Optimal gas storage valuation and futures trading under a high-dimensional price process

Nils Löhndorf¹, David Wozabal²
¹Vienna University of Economics and Business, Vienna, Austria
²Technische Universität München, Munich, Germany
{nils.loehndorf@wu.ac.at, david.wozabal@wu.ac.at}

We study the problem of optimal gas storage valuation under a high-dimensional multifactor price model. The problem is modeled as a Markov decision process which leads to a stochastic version of the popular rolling intrinsic value. We show that the rolling intrinsic solution is optimal for the case of extreme risk aversion when using the nested conditional value-at-risk. We solve the problem by combining optimal quantization with approximate dual dynamic programming. Our quantization method reduces the high-dimensional multifactor price model to a discrete lattice. We find that it is necessary to match price expectations on the lattice with price expectations of the continuous process. In a numerical study, we demonstrate that our approach yields 26%-55% higher values as compared with another state-of-the-art approach from the literature on the exact same problem instances.

Key words: stochastic programming, dynamic programming, real options, finance, asset pricing, Markov decision process, OR in energy

History:

1. Introduction

In contrast to storage space for consumer goods, natural gas storage is a particularly scarce resource as it requires natural geological formations like depleted gas fields, salt domes, or aquifers. However, the global market for natural gas storage is growing and projected to reach 764 billion dollars by 2019 [Markets and Markets, 2014].

In many cases, the owner of storage capacity does not own the gas, and thus a contract is used to transfer usage rights from storage owners to third parties. The contract is typically time-limited and often referred to as virtual storage. Despite its practical importance, pricing such contracts is difficult, because storage derives its value from the flexibility it provides to gas trades.

As a commodity, the value of natural gas is reflected by spot and forward markets, such
as the New York Mercantile Exchange (NYMEX) in the United States. The value of storage thereby equals the discounted cash flow from buying and selling obligations for the physical delivery of gas that gets physically withdrawn and injected. Therefore, pricing a contract on storage capacity is equivalent to finding an optimal operational and trading policy.

Finding an optimal policy entails three challenges: modelling random future gas prices of spot and forward markets, capturing the dynamic nature of the problem implied by storage and forward contract balances, and incorporating complex technical constraints that may be part of the contract (Secomandi 2010).

To determine the optimal value of storage, forward trading has to anticipate the uncertain dynamics of the forward curve as well as all possible future trades, which necessitates solution of a complex stochastic optimization problem. Stochastic-dynamic programming is the natural choice for solving such problems, but because of the high dimensionality of the forward price process it is considered as computationally intractable for the problem at hand (see, e.g., Chen and Forsyth 2007 Carmona and Ludkovski 2010 Lai et al. 2010 2011).

Existing approaches for gas storage valuation either ignore the full dynamics of forward and spot prices and implement a greedy approach based on observed prices, or they simplify the stochastic optimization problem, for example, by using low-dimensional spot price models and ignoring the possibility of trading in forward markets altogether.

Because gas can be used for heating, demand for natural gas is typically higher in winter than in summer, which leads to higher forward prices for winter months than for summer months. Greedy approaches are based on exploiting these types of price differences. The simplest approach in this regard is to compute the intrinsic value utilizing the differences in the current forward prices. However, forward prices change over time, and additional profits can be made from adjusting the forward positions. If we simulate the corresponding adjustments made from computing the intrinsic value, we arrive at the so-called rolling intrinsic value. Here multifactor stochastic models are a popular choice to model and simulate possible future price changes (Gray and Khandelwal 2004a).

Another approach often found in practice, is based on finding a basket of (time) spread
options (see Eydeland and Wolyniec 2003, Gray and Khandelwal 2004b,a). Gray and Khandelwal (2004a) find that the difference in results between the rolling basket-of-spread option approach and the rolling intrinsic approach is small.

Most other valuation approaches in the extant literature aim at finding an approximate solution to the stochastic dynamic program, typically by restricting the policy to trading in the spot market.

A popular technique in this regard is to find a solution to a simplified stochastic dynamic program without futures trading using least squares Monte Carlo (LSMC), (see, e.g., Boogert and de Jong 2008, 2012, Carmona and Ludkovski 2010, Bjerksund et al. 2011). LSMC approximates the value function of the dynamic program by training a linear regression model of the price state for a number of discretized storage states. Another approximation strategy is to discretize the spot price process by a recombining scenario tree and to use backward recursion to solve the discrete-state dynamic program (Felix and Weber 2012). Thompson et al. (2009) avoid discretization altogether by approximating the partial differential equations of a storage valuation model in continuous-time with spot prices modeled as jump diffusion processes. A similar approach is pursued by Chen and Forsyth (2007).

While low-dimensional spot price model keeps stochastic dynamic programming computationally tractable, traders are suspicious of these approaches, since spot price models fail to capture the real dynamics of the market (Lai et al. 2010).

For this reason, Lai et al. (2010) consider gas storage valuation based on monthly forward trading where the evolution of the forward curve is modeled as a multivariate geometric Brownian motion (MGBM). The authors benchmark the (rolling) intrinsic and basket of spread option approaches against their own method, which is based on approximate dynamic programming. They demonstrate that their method outperforms the benchmarks but only yields marginal improvements over the rolling intrinsic approach.

While Lai et al. (2010) employ a high-dimensional price model, the state space of their dynamic program only considers the front month future price and storage but neither the full dynamics of the forward curve nor trading in other forward contracts. Our study intends to
fill this gap by considering full dynamics of the forward curve as well as forward trading. In line with Lai et al. (2010), we model the dynamics of the forward curve as an MGBM and use intrinsic valuation as the baseline.

The contribution of our study is threefold. Our first contribution is the formulation of a Markov decision process (MDP) for storage valuation based on forward trading. We set up the problem in such a way to ensure that the forward positions held at any point in time define an implementable schedule of injections and withdrawals. Myopic planning at the first stage yields an immediate profit that equals the intrinsic value and myopic planning, repeated over the whole time horizon yields the rolling intrinsic value. Therefore, the solution to the MDP provides a stochastic rolling intrinsic value which is a generalization of the classic rolling intrinsic approach.

The second contribution concerns our solution approach. To mitigate the curse of dimensionality inherent in our model formulation, we develop an approach based on approximate dynamic programming (Powell, 2011). Our solution approach exploits the fact that the state space of the MDP can be separated into an environmental state defined by the price forward curve, and a resource state defined by forward contracts and storage.

To discretize the environmental state, we develop a learning algorithm that builds a discrete process, referred to as a lattice, using optimal quantization. A learning algorithm for optimal quantization of a stochastic process was first proposed in Bally and Pagès (2003) but with a different notion of distance than ours. See Bonnans et al. (2012) for an application where a univariate Brownian motion price process is discretized. Our algorithm combines optimal quantization with moment matching. In contrast to the moment matching heuristic of Høyland et al. (2003), which generates scenarios for two-stage stochastic programming, our algorithm finds a discretization that matches the conditional means of the discrete process with the means of the continuous process while minimizing the distance to the optimal quantizers.

In a subsequent step, our solution approach assigns each state of the lattice an optimal value function approximation that is piecewise-linear in the resource state. To construct this function, we resort to a revised version of the approximate dual dynamic programming (ADDP) algorithm.
ADDP was first proposed in Löndorf et al. (2013) and extends the celebrated stochastic dual dynamic programming (SDDP) algorithm of Pereira and Pinto (1991). In contrast to SDDP, ADDP does not require randomness to be stagewise independent but learns a piecewise-linear approximation of the value function around each environmental state of the lattice. The learning process iterates between simulating the decision process under the current policy and then improving the policy by updating the approximation in regions of the state space that are reached by the current policy. It can be shown that this solution strategy converges toward an optimal policy on the lattice.

Since we can assign each state of the stochastic process a state on the lattice by way of rounding to the nearest lattice node, the resulting value function approximation is piecewise-constant in the environmental state dimensions and piecewise-linear in the resource state dimensions. Therefore, we can use our value function to derive a policy that enables us to approximate the optimal value of storage under the original stochastic process that has been used to generate the lattice. Using the case study in Lai et al. (2010), we can show that our policies significantly outperform the rolling intrinsic and the approximate dynamic programming policies proposed in that paper.

Our third contribution concerns connecting our valuation approach to rolling intrinsic valuation. In contrast to the rolling intrinsic policy, the optimal policy maximizes the expected discounted reward of the underlying stochastic optimization problem. Since the optimal policy anticipates possible movements of the forward curve it can speculate on such movements, which exposes the decision maker to much more risk than the greedy rolling intrinsic policy. We can unify both approaches in one model by replacing the expectation with the nested conditional value-at-risk (N-CVaR). This reformulation shields the decision-maker from possible losses at the expense of losing possible higher upside gains. We prove that the rolling intrinsic value results as a special case of optimizing the N-CVaR. Therefore, the formulation contains the rolling intrinsic solution as a boundary case for extreme risk aversion.

Our results are intended to provide energy traders with a new model to price gas storage contracts that relies on assumptions which are widely accepted among traders. In particular,
the new model allows traders to choose the level of risk exposure that matches their preferences somewhere between conservative rolling intrinsic planning and risk neutral planning. Furthermore, the proposed solution approach could be of interest to researchers working on other problems, as it is general enough to address other complex stochastic-dynamic decision problems.

The paper is organized as follows. In Section 2, we summarize our assumptions and state our model formulation. In Section 3, we describe our solution algorithm and discuss some of its properties. In Section 4, we present a numerical analysis which is based on the model parameters of Lai et al. (2010). Some further discussion and concluding remarks are found in Section 5.

2. Model Formulation

2.1. Assumptions

We consider the problem of an energy trader who manages a real option contract for the capacity of a gas storage facility. A virtual storage contract gives the buyer the right to inject, withdraw, and store gas over a finite time horizon. The injection, withdrawal, and storage capacity is limited and defined within the contract. We assume that injection and withdrawal limits are independent of the storage level, i.e., ratchets are ignored as is common with virtual storage contracts.

