The University of Auckland

Department of Engineering Science

Part IV Project

Infinite-Horizon in Stochastic Dual Dynamic Programming

Author:
Shasa Foster

With
Ben Fulton

Supervisor:
Dr Tony Downward

September 21, 2018

Contribution Declaration

SDDP models have been used to model the NZEM (New Zealand Electricity System) since 2007. First was the DOASA model, developed in C++ by Andy Philpott and Geoffrey Pritchard [1]. The JADE.jl package, An implementation of DOASA in Julia was later developed by Lea Kapelevich [2]. JADE used the SDDP.jl package developed by Oscar Dowson [3] to construct the NZ hydro-thermal scheduling problem into a SDDP.

I extended the SDDP.jl and JADE.jl packages with functionality that allowed for modeling the NZEM with infinite-horizon SDDP. My project supervisor Dr Tony Downward guided my extensions to the SDDP.jl and JADE.jl packages. I also implemented cut selection algorithms introduced by Matos et al.[4] and my project supervisor, Dr Tony Downward. Finally, I extended JADE.jl to consider stagewise dependent inflows via a Markov chain. My work is mostly presented in Section 4, 5, 6 and 7.

This project was a paired project with Ben Fulton. My report focuses on my part of the project which has the main objective of extending the JADE model from using SDDP to infinite-horizon SDDP. Fulton focused on the simulation, interpretation and analysis of the results of this extended model. Fulton wrote the VBA macros for analysis and interpretation of simulation results and incorporation of future renewable sources into the JADE model. Ben Fulton's work and findings are presented in his report [5].

Abstract

The New Zealand government aims for 100% of New Zealand's electricity generation to come from renewable sources by 2035[1]. This objective is causing additional uncertainty around the future of Huntly Power Stations' coal-fired units which already have intermittent use because of their function as a 'peaker' during periods of extended low reservoir levels.

Determining the future of Huntly requires a model of the New Zealand Electricity Market (NZEM). This project builds on previous research. Hydro-thermal scheduling models of the NZEM such as JADE and DOASA have been used to research the value of Huntly in the NZEM as well as future renewable generation mixes.

JADE and DOASA models are solved using the stochastic dual dynamic programming algorithm (SDDP). An explicit assumption of SDDP is an exogenous, predefined terminal marginal cost function. This assumption reduces the accuracy of these models and their results.

We extended the JADE model to an 'infinite-horizon' SDDP with an endogenous terminal marginal cost function. Computational improvements reduced the run-time of the 'infinite-horizon' SDDP from greater than 40 hours down to 30 minutes thus enabling accurate solutions to determine the value of Huntly's coal-fired units and the future renewable generation mixes.

[1] in a normal hydrological year

Acknowledgements

Foremost, I would like to thank my project supervisor Tony Downward for his consistent support, patience and feedback throughout the duration of the project.

I am grateful to Oscar Dowson and Lea Kapelevich, for their SDDP.jl and JADE.jl packages I built onto. I would also like to acknowledge Tony Downward, Oscar Dowson and well as Andy Philpott for the development of the theory behind the SDDP infinite-horizon algorithm.

Finally, thank you to my project partner Ben Fulton, for his camaraderie and support through this endeavour.

Contents

List of Figures

1 Introduction

This report discusses the application of infinite-horizon stochastic dual dynamic programming in determining the security of supply of electricity in the NZEM under the current context of the uncertain future of the thermal Huntly Power Station. Additionally, as New Zealand aims for generation of electricity from 100% renewable sources by 2035[2], research on the suitability of different renewable generation mixes have also been carried out.

According to Genesis Energy, the owner of the Huntly power station, its final two Rankine units will be shut down in 2018 and 2022 respectively, which may pose serious risks to New Zealand's electricity supply during dry years due to New Zealand's dependence on hydroelectric generation.

Generators face the problem of hydro-thermal scheduling which can be modelled as a multi-stage stochastic problem that seeks to determine the marginal values of water. The marginal values of water inform generators on the optimal policy of hydro-thermal scheduling. JADE, a stochastic dual dynamic programming model of the NZEM discussed in Section 3, has been used to model different scenarios of the NZEM. The focus of this project was to improve the JADE model by its extension to an infinite-horizon case and then use it to determine the performance and the security of supply in the NZEM under different scenarios.

However, this report will focus exclusively on the algorithm used to solve this problem. There is a companion report by my project partner Ben Fulton [5], that details the simulations, analysis and results of applying the algorithm to different scenarios in the NZEM.

A detailed introduction and motivation are present in the Literature Review and SORI (Statement of Research Intent) document accompanying this report [6].

1.1 Report Structure

The main objective for my part of this project is to extend an existing hydro-thermal scheduling model of the NZEM. Precisely, this involved the extension of an SDDP (Stochastic Dual Dynamic Programming) model of the NZEM (the JADE model) into an infinite-horizon SDDP. Thus, my report focuses on the infinite-horizon SDDP algorithm, while simulation results and analysis are contained in project partner Ben Fulton's report [5].

Section 2 introduces the theory of dynamic programming, stochastic dynamic programming, stochastic dual dynamic programming, infinite-horizon dynamic programming and infinite-horizon stochastic dynamic programming. Section 3 presents JADE, an implementation of SDDP used in modeling the NZEM. Section 4 describes the developments made to JADE and the SDDP algorithm, extending the model into an 'infinite-horizon' model. Convergence tests for this enhanced JADE model are run in Section 5 to determine how several user defined parameters affect the convergence properties. Section 6 discusses the implementation of three improvements that speed up the solve-time of the algorithm. Section 7 describes the implementation of stagewise dependent inflows using a Markov model. Finally Section 8 and Section 9 present ideas for future work and conclusions.

[2] in a normal hydrological year

2 Stochastic Dual Dynamic Programming Formulation

This section introduces Dynamic Programming, Stochastic Dynamic programming and SDDP (Stochastic Dual Dynamic Programming). The mathematical equations displayed have been drawn from the course notes for the ENGSCI 760 and 763 courses [7] [8], the Downward et al presentation at the 2018 EPOC conference [9], Simmonds' thesis [10] and Dowson's thesis [11].

2.1 Dynamic Programming

Dynamic programming is a solution approach for staged decision problems. The staged problem is broken down into a series of simpler similar subproblems called stages. In JADE, the problem is a 52-week decision problem broken down into 1-week stages.

At each stage, the system may be in several different states. In JADE, the states are the reservoir levels (in m^3). The decision made at each stage determines how the system moves to a new state at the next stage.

The objective of a dynamic programming problem is to find an optimal decision policy that minimises cost for each state in each stage of the problem. The minimum cost for a stage is found using the Bellman recursion function, also known as the Bellman cost-to-go function:

$$V_t(\mathbf{x}_t) = \min_{\mathbf{a}_t \in A_t} \{C_t(\mathbf{x}_t, \mathbf{a}_t) + V_{t+1}(f_t(\mathbf{x}_t, \mathbf{a}_t))\} \tag{2.1}$$

$\mathbf{x}_t$ is the state in stage t
$f(\mathbf{x}_t, \mathbf{a}_t)$ is the new state in the next stage ($\mathbf{x}_{t+1}$) resulting from action $\mathbf{a}_t$ in state $\mathbf{x}_t$ and stage t
$C(\mathbf{x}_t, \mathbf{a}_t)$ is the is cost in stage t of taking action $\mathbf{a}_t$ in state $\mathbf{x}_t$
$V_{t+1}(\cdot)$ is the future cost-to-go function in stage $t+1$

Note a terminal future cost-to-go is needed for the final stage when $t = T$.
$V_{T+1}(\mathbf{x}_{T+1})$ is set at the start of the recursion to be a known predefined function. The Bellman cost-to-go function in the final stage is $V_T(\mathbf{x}_T) = \min_{\mathbf{a}_t \in A_t}\{C_t(\mathbf{x}_T, \mathbf{a}_T) + V_{T+1}(f_t(\mathbf{x}_{T+1}, \mathbf{a}_T))\}$

2.2 Stochastic Dynamic Programming

Stochastic dynamic programming introduces randomness into the formulation. The randomness in a stochastic dynamic program is called the noise. In JADE, this uncertainty is the inflows of water to the seven reservoirs due to rainfall and snowmelt. These inflows are modelling by random variables (vectors) labelled ω_t.

In each stage t, the random variable ω_t is realised at the beginning of the stage ($P(\omega_t)$ is the probability of observing the random variate ω_t). However, the realisation of future random variables (inflows in future weeks) is still uncertain. In assuming stagewise independence of the noise (an assumption in JADE), the Bellman function now becomes:

$$V_t(\mathbf{x}_t, \omega_t) = \min_{\mathbf{a}_t \in A_t} \{C_t(\mathbf{x}_t, \mathbf{a}_t, \omega_t) + \sum_{\omega_{t+1} \in \Omega_{t+1}} P(\omega_{t+1}) \cdot V_{t+1}(f_t(\mathbf{x}_t, \mathbf{a}_t), \omega_{t+1})\} \tag{2.2}$$

Let $\mathcal{V}_{t+1}(\mathbf{x}_{t+1})$ be the expected future cost-to-go in stages $t+1 \ldots T$.

$$\mathcal{V}_{t+1}(\mathbf{x}_{t+1}) = \sum_{\omega_{t+1} \in \Omega_{t+1}} P(\omega_{t+1}) \cdot V_{t+1}(f(\mathbf{x}_t, \mathbf{a}_t), \omega_{t+1}) \tag{2.3}$$

Now $V_t(\mathbf{x}_t, \omega_t) = \min_{\mathbf{a}_t \in A_t} C(\mathbf{x}_t, \mathbf{a}_t, \omega_t) + \mathcal{V}_{t+1}(\mathbf{x}_{t+1})$. Taking expectations on both sides of the equation with respect to the noise ω_t, the Bellman recursion becomes:

$$\mathcal{V}_t(\mathbf{x}_t) = \mathop{\mathbb{E}}_{\omega_t \in \Omega_t} [\min_{\mathbf{a}_t \in A_t} \{C_t(\mathbf{x}_t, \mathbf{a}_t, \omega_t) + \mathcal{V}_{t+1}(f_t(\mathbf{x}_t, \mathbf{a}_t))\}] \tag{2.4}$$

2.3 Stochastic Dual Dynamic Programming

The standard and stochastic dynamic programming formulations detailed previously require a discrete set of states. However, in JADE the states are reservoir levels which are continuous. Creating a meaningful discrete approximation of the reservoir levels would make the problem computationally infeasible. For example, if each of the states of the seven reservoirs in week 1 ($\mathbf{x}_1$) was discretised into 50 values there will be 50^7 possible discrete values of $\mathbf{x}_1$.

This problem, referred to the curse of dimensionality, can be avoided by approximating the future expected cost-to-go, $\mathcal{V}_{t+1}$, by a piecewise linear function compared to a set of discrete values.

In stage T, we have a predefined function for $\mathcal{V}_{T+1}(\mathbf{x}_{T+1})$, however in all other stages 1,2,...,T an approximation of $\mathcal{V}_t(\mathbf{x}_{t+1})$ will be refined over the course of the algorithm.

