GPU-Accelerated Dynamic Programming for Multistage Stochastic Energy Storage Arbitrage

Thomas Lee, *Student Member, IEEE*, Andy Sun, *Senior Member, IEEE.*

***Abstract*—We develop a GPU-accelerated dynamic programming (DP) method for valuing, operating, and bidding energy storage under multistage stochastic electricity prices. Motivated by computational limitations in existing models, we formulate DP backward induction entirely in tensor-based algebraic operations that map naturally onto massively parallel GPU hardware. Our method accommodates general, potentially non-concave payoff structures, by combining a discretized DP formulation with a convexification procedure that produces market-feasible, monotonic price-quantity bid curves. Numerical experiments using ISO-NE real-time prices demonstrate up to a 100× speedup by the proposed GPU-based DP method relative to CPU computation, and an 8,000x speedup compared to a commercial MILP solver, while retaining sub-0.3% optimality gaps compared to exact benchmarks.**

***Index Terms*—Energy storage; Dynamic programming; GPU acceleration; Optimization algorithms; Stochastic optimization**

I. Introduction

Energy storage plays a growing role in modern power systems. Energy arbitrage, which shifts energy between lower and higher-valued periods, is a critical revenue source for storage projects. Increased grid volatility, driven by load growth and renewables integration, raises computational challenges for accurate energy storage modeling.

Since dispatch decisions must be made before prices are known, deterministic perfect-foresight models, such as typical LP formulations, can significantly overestimate arbitrage value [1]. Accurately modeling non-simultaneous charging and discharging remains challenging: LP relaxations cannot fully eliminate complementarity violations since the convex hull may involve exponentially many cuts [2], and the issue may become more severe as negative prices increase in frequency due to renewables deployment and grid congestion.

Further, electricity markets rely on *price-quantity* bids instead of solely self-scheduled quantities. Prior works tackling the storage bid formation problem use approximate dynamic programming [3] or binary variables to indicate bid-clearing status [4], but these models only allow one bid segment per time period rather than more general bid curves. A recent analytic dynamic programming (DP) approach constructs energy arbitrage bid curves from value-function subgradients and reports significant speedups compared to stochastic dual dynamic programming (SDDP) [5], but it requires linear payoffs, prohibits discharging at negative prices, and still relies on state space discretization for numerical implementation (concave payoffs require discrete actions as well).

Thomas Lee is with the Institute for Data, Systems, and Society, MIT.
Andy Sun is with the Sloan School of Management, MIT.

979-8-3315-8138-1

With theoretical roots in DP, deep reinforcement learning (RL) offers fast inference due to GPU-accelerated neural networks and has been applied to stochastic energy arbitrage [6], but it requires substantial upfront training effort, and does not generate market-compatible bid curves.

In contrast to using GPU computation for neural networks (e.g. for RL or ML-based optimization surrogates), recent research efforts have applied GPU acceleration directly within novel optimization solvers, including for linear programming (LP) [7] and nonlinear programming (NLP) [8]. Inspired by these recent advances that utilize the optimization algorithm structure to leverage GPU hardware, in this work we develop a discretized DP method with a matrix-tensor structure that naturally enables efficient GPU calculations.

A. Contributions

This work makes the following contributions:

- We design a tensor-based backward induction algorithm to solve a dynamic programming formulation of the multistage stochastic energy arbitrage problem.
- We implement our algorithm to leverage GPU hardware, achieving up to 100x speedup versus CPU baselines.
- We propose a general convexification approach to derive storage bidding curves under general payoff functions.
- Using a test case with extended negative prices, we demonstrate the advantage of the discretized DP method over existing approximate approaches, and show an 8,000x speedup over a commercial MILP solver.
- We use our DP method to quantify the economic value of stochastic modeling and price-quantity bidding.