To make decisions about buying and selling virtual storage capacity, the trader has to price the contract, which requires quantifying its value at the time of inception. The value derives from the expected value of the discounted cash flows from buying and selling forward contracts while fulfilling matured contracts. In line with the literature on gas storage valuation, we assume that cash flows are generated in monthly time increments (Eydeland and Wolyniec 2003, Lai et al. 2010).

We assume that all forward positions yield a physically implementable policy under the given operational constraints of the virtual storage. At any point in time, the feasible set of buying and selling decisions is defined through the current forward positions, the storage level, and the operational limits. This implies that the trader is required to be able to fulfill the contractual obligations at all times by injecting or withdrawing accordingly without clearing contracts in
the market. We assume that each physical injection and withdrawal incurs marginal cost in
addition to an in-kind fuel loss. Holding natural gas in storage incurs no holding cost.

As in (Lai et al. 2010), the dynamics of the forward curve are assumed to follow a driftless
MGBM. In addition to the prompt month’s future price, we assume that in each month forward
prices are available for all future months.

2.2. The Stochastic Price Process

Denote $F_t$ as the forward curve in $t$, i.e., the vector of forward prices, with $F_{tj}$ as the forward
price in period $t = 1, \ldots, T$ with maturity in period $j = t, \ldots, T$, where $T$ is the number of
months in the planning period. Denote $dZ_{tj}$ as a standard Brownian motion increment related
to a forward contract in $t$ with maturity in $j \geq t$ and $\sigma_j$ as the constant monthly volatility and
$\rho_{jk}$ as the instantaneous correlation of increments $dZ_{tj}$ and $dZ_{tk}$. Then, the MGBM is defined
by the stochastic differential equations,

\[
\frac{dF_{tj}}{F_{tj}} = \sigma_j dZ_{tj}, \quad dZ_{tj} dZ_{tk} = \rho_{jk}, \quad j = 1, \ldots, T. \tag{1}
\]

Remark 1. All elements of $Z_t$ are required to determine the state transition from $t$ to $t+1$, so that the dimensionality of the MGBM remains the same throughout and contains contracts beyond the end of the planning horizon in later periods.

2.3. Markov Decision Process

Under the given assumptions, the decision problem can be modelled as a finite horizon,
continuous-state, discrete-time MDP. The state of the MDP consists of the resource state, which
includes state variables that can be influenced by the decision-maker, and the environmental
state, which is characterized by the stochastic process that defines the evolution of the forward
curve and is exogenously given. Denote $R_t$ as the resource state in $t$ that includes all forward
positions, $f_{tj}$, with $j \geq t$, as well as the gas storage level, $s_t$.

Denote $\pi = \{\pi_1, \ldots, \pi_T\}$ as the decision policy that includes all trading and operational deci-
sions subject to the state-dependent feasible set $\Pi_t(R_t)$, and denote $C(F_t, \pi_t)$ as the immediate
reward and $\gamma_t$ as discount factor. For a given initial forward curve, $F_1$, and resource state, $R_1$, the expected discounted value is given by

$$V_t(F_t, R_t) = \max_{\pi_t \in \Pi_t(R_t)} \left( C_t(F_t, \pi_t) + E \left[ V_{t+1}(F_{t+1}, R_{t+1}(|\pi_t)) \right] \right),$$

$t = 1, \ldots, T$, with $V_{T+1} \equiv 0$. The objective of the trader is to maximize the expected discounted rewards at any point in time for a given state of the MDP.

The decision policy $\pi$ involves taking a set of actions subject to certain constraints to jointly maximize immediate and future rewards.

Denote $i_{tj}$ and $w_{tj}$ as the injection and withdrawal decisions in $t$ to close the forward position $f_{tj}$ that matures in $j$. Physical injection and withdrawal incurs marginal cost of $c^i$ and $c^w$ as well as in-kind losses $d^i$ and $d^w$, respectively. Furthermore, denote $x_{tj}$ and $y_{tj}$ as the forward buying and selling decisions.

Accordingly, the immediate reward is given by

$$C_t(F_t, \pi_t) = \sum_{j=t}^{T} \gamma_j F_{tj}(y_{tj} - x_{tj}) - \gamma_t(c^i i_{tt} + c^w w_{tt}),$$

where the monthly discount factor, $\gamma_t$, is defined by the annual interest rate, $\delta$, i.e.,

$$\gamma_t = (1 + \delta)^{-\frac{t}{12}}, \; t = 1, \ldots, T.$$

Note that to avoid charging a cost for the same transaction more than once, we do not account for injection and withdrawal cost incurred by forward trades.

When a forward contract matures, it must either be balanced physically through storage operation or virtually through the spot market, i.e.,

$$f_{tt} + x_{tt} - y_{tt} = i_{tt} - w_{tt}, \; t = 1, \ldots, T.$$

The storage balance that derives from physical operation is given by

$$s_{t+1,t+1} = s_{tt} + d^i i_{tt} - d^w w_{tt}, \; t = 1, \ldots, T.$$

To enforce the assumption that all forward positions yield an implementable schedule, all tradable contracts that mature in some future period must be balanced with future injections and withdrawals,

$$f_{t,j} = i_{t,j} - w_{t,j}, \; t = 1, \ldots, T - 1, \; j = t + 1, \ldots, T.$$
Moreover, all contracts are subject to the balance constraints,

\[ f_{t+1,j} = f_{t,j} + x_{t,j} - y_{t,j}, \quad t = 1, \ldots, T - 1, \quad j = t + 1, \ldots, T, \quad (8) \]

which keep track of the changes in forward positions through purchases or sales in \( t \) for \( j > t \).

Injections and withdrawals of the wait-and-see part are subject to the storage balance,

\[ s_{t,j+1} = s_{t,j} + d^i i_{t,j} - d^w w_{t,j}, \quad t = 1, \ldots, T - 1, \quad j = t, \ldots, T, \quad (9) \]

where \( s_{t,j} \) is the storage level in period \( j \) as projected in period \( t \), based on the assumption that all the trades are physically balanced by the storage. The operational limits of the storage are given by the following set of constraints:

\[ \bar{i} \leq i_{t,j} \leq \bar{i}, \quad t = 1, \ldots, T, \quad j = t, \ldots, T, \quad (10) \]

\[ \underline{w} \leq w_{t,j} \leq \bar{w}, \quad t = 1, \ldots, T, \quad j = t, \ldots, T, \quad (11) \]

\[ \underline{s} \leq s_{t,j} \leq \bar{s}, \quad t = 1, \ldots, T, \quad j = t, \ldots, T. \quad (12) \]

At the end of period \( t \), the physical state of the storage in addition to the forward position encompasses the final resource state, which is given by

\[ R_{t+1} = \{s_{t+1,t+1}\} \times \{f_{t+1,j}\}_{j=t+1}^T, \quad t = 1, \ldots, T. \quad (13) \]

Note the dimensionality of the resource state decreases in \( t \) because one forward position matures in each period.

**Remark 2.** The above constraints define the feasible sets \( \Pi_t(R_t), \quad t = 1, \ldots, T, \) for the policy \( \pi \). Since all constraints and the objective function are linear, the maximization problem at stage \( T \) is a linear program and therefore convex. The resource state \( R_t \) appears only on the right-hand side of the constraints, such that the objective value and therefore the value function \( V_T \) is a piecewise-linear and concave function in \( R_T \). Since concavity is preserved by the expectation operator, the expected value function in \( T - 1 \) is also piecewise-linear and concave in \( R_T \). It follows by backward induction that all expected value functions \( V_t \) are piecewise-linear functions and concave in \( R_t \) and all problems in (2) are linear optimization problems.
Denote $\bar{V}_t(F_t, R_{t+1})$ the expected value in (2), i.e.,

$$\bar{V}_t(F_t, R_{t+1}) = E\left[V_{t+1}(F_{t+1}, R_{t+1})\mid F_t\right],$$

which we will refer to as the post-decision value, in line with Powell (2011).

Since the post-decision value function $R \mapsto \bar{V}_t(F_t, R)$ is piecewise-linear and concave, it can be defined as the minimum of a set of $N$ hyperplanes with intercepts $a_n(F_t)$ and slopes $b_n(F_t)$ for $n = 1, \ldots, N$. The post-decision value can then be expressed by

$$\bar{V}_t(F_t, R_{t+1}) = \min_{1 \leq n \leq N} \left\{ a_n(F_t) + b_n(F_t)^\top R_{t+1} \right\}, \quad t = 1, \ldots, T - 1.$$  (15)

For a fixed $F_t$, the post-decision value, $\bar{V}_t$, can thus be expressed by the solution of the following linear program,

$$\bar{V}_t(F_t, R_{t+1}) = \max_{v_t, R_{t+1}} v_t$$

s.t. $v_t \leq a_n(F_t) + b_n(F_t)^\top R_{t+1}$, $n = 1, \ldots, N, t = 1, \ldots, T - 1.$  (17)

Combining (3) with (17), Problem (2) can be reformulated as

$$V_t(F_t, R_t) = \max \sum_{j=t}^{T} \gamma_j F_t(y_{tj} - x_{tj}) - \gamma_t(c^{t}i_{jt} + c^w w_{jt}) + v_t$$

s.t. $[5, 6, 7, 8, 9, 10, 11, 12, 17]$  
$x_{tj}, y_{tj}, i_{tj}, w_{tj}, s_{tj} \geq 0, \ t = 1, \ldots, T - 1, j = t, \ldots, T$  
$f_{t+1,j} \in \mathbb{R}, \ t = 1, \ldots, T - 1, j = t, \ldots, T - 1,$  
v_t \in \mathbb{R}, \ t = 1, \ldots, T - 1.$  (18)

### 2.4. Risk Measures

In the above formulation, the value of storage is defined as the expected discounted reward that results from jointly trading in the market and operating the storage. This implies a risk-neutral valuation, which is common in the extant literature but may not meet the actual risk preferences of energy traders who are possibly risk averse.