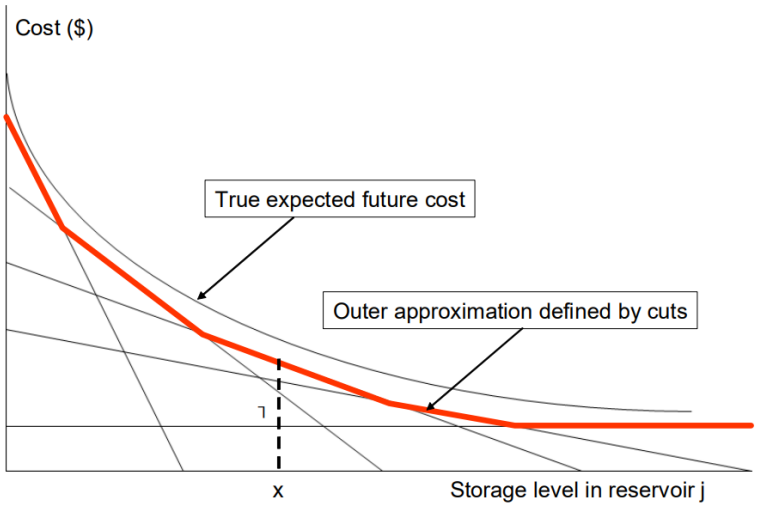

Figure 2.1: Linear cuts approximating the expected future cost-to-go function

Figure 2.1 demonstrates how a piecewise linear function can approximate the expected future cost. When there a few linear cuts the approximation is imprecise, however over the course of the SDDP algorithm the cuts are successively added to the function which improves the accuracy of the approximation of the future cost-to-go $\mathcal{V}_{t+1}(\mathbf{x}_{t+1})$. $\mathcal{V}_{t+1}(\mathbf{x}_{t+1})$ for a given state $\mathbf{x}_{t+1}$ is determined by the minimum of the cuts at the state $\mathbf{x}_{t+1}$ through solving the linear program in equation (2.5).

$$\begin{aligned} \mathcal{V}_{t+1}(\mathbf{x}_{t+1}) = \min\ & \theta \\ \text{s.t.} \quad & \theta \geq \alpha_{t+1}^j + \vec{\beta}_{t+1}^j \cdot \mathbf{x}_{t+1} \qquad \text{for cuts j} = 1,2,...\text{J} \end{aligned} \tag{2.5}$$

Each cut j (produced from iteration j of SDDP) from stage $t+1$ produces a lower bound for $\mathcal{V}_{t+1}(\mathbf{x}_{t+1})$. Cut j at stage t is defined by a y-intercept α_t^j, a gradient $\vec{\beta}_t^j$ and its sampled state $\mathbf{x}_t^j$. The gradient is a vector of the same dimension as the state. The dominating cuts produce a piecewise linear function that is the solution to this problem and a lower bound for the future expected cost-to-go function.

2.3.1 Forward Pass

In the forward pass the optimal decision, $\hat{\mathbf{a}}_t$, is based of the current state $\mathbf{x}_t$, the realisation of the random variable ω_t, and the future cost-to-go approximation $\mathcal{V}_{t+1}(\mathbf{x}_{t+1})$. The new state, $\mathbf{x}_{t+1}$, from the optimal decision $\hat{\mathbf{a}}_t$, is *passed forward* to be the starting state in the next stage.

Algorithm 1: Forward Pass Algorithm

(1) Sample the random variable ω_t

(2) Make the optimal decision $\hat{\mathbf{a}}_t$ based on the $\mathbf{x}_t$, ω_t and $\mathcal{V}_{t+1}$ by:

$$\hat{\mathbf{a}}_t = \arg\max_{\mathbf{a}_t \in A_t} \{C_t(\mathbf{x}_t, \mathbf{a}_t, \omega_t) + \mathcal{V}_{t+1}(f_t(\mathbf{x}_t, \mathbf{a}_t))$$

2.3.2 Backward Pass

The backwards pass produces linear cuts defined by the dual of the linear program, π_t, that is used to approximate the expected cost-to-go approximation $\mathcal{V}_t$ in the current stage. The cut is *passed back* a stage and used to approximate the future cost-to-go function in the previous stage. Cuts are lower bounds for the future expected cost-to-go function.

Algorithm 2: Backward Pass Algorithm

(1) Solve the following Linear Program in stage t and iteration j,

$$\begin{aligned} V_t^j(\mathbf{x}_t, \omega_t) = \min_{\mathbf{a}_t \in A_t} \quad & C_t(\bar{\mathbf{x}}_t, \mathbf{a}_t, \omega_t) + \theta_t \\ \text{s.t.} \quad & \hat{\mathbf{x}}_t = T_t(\bar{\mathbf{x}}_t, \mathbf{a}_t, \omega_t) \\ & \mathbf{a}_t \in \mathbf{A}_t(\bar{\mathbf{x}}_t, \omega_t) \\ & \bar{\mathbf{x}}_t = \mathbf{x}_t \qquad [\pi_t] \\ & \theta_t \geq \alpha_{t+1}^j + \vec{\beta}_{t+1}^j \cdot \hat{\mathbf{x}}_{t+1} \quad j \in \{1, 2, ..., J\} \end{aligned} \tag{2.6}$$

(2) Construct the cut $\theta \geq \alpha_{t+1}^i + \vec{\beta}_{t+1}^i \cdot \mathbf{x}_{t+1}$ for the approximate expected cost-to-go function $\mathcal{V}_t(\mathbf{x}_t)$ by,

$$\vec{\beta}_t^j = \mathop{\mathbb{E}}_{\omega_t \in \Omega_t} [\pi_t(\omega_t)]$$

$$\alpha_t^j = \mathop{\mathbb{E}}_{\omega_t \in \Omega_t} [\hat{V}_t(\mathbf{x}_t^j, \omega_t)] - \vec{\beta}_t^j \cdot \mathbf{x}_t^j$$

2.3.3 SDDP Algorithm

In a given iteration j of the SDDP algorithm, the forward pass algorithm is applied first to stages $1, 2, \ldots, T-2, T-1$. Then, the backward pass algorithm is applied to stages $T, T-1$, $\ldots, 2$ [12]. This defines one iteration of the SDDP algorithm. The SDDP algorithm (shown below) continues iterating until the decision policy of the algorithm has converged, which usually takes thousands of iterations.

Algorithm 3: SDDP algorithm overview

```
j = 0
while policy has not converged do
    % Do Forward Pass
    for t = 1 to T − 1 do
        Apply forward pass algorithm (algorithm 1) to stage t
    end
    % Do Backward Pass
    for t = T to 2 do
        Apply backward pass algorithm (algorithm 2) to stage t
    end
    j = j + 1
end
```

2.4 Infinite-Horizon Dynamic Programming

Infinite-horizon dynamic programs are characterised by having an 'infinite number of stages', $T = \infty$. Hence, there is no specified terminating stage T and no terminal future cost $V_{T+1}(\mathbf{x}_{T+1})$. There are two ways of characterizing an infinite-horizon dynamic program, through a discounted cost model or an average cost model.

Only the average cost model is discussed, because this was the model used later in Section 4 to development the 'infinite-horizon algorithm'. The average cost model was chosen over the discounted cost model because it converges faster.

Average cost model The average cost model defines the new terminal cost to be the cost-to-go in stage 1 subtracted by a constant, Δ. Δ can be interpreted as the 'expected cost' incurred from stages 1 to T. The associated bellman recursion is:

$$V_t^i(\mathbf{x}_t) = \begin{cases} \mathcal{V}_1^i(\mathbf{x}_{t+1}) - \Delta, & \text{if } t = \text{T+1} \\ \min_{\mathbf{a}_t \in A_t}\{C_t(\mathbf{x}_t, \mathbf{a}_t) + V_{t+1}^i(f_t(\mathbf{x}_t, \mathbf{a}_t))\} - \Delta, & \text{otherwise} \end{cases} \tag{2.7}$$

In the average cost model, the terminal cost function is initialised to zero, $V_{T+1}^1(\mathbf{x}_{T+1}) = 0$. Over successive iterations i of the algorithm $V_{T+1}^i(\mathbf{x}_{T+1})$ is updated by the equation (2.7).

2.5 Stochastic Infinite-Horizon Dynamic Programming

Extending an infinite-horizon dynamic program the stochastic case is simple. Expectations with respect to the random variable ω are taken on both sides of the Bellman recursion.

For the average cost model the associated bellman recursion is:

$$\mathcal{V}_t^i(\mathbf{x}_t) = \begin{cases} V_1^i(\mathbf{x}_{t+1}) - \Delta, & \text{if } t = \text{T+1} \\ \mathop{\mathbb{E}}_{\omega \in \Omega}[\min_{\mathbf{a}_t \in A_t}\{C_t(\mathbf{x}_t, \mathbf{a}_t) + V_{t+1}^i(f_t(\mathbf{x}_t, \mathbf{a}_t))\}] - \Delta, & \text{otherwise} \end{cases} \tag{2.8}$$

The application of an infinite-horizon in a stochastic dual dynamic programming setting was the objective of my research and is discussed in Section 4.

3 Hydro-Thermal Scheduling with SDDP

In New Zealand, wind and geothermal generation have steady outputs over the year, but hydro and thermal generation can be scheduled. Since New Zealand's electricity system is hydro dominated, the uncertainty and variability of future inflows into reservoirs makes decision-making regarding reservoir management difficult.

The problem of hydro-thermal scheduling is to minimise the cost of thermal fuel plus shortage costs. This decision problem can be modeled by a multi-stage stochastic program.

JADE is a model of the NZEM using SDDP in the high-level mathematical programming language Julia. JADE use implementations of the SDDP algorithm (from Section 2.3) configured to solve the New Zealand hydro-thermal scheduling problem. JADE seeks a decision policy that minimises the cost of thermal generation and shortage costs. JADE uses the JuMP (Julia for Mathematical Optimization) package to characterise and solve the stage subproblems. Gurobi was used as the solver for the stage subproblems.

This section will provide a high-level formulation of JADE in the context of the SDDP algorithm described in Section 2. A detailed description of JADE is available at references [2], [13].

3.1 Problem Definition

Stages There are 52 stages in the model, $t = 1, 2, ..., 52$. The 52 stages correspond to the 52 weeks in a year. At the beginning of each week, the inflows for the reservoirs are realised before this week's hydro-thermal scheduling decision is made. This is illustrated in Figure 3.1 where the realisation of the stochastic inflows is represented by the wavy arrow.

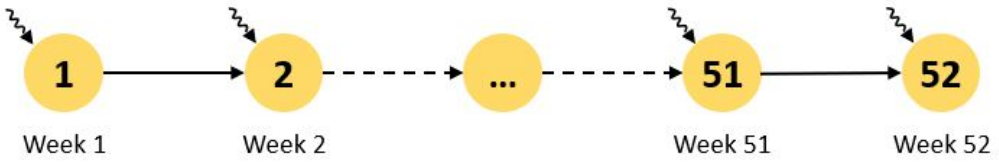

Figure 3.1: Stage graphic for JADE model

State Variables The state variables $\mathbf{x}_t$ represent the amount of water in m^3 of the seven reservoirs with significant inter-week storage capability: Manapouri-Te Anau, Hawea, Ohau, Pukaki, Tekapo, Benmore and Taupo.

