2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)

Power System State Estimation Using Gauss-Newton Unrolled Neural Networks with Trainable Priors

Qiuling Yang
School of Automation
Beijing Institute of Technology
Beijing 100081, China
yang6726@umn.edu

Alireza Sadeghi
ECE Dept. and Digital Tech. Center
University of Minnesota
Minneapolis, MN 55455, USA
sadeghi@umn.edu

Gang Wang
School of Automation
Beijing Institute of Technology
Beijing 100081, China
gangwang@bit.edu.cn

Georgios B. Giannakis
ECE Dept. and Digital Tech. Center
University of Minnesota
Minneapolis, MN 55455, USA
georgios@umn.edu

Jian Sun
School of Automation
Beijing Institute of Technology
Beijing 100081, China
sunjian@bit.edu.cn

***Abstract*—Power system state estimation (PSSE) aims at finding the voltage magnitudes and angles at all generation and load buses, using meter readings and other available information. PSSE is often formulated as a nonconvex and nonlinear least-squares (NLS) cost function, which is traditionally solved by the Gauss-Newton method. However, Gauss-Newton iterations for minimizing nonconvex problems are sensitive to the initialization, and they can diverge. In this context, we advocate a deep neural network (DNN) based “trainable regularizer” to incorporate prior information for accurate and reliable state estimation. The resulting regularized NLS does not admit a neat closed form solution. To handle this, a novel end-to-end DNN is constructed subsequently by unrolling a Gauss-Newton-type solver which alternates between least-squares loss and the regularization term. Our DNN architecture can further offer a suite of advantages, e.g., accommodating network topology via graph neural networks based prior. Numerical tests using real load data on the IEEE 118-bus benchmark system showcase the improved estimation performance of the proposed scheme compared with state-of-the-art alternatives. Interestingly, our results suggest that a simple feed forward network based prior implicitly exploits the topology information hidden in data.**

***Index Terms*—Regularized state estimation, trainable priors, Gauss-Newton unrolled neural networks.**

I. Introduction

Operation of power systems critically hinges on accurate power system state estimation (PSSE), which is a prerequisite for a number of tasks, such as optimal power flow, unit commitment, economic dispatch, and contingency analysis [1], [2]. However, contemporary power systems are being challenged by frequent and sizable voltage fluctuations. This is due mainly to rapid variations of renewable generation, increasing deployment of electric vehicles, and human-in-the-loop demand response incentives. Moreover, directly measuring state variables is difficult. In this context, fast and accurate state estimation (SE) is timely and of major importance to maintain a comprehensive view of the system in real time.

The goal of PSSE is to retrieve the state variables, namely complex voltages at all buses based on available system measurements, including voltage magnitudes, power flows, and power injections, offered by the supervisory control and data acquisition (SCADA) system [2]. Traditionally, the least-squares (LS) or least-absolute-value (LAV) criterion was employed to formulate the PSSE, yielding a nonlinear and nonconvex optimization problem [3]–[5]. The LAV-based estimator is known for its robustness relative to the LS one. However, due to nonconvexity and nonsmoothness, existing LAV solvers are typically slow and, hence, inadequate for real-time system monitoring [6]. On the other hand, focusing on the LS-based PSSE formulation, the Gauss-Newton iterative solver is widely employed in practice [2]. However, due to the nonconvexity and quadratic loss function, the Gauss-Newton method is sensitive to initialization and may diverge [7]. Semidefinite programming approaches trade off these burdens with computational complexities to some extent [8]. Recently, efforts have been devoted to developing data- (and model-) driven neural network (NN) solutions to bypass the nonconvex optimization hurdles in power system monitoring and control [9]–[14]. The main idea is to approximate the mapping from measurements to the state variables through a deep neural network (DNN).

Different from existing methods and motivated by recent advancements in challenging inverse imaging problems [15]–[17], in this work we develop a judiciously *regularized* state estimation problem. Specifically, we regularize the conventional LS-based PSSE formulation with a data-driven prior [15]–[18]. Deep (D) NN is advocated as prior to promote accurate, reliable, and physically meaningful PSSE solutions using historical data. Despite its advantages, the resulting regularized nonlinear LS does not admit a neat closed form solution. To handle this, an alternating minimization-based solver with Gauss-Newton iterations being as a critical algorithmic component is first developed. Unfortunately, this solver incurs a heavy compu-

The work of Q. Yang, G. Wang, and J. Sun was supported in part by the National Natural Science Foundation of China under Grants 61522303, 61720106011, 61621063, and U1613225. Q. Yang was also supported by the China Scholarship Council. The work of A. Sadeghi and G. B. Giannakis was supported in part by the National Science Foundation under Grants 1711471 and 1901134.

tational load since it requires performing matrix inversion per iteration. To accommodate real-time monitoring of large networks and building on our previous work [11], we *unroll* the proposed alternating minimization solver to construct a new DNN architecture. Our developed Gauss-Newton unrolled neural network (GNU-NN) with deep priors consists of a Gauss-Newton iteration as a basic building block, followed by a proximal step to account for the regularization term. Interestingly, upon incorporating a graph (G) NN-based prior, the proposed method exploits the structure of the underlying smart grid. Different from [11], our proposed method provides a systematic and flexible framework for incorporating prior information into standard PSSE problems.

The rest of this paper is structured as follows. Section II outlines the LS-based PSSE formulation. Section III presents a general framework for incorporating data-driven and topology-aware priors into PSSE task through trainable regularizers, followed by an alternating minimization solver for the resultant problem. Simulated tests are presented in Section IV, with concluding remarks drawn in Section V.

