Probabilistic Power Flow Computation via Low-Rank and Sparse Tensor Recovery

Zheng Zhang, Hung Dinh Nguyen, Konstantin Turitsyn and Luca Daniel

***Abstract*—This paper presents a tensor-recovery method to solve probabilistic power flow problems. Our approach generates a high-dimensional and sparse generalized polynomial-chaos expansion that provides useful statistical information. The result can also speed up other essential routines in power systems (e.g., stochastic planning, operations and controls).**

Instead of simulating a power flow equation at all quadrature points, our approach only simulates an extremely small subset of samples. We suggest a model to exploit the underlying low-rank and sparse structure of high-dimensional simulation data arrays, making our technique applicable to power systems with many random parameters. We also present a numerical method to solve the resulting nonlinear optimization problem.

Our algorithm is implemented in MATLAB and is verified by several benchmarks in MATPOWER 5.1. Accurate results are obtained for power systems with up to 50 independent random parameters, with a speedup factor up to 9×10^{20}.

***Index Terms*—Power flow, power system, stochastic collocation, tensors, polynomial chaos, uncertainty, optimization.**

I. Introduction

REALISTIC power systems are affected by various uncertainties, such as the randomness of generations and loads, insufficient knowledge about network parameters, and noisy measurement [1]–[14]. Uncertainties may increase in future power systems, since many renewables highly depend on the uncertain weather conditions [6], [9]. These uncertainties must be considered in simulation, such that subsequent tasks can be completed in an efficient and robust way.

This work investigates the probabilistic power flow problem [2], which quantifies the uncertainties of bus voltages and line flows under uncertain loads, generations or network parameters. Currently, this problem is routinely solved in a number of decision-making procedures. Examples include transmission expansion and planning under long-term uncertainties in renewables penetration and regulation policies [15], [16]. In operations, the operators assess the security of the system and calculate Available Transfer Capability using random scenario sampling [17] where the ability to average the steady state solution over a large number of random scenarios is essential for secure power operations.

Probabilistic power flow problems have been solved by Monte Carlo and many analytical methods (including multi-linearization [10], the comulant method [11], fuzzy load flow analysis [12], and so forth). Recently, point estimation has become a popular technique for probabilistic power flow analysis [4]–[8]. This method assumes the solution being a summation of some univariate functions, then it computes the moments using a set of one-dimensional quadrature points.

Stochastic spectral methods [18] have emerged as a promising technique for the uncertainty analysis of many engineering problems including power systems [13], [14], [19], [20]. They approximate the stochastic solution by a generalized polynomial-chaos expansion [21]. This representation can provide various statistical information (e.g., moments and probability density function); it can also accelerate many stochastic problems in power systems (e.g., stochastic unit commitment [9] and parameter inference [22]), whereas previous approaches generally cannot. However, stochastic spectral methods may require lots of basis functions and simulation samples for problems with many uncertainties. In the uncertainty quantification community, some techniques based on compressed sensing [23], [24], proper generalized decomposition [25], [26] and tensor-train decomposition [20], [27], [28] have been developed for high-dimensional problems.

This paper develops an alternative stochastic spectral method to solve probabilistic power flow problems with possibly high-dimensional random parameters. Our main contributions are summarized as the following: i) We use tensors [29] (i.e., high-dimensional data arrays) to represent the huge set of data samples required in stochastic simulation. With a tensor format, we propose a low-rank and sparse tensor recovery scheme to generate a high-dimensional and sparse approximation while using an extremely small subset of quadrature samples. ii) We present the detailed numerical implementation of the tensor recovery method. Our algorithm relies on alternating minimization and the alternating direction method of multipliers (ADMM) [30]. Although only locally optimal solutions are guaranteed, the developed solver performs well for many practical cases. We demonstrate the performance of the proposed technique with numerical simulations on 3 benchmarks in MATPOWER 5.1 [31].

II. Problem formulation

A. Probabilistic Power Flow Problem

A steady-state power system with uncertainties can be described with parameterized power flow equations:

$$\begin{aligned} P_i(\boldsymbol{\xi}) &= \sum_{k=1}^{n} V_i(\boldsymbol{\xi}) V_k(\boldsymbol{\xi}) \left(G_{ik} \cos\theta_{ik}(\boldsymbol{\xi}) + B_{ik} \sin\theta_{ik}(\boldsymbol{\xi})\right) \\ Q_i(\boldsymbol{\xi}) &= \sum_{k=1}^{n} V_i(\boldsymbol{\xi}) V_k(\boldsymbol{\xi}) \left(G_{ik} \sin\theta_{ik}(\boldsymbol{\xi}) - B_{ik} \cos\theta_{ik}(\boldsymbol{\xi})\right) \end{aligned} \quad (1)$$

Z. Zhang was with Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA. He is now with Argonne National Laboratory, Lemont, IL 60439. E-mail: z_zhang@mit.edu.

H. D. Nguyen, K. Turitsyn and L. Daniel are with MIT, Cambridge, MA 02139, USA. E-mails: {hunghtd, turitsyn, luca}@mit.edu.

where P_i, Q_i, V_i, θ_i are the active and reactive power, voltage magnitude and angle at load bus i, respectively; G_{ik} and B_{ik} are conductances and susceptances; $\theta_{ik} = \theta_i - \theta_k$ is the voltage angle difference between buses i and k.

We employ random parameters $\boldsymbol{\xi}=[\xi_1, \cdots, \xi_d] \in \mathbb{R}^d$ to describe the uncertainties of load power consumptions that further influence bus voltages and angles. After computing V_i's and θ_i's, an out of interest y (e.g., the line flows) can be easily extracted. Obviously y also depends on $\boldsymbol{\xi}$ and thus can be written as $y = g(\boldsymbol{\xi})$. We assume that a deterministic solver is available to solve (1) given a sample of $\boldsymbol{\xi}$. For simplicity, we assume that all elements of $\boldsymbol{\xi}$ are mutually independent, then their joint probability density function is $\rho(\boldsymbol{\xi}) = \prod_{k=1}^{d} \rho_k(\xi_k)$, where $\rho_k(\xi_k)$ is the marginal probability density function of ξ_k. Moreover, the slack bus is assigned to compensate for the variations of loads and losses.

B. Stochastic Collocation Method

If the power flow problem is solvable, and $y=g(\boldsymbol{\xi})$ smoothly depends on $\boldsymbol{\xi}$, then we can approximate y by a truncated generalized polynomial-chaos expansion [21]

$$y = g(\boldsymbol{\xi}) \approx \sum_{|\boldsymbol{\alpha}|\le p} c_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}), \text{ with } \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) = \prod_{k=1}^{d} \phi_{k,\alpha_k}(\xi_k). \quad (2)$$

The multivariate polynomial basis $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$ is indexed by $\boldsymbol{\alpha}=[\alpha_1, \cdots, \alpha_d] \in \mathbb{N}^d$, with the total polynomial degree $|\boldsymbol{\alpha}| = \sum_{k=1}^{d} |\alpha_k| \le p$. The total number of basis functions is

$$K = \frac{(p+d)!}{p!d!}. \quad (3)$$

As shown in Appendix A, the degree-α_k univariate polynomials $\{\phi_{k,\alpha_k}(\xi_k)\}_{\alpha_k=0}^{p}$ are orthonormal to each other. Therefore, the multivariate basis functions are also orthonormal, and $c_{\boldsymbol{\alpha}}$ can be computed with projection

$$c_{\boldsymbol{\alpha}} = \int_{\mathbb{R}^d} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi}) g(\boldsymbol{\xi}) \rho(\boldsymbol{\xi}) d\boldsymbol{\xi}. \quad (4)$$

This integral can be evaluated by a proper quadrature rule which requires computing $g(\boldsymbol{\xi})$ at a set of samples.

