A Illustration of RCL

We illustrate the online optimization process of RCL in Fig. 1.

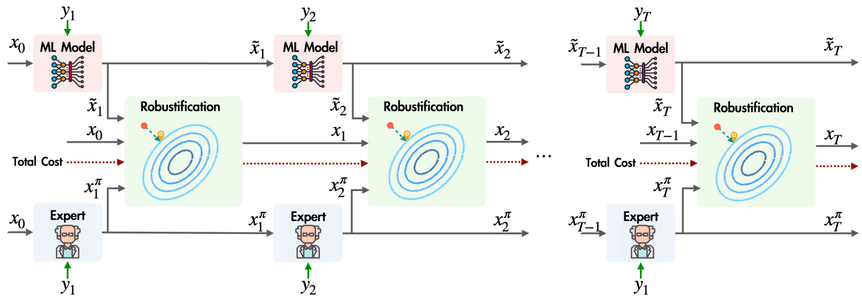


Figure 1: Robustness-constrained online optimization using RCL. The expert algorithm and ML model run independently. At each time $t = 1, \cdots, T$, RCL projects the ML prediction $\tilde{x}_t$ into a robustified action set.

B Case Study: Battery Management for EV Charging Stations

We now explore the performance of RCL using a case study focused on battery management in electric vehicle (EV) charging stations [48]. We first formulate the problem as an instance of SOCO, and then present the baseline algorithms. Finally, we discuss the performance of RCL. Our results highlight the advantage of RCL in terms of robustness guarantees compared to pure ML models, as well as the benefit of training a robustification-aware ML model in terms of the average cost.

B.1 Problem Formulation

Batteries are routinely used in EV charging stations to handle the rapidly fluctuating charging demands and protect the connected grid. Thus, properly managing battery charging/discharging decisions is crucial for reliability, lifespan, and safety of batteries and grids.

We consider the management of N batteries. At each time step t, suppose that $x_t \in \mathbb{R}_+^N$ represents the State of Charge (SoC) and $u_t \in \mathbb{R}^N$ represents the battery charging/discharging schedule, depending on the sign of u_t (i.e., positive means charging, and vice versa). The canonical form of the battery dynamics can be written as $x_{t+1} = Ax_t + Bu_t - w_t$, where A is a $N \times N$ matrix which models the self-degradation of the N-battery system, B is a $N \times N$ matrix which represents the charging efficiency of each battery unit, w_t is a $N \times 1$ vector which denotes the current demand in terms of the charging rate (kW) of all the EVs connected to the charging stations. Assuming that the initial SoC as x_0, the goal is to control the batteries to minimize the difference between the current SoC of all batteries and a nominal value $\bar{x}$, plus a charging/discharging cost to account for battery usage [49, 50], which can be expressed mathematically as $\min_{u_1, u_2, \cdots, u_{T+1}} \sum_{t=1}^{T+1} \|x_t - \bar{x}\|^2 + b\|u_t\|^2$.

This problem falls into SOCO based on the reduction framework described in [49]. Specifically, at time step $t+1$, we can expand x_{t+1} based on the battery dynamics as $x_{t+1} = A^t x_1 + \sum_{j=1}^{t} A^{t-j} B u_j - \sum_{j=1}^{t} A^{t-j} w_j$, We define the context parameter as $y_t = \bar{x} - A^t x_1 + \sum_{i=1}^{t} A^{t-i} w_i$ and the action as $a_t = \sum_{i=1}^{t} A^{t-i} B u_i$. Then, assuming an identity matrix B (ignoring charging loss), the optimization problem becomes $\min_{a_1, \cdots, a_T} \|x_1 - \bar{x}\|^2 + b\|u_T\|^2 + \sum_{t=1}^{T} \|a_t - y_t\|^2 + b\|a_t - Aa_{t-1}\|^2$. Given an initial value of x_1, this problem can be further simplified and reformulated as

$$\min_{a_1, a_2, \cdots, a_T} \sum_{t=1}^{T} \frac{1}{b} \|a_t - y_t\|^2 + \|a_t - Aa_{t-1}\|^2, \tag{7}$$

which is in a standard SOCO form by considering y_t as the context and a_t as the action at time t.

To validate the effectiveness of RCL, we use a public dataset [51] provided by ElaadNL, a Dutch EV charging infrastructure company. We collect a dataset containing transaction records from ElaadNL charging stations in the Netherlands from January to June of 2019. Each transaction record contains the energy demand, transaction start time and charging time. As the data does not specify the details of battery units, we consider the battery units as a single combine battery by summing up the energy demand within each hour to obtain the hourly energy demand.

We use the January to February data as the training dataset, March to April data as the validation dataset for tuning the hyperparameters such as learning rate, and May to June as the testing dataset. We consider each problem instance as one day ($T = 24$ hours, plus an initial action). Thus, a sliding window of 25 is applied, moving one hour ahead each time, on the raw data to generate 1416 problem instances, where the first demand of each instance is used as the initial action of all the algorithms. We set $b = 10$ and $A = I$ for the cost function in Eqn. (7).

All the algorithms use the same ML architecture, when applicable, with the same initialized weights in our experiments for fair comparison. To be consistent with the literature [52, 53], all the ML models are trained offline. Specifically, we use a recurrent neural network (RNN) model that contains 2 hidden layers, each with 8 neurons, and implement the model using PyTorch. We train the RNN for 140 epochs with a batch size of 50. When the RNN model is trained as a standalone optimizer in a robustification-oblivious manner, the training process takes around 1 minute on a 2020 MacBook Air with 8GB memory and a M1 chipset. When RNN is trained in a robustification-aware manner, it takes around 2 minutes. The testing process is almost instant and takes less than 1 second.

B.2 Baseline Algorithms

By default, `RCL` uses a robustification-aware ML model due to the advantage of average cost performance compared to a robustification-oblivious model. We compare `RCL` with several representative baseline algorithms as summarized below.

• Offline Optimal Oracle (**OPT**): This is the optimal offline algorithm that has all the contextual information and optimally solves the problem.

• Regularized Online Balanced Descent (**ROBD**): ROBD is the state-of-the-art order-optimal online algorithm with the best-known competitive ratio for our SOCO setting [54, 49]. The parameters of ROBD are all optimally set according to [49]. By default, `RCL` uses ROBD as its expert for robustness.

• Hitting Cost Minimizer (**HitMin**): HitMin is a special instance of ROBD by setting the parameters such that it greedily minimizing the hitting cost at each time. This minimizer can be empirically effective and hence also used in ROBD as a regularizer.

• Machine Learning Only (**ML**): ML is trained as a standalone optimizer in a robustification-oblivious manner. It does not use robustification during online optimization.

• Expert-Calibrated Learning (**EC-L2O**): It is an ML-augmented algorithm that applies to our SOCO setting by using an ML model to regularize online actions without robustness guarantees [55]. We set its parameters based on the validation dataset to have the optimal average performance with an empirical competitive ratio less than $(1 + \lambda)CR^{\pi}$.

• `RCL` with a robustification-oblivious ML model (`RCL-O`): To differentiate the two forms of `RCL`, we use `RCL` to refer to `RCL` with a robustification-aware ML model and `RCL-O` for the robustification-oblivious ML model, where "`-O`" represents robustification-obliviousness.

To highlight our key contribution to the SOCO literature, the baseline algorithms we choose are representative of the state-of-the-art expert algorithms, effective heuristics, and ML-augmented algorithms for the SOCO setting we consider. While there are a few other ML-augmented algorithms for SOCO [56, 57, 58], they do not apply to our problem as they consider unsquared switching costs in a metric space and exploit the natural triangular inequality. Adapting them to the squared switching costs is non-trivial.

B.3 Results

We now present the results of our case study and begin with the case in which the hitting cost function (parameterized by y_t) is immediately known without feedback delay. The results for the case with feedback delay are presented in Section B.4. Throughout the discussion, the reported values are normalized with respect to those of the respective OPT. The average cost (**AVG**) and competitive ratio (**CR**) are all empirical results reported on the testing dataset.

	RCL				RCL-O				**ML**	**EC-L2O**	**ROBD**	**HitMin**
	λ=0.6	λ=1	λ=3	λ=5	λ=0.6	λ=1	λ=3	λ=5				
AVG	1.4704	1.1144	1.0531	**1.0441**	1.4780	1.2432	1.0855	1.0738	1.0668	1.1727	1.6048	1.2003
CR	1.7672	**1.2905**	1.4405	1.3014	2.2103	2.4209	2.4200	3.0322	3.2566	2.0614	1.7291	2.0865

Table 1: Competitive ratio and average cost comparison of different algorithms.

