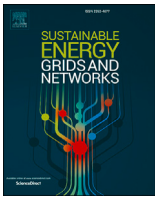
Sustainable Energy, Grids and Networks 44 (2025) 102033

Contents lists available at ScienceDirect

Sustainable Energy, Grids and Networks

journal homepage: www.elsevier.com/locate/segan



A python toolbox for flexibility aggregation and disaggregation: PyFlexAD

Emrah Öztürk[a], Kevin Kaspar[a], Timm Faulwasser[b], Karl Worthmann[c], Peter Kepplinger[a], Klaus Rheinberger[a,*]

[a] Energy Research Centre, Vorarlberg University of Applied Sciences, illwerke vkw Endowed Professorship for Energy Efficiency, Dornbirn, 6850, Austria
[b] Institute of Control Systems, Hamburg University of Technology, Hamburg, 21079, Germany
[c] Institute of Mathematics, Technische Universität Ilmenau, Ilmenau, 98693, Germany

HIGHLIGHTS

- An efficient flexibility (dis-)aggregation method for storage devices is presented.
- Hierarchical aggregation and disaggregation are covered by simple vector-matrix operations.
- The proposed method is provided as an open-source Python package.
- The method is tested against the central control mechanism and an LP-based alternative.
- The case study shows scalability with respect to the number of time periods and the number of devices.

ARTICLE INFO

Keywords:
Flexibility aggregation
Disaggregation
Distributed energy resources
Energy storage systems
Python
Minkowski sum computation

ABSTRACT

The increasing penetration of volatile renewables and growing electricity demand pose several challenges for power systems. Simultaneously, flexible devices – so called distributed energy resources (DER) – are becoming more widespread, making them attractive for providing ancillary services. The flexibility of a single device can be represented by a set of reference power profiles, and the flexibility of multiple devices by the summation of individual flexibility sets. However, set addition, also known as the Minkowski sum, is usually computationally intractable. This has led to the development of various approximation methods in the literature. The current study improves upon our previously published vertex-based inner approximation, by extending it to more general storage devices and hierarchical aggregation settings. We validate the efficacy and accuracy of the proposed method through case studies using real data and provide the source code of the algorithm as a Python package that enables the (dis-)aggregation of various flexible devices in real-world scenarios.

1. Introduction

As the world moves towards an increasing share of sustainable energy sources, renewables such as solar and wind power have become more widespread. Simultaneously, there has been a notable increase in electricity consumption [1]. The inherent volatility of renewable energy generation combined with the increasing electricity demand poses a significant risk to the reliable operation of modern power grids. At the same time, flexible devices – so called distributed energy resources (DERs) –, such as battery energy storage systems (BESSs), electric vehicles (EVs), thermostatically controlled loads (TCLs), etc., offer substantial flexibility potential. By effectively managing these devices, it is possible to shift electricity demand and to store excess generation for later use during periods of high demand.

For this purpose, aggregation is a viable framework that allows large-scale control of such flexible devices while preserving user privacy and reducing communication overhead. In this common framework, an intermediary entity, called an aggregator, coordinates a set of DERs and offers the aggregated flexibility on a flexibility market (e.g., wholesale, reserve, ancillary) to system operators (e.g., grid operators or energy utilities). By selling Demand Response services to an operator the aggregator appears as a ”virtual power plant” acting between the system operator, who purchases the aggregated flexibility, and the DER owners who provide control over their devices. The system operator uses

* Corresponding author.
Email addresses: emrah.oeztuerk@fhv.at (E. Öztürk), kevin.kaspar@fhv.at (K. Kaspar), timm.faulwasser@tuhh.de (T. Faulwasser), karl.worthmann@tu-ilmenau.de (K. Worthmann), peter.kepplinger@fhv.at (P. Kepplinger), klaus.rheinberger@fhv.at (K. Rheinberger).

https://doi.org/10.1016/j.segan.2025.102033
Received 18 December 2024; Received in revised form 21 October 2025; Accepted 25 October 2025
Available online 27 October 2025

the aggregated flexibility to optimize some objective function, e.g. peak power or cost. The DER owners, on the other hand, are compensated by the aggregator for the modification of their consumption patterns. This setting has been widely studied in the literature, cf. [2–5].

From a technical perspective, the aggregator manages contracted user devices by (approximately) assessing their aggregate power flexibility. The aggregator transforms all individual power constraints into constraints that refer only to the aggregate power space. In this way, the aggregate flexibility is a set of fewer dimensions and hides the individually possible power profiles. The system operator who purchases the aggregated flexibility selects an optimal power profile from the aggregate flexibility set, and the aggregator disaggregates this aggregate power profile to the individual devices.

The pivotal bottleneck in this aggregation-based control and coordination is the numerical scalability of the aggregation problem, i.e., the quantification of the aggregated flexibility given by the Minkowski sum of the individual flexibility sets. As the time horizon grows and the number of devices increases, the exact computation becomes computationally intractable [6]. For this reason, several approximate approaches have been proposed in the literature. We provide a brief overview in the following and refer to our previous comparative literature overview and benchmark [7].

Existing approximation methods can be roughly divided into inner and outer approximations. For example, [8,9] propose a simple outer approximation strategy which assumes that the individual flexibility sets are given in half-space representation with a common constraint matrix. As with other outer approximations, e.g. [10–13], the feasible region is typically overestimated and, thus, may contain infeasible power profiles. Auxiliary service providers must ensure the fulfillment of the power profiles to avoid contractual penalties, which favors inner approximations. These are conservative, i.e., the feasible region is potentially reduced, excluding feasible power profiles. The authors of Ref. [14,15] employ a bottom-up approach based on zonotopes. Zonotopes have the attractive feature that their Minkowski sum can be calculated efficiently. In [14,15] the authors show how optimal zonotopic approximations of flexibility can be computed efficiently for different objectives and present an economically fair and computationally efficient disaggregation method. Similarly, [11,16,17] employ strategies based on homothets whose Minkowski sum is also efficient. In [16] the high-dimensionality concerns of union-based computations based on homothets are alleviated to reduce the computational complexity. An inner approximation of thermostatically controlled loads (TCLs) that are described by continuous-power models is provided in [10,12]. The main idea of the ellipsoid-based approach in [18] is to fit an ellipsoid with maximum volume in the implicitly described higher-dimensional Minkowski sum which is then projected back to the aggregation space. Another ellipsoid projection technique is described in [19] where the authors define the inclusion constraint solely in the aggregation space. An alternative inner approximation is given in [20] where the individual power and energy constraints are added initially like in [8], resulting in an outer approximation. In a second step, the approximation of the aggregated flexibility is constructed, whereby feasible disaggregation is guaranteed. Concerning the aggregation of electric vehicles (EVs), in [21] a multi-battery flexibility model is constructed from base sets that reflect different individual EV flexibility sets, and a clustering approach is presented to identify these base sets. The authors of Ref. [22] deal with time-coupled, uncertain, and heterogeneous individual EV models by expressing each EV's flexibility as a polytope under H-representation including various operational constraints. In [23] an analytical polytope approximation aggregation model is developed that also takes uncertainties into account. Finally, we want to mention the exact aggregation method presented in [24], where the authors restrict the modeling to one charging window and do not consider time-dependent power or energy constraints. This exact aggregation clearly outperforms approximations but has limitations concerning modeling and scaling.

However, state-of-the-art inner approximations typically suffer from objective-dependent performance, increased computational complexity, and might exclude nominal power profiles such as the idle state, cf. [7]. In our previous study [25], we developed a vertex-based inner approximation that mitigates these problems. Our approach outperformed ten state-of-the-art inner approximations in the open-source aggregation benchmark [7]. However, there were still limitations with respect to the applicability of this method. In this paper, we extend the method of Ref. [25] to cover a wider range of DERs. The contributions of the present paper are as follows:

- We propose an improved algorithm for aggregating various storage models with varying energy and power limits.
- Hierarchical aggregation and disaggregation are covered by simple vector-matrix operations.
- We demonstrate the applicability of the proposed approach in a case study and show its effectiveness compared to a centralized control scheme and a closely related Linear Programming (LP) based approach.
- The proposed method does not require a (commercial) LP solver and can be implemented without any additional software exploiting parallelization.
- Finally, to facilitate the use of the algorithm, we provide the source code as a fully documented Python package, complete with examples.

The remainder of this paper is organized as follows: Section 2 explores the aggregation and disaggregation of DERs, starting with Section 2.1, which introduces an energy storage model, discusses its applications, and provides a conversion table for mapping various types of DERs to this model. Section 2.2 details the algorithm developed to quantify the aggregated flexibility of a set of energy storage models, while Section 2.3 addresses the disaggregation problem. Section 3 provides an overview of the Python package implementation, including a detailed description of its software architecture along with illustrative code examples. Section 4 applies the proposed algorithm to a case study to evaluate its accuracy and computational complexity, compared to a centralized control framework and an LP based approach. Finally, Section 5 concludes the paper.

2. Aggregation and disaggregation

This section explores the (dis-)aggregation of DERs. We begin by presenting an energy storage model together with a table that shows the correspondence of real-world devices to this model. Next, we introduce an algorithm designed for aggregating a set of energy storage models. Finally, we introduce an LP-based alternative and discuss the disaggregation process, including its application in hierarchical aggregation settings.

2.1. Energy storage models

In this section, we recall an energy storage model capable of modeling a variety of practical devices. We consider a discrete-time setting with d time intervals each of duration Δt. Let x_t (kW) be the (dis-)charging power during period t and let S_t (kWh) be the energy in the storage after period t. Then, we define an energy storage model as the set of potential power profiles $x \in \mathbb{R}^d$ fulfilling the constraints and system dynamics:

$$\underline{x}_t \le x_t \le \overline{x}_t \quad \forall t = 1, \dots, d, \tag{1a}$$

$$S_t = \alpha S_{t-1} + x_t \Delta t \quad \forall t = 1, \dots, d, \tag{1b}$$

$$\underline{S}_t \le S_t \le \overline{S}_t \quad \forall t = 1, \dots, d, \tag{1c}$$

$$S_0 = S_{\text{init}}, \tag{1d}$$

where $\underline{x}, \overline{x}, \underline{S}, \overline{S} \in \mathbb{R}^d$ denote the lower and upper limits for the variables x, and $S \in \mathbb{R}^d$, respectively. $S_{\text{init}} \in \mathbb{R}$ (kWh) denotes the initial energy and $\alpha \in (0, 1]$ the self-discharge factor. Eq. (1) defines the polytope

Table 1
Conversion table for storage parameters: $\underline{x}, \overline{x}, \underline{S}, \overline{S}, \alpha$, and S_{init}.