II. Models

A. Deterministic energy arbitrage

The deterministic, price-taker energy arbitrage problem is

$$\max_{\{p_t,s_t\}_{t=1}^T} \ \sum_{t=1}^T \lambda_t p_t \tag{1a}$$

$$\text{s.t.} \ -\overline{p} \le p_t \le \overline{p}, \quad \forall t \in [T], \tag{1b}$$

$$0 \le s_t \le \overline{s}, \quad \forall t \in [T], \tag{1c}$$

$$s_t = s_{t-1} + F(p_t), \quad \forall t \in [T], \tag{1d}$$

which, given an initial energy level s_0 along with power capacity $\overline{p}$ and energy capacity $\overline{s}$, decides a sequence of net power outputs $\{p_t\}$ and energy levels $\{s_t\}$, to maximize the total storage profits which are linear to the power prices $\{\lambda_t\}$. The *state transition* function in Eq. (1d) is

$$F(p_t) := -\begin{cases} \frac{1}{\eta} p_t, & \text{if } p_t \ge 0, \ \text{(discharge)} \\ \eta p_t, & \text{if } p_t < 0 \ \ \text{(charge)} \end{cases} \tag{2}$$

which describes the power-to-energy conversion with an efficiency factor $\eta \in (0, 1]$, enforcing charge-discharge complementarity for battery energy storage systems.

Prior works typically formulate Eq. (1)-(2) with separate charging p_t^c and discharging p_t^d. An exact MILP reformulation of Eq. (1)-(2) is to combine Eq. (1) with the added constraints

$$p_t = p_t^d - p_t^c, \quad \forall t \in [T] \tag{3a}$$
$$F(p_t) = p_t^c \eta - p_t^d / \eta, \quad \forall t \in [T] \tag{3b}$$
$$0 \le p_t^c \le \overline{p} z_t, \quad \forall t \in [T], \tag{3c}$$
$$0 \le p_t^d \le \overline{p}(1 - z_t), \quad \forall t \in [T] \tag{3d}$$
$$z_t \in \{0, 1\}, \quad \forall t \in [T]. \tag{3e}$$

This is standard in the literature [2]. Note $[T] \equiv \{1, ..., T\}$.

B. Multistage stochastic energy arbitrage

In the stochastic setting, we model power prices as *stagewise independent* random variables, following [5]. That is, the random prices $\{\lambda_t\}_t$ are mutually independent, each with a probability distribution. Given a feasible incoming state $s_{t-1} \in [0, \overline{s}]$, the set of feasible power actions is the continuous interval

$$\mathcal{P}(s_{t-1}) = \left[-\min\{\overline{p}, (\overline{s} - s_{t-1})/\eta\},\ \min\{\overline{p}, s_{t-1}\eta\} \right], \tag{4}$$

because when $p_t \ge 0$, we need $p_t \le \overline{p}$ from (1b) and $0 \le s_{t-1} - \frac{1}{\eta} p_t$ from (1d)-(2). Conversely, when $p_t < 0$, we need $-\overline{p} \le p_t$ and $s_{t-1} - \eta p_t \le \overline{s}$. Thus Eq. (4) exactly encapsulates the constraints (1b)-(2) for two adjacent periods $t-1$ and t.

Assuming stagewise independence, the dynamic programming problem can be written, adapting [5]'s formulation, using the *value function* defined recursively based on Bellman optimality for each $t \in [T]$:

$$Q_{t-1}(s_{t-1}, \xi_t) = \max_{p_t \in \mathcal{P}(s_{t-1})} \{\lambda_t p_t + V_t\left(s_{t-1} + F(p_t)\right)\}, \tag{5}$$

where the uncertainty here is just in prices ($\xi_t = \lambda_t$). Using the optimality condition for Eq. (5), we will be able to find the price-quantity bid curve in Section II-C.

The *expected* value function is defined recursively:

$$V_{t-1}(s_{t-1}) := \mathbb{E}_{\xi_t}\left[Q_{t-1}(s_{t-1}, \xi_t)\right], \quad \forall t \in \{0\} \cup [T], \tag{6}$$

with the final-period base case, describing the value of any residual state-of-charge, defined as $V_T(s_T) := 0,\ \forall s_T \in \mathbb{R}$. Altogether, Eq. (5)-(6) constitute the dynamic programming (DP) problem for multistage stochastic energy arbitrage. The objective expression of Eq. (5) embeds the state transition of (1d). The multistage problem (5) solves for the optimal policies of storage power dispatch p_t, which are contingent upon the incoming energy level s_{t-1} as well as the price realization λ_t.