II. Problem Formulation

Consider a distribution grid comprising N buses (nodes) with E lines (edges) that can be modeled as a graph $\mathcal{G} := (\mathcal{N}, \mathcal{E}, \boldsymbol{W})$, where $\mathcal{N} := \{1, \ldots, N\}$ collects all buses, $\mathcal{E} := \{(n, n')\} \subseteq \mathcal{N} \times \mathcal{N}$ all lines, and $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ is a weight adjacency matrix. If $(n, n') \in \mathcal{E}$ for buses n and n', then $[\boldsymbol{W}]_{nn'} = w_{nn'}$; and $[\boldsymbol{W}]_{nn'} = 0$ otherwise, with $w_{nn'}$ denoting the impedance between the two buses. Let $V_n := v_n^r + jv_n^i$ be the complex voltage with magnitude denoted by $|V_n|$, and $P_n + jQ_n$ the complex power injection, for $n \in \mathcal{N}$. For notational brevity, column vectors $|\boldsymbol{V}| \in \mathbb{R}^N$, $\boldsymbol{P} \in \mathbb{R}^N$, and $\boldsymbol{Q} \in \mathbb{R}^N$ collect the voltage magnitudes, active and reactive power injections across all buses, respectively.

System state variables $\boldsymbol{v} := [v_1^r \ v_1^i \ \ldots v_N^r \ v_N^i]^\top \in \mathbb{R}^{2N}$ are difficult to measure directly, however typically in practice they are to be estimated using the abundant other available measurements provided by the SCADA system, including voltage magnitudes, active and reactive power injections, as well as active and reactive power flows. Let $\mathcal{S}_V$, $\mathcal{S}_P$, $\mathcal{S}_Q$, $\mathcal{E}_P$, and $\mathcal{E}_Q$ represent the sets of buses or lines where install corresponding type meters. For a compact representation, we collect the measurements from all meters into vector $\boldsymbol{z} := [\{|V_n|^2\}_{n \in \mathcal{S}_V}, \{P_n\}_{n \in \mathcal{S}_P}, \{Q_n\}_{n \in \mathcal{S}_Q}, \{P_{nn'}\}_{(n,n') \in \mathcal{E}_P}, \{Q_{nn'}\}_{(n,n') \in \mathcal{E}_Q},]^\top \in \mathbb{R}^M$. The m-th measurement is modeled using the following model

$$z_m = h_m(\boldsymbol{v}) + \epsilon_m, \quad m = 1, \ldots, M. \tag{1}$$

where, non-linear function $h_m(\boldsymbol{v}) := \boldsymbol{v}^\top \boldsymbol{H}_m \boldsymbol{v}$, maps the real-valued state vectors to the m-th SACADA measurements using a symmetric and network dependent measurement matrix $\boldsymbol{H}_m \in \mathbb{R}^{2N \times 2N}$, finally ϵ_m captures the modeling error as well as the measurement noise.

Given $\boldsymbol{z}$, the objective is to recover the state vector $\boldsymbol{v}$. Upon vectorizing (1) and adopting the least-squares criterion, PSSE can be formulated as minimizing the following nonlinear least-squares (NLS)

$$\boldsymbol{v}^* := \arg \min_{\boldsymbol{v} \in \mathbb{R}^{2N}} \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v})\|_2^2. \tag{2}$$

Numerous iterative algorithms are proposed to solve the nonconvex objective in (2), including e.g., Gauss-Newton iterations [2], and semidefinite programming-based solvers [8]. These iterative algorithms typically generate a sequence $\{\boldsymbol{v}_i\}$ by implementing a mapping from $\boldsymbol{v}_i$ to $\boldsymbol{v}_{i+1}$ with an initial $\boldsymbol{v}_0$. Hopefully the sequence $\{\boldsymbol{v}_i\}$ finally will converge to an optimal solution $\boldsymbol{v}^*$ or at worst a locally optimal point. In this paper, we will focus on the most commonly used scheme for minimizing this objective, that is the 'workhorse' Gauss-Newton PSSE iterative solver.

The Gauss-Newton method relies on Taylor's expansion to linearize the function $\boldsymbol{h}(\boldsymbol{v})$ and iteratively updates the state variables until convergence [19, Sec. 1.5.1]. Specifically, at a given point $\boldsymbol{v}_i$, the linear approximation of $\boldsymbol{h}(\boldsymbol{v})$ is given by

$$\tilde{\boldsymbol{h}}(\boldsymbol{v}, \boldsymbol{v}_i) \approx \boldsymbol{h}(\boldsymbol{v}_i) + \boldsymbol{J}_i(\boldsymbol{v} - \boldsymbol{v}_i) \tag{3}$$

where $\boldsymbol{J}_i := \nabla \boldsymbol{h}(\boldsymbol{v}_i)$ is the $M \times 2N$ Jacobian of $\boldsymbol{h}$ evaluated at $\boldsymbol{v}_i$, with $[\boldsymbol{J}_i]_{m,n} := \partial h_m / \partial v_n$. Therefore, after approximating the nonlinear function $\boldsymbol{h}(\boldsymbol{v})$ in (2), using (3) per iteration, the Gauss-Newton method finds the next iterate by solving

$$\boldsymbol{v}_{i+1} = \arg \min_{\boldsymbol{v}} \ \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v}_i) - \boldsymbol{J}_i(\boldsymbol{v} - \boldsymbol{v}_i)\|^2. \tag{4}$$

Clearly, the subproblem (4) is convex quadratic. If matrix $\boldsymbol{J}_i^\top \boldsymbol{J}_i$ is invertible, the closed-form solution is readily available as

$$\boldsymbol{v}_{i+1} = \boldsymbol{v}_i + \left(\boldsymbol{J}_i^\top \boldsymbol{J}_i\right)^{-1} \boldsymbol{J}_i^\top (\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v}_i)). \tag{5}$$

In practice however, per iteration matrix inversion $(\boldsymbol{J}_i^\top \boldsymbol{J}_i)^{-1}$ leads to a high computational complexity. More importantly, sensitivity to initialization limits its reliability in practice. These challenges inhibit its use for real-time monitoring and control, especially in large-scale networks. To bypass these hurdles, first we will incorporate prior information into the PSSE task through judiciously designed regularization, and then will develop an end-to-end DNN to solve the regularized PSSE problem, circumventing the need for iteratively solving the nonconvex and regularized PSSE problem.

