SAND2022-11945J

Network Uncertainty Quantification for Analysis of Multi-Component Systems

John Tencer
Sandia National Laboratories
Albuquerque, New Mexico
Email: jtencer@sandia.gov

Edward Rojas
Los Alamos National Laboratory
Los Alamos, New Mexico
Email: efrojas@lanl.gov

Benjamin B. Schroeder
Sandia National Laboratories
Albuquerque, New Mexico
Email: bbschro@sandia.gov

In order to impact physical mechanical system design decisions and realize the full promise of high-fidelity computational tools, simulation results must be integrated at the earliest stages in the design process. This is particularly challenging when dealing with uncertainty and optimizing for system-level performance metrics, as full-system models (often notoriously expensive and time-consuming to develop) are generally required to propagate uncertainties to system-level quantities of interest. Methods for propagating parameter and boundary condition uncertainty in networks of interconnected components hold promise for enabling design under uncertainty in real-world applications. These methods avoid the need for time consuming mesh generation of full-system geometries when changes are made to components or subassemblies. Additionally, they explicitly tie full-system model predictions to component/subassembly validation data which is valuable for qualification. These methods work by leveraging the fact that many engineered systems are inherently modular, being comprised of a hierarchy of components and subassemblies that are individually modified or replaced to define new system designs. By doing so, these methods enable rapid model development and the incorporation of uncertainty quantification earlier in the design process.

The resulting formulation of the uncertainty propagation problem is iterative. We express the system model as a network of interconnected component models, which exchange solution information at component boundaries. We present a pair of approaches for propagating uncertainty in this type of decomposed system and provide implementations in the form of an open-source software library. We demonstrate these tools on a variety of applications and demonstrate the impact of problem-specific details on the performance and accuracy of the resulting UQ analysis. This work represents the most comprehensive investigation of these network uncertainty propagation methods to date.

Nomenclature

$\mathcal{M}$	deterministic operator
u	exogenous input variable
y	endogenous input variable
c	polynomial chaos expansion coefficients
w	quadrature weights
QoI	Quantity of Interest
UQ	Uncertainty Quantification
PCE	Polynomial Chaos Expansion
DD	Domain Decomposition
T	temperature

1 Introduction

The majority of systems in science and engineering are intentionally designed as collections of components and subassemblies connected in a tree-like structure. This type of design is highly advantageous as it allows for interoperability of components, parallel design efforts, and straightforward approaches to quality assurance. Unfortunately, the most common approaches to modeling and simulation in support of the design of these systems is not similarly constructed. Often, these modeling efforts are interested in system-level quantities of interest (QoIs) and the entire system is treated in a monolithic way. This approach necessitates an undesirable tradeoff: either this monolithic system model is extremely complicated and expensive to construct and evaluate, or it involves significant simplifying assumptions, such as neglecting physics and/or geometric features of components and subassemblies. This latter approach of using simplified representations of components and subassemblies is most common.

While in many cases, these simplifying assumptions are justified and defensible, the impact that these simplifications have on system-level QoIs is difficult to quantify, representing a challenge for high consequence applications for which uncertainty quantification (UQ) is a required piece of the qualification process. Additionally, these simplified models represent a break from the more detailed models commonly used for the design and qualification of individual components and subassemblies. This means that the monolithic system-level model must be validated against system-level test data, which are often very expensive to generate and thus sparse. System-level validation is also inherently challenging because it is difficult to determine the source of any discrepancy. Hierarchical validation allows for discrepancy sources to be more easily isolated. Additional negative consequences of relying exclusively on system-level validation data are increased uncertainty in the predictions of the system-level model and reduced confidence in the resulting predictions.

Researchers have developed a variety of methods to address these challenges [1–3]. In this work, we provide efficient implementations of two of these methods in the form of an open-source software library, PyNetUQ [1]. We publish our open-source tools for uncertainty propagation in networks and remark that these tools are extensible to be used with any simulation tool (including arbitrary surrogate modeling tools) through the construction of a simple Python interface class. Additional interface classes for a variety of simulation tools are currently under active development. We evaluate the convergence behavior of these network UQ problems and the impact of system-level boundary condition types on this network performance. Finally, we present demonstration uncertainty propagation results for a series of increasingly complex exemplar applications.

