## Methods

Manuel Baumann, Sara Grundel, Philipp Sauerteig*, and Karl Worthmann

# Replacing distributed optimization by surrogate models in coupled microgrids

Received 05-07-2019

**Abstract:** The rapid uptake of renewable energy sources and energy storage devices are characteristic traits of the energy transition and require a transformation of the grid from uni- to bidirectional transmission. In addition to the implementation of the necessary physical changes within the grid, also sophisticated decision making plays a major role to make use of inherent flexibilities. To this end, optimization based control has demonstrated its potential on a microgrid level. In this paper, we consider a network of coupled microgrids. Using optimization necessitates several communication and optimization steps within the grid. We construct surrogate models by using Radial Basis Functions or Neural Networks to avoid this step. We prove the efficiency of the proposed methodology numerically based on real-data provided by an Australian distribution grid operator.

**Keywords:** Smart Grids, Model Predictive Control, Distributed Optimization, Surrogate Models, Bidirectional Optimization

**Zusammenfassung:** Der zunehmende Einsatz von erneuerbaren Energien sowie deren lokale Speicherung sind charakteristisch für die Energiewende in Deutschland. Die damit einhergehenden bidirektionalen Stromflüsse fordern nicht nur physische Veränderung im Energienetz, sondern auch die Entwicklung angepasster Algorithmen, um inherente Flexibilitäten nutzbar zu machen. Während sich in diesem Zusammenhang optimale Steuerung auf Microgrid-Ebene bewährt hat, betrachten wir in diesem Paper ein Netzwerk von gekoppelten Microgrids. Die Anwendung von Optimierungstechniken erfordert mehrere Kommunikations- und Optimierungsschritte innerhalb des Netzes. Wir verwenden Ersatzmodelle in Form von radialen Basisfunktionen oder neuronalen Netzen, um diesen Schritt zu vermeiden, und zeigen das Potential dieser Methode numerisch basierend auf Realweltdaten des australischen Netzbetreibers Ausgrid.

**Manuel Baumann, Sara Grundel,** Max Planck Institute for Dynamics of Complex Technical Systems, {baumann,grundel}@mpi-magdeburg.mpg.de
***Corresponding author: Philipp Sauerteig,** Technical University of Ilmenau, philipp.sauerteig@tu-ilmenau.de
**Karl Worthmann,** Technical University of Ilmenau, karl.worthmann@tu-ilmenau.de

## 1 Introduction

The share of renewable energy sources rapidly increases – also due to more and more installed devices like solar panels at household-level. Hence, households become prosumers, i.e. they do not only consume but also provide power. Therefore, energy generation and distribution has to take place in a distributed way all over the grid. In particular, energy can be transmitted bidirectionally between the grid and the prosumers, which results in a paradigm shift in the grid organization. In addition, prosumers may also possess some kind of energy storage unit in order to manipulate their power demand profiles by either charging or discharging. From the grid operator's perspective it might be beneficial that charging decisions are not made based on local information only. Instead taking into account information on the entire grid may improve the system-wide operation, e.g. by flattening the overall power demand in the grid. To this end, communication is needed. In the future, each household shall be equipped with a smart meter. Smart meters collect data and communicate with the grid operator automatically according to a pre-established algorithm. Households with smart meters are referred to as *smart homes*. A straight-forward way to control the systems such that they operate optimally with respect to the entire grid is to formulate one large optimization problem and to solve it in a centralized way, see, e.g. [1]. This technique, however, is hard to realize in practice. One of the disadvantages is that some central node needs the complete information about the grid, which is undesirable. A remedy is given by distributed optimization methods such as distributed dual ascent [2], Alternating Direction Method of Multipliers (ADMM) [3] or the recently published Augmented Lagrangian based Alternating Direction Inexact Newton (ALADIN) [4] method. These algorithms use a star-shaped communication topology, i.e. each smart home communicates only with the

grid operator and does not have to share any information with its neighbours. Nevertheless, in each iteration of those algorithms each household has to transmit specific (personal) data to the grid operator, see also [5] and [6] for an application of ADMM and ALADIN to electrical networked systems, respectively.

An alternate option to steer the power demand of local agents besides battery control is to schedule so-called controllable loads. Controllable loads can be shifted in time to avoid bottlenecks in the energy supply, see e.g. [7, 8]. There is also a large potential in the context of stochastic optimization of smart grids. For weather forecasting methods we refer to [9]. In [10] the impact of cyber attacks on a multi-agent system modelled as a random Markov process and possible strategies to deal with these are discussed. For an integration of electrical vehicles into the electricity network under uncertainties we refer to [11].

Considering the power networks mentioned so far, it is assumed that exchange of energy within the grid is possible at any time and does not cause any losses or additional costs, which might (approximately) hold for domestic nets, e.g. an individual town. In this paper, we refer to these grids as micorgrids (MGs). In [12], the concept of coupled MGs is introduced. Here, the authors show that even if each single MG is operated optimally there is still room for improvement if energy can be exchanged among coupled MGs. To this end, a second optimization problem is solved on a higher grid level in order to compute an optimal energy exchange strategy. These considerations motivate a bilevel optimization problem.

In [13], the authors propose to replace the optimization routine ADMM by a surrogate model in order to speed-up the calculation and further reduce the communication effort. To be more precise, Radial Basis Functions (RBFs) are used to approximate the input-output behaviour of ADMM within the framework of coupled MGs established in [12]. To this end, the RBFs first have to be trained on some given data set via interpolation. Then, they can be used to estimate the optimal solution of the underlying optimization problem. The authors showed the potential of the resulting approximation numerically. The main advantage of using such surrogates is that communication effort can be eliminated. See [14] for an introduction to RBFs. Another approach to learn highly nonlinear mappings is to use Artificial Neural Networks (NNs). NNs are the most popular representatives for modern artificial intelligence and often used in practice due to their success in various fields. They are able to *learn* complicated behaviour based on big data. see e.g. the survey article [15] for applications of NNs. In [16] the authors forecast loads in a power grid using NNs, whereas in [17] NNs are used in an optimal power flow framework.

In this paper, we adopt the idea of coupled microgrids established in [12] and extend the approach presented in [13]. In contrast to the latter, we propose an iterative optimization routine in order to further improve the overall performance and do not use state-of-the-art bilevel optimization techniques as discussed in the overview article [18]. Our approach conversely is application driven and does not assume the existence of an all-knowing superordinate operator. Due to its iterative structure, however, our method comes along with a strong need for communication between smart homes and grid operator. As a remedy we propose two methods in order to reduce the communication effort by substituting the optimization routine using surrogate models.

Furthermore, we include additional information in the input space of the mapping that is replaced by surrogate models. Doing so, we are able to show that indeed each input uniquely determines an (optimal) output computed by ADMM. To evaluate our approach we study the performance in a Model Predictive Control (MPC) framework, see e.g. [19] for an introduction to MPC and [20, 21] for MPC approaches in electrical networks. Our numerical simulations show that the proposed method approximately recovers the performance based on using ADMM but significantly reduces the communication burden.

The paper is structured as follows: In Section 2 we formulate a mathematical model for coupled microgrids that consists of two hierarchy levels, and introduce optimization problems corresponding to those levels. In the consecutive section, we propose an iterative scheme that required the solution of a distributed optimization problem on the lower-level which is solved using ADMM. In Section 4, we investigate the impact of disturbances w.r.t. the lower-level solution on the performance measured in terms of the upper-level objective function. Based on the results, we propose to replace ADMM by surrogates in order to reduce communication effort and computation time. The performance of the optimization scheme incorporating surrogates is analysed in a MPC framework in Section 5.

