BATTPOWER Toolbox: Memory-Efficient and High-Performance Multi-Period AC Optimal Power Flow Solver

Salman Zaferanlouei, *Student Member, IEEE*, Hossein Farahmand, *Senior Member, IEEE*, Vijay Venu Vadlamudi, *Member, IEEE*, and Magnus Korpås, *Member, IEEE*

***Abstract*—With the introduction of massive renewable energy sources and storage devices, the traditional process of grid operation must be improved in order to be safe, reliable, fast responsive and cost efficient, and in this regard power flow solvers are indispensable. In this paper, we introduce an Interior Point-based (IP) Multi-Period AC Optimal Power Flow (MPOPF) solver for the integration of Stationary Energy Storage Systems (SESS) and Electric Vehicles (EV). The primary methodology is based on: 1) analytic and exact calculation of partial differential equations of the Lagrangian sub-problem, and 2) exploiting the sparse structure and pattern of the coefficient matrix of Newton-Raphson approach in the IP algorithm. Extensive results of the application of proposed methods on several benchmark test systems are presented and elaborated, where the advantages and disadvantages of different existing algorithms for the solution of MPOPF, from the standpoint of computational efficiency, are brought forward. We compare the computational performance of the proposed Schur-Complement algorithm with a direct sparse LU solver. The comparison is performed for two different applicational purposes: SESS and EV. The results suggest the substantial computational performance of Schur-Complement algorithm in comparison with that of a direct LU solver when the number of storage devices and optimisation horizon increase for both cases of SESS and EV. Also, some situations where computational performance is inferior are discussed.**

***Index Terms*—multi-period ACOPF, interior point method, energy storage systems.**

I. Introduction

LARGE-SCALE introduction of Renewable Energy Sources (RES), Stationary Energy Storage Systems (SESS) and Electric Vehicles (EV) will influence the way the electricity grid is operated. In the planning and operation of the electricity network, power flow analysis toolboxes that are reliable, computationally fast, and tractable are indispensable.

The optimal power flow is a non-linear and non-convex problem which was introduced in the sixties [1] for the first time. Although it is considered to be a classic power systems problem among researchers, depending on the technical applications and operational dimensions, it may be adapted to various versions such as the Multi-Period AC Optimal Power Flow (MPOPF), introduced by [2], and may become intractable and computationally very demanding even after about 60 years [3], [4].

Many researchers have been trying to either simplify MPOPF by linearising the main problem [5]–[7], or by making it more reliable by finding the global optimum point with different convex relation approaches such as [8], semidefinite programming (SDP) relaxations [9], [10] and second-order cone programming [11]. Moreover, MPOPF is being suggested as a potential online operational tool [12] for use in the near future.

Since the non-linear nature of full ACOPF problems requires non-linear solvers to be called, several Non-Linear Programming (NLP) solvers are used to solve MPOPF problems; these are primarily developed based on IP methods, such as MIPS[1] [13], IPOPT* [14] KNITRO* [15], and recently BELTISTOS* [16], of which the only tailored algorithm to solve MPOPF problems is BELTISTOS. An extensive review of both MPOPF problem formulations and solution methods can be found in [3], [17]. With more and more penetration of renewables into the system, it becomes imperative to rely on SESS to overcome the variability of these renewable energy resources; MPOPF becomes a pivotal tool in this context for system operators, and as such better solvers need to be designed from the point of view of computational ease and efficiency. Though reliability is the most critical factor when it comes to the global optimum solution, computational performance is pivotal for the implementation and application of an algorithm for online operational purposes. It is well-known from the literature [18], [19] that solution of linear Karush-Kuhn-Tucker (KKT) systems and calculation of gradients are the two most computationally expensive aspects in solving a MPOPF problem. Thus, we propose a solver to exploit the sparsity of a MPOPF structure (both KKT systems and gradients) and speed the solution up.

Although considerable efforts have been undertaken in order to solve the MPOPF problem, no currently available work extensively elaborates the complex mathematical details of the problem, because of which it is difficult to compare the various solution algorithms in a systematic (same platform, same

Manuscript received February 28, 2020; revised August 7, 2020 and December 23, 2020; accepted January 16, 2021. Date of publication January 28, 2021; date of current version August 19, 2021. Paper no. TPWRS-00312-2020. *(Corresponding author: Magnus Korpås.)*

The authors are with the Department of Electric Power Engineering, Norwegian University of Science, and Technology, Trondheim 7491, Norway (e-mail: salman.zaf@ntnu.no Student Member; hossein.farahmand@ntnu.no; vijay.vadlamudi@ntnu.no; magnus.korpas@ntnu.no).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TPWRS.2021.3055429.

Digital Object Identifier 10.1109/TPWRS.2021.3055429

[1] a solver name

programming language, single-thread environment) manner. MATPOWER [13] does have a complete implementation of a single-period ACOPF with function evaluations and solution of KKT systems with MIPS. However, there is no package for the MPOPF function evaluations and for handling the sparsity structure of the first and second gradients, as introduced in this proposed package. Reference [16] introduced a fast multi-period KKT solver, Beltistos, as an extension of the IPOPT solver, and specifically for the MPOPF KKT structure, using the PARDISO solver. However, reference [16] does not: 1) consider the sparsity structure of first and second gradients, and 2) discuss the dynamic behaviour of energy storage systems in two different forms of SESS and EV in MPOPF. Moreover, there are several ways to model storage devices within a time-period. Meyer-Huebner *et al.* [20] formulated three different storage device models as: (A) an inequality for the entire optimisation horizon, (B) an inequality and a variable for the entire optimisation horizon, and (C) T number of equalities for each timestamp within the entire optimisation horizon. They concluded that option (C) has more efficient computational performance than options (A) and (B); therefore, we use option (C) formulation to apply the most efficient mathematical formulation with respect to the implementation of storage devices. Note that option (A) has recently been implemented in the Beltistos solver [16]. Finally, the size (number of rows and columns) of KKT systems considered to be solved in [16] is almost double the size of KKT systems we solve here. Reference [21] has developed a computationally fast solution to the MPOPF problem and applied it on a small-scale SESS and test system. Here, in this paper, we expand our work in [21], and propose a new solver for large-scale integration of SESS and EV with consideration of full AC power flow equations.

The main contributions of our paper could be summarised as follows: a) New input matrices are introduced in order to capture the full dynamic of a multi-period system including SESS and EV. b) The first and second analytical (hand-coded) derivatives of linear and non-linear equality constraints, inequality constraints and objective function, w.r.t. variables are extracted in the multi-period form. c) Sparsity structure of analytical derivatives is extracted in order to increase the performance in terms of memory requirements and sparsity calculations in different loops. d) A new re-ordering format is introduced to exploit and reveal the multi-period structure of KKT systems for both SESS and EV. e) A high-performance and memory-efficient sparse Schur-Complement algorithm is introduced in order to solve the multi-period structure of the KKT systems for both SESS and EV. In this paper, we compare the Schur-Complement algorithm, tailored for a specific structure, with a direct sparse LU solver to shed light into a systematic comparison (same implementation platform (MATLAB), single-thread controlled environment, similar PC) to only compare the algorithmic complexity differences.

The structure of this paper is as follows: in the next section, the formulation of MPOPF problem in the presence of SESS and EV is elaborated. The solution proposal, and the mathematical algorithms for speeding up the solution proposal, are presented in the subsequent sections. Next, the performance of the Schur-Complement algorithm is compared with that of a direct sparse LU solver for multiple numbers of storage devices and time horizons for two different cases: performance of SESS on standard mesh transmission benchmarks, and the performance of EV on radial distribution benchmarks. Finally, we summarise the paper in the last section with some concluding remarks.

II. Formulation of Proposed Solver[2]

In general, a given power system can be represented with the following input matrices in ACOPF: $\mathbf{BUS}$ matrix with $n_b \in \mathbb{N}$ number of buses. These buses are connected to each other through the total number of $n_l \in \mathbb{N}$ lines represented by the $\mathbf{BRANCH}$ matrix. The matrix consists of connecting buses, i.e., $\mathbf{BUS}^{\text{from}} \in \mathbb{R}^{n_l \times 1}$ and $\mathbf{BUS}^{\text{to}} \in \mathbb{R}^{n_l \times 1}$ and physical line parameters $\in \mathbb{R}^{n_l \times 1}$, such as resistance, reactance, susceptance and their apparent power capacities (MVA). The $\mathbf{GEN}$ matrix specifies the connection bus of $n_g \in \mathbb{N}$ generators with their dispatch limits and voltage references. The $\mathbf{GENCOST}$ matrix is the associated cost functions of generators in $\mathbf{GEN}$. These matrices have been already defined by MATPOWER [22] and are now extensively used by many researchers in the subjects of power economics and power system analysis.

Here, in addition to the above input matrices, we propose new input matrices,[3] which can be seen in Appendix A of this paper. These new input matrices are designed for the integration of large-scale SESS and EV to capture the dynamic behaviour of storage devices over the optimisation horizon. The input $\mathbf{BATT}$ represents the properties of energy storage systems and more importantly ties single-period optimal power flow equations through a positive integer parameter, called time period, $T \in \mathbb{N}$; t is a time in the interval of $t \in \{1, \ldots, T\}$. The differences between the representation of a SESS and an EV are concentrated in the binary input matrices of $\mathbf{AVBP}$, $\mathbf{CONCH}$, $\mathbf{CONDI}$ and $\mathbf{AVBQ}$. Storage i is considered stationary if the availability condition (1a) holds, otherwise it has a dynamic behaviour over time and can be considered as an EV, while charge, discharge and reactive power provision conditions could be considered as secondary or optional conditions (1b)–(1d).

$$\mathbf{AVBP}\big|_{i,t=1} = \mathbf{AVBP}\big|_{i,t=2} = \cdots = \mathbf{AVBP}\big|_{i,t=T} = 1 \tag{1a}$$

$$\mathbf{CONCH}\big|_{i,t=1} = \mathbf{CONCH}\big|_{i,t=2} = \cdots = \mathbf{CONCH}\big|_{i,t=T} = 1 \tag{1b}$$

$$\mathbf{CONDI}\big|_{i,t=1} = \mathbf{CONDI}\big|_{i,t=2} = \cdots = \mathbf{CONDI}\big|_{i,t=T} = 1 \tag{1c}$$

$$\mathbf{AVBQ}\big|_{i,t=1} = \mathbf{AVBQ}\big|_{i,t=2} = \cdots = \mathbf{AVBQ}\big|_{i,t=T} = 1 \tag{1d}$$

In order to elaborate the mathematical formulation of the MPOPF incorporating the $\mathbf{BATT}$ matrix, we first present single-period power flow equations for a time t, and subsequently expand the equations to represent MPOPF.

[2]Note that all vectors and matrices are shown with bold and non-italic notation: $\mathbf{BOLD}$.

[3]Five non-binary and five binary matrices.

Considering the above types of inputs, we have an object-oriented programming (OOP) package that constructs different types of constraints according to the input matrices and feeds the mathematical formulation of the MPOPF to the solver designed under this package.

A. Single-Period Optimal Power Flow

With the input matrices introduced above, **BUS**, **BRANCH**, **GEN** and **GENCOST**, we show the mathematical formulation of a single-period ACOPF in this subsection.

Consider the vector of complex bus voltages in rectangular coordinates as illustrated by $\underline{\mathbf{V}} \in \mathbb{C}^{n_b \times 1}$, where $\mathbb{C}$ is a complex set. The voltage vector comprises complex elements as: $\underline{\mathrm{v}}_i = |\mathrm{v}_i| e^{j\theta_i}$, where $\underline{\mathrm{v}}_i \in \mathbb{C}$, $\{v_i, \theta_i\} \in \mathbb{R}$ are the voltage magnitude and angle of the corresponding bus in polar coordinates, where $\mathbb{R}$ is a real set. Moreover, $\{\boldsymbol{\mathcal{V}}, \boldsymbol{\Theta}\} \in \mathbb{R}^{n_b \times 1}$ can be defined as vectors of real magnitude and angle of bus voltages. In vector form, the relationship between rectangular and polar coordinates is shown as:

$$\underline{\mathbf{V}} = \mathbf{diag}(\boldsymbol{\mathcal{V}})\ \exp(j\boldsymbol{\Theta}) \tag{2}$$

Line Connectivity matrices of $\{\mathbf{C}^{\mathrm{fr}}, \mathbf{C}^{\mathrm{to}}\} \in \mathbb{B}^{n_l \times n_b}$ can be extracted from $\mathbf{BUS}^{\mathrm{from}}$ and $\mathbf{BUS}^{\mathrm{to}}$ vectors, such that $\mathrm{c}^{\mathrm{fr}}_{ik} = 1$ if bus k is connected to line i, and otherwise $\mathrm{c}^{\mathrm{fr}}_{ik} = 0$, and the same holds for $\mathbf{C}^{\mathrm{to}}$. $\{\underline{\mathbf{V}}^{\mathrm{fr}},\ \underline{\mathbf{V}}^{\mathrm{to}}\} \in \mathbb{C}^{n_l \times 1}$ are the vectors of complex bus voltages at line terminals, including "from" and "to" nodes, correspondingly. These vectors can be extracted using the connectivity matrices explained above shown in Eqs. (3) and (4).

$$\underline{\mathbf{V}}^{\mathrm{fr}} = \mathbf{C}^{\mathrm{fr}} \underline{\mathbf{V}} \tag{3}$$

$$-3pt]\underline{\mathbf{V}}^{\mathrm{to}} = \mathbf{C}^{\mathrm{to}} \underline{\mathbf{V}} \tag{4}$$

and therefore:

$$\underline{\mathbf{V}}^{\mathrm{Line}} = \begin{bmatrix} \underline{\mathbf{V}}^{\mathrm{fr}} \\ \underline{\mathbf{V}}^{\mathrm{to}} \end{bmatrix}_{2n_l \times 1} = \overbrace{\begin{bmatrix} \mathbf{C}^{\mathrm{fr}} \\ \mathbf{C}^{\mathrm{to}} \end{bmatrix}_{2n_l \times n_b}}^{\mathbf{C}^{\mathrm{Line}}} \underline{\mathbf{V}} \tag{5}$$

In order to obtain the entire network flow, the vector of complex voltages $\underline{\mathbf{V}}$ has to be determined. This can be done using the well-known Kirchhoff's current law: the sum of external current injections at a bus $\underline{\mathbf{I}}^{\mathrm{bus}} \in \mathbb{C}^{n_b \times 1}$ is equal to the sum of internal - through lines - current injections to the same bus $\underline{\mathbf{I}}^{\mathrm{bus}} = \underline{\mathbf{Y}}^{\mathrm{bus}} \underline{\mathbf{V}}$, where $\underline{\mathbf{Y}}^{\mathrm{bus}} \in \mathbb{C}^{n_b \times n_b}$ is the bus admittance matrix. The same principle is applied to compute the complex line current using complex bus voltages of line terminals, and line admittance matrix $\underline{\mathbf{Y}}^{\mathrm{Line}} \in \mathbb{C}^{2n_l \times n_b}$. This is shown in (6)

$$\underline{\mathbf{I}}^{\mathrm{Line}} = \begin{bmatrix} \underline{\mathbf{I}}^{\mathrm{fr}} \\ \underline{\mathbf{I}}^{\mathrm{to}} \end{bmatrix}_{2n_l \times 1} = \overbrace{\begin{bmatrix} \underline{\mathbf{Y}}^{\mathrm{fr}} \\ \underline{\mathbf{Y}}^{\mathrm{to}} \end{bmatrix}_{2n_l \times n_b}}^{\underline{\mathbf{Y}}^{\mathrm{Line}}} \underline{\mathbf{V}} \tag{6}$$

The relation between bus admittance and line admittance matrices is defined by (7).

$$\underline{\mathbf{Y}}^{\mathrm{bus}} = (\mathbf{C}^{\mathrm{fr}})^{\top} \underline{\mathbf{Y}}^{\mathrm{fr}} + (\mathbf{C}^{\mathrm{to}})^{\top} \underline{\mathbf{Y}}^{\mathrm{to}} + \underline{\mathbf{Y}}^{\mathrm{shunt}} \tag{7}$$

$\{\underline{\mathbf{Y}}^{\mathrm{fr}}, \underline{\mathbf{Y}}^{\mathrm{to}}\} \in \mathbb{C}^{n_l \times n_b}$, and $\underline{\mathbf{Y}}^{\mathrm{shunt}} \in \mathbb{C}^{n_b \times n_b}$ is the matrix of shunt admittance. Finally, the external complex power injections into a bus i can be computed as $\underline{\mathrm{s}}^{\mathrm{bus}}_i = \underline{\mathrm{v}}_i (\underline{\mathrm{i}}^{\mathrm{bus}}_i)^*$, whereas the complex power flow over a line at the terminal k can be calculated by $\underline{\mathrm{s}}^{\mathrm{Line}}_k = (\mathbf{C}^{\mathrm{Line}}_k \underline{\mathbf{V}})(\underline{\mathrm{i}}^{\mathrm{Line}}_k)^*$, where $\{\underline{\mathrm{s}}^{\mathrm{bus}}_i, \underline{\mathrm{i}}^{\mathrm{bus}}_i, \underline{\mathrm{s}}^{\mathrm{Line}}_k, \underline{\mathrm{i}}^{\mathrm{Line}}_i\} \in \mathbb{C}$ and $\mathbf{C}^{\mathrm{Line}}_k \in \mathbb{B}^{1 \times n_b}$ is the k^{th} element of $\mathbf{C}^{\mathrm{Line}}$ matrix. In summary, power injections into a bus and into a line can be extended in the form of vectors using (8).

$$\underline{\mathbf{S}}^{\mathrm{bus}} = \mathbf{diag}(\underline{\mathbf{V}})(\underline{\mathbf{I}}^{\mathrm{bus}})^* \ \in \ \mathbb{C}^{n_b \times 1} \tag{8}$$

$$\underline{\mathbf{S}}^{\mathrm{Line}} = \mathbf{diag}(\underline{\mathbf{V}}^{\mathrm{Line}})(\underline{\mathbf{I}}^{\mathrm{Line}})^* \ \in \ \mathbb{C}^{2n_l \times 1} \tag{9}$$

