

The network uncertainty quantification method for propagating uncertainties in component-based systems

Kevin Carlberg*, Sofia Guzzetti†, Mohammad Khalil*, and Khachik Sargsyan*

Abstract. This work introduces the network uncertainty quantification (NetUQ) method for performing uncertainty propagation in systems composed of interconnected components. The method assumes the existence of a collection of components, each of which is characterized by exogenous-input random variables (e.g., material properties), endogenous-input random variables (e.g., boundary conditions defined by another component), output random variables (e.g., quantities of interest), and a local uncertainty-propagation operator (e.g., provided by stochastic collocation) that computes output random variables from input random variables. The method assembles the full-system network by connecting components, which is achieved simply by associating endogenous-input random variables for each component with output random variables from other components; no other inter-component compatibility conditions are required. The network uncertainty-propagation problem is: Compute output random variables for all components given all exogenous-input random variables. To solve this problem, the method applies classical relaxation methods (i.e., Jacobi and Gauss–Seidel iteration with Anderson acceleration), which require only black-box evaluations of component uncertainty-propagation operators. Compared with other available methods, this approach is applicable to any network topology (e.g., no restriction to feed-forward or two-component networks), promotes component independence by enabling components to employ tailored uncertainty-propagation operators, supports general functional representations of random variables, and requires no offline preprocessing stage. Also, because the method propagates functional representations of random variables throughout the network (and not, e.g., probability density functions), the joint distribution of any set of random variables throughout the network can be estimated *a posteriori* in a straightforward manner. We perform supporting convergence and error analysis and execute numerical experiments that demonstrate the weak- and strong-scaling performance of the method.

Key words. uncertainty propagation, domain decomposition, relaxation methods, Anderson acceleration, network uncertainty quantification

AMS subject classifications. 35R60, 60H15, 60H35, 65C20, 49M20, 49M27, 65N55

1. Introduction. Many systems in science and engineering—ranging from power grids to gas transfer systems to aircraft—comprise a collection of a large number of interconnected components. Performing uncertainty propagation in these systems is often essential for quantifying the influence of uncertainties on the performance of the full system. Naïvely, performing full-system uncertainty propagation in this context requires (1) integrating all component models into a single deterministic full-system model, and (2) executing uncertainty propagation using the deterministic full-system model. The first step is often challenging or impossible, as components are often characterized by vastly different physical phenomena and spatiotemporal scales, ensuring compatibility between the meshes of neighboring components is difficult, and components are often modeled using different simulation codes that can be difficult to integrate. If the first step is achievable, then the second step may still pose an insurmountable challenge due to the large-scale nature of the deterministic full-system model.

To address these challenges, researchers have developed several classes of methods that aim to decompose the full-system uncertainty-propagation problem into tractable subproblems. Ideally, such a method should satisfy several desiderata. First, the method should *promote component independence*. For example, each component should be able to use a tailored uncertainty-propagation method, as it has been shown that different uncertainty-propagation methods are better suited for different physics (see, e.g., Ref. [14]); in addition, the approach should avoid any coupled-component

¹Sandia National Laboratories, 7011 East Avenue, Livermore, CA 94550. kevintcarlberg@gmail.com, mkhilal@sandia.gov, ksargsy@sandia.gov.

²Emory University, 201 Dowman Dr, Atlanta, GA 30322. sofia.guzzetti@emory.edu.

deterministic solves. Second, to be widely applicable, the method should *not be restricted to systems with particular component-network topologies*, as systems often comprise complex assemblies of components with two-way coupling between neighboring components. Third, the method should characterize uncertainties using *functional representations of random variables*, and should not be restricted to particular functional representations. Characterizing random variables using functional representations (i.e., as functions acting on the sample space) allows for the joint distribution of any set of random variables throughout the system to be estimated straightforwardly, and also facilitates other post-processing activities (e.g., variance-based decomposition). Supporting general representations is also important, as techniques restricted to polynomial-chaos representations [23, 44], for example, are not compatible with other widely used random-variable representations [42, 9, 10].

The first class of methods replaces component high-fidelity models with component surrogate models, and subsequently computes inexpensive full-system samples either by propagating deterministic samples within a feed-forward network [35, 39], or by using domain-decomposition methods to converge these full-system samples [32]. We point out that while these methods can significantly reduce the computational cost of full-system uncertainty propagation, they do not decompose the uncertainty-propagation task itself; thus, these methods do not enable each component to use a tailored uncertainty-propagation method. Relatedly, Ref. [22] generates a surrogate model for the coupling variables across the system to facilitate generating full-system samples; however, this approach requires generating deterministic full-system samples to construct the surrogate model.

The second class of methods considers coupled two-component systems and restricts attention to polynomial-chaos representations of random variables. These techniques focus primarily on reducing the dimensionality of the stochastic space considered by each component; Refs. [4, 3, 5] employ Gauss–Seidel iteration to update the random variables for each component (analogous to overlapping domain decomposition), while Refs. [15, 36] propose intrusive coupled formulations (analogous to non-overlapping domain decomposition). Alternatively, Ref. [12] considers linear feed-forward systems and exploits the fact that—for such systems—the uncertainty-propagation problems for different PCE coefficients decouple. We note that these techniques do not consider the case where input random variables may be shared across components, which may occur, for example if two components share a boundary condition with uncertainty.

The third class of methods models a given component-based system as a network of interconnected components, and propagates probability distributions through the resulting network. The first contribution [1] restricted attention to feed-forward networks, while follow-on work [24] considered extensions to coupled two-component networks. Critically, because a random variable’s marginal distribution does not completely characterize its functional behavior, these approaches encounter challenges for certain uncertainty-quantification tasks, e.g., computing the joint distribution between random variables, compute Sobol’ indices for variance-based decomposition. However, Ref. [1] provided a clever mechanism for recovering joint distributions in the case of feed-forward networks. A related approach [38] replaces coupling variables with conditional probability densities and subsequently generates full-system samples by propagating samples through the resulting feed-forward network; however, this approach does not account for joint distributions between system inputs and coupling variables.

The fourth class of methods stems from the field of multidisciplinary design optimization, and is typically carried out in a reliability-analysis context [46]. Such methods are adopted from related deterministic approaches via either matching moments [26, 19, 13] or reformulating reliability conditions as probabilistic constraints, i.e., enforcing interface-matching violation to occur with specified low probability [30]. In both cases, these approaches enable full-system uncertainty propagation by introducing first-order, local perturbation assumptions with no support for general functional representations, or by introducing strong decoupling assumptions that restrict the set of supported network topologies [33].

To satisfy the aforementioned desiderata and overcome some of the limitations of previous contributions, this work proposes the network uncertainty quantification (NetUQ) method, which is inspired by classical relaxation techniques commonly employed in overlapping domain decomposition. The method simply assumes the existence of a collection of components, each of which is characterized by (1) *exogenous-input random variables* (e.g., boundary conditions, material properties, geometric variables), (2) *endogenous-input random variables* (e.g., boundary conditions imposed by a neighboring component), (3) *output random variables* (e.g., quantities of interest), and (4) an *uncertainty-propagation operator* that computes output random variables from input random variables (e.g., non-intrusive spectral projection, stochastic Galerkin projection). In particular, different components can employ different functional representations of random variables and different uncertainty-propagation operators. To construct the network (i.e., a directed graph) from this collection of components, the NetUQ method simply associates endogenous-input random variables for each component with output random variables from other components; the method makes no other requirements on inter-component compatibility. This again promotes component independence, because, for example, the computational meshes employed to discretize neighboring components need not be perfectly matched. The resulting network uncertainty-propagation problem becomes: Compute output random variables for all components given all exogenous-input random variables.

To solve the network uncertainty-propagation problem, NetUQ applies the classical relaxation techniques of Jacobi and Gauss–Seidel iteration. Within each Jacobi iteration, all network edges corresponding to endogenous-input random variables are “cut” such that the endogenous-input random variables are set to their values at the previous iteration; then, all components perform local uncertainty propagation in an embarrassingly parallel manner to compute their output random variables at the current iteration. Within each Gauss–Seidel iteration, selected network edges are cut to create a feed-forward network (i.e., a directed acyclic graph); then, the method performs feed-forward uncertainty propagation to compute output random variables at the current iteration. We equip each of these methods with Anderson acceleration [2, 21] to accelerate convergence of the resulting fixed-point problem. We provide supporting error analysis for the method, which quantifies the error incurred by the NetUQ formulation for the case where the component uncertainty-propagation operators comprise an approximation of underlying “truth” component uncertainty-propagation operators; this arises, for example, in the case of truncated polynomial-chaos representations.

The remainder of the paper is outlined as follows. Section 2 formulates the problem by first describing the local component uncertainty-propagation problem (Section 2.1) and subsequently forming the network uncertainty-propagation problem by associating endogenous-input random variables for each component with output random variables from other components (Section 2.2). Section 3 describes the two relaxation methods employed by NetUQ to solve the network uncertainty-propagation problem: the Jacobi method (Section 3.1) and the Gauss–Seidel method (Section 3.2). Subsequently, Section 4 performs convergence analysis (Section 4.1) and introduces Anderson acceleration as a mechanism to accelerate convergence of the fixed-point iterations (Section 4.2). Section 5 performs error analysis, first by deriving *a priori* and *a posteriori* error bounds when the component uncertainty-propagation operators serve as approximations of underlying “truth” uncertainty-propagation operators (Section 5.1), and second by performing error analysis in the case where the component uncertainty-propagation operators restrict the output random variables to reside in a subspace of a “truth” subspace associated with the truth uncertainty-propagation operators (Section 5.2). Next, Section 6 reports numerical experiments that demonstrate the performance of NetUQ, namely its ability (1) to accurately propagate uncertainties in large-scale networks while promoting component independence, (2) to yield rapid convergence when deployed with Anderson acceleration, and (3) to generate significant speedups in both strong-scaling (Section 6.3) and weak-scaling (Section 6.4) scenarios. Finally, Section 7 concludes the paper.

2. Problem formulation. We now formulate the problem of interest: uncertainty propagation in systems composed of interconnected components. Section 2.1 provides the formulation for uncertainty propagation at the component level, which comprises the local problem, while Section 2.2 presents the network uncertainty propagation problem, which comprises the global problem.

2.1. Local problem: component uncertainty propagation. We begin by defining a probability space (Ω, \mathcal{F}, P) and assuming that the full system is composed of n interconnected components, the i th of which is characterized by

- *exogenous-input random variables* $\mathbf{u}_i : \Omega \rightarrow \mathbb{R}^{n_{\mathbf{u},i}}$, which correspond to random variables that are input directly to the component (e.g., material properties);
- *endogenous-input random variables* $\mathbf{y}_i : \Omega \rightarrow \mathbb{R}^{n_{\mathbf{y},i}}$, which correspond to random variables that are input from a neighboring component (e.g., boundary conditions imposed by a neighboring component);
- *output random variables* $\mathbf{x}_i : \Omega \rightarrow \mathbb{R}^{n_{\mathbf{x},i}}$, which can correspond to quantities of interest or random variables that associate with endogenous-input random variables for a neighboring component; and
- an *uncertainty-propagation operator*

$$(2.1) \quad \mathbf{f}_i : (\mathbf{y}_i, \mathbf{u}_i) \mapsto \mathbf{x}_i$$

with $\mathbf{f}_i : \mathbb{V}^{n_{\mathbf{y},i}} \times \mathbb{V}^{n_{\mathbf{u},i}} \rightarrow \mathbb{V}^{n_{\mathbf{x},i}}$, whose evaluation comprises the component uncertainty-propagation problem of computing output random variables from input random variables.

Here, \mathbb{V} denotes the vector space of real-valued random variables defined on (Ω, \mathcal{F}, P) , i.e., the set of mappings from the sample space Ω to the real numbers \mathbb{R} . We denote the norm on the vector space \mathbb{V} by $\|\cdot\|$ (e.g., $\|x\| = [\mathbb{E}[|x|^p]]^{1/p}$ for $x \in \mathbb{V}$ and some p). We also denote the norm on the vector space \mathbb{V}^N by $\|\cdot\|$ (e.g., $\|\mathbf{x}\| = [\sum_{i=1}^N \|x_i\|^q]^{1/q}$ for $\mathbf{x} \equiv [x_1 \dots x_N]^T \in \mathbb{V}^N$ and some q), where the use of the respective norms will be apparent from context. We emphasize that input and output random variables are characterized from the functional viewpoint, i.e., as functions acting on the sample space Ω .

Remark 2.1 (Polynomial-chaos expansions). Polynomial-chaos expansions (PCE) [23, 44, 45] comprise a widely adopted approach for computing functional representations of random variables. We now describe how PCE representations correspond to the general framework of component uncertainty propagation outlined above. To apply PCE in this context, we assume the existence of independent “germ” random variables $\boldsymbol{\xi} \equiv (\xi_1, \dots, \xi_{n_\xi}) \in \mathbb{V}^{n_\xi}$ with known distribution, and subsequently express the input and output random variables as polynomial functions of these random variables, i.e.,

$$(2.2) \quad \mathbf{u}_i = \sum_{j \in \mathcal{J}_{\mathbf{u},i}} \psi_j(\boldsymbol{\xi}) \mathbf{u}_{i,j}, \quad \mathbf{y}_i = \sum_{j \in \mathcal{J}_{\mathbf{y},i}} \psi_j(\boldsymbol{\xi}) \mathbf{y}_{i,j}, \quad \mathbf{x}_i = \sum_{j \in \mathcal{J}_{\mathbf{x},i}} \psi_j(\boldsymbol{\xi}) \mathbf{x}_{i,j},$$

where $\mathbf{j} \equiv (j_1, \dots, j_{n_\xi})$ denotes a multi-index; $\mathcal{J}_{\mathbf{u},i}, \mathcal{J}_{\mathbf{y},i}, \mathcal{J}_{\mathbf{x},i} \subseteq \mathbb{N}_0^{n_\xi}$ denote multi-index sets (e.g., defined using total-degree truncation such that $\mathcal{J}_{\mathbf{u},i} = \mathcal{J}_{\mathbf{y},i} = \mathcal{J}_{\mathbf{x},i} = \{\mathbf{j} \in \mathbb{N}_0^{n_\xi} \mid \|\mathbf{j}\|_1 \leq p\}$ for some p); $\mathbf{u}_{i,j} \in \mathbb{R}^{n_{\mathbf{u},i}}$ for $\mathbf{j} \in \mathcal{J}_{\mathbf{u},i}$, $\mathbf{y}_{i,j} \in \mathbb{R}^{n_{\mathbf{y},i}}$ for $\mathbf{j} \in \mathcal{J}_{\mathbf{y},i}$, and $\mathbf{x}_{i,j} \in \mathbb{R}^{n_{\mathbf{x},i}}$ for $\mathbf{j} \in \mathcal{J}_{\mathbf{x},i}$ denote PCE coefficients; and the PCE basis functions $\psi_j : \mathbb{R}^{n_\xi} \rightarrow \mathbb{R}$ are orthogonal with respect to the probability density function (PDF) of the germ random variables¹ and are defined as a product of n_ξ univariate polynomials such that

$$(2.3) \quad \psi_j : \boldsymbol{\xi} \mapsto \prod_{k=1}^{n_\xi} \tilde{\psi}_{j_k}(\xi_k),$$

¹For example, if the germ random variables ξ_i , $i = 1, \dots, n_\xi$ are independent uniform random variables defined on the domain $[-1, 1]$, the Legendre-polynomial PCE basis functions are employed, while if these random variables are independent standard normal random variables, Hermite-polynomial PCE basis functions are employed.

where $\tilde{\psi}_i : \mathbb{R} \rightarrow \mathbb{R}$ denotes a univariate polynomial of degree i .

According to the Cameron–Martin theorem [11], any finite-variance random variable can be represented as a convergent expansion (2.2) as the polynomial order and stochastic dimension n_ξ grow, given that the germ random variables ξ comprise independent Gaussian random variables. While this result has been conjectured for any general independent-component germ [43, 6], the authors in Ref. [20] prove a necessary and sufficient condition that the underlying probability measure of the germ be uniquely determined by its moments; this condition does not hold for a log-normal germ ξ , for example.

To apply PCE within a practical NetUQ setting, we first fix the germ random variables ξ (e.g., $\xi \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$) and multi-index sets $\mathcal{J}_{\mathbf{u},i}$, $\mathcal{J}_{\mathbf{y},i}$, and $\mathcal{J}_{\mathbf{x},i}$, and we subsequently determine the PCE basis functions ψ_j , $j \in \mathcal{J}_{\mathbf{u},i} \cup \mathcal{J}_{\mathbf{y},i} \cup \mathcal{J}_{\mathbf{x},i}$ from the PDF of the germ random variables ξ . With these quantities fixed, the random variables \mathbf{u}_i , \mathbf{y}_i , and \mathbf{x}_i are completely characterized via Eqs. (2.2) by their PCE coefficients $\mathbf{u}_i := (\mathbf{u}_{i,j})_{j \in \mathcal{J}_{\mathbf{u},i}} \in \mathbb{R}^{n_{\mathbf{u},i}}$, $\mathbf{y}_i := (\mathbf{y}_{i,j})_{j \in \mathcal{J}_{\mathbf{u},i}} \in \mathbb{R}^{n_{\mathbf{y},i}}$, and $\mathbf{x}_i \equiv (\mathbf{x}_{i,j})_{j \in \mathcal{J}_{\mathbf{x},i}} \in \mathbb{R}^{n_{\mathbf{x},i}}$, respectively, where $n_{\mathbf{u},i} := n_{\mathbf{u},i} |\mathcal{J}_{\mathbf{u},i}|$, $n_{\mathbf{y},i} := n_{\mathbf{y},i} |\mathcal{J}_{\mathbf{y},i}|$, and $n_{\mathbf{x},i} := n_{\mathbf{x},i} |\mathcal{J}_{\mathbf{x},i}|$.

Then, the uncertainty-propagation operator \mathbf{f}_i associated with this PCE formulation effectively computes the mapping from the input PCE coefficients \mathbf{u}_i , \mathbf{y}_i to the output PCE coefficients \mathbf{x}_i . Specifically, the discrete implementation of the uncertainty-propagation operator \mathbf{f}_i in this case is a discrete uncertainty-propagation operator

$$(2.4) \quad \mathbf{f}_i : (\mathbf{y}_i, \mathbf{u}_i) \mapsto \mathbf{x}_i$$

with $\mathbf{f}_i : \mathbb{R}^{n_{\mathbf{y},i} \times n_{\mathbf{u},i}} \rightarrow \mathbb{R}^{n_{\mathbf{x},i}}$.

The most common approaches for defining the discrete uncertainty-propagation operator \mathbf{f}_i are non-intrusive stochastic collocation [7, 37], non-intrusive spectral projection [34, 28], and intrusive stochastic projection (e.g., stochastic Galerkin [16, 8, 23], stochastic least-squares Petrov–Galerkin [31]).

In the sequel, we present the methodology in the general setting of functional representations of random variables. The discrete counterpart for PCE representations (or for any other functional representation defined by a finite number of deterministic scalar quantities) can be derived as above.

