,t} \qquad \forall t \tag{4.3}$$

$$Pt_{min,t} < Pt_t < Pt_{max,t} \qquad \forall t \tag{4.4}$$

where *Ph* is hydro production and *Pt* is thermal production.

The system is also described by physical constraints. The amount of volume left in the reservoir needs to be in balance with the volume that comes into the reservoir. This balance is known as the reservoir balance. Previously the state transformation equation was described. The state transformation is govern by the reservoir balance. The other balance that needs to be upheld is the energy balance. The energy balance states that the energy that is demanded, needs to be produced. In this case, the energy balance is

$$Ph_t + Pt_t + Pll_t - D_t = 0.0 \qquad \forall t \tag{4.5}$$

where *Ph* is hydro power production drawn from the reservoir, *Pt* is subsided thermal power production and *D* is for demand of end users at a given stage *t*.

Objective function

One can operate a reservoir for many reasons such as flood control, hydro power generation and water supply. Its important that the objective function clearly describes the reason for the operation. For this case study, the reservoir is handled by hydro power generation, which is most common in Nordic reservoirs. There are a few ways to operate the reservoir when dealing with hydro power generation. One can maximize the hydro power generation, or maximize profit of power sales, or as for this case; maximizing the negative values of the cost of thermal production.

For maximizing the cost of production the objective function will be

$$\max \mathbb{E}\Big\{\sum_{t=1}^{N} -C_t \cdot P_t\Big\} \qquad \forall t \tag{4.6}$$

where $\mathbb{E}$ is the expected cost of operation, and the N is the number of total stages in a year. C_t is the cost of production and P_t is the amount of thermal production. t stands for stage in a year. This is the foundation of the model, where the maximization of the operational cost is used to find the future expected profit. The objective function is expanded further, including spillage, lost load and tail-water. The expanded weekly decision problem becomes:

$$\max \mathbb{E}\Big\{\sum_{t=1}^{N} -C_t \cdot P_t - Twa_t - Spi_t - Pll_t\Big\} \qquad \forall t \tag{4.7}$$

where Twa is tail water in between the reservoir and the generator, Spi is spillage of water and Pll is volume of lost load. The model is modeled as a maximization problem, because it makes it simpler to expand the model into a profit maximization problem when adding reserve capacity.

The objective function is the same, but with opposite signs. Penalties are introduced to make sure the that the hydro power is utilized at the fullest capacity before turning to the thermal plant and lost load. It also makes sure of not loosing water when the reservoir is full. The extended objective function is then:

$$\max \mathbb{E}\Big\{\sum_{t=1}^{N} -C_t \cdot P_t - Twa_t * Twa_{pen} - Spi_t * Spi_{pen} - Pll_t * Pll_{pen}\Big\} \qquad \forall t \tag{4.8}$$

Recursive equation

For the deterministic model, the recursive equation describes the calculation of the future expected cost. A recursive equation can take many different forms because of the characteristics of decision variables, state variables and objective function. The decision variables can either to release water from the reservoir or reach a final storage. For state variables, the inflow is either present inflow or previous inflow and the objective is whether to maximize or minimize. For this deterministic model, the recursive equation is:

$$\alpha_t(V_n, I_t) = \max_{x_t} \Big\{ J_t(V_t, I_t, x_t,) + \mathbb{E}\big[\alpha_{t+1}(V_{t+1}, I_{t+1})\big] \Big\} \qquad \forall t \tag{4.9}$$

st.

$$R_t = V_t + I_t - V_{t+1} - Sp_t \qquad \forall t \tag{4.10}$$

where R_t, V_t, I_t, V_{t+1} and Sp_t have been have been defined in equation 4.1. $J_t(V_t, I_t, x_t)$ is the operation cost of decision x_t, inflow I_t and initial reservoir volume V_t. $\alpha_{t+1}(V_{t+1}, I_{t+1})$ is the future expected cost of operation. Solving the weekly decision problem by recursive calculation makes sense because one have to maximize the total future expected profit of operation. The recursive flowchart is shown in figure 4.1. The recursive algorithm is used in all the other models. This is the foundation of the optimization of every weekly decision problems. Solving this is done by looping over all stages and states in the problem and storing the values as every optimum is found.

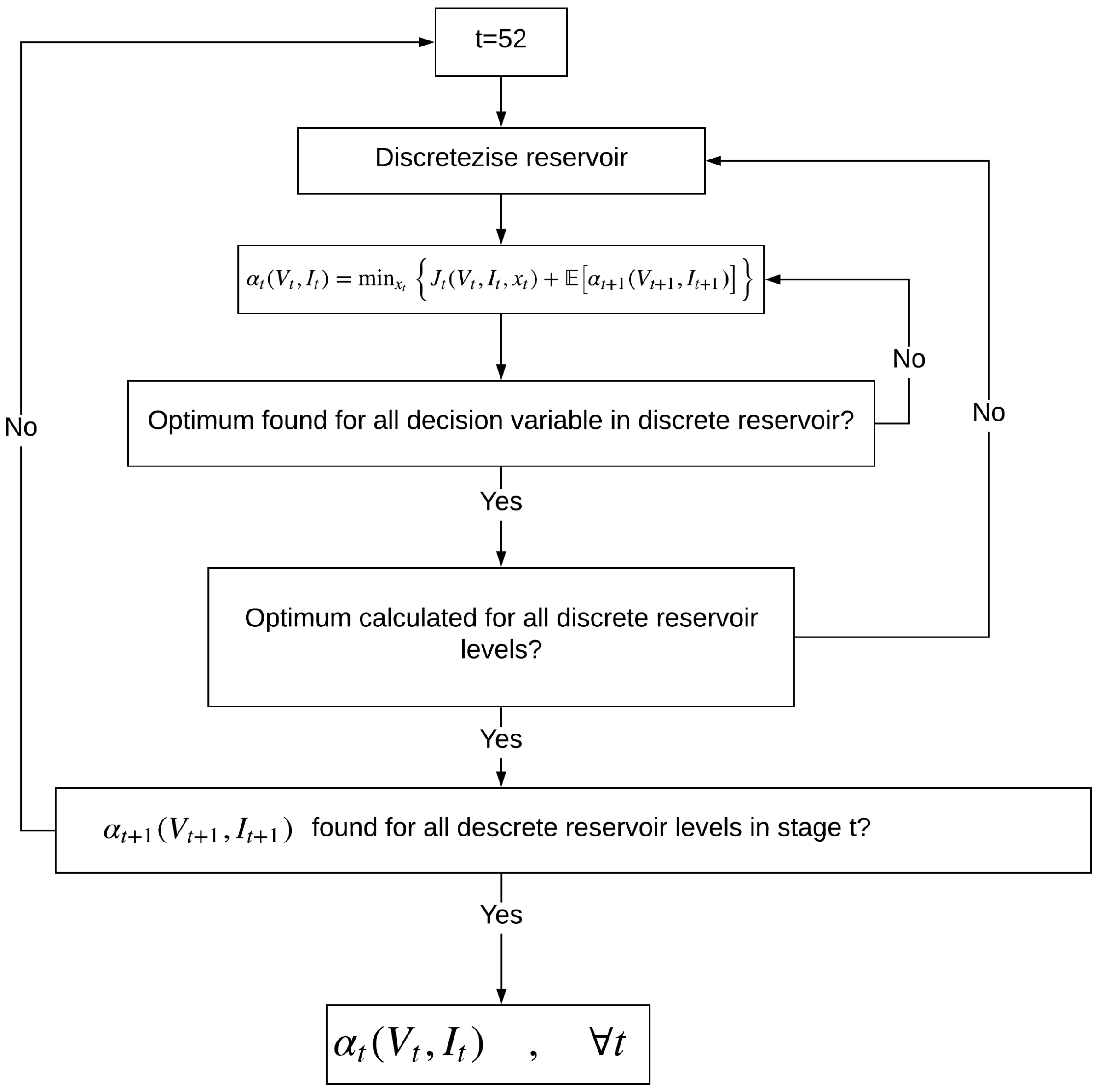

t=52
Discretezise reservoir
$\alpha_t(V_t, I_t) = \min_{x_t} \left\{ J_t(V_t, I_t, x_t) + \mathbb{E}\left[\alpha_{t+1}(V_{t+1}, I_{t+1})\right] \right\}$
Optimum found for all decision variable in discrete reservoir?
No
Yes
Optimum calculated for all discrete reservoir levels?
No
Yes
$\alpha_{t+1}(V_{t+1}, I_{t+1})$ found for all descrete reservoir levels in stage t?
No
Yes
$\alpha_t(V_t, I_t) \quad , \quad \forall t$

Figure 4.1: Flowchart for recursive calculation

Water Value calculation and iteration process

A deterministic model is govern by the fact that all input data is fixed. Therefore, the calculation time is significantly low. A deterministic model is often used for short-term scheduling, to find the optimum production within a week or month. The low computation time means that the model can include detailed information about the systems.

In the deterministic model used in this thesis, there are no detailed description about the system. This is because the goal of the usage is different. In this thesis, the focus is on long-term scheduling, with a planning horizon of one year. When dealing with long-term scheduling, the main goal lies in handling the production or the reservoir. Detailed description of the system will extend the computation time and therefore one must aggregate the model into one reservoir before computing. By aggregation, one can investigate the behavior of the reservoir and the water values, which is the goal of long-term scheduling.

The calculation of water values is a process that takes part at the end an iteration process. This iteration process is an algorithm that includes four loops to calculate the long-term planning horizon. In this section, the iteration process will be described for the deterministic model, to the point where every optimal $\alpha_t(V_n, I_t)$ has been found for every discrete reservoir level in the planning horizon. It is important to understand the behavior of the iteration process to understand the water value calculation and how the water values correlate with the dynamic programming problem.

Iteration process

The deterministic model is based on a non stochastic iteration process. This means that the iteration process is based on four loops. The first loop is the main loop for the iteration. This is the point where one states the range of the iteration. For this model, the maximum iterations that the model can iterate is 20. To find the computation time, one must start the timing at this loop, and print the difference at the end of the loop. The python code of the iteration process will be explained below for every loop until the optimum is found.

```
for it in range(ItMax):
    print("Iteration ,␣", it+1)
    t1=time.time()
```

This is the start of the first loop. The first line initializes the start of the iteration in the range of the maximum iteration limit. *it* is the index of the iterations, *ItMax* is the maximum limit of the iteration. The third line sets the start of the timing.

The second loop is for the stages. This is where you tell the model that the calculations cover every stage that is in the model. In this model there are 52 stages, every stage represents a week. As well as telling the model to calculate every stage, one must initialize the data that is to be used in every stage. For this deterministic model, the data that variate between stages is the inflow, the price and the demand. This loop also contains another loop that tells the model to update the water values and the offset values.

```
for iStage in reversed(range(sys_data["NStages"])):
    instance.Inflow=sys_data["Inflow"][iStage]
    instance.Demand=sys_data["Demand"][iStage]
    instance.Price=sys_data["Price"][iStage]

    if iStage==sys_data["NStages"]-1:
        for iState in range(len(R)-1):
            instance.WV[iState]=WVTableEnd[iState]
            instance.DWV=DValEnd
    else:
        for iState in range(len(R)-1):
            instance.WV[iState]=WVTable[iState][iStage+1]
            instance.DWV=DValTable[iStage+1]
```

The first line initializes the range of the stages, and the reversed function makes the calculation start at stage 52 and calculates recursive. This utilizes the usage of the expected future profit of the scheduling. The three next lines initializes the data from the dictionary and creates instances to be used in the dynamic programming problem. Furthermore, we come to the update of water values. These lines tells the model to use the end values if the calculation is happening in the first stage, and if not, update the next stage until every stage is calculated. *iState* is the index for the discrete reservoir levels at every stage. The loop tells the model that it shall update for every state in the stage.