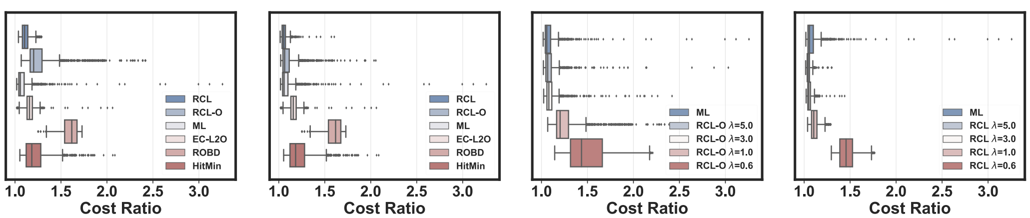
By Theorem 4.1, there is a trade-off (governed by $\lambda > 0$) between exploiting ML predictions for good average performance and following the expert for robustness. Here, we begin with the default setting of $\lambda = 1$ and investigate the impact of different choices of λ on both RCL and RCL-O in Section B.3.3.

B.3.1 The performance of RCL

As shown in Table 1, with $\lambda = 1$, both RCL and RCL-O have a good average cost, but RCL has a lower average cost than RCL-O and is outperformed only by ML in terms of the average cost. RCL and RCL-O have the same competitive ratio (i.e., $(1 + \lambda)$ times the competitive ratio of ROBD). Empirically, RCL has the lowest competitive ratio than all the other algorithms, demonstrating the practical power of RCL for robustifying, potentially untrusted, ML predictions. In this experiment, RCL outperforms ROBD in terms of the empirical competitive ratio because it exploits the good ML predictions for those problem instances that are adversarial to ROBD. This result complements Theorem 4.1, where we show theoretically that RCL can outperform ROBD in terms of the cost by properly setting λ.

By comparison, ML performs well on average by exploiting the historical data, but has the highest competitive ratio due to its expected lack of robustness. The two alternative baselines, EC-L2O and HitMin, are empirically good on average and also in the worst case, but they do not have guaranteed robustness. On the other hand, ROBD is very robust, but its average cost is also the worst among all the algorithms under consideration.

We further show in Fig. 2(a) the box plots for cost ratios with $\lambda = 1$, providing a detailed view of the algorithms' performance. The key message is that RCL obtains the best of both worlds — a good average cost and a good competitive ratio (empirically even better than the expert ROBD).

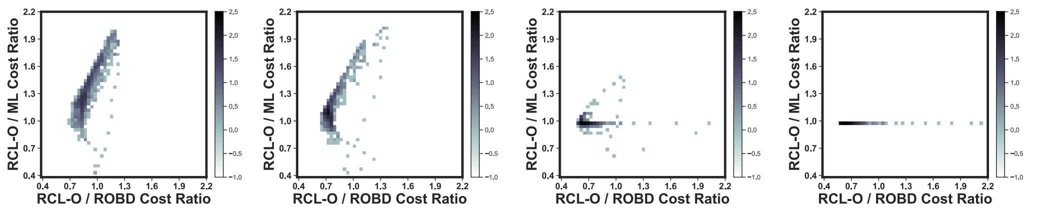


(a) ROBD as the expert (b) HitMin as the expert (c) RCL-O w/ different λ (d) RCL w/ different λ

Figure 2: Cost ratio distributions ($\lambda = 1$ by default).

B.3.2 Utilizing HitMin as the expert

RCL is flexible and can work with any expert online algorithm, even an expert that does not have good or bounded competitive ratios. Thus, it is interesting to see how RCL performs given an alternative expert. For example, in Table 1, HitMin empirically outperforms ROBD in terms of the average, although it is not as robust as ROBD. Thus, using $\lambda = 1$, we leverage HitMin as the expert for RCL and RCL-O, and show the cost ratio distributions in Fig. 2(b). Comparing Fig. 2(b) with Fig. 2(a), we see that RCL and RCL-O both have many low cost ratios by using HitMin as the expert, but the worst case for RCL is not as good as when using ROBD as the expert. For example, the average cost and competitive ratio are 1.0515 and 1.6035, respectively, for RCL. This result is not surprising, as the new expert HitMin has a better average performance but worse competitive ratio than the default expert ROBD.



(a) $\lambda = 0.6$ (b) $\lambda = 1.0$ (c) $\lambda = 5.0$ (d) $\lambda = \infty$ (i.e., ML)

Figure 3: Histogram of bi-competitive cost ratios of `RCL-O` (against ROBD and ML) under different λ. For better visualization, the color map represents logarithmic values of the cost ratio histogram with a base of 10.

B.3.3 Impact of λ

Theorem 4.1 shows the point that we need to set a large enough λ in order to provide enough flexibility for `RCL` to exploit good ML predictions. With a small $\lambda > 0$, despite the stronger competitiveness against the expert, it is possible that `RCL` may even empirically perform worse than both the ML model and the expert. Thus, we now investigate the impact of λ.

We see from Table 1 that the empirical average cost and competitive ratio of `RCL` are both worse with $\lambda = 0.6$ than with the default $\lambda = 1$. More interestingly, by setting $\lambda = 5$, the average cost of `RCL` is even lower than that of ML. This is because ML in our experiment performs fairly well on average. Thus, by setting a large $\lambda = 5$, `RCL` is able to exploit the benefits of good ML predictions for many typical cases, while using the expert ROBD as a safeguard to handle a few bad problem instances for which ML cannot perform well. Also, the empirical competitive ratio of `RCL` is better with $\lambda = 5$ than with $\lambda = 3$, supporting Theorem 4.1 that a larger λ may not necessarily increase the competitive ratio as `RCL` can exploit good ML predictions. In addition, given each λ, `RCL` outperforms `RCL-O`, which highlights the importance of training the ML model in a robustification-aware manner to avoid the mismatch between training and testing objectives.

We also show in Fig. 2(c) an Fig. 2(d) the cost ratio distributions for `RCL-O` and `RCL`, respectively, under different λ. The results reaffirm our main Theorem 4.1 as well as the importance of training the ML model in a robustification-aware manner.

Next, we show the bi-competitive cost ratios of `RCL-O` against both the expert ROBD and the ML predictions. We focus on `RCL-O` as its ML model is trained as a standalone optimizer, whereas `RCL` uses a robustification-aware ML model that is not specifically trained to produce good pre-robustification predictions. According to Theorem 4.1, `RCL-O` obtains a potentially better competitiveness against ML but a worse competitive against the expert ROBD when λ increases, and vice versa. To further validate the theoretical analysis, we test `RCL-O` with different λ and obtain the 2D histogram of its bi-competitive cost ratios against ROBD and ML, respectively. The results are shown in Fig. 3. In agreement with our analysis, the cost ratio of `RCL-O` against ROBD never exceeds $(1 + \lambda)$ for any $\lambda > 0$. Also, with a small $\lambda = 0.6$, the cost ratio of `RCL-O` against ROBD concentrates around 1, while it does not exploit the benefits of ML predictions very well. On the other hand, with a large $\lambda = 5$, the cost ratio of `RCL-O` against ROBD can be quite high, although it follows (good) ML predictions more closely for better average performance. Most importantly, by increasing $\lambda > 0$, we can see the general trend that `RCL-O` follows the ML predictions more closely while still being able to guarantee competitiveness against ROBD. Again, this confirms the key point of our main insights in Theorem 4.1.

B.3.4 Larger distributional shifts

In our dataset, ML performs very well on average as the testing distribution matches well with its training distribution. To consider more challenging cases as a stress test, we manually increase the testing distributional shifts by adding random noise following $\mathcal{N}(0, \sigma)$ to a certain faction p_c of the testing samples. Note that, as we intentionally stress test `RCL` and `RCL-O` under a larger distributional shift, their ML models remain unchanged as in the default setting and are not re-trained by adding noisy data to the training dataset.

		p_c=0.05			p_c=0.1			p_c=0.2		
		σ =0.06	σ =0.08	σ =0.1	σ =0.06	σ =0.08	σ =0.1	σ =0.06	σ =0.08	σ =0.1
AVG	`RCL`	1.1331	1.1444	1.1556	1.1487	1.1693	1.1904	1.1827	1.2254	1.2697
	`RCL-0`	1.2425	1.2436	1.2462	1.2416	1.2434	1.2478	1.2370	1.2394	1.2469
	ML	1.0722	1.0778	1.0855	1.0770	1.0874	1.1018	1.0858	1.1053	1.1325
	EC-L2O	1.1728	1.1737	1.1754	1.1731	1.1750	1.1784	1.1727	1.1757	1.1815
	ROBD	1.6048	1.6048	1.6048	1.6048	1.6049	1.6049	1.6048	1.6048	1.6049
	HitMin	1.2112	1.2195	1.2302	1.2202	1.2357	1.2557	1.2410	1.2724	1.3127
CR	`RCL`	2.5028	2.9697	3.2247	2.6553	3.0283	3.2711	2.5714	3.0123	3.1653
	`RCL-0`	2.4209	2.4209	2.4209	2.4209	2.4209	2.4209	2.4209	2.4209	2.4209
	ML	6.5159	8.9245	11.6627	4.4025	6.5090	9.4168	5.5798	7.3956	9.3903
	EC-L2O	3.4639	4.6034	5.9666	2.6545	3.6740	5.1129	2.9766	3.7713	4.6983
	ROBD	1.7291	1.7291	1.7291	1.7291	1.7291	1.7291	1.7291	1.7296	1.7298
	HitMin	4.8573	6.7746	8.8383	3.1492	4.8253	7.0405	5.0632	6.9699	8.9246

Table 2: Average cost and competitive ratio comparison of different algorithms. We study the effect of introducing out-of-distribution (OOD) samples. Within the testing dataset, we randomly select a fraction of p_c of samples and add some random noise following $\mathcal{N}(0, \sigma)$ to contaminate these data samples (whose input values are all normalized within $[0, 1]$).