2.2. Global problem: network uncertainty propagation. We now describe the uncertainty-propagation problem for the full system composed of n interconnected components, each of which is equipped with the component uncertainty-propagation formulation described in Section 2.1. Denote by $\mathbf{u} := [\mathbf{u}_1^T \cdots \mathbf{u}_n^T]^T \in \mathbb{V}^{n_{\mathbf{u}}}$, $\mathbf{y} := [\mathbf{y}_1^T \cdots \mathbf{y}_n^T]^T \in \mathbb{V}^{n_{\mathbf{y}}}$, and $\mathbf{x} := [\mathbf{x}_1^T \cdots \mathbf{x}_n^T]^T \in \mathbb{V}^{n_{\mathbf{x}}}$ the vectorization of the component exogenous-inputs random variables, endogenous-input random variables, and output random variables, respectively, such that $n_{\mathbf{u}} := \sum_{i=1}^n n_{\mathbf{u},i}$, $n_{\mathbf{y}} := \sum_{i=1}^n n_{\mathbf{y},i}$, and $n_{\mathbf{x}} := \sum_{i=1}^n n_{\mathbf{x},i}$. Then, the full-system uncertainty-propagator operator is

$$(2.5) \quad \mathbf{f}_{\text{vec}} : (\mathbf{y}, \mathbf{u}) \mapsto \mathbf{x},$$

where $\mathbf{f}_{\text{vec}} : \mathbb{V}^{n_{\mathbf{y}}} \times \mathbb{V}^{n_{\mathbf{u}}} \rightarrow \mathbb{V}^{n_{\mathbf{x}}}$ comprises the vectorization of component uncertainty-propagation operators such that

$$(2.6) \quad \mathbf{f}_{\text{vec}} : (\mathbf{y}, \mathbf{u}) \mapsto \sum_{i=1}^n [\mathbf{E}_i^{\mathbf{x}}]^T \mathbf{f}_i(\mathbf{E}_i^{\mathbf{y}} \mathbf{y}, \mathbf{E}_i^{\mathbf{u}} \mathbf{u}).$$

Here, $\mathbf{E}_i^{\mathbf{x}} \in \{0, 1\}^{n_{\mathbf{x},i} \times n_{\mathbf{x}}}$, $\mathbf{E}_i^{\mathbf{y}} \in \{0, 1\}^{n_{\mathbf{y},i} \times n_{\mathbf{y}}}$, and $\mathbf{E}_i^{\mathbf{u}} \in \{0, 1\}^{n_{\mathbf{u},i} \times n_{\mathbf{u}}}$ denote selected rows of the identity matrix that extract quantities associated with the i th component from the network such that $\mathbf{E}_i^{\mathbf{x}} \mathbf{x} \equiv \mathbf{x}_i$, $\mathbf{E}_i^{\mathbf{y}} \mathbf{y} \equiv \mathbf{y}_i$, and $\mathbf{E}_i^{\mathbf{u}} \mathbf{u} \equiv \mathbf{u}_i$.

Critically, the full system is defined by connecting components such that each component's endogenous inputs correspond to the outputs of another component. We encode this relationship by the adjacency matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \in \{0, 1\}^{n_{\mathbf{y}} \times n_{\mathbf{x}}}$, which satisfies the relationship

$$(2.7) \quad \mathbf{y} \equiv \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \mathbf{x}$$

such that $[\mathbf{I}_{\mathbf{x}}^{\mathbf{y}}]_{ij}$ is equal to one if $[\mathbf{y}]_i \equiv [\mathbf{x}]_j$ and is zero otherwise. Note that because uncertainty propagation within each component is self contained, we do not require self connections, and thus the diagonal blocks of the adjacency matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{y}}$ are zero, where the (i, j) block is a $n_{\mathbf{y}, i} \times n_{\mathbf{x}, i}$ submatrix. We also admit the possibilities that a single component output may not correspond to an endogenous input for any other component (in which case the associated adjacency-matrix column is zero), or may constitute the endogenous input for multiple components (in which case the associated adjacency-matrix column has more than one nonzero element).

Substituting Eq. (2.7) into the full-system uncertainty-propagator (2.5) yields the following fixed-point problem: Given exogenous-input random variables $\mathbf{u} \in \mathbb{V}^{n_{\mathbf{u}}}$, compute output random variables $\mathbf{x}_* \equiv \mathbf{x}_*(\mathbf{u}) \in \mathbb{V}^{n_{\mathbf{x}}}$ that satisfy

$$(2.8) \quad \mathbf{r}(\mathbf{x}_*, \mathbf{u}) = \mathbf{0},$$

where $\mathbf{0} \in \mathbb{V}^{n_{\mathbf{x}}}$ denotes a vector of zero-valued random variables and

$$(2.9) \quad \mathbf{r} : (\mathbf{x}, \mathbf{u}) \mapsto \mathbf{x} - \mathbf{f}_{\text{vec}}(\mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \mathbf{x}, \mathbf{u})$$

with $\mathbf{r} : \mathbb{V}^{n_{\mathbf{x}} \times n_{\mathbf{u}}} \rightarrow \mathbb{V}^{n_{\mathbf{x}}}$ denoting the fixed-point residual. To simplify notation, we introduce an alternative version of the full-system uncertainty-propagation operator

$$(2.10) \quad \mathbf{f} : (\mathbf{x}, \mathbf{u}) \mapsto \mathbf{f}_{\text{vec}}(\mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \mathbf{x}, \mathbf{u}),$$

where $\mathbf{f} : \mathbb{V}^{n_{\mathbf{x}}} \times \mathbb{V}^{n_{\mathbf{u}}} \rightarrow \mathbb{V}^{n_{\mathbf{x}}}$. Note that the fixed-point residual is equivalently defined as $\mathbf{r} : (\mathbf{x}, \mathbf{u}) \mapsto \mathbf{x} - \mathbf{f}(\mathbf{x}, \mathbf{u})$.

We note that in many applications, computing specific quantities of interest (QoI) comprises the ultimate goal of the analysis. In the present context, we assign QoI random variables $\mathbf{z} \in \mathbb{V}^{n_{\mathbf{z}}}$ to be a subset of the network outputs, i.e., there exists an extraction matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{z}} \in \{0, 1\}^{n_{\mathbf{z}} \times n_{\mathbf{x}}}$ that satisfies the relationship

$$(2.11) \quad \mathbf{z} \equiv \mathbf{I}_{\mathbf{x}}^{\mathbf{z}} \mathbf{x}$$

such that $[\mathbf{I}_{\mathbf{x}}^{\mathbf{z}}]_{ij}$ is equal to one if $[\mathbf{z}]_i \equiv [\mathbf{x}]_j$ and is zero otherwise.

Figure 2.1 provides a graphical depiction of the NetUQ formulation, which can be interpreted as performing uncertainty propagation in networks, wherein each node associates with a component uncertainty-propagation operator \mathbf{f}_i , and each edge corresponds to a collection of random variables.

Remark 2.2 (Component independence). This formulation promotes component independence, as the only formal inter-component compatibility requirement is the existence of an adjacency matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{y}}$ that associates component output random variables with endogenous input random variables of other components. In particular, each set of random variables can be represented using different functional representations and components can employ completely different uncertainty-propagation operators.

3. Relaxation methods. In principle, the fixed-point system (2.8) can be solved using a variety of techniques. While Newton's method is often employed for the solution of systems of nonlinear algebraic equations due to its local quadratic convergence rate, it relies on the ability to compute the gradient $\partial \mathbf{r} / \partial \mathbf{x}$. In the present context, this requires computing the gradient of the output random variables with respect to the endogenous-input random variables for each component, i.e., $\partial \mathbf{f}_i / \partial \mathbf{y}_i$ must be computable for $i = 1, \dots, n$. This is frequently impractical, e.g., when the simulation code used to compute a component uncertainty-propagation operator is available only as a “black box”.

As such, we proceed by assuming that only the component uncertainty-propagation operators \mathbf{f}_i , $i = 1, \dots, n$ themselves are available, and consider classical relaxation methods (i.e., Jacobi, Gauss–Seidel) to solve the fixed-point problem (2.8), as these methods do not require gradients and promote component independence. If each node in the network corresponds to subdomain in a partial-differential-equation problem, then this approach can be considered an overlapping domain-decomposition strategy; however, the formulation does not rely on any particular interpretation of the components.

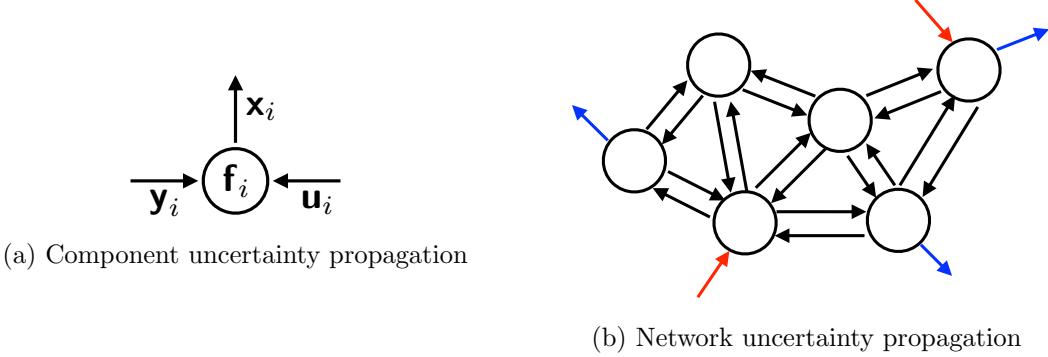


Figure 2.1: *NetUQ formulation*. Figure shows how the network uncertainty-propagation problem can be derived from component uncertainty-propagation problems shown in Fig. 2.1a. In Fig. 2.1b, black edges correspond to endogenous-input random variables \mathbf{y} , red edges correspond to exogenous-input random variables \mathbf{u} , and blue edges correspond to QoI random variables \mathbf{z} . Note that output random variables \mathbf{x} are mapped to endogenous-input random variables \mathbf{y} and/or \mathbf{z} via the operators \mathbf{I}_x^y (see Eq. (2.7)) and \mathbf{I}_x^z (see Eq. (2.11)), respectively. We note that a network is defined from component uncertainty-propagation operators \mathbf{f}_i , $i = 1, \dots, n$, the adjacency matrix \mathbf{I}_x^y , and the extraction matrix \mathbf{I}_x^z .

3.1. Jacobi method. We first consider applying a variant of the Jacobi method (i.e., additive Schwarz) to solve fixed-point problem (2.8); Algorithm 3.1 reports the algorithm. At each iteration, this approach performs independent, embarrassingly parallel component uncertainty propagation (Steps 3–5), applies a relaxation update (Step 6), and subsequently updates the endogenous-input random variables from the output random variables just computed for neighboring components (Step 7). From the network perspective, this approach is equivalent to “splitting” all endogenous-input edges at each iteration and allowing components to perform independent uncertainty propagation; see Figure 3.1a.

At the network level, Jacobi iteration k can be expressed simply as

$$(3.1) \quad \tilde{\mathbf{x}} = \mathbf{f}(\mathbf{x}^{(k)}, \mathbf{u})$$

$$(3.2) \quad \mathbf{x}^{(k+1)} = \omega \tilde{\mathbf{x}} + (1 - \omega) \mathbf{x}^{(k)},$$

where $\omega > 0$ is a relaxation factor, which is set to $\omega = 1$ for the classical Jacobi method. Under-relaxation corresponds to $\omega < 0$, while over-relaxation corresponds to $\omega > 1$. This parameter controls convergence of the algorithm as well as error analysis as will be discussed in Remarks 4.2 and 5.1.

3.2. Gauss–Seidel method. Naturally, we also consider a variant of the Gauss–Seidel method (i.e., multiplicative Schwarz) to solve fixed-point problem (2.8); Algorithm 3.2 reports the algorithm. The benefit of the Gauss–Seidel method with respect to the Jacobi method is that it enables more updated information to be used within each iteration, at the expense of reduced parallelism. To achieve this, each Gauss–Seidel iteration performs feed-forward uncertainty propagation, which requires “splitting” network edges in a manner that generates a directed acyclic graph (DAG), and subsequently performing feed-forward uncertainty propagation within the DAG; see Figure 3.1b. In principle, each Gauss–Seidel iteration can employ a different DAG; for simplicity in exposition, we restrict consideration to a constant DAG.

To generate a DAG, we introduce n -tuple $\mathbf{p} \equiv (p_1, \dots, p_n) \in \mathbb{N}^n$ that provides a permutation

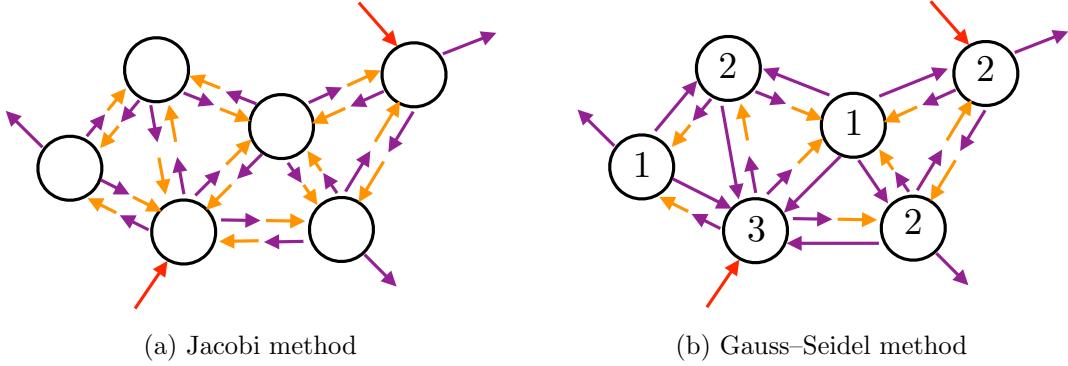


Figure 3.1: *Relaxation methods*. Here, random variables that were computed at the previous iteration are shown in orange, random variables that are computed at the current iteration are shown in violet, and exogenous-input random variables—whose values are fixed—are shown in red. For the Gauss–Seidel method, the order in which a given node is processed is included in Figure 3.1b. For the Jacobi method, note that all edges associated with endogenous-input random variables are “split” at each iteration to enable embarrassingly parallel component uncertainty propagation. On the other hand, for the Gauss–Seidel method, only a subset of such edges are “split” at each iteration such that a directed acyclic graph (DAG) is created. As a result, each iteration corresponds to uncertainty propagation in a feed-forward network. In this example, the chosen edge splitting yields a DAG that incurs three sequential steps per iteration. Note that many such DAGs exist, and each associates with a particular permutation of the adjacency matrix \mathbf{I}_x^y .

Algorithm 3.1 Jacobi (The Jacobi method for NetUQ)

Input: Component uncertainty-propagation operators \mathbf{f}_i , $i = 1, \dots, n$; adjacency matrix \mathbf{I}_x^y ; exogenous-input random variables \mathbf{u}_i , $i = 1, \dots, n$; initial guesses for endogenous-input random variables $\mathbf{y}_i^{(0)}$, $i = 1, \dots, n$; relaxation factor ω

Output: converged endogenous-input random variables $\mathbf{y}_i^{(K)}$; converged output random variables $\mathbf{x}_i^{(K)}$

```

1:  $k \leftarrow 0$ 
2: while not converged do
3:   for  $i = 1, \dots, n$  do {Execute in parallel}
4:      $\tilde{\mathbf{x}}_i = \mathbf{f}_i(\mathbf{y}_i^{(k)}, \mathbf{u}_i)$ 
5:   end for
6:    $\mathbf{x}^{(k+1)} = \omega \tilde{\mathbf{x}} + (1 - \omega) \mathbf{x}^{(k)}$ 
7:    $\mathbf{y}^{(k+1)} = \mathbf{I}_x^y \mathbf{x}^{(k+1)}$ 
8:    $k \leftarrow k + 1$ 
9: end while
10:  $K \leftarrow k$ 

```

of natural numbers one to n . We then define the block permutation matrices

$$(3.3) \quad \mathbf{P}_p^x := \begin{bmatrix} \mathbf{E}_{p1}^x \\ \vdots \\ \mathbf{E}_{pn}^x \end{bmatrix}, \quad \mathbf{P}_p^y := \begin{bmatrix} \mathbf{E}_{p1}^y \\ \vdots \\ \mathbf{E}_{pn}^y \end{bmatrix}$$

and block decomposition of the permuted adjacency matrix

$$(3.4) \quad \mathbf{P}_p^y \mathbf{I}_x^y [\mathbf{P}_p^x]^T = \mathbf{L}_p + \mathbf{U}_p,$$

Algorithm 3.2 GaussSeidel (The Gauss–Seidel method for NetUQ)

Input: Component uncertainty-propagation operators \mathbf{f}_i , $i = 1, \dots, n$; adjacency matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{y}}$; exogenous-input random variables \mathbf{u}_i , $i = 1, \dots, n$; initial guesses for endogenous-input random variables $\mathbf{y}_i^{(0)}$, $i = 0, \dots, n$; relaxation factor ω ; sequence of permutation matrices $\mathbf{p}^{(k)}$, $k = 1, \dots$

Output: converged endogenous-input random variables $\mathbf{y}_i^{(K)}$; converged output random variables $\mathbf{x}_i^{(K)}$

```

1:  $k \leftarrow 0$ 
2: while not converged do
3:    $\mathbf{p} \leftarrow \mathbf{p}^{(k)}$ 
4:    $(\mathbf{P}_{\mathbf{p}}^{\mathbf{x}}, \mathbf{P}_{\mathbf{p}}^{\mathbf{y}}, \mathbf{L}_{\mathbf{p}}, \mathbf{U}_{\mathbf{p}}, \mathcal{I}_{\mathbf{p}}, n_{\text{seq},\mathbf{p}}) = \text{DAG}(\mathbf{p}, \mathbf{I}_{\mathbf{x}}^{\mathbf{y}})$ 
5:    $\bar{\mathbf{y}} \leftarrow \mathbf{y}^{(k)}$ 
6:   for  $\ell = 1, \dots, n_{\text{seq},\mathbf{p}}$  do {Execute sequentially}
7:     for  $i \in \mathcal{I}_{\mathbf{p}}^{\ell}$  do {Execute in parallel}
8:        $\tilde{\mathbf{x}}_i = \mathbf{f}_i(\bar{\mathbf{y}}_i, \mathbf{u}_i)$ 
9:        $\bar{\mathbf{y}} \leftarrow \bar{\mathbf{y}} + \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} [\mathbf{E}_i^{\mathbf{x}}]^T (\tilde{\mathbf{x}}_i - \mathbf{x}_i^{(k)})$ 
10:    end for
11:   end for
12:    $\mathbf{x}^{(k+1)} = \omega \tilde{\mathbf{x}} + (1 - \omega) \mathbf{x}^{(k)}$ 
13:    $\mathbf{y}^{(k+1)} = \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \mathbf{x}^{(k+1)}$ 
14:    $k \leftarrow k + 1$ 
15: end while
16:  $K \leftarrow k$ 

```

where $\mathbf{L}_{\mathbf{p}} \in \{0, 1\}^{n_{\mathbf{y}} \times n_{\mathbf{x}}}$ is strictly block lower triangular and $\mathbf{U}_{\mathbf{p}} \in \{0, 1\}^{n_{\mathbf{y}} \times n_{\mathbf{x}}}$ is strictly block upper triangular.

By “splitting” all edges in the adjacency matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{y}}$ whose indices match the nonzero elements of $[\mathbf{P}_{\mathbf{p}}^{\mathbf{y}}]^T \mathbf{U}_{\mathbf{p}} \mathbf{P}_{\mathbf{p}}^{\mathbf{x}}$, we can create a DAG, as the components can be processed in sequential order p_i , $i = 1, \dots, n$. At the network level, Gauss–Seidel iteration k can be expressed as

$$(3.5) \quad \tilde{\mathbf{x}} = \mathbf{f}_{\text{vec}}([\mathbf{P}_{\mathbf{p}}^{\mathbf{y}}]^T \mathbf{L}_{\mathbf{p}} \mathbf{P}_{\mathbf{p}}^{\mathbf{x}} \tilde{\mathbf{x}} + [\mathbf{P}_{\mathbf{p}}^{\mathbf{y}}]^T \mathbf{U}_{\mathbf{p}} \mathbf{P}_{\mathbf{p}}^{\mathbf{x}} \mathbf{x}^{(k)}, \mathbf{u})$$

$$(3.6) \quad \mathbf{x}^{(k+1)} = \omega \tilde{\mathbf{x}} + (1 - \omega) \mathbf{x}^{(k)},$$

where again $\omega > 0$ is a relaxation factor and the implicit expression (3.5) can be rewritten explicitly via recursion as

$$(3.7) \quad \tilde{\mathbf{x}}_{p_i} = \mathbf{f}_{p_i} \left(\mathbf{E}_{p_i}^{\mathbf{y}} \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \sum_{j=1}^{i-1} [\mathbf{E}_{p_j}^{\mathbf{x}}]^T \tilde{\mathbf{x}}_{p_j} + \mathbf{E}_{p_i}^{\mathbf{y}} \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \sum_{j=i+1}^n [\mathbf{E}_{p_j}^{\mathbf{x}}]^T \mathbf{E}_{p_j}^{\mathbf{x}} \mathbf{x}^{(k)}, \mathbf{u} \right), \quad i = 1, \dots, n.$$

By comparing with the Jacobi update (3.1), it is apparent that the Gauss–Seidel update to $\tilde{\mathbf{x}}$ in Eq. (3.5) is implicit rather than explicit, and so uses more updated information than Jacobi at each iteration; however, this is done at the expense of requiring sequential processing of the components.

