Learning-based MPC using Differentiable Optimisation Layers for Microgrid Energy Management

Vittorio Casagrande and Francesca Boem

Abstract— **In this paper we present a learning-based Model Predictive Control (MPC) algorithm based on differentiable optimisation layers. Recent works show that it is possible to include an optimisation problem as a network layer in a Neural Network (NN) architecture. Here the MPC optimisation problem is integrated on the last layer of a NN which is used to estimate the uncertain parameters of the objective function. The NN is then trained online, end-to-end (E2E), based on previous control actions performance. We show that directly targeting the optimality of the control actions leads to improved control results with respect to the standard method of estimating the uncertain parameters and then perform the optimisation. The effectiveness of the proposed method is illustrated on a microgrid energy management problem where the future profile of the electricity price is not known.**

I. INTRODUCTION

Many control applications require to compute control actions by solving partially-defined optimisation problems that depend on unknown parameters that can be estimated from data. A microgrid Energy Management System (EMS) is a representative example of this problem, where the controller has to plan the amount of energy to store in the storage systems at each time step based on predictions of unknown variables such as electricity price. The standard procedure to solve these control problems is to first estimate the unknown parameters and then provide the solution of the estimation to the subsequent optimisation problem that computes the control actions, according to the so-called Predict then Optimise framework [12]. However, in recent years, performance-based network training has proven to outperform the Predict then Optimise approach [9, 24]. In this setting the parameters of the network are adjusted aiming at optimising the ultimate criteria on which the model is evaluated instead of targeting the estimation performance [21]. In this paper we propose a performance-based self-tuning MPC algorithm that relies on differentiable optimisation layers. Recent works have shown that it is possible to include a parametrised convex optimisation problem as a layer of a NN by implicitly differentiating its optimality conditions [2]. The network is then trained E2E updating its parameters based on the optimality of the decisions. The novel idea in this paper is to include the MPC optimisation problem in the learning framework to allow the real-time operation. The resulting algorithm combines the advantages of MPC and NNs. On the one hand, MPC is a very successful control technique thanks to its ability to compensate for uncertainty and to handle constraints. On the other hand, using deep NNs for estimation of unknown parameters, allows to choose a model structure that has the capability of approximating functions with arbitrary accuracy [23].

In the literature the performance of MPC has been enhanced in many different ways taking advantage of machine learning techniques, by learning or improving the prediction model [26], or by obtaining explicit MPC laws to reduce the computation time [11, 27]. Other works propose learning techniques to adapt the parameters of the MPC optimisation problem to improve its performance: in [15, 25] bayesian optimisation is used to optimise an LQR controller (hence without enforcing constraints on states and inputs); In [5] the best linear model that approximates the nonlinear dynamics of a robotic system is chosen using bayesian optimisation in order to maximise the controller performance. Reinforcement Learning is used in [30, 16] to tune weights and constraints of the MPC. In [20] the differential predictive control method presented in [10] is used to learn explicit NN policies for the economic dispatch problem in order to speed computation time up. On the other hand, in this paper, a NN is used to online adapt the unknown optimisation parameters of the MPC objective and such NN is trained in real time to maximise the controller performance. The algorithm proposed in this paper is based on the work on differentiable optimisation layers presented in [2]. Differentiating through the solution of a convex optimisation problem with respect to its parameters allows to perform backpropagation through convex programs. This can be used to learn convex optimisation control policies [1]. In [4] a self-tuning MPC policy is designed based on differentiable optimisation layers, where the model is trained minimizing an imitation loss, hence assuming that there is an expert system running alongside. As opposed to these contributions, we propose an algorithm which does not require an expert, instead online learns from the optimality of its past control actions and hence it is suitable for real-time implementation. In this paper we adopt the proposed method to optimise the microgrid operation. To do this, the EMS requires the predictions of variables like the energy demand, the renewable power production and the electricity price. In [28] least-square SVM regression is used to forecast load demand and renewable power production; in [18] the day-ahead solar energy is predicted by using Monte Carlo simulation; in [17] a seasonal auto-regressive integrated moving average model is used for wind and load forecasting. Task-based E2E learning has been adopted in [9] where the battery operation is scheduled with price

This work has been supported by Engineering and Physical Sciences Research Council (grant reference: EP/R513143/1, EP/W024411/1).