C. Integration Rules and Curse of Dimensionality

Among different quadrature rules [32]–[34], this work considers computing $c_{\boldsymbol{\alpha}}$ by a tensor-rule Gauss quadrature method. First, use Gauss quadrature [35] (in Appendix B) to decide m quadrature samples and weights $\{\xi_k^{i_k}, w_k^{i_k}\}_{i_k=1}^{m}$ for ξ_k. Next, we compute $c_{\boldsymbol{\alpha}}$ by a tensor rule

$$c_{\boldsymbol{\alpha}} \approx \sum_{i_1=1}^{m} \cdots \sum_{i_d=1}^{m} g(\xi_1^{i_1}, \cdots, \xi_d^{i_d}) \prod_{k=1}^{d} \phi_{k,\alpha_k}(\xi_k^{i_k}) w_k^{i_k}. \quad (5)$$

This method requires simulating the power flow equation m^d times, and obviously it only works well for low-dimensional problems (e.g., when d is below 5 or 6). Sparse grid has been applied to simulate power systems [13], which can compute (4) with about $2^p K$ samples for high-dimensional cases [19]. In this paper, we aim to use only $< K$ samples from a tensor rule to compute (4).

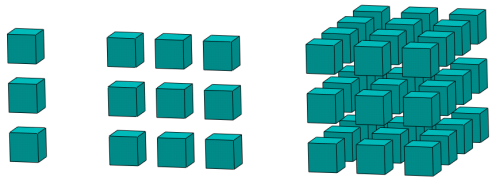

Fig. 1. Demonstration of vectors (left), matrices (middle) and tensors (right).

III. A Tensor-Recovery Approach

This section presents our tensor-recovery method to solve high-dimensional probabilistic power flow problems.

A. Tensor Representations of (5)

As a generalization of vectors and matrices, a tensor $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{m_1 \times \cdots \times m_d}$ represents a high-dimensional data array [29]. The number of dimensions, d, is called the mode of a tensor; m_k is the size of the k-th dimension. Given index $\mathbf{i} = (i_1, \cdots, i_d)$ (with integer $i_k \in [1, m_k]$), we can specify one element $\boldsymbol{\mathcal{A}}(\mathbf{i})$. Fig. 1 shows a 1-mode tensor (i.e., vector), a 2-mode tensor (i.e., matrix) and a 3-mode tensor.

First, we define a d-mode tensor $\boldsymbol{\mathcal{G}} \in \mathbb{R}^{m \times \cdots \times m}$

$$\boldsymbol{\mathcal{G}}(\mathbf{i}) = g(\xi_1^{i_1}, \cdots, \xi_d^{i_d}). \quad (6)$$

Next, for every ξ_k and its degree-α_k polynomial $\phi_{k,\alpha_k}(\xi_k)$, we define a vector $\mathbf{w}_{\alpha_k}^{(k)} \in \mathbb{R}^m$ with its i_k-th element being

$$\mathbf{w}_{\alpha_k}^{(k)}(i_k) = \phi_{k,\alpha_k}(\xi_k^{i_k}) w_k^{i_k}. \quad (7)$$

For every index vector $\boldsymbol{\alpha}$, we further construct a d-mode **rank-1 tensor** $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \in \mathbb{R}^{m \times \cdots \times m}$:

$$\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} = \mathbf{w}_{\alpha_1}^{(1)} \circ \cdots \circ \mathbf{w}_{\alpha_d}^{(d)} \Leftrightarrow \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}(\mathbf{i}) = \prod_{k=1}^{d} \mathbf{w}_{\alpha_k}^{(k)}(i_k). \quad (8)$$

Here $\circ$ denotes an outer product. As a result, the right-hand side of (5) is the **inner product** of $\boldsymbol{\mathcal{G}}$ and $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$:

$$\boxed{c_{\boldsymbol{\alpha}} \approx \langle \boldsymbol{\mathcal{G}}, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \rangle = \sum_{i_1=1}^{m} \cdots \sum_{i_d=1}^{m} \boldsymbol{\mathcal{G}}(\mathbf{i}) \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}(\mathbf{i}).} \quad (9)$$

In summary, in order to obtain the generalized polynomial-chaos approximation (2) we need to compute: 1) tensor $\boldsymbol{\mathcal{G}}$; 2) tensor $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ for each $\boldsymbol{\alpha}$ satisfying $|\boldsymbol{\alpha}| \le p$. Since each $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$ is the outer product of d vectors and many of them can be reused, computing $\boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}}$'s is trivial. However, directly computing $\boldsymbol{\mathcal{G}}$ is almost impossible, since the power flow equation must be simulated m^d times.

B. Low-Rank and Sparse Tensor-Recovery

Instead of computing $\boldsymbol{\mathcal{G}}$ directly, we *approximate* $\boldsymbol{\mathcal{G}}$ by tensor recovery. The key idea is described below.

1) Sub-Sampling: We randomly compute a small portion of elements in $\boldsymbol{\mathcal{G}}$, then seek for a tensor $\hat{\boldsymbol{\mathcal{G}}}$ to approximate $\boldsymbol{\mathcal{G}}$. Let $\mathcal{I} = \{\mathbf{i} | 1 \le i_k \le m\}$ include the indices for all elements in $\boldsymbol{\mathcal{G}}$. The size of $\mathcal{I}$, $|\mathcal{I}|$, is m^d. We choose a subset $\Omega \subset \mathcal{I}$ (with $|\Omega| \ll |\mathcal{I}|$) that includes a small number of indices randomly selected from $\mathcal{I}$, and compute $\boldsymbol{\mathcal{G}}(\mathbf{i}) = g(\xi_1^{i_1}, \cdots, \xi_d^{i_d})$ for any $\mathbf{i} \in \Omega$. Then, we look for a tensor $\hat{\boldsymbol{\mathcal{G}}}$ such that it matches $\boldsymbol{\mathcal{G}}$ at all elements specified by Ω, i.e.,

$$\| \mathbb{P}_\Omega \left(\hat{\boldsymbol{\mathcal{G}}} - \boldsymbol{\mathcal{G}} \right) \|_F^2 = 0. \tag{10}$$

Here $\mathbb{P}_\Omega$ is a linear operator for tensors:

$$\boldsymbol{\mathcal{B}} = \mathbb{P}_\Omega (\boldsymbol{\mathcal{A}}) \;\Leftrightarrow\; \boldsymbol{\mathcal{B}}(\mathbf{i}) = \begin{cases} \boldsymbol{\mathcal{A}}(\mathbf{i}), \text{ if } \mathbf{i} \in \Omega \\ 0, \text{ otherwise.} \end{cases} \tag{11}$$

The Frobenius-norm of a general tensor is defined as

$$\| \boldsymbol{\mathcal{A}} \|_F = \sqrt{\langle \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{A}} \rangle}. \tag{12}$$

An infinite number of tensors exist that satisfies the requirement (10) but significantly differs from $\boldsymbol{\mathcal{G}}$. Therefore, some constraints can be added to regularize this problem.