III. Regularized PSSE with Deep Priors

In this section, we first put forth a framework to incorporate flexible regularizer in the PSSE problems formulated in (2). Then an alternating minimization-based solver is developed to solve the resultant regularized PSSE. Subsequently, we construct an end-to-end DNN architecture by unrolling the alternating minimization solver. Such a novel DNN consists several layers of unrolled Gauss-Newton iterations followed by proximal steps accounting the regularization term.

A. Deep NN based regularizer

As mentioned earlier, in practice, recovering $\boldsymbol{v}$ from $\boldsymbol{z}$ can be ill posed due to e.g., lack of observability, for instance when $\boldsymbol{J}_i$ is a rectangular matrix. To cope with such a challenge, we regularize the PSSE loss (2) with a *trainable* prior as follows

$$\min_{\boldsymbol{v} \in \mathbb{R}^{2N}} \ \underbrace{\|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v})\|_2^2}_{\text{data consistency}} + \lambda \ \underbrace{\|\boldsymbol{v} - \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v})\|_2^2}_{\text{regularizer}} \tag{6}$$

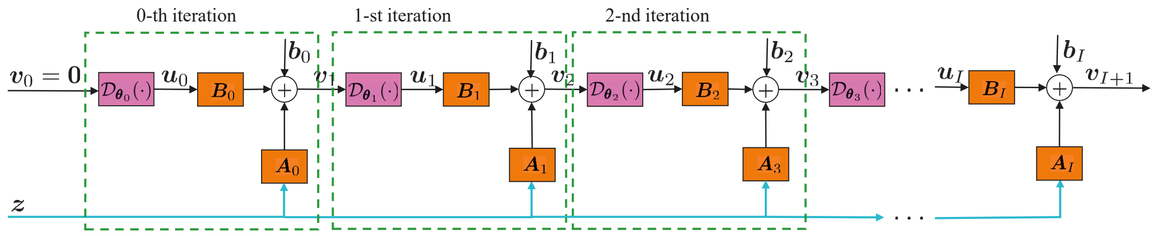


Fig. 1: The structure of the proposed GNU-NN.

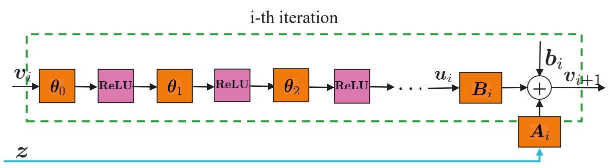


Fig. 2: Plain-vanilla FNN based regularizer.

where $\lambda > 0$ is a hyper-parameter to tune the regularizer term promoting states $\boldsymbol{v}$ to reside close to $\mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v})$. The latter could be considered as nonlinear $\hat{\boldsymbol{v}} := \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v})$ estimator (obtained possibly offline); For instance one may employ a quadratic-linear function of the form $\mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}) = \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{v}\, \boldsymbol{\theta} + \boldsymbol{\theta}^\top \boldsymbol{v}$ as the prior. In this paper, for the sake of expressibility and to encompass a large family of priors, we advocate a DNN-based estimator $\mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v})$, where $\boldsymbol{\theta}$ collects all weights of the NN, that can be learned offline using historical data. This is stemmed from data-driven deep priors advocated in image denoising [15]–[17].

Although this innovative regularized formulation tackles the ill conditioning, the nonconvexity of PSSE objective (6) still remains a challenge. Moreover, the nested structure of $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ brings further challenges, since still one needs to find $\nabla_{\boldsymbol{v}} \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v})$, which is not readily available specifically when $\mathcal{D}_{\boldsymbol{\theta}}$ is a DNN.

To overcome this limitation, we use an alternating minimization algorithm to iteratively approximate the solution of (6), which mimics the Gauss-Newton method for NLS in (2). In particular, starting with an initial guess $\boldsymbol{v}_0$, a linearized data consistency term is introduced at each iteration i to obtain the state at next iteration $\boldsymbol{v}_{i+1}$; that is,

$$\begin{aligned}\boldsymbol{v}_{i+1} &= \arg\min_{\boldsymbol{v}} \|\boldsymbol{z} - \boldsymbol{h}(\boldsymbol{v}_i) - \boldsymbol{J}_i(\boldsymbol{v} - \boldsymbol{v}_i)\|^2 + \lambda \|\boldsymbol{v} - \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}_i)\|^2 \\ &= \boldsymbol{A}_i \boldsymbol{z} + \boldsymbol{B}_i \boldsymbol{u}_i + \boldsymbol{b}_i\end{aligned}$$

where we define

$$\begin{aligned}\boldsymbol{A}_i &:= (\boldsymbol{J}_i^\top \boldsymbol{J}_i + \lambda \boldsymbol{I})^{-1} \boldsymbol{J}_i^\top \\ \boldsymbol{B}_i &:= \lambda (\boldsymbol{J}_i^\top \boldsymbol{J}_i + \lambda \boldsymbol{I})^{-1} \\ \boldsymbol{b}_i &:= (\boldsymbol{J}_i^\top \boldsymbol{J}_i + \lambda \boldsymbol{I})^{-1} \boldsymbol{J}_i^\top (\boldsymbol{J}_i \boldsymbol{v}_i - \boldsymbol{h}(\boldsymbol{v}_i)).\end{aligned}$$