To mitigate the sequential-processing burden, we observe that additional parallelism may be exposed by examining the sparsity pattern of $\mathbf{L}_{\mathbf{p}}$, as this matrix encodes dependencies among the preserved edges in the (permuted) network.

Algorithm 3.3 describes this process. Given a permutation tuple \mathbf{p} , this algorithm effectively “splits” all edges associated with nonzero elements of $[\mathbf{P}_{\mathbf{p}}^{\mathbf{y}}]^T \mathbf{U}_{\mathbf{p}} \mathbf{P}_{\mathbf{p}}^{\mathbf{x}}$, and analyzes the sparsity pattern of the block lower triangular matrix $\mathbf{L}_{\mathbf{p}}$ to determine (1) the minimum number of sequential steps required to propagate uncertainties in the resulting DAG, and (2) which components can be processed within each sequential step. This procedure is called in Step 4 of Algorithm 3.2 to ensure

Algorithm 3.3 DAG (Generate a DAG from a permutation)

Input: Permutation $\mathbf{p} \equiv (p_1, \dots, p_n)$ of the natural numbers one to n ; adjacency matrix $\mathbf{I}_{\mathbf{x}}^{\mathbf{y}}$
Output: Permutation matrices $\mathbf{P}_{\mathbf{p}}^{\mathbf{x}}$ and $\mathbf{P}_{\mathbf{p}}^{\mathbf{y}}$; block lower and upper triangular matrices $\mathbf{L}_{\mathbf{p}}$ and $\mathbf{U}_{\mathbf{p}}$; tuple of components in sequential processing order $\mathcal{I}_{\mathbf{p}} := (\mathcal{I}_{\mathbf{p}}^1, \dots, \mathcal{I}_{\mathbf{p}}^{n_{\text{seq}, \mathbf{p}}})$; number of sequential steps $n_{\text{seq}, \mathbf{p}}$

```

1: Compute  $\mathbf{P}_{\mathbf{p}}^{\mathbf{x}}$ ,  $\mathbf{P}_{\mathbf{p}}^{\mathbf{y}}$ ,  $\mathbf{L}_{\mathbf{p}}$ , and  $\mathbf{U}_{\mathbf{p}}$  from Eqs. (3.3) and (3.4).
2:  $k \leftarrow 0$ ;  $\mathcal{I} = \emptyset$ 
3: while  $\mathcal{I} \neq \{1, \dots, n\}$  do
4:    $k \leftarrow k + 1$ 
5:    $\mathcal{I}_{\mathbf{p}}^k \leftarrow \emptyset$ 
6:   for  $i = 1, \dots, n$  do
7:     if block  $(i, j)$  of  $\mathbf{L}_{\mathbf{p}}$  is zero for all  $j \in \{1, \dots, n\} \setminus \mathcal{I}$  then
8:        $\mathcal{I}_{\mathbf{p}}^k \leftarrow \mathcal{I}_{\mathbf{p}}^k \cup p_i$ 
9:     end if
10:    end for
11:    $\mathcal{I} \leftarrow \mathcal{I} \cup \mathcal{I}_{\mathbf{p}}^k$ 
12: end while
13:  $n_{\text{seq}, \mathbf{p}} = k$ 

```

the Gauss–Seidel iterations employ the fewest number of sequential steps. Clearly, different permutations \mathbf{p} will associate with different numbers of required sequential steps; graph coloring [29] provides a mechanism to determine the minimum number of sequential steps for a given network. In addition to this consideration, different permutations may lead to different convergence rates as will be discussed in Remark 4.2.

4. Convergence analysis and Anderson acceleration. Both Jacobi and Gauss–Seidel methods can be expressed as fixed-point iterations

$$(4.1) \quad \mathbf{x}^{(k+1)} = \mathbf{h}(\mathbf{x}^{(k)}, \mathbf{u})$$

for $k = 0, \dots, K$, where the mappings $\mathbf{x} \mapsto \mathbf{x} - \mathbf{h}(\mathbf{x}, \mathbf{u})$ and $\mathbf{x} \mapsto \mathbf{x} - \mathbf{f}(\mathbf{x}, \mathbf{u})$ have the same roots for any fixed value of the exogenous-input random variables $\mathbf{u} \in \mathbb{V}^{n_{\mathbf{u}}}$. In the case of Jacobi iteration, we see from Eqs. (3.1)–(3.2) that the fixed-point operator is $\mathbf{h} \leftarrow \mathbf{h}_J$ with

$$(4.2) \quad \mathbf{h}_J : (\mathbf{x}, \mathbf{u}; \omega) \mapsto \omega \mathbf{f}(\mathbf{x}, \mathbf{u}) + (1 - \omega) \mathbf{x}.$$

In the case of Gauss–Seidel iteration, we see from Eqs. (3.5)–(3.7) that the associated operator is $\mathbf{h} \leftarrow \mathbf{h}_{GS}$, which is defined recursively for $i = 1, \dots, n$ as

$$(4.3) \quad \begin{aligned} \mathbf{E}_{p_i}^{\mathbf{x}} \mathbf{h}_{GS} : (\mathbf{x}, \mathbf{u}; \omega, \mathbf{p}) \mapsto & \omega \mathbf{f}_{p_i} \left(\mathbf{E}_{p_i}^{\mathbf{y}} \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \sum_{j=1}^{i-1} [\mathbf{E}_{p_j}^{\mathbf{x}}]^T \mathbf{E}_{p_j}^{\mathbf{x}} \mathbf{h}_{GS}(\mathbf{x}, \mathbf{u}; \omega, \mathbf{p}) + \mathbf{E}_{p_i}^{\mathbf{y}} \mathbf{I}_{\mathbf{x}}^{\mathbf{y}} \sum_{j=i+1}^n [\mathbf{E}_{p_j}^{\mathbf{x}}]^T \mathbf{E}_{p_j}^{\mathbf{x}} \mathbf{x}, \mathbf{u} \right) \\ & + (1 - \omega) \mathbf{E}_{p_i}^{\mathbf{x}} \mathbf{x}, \end{aligned}$$

where we have made its dependence on the relaxation factor ω and permutation \mathbf{p} explicit.

4.1. Convergence analysis. We now adopt a standard result from the theory of fixed-point iterations to the present context. This result is valid for any (fixed) value of the exogenous-input random variables $\mathbf{u} \in \mathbb{V}^{n_{\mathbf{u}}}$ and relies on the following assumption:

A1 The fixed-point operator \mathbf{h} is a contraction, i.e., it is Lipschitz continuous such that

$$(4.4) \quad \|\mathbf{h}(\mathbf{x}_1, \mathbf{u}) - \mathbf{h}(\mathbf{x}_2, \mathbf{u})\| \leq L_{\mathbf{h}}(\mathbf{u}) \|\mathbf{x}_1 - \mathbf{x}_2\|$$

for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{V}^{n_{\mathbf{x}}}$ with $L_{\mathbf{h}}(\mathbf{u}) < 1$ for any $\mathbf{u} \in \mathbb{V}^{n_{\mathbf{u}}}$.

Proposition 4.1. *If Assumption A1 holds, then the iterations*

$$(4.5) \quad \mathbf{x}^{(k+1)} = \mathbf{h}(\mathbf{x}^{(k)}, \mathbf{u}), \quad k = 0, 1, \dots$$

converge q-linearly to a fixed point \mathbf{x}_ satisfying $\mathbf{x}_* = \mathbf{h}(\mathbf{x}_*, \mathbf{u})$ with convergence rate $L_{\mathbf{h}}(\mathbf{u})$ for any $\mathbf{u} \in \mathbb{V}^{n_u}$.*

Proof. Subtracting $\mathbf{x}_* = \mathbf{h}(\mathbf{x}_*, \mathbf{u})$ from Eq. (4.5), and employing Lipschitz continuity of the fixed-point operator \mathbf{h} , yields $\|\mathbf{x}^{(k+1)} - \mathbf{x}_*\| \leq L_{\mathbf{h}}(\mathbf{u})\|\mathbf{x}^{(k)} - \mathbf{x}_*\|$, which leads to the desired result. ■

Remark 4.2 (Controlling convergence). For a fixed network, the Jacobi fixed-point operator $\mathbf{h}_J(\cdot, \cdot; \omega)$ is parameterized by the relaxation factor ω , while the Gauss–Seidel fixed-point operator $\mathbf{h}_{GS}(\cdot, \cdot; \omega, \mathbf{p})$ is parameterized by both the relaxation factor ω and the permutation \mathbf{p} . Thus, these parameters provide mechanisms for controlling convergence, as modifying them modifies the Lipschitz constants of these fixed-point operators, which in turn affects satisfaction of Assumption A1 and the convergence rate according to Proposition 4.1. The numerical experiments vary these quantities to numerically assess their effect on performance of NetUQ.

4.2. Anderson acceleration. Proposition 4.1 showed that as long as the fixed-point operator is a contraction, the fixed-point iterations generated by Jacobi or Gauss–Seidel iterations converge q-linearly. Compared with other methods for solving systems of nonlinear equations (e.g., Newton’s method), this rate of convergence is rarely competitive; as such, relaxation methods are used typically only in cases where these types of solvers are not applicable (e.g., if the residual Jacobian cannot be exposed easily).

To improve the convergence rate of fixed-point iterations, researchers have recently rediscovered [21] Anderson acceleration [2], which provides a practical modification to the fixed-point-iteration updates and has been shown to substantially improve convergence. For linear problems, one can demonstrate that convergence with Anderson acceleration is no slower than the original fixed-point iteration [40] and is essentially equivalent to the generalized minimum residual (GMRES) method [41]. For nonlinear problems, it can be shown that Anderson acceleration is a multisecant method [21], and similar convergence-rate results exist [40]. All convergence results hold under the assumption of a contractive mapping, i.e., under Assumption A1.

Algorithm 4.1 reports the Anderson acceleration method adopted to the present context, where the fixed-point operator is either $\mathbf{h} \leftarrow \mathbf{h}_J$ or $\mathbf{h} \leftarrow \mathbf{h}_{GS}$. We note that employing a memory of $m = 0$ recovers the original fixed-point iterations (4.1). Numerical experiments performed in Section 6 illustrate the ability of Anderson acceleration to substantially improve the convergence rate of the proposed NetUQ method in the case of both Jacobi and Gauss–Seidel iteration.

5. Error analysis. We now assess the error incurred by the network formulation when the full-system uncertainty-propagation operator \mathbf{f} constitutes an approximation of an underlying “truth” operator. In particular, it is common practice for the component uncertainty-propagation operators \mathbf{f}_i , $i = 1, \dots, n$ to comprise approximations of underlying “truth” component uncertainty-propagation operators $\tilde{\mathbf{f}}_i : \mathbb{V}^{n_y, i} \times \mathbb{V}^{n_u, i} \rightarrow \mathbb{V}^{n_x, i}$, $i = 1, \dots, n$. For example, this occurs when component uncertainty-propagation operators \mathbf{f}_i , $i = 1, \dots, n$ associate with computing low-dimensional polynomial-chaos representation of the output random variables \mathbf{x}_i , $i = 1, \dots, n$, while $\tilde{\mathbf{f}}_i$, $i = 1, \dots, n$ might associate with computing a high-dimensional polynomial-chaos representation of the output random variables.

We begin by noting that for any function $\mathbf{h} : \mathbb{V}^{n_x} \times \mathbb{V}^{n_u} \rightarrow \mathbb{V}^{n_x}$ for which the mappings $\mathbf{x} \mapsto \mathbf{x} - \mathbf{h}(\mathbf{x}, \mathbf{u})$ and $\mathbf{x} \mapsto \mathbf{x} - \mathbf{f}(\mathbf{x}, \mathbf{u})$ have the same roots for any $\mathbf{u} \in \mathbb{V}^{n_u}$, the solution \mathbf{x}_* to the fixed-point system (2.8) also satisfies the fixed-point system

$$(5.1) \quad \mathbf{x}_* = \mathbf{h}(\mathbf{x}_*, \mathbf{u})$$

for any $\mathbf{u} \in \mathbb{V}^{n_u}$. For example, $\mathbf{h}(\cdot, \cdot) = \mathbf{h}_J(\cdot, \cdot; \omega)$ and $\mathbf{h}(\cdot, \cdot) = \mathbf{h}_{GS}(\cdot, \cdot; \omega, \mathbf{p})$ are the Jacobi and Gauss–Seidel fixed-point operators, respectively, which satisfy this property.

Algorithm 4.1 AndersonAcceleration (Anderson acceleration)

Input: Fixed-point operator \mathbf{h} ; exogenous-input random variables \mathbf{u} ; initial guess for output random variables $\mathbf{x}^{(0)}$; memory $m \in \mathbb{N}_0$

Output: Converged solution $\mathbf{x}^{(K)}$

- 1: $k \leftarrow 0$
- 2: **while** not converged **do**
- 3: $\hat{\mathbf{x}}^{(k+1)} = \mathbf{h}(\mathbf{x}^{(k)}; \mathbf{u})$
- 4: $\mathbf{f}^{(k)} = \hat{\mathbf{x}}^{(k+1)} - \mathbf{x}^{(k)}$
- 5: $m^{(k)} = \min(m, k)$
- 6: Compute $(\alpha_0^{(k)}, \dots, \alpha_{m^{(k)}}^{(k)})$ as the solution to

$$\underset{(\alpha_0, \dots, \alpha_{m^{(k)}})}{\text{minimize}} \quad \left\| \sum_{i=0}^{m^{(k)}} \mathbf{f}^{(k-i)} \alpha_i \right\|_2 \quad \text{subject to} \quad \sum_{i=0}^{m^{(k)}} \alpha_i = 1$$

- 7: $\mathbf{x}^{(k+1)} = \sum_{i=0}^{m^{(k)}} \alpha_i^{(k)} \hat{\mathbf{x}}^{(k+1-i)}$
- 8: $k \leftarrow k + 1$

9: **end while**

10: $K \leftarrow k$

We also define the truth full-system uncertainty-propagator as $\bar{\mathbf{f}}_{\text{vec}} : \mathbb{V}^{n_y} \times \mathbb{V}^{n_u} \rightarrow \mathbb{V}^{n_x}$, which comprises the vectorization of truth component uncertainty-propagation operators such that

$$(5.2) \quad \bar{\mathbf{f}}_{\text{vec}} : (\mathbf{y}, \mathbf{u}) \mapsto \sum_{i=1}^n [\mathbf{E}_i^x]^T \bar{\mathbf{f}}_i(\mathbf{E}_i^y \mathbf{y}, \mathbf{E}_i^u \mathbf{u}).$$

Substituting the relationship between outputs and endogenous inputs (2.7) into the truth full-system uncertainty-propagator (5.2) yields the following truth fixed-point problem: Given exogenous-input random variables $\mathbf{u} \in \mathbb{V}^{n_u}$, compute truth output random variables $\bar{\mathbf{x}}_* \equiv \bar{\mathbf{x}}_*(\mathbf{u}) \in \mathbb{V}^{n_x}$ that satisfy

$$(5.3) \quad \bar{\mathbf{r}}(\bar{\mathbf{x}}_*, \mathbf{u}) = \mathbf{0},$$

where

$$(5.4) \quad \bar{\mathbf{r}} : (\mathbf{x}, \mathbf{u}) \mapsto \mathbf{x} - \bar{\mathbf{f}}_{\text{vec}}(\mathbf{I}_{\mathbf{x}}^y \mathbf{x}, \mathbf{u})$$

with $\bar{\mathbf{r}} : \mathbb{V}^{n_x} \times \mathbb{V}^{n_u} \rightarrow \mathbb{V}^{n_x}$ denotes the truth fixed-point residual. As before, to simplify notation, we introduce an alternative version of the truth full-system uncertainty-propagation operator

$$(5.5) \quad \bar{\mathbf{f}} : (\mathbf{x}, \mathbf{u}) \mapsto \bar{\mathbf{f}}_{\text{vec}}(\mathbf{I}_{\mathbf{x}}^y \mathbf{x}, \mathbf{u}),$$

where $\bar{\mathbf{f}} : \mathbb{V}^{n_x} \times \mathbb{V}^{n_u} \rightarrow \mathbb{V}^{n_x}$. Note that the truth fixed-point residual is equivalently defined as $\bar{\mathbf{r}} : (\mathbf{x}, \mathbf{u}) \mapsto \mathbf{x} - \bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$.

As above, we note that for any function $\bar{\mathbf{h}} : \mathbb{V}^{n_x} \times \mathbb{V}^{n_u} \rightarrow \mathbb{V}^{n_x}$ for which the mappings $\mathbf{x} \mapsto \mathbf{x} - \bar{\mathbf{h}}(\mathbf{x}, \mathbf{u})$ and $\mathbf{x} \mapsto \mathbf{x} - \bar{\mathbf{f}}(\mathbf{x}, \mathbf{u})$ have the same roots for any $\mathbf{u} \in \mathbb{V}^{n_u}$, the solution $\bar{\mathbf{x}}_*$ to the fixed-point system (5.3) also satisfies the fixed-point system

$$(5.6) \quad \bar{\mathbf{x}}_* = \bar{\mathbf{h}}(\bar{\mathbf{x}}_*, \mathbf{u}).$$

We refer to such an operator $\bar{\mathbf{h}}$ as the truth fixed-point operator. For example, one could employ $\bar{\mathbf{h}}(\cdot, \cdot) = \bar{\mathbf{h}}_J(\cdot, \cdot; \bar{\omega})$ with $\bar{\mathbf{h}}_J : (\mathbf{x}, \mathbf{u}; \bar{\omega}) \mapsto \bar{\omega} \mathbf{f}(\mathbf{x}, \mathbf{u}) + (1 - \bar{\omega}) \mathbf{x}$ as a Jacobi fixed-point operator parameterized by the relaxation factor $\bar{\omega} > 0$, or $\bar{\mathbf{h}}(\cdot, \cdot) = \bar{\mathbf{h}}_{\text{GS}}(\cdot, \cdot; \bar{\omega}, \bar{\mathbf{p}})$ as a Gauss–Seidel fixed-point

operator defined analogously to \mathbf{h}_{GS} and parameterized by the relaxation factor $\bar{\omega}$ and permutation $\bar{\mathbf{p}} \in \mathbb{N}^n$.

In the remainder of this section, we drop dependence of all quantities on the exogenous-input random variables \mathbf{u} for notational simplicity. The results can be interpreted as holding for any (fixed) value of $\mathbf{u} \in \mathbb{V}^{n_u}$.

5.1. Error bounds. We first derive *a priori* and *a posteriori* bounds for the error $\|\bar{\mathbf{x}}_* - \mathbf{x}_*\|$. We begin by introducing the following assumption, which will be used to derive the *a posteriori* error bound:

A2 The truth fixed-point operator $\bar{\mathbf{h}}$ is Lipschitz continuous such that $\|\bar{\mathbf{h}}(\mathbf{x}_1) - \bar{\mathbf{h}}(\mathbf{x}_2)\| \leq L_{\bar{\mathbf{h}}} \|\mathbf{x}_1 - \mathbf{x}_2\|$ for all $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{V}^{n_x}$ with $L_{\bar{\mathbf{h}}} < 1$.

