Fraktal

Bridging the Gap Between Reinforcement Learning and SDDP for Hydropower Scheduling

Hydropower Scheduling Conference 2022, Oslo.

Martin.Hjelmeland@fraktal.no

Background

2015

Started my PhD thesis on “Medium-term hydropower scheduling” using the SDDP and SDDiP algorithm.

2018

Pascal Cote holds a talk on “Mid-Term Hydro-Scheduling Problem: The Battle Between Stochastic Dynamic Programming and Reinforcement Learning”.

Worked at Lyse, a hydropower company in Norway.

2020

Started working in a machine learning team at Fraktal in 2020.

Started a project on using reinforcement learning for hydropower scheduling. Partners:

- Lyse
- The Research Council of Norway
- Innovation Norway

Innovation Norway

The Research Council of Norway

Refresher on the Stochastic Dual Dynamic Programming algorithm

Key concepts

Iterative algorithm

Forward iteration provides candidate solutions

Backward iteration trains the expected future profit function

To guarantee convergence
- iid stochastic process
- Linear problem

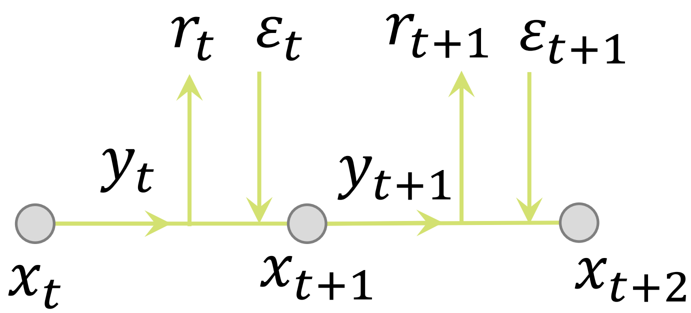
Stagewise formulation

$$Q_t(x_{t-1}, \varepsilon_{t-1}) = \max_{(x_t, y_t) \in X_t(x_{t-1}, \varepsilon_{t-1})} \{f_t(x_t, y_t) + V_t(x_t)\}$$

$$V_t(x_t) = \{\theta_t \leq U_t, \\ \theta_t \leq \pi_t x_t + b_t\}$$

Reinforcement learning are based on the Markov decision process

Illustration



Nomenclature

RL	SDDP	
a_t	y_t	Action
s_t	x_t	State
r_t	f_t	Reward
$V(x_t)$	$Q(x_t)$	State-value function
$Q(x_t, y_t)$		Action value function
$\pi(\cdot \mid x_t)$		Stochastic policy function
$\mu(x_t)$		Deterministic policy function

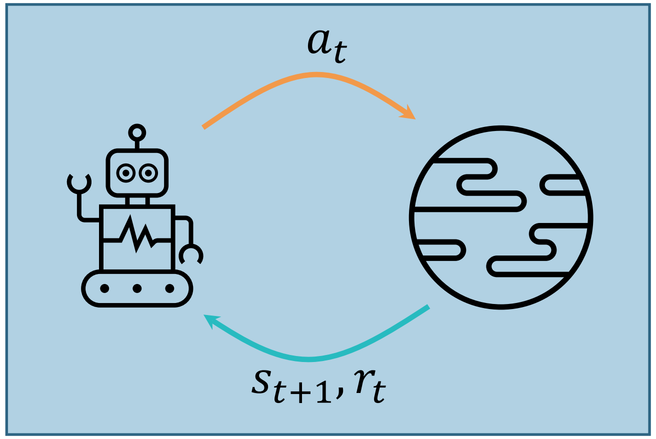
What is the objective of RL and stochastic optimization?

RL	Stochastic Optimization
Maximize expected sum of rewards	Minimize expected cost function

RL

Maximize expected sum of rewards

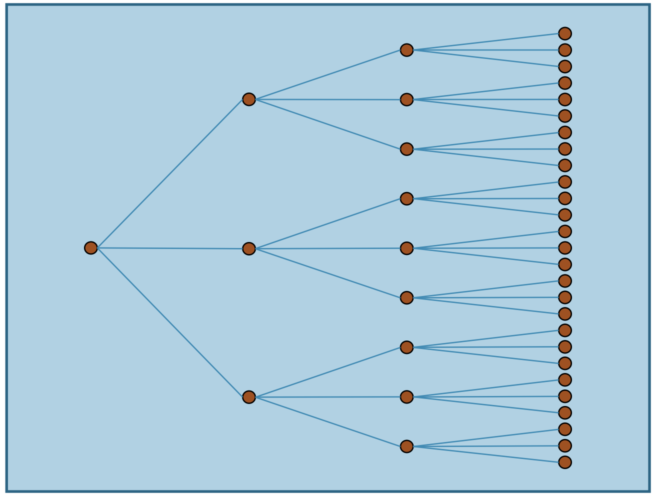
$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t, a_t)\right]$$



Stochastic Optimization

Minimize expected cost function

$$\min_{(x_n,y_n)} \left\{ \sum_{n\in\mathcal{T}} p_n f_n(x_n, y_n) : \left(x_{a(n)}, x_n, y_n\right) \in X_n \forall\, n \in \mathcal{T} \right\}$$



How are they solved?

RL	SDDP

RL

Maximize expected sum of rewards

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t)\right]$$

$$Q(s_t,a_t) = r(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim\rho_\pi}\left[V(s_{t+1})\right]$$

Function approximation

- Neural networks
- Gradient ascent

SDDP

Minimize expected cost function

$$\min_{(x_n,y_n)} \left\{ \sum_{n\in\mathcal{T}} p_n f_n(x_n,y_n) : \left(x_{a(n)}, x_n, y_n\right) \in X_n \forall\, n \in \mathcal{T} \right\}$$

$$Q_n\left(x_{a(n)}\right) = \min_{(x_n,y_n)} \left\{ f_n(x_n,y_n) + \sum_{m\in\mathcal{C}(n)} q_{nm} Q_m(x_n) : \left(x_{a(n)}, x_n, y_n\right) \in X_n \right\}$$

$$Q_t(x_{t-1}) = \min_{(x_t,y_t)} \left\{ f_t(x_t,y_t) + \mathbb{E}_{x_t}\left[Q_{t+1}(x_t)\right] : (x_{t-1}, x_t, y_t) \in X_t \right\}$$

Function approximation

- Hyperplanes
- Mathematical programming solvers

How are they solved?