## 2 A model for coupled microgrids

We consider a system of coupled microgrids (MGs) and call it a *smart grid*. Each MG consists of several residential energy systems (agents) coupled through the grid operator, which can be seen as Central Entity (CE). The coupling

of the microgrids is done through a network, where some MGs are connected by a transmission line and others are not connected, cf. Figure 1.

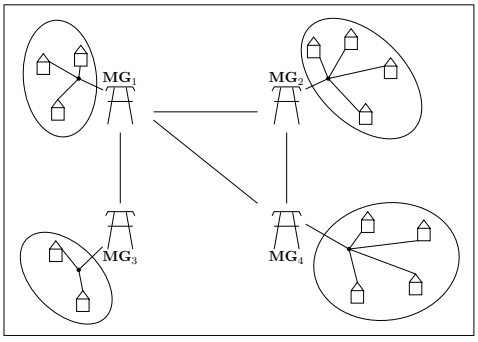


**Fig. 1:** Upper-level model: Schematic representation of four coupled MGs. Energy exchange is possible only via transmission lines.

## 2.1 Upper-level model: Energy exchange

We assume that we have $\Xi \in \mathbb{N}$ many MGs which are partially coupled via transmission lines and can be interpreted as nodes of a non-complete graph, see Figure 1 where $\Xi = 4$. Each MG $\kappa$, for $\kappa \in [1:\Xi]$, consists of $\mathcal{I}_\kappa \in \mathbb{N}$ agents modeled in detail in Subsection 2.2. We assume that each MG has an average power demand $\bar{z}_\kappa(n)$ as well as an average desired power demand $\bar{\zeta}_\kappa(n)$ at time $n$. The average power demand reflects what is needed per residential unit within each MG. Given these two quantities for every MG, we can compute the total power demand $\mathcal{I}_\kappa \bar{z}_\kappa$ and the total desired power demand $\mathcal{I}_\kappa \bar{\zeta}_\kappa$ of the MG. The overall goal is to match these two quantities by controlling the residential storage units as well as exchanging power among the MGs. Let $\delta_{\kappa\nu}$ describe the percentage of power that is transferred from MG $\kappa$ to MG $\nu$. We enforce $\delta_{\kappa\nu}$ equals zero if there is no transmission line between the two MGs. Otherwise, the power demand of a MG $\kappa$ is given by its own total power demand $\delta_{\kappa\kappa}\mathcal{I}_\kappa \bar{z}_\kappa$, where $\delta_{\kappa\kappa}$ is what remains at the MG, and the sum over the power received from connected MGs, $\sum_{\nu\neq\kappa} \delta_{\nu\kappa}\mathcal{I}_\nu \bar{z}_\nu$. For each time step in our prediction horizon of length $N \in \mathbb{N}_{\geq 2}$, we want to match this to the desired power demand of each MG in a least-squares sense. The objective function is thus given by $\mathcal{J} : \mathbb{R}^{\Xi N} \times \mathbb{R}^{\Xi\times\Xi\times N} \to \mathbb{R}$,

$$(\bar{z}, \delta) \mapsto \sum_{n=k}^{k+N-1} \sum_{\kappa=1}^{\Xi} \left( \bar{\zeta}(n)\mathcal{I}_\kappa - \sum_{\nu=1}^{\Xi} \delta_{\nu\kappa}(n)\eta_{\nu\kappa}\mathcal{I}_\nu \bar{z}_\nu(n) \right)^2. \tag{1}$$

Here, the vector $\bar{z} = (\bar{z}(k), \dots, \bar{z}(k+N-1)^\top)^\top$ with $\bar{z}(\cdot) \in \mathbb{R}^\Xi$ stacks the average power demand per MG and time step while the matrix $\eta = (\eta_{\nu\kappa})_{\kappa,\nu=1} \in [0,1]^{\Xi\times\Xi}$ incorporates efficiencies along the transmission lines.

We are interested in minimizing (1) under the following constraints: All exchange rates $\delta_{\nu\kappa}$ are within the interval $[0,1]$, sum up to 1, and only transfer power in one direction, meaning that either $\delta_{\kappa\nu}$ or $\delta_{\nu\kappa}$ are zero. Moreover, note that at this grid level, the average power demands per MG are known. Following [12], the optimization problem of the upper-level is, thus, formulated over the exchange rates $\delta$,

$$\begin{aligned} \min_{\delta\in[0,1]^{\Xi\times\Xi\times N}} \quad & \mathcal{J}(\bar{z}, \delta) && \text{(2a)} \\ \text{s.t.} \quad & \sum_{\nu=1}^{\Xi} \delta_{\nu\kappa}(n) = 1 && \text{(2b)} \\ & \delta_{\kappa\nu}(n) \cdot \delta_{\nu\kappa}(n) \leq 0, \ \kappa \neq \nu && \text{(2c)} \\ & \forall\, \kappa, \nu \in [1:\Xi],\, n \in [k : k+N-1], \end{aligned}$$

where $\delta_{\nu\kappa}(n)$ denotes the power exchange rate from MG $\nu$ to MG $\kappa$ at time instance $n$. Constraints (2b) and (2c) ensure that the whole energy of each MG is scheduled and that exchanges via transmission lines can only occur in one direction during one time step. We denote the feasible set of (2) by

$$\mathbb{D}^\delta \ = \ \left\{ \, \delta \in [0,1]^{\Xi\times\Xi\times N} \, \middle| \, \text{(2b) and (2c) hold} \, \right\}.$$

The efficiency of a transmission line does not depend on the direction of the transfer, i.e. the matrix $\eta$ is symmetric. Furthermore, we assume no loss without transport, i.e. $\eta_{\kappa\kappa} \equiv 1$ for all $\kappa \in [1:\Xi]$.

## 2.2 Lower-level model: Single microgrid

As we have seen in the previous section, we consider a desired and an average power demand at each MG. In order to understand these quantities better, we explain the modeling of the MG in all detail. The basis of our considerations forms the model presented in [22] and its extension [5]. Therefore, let each subsystem be equipped with an energy generation device (e.g. roof top photovoltaic panels) and some storage device (e.g. a battery). Then, the $i$-th system, $i \in [1:\mathcal{I}_\kappa]$ in MG $\kappa$, can be described by the discrete time system dynamics,

$$\begin{aligned} x_{\kappa_i}(n+1) \ &= \ \alpha_{\kappa_i} x_{\kappa_i}(n) + T(\beta_{\kappa_i} u^+_{\kappa_i}(n) + u^-_{\kappa_i}(n)) && \text{(3a)} \\ z_{\kappa_i}(n) \ &= \ w_{\kappa_i}(n) + u^+_{\kappa_i}(n) + \gamma_{\kappa_i} u^-_{\kappa_i}(n), && \text{(3b)} \end{aligned}$$

where $x_{\kappa_i}(n)$ and $z_{\kappa_i}(n)$ denote the State of Charge (SoC) of the battery and the power demand at time instance

$n \in \mathbb{N}_0$, respectively. The latter incorporates the net consumption, $w_{\kappa_i}(n) = \ell_{\kappa_i}(n) - g_{\kappa_i}(n)$, as the difference of load and power generation, cf. Figure 2.