In the first calculation, the water value table is a two-dimension array of zeros. The dimensions are constructed with the length of the stages on the horizontal axis and the discrete reservoir levels on the vertical axis. This array is made from the numpy package and is created as follows:

```
WVTable=np.zeros((len(R)-1,len(T)))
WVTablePrev=np.zeros((len(R)-1,len(T)))
WVTableEnd=sys_data["WV"]
DValTable=np.zeros((len(T)))
DValEnd=0.0
```

Where R and T is discrete reservoir levels and stages. The dimensions of the arrays are important to make the model work. In addition to the dimension, it is important to index the arrays with the same structure as the dimensions of the array. These dimension will evolve as the stochasticity is included, which is described in section 4.2.

The fourth loop is for the discrete reservoir levels. This is the last and final loop for the deterministic model and it contains some important parts. At this point, the model calculates for all stages and for all discrete reservoir levels, so in the final loop one must solve the optimization problem. After solving the optimum for every discrete reservoir level in every stage one must store the values as the future expected profit for water value calculations.

```
for iR in range(len(R)):
    instance.InitVol=R[iR]
    opt.solve(instance,  tee=False,keepfiles=False)
    Alfa=instance.Obj()
```

Where *iR* is index for discrete reservoir levels, and *InitVol* is the initial reservoir volume at one discrete reservoir level. This instance corresponds to V_t in the reservoir balance in equation 4.1. The third line is the command to solve the optimal as an instance value for further use. As previously explained the the optimal $\alpha_t(V_n, I_t)$ is the future expected profit. In the fourth line one store these values for every reservoir level in every stage for further water value calculations.

Water value calculation

So far, the model description is based on calculating every weekly decision problem. These values are now stored and ready to be used in the water value calculations. The first calculation happens when the reservoir is zero at the last stage. This is where one create an offset value for the interception of the y-axis, as a start value for the calculation. The rest of the calculations build on the previous expected profit that is updated for every stage.

```
if iR==0:
    DValTable[iStage]=Alfa
else:
    WVTable[iR-1,iStage]=(Alfa-AlfaPrev)/(ResEnd[iR]-ResEnd[iR-1])
AlfaPrev=Alfa
```

In the first line, the offset value is stored as the first value if the discrete reservoir level is zero. *DVALTable* is the notation of the offset value. If the model is not in the first discrete reservoir level it calculates the water values and store them as a two dimensional array. This array is constructed by indexing the stored values with *[iR-1, iStage]*. The equation for calculating the water values makes sense because it is the slope of the linear function value between discrete reservoir levels.

Convergence and updating water value table

After computing the water values the model needs to check the convergence. It is important to know which loop the convergence check is implemented. This needs to be at the end of the iteration loop, because then you have calculated for all stages and discrete reservoir levels. The difference between the water values from the previous iteration and the current iteration will determine the convergence difference. If the convergence difference is lesser than the convergence limit the iteration stops and the optimum is found. If the difference is more than the convergence limit, the water value table needs to be updated, by replacing the current table with the previous. Note that its just the water value table that needs replacement. The offset value and the last stage of the water value table needs to be replaced. This is common when calculating water values because the iteration process needs to continue with the last values that was calculated in the previous iteration.

```
Cdiff = np.sum(np.absolute(WVTablePrev-WVTable))
 if Cdiff<PolicyEps:
      break
 WVTablePrev=np.copy(WVTable)
 WVTableEnd=np.copy(WVTable[:,0])
 DValEnd=DValTable[0]
```

This is the end of the iteration process for the deterministic model. As line 2 describes if the model converges the iteration process breaks and the water values have been found. The Python code presented in this chapter are compressed and includes only the parts that governs the iteration process. All other visualization commands are excluded. Below is the pseudo code for the DP-problem.

Algorithm 1 DP algorithm

```
1: $it \leftarrow 0, \Delta \leftarrow \infty, WVtable^j \leftarrow 0$
2: while $\Delta > Cdiff$ do
3:     for $t = NStages - 1...0$ do
4:         for $n = 1...NResSates$ do
5:             $WVTable \leftarrow WVTable^j(i, I_{t+1}, t+1), i = 1..NResStates - 1$
6:             $\alpha_t(V_{n+1}, I_{t+1}) \leftarrow Optimize$
7:         end for
8:         if $n > 1$ then
9:             $WVTable^j(V_{n-1}, I_{t+1}, t) \leftarrow \frac{\alpha_t(V_n, I_t) - \alpha_t(V_{n-1}, I_t)}{V_n - V_{n-1}}$
10:        end if
11:    end for
12:    $\Delta \leftarrow \sum_{t=0}^{T} \sum_{n=1}^{NRes} |WVTable^j(V_n, I_t, t) - WVTable^{j-1}(V_n, I_t, t)|$
13:    $WVTable^{j+1}(V_n, I_t, T) \leftarrow WVTable^j(V_n, I_t, 0), \forall n$
14: end while
```

4.2 Extension to Stochastic model

To extend a deterministic model to a stochastic model, implies that the state variables operates with a probability described by a Markovian environment. This extension leads to a number of sub-optimal problems in each state, and therefore an increase in computation time. In this section, the description of the extension will take place, where the focus will be on the addition to the constructed sets, variables and parameters in the deterministic model. Stochastic dynamic programming is the most common calculation process in the strategic phase of hydro power scheduling. This paper do not touch upon stochastic dual dynamic programming, which is also common in the strategic phase. Note that the extension does not only affect the algorithm but also the input data, which will be touched upon in the case studies.

Construction

There are some parts of the deterministic model that remains the same. First of all the discretization of the reservoir remains the same. This makes sense because we are only working with one aggregated reservoir. Since this problem remains a long-term scheduling problem, the model continues with 52 stages, each representing a week in a year. The state transformation equation remains the same, but note that there are different inflow scenarios that will impact each sub-optimal calculation. There are no addition to the constraints since the extension revolves around the price and the inflow.

Stochastic variables

In the stochastic model, the inflow and the price variables are turned into a set of stochastic variables. These sets of stochastic variables are a part of the state variables. In this model, each reservoir level has 15 price nodes and 5 inflow nodes. One can look at them as different scenarios. For example: what is the optimal production in reservoir level 3 if price scenario 4 and inflow scenario 3 occurs. The model has to treat every discrete reservoir states the same way. This is why the computation time increases when you add stochasticity to the model. These unique scenarios are called sub-optimal problems.

Each price node and inflow nodes have a transition probability. This transition probability is govern by a Markovian matrix for each stage. It makes sense using a Markovian matrix, since the Markovian property allows the conditional probability distribution of future states to only depend on the current state [2]. By creating a Markovian nature to the model, the weekly decision problem changes. The future expected cost are now bounded by the probability of the occurrence of the different scenarios. The weekly decision problem will then be

$$\alpha_t(s_t^p, s_t^e) = \max_{x_t} \Big\{ J_t(x_t, s_t^p, s_t^e) + \mathbb{E}\big[\alpha_{t+1}(s_{t+1}^p, s_{t+1^e}) | s_t^e\big] \Big\} \qquad \forall t \tag{4.11}$$

where S_t is a set of system states that comprises all information passed from one decision stage *t-1* to the next *t*. A subset $S_{P,t} \subset S_t$ of these state variables are endogenous to the optimization problem which means that they are created from within the model. The stochastic variables is then defined as the state variables $S_{E,t} \subset S_t$[2]. Adding stochastic variables to a model, implies a probability, in this case Markovian, that is used to express the expected future value function. One needs to find the weighted average of the future expected value function by multiplying the probability over the discrete reservoir volumes and the discrete values $S_{E,t+1}$. The weekly decision problem is then

$$\alpha_t(s_t^p, s_t^e) = \max_{x_t}\left\{J_t(x_t, s_t^p, s_t^e) + \sum_{s_{t+1}^e \in N_E}^{N} \mathbb{P}(s_{t+1}^e | s_t^e) \cdot \alpha_{t+1}(s_{t+1}^p, s_{t+1}^e)\right\} \qquad \forall t \tag{4.12}$$

where $\mathbb{P}$ is the transition probability matrix. N_E is the number of nodes in the model. The Markovian nature of the probability matrix allows one to assume that the stochastic variables are auto- and cross-correlated between the stages.

Water value calculation and iteration process

For a stochastic dynamic model, the water values are calculated the same way as for the deterministic model, except the future expected value function is not relying on the deterministic inflow and price. The stochastic model, treats the water value with a weighted probability average that contains transition probabilities from both inflow and price. The weighted future value is obtain from the sum of every probability scenarios in every discrete reservoir level. In this section, the implementation of the stochastic part of the model will be described, as well as the changes in the iteration algorithm. As previously described, the stochastic variables now include an endogenous state variable, which is the initial reservoir volume at the end of a stage.

Iteration process

When one is expanding a deterministic model to a stochastic model, the iteration process changes. This is because the prices and inflow now have several sub-problems. To obtain the future expected value, one must now calculate all the sub-optimal scenarios for every discrete reservoir in every stage of the iteration. What were 520 calculations , is now 39000 calculations.

There are some changes to the original four loops in the deterministic model. First of all, one needs to move the iteration over the discrete reservoir states from being the fourth loop to the second. The reason for this is because of the transition probability and the price and inflow nodes. Note that with deterministic values in the input, the values are fixed with one value for every stage. In the stochastic model, there are 15 price values and 5 inflow values for every stage. And one needs to make sure that they are looped over for the price and inflow arrays, but also the transition probability matrix.

```
    for iStage in reversed(range(sys_data["NStages"])):
            instance.Demand=sys_data["Demand"][iStage]
            for iR in range(len(R)):
                instance.InitVol=R[iR]
```

As one can see, by comparing this to the deterministic model, the deterministic price and inflow that before were indexed by the stages, are now gone. In this model, lists and arrays have been used as input to the model for price and inflow. The second loop over the discrete reservoir levels is something that will be discussed further when one must calculate the weighted probability average. The first loop to start the iteration process is the same as for the deterministic model.

After looping over the discrete reservoir levels the stochastic price and inflow needs to be looped for. One creates an index for price and inflow that is in the range of the total amount of nodes. In the data input, one must also create a list of the inflow and the price. One can also use a dictionary to stare and input data, but for this model, lists is used. Note that the instances that is created is stochastic variables that goes into the weekly decision optimization problem. The purpose for the loops are to calculate all possible sub-problems in each discrete reservoir level at every stage. Note that the loop starts in the *iR* loop, but because of the width of the paper this loop starts for the left.

```
for pto in range(len(P)):
    instance.Eprice=sys_data["EpriceState"][iStage][pto]
    instance.Cprice=sys_data["CpriceState"][iStage][pto]
    for ito in range(len(I)):
        instance.Inflow=sys_data["InflowState"][iStage][ito]
```

In the first loop one creates the index *pto* to represent the price for going to stage *t* from stage to stage *t-1*. The same goes for index *ito*. *Eprice* is the energy price of selling hydro power to the grid, while *Cprice* is the price of selling reserve capacity.

Furthermore the algorithm must update the water values. Note that since the introduction of the stochastic variables the dimension of the water value table has changed. The water value table is now indexed by the discrete reservoir level, stage, price nodes and inflow nodes. This means that the initialization of the water value tables including offset table and end-table has the additional price node and inflow node dimension. One updates the water value table the same way as in the deterministic model.

As for the deterministic model, the optimal values are calculated in the same loop as the inflow loop. This makes sense because the iteration process is calculating the optimum for every inflow node, and every price node in every discrete reservoir level in every stage. The optimal values are stored in a array to be used further in the water value calculation. The array contains two dimensions; price and inflow.