Remark 5.1 (Practical satisfaction of Assumptions A1 and A2). Analogously to Remark 4.2, we note that the flexibility in the definitions of the fixed-point operators \mathbf{h} and $\bar{\mathbf{h}}$ allows them to be defined such that Assumptions A1 and A2 are satisfied. For example, if $\mathbf{h}(\cdot, \cdot) = \mathbf{h}_J(\cdot, \cdot; \omega)$ (resp. $\bar{\mathbf{h}}(\cdot, \cdot) = \bar{\mathbf{h}}_J(\cdot, \cdot; \bar{\omega})$), then the relaxation factor ω (resp. $\bar{\omega}$) can often be chosen to satisfy Assumption A1 (resp. A2). Alternatively, if $\mathbf{h}(\cdot, \cdot) = \mathbf{h}_{\text{GS}}(\cdot, \cdot; \omega, \mathbf{p})$ (resp. $\bar{\mathbf{h}}(\cdot, \cdot) = \bar{\mathbf{h}}_{\text{GS}}(\cdot, \cdot; \bar{\omega}, \bar{\mathbf{p}})$), then the relaxation factor ω and permutation \mathbf{p} (resp. $\bar{\omega}$ and $\bar{\mathbf{p}}$) can often be chosen to satisfy Assumption A1 (resp. A2). Indeed, ensuring that Assumptions A1 and A2 are satisfied is required to ensure the associated fixed-point iterations converge according to Proposition 4.1.

We proceed by deriving the error bounds, which rely on Assumptions A1 and A2.

Proposition 5.2 (A priori error bound for $\|\bar{\mathbf{x}}_* - \mathbf{x}_*\|$). *If Assumption A1 holds, then*

$$(5.7) \quad \|\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq \frac{1}{1 - L_{\mathbf{h}}} \|\bar{\mathbf{x}}_* - \mathbf{h}(\bar{\mathbf{x}}_*)\|.$$

Proof. We subtract Eq. (5.1) from (5.6) and add and subtract $\mathbf{h}(\bar{\mathbf{x}}_*)$ to obtain

$$(5.8) \quad \bar{\mathbf{x}}_* - \mathbf{x}_* = \bar{\mathbf{h}}(\bar{\mathbf{x}}_*) - \mathbf{h}(\bar{\mathbf{x}}_*) + \mathbf{h}(\bar{\mathbf{x}}_*) - \mathbf{h}(\mathbf{x}_*).$$

Applying the triangle inequality and Lipschitz continuity of \mathbf{h} gives

$$(5.9) \quad \|\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq \|\bar{\mathbf{h}}(\bar{\mathbf{x}}_*) - \mathbf{h}(\bar{\mathbf{x}}_*)\| + L_{\mathbf{h}} \|\bar{\mathbf{x}}_* - \mathbf{x}_*\|.$$

Finally, using $L_{\mathbf{h}} < 1$ and $\bar{\mathbf{h}}(\bar{\mathbf{x}}_*) = \bar{\mathbf{x}}_*$ (from Eq. (5.6)) yields the desired result. ■

We note that because the mapping $\mathbf{x} \mapsto \mathbf{x} - \mathbf{h}(\mathbf{x})$ corresponds to the residual of the fixed-point system (5.1) and the (generally unknown) truth solution $\bar{\mathbf{x}}_*$ appears in the bound, inequality (5.7) can be considered a residual-based *a priori* error bound.

Proposition 5.3 (A posteriori error bound for $\|\bar{\mathbf{x}}_* - \mathbf{x}_*\|$). *If Assumption A2 holds, then*

$$(5.10) \quad \|\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq \frac{1}{1 - L_{\bar{\mathbf{h}}}} \|\mathbf{x}_* - \bar{\mathbf{h}}(\mathbf{x}_*)\|.$$

Proof. We subtract Eq. (5.1) from (5.6) and add and subtract $\bar{\mathbf{h}}(\mathbf{x}_*)$ to obtain

$$(5.11) \quad \bar{\mathbf{x}}_* - \mathbf{x}_* = \bar{\mathbf{h}}(\bar{\mathbf{x}}_*) - \bar{\mathbf{h}}(\mathbf{x}_*) + \bar{\mathbf{h}}(\mathbf{x}_*) - \mathbf{h}(\mathbf{x}_*).$$

As before, applying the triangle inequality and Lipschitz continuity of $\bar{\mathbf{h}}$ gives

$$(5.12) \quad \|\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq L_{\bar{\mathbf{h}}} \|\bar{\mathbf{x}}_* - \mathbf{x}_*\| + \|\bar{\mathbf{h}}(\mathbf{x}_*) - \mathbf{h}(\mathbf{x}_*)\|.$$

Now, using $L_{\bar{\mathbf{h}}} < 1$ and $\mathbf{h}(\mathbf{x}_*) = \mathbf{x}_*$ (from Eq. (5.1)) yields the desired result. ■

We can interpret this result similarly to Proposition 5.2. In particular, because the mapping $\mathbf{x} \mapsto \mathbf{x} - \bar{\mathbf{h}}(\mathbf{x})$ corresponds to the residual of the truth fixed-point system (5.6) and the computed solution \mathbf{x}_* appears in the bound, inequality (5.10) can be considered a residual-based *a posteriori* error bound. The right-hand side norm corresponds to the difference between the computed solution \mathbf{x}_* and the result of applying one iteration with the truth fixed-point operator $\bar{\mathbf{h}}$ to the computed solution \mathbf{x}_* . Thus, the right-hand side can be practically computed (resp. bounded from above) if the Lipschitz constant can be computed (resp. bounded from above) by applying the truth fixed-point operator to the computed solution.

5.2. In-plane error analysis. We now consider the particular case where the uncertainty-propagation operator \mathbf{f} restricts solutions to lie in a subspace of the space considered by the truth uncertainty-propagation operator $\bar{\mathbf{f}}$. Then, we perform analysis that quantifies the difference between the computed solution \mathbf{x}_* and the orthogonal projection of the truth solution $\bar{\mathbf{x}}_*$ onto the considered subspace. This is often referred to as the “in-plane error”, as it represents how close the computed solution is to the orthogonal projection of the truth solution onto the subspace (i.e., “plane”) of interest.

We begin by introducing the following assumption, which will be employed in all results within Section 5.2:

A3 The component uncertainty-propagation operators satisfy $\text{Im}(\mathbf{f}_i) \subseteq \mathbb{X}_i$, while the truth component uncertainty-propagation operators satisfy $\text{Im}(\bar{\mathbf{f}}_i) \subseteq \bar{\mathbb{X}}_i$ with \mathbb{X}_i and $\bar{\mathbb{X}}_i$ linear subspaces of $\mathbb{V}^{n_{\mathbf{x},i}}$ satisfying $\mathbb{X}_i \subseteq \bar{\mathbb{X}}_i \subseteq \mathbb{V}^{n_{\mathbf{x},i}}$ for $i = 1, \dots, n$.

We note that under Assumption A3, it follows trivially that the full-system uncertainty-propagation operator satisfies $\text{Im}(\mathbf{f}) \subseteq \mathbb{X} \equiv \mathbb{X}_1 \times \dots \times \mathbb{X}_n$, while the truth full-system uncertainty-propagation operator satisfies $\text{Im}(\bar{\mathbf{f}}) \subseteq \bar{\mathbb{X}} \equiv \bar{\mathbb{X}}_1 \times \dots \times \bar{\mathbb{X}}_n$ with $\text{Im}(\mathbf{f})$ and $\text{Im}(\bar{\mathbf{f}})$ linear subspaces of $\mathbb{V}^{n_{\mathbf{x}}}$ satisfying $\mathbb{X} \subseteq \bar{\mathbb{X}} \subseteq \mathbb{V}^{n_{\mathbf{x}}}$.

Remark 5.4 (Polynomial-chaos expansions satisfy Assumption A3). Assumption A3 holds if \mathbf{f}_i , $i = 1, \dots, n$ associate with computing low-order polynomial-chaos outputs and $\bar{\mathbf{f}}_i$, $i = 1, \dots, n$ associate with computing high-order polynomial-chaos outputs. In this case, $\mathbb{X}_i = \{\sum_{j \in \mathcal{J}_{\text{low}}} \psi_j(\xi) \mathbf{x}_{i,j} \mid \mathbf{x}_{i,j} \in \mathbb{R}^{n_{\mathbf{x},i}}, \mathbf{j} \in \mathcal{J}_{\text{low}}\} \subseteq \mathbb{V}^{n_{\mathbf{x},i}}$, $i = 1, \dots, n$ with $\mathcal{J}_{\text{low}} := \{\mathbf{j} \in \mathbb{N}_0^{n_{\xi}} \mid \|\mathbf{j}\|_1 \leq p_{\text{low}}\}$, while $\bar{\mathbb{X}}_i = \{\sum_{j \in \mathcal{J}_{\text{high}}} \psi_j(\xi) \mathbf{x}_{i,j} \mid \mathbf{x}_{i,j} \in \mathbb{R}^{n_{\bar{\mathbf{x}},i}}, \mathbf{j} \in \mathcal{J}_{\text{high}}\} \subseteq \mathbb{V}^{n_{\bar{\mathbf{x}},i}}$, $i = 1, \dots, n$ with $\mathcal{J}_{\text{high}} := \{\mathbf{j} \in \mathbb{N}_0^{n_{\xi}} \mid \|\mathbf{j}\|_1 \leq p_{\text{high}}\}$, and $p_{\text{high}} \geq p_{\text{low}}$. The conditions of Assumption A3 can be trivially verified in this case.

Under Assumption A3, the orthogonal projector \mathbf{P}_i onto \mathbb{X}_i , $i = 1, \dots, n$ is a linear operator and satisfies

$$(5.13) \quad \mathbf{P}_i \mathbf{x}_i = \arg \min_{\tilde{\mathbf{x}} \in \mathbb{X}_i} \|\mathbf{x}_i - \tilde{\mathbf{x}}\|,$$

while the orthogonal projector \mathbf{P} onto \mathbb{X} is a linear operator and satisfies

$$(5.14) \quad \mathbf{P} \mathbf{x} = \arg \min_{\tilde{\mathbf{x}} \in \mathbb{X}} \|\mathbf{x} - \tilde{\mathbf{x}}\|$$

with $\mathbf{P} \equiv \sum_{i=1}^n [\mathbf{E}_{p_i}^{\mathbf{x}}]^T \mathbf{P}_i \mathbf{E}_{p_i}^{\mathbf{x}}$. In this case, the error can be decomposed into orthogonal *in-plane* and *out-of-plane* components

$$(5.15) \quad \bar{\mathbf{x}}_* - \mathbf{x}_* = \underbrace{\bar{\mathbf{x}}_* - \mathbf{P} \bar{\mathbf{x}}_*}_{\text{out-of-plane error}} + \underbrace{\mathbf{P} \bar{\mathbf{x}}_* - \mathbf{x}_*}_{\text{in-plane error}}$$

with $\|\bar{\mathbf{x}}_* - \mathbf{x}_*\|^2 = \|\bar{\mathbf{x}}_* - \mathbf{P} \bar{\mathbf{x}}_*\|^2 + \|\mathbf{P} \bar{\mathbf{x}}_* - \mathbf{x}_*\|^2$; the out-of-plane error satisfies $\bar{\mathbf{x}}_* - \mathbf{P} \bar{\mathbf{x}}_* \in \mathbb{X}^\perp$, where \mathbb{X}^\perp denotes the orthogonal complement of \mathbb{X} in $\mathbb{V}^{n_{\mathbf{x}}}$; and the in-plane error satisfies $\mathbf{P} \bar{\mathbf{x}}_* - \mathbf{x}_* \in \mathbb{X}$. Because the out-of-plane error is determined completely from the truth solution $\bar{\mathbf{x}}_*$ and the subspace \mathbb{X} , the in-plane error is the error component that is informative of the accuracy of the computed solution \mathbf{x}_* given the truth solution $\bar{\mathbf{x}}_*$ and the considered subspace \mathbb{X} .

Before stating error bounds, we first derive conditions under which the in-plane error is zero, which implies that the computed solution satisfies $\mathbf{x}_* = \mathbf{P}\bar{\mathbf{x}}_*$ and thus can be considered the optimal approximation of $\bar{\mathbf{x}}_*$ in \mathbb{X} . This result relies on the following assumptions:

A4 Assumption A3 holds and the component uncertainty-propagation operators satisfy

$$(5.16) \quad \bar{\mathbf{f}}_i(\mathbf{E}_i^y \mathbf{I}_{\mathbf{x}}^y \mathbf{x}) = \mathbf{f}_i(\mathbf{E}_i^y \mathbf{I}_{\mathbf{x}}^y \mathbf{P}\mathbf{x}) + \bar{\mathbf{f}}_i^\perp(\mathbf{E}_i^y \mathbf{I}_{\mathbf{x}}^y \mathbf{x}), \quad \forall \mathbf{x} \in \mathbb{V}^{n_x}$$

for $i = 1, \dots, n$.

A5 There is a unique solution \mathbf{x}_* to the fixed-point system (2.8).

It is straightforward to show that Assumption A4 leads to a truth full-system uncertainty-propagation operator of the form

$$(5.17) \quad \bar{\mathbf{f}} : \mathbf{x} \mapsto \mathbf{f}(\mathbf{P}\mathbf{x}) + \bar{\mathbf{f}}^\perp(\mathbf{x}),$$

where

$$(5.18) \quad \bar{\mathbf{f}}^\perp : \mathbf{x} \mapsto \sum_{i=1}^n [\mathbf{E}_i^x]^T \bar{\mathbf{f}}_i^\perp(\mathbf{E}_i^y \mathbf{I}_{\mathbf{x}}^y \mathbf{x})$$

with $\bar{\mathbf{f}}^\perp : \mathbb{V}^{n_x} \rightarrow \mathbb{V}^{n_x}$ and $\text{Im}(\bar{\mathbf{f}}^\perp) \subseteq \mathbb{X}^\perp \equiv \mathbb{X}_1^\perp \times \dots \times \mathbb{X}_n^\perp$.

Note that Assumption A4 is equivalent to assuming the component uncertainty-propagation operators to satisfy $\mathbf{P}_i \bar{\mathbf{f}}_i(\mathbf{E}_i^y \mathbf{I}_{\mathbf{x}}^y \mathbf{x}) = \mathbf{f}_i(\mathbf{E}_i^y \mathbf{I}_{\mathbf{x}}^y \mathbf{P}\mathbf{x})$, $\forall \mathbf{x} \in \mathbb{V}^{n_x}$, which can be obtained by premultiplying Eq. (5.16) by \mathbf{P}_i .

Remark 5.5 (Polynomial-chaos expansions satisfy Assumption A4 for linear problems). Assumption A4 holds if \mathbf{f}_i , $i = 1, \dots, n$ are linear operators that project the outputs onto the space spanned by a low-order PCE basis and $\bar{\mathbf{f}}_i$, $i = 1, \dots, n$ are the same linear operators, but project the outputs onto the space spanned by a high-order PCE basis.

To demonstrate this, we employ the second moment as the norm-squared on the vector space \mathbb{V} , i.e., $\|\mathbf{w}\| := \sqrt{\mathbb{E}[\mathbf{w}^2]} \equiv \sqrt{\int_{\Omega} \mathbf{w}^2 dP(\theta)}$. Because any finite-variance random variable $\mathbf{w} \in \mathbb{V}$ can be represented as a convergent PCE expansion, we can express such random variables as $\mathbf{w} = \sum_{j \in \mathbb{N}_0^{n_\xi}} \psi_j(\xi) w_j$ with $w_j = \mathbb{E}[\mathbf{w}\psi_j]/\|\psi_j\|^2$, in which case the second moment is merely the weighted Euclidean norm of the PCE coefficients, i.e., $\|\mathbf{w}\| = \sqrt{\sum_{j \in \mathbb{N}_0^{n_\xi}} w_j^2 \|\psi_j\|^2}$.

In this case, the output random variables associated with the uncertainty-propagation operators \mathbf{f}_i and $\bar{\mathbf{f}}_i$ are

$$(5.19) \quad \mathbf{x}_i = \sum_{j \in \mathcal{J}_{\text{low}}} \psi_j(\xi) \mathbf{x}_{i,j} \quad \text{and} \quad \bar{\mathbf{x}}_i = \sum_{j \in \mathcal{J}_{\text{high}}} \psi_j(\xi) \bar{\mathbf{x}}_{i,j},$$

respectively, for $i = 1, \dots, n$, with \mathcal{J}_{low} and $\mathcal{J}_{\text{high}}$ defined in Remark 5.4. Note in particular that we assume the same total-degree truncation is employed for the output random variables of all components. Given the choice the norm, any vector of finite-variance random variables $\mathbf{w} \in \mathbb{V}^{n_{\mathbf{x},i}}$ can be expressed as $\mathbf{w} = \sum_{j \in \mathbb{N}_0^{n_\xi}} \psi_j(\xi) \mathbf{w}_j$ with its projection satisfying

$$(5.20) \quad \mathbf{P}_i \mathbf{w} = \sum_{j \in \mathcal{J}_{\text{low}}} \psi_j(\xi) \mathbf{w}_j, \quad \bar{\mathbf{P}}_i \mathbf{w} = \sum_{j \in \mathcal{J}_{\text{high}}} \psi_j(\xi) \mathbf{w}_j$$

for $i = 1, \dots, n$, where we have introduced the orthogonal projector $\bar{\mathbf{P}}_i$ onto $\bar{\mathbb{X}}_i$, $i = 1, \dots, n$ as the linear operator satisfying

$$(5.21) \quad \bar{\mathbf{P}}_i \mathbf{x}_i = \arg \min_{\tilde{\mathbf{x}} \in \bar{\mathbb{X}}_i} \|\mathbf{x}_i - \tilde{\mathbf{x}}\|.$$

Thus, in this case, assuming the endogenous-input random variables \mathbf{y}_i have finite variance such that $\mathbf{y}_i = \sum_{j \in \mathbb{N}_0^{n_\xi}} \psi_j(\xi) \mathbf{y}_{i,j}$, we have

$$(5.22) \quad \mathbf{f}_i(\mathbf{y}_i) = \mathbf{P}_i \mathbf{A}_i \mathbf{y}_i = \mathbf{P}_i \sum_{j \in \mathbb{N}_0^{n_\xi}} \psi_j(\xi) \mathbf{A}_i \mathbf{y}_{i,j} = \sum_{j \in \mathcal{J}_{\text{low}}} \psi_j(\xi) \mathbf{A}_i \mathbf{y}_{i,j} = \mathbf{f}_i(\mathbf{E}_i^{\mathbf{y}} \mathbf{l}_{\mathbf{x}}^{\mathbf{y}} \mathbf{P}_{\mathbf{x}})$$

$$\begin{aligned} \bar{\mathbf{f}}_i(\mathbf{y}_i) &= \bar{\mathbf{P}}_i \mathbf{A}_i \mathbf{y}_i = \bar{\mathbf{P}}_i \sum_{j \in \mathbb{N}_0^{n_\xi}} \psi_j(\xi) \mathbf{A}_i \mathbf{y}_{i,j} = \sum_{j \in \mathcal{J}_{\text{high}}} \psi_j(\xi) \mathbf{A}_i \mathbf{y}_{i,j} \\ (5.23) \quad &= \underbrace{\sum_{j \in \mathcal{J}_{\text{low}}} \psi_j(\xi) \mathbf{A}_i \mathbf{y}_{i,j}}_{\mathbf{f}_i(\mathbf{E}_i^{\mathbf{y}} \mathbf{l}_{\mathbf{x}}^{\mathbf{y}} \mathbf{P}_{\mathbf{x}})} + \underbrace{\sum_{j \in \mathcal{J}_{\text{high}} \setminus \mathcal{J}_{\text{low}}} \psi_j(\xi) \mathbf{A}_i \mathbf{y}_{i,j}}_{\bar{\mathbf{f}}_i^{\perp}(\mathbf{y}_i)}, \end{aligned}$$

where $\mathbf{A}_i \in \mathbb{R}^{n_{\mathbf{x},i} \times n_{\mathbf{y},i}}$, $i = 1, \dots, n$ denote matrices defining the linear uncertainty-propagation operators.

This decoupling implies that higher-order terms of \mathbf{x}_i do not affect lower-order terms of $\bar{\mathbf{f}}_i(\mathbf{x}_i)$. We note that this is similar to [12, Theorem 1].

We now derive conditions under which the in-plane error is zero.

Proposition 5.6 (Conditions for zero in-plane error). *If Assumptions A4 and A5 hold, then*

$$(5.24) \quad \mathbf{x}_* = \mathbf{P}\bar{\mathbf{x}}_*,$$

and thus the in-plane error is zero.