Therefore, we augment the problem formulation discussed above to explicitly include risk preferences. To this end, we substitute the expectation in the objective function of the MDP with the N-CVaR.

For a random variable $X$, let $\text{CVaR}_{\alpha}(X) = \alpha^{-1} \int_0^\alpha F_X^{-1}(t)dt$ and define $\rho_{\alpha,\lambda}(X) =$
\( \lambda \text{CVaR}_\alpha(X) + (1 - \lambda)\mathbb{E}(X) \). For a sequence of random variables \( X_1, \ldots, X_T \) the N-CVaR is defined recursively as

\[
\text{N-CVaR}_{\alpha, \lambda}(X_1, \ldots, X_T) = X_1 + \rho_{\alpha, \lambda}(X_2 + \rho_{\alpha, \lambda}(X_3 + \cdots)).
\]

We change the objective criterion from maximizing expected rewards to maximizing N-CVaR of the profits by altering the definition of the value function (2) to

\[
V_t(F_t, R_t) = \max_{\pi_t \in \Pi_t(R_t)} \left\{ C(F_t, \pi_t) + \rho_{\alpha, \lambda} \left[ V_{t+1}(F_{t+1}, R_{t+1}(\pi_t)) \right] \right\}.
\]

Thus, the expectation is replaced by a convex combination of expectation and CVaR which recursively includes other convex combinations of expectation and CVaR in a nested manner (Ruszczyński and Shapiro 2006). Unlike the terminal CVaR, which measures the risk of the discounted reward distribution, the N-CVaR is time consistent, which means that past losses and gains do not affect the policy (Shapiro 2009). Algorithmic strategies that deal with this formulation can be found in Shapiro (2011), Philpott et al. (2013), Kozmík and Morton (2014). We address this aspect in Section 3.3.

It turns out that the discounted profits accumulated under the optimal N-CVaR policy yield the optimal expected value as well as the rolling intrinsic value as special cases. The optimal expected value can be trivially obtained by setting \( \lambda = 0 \). To relate the N-CVaR policy to the rolling intrinsic solution, we consider the case of \( \lambda = 1 \), i.e., the decision-maker only maximizes the CVaR and ignores the expected value entirely.

Note that for \( \alpha = 0 \), we get

\[
\text{CVaR}_0(X) = \text{ess inf}(X).
\]

Setting \( \alpha = 0 \) under the N-CVAR with \( \lambda = 1 \) yields the following stochastic optimization problem

\[
\max_{\pi} \text{N-CVaR}_{\alpha, \lambda} = \max_{\pi} \sum_{t=1}^{T} \text{ess inf} \gamma C(F_t, \pi_t),
\]

which has the rolling intrinsic solution as the optimal policy as will be shown below.

The intuition behind this is that the forward market offers a price for future deliveries that enables the decision maker to realize a risk-free immediate reward by trading the respective
forward contracts subject to future operational limits. This results in a myopic, greedy policy, referred to as the rolling intrinsic solution, that maximizes the immediate reward irrespective of possible future gains and shields the decision maker against unexpected future losses.

**Proposition 1**

1. The optimal policy for the N-CVaR formulation with $\alpha = 0$ and $\lambda = 1$ yields the rolling intrinsic strategy as optimal policy.

2. The objective value as well as the decisions of the planning problem with an N-CVaR objective and $\lambda = 1$ converges to the rolling intrinsic objective as $\alpha \to 0$.

**Proof.** Since the rolling intrinsic policy is greedy, the optimal first-stage decision is to maximize $C(F_1, \pi_1)$ subject to the storage constraints irrespective of possible gains that might occur because of random price changes in the future. Let $\pi_1^*$ be these decisions.

Any policy with a lower first-stage profit must earn a positive amount of money through price changes in later periods almost surely to make up for the difference. Therefore, it suffices to show that for every decision $\bar{\pi}_1$, there is a positive probability that price changes lead to a negative market-to-market evaluation of currently held forward contracts, i.e., there is no speculative gain in holding the position $\bar{\pi}_1$.

In particular, the set of prices in the next stage that do not lead to an appreciation of the portfolio contains

$$
\mathcal{H} = \{ F_1 = (F_{1,1}, \ldots, F_{1,T}) : f_{1,t}^j (F_{1,t} - F_{0,t}) \leq 0, \ \forall \ t = 2, \ldots, T \}.
$$

Clearly, $\mathcal{H}$ is an intersection of half-spaces in $\mathbb{R}^T$ that by themselves intersect the positive orthant. Since prices are log-normally distributed the probability of $\mathcal{H}$ is positive. This finishes the proof of 1.

To prove 2, note that CVaR$_\alpha$ converges to CVaR$_0$ as $\alpha \to 0$. Therefore, it follows that

$$
\text{N-CVaR}_{\alpha,1} \xrightarrow{\alpha \to 0} \text{N-CVaR}_{0,1}
$$

and consequently from Shapiro et al. (2009), Theorem 7.27, that

$$
\text{N-CVaR}_{\alpha,1} \xrightarrow{\text{epi}} \text{N-CVaR}_{0,1}, \text{ as } \alpha \to 0.
$$
Using this and the fact that the decisions are bounded, it follows that the decisions as well as the objective values converge (see Shapiro et al. 2009, Proposition 7.26).

The above result shows that the greedy rolling intrinsic solution can be regarded as an optimal policy for the N-CVaR problem under extreme risk aversion. Therefore, the N-CVaR formulation in (20) contains the risk-neutral value as well as the rolling intrinsic value as two extreme cases which provide a decision-makers with an upper and lower bound on the value of storage.

3. Method

To mitigate the curse of dimensionality, we exploit the property that the state space of the MDP can be separated into an environmental state, $F_t$, which is exogenously given but random, and a resources state, $R_t$, which is determined through the decision policy.

The separation enables us to split the learning process into two steps. While the first step is about searching for an optimal set of representative discrete states for the environmental state, the second step involves building a piecewise-linear approximation of the value function for each of the previously selected discrete states.

This procedure has several advantages. Since the first step does not involve making decisions, we can quickly simulate many state transitions, possibly millions, to find a representative discretization of the stochastic process. In contrast, not discretizing the resource state, mitigates the curse of dimensionality to some extent, because it eliminates the need to find a representative set of discrete states in high dimensional resource space. Instead, we exploit the fact that the value function is piecewise-linear and jointly concave in the resource state for a given environmental state of the problem.

In Section 3.1 we describe how to learn a discretization of the continuous-time, continuous-state price process via optimal quantization. We also introduce an improvement step based on matching the means of the discrete process with the means of the continuous process, which significantly improves approximation quality, as will be shown in Section 4.

Since we discretize prices but not the resource state, the use of classic dynamic programming techniques is not possible. Instead, we apply ADDP, an iterative decomposition algorithm, to
learn a piecewise-linear approximation of the value function. The algorithm learns to make better decisions from implementing a greedy policy that maximizes rewards under a given value function approximation. The algorithm updates the value function to improve the greedy policy, which thereby gets better with every iteration.

In Section 3.2 we describe the ADDP algorithm, analyze its convergence, and derive upper and lower bounds for the objective value.

3.1. The Lattice Quantization Algorithm

Our objective is to construct a discrete approximation of the stochastic process such that the optimal policy for the problem under the approximate process is also a good solution for the problem under the original process.

The fact that the process described in (1) has the Markov property allows us to construct approximations, which will be referred to as a lattice from hereon, in line with the terminology that is often used in mathematical finance in a similar context.

**Definition 1.** A lattice is a directed, acyclic graph, which is organized in a finite number of layers. Each layer is associated with a discrete point in time and contains a finite number of nodes. Successive layers are connected by arcs. A node represents a possible state of the stochastic process and an arc represents the possibility of a state transition from one node of a given layer to a successor node at the next layer. Each arc is associated with a probability weight, and weights of all outgoing arcs of a node sum to one.

Denote $N_t$ as the set of nodes in $t$ and $\tilde{F}_{tn}$, $n \in N_t$, as the state of the stochastic process at node $n$. Denote $\tilde{F}_t = \{\tilde{F}_{tn} : n \in N_t\}$ as the set of all possible states in one layer of the lattice. The universe of possible scenarios that is defined through the lattice is given by

$$\tilde{F}_1 \times \tilde{F}_2 \times \cdots \times \tilde{F}_{T-1} \times \tilde{F}_T,$$

which leads to $\prod_{t=1}^{T} |N_t|$ scenarios. Evidently the number of possible scenarios increases exponentially in $T$, which is the reason for the curse of dimensionality typically encountered when

$^1$ Note that this definition makes the tacit assumption that all combinations are in fact possible which is the case with MGBM.
using scenario trees. Lattices by contrast only grow moderately in size as $T$ is increased, which makes them attractive for discretization of stochastic processes. See Figure 1 for an illustrative comparison.

Denote $p_t$, $t = 1, \ldots, T - 1$ as the $|N_t| \times |N_{t+1}|$ transition matrix with elements $p_{tnm}$ which defines the probability of a state transition from $\bar{F}_{tn}$ to $\bar{F}_{t+1,m}$. Assume that the initial state of the lattice is deterministic with marginal probability $q_{1,1} = 1$, and denote the corresponding root node by $\bar{F}_{1,1}$. Then, the unconditional probability of all nodes in $t > 2$, can be recursively defined as

$$q_{tm} = \sum_{k=1}^{N_{t-1}} q_{t-1,k} \cdot p_{t-1,k,m}, \quad m = 1, \ldots, N_{t+1}, \quad t = 2, \ldots, T.$$ 

(A22)

A theoretical framework that enables us to measure the discretization quality is the theory of optimal mass transport and probability metrics. For two probability measures $P$ and $Q$ and $r \geq 0$, the mass transportation problem can be written as

$$\min_z \int ||x - y||^r z(dx, dy)$$

s.t. $z(A \times \mathbb{R}^T) = P(A), \forall A$ measurable

$$z(\mathbb{R}^T \times A) = Q(A), \forall A$$ measurable.