With the default $\lambda = 1$, we show the average cost and competitive ratio results in Table 2. We see that ROBD is very robust and little affected by the distributional shifts. In terms of the competitive ratio, ML, HitMin and EC-L2O are not robust, resulting in a large competitive ratio when we add more noisy samples. The average cost performance of `RCL` is empirically better than that of `RCL-0` in almost all cases, except for a slight increase in the practically very rare case where 20% samples are contaminated with large noise. On the other hand, as expected, the competitive ratios of `RCL` and `RCL-0` both increase as we add more noise. While `RCL` has a higher competitive ratio than `RCL-0` empirically in the experiment, they both have the same guaranteed $(1 + \lambda)$ competitiveness against ROBD regardless of how their ML models are trained. Also, their competitive ratios are both better than other algorithms, showing the effectiveness of our novel robustification process.

B.4 Results with Feedback Delay

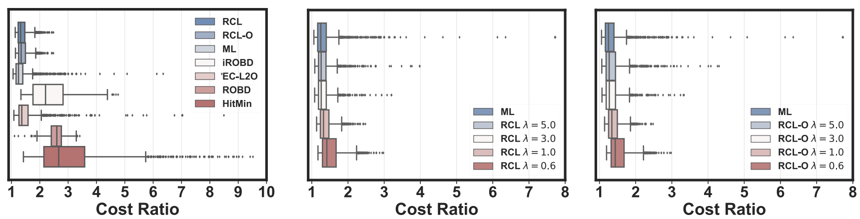
We now turn to the case when there is a one-step feedback delay, i.e., the context parameter y_t is not known to the agent until time $t + 1$. For this setting, we consider the best-known online algorithm iROBD [49] as the expert that handles the feedback delay with a guaranteed competitive ratio with respect to OPT. The other baseline online algorithms — ROBD, EC-L2O, and HitMin— presented in Section B.2 require the immediate revelation of y_t without feedback delay and hence do not directly apply to this case. Thus, for comparison, we use the predicted context, denoted by $\hat{y}_t$, with up to 15% prediction errors in the baseline online algorithms, and reuse the algorithm names (e.g., EC-L2O uses predicted $\hat{y}_t$ as if it were the true context for decision making). We train ML using the same architecture as in Section B.3, with the exception that only delayed context is provided as input for both training and testing. The reported values are normalized with respect to those of the respective offline optimal algorithm OPT. The average cost (**AVG**) and competitive ratio (**CR**) are all empirical results reported on the testing dataset.

We show the results in Table 3 and Fig. 4. We see that with the default $\lambda = 1$, both `RCL` and `RCL-0` have a good average cost, but `RCL` has a lower average cost than `RCL-0` and is outperformed only by ML in terms of the average cost. `RCL` and `RCL-0` have the same competitive ratio guarantee (i.e., $(1 + \lambda)$ times the competitive ratio of iROBD). Nonetheless, `RCL` has the lowest competitive ratio than all the other algorithms, demonstrating the power of `RCL` to leverage both ML prediction and the robust expert. In this experiment, both `RCL` and `RCL-0` outperform iROBD in terms of the empirical competitive ratio because they are able to exploit the good ML predictions for those problem instances that are difficult for iROBD.

By comparison, ML performs well on average by exploiting the historical data, but has a high competitive ratio. The alternative baselines — ROBD, EC-L2O and HitMin— use predicted context $\hat{y}_t$ as the true context. Except for the good empirical competitive ratio of ROBD, they do not have good average performance or guaranteed robustness due to their naively trusting the predicted context (that can potentially have large prediction errors). Note that the empirical competitive ratio of ROBD with predicted context is still much higher than that with the true context in Table 1. These results reinforce the point that blindly using ML predictions (i.e., predicted context in this example) without

	RCL				RCL-O				ML	EC-L2O	iROBD	HitMin	ROBD
	λ=0.6	λ=1	λ=3	λ=5	λ=0.6	λ=1	λ=3	λ=5					
AVG	1.5011	1.3594	1.2874	1.2899	1.5134	1.3690	1.2949	1.3026	**1.2792**	1.4112	2.3076	2.6095	2.5974
CR	2.9797	**2.4832**	3.2049	3.9847	2.9797	2.4832	3.3367	4.3040	8.4200	15.1928	4.7632	26.0264	2.8478

Table 3: Competitive ratio and average cost comparison of different algorithms with feedback delay.



(a) iROBD as the expert (b) RCL w/ different λ (c) RCL-O w/ different λ

Figure 4: Cost ratio distributions with feedback delay ($\lambda = 1$ by default)

additional robustification can lead to poor performance in terms of both average cost and worst-case cost ratio.

We further show in Fig. 4 the box plots for cost ratios of different algorithms, providing a detailed view of the algorithms' performance. The key message is that RCL obtains the best of both worlds — a good average cost and a good competitive ratio. Moreover, we see that by setting $\lambda = 1$, we provide enough freedom to RCL to exploit the benefits of ML predictions while also ensuring worst-case robustness. Thus, like in the no-delay case in Table 1 and Fig. 2, the empirical competitive ratio of RCL with $\lambda = 1$ is even lower than that with $\lambda = 0.6$.

C Proof of Theorems and Corollaries in Section 4

C.1 Proof of Theorem 4.1 (Cost Ratio)

To prove Theorem 4.1, we first give some technical lemmas about the smoothness of cost functions from Lemma C.1 to Lemma C.3.

Lemma C.1 (Lemma 4 in [59])**.** *Assume $f(x)$ is β smooth, for any $\lambda > 0$, we have*

$$f(x) \le (1+\lambda) f(y) + (1+\frac{1}{\lambda})\frac{\beta}{2}\|x-y\|^2 \quad \forall x, y \in \mathcal{X}$$

Lemma C.2. *Assume $f(x)$ is β_1 smooth and $d(x)$ is β_2 smooth, then $f(x) + d(x)$ is $\beta_1 + \beta_2$ smooth.*

Lemma C.3. *Suppose that the hitting cost $f(x, y_t)$ is β_h-smooth with respect to x, The switching cost is $d(x_t, x_{t-1}) = \frac{1}{2}\|x_t - \delta(x_{t-p:t-1})\|^2$, where $\delta(\cdot)$ is L_i-Lipschitz with respect to x_{t-i}. Then for any two action sequences $x_{1:T}$ and $x'_{1:T}$, we must have*

$$\text{cost}(x_{1:T}) - (1+\lambda)\text{cost}(x'_{1:T}) \le \frac{\beta + (1+\sum_{k=1}^{p} L_k)^2}{2}(1+\frac{1}{\lambda})\|x_{1:T} - x'_{1:T}\|^2, \quad \forall \lambda > 0 \tag{8}$$

Proof. The objective to be bounded can be decomposed as

$$\begin{aligned} &\text{cost}(x_{1:T}) - (1+\lambda)\text{cost}(x'_{1:T}) \\ =&\left(\sum_{t=1}^{T} f(x_t, y_t) - (1+\lambda) f(x'_t, y_t)\right) + \frac{1}{2}\left(\sum_{t=1}^{T}\|x_t - \delta(x_{t-p:t-1})\|^2 - (1+\lambda)\|x'_t - \delta(x'_{t-p:t-1})\|^2\right) \end{aligned} \tag{9}$$

Since hitting cost is β_h-smooth, then

$$\sum_{t=1}^{T} f(x_t, y_t) - (1+\lambda) f(x'_t, y_t) \le \frac{\beta_h}{2}(1+\frac{1}{\lambda})\sum_{t=1}^{T}\|x_t - x'_t\|^2 \tag{10}$$

Besides, based on the Lipschitz assumption of function $\delta(\cdot)$, we have

$$\begin{aligned}
&\|x_t-\delta(x_{t-p:t-1})\|^2-(1+\lambda)\|x'_t-\delta(x'_{t-p:t-1})\|^2\\
\le&(1+\frac{1}{\lambda})\|(x_t-x'_t)+(\delta(x_{t-p:t-1})-\delta(x'_{t-p:t-1}))\|^2\\
\le&(1+\frac{1}{\lambda})\left(\|x_t-x'_t\|+\|\delta(x_{t-p:t-1})-\delta(x'_{t-p:t-1})\|\right)^2\\
\le&(1+\frac{1}{\lambda})\left(\|x_t-x'_t\|+\sum_{k=1}^{p}L_k\|x_{t-k}-x'_{t-k}\|\right)^2\\
\le&(1+\frac{1}{\lambda})(1+\sum_{k=1}^{p}L_k)\left(\|x_t-x'_t\|^2+\sum_{k=1}^{p}L_k\|x_{t-k}-x'_{t-k}\|^2\right)
\end{aligned}\tag{11}$$