The engineering exemplar problems used for demonstration in this work are finite element models, solving either thermal or mechanical responses to stimuli external to the system. QoIs for such exemplars are either thermal or mechanical state variables such as temperatures at locations within the system or spatial deflection. Such state variables are often of interest for full system analyses because they can indicate when a component has reached the edge of its designed performance window. Understanding such responses often requires a full system model due to the state variable's response often being dependent upon the response of other neighboring components. Uncertain variables utilized in these systems could have included material properties such as thermal conductivity, specific heat, or emissivity, or geometric tolerances, but only boundary conditions were specified as uncertain in the current work.

The remainder of the document will be organized as follows. Section 2 will introduce the mathematics underpinning the network based UQ approach, how domain decomposition is utilized, and the implementation of those two concepts for the present work. Engineering examples of increasing complexity will then be used in Section 3 to demonstrate the approach's performance for different boundary conditions and equation sets. The UQ approach will also be demonstrated on a large scale exemplar engineering system within this section to show how the approach can apply to relevant applications. Lastly, conclusions and future plans for the UQ approach will be presented in Section 4.

[1]https://jtencer.github.io/PyNetUQ/

2 Methods

We have implemented the network UQ method [1, 4] in an extensible Python library (PyNetUQ), which interfaces with a variety of physics simulation tools to evaluate component UQ problems. Although the network UQ method is agnostic to the functional representation used for the underlying random variables, we will restrict ourselves to polynomial chaos expansions (PCEs) [5–8].

In the subsections to follow, we will outline the approach taken and provide some details on the implementation. Section 2.1 will provide basic background on polynomial chaos and introduce some relevant notation. Section 2.2 will then describe the ways in which the domain is decomposed and the two relevant uncertainty propagation methods implemented in PyNetUQ. Finally, Section 2.3 will contain a discussion on implementation details.

2.1 Polynomial Chaos Expansions

In the broadest terms, we seek to determine a probabilistic characterization of the system response, *QoI*, to uncertainty in parameter values, u. The parameter values are parametrized using the germ, $\xi = \{\xi_1, \xi_2, ...\}$, a vector of independent random variables (RVs), such that

$$u \equiv u(\xi). \tag{1}$$

When using PCEs, we represent the dependence of the solution on the germ through the series

$$QoI(\xi) \approx \sum_k c_k \Psi_k(\xi), \tag{2}$$

where the Ψ_k's are suitably chosen orthogonal functionals of the random variables and c_k are deterministic coefficients given by

$$c_k = \frac{\langle QoI(\xi)\Psi_k(\xi)\rangle}{\langle \Psi_k^2(\xi)\rangle}, \tag{3}$$

with $\langle\cdot\rangle$ an appropriate inner product. The choice of Ψ depends upon the distribution of ξ. For the examples to follow Hermite polynomials are used, which is appropriate for cases where ξ is Gaussian. We approximate the inner product in the numerator using non-intrusive spectral projection (NISP) and quadrature integration

$$\begin{aligned}\langle QoI(\xi)\Psi_k(\xi)\rangle &= \int QoI(\xi)\Psi_k(\xi)\pi(\xi)d\xi \\ &\approx \sum_q QoI(\xi_q)\Psi_k(\xi_q)w_q\end{aligned} \quad . \tag{4}$$

This process is visualized for a single-component system in Figure 1a. The sampling step corresponds to the selection of the quadrature points ξ_q in Eq. 4 and the deterministic solutions correspond to the evaluations of $QoI(\xi_q)$. For domain-decomposed problems, this process becomes somewhat more complicated as shown in Figures 1b and 1c depending upon the approach taken.