Proof. Substituting (5.17), which derives from Assumption A4, in the fixed-point system (5.3) yields

$$(5.25) \quad \bar{\mathbf{x}}_* = \mathbf{f}(\mathbf{P}\bar{\mathbf{x}}_*) + \bar{\mathbf{f}}^{\perp}(\bar{\mathbf{x}}_*).$$

Applying the (linear) operator \mathbf{P} to both sides of Eq. (5.25) yields

$$(5.26) \quad \mathbf{P}\bar{\mathbf{x}}_* = \mathbf{f}(\mathbf{P}\bar{\mathbf{x}}_*),$$

where we have used $\text{Im}(\bar{\mathbf{f}}^{\perp}) \subseteq \mathbb{X}^{\perp}$. Eq. (5.26) shows that $\mathbf{P}\bar{\mathbf{x}}_*$ satisfies $\mathbf{r}(\mathbf{P}\bar{\mathbf{x}}_*) = \mathbf{0}$, and thus the desired result holds by Assumption A5. ■

We now introduce an *a priori* error bound, whose proof follows similar steps to that of Proposition 5.2.

Proposition 5.7 (A priori in-plane error bound). *If Assumptions A1 and A3 hold, then the in-plane error can be bounded as*

$$(5.27) \quad \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq \frac{1}{1 - L_{\mathbf{h}}} \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{h}(\mathbf{P}\bar{\mathbf{x}}_*)\|.$$

Proof. We subtract Eq. (5.1) from $\mathbf{P}(\bar{\mathbf{x}}_* - \bar{\mathbf{h}}(\bar{\mathbf{x}}_*)) = \mathbf{0}$ (which holds from Eq. (5.6) and linearity of \mathbf{P}) and add and subtract $\mathbf{h}(\mathbf{P}\bar{\mathbf{x}}_*)$ to obtain

$$(5.28) \quad \mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_* = \mathbf{P}\bar{\mathbf{h}}(\bar{\mathbf{x}}_*) - \mathbf{h}(\mathbf{P}\bar{\mathbf{x}}_*) + \mathbf{h}(\mathbf{P}\bar{\mathbf{x}}_*) - \mathbf{h}(\mathbf{x}_*).$$

Applying the triangle inequality and Lipschitz continuity of \mathbf{h} gives

$$(5.29) \quad \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq \|\mathbf{P}\bar{\mathbf{h}}(\bar{\mathbf{x}}_*) - \mathbf{h}(\mathbf{P}\bar{\mathbf{x}}_*)\| + L_{\mathbf{h}} \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_*\|.$$

Finally, using $L_{\mathbf{h}} < 1$ and $\bar{\mathbf{h}}(\bar{\mathbf{x}}_*) = \bar{\mathbf{x}}_*$ (from Eq. (5.6)) yields the desired result. ■

We note that this bound is identical to the bound in Proposition 5.2, with $\mathbf{P}\bar{\mathbf{x}}_*$ replacing $\bar{\mathbf{x}}_*$; this result was achievable through the introduction of Assumption A3.

We now introduce the following assumption, which will be used to derive an *a posteriori* bound:

A6 Assumption A3 holds and the projected truth fixed-point operator satisfies

$$(5.30) \quad \|\mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_1) - \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_2)\| \leq L_{\mathbf{P}\bar{\mathbf{h}}} \|\mathbf{P}\mathbf{x}_1 - \mathbf{P}\mathbf{x}_2\|, \quad \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{V}^{n_x}$$

with $L_{\mathbf{P}\bar{\mathbf{h}}} < 1$.

We now introduce the notion of an \mathbb{X}^\perp -invariant operator, which will be used to demonstrate conditions under which Assumption A6 holds.

Definition 5.8 (\mathbb{X}^\perp -invariant truth fixed-point operators). We deem a truth fixed-point operator $\bar{\mathbf{h}}$ to be \mathbb{X}^\perp -invariant if it satisfies

$$(5.31) \quad \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}) = \mathbf{c}, \quad \forall \mathbf{x} \in \mathbb{X}^\perp,$$

where $\mathbf{c} \in \mathbb{V}^{n_x}$ is a constant random vector.

Intuitively, \mathbb{X}^\perp -invariant truth fixed-point operators preserve the decomposition of \mathbb{V}^{n_x} into \mathbb{X} and \mathbb{X}^\perp , as the output components of these operators in the space \mathbb{X} are unaffected by input components in the space \mathbb{X}^\perp . \mathbb{X}^\perp -invariance is a more general notion than that introduced in Assumption A4. We now demonstrate that this type of operator is necessary to satisfy Assumptions A4 and A6.

Proposition 5.9 (\mathbb{X}^\perp -invariance is a necessary condition for Assumptions A4 and A6). If (1) Assumption A4 holds and either Jacobi or Gauss–Seidel iteration defines the truth fixed-point operator $\bar{\mathbf{h}}$, or (2) Assumption A6 holds, then $\bar{\mathbf{h}}$ is an \mathbb{X}^\perp -invariant operator.

Proof. **Case 1.** If Assumption A4 holds and Jacobi iteration is applied, then from definition (4.2), the truth fixed-point operator becomes

$$(5.32) \quad \bar{\mathbf{h}} : \mathbf{x} \mapsto \underbrace{\omega \mathbf{f}(\mathbf{P}\mathbf{x}) + (1 - \omega)\mathbf{P}\mathbf{x}}_{\in \mathbb{X}} + \underbrace{\omega \bar{\mathbf{f}}^\perp(\mathbf{x}) + (1 - \omega)\mathbf{P}^\perp \mathbf{x}}_{\in \mathbb{X}^\perp},$$

where $\mathbf{P}^\perp = \mathbf{I} - \mathbf{P}$, from which condition (5.31) can be trivially verified with $\mathbf{c} = \omega \mathbf{f}(\mathbf{0})$.

Alternatively, if Assumption A4 holds and Gauss–Seidel iteration is applied, then from definition (4.3), the truth fixed-point operator becomes

$$(5.33) \quad \begin{aligned} \mathbf{E}_{p_i}^\mathbf{x} \bar{\mathbf{h}} : \mathbf{x} \mapsto & \underbrace{\omega \mathbf{f}_i \left(\mathbf{E}_i^\mathbf{y} \mathbf{I}_\mathbf{x}^\mathbf{y} \sum_{j=1}^{i-1} [\mathbf{E}_{p_j}^\mathbf{x}]^T \mathbf{P}_{p_j} \mathbf{E}_{p_j}^\mathbf{x} \bar{\mathbf{h}}(\mathbf{x}) + \mathbf{E}_i^\mathbf{y} \mathbf{I}_\mathbf{x}^\mathbf{y} \sum_{j=i+1}^n [\mathbf{E}_{p_j}^\mathbf{x}]^T \mathbf{P}_{p_j} \mathbf{E}_{p_j}^\mathbf{x} \mathbf{x} \right)}_{\in \mathbb{X}_i} \\ & + (1 - \omega) \mathbf{P}_{p_i} \mathbf{E}_{p_i}^\mathbf{x} \mathbf{x} \\ & + \underbrace{\omega \bar{\mathbf{f}}_i^\perp \left(\mathbf{E}_i^\mathbf{y} \mathbf{I}_\mathbf{x}^\mathbf{y} \sum_{j=1}^{i-1} [\mathbf{E}_{p_j}^\mathbf{x}]^T \mathbf{E}_{p_j}^\mathbf{x} \bar{\mathbf{h}}(\mathbf{x}) + \mathbf{E}_i^\mathbf{y} \mathbf{I}_\mathbf{x}^\mathbf{y} \sum_{j=i+1}^n [\mathbf{E}_{p_j}^\mathbf{x}]^T \mathbf{E}_{p_j}^\mathbf{x} \mathbf{x} \right)}_{\in \mathbb{X}_i^\perp} \\ & + (1 - \omega) \mathbf{P}_{p_i}^\perp \mathbf{E}_{p_i}^\mathbf{x} \mathbf{x} \end{aligned}$$

for $i = 1, \dots, n$, where $\mathbf{P}_i^\perp = \mathbf{I} - \mathbf{P}_i$ and $\mathbf{P}^\perp \equiv \mathbf{P}_1^\perp \times \dots \times \mathbf{P}_n^\perp$, from which condition (5.31) can be verified by induction with

$$(5.34) \quad \mathbf{E}_{p_i}^\mathbf{x} \mathbf{c} = \omega \mathbf{f}_i \left(\mathbf{E}_i^\mathbf{y} \mathbf{I}_\mathbf{x}^\mathbf{y} \mathbf{P} \sum_{j=1}^{i-1} [\mathbf{E}_{p_j}^\mathbf{x}]^T \mathbf{E}_{p_j}^\mathbf{x} \mathbf{c} \right)$$

and recalling $\mathbf{P} \equiv \sum_{i=1}^n [\mathbf{E}_{p_i}^\mathbf{x}]^T \mathbf{P}_i \mathbf{E}_{p_i}^\mathbf{x}$.

Case 2. We provide a proof by contradiction. Given any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}^\perp \subseteq \mathbb{V}^{n_x}$, we have $\mathbf{P}\mathbf{x}_1 = \mathbf{P}\mathbf{x}_2 = \mathbf{0}$ such that the right-hand-side of inequality (5.30) is zero. If Eq. (5.31) does not hold, then there will exist some $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}^\perp$ for which $\|\mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_1) - \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_2)\| > 0$, which violates the inequality. ■

Recall that Remark 5.5 demonstrated that Assumption A4 holds if $\bar{\mathbf{f}}$ is a linear operator associated with a polynomial-chaos expansion, and so these conditions also imply \mathbb{X}^\perp -invariance of the truth fixed-point operator $\bar{\mathbf{h}}$ associated with either Jacobi or Gauss–Seidel iteration.

We now derive an *a posteriori* bound on the error that leverages Assumption A6.

Proposition 5.10 (A posteriori in-plane error bound). *If Assumption A6 holds, then the in-plane error can be bounded as*

$$(5.35) \quad \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq \frac{1}{1 - L_{\mathbf{P}\bar{\mathbf{h}}}} \|\mathbf{x}_* - \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*)\|.$$

Proof. As in Proposition 5.7, we first subtract Eq. (5.1) from $\mathbf{P}(\bar{\mathbf{x}}_* - \bar{\mathbf{h}}(\bar{\mathbf{x}}_*)) = \mathbf{0}$; however, we add and subtract $\mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*)$ to obtain

$$(5.36) \quad \mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_* = \mathbf{P}\bar{\mathbf{h}}(\bar{\mathbf{x}}_*) - \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*) + \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*) - \mathbf{h}(\mathbf{x}_*).$$

Applying the triangle inequality, Assumption A6, and noting $\mathbf{P}\mathbf{x}_* = \mathbf{x}_*$ gives

$$(5.37) \quad \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_*\| \leq L_{\mathbf{P}\bar{\mathbf{h}}} \|\mathbf{P}\bar{\mathbf{x}}_* - \mathbf{x}_*\| + \|\mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*) - \mathbf{h}(\mathbf{x}_*)\|.$$

Finally, using $L_{\mathbf{P}\bar{\mathbf{h}}} < 1$ and $\mathbf{h}(\mathbf{x}_*) = \mathbf{x}_*$ yields the desired result. \blacksquare

Proposition 5.6 derived conditions for zero in-plane error by proving conditions under which $\mathbf{P}\bar{\mathbf{x}}_*$ is a fixed point of \mathbf{h} . From Proposition 5.10, we now derive conditions for zero in-plane error by considering the case where \mathbf{x}_* is a fixed point of $\mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*)$, which makes the right-hand side—and thus the left-hand side—of inequality (5.35) zero.

Proposition 5.11 (Additional conditions for zero in-plane error). *If Assumptions A4 and A6 hold, then*

$$(5.38) \quad \mathbf{x}_* = \mathbf{P}\bar{\mathbf{x}}_*,$$

and thus the in-plane error is zero.

Proof. Under Assumption A4, the truth fixed-point operator takes the form

$$(5.39) \quad \bar{\mathbf{h}} : \mathbf{x} \mapsto \mathbf{h}(\mathbf{P}\mathbf{x}) + \bar{\mathbf{h}}^\perp(\mathbf{x})$$

with $\text{Im}(\bar{\mathbf{h}}^\perp) \subseteq \mathbb{X}^\perp$, where

$$(5.40) \quad \bar{\mathbf{h}}^\perp : \mathbf{x} \mapsto \omega \bar{\mathbf{f}}_i^\perp(\mathbf{x}) + (1 - \omega) \mathbf{P}^\perp \mathbf{x}$$

in the case of Jacobi iteration, while

$$(5.41) \quad \bar{\mathbf{h}}^\perp : \mathbf{x} \mapsto \omega \bar{\mathbf{f}}_i^\perp \left(\mathbf{E}_i^y \mathbf{I}_x^y \sum_{j=1}^{i-1} [\mathbf{E}_{p_j}^x]^T \mathbf{E}_{p_j}^x \bar{\mathbf{h}}(\mathbf{x}) + \mathbf{E}_i^y \mathbf{I}_x^y \sum_{j=i+1}^n [\mathbf{E}_{p_j}^x]^T \mathbf{E}_{p_j}^x \mathbf{x} \right) + (1 - \omega) \mathbf{P}_{p_i}^\perp \mathbf{E}_{p_i}^x \mathbf{x}$$

in the case of Gauss–Seidel iteration. Applying projection \mathbf{P} to (5.1) and adding $\mathbf{P}\bar{\mathbf{h}}^\perp(\mathbf{x}_*)(= \mathbf{0})$ yields

$$(5.42) \quad \mathbf{P}\mathbf{h}(\mathbf{x}_*) + \mathbf{P}\bar{\mathbf{h}}^\perp(\mathbf{x}_*) - \mathbf{P}\mathbf{x}_* = \mathbf{0}.$$

Then, using (5.39) with $\mathbf{x}_* = \mathbf{P}\bar{\mathbf{x}}_*$ yields

$$(5.43) \quad \mathbf{P}\bar{\mathbf{h}}(\mathbf{x}_*) - \mathbf{x}_* = \mathbf{0}.$$

Thus, the right-hand side of inequality (5.35)—which is valid under Assumption A6—is zero, which yields the desired result. \blacksquare

6. Numerical experiments. We now assess the performance of the proposed NetUQ method on a two-dimensional stationary nonlinear diffusion equation with uncertainties arising in the diffusion coefficient and boundary conditions. Here, we construct the network formulation by decomposing the physical domain into (overlapping) subdomains, and associating each subdomain with a network component. Thus—for this problem—the NetUQ method is tantamount to an overlapping domain decomposition method for uncertainty propagation. We first define the global uncertainty-propagation problem in Section 6.1, and subsequently define the resulting component and network uncertainty-propagation problems in Section 6.2. Then, Sections 6.3 and 6.4 perform strong- and weak-scaling studies, respectively.

6.1. Global uncertainty-propagation problem. We express the deterministic boundary value problem (BVP) as the partial differential equation (adopted from Ref. [25])

$$(6.1) \quad -\frac{\partial^2 v}{\partial x_1 \partial x_2} + (e^{\mu v} - 1) = 10 \sin(2\pi x_1) \sin(2\pi x_2), \quad x \equiv (x_1, x_2) \in \Omega_x = (0, 1)^2 \\ v(x) = v_\Gamma, \quad x \in \partial\Omega_x.$$

To formulate the uncertainty-propagation problem, we assume the boundary condition and diffusion coefficient are “global” input random variables, which we model using polynomial-chaos expansions as

$$(6.2) \quad v_\Gamma = u_1^{\text{global}} = \sum_{j \in \mathcal{J}_{\text{global}}} \psi_j(\xi) u_{1,j}^{\text{global}}, \quad \mu = u_2^{\text{global}} = \sum_{j \in \mathcal{J}_{\text{global}}} \psi_j(\xi) u_{2,j}^{\text{global}}$$

with $n_\xi = 2$ germ random variables $\xi \equiv (\xi_1, \xi_2) \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Note that the PCE coefficients $u_{1,j}^{\text{global}}$ and $u_{2,j}^{\text{global}}$, $j \in \mathcal{J}_{\text{global}}$ completely characterize the global input random variables $u_1^{\text{global}} \in \mathbb{V}$ and $u_2^{\text{global}} \in \mathbb{V}$, respectively. We consider the case of independent global input random variables such that u_1^{global} and u_2^{global} are expressed strictly in terms of ξ_1 and ξ_2 , respectively. As described in Remark 2.1, we define the multi-index set via total-degree truncation as $\mathcal{J}_{\text{global}} = \{j \in \mathbb{N}_0^{n_\xi} \mid \|j\|_1 \leq p\}$ for $p = 3$ and employ Hermite polynomials such that

$$\begin{aligned} \psi_{(0,0)}(\xi) &= 1, \\ \psi_{(1,0)}(\xi) &= \xi_1, \quad \psi_{(0,1)}(\xi) = \xi_2, \\ \psi_{(2,0)}(\xi) &= \xi_1^2 - 1, \quad \psi_{(1,1)}(\xi) = \xi_1 \xi_2, \quad \psi_{(0,2)}(\xi) = \xi_2^2 - 1, \\ \psi_{(3,0)}(\xi) &= \xi_1^3 - 3\xi_1, \quad \psi_{(2,1)}(\xi) = \xi_2 (\xi_1^2 - 1), \quad \psi_{(1,2)}(\xi) = \xi_1 (\xi_2^2 - 1), \quad \psi_{(0,3)}(\xi) = \xi_2^3 - 3\xi_2. \end{aligned}$$

Table 6.1 reports the polynomial-chaos coefficients used to represent the two non-Gaussian, independent global input random variables. Note that while this characterization yields independent global input random variables, the NetUQ formulation supports dependent input random variables.

| i | $u_{i,(0,0)}^{\text{global}}$ | $u_{i,(1,0)}^{\text{global}}$ | $u_{i,(0,1)}^{\text{global}}$ | $u_{i,(2,0)}^{\text{global}}$ | $u_{i,(1,1)}^{\text{global}}$ | $u_{i,(0,2)}^{\text{global}}$ | $u_{i,(3,0)}^{\text{global}}$ | $u_{i,(2,1)}^{\text{global}}$ | $u_{i,(1,2)}^{\text{global}}$ | $u_{i,(0,3)}^{\text{global}}$ |
|-----|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|-------------------------------|
| 1 | 1.0 | 0.2 | 0.0 | 0.02 | 0.0 | 0.0 | 0.002 | 0.0 | 0.0 | 0.0 |
| 2 | 1.0 | 0.0 | 0.2 | 0.0 | 0.0 | 0.0 | 0.02 | 0.0 | 0.0 | 0.002 |

Table 6.1: Polynomial-chaos coefficients for global input random variables.

The global uncertainty-propagation problem is now: Given global input random variables u_1^{global} and u_2^{global} characterized using polynomial-chaos expansions (6.2) with coefficients provided by Table 6.1, compute the random field $v : \Omega_x \rightarrow \mathbb{V}$, where $v(x)$ is the random variable associated with

the variable $v(x)$ for $x \in \Omega_x$ due to the randomness in the diffusion coefficient and boundary condition. Solving this global uncertainty-propagation with non-intrusive spectral projection (NISP) [34, 28] using a 16-point Gauss–Hermite quadrature rule (derived using a full tensor product of the 1D rule with 4 points per dimension) yields a PCE representation for the random field. Each of the 16 quadrature points yields one instance of the global deterministic problem (6.1). We discretize this (nonlinear) PDE using the finite-element method (FEM) with rectangular linear elements, and we solve the resulting system of nonlinear equations using Newton’s method. The initial guess for the Newton solver is the zero solution.

For simplicity, we truncate the polynomial-chaos expansion of the random field at the same level as the global inputs such that

$$(6.3) \quad v(x) = \sum_{j \in \mathcal{J}_{\text{global}}} \psi_j(\xi) v_j^{\text{global}}(x), \quad x \in \Omega_x.$$

Figure 6.1 plots the resulting PCE coefficients $v_j^{\text{global}}(x)$, $j \in \mathcal{J}_{\text{global}}$, $x \in \Omega_x$.