V. Casagrande and F. Boem are with Dept. of Electronic and Electrical Engineering, University College London, UK. Email: {vittorio.casagrande.19, f.boem} @ucl.ac.uk.

uncertainty and sub-optimal performance is obtained when a NN model is trained by minimising the estimation error instead of the energy cost. However, the NN proposed in [9] is trained on a large dataset and it is not suitable for online application. In contrast, we propose a method to overcome this limitation, thus allowing an online implementation of the E2E performance-based approach, where a training step is performed at each sample time of the controller, taking advantage of newly collected data. To the best of the Author's knowledge, a performance-based learning approach has never been used online to improve energy management performance.

Summing up, in this paper we propose a learning-based MPC algorithm taking advantage of differentiable optimisation layers and we tailor it for the microgrid energy management problem. The controller is embedded in a NN to estimate the optimisation objective parameters, while optimising the ultimate control performance. To do this we introduce a novel training algorithm and define a suitable loss function. We compare the simulation results obtained by minimising the performance-based loss function with the results obtained by minimising a standard MSE loss function and an hybrid loss function.

The remainder of the paper is structured as follows. Section II gives a background on differentiable optimisation layers and introduces the proposed method. Section III describes the microgrid model and characterizes the algorithm for the specific energy management case study, Section IV reports the simulation results and Section V draws the conclusions.

Notation. We use the subscript to denote time instants, i.e. v_t is the vector v at time t. We denote $v_{k|t}$ as the value of the variable v, k steps ahead of time step t based on the information available at time t. The estimation of the variable v, k steps ahead of time step t (i.e. at $t+k$) is denoted as $\hat{v}_{k|t}$. We use bold variables to denote time sequences of N samples, namely $\mathbf{v}_{N|t} = \{v_{k|t}\}_{k\in\{0,\dots,N-1\}}$ is the sequence of values of v computed at time t for the next N steps.

II. DIFFERENTIABLE LEARNING-BASED MPC

In this section we give a background on the performance-based framework and describe the proposed algorithm.

A. Background on performance-based learning

In the performance-based framework, machine learning methods are used to improve the decisions computed through the solution of an optimisation problem which is defined by parameters that are estimated from data. In particular, we consider a constrained optimisation problem it the form:

$$\min_{\xi} \quad f(\xi,\omega) \tag{1a}$$

$$\textit{s.t.} \quad \xi \in C(\omega) \tag{1b}$$

where ξ is the optimisation variable, f is the objective function, ω is a parameter vector and C is the constraint set. The optimisation problem is integrated in the NN as the last layer of the network. We assume to have a dataset of N data points $\mathcal{X} = \{\psi_i, \omega_i\}_{i=1}^N$, where ψ_i is a feature tensor and ω_i is the associated target output. The goal is to use supervised learning to estimate ω such that we compute the optimal solution $\xi^*(\hat{\omega})$ that best matches the targeted optimal solution $\xi^*(\omega)$ ideally obtained with the actual value of the unknown parameter. This framework replaces the two stage approach in which firstly the parameter vector $\hat{\omega}$ is estimated from data ψ_i (minimizing a conventional loss function, e.g. MSE), then such estimation is used to compute the solution of the optimisation problem. In this case the accuracy of the solution of the optimisation problem is not taken into account during training, resulting in a sub-optimal solution. Instead, integrating the optimisation problem as a network layer allows to perform the training minimizing the ultimate objective by introducing a loss function often denoted as *regret* [21, 24]:

$$\mathcal{L}(\hat{\omega},\omega) = f_{\hat{\omega}}(\xi^*(\hat{\omega})) - f_{\omega}(\xi^*(\omega)). \tag{2}$$