2.2 Model Decomposition

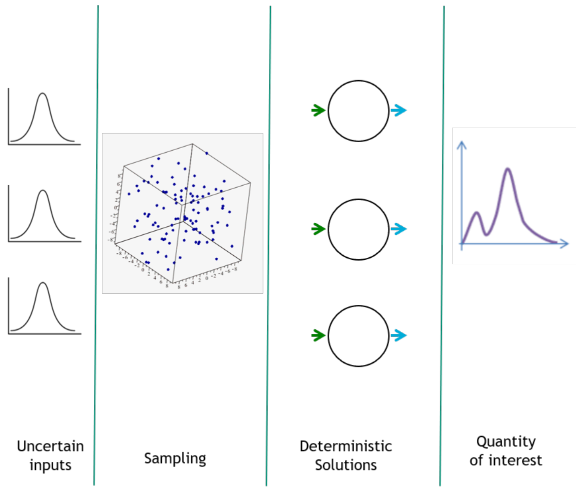
In this section, we will describe the domain decomposition approach used and two different uncertainty propagation algorithms applicable to domain-decomposed problems. Our approach is analogous to domain decomposition methods developed for parallel computing [9, 10] but is implemented in a less intrusive manner. The ideal monolithic system model is described by $\mathcal{M}(u) : u \mapsto QoI$. As previously described, this system model is generally impractical to construct for complex systems. Instead, the system model is decomposed into a set of n local models $\mathcal{M}_i : (y_i, u_i) \mapsto x_i$, $i = 1, \cdots, n$. We define $u_i \subseteq u$ to be the set of exogenous input parameters relevant to component i and y_i to be the corresponding set of endogenous inputs. The endogenous inputs are derived from the outputs x_i of neighboring components (e.g., interface conditions).

For all of the results in this paper, an overlapping domain decomposition technique with Dirichlet-Dirichlet coupling terms is utilized. However, the PyNetUQ software is agnostic to the domain decomposition scheme used, working instead at a higher level of abstraction (inputs/outputs) and leaving the coupling details to the individual component models. As such, the library is also applicable to non-overlapping domain decomposition techniques with Dirichlet-Neumann or Robin-Robin coupling terms as well. More information on the specific coupling scheme used is proved in Section 2.3.

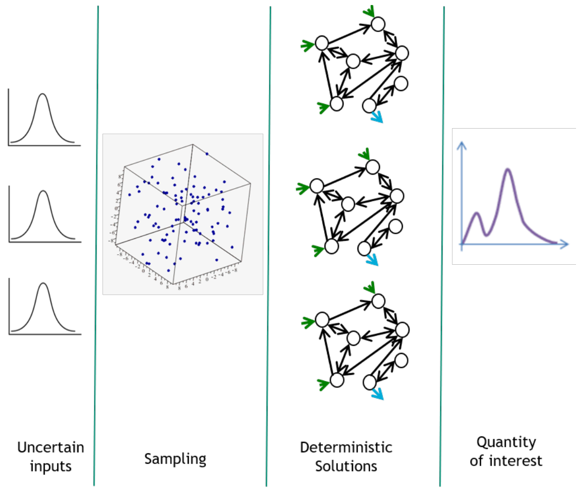
There are two uncertainty propagation methods applicable to systems decomposed in this way. The first approach, which we will term deterministic domain decomposition (deterministic DD) effectively decouples the domain decomposition algorithm from the uncertainty propagation algorithm. The second approach, which we term network uncertainty quantification (NetUQ) combines the domain decomposition and uncertainty propagation algorithms in order to realize improved efficiency.

2.2.1 Uncertainty Propagation via Deterministic Domain Decomposition

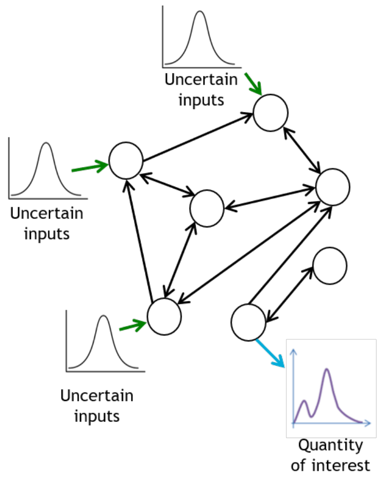
Deterministic domain decomposition (deterministic DD) is an approach for simulating a large complex physi-



(a) Monolithic



(b) Deterministic Domain Decomposition



(c) NetUQ

Fig. 1: Graphical illustrations of the different approaches used for uncertainty propagation in this paper

cal system which has been spatially decomposed into a network of interconnected components. The components exchange information along their interfaces and the full-system solution is achieved through iteration. In our implementation, the forward problems at each iteration for all components are solved simultaneously using the additive Schwarz method [9, 11].