Water value calculation

After storing the optimum values for every price node and every inflow node, one can calculate the water values. The difference between the deterministic model and the stochastic model is that one must generate two more loops in order to calculate the weighted optimum values. These loops represent coming form a price and inflow node, to another node. This is because the transition probability matrix has a Markovian nature, and one assumes that they cross correlate between weeks. The weighted objective values are generated by multiplying of the transition probabilities in price and inflow and multiplying them with the objective values. Note that the start of this loop is under the discrete reservoir loop. This makes sense because the water values are the marginal cost of increasing the production of hydro power.

```
for pfrom in range(len(P)):
    for ifrom in range(len(I)):
        Alfafrom = 0
        for pto in range(len(P)):
            for ito in range(len(I)):

                Alfafrom = Alfafrom +

                    ((TrProbPrice[iStage][pfrom][pto])*

                    (TrProbInflow[iStage][ifrom][ito])*

                    (Alfa[pto][ito]))
        if iR==0:
            DValTable[pfrom][ifrom][iStage]=Alfafrom
        else:
            WVTable[iR-1][pfrom][ifrom][iStage]=
             ((Alfafrom-AlfaPrev[pfrom][ifrom])/(ResEnd[iR]-ResEnd[iR-1]))
        AlfaPrev[pfrom][ifrom]=Alfafrom
```

Where *pfrom* is the index of coming from a price node, and *ifrom* is the index of coming from an inflow node. *Alfafrom* is the weighted objective value and is initialized as zero before looping over price and inflow again. One must update the weighted objective value after every calculation in order to calculate the water values which happens in the last line. The expression for calculating the water values are the same as before except the new dimensions.

It is important to understand the iteration process in the stochastic model to understand how one arrives at the water values. The common parts of the deterministic model and the stochastic model is the way they calculate the water values and that they loop over stage and discrete reservoir levels. But the extension to the stochastic part impacts the algorithm in a way that it calculates all possible scenarios. These optimal values are then used further in another loop to find the weighted optimal value. This process is the stochastic part of the model and it tells us something about the water values and how its computed. On the one hand it tells us that the water values are govern by the probability of going from one state of inflow and price to another. But on the other hand, it tells us that the transition probabilities need to be as accurate as possible to get the best outcome of the scheduling.

To get a complete overview of the algorithm, a pseudo code based on [2] is given below.

Algorithm 2 SDP algorithm

```
1:  $it \leftarrow 0, \Delta \leftarrow \infty, WVtable^j \leftarrow 0$
2:  while $\Delta > Cdiff$ do
3:      for $t = NStages - 1...0$ do
4:          for $n = 1...NResSates$ do
5:              for $s^e_{t+1} = 1..N_E$ do
6:                  $\{Eprice_{t+1}, Cprice_{t+1}, I_{t+1}\} \leftarrow s^e_{t+1}$
7:                  $WVTable \leftarrow WVTable^j(i, s^e_{t+1}, t{+}1), i = 1..NResStates - 1$
8:                  $\alpha_t(V_{n+1}, s^e_{t+1}) \leftarrow Optimize$
9:              end for
10:             for $s^e_t = 1...N_E$ do
11:                 $\alpha_t(V_n, s^e_t) \leftarrow \sum_{s_{t+1}=1^e}^{N_E} \mathbb{P}(s^e_{t+1}|s^e_t) \cdot \alpha_{t+1}(V_n, s^e_{t+1})$
12:                 if $n > 1$ then
13:                     $WVTable^j(V_{n-1}, I_{t+1}, t) \leftarrow \frac{\alpha_t(V_n, I_t) - \alpha_t(V_{n-1}, I_t)}{V_n - V_{n-1}}$
14:                 end if
15:             end for
16:         end for
17:     end for
18:     $\Delta \leftarrow \sum_{t=0}^{T} \sum_{n=1}^{NRes} |WVTable^j(V_n, I_t, t) - WVTable^{j-1}(V_n, I_t, t)|$
19:     $WVTable^{j+1}(V_n, I_t, T) \leftarrow WVTable^j(V_n, I_t, 0), \forall n$
20: end while
```

4.3 Extension to Multi-Area model

The stochastic model is operating in one area with prices and inflow governing that area. In this master thesis, the objective is to make a fundamental market model, including several bidding zones. In this section, the extension from one area, to two and three areas will be described. A multi-area model takes into the account that the interconnected line(s) can manage purchase and sales to other area(s). Furthermore, the losses on the lines will be taken into account to study how the model operates under losses.

In this model, it is more important to study the behaviour of the interconnected areas than the water values. The water values are of course the output of this model, but the study of the behaviour tells us more about the operation than the water values. By adding losses to the interconnected lines, there will be an increase in hydro or thermal production to cover the losses, which will impact the water values.

Two-area model

The extension to a two area is a process that impacts the objective function, constraints and the iteration process. It is important to have a clear knowledge of how to optimize one area, before adding another. The challenging part is how to model the interconnected lines between the two areas. In this section, the extension to two areas will be described, and also the process of constructing it. It will include a description of how the knowledge about one area optimization helped to construct two area model.

Construction

Adding another area means that the global optimization problem is now trying to optimize both areas simultaneously. To make it easier to understand, one must look at one area and generalize it for every new areas. In the simple matter where there are only two areas, it means that there are only one line. In this case one looks at the line with no losses, to simplify the problem. It can be difficult to optimize two areas if one looks at the operational problem, considering both areas simultaneously. Therefor one optimizes one area at the time, and create an additional constraint for the line. The one area model is visualized below.

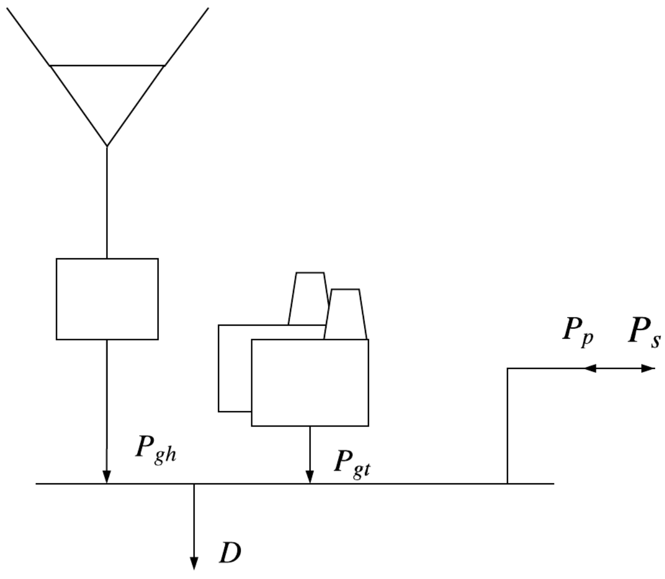


Figure 4.2: One Area optimization problem with exchange

Since both areas are connected, the one area optimization problem is equal for both areas. This leads to the assumption that all energy being sold in area one, is being purchased in area two. This assumption simplifies the construction of the model and leads to the additional constraint.

$$Ptr_j + Ptr_{j+1} = 0 \tag{4.13}$$

Where *Ptr* is the energy being transmitted in the transmission line and *i* represents the area. This constraint makes sense when the transmission line is loss free. The transmission line is in this case positive for export and negative for import. The variable *Ptr* is bound to the physical constraint below.

$$-Ptr^{max} < Ptr < Ptr^{max} \tag{4.14}$$

The variable *Ptr* needs to be included in both energy balance and objective function. Since the transmission variable can variate between positive and negative values for export and import, the energy balance constraint from equation 4.5 becomes:

$$Ph_t + Pt_t + Pll_t - Ptr_t - D_t = 0.0 \qquad \forall t \tag{4.15}$$

where *Ph*, *Pt*, *Pll*, *D* and *t* are the same as in equation 4.5.

The objective function changes when adding the idea of purchase and selling to another area. The price for purchasing or selling works as a penalty function. Since the transmission line variable is positive for export and negative for import, it makes sense that the objective function from equation 4.8 now becomes as below.

$$\max \mathbb{E}\left\{\sum_{t=1}^{N} -C_t \cdot P_t - Twa_t * Twa_{pen} - Spi_t * Spi_{pen} - Pll_t * Pll_{pen} + Ptr_t * Ptr_{pen}\right\} \qquad \forall t \quad (4.16)$$

This makes sense because the sale of energy will increase the profit of the objective function.

Iteration process

The extension to another area does not affect the iteration algorithm as much as the stochastic extension. For another area one needs an additional loop that loops over the areas. Since one would like to optimize for all areas and all stages in the areas the loop needs to be placed before the looping over the stages. Then we need to index the demand, price, inflow and water value tables with the areas. The same goes for the weighted objective value.

```
for it in range(ItMax):
    t1=time.time()
    for area in range(len(A)):
        for iStage in reversed(range(sys_data["NStages"])):
            instance.Demand=sys_data["Demand"][area][iStage]
```

Three-area model

The model contains at this point two areas, each with a hydro power station and a thermal power plant connected with no losses on the line. In this section the extension to a third area will be described, and losses on the lines will also be included. In this model, one assumes that all the three areas are connected together. The purpose of this extension is to create a more complex system and study the behaviour of the import, export and water values. With a three-area model and losses on the line, there are some new variables and constraints that needs to be described. By adding another area and two more lines, the computation time will increase. The tree-area system is visualized below.

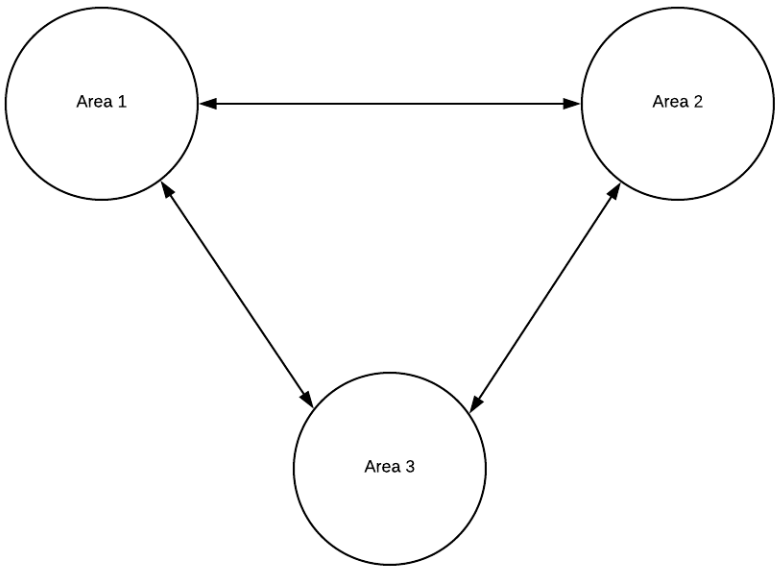


Figure 4.3: Three-area optimization problem with exchange

Construction

To simplify this extension, one must look at one area in the beginning. In this special case there are two interconnected lines going from each area. By creating the constraints to represent an incident matrix, one can control the flow on the lines. This has been done in the construction of the three-area model, and will be described in the following section. The construction also includes losses, and this will be touch upon as well.

Each area can be described with one hydro power plant, one thermal power plant, fixed demand and the opportunity to sell and purchase energy from the other two areas. Below, is a figure describing each area in the three-area model.

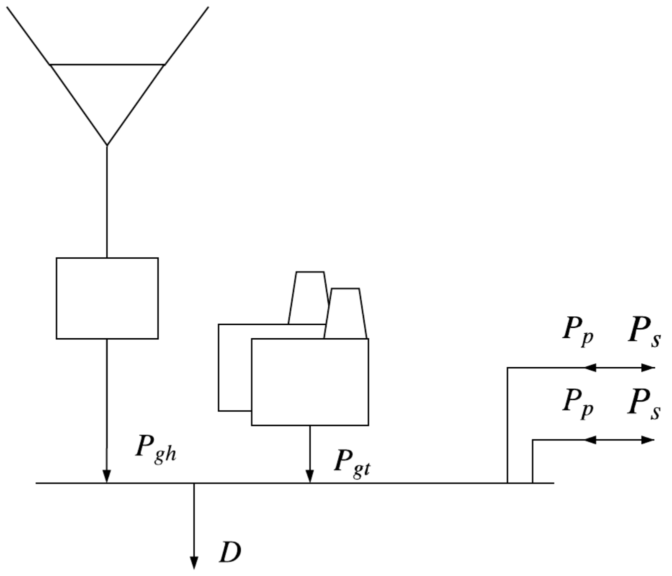


Figure 4.4: Single area optimization problem with exchange between two areas

In this model the constraints are different, since it contains losses and there are two additional lines to model. As mentioned before, in this case there is no incident matrix, but a hard coded constraint on the line. The purpose for it is to use the potential of Pyomo to the fullest.