	$\underline{x}_t$	$\overline{x}_t$	$\underline{S}_t$	$\overline{S}_t$	α	S_{init}
BESS	$x_{\min}$	$x_{\max}$	$S_{\min}\ \forall t = 1, \dots d-1,\ S_f$ for $t = d$	$S_{\max}$	α	S_{init}
TCL (cooling)	$\frac{\theta_a-\theta_r}{\eta R} - p_{\max}$	$\frac{\theta_a-\theta_r}{\eta R}$	$-\frac{C\Delta}{2\eta}$	$\frac{C\Delta}{2\eta}$	$1-\frac{1}{RC}$	$\frac{C(\theta_{\text{init}}-\theta_r)}{\eta}$
TCL(heating)	$\frac{\theta_a-\theta_r}{\eta R}$	$p_{\max} + \frac{\theta_a-\theta_r}{\eta R}$	$-\frac{C\Delta}{2\eta} + \frac{\Delta t}{\eta}\sum_{\tau=1}^{t}\alpha^{t-\tau}D_\tau$	$\frac{C\Delta}{2\eta} + \frac{\Delta t}{\eta}\sum_{\tau=1}^{t}\alpha^{t-\tau}D_\tau$	$1-\frac{1}{RC}$	$\frac{C(\theta_{\text{init}}-\theta_r)}{\eta}$
EV	$\lambda_t x_{\min}$	$\lambda_t x_{\max}$	$S_{\min} + \sum_{\tau=1}^{t}\alpha^{t-\tau}D_\tau\Delta t\ \forall t = 1, \dots d-1,\ S_f + \sum_{\tau=1}^{t}\alpha^{t-\tau}D_\tau\Delta t$ for $t = d$	$S_{\max} + \sum_{\tau=1}^{t}\alpha^{t-\tau}D_\tau\Delta t$	α	S_{init}
PHES	$x_{\min}$	$x_{\max}$	$R_{\min}\rho g h$	$R_{\max}\rho g h$	1	$R_{\text{init}}\rho g h$

$$\mathcal{B}(\underline{x}, \overline{x}, \underline{S}, \overline{S}, \alpha, S_{\text{init}}) := \left\{x \in \mathbb{R}^d : x \text{ fulfills (2a) to (2d)}\right\},$$

$$A(\alpha)x \le b(\underline{x}, \overline{x}, \underline{S}, \overline{S}, \alpha, S_{\text{init}}), \tag{2a}$$

$$A(\alpha) := \left(-I, I, \Gamma^\top, -\Gamma^\top\right)^\top, \tag{2b}$$

$$b(\underline{x}, \overline{x}, \underline{S}, \overline{S}, \alpha, S_{\text{init}}) := \left(-\underline{x}^\top, \overline{x}^\top, \frac{1}{\Delta t}(\overline{S} - S_{\text{init}}a_d)^\top, \frac{1}{\Delta t}(-\underline{S} + S_{\text{init}}a_d)^\top\right)^\top, \tag{2c}$$

$$a_d := (\alpha, \alpha^2, \dots, \alpha^d)^\top. \tag{2d}$$

Here, $I \in \mathbb{R}^{d\times d}$ is the identity matrix, and $\Gamma \in \mathbb{R}^{d\times d}$ a Toeplitz matrix with first column and row defined by $(1, \alpha, \dots, \alpha^{d-1})^\top$ and $(1, 0, \dots, 0)$, respectively. Note that this model does not account for (dis-)charging efficiencies, assuming no energy loss during these processes. These efficiencies lead to non-convexities, which are typically addressed by reformulating the problem into Mixed Integer Linear Programming (MILP) format (cf. [26]). Alternative continuous approximations are discussed in [27].

The energy storage model can be applied to various appliances in practice. Below, we provide examples and a conversion table (cf. Table 1), which maps various types of DERs to the energy storage model.

2.1.1. Battery energy storage systems

A BESS is an electrochemical storage system that can store energy for later use, allowing operational flexibility. These devices can be characterized by the parameter vector $(x_{\min}, x_{\max}, S_{\min}, S_{\max}, \alpha, S_{\text{init}}, S_f) \in (-\infty, 0]\times[0, \infty)\times[0, \infty)\times(S_{\min}, \infty)\times(0, 1]\times[S_{\min}, S_{\max}]^2$, where $x_{\min}, x_{\max}$ (kW) are the lower and upper power limits for (dis-)charging, $S_{\min}, S_{\max}$ (kWh) are the lower and upper limits for the energy in the BESS, α is the self-discharge factor, S_{init} (kWh) is the initial energy, and S_f (kWh) is the minimum final energy. The constraints and system dynamics of a BESS can then be described by (1) using the storage parameters according to Table 1, cf. [25].

2.1.2. Thermostatically controlled loads

TCLs, e.g., refrigerators, air conditioners, water heaters, heat pumps, etc., are appliances designed to maintain some thermal mass within a desired temperature range. The temperature change in a TCL is usually described by a linear time dependence on the temperature difference, together with an additional term for the power consumption. During operation, the temperature shall remain within a certain range around the set point temperature. However, the power supply can be shifted in time, which offers operational flexibility. TCLs can be characterized by a parameter vector $(p_{\max}, R, C, \eta, \Delta, \theta_a, \theta_r, \theta_{\text{init}}) \in [0, \infty)^7 \times [\theta_r - \Delta, \theta_r + \Delta]$. In this vector, $p_{\max}$ (kW) is the maximum power, $R\left(\frac{K}{kW}\right)$ is the thermal resistance, $C\left(\frac{kWh}{K}\right)$ is the thermal capacitance, η is the coefficient of performance, Δ (K) is the dead band, θ_a (K) denotes ambient temperature, θ_r (K) indicates the set point temperature, and θ_{init} (K) represents the initial temperature. The power required to keep the appliance at its set point temperature is given by $x_0 := \pm\frac{\theta_a-\theta_r}{\eta R}$ (plus for a cooling device and minus for a heating device), and the self-discharge factor by $\alpha = 1 - \frac{\Delta t}{RC}$. For appliances like water heaters, an additional demand $D \in \mathbb{R}^d_{\ge 0}$ can be taken into account. With the variable transformations $S_t = \frac{C(\theta_t-\theta_r)}{\eta}$, and $x_t = \pm(-p_t + x_0)$ (plus for a cooling device and minus for a heating device), the constraints and system dynamics are given by (1) and the parameters according to Table 1, cf. [10,11]. Note that due to these substitutions, the power profiles become $x_0 \pm x(t)$ (minus for a cooling device and plus for a heating device).

2.1.3. Electric vehicles

An EV can be modeled as a BESS with limited availability and additional trip consumption. Therefore, apart from the characteristics of a BESS, the availability vector $\lambda \in \{0, 1\}^d$ and the trip consumption vector $D \in \mathbb{R}^d_{\ge 0}$ (kW) are introduced. Given the parameters and conversions listed in Table 1, the constraints and system dynamics can be expressed using (1).

2.1.4. Pumped hydro energy storages

A pumped hydro energy storage (PHES) pumps water from a lower reservoir to an upper reservoir, thereby increasing the water's potential energy. This potential energy can later be used to generate electricity. A PHES can be characterized by the parameter vector $(x_{\min}, x_{\max}, R_{\min}, R_{\max}, h, R_{\text{init}}) \in (-\infty, 0]\times[0, \infty)^4 \times [R_{\min}, R_{\max}]$, where $x_{\min}, x_{\max}$ (kW) are the lower and upper power limits, $R_{\min}, R_{\max}$ (m^3) are the lower and upper limits for the volume in the upper reservoir, h (m) is the height between the upper and lower reservoir, and R_{init} is the initial water volume in the upper reservoir. In addition, the unit conversion parameter $\eta^\star = \frac{1}{\rho g h}\left(\frac{m^3}{Ws}\right)$ is introduced, where $\rho = 1000\left(\frac{kg}{m^3}\right)$ is the water density and $g = 9.81\left(\frac{m}{s^2}\right)$ is the gravitational acceleration. The constraints and system dynamics are then given by (1) together with the conversions in Table 1. Note that, this model disregards the efficiencies in pumping and generation, i.e., it is assumed that energy is not lost during these processes. In reality, the pump and turbine efficiencies range from 75 % to 85 % and from 93 % to 95 %, respectively, cf. [28]. For a more detailed model that takes these efficiencies into account, we refer to [29].

2.2. Aggregation

In this section, we present an enhanced algorithm to quantify the aggregated flexibility of a set of energy storage models. To this end, extreme actions, ideally vertices or elements located on the boundary, are computed for each energy storage model. Corresponding extreme actions are then summed to obtain elements in the aggregated flexibility

$$\left\{x \in \mathbb{R}^d : x = \sum_{i=1}^{n} x_i, x_i \in \mathcal{B}\left(\underline{x}_i, \overline{x}_i, \underline{S}_i, \overline{S}_i, \alpha_i, S_{\text{init},i}\right)\right\},$$

i.e., the Minkowski sum of energy storage models. The convex hull of the summed extreme actions then results in an inner approximation of the aggregated flexibility.

Algorithm 1 (extremeActions).