2) Constraint 1*– Sparsity:* Let vector $\mathbf{c} = [\cdots, c_{\boldsymbol{\alpha}}, \cdots] \in \mathbb{R}^K$ includes all coefficients in the generalized polynomial-chaos approximation. In high-dimensional cases, $\mathbf{c}$ is generally very sparse – most of its elements are close to zero. Using l_1-norm as a measure of sparsity [36], we have

$$|\mathbf{c}| = \sum_{\boldsymbol{\alpha} \le p} |c_{\boldsymbol{\alpha}}| \approx \sum_{\boldsymbol{\alpha} \le p} \left| \left\langle \hat{\boldsymbol{\mathcal{G}}}, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \right\rangle \right| \text{ is small} \tag{13}$$

3) Constraint 2*–Low Tensor Rank:* In many cases, $\boldsymbol{\mathcal{G}}$ has a low tensor rank and can be well approximated by the summation of a few rank-1 tensors. Therefore, we assume that the solution $\hat{\boldsymbol{\mathcal{G}}}$ has a **rank-r decomposition**:

$$\hat{\boldsymbol{\mathcal{G}}} = \mathbb{T} \left(\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)} \right) := \sum_{j=1}^{r} \mathbf{u}_j^{(1)} \circ \cdots \circ \mathbf{u}_j^{(d)} \tag{14}$$

where $\mathbf{u}_j^{(k)} \in \mathbb{R}^{m \times 1}$ is the j-th column of matrix $\mathbf{U}^{(k)} \in \mathbb{R}^{m \times r}$. Therefore, we may use d matrices to represent the whole tensor instead of computing and storing all elements of $\hat{\boldsymbol{\mathcal{G}}}$.

4) Final Tensor-Recovery Model: We describe the low-rank and sparse tensor-recovery model as follows:

Given $\boldsymbol{\mathcal{G}}(\mathbf{i})$ for every $\mathbf{i} \in \Omega$, solve

$$\begin{aligned} \min_{\left\{ \mathbf{U}^{(k)} \in \mathbb{R}^{m \times r} \right\}_{k=1}^{d}} \quad & f \left(\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)} \right) \\ = \tfrac{1}{2} & \left\| \mathbb{P}_\Omega \left(\mathbb{T}(\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)}) - \boldsymbol{\mathcal{G}} \right) \right\|_F^2 \\ + \lambda & \sum_{|\boldsymbol{\alpha}| \le p} \left| \langle \mathbb{T}(\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)}), \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \rangle \right|. \end{aligned} \tag{15}$$

Here $\lambda > 0$ is a regularization parameter.

C. Summary of Main Steps

We summarize the main steps of our approach as below.

1) Simulation Step: Randomly generate a small subset $\Omega \subset \mathcal{I}$ such that $|\Omega| < K \ll m^d$. For every index $\mathbf{i} = [i_1, \cdots, i_d] \in \Omega$, simulate the power flow equation (1) once to obtain a deterministic value $\boldsymbol{\mathcal{G}}(\mathbf{i}) = g(\xi_1^{i_1}, \cdots, \xi_d^{i_d})$.

Algorithm 1 Alternating Minimization for Solving (15).

1: Initialize: $\mathbf{U}^{(k),0} \in \mathbb{R}^{m \times r}$ for $k = 1, \cdots d$;
2: **for** $l = 0, 1, \cdots$
3: **for** $k = 1, \cdots, d$ **do**
4: solve (17) by Alg. 2 to obtain $\mathbf{U}^{(k),l+1}$;
5: **end for**
6: **break** if converged;
7: **end for**
8: **return** $\mathbf{U}^{(k)} = \mathbf{U}^{(k),l+1}$ for $k = 1, \cdots, d$.

2) Optimization Step: Solve (15) to obtain matrices $\mathbf{U}^{(1)}, \cdots, \mathbf{U}^{(d)}$ that represent tensor $\hat{\boldsymbol{\mathcal{G}}}$ in (14).

3) Model Generation: Replace $\boldsymbol{\mathcal{G}}$ by $\hat{\boldsymbol{\mathcal{G}}}$, and calculate $c_{\boldsymbol{\alpha}}$'s according to (9). With low-rank tensor factors, the computation can be simplified to

$$c_{\boldsymbol{\alpha}} \approx \left\langle \hat{\boldsymbol{\mathcal{G}}}, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \right\rangle = \sum_{j=1}^{r} \left[\prod_{k=1}^{d} \left(\left(\mathbf{u}_j^{(k)} \right)^T \mathbf{w}_{\alpha_k}^{(k)} \right) \right], \tag{16}$$

which involves only cheap vector inner products.

Since we can approximate $\boldsymbol{\mathcal{G}}$ by using only a small number of simulation samples, our method can be applied to many high-dimensional problems.

IV. Optimization Solver

This section describes how to solve (15).

A. Outer Loop: Alternating Minimization

1) Algorithm Flow: Starting from an initial guess $\{\mathbf{U}^{(k),0}\}_{k=1}^d$, we perform the following iterations: at iteration $l+1$ we use $\{\mathbf{U}^{(k),l}\}_{k=1}^d$ as an initial guess and obtain updated tensor factors $\{\mathbf{U}^{(k),l+1}\}_{k=1}^d$ by alternating minimization. Each iteration consists of d steps, and at the k-th step, $\mathbf{U}^{(k),l+1}$ is obtained by solving

$$\boxed{\mathbf{U}^{(k),l+1} = \arg\min_{\mathbf{X}} f \left(\cdots, \mathbf{U}^{(k-1),l+1}, \mathbf{X}, \mathbf{U}^{(k+1),l}, \cdots \right).} \tag{17}$$

Since all factors expect $\mathbf{U}^{(k)}$ are fixed, (17) becomes a convex optimization problem, and its global minimum can be computed by the solver in Section IV-B. The alternating minimization method ensures that the cost function decreases monotonically to a local minimal. The pseudo codes are summarized in Alg. 1, which terminates when the convergence criteria in Appendix C is satisfied.

2) Prediction Error: An interesting question is: how accurate is $\hat{\boldsymbol{\mathcal{G}}}$ compared with the exact tensor $\boldsymbol{\mathcal{G}}$? Our tensor recovery formulation enforces consistency between $\hat{\boldsymbol{\mathcal{G}}}$ and $\boldsymbol{\mathcal{G}}$ at the indices specified by Ω. We hope that $\hat{\boldsymbol{\mathcal{G}}}$ also has a good predictive behavior – $\hat{\boldsymbol{\mathcal{G}}}(\mathbf{i})$ is also close to $\boldsymbol{\mathcal{G}}(\mathbf{i})$ for $\mathbf{i} \notin \Omega$. In order to measure the predictive property of our results, we define a heuristic prediction error

$$\epsilon_{\text{pr}} = \sqrt{\frac{\sum_{\mathbf{i} \in \Omega'} \left(\hat{\boldsymbol{\mathcal{G}}}(\mathbf{i}) - \boldsymbol{\mathcal{G}}(\mathbf{i}) \right)^2 w_{\mathbf{i}}}{\sum_{\mathbf{i} \in \Omega'} \left(\boldsymbol{\mathcal{G}}(\mathbf{i}) \right)^2 w_{\mathbf{i}}}}, \text{ with } w_{\mathbf{i}} = \prod_{k=1}^{d} w_k^{i_k}. \tag{18}$$

Algorithm 2 ADMM for Solving (17).