(A23)

The decision variable $z$ in the above optimization problem is a measure and is called a transport plan. The objective value is the Wasserstein metric between the two measures $P$ and $Q$. Many authors advocate the use of the above probability metrics to find suitable approximations for probability measures in stochastic programming (Pflug 2001, Dupačová et al. 2003, Pflug and Pichler 2012, Bally and Pagès 2003). The problem of finding a discrete probability measure that is as close as possible in terms of the transportation cost to a given measure $P$ is called the
Optimal gas storage valuation under a high-dimensional price process

problem of optimal quantization (Graf and Luschgy 2000). It can be shown that the problem of optimal quantization is equivalent to choosing atoms \(q_i\) of the approximating measure. The transport plan then follows by transporting every point to its closest atom. The points that are transported to the same atom \(q_i\) form a Voronoi cell \(V(q_i)\).

Denote by \(P_{t+1}(\cdot|F_t = \bar{F}_{tn})\) the image measure of \(F_{t+1}\) in \(\mathbb{R}^T\) given \(F_t = \bar{F}_{tn}\). Consequently, conditional on the nodes of the lattice at stage \(t\) with unconditional probabilities \(q_{tn}\), the distribution of \(F_{t+1}\) is given by

\[
P_{t+1}(A) = \sum_{n \in N_t} q_{tn} P(A|\bar{F}_t = \bar{F}_{tn}), \quad \forall A \subseteq \mathbb{R}^T \text{ measurable.} \tag{24}
\]

Hence, \(P_{t+1}\) is the image measure of the random variable, which is the value of the MGBM starting at a random node of the lattice \(\bar{F}_{tn}\) in period \(t\). Given the nodes \(\bar{F}_{tn}, n \in N_t\) at stage \(t\), we want to find \(|N_{t+1}|\) nodes \(\bar{F}_{t+1,m}\) such that the probability weighted sum of the mass transport problems is minimized, i.e.,

\[
\min_{z_{\bar{F}_{t+1}}} \sum_{n \in N_t} q_{tn} \int ||\bar{F}_{t+1} - F_{t+1}||^p z_{n}(d\bar{F}_{t+1}, dF_{t+1})
\text{s.t.} \quad z_n(\mathbb{R}^T \times A) = P_{t+1}(A|\bar{F}_t = \bar{F}_{tn}), \quad \forall A, \forall n \in N_t
\]

\[
z_n(\{ F_{t+1,m} \} \times \mathbb{R}^T) = P_{t+1}(V(\bar{F}_{t+1,m})|\bar{F}_t = \bar{F}_{tn}), \forall m \in N_{t+1}, \forall n \in N_{k}, \tag{25}
\]

where \(z_n\) is the transportation measure between \(P\) and the distribution of \(\bar{F}_{t+1}\) given \(\bar{F}_t = \bar{F}_{tn}\) on the lattice. This ensures that the conditional distributions of \(F_{t+1}\) given the discretization in state \(t\) are captured as accurately as possible, whereby the trade-off between the \(|N_t|\) different transportation problems embedded in the above problem is determined by the unconditional probabilities \(q_{tn}\).

To efficiently solve (25), we reformulate the problem using the following proposition.

**Proposition 2** Problem (25) is equivalent to the problem of transporting the unconditional distributions \(P_{t+1}\) at stage \(t+1\) to a single discrete distribution, i.e.,

\[
\min_{z_{\bar{F}_{t+1}}} \int ||\bar{F}_{t+1} - F_{t+1}||^p z(d\bar{F}_{t+1}, dF_{t+1})
\text{s.t.} \quad z(\mathbb{R}^T \times A) = P_{t+1}(A), \quad \forall A, \forall n \in N_t
\]

\[
z(\{ F_{t+1,m} \} \times \mathbb{R}^T) = P_{t+1}(V(\bar{F}_{t+1,m})), \forall m \in N_{t+1}. \tag{26}
\]

**Proof.** We start by showing that every solution of (26) can be transformed to a solution of (25) with the same objective. In fact, for every solution \(z\) to problem (26), we can consider the
disintegration of \( z \) with respect to the projection on the second coordinate, i.e., the conditional distributions, and rewrite the objective as

\[
\int ||\tilde{F}_{t+1} - F_{t+1}||^p \, \mu(\tilde{F}_{t+1}, dF_{t+1}) = \int \int ||\tilde{F}_{t+1} - F_{t+1}||^p \, \nu_\lambda(\tilde{F}_{t+1}, dF_{t+1}) \lambda(dF_{t+1})
\]

\[
= \sum_{n \in N_t} q_{tn} \int \int ||\tilde{F}_{t+1} - F_{t+1}||^p \, \nu_\lambda(\tilde{F}_{t+1}, dF_{t+1}) P_{t+1}(d\lambda|F_t = \tilde{F}_{tn})
\]

\[
= \sum_{n \in N_t} q_{tn} \int ||\tilde{F}_{t+1} - F_{t+1}||^p \, \nu(z|F_t = \tilde{F}_{tn})
\]

with

\[
z_n(A \times B) = \int \int 1_{A \times B} \, \nu_\lambda(\tilde{F}_{t+1}, dF_{t+1}) P_{t+1}(d\lambda|F_t = \tilde{F}_{tn}).
\]

Since

\[
z_n(R^T \times A) = \int \int 1_{R^T \times A} \, \nu_\lambda(\tilde{F}_{t+1}, F_{t+1}) \, \nu(z_{t+1}(\tilde{F}_{t+1}, dF_{t+1+1}) P_{t+1}(dF_{t+1}|F_t = \tilde{F}_{tn})
\]

\[
 = \int 1_A \, (F_{t+1}) P_{t+1}(dF_{t+1}|F_t = \tilde{F}_{tn}) = P_{t+1}(A|F_t = \tilde{F}_{tn})
\]

\[
z_n(\{\tilde{F}_{t+1,m}\} \times R^T) = \int \int 1_{\{\tilde{F}_{t+1,m}\} \times R^T} \, \nu_\lambda(\tilde{F}_{t+1}, F_{t+1}) \, \nu(z_{t+1}(\tilde{F}_{t+1}, dF_{t+1+1}) P_{t+1}(dF_{t+1}|F_t = \tilde{F}_{tn})
\]

\[
 = \int z_{t+1}(\{\tilde{F}_{t+1,m}\} \times \{F_{t+1}\}) P_{t+1}(dF_{t+1}|F_t = \tilde{F}_{tn})
\]

\[
 = P_{t+1}(V(\tilde{F}_{t+1,m})|F_t = \tilde{F}_{tn})
\]

the measures \( z_n \) are a feasible solution of \((25)\).

To show the other direction, let \( (z_n)_{n \in N_t} \) be a solution of \((26)\) and define

\[
z(A \times B) = \sum_{n \in N_t} q_{tn} z_n(A \times B).
\]

Note that the objective function \((26)\) equals

\[
\int ||\tilde{F}_{t+1} - F_{t+1}||^p \, \nu(z_{t+1}, dF_{t+1}).
\]

To show that \( z \) is a feasible solution to \((26)\), note that

\[
z(R^T \times A) = \sum_{n \in N_t} q_{tn} z_n(R^T \times A) = \sum_{n \in N_t} q_{tn} P_{t+1}(A|F_t = \tilde{F}_{tn}) = P_{t+1}(A)
\]

\[
z(\{\tilde{F}_{t+1,m}\} \times R^T) = \sum_{n \in N_t} q_{tn} z_n(\{\tilde{F}_{t+1,m}\} \times R^T) = \sum_{n \in N_t} q_{tn} P_{t+1}(V(\tilde{F}_{t+1,m})|F_t = \tilde{F}_{tn})
\]

\[
 = P_{t+1}(V(\tilde{F}_{t+1,m}))
\]

which finishes the proof.  \( \square \)
The advantage of (26) over (25) is that we have to solve only one quantization problem per stage instead of $|N_t|$ problems, which allows us to rely on existing methods for optimal quantization.

It is well known that optimal quantization belongs to the class of $\mathcal{NP}$-hard problems and only local solutions can be found efficiently. A common solution strategy in this regard is stochastic gradient descent (Ahalt et al. 1990, Bally and Pagès 2003).

Denote $\beta$ as the stepsize of the stochastic gradient update after each state transition, with $\beta(i; h) \in (0, 1]$. Let us assume that the stepsize is given by

$$\beta(i; h) = \frac{h}{h + i - 1}, \ h \geq 1. \quad (27)$$

See Powell (2011) for a detailed overview of different stepsize rules. In Pagès and Printems (2003), Theorem 2.4, it is shown that if the sequence $(\beta(i; h))_{i=1}^{N_t}$ satisfies $\sum_{i=1}^{\infty} \beta(i; h) = \infty$ and $\sum_{i=1}^{\infty} \beta^2 < \infty$ and if $F_t|F_{t-1}$ is continuously differentiable, then the resulting quantizers converge to local optimizers of (26).

The lattice quantization learning (LQL) algorithm proposed in this study applies gradient descent to reduce an MGBM process to a lattice. The algorithm searches for an optimal quantization of the conditional distributions which are given through the nodes of the lattice. In contrast, Bally and Pagès (2003) propose a quantization learning algorithm for the unconditional distributions.

The LQL algorithm is outlined in Figure 2. As arguments, the algorithm requires an initial state, the number of desired nodes at each layer of the lattice, and access to the state transition function of the Markov process. All parameters of the lattice are initialized with zero.