Require: $\underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, \mathcal{J} \subseteq \{-1, 1\}^d$
Ensure: Y
1: $Y \leftarrow \mathbf{0}_{d\times|\mathcal{J}|}$ ▷ $d \times |\mathcal{J}|$ matrix of zeros
2: **for** $j \in \mathcal{J}$ **do**
3: $k \leftarrow 1$
4: $y^j \leftarrow \mathbf{0}_d$
5: **for** $t = 1$ **to** d **do** ▷ try, otherwise solve (3), and return x if $t \leq 0$
6: **if** $j_t = 1$ **then**
7: $y_t^j \leftarrow \max\left(\min\left(\overline{x}_t, \frac{\overline{S}_t - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{t-1} \alpha^{t-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \underline{x}_t\right)$
8: $y^j \leftarrow \text{correctiveIncrease}(y^j, \underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, t)$
9: $y^j \leftarrow \text{correctiveDecrease}(y^j, \underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, t)$
10: **else if** $j_t = -1$ **then**
11: $y_t^j \leftarrow \min\left(\max\left(\underline{x}_t, \frac{\underline{S}_t - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{t-1} \alpha^{t-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \overline{x}_t\right)$
12: $y^j \leftarrow \text{correctiveIncrease}(y^j, \underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, t)$
13: $y^j \leftarrow \text{correctiveDecrease}(y^j, \underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, t)$
14: **end if**
15: **end for**
16: $Y[:, k] \leftarrow y^j$
17: $k \leftarrow k + 1$
18: **end for**

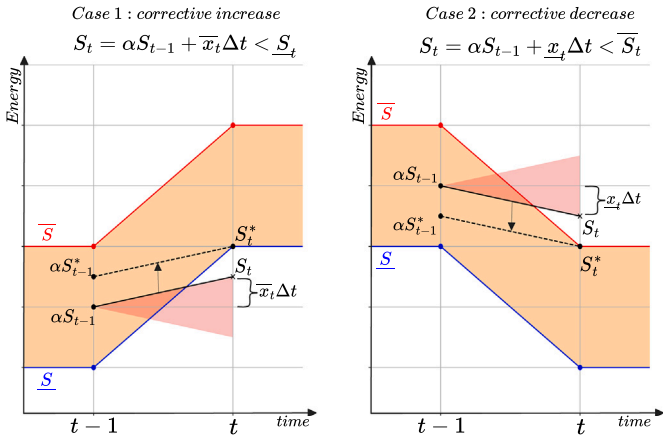


Fig. 1. Left: Corrective increase from S_t to S_t^* in order to comply with high energy bounds. Right: Corrective decrease from S_t to S_t^* in order to comply with low energy bounds.

Our approach to generating extreme actions involves moving as far as possible in specific directions within the polytopes. To this end, we use -1 to indicate the negative direction and 1 to indicate the positive direction of an axis in $\mathbb{R}^d$. A corresponding direction in d-dimensional space can then be represented by a vector $j \in \{-1, 1\}^d$.

Algorithm 1 calculates extreme actions y^j for the set of directions $\mathcal{J} \subseteq \{-1, 1\}^d$. This is done by moving as far as possible in each direction j_t, i.e., charging or discharging the storage to the limits in all time periods. Lines 7 and 11 enforce the power constraints, i.e., $\underline{x}_t \leq y_t^j \leq \overline{x}_t$. However, as the energy limits are allowed to vary over time, future energy constraints could be violated. Two cases are distinguished as illustrated in Fig. 1.

The case where the energy S_t is less than the lower limit $\underline{S}_t$, shown in Fig. 1, is handled in Algorithm 2. In Line 2, i is initialized with the first index of reversed (i.e., inverted) $\overline{x}_{[t]}$, where $\overline{x}_i > 0$ applies. Here $x_{[t]} \in \mathbb{R}^t$ refers to the vector consisting of the first t elements of $\overline{x}$. This step is necessary to identify the last index prior that allows for correction, i.e., positive power. A correction to increase the energy must be carried out

Algorithm 2 (correctiveIncrease).

Require: $y^j, \underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, t$
Ensure: y^j
1: **if** $\underline{S}_t > \alpha^t S_{\text{init}} + \sum_{\tau=1}^{t} \alpha^{t-\tau} y_\tau^j \Delta t$ **then**
2: init i ▷ first index with $\overline{x}_i > 0$ in reversed $\overline{x}_{[t]}$
3: $y_{t-i+1}^j \leftarrow \min\left(\max\left(\underline{x}_{t-i+1}, \frac{\underline{S}_t - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{t-1} \alpha^{t-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \overline{x}_{t-i+1}\right)$
4: $k \leftarrow t - i$
5: **while** $\underline{S}_t > \alpha^t S_{\text{init}} + \sum_{\tau=1}^{t} \alpha^{t-\tau} y_\tau^j \Delta t$ **do**
6: **for** $l = k$ to $t - i$ **do**
7: $y_l^j \leftarrow \max\left(\min\left(\overline{x}_l, \frac{\overline{S}_l - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{l-1} \alpha^{l-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \underline{x}_l\right)$
8: **end for**
9: $y_{t-i+1}^j \leftarrow \min\left(\max\left(\underline{x}_{t-i+1}, \frac{\underline{S}_t - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{t-1} \alpha^{t-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \overline{x}_{t-i+1}\right)$
10: $k \leftarrow k - 1$
11: **end while**
12: **end if**

Algorithm 3 (correctiveDecrease).

Require: $y^j, \underline{x}, \overline{x}, \underline{S}, \overline{S}, S_{\text{init}}, \alpha, t$
Ensure: y^j
1: **if** $\overline{S}_t < \alpha^t S_{\text{init}} + \sum_{\tau=1}^{t} \alpha^{t-\tau} y_\tau^j \Delta t$ **then**
2: init i ▷ first index with $\underline{x}_i < 0$ in reversed $\underline{x}_{[t]}$
3: $y_{t-i+1}^j \leftarrow \max\left(\min\left(\overline{x}_{t-i+1}, \frac{\overline{S}_t - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{t-1} \alpha^{t-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \underline{x}_{t-i+1}\right)$
4: $k \leftarrow t - i$
5: **while** $\overline{S}_t < \alpha^t S_{\text{init}} + \sum_{\tau=1}^{t} \alpha^{t-\tau} y_\tau^j \Delta t$ **do**
6: **for** $l = k$ to $t - i$ **do**
7: $y_k^j \leftarrow \min\left(\max\left(\underline{x}_l, \frac{\underline{S}_l - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{l-1} \alpha^{l-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \overline{x}_l\right)$
8: **end for**
9: $y_{t-i+1}^j \leftarrow \max\left(\min\left(\overline{x}_{t-i+1}, \frac{\overline{S}_t - \left(\alpha^t S_{\text{init}} + \sum_{\tau=1}^{t-1} \alpha^{t-\tau} y_\tau^j \Delta t\right)}{\Delta t}\right), \underline{x}_{t-i+1}\right)$
10: $k \leftarrow k - 1$
11: **end while**
12: **end if**

with strictly positive values and is not possible with indices $l > t - i + 1$, as they are either zero or negative due to $\overline{x}_l \leq 0$. The power profile is then changed backwards, starting with the index $t - i + 1$ to reach $\underline{S}_t$ without violating the constraints. Lines 3, 7, and 9 enforce the power constraints, i.e., it holds that $\underline{x}_{t-i+1} \leq y_{t-i+1}^j \leq \overline{x}_{t-i+1}$. In Line 7, the y_k^j are modified to increase the energy in the storage.

The second case, on the right in Fig. 1, where the energy S_t is greater than $\overline{S}_t$, is covered similarly in the corrective algorithm to decrease the energy, cf. Algorithm 3. These corrections, if needed, are applied in Lines 8, 9, 12, and 13 of Algorithm 1.

In the event that the while loop on Line 5 of Algorithms 2 and 3 terminates with an error (i.e., when $k < 1$), we propose employing a try-catch mechanism within the for loop on Line 5 of Algorithm 1 to handle the error and address the subsequent feasibility problem:

$$\min_{x,t} \; t \tag{3a}$$

subject to

$$A(\alpha)x \leq b(\underline{x}, \overline{x}, \underline{S}, \overline{S}, \alpha, S_{\text{init}}) + t\mathbf{1}_{4d}. \tag{3b}$$

If the minimizer (x, t) satisfies $t \leq 0$, then feasibility is established. Therefore, if the while loop terminates with an error and the polytope is non-empty, x can be returned as a feasible solution.

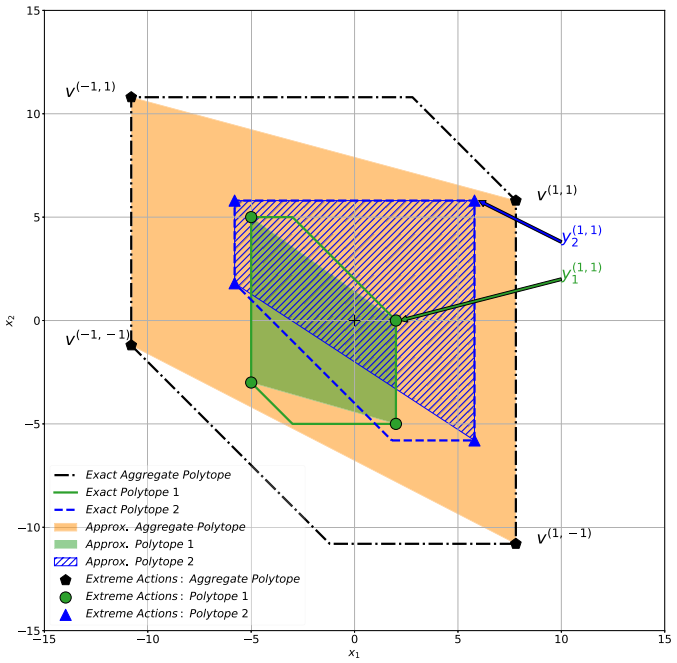


Fig. 2. The individual polytopes are shown in dashed blue and solid green, along with some extreme actions y_1^j, y_2^j and their Minkowski sum in dashed-dotted black. The summed extreme actions $v^j, j \in \{-1,1\}^2$ with their convex hull are shown in yellow. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Given n devices, we now assume that sets of extreme actions $\{y_i^j : j \in \mathcal{J}\}$ were computed for $i = 1, \dots, n$ using Algorithm 1. The sum of extreme actions with identical vector j across all devices is denoted as:

$$v^j := \sum_{i=1}^{n} y_i^j. \tag{4}$$

The matrix of summed vectors can then be obtained by adding the matrices returned by Algorithm 1, i.e.,

$$V := \sum_{i=1}^{n} Y_i.$$

Note that, by construction, the vector v^j is an element of the aggregated flexibility. Furthermore, the polytope $\mathcal{B}(\underline{x}_i, \overline{x}_i, \underline{S}_i, \overline{S}_i, \alpha_i, S_{\text{init},i})$ is convex, and the Minkowski sum of polytopes is itself a polytope, cf. [30]. Thus, the convex hull formed by the summed vectors,

$$\text{Conv}(v^j : j \in \mathcal{J}) := \left\{ x \in \mathbb{R}^d : x = \sum_{j \in \mathcal{J}} \alpha_j v^j, \sum_{j \in \mathcal{J}} \alpha_j = 1, \alpha_j \geq 0 \right\},$$

is an inner approximation of the aggregated flexibility, i.e., the Minkowski sum.