Here we define generator connectivity matrix $\mathbf{C}^{\mathrm{g}} \in \mathbb{B}^{n_b \times n_g}$ which is a binary matrix of 0 and 1. $\mathbf{C}^{\mathrm{g}}_{ik} = 1$ if generator i is connected to the bus k and otherwise 0. $\underline{\mathbf{S}}^{\mathrm{g}} \in \mathbb{C}^{n_g \times 1}$ and $\underline{\mathbf{S}}^{\mathrm{d}} \in \mathbb{C}^{n_b \times 1}$ are the complex vectors of generation units and loads. The first set of non-linear equality constraints for the ACOPF problem, can be defined as:

$$\widetilde{\mathbf{g}}(\mathbf{x}) = \underline{\mathbf{S}}^{\mathrm{bus}} + \underline{\mathbf{S}}^{\mathrm{d}} - \mathbf{C}^{\mathrm{g}} \underline{\mathbf{S}}^{\mathrm{g}} = 0 \tag{10}$$

In this article, $\sim$ is the sign for a non-linear equation. In the literature on power systems, (10) has been divided into two parts, power balance of active and reactive power as indicated in Eqs. (11a) and (11b), respectively. Therefore, non-linear equality constraints take the form of $\widetilde{\mathbf{g}}(x) \in \mathbb{R}^{n_{gn} \times 1}$ and $n_{gn} = 2n_b$.

$$\mathbf{C}^{\mathrm{g}} \boldsymbol{\mathcal{P}}^{\mathrm{g}} - \boldsymbol{\mathcal{P}}^{\mathrm{d}} = \Re[\mathbf{C}^{\mathrm{g}} \underline{\mathbf{S}}^{\mathrm{g}}] - \Re[\underline{\mathbf{S}}^{\mathrm{d}}] = \Re[\underline{\mathbf{S}}^{\mathrm{bus}}] \tag{11a}$$

$$\mathbf{C}^{\mathrm{g}} \boldsymbol{\mathcal{Q}}^{\mathrm{g}} - \boldsymbol{\mathcal{Q}}^{\mathrm{d}} = \Im[\mathbf{C}^{\mathrm{g}} \underline{\mathbf{S}}^{\mathrm{g}}] - \Im[\underline{\mathbf{S}}^{\mathrm{d}}] = \Im[\underline{\mathbf{S}}^{\mathrm{bus}}] \tag{11b}$$

where $\{\boldsymbol{\mathcal{P}}^{\mathrm{g}}, \boldsymbol{\mathcal{Q}}^{\mathrm{g}}\} \in \mathbb{R}^{n_g \times 1}$ are the vectors of active and reactive power generations, $\{\boldsymbol{\mathcal{P}}^{\mathrm{d}}, \boldsymbol{\mathcal{Q}}^{\mathrm{d}}\} \in \mathbb{R}^{n_b \times 1}$ are the vectors of active and reactive power consumption. If we take $(|\underline{\mathbf{S}}^{\mathrm{Line}}_{\max}|)^2 \in \mathbb{R}^{2n_l \times 1}$ as the squared vector of apparent power flow limits, then apparent line power flow constraint can be defined as:

$$\widetilde{\mathbf{h}}(\mathbf{x}) = \left[(\underline{\mathbf{S}}^{\mathrm{Line}})^* \underline{\mathbf{S}}^{\mathrm{Line}} - (|\underline{\mathbf{S}}^{\mathrm{Line}}_{\max}|)^2\right] \leq 0 \in \mathbb{R}^{n_{hn} \times 1} \tag{12}$$

where $n_{hn} = 2n_l$. Another type of constraint is the linear equality constraint $\overline{\mathbf{g}}(\mathbf{x}) = 0$ where $-$ is the sign for a linear vector. Voltage angle is kept equal to zero $\theta^{\mathrm{slack}} = 0 \ \in \mathbb{R}$ which is a linear equality constraint for the slack bus. The last set of constraints is that of linear inequality, related to upper and lower bounds variables called box constraints [16] with vectors $\boldsymbol{\Theta}^{\min} \leq \boldsymbol{\Theta} \leq \boldsymbol{\Theta}^{\max} \in \mathbb{R}^{(2n_b-1) \times 1}$, $\boldsymbol{\mathcal{V}}^{\min} \leq \boldsymbol{\mathcal{V}} \leq \boldsymbol{\mathcal{V}}^{\max} \in \mathbb{R}^{2n_b \times 1}$, $\{(\boldsymbol{\mathcal{P}}^{\mathrm{g}})^{\min} \leq \boldsymbol{\mathcal{P}}^{\mathrm{g}} \leq (\boldsymbol{\mathcal{P}}^{\mathrm{g}})^{\max}$, $(\boldsymbol{\mathcal{Q}}^{\mathrm{g}})^{\min} \leq \boldsymbol{\mathcal{Q}}^{\mathrm{g}} \leq (\boldsymbol{\mathcal{Q}}^{\mathrm{g}})^{\max}\} \in \mathbb{R}^{2n_g \times 1}$. All the box constraints could be put together and represented with one vector $\overline{\mathbf{h}}(\mathbf{x}) \in \mathbb{R}^{n_{hl} \times 1}$ where $n_{hl} = (4n_b - 1) + 4n_g$.

Thus, if the objective function $f(\mathbf{x})$ is an arbitrary linear or non-linear function related to the cost of power generation, the general optimisation framework could be:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\text{s.t. } \mathbf{g}(\mathbf{x}) = \begin{bmatrix} \widetilde{\mathbf{g}}(\mathbf{x}) \\ \overline{\mathbf{g}}(\mathbf{x}) \end{bmatrix} = 0 \qquad \in \mathbb{R}^{n_{gx} \times 1} \\ &\mathbf{h}(\mathbf{x}) = \begin{bmatrix} \widetilde{\mathbf{h}}(\mathbf{x}) \\ \overline{\mathbf{h}}(\mathbf{x}) \end{bmatrix} \leq 0 \qquad \in \mathbb{R}^{n_{hx} \times 1} \end{aligned} \tag{13}$$

where $n_{gx} = n_{gn} + n_{gl}$[4], $n_{hx} = n_{hn} + n_{hl}$[5], and

$$\mathbf{x} = \left[\boldsymbol{\Theta}\ \boldsymbol{\mathcal{V}}\ \boldsymbol{\mathcal{P}}^{\mathrm{g}}\ \boldsymbol{\mathcal{Q}}^{\mathrm{g}}\right]^{\top} \in \mathbb{R}^{n_x \times 1} \tag{14}$$

and $n_x = 2n_b + 2n_g$. With the discussed input matrices and use of OOP[6], we construct an instance of an object called $\mathbf{MP}$ which incorporates all the introduced network variables and constraints in the formulated problem. In the next subsection, we come up with a new input matrix, and make a generalised case for storage devices.

B. Multi-Period Optimal Power Flow

"$\mathbf{BATT}$" is the main introduced matrix here in this section, to optimally operate an electricity network over a time horizon of T which defines the number of time-steps in the optimisation. Therefore, multiple single-period optimal power flow problems are coupled together over a given time horizon to determine the optimal combined operation schedule for energy storage systems. The coupling constraints over time are introduced by linear equality constraints of storage devices in $e_{i,t} - e_{i,t-1} - \psi_i^{\mathrm{ch}} p_{i,t}^{\mathrm{ch}} \Delta t + \frac{p_{i,t}^{\mathrm{dch}} \Delta t}{\psi_i^{\mathrm{dch}}} = 0$, where $soc_{i,t} = \frac{e_i}{e_i^{\max}}$ and $\{soc,\ p^{\mathrm{ch}},\ p^{\mathrm{ch}}, q^s, e, \psi\} \in \mathbb{R}$ representing storage device variables connected to bus i at time t. The presence of: 1) cost minimisation objective function, and 2) the efficiency of charge and discharge helps to avoid the simultaneous charging and discharging. However, charging and discharging might still occur at the same time in some cases, although this is outside the scope of this paper.

To extend the vector notation, for n_y storage devices, we have:

$$\overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_t) = \mathbf{E}_t - \mathbf{E}_{t-1} - \boldsymbol{\Psi}^{\mathrm{ch}} \boldsymbol{\mathcal{P}}_t^{\mathrm{ch}} \Delta t + \frac{\boldsymbol{\mathcal{P}}_t^{\mathrm{dch}} \Delta t}{\boldsymbol{\Psi}^{\mathrm{dch}}} = 0 \tag{15}$$

where $\boldsymbol{\mathcal{SOC}}_t = \frac{\mathbf{E}_t}{\mathbf{E}^{max}}$ and $\{\boldsymbol{\mathcal{SOC}}_t, \boldsymbol{\mathcal{P}}_t^{\mathrm{ch}}, \boldsymbol{\mathcal{P}}_t^{\mathrm{dch}}, \boldsymbol{\mathcal{Q}}_t^{\mathrm{s}}\} \in \mathbb{R}^{n_y \times 1}$. Moreover, $\boldsymbol{\tau}_t = \{\mathbf{x}_{t-1}, \mathbf{x}_t\}$ and $\mathcal{T} = \{\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_T\} = \{\{\mathbf{x}_1\}, \{\mathbf{x}_1, \mathbf{x}_2\}, \ldots, \{\mathbf{x}_{T-1}, \mathbf{x}_T\}\} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, thus $\overline{\mathbf{G}}^{s}(\mathcal{T})$[7] $= \overline{\mathbf{G}}^{s}(\mathbf{X})$. Thus, four vectors above representing n_y are added to the variables shown in subsection II-A. Note that initial state of charge of each storage i at time t is defined as $e_{i,t-1} = e_i^{max} \mathbf{SOCi}_{i,t}$ where $\mathbf{SOCi}$ is the input matrix introduced in II such that an initial value of $\mathbf{SOCi}_{i,t}$ is allocated if one of the arrival conditions are satisfied: 1) $\mathbf{AVBP}_{i,t=1} = 1$. 2) $\mathbf{AVBP}_{i,t-1} = 0$ and $\mathbf{AVBP}_{i,t} = 1$.

The procedure to initiate a MPOPF problem is as follows: First, we read information on storage devices such as bus location, charge and discharge efficiencies, capacities and initial state of charge from $\mathbf{BATT}$ matrix, and save them in the $\mathbf{MP}$ instance, previously initiated by other input matrices as introduced previously. Then, we construct the above-mentioned storage variables and their linear equality constraints in the current $\mathbf{MP}$ instance.

[4] n_{gn} is the number of nonlinear equality constraints and n_{gl} is the number of linear equality constraints.

[5] n_{hn} is the number of nonlinear inequality constraints and n_{hl} is the number of linear inequality constraints.

[6] object oriented programming.

[7] Note that $\boldsymbol{\tau}_t$ and $\mathcal{T}$ are representations of two sets such that $\boldsymbol{\tau}_t \in \mathcal{T}$.

Furthermore, four new three-dimensional connectivity matrices of $\{\mathbf{C}^{\mathrm{ch}}, \mathbf{C}^{\mathrm{dch}}, \mathbf{C}^{\mathrm{s}}\} \in \mathbb{B}^{n_b \times n_y \times T}$ and $\mathbf{C}^{\mathrm{g}} \in \mathbb{B}^{n_b \times n_g \times T}$ are constructed. The first two are built based on: 1) bus connectivity matrix $\mathbf{BATT}$, 2) active power provision $\mathbf{AVBP}$, and 3) whether charge or discharge options are available from $\mathbf{CONCH}$ and $\mathbf{CONDI}$ matrices, such that $\mathbf{C}^{\mathrm{ch}}_{ikt} = 1$ if both the following conditions are satisfied: (a) battery i is connected to bus k at time t, and (b) charge of battery i at time t is activated, otherwise 0. $\mathbf{C}^{\mathrm{dch}}_{jkt} = 1$ if both the following conditions are satisfied (a) battery j is connected to bus k at time t, and (b) discharge of battery j at time t is activated, otherwise 0.

Connectivity matrix for reactive power provision $\mathbf{C}^{\mathrm{s}}$ is constructed by: 1) bus connectivity matrix $\mathbf{BATT}$, 2) reactive power provision $\mathbf{AVBQ}$ such that $\mathbf{C}^{\mathrm{s}}_{mkt} = 1$ if both the following conditions are satisfied: (a) battery m is connected to bus k at time t, and (b) reactive power injection or absorption of battery m at time t is activated, otherwise 0. Similarly, a generator connectivity matrix $\mathbf{C}^{\mathrm{g}}$ is built with: 1) bus connectivity matrix of generators $\mathbf{GEN}$, 2) active and reactive power provision $\mathbf{AVG}$ such that $\mathbf{C}^{\mathrm{g}}_{nkt} = 1$ if the generator n is connected to bus k is running at time t, otherwise 0. Thus, Eqs. (11a) and (11b) can be re-formulated now to Eqs. (16a) and (16b) respectively.

$$2\mathbf{C}_t^{\mathrm{g}} \boldsymbol{\mathcal{P}}_t^{\mathrm{g}} - \boldsymbol{\mathcal{P}}_t^{\mathrm{d}} - \mathbf{C}_t^{\mathrm{ch}} \boldsymbol{\mathcal{P}}_t^{\mathrm{ch}} + \mathbf{C}_t^{\mathrm{dch}} \boldsymbol{\mathcal{P}}_t^{\mathrm{dch}} - \Re[\underline{\mathbf{S}}_t^{\mathrm{bus}}] = 0 \tag{16a}$$

$$\mathbf{C}_t^{\mathrm{g}} \boldsymbol{\mathcal{Q}}_t^{\mathrm{g}} - \boldsymbol{\mathcal{Q}}_t^{\mathrm{d}} + \mathbf{C}_t^{\mathrm{s}} \boldsymbol{\mathcal{Q}}_t^{\mathrm{s}} - \Im[\underline{\mathbf{S}}_t^{\mathrm{bus}}] = 0 \tag{16b}$$

where, $\boldsymbol{\mathcal{P}}_t^{\mathrm{d}} = \mathbf{PD}_t$. and $\boldsymbol{\mathcal{Q}}_t^{\mathrm{d}} = \mathbf{QD}_t$ In summary, the total number of variables for each time-step becomes:

$$\mathbf{x}_t = \left[\boldsymbol{\Theta}_t\ \boldsymbol{\mathcal{V}}_t\ \boldsymbol{\mathcal{P}}_t^{\mathrm{g}}\ \boldsymbol{\mathcal{Q}}_t^{\mathrm{g}}\ \boldsymbol{\mathcal{SOC}}_t\ \boldsymbol{\mathcal{P}}_t^{\mathrm{ch}}\ \boldsymbol{\mathcal{P}}_t^{\mathrm{dch}}\ \boldsymbol{\mathcal{Q}}_t^{\mathrm{s}}\right]^{\top}_{1 \times N_{x_t}} \tag{17}$$

$N_{x_t} = n_x + 4n_y$. Subscript t stands for a specific time-step in this paper. In addition to box constraints defined in II-A, we define more box constraints corresponding to the new defined storage variables. $\{\mathbf{SOCMi}_t \leq \boldsymbol{\mathcal{SOC}}_t \leq \boldsymbol{\mathcal{SOC}}^{\max}, (\boldsymbol{\mathcal{P}}^{\mathrm{ch}})^{\min} \leq \boldsymbol{\mathcal{P}}_t^{ch} \leq (\boldsymbol{\mathcal{P}}^{\mathrm{ch}})^{\max}, (\boldsymbol{\mathcal{P}}^{\mathrm{dch}})^{\min} \leq \boldsymbol{\mathcal{P}}_t^{\mathrm{dch}} \leq (\boldsymbol{\mathcal{P}}^{\mathrm{dch}})^{\max}$ and $(\boldsymbol{\mathcal{Q}}^{\mathrm{s}})^{\min} \leq \boldsymbol{\mathcal{Q}}_t^{\mathrm{s}} \leq (\boldsymbol{\mathcal{Q}}^{\mathrm{s}})^{\max}\} \in \mathbb{R}^{n_y \times 1}$ The vector of total variables in the MPOPF problem $\mathbf{X} \in \mathbb{R}^{N_x \times 1}$ where $N_x = TN_{x_t}$, is shown in (18):

$$\mathbf{X} = [\mathbf{x}_1\ \ \mathbf{x}_2\ \ \cdots\ \ \mathbf{x}_t\ \ \cdots\ \ \mathbf{x}_T]^{\top} \tag{18}$$