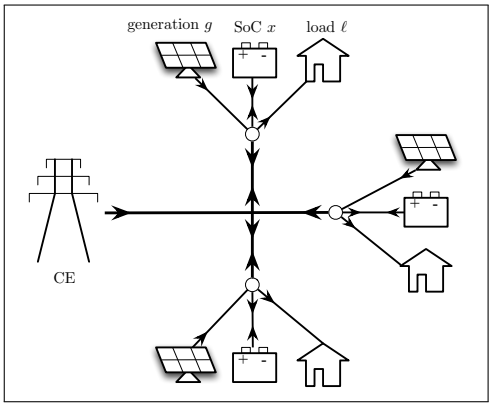


**Fig. 2:** Lower-level model: Star-shaped network of residential energy systems and central entity (CE). The quantity $w = \ell - g$ in (3b) is obtained from the data set [23].

The system can be controlled by charging $u^+_{\kappa_i}$ and discharging $u^-_{\kappa_i}$ the battery at each time step. The length of a time step in hours is denoted by $T > 0$, e.g. $T = 0.5$ corresponds to a half-hour time window. The constants $\alpha_{\kappa_i}, \beta_{\kappa_i}, \gamma_{\kappa_i} \in (0, 1]$ represent efficiencies w.r.t. self discharge and energy conversion. Furthermore, the initial SoC $x_{\kappa_i}(k) = \hat{x}_{\kappa_i}$, where $k \in \mathbb{N}_0$ denotes the current time instance, is assumed to be known. State and input are subject to the inequality constraints,

$$\begin{aligned} 0 &\leq x_{\kappa_i}(n) \leq C_{\kappa_i} && \text{(4a)}\\ \underline{u}_{\kappa_i} &\leq u^-_{\kappa_i}(n) \leq 0 && \text{(4b)}\\ 0 &\leq u^+_{\kappa_i}(n) \leq \bar{u}_{\kappa_i} && \text{(4c)}\\ 0 &\leq \frac{u^-_{\kappa_i}(n)}{\underline{u}_{\kappa_i}} + \frac{u^+_{\kappa_i}(n)}{\bar{u}_{\kappa_i}} \leq 1. && \text{(4d)} \end{aligned}$$

Here, $C_{\kappa_i} \geq 0$ denotes the battery capacity. The last constraint ensures that the bounds on discharging (4b) and charging (4c) hold even if the battery is both discharged and charged during one time step. Note that the case $C_{\kappa_i} = 0$ covers the case, where not all systems have a storage device. Since the future net consumption is not known in advance, it is assumed to be reliably predictable on a short time horizon of size $N$, $N \in \mathbb{N}_{\geq 2}$, time steps.

For a concise notation we introduce the set $\mathbb{X}_{\kappa_i} = [0, C_{\kappa_i}]$ of feasible states, the set $\mathbb{U}_{\kappa_i} = \left\{ (u^+_{\kappa_i}, u^-_{\kappa_i})^\top \in \mathbb{R}^2 \;\middle|\; \text{(4b)} - \text{(4d) hold} \right\}$ of feasible control pairs and the set

$$\mathbb{D}_{\kappa_i} = \left\{ z_{\kappa_i} \in \mathbb{R}^N \;\middle|\; \begin{array}{c} z_{\kappa_i} = (z_{\kappa_i}(k), \ldots, z_{\kappa_i}(k+N-1))^\top \\ \exists\, u_{\kappa_i} \in \mathbb{U}^N \text{such that} \\ x_{\kappa_i}(k) = \hat{x}_{\kappa_i}, \text{(3) and (4) hold} \end{array} \right\}$$

of feasible outputs over the next $N$ time steps, $i \in [1 : \mathcal{I}_\kappa]$, $\kappa \in [1 : \Xi]$. Referring to the feasible sets of a MG $\kappa$ we use the Cartesian product, e.g. $\mathbb{D}^{(\kappa)} = \mathbb{D}_{\kappa_1} \times \ldots \times \mathbb{D}_{\kappa_{\mathcal{I}_\kappa}}$ and $z^{(\kappa)} \in \mathbb{D}^{(\kappa)}$.

The output quantity in (3b) is the power demand $z_{\kappa_i}$ of an individual agent in MG $\kappa$. The average power demand $\bar{z}_\kappa = \frac{1}{\mathcal{I}_\kappa} \sum_{i=1}^{\mathcal{I}_\kappa} z_{\kappa_i}$ in each MG can then be computed from the individual power demands, and is used as an input to (1). One way to define $\bar{\zeta}_\kappa$ independent of $\kappa$ in (1) is to create a stable reference trajectory by averaging over a past time horizon and over all individual residential units of all microgrids, the so-called overall average net consumption as proposed in [5],

$$\bar{\zeta}(n) = \frac{1}{\mathcal{I} \cdot \min\{N, n+1\}} \sum_{j=n-\min\{n,N-1\}}^{n} \sum_{i=1}^{\mathcal{I}} w_i(j) \tag{5}$$

where $\mathcal{I} = \sum_{\kappa=1}^{\Xi} \mathcal{I}_\kappa$ denotes the total number of agents within the entire smart grid.

Let us for the moment ignore the coupling described in Subsection 2.1. Then, $\delta$ equals the identity, and the objective function $g : \mathbb{R}^N \to \mathbb{R}_{\geq 0}$,

$$\min_{z^{(\kappa)} \in \mathbb{D}^{(\kappa)}} \quad g(\bar{z}_\kappa) = \left\| \bar{\zeta} - \bar{z}_\kappa \right\|_2^2, \tag{6}$$

yields a lower-level optimization problem for MG $\kappa$ that at the same time minimizes (1).

## 2.3 Fully coupled optimization problem

We are interested in optimizing the function (1). This function, in general, depends on $\delta$ as well as on $\bar{z}$. As seen in the previous section, the average power demand $\bar{z} = \bar{z}(u)$ depends on the control $u$, which we have to find in such a way that $\mathcal{J}$ is optimal. The overall optimization problem can be written as

$$\min_{\delta \in \mathbb{D}^\delta, \bar{z} \in \mathbb{D}^{(\kappa)}} \quad \mathcal{J}(\bar{z}, \delta). \tag{7}$$

Note that due to constraint (2c) the optimization of $\mathcal{J}$ w.r.t. $\delta$ is non-convex. Furthermore, the large scale of the optimization w.r.t. $\bar{z}$ causes the use of a centralized solver to be expensive. In addition, using a centralized solver assumes the existence of a global entity gathering the

information of the whole grid, in particular the personal data of each household, which is undesirable in practice. Hence, solving (7) centralized is impractical. In the subsequent section we present an approach to tackle (7) by solving the upper and lower-level problem iteratively. Doing so we avoid a node with full knowledge in the grid and only communicate specific aggregated information in each iteration among the agents.

# 3 Bidirectional optimization

We propose to tackle the optimization problem (7) in a *bidirectional* way, i.e. we first find an optimal $\bar{z}$ for $\delta$ being the identity and then optimize (7) w.r.t. $\delta$ for fixed $\bar{z}$ in order to find the optimal exchange strategy. This typically already gives a considerable improvement, and has been also done e.g. in [12]. To solve (2) for a fixed $\bar{z}$ is straight forward, and we use a standard Sequential Quadratic Programming (SQP) solver. We refer to [24] for an introduction to SQP methods. In this paper we show how to incorporate the computed energy exchange from the upper level into the lower-level optimization problem in order to improve the overall performance.