For each layer of the lattice, the algorithm simulates $I$ state transitions by choosing a random node as the initial state. The probability of a node being chosen is proportional to its unconditional probability (which is equal to one for the root node). In step (1.3) the simulated successor state is assigned to its nearest node which is moved towards this state by a small amount (the stochastic gradient learning step). The probability of a state transition from the predecessor node to all successor nodes is updated in a similar manner. After $I$ state transitions
Input arguments: \( N = \{1, |N_2|, \ldots, |N_T|\}, \beta(\cdot; h), F_t(\cdot), \hat{F}_{1,1}, q_{1,1} = 1, \bar{F}_i = 0, p_i = 0, t = 2, \ldots, T \)

Do for \( t = 2, 3, \ldots, T \)

**Forward simulation**

(1) Do for \( i = 1, 2, \ldots, I \)

(1.1) Draw \( \hat{u} \sim U(0, 1) \) and set \( n \leftarrow \arg \max_k \left\{ \sum_{j=1}^{k} q_{t-1,j} \leq \hat{u} : k \in N_i \right\} \)

(1.2) Draw \( \hat{z}_{t-1,j} \sim N(0, 1) \), \( j = 1, \ldots, T \) and set \( \hat{F}_i \leftarrow F_i(\bar{F}_{t-1,n}, \hat{z}_{t-1,1}, \ldots, \hat{z}_{t-1,T}) \)

Update centers and transition probabilities

(1.3) Set \( m \leftarrow \arg \min_k \left\{ ||\bar{F}_{t+1,m} - \bar{F}_{t+1,k}||_2^2 : k \in N_i \right\} \)

(1.4) Update \( \bar{F}_{tm} \leftarrow \bar{F}_{tm} + \beta(i; h)(\hat{F}_i - \bar{F}_{tm}) \)

(1.5) Update \( p_{t-1,nk} \leftarrow p_{t-1,nk} + \beta(i; h)(1_k = m - p_{t-1,nk}) \), \( k \in N_t \)

(2) Compute \( q_n = \sum_{k \in N_t} q_{t-1,k} p_{t-1,kn}, n \in N_t \)

Return \( \bar{F}_i, p_i, t = 2, \ldots, T \)

---

**Figure 2**  Lattice quantization learning.

---

have been simulated, we obtain a transition matrix from all nodes in this layer to nodes in the subsequent layer. The unconditional probabilities of all successor nodes at the subsequent layer can be obtained from the transition matrix.

3.1.1. Matching Conditional Means  The LQL algorithm does not guarantee that the mean of the successor nodes of a lattice node \( \bar{F}_{tn} \) corresponds to the mean of the price process if \( \bar{F}_{tn} \) was used as an initial state. A risk-neutral decision maker would try to utilize differences in means that may not be present in the actual price process, which leads to biased policies that may perform poorly under the original MGBM process.

To remove this bias, the parameters of the lattice must be adjusted in such a way, that the expected movement of the forward curve at each node corresponds to the expected movement under the original process. This can be achieved by modifying the transition probabilities at each node.

We adjust the transition probabilities associated with node \( n \) so that the expectations match as closely as possible. Denote \( p'_{tnm} \) as the adjusted transition probability from node \( n \) in \( t \) to node \( m \) in \( t+1 \). To find these updated probabilities, we solve the following linear optimization problem for every node \( n \in N_t \):

\[
\min \quad \sum_{m,k \in N_{t+1}} ||\bar{F}_{t+1,m} - \bar{F}_{t+1,k}||_2^2 \ z_{mk} + M \sum_{j=1}^{T} (e_j^+ + e_j^-) \tag{28}
\]
The above optimization problem modifies the probabilities in such a way that the expectation at node \( n \) equals the expectation of the MGBM, conditional on \( \bar{F}_{tn} \) as modelled in (32). To keep the problem feasible, we introduce the two slack variables \( e^+_j \) and \( e^-_j \), which are penalized with a large enough \( M \) in the objective, e.g., the distance between the expected successor state as given by the lattice and the expected successor state as given by the process, 

\[
M = ||E(F_{t+1,j}|F_t = \bar{F}_{tn}) - E(\bar{F}_{t+1,j})||^2_2. \tag{35}
\]

The \( z_{mk} \) represent the transportation plan between the old and the new distribution which is used in the first term of the objective to model the Wasserstein distance between the two measures, which is kept at a minimum. The above formulation attempts to find a distribution with the correct first moments with minimum deviation from the optimal distribution in terms of the Wasserstein metric.

We can include the probability adjustment after step (1) of the LQL algorithm just before the marginal probabilities of the successor nodes are being computed. This ensures that the conditional expectation of all successor nodes matches the conditional expectations of the movement of the forward curve over time. This property is supported by the following proposition.

**Proposition 3** Let \( (F_t)_{0 \leq t \leq T} \) be a GBM process that is discretized to a lattice process \( (\bar{F}_t)_{t=1,...,T} \). Assume that the lattice nodes and probabilities are such that

\[
E(\bar{F}_{t+r}|\bar{F}_{t+r-1} = \bar{F}_{t+r-1,j}) = E(F_{t+r}|F_{t+r-1} = F_{t+r-1,j}), \quad \forall \ j \in N_{t+r-1}, \forall \ r = 1,...,s,
\]

then

\[
E(\bar{F}_{t+s}|\bar{F}_t = \bar{F}_{t,k}) = E(F_{t+s}|F_t = \bar{F}_{t,k}), \quad \forall \ k \in N_t.
\]
Proof. W.l.o.g. \( (F_t)_{0 \leq t \leq T} \) takes values in \( \mathbb{R} \). Note that because the process is GBM, we have
\[
E(F_{t+s}|F_t = x) = e^{s\sigma^2/2}x.
\]
We show the result for \( s = 2 \), the general case follows by induction.
\[
E(\bar{F}_{t+2}|\bar{F}_t = \bar{F}_{t,k}) = \sum_{j \in \mathbb{N}_{t+1}} p_{k,j} \sum_{m \in \mathbb{N}_{t+2}} p_{j,m} \bar{F}_{t+2,m} = \sum_{j \in \mathbb{N}_{t+1}} p_{k,j} E(\bar{F}_{t+2}|\bar{F}_{t+1} = \bar{F}_{t+1,j})
= \sum_{j \in \mathbb{N}_{t+1}} p_{k,j} e^{\sigma^2/2} \bar{F}_{t+1,j} = e^{\sigma^2/2} \sum_{j \in \mathbb{N}_{t+1}} p_{k,j} \bar{F}_{t+1,j} = e^{\sigma^2/2} E(\bar{F}_{t+1}|\bar{F}_t = \bar{F}_{t,k})
=e^{\sigma^2/2} e^{\sigma^2/2} \bar{F}_{t,k} = E(F_{t+2}|F_t = \bar{F}_{t,k}) \quad \Box
\]
In our numerical analysis, we will see that this property provides a tremendous advantage in finding good policies. Moreover, this presents the first approach to merge elements from moment matching with probability metrics in a multistage setting.

3.2. Approximate Dual Dynamic Programming

Given a lattice, the ADDP algorithm learns the post-decision value functions by iteratively simulating the decisions of the current policy and then updating the value functions around resource states, which have been visited by the decision process.

With ADDP, each node of the lattice holds its own post-decision value function. In contrast, SDDP and related algorithms require only one post-decision value function per stage, since cuts can be shared among scenarios. This restricts SDDP to stochastic processes with stagewise independent randomness. ADDP does not have this restriction at the expense of having to store one value function at each node.

Denote \( \hat{V}_t \) as the approximate post-decision value function (VFA), i.e.,
\[
\hat{V}_t(\hat{F}_{tn}, R) \approx \hat{V}_t(\hat{F}_{tn}, R), \quad n \in \mathbb{N}_t, \quad t = 1, \ldots, T - 1.
\]
(36)

Suppose we have piecewise linear concave approximations of the post-decision value functions \( R \mapsto \hat{V}_{t+1}(\hat{F}_{tn}, R) \) for the lattice nodes \( \hat{F}_{tn} \) at stage \( t \) and a piecewise-linear approximation \( R \mapsto \hat{V}_t(\hat{F}_{t-1,k}, R) \) for the lattice node \( \hat{F}_{t-1,k} \) at stage \( t - 1 \)
\[
\hat{V}_t(\hat{F}_{t-1,k}, R) = \min_{i \in \mathbb{Z}^k} \{ a_i(\hat{F}_{t-1,k}) + b_i(\hat{F}_{t-1,k})^\top R \}.
\]
(37)
Given a resource state \( R_t \), we can solve the approximate problems for each node \( \hat{F}_{tn} \) at stage \( t \)

\[
a_n = \max_{z_t \in z_t(R_{t-1})} \left\{ C(F_{tn}, z_t) + \hat{V}_{t+1}(\hat{F}_{tn}, R_{t+1}(z_t)) \right\}
\]

and obtain a slope vector \( b_n \) for the value function \( \hat{V}_t \) at point \( R_t \) from the dual multipliers of the constraints (5), (6), (7), and (8) with the resource variables \( R_t \) in the right-hand side.

The resulting supergradients can now be used to improve the post decision value function approximation by adding the cut with intercept \( a|I_{n}|+1(F_{t-1,k}) \) and slope \( b|I_{n}|+1(F_{t-1,k}) \)

\[
a|I_{n}|+1(F_{t-1,k}) = \sum_n p_{kn}(a_n - b_n^T R_t), \quad b|I_{n}|+1(F_{t-1,k}) = \sum_n p_{kn} b_n
\]

to the piecewise linear representation of \( \hat{V}_t(F_{t-1,k}, \cdot) \), i.e., redefining

\[
\hat{V}_t(F_{t-1,k}, R) = \min_{i \in I^{k \cup \{[I_{n}]-1\}}} \left\{ a_i(F_{t-1,k}) + b_i(F_{t-1,k})^T R \right\}.
\]

The resource states, which serve as the tangential points of the hyperplanes of the piecewise-linear approximation in (38), are determined during forward passes of the MDP. After a sample of resource states has been collected during a forward pass, all VFAs are updated recursively beginning with the terminal nodes, backwards through time, until the root node is reached.