To start of the construction of the line, one have to imagine the transmission line to be two imaginary lines; one for import and one for export. The extension from one line to two imaginary lines helps us with the modelling of the losses. One introduces these two lines as variables with bounds.

$$-Pex_{i,j}^{max} < Pex_{i,j} < Pex_{i,j}^{max} \tag{4.17}$$

$$-Pim_{i,j}^{max} < Pim_{i,j} < Pim_{i,j}^{max} \tag{4.18}$$

Where $Pex_{i,j}$ is the imaginary line of exporting energy, $Pim_{i,j}$ is the imaginary line of importing energy, i is the area that is being optimized, j is the area connected to the area.

The constraint on the line can be created by introducing the losses as μ.

$$Pim_{i,j} - (1-\mu)Pex_{i,j} = 0 \tag{4.19}$$

This constraint ensures that the exported value needs to be accountant for producing the losses on the line. In the model, the areas *i,j* is hard coded to ensure that the import and export are floating to the right areas. In addition to this constraint, there are some extent to the energy balance in the model. One needs to make sure that the sum of the export and import are included in the energy balance. It is important to understand that constraint for the line only affects the exported value, and that the imported value needs to be accounted for in the energy balance. The extended energy balance is given below.

$$Ph_t + Pt_t + Pll_t - Ptr_t + \sum_{j=1}^{2}(1-\mu)\cdot Pim_{i,j} - \sum_{j=1}^{2} Pex_{i,j} - D_t = 0.0 \qquad \forall t \tag{4.20}$$

The objective function also changes to keep the profit as a benefit or a cost in the objective. To make this a cost variable on can simply introduce another constraint that states that the transmission cost is the sum of the variable cost in export or import.

$$ctr_j - \sum_{i=1}^{2}\phi_{i,j}\cdot(Pim_{i,j} - Pex_{i,j}) = 0 \tag{4.21}$$

Where ctr_j is the cost of transmission and $\phi_{i,j}$ is the variable cost of transmission. By introducing this constraint the objective function is then described as below.

$$\max\mathbb{E}\Big\{\sum_{t=1}^{N} -C_t\cdot P_t - Twa_t * Twa_{pen} - Spi_t * Spi_{pen} - Pll_t * Pll_{pen} - \sum_{j=1}^{2} ctr_j\Big\} \qquad \forall t \tag{4.22}$$

The water values are calculated as the same way as for two areas. This extension is the the final addition to the market model, and covers the task of making a fundamental market model. Further more the input data to the model will be described in the next chapter, and the testing of the model will also be described.

Chapter 5

Case Studies

The motivation behind these case studies is to make a fundamental market model to study the possibilities of decommission thermal production and look at the performance of the different models and analyze the results. Another factor to the studies, is to participate in the reserve market and sell reserve capacity. There is also a need for checking different models and to analyze the potential revenue, and the behaviour of the market models.

The work that will be analyzed is made out of prototypes of an optimization model. Since these models are only prototypes, the creation of the models have some assumptions and simplifications that will impact the results, but these will be mentioned and discussed.

These case studies are only imaginary cases, and do not represent any realistic hydro power stations. These case studies are used to describe the system and how the model is set up. But since the input needs to be created of real catchments, the price and inflow of Mauldal hydro power station in Rogaland have been used. The other parameters, such as reservoir volume, and generators does not represent Mauldal.

5.1 Hydro system

The hydro system is the same for every area and model. It has a reservoir that can run to spillage. there is only one generator and in addition to the hydro power station, there is a thermal power plant connected to the grid. The reservoir volume is in every case 50 Mm^3. Figure 5.1 shows in detail the hydro system that is in the model.

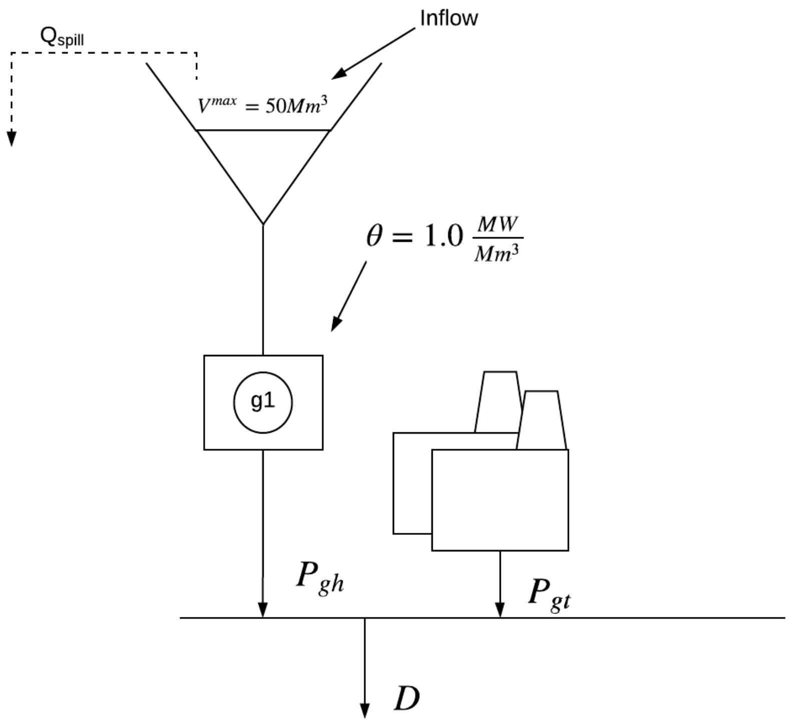


Figure 5.1: Hydro system for all areas and models

Minimum river flow

Since these models are only prototypes and constructed to simulate a market model, some simplifications and assumptions have been made. The first assumption is that one excludes the minimum river flow. This is mainly because the focus is on the market model and detailed information can make the computation time even bigger. On the one hand, it makes sense having minimum river flow, because it creates a variable that constrains the maximum discharge. On the other hand, the minimum river flow is rarely too big to impact the production in a large scale. The claim of a minimum river flow will impact the model at a low reservoir volume, since the bypass will be needed and there could be to little water left in the reservoir to produce hydro power.

Generator efficiency

In this hydro system there are only one generator to produce hydro power and just one generator producing thermal power. This is of course a simplification to reduce computation time. Normally the generators works with a PQ-efficiency curve, witch is a curve that represents the power output (P) for any discharge (Q). This simplification avoids a piece-wise lionization of the PQ-efficiency curve. One could have made use of an SOS-2 variable to find the linear point between to generator states. If this where to be added in the model, it would have included a binary state of the generator. The same goes for the thermal generator. It would have made the computation time much larger since there would be two generators with to different PQ-curves.

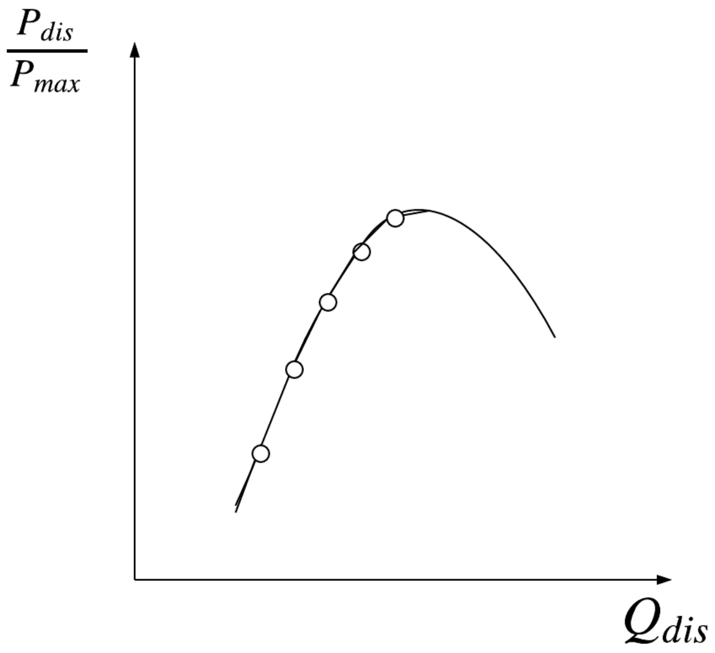


Figure 5.2: PQ-efficiency curve example

Magazine volume and discretization

As seen in figure 5.1, the maximum volume of the reservoir is set to 50 Mm^3. The reason for it is because the discrete reservoir levels are set to 10, which means each segment is 5 Mm^3. The discrete reservoir levels were set to 10, because of the computation time. Since these models are just a prototype, the computation time is a matter in trying to verify the model. Too big computation time would have made it difficult to check for mistakes, if the model was dysfunctional. The reservoir segments can be increased if the maximum volume of the reservoir increases. This will of course play a part on the water values, as it is calculating the value of the water left in the reservoir. The discrete reservoir levels also has something to do with the accuracy of the calculations. On the one hand, if one have too few discrete reservoir levels the calculation accuracy of the water values will decrease, but on the other hand if there is too many discrete reservoir levels the computation time will increase. Therefore it is necessary to fin a balance between accuracy and computation time.

Cost of thermal power production

The cost of thermal production will in this case have a impact on the water values. Since we look to decommission the thermal production in the areas, the price of thermal production mus reflect the cheapness of producing thermal power, but also reflect the CO_2-taxes as well. The price of selling in the DAM lies between 65-290 Nok/MWh in the seasonal change and with 15 price nodes. Therefore the cost of thermal energy production is set to a constant 200 NOK/MWh.

Energy conversion factor

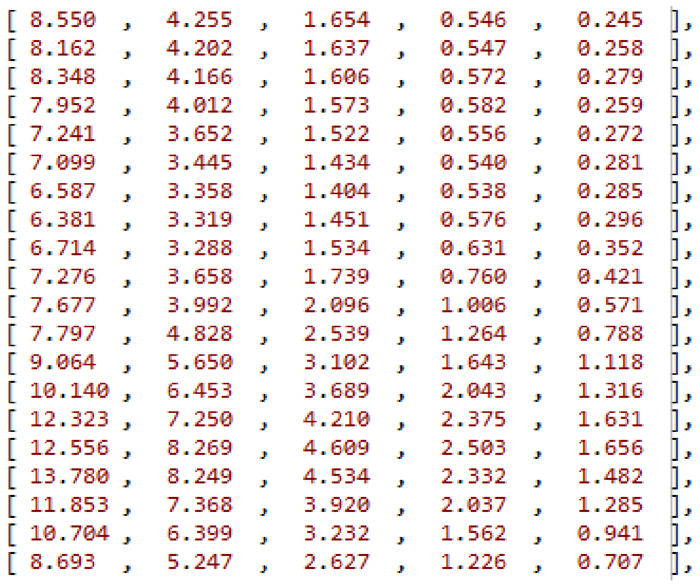
As seen in figure 5.1, the energy conversion factor is set to 1.0 to simplify the problem. This can be changed in the model, but for this purpose it stays at 1.0. This means that the discharged water that is produced generates one MWh per Mm^3 of water. Increasing the energy conversion factor will lead to more production of electricity pr Mm^3 water and the opposite happens if one decrease it.

5.2 Input data

Since the cases variate between different models, the input data will be different according to which model they represent. A deterministic model has fixed values for each stage, but the stochastic data has different nodes of data. The multi-area model has different values for each area, that needs to be accessed in the right way. In this section, the representation of the input data will be presented in its origin, and the reason for how its set up will be explained. The data is retrieved as inspiration from [2] and clustered by the author.