RL	SDDP

Actor $a_t = \underset{a_t}{argmax}\, \pi_\theta(\cdot\,|s_t)$

Critic $v = Q_\varphi(s_t, a_t)$

$$x_t, y_t = \underset{(x_t, y_t) \in X_t}{argmin} \{f_t(x_t, y_t) + V_t(x_t)\}$$

Policy optimization is reinforcement learnings “version” of the mathematical solver

	Formulation
Reward function	$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}\left[r(s_t,a_t)\right]$
Compute gradient	$\nabla_\theta J(\pi_\theta) = \nabla_\theta \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_{\pi_\theta}}\left[r(s_t,a_t)\right]$
Arithmetic	$\ldots$
Policy gradient	$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta log(\pi_\theta(a_t\|s_t))r(s_t,a_t)]\right]$
Gradient ascent	$\theta = \theta + \tau\nabla_\theta J(\pi_\theta)$

Q-learning is reinforcement learnings version of SDDP’s backward pass

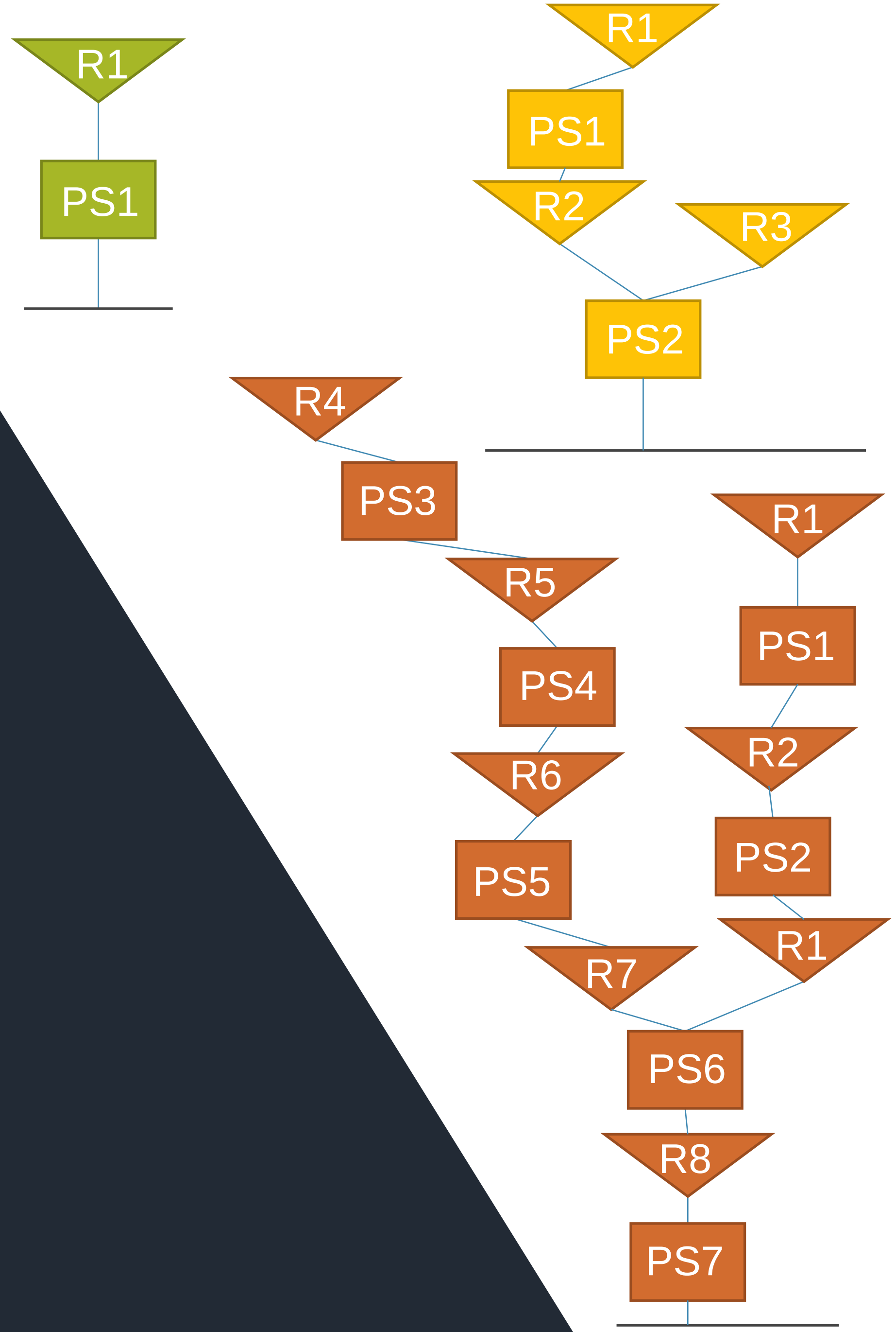
	Formulation
Bellman equation	$Q(s_t, a_t) = r(s_t, a_t) + \mathbb{E}_{s_{t+1} \sim \rho_\pi}[V(s_{t+1})]$
Approximate action value	$Q_\varphi(s_t, a_t) = r(s_t, a_t) + Q_\varphi(s_{t+1}, a_{t+1})$
Bellman error	$\delta_\varphi = Q_\varphi(s_t, a_t) - [r(s_t, a_t) + Q_\varphi(s_{t+1}, a_{t+1})]$
Gradient ascent	$\varphi = \varphi + \tau \nabla \delta_\varphi$