1: Initialize: form $\mathbf{A}, \mathbf{F}$ and $\mathbf{b}$ according to Appendix D, specify initial guess $\mathbf{x}^0$, $\mathbf{u}^0$ and $\mathbf{z}^0$;
2: **for** $j = 0, 1, \cdots$ **do**
3: compute $\mathbf{x}^{j+1}$, $\mathbf{z}^{j+1}$ and $\mathbf{u}^{j+1}$ according to (20);
4: **break** if $\|\mathbf{F}\mathbf{x}^{j+1} - \mathbf{z}^{j+1}\| < \epsilon_1$ & $\|\mathbf{F}^T(\mathbf{z}^{j+1} - \mathbf{z}^j)\| < \epsilon_2$;
5: **end for**
6: **return** $\mathbf{U}^{(k),l+1} = \text{reshape}(\mathbf{x}^{j+1}, [m, r])$.

Here $\Omega' \in \mathcal{I}$ is a small-size index set such that $\Omega' \cap \Omega = \emptyset$. Obviously, $\hat{\boldsymbol{\mathcal{G}}}$ has good predictive behavior if ϵ_{pr} is small. Estimating ϵ_{pr} requires simulating the power flow equation at some extra quadrature samples. However, a small-size Ω' can provide a good heuristic estimation.

B. Inner Loop: Numerical Solver for (17)

Following the procedures in Appendix D, we rewrite Problem (17) as the generalized LASSO problem:

$$\boxed{\boldsymbol{vec}\left(\mathbf{U}^{(k),l+1}\right) = \arg\min_{\mathbf{x}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda|\mathbf{F}\mathbf{x}|} \quad (19)$$

where $\mathbf{A} \in \mathbb{R}^{|\Omega| \times mr}$, $\mathbf{F} \in \mathbb{R}^{K \times mr}$ and $\mathbf{b} \in \mathbb{R}^{|\Omega| \times 1}$, and $\mathbf{x} = \boldsymbol{vec}(\mathbf{X}) \in \mathbb{R}^{mr \times 1}$ is the vectorization of $\mathbf{X}$ (i.e., $\mathbf{x}(jm - m + i) = \mathbf{X}(i, j)$ for any integer $1 \le i \le m$ and $1 \le j \le r$). Note that $|\Omega|$ is the number of simulations samples in tensor recovery, and K is the total number of basis functions.

We solve (19) by the alternating direction method of multipliers (ADMM) [30]. Problem (19) can be rewritten as

$$\min_{\mathbf{x},\mathbf{z}} \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 + \lambda|\mathbf{z}| \quad \text{s.t. } \mathbf{F}\mathbf{x} - \mathbf{z} = \mathbf{0}.$$

By introducing an auxiliary variable $\mathbf{u}$ and starting with initial guesses $\mathbf{x}^0$, $\mathbf{u}^0 = \mathbf{z}^0 = \mathbf{F}\mathbf{x}^0$, the following iterations are performed to update $\mathbf{x}$ and $\mathbf{z}$:

$$\begin{aligned}
\mathbf{x}^{k+1} &= \left(\mathbf{A}^T\mathbf{A} + s\mathbf{F}^T\mathbf{F}\right)^{-1}\left(\mathbf{A}^T\mathbf{b} + s\mathbf{F}^T(\mathbf{z}^k - \mathbf{u}^k)\right) \\
\mathbf{z}^{k+1} &= \text{shrink}_{\lambda/s}(\mathbf{F}\mathbf{x}^{k+1} + \mathbf{z}^k + \mathbf{u}^k) \\
\mathbf{u}^{k+1} &= \mathbf{u}^k + \mathbf{F}\mathbf{x}^{k+1} - \mathbf{z}^{k+1}.
\end{aligned} \quad (20)$$

Here $s > 0$ is an augmented lagrangian parameter, and the soft thresholding operator is defined as

$$\text{shrink}_{\lambda/s}(a) = \begin{cases} a - \lambda/s, & \text{if } a > \lambda/s \\ 0, & \text{if } |a| < \lambda/s \\ a + \lambda/s, & \text{if } a < -\lambda/s. \end{cases}$$

The pseudo codes for solving (17) are given in Alg. 2.

C. Limitations

Firstly, the cost function of (15) is non-convex, and it is non-trivial to compute its global minimum with theoretical guarantees. Although researchers and engineers are very often satisfied with a local minimal, the obtained result may not be good enough for some cases. Secondly, in this work the parameters λ [the regularization parameter in (15)] and r [the tensor rank in (14)] are set based on some heuristic experiences. This treatment is definitely not optimal and does not guarantee high accuracy for all cases.

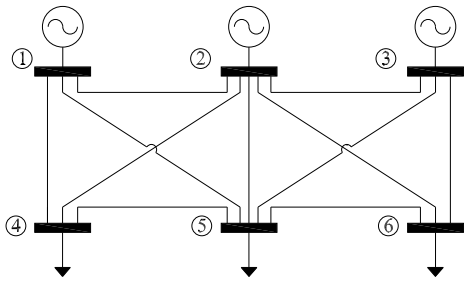

Fig. 2. The 6-bus system.

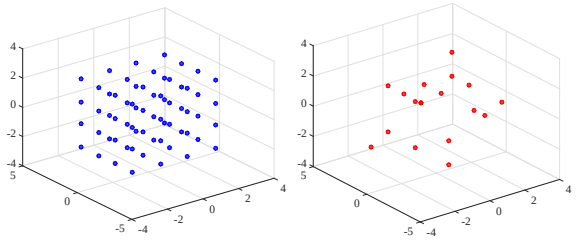

Fig. 3. Left: all quadrature samples; right: samples used in tensor recovery.

V. Simulations

This section reports the simulation results for several test cases from MATPOWER 5.1 [31]. All codes are implemented in MATLAB. We find that a 2nd- or 3rd-order generalized polynomial-chaos expansion can provide good accuracy for many cases, therefore we set $p = 2$ (or 3) in (2) and $m = 3$ (or 4) in Equation (9) .

A. 6-Bus Case (with 3 Random Parameters)

The **case6ww** example in MATPOWER 5.1 (c.f. Fig. 2) is used as a demonstrative example. We use 3 random parameters to describe the uncertain active powers at the load buses 4 to 6. We aim to obtain a 3rd-order generalized polynomial-chaos expansion for the real power injected from Bus 2 to Bus 4, leading to 20 basis functions in total. Applying a 4-point Gauss-quadrature rule to perform numerical integration for each dimension, we generate 64 quadrature points in total.

In order to compute the generalized polynomial-chaos expansion, only 18 quadrature points (as shown in Fig. 3) are randomly sub-selected. The simulation results at these selected samples are used to perform tensor recovery. For this case, we find that setting the tensor rank $r = 3$ and the regularization parameter $\lambda = 0.25$ is a good choice. Starting from a randomly generated rank-3 tensor, our algorithm converged after 25 iterations as shown in Fig. 4. The obtained low-rank tensor approximation has an estimated prediction error of 0.2%. With the obtained tensor approximation, the coefficients for all generalized polynomial-chaos basis functions are easily calculated based on (16). The coefficients for $\boldsymbol{\alpha} = 0$ is 31.83, which is the mean value of the output. All other coefficients are plotted in Fig. 5, where a sparsity pattern is observed.