## 3.1 Bidirectional optimization scheme

Assume that each MG $\kappa$, $\kappa \in [1:\Xi]$, within the smart grid has already solved its inherent optimization problem (6) and based on the corresponding solutions $\bar{z}_\kappa$ an energy exchange policy $\delta^\star$ has been computed according to (2). This exchange yields an updated power demand

$$\bar{z}_\kappa^+(n) = \frac{1}{\mathcal{I}_\kappa} \sum_{\nu=1}^{\Xi} \delta^\star_{\nu\kappa}(n)\, \eta_{\nu\kappa}\, \bar{z}_\nu(n)\, \mathcal{I}_\nu,\ n \in [k:k+N-1], \tag{8}$$

and hence, the difference $\Delta\bar{z}_\kappa(n) = \bar{z}_\kappa(n) - \bar{z}_\kappa^+(n)$ in power demand for all MGs. We are interested in updating $\bar{z}$ in such a way that our cost function (1),

$$\begin{aligned}\mathcal{J}(\bar{z},\delta^\star) &= \sum_{n=k}^{k+N-1} \sum_{\kappa=1}^{\Xi} \left( \bar{\zeta}(n)\,\mathcal{I}_\kappa - \sum_{\nu=1}^{\Xi} \delta^\star_{\nu\kappa}(n)\,\eta_{\nu\kappa}\,\bar{z}_\nu(n)\,\mathcal{I}_\nu \right)^2 \\ &= \sum_{\kappa=1}^{\Xi} \mathcal{I}_\kappa^2 \left\| \bar{\zeta} - \bar{z}_\kappa^+ \right\|_2^2 = \sum_{\kappa=1}^{\Xi} \mathcal{I}_\kappa^2 \left\| \bar{\zeta} + \Delta\bar{z}_\kappa - \bar{z}_\kappa \right\|_2^2\end{aligned}$$

is minimized further. One could think about fixing $\delta$ and finding an optimal $\bar{z}$. This, however, leads to an optimization problem not avoiding communication and coupling all microgrids. The trick here is now to fix not only the $\delta$ but also the $\bar{z}$-components from all the microgrids but one. This leads to optimizing $\mathcal{I}_\kappa^2 \left\| \bar{\zeta} + \Delta\bar{z}_\kappa - \bar{z}_\kappa \right\|_2^2$ locally in each MG, where $\Delta\bar{z}_\kappa$ is computed by using the $\bar{z}_\kappa$'s and $\delta$'s from the previous optimization step. Intuitively, the difference $\Delta\bar{z}_\kappa$, $\kappa \in [1:\Xi]$, can be interpreted as an additional load or generation, and, therefore, as a change of the desired power demand profile for MG $\kappa$. This yields the modified lower-level optimization problem

$$\min_{z^{(\kappa)} \in \mathbb{D}^{(\kappa)}} \quad g_\kappa(\bar{z}_\kappa) = \left\| \bar{\zeta}_\kappa^+ - \bar{z}_\kappa \right\|_2^2, \tag{9}$$

where $\bar{\zeta}_\kappa^+ = \bar{\zeta} + \Delta\bar{z}_\kappa$ In this formulation the updated reference trajectories $\bar{\zeta}_\kappa^+$, $\kappa \in [1:\Xi]$, differ among the single MGs and depend on a given $\delta$ and a given previous $\bar{z}_\kappa$.

Based on the updated reference value we solve (9) and (2) to improve the battery usage and the energy exchange and repeat the optimization until some terminal condition is satisfied, e.g. performance improvement less than a pre-defined tolerance or maximal number of iterations exceeded. These considerations are summarized in Algorithm 1. Note that we only update the reference $\bar{\zeta}$ on the lower-level, since the upper-level optimization problem (1) does not change.

**Algorithm 1** Iterative bidirectional optimization scheme

**Input**: Current time instance $k \in \mathbb{N}_0$, current SoC $x_{\kappa_i}(k) \in \mathbb{X}_{\kappa_i}$, prediction horizon $N \in \mathbb{N}_{\geq 2}$, predicted net consumption $(w_{i_\kappa}(k), \dots, w_{i_\kappa}(k+N-1))^\top \in \mathbb{R}^N$, $i_\kappa \in [1:\mathcal{I}_\kappa]$, $\kappa \in [1:\Xi]$, reference trajectory $(\bar{\zeta}(k), \dots, \bar{\zeta}(k+N-1))^\top \in \mathbb{R}^N$, maximal number $j_{\max} \in \mathbb{N}$ of iterations, and tolerance $\varepsilon > 0$.

**Initialization**:

1. Set $j = 0$ and $\delta^0(n) = I_\Xi$ for all $n \in [k:k+N-1]$.
2. *Lower level (parallel in $\kappa$).* Compute $\bar{z}_\kappa^j$ as the solution of (6) using ADMM.
3. *Upper level.* Given $\bar{z}_\kappa^j$, solve (2) for $\delta^j$ using SQP.

**While** $j < j_{\max}$ and $\mathcal{J}(\bar{z}^{j-1}, \delta^{j-1}) - \mathcal{J}(\bar{z}^j, \delta^j) > \varepsilon$ **Do**:

4. *Lower level (parallel in $\kappa$).*
   (a) Compute $\bar{\zeta}_\kappa^+$ from $\bar{z}_\kappa^j$ and $\delta^j$.
   (b) Solve (9) using ADMM and send $\bar{z}_\kappa^{j+1}$ to the upper level.
5. *Upper level.* Given $\bar{z}_\kappa^{j+1}$, solve (2) for $\delta^{j+1}$ using SQP.
6. $j \to j+1$

Neither convergence nor the interpretation of a potential limit of Algorithm 1 is clear a priori. Figure 3, however, experimentally shows convergence of the proposed

scheme and a continuous improvement of the upper-level performance index. Here, we ran 10 iterations of the optimization scheme and plotted both the objective function values before and after the energy exchange within each iteration. The values stagnate after 4 iterations indicating that additional iterations do not further improve the overall performance. The next subsection elaborates on how to solve (9) in a fully distributed way using ADMM.

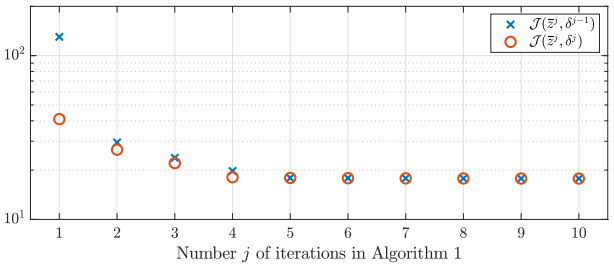


**Fig. 3:** Evolution of the costs computed according to the bidirectional optimization scheme described in Algorithm 1.

## 3.2 Distributed optimization via ADMM

In this section we briefly discuss how to solve the lower-level optimization problem (6) or (9) using an Alternating Direction Method of Multipliers (ADMM) approach. We consider a single MG and, therefore, ommit the index $\kappa$. Since the averaged output quantity appears in the objective function (6) or (9), we need to introduce an auxiliary variable $a$ in order to decouple the lower-level optimization in the following way,

$$\min_{z,a} \quad g(\bar{a}) = \|\bar{a} - \bar{\zeta}\|_2^2 \tag{10a}$$

$$\text{s.t.} \quad \frac{1}{\mathcal{I}} \sum_{i=1}^{\mathcal{I}} a_i - \bar{a} = 0, \quad z_i - a_i = 0 \tag{10b}$$