Finally, a general MPOPF problem can be formulated as:

$$4 \min_{\mathbf{X}} F(\mathbf{X}) \tag{19a}$$

$$\text{s.t. } G(\mathbf{X}) = \left[\widetilde{G}(\mathbf{X})\ \overline{G}(\mathbf{X})\ \overline{G}^{s}(\mathbf{X})\right]^{\top} = 0 \in \mathbb{R}^{N_g \times 1} \tag{19b}$$

$$H(\mathbf{X}) = \left[\widetilde{H}(\mathbf{X})\ \ \overline{H}(\mathbf{X})\right]^{\top} \leq 0 \in \mathbb{R}^{N_h \times 1} \tag{19c}$$

where $F(\mathbf{X}) = f_{t=1}(\mathbf{x}_1) + f_{t=2}(\mathbf{x}_2) + \cdots + f_{t=T}(\mathbf{x}_T)$, $\widetilde{\mathbf{G}}(\mathbf{X}) \in \mathbb{R}^{N_{gn} \times 1}$, $\overline{\mathbf{G}}(\mathbf{X}) \in \mathbb{R}^{N_{gl} \times 1}$, $\overline{\mathbf{G}}^{s}(\mathbf{X}) \in \mathbb{R}^{N_{gs} \times 1}$, $\widetilde{\mathbf{H}}(\mathbf{X}) \in \mathbb{R}^{N_{hn} \times 1}$ and $\overline{\mathbf{H}}(\mathbf{X}) \in \mathbb{R}^{N_{hl} \times 1}$ are as shown:

$$4\widetilde{\mathbf{G}}(\mathbf{X}) = \left[\widetilde{\mathbf{g}}(\mathbf{x}_1)\ \widetilde{\mathbf{g}}(\mathbf{x}_2)\ \ldots\ \widetilde{\mathbf{g}}(\mathbf{x}_T)\right]^{\top} \tag{20a}$$

$$\overline{\mathbf{G}}(\mathbf{X}) = \left[\overline{\mathbf{g}}(\mathbf{x}_1)\ \overline{\mathbf{g}}(\mathbf{x}_2)\ \ldots\ \overline{\mathbf{g}}(\mathbf{x}_T)\right]^{\top} \tag{20b}$$

$$\overline{\mathbf{G}}^{s}(\mathbf{X}) = \left[\overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_1)\ \overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_2)\ \ldots\ \overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_T)\right]^{\top} \tag{20c}$$

$$\widetilde{\mathbf{H}}(\mathbf{X}) = \left[\widetilde{\mathbf{h}}(\mathbf{x}_1)\ \widetilde{\mathbf{h}}(\mathbf{x}_2)\ \ldots\ \widetilde{\mathbf{h}}(\mathbf{x}_T)\right]^{\top} \tag{20d}$$

$$\overline{\mathbf{H}}(\mathbf{X}) = \left[\overline{\mathbf{h}}(\mathbf{x}_1)\ \overline{\mathbf{h}}(\mathbf{x}_2)\ \ldots\ \overline{\mathbf{h}}(\mathbf{x}_T)\right]^{\top} \tag{20e}$$

where $N_g = N_{gn} + N_{gl} + N_{gs}$, $N_{gn} = Tn_{gn}$, $N_{gl} = n_{gl_{t=1}} + n_{gl_{t=2}} + \cdots + n_{gl_{t=T}}$, $N_{gs} = Tn_y$, $N_h = N_{hn} + N_{hl}$, $N_{hn} = Tn_{hn}$, $N_{hl} = n_{hl_{t=1}} + n_{hl_{t=2}} + \cdots + n_{hl_{t=T}} + T(8n_y)$, $\boldsymbol{\tau}_1 = \{\mathbf{x}_1\}$. Furthermore, $\widetilde{\mathbf{g}}(\mathbf{x}_t)$ contains the two new defined constraints of (16a) and (16b). $\overline{\mathbf{g}}(\mathbf{x}_t)$ includes (21a)–(21e) plus any other upper and lower bounds of variable $\mathbf{x}_t$ such that $x_t^{\min} = x_t^{\max}$, which can be user defined, and as such can be removed from the list of box constraints in (20e) and is introduced here as a new linear equality (21g).

$$\theta_t^{\text{slack}} = 0 \tag{21a}$$

$$p_{i,t}^{\text{ch}} = 0, \ \text{if} \quad \{\mathbf{AVBP}_{i,t} \vee \mathbf{CONCH}_{i,t}\} = 0 \tag{21b}$$

$$p_{i,t}^{\text{dch}} = 0, \ \text{if} \quad \{\mathbf{AVBP}_{i,t} \vee \mathbf{CONDI}_{i,t}\} = 0 \tag{21c}$$

$$q_{i,t}^{\text{s}} = 0, \ \text{if} \quad \{\mathbf{AVBP}_{i,t} \vee \mathbf{AVBQ}_{i,t}\} = 0 \tag{21d}$$

$$p_{i,t}^{\text{g}} = 0, \ \text{if} \quad \mathbf{AVG}_{i,t} = 0 \tag{21e}$$

$$q_{i,t}^{\text{g}} = 0, \ \text{if} \quad \mathbf{AVG}_{i,t} = 0 \tag{21f}$$

$$x_t = x_t^{\min} = x_t^{\max} \ \text{if} \ x_t^{\min} = x_t^{\max} \tag{21g}$$

$\overline{\mathbf{g}}^{\text{s}}(\mathbf{x}_t)$ is defined from (15), $\widetilde{\mathbf{h}}(\mathbf{x}_t)$ is the non-linear inequality constraints for time t and is similar to (12). Finally $\overline{\mathbf{h}}(\mathbf{x}_t)$ is the set of box constraints of all variables except the slack bus. Our proposed formulation through the instance of $\mathbf{MP}$ is fed to the solver that is introduced in the next section.

Note that the number of linear equality constraints of n_{gl_t} and the number of linear inequality constraints of n_{hl_t}, in each time $t = \{1, \ldots, T\}$, are dependent on the availability matrices $\mathbf{AVBP}$, $\mathbf{CONCH}$, $\mathbf{CONDI}$, $\mathbf{AVBQ}$ and $\mathbf{AVG}$ introduced in II. These numbers play an important role in the Jacobian structure of solution proposal and the follow-up re-ordering section, which will be explained in IV-B. In brief, they are constant numbers over time $t = \{1, \ldots, T\}$ in the optimisation, if all the storage devices and generators have similar input availability matrices over time t, as shown in (1) (if all storage devices are SESS).

III. Solution Proposal

A. Primal-Dual Interior Point

The problem formulated in Section II-B can be solved using primal-dual interior method [13]. This can be implemented by converting the inequality equations to equality in (13) using slack variable of $z_i \in \mathbb{R}$, where i denotes the number of inequality equation $\{i | i \in \mathbb{N}, 1 \leq i \leq N_h\}$ and applying barrier function for slack variables:

$$\min_{\mathbf{X}} \left[F(\mathbf{X}) - \gamma \sum_{i=1}^{N_h} ln(z_i)\right] \tag{22a}$$

$$\text{s.t. } \mathbf{G}(\mathbf{X}) = 0, \tag{22b}$$

$$\mathbf{H}(\mathbf{X}) + \mathbf{Z} = 0 \tag{22c}$$

$$\mathbf{Z} \geq 0 \tag{22d}$$

where $\mathbf{Z} \in \mathbb{R}^{N_g \times 1}$ is the vector of slack variables and γ is the perturbation parameter which reduces to zero when the problem approaches to optimal point. Lagrangian function of the sub-problems (22a)–(22d) becomes:

$$\begin{aligned}&\boldsymbol{\mathcal{L}}^{\gamma}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) \\ &= f(\mathbf{X}) + \boldsymbol{\lambda}^{\top}\mathbf{G}(\mathbf{X}) + \boldsymbol{\mu}^{\top}(\mathbf{H}(\mathbf{X}) + \mathbf{Z}) - \gamma \sum_{i=1}^{N_g} ln(z_i)\end{aligned} \tag{23}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{N_g \times 1}, \boldsymbol{\mu} \in \mathbb{R}^{N_h \times 1}$ are the vectors of Lagrange multipliers for equality and inequality constraints. To write Karush-Kuhn-Tucker (KKT) conditions, partial differentials of (23) can be extracted with respect to the all variables:

$$\boldsymbol{\mathcal{L}}_{\mathbf{X}}^{\gamma}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_{\mathbf{X}} + \boldsymbol{\lambda}^{\top}\mathbf{G}_{\mathbf{X}} + \boldsymbol{\mu}^{\top}\mathbf{H}_{\mathbf{X}} = 0 \tag{24a}$$

$$\boldsymbol{\mathcal{L}}_{\mathbf{Z}}^{\gamma}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \boldsymbol{\mu}^{\top} - \gamma \mathbf{e}^{\top}\mathbf{diag}(\mathbf{Z})^{-1} = 0 \tag{24b}$$

$$\boldsymbol{\mathcal{L}}_{\boldsymbol{\lambda}}^{\gamma}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{G}^{\top}(\mathbf{X}) = 0 \tag{24c}$$

$$\boldsymbol{\mathcal{L}}_{\boldsymbol{\mu}}^{\gamma}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{H}^{\top}(\mathbf{X}) + \mathbf{Z}^{\top} = 0 \tag{24d}$$

where $f_{\mathbf{X}} \in \mathbb{R}^{N_x \times 1}$, $\mathbf{G}_{\mathbf{X}} \in \mathbb{R}^{N_g \times N_x}$ and $\mathbf{H}_{\mathbf{X}} \in \mathbb{R}^{N_h \times N_x}$ are partial differentials of objective function, equality constraints and inequality constraints with respect to $\mathbf{X}$, and $\mathbf{e} \in \{\mathbb{1}\}^{N_h \times 1}$. Eqs. (24a)–(24d) can be written as (25a) in a matrix form.

$$4\boldsymbol{\Omega}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \begin{bmatrix} f_{\mathbf{X}} + \boldsymbol{\lambda}^{\top}\mathbf{G}_{\mathbf{X}} + \boldsymbol{\mu}^{\top}\mathbf{H}_{\mathbf{X}} \\ \mathbf{diag}(\mathbf{Z})\boldsymbol{\mu}^{\top} - \gamma \mathbf{e}^{\top} \\ \mathbf{G}^{\top}(\mathbf{X}) \\ \mathbf{H}^{\top}(\mathbf{X}) + \mathbf{Z}^{\top} \end{bmatrix} = 0 \tag{25a}$$

$$\mathbf{Z} > 0 \tag{25b}$$

$$\boldsymbol{\mu} > 0 \tag{25c}$$

We applied Newton-Raphson method [23] to solve sets of equations in (25a), and hence we have:

$$[\boldsymbol{\Omega}_{\mathbf{X}}\ \boldsymbol{\Omega}_{\mathbf{Z}}\ \boldsymbol{\Omega}_{\boldsymbol{\lambda}}\ \boldsymbol{\Omega}_{\boldsymbol{\mu}}]^{k}[\Delta\mathbf{X}\ \Delta\mathbf{Z}\ \Delta\boldsymbol{\lambda}\ \Delta\boldsymbol{\mu}]^{\top^{k}} = -\boldsymbol{\Omega}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu})^{k} \tag{26}$$

where k is the iteration number in each step. In order to use Newton-Raphson's method to solve equation, partial differential equations of $\boldsymbol{\Omega}_X$, $\boldsymbol{\Omega}_Z$, $\boldsymbol{\Omega}_{\boldsymbol{\lambda}}$ and $\boldsymbol{\Omega}_{\boldsymbol{\mu}}$ must be calculated as shown in (27).

$$\begin{bmatrix} \boldsymbol{\mathcal{L}}_{\mathbf{XX}}^{\gamma} & 0 & \mathbf{G}_{\mathbf{X}}^{\top} & \mathbf{H}_{\mathbf{X}}^{\top} \\ 0 & \mathbf{diag}(\boldsymbol{\mu}) & 0 & \mathbf{diag}(\mathbf{Z}) \\ \mathbf{G}_{\mathbf{X}} & 0 & 0 & 0 \\ \mathbf{H}_{\mathbf{X}} & I & 0 & 0 \end{bmatrix}^{k} \begin{bmatrix} \Delta\mathbf{X} \\ \Delta\mathbf{Z} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{\mu} \end{bmatrix}^{k} = \begin{bmatrix} \boldsymbol{\mathcal{L}}_{\mathbf{X}}^{\gamma\top} \\ -\boldsymbol{\mathcal{L}}_{\mathbf{Z}}^{\gamma\top} \\ \boldsymbol{\mathcal{L}}_{\boldsymbol{\lambda}}^{\gamma\top} \\ \boldsymbol{\mathcal{L}}_{\boldsymbol{\mu}}^{\gamma\top} \end{bmatrix}^{k} \tag{27}$$

where $\boldsymbol{\mathcal{L}}_{\mathbf{XX}}^{\gamma}(\mathbf{X}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f_{\mathbf{XX}} + \mathbf{G}_{\mathbf{XX}}(\boldsymbol{\lambda}) + \mathbf{H}_{\mathbf{XX}}(\boldsymbol{\mu})$. Looking at the structure of the coefficient matrix in (27), we

are able to eliminate two of the rows. In the second row of (27), we have the equation as $\mathbf{diag}(\boldsymbol{\mu})\Delta\mathbf{X} + \mathbf{diag}(\mathbf{Z})\Delta\boldsymbol{\mu} = -\mathbf{diag}(\boldsymbol{\mu})\mathbf{Z} + \gamma\mathbf{e}$, where $\Delta\boldsymbol{\mu}$ can re-written as a function of $\Delta\mathbf{Z}$ as in (28).

$$\Delta\boldsymbol{\mu} = -\boldsymbol{\mu} + \mathbf{diag}(\mathbf{Z})^{-1}(\gamma\mathbf{e} - \mathbf{diag}(\boldsymbol{\mu})\Delta\mathbf{Z}) \tag{28}$$

The same holds for the fourth row of (27) which can be replaced by $\Delta\mathbf{Z}$ as a function of $\Delta\mathbf{X}$ as in (29).

$$\Delta\mathbf{Z} = -\mathbf{H}(\mathbf{X}) - \mathbf{Z} - \mathbf{H}_{\mathbf{X}}\Delta\mathbf{X} \tag{29}$$

Thus, two rows of matrix (27) are taken out of the sets of equation and can be calculated by substituting ΔX in (29) and then ΔZ in (28). After the eliminations of above mentioned rows and several stages of simplifications, (27) can finally be written as :

$$\begin{bmatrix} \mathbf{M} & \mathbf{G}_{\mathbf{X}}^{\top} \\ \mathbf{G}_{\mathbf{X}} & 0 \end{bmatrix}^{k} \begin{bmatrix} \Delta\mathbf{X} \\ \Delta\boldsymbol{\lambda} \end{bmatrix}^{k} = \begin{bmatrix} -\mathbf{N} \\ -\mathbf{G}(\mathbf{X}) \end{bmatrix}^{k} \tag{30}$$

$\mathbf{M} \in \mathbb{R}^{N_x \times N_x}$ and $\mathbf{N} \in \mathbb{R}^{N_x \times 1}$ are defined as:

$$\mathbf{M} = \boldsymbol{\mathcal{L}}_{\mathbf{XX}}^{\gamma} + \mathbf{H}_{\mathbf{X}}^{\top}\mathbf{diag}(\mathbf{Z})^{-1}\mathbf{diag}(\boldsymbol{\mu})\mathbf{H}_{\mathbf{X}} \tag{31a}$$

$$\mathbf{N} = f_{\mathbf{X}}^{\top} + \mathbf{G}_{\mathbf{X}}^{\top}\boldsymbol{\lambda} + \mathbf{H}_{\mathbf{X}}^{\top}\boldsymbol{\mu} + \mathbf{H}_{\mathbf{X}}^{\top}\mathbf{diag}(\mathbf{Z})^{-1}(\gamma\mathbf{e} + \mathbf{diag}(\boldsymbol{\mu})\mathbf{H}(\mathbf{X})) \tag{31a}$$

$$\boldsymbol{\mathcal{L}}_{\mathbf{XX}}^{\gamma} = f_{\mathbf{XX}} + \mathbf{G}_{\mathbf{XX}}(\boldsymbol{\lambda}) + \mathbf{H}_{\mathbf{XX}}(\boldsymbol{\mu}) \tag{31c}$$

Solving (30) numerically results to the locally optimum point $\mathbf{X}^*$. By replacing $\Delta\mathbf{X}^k$ into Eq. (29), $\Delta\mathbf{Z}^k$ is obtained and subsequently $\Delta\mathbf{Z}^k$ into Eq. (28) finally $\Delta\boldsymbol{\mu}^k$ is computed. Therefore, $\mathbf{X}^{k+1} = \mathbf{X}^k + \alpha\Delta\mathbf{X}^k$, $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \alpha\Delta\boldsymbol{\lambda}^k$, $\mathbf{Z}^{k+1} = \mathbf{Z}^k + \alpha\Delta\mathbf{Z}^k$ and $\boldsymbol{\mu}^{k+1} = \boldsymbol{\mu}^k + \alpha\Delta\boldsymbol{\mu}^k$, where α is the step-control parameter which can be chosen arbitrary or based on a optimal multiplier method, which is beyond the scope of this paper [24]. The detail of numerical solution in Newton's method can be found in [25].

It is common to construct Jacobian matrix of (30) using numerical derivatives and solve the KKT equations using LU factorization [26]. However, note that $\mathbf{M}$ is an asymmetric matrix even though it is structurally symmetric, The reason is the term $\mathbf{H}_{\mathbf{X}}^{\top}\mathbf{diag}(\mathbf{Z})^{-1}\mathbf{diag}(\boldsymbol{\mu})\mathbf{H}_{\mathbf{X}}$ in Eq. (31a) which makes it asymmetrical. Therefore it is not possible to apply the $LDL^{\top}$ factorization technique [26].

IV. Speed up of the Solution Proposal

In terms of both memory allocation and computational operations, a MPOPF problem can be attributed to two parts: 1) Input data[8] preparation, and 2) Core optimisation solver which itself consists of: (a) function evaluation, which is the calculus of partial differentials of constraints and objective function w.r.t. all variables in the solution proposal section, (b) computing the inverse of Newton-Raphson Jacobian (30), and (c) computational efforts regarding the update of step-control parameter in each iteration. It is well known that step (b) is the most computationally expensive step of an interior point (IP) algorithm with a large-scale number of variables and constraints [18], [27]. In this section, first, we mathematically derive the analytical derivative of partial differentiation of constraints w.r.t. all existing variables which, in turn, is the fastest way to solve step (a) [28], and then, we tailor an algorithm to exploit the structure of KKT systems, specifically using a Schur-Complement approach to accelerate step (b) of the IP method. In the following subsections we will elaborate these two steps.

A. Analytical Derivatives

In this subsection, the first and second analytical derivatives of $\mathbf{H}(\mathbf{X})$, $\mathbf{G}(\mathbf{X})$ and $F(\mathbf{X})$ will be extracted. These derivations will be used to construct and consequently solve Eqs. (31a), (31b), (31c) and finally (30). Details of equations regarding the extraction of analytical derivatives can be found in Appendix B[9].