Random Variables The noise, ω_t, is a vector inflows of water into each reservoir. In JADE inflows are assumed to be stagewise independent and are sampled from a record from 1970 - 2013. The assumption of stagewise independent inflow JADE is discussed further in this report's associated literature review on page 7.

3.2 Stage Subproblem

For each week t, with given reservoir levels $\mathbf{x}_t$ (equal to the reservoir levels at the end of the previous week $t-1$), the stage subproblem is solved (by algorithm 1) to determine the amount of hydro and thermal generation over the week. The inflows and hydro releases will result in a new reservoir level for each of the seven reservoirs which is demonstrated in Figure 3.2. After the subproblem is solved in week 1, the subproblem is then solved in week 2 using the

resulting reservoir levels from the solution of the week 1 subproblem. This continues until the subproblem has been solved for all 52 weeks of the year. The stage subproblem is solved during the forward pass of SDDP.

Stage cost The expected stage cost is defined by the Bellman function (also referred to as the cost-to-go), $V_t(\mathbf{x}_t, \omega_t) = \min_{\mathbf{a}_t \in A_t}\{C_t(\mathbf{x}_t, \mathbf{a}_t, \omega) + \mathcal{V}_{t+1}(f_t(\mathbf{x}_t, \mathbf{a}_t, \omega_t))\}$.

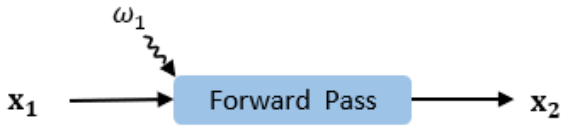

Figure 3.2: Graphic of Forward Pass where stage subproblem is solved

3.3 Improving the Expected Future Cost-To-Go Approximation

Recall from Section 2.3.2 in the backward pass of SDDP, cuts are produced in weeks 52,51,...,3,2. These cuts improve the expected future cost-to-go for the weeks 51,50,...,2,1. In JADE, the expected future cost-to-go $\mathcal{V}_t(\cdot)$ is the expected future cost in New Zealand dollars of operating an optimal policy from this week t to the end the year at week 52. The reservoir level at the end of the weekly subproblem and the realisation of the inflows are used in algorithm 2 to generate a cut to approximate the expected future cost-to-go function for the previous week.

Figure 3.3: Graphic interpretation of the backwards pass

3.4 Terminal Water Value

As mentioned in Section 2, all dynamic programs have a terminating expected cost-to-go, $\mathcal{V}_T$. The expected future cost-to-go function for weeks 1,2,....,51, $\mathcal{V}_{t+1}$ is approximated by a series of cuts. The terminating cost-to-go $\mathcal{V}_T$ is deterministic in JADE and represents the marginal value of water in the reservoirs at the end of the year (end of week 52). Without a terminal marginal value of water, JADE would have no incentive not to leave all reservoirs empty at the end of the year.

The terminal marginal value of water in JADE is an assumption of the model. It is an exogenous input to the model and does not depend on the nature of the NZEM. This is a significant assumption of JADE, and the extension of JADE and SDDP leading to an endogenous terminal marginal value of water was the objective of my research.

Shown in Figure 3.4, the terminal marginal value of water is a convex step function. The first 1000GWh of stored water has a value of \$137/MWh. The next 500GWh has a value of \$87/MWh and so forth. Note water associated with stored energies over 3500GWh have no value because it was assumed excess water would be spilled from reservoirs or used when the electricity price was zero.

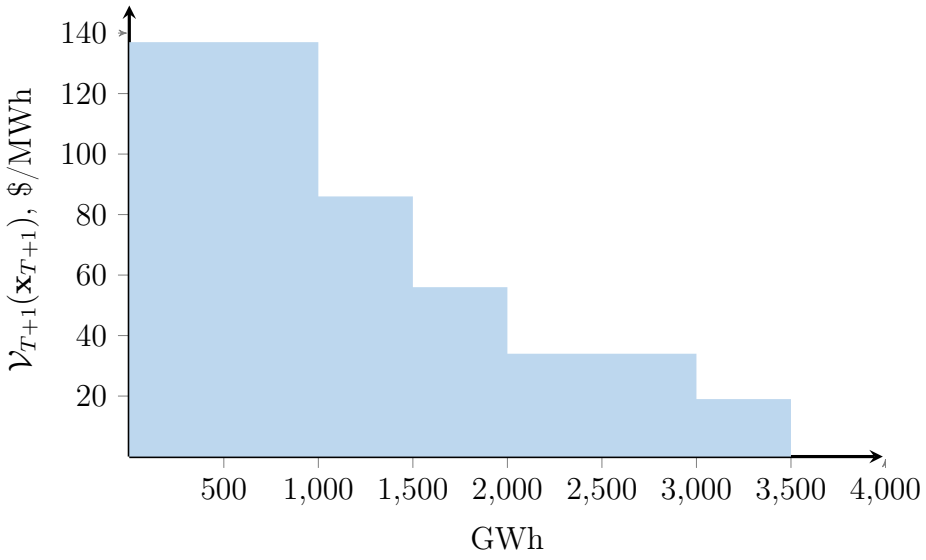

Figure 3.4: Terminal Marginal Water Value in JADE, $\mathcal{V}_{T+1}(\mathbf{x}_{T+1})$

3.5 Summary

Figure 3.5 ties in Figures 3.1, 3.2 and 3.3 to illustrate an iteration of SDDP on the JADE model at a high level. An iteration begins with the deterministic starting reservoir levels $\mathbf{x}_1$ which are passed to the subproblem in stage 1, represented by the yellow circle. The optimal action of stage 1 is determined when solving the subproblem in the forward pass represented by the blue rectangle by the text 'FP'. The random inflows ω_1, are realised at the start of the forward pass, demonstrated by the ω_1 connected to the top left corner of the blue rectangle by a squiggly line. The solution of the subproblem in stage 1 results in the new reservoir levels $\mathbf{x}_2$, which is passed as the input reservoir into the subproblem of the second stage. This process continues for weeks, 1,2,..., 51.

Once the forward passes are complete for weeks 1,2,...,51, the backward passes begin from the final week, week 52. In the backwards pass, a cut is generated that improves the approximation of the future cost-to-go function in the previous week.

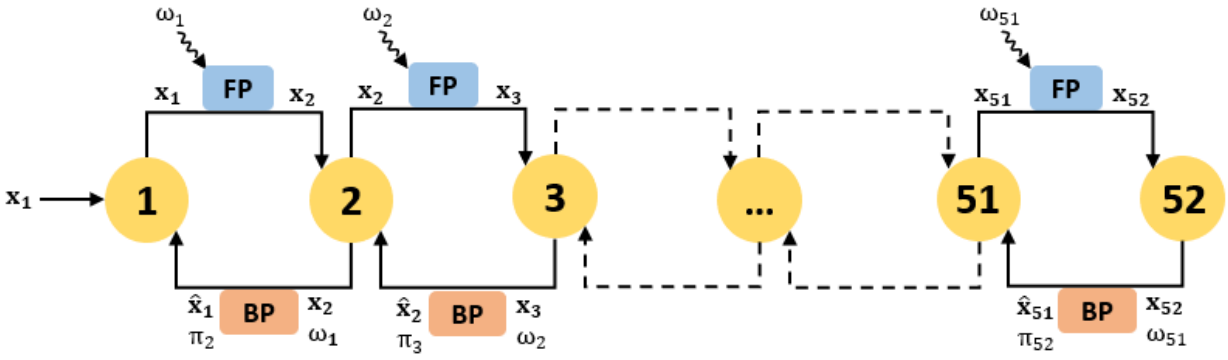

Figure 3.5: Graphic interpretation of SDDP algorithm in the context of JADE

4 Infinite-Horizon SDDP

The aim of my part of the project was to extend the model of the NZEM in JADE from a SDDP to an 'infinite-horizon' SDDP.

In SDDP, the terminal future marginal cost $\mathcal{V}_{T+1}(\mathbf{x}_{T+1})$ is fixed and determined by a predefined function. This function was displayed in Figure 3.4 which illustrates the terminal marginal future value of water in JADE. This estimated marginal value of water is a large assumption in the standard JADE model. For different configurations of the NZEM the terminal marginal value of water changes. The current method in JADE of using the same marginal value of water for modelling different scenarios in the NZEM is an assumption, and does not provide the most realistic model as the terminal marginal value of water would change under different scenarios. This issue is expanded on in the motivation section of the Literature Review and Statement of Research Intent accompanying this report [6].

In Section 2.4 and Section 2.5 the theory of infinite-horizon dynamic programming was discussed for discrete states. SDDP deal with problems with continuous states. AS mentioned in Section 2.4, SDDP was extended to an infinite-horizon SDDP using the method of 'average expected cost' because the 'average expected cost method' converges faster than the 'discounted cost' method.

The expected average cost method updates the terminal cost-to-go approximation via the following equation:

$$\mathcal{V}_{T+1}(\mathbf{x}) = V_1(\mathbf{x}) - \Delta, \qquad \forall \mathbf{x}. \tag{4.1}$$

The terminal marginal water value is no longer deterministic and depends on the configuration of the NZEM that we are modeling. In extending JADE to an infinite-horizon model, the terminal marginal water value becomes an endogenous part of the model by using equation (4.1). $V_1(\mathbf{x})$ is represented by a piecewise linear function built up from a series of cutting planes over iterations of SDDP as discussed in Section 2.3. Hence the new endogenous terminal marginal value of water is built up over successive iterations of the algorithm.

For the rest of this section, developments to SDDP in producing the infinite-horizon SDDP are discussed. These developments involved extending the SDDP.jl and JADE.jl packages.

4.1 Infinite-Horizon Forward Pass

In the standard JADE model, years (years≈iterations) are distinct in the sense that the state (reservoir levels of each of the seven reservoirs) resulting at the end of the year do not carry forward into the next year because the starting reservoir levels (the state) in an of JADE is deterministic and fixed. In SDDP each iteration begins from the same initial state. This initial state of each of the seven reservoirs is specified in the input files of JADE in reservoirs.csv. In SDDP if 5000 iterations are run, all 5000 iterations begin from the same state $\mathbf{x}_1$.

This means in JADE the model does not bear the direct consequence of the scenario where the model completely drains the reservoirs in one year resulting in high thermal costs in the next year because of a lack of water for hydro generation. In the infinite-horizon SDDP, the next year (iteration) starts where the previous year finished. This is what makes the model an 'infinite-horizon' as one year/iteration transitions smoothly into the next in a 'looping' mechanism.