The algorithm iterates between a forward pass and a backward pass until the pre-decision value of the root node sub-problem has converged to the sample average of the discounted rewards obtained through simulating the decision process. See Figure 3 for a detailed description of the full algorithm.

With \( S^M \) as the state transition function, a sample state transition from node \( n \) to a possible successor node of \( n \) is given by

\[
S^M(n, t; N_t, P_t) = \arg \max_k \left\{ \sum_{j=1}^k p_{tnj} \leq \hat{u}, k \in N_t \right\}, \quad \hat{u} \sim U(0, 1).
\]

Note that in step (3.2) of the algorithm new cuts are only added if they improve the VFA by at least \( \varepsilon \). If \( \varepsilon \) is set to zero, then each node will contain a value function consisting of exactly \( n \) cuts after the \( n \)-th iteration. In contrast, if \( \varepsilon > 0 \), then some cuts will be discarded, which speeds up computation times of the optimization problems at the expense of having a slightly looser approximation.
Input arguments: initial states $F_1$ and $R_1$, initial approximation $\hat{V}_t$, $t = 1, \ldots, T - 1$, $\varepsilon \geq 0$

Do for $i = 1, 2, \ldots, M$

**Forward Pass**

1. Begin with root node $n \leftarrow 1$
2. Do for $t = 1, 2, \ldots, T - 1$
   1. Solve $R_{t+1} = \arg \max_{\pi_t \in \Pi_t(R_t)} \{ C(F_{tn}, \pi_t) + \gamma_t \hat{V}_{t+1}(F_{t+1}, \pi_t) \}$
   2. Draw successor node $n \leftarrow S_n(n, t, \hat{u}; N_t, P_t)$

**Backward Pass**

3. Do for $t = T, T - 1, \ldots, 2$
   1. Do for $n \in N_t$
      1.1. Get hyperplane $(a_{|I_{tn}|+1}(F_{tn}), b_{|I_{tn}|+1}(F_{tn}))$ as in (39)
      1.2. If $|\hat{V}_t(F_{t-1}, R_t) - a_{|I_{tn}|+1}| > \varepsilon$ then
         1.2.1. $I_{tn} \leftarrow I_{tn} \cup \{ |I_{tn}| + 1 \}$
         1.2.2. $\hat{V}_t(F_{t-1}, R) \leftarrow \min_{k \in I_{tn}} \{ a_i(F_{t-1,k}) + b_i(F_{t-1,k})^T R \}$

Return post-decision value functions $\hat{V}_t (t = 1, \ldots, T - 1)$

**Figure 3** Single-cut approximate dual dynamic programming.

The algorithm in Figure 3 is similar to the algorithm used in Löhndorf et al. (2013) for the dispatch optimization of hydro plants. In contrast to the algorithm in Löhndorf et al. (2013), cuts in VFAs are not updated with new information once they are added, i.e., the only cuts that are calculated in each iteration are the newly added cuts around the sample resource states from the forward pass. While this clearly slows down convergence, it also reduces the computation time of the backward pass, particularly in models that take many iterations to converge and hence require more cuts. For the test problems considered for this study, the single cut strategy proved to be significantly faster. We term this variant of the algorithm **single cut ADDP**.

Let us establish the theoretical properties of the proposed algorithm.

**Proposition 4**

1. In any given iteration, the VFAs are upper bounds for the real value functions of the problem.

2. The policies obtained from single cut ADDP converge to the optimal policies in finitely many iterations.

**Proof.** Note that the function $R_T \mapsto \bar{V}_{T-1}(F_{T-1,n}, R_T)$ is concave, since the starting resource state only appears in the RHS of the optimization problem at the last stage. The approximating
cuts are supergradients, and therefore upper bounds for the function. The minimum of these upper bounds is still an upper bound. An inductive argument over the stages establishes 1.

To prove 2, note that since the problems on the nodes are linear, the functions $R_T \mapsto \bar{V}_{T-1}(F_{T-1,n}, R_T)$ are actually piecewise-linear, and therefore equal to the minimum of a finite number of supergradients (see Philpott and Guan 2008, Lemma 1, and Remark 2 above). Assume that the algorithm stops adding cuts to the VFA after iteration $n \in \mathbb{N}$ and there is a sampling path $(\bar{F}_{1,n_1}, \ldots, \bar{F}_{T-1,n_{T-1}})$ such that $\bar{F}_{T-1,n_{T-1}} = \bar{F}_{T-1,n}$ leading to a resource state $R_T$ at the end of period $T-1$ with $\bar{V}_{T-1}(F_{T-1,n}, R_T) < \bar{V}_{T-1}(F_{T-1,n}, R_T)$. Then, by the Borel-Cantelli Lemma, the sequence $(F_1, n_1, \ldots, F_{T-1,n_{T-1}})$ will be sampled in an iteration $n' > n$, and therefore a new cut will be added at $R_T$, which is a contradiction to the choice of $n$. □

Remark 3. The resource states that are sampled in the forward passes are optimal with respect to the current VFA. Since in the first iteration the VFA is inexact, the sampled resource states tend to be in regions where the upward bias of the VFA is the greatest. Therefore, this sampling strategy can be seen as an exploitation strategy. This results in the VFA being more accurate in regions of the state space that get sampled by the optimal policy and less accurate in other regions.

As is common with SDDP type algorithms, we can obtain upper and lower bounds on the solution in every iteration of the algorithm. In particular, since the VFAs obtained by ADDP are upper bounds for the real value functions, the upper bound of the objective value on the lattice is called the lattice upper bound (LUB) and is given by

$$LUB = V_1(F_1, R_1) = \max_{\pi_1 \in \Pi_1(R_1)} \left\{ C(F_1, \pi_1, R_1) + \hat{V}_2(F_1, R_2) \right\}.$$

The LUB is an upper bound for the MDP on the lattice, i.e., for the problem where the continuous process is replaced by its discrete counterpart. Note that the LUB is not a bound on the objective value of the MDP on the continuous process.

To obtain a lower bound, we can perform $N$ forward passes of the algorithm, compute the obtained revenues $\Pi_n$, $n = 1, \ldots, N$ and then take an average to obtain the lattice lower bound (LLB) defined as
\[ LLB = N^{-1} \sum_{n=1}^{N} \Pi_n \] (42)

Note that, the LLB is itself random, as it is derived from a sample of state transitions of the lattice.

**Remark 4.** The difference between the LUB and the LLB can be used to define stopping criteria (Shapiro 2011). To be able to compare computation times across instances, we decided to manually select a fixed number of iterations that is large enough to ensure statistical convergence for the selected test instances.

### 3.3. Risk Measures

To solve the N-CVaR problem, we adapt the solution approach proposed by Philpott et al. (2013) for SDDP to the ADDP algorithm. In contrast to other approaches, their method neither requires a reformulation of the model nor does it increase the dimension of the state space.

For a discrete random variable \( X \) with possible realizations \( X_1, \ldots, X_M \) and corresponding probabilities \( p_1, \ldots, p_M \), the risk measure \( \rho_{\alpha,\lambda} \) has a dual representation as

\[ \rho_{\alpha,\lambda}(X) = \min_{\xi \in \mathcal{U}} \sum_{i=1}^{M} p_i \xi_i X_i \] (43)

where

\[ \mathcal{U} = \left\{ \xi \in \mathbb{R}^M : \sum_{i=1}^{M} \xi_ip_i = 1, \xi_i = (1 - \lambda) + \lambda \eta_i, 0 \leq \eta_i \leq \alpha^{-1}, \forall 1 \leq i \leq M \right\}. \] (44)

Given a lattice node \( \bar{F}_{tn} \) at stage \( t \), we can write

\[ \rho_{\alpha,\lambda}(V_{t+1}((\bar{F}_{t+1}, R_{t+1}(\pi_T)))) = \min_{\xi \in \mathcal{U}} \sum_{m \in N_{t+1}} p_{nm} \xi_m V_{t+1}((\bar{F}_{t+1,m}, R_{t+1}(\pi_T))). \] (45)

Hence, the \( \rho_{\alpha,\lambda} \) can be viewed as an expectation with a changed probability measure. The supergradients needed by the ADDP algorithm can be constructed from this representation by noting that

\[ \partial_R \rho_{\alpha,\lambda}(V_{t+1}(\bar{F}_{t+1}, R)) = \sum_{m \in N_{t+1}} p_{nm} \xi^*_m \partial_R V_{t+1}(\bar{F}_{t+1,m}, R) \] (46)

where \( \xi^* \) is optimal for \( R \) in (45) (see Philpott et al. 2013).

Since the optimal weights are easily found by sorting the values \( V_{t+1}(\bar{F}_{t+1,m}, R) \), the above can be used to construct value function approximations during the backward pass of the ADDP
similar to the expectation case. This is an improvement over alternative methods of handling the N-CVaR in SDDP type algorithms. However, the sample average of the discounted rewards is not a lower bound of the N-CVAR objective, since the forward passes only sample the immediate rewards and not the risk penalty in the objective. Philpott et al. (2013) propose a method to circumvent this problem, but the method is computationally expensive, and therefore will not be considered for this study.

4. Numerical Results

4.1. Instances

For our numerical analysis, we use the exact same parameters as reported in Lai et al. (2010) and its electronic companion. Their setup combines real price data with storage contract characteristics from the energy literature. Three storage characteristics, called A, B, and C, are used, which differ in injection capacity and the monthly withdrawal limit.