Inflow data

The inflow is originally from 1000 inflow samples generated from a VAR(1) model based on historical inflow data. These samples were then used as a baseline to an appropriate number of used in this model, which was set to 5 inflow nodes per stage as shown in Figure 5.3. This ensures that the instances of unusually high or low inflow is taken into consideration in the model[16]. For the deterministic model, the data input is different, because it is only one inflow per stage. The inflow nodes are converted into lists. For the one-area stochastic model this list dimensions contains stage and inflow node. For the multi-area model, the dimensions of the list contains of area, stage and inflow node.



```
[ 8.550  ,  4.255  ,  1.654  ,  0.546  ,  0.245  ],
[ 8.162  ,  4.202  ,  1.637  ,  0.547  ,  0.258  ],
[ 8.348  ,  4.166  ,  1.606  ,  0.572  ,  0.279  ],
[ 7.952  ,  4.012  ,  1.573  ,  0.582  ,  0.259  ],
[ 7.241  ,  3.652  ,  1.522  ,  0.556  ,  0.272  ],
[ 7.099  ,  3.445  ,  1.434  ,  0.540  ,  0.281  ],
[ 6.587  ,  3.358  ,  1.404  ,  0.538  ,  0.285  ],
[ 6.381  ,  3.319  ,  1.451  ,  0.576  ,  0.296  ],
[ 6.714  ,  3.288  ,  1.534  ,  0.631  ,  0.352  ],
[ 7.276  ,  3.658  ,  1.739  ,  0.760  ,  0.421  ],
[ 7.677  ,  3.992  ,  2.096  ,  1.006  ,  0.571  ],
[ 7.797  ,  4.828  ,  2.539  ,  1.264  ,  0.788  ],
[ 9.064  ,  5.650  ,  3.102  ,  1.643  ,  1.118  ],
[ 10.140 ,  6.453  ,  3.689  ,  2.043  ,  1.316  ],
[ 12.323 ,  7.250  ,  4.210  ,  2.375  ,  1.631  ],
[ 12.556 ,  8.269  ,  4.609  ,  2.503  ,  1.656  ],
[ 13.780 ,  8.249  ,  4.534  ,  2.332  ,  1.482  ],
[ 11.853 ,  7.368  ,  3.920  ,  2.037  ,  1.285  ],
[ 10.704 ,  6.399  ,  3.232  ,  1.562  ,  0.941  ],
[ 8.693  ,  5.247  ,  2.627  ,  1.226  ,  0.707  ],
```

Figure 5.3: Screenshot of inflow node list

Price data

The price nodes in the model originate from historical data from 2013-2015 for NO2, the price area in Norway in which Maudal is located. These data were used in an ARIMA(2,0,3) time series model, which samples them into a wished number of price nodes. The appropriate number of price nodes ended at 15 in total per stage. These 15 nodes consist of 5 different energy and 3 different reserve capacity prices. The reserve price nodes are also originally taken from the same historical data, but sampled through GARCH(1,1) and ARMA(1,1)[16]. Also in this case are the data made into lists. The price nodes variate between areas for a more realistic approach.

Markov

The transition probabilities for the price and the inflow nodes are given with a Markov chain. Markov chains are explained in section 3.4 and figure 5.4 shows the set up for the Markovian transition probabilities. The sum of every column should equal 1.0. Every stage has a transition probability matrix and they are all made into lists where the dimensions are stage, node and node for the stochastic model. For the multi-area model the dimensions are area, stage, node and node.

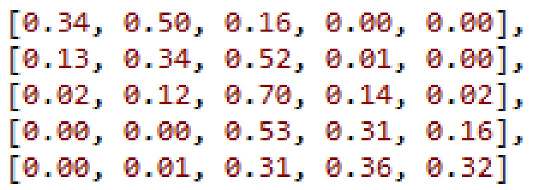

```
[0.34, 0.50, 0.16, 0.00, 0.00],
[0.13, 0.34, 0.52, 0.01, 0.00],
[0.02, 0.12, 0.70, 0.14, 0.02],
[0.00, 0.00, 0.53, 0.31, 0.16],
[0.00, 0.01, 0.31, 0.36, 0.32]
```

Figure 5.4: Markovian matrix for one stage

Chapter 6

Testing of Models

To get a better understanding of how the behaviour of the models, it is necessary to test the models. By testing the model one can analyze the results and check for faults or improvements. This chapter will contain the results of the testing of four models; the deterministic, the stochastic one area, the two-area stochastic model without losses and at last the three-area stochastic model with losses on the interconnected lines. One would like to analyze the iteration process, that is, if it converges, how much time does the model use and how many calculated iterations before convergence. The water values will also be presented from each model, and compared at the end of the chapter.

It is important for the reader to understand that these models are prototypes, and also consist of many simplifications and assumptions. These simplifications and assumptions will of course impact the result of the model, but the main focus in this chapter is to get a better understanding of the market model, and how the stochasticity characterizes the water values. At the same time one aims to find an understanding for how multi-area market models work, and how the transmission losses impacts the scheduling problem.

At the end of each test, there will be a comment on the output of the water values, and some explanation of the behavior of the models. The main discussion takes place in chapter 8.

6.1 Testing of Deterministic Model

The deterministic model is described in section 4.1, and consist of fixed values in the input data for price, inflow and demand for every stage. The water values that is produced in this model, will be of two dimensions; state and stage. There are 520 discrete states to optimize, and therefor the convergence time is low for this model.

The goal of this test is to analyze that the DP-problem is working correctly. This model was made by Martin N. Hjelmeland. At first there was a problem with the model,because the water values was calculated towards the price of lost load. The problem was that the demand was too high for the hydro- and thermal power plants to supply, and therefor it resulted in VOLL. After lowering the demand, the model converged.The water values that the deterministic model produced, was exported to Excel and visualized as a graph.

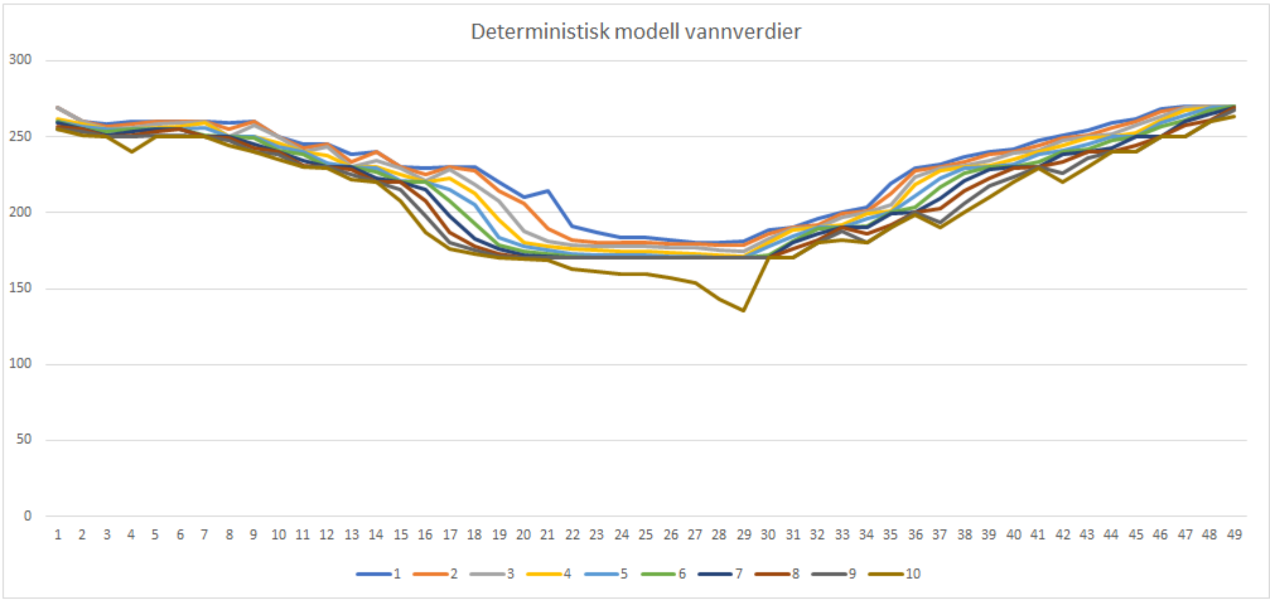


Figure 6.1: Water values for deterministic model

The identification below indicate each discrete reservoir level. As one can see from the graph, the water values are higher for lower reservoir volume and low for higher reservoir volume, which indicates that the DP- optimization is working as it should.

6.2 Testing of Stochastic Model

The Stochastic dynamic model is an extension to the deterministic model made by the author. It is explained in section 4.2 and consists of 15 price nodes and 5 inflow nodes. The demand is still fixed since it is a fixed load. In this model, the water values are calculated as a four dimension array; stage, state, price and inflow. This results in 39000 discrete states to optimize, which leads to a much higher solution time. For this model the solution time lies between 1300 and 1400 seconds pr iteration. It converged in typically 4-5 iteration, depending on the input data.

Since this model is the base model for the extension to multiple areas, the main goal was to make sure it converged and that it produced water values that was reasonable for the data input. The model did produce wrong water values at the beginning, because the update of the weighted alpha was incorrect. After fixing the problem it produced water values that made sense.

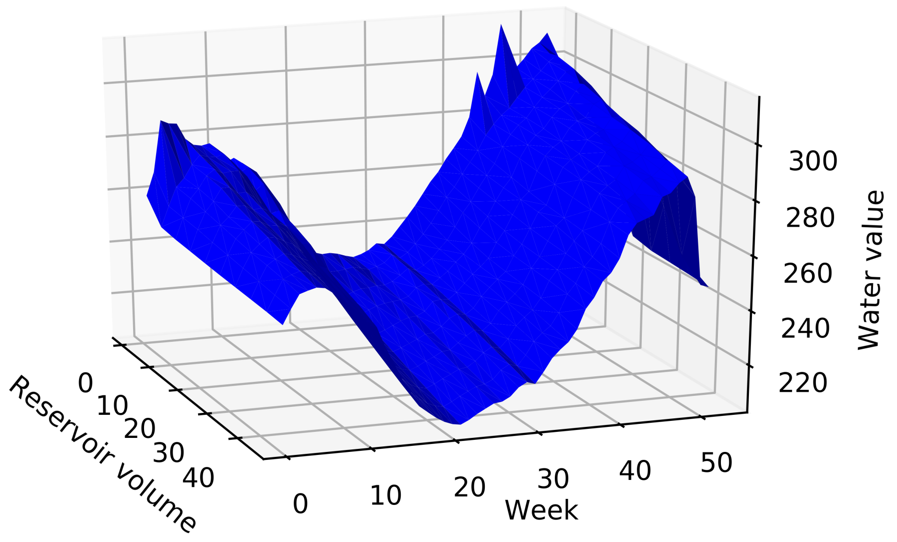


Figure 6.2: Water values for price node 1 and inflow node 1

6.3 Testing of Two-Area Model Without Losses

The first extension to the multi-area model is two areas with no losses on the lines. This model is explained in section 4.3. After testing and verifying that the stochastic model works, the two are model would be able to produce the first results for the market model. Since there is no losses on the lines, the behavior of the two areas would not be affected in a large scale. Remember that selling or buying from another area is purely beneficial to the optimal solution. Buying from another area means that the price at that area is cheaper than running the thermal power plant, and selling is for gaining more profit, if the value of the water in the reservoir is lower than the price of selling.

The results of the two-area model is given below. The expansion from one are to another affects the computation time as the model is now calculation 78000 discrete states. The reservoir volume is equal in both areas but the demand is not. This is because the author wants to force the other area to sell energy that is left in the reservoir, to analyze the behavior of the market model.

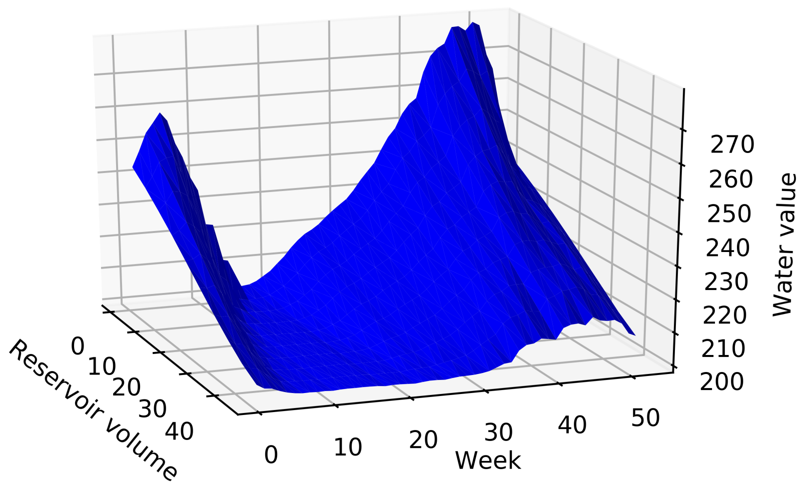


Figure 6.3: Water values for price node 1, inflow node 1 and area 2

6.4 Testing of Three-Area Model With Losses

The last test is for the three-area model with losses on the line. This model is explained in section 4.3 and is the final model for this master thesis. When testing this model, one are looking for divergence from the two-area model to see the impact of the losses on the lines. The function of the losses makes the hydro power plant generate more power to cover the loss, which will impact the value of the water left in the reservoir.