Next we validate our results by Monte Carlo. Here we use 5000 samples in Monte Carlo simulation and treat its result as a golden reference solution. As shown in Table I, the

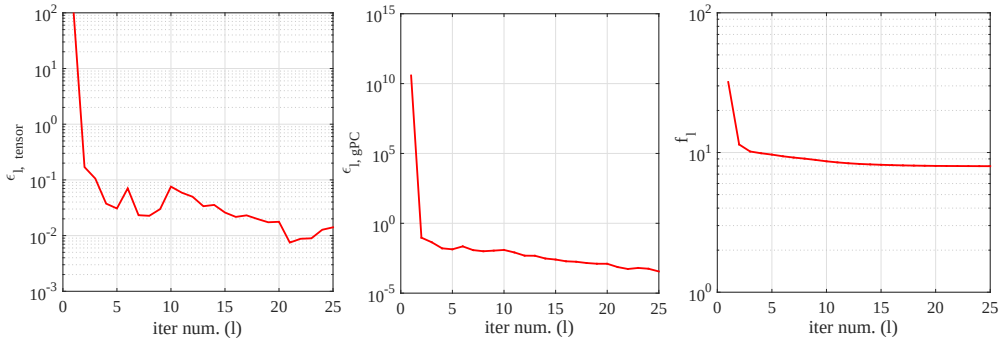

Fig. 4. Convergence for the 6-bus example. The left figure shows the relative update of tensor factors (i.e., $\epsilon_{l,\text{tensor}}$); the middle figure shows the update of the generalized-polynomial-chaos coefficients (i.e., $\epsilon_{l,\text{gPC}}$); the right figure shows the value of the objective function (i.e., f_l).

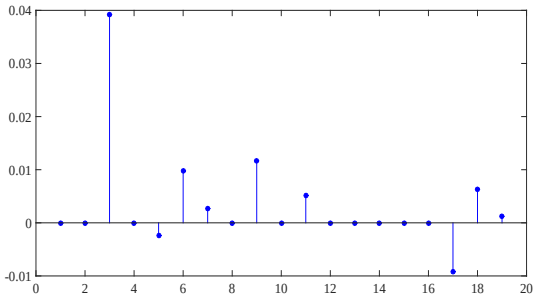

Fig. 5. Coefficients of $\{\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\}_{|\boldsymbol{\alpha}|=1}^{3}$ for the 6-bus example.

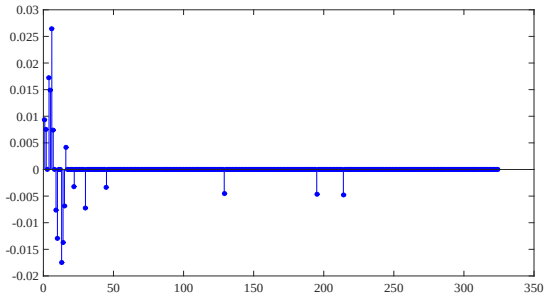

Fig. 6. Coefficients of $\{\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\}_{|\boldsymbol{\alpha}|=1}^{2}$ for the 30-bus example.

TABLE I
MOMENTS OF THE 6-BUS EXAMPLE.

	Tensor Recovery	Monte Carlo
samples	18	5000
Mean	31.83	31.87
stand. dev.	0.0439	0.0448

mean value and standard deviation from our tensor recovery approach is very close to that from Monte Carlo.

Complexity Reduction. Since we use 18 samples out of 64 quadrature points, the reduction ratio for this problem is 3.6. Note that the number of samples in tensor recovery is less than the number of basis functions (i.e., 20).

B. 30-Bus Case (with 24 Random Parameters)

Next we consider the **case30** example in MATPOWER 5.1, with the active powers of 24 load buses modeled by Gaussian random variables. We apply a 2nd-order generalized polynomial-chaos expansion for the real power from bus 15 to bus 23, requiring totally 325 basis functions. For each parameter, 3 quadrature points are used, leading to $3^{24} \approx 2.8 \times 10^{11}$ samples in total. Obviously, it is prohibitively expensive to simulate the power system at all quadrature points.

In our tensor recovery scheme, we randomly pick 280 quadrature points from the full tensor-rule quadrature samples and approximate $\boldsymbol{\mathcal{G}}$ by a rank-4 tensor. Setting $\lambda = 0.3$ and starting with a random initial guess, our algorithm converges nicely' after 26 iterations which are similar to Fig. 4. With 50 newly sub-sampled quadrature points as the testing samples,

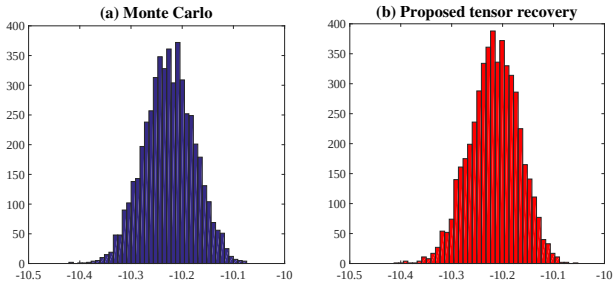

Fig. 7. Histograms of the simulated output for the 30-bus example .

the estimated prediction error is 0.55%. Although this example has many random parameters, its generalized polynomial-chaos expansion is very sparse, as shown in Fig. 6.

In order to check the accuracy, we perform Monte Carlo simulation using 5000 random samples. Table II compares the mean values and standard deviations from both approaches, and they are very close. An advantage of generalized polynomial-chaos expansion is that one can easily evaluate the expression with many samples to get a density function or histogram. Such information cannot be easily obtained by a point-estimation method. The histogram from our method is close to that from Monte Carlo (c.f. Fig. 7).

Complexity Reduction. Since we use 280 samples out of 3^{24} quadrature points, the reduction ratio for this example is 10^9. The number of samples in tensor recovery is also smaller than the number of basis functions (i.e., 325).

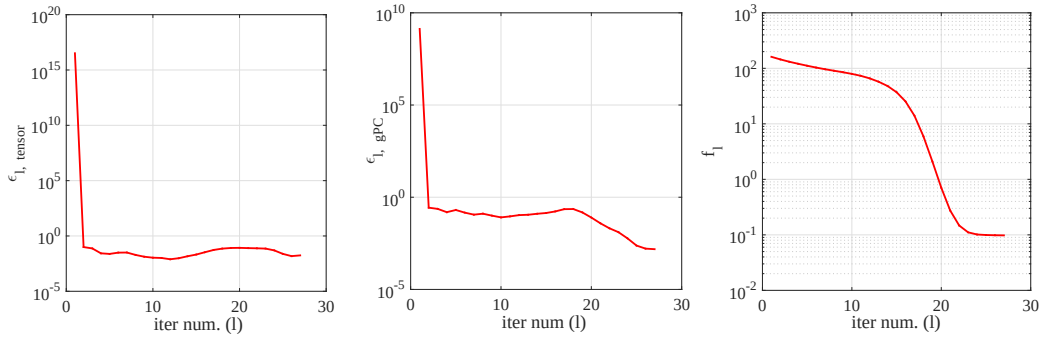

Fig. 8. Convergence for the 57-bus example. The left figure shows the relative update of tensor factors (i.e., $\epsilon_{l,\text{tensor}}$); the middle figure shows the update of the generalized-polynomial-chaos coefficients (i.e., $\epsilon_{l,\text{gPC}}$); the right figure shows the value of the objective function (i.e., f_l).