Alternating between ensuing two steps

$$\boldsymbol{u}_i = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}_i) \tag{8a}$$

$$\boldsymbol{v}_{i+1} = \boldsymbol{A}_i \boldsymbol{z} + \boldsymbol{B}_i \boldsymbol{u}_i + \boldsymbol{b}_i \tag{8b}$$

until some convergence criterion is met, and a solution of (6) is reached. For instance, given measurement $\boldsymbol{z}$, and initialization $\boldsymbol{v}_0 = \boldsymbol{0}$, the $i = 0$ iteration yields $\boldsymbol{v}_1 = \boldsymbol{A}_0 \boldsymbol{z} + \boldsymbol{B}_0 \boldsymbol{u}_0 + \boldsymbol{b}_0$. The DNN $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ takes as input the $\boldsymbol{v}_1$, to generate $\boldsymbol{u}_1 = \mathcal{D}_{\boldsymbol{\theta}}(\boldsymbol{v}_1)$ according to (8a), which is also the input to $i = 1$ iteration. Hence, by repeating these alternating iterations whenever a new system measurement $\boldsymbol{z}$ becomes available, the state estimates can be obtained. Notice that every iteration i must evaluate the Jacobian matrix $\boldsymbol{J}_i$, followed by matrix inversions to form $\boldsymbol{A}_i$, $\boldsymbol{B}_i$, and $\boldsymbol{b}_i$. These steps are computationally expensive.

Encouraged by results reported in our preceding work [20], we pursue an unrolling method that builds a DNN architecture, as depicted in Fig. 1. The constructed DNN is obtained by unrolling I iterations of the proposed alternating minimizer in (8). Recall that the DNN prior information $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ in (8a) is considered pre-trained, with weight parameters $\boldsymbol{\theta}$ being fixed. In the constructed DNN in Fig. 1 however, all the coefficients $\{\boldsymbol{A}_i\}_{i=0}^I$, $\{\boldsymbol{B}_i\}_{i=0}^I$, $\{\boldsymbol{b}_i\}_{i=0}^I$, as well as the DNN weights $\{\boldsymbol{\theta}_i\}_{i=0}^I$ are considered learnable during a *single* training phase. We call this architecture as GNU-NN, since it is obtained unrolling Gauss-Newton like iterations.

During training, our GNU-NN takes as input the measurements-state pairs $\{(\boldsymbol{z}^t, \boldsymbol{v}^{*t})\}_{t=1}^T$, where $\boldsymbol{v}^{*t}$ is the ground-truth state vector. For notational brevity, we concatenate all trainable parameters of the GNU-NN in vector $\boldsymbol{\omega} := [\{\boldsymbol{A}_i\}_{i=0}^I, \{\boldsymbol{B}_i\}_{i=0}^I, \{\boldsymbol{b}_i^1\}_{i=0}^I]$. After specifying a certain loss $\ell(\boldsymbol{v}^*, \boldsymbol{v}_{I+1})$ to measure how accurate GNU-NN predicts are, the GNU-NN weights $\boldsymbol{\omega}$ can be updated using backpropagation to minimize this loss. The proposed method is tabulated in Algorithm 1.

During testing phase, one just feeds the real-time measurement $\boldsymbol{z}^t$ into the learned GNU-NN, after only a few matrix-vector multiplications, the estimated voltage $\boldsymbol{v}^t$ can be obtained. Our GNU-NN enjoys competitive estimation performance compared with other iterative algorithms, e.g., the Gauss-Newton method. Furthermore, due to skipping connections from the input layer to intermediate and output layers, our GNU-NN can avoid vanishing and exploding gradients.

Interestingly, by judiciously choosing model for $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$, desired merits can further be attained. For instance, we can use plain-vanilla feed forward (F) NNs as $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$, which is referred to GNU-FNN. The i-th iteration structure of GNU-FNN is illustrated in Fig. 2. The main advantage of FNN structure is simplicity and computational efficiency. However, it is difficult to design a decentralized algorithm using FNN. Fortunately, upon utilizing recent DNN architectures as priors, such as graph neural networks (GNNs), one can easily design decentralized algorithms and enjoy scalability. Using GNNs as priors for

Algorithm 1 PSSE Solver with NN Priors.

```
Training phase:
1:  Input: Training samples $\{(\boldsymbol{z}^t, \boldsymbol{v}^{*t})\}_{t=1}^T$
2:  Initialize:
    $\boldsymbol{\omega} := [\{\boldsymbol{\theta}_i^1\}_{i=0}^I, \{\boldsymbol{A}_i^1\}_{i=0}^I, \{\boldsymbol{B}_i^1\}_{i=0}^I, \{\boldsymbol{b}_i^1\}_{i=0}^I]$, $\boldsymbol{v}_0 = 0$.
3:  for $t = 1, 2, \ldots, T$ do
4:      Feed $\boldsymbol{z}^t$ and $\boldsymbol{v}_0$ as input into GNU-NN.
5:      for $i = 0, 1, \ldots, I$ do[1]
6:          Obtain $\boldsymbol{u}_i$ using (8a).
7:          Obtain $\boldsymbol{v}_{i+1} \in \mathbf{R}^{2N}$ via (8b).
8:      end for
9:      Obtain $\boldsymbol{v}_{I+1}^t$ from the GNU-NN output.
10:     Minimize the loss $\ell(\boldsymbol{v}^{*t}, \boldsymbol{v}_{I+1}^t)$ and update $\boldsymbol{\omega}^t$.
11: end for
12: Output: $\boldsymbol{\omega}^T$
Inference phase:
1:  for $t = T+1, \ldots, T'$ do
2:      Feed real-time $\boldsymbol{z}^t$ to the trained GNU-NN.
3:      Obtain the estimated voltage $\boldsymbol{v}^t$.
4:  end for
```

PSSE will be elaborated in ensuing subsection.