Since the extreme actions are computed for every $j \in \mathcal{J}$ and each device $i = 1, \dots, n$, the amount of extreme action computations is given by $|\mathcal{J}|n$. Given that the number of possible extreme directions is $|\{-1,1\}^d| = 2^d$, in high-dimensional spaces (e.g., $d > 8$), we propose selecting the set of extreme directions $\mathcal{J}$ by uniformly choosing g distinct elements from $\{-1,1\}^d$. Here, g should increase with the number of time periods, as the number of extreme points increases with dimensionality. For instance, a hypercube has 2^d vertices in d-dimensional space.

Fig. 2 illustrates the proposed algorithm. The extreme actions y_1^j, y_2^j together with their sum v^j in the aggregate feasible region are shown on the left for $j = (1,1)$. Whereas on the right, all possible summed extreme actions $v^j, j \in \{-1,1\}^2$ are shown together with their convex hull.

It should also be noted that for BESSs (cf. Section 2.1.1), the proposed approach simplifies to the method proposed in [25]. We showed therein, that the summed extreme actions are vertices of the aggregate feasible region.

An alternative approach to calculate extreme actions takes a given j vector as the coefficient vector of a linear objective function that is maximized over the individual flexibility sets $\mathcal{B}_i$. In this way, the individual LPs result in vertices of the individual devices' flexibility sets. The vertices can be combined, analogous to the proposed algorithm above, to obtain boundary points of the aggregate flexibility set. Finally, the convex hull of the latter forms again an inner approximation of the aggregate flexibility set. This approach, which we name LPCH for "LP based Convex Hull", is also implemented in the Python package and will be compared to our non-LP algorithm which we name EACH for "Extreme Actions Convex Hull".

2.3. Disaggregation

Given a matrix of summed extreme actions $V = \sum_{i=1}^{n} Y_i$, the purchaser of the aggregated flexibility can choose a power profile x within the aggregated feasible region; that is, she selects a $\alpha \in \mathbb{R}^{|\mathcal{J}|}$ such that:

$$x = V\alpha, \tag{5a}$$

$$\sum_{j \in \mathcal{J}} \alpha_j = 1, \tag{5b}$$

$$\alpha_j \geq 0, \quad \forall\, j \in \mathcal{J}. \tag{5c}$$

The vector α parametrizes the convex hull of the summed extreme actions. This parametrization can be readily used by the optimizing entity. The so-chosen power profile x must then be disaggregated, i.e., distributed to the individual flexible devices. We apply the method previously published in [25] for this purpose. Using $V = \sum_{i=1}^{n} Y_i$ in (5a) leads to

$$x = V\alpha = \left(\sum_{i=1}^{n} Y_i \right) \alpha = \sum_{i=1}^{n} Y_i \alpha.$$

Hence, the power profile applied by device i is given by $Y_i\alpha$, i.e., the device's matrix multiplied by the weight vector α.

Note that this expression remains unchanged for scenarios with multiple aggregators (hierarchical aggregation settings), as each aggregation level can be represented by the summation of the previous levels. By applying this rule successively, the matrix corresponding to the top-level aggregator V can be represented by the summation of the device-level matrices.

Fig. 3 displays a general hierarchical aggregation setting, where the number of indices corresponds to the hierarchical level, e.g., no indices for the top level, one index for the subsequent level, etc. The matrix V corresponding to the top-level aggregator can be expressed as $V = \sum_{i=1}^{m} V_i$. Following the path to the device level aggregators, this can be re-expressed as $V = \sum_{i=1}^{m} \cdots \sum_{k=1}^{r} \sum_{h=1}^{l} \sum_{f=1}^{a} Y_{i,\dots,k,h,f}$. Using the same reasoning as above, we get:

$$x = V\alpha = \left(\sum_{i=1}^{m} \cdots \sum_{k=1}^{r} \sum_{h=1}^{l} \sum_{f=1}^{a} Y_{i,\dots,k,h,f} \right) \alpha = \sum_{i=1}^{m} \cdots \sum_{k=1}^{r} \sum_{h=1}^{l} \sum_{f=1}^{a} Y_{i,\dots,k,h,f} \alpha.$$

Hence, the contribution of a specific device is given by the device's matrix times the weight vector α.

3. Implementation

In this section, we present the software package `PyFlexAD`, which implements the proposed (dis-)aggregation method EACH as well as the LP-based alternative LPCH for various DERs. The software is developed in Python 3 and licensed under the MIT License. The Python package is available for download from the Python Package Index (PyPI), cf. [31].

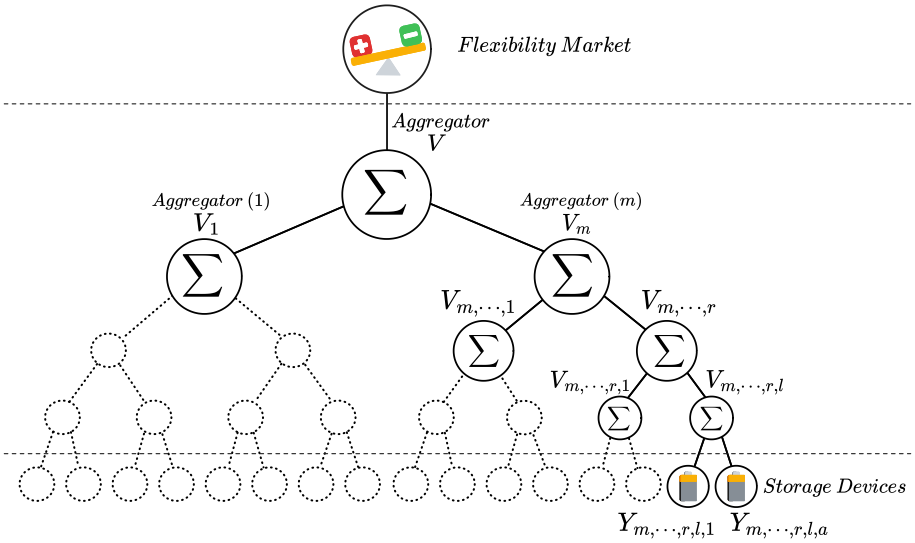


Fig. 3. General configuration for hierarchical aggregation, where each aggregator is associated with a matrix that represents the summed extreme actions. At the bottom, storage devices are displayed along with their corresponding device matrices.

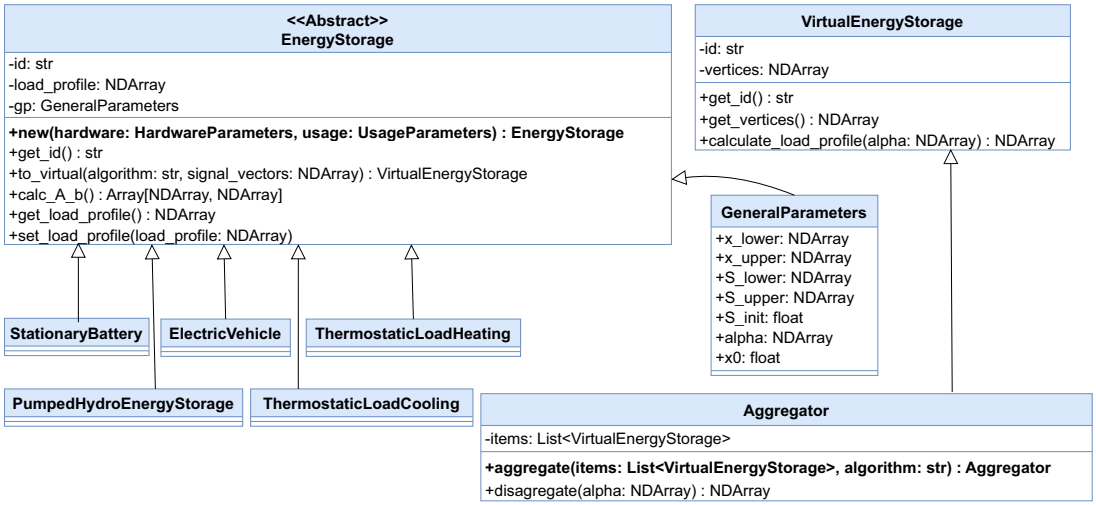


Fig. 4. Main classes in the package `PyFlexAD` using Unified Modeling Language (UML) notation.

The package's source code, along with sample code, can be accessed on GitHub, cf. [32].

To the best of the authors' knowledge, this is the first Python tool developed for the approximate quantification of the aggregate flexibility of energy storage models. Nevertheless, there are related software options available. For example, [33] offers a Python tool specifically designed to calculate the exact Minkowski sum of polytopes. However, due to the high computational complexity of these calculations, this software is limited to low-dimensional spaces. Another tool for quantifying the overall flexibility of selected devices is described in [34]. This tool first determines the optimal power profiles for each device in various settings using an MILP formulation to minimize the total costs. In a second step, the optimal power profiles are summed to yield a set of feasible aggregated power profiles.

Below, we provide an overview of our software, covering the class structure, system architecture, and implemented control strategies. Additionally, to demonstrate the software tool's functionality, we present an example code, which we discuss in detail.

3.1. Energy storage and subclasses

The package supports multiple types of energy storage systems, including BESSs, TCLs, EVs and PHESs. These models extend the `EnergyStorage` class, as depicted in Fig. 4.

The package focuses on modeling energy storage systems and does not include other energy resources without storage properties. Inflexible/non-controllable loads are either included as power demands of the devices (e.g. EVs or TCLs) or must be managed by the optimizing entity alongside the energy storage systems.