Furthermore, to exploit the sparsity structure of each block of $\mathbf{G}_{\mathbf{X}} = \frac{\partial\mathbf{G}}{\partial\mathbf{X}}$, $\mathbf{H}_{\mathbf{X}} = \frac{\partial\mathbf{H}}{\partial\mathbf{X}}$, $F_{\mathbf{X}} = \frac{\partial F}{\partial\mathbf{X}}$, $\mathbf{G}_{\mathbf{XX}} = \frac{\partial}{\partial\mathbf{X}}(\mathbf{G}_{\mathbf{X}}^{\top}\boldsymbol{\lambda})$, $\mathbf{H}_{\mathbf{XX}} = \frac{\partial}{\partial\mathbf{X}}(\mathbf{H}_{\mathbf{X}}^{\top}\boldsymbol{\lambda})$, $F_{\mathbf{XX}} = \frac{\partial}{\partial\mathbf{X}}(F_{\mathbf{X}}^{\top})$ and their subsequent sub-blocks according to (20a)–(20e), a robust and simplified form of structure is developed in order to use highly efficient mathematical operations and matrix substitutions to form (30). The general format of these structures can be found in Appendix C[9] of this paper.

B. Structure Exploitation and Following Re-Ordering

Theoretically, the concept of MPOPF is several snapshots of OPF coupled in time. Each snapshot is a dispatch operational problem with its own variables introduced with (17). If we exploit the structure of the Hessian matrix in (30) with ordering of constraints and variables as shown in (20), then, the detailed structure of $\mathbf{M}$ and $\mathbf{G}_{\mathbf{X}}$ can be seen as:

$$\begin{bmatrix} \begin{bmatrix} \mathbf{M}_1 & & \\ & \ddots & \\ & & \mathbf{M}_T \end{bmatrix} & \mathbf{G}_{\mathbf{X}}^{\top} \\ \begin{bmatrix} \widetilde{\mathbf{G}}_{\mathbf{x}_1} & & \\ & \ddots & \\ & & \widetilde{\mathbf{G}}_{\mathbf{x}_T} \end{bmatrix} & \\ \begin{bmatrix} \overline{\mathbf{G}}_{\mathbf{x}_1} & & \\ & \ddots & \\ & & \overline{\mathbf{G}}_{\mathbf{x}_T} \end{bmatrix} & \mathbf{O} \\ \begin{bmatrix} \overline{\mathbf{G}}_{\boldsymbol{\tau}_1}^{s} & & \\ & \ddots & \\ & & \overline{\mathbf{G}}_{\boldsymbol{\tau}_T}^{s} \end{bmatrix} & \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_1 \\ \vdots \\ \Delta\mathbf{x}_T \\ \Delta\widetilde{\boldsymbol{\lambda}}_1 \\ \vdots \\ \Delta\widetilde{\boldsymbol{\lambda}}_T \\ \Delta\overline{\boldsymbol{\lambda}}_1 \\ \vdots \\ \Delta\overline{\boldsymbol{\lambda}}_T \\ \Delta\overline{\boldsymbol{\lambda}}_1^{s} \\ \vdots \\ \Delta\overline{\boldsymbol{\lambda}}_T^{s} \end{bmatrix} = \begin{bmatrix} -\mathbf{N}_1 \\ \vdots \\ -\mathbf{N}_T \\ -\widetilde{\mathbf{g}}(\mathbf{x}_1) \\ \vdots \\ -\widetilde{\mathbf{g}}(\mathbf{x}_T) \\ -\overline{\mathbf{g}}(\mathbf{x}_1) \\ \vdots \\ -\overline{\mathbf{g}}(\mathbf{x}_T) \\ -\overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_1) \\ \vdots \\ -\overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_T) \end{bmatrix} \tag{32}$$

where $\mathbf{M}_t \in \mathbb{R}^{N_{x_t} \times N_{x_t}}$, $\mathbf{G}_{\mathbf{X}}^{\top} \in \mathbb{R}^{N_x \times N_g}$ is the transpose of the left bottom block in the coefficient matrix of (32), $\mathbf{O} \in \mathbb{O}^{N_g \times N_g}$, (set of zeros: $\mathbb{O}$), $\widetilde{\mathbf{G}}_{\mathbf{x}_t} = \frac{\partial\widetilde{\mathbf{g}}}{\partial\mathbf{x_t}} \in \mathbb{R}^{N_x \times n_{gn}}$, $\overline{\mathbf{G}}_{\mathbf{x}_t} = \frac{\partial\overline{\mathbf{g}}}{\partial\mathbf{x_t}} \in \mathbb{R}^{N_x \times n_{gl_t}}$. As defined in section II-B, for the sake

[8] 14 input matrices for BATTPOWER solver, see Appendix A.

[9] www.arxiv.org/abs/2012.12941.

of simplicity of notation we take $\overline{\mathbf{G}}^{s}_{\boldsymbol{\tau}_t} = \frac{\partial \overline{\mathbf{g}}^{s}}{\partial \boldsymbol{\tau}_t} = \frac{\partial \overline{\mathbf{g}}^{s}}{\partial (\mathbf{x}_{t-1}, \mathbf{x}_t)} \in \mathbb{R}^{N_x \times n_{gs}}$. If we reorder the current variables and consequently re-construct the coefficient and righthand side matrix such that all variables corresponding to time t are assembled together except the variables of inter-temporal constraints, then the vector of variables with its corresponding righthand side and coefficient matrix could be written as (33a), (33b) and (33c), respectively.

$$[\Delta \mathbf{x}_1\ \Delta\widetilde{\boldsymbol{\lambda}}_1\ \Delta\overline{\boldsymbol{\lambda}}_1 \ldots \Delta \mathbf{x}_T\ \Delta\widetilde{\boldsymbol{\lambda}}_T\ \Delta\overline{\boldsymbol{\lambda}}_T,\quad \Delta\overline{\boldsymbol{\lambda}}^{s}_1 \ldots \Delta\overline{\boldsymbol{\lambda}}^{s}_T]^{\top} \tag{33a}$$

$$-\Big[\mathbf{N}_1\ \widetilde{\mathbf{g}}(\mathbf{x}_1)\ \overline{\mathbf{g}}(\mathbf{x}_1) \ldots \mathbf{N}_T\ \widetilde{\mathbf{g}}(\mathbf{x}_T)\ \overline{\mathbf{g}}(\mathbf{x}_T), \overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_1) \ldots \overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_T)\Big]^{\top} \tag{33b}$$

$$\begin{bmatrix} \begin{bmatrix} \boldsymbol{\Upsilon}_1 & & \\ & \ddots & \\ & & \boldsymbol{\Upsilon}_T \end{bmatrix} & \overline{\mathbf{G}}^{sr\top}_{\mathbf{X}} \\ \begin{bmatrix} \overline{\mathbf{G}}^{sr}_{\boldsymbol{\tau}_1} & & \\ & \ddots & \\ & & \overline{\mathbf{G}}^{sr}_{\boldsymbol{\tau}_T} \end{bmatrix} & \mathbf{O}^{r} \end{bmatrix} \tag{33c}$$

where $\boldsymbol{\Upsilon}_t = \begin{bmatrix} \mathbf{M}_t & \begin{bmatrix} \widetilde{\mathbf{G}}^{\top}_{\mathbf{x}_t} & \overline{\mathbf{G}}^{\top}_{\mathbf{x}_t} \end{bmatrix} \\ \begin{bmatrix} \widetilde{\mathbf{G}}_{\mathbf{x}_t} \\ \overline{\mathbf{G}}_{\mathbf{x}_t} \end{bmatrix} & \mathbf{O}^{r'} \end{bmatrix} \in \mathbb{R}^{N_{\boldsymbol{\Upsilon}_t} \times N_{\boldsymbol{\Upsilon}_t}}$, $N_{\boldsymbol{\Upsilon}_t} = N_{x_t} + n_{gn} + n_{gl_t}$, $\mathbf{O}^{r'} \in \mathbb{O}^{[n_{gn}+n_{gl_t}] \times [n_{gn}+n_{gl_t}]}$ and $\overline{\mathbf{G}}^{sr}_{\mathbf{X}} = \begin{bmatrix} \overline{\mathbf{G}}^{sr}_{\boldsymbol{\tau}_1} & & \\ & \ddots & \\ & & \overline{\mathbf{G}}^{sr}_{\boldsymbol{\tau}_T} \end{bmatrix} \in \mathbb{R}^{N_{gs} \times N_{gsr}}$, $N_{gsr} = N_{\boldsymbol{\Upsilon}_{t=1}} + N_{\boldsymbol{\Upsilon}_{t=2}} + \cdots + N_{\boldsymbol{\Upsilon}_{t=T}}$, $\mathbf{O}^{r} \in \mathbb{O}^{N_{gs} \times N_{gs}}$. In order to illustrate the re-ordering more clearly, we define: $\delta\boldsymbol{\omega}_t = [\Delta \mathbf{x}_t \Delta\widetilde{\boldsymbol{\lambda}}_t \Delta\overline{\boldsymbol{\lambda}}_t]^{\top} \in \mathbb{R}^{N_{\boldsymbol{\Upsilon}_t} \times 1}$, $\delta\boldsymbol{\lambda} = [\Delta\overline{\boldsymbol{\lambda}}^{s}_1 \ldots \Delta\overline{\boldsymbol{\lambda}}^{s}_T]^{\top} \in \mathbb{R}^{N_{gs} \times 1}$, $\boldsymbol{\zeta}_t = -[\mathbf{N}_t \widetilde{\mathbf{G}}_{\mathbf{x}_t} \overline{\mathbf{G}}_{\mathbf{x}_t}]^{\top} \in \mathbb{R}^{N_{\boldsymbol{\Upsilon}_t} \times 1}$ and $\boldsymbol{\Gamma} = -[\overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_1) \ldots \overline{\mathbf{g}}^{s}(\boldsymbol{\tau}_T)]^{\top} \in \mathbb{R}^{N_{gs} \times 1}$. Therefore we can convert (33) to (34) which is a well-known "arrowhead" structure that can be found in the literature [29], [30].

$$\begin{bmatrix} \boldsymbol{\Upsilon}_1 & & & & \boldsymbol{\rho}^{\top}_1 \\ & \boldsymbol{\Upsilon}_2 & & & \boldsymbol{\rho}^{\top}_2 \\ & & \ddots & & \vdots \\ & & & \boldsymbol{\Upsilon}_T & \boldsymbol{\rho}^{\top}_T \\ \boldsymbol{\rho}_1 & \boldsymbol{\rho}_2 & \ldots & \boldsymbol{\rho}_T & 0 \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\omega}_1 \\ \delta\boldsymbol{\omega}_2 \\ \vdots \\ \delta\boldsymbol{\omega}_T \\ \delta\boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\zeta}_1 \\ \boldsymbol{\zeta}_2 \\ \vdots \\ \boldsymbol{\zeta}_T \\ \boldsymbol{\Gamma} \end{bmatrix} \tag{34}$$

where the coupling matrices of $\boldsymbol{\rho}_t \in \mathbb{R}^{N_{gs} \times N_{\boldsymbol{\Upsilon}_t}}$ are:

$$\boldsymbol{\rho}_1 = \begin{bmatrix} \overline{\mathbf{G}}^{s}_{\boldsymbol{\tau}_1} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{\rho}_2 = \begin{bmatrix} 0 \\ \overline{\mathbf{G}}^{s}_{\boldsymbol{\tau}_2} \\ 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{\rho}_T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \overline{\mathbf{G}}^{s}_{\boldsymbol{\tau}_T} \end{bmatrix} \tag{35}$$

Fig. 1 illustrates the reordering of Hessian matrix of Eqs. (32) to (34). In the next subsection, we propose a Schur-Complement technique tailored for the reordered structure of Eq. (34) to save computational time. Considering Eqs. (33) and (34), it should be kept in mind that in the introduced re-ordering structure, there are

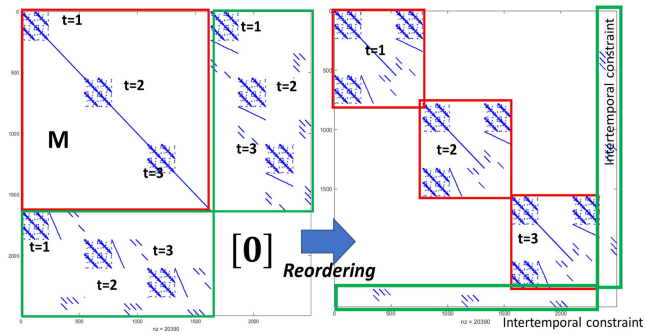

Fig. 1. Structure of Jacobian of the Newton-Raphson's algorithm before and after reordering.

some vectors which could have a different length in each time t as discussed in Section II-B when the availability of either storage devices, charge, discharge, reactive power provision, and generators would alternate over the optimisation horizon ($\mathbf{AVBP}\big|_{t=1} \neq \mathbf{AVBP}\big|_{t=2} \neq \cdots \neq \mathbf{AVBP}\big|_{t=T}$), and consequently, n_{gl_t} and n_{hlt} are not constant through time $t = \{1, \ldots, T\}$. Therefore, specific indices are introduced to keep track of them at each time and over each iteration k, in Eq. (32).

C. Schur-Complement Technique

The sparse arrowhead structure of the coefficient matrix of (34), is suited for block elimination using the Schur-Complement technique [26]. The algorithm is tailored for solving any problem featured with repeatable matrices and coupling constraints between them, such as: (a) multi-period systems as this paper presents, and in the literature [16], (b) stochastic problems with a large number of scenarios [29], [30], and (c) security constrained problems with a large number of contingencies [31]. Algs. 1 and 2 are proposed to solve for a generic multi-period KKT system with a structurally symmetric structure. Alg. 1 is for Schur-Complement factorisation and Alg. 2 is for forward and backward substitution. The substructures of (34) are inputs to both Algs. 1 and 2.

1) Alg. 1: Schur-Complement Factorisation: Algorithm 1 starts with finding a permutation matrix of $\mathbf{Q}^{am}_t \in \mathbb{B}^{N_{\boldsymbol{\Upsilon}_t} \times N_{\boldsymbol{\Upsilon}_t}}$, generated in order to capture the sparse structure of $\boldsymbol{\Upsilon}_t$ to reduce the number of non-zeroes in LU factorisation in L.8. Permutation matrix $\mathbf{Q}^{am}_t$ is produced based on an approximate minimum degree permutation method [32]. It should be kept in mind that the structure of $\mathbf{Q}^{am}_t$ is dependent on $\boldsymbol{\Upsilon}_t$ which, in turn, is dependent on the structure of input matrices and the conditions of (1a)–(1c). If these conditions hold (all storage devices are SESS), then $N_{\boldsymbol{\Upsilon}_t}$ is constant through time t. Therefore, $\boldsymbol{\Upsilon}_t$ and $\mathbf{Q}^{am}_t$ have constant structures over time and L.6 will not be executed; in other words $\mathbf{Q}^{am}_1 = \mathbf{Q}^{am}_2 = \mathbf{Q}^{am}_t$. If these conditions do not hold, which in turn mean dynamic storage devices (EV), then L.2 is not executed and instead, L.6 will be executed. The factorisation here is based on an *incomplete augmented factorisation* technique [16] in order to compute the Schur-Complement of

Algorithm 1: Schur-Complement Factorization.

1 **Function** SchurComI($\{\boldsymbol{\Upsilon}_t, \boldsymbol{\rho}_t, \boldsymbol{\zeta}_t, \forall\, t = 1, \ldots, T\}$):
2 $[\mathbf{Q}_t^{am}]$=ApproxMinDegrPermut($\boldsymbol{\Upsilon}_1$)// If (1a) – (1c) hold
3 $\boldsymbol{\sigma}^c = 0$
4 $\boldsymbol{\sigma}^l = 0$
5 **for** t= 1 :T **do**
6 $[\mathbf{Q}_t^{am}]$=ApproxMinDegrPermut($\boldsymbol{\Upsilon}_t$)
7 // If (1a) – (1c) does NOT hold
8 $\boldsymbol{\Pi}$ = SparsePermute($\left[[\mathbf{Q}_t^{am}]^\top \boldsymbol{\Upsilon}_t \mathbf{Q}_t^{am}\right]$)
9 $[\mathbf{L}_t^{lu}, \mathbf{U}_t^{lu}, \mathbf{P}_t^{lu}, \mathbf{Q}_t^{lu}, \mathbf{R}_t^{lu}]$
10 = SparseLUFactorize($\boldsymbol{\Pi}$)
11 $\mathbf{inf}_t^{\boldsymbol{\Upsilon}}$ =struct($\mathbf{L}_t^{lu}, \mathbf{U}_t^{lu}, \mathbf{P}_t^{lu}, \mathbf{Q}_t^{lu}, \mathbf{R}_t^{lu}, \mathbf{Q}_t^{am}$)
12 $\mathbf{S}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\rho}_t^\top$ // SchurCompAuxiliary $\mathbf{A}_t^a$
13 $\boldsymbol{\sigma}^c = \boldsymbol{\sigma}^c + \mathbf{S}_t$ // $MainSchurCompArrowhead - 34$
14 $\boldsymbol{\Xi}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\zeta}_t$ // SchurCompAuxiliary $\mathbf{A}_t^b$
15 $\boldsymbol{\sigma}^l = \boldsymbol{\sigma}^l + \boldsymbol{\Xi}_t$// righthand side Alg 2,L.2
16 **End for**
17 $[\mathbf{L}^{ldl}, \mathbf{D}^{ldl}, \mathbf{P}^{ldl}, \mathbf{S}^{ldl}]$ =SparseLDLFactorize($\boldsymbol{\sigma}^c$)
18 $\mathbf{inf}^c$ = struct ($\mathbf{L}^{ldl}, \mathbf{D}^{ldl}, \mathbf{P}^{ldl}, \mathbf{S}^{ldl}$)
19 **return** $\{\boldsymbol{\sigma}^l, \mathbf{inf}^c\}$ and $\{\mathbf{inf}_t^{\boldsymbol{\Upsilon}},\ \forall\, t = 1, \ldots, T\}$
20 **End Function**

each augmented matrix of $\mathbf{A}_t^a = \left[\begin{smallmatrix} \boldsymbol{\Upsilon}_t & \boldsymbol{\rho}_t^\top \\ \boldsymbol{\rho}_t & 0 \end{smallmatrix}\right]$. Permuted $\boldsymbol{\Upsilon}_t$ factorises with LU factorisation technique in L.8 and afterwards, the Schur-Complement of each block of $\mathbf{A}_t^a$ is computed as $\mathbf{S}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\rho}_t^\top \in \mathbb{R}^{N_{gs} \times N_{gs}}$ in L.10 in each iteration and summed together in $\boldsymbol{\sigma}^c \in \mathbb{R}^{N_{gs} \times N_{gs}}$ L.11 to shape the main Schur-Complement of arrowhead structure of Eq. (34).