$$z_i \in \mathbb{D}_i \quad \forall\, i \in [1:\mathcal{I}]. \tag{10c}$$

Note that (10c) is a short-hand notation for the battery dynamics (3)-(4), and yields a fully decoupled constraint in the variable $z$. ADMM is an optimization scheme to solve (10) based on the augmented Lagrangian $\mathcal{L}_\rho : \mathbb{R}^{\mathcal{I}N} \times \mathbb{R}^{\mathcal{I}N} \times \mathbb{R}^{\mathcal{I}N} \to \mathbb{R}$, for $\rho > 0$,

$$\mathcal{L}_\rho(z, a, \lambda) = g(\bar{a}) + \sum_{i=1}^{\mathcal{I}} \left( \lambda_i^\top (z_i - a_i) + \frac{\rho}{2} \|z_i - a_i\|_2^2 \right),$$

in a distributed way, cf. [3]. Following [5], the ADMM algorithm for (10) yields the three-step iteration $\ell \mapsto \ell+1$,

$$z_i^{\ell+1} = \arg\min_{z_i \in \mathbb{D}_i} z_i^\top \lambda_i^\ell + \frac{\rho}{2} \left\| z_i - a_i^\ell \right\|_2^2 \tag{11a}$$

$$a^{\ell+1} = \arg\min_{a \in \mathbb{R}^{\mathcal{I}N}} g(\bar{a}) - \sum_{i=1}^{\mathcal{I}} a_i^\top \lambda_i^\ell - \frac{\rho}{2} \|z_i^{\ell+1} - a_i\|_2^2 \tag{11b}$$

$$\lambda_i^{\ell+1} = \lambda_i^\ell + \rho(z_i^{\ell+1} - a_i^{\ell+1}) \tag{11c}$$

until some termination condition is satisfied. Note that (11b) is an unconstrained optimization problem and can be solved explicitly. The problem (11a) can be solved in parallel by each battery in the MG introduced for our model in Section 2.2. Note that scheme (11) assumes communication within the MG, more precisely, each system $i$, $i \in [1:\mathcal{I}]$, sends its optimal solution $z_i$ to the CE and receives both the updated auxiliary $a_i$ and dual variable $\lambda_i$ in return. The variant discussed in [5] avoids unnecessary communication overhead by returning a broadcast variable instead, which only incorporates information on the aggregated values.

According to Theorem 3.1 in [5] the optimization scheme (11) converges in the following sense.

**Theorem 1.** Consider Problem (10) with $g$ being strictly convex, closed and proper, and assume there exists a saddle point $(z^\star, a^\star, \lambda^\star)$ of the Lagrangian $\mathcal{L}_0$, i.e.

$$\mathcal{L}_0(z^\star, a^\star, \lambda) \;\le\; \mathcal{L}_0(z^\star, a^\star, \lambda^\star) \;\le\; \mathcal{L}_0(z, a, \lambda^\star)$$

for all $(z, a) \in \mathbb{R}^{\mathcal{I}N} \times \mathbb{R}^{\mathcal{I}N}$ and $\lambda \in \mathbb{R}^{\mathcal{I}N}$. Furthermore, let the iterates $(z^\ell, a^\ell, \lambda^\ell)$ be computed according to (11). Then the following following statements hold true:

1. $(z^\ell - a^\ell)_{\ell \in \mathbb{N}_0}$ converges to zero for $\ell \to \infty$,
2. $(g(\bar{a}^\ell))_{\ell \in \mathbb{N}_0}$ converges to the optimal value $g^\star$ of (10),
3. $(\lambda^\ell)_{\ell \in \mathbb{N}_0}$ converges to the dual optimal $\lambda^\star$ of (10).

According to [5] and the references [3, Section 3] and [2, Appendix C] therein, problem (6) fulfils the assumptions of Theorem 1.

# 4 Surrogate models for ADMM

Due to the distributive structure of ADMM, the residential energy systems do not need to share information with their neighbours but only with the CE of the corresponding MG, see also the star-structure in Figure 2. In each iteration $\ell$ of ADMM, subsystem $i_\kappa$ of MG $\kappa$ has to transmit its solution $z_{\kappa_i}^\ell$ of (11a) to the CE of MG $\kappa$. The optimization scheme presented in Algorithm 1, however, suggests to run ADMM more than once in order to improve the performance w.r.t. to the objective function (1). In order to avoid unnecessary

communication, we propose to use surrogate models to approximate the optimization routine (11). More precisely, we are interested in a function $\varphi_\kappa : \mathbb{R}^N \times \mathbb{R}^{\mathcal{I}_\kappa} \times \mathbb{R}^N \to \mathbb{R}^N$ which approximates

$$\varphi_\kappa(\bar{w}_\kappa, x_\kappa(k), \bar{\zeta}) \approx \bar{z}_\kappa, \tag{12}$$

for all feasible $(\bar{w}_\kappa, x_\kappa(k), \bar{\zeta}) \in \mathbb{R}^N \times \mathbb{X}^{(\kappa)} \times \mathbb{R}^N$, $\kappa \in [1 : \Xi]$.

**Remark 1.** Note that $\bar{\zeta} \mapsto \bar{z}_\kappa$ as introduced in [13] does not define a mapping since the solution $\bar{z}_\kappa$ of (6) not only depends on the reference value $\bar{\zeta}$ but also on the future net consumption $\bar{w}_\kappa$ and the current SoC $x_{i_\kappa}(k)$.

Figure 4 (top) shows that if the approximation (12) is sufficiently accurate, the impact on the performance of the optimization scheme is negligible. Here, the costs $\mathcal{J}(\bar{z}, \delta)$

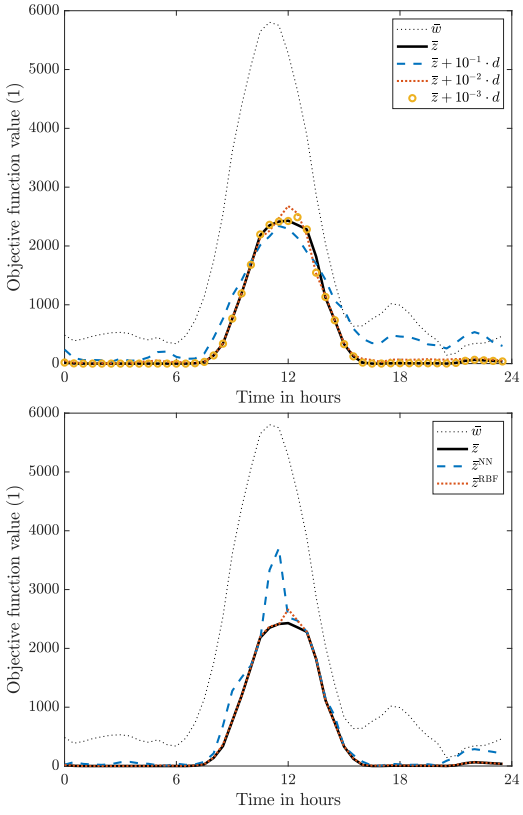


**Fig. 4:** Effect of mapping error in (12) (top) and of the approximation via radial basis functions (RBFs) and a neural network (NN) on the open-loop performance $\mathcal{J}(\bar{z}, \delta)$ within 48 consecutive time steps (bottom). We use $T = 0.5h$ in (3a).

after optimization are visualized for 48 consecutive time steps (equals 24-hours simulation time). In the experiment, we disturbed the ADMM solution in Algorithm 1 by uniformly distributed additive noise, i. e., $\bar{z}_\kappa + 10^{-p} \cdot d$, where the vector $d \sim \mathcal{U}(-1, 1)$, and $p \in \mathbb{N}_0$ denotes the intensity of the disturbance.