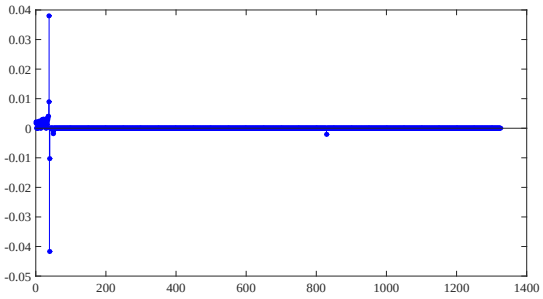

Fig. 9. Coefficients of $\{\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})\}^2_{|\boldsymbol{\alpha}|=1}$ for the 57-bus example.

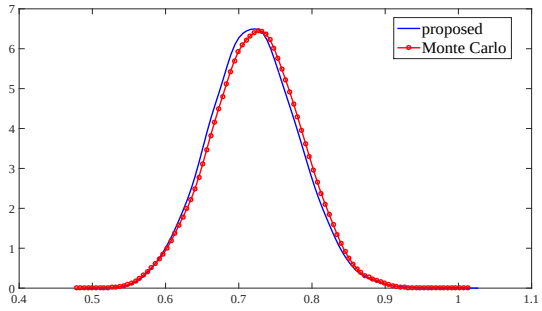

Fig. 10. Computed probability density functions for the 57-bus example.

TABLE II
MOMENTS OF THE 30-BUS EXAMPLE.

	Tensor Recovery	Monte Carlo
samples	280	5000
Mean	-10.23	-10.22
stand. dev.	0.048	0.049

TABLE III
MOMENTS OF THE 57-BUS EXAMPLE.

	Tensor Recovery	Monte Carlo
samples	800	5000
Mean	0.721	0.724
stand. dev.	0.0596	0.0609

C. 57-Buse Case (with 50 Random Parameters)

Finally we consider the **case57** example in MATPOWER 5.1, with 50 Gaussian random variables describing the active powers at load buses. With a 2nd-order polynomial-chaos expansion, we aim to approximate the real power injected from Bus 19 to Bus 20 with 1326 basis functions. Using 3 Gauss-quadrature points for each parameter, a tensor-rule quadrature method requires $3^{50} > 7 \times 10^{23}$ samples in total. It is impossible to store the samples on a personal computer, let alone simulating the power flow equation at all samples.

Our tensor recovery scheme randomly sub-selects 800 samples to perform power flow simulations. Starting with a random initial guess, we approximate the full tensor $\mathcal{G}$ by a rank-5 tensor, with an estimated prediction error of 1%. Fig. 8 shows the convergence of our solver. Fig. 9 plots the coefficients for all non-constant basis functions. Clearly, the result is extremely sparse for this high-dimensional example.

In order to get a full picture about the statistical behavior of the output, we evaluate the computed generalized polynomial-chaos expansion with 5000 random samples and plot its probability density function. As shown in Fig. 10, the result is close to that from Monte Carlo simulation on the original power flow equations. The mean values and standard deviations from both approaches are very close (c.f. Table III).

Complexity Reduction. Since we use 800 samples out of 3^{50} quadrature points, the reduction ratio for this example is about 9×10^{20}. The number of samples in tensor recovery is again smaller than the number of basis functions (i.e., 1326).

VI. CONCLUSIONS AND FUTURE WORK

This paper has presented a probabilistic power flow simulation algorithm based on tensors and stochastic collocation. In order to break the curse of dimensionality, we have developed a high-dimensional method that exploits the low-rank and sparse property of tensors. This tensor framework has completed the huge-data-size simulation task with an extremely small-size simulation data set. We have further developed a numerical solver for the tensor recovery problem and tested it on three power flow benchmarks. This algorithm has successfully generated high-dimensional and sparse generalized polynomial-chaos expansions for the solutions. Good accuracy (measured by prediction errors and comparison against Monte Carlo) as well as significant computational cost reduction (with up to 9×10^{20} times) have been observed in this work.

To the best of our knowledge, this is the first work studying stochastic power systems from a tensor perspective. While several limitations exist in the current work, the authors believe that many problems are worth investigation in this direction. Firstly, it is worth investing global non-convex optimization to solve (15) or trying to convexify the formulation. Secondly, some future work about the optimal choice of λ and r may help improve the robustness of our framework. Finally, it is worth developing better sampling schemes to improve the performance of tensor recovery.

One particularly attractive application of this technique is the construction of probabilistic static equivalents of a sub-networks. These equivalents can be used both for distribution grid models with high penetration of intermittent renewables and for probabilistic modeling of random disturbances in neighbor areas in multi-area power systems.

A second natural application is the stochastic contingency analysis over a range of operating conditions. Currently deterministic contingency analysis is considered in some basic cases with given operating condition. However, as the systems change and are subject to uncertainties, one need to take multiple cases or scenarios into account. The fast averaging technique developed in this paper can effectively alleviate the heavy computation in such situations.

Appendix A
Orthonormal Polynomials

Consider a single random parameter $\xi_k \in \mathbb{R}$ with a probability density function $\rho_k(\xi_k)$, one can construct a set of polynomial functions subject to the orthonormal condition:

$$\int_{\mathbb{R}} \phi_{k,\alpha}(\xi_k)\phi_{k,\beta}(\xi_k)\rho_k(\xi_k)d\xi_k = \delta_{\alpha,\beta}$$

where $\delta_{\alpha,\beta}$ is a Delta function, integer α is the highest degree of $\phi_{k,\alpha}(\xi_k)$. Such polynomials can be constructed as follows [37]. First, one constructs orthogonal polynomials $\{\pi_{k,\alpha}(\xi_k)\}_{\alpha=0}^{p}$ with an leading coefficient 1 recursively

$$\pi_{k,\alpha+1}(\xi_k) = (\xi_k - \gamma_\alpha)\,\pi_{k,\alpha}(\xi_k) - \kappa_\alpha \pi_{k,\alpha-1}(\xi_k)$$

for $\alpha = 0, 1, \cdots p-1$, with initial conditions $\pi_{k,-1}(\xi_k) = 0$, $\pi_{k,0}(\xi_k) = 1$ and $\kappa_0 = 1$. For $\alpha \geq 0$, the recurrence parameters are defined as

$$\gamma_\alpha = \frac{\mathbb{E}\left(\xi_k \pi_{k,\alpha}^2(\xi_k)\right)}{\mathbb{E}\left(\pi_{k,\alpha}^2(\xi_k)\right)},\ \kappa_{\alpha+1} = \frac{\mathbb{E}\left(\xi_k \pi_{k,\alpha+1}^2(\xi_k)\right)}{\mathbb{E}\left(\xi_k \pi_{k,\alpha}^2(\xi_k)\right)}. \quad (21)$$

Here $\mathbb{E}$ denotes the operator that calculates expectation. Second, one can obtain $\{\phi_{k,\alpha}(\xi_k)\}_{\alpha=0}^{p}$ by normalization:

$$\phi_{k,\alpha}(\xi_k) = \frac{\pi_{k,\alpha}(\xi_k)}{\sqrt{\kappa_0\kappa_1\cdots\kappa_\alpha}},\ \text{for } \alpha = 0, 1, \cdots, p.$$

Appendix B
Gauss Quadrature Rule [35]