C. Price-quantity function bidding

In order to participate in real-time (RT) electricity markets, resources including energy storage submit *price-quantity function bids*, consisting of *price-quantity pairs* that enter the grid operator's economic dispatch problem. We assume the bid curve for delivery during time t should be calculated prior to time t, following [5]. Define the composed function

$$U_t(p_t; s_{t-1}) := V_t(s_{t-1} + F(p_t)), \tag{7}$$

which maps action p_t to the value obtained at the next stage.

Consider the setting where $V_t(s_t)$ is concave and monotonic non-decreasing in its energy level argument s_t. The concave $V_t(\cdot)$ is composed with $F(p_t)$, which preserves the concavity of $U(p_t; s_{t-1})$ in its argument p_t (see p.32 in [9]). Then, given any s_{t-1}, the optimality condition for Eq. (5) is $\lambda_t \in -\partial U_t(p_t; s_{t-1})$, which depends on the superdifferential of the concave $U_t(p_t; s_{t-1})$ with respect to p_t. Then

$$b(p_t; s_{t-1}) := -\partial U_t(p_t; s_{t-1}), \tag{8}$$

is a set-valued function that maps a feasible power output $p_t \in \mathcal{P}(s_{t-1})$ to a corresponding set of marginal costs, within which p_t is optimal. The concavity of $U_t(\cdot; s_{t-1})$ means $b(p_t)$ is monotonic non-decreasing in p_t. The overall optimal bidding curve is formed by the graph

$$\left(\mathcal{P}(s_{t-1}),\ b(\mathcal{P}(s_{t-1}); s_{t-1})\right). \tag{9}$$

Critically, the bid curve written in Eq. (9) must be decided during interval $t-1$ and corresponds to physical delivery during interval t. From Eq. (8), this bid curve only depends on information embedded in $V_t(\cdot)$, which is an expectation over the λ_t and remains uncertain as of time $t-1$. Thus the bid curve construction is non-anticipative and is feasible to the real-life imperfect information structure.

In general, $U_t(\cdot, s_{t-1})$ could be non-concave. For example, if prices are sufficiently negative then $V_t(s_t)$ could actually decrease as s_t increases, i.e. when having more headroom is beneficial because charging is profitable. In this case, the function could be *convexified* first, to $\tilde{U}_t(p_t; s_{t-1})$, such that

$$\operatorname{hyp} \tilde{U}_t(p_t; s_{t-1}) = \operatorname{conv}(\operatorname{hyp} U_t(p_t; s_{t-1})), \tag{10}$$

where hyp denotes the hypograph of a function and conv denotes the convex hull. This step can be performed numerically, for example with the Graham scan [10]. Then, Eq. (8) can be modified to produce a *monotonized* bid curve using

$$\tilde{b}(p_t; s_{t-1}) := -\partial \tilde{U}_t(p_t; s_{t-1}), \tag{11}$$

which now meets the monotonicity required by market rules. The next section describes the numerical implementation of the overall dynamic programming algorithm.

III. Dynamic programming algorithm

A. Discretization

Assume that the state space is discretized to uniform multiples of a divisible scalar $\delta > 0$, such that $n^s = \overline{s}/\delta \in \mathbb{N}$. Explicitly write out the vector of these state levels as

$$\hat{s} := [0, \delta, 2\delta, ..., \overline{s}]^\top \in \mathbb{R}^S, \qquad S := n^s + 1. \tag{12}$$

Define the discretized power actions as the vector

$$\hat{p} := [-\overline{p}, ..., -2\delta/\eta, -\delta/\eta,\ 0,\ \delta\eta, 2\delta\eta, ..., \overline{p}]^\top, \tag{13a}$$
$$P := \dim(\hat{p}) = n^c + n^d + 1, \tag{13b}$$
$$n^c := \lceil \overline{p}\eta/\delta \rceil, \quad n^d = \lceil \overline{p}/(\delta\eta) \rceil, \tag{13c}$$

where n^c, n^d are the number of charge and discharge actions, plus the zero-power action. The two $\{-\overline{p}, \overline{p}\}$ endpoints are deliberately included based on the empirical observation that optimal storage dispatch very often occurs at the $\{-\overline{p}_t, 0, \overline{p}_t\}$ levels (which is reinforced by the theoretical existence of extreme point LP solutions).