Note that (12) might yield approximations to the solution $\bar{z}_\kappa$ that either aggravate the performance w.r.t. (6) compared to the solution $\bar{w}$ associated with $u \equiv 0$ or solutions that correspond to an infeasible control $\hat{u} \notin \mathbb{U}$. As a remedy, we propose to apply ADMM once after replacing it by a surrogate in the optimization scheme. More precisely, first we run Algorithm 1 using a surrogate in Step 4(b) until the while loop terminates and then we additionally repeat Steps 4 and 5 using ADMM.

## 4.1 Well-posedness

The following proposition states that (12) defines a proper mapping. For a concise notation we omit the index $\kappa$ and replace the index $\kappa_i$ by $i$ here.

**Proposition 1.** Consider $\varphi$ given by (12), where $\bar{z}$ describes the optimal solution of (6) computed via ADMM, i.e. $\bar{z} = \bar{z}(u^\star)$. We assume all hyper-parameter to be fixed meaning that $\{T, \alpha_i, \beta_i, \gamma_i, C_i, \bar{u}_i, \underline{u}_i\}$ in (3)-(4) are constant over time for all $i \in [1 : \mathcal{I}]$. Then $\varphi$ is a mapping, i.e. for all $(\bar{w}, x(k), \bar{\zeta}) \in \mathbb{R}^N \times \mathbb{X} \times \mathbb{R}^N$ exists a uniquely determined $\bar{z} \in \mathbb{R}^N$ such that $\bar{z}$ is the solution to the optimization problem (6).

*Proof.* First note that ADMM yields the unique solution of (6), see e.g. [3]. Furthermore, there are no constraints on $z_i$, $i \in [1 : \mathcal{I}]$, and the future SoC can be interpreted as an affine function of the current SoC and the future (dis-)charging rate. Hence, expansion of (3a) and averaging of (3b) yield,

$$\begin{aligned}
&\min_u \quad \left\| \bar{z}(u) - \bar{\zeta} \right\|_2^2, \quad \text{subject to} \\
&x_i(k+1+n) = \alpha_i^{n+1} x_i(k) + T \sum_{\ell=k}^{k+n} \alpha_i^{n+k-\ell} \left( \beta_i u_i^+(\ell) + u_i^-(\ell) \right), \\
&x_i(k) = \hat{x}_i, \quad \text{and constraints (4)}, \\
&\bar{z}(k+n) = \bar{w}(k+n) + \bar{u}^+(k+n) + \bar{\gamma} \bar{u}^-(k+n), \\
&n \in [0 : N-1],
\end{aligned}$$

where $\bar{\cdot}$ denotes the corresponding average value w.r.t. all subsystems, in particular $\bar{z}(n) = \frac{1}{\mathcal{I}} \sum_{i=1}^{\mathcal{I}} z_i(n)$. This representation of (6) illustrates that the (predicted) average values $\bar{\zeta} = (\bar{\zeta}(k), \dots, \bar{\zeta}(k+N-1))^\top$ and $\bar{w} = (\bar{w}(k), \dots, \bar{w}(k+N-1))$ and the current SoC $\{x_i(k)\}_{i=1}^{\mathcal{I}}$, uniquely determine the optimal solution $\bar{z}(u^\star)$ obtained by ADMM. □

## 4.2 Radial basis functions approximation

Radial Basis Functions (RBFs) are used to interpolate functions based on a set of sampling data. We briefly recap some basics on RBFs. For a detailed introduction to theory and application see e.g. [14], for a similar approach where RBFs are used to replace ADMM we refer to [13].

Let $M \in \mathbb{N}$ denote the number of samples. Then, the interpolation function of (12) is given as the sum of basis functions $\psi_m : \mathbb{R}^N \times \mathbb{X}^{(\kappa)} \times \mathbb{R}^N \to \mathbb{R}$, $m \in [1:M]$, and a regularization term $q : \mathbb{R}^N \times \mathbb{X}^{(\kappa)} \times \mathbb{R}^N \to \mathbb{R}^N$. More precisely,

$$\bar{z}_\kappa \approx \varphi_\kappa^{\mathrm{RBF}}(\chi_\kappa) = \sum_{m=1}^{M} \psi_m(\chi_\kappa)\alpha_m + q(\chi_\kappa), \tag{13}$$

where $\chi_\kappa = (\bar{w}_\kappa, x_\kappa(k), \bar{\zeta})$ is the joint inputs of Proposition 1, and $\alpha_m \in \mathbb{R}^N$, $m \in [1:M]$. The basis functions are so-called radial basis functions of the form, $\psi_m(\chi_\kappa) = \psi(\|\chi_\kappa - \chi_m\|)$, where the kernel $\psi$ yields support close to the sampling data $\chi_m$, $m \in [1:M]$. We choose an affine linear regularization $q(\chi_\kappa) = \beta_0 + B\chi_\kappa$. Note that different choices are possible. The missing parameters $\alpha_m$, $\beta_0$ and $B$ are determined by interpolation conditions, cf. [13, 14].

In Figure 5, a possible fit via RBFs is visualized. Here, we interpolated given data from two-weeks of optimization (4540 data points) based on sampling data picking each 25-th data point to train (13). Then, we tested $\varphi_1^{\mathrm{RBF}}$ on the following day, and plotted the fitting. Our implementation is based on the `Matlab` toolbox `DACE` [25]. Note that the evaluation time of the RBF approximation grows with the number of data points used. Already with 180 data points to train (13) with $N = 6$ causes the function evaluation of $\varphi^{\mathrm{RBF}}$ to be expensive, see Table 1. Using more data points would no longer yield an advantage over using ADMM w.r.t. computation time.

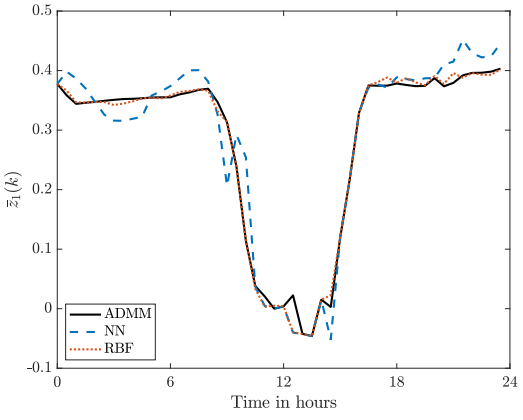


**Fig. 5:** RBF and neural net fitting of the first component (MG 1) $\bar{z}_1(k)$ of $\bar{z} = (\bar{z}(k), \ldots, \bar{z}(k+N-1))^\top$ within 48 consecutive time steps indicating the quality of the approximation (12).

## 4.3 Neural networks approximation

Neural Networks (NNs) are a state-of-the-art method in artificial intelligence frameworks. Based on huge amounts of data $M \gg 1$ they are able to learn and recognize patterns in complex systems. We consider a NN of $l$-layers as an approximation to the mapping (12), i.e.,

$$\bar{z}_\kappa \approx \varphi_\kappa^{\mathrm{NN}}(\chi_\kappa) = \sigma\left(W^{[l]} \ldots \sigma(W^{[2]}\chi_\kappa + b^{[2]}) \ldots + b^{[l]}\right), \tag{14}$$

where $\sigma$ denotes the sigmoid function, and the weights $W^{[l]}$ and biases $b^{[l]}$ are determined during the training phase. Here, the number of neurons at layer $l-1$ and at layer $l$ determine the number of rows and columns of $W^{[l]}$, respectively. Note that a separate neural network is trained for each MG. For an introduction to deep learning and neural networks we refer the reader to [26, 27]. To train the NNs, we used `Matlab`'s built-in toolbox `nftool`.