Given $\xi_k \in \mathbb{R}$ with a density function $\rho_k(\xi_k)$ and a smooth function $q(\xi_k)$, Gauss quadrature evaluates the integral

$$\int_{\mathbb{R}} q(\xi_k)\rho_k(\xi_k)d\xi_k \approx \sum_{i_k=1}^{m} q(\xi_k^{i_k}) w_k^{i_k}$$

with an error decreasing exponentially as m increases. An exact result is obtained if $q(\xi_k)$ is a polynomial function of degree $\leq 2m-1$. One can obtain $\{(\xi_k^{i_k}, w_k^{i_k})\}_{i_k=1}^{p+1}$ by reusing the recurrence parameters in (21) to form a symmetric tridiagonal matrix $\mathbf{J} \in \mathbb{R}^{(p+1)\times(p+1)}$:

$$\mathbf{J}(i,j) = \begin{cases} \gamma_{i-1}, \text{ if } i = j \\ \sqrt{\kappa_i}, \text{ if } i = j+1 \\ \sqrt{\kappa_j}, \text{ if } i = j-1 \\ 0, \text{ otherwise} \end{cases} \quad \text{for } 1 \leq i, j \leq p+1.$$

Let $\mathbf{J} = \mathbf{Q}\Sigma\mathbf{Q}^T$ be an eigenvalue decomposition and $\mathbf{Q}$ a unitary matrix, then $\xi_k^{i_k} = \Sigma(i_k, i_k)$ and $w_k^{i_k} = (\mathbf{Q}(1, i_k))^2$.

Appendix C
Error Control in Alg. 1

With tensor factors $\{\mathbf{U}^{(1),k}\}_{k=1}^{d}$ obtained after l iterations of the outer loops of Alg. 1, we define

$$f_l := f\left(\mathbf{U}^{(1),l}, \cdots, \mathbf{U}^{(d),l}\right) \quad \text{[updated cost func. of (15)]}$$
$$\hat{\boldsymbol{\mathcal{G}}}_l := \mathbb{T}\left(\mathbf{U}^{(1),l}, \cdots, \mathbf{U}^{(d),l}\right) \quad \text{(approximated tensor)}$$
$$c_{\boldsymbol{\alpha}}^l := \left\langle \hat{\boldsymbol{\mathcal{G}}}_l, \boldsymbol{\mathcal{W}}_{\boldsymbol{\alpha}} \right\rangle \quad \text{[updated coefficient for } \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\alpha})\text{]}$$

and let $\mathbf{c}^l = [\cdots, c_{\boldsymbol{\alpha}}^l, \cdots] \in \mathbb{R}^K$ for all $|\boldsymbol{\alpha}| \leq p$. Then, we define the following quantities for error control:

- Relative update of the tensor factors:

$$\epsilon_{l,\text{tensor}} = \sqrt{\sum_{k=1}^{d} \|\mathbf{U}^{(k),l} - \mathbf{U}^{(k),l-1}\|_F^2 / \sum_{k=1}^{d} \|\mathbf{U}^{(k),l-1}\|_F^2}.$$

- Relative update of $\mathbf{c} = [\cdots, c_{\boldsymbol{\alpha}}, \cdots]$

$$\epsilon_{l,\text{gPC}} = \|\mathbf{c}^l - \mathbf{c}^{l-1}\| / \|\mathbf{c}^{l-1}\|.$$

- Relative update of the cost function:

$$\epsilon_{l,\text{cost}} = |f_l - f_{l-1}| / |f_{l-1}|.$$

The solution $\{\mathbf{U}^{(k),l}\}_{k=1}^{d}$ is regarded as a local minimal if $\epsilon_{l,\text{tensor}}$, $\epsilon_{l,\text{gPC}}$ and $\epsilon_{l,\text{cost}}$ are small enough.

Appendix D
Assembling The Matrices and Vector in (19)

Consider the tensor factors $\mathbf{U}^{(1),l+1}$, $\cdots$, $\mathbf{U}^{(k-1),l+1}$, $\mathbf{X}$, $\mathbf{U}^{(k+1),l}$, $\cdots$, $\mathbf{U}^{(d),l}$ in (17). We denote the (i,j) element of $\mathbf{U}^{(k'),l}$ (or $\mathbf{X}$) by $u_{i,j}^{(k'),l}$ (or $x_{i,j}$), and its j-th column by $\mathbf{u}_j^{(k'),l}$ (or $\underline{\mathbf{x}}_j$). Then, the cost function in (17) is

$$\begin{aligned} &f\left(\cdots, \mathbf{U}^{(k-1),l+1}, \mathbf{X}, \mathbf{U}^{(k+1),l}, \cdots\right) \\ =&\frac{1}{2}\sum_{\mathbf{i}\in\Omega}\left(\sum_{j=1}^{r} x_{i_k,j}\mu_{\mathbf{i},j} - \boldsymbol{\mathcal{G}}(\mathbf{i})\right)^2 + \lambda \sum_{|\boldsymbol{\alpha}|\leq p}\left|\sum_{j=1}^{r} \nu_{\boldsymbol{\alpha},j}\langle \underline{\mathbf{x}}_j, \mathbf{w}_{\alpha_k}^{(k)}\rangle\right| \end{aligned}$$

where the scalars $\mu_{\mathbf{i},j}$ and $\nu_{\boldsymbol{\alpha},j}$ are computed as follows:

$$\mu_{\mathbf{i},j} = \prod_{k'=1}^{k-1} u_{i_{k'},j}^{(k'),l+1} \prod_{k'=k+1}^{d} u_{i_{k'},j}^{(k'),l},$$
$$\nu_{\boldsymbol{\alpha},j} = \prod_{k'=1}^{k-1} \langle \mathbf{u}_j^{(k'),l+1}, \mathbf{w}_{\alpha_{k'}}^{(k')}\rangle \prod_{k'=k+1}^{d} \langle \mathbf{u}_j^{(k'),l}, \mathbf{w}_{\alpha_{k'}}^{(k')}\rangle.$$

Since each row (or element) of $\mathbf{A}$ (or $\mathbf{b}$) corresponds to an index $\mathbf{i} \in \Omega$, and each row of $\mathbf{F}$ corresponds to a basis function $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$, in this appendix we use $\mathbf{i}$ as the row index (or element index) of $\mathbf{A}$ (or $\mathbf{b}$) and $\boldsymbol{\alpha}$ as the row index of $\mathbf{F}$. Now we specify the elements of $\mathbf{A}$, $\mathbf{b}$ and $\mathbf{F}$ of (19).

- For every $\mathbf{i} \in \Omega$, $\mathbf{b}(\mathbf{i}) = \boldsymbol{\mathcal{G}}(\mathbf{i})$.
- Since $x_{i_k,j}$ is the $(j-1)m + i_k$-th element of $\mathbf{x} = \boldsymbol{vec}(\mathbf{X})$, for every $\mathbf{i} \in \Omega$ we have

$$\mathbf{A}(\mathbf{i}, (j-1)m + i_k) = \begin{cases} \mu_{\mathbf{i},j}, & \text{for } j = 1, \cdots, r \\ 0, & \text{otherwise.} \end{cases}$$

- Since $\underline{\mathbf{x}}_j$ includes the elements of $\mathbf{x}$ ranging from index $(j-1)m+1$ to jm, given an index vector $\boldsymbol{\alpha}$ the corresponding row of $\mathbf{F}$ can be specified as

$$\mathbf{F}(\boldsymbol{\alpha}, jm - m + i_k) = \nu_{\boldsymbol{\alpha},j} \mathbf{w}_{\alpha_k}^{(k)}(i_k) = \nu_{\boldsymbol{\alpha},j} \phi_{k,\alpha_k}(\xi_k^{i_k}) w_k^{i_k}$$

for all integers $j \in [1, r]$ and $i_k \in [1, m]$.