There was some problems making the losses on the line at first, because the losses do not impact the variable for the line. The solution to this problem was to create two imaginary lines and as well make sure that the losses were covered in the energy balance. This model calculates 117000 discrete reservoir states. Therefore, the computation time is highest in this model. The figure below is the water value output for the model.

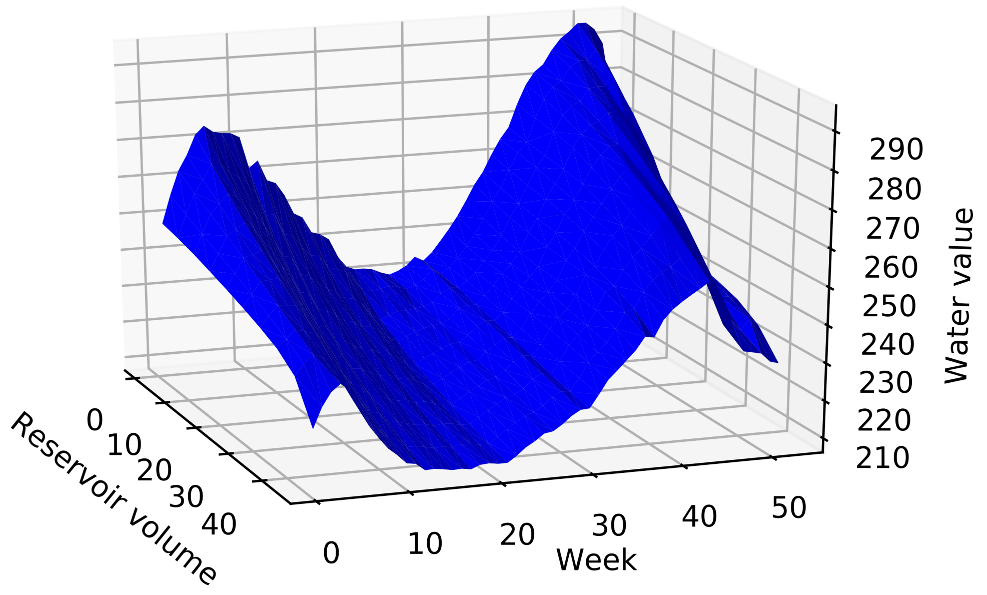


Figure 6.4: Water values for price node 1, inflow node 1 and area 2

Chapter 7

Sensitivity Analysis: Including Wind Energy

In this chapter, there will be conducted a sensitivity analysis on the impact of including wind energy into the model. This analysis will include a theoretical aspect on how the author included wind to the model, and the results. Since the goal of this thesis is to make a fundamental market model, the addition wind generation is important. Throughout Europe, decommission of thermal energy is an its march with an increase in solar and wind production. Therefore, the author finds it necessary to look into the aspect of including wind in the model.

7.1 Model extension

The wind energy variable needs to have some physical constraints to the model. In the scientific language, it is called "cut in" when the there is enough wind to produce energy, and "cut out" when the maximum limit of production is reached. These symbolizes the minimum and maximum production limit. The constraints on the wind energy variable is given below.

$$Pw^{min} < Pw < Pw^{max} \tag{7.1}$$

Where Pw^{min} is the minimum production of wind energy, Pw is the wind energy variable and Pw^{max} is the maximum production limit.

The wind energy does not in this case contribute as a profit to the objective function, because it is meant to be utilized to cover the demand in each stage. In similarity to the inflow, the wind is a renewable resource that can be assumed "cost free". Unlike inflow, the wind resource cannot be stored for production, so it makes sense to utilize the wind energy when it is enough wind to produce electricity. Since wind energy is covering the demand, it needs to be included in the energy balance.

$$Ph_t + Pt_t + Pw + Pll_t - Ptr_t + \sum_{j=1}^{2}(1-\mu)\cdot Pim_{i,j} - \sum_{j=1}^{2} Pex_{i,j} - D_t = 0.0 \qquad \forall t \tag{7.2}$$

The one area system including wind production is shown in figure 7.1 below.

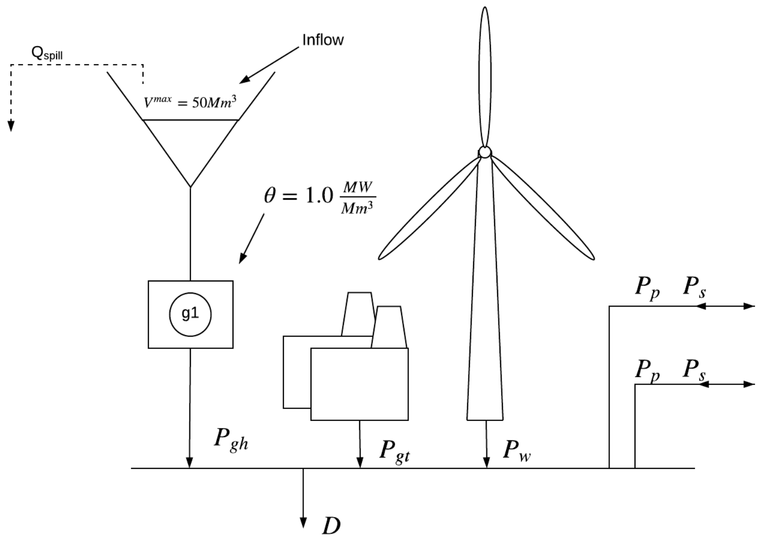


Figure 7.1: Area system including wind

7.2 Results

Since wind energy is included in the model, this should give an effect on the water values. This is because one utilizes the wind energy when it is produced, lowering the need for hydro power production. Therefore the decision of producing or storing water in the reservoir changes. This should give a negative effect on the water values. With the expectation of lesser water values, the result of the model is given below.

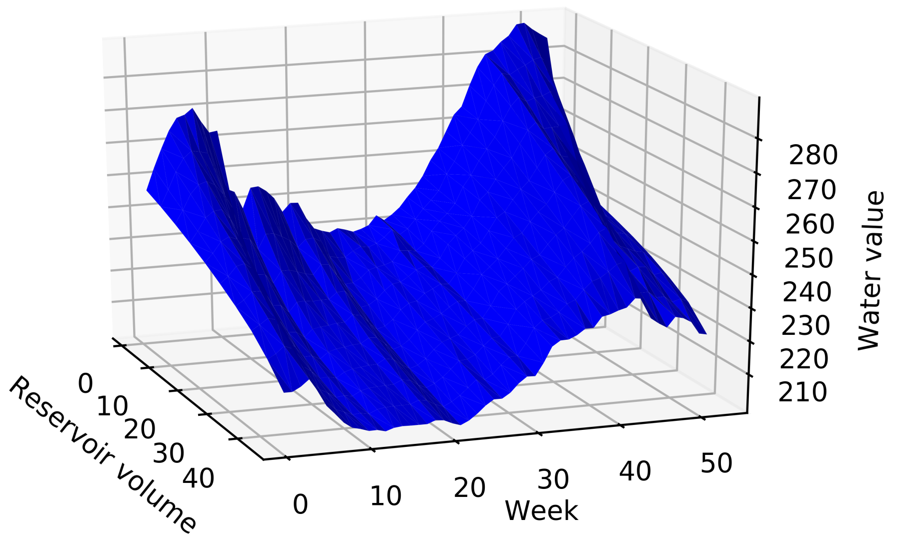


Figure 7.2: Water values for price node 1, inflow node 1 area 2 with wind

From figure 7.2, one can see the water values is true to the expectation. If one compares the water values to the water values in figure 6.4, one can clearly see that the peak value is lowered by 10 NOK/MWh. Another point to be made is that the water values is less smooth, which indicates the variation in the production of wind. This confirms the expectations that the additional wind energy, contributes to the decommission of thermal on fossil energy. This analyze needs to include stochastic variables for a more complete analysis, but one can clearly see the effect of wind energy with fixed values. The time series is given in section B.2

Chapter 8

Comparing Models

In this thesis, the main focus is on creating a fundamental market model, to conduct simulations. Until now, the models have been constructed and tested. There have been made assumptions and simplifications, which will impact the results. It is important to analyze these results to verify the models, or discuss their behavior. Since these models are prototypes, they are not complete, which means that the results of the models can be improved. Furthermore the difference in each model are of interest, because they represent different ways of calculating water values.

In this chapter, the discussion of the models will take place. It will contain a comparison between different model, to analyze their behavior and the difference in the water values produced. There are a few aspects that plays a part in this discussion, which is the slopes of the water values, the solution time, the accuracy of the calculations and the input data of each model compared. For the multi-area models, the big difference is clearly the use of losses, which also will be discussed in this chapter.

The comparison that will be discussed in this chapter is :

- Deterministic model and stochastic one-area model
- Two-area neglecting losses and three-area with losses

8.1 Deterministic model and stochastic model

Since the deterministic model and the stochastic model produces different types of dimensions of the water value table, they are not that similar when you look at them for the first time. The water values from the deterministic model are of two dimensions with 10 lines, each representing different reservoir states. While the stochastic model results show a three dimension graph where the reservoir states are represented in the x-axis. There are some similarities to the results, which will be discussed further

Comparing water values

For the deterministic water values shown in figure 6.1, the water value slope are from a scientific aspect correct. They appear to be high in the cold winter days, and lower in the summer periods. This is of course a product of the fixed demand, but the importance of the table is the difference between the reservoir states. As one can clearly see, the difference of the water values at a low reservoir state is much higher that with a full. There are a difference of up to 50 NOK/MWh, which indicates the cost of producing more thermal energy than hydro energy. Comparing this to the stochastic water values in figure 6.2, one can see the effects of having different nodes for the price and inflow. For the stochastic model, the difference is not that high, which indicates that the thermal usage is lesser that for the deterministic model. Another difference is the variations in the water values. Even though the deterministic water values look less steeper than the stochastic water values, they variate with over 100 NOK/MWh. The stochastic water values also variate with more than 100 NOK/MWh, but the slopes at the beginning and end of the week are different. This difference can be explained by the opportunity to store water for the next stages when the demand is higher.

For both models, there are some water values that sticks out from the rest. In the deterministic water values, there are a dump in the values for the full reservoir in week 29. This can be explained by the fact that the inflow is too high to produce and some of the water runs to spillage. Spillage is something that one aims to prevent, and as one can see in figure 6.1, it has a n impact on the water values. For the stochastic water values there are two higher water values that sticks out. One can clearly see that the values are at the lower reservoir states which could indicate that the water left in the reservoir are running low. Having less water in the reservoir will increase the water values.

Solution time and accuracy

For these two models, there are an extensive difference in solution time. The different lies in the calculation process. Since the deterministic model have fixed values for every stage it only takes 520 calculations pr iteration. This gives a solution time of 30 to 40 seconds each iteration. The solution time are of importance when for scheduling a hydro power station. Keep in mind that the deterministic approach is mostly used in short-term scheduling problems, where the hydro system is programmed in detail, which will of course give a higher solution time, but for this case it does not. Comparing this to the stochastic model, it has 38 times as high calculation time as the deterministic model. In other words it takes up to 1400 seconds, or 23.3 minutes per iteration. Taking into account the simplifications and the assumptions, this makes sense. If the model were more detailed with start up costs, more generators, bypass and PQ-curves the solution time would have been much higher.