Case Studies

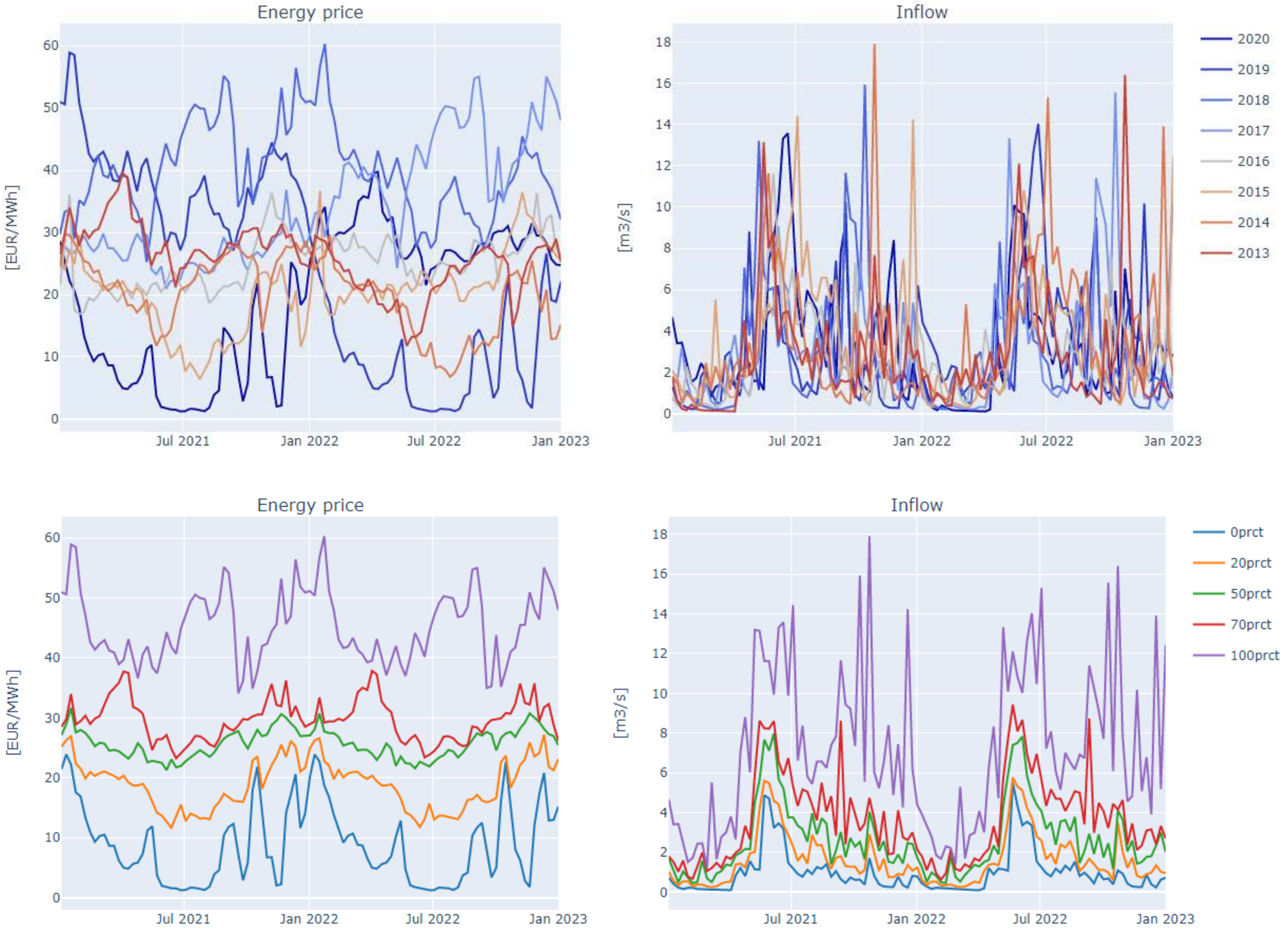
- Three hydro power systems
  - Small
  - Medium
  - Large

- Weekly time resolution over two years
  - 104 stages

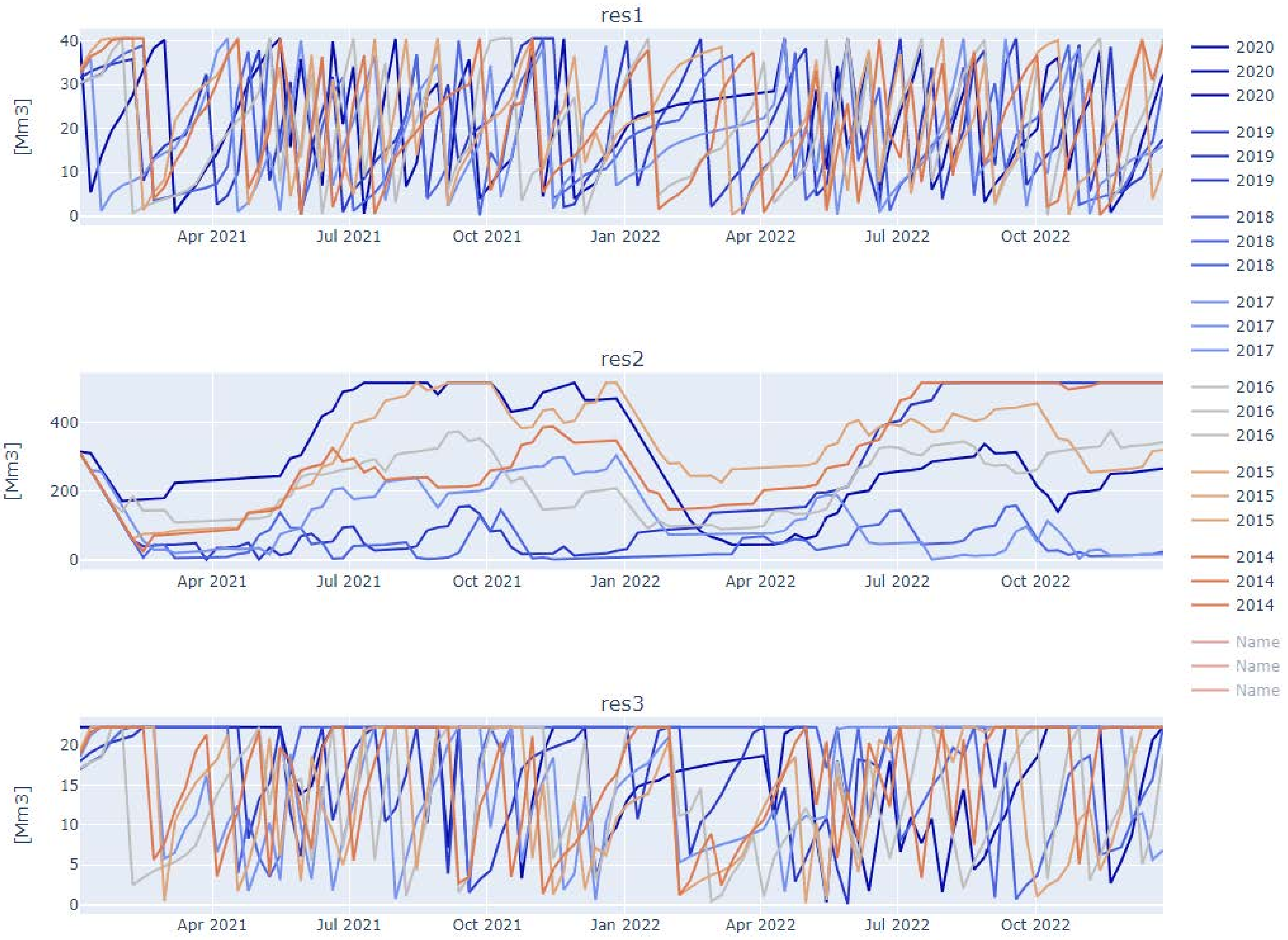
- Historical price and inflow scenarios
  - Markov chain for training
  - Actual scenarios for evaluation