Finally, assume that each random variable λ_t is modeled with a discrete probability mass function with a finite support (e.g. by randomly sampling from a continuous distribution of λ_t), accessible as a vector $\hat{\lambda}_t$ of price levels along with a vector $\hat{\pi}_t$ of their discrete probabilities,

$$\hat{\lambda}_t,\ \hat{\pi}_t \in \mathbb{R}^R, \quad \forall t \in [T], \tag{14a}$$

$$\hat{\pi}_{tr} = \mathbb{P}(\lambda_t = \hat{\lambda}_{tr}), \quad \forall t \in [T], r \in [R] \tag{14b}$$

where R is the cardinality of the finite discrete support of λ_t, i.e. the number of distinct price levels used to model its distribution, and such that $\sum_{r=1}^{R} \hat{\pi}_{tr} = 1$ for all $t \in [T]$.

B. Backward induction as matrix algebra

We now impose the discretization scheme of Section III-A onto the DP problem of Eq. (4)-(6). The resulting discretized problem is thus a restriction of the original DP, since all actions and states are still feasible. Algorithm 1 solves the discretized DP using purely matrix-algebraic steps.

Algorithm 1 Tensor-based DP backward induction

Input: $\hat{s} \in \mathbb{R}^S,\ \hat{p} \in \mathbb{R}^P,\ \{\hat{\lambda}_t, \hat{\pi}_t\}_{t=1}^T$, where $\hat{\lambda}_t, \hat{\pi}_t \in \mathbb{R}^R$.
Output: Expected value vectors $\{\hat{V}_t\}_{t=0}^T$.

1: $\hat{V}_T \leftarrow \mathbf{0}_S$. (*base case*)
2: $\sigma \leftarrow \hat{s}(\mathbf{1}_P)^\top + (\mathbf{1}_S)F(\hat{p})^\top$. (*next-state levels*)
3: $z \leftarrow \frac{1}{\delta}\min\{0, \max\{\sigma, \overline{s}\}\} + 1$. (*next-state proxies*)
4: $z^- \leftarrow \lfloor z \rfloor, \quad z^+ \leftarrow \lceil z \rceil$. (*next-state indices*)
5: $b \leftarrow (z - z^-)/(z^+ - z^-)$. (*interpolation factors*)
6: **for** $t = T, ..., 0$ **do**
7: $\quad \hat{V}_t^{\text{next}} \leftarrow (1-b) \odot \hat{V}_t[z^-] + b \odot \hat{V}_t[z^+]$.
8: $\quad \hat{V}_t^{\text{next}}[\mathbb{1}(\sigma < 0) + \mathbb{1}(\sigma > S)] \leftarrow -\infty$. (*infeasibility*)
9: $\quad \hat{Q}_{t-1}^{\text{all}} \leftarrow \mathbf{1}_S \otimes \hat{p}\hat{\lambda}_t^\top + \hat{V}_t^{\text{next}} \otimes \mathbf{1}_R^\top$.
10: $\quad \hat{Q}_{t-1}[i, r] \leftarrow \max_{j \in [P]} \hat{Q}_{t-1}^{\text{all}}[i, j, r], \quad \forall i \in [S], r \in [R]$.
11: $\quad \hat{V}_{t-1} \leftarrow \hat{Q}_{t-1}\hat{\pi}_t$. (*expectation*)
12: **end for**

A key part of the algorithm design is efficiently evaluating the next-stage value functions. This is achieved by Lines 2-5, which pre-compute relevant vector indices and interpolation factors as constants across all t. Line 2 performs the "outer sum" on the state and power vectors to find $\sigma \in \mathbb{R}^{S \times P}$, consisting of all pairwise sums from $\hat{s}$ and $\hat{p}$. Line 3 projects σ, elementwise, onto $[0, \overline{s}]$, then transforms it to $z \in \mathbb{R}^{S \times P}$ with elements in the continuous range $[1, S]$, forming continuous "proxies" of the next-state indices. Line 4 converts z into matrices of integer indices $z^-, z^+ \in [S]^{S \times P}$, where floor and ceiling operations are applied elementwise. Then Line 5 calculates interpolation factors (with slight abuse of notation, where division applies elementwise). Note that

$$F(\hat{p}) = [\overline{p}\eta, ..., 2\delta, \delta, 0, -\delta, -2\delta, ..., \overline{p}/\eta]^\top,$$

which guarantees that all of σ's columns (except the first and last) consist of combining integer multiples of δ. This means columns 2 to $P-1$ in matrix z all contain integer elements in set $[S]$, associated with 0 interpolation factor within b. This choice of "recombining" states and power actions helps to reduce any numerical interpolation errors, since we would only need to interpolate for the extreme power endpoints $\{-\overline{p}, \overline{p}\}$.