6.2. Component and network uncertainty-propagation problems. We construct the network uncertainty-propagation problem by decomposing the global problem into overlapping subdomains $\{\Omega_x^i\}_{i=1}^n$ satisfying $\cup_{i=1}^n \Omega_x^i = \Omega_x$, and treating each subdomain as a component. Then, the component uncertainty-propagation problem is defined as uncertainty propagation performed on the associated subdomain. In particular, the deterministic BVP for the i th component is

$$(6.4) \quad \begin{aligned} -\frac{\partial^2 v^i}{\partial x_1 \partial x_2} + (e^{\mu v} - 1) &= 10 \sin(2\pi x_1) \sin(2\pi x_2), \quad x \in \Omega_x^i \subseteq \Omega_x \\ v^i(x) &= v_\Gamma, \quad x \in \partial\Omega_x^i \cap \partial\Omega_x, \quad v^i(x) = v^j(x), \quad x \in \partial\Omega_x^i \cap \Omega_x^j \setminus \partial\Omega_x^j, \quad j \in \mathcal{N} \end{aligned}$$

for $i = 1, \dots, n$, where $\mathcal{N} \subset \{1, \dots, n\}$ denotes the components neighboring the i th component.

The i th component uncertainty-propagation problem is now: Given global input random variables u_1^{global} and u_2^{global} characterized using polynomial-chaos expansions (6.2) with coefficients provided by Table 6.1, as well as random fields $v^j : \partial\Omega_x^i \cap \Omega_x^j \setminus \partial\Omega_x^j$ for $j \in \mathcal{N}$, where $v^j(x)$ is the random variable associated with the variable $v^j(x)$ on the boundary $x \in \partial\Omega_x^i \cap \Omega_x^j \setminus \partial\Omega_x^j$, compute the random field $v^i : \Omega_x^i \rightarrow \mathbb{V}$, where $v^i(x)$ is the random variable associated with the variable $v^i(x)$ for $x \in \Omega_x^i$.

As with the global uncertainty-propagation problem, we solve this component uncertainty-propagation with non-intrusive spectral projection (NISP) using a 16-point, 4-level Gauss–Hermite quadrature rule, which yields a PCE representation for the random field. Each of the 16 quadrature points yields one instance of the component deterministic problem (6.4). We use the same spatial discretization as the global problem, and we solve the resulting system of nonlinear equations using Newton’s method. The initial guess for the Newton solver is the solution of the problem at the previous fixed-point iteration; at the first iteration, the initial guess is zero.

Again, we truncate the polynomial-chaos expansion of the i th component’s random field at the same level as the global inputs such that

$$(6.5) \quad v^i(x) = \sum_{j \in \mathcal{J}_{\text{global}}} \psi_j(\xi) v_j^i(x), \quad x \in \Omega_x^i.$$

Abstractly, we view the mapping from global input random variables u_1^{global} and u_2^{global} and neighbor field random variables $v^j(x)$, $x \in \partial\Omega_x^i \cap \Omega_x^j \setminus \partial\Omega_x^j$, $j \in \mathcal{N}$ to the field random variables $v^i(x)$, $x \in \partial\Omega_x^i \cap \Omega_x^j \setminus \partial\Omega_x^i$, $j \in \mathcal{N}$ computed using NISP in this manner as defining the component uncertainty-propagation problem defined in Section 2.1. In particular, the exogenous-input random variables correspond to the component global input random variables such that $\mathbf{u}_i = (u_1^{\text{global}}, u_2^{\text{global}})$, the endogenous-input random variables correspond to the neighbor field random variables such

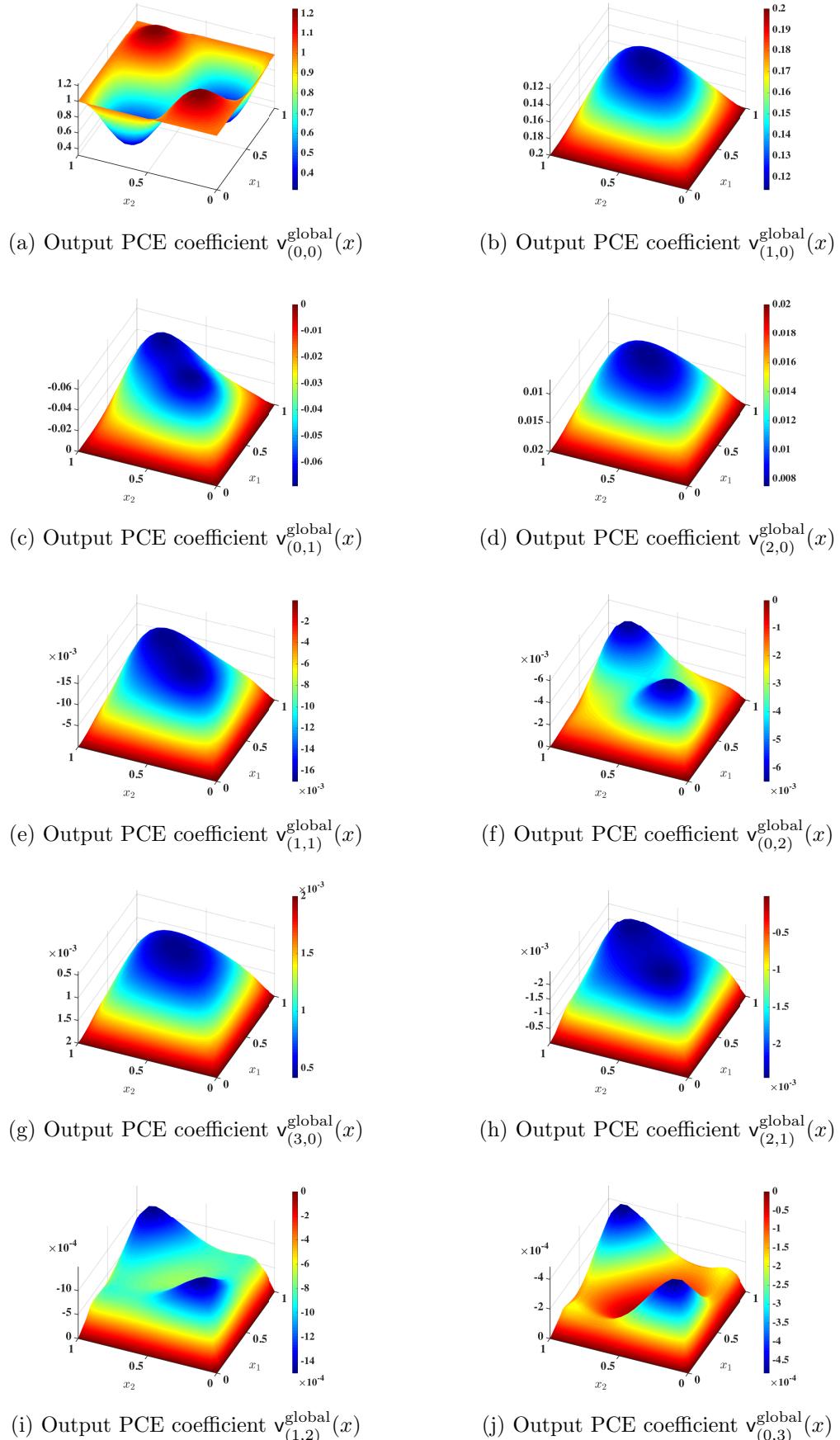


Figure 6.1: PCE coefficients for the random field $v(x)$ computed via NISP using a 16-point, 4-level Gauss–Hermite quadrature rule. 21

that $\mathbf{y}_i = (v^j(x), x \in \partial\Omega_x^i \cap \Omega_x^j \setminus \partial\Omega_x^j)_{j \in \mathcal{N}}$, and the output random variables correspond to the field random variables on the current subdomain that are used to define boundary conditions on neighboring subdomains such that $\mathbf{x}_i = (v^i(x), x \in \partial\Omega_x^i \cap \Omega_x^i \setminus \partial\Omega_x^i)_{j \in \mathcal{N}}$. Because we are employing a finite-element spatial discretization, all field variables are represented in the finite-element trial space such that the endogenous-input random variables \mathbf{y}_i and output random variables \mathbf{x}_i defined above are also finite dimensional.

As described in Remark 2.1, because we are employing PCE representations of random variables, we can equivalently express the component uncertainty-propagation problem in terms of the PCE coefficients themselves via Eq. (2.4). The associated network uncertainty-propagation problem in terms of PCE coefficients is derived analogously to (2.8) and is

$$(6.6) \quad \mathbf{r}(\mathbf{x}_*, \mathbf{u}) = \mathbf{0},$$

where $\mathbf{u} := [\mathbf{u}_1^T \cdots \mathbf{u}_n^T]^T \in \mathbb{R}^{n_u}$ and $\mathbf{x} := [\mathbf{x}_1^T \cdots \mathbf{x}_n^T]^T \in \mathbb{R}^{n_x}$ such that $n_u := \sum_{i=1}^n n_{u,i}$ and $n_x := \sum_{i=1}^n n_{x,i}$. We denote the value of the outputs that satisfies the fixed point problem by $\mathbf{x}_* \equiv \mathbf{x}_*(\mathbf{u}) \in \mathbb{R}^{n_x}$.

Figure 6.2 illustrates construction of the network uncertainty-propagation problem a decomposition involving 4 subdomains defined on a 2×2 grid. Note that we employ an overlap region spanning one element between neighboring subdomains. Analogously, Fig. 6.3 illustrates the network connectivity for a decomposition involving 16 subdomains defined on a 4×4 grid.

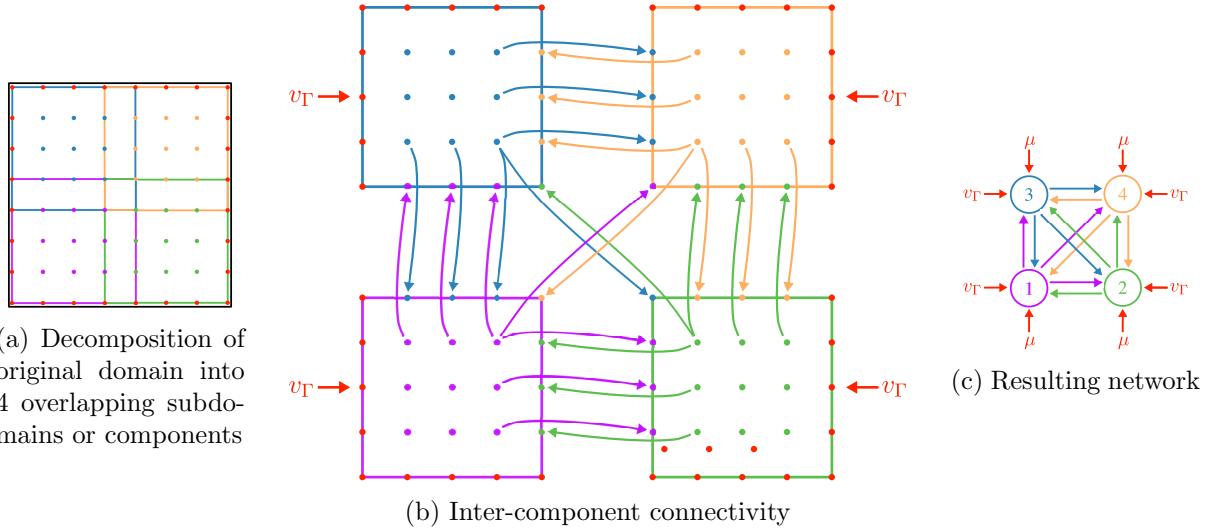


Figure 6.2: Network formulation for the 2×2 component case.

To solve the resulting network uncertainty-propagation problem, we investigate both the Jacobi and Gauss–Seidel NetUQ methods, with two values of the relaxation factor, with and without Anderson acceleration. In the case of Gauss–Seidel, we also assess the effect of different permutations \mathbf{p} on performance. The initial guess for the relaxation methods is the zero solution (i.e., all PCE coefficients for output random variables are zero). To assess convergence of the relaxation methods at iteration k , we compute the relative residual $r(\mathbf{x}^{(k)}, \mathbf{u})$, where

$$(6.7) \quad r : (\mathbf{x}, \mathbf{u}) \mapsto \|\mathbf{r}(\mathbf{x}, \mathbf{u})\|_2 / \|\mathbf{r}(\mathbf{0}, \mathbf{u})\|_2.$$

All timings are obtained by performing calculations on an Intel(R) Xeon(R) Core(TM) i7-5557U CPU @ 3.10GHz with 16 GB RAM. The NetUQ software is written in Matlab; it wraps around the Uncertainty Quantification Toolkit (UQTk) [18, 17], which performs component uncertainty propagation.

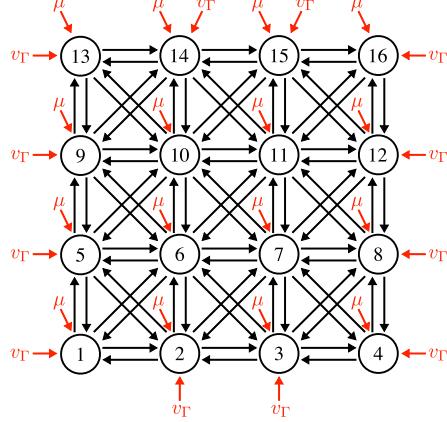
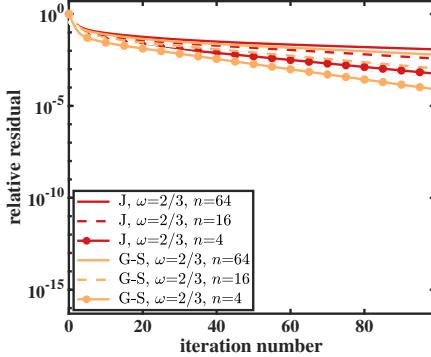


Figure 6.3: Network formulation 4×4 component case. Figure shows coupling between nodes for the network uncertainty-propagation problem, with black edges corresponding to endogenous-input random variables and red edges corresponding to exogenous-input random variables.

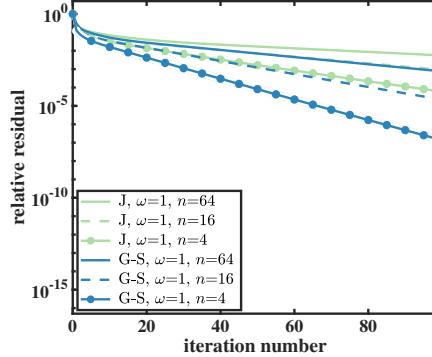
6.3. Strong-scaling study. Here we investigate the strong scaling performance of NetUQ. In the context of NetUQ, strong scaling associates with the case where a fixed system can be broken down into smaller and smaller components or assemblies on which uncertainty-propagation can be performed. To perform strong scaling, we fix the finite-element discretization of the global problem using 1681 nodes arising from a mesh characterized by a uniform 41×41 grid. Subsequently, we increase the number of subdomains used to define the network on this fixed global finite-element discretization, always employing an overlap region spanning one element, and always employing decompositions that are uniform in x_1 and x_2 . In particular, we consider 2×2 , 4×4 , and 8×8 decompositions. Note that the size of each component deterministic problem increases as the number of components decreases, and employing $n = 1$ corresponds to the global uncertainty-propagation problem described in Section 6.1.

Fig. 6.4 reports convergence results for this study, where we have employed a permutation \mathbf{p} for Gauss–Seidel that yields the minimum number of sequential steps per iteration of $n_{\text{seq}} = 4$. This figure elucidates several trends. First, we note that convergence is faster for smaller networks; this is sensible, as more iterations are required for information to propagate throughout the domain when the method uses a larger number of (smaller) subdomains. Indeed, in the limiting case of $n = 1$, the NetUQ formulation is equivalent to the global uncertainty-propagation problem, which—by definition—converges in a single iteration. Second, we note that Gauss–Seidel yields faster convergence than Jacobi when measured as a function of the number of iterations. As discussed in Section 3.2, this occurs because Gauss–Seidel employs more updated information within each iteration. However, because Gauss–Seidel employs more sequential steps per iteration than the Jacobi method, each Gauss–Seidel iteration incurs a larger wall time, and thus these results are insufficient for assessing which method yields the best parallel wall-time performance. Third, we observe that the classical iterations (i.e., employing a relaxation factor of $\omega = 1$) yield faster convergence than under-relaxation performed with a relaxation factor $\omega = 2/3$. However, over-relaxation (i.e., employing a relaxation factor of $\omega > 1$) sometimes resulted in divergence; thus, we have omitted these results (see Remark 4.2). Fourth, we note that employing Anderson acceleration yields substantially faster convergence.

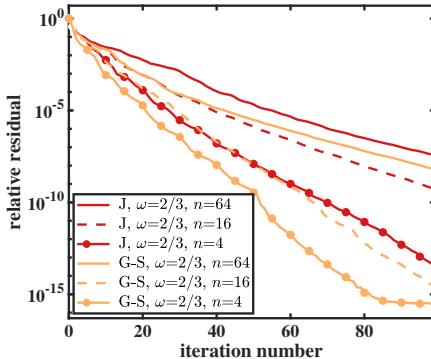
Next, Fig. 6.5 reports the dependence of several performance quantities as a function of the number of components. This figure reports results for NetUQ without Anderson acceleration, and for the case where iterations are terminated when the relative residual reaches a value of 10^{-3} . Consistent with Fig. 6.4, Fig. 6.5a shows that the lowest iteration count is achieved for smaller networks, Gauss–Seidel, and a relaxation factor of $\omega = 1$.



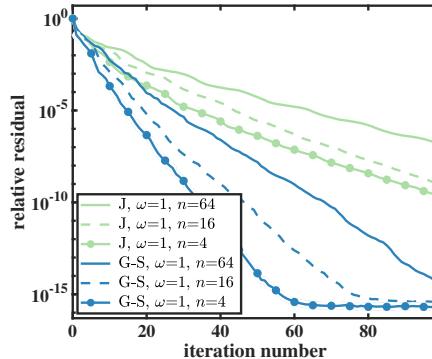
(a) $\omega = 2/3$; no Anderson acceleration



(b) $\omega = 1$; no Anderson acceleration



(c) $\omega = 2/3$; Anderson acceleration, $m = 5$



(d) $\omega = 1$; Anderson acceleration, $m = 5$

Figure 6.4: Strong-scaling study: convergence results. J and G-S labels refer to the Jacobi and Gauss–Seidel methods, respectively. Note that convergence is faster when NetUQ employs smaller networks, Gauss–Seidel (rather than Jacobi), $\omega = 1$ (rather than $\omega = 2/3$), and Anderson acceleration.

Fig. 6.5b reports the associated parallel wall times, employing a one-to-one component-to-processor map. These results show that the trends suggested by Fig. 6.5 can be reversed when accounting for the lower computational cost of solving the component deterministic problems (6.4) for smaller subdomains and the larger number of sequential steps per iteration for the Gauss–Seidel method. Specifically, this figure shows that the smallest parallel wall times are achieved for Jacobi iteration and larger network sizes. This suggests that the embarrassingly parallel nature of each Jacobi iteration outweighs the larger number of iterations it requires for convergence relative to Gauss–Seidel, and the lower computational cost of solving component deterministic problems (6.4) with smaller subdomains outweighs the larger number of iterations needed for convergence.

Next, Fig. 6.5c reports the speedup as a function of the number of components, where the speedup is defined as the ratio of serial execution time to parallel execution time. Here, we consider the serial execution time to be the wall time required to solve the global uncertainty-propagation problem described in Section 6.1 using one computing core. We observe that—for four components—the relatively large number of iterations required for convergence outweighs gains achieved through parallelization, resulting in a speedup less than one. Speedups greater than one are achieved only for $n = 16$ in the case of Jacobi with $\omega = 1$ and for $n = 64$ in the case of Jacobi and Gauss–Seidel with $\omega = 1$. However, we emphasize that the purpose of NetUQ is not necessarily to achieve speedups in this particular context, i.e., where solving the global uncertainty-propagation

problem is feasible. As described in the introduction, the true objective of NetUQ is to enable full-system UQ in scenarios where it would be otherwise impossible due to challenges in integrating component models, for example. Nonetheless, we show in the next figure that substantial speedups can indeed be achieved when Anderson acceleration is employed.