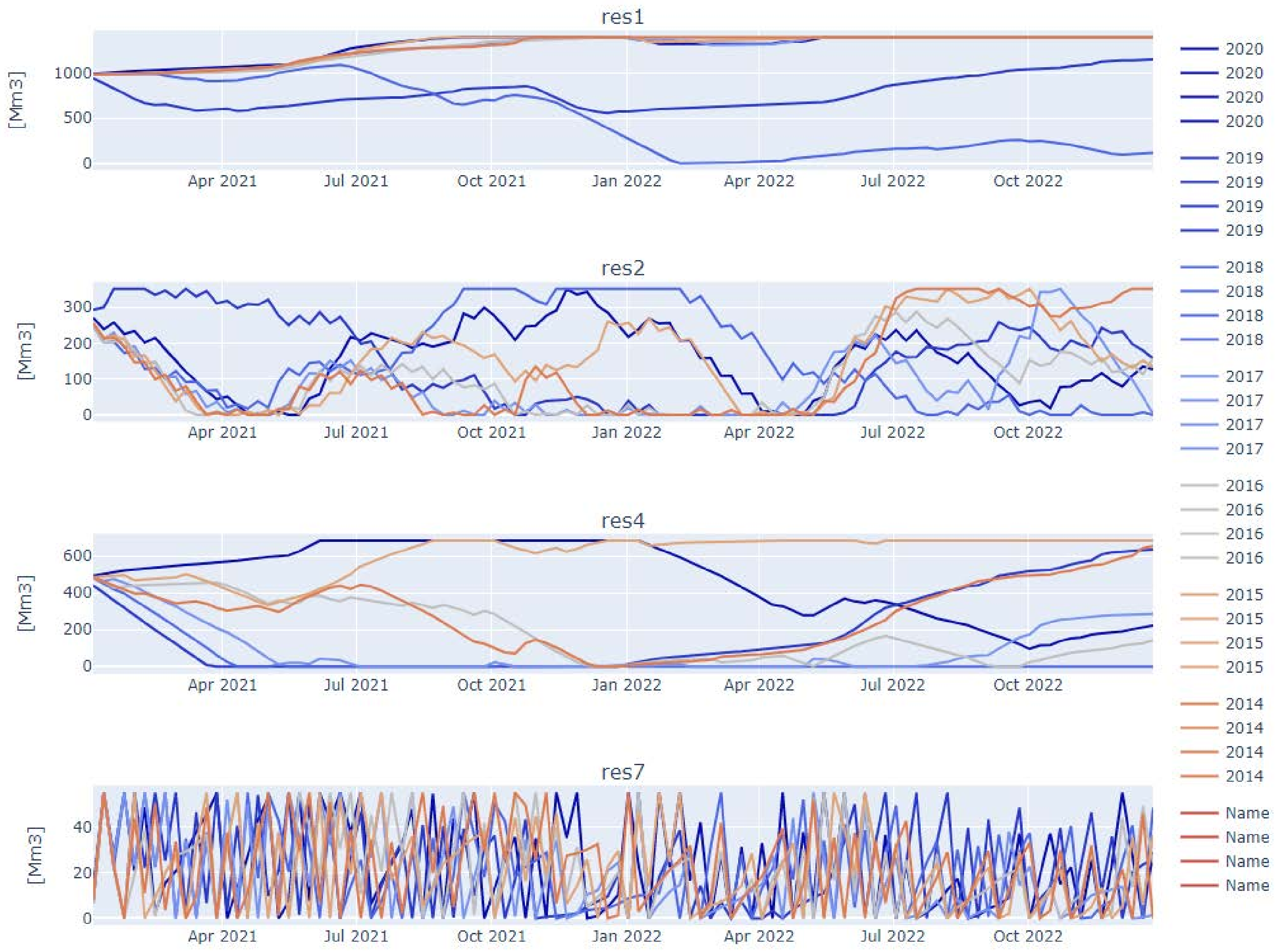
Inflow and energy prices used in case study



Hydro system “Medium”