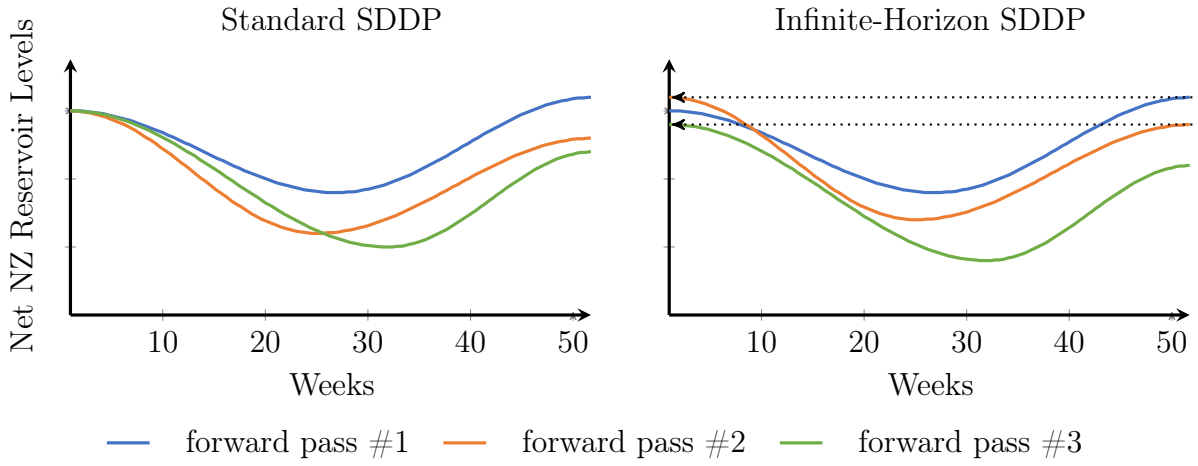

Figure 4.1: Net NZ reservoir levels over the successive forward passes

In the infinite-horizon SDDP, each the forward pass of a new iteration of SDDP begins where the forward pass of the previous iteration finished. More precisely, the initial state to the forward pass in stage 1, iteration j is equal to the state at the end of stage T in iteration *j-1* (iterations of the SDDP algorithm are referred to by j). Figure 4.1 shows SDDP iteration 2 beginning in the state where SDDP iteration 1 finished. Similarly, SDDP iteration 3 begins in the state where SDDP iteration 2 finishes.

Figure 4.2: Graphic displaying the continuation of state across iterations, $\mathbf{x}_1^j = \mathbf{x}_T^{j-1}$

By starting a new iteration of SDDP in the state where we finished the previous iteration, we can interpret successive iterations of SDDP in the infinite-horizon model as being part of one continuous or 'infinite' loop as shown in figure 4.2. This is contrasted to standard SDDP, discussed in Section 2.3 where all states are 'reset' to fixed reservoir levels in a new iteration.

4.2 Infinite-Horizon Backward Pass

Recall from Section 2.3 that in the backward pass cuts are generated and passed back to the previous stage, e.g. cuts are passed from stage $t+1$ to stage t. In infinite-horizon SDDP this process occurs as before with the addition of passing cuts from stage 1 to stage T as per equation (2.8) (replicated below) to build up the terminal future cost function.

$$\mathcal{V}_{T+1}^i(\mathbf{x}_{t+1}) = V_1^i(\mathbf{x}_{t+1}) - \hat{\delta}^i$$

Discussed in Section 2.3, cuts (from iteration j of SDDP, stage t) are linear functions characterized by a gradient $\vec{\beta}_t^j$, a y-intercept α_t^j and the state the cut was sampled at, $\mathbf{x}_t^j$. The cuts from stage 1 in iteration j must be shifted down by $\hat{\delta}$ as per the terminal future cost update function, equation (2.8). The cuts (in SDDP iteration j) are shifted down by $\hat{\delta}$ not Δ as Δ is defined to be the converged value of $\hat{\delta}$ (as $j \to \infty, \hat{\delta} \to \Delta$). The method of shifting the cuts down is done by subtracting the y-intercept, α_1^j, of the stage 1 cuts by $\hat{\delta}$. The gradients and shifted y-intercepts of these cuts are passed to the terminal future cost function $\mathcal{V}_{T+1}(\mathbf{x})$

thus updating $\mathcal{V}_{T+1}(\mathbf{x})$ and are calculated by equations (4.2) and (4.3). Determining $\hat{\delta}$ is an involved problem and is discussed later, in Section 4.4.

$$\vec{\beta}^{j}_{T+1} = \vec{\beta}^{j}_{1} \tag{4.2}$$

$$\alpha^{j}_{T+1} = \alpha^{j}_{1} - \hat{\delta} \tag{4.3}$$

In the context of JADE, as the number of iterations j of SDDP increases, $\hat{\delta}$ converges to Δ, the expected cost of the hydro-thermal scheduling over the year. By rearranging equation (2.8) into equation (4.4) (below) this becomes obvious.

$$\hat{\delta} = V_1(\mathbf{x}_{t+1}) - \mathcal{V}_{T+1}(\mathbf{x}_{t+1}) \tag{4.4}$$

No cuts are produced from the first stage of SDDP because cuts are produced in the backwards pass, and there is no backwards pass for stage 1. As cuts from stage 1 are required to be shifted and then passed to stage T a dummy stage 0 is introduced. As stage 1 is now no longer the 'first stage', cuts are produced at stage 1 and are passed to the dummy stage 0, shifted, then passed to stage T. Without the dummy stage 0, cuts would have to be shifted then passed from stage 2 to stage T which would create a 1-stage discrepancy. In the dummy stage 0 the objective function is set to a constant and in the context of JADE there is no demand, inflows, or change of reservoir levels. The dummy stage's purpose is solely to allow cuts to be generated in stage 1. As now the backwards and forward passes have been discussed, we now consider the infinite-horizon SDDP algorithm as a whole.

4.3 Infinite-Horizon SDDP Algorithm

Figure 4.3 illustrates an iteration of the infinite-horizon SDDP algorithm at a high level. First, note the arrow connected from week 52 to the dummy week which shows the passing of the reservoir levels at the end of week 52 to the initial reservoir levels of the dummy stage. Secondly, note the lack of a forward pass and subproblem between the dummy week 0 and week 1. This is because the dummy week 0 is just a placeholder for cuts from week 1 so the starting reservoir levels in the dummy week are the same as the final reservoir levels at the end of the dummy week. Finally note how the cuts π_1 from week 1 are passed to the final stage, week 52. This demonstrates the building up of the future cost-to-go function at week 52 which is a lower bound approximation of the terminal marginal water value.

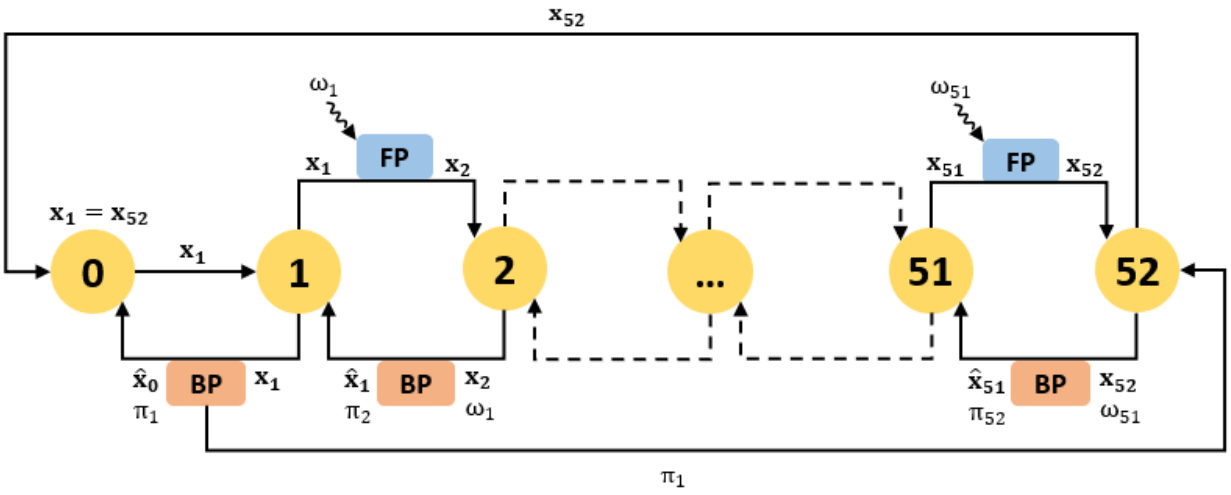

Figure 4.3: Graphic of an iteration of the infinite-horizon SDDP algorithm

The algorithm starts with no cuts at week 52 or any other week. This implies water in reservoirs at the end of the year has no value. Unsurprisingly, this results in the model emptying all reservoirs in the final weeks of the year. However, the future cost-to-go function in week 52 is built up from zero over successive iterations from the shifted cuts from the dummy week 0. With enough iterations, the future terminal cost-to-go function converges, which then leads to the algorithm's policy converging.

Figure 4.3 and its following description implies that the shifted cuts are passed to week 52, the final stage during the backwards pass of a given iteration of SDDP. However, for computational efficiency and to result in a more precise terminal future cost approximation, cuts were cached for 500 iterations of SDDP, then the corresponding $\hat{\delta}$ shift determined. The 500 stage 1 cuts are then shifted down by $\hat{\delta}$ and then passed to stage T. This methodology in the infinite-horizon SDDP algorithm (Algorithm 4) below.

So far, when referring to 'iterations' we have used both the indices i an j. These two types of iterations will be properly defined now and will also be refereed to in the later Section 5 and 6 to detail on convergence and computational improvements.

1. Iterations of SDDP consisting of a forward and backward pass are indexed by the superscript j. Iterations of SDDP are run in a 'for loop' (see Algorithm 4 below).
2. Iterations of the 'outer loop' of the infinite-horizon algorithm are indexed by the superscript i. In the 'outer loop' J iterations of SDDP are run, then the cached stage 1 cuts are shifted by $\hat{\delta}^i$ and then passed to stage 52. The 'outer loop' is a 'while loop' (see Algorithm 4 below).

Algorithm 4: Infinite-Horizon SDDP Algorithm

$i = 0$
$J = 500$
while *policy has not converged* **do**
  for $j = 1$ **to** J **do**
    if j == *1* **then**
      % Randomise initial reservoir levels
      storeState = random()
    end
    $\mathbf{x}_1^j$ = storeState
    SDDP Forward Pass
    storeState = $\mathbf{x}_T^j$
    SDDP Backward Pass
  end
  % Determine $\hat{\delta}^i$ (see algorithm 5)
  $\hat{\delta}^i = \min_x\{\mathcal{V}_1^i(\mathbf{x}) - \mathcal{V}_{T+1}^i(\mathbf{x})\}\ \forall \mathbf{x} \in$ sampled states
  % Update terminal future cost
  $\mathcal{V}_{T+1}^{i+1}(\mathbf{x}) = \mathcal{V}_1^i(\mathbf{x}) - \hat{\delta}^i$
  $i = i + 1$
end

4.4 Determining $\hat{\delta}$

$\hat{\delta}$ is the distance the y-intercept, α_1, of the stage 1 cuts are shifted down by before being passed to stage T. The reasoning for this shift will now be fully discussed and the exact algorithm used to calculate $\hat{\delta}$ is shown.