The overall goal of the approximation (14) is to be sufficient in the sense of the MPC performance shown in Figure 6. Our experiments in Figures 4 (bottom) and Figure 5 show that with one hidden layer of ten neurons only, a satisfying approximation on a 24-hours time window can be achieved if the training data is large enough. To our understanding, this is due to the periodic nature of the input $\bar{w}$ that is given by repeating load and generation profiles. Note that NNs benefit from big data. In our case study, we trained the NN only on data corresponding to two weeks.

# 5 Numerical proof-of-concept

Model Predictive Control (MPC) is a method to tackle optimal control problems on an infinite time horizon by solving a series of finite dimensional optimization problems instead, see e.g. [19] for an introduction to non-linear MPC.

## 5.1 Model predictive control (MPC)

Consider the optimal control problem (6). In order to provide an optimal control sequence over an arbitrary long

time horizon we use MPC. To this end, at current time instance $k \in \mathbb{N}_0$ we assume the future net consumption $(w_i(k), w_i(k+1), \dots, w_i(k+N-1))^\top \in \mathbb{R}^N$ to be predicted for all subsystems $i \in [1 : \mathcal{I}]$. Based on the prediction, Algorithm 1 is executed (per MG) to determine control sequences $u_i$, $i \in [1 : \mathcal{I}]$, and an exchange strategy $\delta$. Then, only the first instances $u_i(k)$ and $\delta(k)$ are implemented and the time instance is incremented. Algorithm 2 outlines this MPC scheme.

**Algorithm 2** MPC for coupled MGs

**Input**: Current time instance $k \in \mathbb{N}_0$, current SoC $x_i(k) \in \mathbb{X}_{\kappa_i}$, prediction horizon $N \in \mathbb{N}_{\geq 2}$, and reference trajectory $(\bar{\zeta}(k), \dots, \bar{\zeta}(k+N-1))^\top \in \mathbb{R}^N$.

**Repeat**:

1. Measure current state $x_i(k)$ and update the forecast $(w_i(k), w_i(k+1), \dots, w_i(k+N-1))^\top$, $i \in [1 : \mathcal{I}]$.
2. Run Algorithm 1 for all MG in parallel to get optimal control sequences $u_i^\star = (u_i^\star(k), \dots, u_i^\star(k+N-1))^\top$ for all subsystems $i \in [1 : \mathcal{I}]$, and an optimal exchange strategy $\delta^\star = (\delta^\star(k), \dots, \delta^\star(k+N-1))$.
3. Implement $u_i^\star(k)$, $i \in [1 : \mathcal{I}]$, and $\delta^\star(k)$ and shift the time instance $k \to k+1$.

Note that Problems (6) or (9) and (2) have to be solved in order to determine $\bar{z}^\star$ and $\delta^\star$ in each MPC iteration. Therefore, the open-loop costs $\mathcal{J}(\bar{z}^\star, \delta^\star)$ can be computed in each iteration as well (cf. Figure 4). However, since Step 3 in Algorithm 2 suggests to only implement the first instance of the controls computed in Step 2, these costs are not attained. Instead the closed-loop costs

$$\sum_{\kappa=1}^{\Xi} \left( \bar{\zeta}(k)\mathcal{I}_\kappa - \sum_{\nu=1}^{\Xi} \delta_{\nu\kappa}^\star(k)\eta_{\nu\kappa}\mathcal{I}_\nu \bar{z}_\nu^\star(k) \right)^2 \tag{15}$$

are realized at each time step $k \in \mathbb{N}_0$ (cf. Figure 6).

## 5.2 Usage of surrogate models in MPC

We compare the performances using ADMM, RBFs, and NNs on the lower-level, i.e. in Step 4(b) of Algorithnm 1. In all numerical simulations we set $T = 0.5$, $N = 6$, $\Xi = 4$, $\mathcal{I}_1 = 50$, and $\mathcal{I}_2 = \mathcal{I}_3 = \mathcal{I}_4 = 10$. The battery parameters were randomly chosen with mean values $C = 0.98$, $\underline{u} = -0.24$, and $\bar{u} = 0.25$. Based on the battery capacities we set $\hat{x}_i = 0.5C_i$. In order to incorporate losses along the transmission lines, we used the efficiency matrix,

$$\eta = \begin{bmatrix} 1.0 & 0.9 & 0.9 & 0.85 \\ 0.9 & 1.0 & 0.0 & 0.85 \\ 0.9 & 0.0 & 1.0 & 0.0 \\ 0.85 & 0.85 & 0.0 & 1.0 \end{bmatrix} \text{ in (1).}$$

For simplicity, we only replaced the lower-level optimization routine for MG 1.

Results on the MPC closed loop can be found in Figure 6 and Table 1. In Figure 6 the closed-loop per-

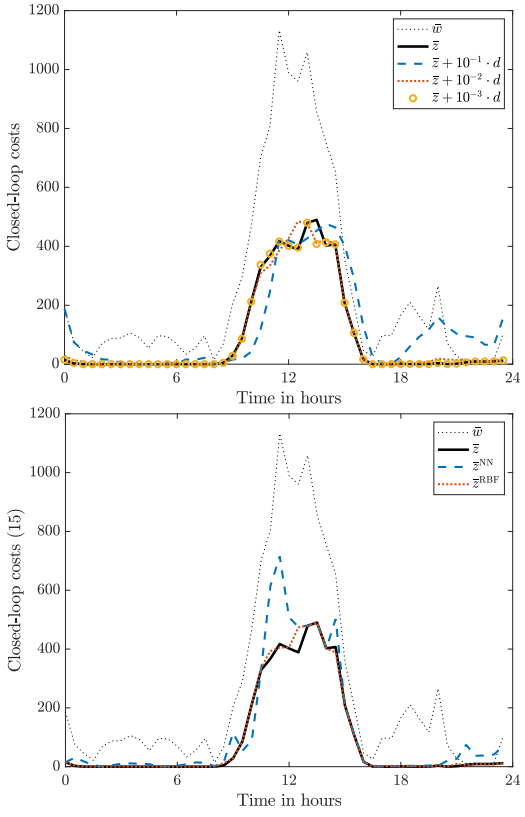


**Fig. 6:** Impact of mapping error (top) and approximation via RBF and NN (bottom) on the closed-loop performance (15) within 48 consecutive time steps.

| | closed-loop cost | runtime [ms] |
|---|---|---|
| **no control** | **12,228** | — |
| **ADMM** | **4,416** | **2.5** |
| **RBFs** | **4,529** | **1.2** |
| **NNs** | **5,598** | **0.05** |

**Tab. 1:** Comparison of the summed MPC closed-loop performance $\sum_{k=0}^{47}\sum_{\kappa=1}^{\Xi}\left(\bar{\zeta}(k)\mathcal{I}_\kappa - \sum_{\nu=1}^{\Xi}\delta_{\nu\kappa}^\star(k)\eta_{\nu\kappa}\mathcal{I}_\nu\bar{z}_\nu^\star(k)\right)^2$ and runtime (per call): ADMM vs. RBFs vs. NNs.

formances of ADMM (black line) compared to perturbed ADMM, and ADMM (black line) compared to the two surrogate models are visualized. Similar to the open-loop case, small disturbances in ADMM have little impact and RBFs outperform the NN. The first column of Table 1 compares the sum of all MPC closed-loop performances using ADMM, RBFs and a NN while in column 2 the average runtimes of these approaches are reported. Note that when using a surrogate, we call ADMM once per MPC iteration. As elaborated in [5] in each ADMM iteration an $N$-dimensional vector has to be transmitted twice. Hence, both surrogates reduce the need for communication. Two great advantages of ADMM are that the local optimization (11a) can be parallelized and the global optimization is independent of the size of the MG. However, a single function evaluation such as (13) or (14) is faster than running the entire ADMM optimization routine.