In order to calculate the parameters of an energy storage system, both hardware and usage parameters must be provided, cf. Table 1. Hardware parameters include, e.g., lower and upper energy limits, while usage parameters are user-specific and encompass factors such as EV availability and EV trip consumption. Each sub-class of `EnergyStorage` requires a specific set of parameters, cf. Fig. 5. To instantiate the class `StationaryBattery` instances of `BESSHardware` and `BESSUsage` are required. For ease of use, the package already includes a sample of hardware parameters for models from BESS manufacturers like Tesla

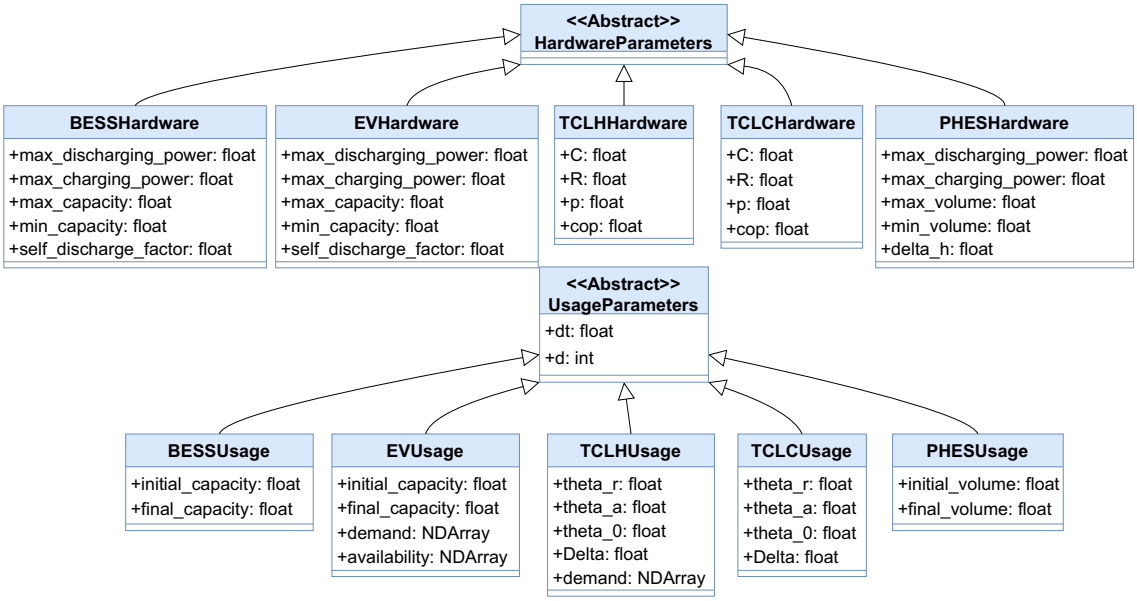


Fig. 5. Parameter classes in the package `PyFlexAD` using Unified Modeling Language (UML) notation.

Table 2
Battery energy storage system parameters.

	$x_{\min}$ (kW)	$x_{\max}$ (kW)	$S_{\min}$ (kWh)	$S_{\max}$ (kWh)	α
Tesla Powerwall 2	−5	5	0	13.5	1
Tesla Powerwall 3	−11.5	11.5	0	13.5	1
Tesla Powerwall +	−5.8	5.8	0	13.5	1
GENERAC PWRcell M3	−3.4	3.4s	0	9	1
GENERAC PWRcell M4	−4.5	4.5	0	12	1
GENERAC PWRcell M5	−5.6	5.6	0	15	1
GENERAC PWRcell M6	−6.7	6.7	0	18	1

Table 3
Electric vehicle system parameters.

	$x_{\min}$ (kW)	$x_{\max}$ (kW)	$S_{\min}$ (kWh)	$S_{\max}$ (kWh)	α
Nissan Leaf 6.6 kW AC	−6.6	6.6	0	39	1
Tesla Model Y 11 kW AC	−11	11	0	57.5	1
Tesla Model S P100D 16.5 kW AC	−16.5	16.5	0	95	1
Renault Zoe ZE40 R110 22 kW AC	−22	22	0	41	1
Renault Zoe ZE40 R110 40 kW DC	−40	40	0	41	1
Renault Zoe ZE50 R135 22 kW AC	−22	22	0	52	1
Renault Zoe ZE50 R135 41 kW DC	−41	41	0	52	1

Table 4
Thermostatically controlled load system parameters.

	C (kWh/K)	R (K/kW)	$p_{\max}$ (kW)	η
Generic air conditioner	2	2	5	2.5
Generic water heater	6	800	3	3

and GENERAC, cf. Table 2, EV manufacturers like Tesla, Nissan, and Renault for EV models, cf. Table 3, and generic air conditioning and water heaters for TCL models, cf. Table 4.

3.2. System architecture for flexibility control structures

Fig. 6 illustrates the basic system architecture of two different flexibility control structures—aggregation-based control and central control—organized into layers. The base layer, labeled as Physical Device Level, contains the physical energy storage devices, including their individual hardware and usage information. The top layer, i.e., the Energy Market Level, is where an optimizing entity gathers flexibility information and provides optimized power profiles to control the devices.

On the left side of Fig. 6, a control structure with a central controller is shown. In this structure, each physical device is directly controlled by the central controller, which requires access to the hardware and usage information of the devices. For convenience, the class `EnergyStorage` implements the method `calc_A_`, which computes the matrix A_i and the vector b_i from the storage parameters, cf. (2). Using the matrix A_i and the vector b_i of each device i, optimizations, such as peak-shaving or cost-reduction, can be performed through central controller implementations to obtain an optimal power profile. In detail, given matrices A_i, vectors b_i, and an objective function $f : \mathbb{R}^d \mapsto \mathbb{R}$, the central controller solves the following problem:

$$\min_{x, x_i} \; f(x) \tag{6a}$$

subject to

$$x = \sum_{i=1}^{n} x_i \tag{6b}$$

$$A_i x_i \leq b_i, \; \forall i = 1, \dots, n. \tag{6c}$$

In contrast, the right side of Fig. 6 illustrates an aggregation-based control structure. This structure includes a virtual abstraction layer (Virtual Aggregation Level) situated between the Physical Device Level and the Energy Market Level. In this layer, physical devices are virtualized, i.e., represented by their extreme actions and then aggregated. The collective storage within this layer resembles a virtual power plant.

3.3. Abstraction via virtualization

A fundamental concept in the `PyFlexAD` package is the virtualization of energy storage systems. This involves abstracting individual physical energy storages into virtual representations, which encapsulate the essential characteristics of the physical devices. The virtualization process begins with the calculation of polytope extreme actions to delineate the feasible operation regions of energy storage systems. To virtualize an instance of `EnergyStorage`, the method `to_virtual` must be called. When provided with a matrix of extreme directions, this method returns an instance of `VirtualEnergyStorage`. The link to its physical counterpart will remain only through a shared identification key.

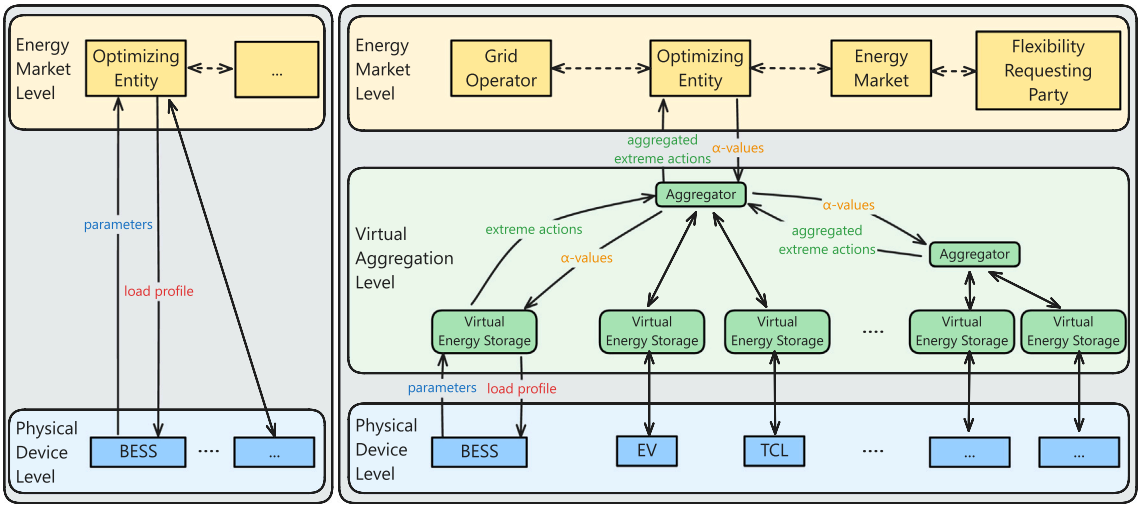


Fig. 6. System architecture overview: The left side displays the central control structure without aggregation, while the right side illustrates the aggregation-based control structure, featuring the Virtual Aggregation Level (center) and a hierarchical framework (far right). Physical devices at the bottom provide usage and hardware parameters, while the Virtual Aggregation Level virtualizes and further aggregates energy storage devices, resembling a virtual power plant.

3.4. Flexibility provision via aggregation

The class `Aggregator` is a subclass of `VirtualEnergyStorage`, not only inheriting all its properties but extending it by methods for (dis-)aggregation, cf. Fig. 4. For instance, the class method `aggregate` sums a provided set of virtual energy storages returning an `Aggregator` instance. Furthermore, since each `Aggregator` instance is a `VirtualEnergyStorage` object, aggregation of aggregators (hierarchical aggregation) is also possible. This allows constructs such as initial technical aggregations to provide flexibility services, followed by a market-focused aggregation to bundle multiple technical aggregators into a portfolio with enhanced flexibility resources, as illustrated on the right of Fig. 6.

In a similar manner as with the centralized control structure, the `PyFlexAD` package also provides implementations for the aggregation-based controller to optimize power profiles towards specific objectives. To apply the optimized power profile to the physical energy storage devices via the aggregated virtual storages, the aggregator performs the `disaggregate` method. This method signals each of its virtual energy storages to execute the `calc_power_profile` method using the α-values provided by the optimizing entity, cf. Section. 2.3. The disaggregated power profiles are then forwarded to the physical devices, cf. Fig. 6.

3.5. Illustrative code example

The Python code in Listing 1 demonstrates a workflow for optimizations with both centralized and aggregation-based control. In this scenario, the controller's objective is to minimize the peak power of the system, utilizing the flexibility provided by the energy resources. In order to plot the feasible regions, two discrete time intervals are analyzed, with each interval spanning 15 min (Line 13). The systems' power demand is set to 23 kW in the first and 21 kW in the second interval (Line 14). For simplicity the scenario includes two BESS devices of the same hardware type with the same initial and final energy conditions (Lines 20–22). Due to $d = 2$ and $g = d$, cf. Section 2.2, the set of extreme directions for the EACH algorithm results in $\mathcal{J} = \{-1, 1\}^2$ (Line 25). For the aggregation-based optimization (Lines 37–38) the flexible energy resources are virtualized (Lines 26–27) and aggregated (Line 30) using the proposed algorithm, whereas for the centralized optimization, the physical devices have to directly provide data of their hardware and usage (Lines 33–34).

In two dimensions, the exact calculation of vertices and the Minkowski sum of polytopes is possible. For verification purposes, the `PyFlexAD` package provides an implementation for this specific case, cf. [33]. To apply the exact calculations, the algorithm variable (Line 15) must be set to `Algorithms.EXACT`.