With almost the same procedure explained above, in order to compute $\boldsymbol{\xi}$ which is the righthand side of the main Schur-Complement equation in $[\boldsymbol{\sigma}^c][\delta\boldsymbol{\lambda}] = [\boldsymbol{\xi}]$, (refer to Alg 2, L.2), we define another auxiliary block matrix of $\mathbf{A}_t^b = \left[\begin{smallmatrix} \boldsymbol{\Upsilon}_t & \boldsymbol{\zeta}_t \\ \boldsymbol{\rho}_t & 0 \end{smallmatrix}\right]$ and consequently compute its Schur-Complement as $\boldsymbol{\Xi}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\zeta}_t$ where $\boldsymbol{\Upsilon}_t^{-1}$ is factorised in previous step L.8. Thus, we only recall the stored "struct" of $\mathbf{inf}_t^{\boldsymbol{\Upsilon}}$ L.9 from memory. Value of $\boldsymbol{\Xi}_t$ is aggregated in each iteration with $\boldsymbol{\sigma}^l$ in L.13. $\boldsymbol{\sigma}^c$ is the Schur-Complement of arrowhead structure of Eq. (34) and has an interesting pattern that can be exploited further, and it is dependent on the input matrices of **AVBP**, **CONCH** and **CONDI**.

a) Static Schur-Complement Structure: SESS. If $\{\mathbf{AVBP}, \mathbf{CONCH}, \mathbf{CONDI}\} \in \{\mathbb{1}\}^{n_y \times T}$ holds, then all the storage devices are considered as SESS. As elaborated above, $\boldsymbol{\sigma}^c$ is the aggregation of Schur-Complement of each auxiliary block of $\mathbf{A}_t^a$, therefore, in each iteration $\mathbf{S}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\rho}_t^\top$. $\boldsymbol{\sigma}^c$ is a sparse bandwidth matrix such that $\{\forall i, j\ \boldsymbol{\sigma}_{i,j}^c = 0$ if $|i - j| > n_y\}$ where the number of non-zero matrix elements is only dependent on the number of storage devices n_y and simulation horizon T, and not on network properties. More precisely, each element of $s_{ij,t}$ is computed through $s_{ij,t} = -\boldsymbol{\rho}_{i,t} \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\rho}_{j,t}^\top$ considering the only non-zero part of $\boldsymbol{\rho}_t$ is $\overline{\mathbf{G}}_{\boldsymbol{\tau}_t}^s$ which moves from top to bottom while t moves from $t = 1$ to $t = T$ as illustrated in Eq. (35) and in the for loop of Alg. 1, with the essential assumption that the condition

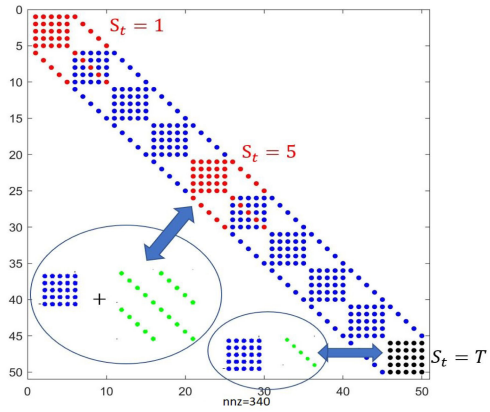

Fig. 2. Overall structure of the main Schur-Complement: $\boldsymbol{\sigma}^c$. The structure of $\mathbf{S}_t$ is shown with red dots, and $\mathbf{S}_{t=T}$ with black dots, for a network with $n_y = 5$ and $T = 10$.[10]

$\mathbf{AVBP} = \mathbf{CONDI} = \mathbf{CONCH} = [\mathbb{1}]_{n_y \times T}$ holds, which intuitively means that $\overline{\mathbf{G}}_{\boldsymbol{\tau}_t}^s$ has a constant structure over optimisation horizon. $\boldsymbol{\rho}_t^\top$ also has the same pattern through a loop from $t = 1$ to $t = T$. Therefore, $\mathbf{S}_t$ moves in a bandwidth structure as shown in Fig. 2 from the top left corner to the bottom right. Each block of $\mathbf{S}_{t \neq T}$ is constructed by two sub-blocks of $\overline{\mathbf{G}}_{\boldsymbol{\tau}_t}^s$, the blue rectangular sub-block with $\boldsymbol{\mathcal{P}}^{ch}$ and $\boldsymbol{\mathcal{P}}^{dch}$ variables and the light green sub-block with $\boldsymbol{\mathcal{SOC}}$ variables at each time. The structure of $\mathbf{S}_{t=T}$ is different since $\overline{\mathbf{G}}_{\boldsymbol{\tau}_T}^s$ is the last linking equation over time, and therefore, $\boldsymbol{\mathcal{SOC}}$ is no longer linked to a next time, but $\boldsymbol{\mathcal{P}}^{ch}$ and $\boldsymbol{\mathcal{P}}^{dch}$ still exist.

b) Dynamic Schur-Complement Structure—EV. If conditions of (1a)–(1c) do not hold, the structure shown in Fig. 2 would change depending on the dynamic behaviour of the input matrices of **AVBP**, **CONCH** and **CONDI**. Fig. 3 illustrates the sparse Schur-Complement pattern of $\boldsymbol{\sigma}^c$, made by input matrices shown in Eqs.(36a)-(36b). Each colour/legend shows the non-zero elements of $\mathbf{S}_t$ for each time $t = 1, \ldots, T$, which show up in the main Schur-Complement[11] structure $\boldsymbol{\sigma}^c$. As presented in Alg. 1, L.11, the structure of $\boldsymbol{\sigma}^c$, and also as shown in Fig. 3, is constructed inside a for loop from $t = 1$ to $t = T$. The non-zero elements of $\mathbf{S}_t$ have overlaps with non-zero elements of $\mathbf{S}_{t-1}$ and $\mathbf{S}_{t+1}$, see Fig. 3. Therefore, they are added together in the for loop while t progresses from $t = 1$ to $t = T$. In this algorithm, it is important to allocate a size of memory proportional to the number of non-zero elements of $\mathbf{S}_t$ and $\boldsymbol{\sigma}^c$ to achieve high performance. It should be noted that the number of non-zero elements in $\mathbf{S}_t$ and $\boldsymbol{\sigma}^c$ (and the size of allocated memory) for each time $t = 1, \ldots, T$ are in turn function of input matrices and they can be predetermined before the starting of operation of

[10] $nnz = 340$ stands for number of non-zero elements
[11] $nnz = 202$ stands for number of non-zero elements, different value for each time

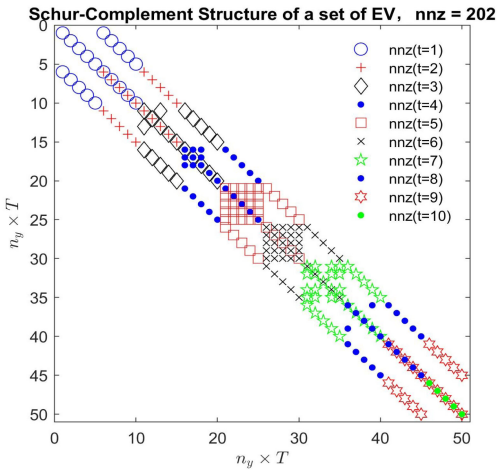

Fig. 3. Overal structure of the main Schur-Complement: $\boldsymbol{\sigma}^c$, for a network with $n_y = 5$ and $T = 10$, corresponding to input matrices of (36a)-(36b).[11] Number of non-zero elements of $\mathbf{S}_t$ changes through time $t = \{1, \ldots, T\}$.

Alg. 1.

$$\mathbf{AVBP} = \mathbf{CONCH} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{36a}$$

$$\mathbf{CONDI} = \Big[\mathbb{0}\Big]_{(n_y=5)\times(T=10)} \tag{36b}$$

There is no EV in the system at time $t = 1, 2$; two EV appear at time $t = 3$, full EV connected system at $t = 5, 6$; similarly, corresponding structures alter, as can be seen in Fig. 2. The reason for the different structure shown Fig. 2 is the same reason elaborated in Section IV-C1. The only non-zero part of $\boldsymbol{\rho}_t$ is $\overline{\mathbf{G}}^s_{\boldsymbol{\tau}_t}$ which moves from top to bottom while t moves from $t = 1$ to $t = T$ illustrated in Eq. (35) and in the for loop of Alg. 1. $\overline{\mathbf{G}}^s_{\boldsymbol{\tau}_t}$ is the first derivative of energy storage systems in the time t and its structure alternates over time, due to the dynamic behaviour of input matrices of $\mathbf{AVBP}$, $\mathbf{CONCH}$ and $\mathbf{CONDI}$. Therefore, the Schur-Complement structure of $\boldsymbol{\sigma}^c$ gets more sparse than that of SESS and could be factorised even faster than that of SESS. In the rest of Alg. 1, $\boldsymbol{\sigma}^c$ is factorised by sparse LDL factorisation technique, since it is a sparse symmetric matrix. Finally Alg. 1 returns $\boldsymbol{\sigma}^l$ matrix, and $\mathbf{inf}^c$ and $\mathbf{inf}^{\boldsymbol{\Upsilon}}_t$ structs containing factorisation information to be called in Alg. 2.

2) Alg. 2 Forward and Backward Substitution: Inputs are matrices of $\boldsymbol{\Gamma}, \boldsymbol{\sigma}^l, \boldsymbol{\rho}_t, \boldsymbol{\zeta}_t$, and structs of $\mathbf{inf}^c, \mathbf{inf}^{\boldsymbol{\Upsilon}}_t$. Alg. 2 is to solve $\delta\boldsymbol{\lambda}$ through sparse LDL forward and backward substitution, and further, to compute $\delta\boldsymbol{\omega}_t$ through a for loop with the help of sparse LU forward and backward substitution.

First the righthand side of the main Schur-Complement equation of $\boldsymbol{\sigma}^c\delta\boldsymbol{\lambda} = \boldsymbol{\xi}$, which is $\boldsymbol{\xi}$, is computed in Alg 2, L.2. Using the "struct" of $\mathbf{inf}^c$, the sparse LDL forward and backward substitution is performed in L.3 and $\delta\boldsymbol{\lambda}$ is cleared. Second,

Algorithm 2: Forward and Backward Substitution.

1 **Function** `SchurComII` $(\{\boldsymbol{\Gamma}, \boldsymbol{\sigma}^l, \mathbf{inf}^c\}, \{\boldsymbol{\rho}_t, \boldsymbol{\zeta}_t, \mathbf{inf}^{\boldsymbol{\Upsilon}}_t, \forall\, t = 1, \ldots, T\})$ **:**
2 $\quad \boldsymbol{\xi} = \boldsymbol{\Gamma} - \boldsymbol{\sigma}^l$
3 $\quad \delta\boldsymbol{\lambda}$=SparseLDLForBackSolve($\mathbf{inf}^c$, $\boldsymbol{\xi}$)
4 $\quad$ **for** t= 1 :T **do**
5 $\quad\quad \boldsymbol{\kappa}_t = \boldsymbol{\zeta}_t - \boldsymbol{\rho}_t^\top \delta\boldsymbol{\lambda}$
6 $\quad\quad \delta\boldsymbol{\omega}_t$= SparseLUForBackSolve($\mathbf{inf}^{\boldsymbol{\Upsilon}}_t, \boldsymbol{\kappa}_t$)
7 $\quad\quad$ **End for**
8 $\quad$ **return** $\{\delta\boldsymbol{\lambda}\}$ and $\delta\boldsymbol{\omega}_t,\ \forall\, t = 1, \ldots, T\}$
9 $\quad$ **End Function**

in a for loop, a slack variable called $\boldsymbol{\kappa}_t$ is constructed with $\boldsymbol{\kappa}_t = \boldsymbol{\zeta}_t - \boldsymbol{\rho}_t^\top \delta\boldsymbol{\lambda}$ in L.5 and recalled to solve for the sparse LU forward and backward substitution using the "struct" of $\mathbf{inf}^{\boldsymbol{\Upsilon}}_t$ in each t and thus, $\delta\boldsymbol{\omega}_t$ is cleared. Finally, Alg. 2 returns vectors $\delta\boldsymbol{\lambda}$ and $\delta\boldsymbol{\omega}_t$.

D. Computational Performance and Memory Efficiency

1) Sparse Matrix Operations: Since most of the operations here are done in sparse format efficiently, sparse indexing and libraries are developed to construct and handle re-ordering explained above. Sparsity is a method to save significant memory in large-scale simulations.

2) Function Evaluation: First and foremost, analytical derivative functions, known as hand-coded functions, are implemented here, and are the fastest possible way to compute all partial derivatives of the first and second order of objective function and constraints w.r.t. all variables, especially when it comes to large-scale optimisation systems [28]. Efficiently exploiting and handling the operations of sparse matrices and subsequently updating their indices over the optimisation time horizon accelerates the computational performance.

3) KKT Systems: The computational complexity of sparse matrix operations is proportional to the number of non-zero elements in each sparse matrix and independent of the size of matrices [33]. Note that the number of non-zero elements of Jacobian matrix for each case and per row is almost constant and is similar to the number of non-zero elements of a coefficient matrix obtained from discretisation of finite element methods in three-dimensional meshes.

Therefore, the only possible way to assess the performance of a solution proposal for KKT systems is through the estimation of sparse matrix operations. Factorisation L.8 in Alg. 1 is the most computationally expensive step in the loop when $n_y \leq \mathcal{K}$, where $\mathcal{K} \propto N_{\boldsymbol{\Upsilon}_t} = N_{x_t} + n_{gn} + n_{gl_t} = 2n_b + 2n_g + 4n_y + n_{gn} + n_{gl_t}$.[12] Note that the rest of Alg. 1 and Alg. 2 are dominated by this step and can be ignored in this case. If the number of non-zero elements per row of coefficient matrix of (35) is approximately similar to Laplacian matrices discretised by finite elements, then LU factorisation of L.8 has

[12]In other words, Factorisation L.8 is the most computationally expensive step, when the number of storage devices are smaller than a certain number, where this number is proportional to size of blocks of $\boldsymbol{\Upsilon}_t$ in Eq. (34).

the complexity of $O(N^2_{\boldsymbol{\Upsilon}_t})$. Consequently, the complexity of factorisation for all blocks of $\boldsymbol{\Upsilon}_t$ will be $O(TN^2_{\boldsymbol{\Upsilon}_t})$.

However, for $n_y > \mathcal{K}$ the complexity of Alg. 1 is dominated by (a) L.10, (b) L.12, since the size of matrix $\boldsymbol{\rho}_t \in \mathbb{R}^{N_{gs} \times N_{\boldsymbol{\Upsilon}_t}}$ gets larger where $N_{gs} = Tn_y$, and (c) $LDL^\top$ factorisation of $\boldsymbol{\sigma}^c$, in L.15; this is because the main Schur-Complement structure of $\boldsymbol{\sigma}^c$ becomes dense and therefore, the complexity of $LDL^\top$ will be $O(\frac{1}{3}(Tn_y)^3)$. Note that overhead of factorisation of $\boldsymbol{\sigma}^c$ is more dependent on n_y than on T since non-zero elements of each sub-structure of $\mathbf{S}_t$ are $\{nnz(\mathbf{S}_t) = n_y(n_y+3) \mid t \neq T,\ nnz(\mathbf{S}_T) = n_y^2\}$. Note that these arguments are only valid when $\mathbf{AVBP} = \mathbf{CONDI} = \mathbf{CONCH} = \left[\mathbb{1}\right]_{n_y \times T}$ holds, which in turn would lead to a static Schur-Complement structure discussed in Section IV-C1. On the contrary, the complexity of dynamic Schur-Complement structure shown in Section IV-C1 has even less overhead than the static one.

4) Memory Efficiency:

a) Input Matrices. BATTPOWER has 14 input matrices in total, as described in the Appendix A: Four are similar to MATPOWER ($\mathbf{BUS}$, $\mathbf{BRANCH}$, $\mathbf{GEN}$, $\mathbf{GENCOST}$) and ten new matrices to capture the mulitperiod formulation and energy storage. Except the mentioned MATPOWER matrices and the new matrises $\mathbf{PD}$ and $\mathbf{QD}$, which are all dense matrices, the rest is stored in a memory-efficient manner, either binary or spare format matrices. $\mathbf{AVBP}$, $\mathbf{CONCH}$, $\mathbf{CONDI}$, $\mathbf{AVBQ}$, and $\mathbf{AVG}$ matrices are the binary ones. Finally, $\mathbf{SOCi}$ and $\mathbf{SOCMi}$ are neither dense nor binary, so these are stored with sparse format.

b) Core Optimisation Solver. As noted in Section IV, the solution of the linear KKT system (30) is the most computationally expensive step in an IP algorithm. This step attributes also to the highest peak memory footprint, since the Newton-Raphson Jacobian (30) is the largest structure built.[13] The solution of the multi-period KKT structure of (30) through Schur-Complement breaks it into smaller $\boldsymbol{\Upsilon}_t$ blocks in (34), and thus, significantly less memory allocations. In fact, the line with peak memory allocation is located in Alg. 1,L.9 where struct $\mathbf{inf}_t^{\boldsymbol{\Upsilon}}$ stores info to be called in Alg. 2 L.6.