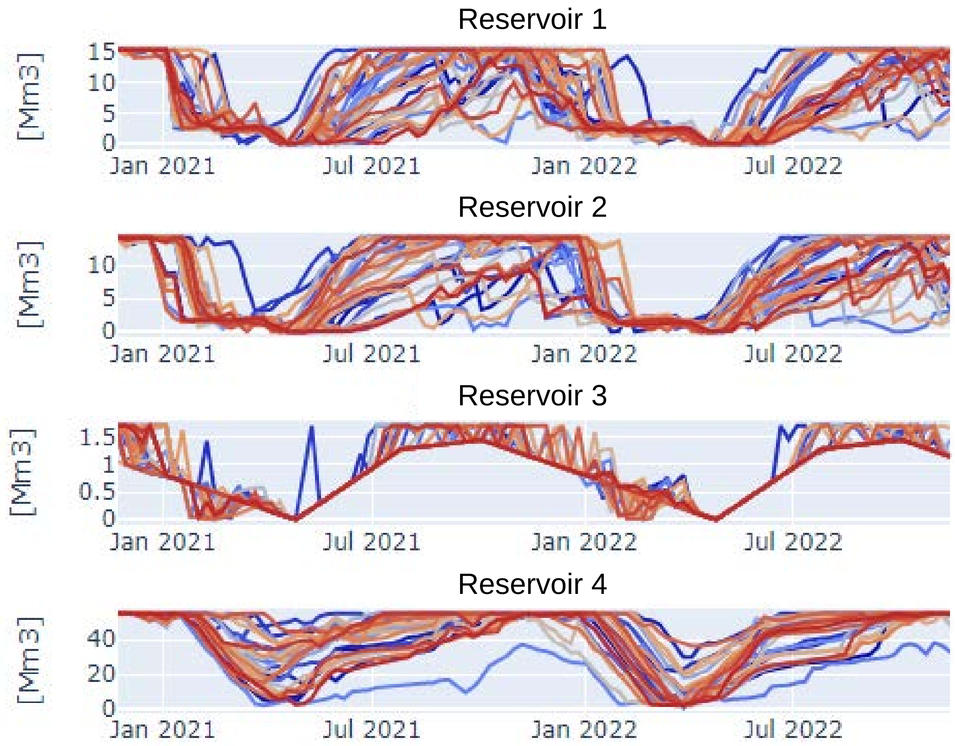


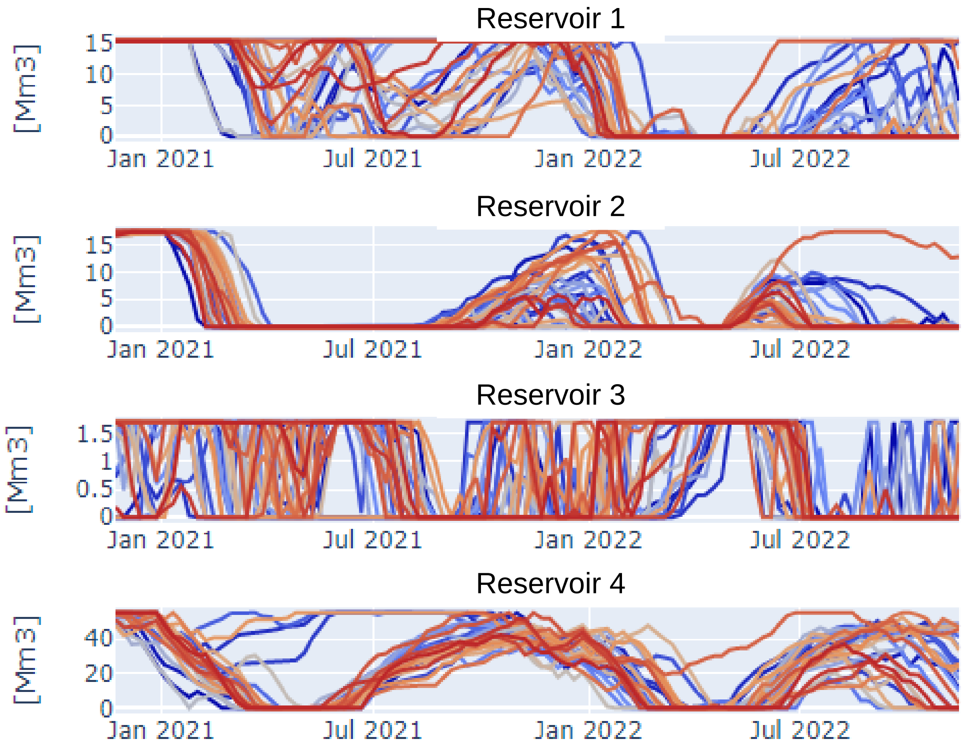
Hydro System Large



Benchmarking against a conventional SDP/SDDP model. Performed with a Norwegian hydropower producer.

SDP/SDDP	RL





Results were on par when comparing weekly values

However, RL algorithm was not able to handle hourly decision adequately

- ~20% worse score

Is there some way to combine the benefits of reinforcement learning and SDDP?

Idea
Build the value function with neural nets and solve stagewise decision problem with a mathematical solver

Goal
Leverage nonlinear neural nets and precision of mathematical solvers

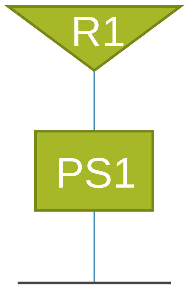
How
Iterative approach with approximated gradients

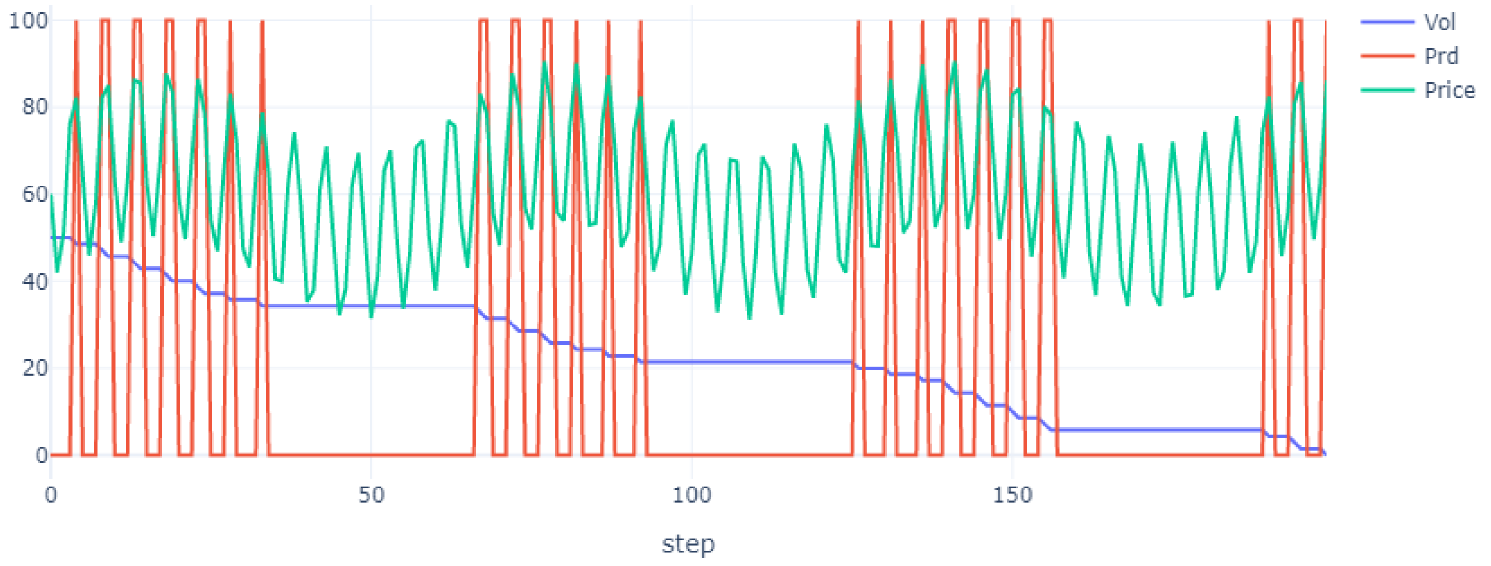
Actor $\cancel{a_t = \underset{a_t}{\arg\max}\, \pi_\theta(\cdot \mid s_t)}$

Critic $\boxed{v = V_\varphi(s_t)}$

$$x_t, y_t = \underset{(x_t, y_t) \in X_t}{\arg\min} \{ f_t(x_t, y_t) + \cancel{V_t(x_t)} \}$$