Fig. 7 shows the aggregated and individual feasible regions for exact vertices and extreme actions via the EACH algorithm, including the optimized operational points. The `PyFlexAD` package includes functions with various customization options to generate such two-dimensional plots effectively. Using centralized control yields a power profile of $x = (-7.0, -5.0)^\top$ kW for the combined BESS devices, cf. Fig. 7, reducing the peak power from 23.0 kW to 16.0 kW. Disaggregation of the centralized approach leads to individual profiles with $x_{\text{BESS1}} = (-2, -4)^\top$ kW and $x_{\text{BESS2}} = (-1, -5)^\top$ kW. However, due to the underestimated flexibility in the inner approximation, the power profile utilizing the proposed algorithm is $x = (-5.7, -3.7)^\top$ kW, cf. Fig. 7, reducing the peak power slightly less to 17.3 kW. In this case, the disaggregated load profiles for both devices are $x_{\text{BESS1}} = x_{\text{BESS2}} = (-2.9, -1.9)^\top$.

4. Case study

In this section, we present a case study to evaluate the accuracy and computational efficiency of the proposed aggregation method EACH. Our analysis has two primary goals: first, to assess the software's practical usage, and second, to demonstrate its scalability and accuracy.

To apply the methods to realistic market and consumption signals, data from the open-source repository [35] was employed. This repository compiles and preprocesses publicly available datasets relevant to demand-side management: The day-ahead price data originate from the ENTSO-E Transparency Platform [36], representing the Germany–Austria–Luxembourg bidding zone for the year 2016 with a 15-minute resolution. The EV trip consumption data are derived from the My Electric Avenue project conducted in the United Kingdom between 2013 and 2015, involving approximately 200 electric vehicle users across ten trial clusters [37]. The household electricity demand data are based on the Irish Social Science Data Archive (ISSDA) dataset CER Smart Metering Project – Electricity Customer Behaviour Trial [38], which recorded consumption profiles of around 4200 residential customers in Ireland between 2009 and 2010. Both the EV and household datasets were resampled and aligned within DSM-data to a uniform 15-minute resolution.

Our analysis is based on a one day scenario in a residential area with 5000 residents and one aggregator. The residents may own EVs, BESSs,

```python
import numpy as np
import PyFlexAD.models.bess.tesla as tesla
from PyFlexAD.physical.stationary_battery import StationaryBattery
from PyFlexAD.physical.stationary_battery import BESSUsage
from PyFlexAD.virtual.aggregator import Aggregator
from PyFlexAD.math.signal_vectors import SignalVectors
from PyFlexAD.utils.algorithms import Algorithms
from PyFlexAD.optimization.vertex_based_power_controller import VertexBasedPowerController
from PyFlexAD.optimization.centralized_power_controller import CentralizedPowerController

"""settings"""
d = 2  # number of time intervals
dt = 0.25  # interval duration in hours
system_power_demand = np.array([[23.0, 21.0]])  # total power demand for each interval in
    kW
algorithm = Algorithms.IABVG  # virtualization and aggregation algorithm
S_0 = 6.5  # initial battery capacity in kWh
S_f = 5.0  # final battery capacity in kWh

"""instantiate energy storage resources -> 2x Tesla Power Wall 2"""
usage = BESSUsage(initial_capacity=S_0, final_capacity=S_f, d=d, dt=dt)
bess_1 = StationaryBattery.new(hardware=tesla.power_wall_2, usage=usage)
bess_2 = StationaryBattery.new(hardware=tesla.power_wall_2, usage=usage)

"""virtualize"""
direction_vectors = SignalVectors.new(d)
virtual_ess_1 = bess_1.to_virtual(algorithm, direction_vectors)
virtual_ess_2 = bess_2.to_virtual(algorithm, direction_vectors)

"""aggregate"""
aggregator = Aggregator.aggregate([virtual_ess_1, virtual_ess_2], algorithm)

"""optimize power with centralized controller"""
centralized_controller = CentralizedPowerController(system_power_demand)
cc_power = centralized_controller.optimize([bess_1, bess_2])

"""optimize power with vertex-based controller"""
vertex_controller = VertexBasedPowerController(system_power_demand)
vc_power = vertex_controller.optimize(aggregator)
```

Listing 1: Example Code.

and TCLs (air conditioning systems) that can be controlled by the aggregator. The aggregator computes the approximate aggregate flexibility and offers it on the flexibility market. The system parameters used are listed in Table 2 for the BESS, in Table 3 for the EVs, and in Table 4 for the TCLs. Residential devices are chosen at random from this selection of possible values. We assume an ambient temperature of 30 °C. The initial temperature and the dead band of the air conditioning systems are selected uniformly from the intervals $[19, 20]$ °C and $[1.5, 2.5]$ °C, respectively. The set point temperatures for the air conditioning systems are set to 20 °C. Initially, the EV batteries are charged to 50 % SoC, while the BESSs initial energy is randomly selected from the interval $[\frac{1}{2}S_{\max,i}, S_{\max,i}]$. At the end of the operating window, EVs must retain a minimum energy level, which is determined by subtracting the trip consumption from the initial energy and adding the charged energy given by the data in [39]. The BESS, on the other hand, must retain the initial energy at the end of the operating window.

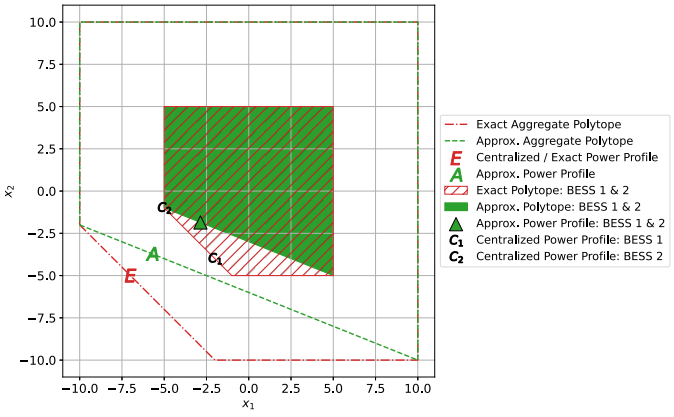


Fig. 7. An illustration of the aggregated feasible regions highlights the exact aggregation (dashed-dotted red line) and the approximate aggregation (dashed green line). Point E marks the central solution, while point A indicates the solution found in the approximation. The green triangle represents the disaggregated power profiles for the approximate aggregation, while the central disaggregated points are labeled as C_1 and C_2. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

All linear optimizations are solved using Gurobi [40]. The Python script for the case study, along with scripts for additional applications, is available at [32]. All computations are conducted on a system equipped with an i7-1255U CPU processor.

We assess the effectiveness of the proposed algorithm via the unused potential ratio (UPR), cf. [7]:

$$\text{UPR} := \frac{z_{\text{approx}} - z_{\text{exact, best}}}{z_{\text{exact, worst}} - z_{\text{exact, best}}} \cdot 100, \tag{7}$$

where z_{approx} is the solution of an optimization within the feasible region defined by the proposed algorithm (cf. (5)), $z_{\text{exact, best}}$ denotes the solution within the exact feasibility region computed by a central controller (cf. (6)), and $z_{\text{exact, worst}}$ denotes the solution within the exact feasibility region when the objective is maximized rather than minimized. The two objectives total peak load and energy costs are minimized in the residential area, i.e., $f(x) = \|x + q\|_\infty$, where q represents the total household demand, and $f(x) = c^\top(x + q)$, where c denotes vector of day-ahead prices. The UPR is limited to a range of $0 \leq \text{UPR} \leq 100$. A UPR value near zero suggests that the approximation closely matches the output of the central controller. In contrast, when the UPR nears 100 %, it indicates that a considerable amount of potential within the exact feasibility region is left unused and not represented by the approximation.

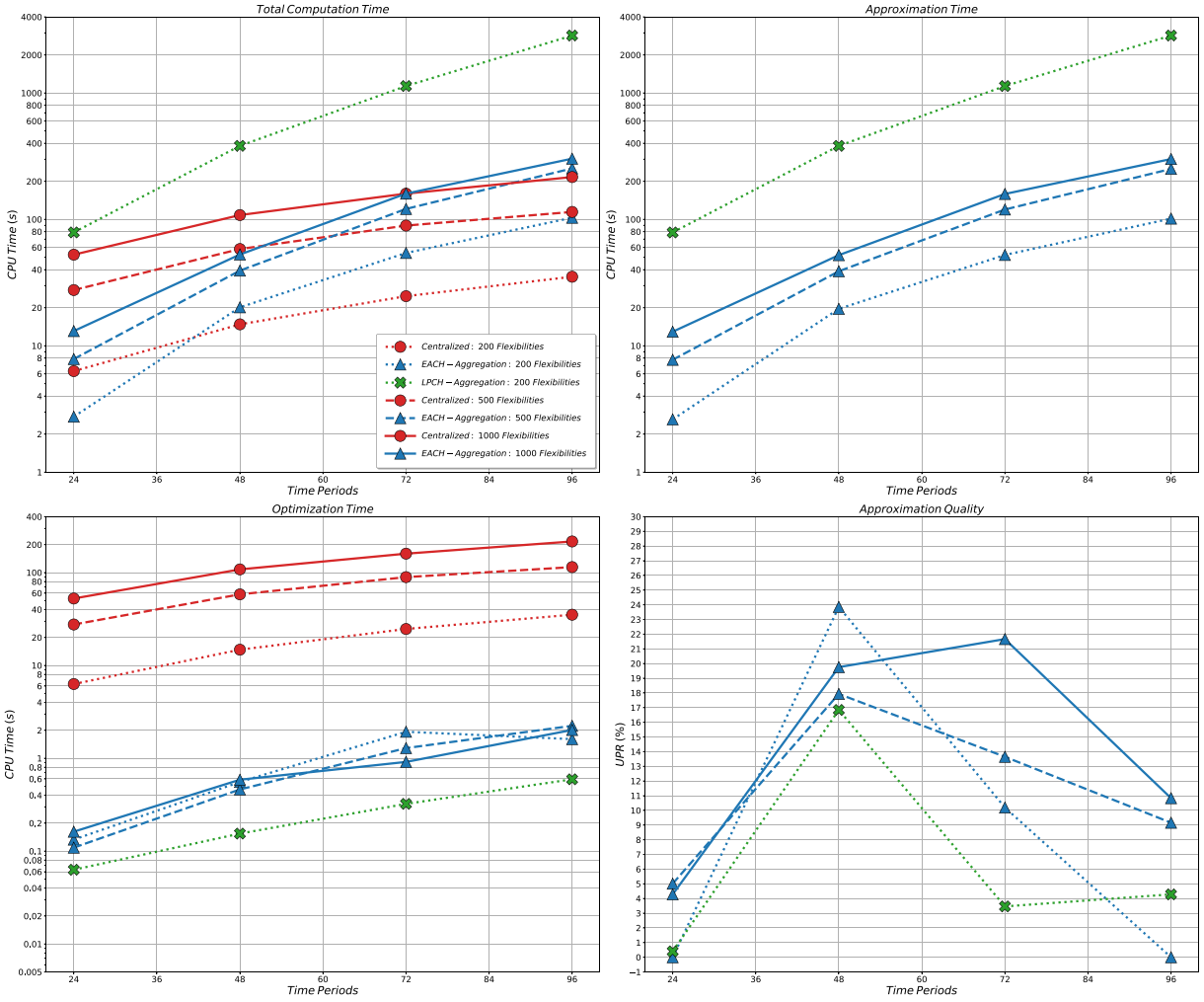


Fig. 8. Quality measures to compare the centralized optimization and the approximation algorithms EACH and LPCH for total peak load minimization. Top left: total computation time. Top right: computation time for the approximation of the aggregate flexibility set. Bottom left: computation time for the subsequent optimization. Bottom right: approximation quality measured by UPR, cf. [7].