Next, Algorithm 1 performs backward induction. At each stage t, Line 7 uses the index matrices $\hat{z}^-, \hat{z}^+$ to extract the appropriate elements of $\hat{V}_t$ and applies linear interpolation for the first and last columns (corresponding to $-p_t$ and p_t). In Line 7, brackets denote indexing, and $\odot$ denotes the elementwise Hadamard product. This produces a tentative next-state value matrix $\hat{V}_t^{\text{next}} \in \mathbb{R}^{S \times P}$, which is a discretization of $V_t(s_{t-1} + F(p_t))$. Line 8 then assigns $-\infty$ to the infeasible (state, power) combinations, in $\hat{V}_t^{\text{next}}$, to ensure they are not chosen during maximization. The outer product matrix $\hat{p}\hat{\lambda}_t^\top \in \mathbb{R}^{P \times R}$ describes the stage-specific payoffs, and the tensor $\hat{Q}_{t-1}^{\text{all}} \in \mathbb{R}^{S \times P \times R}$ is created in Line 9 by broadcasting the matrices with tensor products $\otimes$ onto common dimensions.

At this point, $\hat{Q}_{t-1}^{\text{all}}$ corresponds to the inner expression within Eq. (5)'s maximization objective. Line 10 then maximizes over the actions, resulting in the matrix $\hat{Q}_{t-1} \in \mathbb{R}^{S \times R}$. Next, Line 11 computes the expectation over the discrete random variable λ_t; this is done as a matrix-vector product, producing $\hat{V}_{t-1} \in \mathbb{R}^S$. Then the algorithm inductively proceeds backward, as illustrated in Fig. 1. Finally, the bidding function in Eq. (8) or (11) can be evaluated using numerical finite differences based on the $\hat{V}_t$ vectors.

Crucially, all operations in Algorithm 1 can be written as fully vectorized code, including Line 10's maximization along an axis. Only the time stages t require a sequential for-loop. Thus, by design, Algorithm 1 is a natural setting for highly parallelized single instruction, multiple data (SIMD) algebraic operations, for which GPU computation excels.

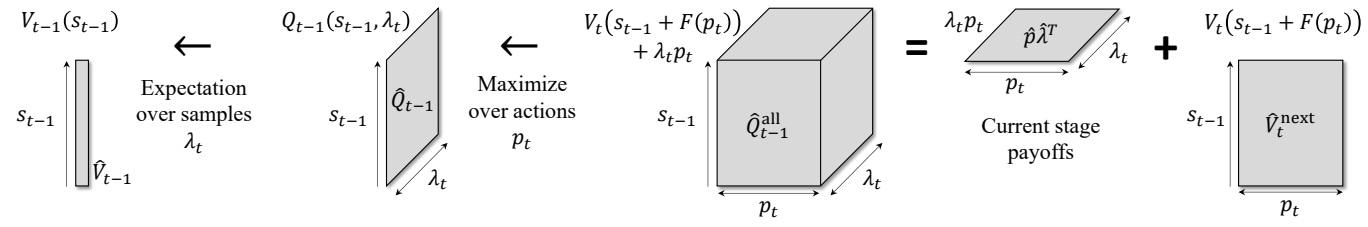

Fig. 1. Diagram illustrating Algorithm 1. Vector / matrix / tensor objects from Alg. 1 are illustrated as gray shapes. Labels at the top correspond to function names from Section II, and each axis is labeled with its corresponding variable name. The "+" operation here represents the tensor sum defined in Line 9.