B. Graph NN based deep prior

In order to use expressive state estimators in our regularization term, we model $\mathcal{D}_{\boldsymbol{\theta}}(\cdot)$ by a GNN, which is a careful choice due to having a networked data. Recently, GNNs have demonstrated remarkable performance in several tasks specifically semi-supervised learning [21], [22]. By directly operating over the graph, GNN can explicitly utilize the power system topology information. Therefore, it is an attractive choice for parameterization in our application domain, where the data follows the power network graph structure [21].

From the graph signal processing perspective, the measurements $\boldsymbol{X} \in \mathbb{R}^{N \times F}$ can be seen as a signal on the power network graph. Its n-th row denoted by $\boldsymbol{x}_n^\top := [\boldsymbol{X}]_{n:}$ represents an $F \times 1$ feature vector per node n, where for the PSSE problem, the feature vector is $\boldsymbol{v}_n^r$ and $\boldsymbol{v}_n^i$, i.e., $F = 2$. By pre-multiplying the graph signal $\boldsymbol{X}$ from left with weighted adjacency $\boldsymbol{W}$, features are propagated over the underlying graph, yielding a diffused version $\check{\boldsymbol{Y}} \in \mathbb{R}^{N \times F}$ obtained as follows

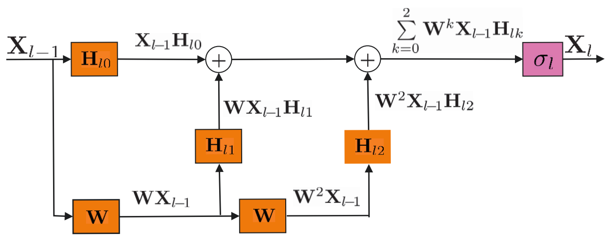
$$\check{\boldsymbol{Y}} = \boldsymbol{W}\boldsymbol{X}. \tag{9}$$

Interestingly, one can replace the weighted adjacency matrix with any matrix that preserves the structure of the power network (i.e. $\boldsymbol{W}_{nn'} = 0$ if $(n, n') \notin \mathcal{E}$), such as the graph Laplacian, the random walk Laplacian matrix, and their normalized versions.

The transformation in (9) is a feature propagation transformation. It gives the f-th feature at every node by linearly combining f-th features of neighboring nodes. For instance, the shifted f-th feature $[\check{\boldsymbol{Y}}]_{nf}$ for bus n, is given by

$$[\check{\boldsymbol{Y}}]_{nf} = \sum_{i=1}^{N} [\boldsymbol{W}]_{ni}[\boldsymbol{X}]_{if} = \sum_{i \in \mathcal{N}_n} w_{ni} x_i^f \tag{10}$$

[1]For brevity the superscript t is removed from inner iteration i.



$\mathbf{X}_{l-1}$
$\mathbf{H}_{l0}$
$\mathbf{X}_{l-1}\mathbf{H}_{l0}$
$\sum_{k=0}^{2} \mathbf{W}^k\mathbf{X}_{l-1}\mathbf{H}_{lk}$
σ_l
$\mathbf{X}_l$
$\mathbf{W}\mathbf{X}_{l-1}\mathbf{H}_{l1}$
$\mathbf{W}^2\mathbf{X}_{l-1}\mathbf{H}_{l2}$
$\mathbf{H}_{l1}$
$\mathbf{H}_{l2}$
$\mathbf{W}$
$\mathbf{W}\mathbf{X}_{l-1}$
$\mathbf{W}$
$\mathbf{W}^2\mathbf{X}_{l-1}$

Fig. 3: The signal diffuses from layer $l-1$ to l with $K = 3$.

where $\mathcal{N}_n = \{i \in \mathcal{N} : (i, n) \in \mathcal{E}\}$ represents the set of neighboring buses for bus n. Clearly, this interpretation generates a diffused $\boldsymbol{X}$ over the graph. The 'graph convolution' operation in GNNs exploits network topology to linearly combine K hop neighborhood information, as follows

$$[\boldsymbol{Y}]_{nd} := [\mathcal{H} \star \boldsymbol{X}; \boldsymbol{W}]_{nd} := \sum_{k=0}^{K-1} [\boldsymbol{W}^k \boldsymbol{X}]_{n:}[\boldsymbol{H}_k]_{:d} \tag{11}$$

where $\mathcal{H} := [\boldsymbol{H}_0 \ \cdots \ \boldsymbol{H}_{K-1}]$ with $\boldsymbol{H}_k \in \mathbb{R}^{F \times D}$ concatenates the filter coefficient parameters; $\boldsymbol{Y} \in \mathbb{R}^{N \times D}$ is the intermediate (hidden) matrix with D features each bus; and, $\boldsymbol{W}^k \boldsymbol{X}$ linearly combines features of buses within the k-hop neighborhood by recursively applying the shift operator $\boldsymbol{W}$.