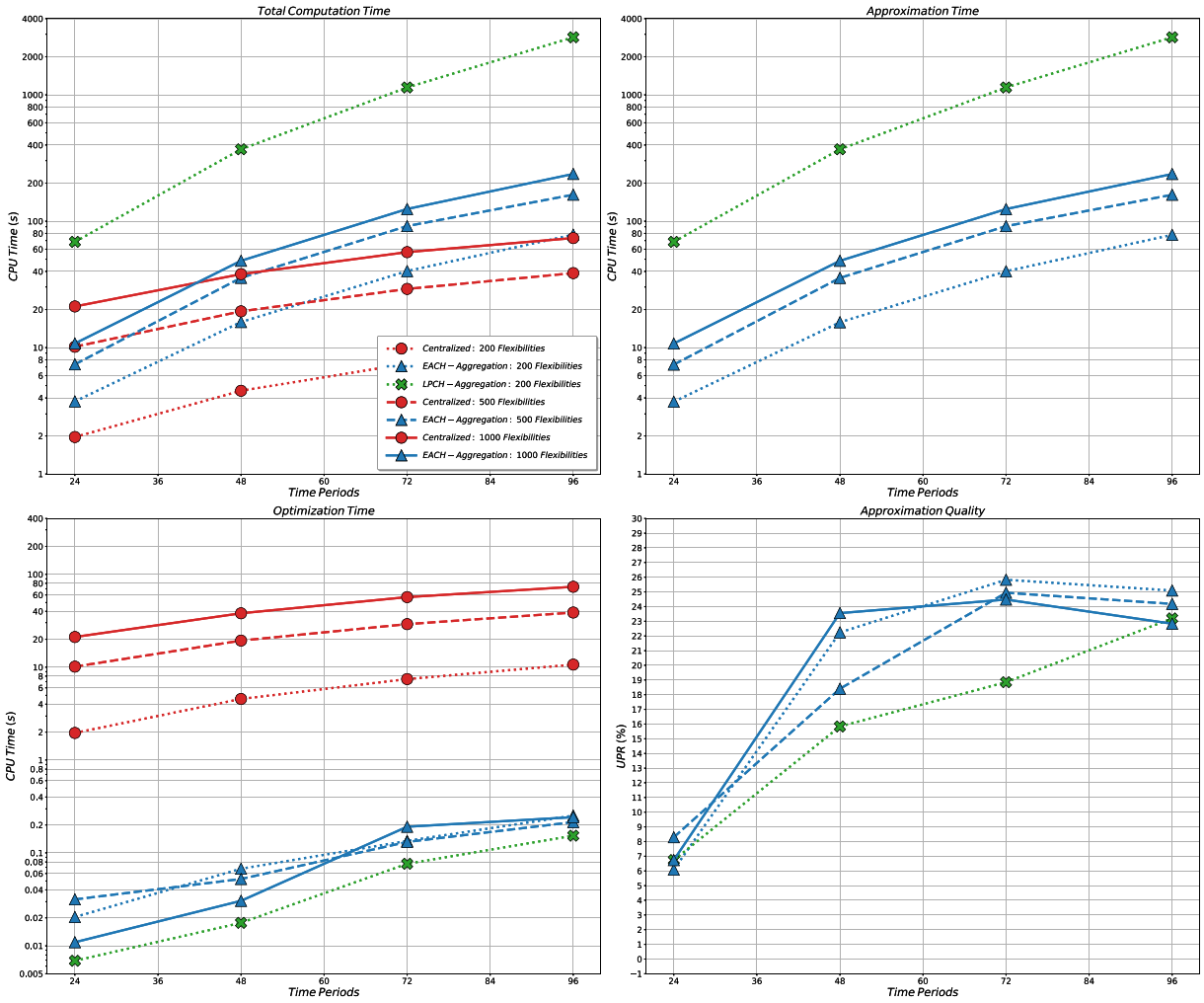


Fig. 9. Quality measures to compare the centralized optimization and the approximation algorithms EACH and LPCH for cost minimization. Top left: total computation time. Top right: computation time for the approximation of the aggregate flexibility set. Bottom left: computation time for the subsequent optimization. Bottom right: approximation quality measured by UPR, cf. [7].

In our experiments, we examine tuples $(n, d) \in \{200, 500, 1000\} \times \{24, 48, 72, 96\}$, representing a maximum of 1000 devices and up to 96 time periods. To account for variability, for each tuple (n, d) we compute median values of five random scenarios. Furthermore, a quadratic time dependence is employed for the set of extreme directions ($|\mathcal{J}| = 2^d$), as described in [25].

The UPR values and CPU times are shown in Fig. 8 and Fig. 9 for the two objectives, respectively. The total computation times of the central controller and the EACH method are comparable. Note, however, that the computations of the individual extreme actions (approximation time) were done serially and could be implemented more efficiently in parallel. The times for computing the optimization were decreased by both aggregation methods and do not depend on the number of flexibilities, as expected. The LPCH method is significantly slower than the EACH method while resulting in similar approximation quality. Due to the extensive aggregation time, we computed LPCH results only for 200 flexibilities. Note that, unlike the proposed aggregation EACH method, the central controller requires complete information and manages all devices directly. While this enables the use of the exact feasible region, it also introduces privacy concerns and excessive communication overhead. Furthermore, these experiments demonstrate the scalability of our EACH method across different time periods and devices, addressing a well-documented issue with current approximation techniques [7].

5. Discussion and conclusion

In this study, we introduced the inner approximation method EACH for flexibility aggregation which is applicable to a broad class of DERs with varying energy and power limits. The proposed scheme extends our previous findings from Ref. [25]. The accompanying open-source Python package includes the source code and practical scripts that demonstrate the capabilities of our approach, showcasing different use cases. The effectiveness of the aggregation method EACH was assessed through a case study and a comparison with a centralized control scheme and the closely related LP-based method LPCH.

The results indicate that with EACH the approximate aggregate flexibility of multiple appliances can be efficiently computed and used to reduce peak power demands in residential settings. While resulting in similar approximation quality, the LPCH method is significantly slower than the EACH method, even when using a fast commercial LP solver.

The strengths of the EACH method can be summarized as follows: The applicable storage models are allowed to have time-dependent bounds for power and energy. The method is therefore applicable to a very general modeling settings including EVs, TCLs, and BESS. The EACH method has very good scaling characteristics concerning the number of time periods and the number of devices. The method does not require a (commercial) LP solver and can be implemented without any

additional software exploiting parallelization. Hierarchical aggregation and disaggregation are covered by simple vector-matrix operations.

Although the central controller achieved better peak demand reduction compared to the inner approximations, it serves only as a benchmark due to its impracticality in real-world applications. The centralized approach faces limitations related to data security and high communication demands in large-scale implementations. In contrast, aggregation mitigates these issues by protecting data and significantly reducing communication overhead, as only the aggregator interacts with the purchaser of the aggregate flexibility. Thus, the presented inner approximation EACH provides a viable alternative for controlling numerous storage-like loads, offering significant advantages in data security and communication efficiency. Its scalability across different time periods and devices demonstrates its applicability in various contexts.

The FERC order 2222 [41] introduces requirements for DER aggregation and participation in wholesale markets. The representation of flexibility by the convex hull of a set of extreme actions, e.g. vertices, offers a simple means to (dis-)aggregate and communicate DER flexibility. To this end, one has to specify the matrices containing the extreme actions as columns, cf. Fig. 3. However, in order to standardize this representation one also needs to select which extreme directions $j \in \mathcal{J}$ are used. To do this in a deterministic and/or optimized way is an interesting task for future work.

We want to note important limitations and some more possible future work directions: We focused on the computational challenges of efficient algorithms for (dis-)aggregation of flexibility which, in our aggregation framework, is performed by an aggregator. The aggregator acts as an intermediary between end-users and system operators and offers the end-users' aggregated flexibility to flexibility markets. The details of the latter are not covered in this work. The aggregation method EACH does not yet consider couplings between the flexibility of individual devices. For the aggregation of EVs, these are given by power constraints of charging stations. Furthermore, the evaluation of the method is not yet done in an entirely realistic setting. In future work, the case study should be extended to a more comprehensive real-world evaluation that also compares the method to other methods in the literature. Future research should also focus on expanding the software to include additional storage types, such as chemical (e.g., hydrogen energy storage) and mechanical (e.g., flywheel energy storage) systems. Additionally, one could try to adapt our method to handle non-convex storage models and address scenarios with uncertain data.