IV. Data

Historical day-ahead (DA) and real-time (RT) prices are downloaded from ISO-NE ("Kendall" price node) for the years 2023-2024. For simplicity we consider hourly averages of RT prices. While our DP method allows any probabilistic forecast of $\{\lambda_t\}_t$, in this work we employ a simple data-driven quantile forecast approach, which uses Y2023 for "training" and Y2024 for simulations. For each RT forecast hour in Y2024, we add the corresponding DA price (which is already available) onto 200 samples of empirical quantiles of (RT−DA) spreads from Y2023, conditioned on the same (month, hour), in order to obtain probabilistic forecasts of the RT price.

V. Numerical results

We assume 85% roundtrip efficiency ($\eta = \sqrt{0.85}$), $\overline{p}$ = 1MW power capacity, and $s_0 = 0$MWh initial state, unless otherwise noted. ***Software***: LP and MILP models are written in `CVXPY` and solved with `Gurobi`; solve times are reported for the time spent inside the solver (which excludes `CVXPY` compilation time). The discretized DP method is implemented in Python using `NumPy` for CPU and `CuPy` [11] for GPU. `CuPy` uses identical software syntax as `NumPy`, enabling a controlled study of the impact from CPU vs. GPU hardware. ***Hardware resources***: Computations are executed on one node of the MIT Engaging high-performance computing cluster. CPU computations are on an Intel Xeon Platinum 8562Y+ CPU processor (32 physical cores, 2.8GHz), GPU computations use an NVIDIA L40S GPU (91.6 TFLOPS for FP32).

A. Accuracy of discretized DP (CPU)

Imposing a discretization scheme may raise concerns about how accuracy depends on the chosen discretization granularity. So we first benchmark our implemented DP method (still on CPU) relative to an exact LP formulation in the deterministic perfect foresight setting, solving with the full 8784-hour RT prices from Y2024 as $\{\lambda_t\}$ with a single stochastic sample, $R = 1$. The problem is solved for a battery with a 4-hour duration, and the DP method is tested at increasingly granular levels of δ. We also obtain bids as in Eq. (9), and evaluate them using a merit-order market clearing simulation based on the realized prices. That is, assuming the storage is a price-taker, for each t intersect the realized RT price with the bid curve to obtain the cleared quantity award p_t.

TABLE I
Deterministic prices (8784 hours): Accuracy of discretized DP

Method	δ	Actions (P)	Time (sec)	Quantity opt. Obj. (k$)	Quantity opt. Gap	Bidding Obj. (k$)	Bidding Gap
LP	–	–	0.25	57.65	–	–	–
DP	0.10	22	0.15	57.54	−0.19%	57.55	−0.17%
	0.05	42	0.23	57.57	−0.13%	57.59	−0.09%
	0.02	103	0.64	57.63	−0.04%	57.63	−0.03%
	0.01	203	2.14	57.63	−0.02%	57.64	−0.02%

Table I reports δ resolutions and action space sizes P, with solve times in seconds. Objectives from quantity optimization ("Quantity opt.") and price-quantity bidding ("Bidding") are reported in thousand USD per MW-year, along with relative optimality gaps compared to the exact LP. The DP objectives remain below the LP upper bound, which numerically verifies that Section III-A's discretization framework indeed results in a restriction of the original continuous problem. A 0.2% accuracy is already achieved with a fairly coarse discretization (22 power actions), suggesting that such a resolution is sufficient for practical valuation and bidding purposes. Accuracy further improves at higher resolutions, to within 0.02%, suggesting that accuracy scales roughly linearly with δ.

B. Case study of negative prices (CPU)

A limitation of the prior analytic DP approach in [5] is the ex-ante prevention of discharging during negative price periods, as well as the requirement of linear (or at least concave) payoff functions. To demonstrate the generalized discretized DP approach's economic value, we create an artificial price time series by level-shifting the prices to create a deterministic price series consisting of all negative prices:

$$\lambda_t^{neg} = \lambda_t - \left(\max_t \lambda_t\right) \leq 0, \quad \forall t \in [T].$$

While artificially constructed, a setting of frequent negative pricing is generally plausible, e.g. at a highly congested node. Table II is a case study to optimize the first 72 hours of the adjusted Y2024 prices series $\{\lambda_t^{neg}\}$, with initial $s_0 = \overline{s}$. We compare our discretized DP method (along with its produced bid curves) with a set of different energy arbitrage formulations. ***MILP exact***: uses Eq. (1) and (3). ***LP relaxation***: replaces Eq. (3e) with a continuous $z_t \in [0, 1]$. ***LP restriction***: prohibits discharge during negative prices, i.e.

$$p_t^d = 0, \quad \forall t \in [T] : \ \lambda_t \leq 0,$$

which replicates the same restriction in the analytic DP method of [5]. Ref. [5] explains how this restriction is sufficient to eliminate simultaneous charge-discharge. Thus, this is a valid restriction of the MILP exact feasible region. **DP**: our DP method solved with CPU, and then using the convexification approach to produce monotonic bids as in Eq. (11).