V. Case Study and Results

In this section,[14] we present the results of benchmarking, obtained from implementations of different algorithms on similar platforms and workstations. The aim is to show the computational differences among mathematical algorithms when implemented on similar platforms.

In this respect, standard case-files are adopted, Case9 [34], IEEE30 [35], IEEE118 [36] and PEGASE1354 [37], [38] are chosen for the study of SESS. Moreover, three distribution networks are considered for the simulation of EV: Case85 [39], Case141 [40], and a case study based on a real distribution grid in Mid-Norway [41]; except for the last case study, the other cases can be found in the MATPOWER data folder [22]. Details of these benchmarks are shown in Table I. The transmission networks are considered in order to simulate the SESS and consequently the performance of static Schur-Complement structure that was discussed in Section IV. The distribution networks are considered to simulate and compare the performance of dynamic Schur-Complement algorithm on arrival and departure of EV in one 24-hour period.

TABLE I
Power Grid Benchmark Models

Type	case study	n_b	n_l	n_g
Transmission Grid	Case9	9	9	4
	IEEE30	30	41	6
	IEEE118	118	186	54
	PEGASE1354	1354	1991	260
Distribution Grid	Case85	85	84	1
	Case141	141	140	1
	Mid-Norway	974	1023	2

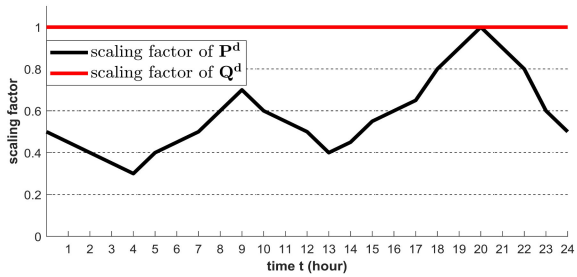

Fig. 4. Scaling factor multiplied by vector of consumption of active load $\boldsymbol{\mathcal{P}}^{\mathrm{d}}$ in order to simulate one period of 24 hours.

For all simulation results, both transmission and distribution networks in the next two subsections, $\mathbf{BUS}$, $\mathbf{BRANCH}$, $\mathbf{GEN}$, and $\mathbf{GENCOST}$ matrices are taken from the original test-cases and are not modified. Moreover, the flat initialisation strategy ($\frac{\mathbf{X}^{\max} - \mathbf{X}^{\min}}{2}$) is taken for all results presented in this paper. Vector of bus active load is obtained through $\boldsymbol{\mathcal{P}}^{\mathrm{d}} = c^p(t).\boldsymbol{\mathcal{P}}$, and $c^p(t)$ is illustrated in Fig. 4, which fluctuates similar to a base load of households. Vector of bus reactive load is simulated as $\boldsymbol{\mathcal{Q}}^{\mathrm{d}} = c^q(t).\boldsymbol{\mathcal{Q}}$ with a constant scaling factor shown in Fig. 4 by a red line. Note that the objective function of the optimisation problem in this paper includes only active power minimisation. $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{Q}}$ are taken from the original values in $\mathbf{BUS}$ matrix. All simulations codes are developed in MATLAB environment. They are performed on a computer with Intel(R) Xeon(R) CPU E5-2690 v4 @ 2.60 GHz and 384 GB RAM, and controlled with the single-thread environment to compare the computational differences in only the single-thread mode.

A. Transmission Network With Stationary Storage

For transmission networks, the time-step is taken to be $\Delta t = 1$ hour. The capacity of storage devices is $e_i^{\max} = 100$ MWh,

[13] In general, both sparse matrix size ($m \times n$) and density have a direct relationship with the size of allocated memory in a computational program.

[14] Please Note That, the Efficiency of Calculating Analytical Derivatives and Their Structures is Illustrated in Appendix D: www.arxiv.org/abs/2012.12941

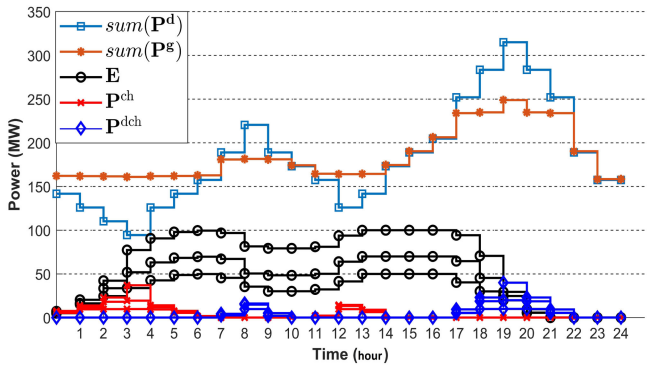

Fig. 5. IEEE Case9, $n_y = 3, T = 24$ and $n_g = 3$. Total loads and generation vs operational variables of batteries—$\boldsymbol{\mathcal{SOC}}$, $\boldsymbol{\mathcal{P}}^{\mathrm{ch}}$ and $\boldsymbol{\mathcal{P}}^{\mathrm{dch}}$.

TABLE II
STRATEGIES ON DISTRIBUTION OF STORAGE DEVICES

Case	Distribution	T (period)	n_y	No. of Iter.	Time (s)	$\frac{\text{Time}}{\text{iter}}$
A[a]	First-Last	240	10	69	145	2.10
A[a]	Last-First	240	10	90	190	2.11
A[a]	Load-Bus	240	10	67	141.9	2.11
A[a]	Fair-Dist	240	10	81	170	2.10
B[b]	First-Last	24	50	43	338	7.86
B[b]	Last-First	24	50	43	338	7.86
B[b]	Load-Bus	24	50	44	348	7.91
B[b]	Fair-Dist	24	50	43	342	7.95

[a] A:IEEE118
[b] B:PEGASE1354

and consequently $e_i^{\min} = 0$ MWh. Charge and discharge limits are considered as $(\mathcal{P}_t^{\mathrm{ch}})^{\min} = (\mathcal{P}_t^{\mathrm{dch}})^{\min} = 0$ MW and $(\mathcal{P}_t^{\mathrm{ch}})^{\max} = (\mathcal{P}_t^{\mathrm{dch}})^{\max} = 10$ MW. All charging and discharging efficiencies are taken as $\psi_i^{\mathrm{ch}} = 0.95$ and $\psi_i^{\mathrm{dch}} = 0.97$, respectively. Moreover, the initial status of storage devices is taken to be zero for all cases: $\mathbf{SOCi} = \left[\mathbb{0}\right]_{n_y \times T}$. Fig. 5 shows a typical optimisation outcome where the case study is Case9. Summation of loads and generations in each hour is depicted here with the operational strategy of energy storage systems with $n_y = 3$. Storage devices are located at buses 1, 2 and 3, where generators are located originally in the case-file. Since the objective function is a quadratic cost function, the storage devices are charging when the sum of loads is at the minimum and discharging when at the maximum. Fig. 5 is only for illustration of the optimisation outcome and does not provide more insight.

1) Independency of Distribution of Storage Devices and Computational Performance: Optimisation problems are solved for different distribution of storage devices at buses. Various types of scenarios are tested in order to verify that the distribution of storage devices does not have an impact on the overall computational time of each benchmark for convergence. In this respect, Table II illustrates the iterations and overall time spent in solving each case study with specific strategies for the distribution of storage devices. In Table II, First-Last strategy

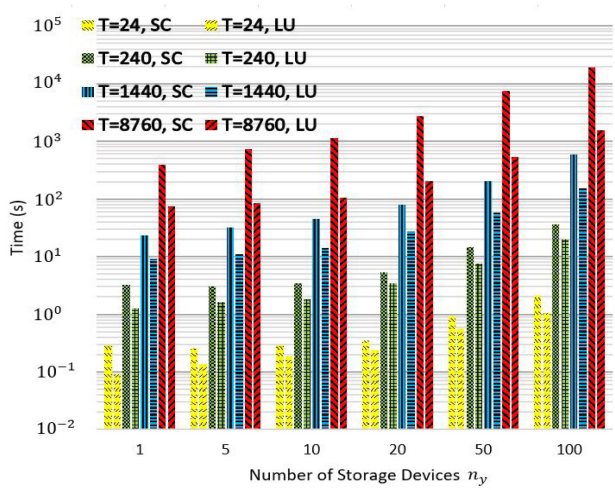

Fig. 6. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on Case9.

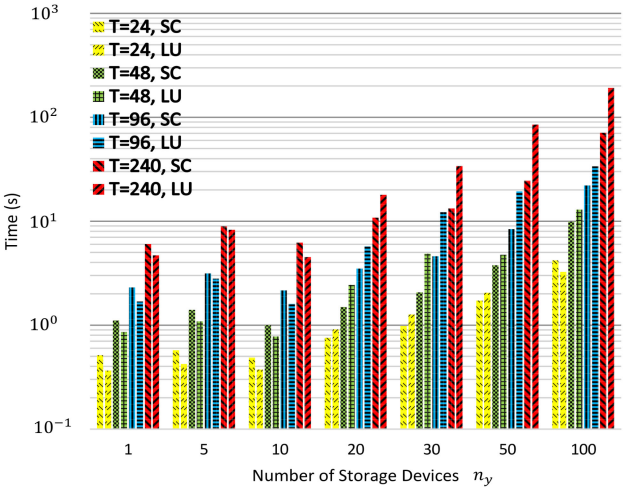

Fig. 7. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on IEEE30 .

is for the one with distribution from first bus number to the n_y^{th} bus number when $n_y \leq n_b$. If $n_y > n_b$, then First-Last strategy repeats until all storage devices are located at the buses. For the Last-First strategy, the starting point is from the last bus gradually to the first bus. The Load-Bus strategy is for locating the storage devices at the buses that only have non-zero load in their original case-files: bus i adopts one storage if $p_i^{\mathrm{d}} \neq 0$; this process repeats until all storage devices have been located. The last strategy is Fair-Dist, which uniformly distributes the storage devices among the buses. For instance if we have $n_b = 100$ and $n_y = 10$ then every tenth bus adopts a storage device. Table II shows that the time spent for each iteration, and for each scenario explained above, is very similar.

2) Reported Time in the Benchmark: All the reported time values in this paper are shown in Appendix D (Table VII) and Figs. 6, 7, 8, 9, 13, 14 and 15 show the total time taken

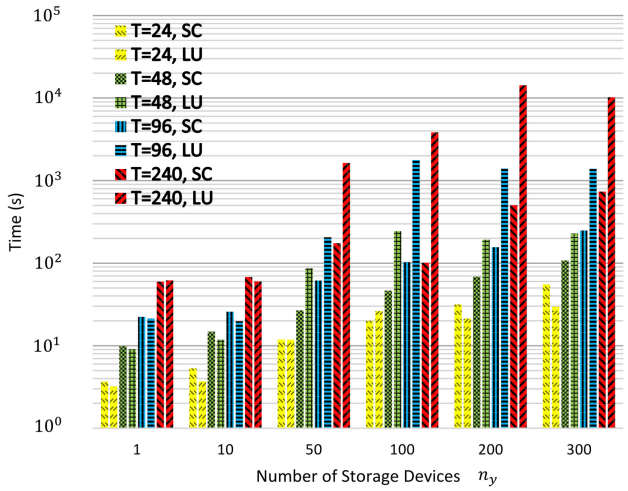

Fig. 8. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on IEEE118.

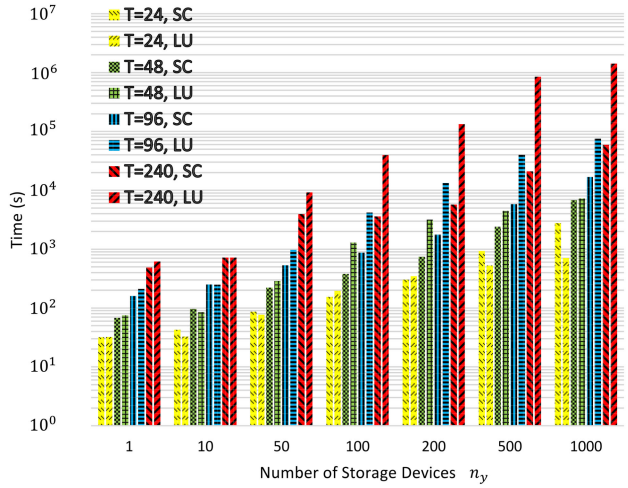

Fig. 9. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on PEGASE1354.

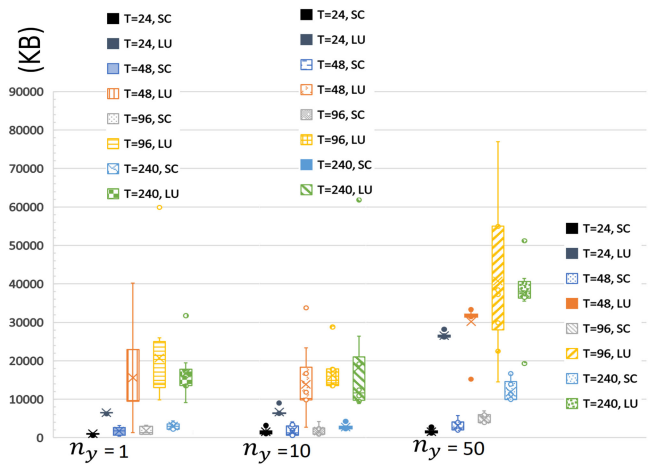

Fig. 10. Peak Memory (KB) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, IEEE 118, where $n_y = 1$, $n_y = 10$, and $n_y = 50$.

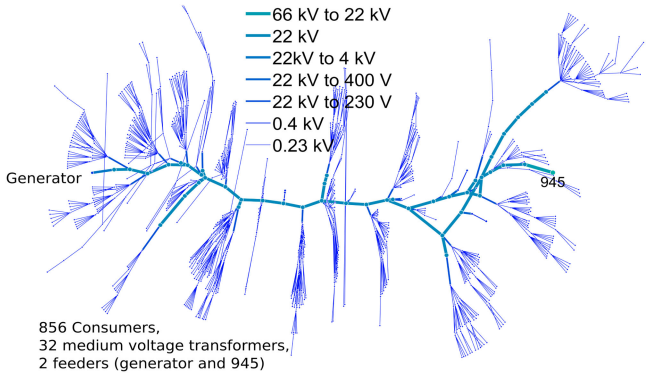

Fig. 11. Local distribution grid located in Norway with 856 costumers. using the visulaisation technique from [47].

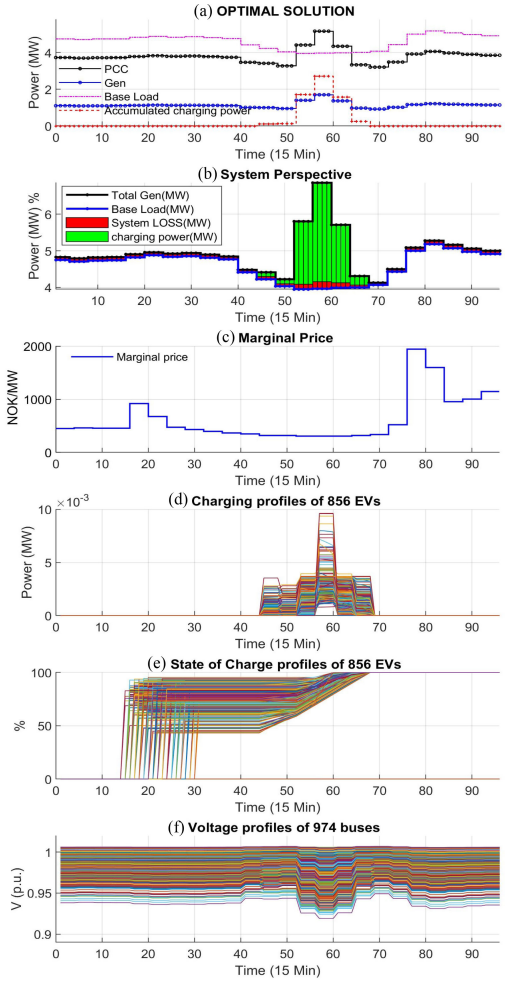

Fig. 12. Outcome of optimisation of large local distribution grid in mid-Norway, Data for 12:00 PM Feb 1, 2012, until 12:00 PM Feb 2, 2012, with highest pick of electricity price in the year of 2012: a) General perspective of optimisation, total hourly consumption profile, shown as Base Load and optimal production profile of PCC and generator, plus accumulated charging power of 856 EV, 1 EV per costumer. b) accumulation of total generation vs base load and in between two curves, losses in red and charging power in green c) hourly spot price, 8:00 am of Feb 2, 2012, is highest price of the year 2012 d) charging profile of 856 EVs. Outcome of optimisation suggests charging times and power values such that to compromise between total cost and total loss. e) state of charge of 856 EV, f) voltage fluctuations of 974 buses.

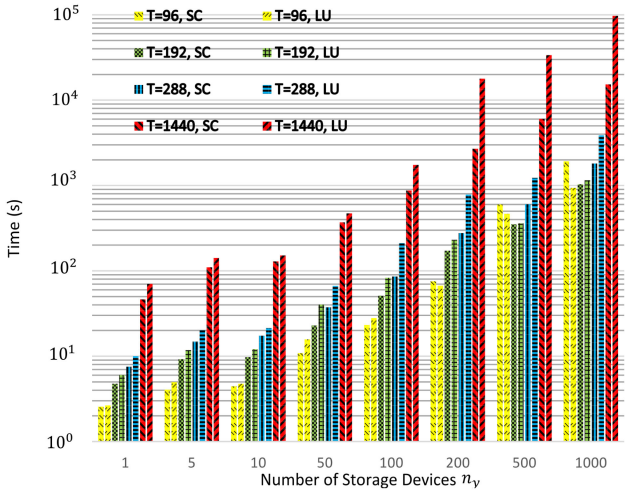

Fig. 13. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on CASE85.