To obtain a GNN with L hidden layers, let $\boldsymbol{X}_{l-1}$ denote the output of the $(l-1)$-th layer, which is also the l-th layer input for $l = 1, \ldots, L$, and $\boldsymbol{X}_0 = \boldsymbol{X}$ is the input matrix. The hidden signal $\boldsymbol{Y}_l \in \mathbb{R}^{N \times D_l}$ with D_l features is obtained by applying the graph convolution operation (11) at layer l, namely

$$[\boldsymbol{Y}_l]_{nd} = \sum_{k=0}^{K_l - 1} [\boldsymbol{W}^k \boldsymbol{X}_{l-1}]_{n:}[\boldsymbol{H}_{lk}]_{:g} \tag{12}$$

where $\boldsymbol{H}_{lk} \in \mathbb{R}^{F_{l-1} \times F_l}$ are the graph convolution coefficients for $k = 0, \ldots, K_l - 1$. The output $\boldsymbol{X}_l$ at layer l is found by applying a graph convolution followed by a point-wise nonlinear operation $\sigma_l(\cdot)$, such as the rectified linear unit (ReLu) $\sigma_l(t) := \max\{0, t\}$ for $t \in \mathbb{R}$; see Fig. 3 for an illustration. Upon rewriting (12) in a compact form, one can arrives at

$$\boldsymbol{X}_l = \sigma_l(\boldsymbol{Y}_l) = \sigma_l\left(\sum_{k=0}^{K_l - 1} \boldsymbol{W}^k \boldsymbol{X}_{l-1} \boldsymbol{H}_{lk}\right). \tag{13}$$

The GNN-based PSSE provides a nonlinear functional mapping $\boldsymbol{X}_L = \boldsymbol{\Phi}(\boldsymbol{X}_0; \boldsymbol{\Theta}, \boldsymbol{W})$ that maps the GNN input $\boldsymbol{X}_0$ to voltage estimates by taking into account the graph structure, that is

$$\boldsymbol{\Phi}(\boldsymbol{X}_0; \boldsymbol{\Theta}, \boldsymbol{W}) = \tag{14}$$
$$\sigma_L\left(\sum_{k=0}^{K_L - 1} \boldsymbol{W}^k \left(\ldots \left(\sigma_1\left(\sum_{k=0}^{K_1 - 1} \boldsymbol{W}^k \boldsymbol{X}_0 \boldsymbol{H}_{1k}\right) \ldots\right)\right) \boldsymbol{H}_{Lk}\right)$$

where the parameter set $\boldsymbol{\Theta}$ collects all the filter weights, i.e., $\boldsymbol{\Theta} := \{\boldsymbol{H}_{lk}, \forall l, k\}$, and also recall that the input $\boldsymbol{X}_0 = \boldsymbol{X}$.

To accommodate the GNN implementation over the proposed unrolled architecture, we concatenate all trainable parameters of the GNU-GNN in vector $\boldsymbol{\omega}' := [\{\boldsymbol{\Theta}_i\}_{i=0}^I, \{\boldsymbol{A}_i\}_{i=0}^I, \{\boldsymbol{B}_i\}_{i=0}^I, \{\boldsymbol{b}_i^1\}_{i=0}^I]$, which can be updated

Algorithm 2 Reshaping the inputs and outputs of GNNs.

```
1: for $i = 0, 1, \ldots, I$ do
2:     Reshape $\boldsymbol{v}_i \in \mathbf{R}^{2N}$ to get $\boldsymbol{X}_0^i \in \mathbb{R}^{N\times 2}$.
3:     Feed $\boldsymbol{X}_0^i$ into GNN.
4:     Vectorize the GNN output $\boldsymbol{X}_L^i \in \mathbb{R}^{N\times 2}$ to get $\boldsymbol{u}_i$.
5:     Obtain $\boldsymbol{v}_{i+1} \in \mathbf{R}^{2N}$ using (8b).
6: end for
```

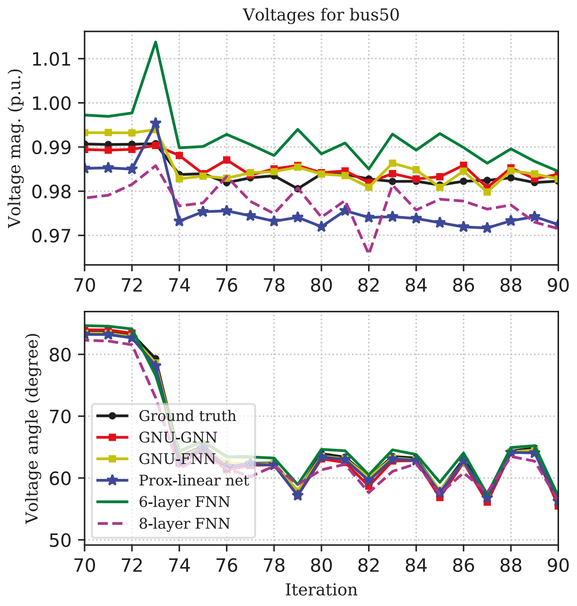


Fig. 4: The estimated voltage profiles at bus 50 from slot 70 to 90.

using backpropagation. The whole process is the same with Algorithm 1 except steps 5-8. Specifically, at the i-th iteration, we reshape the states $\boldsymbol{v}_i \in \mathbb{R}^{2N}$ to form the $N \times 2$ GNN input matrix $\boldsymbol{X}_0^i \in \mathbb{R}^{N\times 2}$. Next, to obtain the vector $\boldsymbol{u}_i \in \mathbb{R}^{2N}$, we vectorize the GNN output $\boldsymbol{X}_L^i \in \mathbb{R}^{N\times 2}$ (cf. (8a)). This difference is depicted in Algorithm 2.

IV. Numerical Tests

In this section, we used the IEEE 118-bus system to assess the performance of our proposed PSSE solver. The simulations were executed on an NVIDIA Titan X GPU with a 12GB RAM. For numerical tests, we used real load consumption data from the 2012 Global Energy Forecasting Competition (GEFC) [23], using which training and testing data were created as follows. To match the scale of power demands, we first normalized the load data. Next, we fed it into MATPOWER, to generate $1,000$ pairs of measurements and ground-truth voltages, by solving the exact AC power flow equations. Finally, we randomly selected 80% of the measurement-state pairs to be the training set and the remaining 20% to be the test set, the algorithm was then trained and tested on these sets.