REFERENCES

[1] A. J. Conejo, M. Carrión, and J. M. Morales, *Decision making under uncertainty in electricity markets*. Springer, 2010, vol. 1.

[2] B. Borkowska, "Probabilistic load flow," *IEEE Trans. Power Apparatus and Syst.*, vol. 93, no. 3, pp. 752–759, 1974.

[3] E. Haesen, J. Driesen, and R. Belmans, "Stochastic, computational and convergence aspects of distribution power flow algorithms," in *Proc. IEEE Power Tech.*, 2007, pp. 1447–1452.

[4] C. S. Saunders, "Point estimate method addressing correlated wind power for probabilistic optimal power flow," *IEEE Trans. Power Syst.*, vol. 29, no. 3, pp. 1045–1054, 2014.

[5] J. M. Morales and J. Perez-Ruiz, "Point estimate schemes to solve the probabilistic power flow," *IEEE Trans. Power Syst.*, vol. 22, no. 4, pp. 1594–1601, 2007.

[6] J. M. Morales, L. Baringo, A. J. Conejo, and R. Mínguez, "Probabilistic power flow with correlated wind sources," *IET Generation, Transmission & Distribution*, vol. 4, no. 5, pp. 641–651, 2010.

[7] G. Verbic and C. Canizares, "Probabilistic optimal power flow in electricity markets based on a two-point estimation method," *IEEE Trans. Power Syst.*, vol. 21, no. 4, pp. 1883–1893, 2006.

[8] C. Su, "Probabilistic load-flow computation using point estimate method," *IEEE Trans. Power Syst.*, vol. 20, no. 4, pp. 1843–1851, 2005.

[9] E. M. Constantinescu, V. M. Zavala, M. Rocklin, S. Lee, and M. Anitescu, "A computational framework for uncertainty quantification and stochastic optimization in unit commitment with wind power generation," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 431–441, Feb. 2011.

[10] R. Allan and A. Leite da Silva, "Probabilistic load flow using multilinearizations," *IET Gen. Transm. Distrib.*, vol. 128, no. 5, pp. 280–287, 1981.

[11] R. Allan and M. Al-Shakarchi, "Probabilistic techniques in AC load flow analysis," *Proc. IEE*, vol. 124, no. 2, pp. 154–160, Feb 1977.

[12] V. Miranda, M. Matos, and J. Saraiva, "Fuzzy load flow: new algorithms incorporating uncertain generation and load representation," in *Proc. Power Syst. Comp. Conf.*, 1990, pp. 621–627.

[13] G. Lin, M. Elizondo, S. Lu, and X. Wan, "Uncertainty quantification in dynamic simulations of large-scale power system models using the high-order probabilistic collocation method on sparse grids," *Int. J. Uncert. Quant.*, vol. 4, no. 3, pp. 185–204, Feb. 2014.

[14] J. Hockenberry and B. Lesieutre, "Evaluation of uncertainty in dynamic simulation of power system models: the probabilistic collocation method," *IEEE Trans. Power Syst.*, vol. 19, no. 3, pp. 242–272, 2004.

[15] M. Milligan, P. Donohoo, and M. OMalley, "Stochastic methods for planning and operating power system with large amounts of wind and solar power," in *Proc. Int. Workshop Large-Scale Integr. Wind Power into Power Syst.*, 2012.

[16] B. Gorenstin, N. Campodonico, J. da Costa, and M. Pereira, "Power system expansion planning under uncertainty," *Power Systems, IEEE Transactions on*, vol. 8, no. 1, pp. 129–136, Feb 1993.

[17] J. Stahlhut, F. Gao, K. Hedman, B. Westendorf, G. Heydt, P. Sauer, and G. Sheblé, "Uncertain power flows and transmission expansion planning," in *Power Symposium, 2005. Proceedings of the 37th Annual North American*. IEEE, 2005, pp. 489–496.

[18] D. Xiu, "Fast numerical methods for stochastic computations: A review," *Commun. Comput. Phys.*, vol. 5, no. 2-4, pp. 242–272, Feb. 2009.

[19] Z. Zhang, T. A. El-Moselhy, I. M. Elfadel, and L. Daniel, "Stochastic testing method for transistor-level uncertainty quantification based on generalized polynomial chaos," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst*, vol. 32, no. 10, pp. 1533–1545, Oct 2013.

[20] Z. Zhang, I. Osledets, X. Yang, G. E. Karniadakis, and L. Daniel, "Enabling high-dimensional hierarchical uncertainty quantification by ANOVA and tensor-train decomposition," *IEEE Trans. CAD Integr. Circuits Syst.*, vol. 34, no. 1, p. 63 76, Jan. 2015.

[21] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM J. Scientific Computing*, vol. 24, no. 2, pp. 619–644, Feb 2002.

[22] Z. Zhang, C. Petra, N. Petra, and E. Constantinescu, "Fast state estimation of power systems with polynomial chaos," Aug. 2015, technical Report, Argonne National Laboratory.

[23] X. Yang and G. E. Karniadakis, "Reweighted l_1 minimization method for sothcastic elliptic differential equations," *J. Comp. Phys.*, vol. 248, no. 1, pp. 87–108, Sept. 2013.

[24] J. Peng, J. Hampton, and A. Doostan, "A weighted l_1 minimization approach for sparse polynomial chaos expansion," *J. Comp. Phys.*, vol. 267, no. 1, pp. 92–111, Jun. 2014.

[25] A. Nouy, "Proper generalized decomposition and separated representations for the numerical solution of high dimensional stochastic problems," *Arch. Comp. Meth. Eng.*, vol. 27, no. 4, pp. 403–434, Dec 2010.

[26] F. Chinesta, P. Ladeveze, and E. Cueto, "A short review on model order reduction based on proper generalized decomposition," *Arch. Comput. Meth. Eng.*, vol. 18, no. 4, pp. 395–404, 2011.

[27] B. N. Khoromskij and C. Schwab, "Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs," *SIAM J. Sci. Comput*, vol. 33, no. 1, pp. 364–385, Oct 2011.

[28] D. Bigoni, A. P. Engsig-Karup, and Y. M. Marzouk, "Spectral tensor-train decomposition," *arXiv preprint, arXiv:1405.5713v1*, May 2014.

[29] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[30] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1 –122, 2010.

[31] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-state operations, planning and analysis tools for power system research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–16, Jun. 2011.

[32] T. Gerstner and M. Griebel, "Numerical integration using sparse grids," *Numerical Algorithms*, vol. 18, pp. 209–232, Mar. 1998.

[33] S. Weinzierl, "Introduction to Monte Carlo methods," NIKHEF, Theory Group, Amsterdam, The Netherlands, Tech. Rep., 2000.

[34] W. J. Morokoff and R. E. Caflisch, "Quasi-Monte Carlo integration," *J. Comput. Phys.*, vol. 122, no. 2, pp. 218–230, Dec 1995.

[35] G. H. Golub and J. H. Welsch, "Calculation of Gauss quadrature rules," *Math. Comp.*, vol. 23, pp. 221–230, 1969.

[36] D. L. Donoho, "Compressed sensing," *IEEE Trans. Informa. Theory*, vol. 52, no. 4, pp. 578 –594, April 2006.

[37] W. Gautschi, "On generating orthogonal polynomials," *SIAM J. Sci. Stat. Comput.*, vol. 3, no. 3, pp. 289–317, Sept. 1982.