For convergence of the algorithm's policy, it is not necessary to shift the stage 1 cuts down before passing them to the final stage T, however by shifting the stage 1 cuts down the algorithm converges much faster. Not shifting the stage 1 cuts down will result in the stage 1 cuts passed to stage T dominating previous cuts hence making them redundant. The cuts present at stage T are valid and useful in the terminal future cost approximation. By making these cuts redundant, the algorithm will take much longer to converge than if the information of these cuts was used. Hence, new cuts from stage 1 are shifted down by $\hat{\delta}$. If $\hat{\delta}$ is too large, the new cuts will be *below* the current cuts defining $\mathcal{V}_{T+1}(\mathbf{x})$, hence will provide no new information to the terminal future cost approximation. If the $\hat{\delta}$ is too small the new cuts will dominate all the current cuts defining $\mathcal{V}_{T+1}(\mathbf{x})$. The $\hat{\delta}$ must be determined such that the shifted new cuts provided new information to the terminal future cost approximation while also not dominating all the current cuts in this approximation.

As discussed in Section 4.3, the stage 1 cuts are cached for 500 iterations of SDDP before determining the $\hat{\delta}$ and then passing the shifted cuts to stage T. The exact methodology for determining this $\hat{\delta}$ follows. We have 500 new cuts from stage 1 with each cut j having a gradient $\vec{\beta}_1^j$ and a y-intercept α_1^j, sampled from a state $\mathbf{x}_1^j$. The δ^j for each cut j is determined by finding the maximum distance between the new cuts from stage 1 approximating $\mathcal{V}_1(\mathbf{x})$ and the current cuts approximating $\mathcal{V}_T(\mathbf{x})$ at the sampled state $\mathbf{x}_1^j$.

Then $\hat{\delta}^i$ is determined by:

$$\hat{\delta}^i = \min \delta^j \tag{4.5}$$

Intuitively, this means $\hat{\delta}^i$ is the smallest distance between the dominating cuts of $\mathcal{V}_1^i$ and the $\mathcal{V}_{T+1}^{i-1}$.

The y-intercept of the 500 new cuts are shifted down by the $\hat{\delta}^i$ and passed to the expected future cost-to-go function at stage T by equation (4.3), replicated for convenience:

$$\alpha_{T+1}^i = \alpha_1^i - \hat{\delta}^j$$

Algorithm 5 (shown below) concisely brings together all the methodology discussed in Section 4.4. The result of the algorithm is the exact value for $\hat{\delta}^i$ for the given outer loop i of the infinite-horizon SDDP algorithm (Algorithm 4) which in JADE, once converged, leads to convergence of the hydro-thermal scheduling policy.

Algorithm 5: $\hat{\delta}$ Calculation algorithm

% In iteration i of 'outer loop'
% $J = 500$
% Hence we have 500 cached stage 1 cuts
for $j = 1$ **to** J **do**
  % cut $j = \alpha_1^j + \vec{\beta}_1^j$, sampled at $\mathbf{x}_1^j$
  $y^j = \max\{\alpha_1^c + \vec{\beta}_1^c \cdot \mathbf{x}^j\}$ for cuts c = 1,2,...,500 in new stage 1 cuts
  $\delta^j = \max\{y^j - (\alpha_1^k + \vec{\beta}_1^k \cdot \mathbf{x}^j)\}$ for cuts k = 1,2,...,K defining $\mathcal{V}_{T+1}$(**x**
end
$\hat{\delta}^i = \min_{\mathbf{j}\in\mathbf{J}}\{\delta^j\}$

5 Convergence

The previous section discussed the theory and algorithms developed for the implementation of the infinite-horizon SDDP algorithm. After I implemented these changes by extending the SDDP.jl and JADE.jl packages, the infinite-horizon algorithm was tested to see if it converged. A mathematical proof for convergence of the algorithm was out of the scope of this project but is included as future work, discussed in Section 8. Convergence of algorithm is important for ensuring accuracy and confidence in produced results. In the context of JADE, convergence of the algorithm means the convergence of a hydro-thermal scheduling decision policy.

There are several methods of convergence criteria used to determine convergence of the SDDP algorithm. According to Dowson, 2018 [11], running for a fixed time limit or a fixed number of iterations and then simulating the model to see if poor decisions are made is a preferred method of testing for convergence compared to statistical stopping rules. Previous work with JADE has found >3000 iterations causes sufficient convergence.

However, as JADE uses an exogenous deterministic terminal marginal water value and our infinite-horizon model uses an endogenous terminal marginal value of water that is developed over successive iterations, we used 8000 iterations of SDDP for generating the hydro-thermal scheduling policy.

Our exact method used 15 updates of $\mathcal{V}_T(\mathbf{x})$ (I=15) with 500 iterations of SDDP between updates (J=500). Hence for SDDP iterations 1,2,...7499,7500 updates of $\mathcal{V}_T(\mathbf{x})$ occurred every 500 iterations of SDDP (e.g. 500,1000,...,7000,7500).

Several other convergence criteria were checked, to convince ourselves the algorithm had converged. These convergence criteria are discussed below.

5.1 $\hat{\delta}$ Convergence

$\hat{\delta}$ is the distance to shift the new stage 1 cuts down before passing the cuts to stage T. As the number of updates of $\mathcal{V}_T(\mathbf{x})$ increases (recall $\mathcal{V}_T(\mathbf{x})$ is updated every J iterations of SDDP) $\hat{\delta} \to \Delta$, where Δ is the converged value, the expected cost accrued of operating an optimal hydro-thermal scheduling policy of over the given time horizon.

A plot of the standard deviation of δ^j over updates of $\mathcal{V}_T(\mathbf{x})$ (occurring every 500 iterations of SDDP as $J = 500$) is shown below in Figure 5.1. The standard deviation of $\hat{\delta}$ appears to have converged. Note the magnitude of the standard deviation is large, but this is because δ^j's are mostly greater than 10^9.

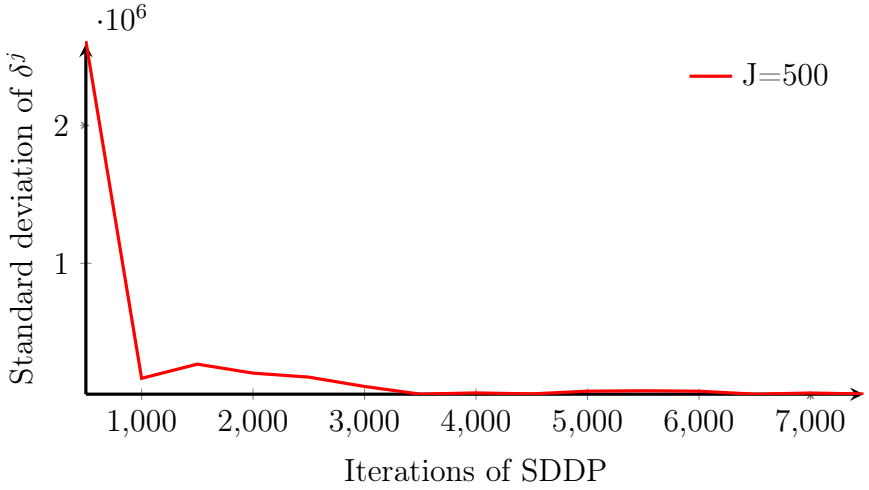

Figure 5.1: Plot of the convergence of δ^j for $J = 500$

5.2 Terminal Future Expected Cost-To-Go Integral Convergence

An approximation of the terminal future expected cost-to-go, $\mathcal{V}_{T+1}(\mathbf{x})$ was also used as a criterion for convergence. Integrating the volume under the terminal future expected cost-to-go function would give a new stopping criterion that was not mentioned in Dowson's thesis [11].

After the terminal future expected cost-to-go has converged according to some bound this implies the algorithm has converged. As an integration over thousands of superimposed 7D linear functions is computationally infeasible, a 1D linear approximation of the 7D linear functions was made, and then the area under this set of 1D linear functions was determined. The algorithm used to determine this numerical integral (an implementation of the 'rectangle rule') can see seen in Appendix II. This area was calculated every J iterations of SDDP (where $J = 500$) because $\mathcal{V}_T(\mathbf{x})$ is updated every J iterations of SDDP. This criterion was found to quickly converge within 2500 - 3000 iterations, much earlier than the previous iteration stopping criterion at 7500 iterations. The precise scale on the y-axis of Figure 5.2 demonstrates how the integral converges quickly (between 2.0794×10^{10} and 2.0795×10^{10}).

Since cuts are only added to the future expected cost-to-go functions (for any state including the terminal stage), the integral can only every increase, so if two iterations (of the outer loop) have the same value then the functions are the same (however, this doesn't guarantee convergence, since a different random inflow sequences can find a new cut). This criterion is also a confirming indicator that $\hat{\delta} \to \Delta$ as if $\hat{\delta}$ has not converged, it will continue to increase and with it the integral of $\mathcal{V}_{T+1}(\mathbf{x})$.

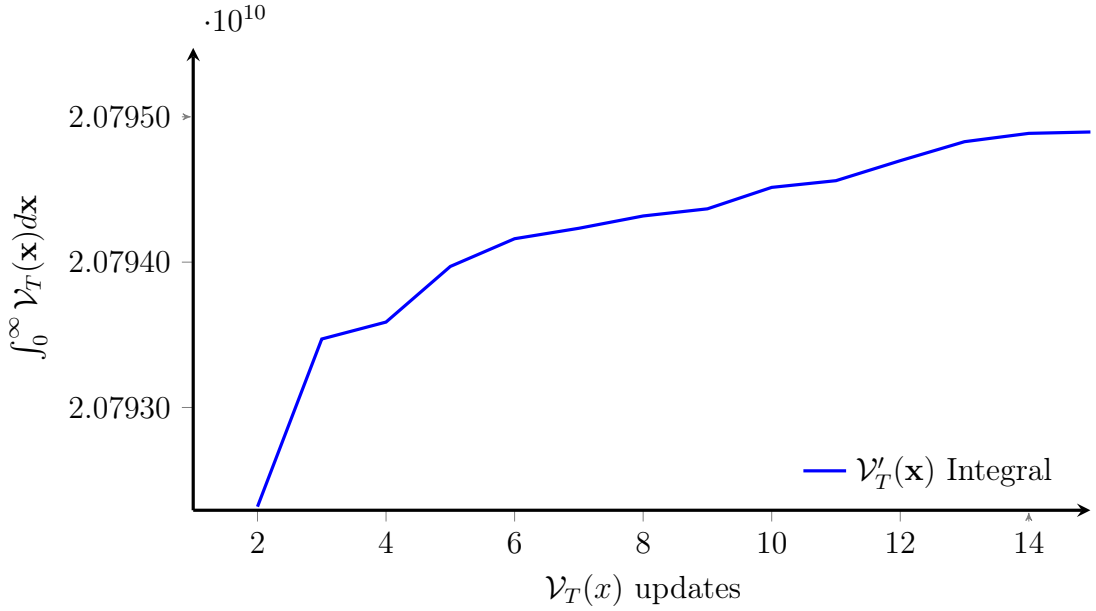

Figure 5.2: Plot of convergence of numerical integral of $\mathcal{V}'_T(\mathbf{x})$

5.3 Terminal Marginal Cost Function Convergence

The convergence of the endogenous future marginal cost in the final stage, $\mathcal{V}_{T+1}(\mathbf{x})$, is another criterion that can be used to determine convergence of the infinite-horizon SDDP algorithm. As $\mathcal{V}_{T+1}(\mathbf{x})$ is 7D in JADE we approximated it as a 1D function by a weighted average based on the energy per m^3 in each reservoir. We then plotted the 1D approximation of $\mathcal{V}_{T+1}(\mathbf{x})$ on for every iteration i of the outer loop which corresponds to on every update of $\mathcal{V}_{T+1}(\mathbf{x})$.