TABLE II
Negative prices (72 hours): Accuracy of discretized DP, starting at $\overline{s}$

Method	Time (sec)	Obj. (k$)	Gap	Freq. of $p_t^c p_t^d > 0$
MILP exact	25.068	12.811	–	0%
LP relaxation	0.071	23.045	79.89%	86%
LP restriction	0.142	0.000	−100.00%	0%
DP	0.003	12.777	−0.27%	0%
	(Bidding)	12.799	−0.10%	0%

Table II demonstrates that the LP relaxation can significantly overstate storage profitability, as a consequence of very frequent complementarity violations ("Frequency of $p_t^c p_t^d > 0$"). Conversely, the LP restriction approach earns exactly 0 profit when starting with the full $s_0 = \overline{s}$, because the storage device is always prohibited to discharge in this case. In contrast, our discretized DP approach guarantees feasible dispatch, since simultaneous charge-discharge is impossible with only one net p_t variable per t. Furthermore, our DP method achieves profits that essentially match the exact MILP solution within $0.1{\sim}0.3\%$ accuracy, while being ∼8,000x faster than MILP.

C. GPU acceleration with multistage stochastic prices

Table III tests the DP method in the multistage stochastic price setting, with 200 price samples per time period. A range of different storage durations (energy-to-power ratio, measured in hours) is tested, at two different δ resolutions. The table lists the discretization granularity in terms of number of S states and P actions. Times to solve the full-year problem, using CPU (`NumPy`) versus GPU (`CuPy`) computation, are reported in seconds. As problem size scales up, either in terms of granularity or storage duration, the GPU times remain the same or modestly increase, which significantly outperforms the CPU's solve times by up to about 100x.

TABLE III
CPU vs GPU solve times (8784 hours, 200 price samples per hour)

Duration (hours)	**States** (S)	**Actions** (P)	**CPU time** (sec)	**GPU time** (sec)	***GPU speedup***
4	41	22	3.98	2.77	1.4x
20	201	22	18.29	2.81	6.5x
100	1,001	22	98.12	2.94	33.4x
4	401	203	397.80	4.88	81.5x
20	2,001	203	1,996.02	19.17	104.1x
100	10,001	203	9,972.64	86.67	115.1x

D. Quantifying the value of stochastic DP bid curves (GPU)

We study the interaction between price uncertainty and storage duration. We compare a range of strategies representing decreasing levels of information access. ***Perfect foresight***: LP on realized RT prices (giving an upper bound on profits). ***Stochastic DP bid curves***: DP using Section IV's stochastic prices, producing price-quantity bid curves as in Eq. (9). ***Stochastic DP self-scheduled***: the same stochastic DP model, but we intersect each bid curve for time t with the realized 1-hour-lagged RT price (as an observable price forecast) in order to obtain a single self-scheduled quantity p_t. ***Myopic***: deterministic LP on DA prices to obtain self-scheduled quantities, which are multiplied by RT prices.

As seen in Fig. 2, across a range of storage durations, using stochastic modeling significantly outperforms the myopic dispatch (by up to 24%), which represents the value of incorporating uncertainty. Furthermore, utilizing bidding curves rather than single quantities leads to additional gains in profitability (by up to 32%). An intuition for this additional value is that bid curves can embody the price-contingent optimal policy function. As duration increases, the realizable arbitrage profits also increase as a percentage capture of the perfect foresight upper bound. The evolution of these profit capture ratios has crucial implications for energy storage investment decisions.

Solving all DP models for Fig. 2 takes 35 seconds with GPU acceleration, compared to about 866 seconds with CPU (14 minutes, or 25x longer). This speedup highlights the power of the proposed method to utilize GPU acceleration in an efficient, on-the-fly manner without having to "retrain", unlocking wider studies across practically relevant scenarios and parameters.