Summing up the switching costs of all time steps together, we have

$$\begin{aligned}
&\sum_{t=1}^{T}\|x_t-\delta(x_{t-p:t-1})\|^2-(1+\lambda)\|x'_t-\delta(x'_{t-p:t-1})\|^2\\
\le&(1+\frac{1}{\lambda})(1+\sum_{k=1}^{p}L_k)\sum_{t=1}^{T}\left(\|x_t-x'_t\|^2+\sum_{k=1}^{p}L_k\|x_{t-k}-x'_{t-k}\|^2\right)\\
\le&(1+\frac{1}{\lambda})(1+\sum_{k=1}^{p}L_k)\sum_{t=1}^{T}(1+\sum_{k=1}^{p}L_k)\|x_t-x'_t\|^2\\
=&(1+\frac{1}{\lambda})(1+\sum_{k=1}^{p}L_k)^2\sum_{t=1}^{T}\|x_t-x'_t\|^2
\end{aligned}\tag{12}$$

Substituting Eqn. (12) and Eqn. (10) into Eqn. (9), we finish the proof. □

Now we propose Lemma C.4 based on these above lemmas, which ensures the feasibility of robustness constraint in Eqn. (1)

Lemma C.4. *Let π be any expert algorithm for the SOCO problem with multi-step feedback delays and multi-step switching costs, for any $\lambda \ge 0$ and $\lambda \ge \lambda_0 \ge 0$, the total cost by the projected actions x_t must satisfy cost$(x_{1:T}) \le (1+\lambda)$cost$(x^\pi_{1:T})$*

Proof. We prove by induction that the constraints in Eqn. (1) are satisfied for each t. For $t=1$, since we assume the initial actions are the same ($x_{-p+1:0}=x^\pi_{-p+1:0}$), it is obvious that $x=x^\pi_1$ satisfies the robustness constraints Eqn. (1).

Then for any time step $t\ge 2$, suppose it holds at $t-1$ that

$$\begin{aligned}
&\sum_{\tau\in\mathcal{A}_{t-1}}f(x_\tau,y_\tau)+\sum_{\tau\in\mathcal{A}_{t-1}\cup\mathcal{B}_{t-1}}d(x_\tau,x_{\tau-p:\tau-1})+\sum_{\tau\in\mathcal{B}_{t-1}}H(x_\tau,x^\pi_\tau)+G(x,x_{t-p:t-1},x^\pi_{t-p:t})\\
\le&(1+\lambda)\Bigg(\sum_{\tau\in\mathcal{A}_{t-1}}f(x^\pi_\tau,y_\tau)+\sum_{\tau\in\mathcal{A}_{t-1}\cup\mathcal{B}_{t-1}}d(x^\pi_\tau,x^\pi_{\tau-p:\tau-1})\Bigg)
\end{aligned}\tag{13}$$

Now the robustness constraints Eqn. (1) is satisfied if we prove $x_t=x^\pi_t$ satisfies the constraints in Eqn. (1) at time step t. Since for the sets $\mathcal{A}$ and $\mathcal{B}$, we have

$$(\mathcal{A}_t\cup\mathcal{B}_t)\backslash(\mathcal{A}_{t-1}\cup\mathcal{B}_{t-1})=\{t\},\quad \mathcal{A}_{t-1}\subseteq\mathcal{A}_t,\tag{14}$$

so it holds that

$$\sum_{\tau\in\mathcal{A}_t\cup\mathcal{B}_t}d(x_\tau,x_{\tau-p:\tau-1})-\sum_{\tau\in\mathcal{A}_{t-1}\cup\mathcal{B}_{t-1}}d(x_\tau,x_{\tau-p:\tau-1})=d(x_t,x_{t-p:t-1})\tag{15}$$

By Lemma C.1, we have

$$\begin{aligned}
&d(x_t^\pi, x_{t-p:t-1}) - (1+\lambda)d(x_t^\pi, x_{t-p:t-1}^\pi)\\
\leq&\frac{1}{2}(1+\frac{1}{\lambda})\|\delta(x_{t-p:t-1}) - \delta(x_{t-p:t-1}^\pi)\|^2\\
\leq&\frac{1}{2}(1+\frac{1}{\lambda})\left(\sum_{i=1}^p L_i\|x_{t-i} - x_{t-i}^\pi\|\right)^2
\end{aligned}\tag{16}$$

Denote $\alpha = 1 + \sum_{k=1}^p L_k$. For the reservation cost, we have

$$\begin{aligned}
&G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) - G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi)\\
=&\frac{\alpha(1+\frac{1}{\lambda_0})}{2}\left(\sum_{k=1}^p\sum_{i=0}^{p-k} L_{k+i}\|x_{t-i-1} - x_{t-i-1}^\pi\|^2 - \sum_{k=1}^p\sum_{i=1}^{p-k} L_{k+i}\|x_{t-i} - x_{t-i}^\pi\|^2\right)\\
=&\frac{\alpha(1+\frac{1}{\lambda_0})}{2}\left(\sum_{k=0}^{p-1}\sum_{i=1}^{p-k} L_{k+i}\|x_{t-i} - x_{t-i}^\pi\|^2 - \sum_{k=1}^p\sum_{i=1}^{p-k} L_{k+i}\|x_{t-i} - x_{t-i}^\pi\|^2\right)\\
=&\frac{\alpha(1+\frac{1}{\lambda_0})}{2}\left(\sum_{k=0}^{p-1}\sum_{i=1}^{p-k} L_{k+i}\|x_{t-i} - x_{t-i}^\pi\|^2 - \sum_{k=1}^{p-1}\sum_{i=1}^{p-k} L_{k+i}\|x_{t-i} - x_{t-i}^\pi\|^2\right)\\
=&\frac{\alpha(1+\frac{1}{\lambda_0})}{2}\sum_{i=1}^p L_i\|x_{t-i} - x_{t-i}^\pi\|^2
\end{aligned}\tag{17}$$

Continuing with Eqn. (17), we have

$$\begin{aligned}
&G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) - G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) = \frac{\alpha(1+\frac{1}{\lambda_0})}{2}\sum_{i=1}^p L_i\|x_{t-i} - x_{t-i}^\pi\|^2\\
\geq&\frac{(1+\frac{1}{\lambda_0})(\sum_{i=1}^p L_i)^2}{2}\sum_{i=1}^p \frac{L_i}{\sum_{i=1}^p L_i}\|x_{t-i} - x_{t-i}^\pi\|^2\\
\geq&\frac{(1+\frac{1}{\lambda_0})(\sum_{i=1}^p L_i)^2}{2}\left(\sum_{i=1}^p \frac{L_i}{\sum_{i=1}^p L_i}\|x_{t-i} - x_{t-i}^\pi\|\right)^2\\
=&\frac{1}{2}(1+\frac{1}{\lambda_0})\left(\sum_{i=1}^p L_i\|x_{t-i} - x_{t-i}^\pi\|\right)^2 \geq \frac{1}{2}(1+\frac{1}{\lambda})\left(\sum_{i=1}^p L_i\|x_{t-i} - x_{t-i}^\pi\|\right)^2
\end{aligned}\tag{18}$$

where the second inequality holds by Jensen's inequality. Therefore, combining with (16), we have

$$d(x_t^\pi, x_{t-p:t-1}) + G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) \leq G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) + (1+\lambda)d(x_t^\pi, x_{t-p:t-1}^\pi)\tag{19}$$

By Eqn. (19), we have

$$\begin{aligned}
&G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) + \sum_{\tau\in\mathcal{A}_t\cup\mathcal{B}_t} d(x_\tau, x_{\tau-p:\tau-1}) - \sum_{\tau\in\mathcal{A}_{t-1}\cup\mathcal{B}_{t-1}} d(x_\tau, x_{\tau-p:\tau-1})\\
\leq&G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) + (1+\lambda)\left(\sum_{\tau\in\mathcal{A}_t\cup\mathcal{B}_t} d(x_\tau^\pi, x_{\tau-p:\tau-1}^\pi) - \sum_{\tau\in\mathcal{A}_{t-1}\cup\mathcal{B}_{t-1}} d(x_\tau^\pi, x_{\tau-p:\tau-1}^\pi)\right)
\end{aligned}\tag{20}$$

Now we define a new set $\mathcal{D}_t = \mathcal{A}_t \backslash \mathcal{A}_{t-1}$, which denotes the timestep set for the newly received context parameters at t.