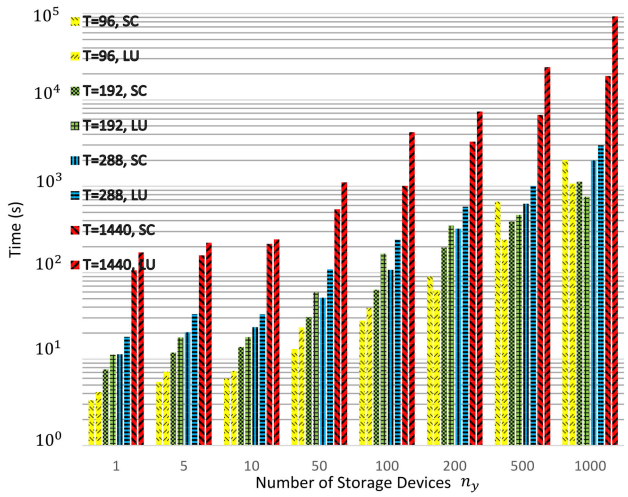

Fig. 14. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on CASE141.

for different benchmarks, which is the elapsed time to execute an algorithm until the optimum point is found. In other words, total time is TotalTime = No.of Iter. × TimePerIter. It should be kept in mind that the same number of iterations is considered for benchmarks reported and compared in Appendix D* (Table VII) and Figs. 6, 7, 8, 9, 13, 14 and 15. Moreover, the selected strategy is First-Last distribution for all the above benchmarks.

Lastly, it should be noted that different distribution strategies have a different impact on the outcome of optimum operational values, such as objective function, generator scheduling, total losses, and voltage fluctuations in the grid.

3) Linear Algebra Overhead of SESS $\mathbf{AX} = \mathbf{B}$*:* As we stated in Section IV, the most computationally expensive part of the IP algorithm is the solution of linear algebraic equations of the KKT system, which is similar to $\mathbf{AX} = \mathbf{B}$, where $\mathbf{A}$ is

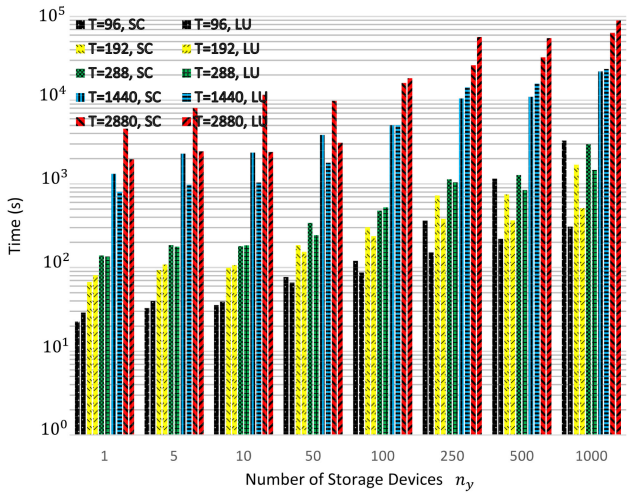

Fig. 15. Total time (TotalTime = No.of Iter. × TimePerIter) for solution of the linear KKT systems of (34) solved by Schur-Complement algorithm vs direct sparse LU solver, applied on Mid-Norway local Norwegian distribution grid.

the coefficient matrix, and $\mathbf{B}$ is the right-hand side vector. In order to assess the computational performance of the Schur-Complement technique, here we compare the performance of the tailored algorithm with direct sparse LU factorisation of complete structure of Eq. (34) (well-known arrowhead structure) and consequently the forward and backward solution, to present the computational time for the two algorithms implemented in the same platform. Note that most of the current IP-based solvers such as IPOPT, MIPS and KNITRO embed a direct solution method (LU/LDL). The main focus of the numerical results in this section is short-term horizon $T < 240$ for IEEE30, IEEE118 and PEGASE1354 since full ACOPF would be more application oriented within short time-periods.

Fig. 6 depicts the computational time needed to solve Eq. (34) of Case9 with Schur-Complement and a direct sparse-LU solver for number of storage devices $n_y = 1, 5, 10, 20, 50$ and 100; each case takes into account time horizons of $T = 24, 240, 1440$ and 8760. Direct sparse-LU solver outperforms in all cases as n_y and T increase. This informs us that the Schur-Complement method is not efficient in comparison with a direct sparse-LU solver when the case study is a comparatively small network.

However, results for IEEE30, IEEE118 and PEGASE1354 suggest that the Schur-Complement method outperforms the direct sparse-LU solver when T significantly increases. Since very large time-periods $T > 240$ would not be considered as applied cases, we do not include them here. Moreover, for relatively small number of time periods $T \leq 24$ direct sparse-LU solver outperforms the Schur-Complement method in almost all cases [16]; therefore, we focus our results for IEEE30, IEEE118 and PEGASE1354 when the $24 < T < 240$. It can be seen that when number of storage devices increases $n_y > 10$, then the Schur-Complement method provides a more computationally efficient outcome.

Fig. 7 illustrates the comparative computational performance of the Schur-Complement solver and direct sparse-LU solver

in order to solve the IEEE30 case study. As $n_y > 10$ the Schur-Complement solver has higher performance which increases considerably when $T > 24$. Note that the direct sparse-LU solver is dominant again when $n_y > 300$, due to the reason that $n_y > \mathcal{K}$ where $\mathcal{K} \propto N_{\boldsymbol{\Upsilon}_t} = 2n_b + 2n_g + 4n_y + n_{gn} + n_{gl_t}$, where n_b, n_g, n_y, n_{gn} and n_{gl_t} are respectively the number of buses, generators, storage devices, grid non-linear equalities and grid linear equalities at time t. Put simply, when $n_y > 300$, the number of storage devices is larger than a certain number which is proportional to the size of blocks of $\boldsymbol{\Upsilon}_t$ in (34). Therefore, the computationally demanding terms would be the calculation of Schur-Complement auxiliary blocks of $\mathbf{A}_t^a$ and $\mathbf{A}_t^b$ as, respectively, in terms of $\mathbf{S}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\rho}_t^\top$ and $\boldsymbol{\Xi}_t = -\boldsymbol{\rho}_t \boldsymbol{\Upsilon}_t^{-1} \boldsymbol{\zeta}_t$ in Alg. 1.

Numerical results of IEEE118 follow approximately a similar pattern as for IEEE30 when n_y and T increase, cf. Fig. 8. It can be observed that the difference between the direct sparse LU method and Schur-Complement method gets larger in IEEE118, which in turn, proves that the latter outperforms when size of $\boldsymbol{\Upsilon}_t$ blocks in (34) becomes larger.

Fig. 9 shows the performance of the Schur-Complement method in comparison with the direct sparse LU solver where the case study is a large-scale optimisation of PEGASE1354. As expected, the Schur-Complement method outperforms the sparse-LU solver considerably when $n_y > 10$ and $T > 24$.

The simulation shows that computational time is highly dependent on the number of battery energy storage devices in the grid and the time-period. The Schur-Complement method is an efficient method to accelerate the MPOPF solution time when n_y and T are large numbers in optimisation.

B. memory Efficiency

In order to back the statement in subsection IV-D4, the maximum memory consumption is tested. Therefore, the peak memory usage of Schur-Complement method is compared with that of sparse LU solver. Fig. 10 presents the peak memory consumption to solve IEEE118, where $n_y = 1$, $n_y = 10$, and $n_y = 50$. In each n_y, four different time horizons $T = 24$, $T = 48$, $T = 96$, and $T = 240$ are tested. Moreover, each simulation test $\in \{1, .., 24\}$ is repeated 10 times, and the results are shown as box plots in Fig. 10. The results show that the average peak memory consumption of Schur-Complement method is more than 7 times less than sparse LU solver. It should be noted that increase in both T and n_y would result in higher maximum memory usage. Lastly, a larger variation of the peak memory usage is observed in the case of direct sparse-LU solver. It should be noted that LU solver is an internal MATLAB library.

C. Distribution Network With EV Storage

In this section, three distribution networks are considered for the benchmarking study: Case85 [39], Case141 [40], and a real Mid-Norway distribution grid.

References [42], [43] report that the average driving distance is 52 km and as per reference [44], the standard deviation is 22 km. EV fleet's arrival and departure are derived from the work hour lifestyle survey results presented in [45]. A summary of the data for EV charge profile generation is provided in Table III. Note that the departure times of the EV owners are calculated as 9.5 hours after their arrival time by considering 8 hours of working and 1.5 hours for total commuting time.

TABLE III
DATA FOR EV CHARGE PROFILE GENERATION

Mean daily drive distance	52 km
Standard deviation of daily drive distance	22 km
Standard deviation of daily drive distance distribution	10% [1]
Percentage of EV population that consume $\leq$ 18 kWh/100km	80%
Percentage of EV population that consume $\geq$ 18 kWh/100km	20%
Mean arrival time for the EV population	17:00 hours
Standard deviation of arrival time for the EV population	90 min
Standard deviation of daily arrival time for individual EV	15 min
Percentage of 230V, 10A chargers	70%
Percentage of 230V, 16A chargers	20%
Percentage of 230V, 48A chargers	10%

[1] of daily drive distance

TABLE IV
TIME RESOLUTION IN DIFFERENT OPTIMISATION SESSIONS

T	Δt
96	15 min
192	7.5 min
288	5 min
1440	1 min
2880	30 sec

The number of EV, departure time, and arrival time are selected as an input and solved for one period of 24 hours. One full EV optimisation period is applied for the entire simulation from 12:00 PM until 12:00 PM the next day. Time resolution in each profile can be seen in Table IV . Three types of charger power capacity are chosen and shown in Table III. Discharge is considered to be inactive for all optimisation senarios: $\mathbf{CONDI} = [\mathbb{0}]_{n_y \times T}$, input matrices of $\mathbf{AVBP}$ and $\mathbf{CONCH}$ are considered to be similar, i.e. $\mathbf{AVBP} = \mathbf{CONCH}$, which are derived from arrival and departure distribution functions. The initial state of charge input $\mathbf{SOCi}$ is calculated according the distance each EV has traveled such that $[\mathbf{SOCi}_i]^{\mathrm{day_n}} = [e_i^{\max} - \mathrm{Energy(x)}]^{\mathrm{day_{n-1}}}$, where $\mathrm{Energy(x)}$ is a function that calculates the energy roughly consumed. Lastly, the departure state of charge is controlled through the last input matrix $\mathbf{SOCMi}$, which is the minimum state of charge, cf. the box constraint introduced as $\mathbf{SOCMi}_t \leq \boldsymbol{\mathcal{SOC}}_t \leq \boldsymbol{\mathcal{SOC}}^{\max}$, such that $\mathbf{SOCi}_{i,t} = \boldsymbol{\mathcal{SOC}}_i^{\max}$ if $\{\mathbf{AVBP}_{i,t} = 1 \wedge \mathbf{AVBP}_{i,t+1} = 0\} \vee \{\mathbf{AVBP}_{i,t=T} = 1\}$.

1) Standard Distribution Cases: Case85 and Case141 are open-source radial distribution cases in the MATPOWER data folder; they are 11 kV and 12.5 kV medium voltage distribution grids, respectively. They are each fed by only one feeder, cf. Table I. Linear cost functions are chosen for both of them such that $f(\boldsymbol{\mathcal{P}}^{\mathrm{g}}) = \boldsymbol{\mu}^\top \boldsymbol{\mathcal{P}}^{\mathrm{g}}$ where $\boldsymbol{\mu} \in \mathbb{R}^{T \times 1}$ is a vector of marginal price from 12:00 PM until 12:00 PM the next day, randomly selected from the Nordpool [46] spot price, $\boldsymbol{\mathcal{P}}^{\mathrm{g}} \in \mathbb{R}^{T \times 1}$ is the

power bought from the upstream network, and T is the optimisation horizon. $\boldsymbol{\mathcal{P}}^{\mathrm{d}}$ and $\boldsymbol{\mathcal{Q}}^{\mathrm{d}}$ of Case85 and Case141 are calculated similar to the procedure in Section V-A. Arrival and departure timetables of EV are calculated according to the description in Section V-C.

2) Local Mid-Norway Distribution Grid: The real-case Mid-Norway distribution grid is a large distribution case study, shown in Fig. 11, which is a 22 kV medium voltage to 230 low voltage grid, fed by: 1) a high-voltage 66 kV feeder, Point of Common Coupling (PCC), shown as red circle dot, and 2) a local generator, shown as light blue circle dot. Cost functions of PCC and generator are similar and are a linear function of $f(\boldsymbol{\mathcal{P}}^{\mathrm{g^{PCC}}}, \boldsymbol{\mathcal{P}}^{\mathrm{g^{gen}}}) = \boldsymbol{\mu}^{\top}(\boldsymbol{\mathcal{P}}^{\mathrm{g^{PCC}}} + \boldsymbol{\mathcal{P}}^{\mathrm{g^{gen}}})$ where $\boldsymbol{\mu} \in \mathbb{R}^{T\times 1}$ is the marginal hourly spot price (NOK/MW). We assumed that the feeder and generator have similar hourly cost functions. The network has 32 MV/LV transformers (shown as dark blue squares in Fig. 11) and feeds 856 registered consumers in the low-voltage area. $\boldsymbol{\mathcal{P}}^{\mathrm{d}}$ and $\boldsymbol{\mathcal{Q}}^{\mathrm{d}}$ for Mid-Norway grid are acquired from the local DSO and are hourly real consumption data of 856 consumers.

Fig. 12 depicts the optimisation outcome for an EV period of 12:00 PM Feb 1, 2012, until 12:00 PM Feb 2, 2012, where optimisation resolution is 15 minutes, and thus $T = 96$. Fig. 12 a) shows the overall picture of a day with the base load of consumers, and optimal production from the generator and PCC as well as optimal charging. Fig. 12 b) provides more insight into the optimisation results, where the optimum is a middle ground between: 1) placing all EV chargings with highest capacity of charge for the lowest price (shown in Fig. 12 c)), and 2) minimising losses at the same time. The compromise outcome is a pick of charging EV distributed between the time index of 45 until 68 and in a relatively sharp manner. It should be kept in mind that if the same optimisation formulation is run with DCOPF, then the charging pick would be sharp (all in the lowest price timestep), such that the loss would not be seen. Fig. 12 d) and e) illustrate the charge and state of charge profile of each EV owner, which total 856. Lastly, Fig. 12 f) shows the voltage variation of 974 buses over 24 hours and 96 timesteps.

3) Linear Algebra Overhead of EV $\mathbf{AX} = \mathbf{B}$*:* Similar to Section V-A3, the computational performance of Schur-Complement algorithms of Algorithm 1 and Algorithm 2 are compared with that of a direct sparse LU solver. Despite the fact that the Schur-Complement structures of SESS and EV are different, the main difference could be that EV have lower number of coupling constraints and some simulation hours could be completely decoupled; it is more efficient to solve them separately from coupled times. Here in this paper, we solve one EV period completely, both with the Schur-Complement solver and the direct sparse LU solver.

Figs. 13 and 14 show the computational time to solve a similar structure of (34) with the Schur-Complement algorithm vs the direct sparse LU solver, where number of EVs n_y are increased from 1 to 1000 in the benchmark case study of Case85, with the strategy of First-Last as shown in Table II. The results follow the same pattern as that of optimisation of SESS in IEEE118 shown in fig. 8.

In small time horizons when n_y increases the efficiency of computing of Schur-Complement algorithm surpass that of the direct sparse LU solver until a certain point, and then decreases with a slope again. This is more evident when $T = 96$ and $n_y > 100$. Furthermore, it can be seen $T = 192$ and $n_y > 200$.

The Mid-Norway case study is an interesting case where the direct LU solver mostly performs better. This is due to the fact the T times LU factorisation of the block $\boldsymbol{\Upsilon}_t$ in Algorithm 1 of the Schur-Complement algorithm is more computationally expensive than the solution of the entire coefficient matrix of (34). This is due to the fact that the number of coupled blocks of $\boldsymbol{\Upsilon}_t$ is reduced, which in turn is because of input matrices, defined by the arrival and departure of EV. In fact the ratio of coupled blocks can be calculated as $\frac{[\text{average of dep time steps} - \text{average of arrival time steps}]}{\text{total time steps}} = 0.52$ which shows that only 52% of blocks are coupled. One more interesting aspect is the clear observation of the turning point, when $n_y = 100$ for two time horizons of $T = 1440$ and $T = 2880$, such that when $n_y < 100$ the direct sparse LU solver performs better, and when $n_y > 100$ then the Schur-Complement is supreme.

VI. Conclusion and Future Work

A high performance and memory-efficient multi-period ACOPF solver based on a primal-dual IP method is proposed in this paper. In order to boost the computational performance, two mathematical approaches have been investigated. Partial derivatives of linear and non-linear constraints, objective function, and KKT conditions have been extracted analytically and consequently their sparse structures have been explored and exploited. A tailored algorithm has been suggested, using a new re-ordering format, in order to solve the sparse multi-period structure of Newton step in the IP method, with high computational performance. From the numerical results, the performance of the proposed Schur-Complement method is compared with a general sparse LU solver. Numerical results suggest that a tailored Schur-Complement algorithm could be computationally supreme in a problem with certain specifications, such as (1) large networks (large number of bus and branches) (2) different optimisation horizon (T), and (3) large number of storage devices. In future works, we propose a parallelised Schur-Complement algorithm and benchmark it thoroughly.

Appendix A
BATTPOWER Input

BATTPOWER input matrices are introduced and elaborated in this section. The main introduced input is matrix of $\mathbf{BATT}$, which represents a connection matrix of $n_y \in \mathbb{N}$ energy storage devices. It contains charge and discharge rates and efficiencies of storage devices along with their maximum and minimum energy capacities. Moreover, it includes initial points of the charge, discharge, and state of charge variables. Table V summarises the input matrices fed into the BATTPOWER solver. Note that $T \in \mathbb{N}$ is the time period is optimisation and t is a time in the interval of $t \in \{1, \ldots, T\}$.