As discussed above, the details of the system are of importance. To compare these two models, one looks to the solution time, and the accuracy of the result. For a deterministic model, where the input are fixed, there are less accuracy in the calculations. Therefore it is important to analyze what are the risks of calculating the water values. First of all, the inflow and price data, can change drastically. These input parameters are not something that the programmer can control and therefore it has a unreliable effect on the model. When using a deterministic model, one often has a clear knowledge of the inflow and price for the given stages. For the stochastic model, the accuracy of the calculations increases. This accuracy is dependent on the amount of discrete reservoir states, the amount of price nodes and the amount of inflow nodes. By increasing these, the solution time increases as discussed. Therefore it is important to compare solution time against accuracy. In the stochastic model the computation time is rather low, which can give the programmer the ability to increase the accuracy. The inflow and price nodes are accurate enough, but by increasing the discrete reservoir levels, one can increase the accuracy of the calculation. This has not been done with this model, but can be interesting to look into further.

8.2 Multi-area with or without losses

In this section, one aims to find the impact on the water values, with losses on the interconnected lines. This impact will be shown by comparing the two multi-area models and their behaviour. The result of the two-area model is shown in figure 6.3, and the result of the three-area model is shown in figure 6.4. These two models are not similar in the structure, since one of the models include another area. This is one aspect that will be included in the discussion, including the effect of losses.

Comparing water values

As one can see from the water value figures in figure 6.3 and 6.4, they appear to have the same structure as the one area stochastic model. This makes sense since the data is not that different from the each area. The water values in both models appear to be more impacted by the change in discrete reservoir levels. This is shown by the steepness of the curves decreasing rapidly from empty reservoir to full reservoir. Another aspect that concerns both models is the fact that they are more smoother in transitions between stages. This can be backed up by the construction of the models. Since one extends to another area, one shifts the objective function to include all areas simultaneously. This change enhances the global optimum and the fact that these areas have the opportunity to sell and purchase energy makes the stage-transition smoother. In the decision-making aspect of the models, they are always taking into account the value of the water left in the reservoir, and the price of purchasing energy. If the value of the water left in the reservoir is higher than the price of purchase, the model will purchase. At the same time, the model wants to make use of the water in the reservoir plus the inflow, to not run the risk off spillage. Remember that the objective is to maximize the profit of producing hydro power, which means that running the risk of producing spillage will lower the value of the water, and therefore the objective as well.

Impact of losses

Including losses to the line means that the producer that is selling an amount of energy, needs to compensate for the losses. To compensate this, the hydro power station needs to produce more, which leads to higher water values. Another restriction is that one cannot sell more than the maximum capacity of the hydro power plant. That being said, selling energy to another area, can only be allowed if the demand in the production area is fulfilled. By comparing the two models, one can clearly see the effect of the losses in the water values. The water values for the model with no losses are lower than the model with losses.

In the Nordic system, the losses on the lines needs to be covered by the TSO. That being said, the profitability of selling hydro power instead of buying, is the change in prices for both areas times the losses[17] and this will impact the water values. As one can see from figure 6.3, when the reservoir is full, it has the opportunity to sell energy and is there is no losses which compensate the water values as for figure 6.4.

Solution time and accuracy

For these models, the solution time is not that different from each other. They both converged in 4 iteration with a solution time of 1900 seconds for the two-area model and 2300 seconds for the three-area model. The accuracy for the water values are different, because the three-area model is including losses. This is a more realistic approach, even though it leads to a bigger solution time. As for the stochastic one-area model, the solution time are still low, so it would be beneficial to increase the discrete reservoir levels.

Chapter 9

Conclusion

In this thesis, the research of water values has been conducted. One have gained insight on how to model a stochastic dynamic program and how to extend the model to different areas. This research has been conducted to look at the impact on areas where the decommission of thermal power are at stake. For further knowledge, the introduction of wind energy has been looked at to see how the model reacts to another energy source. In this chapter, the author will conduct his conclusion of the thesis, and his understanding of the work that has been done.

Decommission of thermal and fossil fuel, is dependent on having enough energy sources and to be able to compete on the market with the low cost of thermal energy. This model shows how the impact of scheduling a hydro power station with a thermal power plant. The water values indicates the price of producing hydro power instead of thermal energy. One can clearly see that the introduction of wind makes the water values lower, which has an impact on the decommission. This shows that the rapid growth in wind and solar energy in Europe, will play a big part in moving from fossil fuel to renewable energy and reaching the European goals.

The results from the model is assumed to be correct, considering the assumptions and simplifications that were made. By introducing losses to the lines, one can also see the water values react to the increase in production, which lowers the reservoir level. This indicates the results to be reasonable and dependent on the model.

The stochastic variables show stat the accuracy in the calculations increases from the deterministic model. The model has a low computation time, which means the accuracy in the calculations can be increased by increasing the price- and inflow nodes. Also, it would be more accurate if one increases the discrete reservoir levels for further analysis.

Chapter 10

Further work

When including wind to the three-area problem, there are some assumptions that has to be made. First of all, because of the delivery deadline of the master thesis, the author did not have time to make the wind variable stochastic. It would be preferred to make it stochastic to create a more realistic model. With that being said, the author believes that even though the wind values are fixed, it will still contribute to the model.

Another work that needs to be done is to differentiate between the prices in the different areas. It would be preferred using values from three different NO-zones and check the results.

Appendix A

Acronyms

SDP Stochastic dynamic programming

DP Dynamic programming

LP Linear programming

Water value Marginal operational cost of producing one unit more of hydro power

SDDP Stochastic dual dynamic programming

Iteration The repetition of a process

State Description of how the state of the system is

Stage One point in a series of positions, here referred as time

Aggregate Several systems made into one system

Long-term scheduling The operational planning process of a year or more

Short term scheduling The operational planning process of a week or month

Head Height different between reservoir level and turbine

Syntax The structure of statements in a computer language

NTNU Norwegian University of Science and Technology

Appendix B

Additional Information

This Chapter contains the Python code that has been used in this master thesis.