Fig. 1 shows the architecture of the resulting network. The

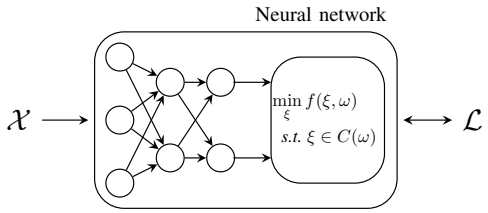

Fig. 1. End-to-End Predict and Optimise algorithm framework.

first challenge to implement this method is the computation of the gradients required for the backpropagation algorithm which implies the differentiation through the *argmin* function. Solution to this problem are provided in the literature for quadratic programming [3, 9], linear programming [12] and combinatorial problems [14]. The second challenge is related to the real-time implementation for predictive control applications. In particular, the future values of the unknown parameter ω in the future, required to compute the cost in the MPC horizon, are not known in advance, hence it is not possible to calculate the second term of (2). In this paper we deal with the second problem and we present the proposed algorithm in the following subsection.

B. Proposed methodology

We consider a scheduling problem solved by an MPC algorithm. The MPC problem is formulated as follows:

$$\min_{\mathbf{u}_{T|t}} \quad \sum_{k=0}^{T-1} J(u_{k|t}, \hat{\omega}_{k|t}) \tag{3a}$$

$$\text{s.t.} \quad x_{k+1|t} = Ax_{k|t} + Bu_{k|t} \tag{3b}$$

$$x_{k|t} \in X, u_{k|t} \in U \tag{3c}$$

$$x_{0|t} = x_t \tag{3d}$$

where T is the prediction horizon, $x_t \in \mathbb{R}^n$ is the system state, $u_t \in \mathbb{R}^m$ is the system input, $X \subset \mathbb{R}^n$ is the state constraint set, $U \subset \mathbb{R}^m$ is the input constraint set and $\hat{\omega}_{k|t} \in \mathbb{R}^p$ is the estimation of the unknown parameter

$\omega_{k|t}$. We omit here the terminal constraint since in this paper we only consider the scheduling problem and assume that the microgrid stability is guaranteed by lower level controllers. Moreover we assume that the cost function (3a) and constraints (3c) are convex hence resulting in a convex optimisation problem. Once the optimal solution $\mathbf{u}^*_{T|t}(\hat{\boldsymbol{\omega}}_{T|t})$ is found, the control law is defined as $u_t = u^*_{0|t}$. The current estimation of the unknown parameter $\hat{\boldsymbol{\omega}}_{T|t}$ over the prediction horizon T is estimated online using a NN according to the framework presented in Section II-A. We consider a NN like the one of Fig. 1, taking at each time step a feature tensor as input and computing the related optimal control action. Given a certain look-back window, we define the feature tensor $\psi_t \in \mathbb{R}^{L\times F}$ where F is the number of learning features. For example, the input feature tensor may include the past observations of the unknown parameter $\boldsymbol{\omega}_{L|t-L}$ and the past values of the system state $\mathbf{x}_{L|t-L}$. In the forward pass the NN takes the feature tensor ψ_t as input and computes the optimal control sequence $\mathbf{u}^*_{T|t}$. New data samples are collected at each time step, and enter the network training process at each algorithm iteration. We now define the loss function $\mathcal{L}$ and the training set $\mathcal{X}_t$ at time step t. Since the loss function is defined as the regret function in Eq. (2), where the functions $f_{\hat{\omega}}$ and f_ω are the MPC cost functions evaluated with the estimated and actual values of the parameter ω respectively, we can compute it only for the past time steps because the future values of ω are unknown. The loss function is defined as:

$$\mathcal{L}(\hat{\boldsymbol{\omega}}_{T|t-T}, \boldsymbol{\omega}_{T|t-T}) = \sum_{k=0}^{T-1} J(u^*_{k|t-T}, \hat{\omega}_{k|t-T}) - J(u^*_{k|t-T}, \omega_{k|t-T}) \quad (4)$$