The authors calibrate four versions of the MGBM price process from Section 2.2, each using the information available at the closing of NYMEX Henry Hub on 3/1/2006 (Spring), 6/1/2006 (Summer), 8/31/2006 (Fall), and 12/1/2006 (Winter). The initial state of the MGBM, $F_1$, is given by the spot price and the futures prices of the first 23 maturities on each of the four days. Implied volatilities of the 23 futures prices are obtained from prices of NYMEX call options on natural gas futures. The correlation matrix is constructed from the historical correlation of the first 23 maturities of each trading day from 1/2/1997 to 12/14/2006. Accordingly, the interest rates as reported by the Dept. of Treasury on the four selected days define the monthly discount factor of the model, which is given in (4).

All constant model parameters are summarized in Table 1. If not stated otherwise, model instance A with Spring prices will serve as our base case.

4.2. Implementation

ADDP and LQL are implemented in Java 7 with the linear programs being solved by Sulum. The implementation of the algorithms is available as a Java library called QUASAR (http://www.quantego.com). For ease of use, QUASAR can be accessed as a web application.
based on iPython, which was also used to perform the numerical analysis described in this section. Computations were conducted remotely on a Dual-Xeon E5-2650 with 16 cores and 128g memory.

### 4.3. Algorithm Settings

If not stated otherwise, the ADDP algorithm was run for 200 iterations and a lattice with 500 nodes and conditional moment matching is used as the base case. ADDP rejects cuts that are only $\varepsilon = 0.0001$ tighter than the approximated value.

The lattices with 5, 50, and 500 nodes per stage are constructed in such a way that the number of nodes increases approximately in a square-root fashion from one stage to the next to account for the increasing standard deviations of the unconditional distributions of the MGBM process. Therefore, the number of terminal nodes is greater than the number of nodes at the second stage, e.g., the 5-node lattice has seven nodes at $t = 24$, but only two nodes at $t = 2$.

The lattice with 5, 50, and 500 nodes per stage (on average) was built by sampling 50k, 500k, and 5m state transitions from the MGBM, respectively. The stepsize parameter, $h$, of the learning algorithm was set to 50, 500, and 5000, respectively.

The states of the MDP have 23 dimensions in the environmental variables and 1 to 23 dimensions in the resource variables, which is clearly out of scope for conventional methods of stochastic dynamic programming. If we were to replace a lattice with a scenario tree, we would

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notation</th>
<th>Instance</th>
<th>Value(s)</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage limit</td>
<td>$\pi$</td>
<td></td>
<td>1.00</td>
<td>mmBtu</td>
</tr>
<tr>
<td>Initial resource state</td>
<td>$R_1$</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Injection (withdrawal) limit</td>
<td>$i, \bar{w}$</td>
<td>A</td>
<td>0.15 (0.30)</td>
<td>mmBtu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B</td>
<td>0.30 (0.60)</td>
<td>mmBtu</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C</td>
<td>0.45 (0.90)</td>
<td>mmBtu</td>
</tr>
<tr>
<td>Injection cost</td>
<td>$c^i$</td>
<td></td>
<td>0.02</td>
<td>$/mmBtu</td>
</tr>
<tr>
<td>Withdrawal cost</td>
<td>$c^w$</td>
<td></td>
<td>0.01</td>
<td>$/mmBtu</td>
</tr>
<tr>
<td>Injection loss</td>
<td>$d^i$</td>
<td></td>
<td>0.99</td>
<td>mmBtu/mmBtu</td>
</tr>
<tr>
<td>Withdrawal loss</td>
<td>$d^w$</td>
<td></td>
<td>1.01</td>
<td>mmBtu/mmBtu</td>
</tr>
<tr>
<td>Interest rates</td>
<td>$\delta$</td>
<td>Spring</td>
<td>4.74</td>
<td>% p.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Summer</td>
<td>5.05</td>
<td>% p.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fall</td>
<td>5.01</td>
<td>% p.a.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Winter</td>
<td>4.80</td>
<td>% p.a.</td>
</tr>
<tr>
<td>Time horizon</td>
<td>$T$</td>
<td></td>
<td>24</td>
<td>month</td>
</tr>
</tbody>
</table>

Table 1 Parameters for the numerical study.
obtain trees with $2.82 \times 10^{15}$, $2.15 \times 10^{38}$, or $2.15 \times 10^{61}$ terminal nodes to replace the 5, 50, or 500-node lattices, respectively. None of these would fit into the memory of a modern computer.

4.4. Simulation

The rolling intrinsic policy can be obtained by simulating the MDP defined in Section 2 for a sample of prices but without a value function, i.e., $\bar{V}_t \equiv 0$, $t = 1, \ldots, T$. The rolling intrinsic policy thereby serves as benchmark to which we can compare the proposed solution approach.

One has to note that our rolling intrinsic solution does not consider future injection and withdrawal cost as part of the wait-and-see model. In this way, we avoid accumulating cost that never incur during forward simulation, but at the same time we relax our intrinsic solution, which evidently affects the decisions under the rolling intrinsic policy. For example, the intrinsic value for Spring prices and instance A as reported in Lai et al. (2010) is $3.69$, whereas we obtain a value of $3.75$. Since our results for the rolling intrinsic values deviate at most 2.2% from the ones in Lai et al. (2010), we decided to ignore the inaccuracy to be able to use the exact same model parameters for all policies.

ADDP solves the MDP using the lattice as approximation of the price process, so that the resulting policy is optimal for the discrete-state process but not necessarily for the continuous-state process. To obtain a feasible policy under the continuous-state process, we apply the following strategy:

1. Simulate a state transition from the continuous-state process.
2. Find the node on the lattice with minimal Euclidean distance to the simulated state.
3. Solve the subproblem associated with the given node.
4. Evaluate the objective value by replacing the lattice state with the simulated state.

The policy implements the exact same decisions as under the discrete-state process, but uses a different objective function to assess the performance.

If we execute this policy for $N$ forward passes of our algorithm, it provides us with a lower bound on the optimal value under the continuous process that we refer to as the process lower bound (PLB). The gap between the LLB and the PLB allows us to study the influence of lattice
quantization on solution quality. Moreover, if the gap closes with an increasing number of nodes in the lattice, we can conclude that our approach is capable of finding a near-optimal policy to the continuous-state MDP. This is in stark contrast to solutions typically obtained when using scenario trees, where, to the best of our knowledge, solution quality is never assessed with respect to the original stochastic process.

For policy evaluation, all policies are simulated by drawing 10k sample paths from the MGBM.

4.5. Influence of Lattice Quantization Learning

Our first analysis concerns the quality of the lattice quantization algorithm and its effect on the LLB/PLB gap.

Figure 4 shows 10k sample paths of the front month future from simulating the MGBM process versus simulating state transition from the lattice after quantization with 5, 50, and 500 nodes per stage (on average). Since the lines of the plot are transparent, price states that get visited frequently appear darker than less frequently sampled states.

All three versions of the lattice yield scenarios that cover the outcome space of the front month futures in regions of high probability density. Good representations of the tails of the unconditional distributions are only obtained with 50 and 500 nodes per stage.

This concerns only a single dimension of the process. To visually inspect the quality of the scenarios in 23-dimensional space, we conducted a principal components analysis (PCA) of the joint unconditional distribution of all future prices at stage $t = 24$ and created scatter plots of the first two principal components. The subplots in Figure 5 show the first two components of realizations from the MGBM as red dots and realizations from the lattice with 5, 50, and 500 nodes as black squares. Dots and squares are again transparent to better distinguish regions of the outcome space with different sampling frequencies.

With only 5 nodes per stage (on average), the LQL algorithm puts all emphasis on capturing the variance of the distribution in the direction of the first principal component. With 50 nodes (on average), LQL also puts some emphasis on the variance in the direction of the second
principal component, but it still does not provide scenarios for the tails in this direction. Only with 500 nodes (on average), the tails of unconditional distribution get covered along the first two components.

In summary, both Figure 4 and 5 show that LQL is capable of creating a lattice that retains important properties of the original stochastic process. Moreover, if the number of nodes is chosen to be large enough, the lattice also provides scenarios for the tails of the unconditional distributions, which is an important feature when optimizing under a risk measure.

Our final analysis concerns the quality of lattice quantization by directly addressing the gap between the LLB and the PLB. For the analysis, we only considered problem instance A. We generated lattices with 5, 50, and 500 nodes with and without matching means and ran ADDP for 200 iterations, which was sufficient to close the gap between the LUB and LLB in all cases. Next, we calculated the PLB using the optimal policy from the lattice. The computational results are summarized in Table 2.

The results indicate that matching means has a significant effect on the quality of the solution.
Figure 5  The first two principal components of the unconditional sample distribution of the MGBM at $t = 24$. 