Case 1: If $t \in \mathcal{D}_t$, then $B_{t-1} \backslash B_t = \mathcal{D}_t \backslash \{t\}$, then we have

$$\begin{aligned}
&\left(\sum_{\tau \in \mathcal{A}_t} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_t} H(x_\tau, x_\tau^\pi)\right) - \left(\sum_{\tau \in \mathcal{A}_{t-1}} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_{t-1}} H(x_\tau, x_\tau^\pi)\right) \\
=& \sum_{\tau \in \mathcal{D}_t} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} H(x_\tau, x_\tau^\pi) = f(x_t^\pi, y_t) + \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} H(x_\tau, x_\tau^\pi)
\end{aligned} \tag{21}$$

Since hitting cost $f(\cdot, y_t)$ is β_h-smooth, we have

$$\begin{aligned}
&\sum_{\tau \in \mathcal{D}_t \backslash \{t\}} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} (1+\lambda) f(x_\tau^\pi, y_\tau) \\
\leq& \frac{\beta_h (1 + \frac{1}{\lambda})}{2} \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} \|x_\tau^\pi - x_\tau\|^2 \leq \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} H(x_\tau, x_\tau^\pi)
\end{aligned} \tag{22}$$

Substituting Eqn. (22) back to Eqn. (21), we have

$$\begin{aligned}
&\left(\sum_{\tau \in \mathcal{A}_t} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_t} H(x_\tau, x_\tau^\pi)\right) - \left(\sum_{\tau \in \mathcal{A}_{t-1}} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_{t-1}} H(x_\tau, x_\tau^\pi)\right) \\
\leq& (1+\lambda) \left(\sum_{\tau \in \mathcal{A}_t} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{A}_{t-1}} f(x_\tau, y_\tau)\right)
\end{aligned} \tag{23}$$

Case 2: If $t \notin \mathcal{D}_t$, then $(B_{t-1} \cup \{t\}) \backslash B_t = \mathcal{D}_t$ and we have

$$\begin{aligned}
&\left(\sum_{\tau \in \mathcal{A}_t} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_t} H(x_\tau, x_\tau^\pi)\right) - \left(\sum_{\tau \in \mathcal{A}_{t-1}} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_{t-1}} H(x_\tau, x_\tau^\pi)\right) \\
=& \sum_{\tau \in \mathcal{D}_t} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{D}_t} H(x_\tau, x_\tau^\pi) + H(x_t^\pi, x_t^\pi) \\
=& \sum_{\tau \in \mathcal{D}_t} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{D}_t} H(x_\tau, x_\tau^\pi)
\end{aligned} \tag{24}$$

Since hitting cost $f(\cdot, y_t)$ is β_h-smooth, we have

$$\sum_{\tau \in \mathcal{D}_t} f(x_\tau, y_\tau) - \sum_{\tau \in \mathcal{D}_t} (1+\lambda) f(x_\tau^\pi, y_\tau) \leq \frac{\beta_h (1 + \frac{1}{\lambda})}{2} \sum_{\tau \in \mathcal{D}_t} \|x_\tau^\pi - x_\tau\|^2 \leq \sum_{\tau \in \mathcal{D}_t} H(x_\tau, x_\tau^\pi) \tag{25}$$

Since $\lambda \geq 0$, we substitute Eqn. (25) back to Eqn. (24), we have the same conclusion as Eqn (23).

Adding Eqn. (13), Eqn. (20) and Eqn. (23) together, we can prove $x = x_t^\pi$ satisfies the constraints in Eqn. (1). At time step T, we have

$$\begin{aligned}
&\sum_{\tau \in \mathcal{A}_T} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{A}_T \cup \mathcal{B}_T} d(x_\tau, x_{\tau-p:\tau-1}) + \sum_{\tau \in \mathcal{B}_T} \left(f(x_\tau, y_\tau) - (1+\lambda) f(x_\tau^\pi, y_\tau)\right) \\
\leq& \sum_{\tau \in \mathcal{A}_T} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{A}_T \cup \mathcal{B}_T} d(x_\tau, x_{\tau-p:\tau-1}) + \sum_{\tau \in \mathcal{B}_T} H(x_\tau, x_\tau^\pi) \\
\leq& (1+\lambda) \left(\sum_{\tau \in \mathcal{A}_T} f(x_\tau^\pi, y_\tau) + \sum_{\tau \in \mathcal{A}_T \cup \mathcal{B}_T} d(x_\tau^\pi, x_{\tau-p:\tau-1}^\pi)\right)
\end{aligned} \tag{26}$$

In other words

$$\sum_{\tau \in \mathcal{A}_T \cup \mathcal{B}_T} \left(f(x_\tau, y_\tau) + d(x_\tau, x_{\tau-p:\tau-1})\right) \leq (1+\lambda) \sum_{\tau \in \mathcal{A}_T \cup \mathcal{B}_T} \left(f(x_\tau^\pi, y_\tau) + d(x_\tau, x_{\tau-p:\tau-1})\right) \tag{27}$$

$\square$

In the next lemma, we bound the difference between the projected action and the ML predictions.

Lemma C.5. *Suppose hitting cost is β_h-smooth, given the expert policy π, ML predictions $\tilde{x}_{1:T}$, for any $\lambda > 0$ and $\lambda_1 > 0$, the total distance between actual actions $x_{1:T}$ and ML predictions $\tilde{x}_{1:T}$ are bounded,*

$$\sum_{i=1}^{T} \|x_t - \tilde{x}_t\|^2 \leq \sum_{i=1}^{T} \left(\left[\|\tilde{x}_t - x_t^\pi\| - \sqrt{K \left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau \in \mathcal{D}_t} f(x_\tau^\pi, y_\tau) \right)} \right]^+ \right)^2 \tag{28}$$

where $[\cdot]^+$ is the ReLU function and $K = \frac{2(\lambda - \lambda_0)}{\beta_h(1+\frac{1}{\lambda_0}) + \alpha^2(1+\frac{1}{\lambda_0})}$, $\alpha = 1 + \sum_{i=1}^{p} L_i$

Proof. Suppose we at $t-1$ have the following inequality:

$$\begin{aligned}
&\sum_{\tau \in \mathcal{A}_{t-1}} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{A}_{t-1} \cup \mathcal{B}_{t-1}} d(x_\tau, x_{\tau-p:\tau-1}) + \sum_{\tau \in \mathcal{B}_{t-1}} H(x_\tau, x_\tau^\pi) + G(x, x_{t-p:t-1}, x_{t-p:t}^\pi) \\
\leq &(1+\lambda) \left(\sum_{\tau \in \mathcal{A}_{t-1}} f(x_\tau^\pi, y_\tau) + \sum_{\tau \in \mathcal{A}_{t-1} \cup \mathcal{B}_{t-1}} d(x_\tau^\pi, x_{\tau-p:\tau-1}^\pi) \right)
\end{aligned} \tag{29}$$

Remember that $\mathcal{D}_t = \mathcal{A}_t \backslash \mathcal{A}_{t-1}$ is the set of the time steps for the newly received context parameters at t. The robustness constraint in Eqn. (1) is satisfied if x_t satisfies the following inequality.

$$\begin{aligned}
&\left(\sum_{\tau \in \mathcal{D}_t} f(x_\tau, y_\tau) + \sum_{\tau \in \mathcal{B}_t} H(x_\tau, x_\tau^\pi) - \sum_{\tau \in \mathcal{B}_{t-1}} H(x_\tau, x_\tau^\pi) \right) + d(x_t, x_{t-p:t-1}) + G(x_t, x_{t-p:t-1}, x_{t-p:t}^\pi) \\
&- G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) \leq (1+\lambda) \left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau \in \mathcal{D}_t} f(x_\tau^\pi, y_\tau) \right)
\end{aligned} \tag{30}$$

For the switching cost, we have

$$\begin{aligned}
&d(x, x_{t-p:t-1}) - (1+\lambda_0) d(x_t^\pi, x_{t-p:t-1}^\pi) \\
\leq &\frac{1}{2}(1 + \frac{1}{\lambda_0}) \left(\|x - x_t^\pi\| + \|\delta(x_{t-p:t-1}) - \delta(x_{t-p:t-1}^\pi)\| \right)^2 \\
\leq &\frac{1}{2}(1 + \frac{1}{\lambda_0}) \left(\|x - x_t^\pi\| + \sum_{i=1}^{p} L_i \|x_{t-i} - x_{t-i}^\pi\| \right)^2 \\
\leq &\frac{\alpha(1+\frac{1}{\lambda_0})}{2} \left(\|x - x_t^\pi\|^2 + \sum_{i=1}^{p} L_i \|x_{t-i} - x_{t-i}^\pi\|^2 \right)
\end{aligned} \tag{31}$$

The first inequality comes from Lemma C.1, the second inequality comes from the L_i-Lipschitz assumption, and the third inequality is because $\alpha \geq 1$. Besides, from Eqn (17), we have

$$G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) - G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) = \frac{\alpha(1+\frac{1}{\lambda_0})}{2} \sum_{i=1}^{p} L_i \|x_{t-i} - x_{t-i}^\pi\|^2 \tag{32}$$