The first term, $J(u^*_{k|t-T}, \hat{\omega}_{k|t-T})$ is the value of the objective function computed at $t-T$ using the estimation of the unknown variable. The second term $J(u^*_{k|t-T}, \omega_{k|t-T})$ is the value of the objective function computed at $t-T$ using the real values of ω. In other words, the first term is computed using the NN available at time step t, while the second term is computed solving the optimisation problem (3a)-(3d) with the past true values of the parameter ω. To compute the loss (4) we need to compute the current output of the network (first term) and the target output (second term). Hence we need a dataset composed as $\mathcal{X}_t = \{\psi_{t-T}, \boldsymbol{\omega}_{T|t-T}\}$ where ψ_{t-T} is used to compute the first term as the output of the NN and $\boldsymbol{\omega}_{T|t-T}$ is used to compute the second term by solving (3a)-(3d). Fig. 2 shows the samples required in the forward and backward pass at each time step t.

The algorithm can be divided into three steps:

1) compute the loss function (4). To do this we need: (i) the predicted sequence $\mathbf{u}^*_{T|t-T}(\hat{\omega}_{T|t-T})$ at time step $t-T$ computed using the current NN with input ψ_{t-T}; (ii) the sequence $\mathbf{u}^*_{T|t-T}(\omega_{T|t-T})$ which in theory could have been computed at $t-T$ if the future values of the unknown parameter were known. This is computed at time step t, solving the MPC optimisation problem, once the true unknown parameter has been measured.

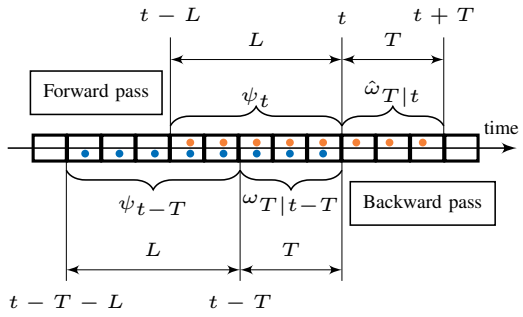

Fig. 2. Set of samples $\mathcal{X}_t$ used at each algorithm iteration. In this example the prediction horizon is 3 and the look back window length is 5. Samples marked with a red spot are used in the forward pass. Samples marked with a blue spot are used in the backward pass.

2) compute the gradients via backpropagation and update the network parameters.
3) feed the network with the feature tensor ψ_t and compute the control action.

The second step of the algorithm requires the computation of the gradient of the loss function with respect to the parameters of the network, thus involving differentiation through the optimisation layer. Since the optimisation problem (3a)-(3d) is convex in our case, the method proposed in [2] is used.

III. MICROGRID ENERGY MANAGEMENT

In this section we describe the model of the microgrid used to evaluate the proposed method in simulation.

A. Microgrid model

We consider a microgrid model similar to [8] composed of a load, a renewable generator, a connection to the utility grid and an energy storage system. The goal is to design an MPC algorithm ensuring to meet load demand while minimizing the energy cost for the microgrid. Renewable generators are characterized by a power profile $\bar{P}^r_t$, i.e. the power that is produced by the renewable generator at each time step. The load is characterized by a profile $\bar{P}^l_t$ that is the power demand at each time step t. We model the storage system, similarly to [29, 28] as a first-order linear system:

$$s_{t+1} = (1-\sigma)s_t + \eta T_s P^s_t \quad (5)$$

where s_t is the state of charge of the storage system, $\sigma \in [0,1]$ is the self-discharge rate of the battery, $\eta \in [0,1]$ is the energy conversion efficiency, T_s is the sample time of the controller and P^s_t is the power exchanged between the storage and the microgrid. The power that the storage system can inject or draw from the grid is limited by its maximum value $\bar{P}^s$:

$$-\bar{P}^s \leq P^s_t \leq \bar{P}^s \quad (6)$$

and each storage capacity is limited as:

$$\underline{s} \leq s_t \leq \bar{s}. \quad (7)$$

Microgrids are connected to the utility grid and exchange a power P_t^g at each time step. To ensure power balance in the microgrid, the following constraint is enforced:

$$P_t^g = \bar{P}_t^r - \bar{P}_t^l - P_t^s. \tag{8}$$

In other words, the amount of power that is injected in the microgrid has to be equal to the amount of power that is drawn from the microgrid. In case of complex microgrid topologies power flow constraints can be enforced as in [8], however in this paper we only consider power balance constraints. Our goal is to fulfill the load demand minimising the total energy cost. To do this we take advantage of the storage system that is capable of storing energy when renewable power production is high or prices are low. The controller objective is to minimise the cost:

$$\sum_{t=0}^{\infty} -p_t P_t^g, \tag{9}$$

where p_t is the electricity price at time t and the minus sign has been introduced since power is positive when it is sold to the utility grid. Eq. (9) cannot be used as objective function since electricity price is unknown in the future time steps. In the MPC framework this cost is approximated by a finite horizon problem which can be computed online:

$$\sum_{k=0}^{T-1} -\hat{p}_{k|t} P_{k|t}^g \tag{10}$$

where $\hat{p}_{k|t}$ is the prediction of the electricity price.

B. Neural network architecture

In this paragraph we give the details of the NN employed for the microgrid energy management problem. The NN architecture is chosen depending on the unknown parameters of the optimisation problem. In our case, since we deal with price profiles we make use of LSTM networks which are suitable to deal with data sequences. The NN employed for this work is designed as follows: (i) LSTM layers; (ii) dense layer; (iii) convex optimisation layer. LSTM networks are recurrent NNs that are often used for time sequence forecasting. The last layer of the NN is a convex optimisation layer that takes the estimated future price profile as input and computes the optimal power profile for utility grid connection and the storage system. The optimisation problem (3a)-(3d) is formulated as:

$$\min_{\mathbf{P}_t^s} \quad \sum_{k=0}^{T-1} -\hat{p}_{k|t} P_{k|t}^g \tag{11a}$$

$$\text{s.t.} \quad s_{k+1|t} = (1-\sigma) s_{k|t} + \eta T_s P_{k|t}^s \tag{11b}$$

$$-\bar{P}^s \leq P_{k|t}^s \leq \bar{P}^s \tag{11c}$$

$$\underline{s} \leq s_{k|t} \leq \bar{s} \tag{11d}$$

$$P_{k|t}^g = \bar{P}_{k|t}^r - \bar{P}_{k|t}^l - P_{k|t}^s \tag{11e}$$

$$s_{0|t} = s_t \tag{11f}$$

where (11b) is the system model, (11c) and (11d) are the power and state limits, (11e) is the power balance constraint and (11f) is the current state of the storage system. Note that in constraint (11e) we use the true future values of the load and renewable generator profiles since in this work we assume that the power profiles are known in advance. Once the optimal solution $\mathbf{P}_{T,t}^{s,*}$ is found, the control law is defined as $P_t^s(\hat{\mathbf{p}}_{T,t}) = P_{0|t}^{s,*}$. The optimisation is repeated at the following time step with the new price prediction and initial condition. Moreover, at each step, before computing the control action, a training step of the network is performed.

C. Network training

By defining a suitable training set and loss function we train the NN online. In this subsection, first we introduce the loss function and then the training set that are used for the EMS. Our method employs a performance-based loss function as in Eq. (4) differently than the standard MSE loss function. The MSE loss is:

$$\mathcal{L}^{\text{MSE}} = \frac{1}{T} \sum_{k=0}^{T-1} \left[p_{k|t-T} - \hat{p}_{k|t-T} \right]^2 \tag{12}$$

and it is used as a benchmark to assess the controller performance by estimating the price profile with a NN based on this loss function and subsequently using this estimation in the optimisation. Instead, the proposed performance-based (or *task*-based) loss function is defined as:

$$\mathcal{L}^{\text{task}} = \sum_{k=0}^{T-1} \big[p_{k|t-T} P_{k|t-T}^g(\mathbf{p}_{T|t-T}) - \hat{p}_{k|t-T} P_{k|t-T}^g(\hat{\mathbf{p}}_{T|t-T}) \big], \tag{13}$$