Figure 5.3 shows the convergence of the terminal marginal water value for the current scenario of the NZEM. Note how in the legend that Iterations 14 and 15 are coloured blue and red respectively but in Figure 5.3 a pink line is seen. This means the lines are directly on top of one another and the 1D approximation of the marginal water values are identical (implying convergence) for iterations 14 and 15.

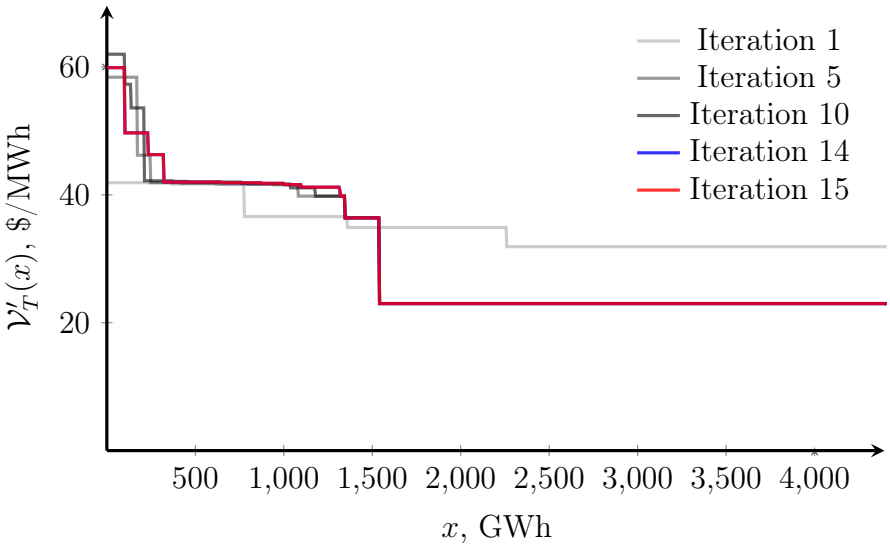

Figure 5.3: Plot of convergence of terminal marginal water value, $\mathcal{V}'_T(x)$

The convergence of this criterion implies that the policy of the of the problem has converged. This is because the marginal water values for a given set of reservoir levels informs the hydro-thermal scheduling decision.

To summarise this section so far of all three criteria have converged. The convergence of these criteria implies convergence of the infinite-horizon SDDP algorithm. The rest of this Section discusses how variation of the user chosen parameter J, (insignificantly) affects our three convergence criteria.

5.4 Expected Terminal Future Cost-To-Go Update Frequency

We chose to cache stage 1 cuts for 500 iterations of SDDP before determining the $\hat{\delta}$ to shift the new stage 1 cuts down by (then update $\mathcal{V}_{T+1}(\mathbf{x})$. As J is the number of iterations of SDDP to run per 'outer loop' this means we have been setting $J = 500$. This choice for J was chosen from the supervisor of this project's specialised knowledge of JADE and SDDP. Choices such as caching the stage 1 cuts for 200 or 1000 iterations of SDDP (or anything in-between) before determining the $\hat{\delta}$ may result in faster convergence of $\hat{\delta} \to \Delta$. Inspection on how the convergence of the other two criteria with the different choices of J is also of interest.

The infinite-horizon model was run with J = 100, 200, 320, 400, 500, 615, 800, and 1000. The total number of iterations pf SDDP was kept constant at 8000 iterations. Hence the number of iterations of the outer loop (I) also varied.

5.4.1 Terminal Future Expected Cost-To-Go Integral Convergence

The terminal cost integral converged for all cases. Recall from 5.2 that this implies the expected future terminal cost-to-go function has converged. However, the value of the terminal marginal integral for each case was different. This was expected because of initial (bad) values for $\hat{\delta}^i$ shift the expected future terminal cost-to-go function to different heights.

On average, there was an inversely proportional relationship between the value of the terminal marginal cost integral and the number of iterations J of SDDP per outer loop. This is expected because the cases with smaller values of J underestimate $\hat{\delta}$. This is because for smaller values of J there are less new stage 1 cuts generated per 'outer loop'. Hence, the 7D 'surface' produced by the smaller set of stage 1 cuts will be more 'patchy', with places where there are not many cuts. The distance between the 'patchy' area's of the new stage 1 cuts 7D 'surface' and the current terminal cost-to-go function will be small. Recalling from Section 4.4, $\hat{\delta}^i$ is the smallest distance between the dominating cuts from $\mathcal{V}_{T+1}^{i-1}$ and $\mathcal{V}_1^i$ at the sample points so for a small J a small value of $\hat{\delta}^i$ is more likely.

5.4.2 Convergence of $\hat{\delta}^j$

The same problem is being solved for the various value of J so, $\hat{\delta}$, an estimate of the expected accrued cost should converge to the same value for all values of J if our model is correct. As the number of iterations of iterations of SDDP increases, $\hat{\delta} \to \Delta$. $\hat{\delta}$ was observed to converge for all values of J as demonstrated in Figure 5.4.

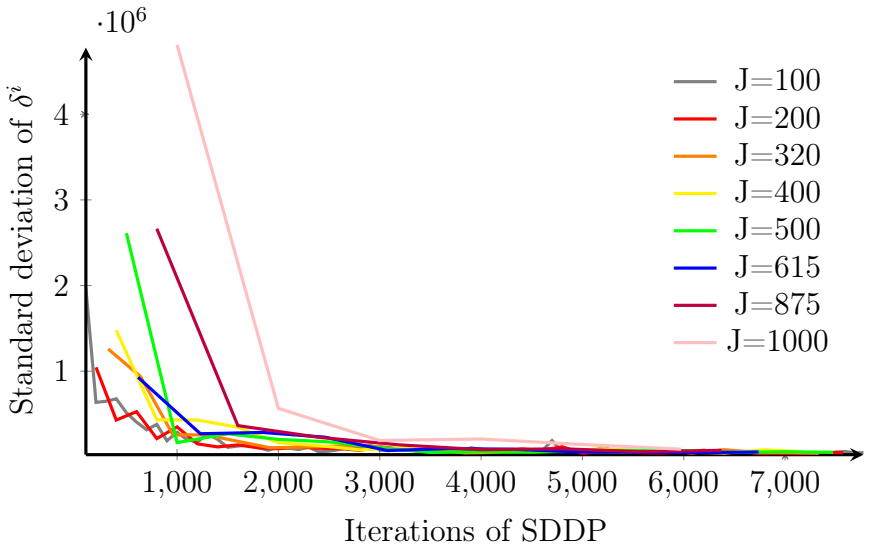

Figure 5.4: Plot of the convergence of δ^j for various values of J

For all cases of J the final set of δ^j converged to the narrow range of 3.577×10^9 - 3.600×10^9. The final value of $\hat{\delta}^i$ over all cases of J was in the narrow range of 3.5770×10^9 - 3.5784×10^9. Convergence of $\hat{\delta} \to \Delta$ demonstrates the correctness of my implementation and gives confidence in results produces from the model.

To summarise the testing of different values of J, the testing provided additional evidence the algorithm converged and showed the algorithm is not meaningfully sensitive to the most significant user-defined parameter. It also implied a 'best choice' for J. The choice of $J = 500$ (the original choice) resulted in the smallest standard deviation for $\hat{\delta}^j$.

6 Computational Improvements

Section Section 4 discussed the theoretical methodology of the infinite-horizon SDDP. Section 5 demonstrated the algorithm's convergence through the simultaneous convergence of three metrics. This section discusses the three computational speed improvements that reduced the solve time of the initial infinite-horizon SDDP model from more than 18 hours to 30 minutes after all three speed improvements were implemented. These computational improvements enable the JADE hydro-thermal scheduling model to be used at an entirely new scale, allowing deeper analysis and understanding of the NZEM.

1. The first computational improvement involved decreasing the problem size by half.
2. The second computational improvement involved parallelising the infinite-horizon algorithm. This improvement reduced the runtime to 3.5-4 hours when using a 16-core virtual machine at 2.60GHz and 64GB of RAM.
3. The third computational improvement was the implementation of a cut selection heuristic which reduced the number of cuts in the subproblem which meant subproblems could be solved faster. This improvement reduced the runtime to 30 minutes when using a 16-core virtual machine at 2.60GHz and 64GB of RAM with the cut selection heuristic.

6.1 Initial 'Hot Started' Infinite-Horizon Algorithm Implementation

Initially, a simpler model of infinite-horizon SDDP algorithm was developed to get simulation results to allow my project partner Ben Fulton to develop the excel macros for analysing simulation output while I developed the infinite-horizon SDDP algorithm discussed in Section 4 (algorithm 4).

This simple version of the infinite-horizon SDDP algorithm involved a 104 week (two-year) model of the NZEM. Cuts from week 52 were cached for 500 iterations of SDDP then used to update the exogenous terminal marginal water value function. The SDDP algorithm was then restarted with this new exogenous terminal marginal water value function. Restarting the SDDP algorithm involves throwing away all cuts previously generated. Hence, this method was much slower than the second version of the infinite-horizon SDDP algorithm discussed in Section 4, taking upwards of 18 hours to converge to a stable terminal marginal water value function. The term 'roughly converge' is used, because more stringent convergence criteria were later used.

When the two-year model was simulated (carried out by Ben Fulton), only the second year of simulation results was used in the analysis because the intention was to analyse only one year of the NZEM. However, the first year in the two-year model was needed to develop the terminal marginal water value function and produce a distribution of reservoir levels at the start of year 2. For completeness, the initial infinite-horizon SDDP algorithm (6) is shown below.

The development from algorithm 6 to algorithm 4 halved the problem size from a 104 stages to 52 stages. However as cuts were no longer being thrown away, the individual subproblems became much larger. This led to the convergence of the algorithm taking at least 40 hours however much stronger convergence criteria (discussed in Section 5) were now used.