In order to improve the performance of the NN, more sampling data has to be generated to increase the training set significantly. To avoid large offline computation times, we chose $N = 6$, i.e. a prediction horizon of three hours, which is rather short compared to [5, 13].

**Remark 2.** We point out two implementation details to solve (2) efficiently. First, note that the optimization (2) can be parallelized in $n$, since there is no coupling. Furthermore, we replace (2c) by

$$\delta_{\kappa\nu}(n) \cdot \delta_{\nu\kappa}(n) \ \leq \ \varepsilon$$

for some tolerance $\varepsilon > 0$ to smooth the feasible set.

# 6 Conclusions

In this paper we recalled an optimization problem arising in large-scale electrical networks. We proposed an iterative *bidirectional* optimization scheme to tackle this problem in a distributed way, and showed numerically that a sufficiently small error on the lower-level does not have noticeable impact on the performance of the overall optimization scheme. Based on this observation, we replaced the lower-level optimization by surrogates in order to reduce communication effort as well as computation time. We present to choices for surrogate models, radial basis functions and neural networks, and obtained a comparable approximation quality. The numerical results showed that the proposed optimization scheme can improve the overall performance compared to that in [13] and that surrogate models have the potential to reduce computational time and communication effort in MPC while preserving the quality of the optimization w.r.t (1). Based on our simulations the performances of RBFs and NNs are comparable.

The approach is promising and there are still some open problems that need to be investigated: Each MG needs to train its own surrogate and adjust it whenever there are changes in the MG which requires a large amount of training data. A possible way to tackle this problem might be to update the surrogates online during the MPC loop in a suitable way.

**Acknowledgment:** The authors are funded by the Federal Ministry for Education and Research (BMBF; grants 05M18EVA and 05M18SIA). Karl Worthmann is indebted to the the German Research Foundation (DFG; grant WO 2056/6-1).

# References

[1] D. E. Olivares, C. A. Cañizares, and M. Kazerani. A centralized optimal energy management system for microgrids. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–6, 2011.

[2] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Belmont, MA, USA: Athena Scientific, 1989.

[3] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011.

[4] B. Houska, J. Frasch, and M. Diehl. An Augmented Lagrangian Based Algorithm For Distributed Nonconvex Optimization. *SIAM J. Optim.*, 26(2):1101–1127, 2016.

[5] P. Braun, T. Faulwasser, L. Grüne, C. M. Kellett, S. R. Weller, and K. Worthmann. Hierarchical distributed ADMM for predictive control with applications in power networks. *IFAC J. Syst. Control*, 3:10 – 22, 2018.

[6] A. Engelmann, Y. Jiang, T. Mühlpfordt, B. Houska, and T. Faulwasser. Towards Distributed OPF using ALADIN. *IEEE Trans. Power Syst.*, 34(1):584–594, 2019.

[7] G. Graditi, M. L. Di Silvestre, R. Gallea, and E. R. Sanseverino. Heuristic-Based Shiftable Loads Optimal Management in Smart Micro-Grids. *IEEE Trans. Ind. Informat.*, 11(1):271–280, 2015.

[8] P. Braun, L. Grüne, C. M. Kellett, S. R. Weller, and K. Worthmann. Model Predictive Control of Residen-

tial Energy Systems Using Energy Storage & Controllable Loads. *Progress in Industrial Mathematics at ECMI 2014. Mathematics in Industry*, 22:617–623, 2016.

[9] R. R. Appino, J. Á. G. Ordiano, R. Mikut, T. Faulwasser, and V. Hagenmeyer. On the use of probabilistic forecasts in scheduling of renewable energy sources coupled to storages. *Applied Energy*, 210:1207–1218, 2018.

[10] Z. Feng, G. Wen, and G. Hu. Distributed Secure Coordinated Control for Multiagent Systems Under Strategic Attacks. *IEEE Trans. Cybern.*, 47(5):1273–1284, 2017.

[11] R. R. Appino, M. Muñoz-Ortiz, J. Á. G. Ordiano, R. Mikut, V. Hagenmeyer, and T. Faulwasser. Reliable Dispatch of Renewable Generation via Charging of Time-varying PEV Populations. *IEEE Trans. Power Syst.*, 34(2):1558–1568, 2018.

[12] P. Braun, P. Sauerteig, and K. Worthmann. Distributed optimization based control on the example of microgrids. In M. J. Blondin, P. M. Pardalos, and J. S. Sáez, editors, *Computational Intelligence and Optimization Methods for Control Engineering*, volume 150 of *Springer Optimization and Its Applications*. Springer International Publishing, 2019. To appear.

[13] S. Grundel, P. Sauerteig, and K. Worthmann. Surrogate Models For Coupled Microgrids. In I. Faragó, F. Izsák, and P. Simon, editors, *Progress in Industrial Mathematics at ECMI 2018*, 2019. To appear.

[14] M. D. Buhmann. *Radial basis functions: Theory and implementations*, volume 12. Cambridge university press, 2003.

[15] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), 2018.

[16] H. Zhang, F. Xu, and L. Zhou. Artificial neural network for load forecasting in smart grid. In *2010 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3200–3205, 2010.

[17] P. Siano, C. Cecati, H. Yu, and J. Kolbusz. Real Time Operation of Smart Grids via FCN Networks and Optimal Power Flow. *IEEE Trans. Ind. Informat.*, 8(4):944–952, 2012.

[18] A. Sinha, P. Malo, and K. Deb. A Review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications. *arXiv preprint arXiv: 1705.06270*, 2017.

[19] L. Grüne and J. Pannek. *Nonlinear Model Predictive Control. Theory and Algorithms*. Springer, London, 2 edition, 2017.

[20] M. Khalid and A. V. Savkin. A model predictive control approach to the problem of wind power smoothing with controlled battery storage. *Renewable Energy*, 35(7):1520–1526, 2010.

[21] A. Parisio, E. Rikos, and L. Glielmo. A model predictive control approach to microgrid operation optimization. *IEEE Trans. Control Syst. Technol.*, 22(5):1813–1827, 2014.

[22] K. Worthmann, C. M. Kellett, P. Braun, L. Grüne, and S. R. Weller. Distributed and Decentralized Control of Residential Energy Systems Incorporating Battery Storage. *IEEE Trans. Smart Grid*, 6(4):1914–1923, 2015.

[23] E. L. Ratnam, S. R. Weller, C. M. Kellett, and A. T. Murray. Residential load and rooftop PV generation: an Australian distribution network dataset. *Internat. J. Sustain. Energy*, 2015.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, 2006.

[25] S. N. Lophaven, H. B. Nielsen, and J. Søndergaard. DACE-A Matlab Kriging toolbox, version 2.0. Technical report, 2002.

[26] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[27] C. F. Higham and D. J. Higham. Deep Learning: An Introduction for Applied Mathematicians. *arXiv preprint arXiv:1801.05894*, 2018.