where $P_{k|t-T}^g(\hat{\mathbf{p}}_{T|t-T})$ is the estimated power exchanged with the electricity grid given the price profile prediction $\hat{\mathbf{p}}_{T|t-T}$ predicted given the information available till time t. Similarly $P_{k|t-T}^g(\mathbf{p}_{T|t-T})$ is the power exchanged with the electricity grid computed at time t when the full sequence of the electricity price $\mathbf{p}_{T|t-T}$ is known. We now describe the training dataset $\mathcal{X}_t$. The input feature tensor ψ_t is composed of the past electricity prices and the system state:

$$\psi_t = \begin{bmatrix} \mathbf{p}_{L|t-L} & \mathbf{s}_{L|t-L} \end{bmatrix}. \tag{14}$$

The feature tensor ψ_t is used for the forward pass, whereas the tensor ψ_{t-T} is used for the training step. The optimisation parameter $\hat{\boldsymbol{\omega}}_{T|t-T}$ is the future price profile $\hat{\mathbf{p}}_{T|t-T}$.

IV. NUMERICAL RESULTS

We consider a microgrid composed of 4 components: (i) a load; (ii) a renewable generator; (iii) an energy storage system; (iv) the connection to the utility grid. Electricity price data has been downloaded from the "ENTSO-E Transparency Platform" [13, 19] and refer to the *IT-Centre-North* bidding zone in 2017. The load and renewable generator profiles refer to the site 15 of the EMSx benchmark dataset [22], provided by Schneider Electric, collecting historical observations of real microgrids in the United States and in Europe. The price profile is presented in Fig. 3. Other numerical values used for this simulation are: $T_s = 1\,\text{h}$, $T = 12$, $L = 24$, $\underline{s} = 0\,\text{kW h}$, $\bar{s} = 400\,\text{kW h}$, $\bar{P}^s = 100\,\text{kW}$, $\sigma = 0.0042$, $\eta = 0.95$.

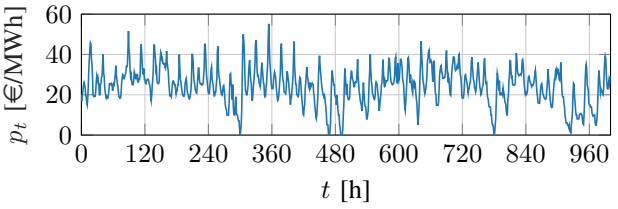

Fig. 3. Electricity price profile.

We consider three different loss functions to train the NN:

1) the MSE loss as in Eq. (12), used as benchmark;
2) the task-based loss proposed in Eq. (13);
3) a hybrid loss obtained as a weighted sum ($w_1, w_2 \in \mathbb{R}$) of the MSE loss and the task-loss:

$$\mathcal{L}^{\text{hybrid}} = w_1 \mathcal{L}^{\text{task}} + w_2 \mathcal{L}^{\text{MSE}} \tag{15}$$

The NN trained with the MSE-based loss function is composed of a LSTM layers and a dense layer. The network input feature tensor $\psi_t = \mathbf{p}_{L|t-L}$ is composed of the past electricity prices in the look back window. We compare the simulation results obtained for different combinations of the network hyperparameters: (i) hidden dimension $n_h \in \{2, 10, 20\}$; (ii) number of LSTM layers $n_l \in \{1, 2\}$; (iii) weights w_1 and w_2 of Eq. (15), $(w_1, w_2) = \{(0.1, 1), (10, 1)\}$. Clearly the MSE and task loss functions can be obtained from the hybrid loss function by setting (w_1, w_2) equal to $(0, 1)$ and $(1, 0)$, respectively. We evaluate the NNs on three performance indicators. Firstly, to describe the network complexity, we compute the memory in kilobytes (*memory cost*) required to store the NN parameters. Networks with less parameters are preferred since they require less memory for storing weights and less computational power [7]. As a second performance indicator, we compute the total energy cost of the last 20% of simulation steps (as this allows to disregard of potential initialisation or pre-training advantages) as in (9):