Algorithm 6: Initial Infinite-Horizon SDDP Algorithm

$i = 0$
$J = 500$
storeState = random()
$\mathcal{V}_{105}(\mathbf{x}_{105}) = 0$
while *policy has not converged* **do**
  Throw all away all cuts approximating $\mathcal{V}_t(\mathbf{x}_{t+1})$ $\quad \forall$t
  for $j = 1$ **to** J **do**
    $\mathbf{x}_1^j$ = storeState
    SDDP Forward Pass
    storeState = $\mathbf{x}_T^j$
    SDDP Backward Pass
  end
  $\mathcal{V}_{105}(\mathbf{x}_{105})$ = 1D Approximation of $\mathcal{V}_{52}(\mathbf{x}_{t+1})$
  $i = i + 1$
end

6.2 Parallel Processing

Additionally, my developments to the SDDP.jl and JADE.jl packages work with the parallelism features of SDDP.jl. This method works by running multiple 'slave' copies of the algorithm which pass and receive cuts to a 'master' copy of the model. At the end of each iteration of SDDP, the 'slaves' pass all their cuts to the 'master' process and receive new cuts discovered by other 'slave' processes. The master and slave processes each run on their own core and given the availability of a 16-core Virtual Machine, this extension reduced the solve time significantly by an order of magnitude compared to the previous implementation running on only 1 processor for the same convergence standard. The parallelised Infinite-Horizon algorithm achieved convergence after 3.5-4 hours using a 16-cores running at 2.6GHz, with 64GB of memory. The reduction in time to convergence from >40 hours to 3.5-4 hours by the parallelism of the JADE hydro-thermal scheduling model now allowed many different scenarios of the NZEM to be modelled and simulated. Since the convergence criteria are now much higher, we are confident in the accuracy of our results.

6.3 Cut Selection

Many cuts (typically thousands) are added to each weekly subproblem as the SDDP algorithm progresses. This computational load causes subproblems to take longer to solve. However, many of the added cuts may be completely dominated and are hence redundant. Using a cut selection heuristic, sub-problems can be rebuilt using a subset of the given sub-problems' present cuts. I implemented two cut selection algorithms introduced by Matos et al. in 2015 [4].

6.3.1 Level 1 Cut Selection

Recall from Section 2.3 that cuts (from iteration i, stage t) are linear functions characterized by a gradient $\vec{\beta}_t^i$, a y-intercept α_t^i, and a stage $\mathbf{x}_t$ where the cut was sampled at. Given a set of N cuts at stage t we say cut k is dominated if for every $\mathbf{x}_t$ that is feasible for the stage problem there is at least one $n \neq k$ with:

$$\alpha_t^k + \vec{\beta}_t^k \cdot \mathbf{x}_t \leq \alpha_t^n + \vec{\beta}_t^n \cdot \mathbf{x}_t \tag{6.1}$$

As it is computationally infeasible to sample for all $\mathbf{x}_t$, heuristics are used. The 'Level 1 Cut Selection Algorithm' selects the dominating cuts at the sampled states $\mathbf{x}_t$ of the set of cuts to be the Level 1 dominating cuts. This is demonstrated in Figure 6.1.

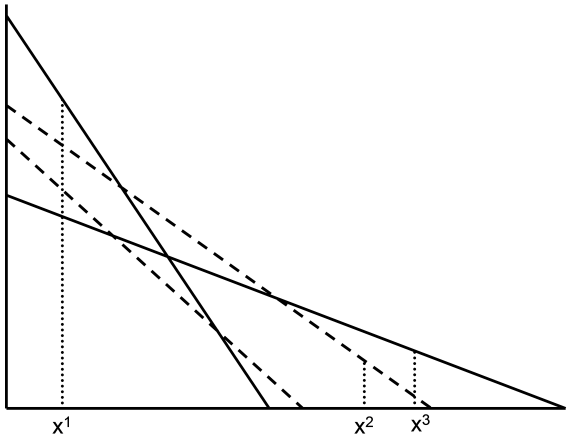

Figure 6.1: 1D representation of Level 1 Cut Selection Heuristic [4]

In Figure 6.1, the lower dashed cut is dominated. The remaining cuts are computed at points $\mathbf{x}^1$ (upper left solid), $\mathbf{x}^2$ (upper dashed), $\mathbf{x}^3$ (lower right solid). The upper dashed cut is not dominated by the solid cuts but would not be included in a Level 1 dominance selection as it is not the highest cut at $\mathbf{x}^1$, $\mathbf{x}^2$ or $\mathbf{x}^3$. The upper-left solid cut is the highest cut at $\mathbf{x}^1$ and the lower-right solid cut is the highest cut at x^2 and x^3, so they are both Level 1 dominating cuts (paraphrased from Matas et al. [4]).

Algorithm 7: L1 Cut Selection Algorithm

```
% For a set of N cuts, determine the Level 1 dominating cuts
nondomIndices = zeros(N)
for s = 1 to N do
    % Determine the dominating cut n at state s
    Y^n = α_t^n + β⃗_t^n · x_t^s    ∀n
    % Record the index of the dominating cut at state s
    nondomIndices[s] = arg max{Y^n}
end
Return the dominating cuts, from the unique indices of nondomIndices
```

I implemented the L1 cut selection strategy into JADE. The heuristic was applied in the outer loop of the infinite-horizon SDDP algorithm and sped up the model by an order of magnitude reproducing the speed improvements found by Matos et al. when using the L1 cut selection heuristic. [4]. Using the L1 cut selection heuristic with 8000 iterations of SDDP in the infinite-horizon algorithm ran in 25-40 minutes while without the cut selection heuristic the algorithm took 3-3.5 hours to solve (both cases used a parallel processor implementation with 16 cores at 2.60GHz).

6.3.2 Level H Cut Selection

The Level 1 cut selection strategy only selects the 'best' cut at each sampled state $\mathbf{x}^j$. Hence many cuts that would be binding are not selected as the $\mathbf{x}$ they are binding for is not in the set of sampled state. Selecting the 'best' and 'second best' cuts for each $\mathbf{x}^j$ was also proposed

by Matos et al [4]. This method can be extended to the general case where the H highest cuts at each sampled point $\mathbf{x}^j$ is selected. This method is called the Level H Dominance strategy. I also implemented the Level H Dominance cut selection strategy into JADE. The heuristic was applied to the infinite-horizon SDDP algorithm by the same method as L1 cut selection heuristic. As observed by Matos et al., similar performance was produced with the Level 1 and Level H (for a variety of choices of H) cut selection heuristics. The Level H cut selection algorithm can be seen in Appendix I.

6.3.3 Number of Cuts Selected by the L1 Cut Selection Heuristic

The number of cuts selected by the L1 cut selection heuristic increases with the number outer loops of the infinite-horizon SDDP algorithm. Figure 6.2 shows a clear linear relationship between the iteration number of the outer loop and the number of cuts selected by the L1 cut selection heuristic for the intervals $I \in \{1, 2, 3\}$ and $I \in \{3, 4, ..., 13, 14\}$.

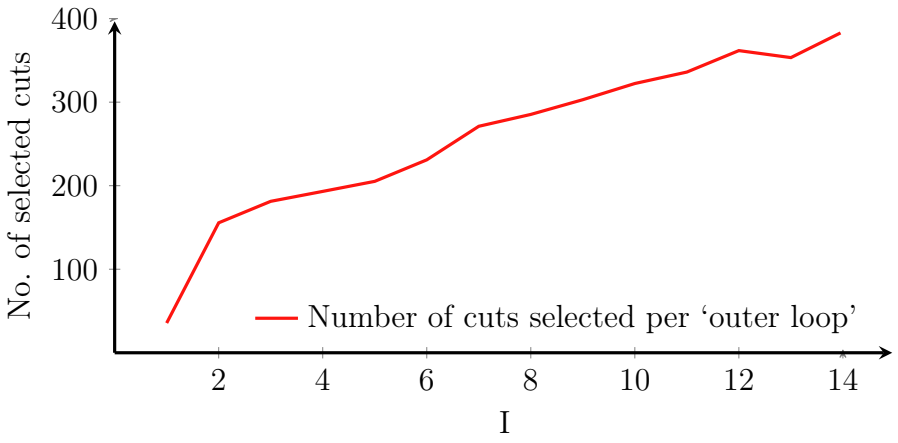

Figure 6.2: Plot of number of L1 dominating cuts selected per outer loop

6.3.4 L1 Cut selection heuristic across a larger range of sampled points

In the Level 1 (and Level H cut) selection heuristics, the L1 dominating cuts are found using only the set of states the set of cuts were sampled at. For example, if the L1 cut selection heuristic is applied to a set of 2000 cuts, only 2000 states are used to determine the L1/LH dominating cuts. This may not be enough sampled states to produce an effective set of dominating cuts. By sampling over more states (selected randomly from an appropriate distribution) may result in faster convergence. Given the L1 and LH cut selection heuristics compute quickly, this method will determine a larger set of dominating cuts with a marginal on the algorithm's runtime.

Extra sample states were chosen by sampling the 7D state uniformly between the lower bound (ub) and upper bound (ub) for each dimension d by:

$$U_i \sim U(0, 1), \qquad \mathbf{x}_i^d = \text{lb}^d + (\text{ub}^d - \text{lb}^d) * U_i \qquad \forall d \tag{6.2}$$

Using an extra 2000 sample states did not adversely affect the run-time of the algorithm because the L1 cut selection heuristic was optimised for efficiency. Use of the extra 2000 sample states appeared to result in faster convergence for the first 2000 iterations of SDDP, however, after that the extra 2000 sample states did not make a difference and all convergence metrics were equivalent.

7 Markov Inflows

An assumption of JADE is to assume inflows are stagewise independent. In reality, this is not the case as weather patterns persist. This is a particular concern when modelling the NZEM because of the dependence on hydro-generation. Droughts are a threat to the security of supply in New Zealand, and a drought is an example of a persistent state of weather. The assumption of stagewise independence of JADE is noted to produce overly optimistic policies [13] and hence introducing stagewise-dependent inflows using a Markov chain which better models reality.

For this model we considered a simple Markov chain of two climate states, *wet* and *dry*. If we are in a *wet* state in a given week, we are more likely to be in a *wet* state in the next week. The converse is true if we are in a dry state. Dry spells are less common than the standard wet weather in New Zealand, so we assume every year, in week 1 we start in a *wet* state.

The probability of transition between a wet and dry state is defined by the Markov transition matrix (example below in Table 1).

		To	
		Wet	Dry
From	Wet	0.64	0.36
	Dry	0.36	0.64

Table 1: Markov Transition matrix for week 1

A *wet* week was defined by the inflows being greater than or equal to the median historical inflows for a given week. A *dry* week was defined by the complement, if the given week's inflows were lower than the median historical inflows for this week. The Markov transition matrix for each week was determined by the historical occurrence of the transitioning from an inflow state in week i to the inflows state in the next week, week $i+1$. Inflow data from 1986 to 2013 (28 data points for each week) was used to determine the 2×2 Markov transition matrix for each week.

If the given week i, was in a *wet* state, week i's inflows were sampled from the subset of historical weekly inflows where these inflows were greater than or equal to the median inflows for the given week i. Similarly, if the given week i, was in a *dry* state, week i's inflows were sampled from the subset of historical weekly inflows where these inflows were less than median inflows for the given week i.

The JADE model with Markov inflows was solved with the infinite-horizon algorithm. The modelled scenario was the current situation in the NZEM with two out of the four coal-fired units at units available to the market. The converged value for $\hat{\delta}$ was considerably higher when using the Markov JADE model (4.37×10^9) than the standard JADE model with stagewise independent inflows (3.57×10^9). As $\hat{\delta}$ is a proxy for the expected cost accrued over a year, this result is not surprising as the persistence of a low inflow state (i.e a drought) requires more thermal generation which is costly.