Overview of the simple deterministic case study used for testing





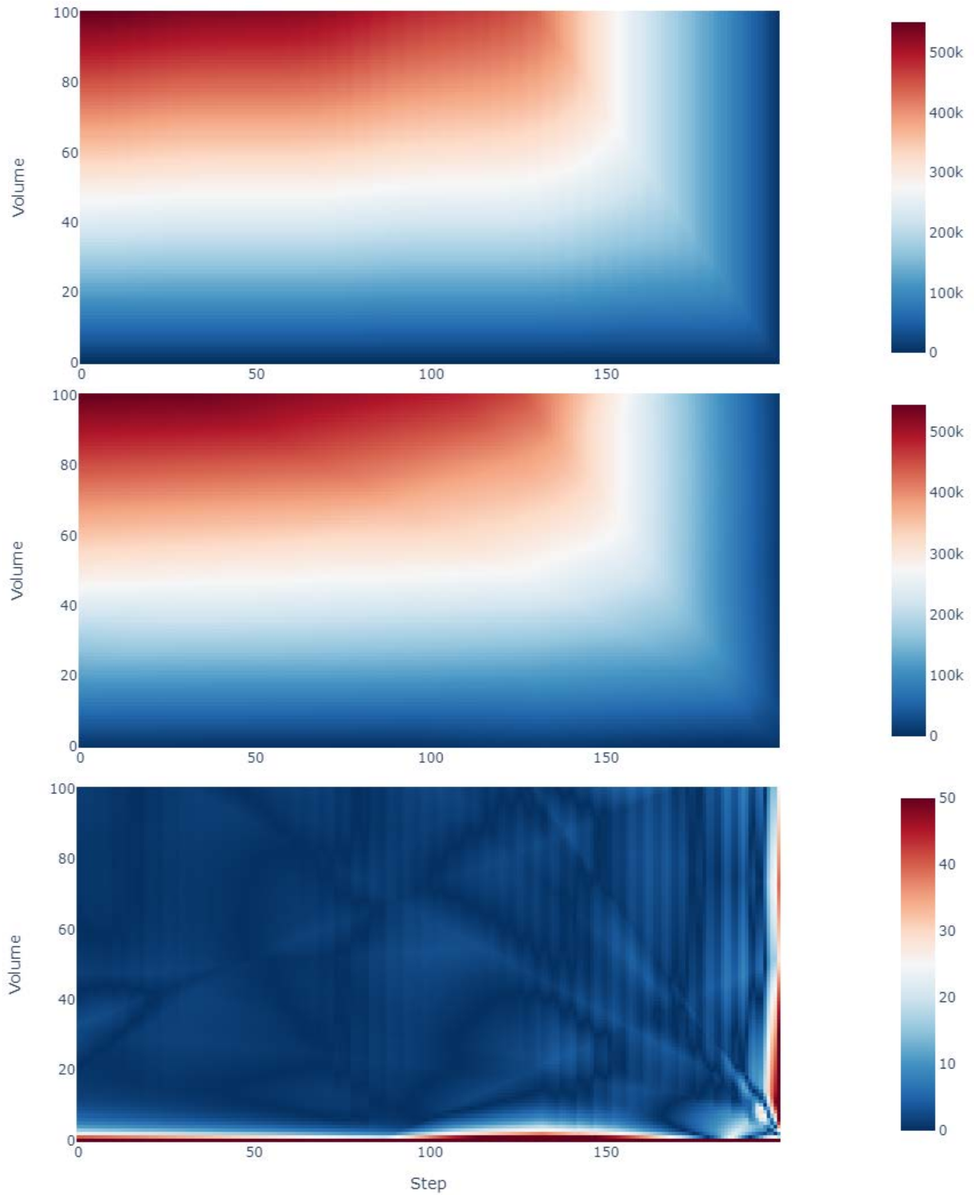
Tractable to compute the actual value functions

The value function is approximated by a neural network

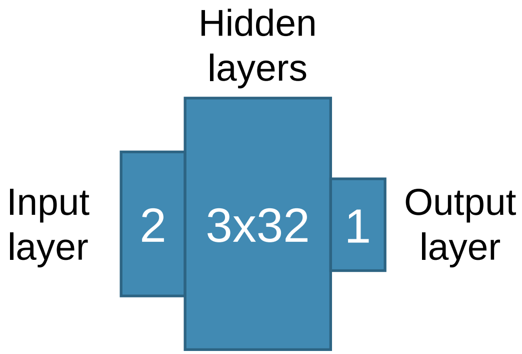
Actual value function

Neural net value function trained with supervised learning

Difference[%]



Neural network structure



You have the value function described by a neural network, now what?

$$x_t, y_t = \underset{(x_t, y_t) \in X_t}{\operatorname{argmin}} \left\{ f_t(x_t, y_t) + V_\varphi\left(x_t^k\right) \right\} \qquad (1)$$

Algo 1

1. $k = 0$, set x_t^0.
2. Fix $V_\varphi\left(x_t^k\right)$
3. Solve (1)
4. If $x_t - x_t^k > \delta$

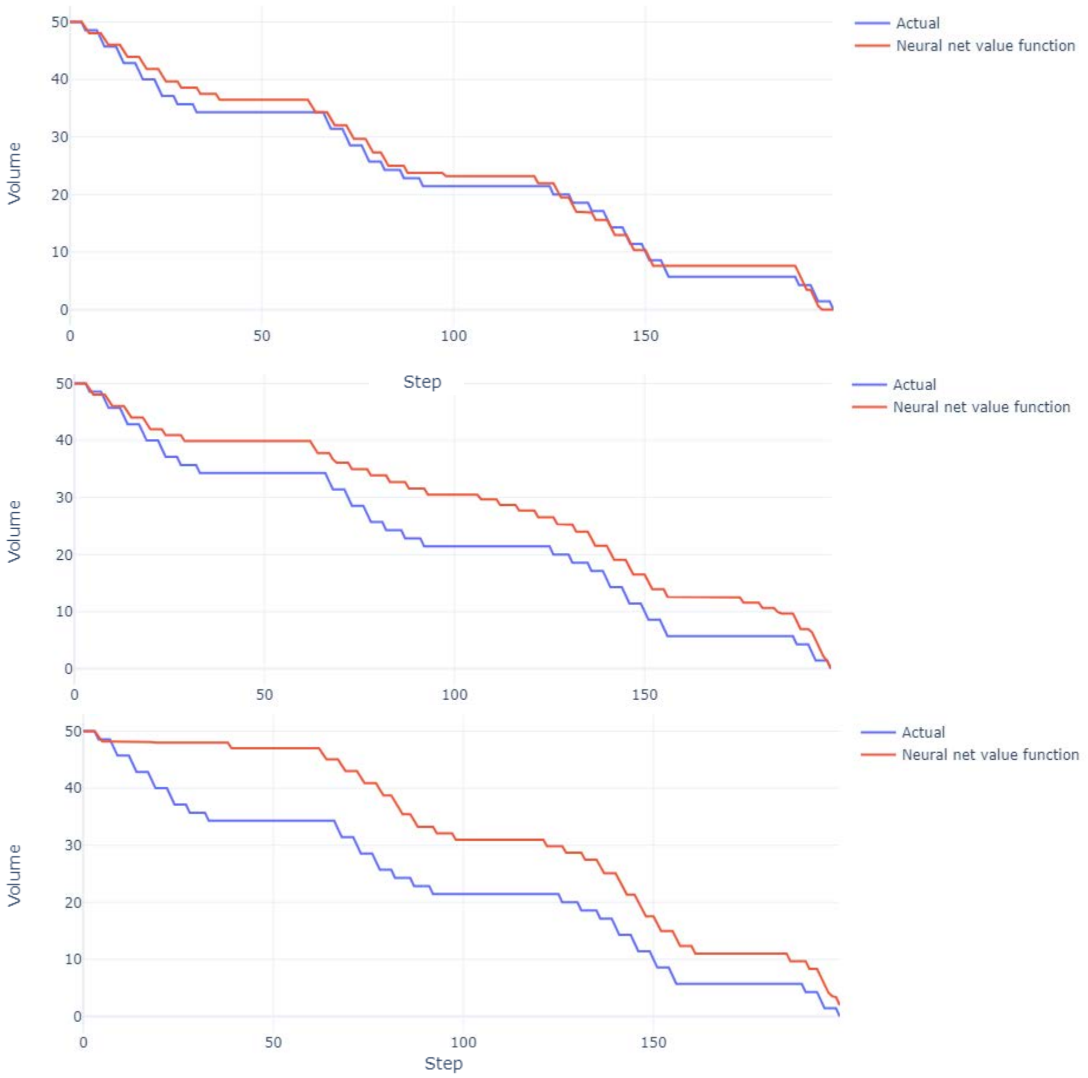
   Solve gradient ascent

   Update x_t^k and go to 2.