Note that our GNU-GNN architecture explicitly captures the topology and the physics of the smart grid, while our GNU-FNN leverages the network topology only indirectly through simulated data. It is therefore natural to ask how much gain will be obtained using topology information explicitly? Furthermore, what are the gains of using trainable regularizer for PSSE compared with alternatives? To answer these questions, we have carried out numerical tests, where three PSSE solvers are employed as baselines, namely: i) the prox-linear network in [11]; ii) a 6-layer vanilla feed-forward (F)NN; and iii) an 8-layer FNN. The weights of all these NNs were updated using the 'Adam' optimizer [24] to minimize the Hüber loss with learning rate fixed to 10^{-3}. The training phase was carried out with 500 epochs, and the batch size was set to 32.

Our GNU-GNN and GNU-FNN were implemented by unrolling $I = 6$ iterations of the proposed alternating minimizing solver, respectively. Per unrolled iteration, a GNN with $K = 2$

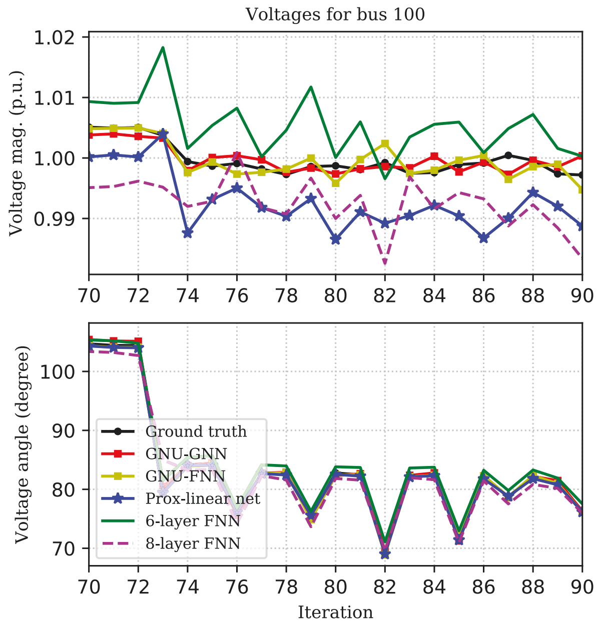


Fig. 5: The estimated voltage profiles at bus 100 from slot 70 to 90.

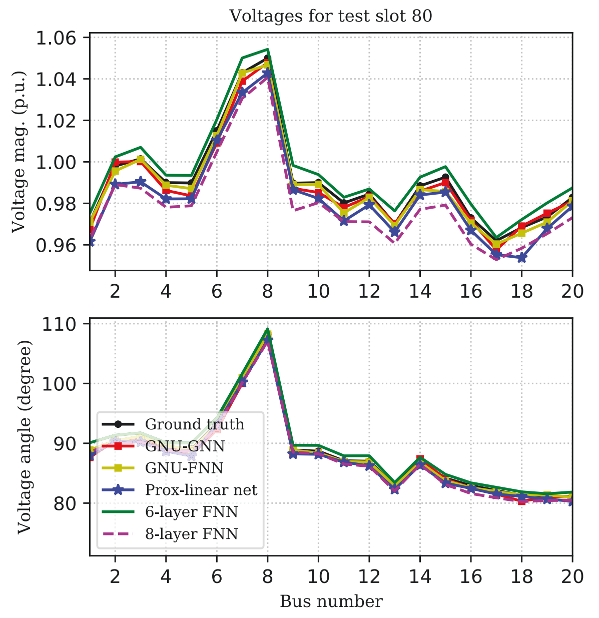


Fig. 6: The estimated voltage profiles for the first 20 buses at slot 80.

hops, and $D = 8$ hidden units with ReLU activation functions was used as the deep prior of GNU-GNN, while a FNN with one hidden layer and 8 hidden units with ReLU activation was employed as the deep prior for GNU-FNN. The GNU-GNN, GNU-FNN, and prox-linear network architectures were designed to have roughly the same number of weight parameters.

The estimated voltage profiles obtained at buses 50 and 100 from test slots 70 to 90 are reported in Figs. 4 and 5, respectively. The ground-truth and estimated voltages for the first 20 buses at slot 80 are presented in Fig. 6. These plots corroborate the improved PSSE performance using our GNU-GNN and GNU-FNN relative to alternative approaches. Furthermore, based on the results in Figs. 4-6, the GNU-FNN and GNU-GNN perform similarly. This implies that explicitly incorporating topology information through GNNs does not provide any performance gain compared with implicitly exploiting it through FNNs. This suggests that a GNN architecture may inherit unnecessary complexity and redundant computation, while a FNN offers the same performance without any need for topology information. Recently, it has been shown that successively removing nonlinearities and collapsing weight matrices between consecutive layers of a GNN architecture does not influence its performance in practice [25]. These observations suggest that measurements already contain the required information about the network topology, thus there is no need to employ a GNN to explicitly use network topology. In terms of runtime, our GNU-GNN, GNU-FNN, the prox-linear net, the 6-layer FNN, and the 8-layer FNN over 200 testing samples took 1.5×10^{-2}s, 1.3×10^{-2}s, 1.6×10^{-2}s, 2.8×10^{-2}s, and 3.9×10^{-2}s, respectively. These results corroborate the improved performance of the our GNU-GNN and GNU-FNN relative to the simulated PSSE solvers.