B.1 Pyton Code final model with wind

```
# Import neccessary tools and packages
import pyomo.environ as pyo
#import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
#from matplotlib import style
from inputdata import processData, createModelData
from mpl_toolkits.mplot3d import Axes3D
from functions import plot3DWVTable
import time as time
import random
# ==========================================================================
# Formulate optimization model as abstract
# ==========================================================================
model = pyo.AbstractModel(name="SDP")
# Define solver
```

```
opt=pyo.SolverFactory('gurobi',solver_io="python")
# ==========================================================================
# Define sets of the model
# ==========================================================================
model.STAGE=pyo.Set() # Time stage
model.STATE=pyo.Set(ordered=True) # Reservoir states (discretizations)
def areaidex(model):
    return [(i) for i in range(sys_data["Area"])]
model.Area = pyo.Set(ordered=True)

def SOS_indices_init(model):
    return [(i) for i in range(sys_data["NResStates"]-1)]
model.SOS_indices=pyo.Set(ordered=True,initialize=SOS_indices_init)
# ==========================================================================
# Parameters of the model
# ==========================================================================
sys_data=processData()
dict_data=createModelData(sys_data)
T = [i for i in range(sys_data["NStages"])] #= pyo.RangeSet(1,52)
R = {idx:i for idx,i in enumerate(range(0,sys_data["VolMax"]+1,int(sys_data["VolMax"]/
model.Eff = pyo.Param(model.G,initialize=0.0,mutable=True)
model.ProdMin = pyo.Param(model.G,initialize=0.0,mutable=True)
model.ProdMax = pyo.Param(model.G,initialize=0.0,mutable=True)
model.SCost = pyo.Param(model.G,initialize=0.0,mutable=True)
model.MinEx = pyo.Param()
model.MaxEx = pyo.Param()
model.Inflow = pyo.Param(initialize=0.0,mutable=True) #  mutable: Able to redine later
model.Demand = pyo.Param(initialize=0.0,mutable=True)
model.Pw = pyo.Param(initialize=0.0,mutable=True)
model.Eprice = pyo.Param(initialize=0.0, mutable=True)
model.Cprice = pyo.Param(initialize=0.0, mutable=True)
model.InitVol = pyo.Param(initialize=0.0,mutable=True)
model.Ct = pyo.Param()
model.penspi = pyo.Param()
model.PVOLL = pyo.Param()
model.PgtMin = pyo.Param()
model.PgtMax = pyo.Param()
model.PghMin = pyo.Param()
model.PghMax = pyo.Param()
model.VolMin = pyo.Param()
model.VolMax = pyo.Param()
model.TwaPen = pyo.Param()
model.eff = pyo.Param()
model.WV = pyo.Param(model.STATE,mutable=True)
model.DWV = pyo.Param(initialize=0.0,mutable=True)
model.I=pyo.RangeSet(0,5)
model.P=pyo.RangeSet(0,15)
model.InflowState = pyo.Param(model.I,T,mutable=True)
model.EpriceState = pyo.Param(model.I,T,mutable=True)
model.TrProbInflow = pyo.Param(model.I,model.I,T,mutable=True)
model.TrProbPrice = pyo.Param(model.P,model.P,T,mutable=True)
# ======================================================================
# Define Variables of the model
# ======================================================================

def ptrrule11(model):
    return(model.MinEx,model.MaxEx)
model.Ptr11 = pyo.Var(bounds=ptrrule11)
def ptrrule21(model):
    return(model.MinEx,model.MaxEx)
```

```
model.Ptr21 = pyo.Var(bounds=ptrrule21)
def ptrrule12(model):
    return(model.MinEx,model.MaxEx)
model.Ptr12 = pyo.Var(bounds=ptrrule12)
def ptrrule22(model):
    return(model.MinEx,model.MaxEx)
model.Ptr22 = pyo.Var(bounds=ptrrule22)
model.Pll = pyo.Var(within=pyo.NonNegativeReals)
def pcgRule(model):
    return(model.PghMin, model.PghMax)
model.Pcg= pyo.Var(bounds=pcgRule,within=pyo.NonNegativeReals)
def pgtRule(model):
    return(model.PgtMin, model.PgtMax)
model.Pgt = pyo.Var(bounds=pgtRule, within=pyo.NonNegativeReals)
def pghRule(model):
    return(model.PghMin, model.PghMax)
model.Pgh = pyo.Var(bounds=pghRule, within=pyo.NonNegativeReals)
model.Spi = pyo.Var(within=pyo.NonNegativeReals)
model.Twa = pyo.Var(bounds=(0,0), within=pyo.NonNegativeReals)
model.Delta= pyo.Var(model.SOS_indices, within=pyo.Binary)
def VolRule(model, i):
    return(R[0], R[1]) # Discretization step for reservoir state
model.Vol = pyo.Var(model.SOS_indices, bounds=VolRule,  within=pyo.NonNegativeReals)
# ==========================================================================
# Objective function of the model
# ==========================================================================
def hydroObjRule(model):
    objExpr=-model.Ct*model.Pgt - model.TwaPen*model.Twa
#    objExpr-=model.Eprice*model.Ptr1
    objExpr-=model.Eprice*(model.Ptr11+model.Ptr21-model.Ptr12-model.Ptr22)
```

```
#       objExpr-=model.Eprice*model.Pep
        objExpr-=model.PVOLL*model.Pll
        objExpr+=sum(model.WV[j]*model.Vol[j] for j in model.SOS_indices)
        objExpr+=model.DWV
        objExpr-=model.Spi*model.penspi
        return(objExpr)
model.Obj=pyo.Objective(rule=hydroObjRule, sense=pyo.maximize)

# ==============================================================================
# Constraints of the model
# ==============================================================================
#def EnergyBalanceRule(model):
#       return( -model.Pgt-model.Pgh-model.Pw - model.Pll -model.Ptr1 + model.Demand ==
#model.EnergyBalance = pyo.Constraint(rule=EnergyBalanceRule)
def EnergyBalanceRule(model):
        return(-model.Pgt - model.Pgh - model.Pw - model.Pll -
        (((model.Ptr11+model.Ptr21))-(model.Ptr12+model.Ptr22))
        + model.Demand == 0.0)
model.EnergyBalance = pyo.Constraint(rule=EnergyBalanceRule)

def transruleexp(model,i):
        if i == 0:
                return(model.Ptr11[i]-(model.Ptr12[i+1]*0.9)==0)
                return(model.Ptr21[i]-(model.Ptr22[i+2]*0.9)==0)
        elif i == 1:
                return(model.Ptr11[i]-(model.Pt12[i-1]*0.9)==0)
                return(model.Ptr21[i]-(model.Ptr22[i+1]*0.9)==0)
        else:
                return(model.Ptr11[i]-(model.Ptr12[i-1]*0.9)==0)
                return(model.Ptr21[i]-(model.Ptr22[i-2]*0.9)==0)
```

```
model.Transmissionexp = pyo.Constraint(model.Area, rule=transruleexp)


#def transruleexp(model, i):
#     if i == 0:
#          return(model.Ptr1[i]-(model.Ptr1[i+1])==0)
#
#     else:
#          return(model.Ptr1[i]-(model.Ptr1[i-1])==0)
#model.Transmissionexp = pyo.Constraint(model.Area, rule=transruleexp)


def ResBalanceRule(model):
     return(- model.InitVol + (model.Pgh*model.eff) - model.Inflow
          + sum(model.Vol[i] for i in model.SOS_indices) - model.Twa + model.Spi== 0)
model.ResBalance = pyo.Constraint(rule=ResBalanceRule)


def volSOSUB(model, i):
     if i<sys_data["NResStates"]-1 and i>1:
          return(model.Delta[i-1] >= model.Delta[i])
     else:
          return(model.Delta[i] >= model.Delta[i+1])
model.volSOS = pyo.Constraint(model.SOS_indices, rule=volSOSUB)


def VolUBRule(model, i):
     return(model.Vol[i] <=R[1]*model.Delta[i])
model.VolUB = pyo.Constraint(model.SOS_indices, rule=VolUBRule)


def VolLBRule(model, i):
     if i < sys_data["NResStates"]-2:
          return(R[1]*model.Delta[i+1] <= model.Vol[i])
     else:
        return(0 <= model.Vol[i])
model.VolLB = pyo.Constraint(model.SOS_indices,rule=VolLBRule)

# ============================================================================
#
# ============================================================================

instance=model.create_instance(dict_data,namespace="model")
instance.pprint(filename="model.txt")
A=[i for i in range(sys_data["Area"])]
P=[i for i in range(sys_data["P"])]
I=[i for i in range(sys_data["I"])]
Alfa=np.zeros((len(P),len(I),len(A)))
#AlfaPrev=0.0
Alfafrom=0.0

AlfaPrev=np.zeros((len(P),len(I),len(A)))
#Alfafrom=np.zeros((len(P),len(I)))
ResEnd=[i/(sys_data["NResStates"]-1)*sys_data["VolMax"] for i in range(sys_data["NRes

WVTable=np.zeros((len(R)-1,len(P),len(I),len(T),len(A)))
WVTablePrev=np.zeros((len(R)-1, len(P),len(I),len(T),len(A)))
#WVTableEnd=sys_data["WV"]
WVTableEnd=np.zeros((len(R)-1,len(P),len(I),len(A)))
WVTableStart = sys_data["WV"]
DValTable=np.zeros((len(P),len(I),len(T),len(A)))
DValEnd=np.zeros((len(P),len(I),len(A)))
#InflowState=np.zeros((len(T),len(I)))
#EpriceState=np.zeros((len(T),len(P)))
#MaxE=model.MaxE
```

```
#MaxI=model.MaxI
#P=instance.P
#I=instance.I
TrProbInflow = sys_data["TrProbInflow"]
TrProbPrice = sys_data["TrProbPrice"]
ItMax=20
Cdiff=0
PolicyEps=0.1



print("="*60)
print("Areas␣-", sys_data["Area"])
print("Max␣iterations␣-␣", ItMax)
print("NStages␣-␣",sys_data["NStages"])
print("NReservoir␣States␣-␣,",sys_data["NResStates"])
print("="*60)
print("---␣Starting␣SDP␣Procedure␣---")
print("="*60)
for it in range(ItMax):
    print("Iteration,␣", it+1)
    t1=time.time()
    for area in range(len(A)):
        for iStage in reversed(range(sys_data["NStages"])):
            instance.Demand=sys_data["Demand"][area][iStage]
            instance.Pw = sys_data["Pw"][area][iStage]
            for iR in range(len(R)):
                instance.InitVol=R[iR]
                for pto in range(len(P)):
                    instance.Eprice=sys_data["EpriceState"][area][iStage][pto]
instance.Cprice=sys_data["CpriceState"][area][iStage][pto]
```

```
            for ito in range(len(I)):
                instance.Inflow=sys_data["InflowState"][area][iStage][ito]
# Update WV Table for last stage.
                if iStage==sys_data["NStages"]-1:
                    for iState in range(len(R)-1):
                        instance.WV[iState]=
                        WVTableEnd[iState][pto][ito][area]
                        instance.DWV=DValEnd[pto][ito][area]
                else:
                    for iState in range(len(R)-1):
                        instance.WV[iState]=
                        WVTable[iState][pto][ito][iStage+1][area]
                        instance.DWV=
                        DValTable[pto][ito][iStage+1][area]

                opt.solve(instance, tee=False,keepfiles=False)
                Alfa[pto][ito][area]=instance.Obj()
                instance.write('model.lp',
                io_options={'symbolic_solver_labels':True})
        for pfrom in range(len(P)):
            for ifrom in range(len(I)):
                Alfafrom = 0
                for pto in range(len(P)):
                    for ito in range(len(I)):
                        Alfafrom = Alfafrom +
                        ((TrProbPrice[iStage][pfrom][pto])*
                        (TrProbInflow[iStage][ifrom][ito])*
                        (Alfa[pto][ito][area]))
                if iR==0:
                    DValTable[pfrom][ifrom][iStage][area]=Alfafrom
```

```
                        else:
                            WVTable[iR-1][pfrom][ifrom][iStage][area]=
                            ((Alfafrom-AlfaPrev[pfrom][ifrom][area])/
                            (ResEnd[iR]-ResEnd[iR-1]))

                        AlfaPrev[pfrom][ifrom][area]=Alfafrom
                # Compute WV
    # Convergence check
    Cdiff = np.sum(np.absolute(WVTable-WVTablePrev))
    if Cdiff<PolicyEps:
        print("Converged␣in␣", it, "␣iterations")
        print("Cdiff:␣", Cdiff)
        break
    else:
        print("Policy␣difference:␣", Cdiff)
    # Update WV tables
    WVTablePrev=np.copy(WVTable[:,:,:,:,:])
    WVTableEnd=np.copy(WVTable[:,:,:,0,:])
    DValEnd=DValTable[:,:,0,:]
    t2=time.time()-t1
    print("Time:␣", t2)
    print("="*60,)
    print("WV␣Table")
    np.set_printoptions(suppress=True)
    print(WVTable)
    print("="*60,)
    print("\nDisplaying␣Solution\n" + "-"*60)
    print("Cdiff:",Cdiff)
    print("Solution␣time:␣", t2)
instance.write('model.lp',io_options={'symbolic_solver_labels':True})
```

B.2 Time series of wind energy

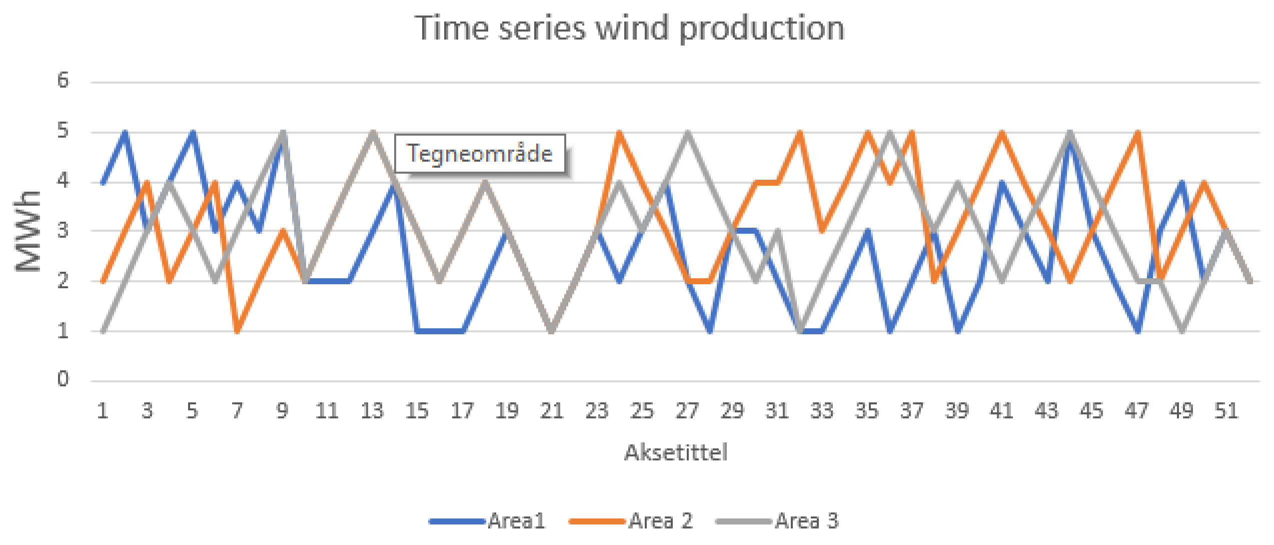


Figure B.1: Time series of wind production

Bibliography

[1] Chiara Bordin Arild Helseth, Marte Fodstad. *Comparison of two approaches for implementing multireservoir operating policies derived using stochastic dynamic programming.* 2017.

[2] Magnus Askeland Birger Mo Odd Bjarte Nilsen Juan Ignacio Pérez-Díaz Manuel Chazarra Ignacio Guisández Arild Helseth, Marte Fodstad. *IET Reseach Journals*, pages 1–9.

[3] European Commision. *EU Energy, Transport and GHG Emissions Trends to 2050.* 2013.

[4] European Commision. *A policy framework for climate and energy in the period from 2020 to 2030.* 2014.

[5] Gerald L. Doorman. *Hydro Power Scheduling.* Department of Electric Power Engineering, NTNU, Autumn 2017.

[6] Magnus Korpås Martin N. Hjelmeland Hans H. Faanes, Gerald L. Doorman. *Energy Systems Planning and Operation.* Department of Electric Power Engineering, Mathematics and Electrical Engineering, Faculty of information Technology, NTNU, January 2016.

[7] Nordic Energy Reseach International Energy Agency. *Nordic Energy Technology Perspectives 2016.* 2016.

[8] J. Lindqvist. *Operation of a Hydrothermal Electric System: A Multistage Decision Process.* 1962.

[9] L.M.V.G Pinto Mario Pereira. *Optimal stochastic operations scheduling of large hydroelectric systems.* 1984.

[10] L.M.V.G Pinto Mario Pereira. *Multi-stage stochastic optimization applied to energy planning.* 1991.

[11] Rafael Kelman Mario Pereira, Nora Campodonico. *Long-term Hydro Scheduling based on Stochastic Models.* 1998.

[12] Magnus Korpås Martin N Hjelmeland, Camilla T Larsen. *Provision of rotating reserves from wind power in a hydro-dominated power system.* 2016.

[13] Birger Mo Anders Gjelsvik Ivar Wangensteen Gerard Doorman Ove Wolfgang, Arne Haugstad. *Hydro reservoir handling in Norway before and after deregulation.* 2009.

[14] Arnulf Jäger-Waldau Roberto Lacal Arantegui. *Photovoltaics and wind status in the European Union after the ParisAgreement.* 2017.

[15] Y. Larsson S. Stage. *Incremental Cost of Water Power.* 1961.

[16] Kasper Emil Thorvaldsen. *Profitability of a Hydropower Producer Operating in Two Different Markets.* 2017.

[17] Ivar Wangensteen. *Power system economics - the Nordic Electricity Market.* Fagbokforlaget, 2nd edition, 2012.