<table>
<thead>
<tr>
<th>Nodes</th>
<th>MM</th>
<th>LUB</th>
<th>LLB (SE)</th>
<th>PLB (SE)</th>
<th>LPs solved</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>yes</td>
<td>6.325</td>
<td>6.306 (0.031)</td>
<td>6.326 (0.044)</td>
<td>23399</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>7.201</td>
<td>7.201 (0.027)</td>
<td>5.629 (0.032)</td>
<td>23399</td>
<td>6</td>
</tr>
<tr>
<td>50</td>
<td>yes</td>
<td>6.430</td>
<td>6.428 (0.042)</td>
<td>6.385 (0.043)</td>
<td>230199</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>7.101</td>
<td>7.120 (0.035)</td>
<td>5.649 (0.033)</td>
<td>230399</td>
<td>22</td>
</tr>
<tr>
<td>500</td>
<td>yes</td>
<td>6.565</td>
<td>6.592 (0.043)</td>
<td>6.563 (0.044)</td>
<td>2296399</td>
<td>1520</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>6.860</td>
<td>6.865 (0.038)</td>
<td>5.969 (0.037)</td>
<td>2300399</td>
<td>362</td>
</tr>
</tbody>
</table>

MM = matching means, LUB = lattice upper bound, LLB = lattice lower bound, PLB = process lower bound, SE = std error

Table 2  The effect of lattice size and quantization method on solution quality.
<table>
<thead>
<tr>
<th>Instance</th>
<th>Futures</th>
<th>Rolling intrinsic</th>
<th></th>
<th>Approximate optimum</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg value (SE)</td>
<td>CVaRα=0.05 (SE)</td>
<td>Avg value (SE)</td>
<td>CVaRα=0.05 (SE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spring</td>
<td>4.157 (0.001)</td>
<td>3.790 (0.001)</td>
<td>6.563 (0.044)</td>
<td>-5.408 (0.027)</td>
<td></td>
</tr>
<tr>
<td>Summer</td>
<td>4.638 (0.001)</td>
<td>4.198 (0.001)</td>
<td>7.285 (0.047)</td>
<td>-5.387 (0.028)</td>
<td></td>
</tr>
<tr>
<td>Fall</td>
<td>4.082 (0.002)</td>
<td>3.623 (0.001)</td>
<td>6.503 (0.044)</td>
<td>-5.934 (0.030)</td>
<td></td>
</tr>
<tr>
<td>Winter</td>
<td>1.714 (0.002)</td>
<td>1.195 (0.002)</td>
<td>2.811 (0.028)</td>
<td>-5.173 (0.020)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg value (SE)</td>
<td>CVaRα=0.05 (SE)</td>
<td>Avg value (SE)</td>
<td>CVaRα=0.05 (SE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spring</td>
<td>5.255 (0.002)</td>
<td>4.610 (0.002)</td>
<td>7.557 (0.051)</td>
<td>-6.284 (0.032)</td>
<td></td>
</tr>
<tr>
<td>Summer</td>
<td>6.270 (0.002)</td>
<td>5.629 (0.002)</td>
<td>8.774 (0.053)</td>
<td>-5.625 (0.033)</td>
<td></td>
</tr>
<tr>
<td>Fall</td>
<td>6.376 (0.002)</td>
<td>5.598 (0.003)</td>
<td>8.919 (0.052)</td>
<td>-5.475 (0.035)</td>
<td></td>
</tr>
<tr>
<td>Winter</td>
<td>2.442 (0.002)</td>
<td>1.647 (0.003)</td>
<td>3.429 (0.032)</td>
<td>-5.636 (0.022)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Avg value (SE)</td>
<td>CVaRα=0.05 (SE)</td>
<td>Avg value (SE)</td>
<td>CVaRα=0.05 (SE)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spring</td>
<td>5.745 (0.002)</td>
<td>4.870 (0.003)</td>
<td>8.013 (0.054)</td>
<td>-6.551 (0.034)</td>
<td></td>
</tr>
<tr>
<td>Summer</td>
<td>6.796 (0.003)</td>
<td>5.931 (0.003)</td>
<td>9.520 (0.058)</td>
<td>-5.806 (0.034)</td>
<td></td>
</tr>
<tr>
<td>Fall</td>
<td>7.539 (0.003)</td>
<td>6.468 (0.003)</td>
<td>10.25 (0.057)</td>
<td>-5.228 (0.037)</td>
<td></td>
</tr>
<tr>
<td>Winter</td>
<td>2.827 (0.003)</td>
<td>1.823 (0.004)</td>
<td>3.688 (0.034)</td>
<td>-5.975 (0.024)</td>
<td></td>
</tr>
</tbody>
</table>

Table 3  Comparison of rolling intrinsic and approximately optimal value.

When means remain unchanged, having more nodes decreases the gap between the LLB and PLB, although not even 500 nodes are sufficient to close it. On the other hand, when LQL matches conditional means there appears to be no significant gap between the LLB and PLB, for any of the used lattices. This implies that matching means is necessary for the algorithm to arrive at an objective value that can be reproduced under the original process.

Although lattices with more nodes provide slightly higher values, the difference in the PLB between 5 and 500 nodes is below 5%, which indicates that a good approximation of the optimal (risk-neutral) storage value can already be obtained at low computational cost.

### 4.6. Comparison of the Optimal Policy with Rolling Intrinsic

Our next analysis concerns the difference between the rolling intrinsic value and the approximate optimum. Exactly like Lai et al. (2010), we generated 12 instances of the model by combining the four variants of the price process (Spring, Summer, Fall, Winter) with the three pairs of storage capacity limits. We used a lattice with 500 nodes per stage generated using LQL and moment matching for each variant of the price process and ran ADDP for 200 iterations on each variant. The computational results are summarized in Table 3.

As we can see, the approximate optimum is substantially greater than the rolling intrinsic value across all instances, which demonstrates that the rolling intrinsic value severely underestimates the risk-neutral value of storage under the given price process. Moreover, our values
are 26-55% higher than the dual upper bounds reported in [Lai et al. (2010)], as their model does not account for forward trading.

In addition to the average total profits, Table 3 also reports the corresponding CVaR for both policies. We can see that under the rolling intrinsic policy the CVaR is much closer to the average value than under the optimal risk-neutral policy. In contrast to the optimal risk-neutral policy, the rolling intrinsic solution provides maximal risk aversion, which directly follows from Proposition 1.

We conclude that, unless the decision maker intends to minimize her maximum loss, the rolling intrinsic value substantially underrates the value of storage. However, a decision-maker is not necessarily perfectly risk-neutral either, so that the rolling intrinsic value and the optimal value provide two bounds for the optimal value of storage, irrespective of risk preferences.

4.7. The Effect of N-CVaR

Our final analysis concerns the effect of the N-CVaR on the value of storage. The trade-off between maximizing the expectation and minimizing the tail loss is defined by parameters $\lambda$ and $\alpha$, with a higher $\lambda$ resulting in more risk aversion. For the analysis, we used a lattice with 500 nodes per stage (on average) and ran ADDP for 1000 iterations for model instance A with Spring prices, $\lambda \in \{0.0, 0.05, 1.0\}$, and $\alpha = 0.05$.

Figure 6 summarizes the effect of the N-CVaR on profits. The figure shows four fan charts of the unconditional distributions of profits accumulated over time for different values of $\lambda$. Much like a heat map, a fan chart gives realizations with a higher probability of occurrence a darker color than realizations with a lower probability. The mean is indicated by a white dotted line with the terminal value annotated on the right-hand side of each graph.

Panel (a) shows the cumulative profits if we ignore risk and only maximize the expectation, whereas panels (b) and (c) show cumulative profits under the N-CVaR risk measure for $\lambda = 0.05$ and $\lambda = 1.0$, respectively. Comparing (a) and (b), we can see that the risk-neutral policy invests at stage one with a high negative immediate reward but also yields the higher mean terminal profit. In contrast, the N-CVaR policy with $\lambda = 0.05$ keeps the cumulative profits (mostly)
Optimal gas storage valuation under a high-dimensional price process

(a) Optimal expectation \( (\lambda = 0.0) \)
(b) Optimal combination of expectation and N-CVaR \( (\lambda = 0.05) \)
(c) Optimal N-CVaR \( (\lambda = 1.0) \)
(d) Rolling intrinsic

Figure 6  Comparison of accumulated profits for different risk preferences.

positive on average, but terminates with lower profits on average. Panel (c) shows the extreme case of a policy that only maximizes the N-CVaR \( (\lambda = 1.0) \). Compared to panel (b), the policy is almost loss-free with rare occurrences of negative profits. However, the policy is still not as risk-averse as the rolling intrinsic policy shown in panel (d), which only accumulates positive profits and never falls behind the intrinsic value.

The analysis shows that the proposed approach is capable of delivering risk-averse policies that provide a balance between extreme loss aversion and risk neutrality. Together with the rolling intrinsic policy, these policies offer decision makers a range of implementable trading policies that best match their risk preferences.

5. Conclusion

In this study, we have proposed a new approach for gas storage valuation. In contrast to prior research, the new approach solves the stochastic dynamic optimization problem underlying storage valuation under full consideration of the high-dimensional state space that is defined
through forward prices as well as forward contracts. By incorporating the N-CVaR into our model formulation, we can show that the widely used rolling intrinsic solution is a boundary case of our model under extreme loss aversion.

To solve the high-dimensional stochastic optimization problem, we developed a solution method that combines approximate dual dynamic programming with a lattice quantization algorithm that discretizes the high-dimensional price process in an optimal way. To ensure that price expectations on the lattice match price expectations of the process for a given state, we combine our lattice quantization algorithm with a moment-matching heuristic. We find that moment-matching is important to close the gap between a dual upper bound and a simulated lower bound on the continuous MGBM process.

In a numerical analysis, we reproduce the numerical results of Lai et al. (2010) by using the parameters of their multifactor price model along with the same contract parameters for our own case study. We compare our solution to their dual upper bound as well as the rolling intrinsic value and show that the expected value of storage is higher when our approach is used.

We show that the gap between the rolling intrinsic solution and our solution can be explained by different attitudes toward risk. While the rolling intrinsic solution can be seen as a lower bound of a loss-free value of storage, our solution can be viewed as a risk-neutral upper bound. Intermediate results can be obtained by using the N-CVaR as a risk measure.

However, the effect of using a discrete representation of a continuous stochastic process for control is still not well understood. Improving the way in which latices are constructed provides a number of starting points for future research.

References


Markets* **12**(1).

Carmona, R., M. Ludkovski. 2010. Valuation of energy storage: an optimal switching approach. *Quan-

Chen, Z., P. Forsyth. 2007. A semi-lagrangian approach for natural gas storage valuation and optimal


Pricing, and Hedging*. Wiley Finance, Wiley.


Mathematics, Springer Berlin Heidelberg.

September* 1–5.


Markets and Markets. 2014. Natural gas storage market by type (underground, above ground, and floating), and by geography - global trends & forecasts till 2019.