   Else:

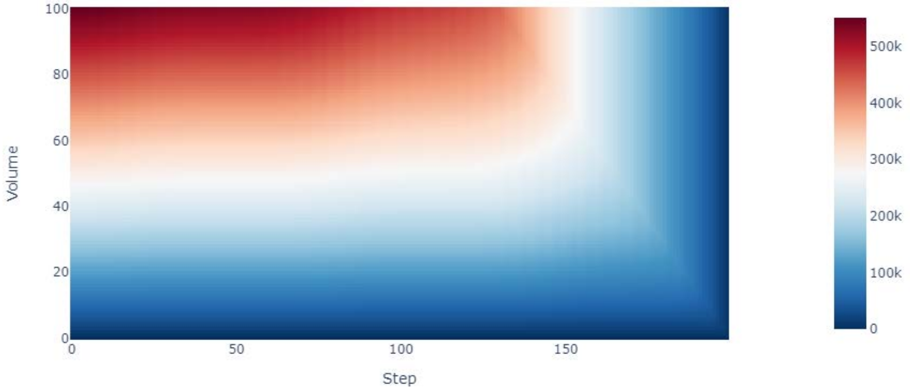
   Finished.

The solution is sensitive to the random initialization of the neural net and hyperparameters of **Algo 1**

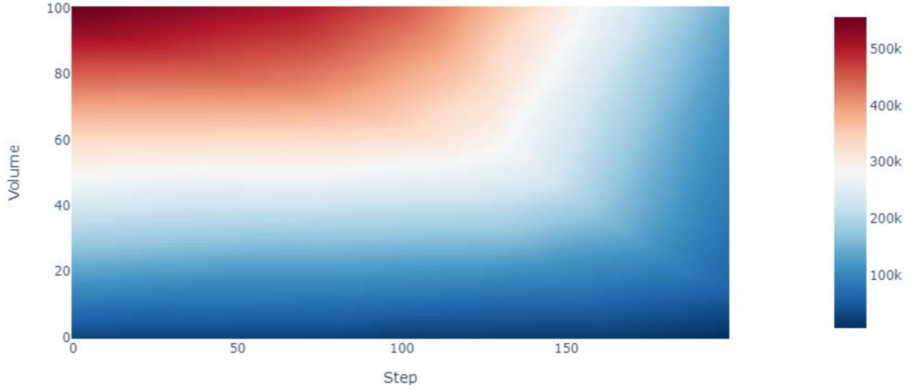


The value function learned with Q-learning

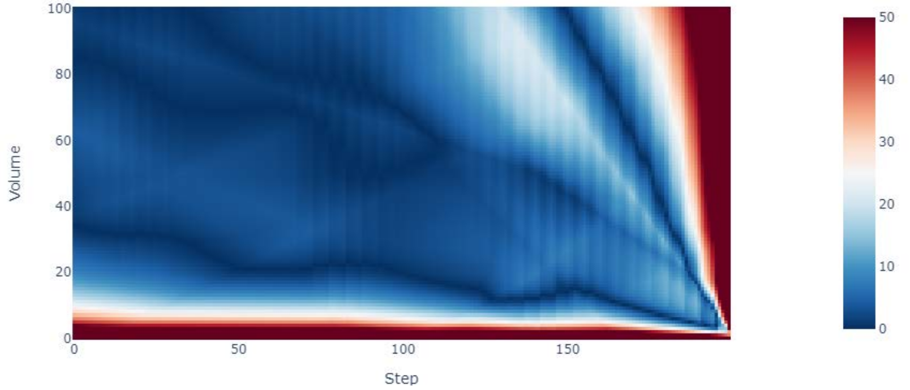
Actual value function



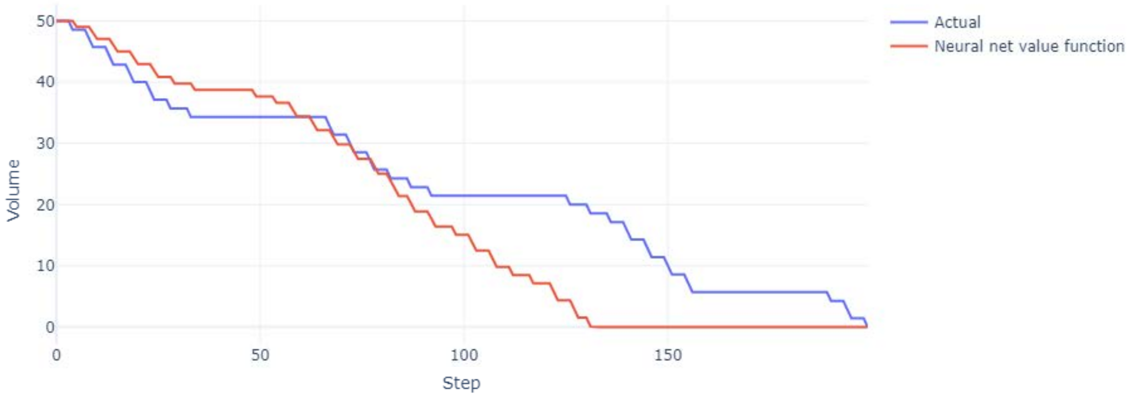
Q-learning value function



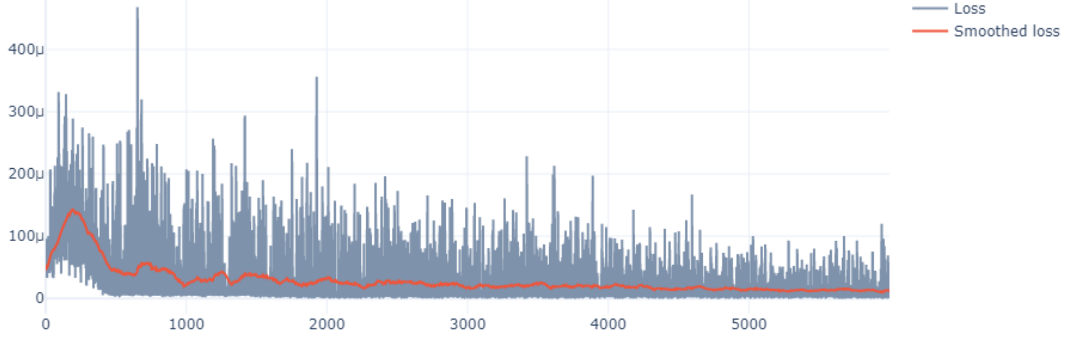
Difference [%]