TABLE V
DEFINITION OF INPUT MATRICES

Input	Size of Matrix n	Size of Matrix m	Description
$\mathbf{BUS}$	n_b	[1]	Examples can be found in [22]
$\mathbf{BRANCH}$	n_l	[1]	Examples can be found in [22]
$\mathbf{GEN}$	n_g	[1]	Examples can be found in [22]
$\mathbf{GENCOST}$	n_g	[1]	Examples can be found in [22]
$\mathbf{BATT}$	n_y	[1]	BATT_BUS, SOC_OPT, PCH_OPT, PDICH_OPT, Q_INJ_OPT, $\boldsymbol{\mathcal{SOC}}^{\max}$, $\boldsymbol{\mathcal{SOC}}^{\min}$, $(\boldsymbol{\mathcal{Q}}^{\mathrm{s}})^{\max}$, $(\boldsymbol{\mathcal{Q}}^{\mathrm{s}})^{\min}$, MBASE, $(\boldsymbol{\mathcal{P}}^{\mathrm{ch}})^{\max}$, $(\boldsymbol{\mathcal{P}}^{\mathrm{dch}})^{\max}$ EFF_CH ($\boldsymbol{\Psi}^{\mathrm{ch}}$) EFF_DICH ($\boldsymbol{\Psi}^{\mathrm{dch}}$)
$\mathbf{AVBP}$	n_y	T	$\mathbf{AVBP} \in \mathbb{B}^{n_y \times T}$ ($\mathbb{B}$ is a binary set.)[2] which is the availability matrix of active power provision of storage devices, such that $\mathbf{AVBP}_{i,t} = 1$ if the i^{th} storage at t^{th} time is available and connected to the grid, otherwise $\mathbf{AVBP}_{i,t} = 0$, where T is the optimisation horizon.
$\mathbf{CONCH}$	n_y	T	$\mathbf{CONCH} \in \mathbb{B}^{n_y \times T}$ is the charge connectivity matrix in which $\mathbf{CONCH}_{i,t} = 1$ if the i^{th} storage at t^{th} time has a charging option, otherwise $\mathbf{CONCH}_{i,t} = 0$.
$\mathbf{CONDI}$	n_y	T	$\mathbf{CONDI} \in \mathbb{B}^{n_y \times T}$ is the discharge connectivity matrix such that $\mathbf{CONDI}_{i,t} = 1$ if the i^{th} storage at t^{th} time has the available discharging option, otherwise $\mathbf{CONDI}_{i,t} = 0$[3].
$\mathbf{AVBQ}$	n_y	T	$\mathbf{AVBQ} \in \mathbb{B}^{n_y \times T}$ is the availability matrix of reactive power provision of storage devices such that $\mathbf{AVBQ}_{i,t} = 1$ if the i^{th} storage at t^{th} time has the available option for reactive power provision, otherwise $\mathbf{AVBQ}_{i,t} = 0$.
$\mathbf{AVG}$	n_g	T	$\mathbf{AVG} \in \mathbb{B}^{n_g \times T}$ which is the availability matrix of generators within the optimisation time horizon and consequently $\mathbf{AVG}_{i,t} = 1$ if the i^{th} generator at t^{th} time is available to inject power in the grid.
$\mathbf{SOCi}$	n_y	T	$\mathbf{SOCi} \in \mathbb{R}^{n_y \times T}$ is the matrix consisting of initial state of charge of n_y storage devices over time $t \in \{1, ..., T\}$. A value for initial state of charge $\{0 \leq \mathbf{SOCi}_{i,t} \leq 1\}$ is allocated for the i^{th} storage device at time t if and only if one of these conditions is satisfied: 1) $\mathbf{AVBP}_{i,t=1} = 1$. 2) $\mathbf{AVBP}_{i,t-1} = 0$ and $\mathbf{AVBP}_{i,t} = 1$ (arrival definition), otherwise $\mathbf{SOCi}_{i,t} = 0$.
$\mathbf{SOCMi}$	n_y	T	$\mathbf{SOCMi} \in \mathbb{R}^{n_y \times T}$ matrix which includes the minimum state of charge of n_y storage devices through time $t \in \{1, ..., T\}$. The state of charge of the i^{th} storage device at the departure time of t can be settled if one of these two conditions is satisfied: 1) $\mathbf{AVBP}_{i,t} = 1$, $\mathbf{AVBP}_{i,t+1} = 0$. 2) $\mathbf{AVBP}_{i,t=T} = 1$.
$\mathbf{PD}$	n_b	T	Time series of active loads.
$\mathbf{QD}$	n_b	T	Time series of reactive loads.

[1]User Defined. [2] $\mathbb{B}$ is a binary set. [3]Note that $\mathbf{AVBP}_{i,t} = 0$ means that the i^{th} storage\EV at time t is not available; therefore, the same element in charge and discharge connectivity matrices must be zero: $\mathbf{CONCH}_{i,t} = 0$ and $\mathbf{CONDI}_{i,t} = 0$. The converse logic is not valid.

ACKNOWLEDGMENT

The authors acknowledge the contributions of Iver Bakken Sperstad and Venkatachalam Lakshmanan, researchers at SINTEF Energy Research, Norway, and Jamshid Aghaei, a former postdoctoral researcher at NTNU.

REFERENCES

[1] J. Carpentier, "Contribution a l'etude du dispatching economique," *Bull. De La Societe Francaise Des Electriciens*, vol. 3, no. 1, pp. 431–447, 1962.
[2] K. M. Chandy, S. H. Low, U. Topcu, and H. Xu, "A simple optimal power flow model with energy storage," in *Proc. 49th IEEE Conf. Decis. Control*, Dec. 2010, pp. 1051–1057.
[3] I. B. Sperstad and H. Marthinsen, "Optimal power flow methods and their application to distribution systems with energy storage: A survey of available tools and methods," Rep. No. TR A7604, SINTEF Energy Research, Trondheim., 2016.
[4] F. Capitanescu, "Critical review of recent advances and further developments needed in ac optimal power flow," *Electric Power Syst. Res.*, vol. 136, p. 57–68, 2016.
[5] G. Carpinelli, G. Celli, S. Mocci, F. Mottola, F. Pilo, and D. Proto, "Optimal integration of distributed energy storage devices in smart grids," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 985–995, Jun. 2013.
[6] P. Fortenbacher, M. Zellner, and G. Andersson, "Optimal sizing and placement of distributed storage in low voltage networks," in *Proc. Power Syst. Comput. Conf.*, Jun. 2016, pp. 1–7.
[7] P. Fortenbacher, J. L. Mathieu, and G. Andersson, "Modeling and optimal operation of distributed battery storage in low voltage grids," *IEEE Trans. Power Syst.*, vol. 32, no. 6, pp. 4340–4350, Nov. 2017.
[8] F. Geth, S. Leyder, C. Del Marmol, and S. Rapoport, "The PlanGridEV distribution grid simulation tool with EV models," in *Proc. CIRED Workshop*, Jun. 2016, pp. 1–4.
[9] J. Warrington, P. Goulart, S. Mariéthoz, and M. Morari, "A market mechanism for solving multi-period optimal power flow exactly on AC networks with mixed participants," in *Proc. Amer. Control Conf.*, Jun. 2012, pp. 3101–3107.
[10] A. Gopalakrishnan, A. U. Raghunathan, D. Nikovski, and L. T. Biegler, "Global optimization of multi-period optimal power flow," in *Proc. Amer. Control Conf.*, Jun. 2013, pp. 1157–1164, iSSN: 2378–5861.
[11] S. Moghadasi and S. Kamalasadan, "Real-time optimal scheduling of smart power distribution systems using integrated receding horizon control and convex conic programming," in *Proc. IEEE Ind. Appl. Soc. Annu. Meeting*, Oct. 2014, pp. 1–7, iSSN: 0 197–2618.
[12] S. Moghadasi and S. Kamalasadan, "Optimal fast control and scheduling of power distribution system using integrated receding horizon control and convex conic programming," *IEEE Trans. Ind. Appl.*, vol. 52, no. 3, pp. 2596–2606, May 2016.
[13] H. Wang, C. E. Murillo-Sanchez, R. D. Zimmerman, and R. J. Thomas, "On computational issues of market-based optimal power flow," *IEEE Trans. Power Syst.*, vol. 22, no. 3, pp. 1185–1193, Aug. 2007.
[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, Mar. 2006.
[15] R. H. Byrd, J. Nocedal, and R. A. Waltz, "Knitro: An integrated package for nonlinear optimization," in *Large-Scale Nonlinear Optimization, Ser. Nonconvex Optimization and Its Applications*, G. Di Pillo and M. Roma, Eds. Boston, MA: Springer US, 2006, pp. 35–59.
[16] D. Kourounis, A. Fuchs, and O. Schenk, "Towards the next generation of multiperiod optimal power flow solvers," *IEEE Trans. Power Syst.*, vol. 33, no. 4, pp. 4005–4014, Jul. 2018.
[17] I. B. Sperstad and M. Korpås, "Energy storage scheduling in distribution systems considering wind and photovoltaic generation uncertainties," *Energies*, vol. 12, no. 7, p. 1231, Jan. 2019.
[18] F. Capitanescu and L. Wehenkel, "Experiments with the interior-point method for solving large scale optimal power flow problems," *Electric Power Syst. Res.*, vol. 95, pp. 276–283, Feb. 2013.

[19] A. Castillo and R. P. O'Neill, "Computational performance of solution techniques applied to the ACOPF," *Federal Energy Reg. Commission, Optimal Power Flow Paper*, vol. 5, 2013.
[20] N. Meyer-Huebner, M. Suriyah, and T. Leibfried, "On efficient computation of time constrained optimal power flow in rectangular form," in *Proc. IEEE Eindhoven PowerTech*, Jun. 2015, pp. 1–6.
[21] S. Zaferanlouei, M. Korpås, J. Aghaei, H. Farahmand, and N. Hashemipour, "Computational efficiency assessment of multi-period AC optimal power flow including energy storage systems," in *Proc. Int. Conf. Smart Energy Syst. Technol.*, Sep. 2018, pp. 1–6.
[22] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, Feb. 2011.
[23] R. D. Zimmerman, "Ac power flows, generalized opf costs and their derivatives using complex matrix notation," 2010.
[24] J. E. Tate and T. J. Overbye, "A comparison of the optimal multiplier in polar and rectangular coordinates," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 1667–1674, Nov. 2005.
[25] R. D. Zimmerman and H. Wang, "Matpower interior point solver MIPS 1.3 User's Manual," 2016.
[26] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.; New York: Cambridge Univ. Press, 2004.
[27] F. Capitanescu, M. Glavic, D. Ernst, and L. Wehenkel, "Interior-point based algorithms for the solution of optimal power flow problems," *Electric Power Syst. Res.*, vol. 77, no. 5, pp. 508–517, Apr. 2007.
[28] Q. Jiang, G. Geng, C. Guo, and Y. Cao, "An efficient implementation of automatic differentiation in interior point optimal power flow," *IEEE Trans. Power Syst.*, vol. 25, no. 1, pp. 147–155, Feb. 2010.
[29] C. G. Petra, O. Schenk, M. Lubin, and K. Gäertner, "An augmented incomplete factorization approach for computing the Schur complement in stochastic optimization," *SIAM J. Sci. Comput.*, vol. 36, no. 2, pp. C 139–C162, Jan. 2014.
[30] C. G. Petra, O. Schenk, and M. Anitescu, "Real-time stochastic optimization of complex energy systems on high-performance computers," *Comput. Sci. Eng.*, vol. 16, no. 5, pp. 32–42, Sep. 2014.
[31] J. Kardoš, D. Kourounis, and O. Schenk, "Structure-exploiting interior point methods," in Parallel Algorithms in Computational Science and Engineering, Ser. Modeling and Simulation in Science, Engineering and Technology, A. Grama and A. H. Sameh, Eds. Springer International Publishing, pp. 63–93. [Online]. Available: https://doi.org/10.1007/978-3-030-43736-7_3
[32] "Approximate minimum degree permutation - MATLAB Amd - MathWorks Switzerland,"
[33] J. R. Gilbert, C. Moler, and R. Schreiber, "Sparse matrices in MATLAB: Design and implementation," *SIAM J. Matrix Anal. Appl.*, vol. 13, no. 1, pp. 333–356, Jan. 1992.
[34] R. P. Schulz, A. E. Turner, and D. N. Ewart, "Long term power system dynamics. volume i. summary and technical report," *Gen. Electric Co.*, Schenectady, N.Y. (USA), Tech. Rep. EPRI-90-7-0(Vol. 1), Jun. 1974.
[35] O. Alsac and B. Stott, "Optimal load flow with steady-state security," *IEEE Trans. Power App. Syst.*, vol. PAS-93, no. 3, pp. 745–751, May 1974.
[36] "pg_tca118bus." [Online]. Available: http://labs.ece.uw.edu/pstca/pf118/pg_tca118bus.htm
[37] C. Josz, S. Fliscounakis, J. Maeght, and P. Panciatici, "AC power flow data in MATPOWER and QCQP format: Itesla, RTE snapshots, and PEGASE," Mar. 2016, *arXiv:1603.01533*.
[38] S. Fliscounakis, P. Panciatici, F. Capitanescu, and L. Wehenkel, "Contingency ranking with respect to overloads in very large power systems taking into account uncertainty, preventive, and corrective actions," *IEEE Trans. Power Syst.*, vol. 28, no. 4, pp. 4909–4917, Nov. 2013.
[39] D. Das, D. P. Kothari, and A. Kalam, "Simple and efficient method for load flow solution of radial distribution networks," *Int. J. Elect. Power Energy Syst.*, vol. 17, no. 5, pp. 335–346, Oct. 1995.
[40] H. M. Khodr, F. G. Olsina, P. M. D. O.-D. Jesus, and J. M. Yusta, "Maximum savings approach for location and sizing of capacitors in distribution systems," *Electric Power Syst. Res.*, vol. 78, no. 7, pp. 1192–1203, Jul. 2008.
[41] S. Zaferanlouei, M. Korpås, H. Farahmand, and V. V. Vadlamudi, "Integration of PEV and PV in norway using multi-period ACOPF - case study," in *Proc. IEEE Manchester PowerTech*, Jun. 2017, pp. 1–6.
[42] T. Report, E. Figenbaum, and M. Kolbenstvedt, "Learning from norwegian battery electric and plug-in hybrid vehicle users - results from a survey of vehicle owners," p. 8, 2016.
[43] A. Thingvad, C. Ziras, and M. Marinelli, "Economic value of electric vehicle reserve provision in the nordic countries under driving requirements and charger losses," *J. Energy Storage*, vol. 21, pp. 826–834, Feb. 2019.
[44] T. Bretteville-Jensen, "The norwegian electric car controversy: The arguments and some empirical illustrations," no. 66, 2016.
[45] T. Sterud, "Working time in the european union: Norway." [Online]. Available: https://www.eurofound.europa.eu/publications/report/2009/working-time-in-the-european-union-norway
[46] " Nord Pool." [Online]. Available: https://www.nordpoolgroup.com/
[47] P. Cuffe and A. Keane, "Visualizing the electrical structure of power systems," *IEEE Syst. J.*, vol. 11, no. 3, pp. 1810–1821, Sep. 2017.

Salman Zaferanlouei (Student Member, IEEE) received the M.Sc. degree from the Department of Energy Engineering and Physics, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2009 and the Ph.D. degree from the Department of Electric Power Engineering (IEL), Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2020 on the topic of Integration of Electric Vehicles into Power Distribution Systems - The Norwegian Case Study, using High-Performance MultiPeriod AC Optimal Power Flow Solver. He is currently a Researcher with IEL, NTNU. His research interests include core optimisation problems, high performance computing and power system simulations, economics, and modeling.

Hossein Farahmand (Senior Member, IEEE) received the Ph.D. degree from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2012. He is currently an Associate Professor with the Department of Electric Power Engineering, NTNU, and is a Member of the Electricity Markets and Energy System Planning Research Group. His research interests include power system balancing, power market analysis, demand side management, and flexibility operation in distribution systems. He has been involved in several EU projects including INVADE H2020, EU FP7 TWENTIES, EU FP7 eHighway2050, and IRPWIND. He is the Leader of the research task on the evaluation of the value of flexibility in smart grids in the Centre for Intelligent Electricity Distribution (CINELDI).

Vijay Venu Vadlamudi (Member, IEEE) received the Ph.D. degree from the Indian Institute of Technology Bombay, Mumbai, India, in 2011. He is currently an Associate Professor with the Department of Electric Power Engineering, Norwegian University of Science and Technology (NTNU), Trondheim, Norway. He has also been the Deputy Head of Education Department since 2017, and is a Member of the Power System Operation and Analysis research group. His areas of research interest include reliability and risk based power system operation and planning practices, and probabilistic methods applied to power system analysis. He is on the Editorial Board of the *journal IET Generation, Transmission and Distribution*, and a Subject Editor for the field of power system reliability.

Magnus Korpås (Member, IEEE) received the Ph.D. degree from the Norwegian University of Science and Technology (NTNU), Trondheim, Norway, in 2004 on the topic of optimizing the use of energy storage for distributed wind energy in the power market. He is currently a Professor with the Department of Electric Power Engineering, NTNU, where he also leads the Electricity Markets and Energy System Planning Research Group. He is a Leader and active participant in several large energy research projects at national and European levels. He is a Former Research Director of the Department of Energy Systems, SINTEF Energy Research,Trondheim, Norway. From 2018 to 2019, he was a Visiting Researcher with MIT Laboratory for Information and Decision Systems. He is also the Leader of the scientific committee and the Leader of the work package on flexible resources in the power system in the Centre for Intelligent Electricity Distribution.