Thus we have

$$\begin{aligned}
&G(x, x_{t-p:t-1}, x_{t-p:t}^\pi) - G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) \\
= &G(x, x_{t-p:t-1}, x_{t-p:t}^\pi) - G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) + G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) - G(x_{t-1}, x_{t-p-1:t-2}, x_{t-p-1:t-1}^\pi) \\
= &G(x, x_{t-p:t-1}, x_{t-p:t}^\pi) - G(x_t^\pi, x_{t-p:t-1}, x_{t-p:t}^\pi) - \frac{\alpha(1+\frac{1}{\lambda_0})}{2} \sum_{i=1}^{p} L_i \|x_{t-i} - x_{t-i}^\pi\|^2.
\end{aligned} \tag{33}$$

Combining with inequality (31), we have

$$
\begin{aligned}
&G(x_t, x_{t-p:t-1}, x^\pi_{t-p:t}) - G(x_{t-1}, x_{t-p-1:t-2}, x^\pi_{t-p-1:t-1}) + d(x_t, x_{t-p:t-1}) - (1+\lambda_0) d(x^\pi_t, x^\pi_{t-p:t-1}) \\
\le & G(x_t, x_{t-p:t-1}, x^\pi_{t-p:t}) - G(x^\pi_t, x_{t-p:t-1}, x^\pi_{t-p:t}) + \frac{\alpha(1+\frac{1}{\lambda_0})}{2} \|x_t - x^\pi_t\|^2 \\
= & \frac{\alpha(1+\frac{1}{\lambda_0}) \sum_{k=1}^p L_k}{2} \|x_t - x^\pi_t\|^2 + \frac{\alpha(1+\frac{1}{\lambda_0})}{2} \sum_{k=1}^p \|x_t - x^\pi_t\|^2 \\
= & \frac{\alpha^2(1+\frac{1}{\lambda_0})}{2} \|x_t - x^\pi_t\|^2
\end{aligned}
\tag{34}
$$

Substituting Eqn. (34) back to Eqn. (30), we have

$$
\begin{aligned}
&\sum_{\tau \in \mathcal{D}_t} \left(f(x_\tau, y_\tau) - (1+\lambda_0) f(x^\pi_\tau, y_\tau) \right) + \sum_{\tau \in \mathcal{B}_t} H(x_\tau, x^\pi_\tau) - \sum_{\tau \in \mathcal{B}_{t-1}} H(x_\tau, x^\pi_\tau) \\
&+ \frac{\alpha^2(1+\frac{1}{\lambda_0})}{2} \|x - x^\pi_t\|^2 \le (\lambda - \lambda_0) \left(d(x^\pi_t, x^\pi_{t-p:t-1}) + \sum_{\tau \in \mathcal{D}_t} f(x^\pi_\tau, y_\tau) \right)
\end{aligned}
\tag{35}
$$

Case 1: If $t \in \mathcal{D}_t$, then $B_{t-1} \backslash B_t = \mathcal{D}_t \backslash \{t\}$, then Eqn.(35) becomes

$$
\begin{aligned}
&f(x_t, y_t) - (1+\lambda_0) f(x^\pi_t, y_t) + \frac{\alpha^2(1+\frac{1}{\lambda_0})}{2} \|x - x^\pi_t\|^2 \\
&+ \sum_{\tau \in \mathcal{D}_t \backslash \{t\}} f(x_\tau, y_\tau) - (1+\lambda_0) f(x^\pi_\tau, y_\tau) - H(x_\tau, x^\pi_\tau) \le (\lambda - \lambda_0) \left(d(x^\pi_t, x^\pi_{t-p:t-1}) + \sum_{\tau \in \mathcal{D}_t} f(x^\pi_\tau, y_\tau) \right)
\end{aligned}
\tag{36}
$$

Since hitting cost is β_h-smooth, the sufficient condition for Eqn. (35) becomes

$$
\frac{(\beta_h + \alpha^2)(1+\frac{1}{\lambda_0})}{2} \|x - x^\pi_t\|^2 \le (\lambda - \lambda_0) \left(d(x^\pi_t, x^\pi_{t-p:t-1}) + \sum_{\tau \in \mathcal{D}_t} f(x^\pi_\tau, y_\tau) \right) \tag{37}
$$

Since the hitting cost is non-negative, the sufficient condition can be further simplified, which is

$$
\frac{(\beta_h + \alpha^2)(1+\frac{1}{\lambda_0})}{2} \|x - x^\pi_t\|^2 \le (\lambda - \lambda_0) \left(f(x^\pi_t, y_t) + d(x^\pi_t, x^\pi_{t-p:t-1}) \right) \tag{38}
$$

Case 2: If $t \notin \mathcal{D}_t$, then $(B_{t-1} \cup \{t\}) \backslash B_t = \mathcal{D}_t$, then Eqn.(35) becomes

$$
\begin{aligned}
&\frac{\alpha^2(1+\frac{1}{\lambda_0})}{2} \|x - x^\pi_t\|^2 + H(x, x^\pi_t) + \sum_{\tau \in \mathcal{D}_t} \left(f(x_\tau, y_\tau) - (1+\lambda_0) f(x^\pi_\tau, y_\tau) - H(x_\tau, x^\pi_\tau) \right) \\
&\le (\lambda - \lambda_0) \left(d(x^\pi_t, x^\pi_{t-p:t-1}) + \sum_{\tau \in \mathcal{D}_t} f(x^\pi_\tau, y_\tau) \right)
\end{aligned}
\tag{39}
$$

Since hitting cost is β_h-smooth, the sufficient condition for Eqn. (39) becomes

$$
\frac{(\beta_h + \alpha^2)(1+\frac{1}{\lambda_0})}{2} \|x - x^\pi_t\|^2 \le (\lambda - \lambda_0) \left(d(x^\pi_t, x^\pi_{t-p:t-1}) + \sum_{\tau \in \mathcal{D}_t} f(x^\pi_\tau, y_\tau) \right) \tag{40}
$$

Now we define

$$
K = \frac{2(\lambda - \lambda_0)}{(\beta_h + \alpha^2)(1+\frac{1}{\lambda_0})}
$$

At time step t, if x'_t is the solution to this alternative optimization problem

$$\begin{aligned} x'_t &= \arg\min_x \frac{1}{2}\|x - \tilde{x}_t\|^2 \\ s.t. \quad & \|x - x_t^\pi\|^2 \le K\left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)\right) \end{aligned} \tag{41}$$

The solution to this problem can be calculated asd

$$\begin{aligned} x'_t &= \theta x_t^\pi + (1-\theta)\tilde{x}_t \\ \theta &= \left[1 - \frac{\sqrt{K\left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)\right)}}{\|\tilde{x}_t - x_t^\pi\|}\right]^+. \end{aligned} \tag{42}$$

Then $\|x'_t - \tilde{x}_t\| = \left[\|\tilde{x}_t - x_t^\pi\| - \sqrt{K\left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)\right)}\right]^+$. Since x'_t also satisfies the original robustness constraint, we have $\|x_t - \tilde{x}_t\| \le \|x'_t - \tilde{x}_t\|$ and we finish the proof.

□

Proof of Theorem 4.1

Now summing up the distance through 1 to T, we have

$$\sum_{i=1}^T \|x_t - \tilde{x}_t\|^2 \le \sum_{i=1}^T \left(\left[\|\tilde{x}_t - x_t^\pi\| - \sqrt{K\left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)\right)}\right]^+\right)^2 \tag{43}$$

Based on Lemma C.3 we have $\forall \lambda_2 > 0$,

$$\text{cost}(x_{1:T}) - (1+\lambda_2)\text{cost}(\tilde{x}_{1:T}) \le \frac{\beta+\alpha^2}{2}(1+\frac{1}{\lambda_2})\sum_{i=1}^T \|x_t - \tilde{x}_t\|^2. \tag{44}$$

Suppose the offline optimal action sequence is $x^*_{1:T}$, the optimal cost is $\text{cost}(x^*_{1:T})$. Then we divide both sides of Eqn. (44) by $\text{cost}(x^*_{1:T})$, and get $\forall \lambda_2 > 0$,

$$\begin{aligned} \text{cost}(x_{1:T}) \le & (1+\lambda_2)\text{cost}(\tilde{x}_{1:T}) + \frac{\beta+\alpha^2}{2}(1+\frac{1}{\lambda_2})\cdot \\ & \sum_{i=1}^T \left(\left[\|\tilde{x}_t - x_t^\pi\| - \sqrt{K\left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)\right)}\right]^+\right)^2 \end{aligned} \tag{45}$$

By substituting $K = \frac{2(\lambda-\lambda_0)}{(\beta_h+\alpha^2)(1+\frac{1}{\lambda_0})}$ back to Eqn (46), we have

$$\begin{aligned} \text{cost}(x_{1:T}) \le & (1+\lambda_2)\text{cost}(\tilde{x}_{1:T}) + (1+\frac{1}{\lambda_2})\sum_{i=1}^T \left[\frac{\beta+\alpha^2}{2}\|\tilde{x}_t - x_t^\pi\|^2 \right. \\ & \left. - \frac{\lambda-\lambda_0}{1+\frac{1}{\lambda_0}}\left(d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)\right)\right]^+ \end{aligned} \tag{46}$$

By defining single step cost of the expert π as $\text{cost}_t^\pi = d(x_t^\pi, x_{t-p:t-1}^\pi) + \sum_{\tau\in\mathcal{D}_t} f(x_\tau^\pi, y_\tau)$ and the auxiliary cost as $\Delta(\lambda) = \sum_{i=1}^T \left[\|\tilde{x}_t - x_t^\pi\|^2 - \frac{2(\lambda-\lambda_0)}{(\beta_h+\alpha^2)(1+\frac{1}{\lambda_0})}\text{cost}_t^\pi\right]^+$

$$\text{cost}(x_{1:T}) \le \left(\sqrt{\text{cost}(\tilde{x}_{1:T})} + \sqrt{\frac{\beta+\alpha^2}{2}\Delta(\lambda)}\right)^2 \tag{47}$$

Combined with Lemma C.4, we obtain the following bound, which finished this proof.

$$\text{cost}(x_{1:T}) \le \min\left((1+\lambda)\text{cost}(x_{1:T}^\pi), \left(\sqrt{\text{cost}(\tilde{x}_{1:T})} + \sqrt{\frac{\beta+\alpha^2}{2}\Delta(\lambda)}\right)^2\right) \tag{48}$$

C.2 Proof of Theorem 4.2

Proof. We first give the formal definition of Rademacher complexity of the ML model space with robustification.