Next, Fig. 6.5d quantifies the error incurred by the network formulation. Recall from Section 5 that the network formulation can incur error if the uncertainty-propagation operator \mathbf{f} constitutes an approximation of an underlying “truth” operator $\bar{\mathbf{f}}$. This is precisely the case in this context, where the truth operator $\bar{\mathbf{f}}$ associates with performing uncertainty-propagation with infinite-dimensional PCE representations for output random variables and endogenous-input random variables; this “no edge truncation” case is mathematically equivalent to simply solving the global uncertainty-propagation problem described in Section 6.1. Thus, to assess these errors, we compute the relative error in the PCE coefficients of the field random variable at several spatial locations computed using the NetUQ approach (executed with 3rd-order total-degree truncation and satisfying a relative residual of 10^{-10}) with respect to the same PCE coefficients computed by solving the global uncertainty-propagation problem. This relative error is computed as the ℓ^2 -norm of the difference between these PCE coefficients, divided by the ℓ^2 -norm of the PCE coefficients computed via global uncertainty propagation. Fig. 6.5d shows that this error increases as the number of components increases; this is sensible, as a larger number of components implies more spatial locations where truncation is imposed. The figure also shows that this error is larger for variables located further from the domain boundary; this is also intuitive, as it suggests that errors grow as the variable of interest is located further from one of the driving exogenous inputs.

Fig. 6.6 repeats the same study, but with all relaxation methods using Anderson acceleration with $m = 5$. Note that we need not repeat the error-quantification study, as these results are related to the network formulation itself, and are independent of the relaxation method used to numerically solve the network uncertainty-propagation problem. Through a comparison with the corresponding figures in Fig. 6.5, it is clear that Anderson acceleration yields roughly an order-of-magnitude reduction in the number of iterations required for convergence. This in turn implies substantially lower parallel wall times, as well as significantly larger speedups. Indeed, when Anderson acceleration is employed, the NetUQ method realizes speedups as high as nearly 20 in the case of $n = 16$ components and the Jacobi method. One noteworthy observation is that the maximum speedup corresponds to the 16-component network. Since this problem is relatively small (i.e., an FEM discretization with 1681 nodes), decomposing the domain into smaller domains beyond a certain threshold (16 subdomains in this case) does not sufficiently reduce the time required for each component uncertainty propagation to overcome the increase in iteration count for the algorithm to converge. We also note that Anderson acceleration has the effect of reducing the performance discrepancy obtained using relaxation factors of $\omega = 2/3$ and $\omega = 1$, especially for the Jacobi method.

Finally, to assess the effect of the permutation \mathbf{p} on the performance of the Gauss–Seidel method, Fig. 6.7 reports strong-scaling results for Gauss–Seidel with $\omega = 1$, Anderson acceleration, and ten random permutations \mathbf{p} . Recall from Section 3.2 that the permutation effectively defines the DAG on which Gauss–Seidel propagates uncertainties in a feed-forward manner within each iteration. Figure 6.7a shows that—for this problem—the choice of permutation has essentially no effect on the number of iterations required for convergence. However, Figure 6.7b shows that the choice of permutation does have a substantial effect on the number of sequential steps per iteration n_{seq} as computed using Algorithm 3.3, which in turn yields significant differences in parallel wall time (Fig. 6.7c) and speedup (Fig. 6.7d). Thus, for this problem, the permutation should be chosen to minimize the number of sequential steps per iteration n_{seq} . Note that virtually no speedup is observed even when using Anderson acceleration for the worst performing permutation. Thus further emphasizes the need to carefully consider the choice of permutation in practice.

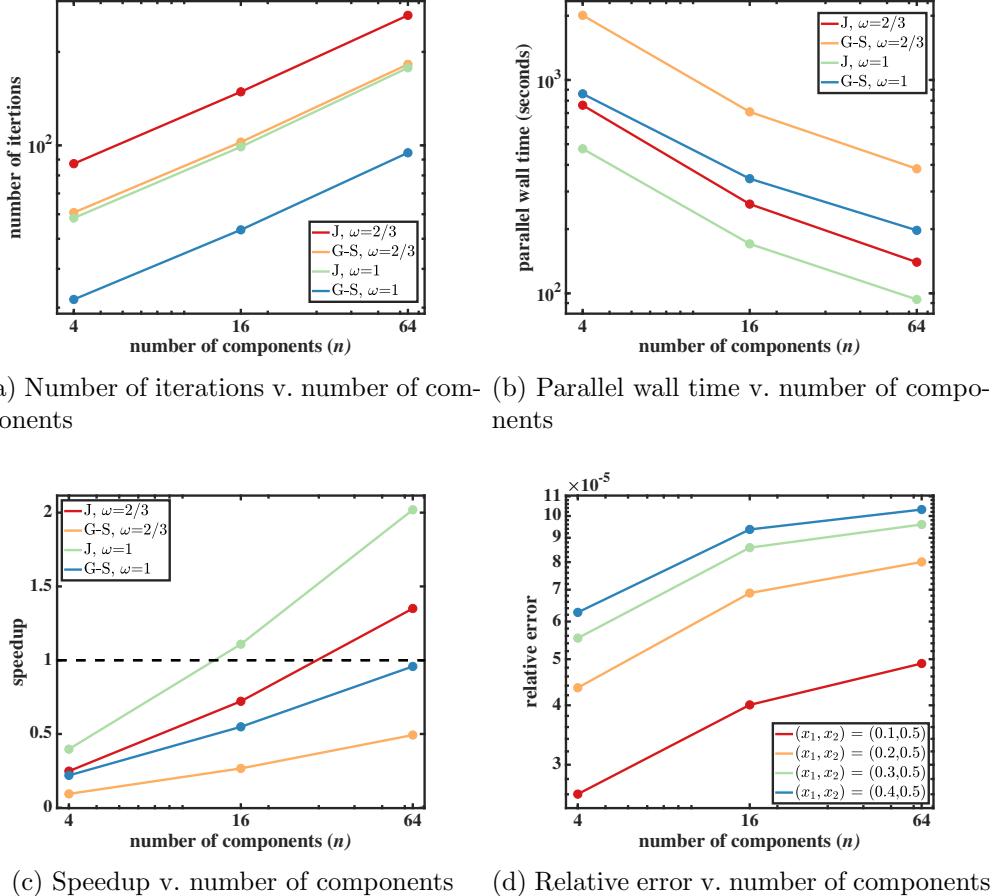


Figure 6.5: Strong-scaling study without Anderson acceleration. Iterations are terminated when the relative residual reaches a value of 10^{-3} . J and G-S labels refer to the Jacobi and Gauss–Seidel methods, respectively. Relative error refers to the error in the PCE coefficients of the field random variable at several spatial locations computed using the NetUQ approach (executed with 3rd-order total-degree truncation and satisfying a relative residual of 10^{-10}) with respect to the same PCE coefficients computed by solving the global uncertainty-propagation problem.

6.4. Weak-scaling study. We now investigate the weak-scaling performance of NetUQ. In the context of NetUQ, weak scaling associates with the case where a fixed library of components on which uncertainty propagation can be performed is assembled into larger and larger full systems. In particular, we adopt the same strategy for constructing networks as in Section 6.3, with one exception: each component deterministic BVP is discretized using a mesh composed of 36 grid points, regardless of the total number of components. As such, the size of each component deterministic problem remains fixed for all networks. However, as the number of components increases, the size of the overlap region between components decreases, and size of the deterministic global problem increases. We note that the weak-scaling case with $n = 64$ is equivalent to the strong-scaling case with $n = 64$, and thus the results for these cases are identical.

Fig. 6.8 reports convergence results, where—as in the strong-scaling case—we use a permutation \mathbf{p} for Gauss–Seidel that yields the minimum number of sequential steps per iteration of $n_{\text{seq}} = 4$. Comparing with Fig. 6.4, we see that—as in the strong-scaling case—convergence is faster for smaller networks, Gauss–Seidel, $\omega = 1$, and Anderson acceleration. The explanation behind these trends is the same as in the strong-scaling case. However, we observe that convergence is

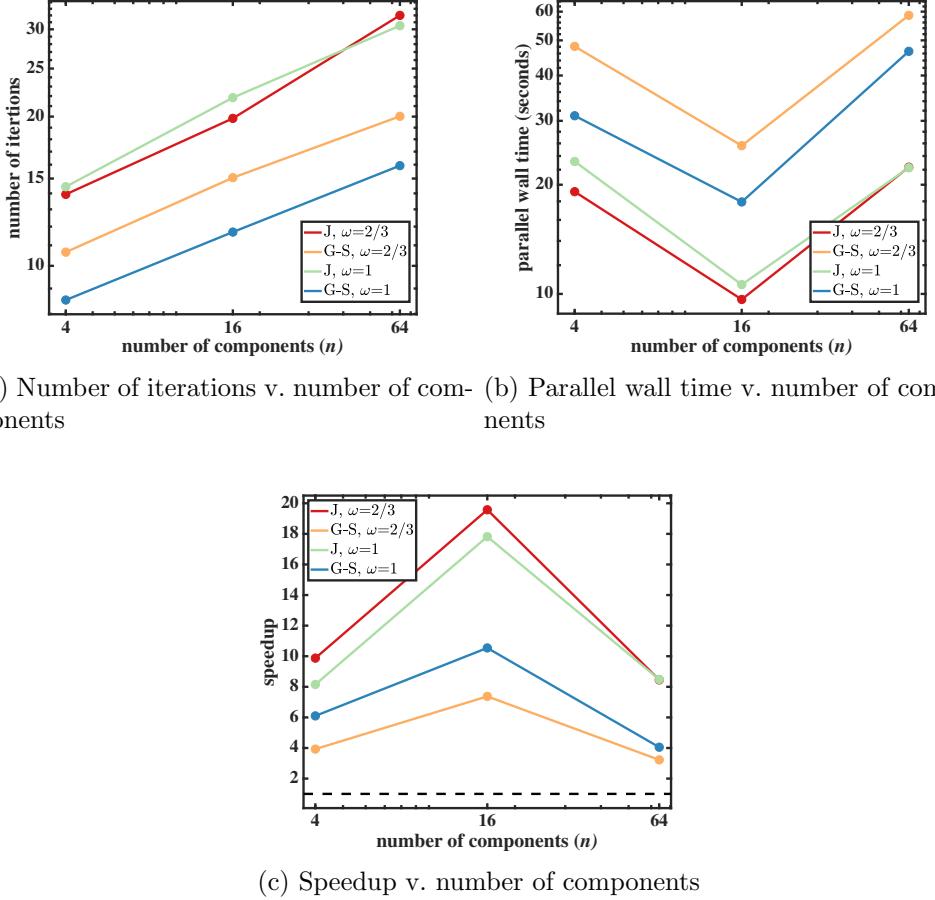
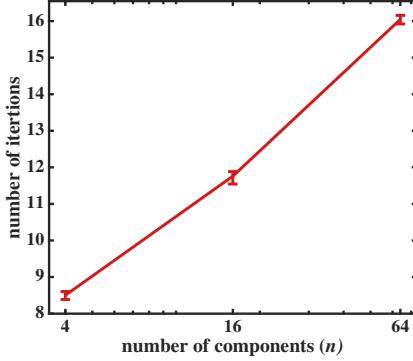


Figure 6.6: Strong-scaling study using Anderson acceleration with $m = 5$. Iterations are terminated when the relative residual reaches a value of 10^{-3} . J and G-S labels refer to the Jacobi and Gauss–Seidel methods, respectively.

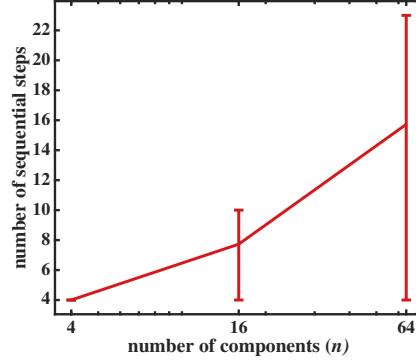
substantially faster (as a function of iteration number) for $n = 4$ and $n = 16$ in the weak-scaling case compared with the strong-scaling case. This is due to the fact that—for these network sizes—the weak-scaling problem corresponds to a larger amount of overlap between neighboring components than the strong-scaling problem. Increasing the amount of overlap is known to promote convergence in domain decomposition, which is an effect we observe here.

Next, Fig. 6.9 reports performance as a function of the number of components. As in Fig. 6.5, results correspond to NetUQ without Anderson acceleration, and for the case where iterations are terminated when the relative residual reaches a value of 10^{-3} . Comparing Figs. 6.5a and 6.9a shows that—in both cases—the number of iterations increases as the number of components increases. This is due to the fact that more iterations are needed for information to propagate throughout the domain when it is decomposed into more components. However, we can see that the weak-scaling case yields substantially fewer iterations than the strong-scaling case for $n = 4$ and $n = 16$; this occurs due to the increased amount of overlap in the weak-scaling case as previously discussed.

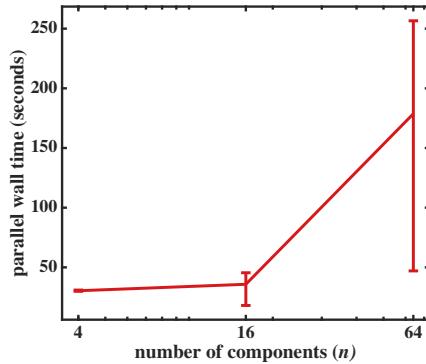
Fig. 6.9b reports wall times, again using a one-to-one component-to-processor map. Comparing with Fig. 6.5b shows that weak and strong scaling exhibit opposite trends. This is sensible because—unlike in strong scaling where the component deterministic problem decreases in size for larger networks—the deterministic component-problem size remains constant for all network sizes. Thus, the parallel wall time is driven purely by iteration count in the case of weak scaling. However, as



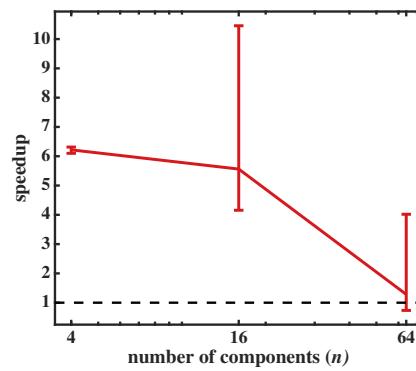
(a) Number of iterations v. number of components



(b) Number of sequential steps per iteration v. number of components



(c) Parallel wall time v. number of components



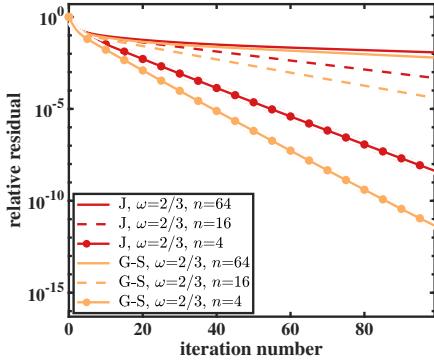
(d) Speedup v. number of components

Figure 6.7: Strong-scaling study using Anderson acceleration with $m = 5$. Gauss–Seidel with $\omega = 1$ performance for ten random permutations \mathbf{p} . Red curve represents the mean value, and vertical lines span the minimum and maximum values computed over these permutations. Note that the permutation has minimal effect on the number of iterations required for convergence, and thus performance is driven by the number of sequential steps per iteration n_{seq} that the permutation admits.

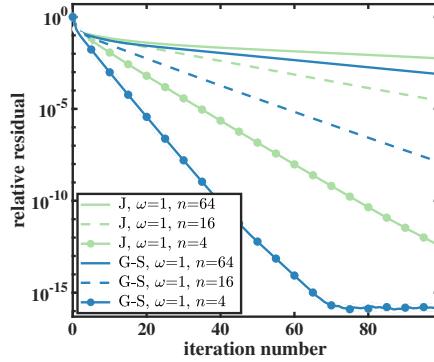
in the strong-scaling case, we again see that the Jacobi method yields superior parallel wall-time performance than the Gauss–Seidel method despite consuming more iterations; this is due to the embarrassingly parallel nature of Jacobi iterations.

Fig. 6.9c reports the speedup as a function of the number of components, where the speedup is defined as the ratio of serial execution time to parallel execution time, where the serial execution time corresponds to the cost of solving the global uncertainty-propagation problem using the same (global) mesh as that corresponding to the NetUQ problem. As with the strong-scaling case reported in Fig. 6.5c, we again see that modest speedups are achieved for Jacobi with the largest network size. While this might seem surprising given the trends shown in Fig. 6.9b, it can be explained by the fact that the global uncertainty-propagation problem incurs lower parallel wall time as the number of components decreases in this case. This also explains why the speedup increases at a slower rate with increasing network size as compared with the strong-scaling case.

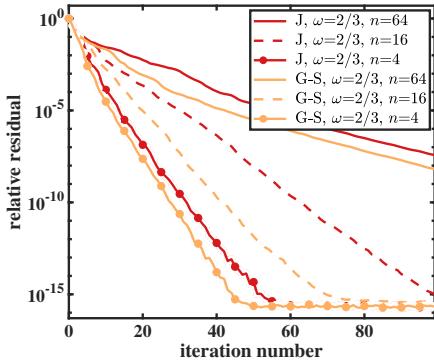
Finally, Fig. 6.9d elucidates the same trends as were observed in Fig. 6.5d for the strong-scaling case: the errors incurred by the NetUQ formulation are extremely small, and increase slightly as



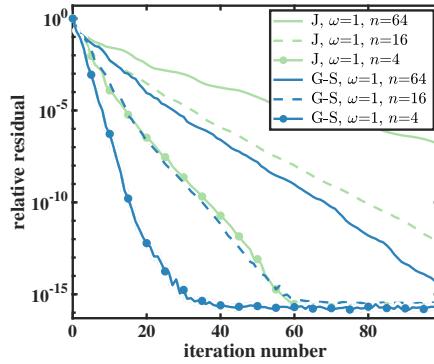
(a) $\omega = 2/3$; no Anderson acceleration



(b) $\omega = 1$; no Anderson acceleration



(c) $\omega = 2/3$; Anderson acceleration, $m = 5$



(d) $\omega = 1$; Anderson acceleration, $m = 5$

Figure 6.8: Weak-scaling study: convergence results. J and G-S labels refer to the Jacobi and Gauss–Seidel methods, respectively. Note that convergence is faster when NetUQ employs smaller networks, Gauss–Seidel (rather than Jacobi), $\omega = 1$ (rather than $\omega = 2/3$), and Anderson acceleration.

the number of components increases.

Fig. 6.10 repeats the same study, but with all relaxation methods using Anderson acceleration with $m = 5$. By comparing with the corresponding results without Anderson acceleration shown in Fig. 6.9, it is again clear that employing Anderson acceleration substantially improves the performance of NetUQ. As in the strong-scaling case, we again observe speedup saturation at $n = 16$. As before, we also observe that Anderson acceleration reduces the performance discrepancy arising from different values of the relaxation factor ω .

Finally, Fig. 6.11 assesses the effect of the permutation \mathbf{p} on the performance of the Gauss–Seidel method in weak scaling. Results are similar to the strong-scaling case: the choice of permutation tuple has essentially no effect on the number of iterations required for convergence, so performance is driven by the number of sequential steps per iteration n_{seq} exposed by the choice of permutation.