The marginal water values (for week 1) are shown for the standard inflow and Markov inflow model in Figure 7.1. There are no major differences between the marginal water values for the standard JADE model, and the Markov model in the wet and dry states respectively. However, when looking at the marginal water values when the stored hydroelectric energy is low, we can see that the marginal water values from the Markov model are higher than the marginal water values from the standard model for both the wet and dry state respectively. After the stored hydroelectric energy reaches 400GWh, the marginal water values for all three functions are very similar.

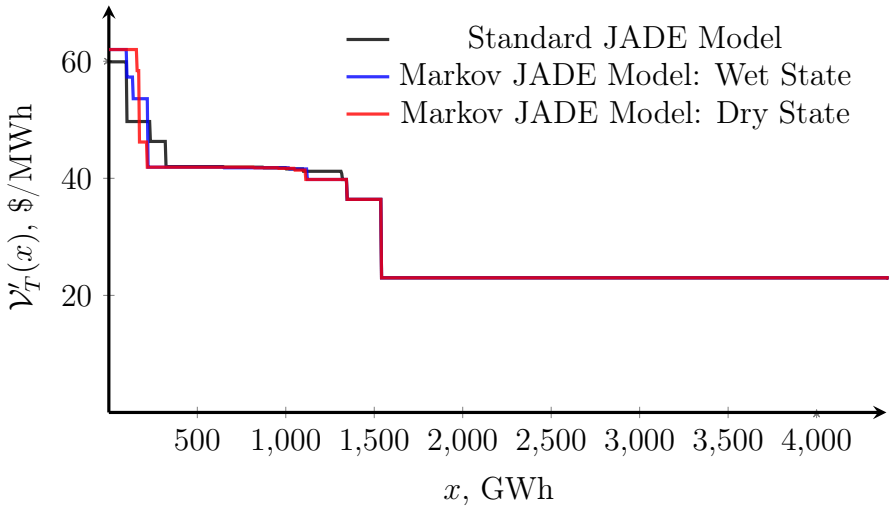

Figure 7.1: Plot of converged marginal water values, $\mathcal{V}'_T(x)$

Our results from the Markov model demonstrate that stagewise dependency has a small effect on the marginal water values, and a more significant effect on the expected accrued cost. Inflows to reservoirs are stagewise dependent. Hence, future work to develop a more valid stagewise dependent model (compared to our simple 2-state Markov model) is an important development for JADE, to give confidence in results produced.

8 Future Work

Develop Stagewise Dependent Inflow Model The Markov model used in Section 7 to model the stagewise dependency of inflows is too simple. Developing the two inflows states (*high* and *low*) by whether the weekly inflows were above or below the median was carried out so the Markov model could be developed quickly and some insight could be seen on how introducing stagewise dependence affected the hydro-thermal scheduling policy. A more suitable inflow model would be an auto-regressive integrated moving average inflow model. Development of an effective stagewise dependent model of inflows is important to resolve the stagewise independence assumption in JADE, develop a more accurate model of the NZEM and give practitioners increased confidence in their results.

Complete Integration of our Infinite-Horizon Method into SDDP.jl Currently, the infinite-horizon SDDP algorithm runs iterations of the standard SDDP algorithm in an 'outer loop' as demonstrated in Section 4.3. The actual implementation in Julia involves the calling of the SDDP.jl **Solve** function every iteration of the infinite-horizon outer loop. Integrating the infinite-horizon SDDP algorithm into SDDP.jl by having a 'flag' that is set in the SDDP.jl **Solve** function that tells the algorithm to apply the infinite-horizon SDDP algorithm would be an important development for the adoption of the infinite-horizon algorithm by practitioners. Practitioner adoption of the infinite-horizon algorithm for the application of hydro-thermal scheduling is important in the NZEM (and other hydro-dominated electricity markets) for increasing the accuracy and confidence in produced results. More generally, as stochastic programming and decision making under uncertainty in general gains traction in the optimisation community, a high-level implementation of the algorithm is required.

Parallel Initialization Speed Up The initialisation of the 16 parallel cores (uses the run the algorithm) takes 28-40 seconds (33 seconds on average). The cores are initialised every outer loop of the infinite-horizon algorithm. Hence for running the algorithm on 16 cores (using the L1 cut selection heuristic) for 15 iterations in the outer loop, and 500 iterations in the inner loop, results in a 30-minute runtime with approximately 8 minutes (27%) spent initializing the parallel cores. Further work to increase the speed of the initialisation shows promise since the initialization of the cores has not been 'optimized' and is a large component (27%) of the runtime.

Proof of Convergence All of our experiments have shown convergence of three convergence metrics; the lower bound, $\hat{\delta}$ and the 1D approximation of the terminal future cost-to-go function, $\mathcal{V}_{T+1}(\mathbf{x})$. However, a proof for the general case is required to ensure the general algorithm converges, to give practitioners applying the SDDP algorithm to other problems confidence in their results.

9 Conclusions

This report presents the implementation of an infinite-horizon stochastic dual dynamic program with the application to the New Zealand hydro-thermal scheduling problem. The implementation extended a current stochastic dual dynamic programming model of the NZEM, the JADE.jl Julia package. Development on the SDDP.jl (a stochastic dual dynamic programming package) was also carried out.

The infinite-horizon SDDP generates a more realistic policy of optimal hydro-thermal scheduling in the NZEM because it is a more accurate model than the standard SDDP model of the NZEM with fewer assumptions. In particular, the infinite-horizon SDDP resolves the assumption in SDDP of a fixed exogenous end of horizon marginal value of water function.

Three performance improvements were successfully implemented that decreased the solve time of the algorithm immensely. First, using an endogenous terminal future cost-to-go instead of a hot-started model reduced the size of the problem by half and resolved the inefficient method of throwing away cuts each time the model was hot-started. Parallel processing development allows the user to take full advantage of their available computing power. Finally, cut selection heuristics reduced the solve time by an order of magnitude by only adding the most important cuts to the stage subproblems. The result of the three performance improvements using a 16 core, 2.60GHz virtual machine, was a solve time of the infinite-horizon JADE model in 30 minutes, compared the first model which took > 18 hours.

The JADE hydro-thermal scheduling model was then extended to model reservoir inflows as stagewise dependent (previously inflows were stagewise independent) using a two-state Markov model. This extension extended the complexity of the model resulting in a solve-time to X minutes.

Given this fast algorithm, my project partner, Ben Fulton's report [5] describes the outcomes in different scenarios of the NZEM associated with the converged polices.

In conclusion, the implementation of an infinite-horizon SDDP, with the application to the New Zealand hydro-thermal scheduling problem is an important development the JADE model to and the SDDP.jl package, and to our knowledge is the first implementation of its kind.

References

[1] A. B. Philpott and Z. Guan, "On the Convergence of Stochastic Dual Dynamic Programming and Related Methods," *Oper. Res. Lett.*, vol. 36, no. 4, pp. 450–455, Jul. 2008, ISSN: 0167-6377. DOI: 10.1016/j.orl.2008.01.013. [Online]. Available: http://dx.doi.org/10.1016/j.orl.2008.01.013.

[2] L. Kapelevich, "About JADE," The University of Auckland, Tech. Rep., 2017.

[3] O. Dowson, "SDDP.jl: a Julia package for Stochastic Dual Dynamic Programming," The University of Auckland, Tech. Rep., 2017.

[4] V. L. de Matos, A. B. Philpott, and E. C. Finardi, "Improving the performance of stochastic dual dynamic programming," *J. Comput. Appl. Math.*, vol. 290, no. C, pp. 196–208, Dec. 2015, ISSN: 0377-0427. DOI: 10.1016/j.cam.2015.04.048. [Online]. Available: http://dx.doi.org/10.1016/j.cam.2015.04.048.

[5] B. Fulton, "Security of Supply in the New Zealand Electricity Market," The University of Auckland, 2018.

[6] S. Foster, "Literature Review and Statement of Research Intent," The University of Auckland, Tech. Rep., 2018.

[7] E. S. Department, *Dynamic programming notes part 4*, 2018.

[8] ——, *Stochastic programming notes*, 2018.

[9] S. F. Tony Downward Ben Fulton, *Computing the impact of changes to New Zealand's generation mix on hydro-reservoir management*, Sep. 2018.

[10] J. Simmonds, "Stochastic Supply Chain Modelling in Julia," University of Auckland, Sep. 2017.

[11] O. Dowson, "Applying Stochastic Optimisation to the New Zealand Dairy Industry," University of Auckland, 2018.

[12] M. V. F. Pereira and L. M. V. G. Pinto, "Multi-stage stochastic optimization applied to energy planning," *Mathematical Programming*, vol. 52, no. 1, pp. 359–375, May 1991, ISSN: 1436-4646. DOI: 10.1007/BF01582895. [Online]. Available: https://doi.org/10.1007/BF01582895.

[13] A. Philpott and G. Pritchard, *EMI-DOASA*, 2017.

Appendices

Appendix I: Level H Cut selection Algorithm

Algorithm 8: Level H Cut Selection Algorithm

% For a set of N cuts, determine the Level H dominating cuts
% Matrix of N rows, H columns
nondomIndices = zeros(N,H)
for $s = 1$ **to** N **do**
    $Y^c = \alpha_t^c + \vec{\beta}_t^c \cdot \mathbf{x}_t^s \quad \forall c$
    yMax = ∞
    for $h = 1$ **to** H **do**
        % Determine the level h dominating cut at state s
        index = $\arg\max_{Y^c < \text{yMax}}\{Y^c\}$
        yMax = $\max_{Y^c < \text{yMax}}\{Y^c\}$
        % Record the index of the dominating cut at state s
        nondomIndices[s,h] = c
    end
end
Return the dominating cuts, from the unique indices of nondomIndices

Appendix II: 1D Approximation of Numerical Integral

Recall from Section 2.3 cuts are defined by a 7D gradient $\hat{\beta}_t$, a y-intercept α_t, and the state the cut was sampled at, $\mathbf{x}_t$.

Algorithm 9: 1D Approximation of Numerical Integral of $\mathcal{V}'_T(\mathbf{x})$ Algorithm

% Have N cuts approximating the expected future cost-to-go function
% Determine the stored energy in each of the 7 reservoirs by multiplying the reservoir levels (the state $\mathbf{x}_t$) by the specific power. The specific power for a given reservoir is the amount of energy (MWh) produced from 1m^3 of water in the given reservoir
$\mathbf{E}_t^n = \mathbf{x}_t^n \times \text{SpecificPower}$

% Construct a 1D approximation of the 7D gradients (one gradients per each of the seven reservoirs) by a weighted average of each dimensional with respect to the amount of stored energy in each reservoir
$\beta_t^n = \hat{\beta}_t^n \cdot \mathbf{E}_t^n \qquad \forall n$

% Find dominating cut for sample points 0,10,20,...,4400,4410
for $x = 10$ **to** 4410 **by** 10 **do**
  y[x] $= \arg\max\{\alpha_1^n + \beta_1^n \text{x}\} \qquad \forall n$
end
Determine the approximate area under the given cut
for $x = 10$ **to** 4410 **by** 10 **do**
  rectangle[x] $= (\alpha_1^{\text{y[x]}} + \beta_1^{\text{y[x]}}(\text{x-5})) \times 10$
end
1D Integral Approximation $= \sum_x$ rectangle[x]