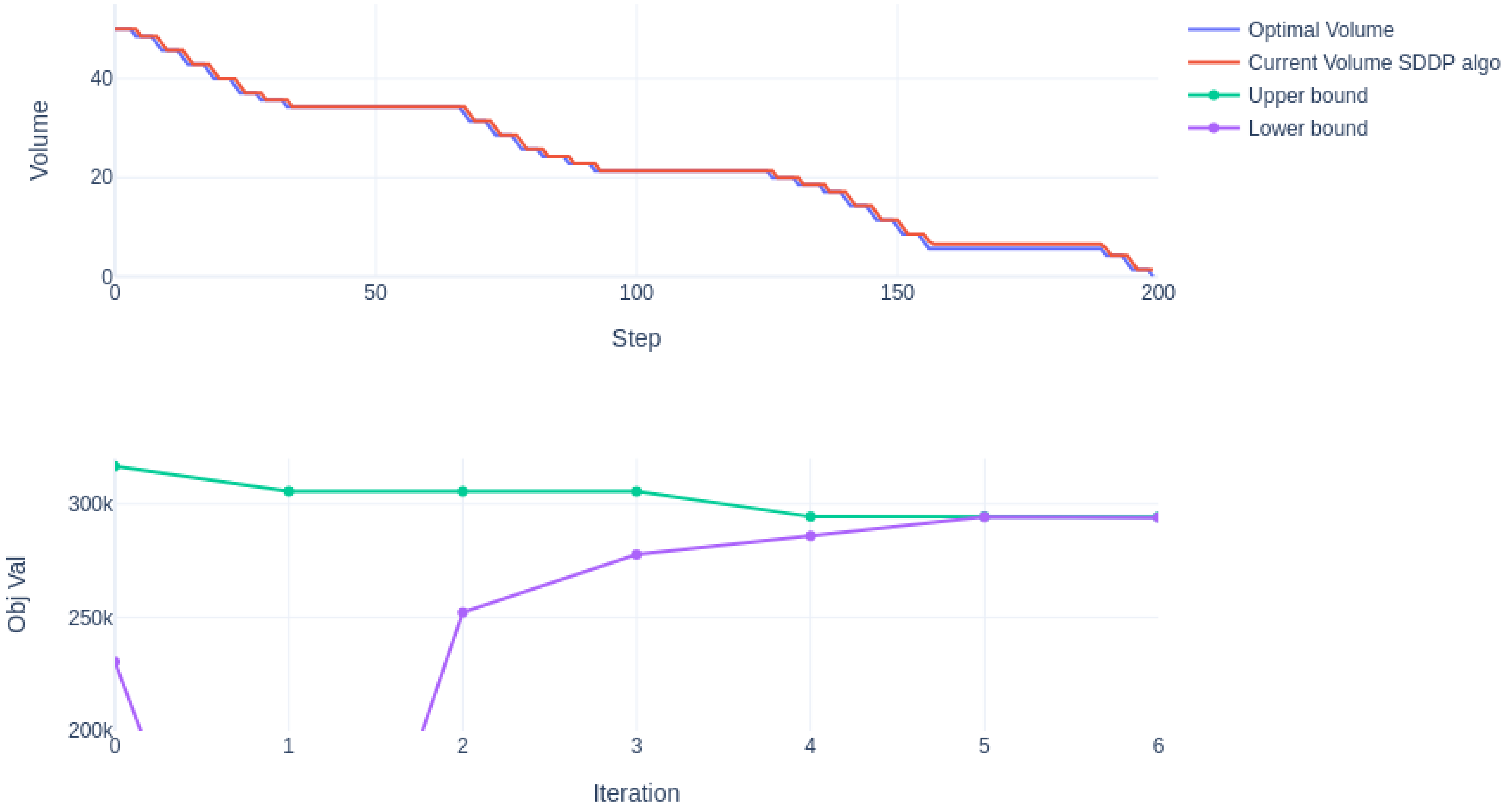
Reservoir trajectory



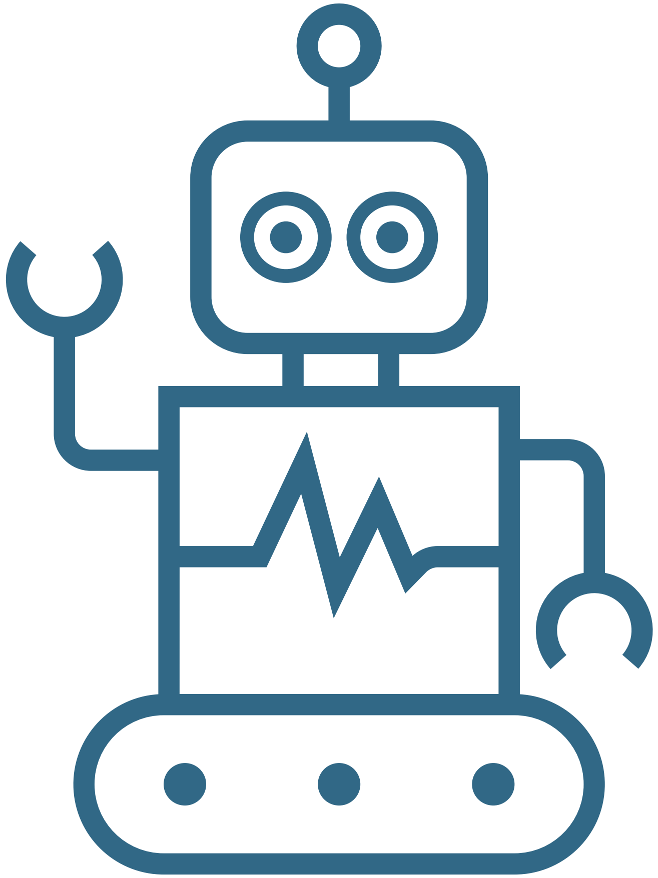
Bellman error (Loss)



SDDP is still king …



…. but the robots are getting closer!



Thank you!

Martin.Hjelmeland@fraktal.no