Definition 5 (Rademacher Complexity). *Let* $\mathsf{Rob}_\lambda(\mathcal{W}) = \{\mathsf{Rob}_\lambda(h_W), W \in \mathcal{W}\}$ *be the ML model space with robustification constrained by* (2)*. Given the dataset* $\mathcal{S}$*, the Rademacher complexity with respect to* $\mathsf{Rob}_\lambda(\mathcal{W})$ *is*

$$\mathrm{Rad}_{\mathcal{S}}(\mathsf{Rob}_\lambda(\mathcal{W})) = \frac{1}{|\mathcal{S}|}\mathbb{E}_\nu\left[\sup_{W\in\mathcal{W}}\left(\sum_{i\in\mathcal{S}}\nu_i \mathsf{Rob}_\lambda\left(h_W(y^i)\right)\right)\right],$$

where y^i *is the* i*-th sample in* $\mathcal{S}$*, and* $\nu_1, \cdots, \nu_n$ *are independently drawn from Rademacher distribution.*

Since the cost functions are smooth, they are locally Lipschitz continuous for the bounded action space, and we can apply the generalization bound based on Rademacher complexity [60] for the space of robustified ML model $\mathsf{Rob}_\lambda(h_W)$. Given any ML model h_W trained on dataset $\mathcal{S}$, with probability at least $1-\delta, \delta \in (0,1)$,

$$\mathbb{E}_{\mathbb{P}'_y}[\mathrm{cost}_{1:T}] \leq \overline{\mathrm{cost}}_{\mathcal{S}}(\mathsf{Rob}_\lambda(h_W)) + 2\Gamma_x \mathrm{Rad}_{\mathcal{S}}(\mathsf{Rob}_\lambda(\mathcal{W})) + 3\bar{c}\sqrt{\frac{\log(2/\delta)}{|\mathcal{S}|}}, \tag{49}$$

where $\Gamma_x = \sqrt{T}|\mathcal{X}|\left[\beta_h + \frac{1}{2}(1+\sum_{i=1}^p L_i)(1+\sum_{i=1}^p L_i)\right]$ with $|\mathcal{X}|$ being the size of the action space $\mathcal{X}$ and β_h, L_i, and p as the smoothness constant, Lipschitz constant of the nonlinear term in the switching cost, and the memory length as defined in Assumptions 1 and 2, and $\bar{c}$ is the upper bound of the total cost for an episode. We can get the average cost bound in Proposition 4.2.

Next, we prove that the Rademacher complexity of the ML model space with robustification is no larger than the Rademacher complexity of the ML model space without robustification expressed as $\{h_W, W \in \mathcal{W}\}$, i.e. we need to prove $\mathrm{Rad}_{\mathcal{S}}(\mathsf{Rob}_\lambda(\mathcal{W})) \leq \mathrm{Rad}_{\mathcal{S}}(\mathcal{W})$. The Rademacher complexity can be expressed by Dudley's entropy integral [61] as

$$\mathrm{Rad}_{\mathcal{S}}(\mathsf{Rob}_\lambda(\mathcal{W})) = \mathcal{O}\left(\frac{1}{\sqrt{|\mathcal{S}|}}\int_0^\infty \sqrt{\log \mathbb{N}(\epsilon, \mathsf{Rob}_\lambda(\mathcal{W}), L_2(\mathcal{S}))}\mathrm{d}\epsilon\right), \tag{50}$$

where $\mathbb{N}(\epsilon, \mathsf{Rob}_\lambda(\mathcal{W}), L_2(\mathcal{S}))$ is the covering number [61] with respect to radius ϵ and the function distance metric $\|h_1 - h_2\|_{L_2(\mathcal{S})} = \frac{1}{|\mathcal{S}|}\sum_{i\in\mathcal{S}}\|h_1(x_i) - h_2(x_i)\|^2$ where h_1 and h_2 are two functions defined on the space including dataset $\mathcal{S}$. We can find that for any two different weights W_1 and W_2, their corresponding post-robustification distance $\|\mathsf{Rob}_\lambda(h_{W_1}) - \mathsf{Rob}_\lambda(h_{W_2})\|_{L_2(\mathcal{S})}$ is no larger than their pre-robustification distance $\|h_{W_1} - h_{W_2}\|_{L_2(\mathcal{S})}$. To see this, we discuss three cases given any input sample y. If both $h_{W_1}(y)$ and $h_{W_2}(y)$ lie in the projection set, then $\mathsf{Rob}_\lambda(h_{W_1})(y) = h_{W_1}(y)$ and $\mathsf{Rob}_\lambda(h_{W_2})(y) = h_{W_2}(y)$. If $h_{W_1}(y)$ lies in the projection set while $h_{W_2}(y)$ is out of the projection set, the projection operation based on the closed convex projection set will make $\|\mathsf{Rob}_\lambda(h_{W_1})(y) - \mathsf{Rob}_\lambda(h_{W_2})(y)\|$ to be less than $\|h_{W_1}(y) - h_{W_2}(y)\|$. If both $h_{W_1}(y)$ and $h_{W_2}(y)$ lie out of the projection set, we still have $\|\mathsf{Rob}_\lambda(h_{W_1})(y) - \mathsf{Rob}_\lambda(h_{W_2})(y)\| \leq \|h_{W_1}(y) - h_{W_2}(y)\|$ since the projection set at each round is a closed convex set [62]. Therefore, after robustification, the distance between two models with different weights will not become larger, i.e. $\|\mathsf{Rob}_\lambda(h_{W_1}) - \mathsf{Rob}_\lambda(h_{W_2})\|_{L_2(\mathcal{S})} \leq \|h_{W_1} - h_{W_2}\|_{L_2(\mathcal{S})}$, which means `RCL` has a covering number $\mathbb{N}(\epsilon, \mathsf{Rob}_\lambda(\mathcal{W}), L_2(\mathcal{S}))$ no larger than that of the individual ML model $\mathbb{N}(\epsilon, \mathcal{W}, L_2(\mathcal{S}))$ for any ϵ. Thus the Rademacher complexity with the robustification procedure does not increase.

By [63], the upper bound of Rademacher complexity with respect to the space of ML model $\mathrm{Rad}_{\mathcal{S}}(\mathsf{Rob}_\lambda(\mathcal{W}))$ is in the order of $\mathcal{O}(\frac{1}{\sqrt{|\mathcal{S}|}})$. Since the Rademacher complexity with the robustification procedure satisfies $\mathrm{Rad}_{\mathcal{S}}(\mathsf{Rob}_\lambda(\mathcal{W})) \leq \mathrm{Rad}_{\mathcal{S}}(\mathcal{W})$, it also decreases with the dataset size in the order of $\mathcal{O}(\frac{1}{\sqrt{|\mathcal{S}|}})$. □

D Robustification-aware Training

Theorem 4.2 also shows the benefits of training the ML model in a robustification-aware manner. Specifically, by comparing the losses in (5) and (6), we see that using (6) as the robustification-aware loss for training W can reduce the term $\overline{\text{cost}}_{\mathcal{S}}(\mathsf{ROB}(h_W))$ in the average cost bound, which matches exactly with the training objective in (6). The robustification-aware approach is only beginning to be explored in the ML-augmented algorithm literature and non-trivial (e.g., unconstrained downstream optimization in [55]), especially considering that (1) is a constrained optimization problem with no explicit gradients.

Gradient-based optimizers such as Adam [64] are the de facto state-of-the-art algorithms for training ML models, offering better optimization results, convergence, and stability compared to those non-gradient-based alternatives [65]. Thus, it is crucial to derive the gradients of the loss function with respect to the ML model weight W given the added robustification step.