7. Conclusions. This work proposed the network uncertainty quantification (NetUQ) method for propagating uncertainties in large-scale networks. The approach simply assumes the existence of a collection of components, each of which is characterized by exogenous-input random variables, endogenous-input random variables, output random variables, and an uncertainty-propagation operator. The method constructs the network simply by associating endogenous-input random variables for each component with output random variables from other components; no other

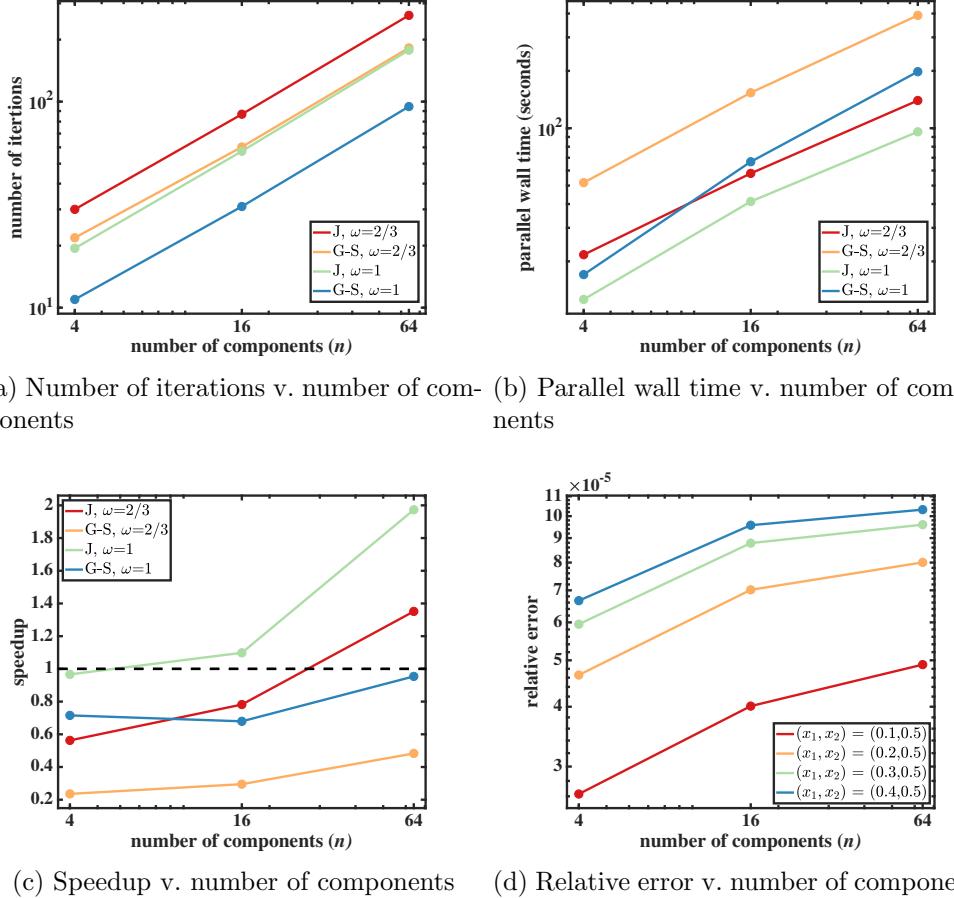
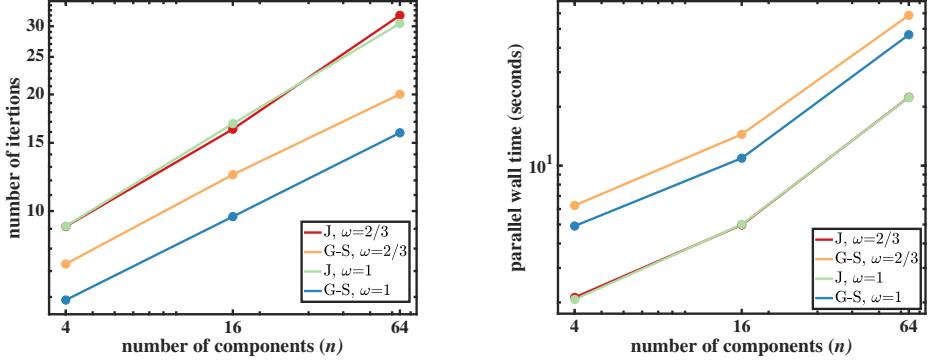


Figure 6.9: Weak-scaling study without Anderson acceleration. Iterations are terminated when the relative residual reaches a value of 10^{-3} . J and G-S labels refer to the Jacobi and Gauss–Seidel methods, respectively. Relative error refers to the error in the PCE coefficients of the field random variable at several spatial locations computed using the NetUQ approach (executed with 3rd-order total-degree truncation and satisfying a relative residual of 10^{-10}) with respect to the same PCE coefficients computed by solving the global uncertainty-propagation problem.

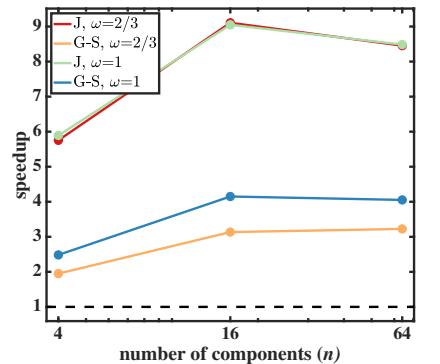
inter-component compatibility conditions are required. This formulation promotes component independence by enabling different components to use different functional representations of random variables and different uncertainty-propagation operators.

To resolve the resulting network uncertainty-propagation problem, the method employs the classical relaxation methods of Jacobi and Gauss–Seidel iteration equipped with Anderson acceleration. Each Jacobi iteration entails embarrassingly parallel component uncertainty propagation, while each Gauss–Seidel iteration incurs feed-forward uncertainty propagation in a DAG created by “splitting” selected edges in the network. Critically, no two-way coupled solves between components are required within a given relaxation iteration.

In addition to proposing the NetUQ method, this work provided supporting error analysis (Section 5), which is applicable to the case where the component uncertainty-propagation operators comprise an approximation of an underlying “truth” uncertainty-propagation operator. These results include *a priori* (Proposition 5.2) and *a posteriori* (Proposition 5.3) error bounds for the general case. In addition, this section performed error analysis for the case where the uncertainty-propagation operator restricts the solution to lie in a subspace of the space considered by the truth



(a) Number of iterations v. number of components (b) Parallel wall time v. number of components



(c) Speedup v. number of components

Figure 6.10: Weak-scaling study using Anderson acceleration with $m = 5$. Iterations are terminated when the relative residual reaches a value of 10^{-3} . J and G-S labels refer to the Jacobi and Gauss–Seidel methods, respectively.

uncertainty-propagation operator, including conditions for zero in-plane error (Propositions 5.6 and 5.11), and *a priori* (Proposition 5.7) and *a posteriori* (Proposition 5.10) in-plane error bounds.

Numerical experiments performed in Section 6 studied the strong-scaling (Section 6.3) and weak-scaling (Section 6.4) performance of the method on a benchmark parameterized diffusion problem. These results illustrate that convergence occurs in the fewest number of iterations for smaller networks, Gauss–Seidel iteration, and a relaxation factor of $\omega = 1$. However, due to the embarrassingly parallel nature of Jacobi iterations, the Jacobi method yields superior parallel wall-time performance compared with the Gauss–Seidel method. Critically, numerical experiments also demonstrated that Anderson acceleration is essential for generating substantial speedups relative to monolithic full-system uncertainty propagation.

Future work entails demonstrating the methodology on problems characterized by greater discrepancies between components, investigating other mechanisms to accelerate convergence of the fixed-point iterations, and integrating surrogate models to reduce the wall-time incurred by component deterministic simulations used for component uncertainty propagation. Preliminary results in this direction have been presented in Ref. [27], which applied NetUQ to propagate uncertainties in large-scale 3D hemodynamics models, and employed reduced-order models to reduce the computational cost of solving the component deterministic problems.

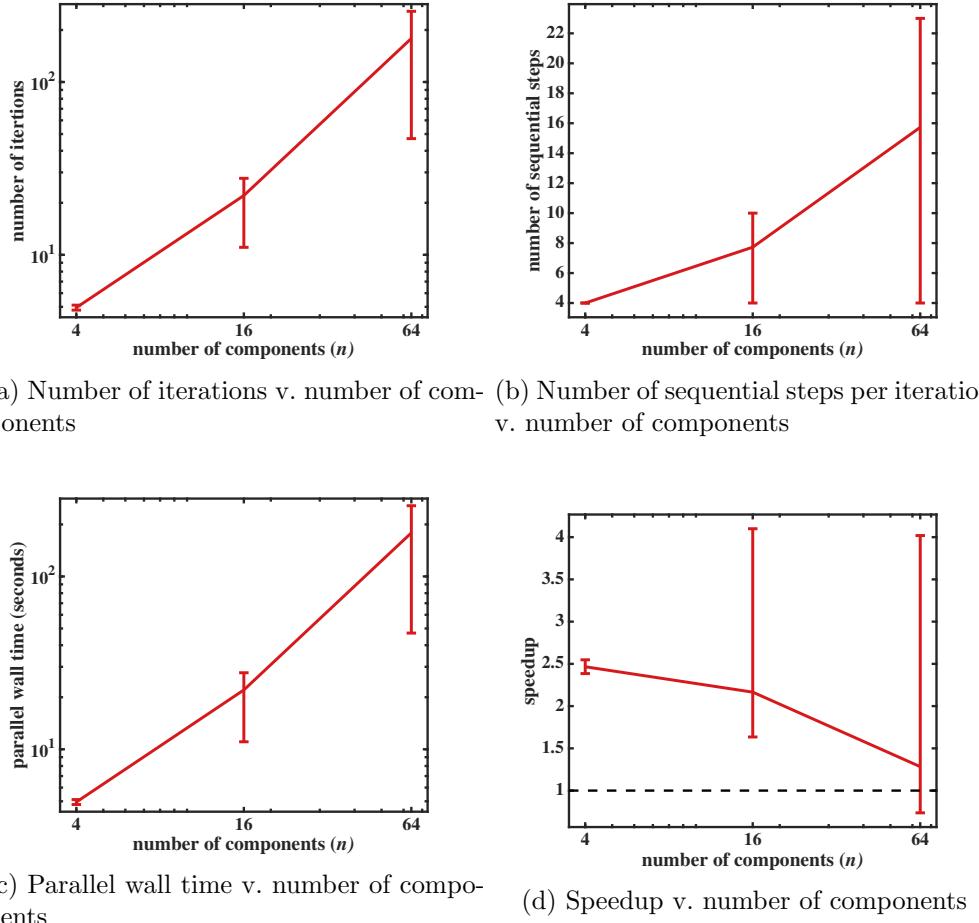


Figure 6.11: Weak-scaling study using Anderson acceleration with $m = 5$. Gauss–Seidel with $\omega = 1$ performance for ten random permutations \mathbf{p} . Red curve represents the mean value, and vertical lines span the minimum and maximum values computed over these permutations.

Acknowledgements. The authors gratefully acknowledge Jiahua Jiang for helpful initial experiments performed with Anderson acceleration. This work was performed at Sandia National Laboratories and was supported by the LDRD program (project 190968). This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

- [1] S. AMARAL, D. ALLAIRE, AND K. WILLCOX, *A decomposition-based approach to uncertainty analysis of feed-forward multicomponent systems*, International Journal for Numerical Methods in Engineering, 100 (2014), pp. 982–1005.
- [2] D. G. ANDERSON, *Iterative procedures for nonlinear integral equations*, Journal of the ACM (JACM), 12 (1965), pp. 547–560.
- [3] M. ARNST, R. GHANEM, E. PHIPPS, AND J. RED-HORSE, *Dimension reduction in stochastic modeling of coupled*

- problems*, International Journal for Numerical Methods in Engineering, 92 (2012), pp. 940–968.
- [4] ———, *Measure transformation and efficient quadrature in reduced-dimensional stochastic modeling of coupled problems*, International Journal for Numerical Methods in Engineering, 92 (2012), pp. 1044–1080.
- [5] ———, *Reduced chaos expansions with random coefficients in reduced-dimensional stochastic modeling of coupled problems*, International Journal for Numerical Methods in Engineering, 97 (2014), pp. 352–376.
- [6] M. ARNST, R. GHANEM, AND C. SOIZE, *Identification of Bayesian posteriors for coefficients of chaos expansions*, Journal of Computational Physics, 229 (2010), pp. 3134 – 3154.
- [7] I. BABUŠKA, F. NOBILE, AND R. TEMPONE, *A stochastic collocation method for elliptic partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 45 (2007), pp. 1005–1034.
- [8] I. BABUSKA, R. TEMPONE, AND G. E. ZOURARIS, *Galerkin finite element approximations of stochastic elliptic partial differential equations*, SIAM Journal on Numerical Analysis, 42 (2004), pp. 800–825.
- [9] I. BILIONIS AND N. ZABARAS, *Multi-output local Gaussian process regression: Applications to uncertainty quantification*, Journal of Computational Physics, 231 (2012), pp. 5718–5746.
- [10] ———, *Multidimensional adaptive relevance vector machines for uncertainty quantification*, SIAM Journal on Scientific Computing, 34 (2012), pp. B881–B908.
- [11] R. H. CAMERON AND W. T. MARTIN, *The orthogonal development of non-linear functionals in series of Fourier-Hermite functionals*, Annals of Mathematics, 48 (1947), pp. 385–392.
- [12] X. CHEN, B. NG, Y. SUN, AND C. TONG, *A flexible uncertainty quantification method for linearly coupled multi-physics systems*, Journal of Computational Physics, 248 (2013), pp. 383–401.
- [13] A. CHIRALAKSANAKUL AND S. MAHADEVAN, *First-order approximation methods in reliability-based design optimization*, Journal of Mechanical Design, 127 (2005), pp. 851–857.
- [14] P. G. CONSTANTINE, A. DOOSTAN, AND G. IACCARINO, *A hybrid collocation/Galerkin scheme for convective heat transfer problems with stochastic boundary conditions*, International Journal for Numerical Methods in Engineering, 80 (2009), pp. 868–880.
- [15] P. G. CONSTANTINE, E. T. PHIPPS, AND T. M. WILDEY, *Efficient uncertainty propagation for network multi-physics systems*, International Journal for Numerical Methods in Engineering, 99 (2014), pp. 183–202.
- [16] M. K. DEB, I. M. BABUŠKA, AND J. T. ODEN, *Solution of stochastic partial differential equations using Galerkin finite element techniques*, Computer Methods in Applied Mechanics and Engineering, 190 (2001), pp. 6359–6372.
- [17] B. DEBUSSCHERE, K. SARGSYAN, C. SAFTA, AND K. CHOWDHARY, *Uncertainty quantification toolkit (UQTk)*, Handbook of Uncertainty Quantification, (2016), pp. 1–21.
- [18] B. J. DEBUSSCHERE, H. N. NAJM, P. P. PÉBAY, O. M. KNIO, R. G. GHANEM, AND O. P. LE MAÎTRE, *Numerical challenges in the use of polynomial chaos representations for stochastic processes*, SIAM Journal on Scientific Computing, 26 (2004), pp. 698–719.
- [19] X. DU AND W. CHEN, *Collaborative reliability analysis under the framework of multidisciplinary systems design*, Optimization and Engineering, 6 (2005), pp. 63–84.
- [20] ERNST, OLIVER G., MUGLER, ANTJE, STARKLOFF, HANS-JÖRG, AND ULLMANN, ELISABETH, *On the convergence of generalized polynomial chaos expansions*, ESAIM: M2AN, 46 (2012), pp. 317–339.
- [21] H.-R. FANG AND Y. SAAD, *Two classes of multisecant methods for nonlinear acceleration*, Numerical Linear Algebra with Applications, 16 (2009), pp. 197–221.
- [22] S. FRIEDMAN, B. ISAAC, S. F. GHOREISHI, AND D. L. ALLAIRE, *Efficient decoupling of multiphysics systems for uncertainty propagation*, in 2018 AIAA Non-Deterministic Approaches Conference, 2018, p. 1661.
- [23] R. G. GHANEM AND P. D. SPANOS, *Stochastic finite elements: a spectral approach*, Courier Corporation, 2003.
- [24] S. F. GHOREISHI AND D. L. ALLAIRE, *Compositional uncertainty analysis via importance weighted Gibbs sampling for coupled multidisciplinary systems*, in 18th AIAA non-deterministic approaches conference, 2016, p. 1443.
- [25] M. A. GREPL, Y. MADAY, N. C. NGUYEN, AND A. T. PATERA, *Efficient reduced-basis treatment of nonaffine and nonlinear partial differential equations*, ESAIM: Mathematical Modelling and Numerical Analysis, 41 (2007), pp. 575–605.
- [26] X. S. GU, J. E. RENAUD, AND C. L. PENNINGER, *Implicit uncertainty propagation for robust collaborative optimization*, Journal of Mechanical Design, 128 (2006), pp. 1001–1013.
- [27] S. GUZZETTI, L. MANSILLA ALVAREZ, P. BLANCO, K. CARLBERG, AND A. VENEZIANI, *Propagating uncertainties in large-scale hemodynamics models via network uncertainty quantification and reduced-order modeling*, Computer Methods in Applied Mechanics and Engineering, in press (2019).
- [28] S. HOSDER, R. WALTERS, AND R. PEREZ, *A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations*, in 44th AIAA Aerospace Sciences Meeting and Exhibit, 2006, p. 891.
- [29] T. R. JENSEN AND B. TOFT, *Graph coloring problems*, vol. 39, John Wiley & Sons, 2011.
- [30] M. KOKKOLARAS, Z. P. MOURELATOS, AND P. Y. PAPALAMBROS, *Design optimization of hierarchically decomposed multilevel systems under uncertainty*, in ASME 2004 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers, 2004, pp. 613–624.
- [31] K. LEE, K. CARLBERG, AND H. C. ELMAN, *Stochastic least-squares Petrov–Galerkin method for parameterized*

- linear systems*, SIAM/ASA Journal on Uncertainty Quantification, 6 (2018), pp. 374–396.
- [32] Q. LIAO AND K. WILLCOX, *A domain decomposition approach for uncertainty analysis*, SIAM Journal on Scientific Computing, 37 (2015), pp. A103–A133.
 - [33] S. MAHADEVAN AND N. SMITH, *Efficient first-order reliability analysis of multidisciplinary systems*, International Journal of Reliability and Safety, 1 (2006), pp. 137–154.
 - [34] O. L. MAÎTRE AND O. KNIO, *Spectral Methods for Uncertainty Quantification: With Applications to Computational Fluid Dynamics (Scientific Computation)*, Springer, 1st edition. ed., April 2010.
 - [35] J. D. MARTIN AND T. W. SIMPSON, *A methodology to manage system-level uncertainty during conceptual design*, Journal of Mechanical Design, 128 (2006), pp. 959–968.
 - [36] A. MITTAL, X. CHEN, A. H. TONG, AND G. IACCARINO, *A flexible uncertainty propagation framework for general multiphysics systems*, SIAM/ASA Journal on Uncertainty Quantification, 4.
 - [37] F. NOBILE, R. TEMPONE, AND C. G. WEBSTER, *A sparse grid stochastic collocation method for partial differential equations with random input data*, SIAM Journal on Numerical Analysis, 46 (2008), pp. 2309–2345.
 - [38] S. SANKARARAMAN AND S. MAHADEVAN, *Likelihood-based approach to multidisciplinary analysis under uncertainty*, Journal of Mechanical Design, 134 (2012), p. 031008.
 - [39] F. SANSON, O. LE MAÎTRE, AND P. CONGEDO, *Uncertainty propagation framework for systems of solvers*, (2018).
 - [40] A. TOTH AND C. KELLEY, *Convergence analysis for Anderson acceleration*, SIAM Journal on Numerical Analysis, 53 (2015), pp. 805–819.
 - [41] H. F. WALKER AND P. NI, *Anderson acceleration for fixed-point iterations*, SIAM Journal on Numerical Analysis, 49 (2011), pp. 1715–1735.
 - [42] X. WAN AND G. E. KARNIADAKIS, *Multi-element generalized polynomial chaos for arbitrary probability measures*, SIAM Journal on Scientific Computing, 28 (2006), pp. 901–928.
 - [43] D. XIU AND G. E. KARNIADAKIS, *Modeling uncertainty in steady state diffusion problems via generalized polynomial chaos*, Computer Methods in Applied Mechanics and Engineering, 191 (2002), pp. 4927–4948.
 - [44] D. XIU AND G. E. KARNIADAKIS, *The Wiener–Askey polynomial chaos for stochastic differential equations*, SIAM Journal on Scientific Computing, 24 (2002), pp. 619–644.
 - [45] ———, *Modeling uncertainty in flow simulations via generalized polynomial chaos*, Journal of Computational Physics, 187 (2003), pp. 137–167.
 - [46] W. YAO, X. CHEN, W. LUO, M. VAN TOOREN, AND J. GUO, *Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles*, Progress in Aerospace Sciences, 47 (2011), pp. 450 – 479.