CRediT authorship contribution statement

Emrah Öztürk: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Kevin Kaspar:** Writing – original draft, Validation, Software, Methodology, Investigation. **Timm Faulwasser:** Writing – review & editing, Supervision, Conceptualization. **Karl Worthmann:** Writing – review & editing, Supervision, Conceptualization. **Peter Kepplinger:** Writing – review & editing, Resources, Project administration, Funding acquisition, Conceptualization. **Klaus Rheinberger:** Writing – review & editing, Supervision, Methodology, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships that may be considered as potential competing interests:

Emrah Oeztuerk reports that financial support was provided by Austrian Research Promotion Agency FFG. Kevin Kaspar reports that financial support was provided by Austrian Research Promotion Agency FFG. Peter Kepplinger reports that financial support was provided by Austrian Research Promotion Agency FFG. Klaus Rheinberger reports that financial support was provided by Austrian Research Promotion Agency FFG. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge the financial support from the Austrian Research Promotion Agency FFG for the Hub4FlECs project (COIN FFG 898053). Parts of this work have been funded by the project "EBusCharge" [FFG, No. 899915]. Parts of this work have been presented at the NEIS 2024-Conference on Sustainable Energy and Storage Systems, cf. [42] for a preprint version.

Data availability

We have shared the links to data and code in the References section.

References

[1] IEA (2024), Iea, electricity 2024, https://www.iea.org/reports/electricity-2024.
[2] D.S. Callaway, I.A. Hiskens, Achieving controllability of electric loads, Proc. IEEE 99 (1) (2011) 184–199, https://doi.org/10.1109/JPROC.2010.2081652, https://ieeexplore.ieee.org/abstract/document/5643088.
[3] L. Gkatzikis, I. Koutsopoulos, T. Salonidis, The role of aggregators in smart grid demand response markets, IEEE J. Sel. Areas Commun. 31 (7) (2013) 1247–1257, https://doi.org/10.1109/JSAC.2013.130708
[4] F. Plaum, R. Ahmadiahangar, A. Rosin, J. Kilter, Aggregated demand-side energy flexibility: a comprehensive review on characterization, forecasting and market prospects, Energy Rep. 8 (2022) 9344–9362, https://doi.org/10.1016/j.egyr.2022.07.038, https://www.sciencedirect.com/science/article/pii/S2352484722012999.
[5] K. Mukhi, G. Loho, A. Abate, Exact characterization of Aggregate flexibility via generalized polymatroids, arXiv:2503.23458 [eess] (2025), https://doi.org/10.48550/arXiv.2503.23458 (Mar.) http://arxiv.org/abs/2503.23458.
[6] H.R. Tiwary, On the hardness of computing intersection, union and Minkowski sum of polytopes, Discrete Comput. Geom. 40 (3) (2008) 469–479, https://doi.org/10.1007/s00454-008-9097-3
[7] E. Öztürk, K. Rheinberger, T. Faulwasser, K. Worthmann, M. Preißinger, Aggregation of demand-side flexibilities: a comparative study of approximation algorithms, Energies 15 (7) (2022) 2501, https://doi.org/10.3390/en15072501
[8] S. Barot, J.A. Taylor, An outer approximation of the Minkowski sum of convex conic sets with application to demand response, in: 2016 IEEE 55th Conference on Decision and Control (CDC), IEEE, 2016, pp. 4233–4238, https://doi.org/10.1109/CDC.2016.7798912
[9] S. Barot, J.A. Taylor, A concise, approximate representation of a collection of loads described by polytopes, Int. J. Electr. Power Energy Syst. 84 (2017) 55–63, https://doi.org/10.1016/j.ijepes.2016.05.001
[10] He Hao, B.M. Sanandaji, K. Poolla, T.L. Vincent, A generalized battery model of a collection of thermostatically controlled loads for providing ancillary service, in: 2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton), IEEE, 2013, pp. 551–558, https://doi.org/10.1109/Allerton.2013.6736573
[11] L. Zhao, W. Zhang, H. Hao, K. Kalsi, A geometric approach to aggregate flexibility modeling of thermostatically controlled loads, IEEE Trans. Power Syst. 32 (6) (2017) 4721–4731, https://doi.org/10.1109/TPWRS.2017.2674699
[12] H. Hao, B.M. Sanandaji, K. Poolla, T.L. Vincent, Aggregate flexibility of thermostatically controlled loads, IEEE Trans. Power Syst. 30 (1) (2015) 189–198, https://doi.org/10.1109/TPWRS.2014.2328865
[13] Y. Wen, Z. Hu, S. You, X. Duan, Aggregate feasible region of DERs: exact formulation and approximate models, IEEE Trans. Smart Grid 13 (6) (2022) 4405–4423, https://doi.org/10.1109/TSG.2022.3179998
[14] F.L. Müller, O. Sundström, J. Szabó, J. Lygeros, Aggregation of energetic flexibility using zonotopes, in: 2015 54th IEEE Conference on Decision and Control (CDC), 2015, pp. 6564–6569, https://doi.org/10.1109/CDC.2015.7403253
[15] F.L. Müller, J. Szabó, O. Sundström, J. Lygeros, Aggregation and disaggregation of energetic flexibility from distributed energy resources, IEEE Trans. Smart Grid 10 (2) (2019) 1205–1214, https://doi.org/10.1109/TSG.2017.2761439
[16] M.S. Nazir, I.A. Hiskens, A. Bernstein, E. Dall'anese, Inner approximation of Minkowski sums: a union-based approach and applications to aggregated energy resources, in: 2018 IEEE Conference on Decision and Control (CDC), IEEE, 2018, pp. 5708–5715, https://doi.org/10.1109/CDC.2018.8618731
[17] L. Zhao, H. Hao, W. Zhang, Extracting flexibility of heterogeneous deferrable loads via polytopic projection approximation, in: 2016 IEEE 55th Conference on Decision and Control (CDC), 2016, pp. 6651–6656, https://doi.org/10.1109/CDC.2016.7799293
[18] S. Barot, Aggregate Load Modeling for Demand Response via the Minkowski Sum, Thesis, University of Toronto, accepted: 2017-11-01T20:00:44Z (2017 (Jun.). https://tspace.library.utoronto.ca/handle/1807/78943.
[19] J. Zhen, D. den Hertog, Computing the maximum volume inscribed ellipsoid of a polytopic projection, INFORMS J. Comput. 30 (1) (2018) 31–42, https://doi.org/10.1287/ijoc.2017.0763
[20] R.R. Appino, V. Hagenmeyer, T. Faulwasser, Towards optimality preserving aggregation for scheduling distributed energy resources, IEEE Trans. Control Netw. Syst. 8 (3) (2021) 1477–1488, https://doi.org/10.1109/TCNS.2021.3070664
[21] F. Al Taha, T. Vincent, E. Bitar, A multi-battery model for the aggregate flexibility of heterogeneous electric vehicles, in: 2023 American Control Conference (ACC), IEEE, 2023, pp. 1243–1250, https://doi.org/10.23919/ACC55779.2023.10156328

[22] M. Zhang, Y. Xu, X. Shi, Q. Guo, A fast polytope-based approach for aggregating large-scale electric vehicles in the joint market under uncertainty, IEEE Trans. Smart Grid 15 (1) (2024) 701–713, https://doi.org/10.1109/TSG.2023.3274198
[23] J. Jian, M. Zhang, Y. Xu, W. Tang, S. He, An analytical polytope approximation aggregation of electric vehicles considering uncertainty for the day-ahead distribution network dispatching, IEEE Trans. Sustain. Energy 15 (1) (2024) 160–172, https://doi.org/10.1109/TSTE.2023.3275566
[24] K. Mukhi, A. Abate, An exact characterisation of flexibility in populations of electric vehicles, arXiv:2306.16824 [cs, eess] (Jun.2023).
[25] E. Öztürk, T. Faulwasser, K. Worthmann, M. Preißinger, K. Rheinberger, Alleviating the curse of dimensionality in Minkowski sum approximations of storage flexibility, IEEE Trans. Smart Grid 15 (6) (2024) 5733–5743, https://doi.org/10.1109/TSG.2024.3420156
[26] D. Pozo, Convex hull formulations for linear modeling of energy storage systems, IEEE Trans. Power Syst. 38 (6) (2023) 5934–5936, https://doi.org/10.1109/TPWRS.2023.3304131
[27] S. Sass, T. Faulwasser, D.E. Hollermann, C.D. Kappatou, D. Sauer, T. Schütz, D.Y. Shu, A. Bardow, L. Gröll, V. Hagenmeyer, et al., Model compendium, data, and optimization benchmarks for sector-coupled energy systems, Comput. Chem. Eng. 135 (2020) 106760.
[28] J. Giesecke, S. Heimerl, Wasserkraftanlagen: Planung, Bau Und Betrieb, Springer, 2014, https://doi.org/10.1007/978-3-642-53871-1
[29] B. Zhou, S. Liu, S. Lu, X. Cao, W. Zhao, Cost–benefit analysis of pumped hydro storage using improved probabilistic production simulation method, J. Eng. 2017 (13) (2017) 2146–2151, https://doi.org/10.1049/joe.2017.0709
[30] G.M. Ziegler, Lectures on Polytopes, Graduate Texts in Mathematics, vol. 152 of Springer, 1995, https://doi.org/10.1007/978-1-4613-8431-1
[31] K. Kaspar, The Python Package Index. PyFlexAD: python flexibility aggregation and disaggregation, https://pypi.org/project/pyflexad/.
[32] K. Kaspar, GitHub of the Energy Research Centre - FH Vorarlberg University of Applied Sciences. Repository PyFlexAD, https://github.com/rce-fhv/PyFlexAD.
[33] pytope, https://pypi.org/project/pytope/.
[34] OpenTUMFlex, https://opentumflex.readthedocs.io/en/latest/.
[35] DSM-data, https://github.com/klaus-rheinberger/DSM-data.
[36] ENTSO-E, Transparency platform - day-ahead market prices, https://transparency.entsoe.eu/.
[37] E.A. Technology, My electric avenue project, 2016 (2016) https://eatechnology.com/resources/projects/my-electric-avenue-data-download/.
[38] I.S.S.D.A.-I.S.S.D. Archive, Cer smart metering project – electricity customer behaviour trial, 2009–2010, https://www.ucd.ie/issda/.
[39] My electric avenue, https://myelectricavenue.info/.
[40] Gurobi Optimization, LLC, Gurobi optimizer reference manual, https://www.gurobi.com.
[41] Eac FERC order 2222 recommendations_approved, https://www.energy.gov/sites/default/files/2021-04/EAC%20FERC%20Order%202222%20Recommendations_Approved.pdf.
[42] E. Öztürk, K. Kaspar, T. Faulwasser, K. Worthmann, P. Kepplinger, K. Rheinberger, Towards efficient Aggregation of storage flexibilities in power grids, arXiv:2403.20104 (Mar.2024).