$$\text{Cost} = \sum_{t=800}^{1000} -p_t P_t^g \tag{16}$$

In Fig. 4 we compare the controller performance for different values of the hyperparameters. The colour of the points is used to specify the loss function adopted for network training. We also compare the performance with respect to the "prescient" controller, that is a controller solving (11a)-(11f) with a perfect knowledge of the future price profile. This controller achieves a total cost of -9.62 €. In the x axis we represent the required memory for storing the NN weights. In the y axis we represent the total cost (16) and the MSE of the one-step ahead prediction: MSE$(p_{t+1}, \hat{p}_{1|t})$. Not surprisingly the MSE cost indicator of the NNs trained minimising the MSE loss function is lower, however the control performance in terms of energy cost of the networks trained with the task and hybrid-based loss functions is superior. We now analyse more in detail one controller for each type of loss function. We consider one of the best controller for each type (represented by squares in Fig. 4) and we run 10 different simulations. For every type of loss we choose the NN controller with hidden dimension $n_h = 2$ and number of LSTM layers $n_l = 1$. Table I contains the details on the hyperparameters, the MSE and the total cost for each controller. The controller trained with task-based loss achieves a good performance with a low variability. Using the hybrid loss allows to achieve better results in terms of control and estimation performance but the variability is higher. The performance using the MSE loss is worse than the other two methods even though a lower prediction MSE is achieved.

Loss function	w_1	w_2	MSE cost	Energy cost
MSE	0	1	74.83±17.35	32.46±5.61
task	1	0	134.93±16.25	25.27±3.80
hybrid	10	1	142.20±40.47	22.87±10.97

TABLE I
NUMERICAL RESULTS OF 10 RUNS FOR EACH CONTROLLER.

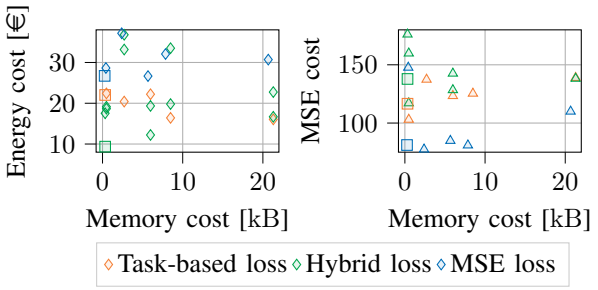

Fig. 4. Energy cost and MSE cost results.

V. CONCLUSIONS

In this paper we presented a novel learning-based MPC algorithm and we showed its effectiveness in the microgrid energy management case study. The method is based on differentiable optimisation layers and allows to train a NN, based on past data, to improve controller's performance. Simulation results show that the proposed method allows to achieve better performance than the standard MSE-based approach. As a future work, we want to explore the algorithm capabilities in the case of model uncertainty. In [6] we expand the purpose of this work, by removing the assumption that the renewable generator and load profiles are known.

REFERENCES

[1] Akshay Agrawal, Shane Barratt, and Stephen Boyd. "Learning convex optimization models". In: *IEEE/CAA Journal of Automatica Sinica* 8.8 (2021), pp. 1355–1364.

[2] Akshay Agrawal et al. "Differentiable convex optimization layers". In: *Advances in neural information processing systems* 32 (2019).

[3] Brandon Amos and J Zico Kolter. "Optnet: Differentiable optimization as a layer in neural networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 136–145.

[4] Brandon Amos et al. “Differentiable mpc for end-to-end planning and control”. In: *Advances in neural information processing systems* 31 (2018).

[5] Somil Bansal et al. “Goal-driven dynamics learning via Bayesian optimization”. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE. 2017, pp. 5168–5173.

[6] Vittorio Casagrande and Francesca Boem. “A novel learn-based MPC with embedded profiles prediction for microgrid energy management”. In: *2023 IFAC World Congress*. IEEE. 2021.

[7] Vittorio Casagrande et al. “Machine learning for computationally efficient electrical loads estimation in consumer washing machines”. In: *Neural Computing and Applications* 33.22 (2021), pp. 15159–15170.

[8] Vittorio Casagrande et al. “Resilient Distributed MPC Algorithm for Microgrid Energy Management under Uncertainties”. In: *2022 European Control Conference (ECC)*. IEEE. 2022, pp. 602–607.