Next, we derive the gradients of x_t with respect to $\tilde{x}_t$. For the convenience of presentation, we use the basic SOCO setting with a single-step switching cost and no hitting cost delay as an example, while noting that the same technique can be extended to derive gradients in more general settings. Specifically, for this setting, the pre-robustification prediction is given by $\tilde{x}_t = h_W(\tilde{x}_{t-1}, y_t)$, where W denotes the ML model weight. Then, the actual post-robustification action x_t is obtained by projection in (1) by setting $q = 0$ and $p = 1$, given the ML prediction $\tilde{x}_t$, the expert's action x_t^π and cumulative $\text{cost}(x_{1:t}^\pi)$ up to t, and the actual cumulative $\text{cost}(x_{1:t-1})$ up to $t-1$.

The gradient of the loss function $\text{cost}(x_{1:T}) = \sum_{t=1}^{T}\left(f(x_t, y_t) + d(x_t, x_{t-1})\right)$ with respect to the ML model weight W is given by $\sum_{t=1}^{T} \nabla_W \big(f(x_t, y_t) + d(x_t, x_{t-1})\big)$. Next, we write the gradient of per-step cost with with respect to W as follows:

$$\begin{aligned}
&\nabla_W \big(f(x_t, y_t) + d(x_t, x_{t-1})\big) \\
=&\nabla_{x_t} \big(f(x_t, y_t) + d(x_t, x_{t-1})\big)\nabla_W x_t + \nabla_{x_{t-1}} \big(f(x_t, y_t) + d(x_t, x_{t-1})\big)\nabla_W x_{t-1} \\
=&\nabla_{x_t} \big(f(x_t, y_t) + d(x_t, x_{t-1})\big)\nabla_W x_t + \nabla_{x_{t-1}} d(x_t, x_{t-1})\nabla_W x_{t-1},
\end{aligned} \tag{51}$$

where the gradients $\nabla_{x_t}\big(f(x_t, y_t) + d(x_t, x_{t-1})\big)$ and $\nabla_{x_{t-1}} d(x_t, x_{t-1})$ are trivial given the hitting and switching cost functions, and the gradient $\nabla_W x_{t-1}$ is obtained at time $t-1$ in the same way as $\nabla_W x_t$. To derive $\nabla_W x_t$, by the chain rule, we have:

$$\nabla_W x_t = \nabla_{\tilde{x}_t} x_t \nabla_W \tilde{x}_t + \nabla_{\text{cost}(x_{1:t-1})} x_t \nabla_W \text{cost}(x_{1:t-1}), \tag{52}$$

where $\nabla_W \tilde{x}_t$ is the gradient of the ML output (following a recurrent architecture illustrated in Fig. 1 in the appendix) with respect to the weight W and can be obtained recursively by using off-the-shelf BPTT optimizers [64], and $\nabla_W \text{cost}(x_{1:t-1}) = \sum_{\tau=1}^{t-1} \nabla_W \big(f(x_\tau, y_\tau) + d(x_\tau, x_{\tau-1})\big)$ can also be recursively calculated once we have the gradient in Eqn. (51). Nonetheless, it is non-trivial to calculate the two gradient terms in Eqn. (52), i.e., $\nabla_{\tilde{x}_t} x_t$ and $\nabla_{\text{cost}(x_{1:t-1})} x_t$, where x_t itself is the solution to the constrained optimization problem (1) unlike in the simpler unconstrained case [55]. As we cannot explicitly write x_t in a closed form in terms of $\tilde{x}_t$ and $\text{cost}(x_{1:t-1})$, we leverage the KKT conditions [66, 67, 68] to implicitly derive $\nabla_{\tilde{x}_t} x_t$ and $\nabla_{\text{cost}(x_{1:t-1})} x_t$ in the next proposition.

Proposition D.1 (Gradients by KKT conditions)**.** *Let $x_t \in \mathcal{X}$ and $\mu \geq 0$ be the primal and dual solutions to the problem (1), respectively. The gradients of x_t with respect to $\tilde{x}_t$ and cost$(x_{1:t-1})$ are*

$$\nabla_{\tilde{x}_t} x_t = \Delta_{11}^{-1}[I + \Delta_{12} Sc(\Delta, \Delta_{11})^{-1}\Delta_{21}\Delta_{11}^{-1}],$$
$$\nabla_{\textit{cost}(x_{1:t-1})} x_t = \Delta_{11}^{-1}\Delta_{12} Sc(\Delta, \Delta_{11})^{-1}\mu,$$

where $\Delta_{11} = I + \mu\Big(\nabla_{x_t,x_t} f(x_t, y_t) + \Big(1 + (1+\frac{1}{\lambda_0})(L_1^2 + L_1)\Big) I\Big)$, $\Delta_{12} = \nabla_{x_t} f(x_t, y_t) + (x_t - \delta(x_{t-1})) + \Big(1 + (1+\frac{1}{\lambda_0})(L_1^2 + L_1)\Big)(x_t - x_t^\pi)$, $\Delta_{21} = \mu\Delta_{12}^\top$, $\Delta_{22} = f(x_t, y_t) + d(x_t, x_{t-1}) + G(x_t, x_t^\pi) + \textit{cost}(x_{1:t-1}) - (1+\lambda)\textit{cost}(x_{1:t}^\pi)$, and $Sc(\Delta, \Delta_{11}) = \Delta_{22} - \Delta_{21}\Delta_{11}^{-1}\Delta_{12}$ is the Schur-complement of Δ_{11} in the blocked matrix $\Delta = \big[[\Delta_{11}, \Delta_{12}], [\Delta_{21}, \Delta_{22}]\big]$.

If the ML prediction $\tilde{x}_t$ happens to lie on the boundary such that the inequality in (1) becomes an equality for $x = \tilde{x}_t$, then the gradient does not exist in this case and $Sc(\Delta, \Delta_{11})$ may not be full-rank. Nonetheless, we can still calculate the pseudo-inverse of $Sc(\Delta, \Delta_{11})$ and use Proposition D.1 to calculate the subgradient. Such approximation is actually a common practice to address non-differentiable points for training ML models, e.g., using 0 as the subgradient of $ReLu(\cdot)$ at the zero point [64].

Reference

[48] W. Tang, S. Bi, and Y. Zhang. Online coordinated charging decision algorithm for electric vehicles without future information. *IEEE Trans. Smart Grid*, 5:2810–2824, May 2014.

[49] Weici Pan, Guanya Shi, Yiheng Lin, and Adam Wierman. Online optimization with feedback delay and nonlinear switching cost. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(1), feb 2022.

[50] Tongxin Li, Ruixiao Yang, Guannan Qu, Yiheng Lin, Steven Low, and Adam Wierman. Equipping black-box policies with model-based advice for stable nonlinear control. In *https://arxiv.org/abs/2206.01341*, 2022.

[51] Chris Develder, Nasrin Sadeghianpourhamami, Matthias Strobbe, and Nazir Refa. Quantifying flexibility in ev charging as dr potential: Analysis of two real-world data sets. In *2016 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pages 600–605. IEEE, 2016.

[52] Mohammad Ali Alomrani, Reza Moravej, and Elias B. Khalil. Deep policies for online bipartite matching: A reinforcement learning approach. *CoRR*, abs/2109.10380, 2021.

[53] Bingqian Du, Zhiyi Huang, and Chuan Wu. Adversarial deep learning for online resource allocation. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 6(4), feb 2022.

[54] Gautam Goel, Yiheng Lin, Haoyuan Sun, and Adam Wierman. Beyond online balanced descent: An optimal algorithm for smoothed online optimization. In *NeurIPS*, volume 32, 2019.

[55] Pengfei Li, Jianyi Yang, and Shaolei Ren. Expert-calibrated learning for online optimization with switching costs. *Proc. ACM Meas. Anal. Comput. Syst.*, 6(2), Jun 2022.

[56] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *ICML*, 2020.

[57] Nicolas Christianson, Tinashe Handina, and Adam Wierman. Chasing convex bodies and functions with black-box advice. In *COLT*, 2022.

[58] Daan Rutten, Nico Christianson, Debankur Mukherjee, and Adam Wierman. Online optimization with untrusted predictions. *CoRR*, abs/2202.03519, 2022.

[59] Guanya Shi, Yiheng Lin, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online optimization with memory and competitive control. In *NeurIPS*, volume 33. Curran Associates, Inc., 2020.

[60] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[61] Martin Wainwright. Rademacher and empirical covering. `https://people.eecs.berkeley.edu/~wainwrig/stat241b/lec20.pdf`, 2009.

[62] Constantine Caramanis and Sujay Sanghavi. Projection onto a convex set. `https://users.ece.utexas.edu/~cmcaram/EE381V_2012F/Lecture_3_Scribe_Notes.final.pdf`, 2012.

[63] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. *Advances in neural information processing systems*, 30, 2017.

[64] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[65] Hyung Ju Suh, Max Simchowitz, Kaiqing Zhang, and Russ Tedrake. Do differentiable simulators give better policy gradients? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 20668–20696. PMLR, 17–23 Jul 2022.

[66] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[67] Brandon Amos. Tutorial on amortized optimization for learning to optimize over continuous domains. *CoRR*, abs/2202.00665, 2022.

[68] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.