VI. Conclusion

This paper applies GPU acceleration to dynamic programming in order to speed up calculations for energy arbitrage with multistage price uncertainty. This is made possible by our proposed tensor-based algorithmic formulation of DP backward induction. Up to about 100x speedup is found for GPU vs. CPU computation.

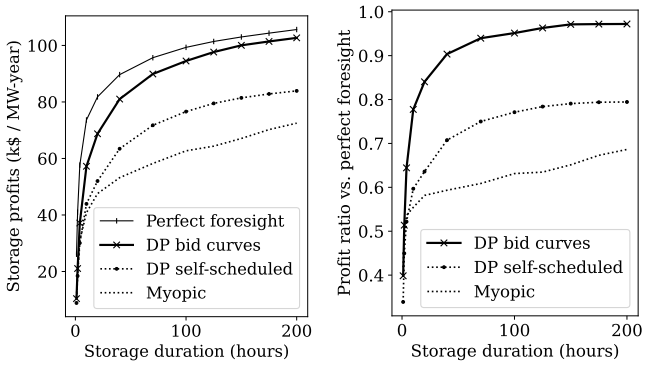

Fig. 2. Impact of duration and dispatch strategy on energy arbitrage profits. "*DP bid curves*" achieve up to 32% higher profits than "*DP self-scheduled*" (this difference represents the value of using price-quantity bid curves), which in turn achieves up to 24% higher profits than "*Myopic*".

A suite of experiments validate the numerical accuracy of our proposed DP method. Compared to an existing analytic DP approach's convex restriction, our discretized method significantly improves profitability under scenarios with frequent negative prices, while being 8,000x faster than an exact MILP solver. We also demonstrate the ability of the DP method to generate price-quantity bid curves that meaningfully outperform quantity-only dispatch.

The GPU-accelerated DP method can enable more effective investigation over a wide range of energy storage parameters, as well as different price forecasting methods, which can be explored in future work. This approach enables fast and accurate storage valuations, providing a practical tool for large-scale scenario analysis and storage investment studies.

References

[1] R. Sioshansi *et al.*, "Energy-storage modeling: State-of-the-art and future research directions," *IEEE Trans. Power Syst.*, vol. 37, no. 2, 2021.
[2] D. Pozo, "Convex hull formulations for linear modeling of energy storage systems," *IEEE Trans. on Power Syst.*, vol. 38, no. 6, 2023.
[3] D. R. Jiang and W. B. Powell, "Optimal hour-ahead bidding in the real-time electricity market with battery storage using approximate dynamic programming," *INFORMS J. Comput.*, vol. 27, no. 3, pp. 525–543, 2015.
[4] D. Krishnamurthy *et al.*, "Energy storage arbitrage under day-ahead and real-time price uncertainty," *IEEE Trans. on Power Syst.*, 2017.
[5] B. Xu, M. Korpås, and A. Botterud, "Operational valuation of energy storage under multi-stage price uncertainties," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 55–60.
[6] J. Cao *et al.*, "Deep reinforcement learning-based energy storage arbitrage with accurate lithium-ion battery degradation model," *IEEE Trans. Smart Grid*, vol. 11, no. 5, 2020.
[7] H. Lu and J. Yang, "cuPDLP.jl: A GPU implementation of restarted primal-dual hybrid gradient for linear programming in Julia," *Oper. Res.*, 2025.
[8] S. Shin, M. Anitescu, and F. Pacaud, "Accelerating optimal power flow with GPUs: SIMD abstraction of nonlinear programs and condensed-space interior-point methods," *Electr. Power Syst. Res.*, vol. 236, 2024.
[9] T. Rockafellar, R. *Convex Analysis*, ser. Princeton Mathematical Series. Princeton, NJ: Princeton University Press, 1970.
[10] R. L. Graham, "An efficient algorithm for determining the convex hull of a finite planar set," *Information Processing Letters*, vol. 1, no. 4, 1972.
[11] R. Okuta *et al.*, "CuPy: A NumPy-compatible array library accelerated by CUDA," in *Proc. Mach. Learn. Syst. (MLSys)*, 2017.