[9] Priya L Donti, Brandon Amos, and J Zico Kolter. “Task-based end-to-end model learning in stochastic optimization”. In: *arXiv preprint arXiv:1703.04529* (2017).

[10] Ján Drgona, Aaron Tuor, and Draguna Vrabie. “Learning constrained adaptive differentiable predictive control policies with guarantees”. In: *arXiv preprint arXiv:2004.11184* (2020).

[11] Ján Drgona et al. “Differentiable predictive control: An mpc alternative for unknown nonlinear systems using constrained deep learning”. In: *arXiv preprint arXiv:2011.03699* (2020).

[12] Adam N Elmachtoub and Paul Grigas. “Smart “predict, then optimize””. In: *Management Science* (2021).

[13] ENTSO-E. *Transparency platform*. URL: `https://transparency.entsoe.eu`. (Accessed: 14.03.2022).

[14] Aaron Ferber et al. “Mipaal: Mixed integer program as a layer”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 02. 2020, pp. 1504–1511.

[15] Lukas P Fröhlich et al. “Bayesian optimization for policy search in high-dimensional systems via automatic domain selection”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2019, pp. 757–764.

[16] Sébastien Gros and Mario Zanon. “Data-driven economic nmpc using reinforcement learning”. In: *IEEE Transactions on Automatic Control* 65.2 (2019), pp. 636–648.

[17] Yi Guo et al. “Two-stage economic operation of microgrid-like electric vehicle parking deck”. In: *IEEE Transactions on Smart Grid* 7.3 (2015), pp. 1703–1712.

[18] Christian A Hans et al. “Scenario-based model predictive operation control of islanded microgrids”. In: *2015 54th IEEE conference on decision and control (CDC)* (2015), pp. 3272–3277.

[19] Lion Hirth, Jonathan Mühlenpfordt, and Marisa Bulkeley. “The ENTSO-E Transparency Platform–A review of Europe’s most ambitious electricity data platform”. In: *Applied energy* 225 (2018), pp. 1054–1067.

[20] Ethan King et al. “Koopman-based Differentiable Predictive Control for the Dynamics-Aware Economic Dispatch Problem”. In: *2022 American Control Conference (ACC)*. IEEE. 2022, pp. 2194–2201.

[21] James Kotary et al. “End-to-end constrained optimization learning: A survey”. In: *arXiv preprint arXiv:2103.16378* (2021).

[22] Adrien Le Franc et al. “EMSx: a numerical benchmark for energy management systems”. In: *Energy Systems* (2021), pp. 1–27.

[23] Lennart Ljung et al. “Deep learning and system identification”. In: *IFAC-PapersOnLine* 53.2 (2020), pp. 1175–1181.

[24] Jayanta Mandi and Tias Guns. “Interior Point Solving for LP-based prediction+ optimisation”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7272–7282.

[25] Alonso Marco et al. “Automatic LQR tuning based on Gaussian process global optimization”. In: *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE. 2016, pp. 270–277.

[26] Ali Mesbah. “Stochastic model predictive control with active uncertainty learning: A survey on dual control”. In: *Annual Reviews in Control* 45 (2018), pp. 107–117.

[27] Thomas Parisini and Riccardo Zoppoli. “A receding-horizon regulator for nonlinear systems and a neural approximation”. In: *Automatica* 31.10 (1995), pp. 1443–1451.

[28] Alessandra Parisio, Evangelos Rikos, and Luigi Glielmo. “A model predictive control approach to microgrid operation optimization”. In: *IEEE Transactions on Control Systems Technology* 22.5 (2014), pp. 1813–1827.

[29] Ionela Prodan, Enrico Zio, and Florin Stoican. “Fault tolerant predictive control design for reliable microgrid energy management under uncertainties”. In: *Energy* 91 (2015), pp. 20–34.

[30] Mario Zanon, Sébastien Gros, and Alberto Bemporad. “Practical reinforcement learning of stabilizing economic MPC”. In: *2019 18th European Control Conference (ECC)*. IEEE. 2019, pp. 2258–2263.