V. Conclusions

PSSE is an important task for monitoring and control of current smart grids, which is typically formulated as a least-absolute-value or a least-square problem, both of which are nonlinear and nonconvex. In this work, DNN-based trainable regularizers were adopted to promote accurate, reliable, and physically meaningful PSSE solutions using historical data. To obtain the solution of the regularized PSSE problem, an alternating minimization solver using Gauss-Newton iterations was introduced. This slover however, requires performing matrix inversion per iteration, thus incurring a heavy computational burden that may discourage its use for real-time monitoring of large networks. To accommodate realtime operations, we construted a new DNN architecture by unrolling the Gauss-Newton iterations, followed by a proximal step. The proposed architecture provides a principled framework for designing deep neural networks that can incorporate prior information into solving inverse problems. The merits of our proposed scheme relative to existing methods were corroborated through numerical tests using real data. This work also opens up interesting directions for future research, including using data-driven and topology-aware regularizer for optimal power flow and unit commitment problems.

References

[1] F. C. Schweppe, J. Wildes, and D. Rom, "Power system static-state estimation: Parts I, II, III," *IEEE Trans. Power App. Syst.*, vol. PAS-89, pp. 120–135, Jan. 1970.

[2] A. Abur and A. G. Exposito, *Power System State Estimation: Theory and Implementation*. New York, USA: CRC Press, 2004.

[3] G. Wang, G. B. Giannakis, J. Chen, and J. Sun, "Distribution system state estimation: An overview of recent developments," *Front. Inform. Technol. Electron. Eng.*, vol. 20, no. 1, pp. 4–17, Jan. 2019.

[4] A. S. Zamzam, Y. Liu, and A. Bernstein, "Model-free state estimation using low-rank canonical polyadic decomposition," *arXiv:2004.05741*, 2020.

[5] G. Wang, A. S. Zamzam, G. B. Giannakis, and N. D. Sidiropoulos, "Power system state estimation via feasible point pursuit: Algorithms and cramér-Rao bound," *IEEE Trans. Signal Process.*, vol. 66, no. 6, pp. 1649–1658, Mar. 2018.

[6] G. Wang, G. B. Giannakis, and J. Chen, "Robust and scalable power system state estimation via composite optimization," *IEEE Trans. Smart Grid*, vol. 10, no. 6, pp. 6137–6147, Nov. 2019.

[7] B. Blaschke, A. Neubauer, and O. Scherzer, "On convergence rates for the iteratively regularized Gauss-Newton method," *IMA J. Numer. Anal.*, vol. 17, no. 3, pp. 421–436, 1997.

[8] H. Zhu and G. B. Giannakis, "Power system nonlinear state estimation using distributed semidefinite programming," *IEEE J. Sel. Topics Signal Process.*, vol. 8, no. 6, pp. 1039–1050, Jun. 2014.

[9] E. Manitsas, R. Singh, B. C. Pal, and G. Strbac, "Distribution system state estimation using an artificial neural network approach for pseudo measurement modeling," *IEEE Trans. Power Syst.*, vol. 27, no. 4, pp. 1888–1896, Nov. 2012.

[10] P. P. Barbeiro, J. Krstulovic, H. Teixeira, J. Pereira, F. J. Soares, and J. P. Iria, "State estimation in distribution smart grids using autoencoders," in *IEEE Intl. Power Eng. and Opt. Conf.*, 2014, pp. 358–363.

[11] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 4069–4077, Aug. 2019.

[12] Q. Yang, G. Wang, A. Sadeghi, G. B. Giannakis, and J. Sun, "Two-timescale voltage control in distribution grids using deep reinforcement learning," *IEEE Trans. Smart Grid*, pp. 1–11, 2019.

[13] A. S. Zamzam and N. D. Sidiropoulos, "Physics-aware neural networks for distribution system state estimation," *IEEE Trans. Power Syst.*, pp. 1–1, 2020.

[14] Q. Yang, M. Coutino, G. Wang, G. B. Giannakis, and G. Leus, "Learning connectivity and higher-order interactions in radial distribution grids," in *Proc. Intl. Conf. Acoustics Speech Signal Process.* Barcelona, Spain: IEEE, May 4-8 2020, pp. 5555–5559.

[15] S. G. Lingala and M. Jacob, "Blind compressive sensing dynamic MRI," *IEEE Trans. Med. Imag.*, vol. 32, no. 6, pp. 1132–1145, Mar. 2013.

[16] J. Schlemper, J. Caballero, J. V. Hajnal, A. N. Price, and D. Rueckert, "A deep cascade of convolutional neural networks for dynamic MR image reconstruction," *IEEE Trans. Med. Imag.*, vol. 37, no. 2, pp. 491–503, Oct. 2017.

[17] H. K. Aggarwal, M. P. Mani, and M. Jacob, "MoDL: Model-based deep learning architecture for inverse problems," *IEEE Trans. Med. Imag.*, vol. 38, no. 2, pp. 394–405, Aug. 2018.

[18] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D: Nonlinear Phenomena*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[19] D. P. Bertsekas, *Nonlinear Programming*. 2nd ed. Belmont, MA, USA: Athena Sci., 1999.

[20] L. Zhang, G. Wang, and G. B. Giannakis, "Real-time power system state estimation and forecasting via deep unrolled neural networks," *IEEE Trans. Signal Process.*, vol. 67, no. 15, pp. 4069–4077, Aug. 2019.

[21] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv:1609.02907*, 2016.

[22] V. N. Ioannidis, A. G. Marques, and G. B. Giannakis, "Tensor graph convolutional networks for multi-relational and robust learning," *arXiv:2003.07729*, 2020.

[23] [Online]. Available: https://www.kaggle.com/c/global-energy-forecasting-competition-2012- load-forecasting/data.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[25] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv:1902.07153*, 2019.