Uncertainty propagation for deterministic DD is performed in exactly the same way as when the ideal monolithic system model is available. The only difference is the substitution of the alternative deterministic solution method used for evaluating the simulation outputs at each sample point. This is seen visually by comparing Figures 1a and 1b. The domain decomposition is deterministic and the uncertainty propagation algorithm wraps the deterministic system solution procedure.

In PyNetUQ, we use a PCE to represent the uncertainty in any QoIs and utilize a quadrature rule for determining the sample points. PyNetUQ leverages UQTk [12] to generate either full or sparse quadrature rules for evaluating Eq. 4. Since the UQ method wraps around the deterministic DD algorithm, a single global quadrature rule for integrating over ξ is defined for the entire system and a separate deterministic DD iterative solution must be performed at each quadrature point, $\xi^{(q)}$.

2.2.2 Uncertainty Propagation via NetUQ

In the network UQ (NetUQ) method illustrated in Figure 1c, the interfaces between components are treated as stochastic rather than deterministic [1, 4]. We define $u_i \subseteq u$ to be the set of exogenous input random variables relevant to component i and y_i to be the corresponding set of endogenous input random variables. The endogenous inputs are derived from the outputs x_i of neighboring components (e.g., interface conditions). Note that in contrast to the deterministic DD approach, x_i and y_i are treated as stochastic in the NetUQ approach.

We represent each random variable using a PCE. In order to propagate uncertainty through each component, we define a local quadrature rule for integrating over ξ for each component. The local quadrature rule allows for different stochastic representations for different components, adding flexibility and potentially reducing the number of simulations required. These quadrature points $\xi^{(q)}$ define samples for both the exogenous inputs $u \approx \sum_k (u_i)_k \Psi_k(\xi^{(q)})$ and endogenous inputs $y_i^{(q)} \approx \sum_k (y_i)_k \Psi_k(\xi^{(q)})$. The forward problem $\mathcal{M}$ is then solved for each quadrature point and the resulting outputs $x_i^{(q)}$ are used to update the PCE coefficients using Galerkin projection as in Eq. 3.

The network uncertainty propagation problem for the full system may be written as a fixed-point problem. We stack up the inputs and outputs from each component $\tilde{u} := \left[u_1^T \cdots u_n^T\right]^T$, $\tilde{y} := \left[y_1^T \cdots y_n^T\right]^T$, and $\tilde{x} := \left[x_1^T \cdots x_n^T\right]^T$ and define a full-system uncertainty propagation operator $\tilde{\mathcal{M}} : (\tilde{y}, \tilde{u}) \mapsto \tilde{x}$. The endogenous outputs of each component are routed to the appropriate endogenous inputs of neighboring components. These relationships are encoded into the adjacency matrix $I_x^y \in \{0,1\}^{n_y \times n_x}$, which satisfies

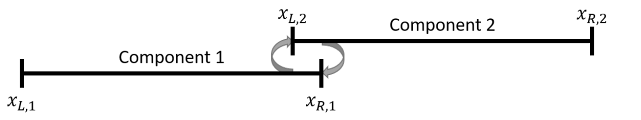


Fig. 2: Schematic of overlapping domain decomposition strategy

$$\tilde{y} \equiv I_x^y \tilde{x}. \tag{5}$$

n_x and n_y are the total number of endogenous output and input values respectively across all n components. The result is the fixed-point problem

$$\tilde{x} = \mathcal{M}(I_x^y \tilde{x}, \tilde{u}), \tag{6}$$

which can be solved via Jacobi iteration.

In our implementation, we never explicitly form the adjacency matrix. Rather, it is represented implicitly in the boundary conditions specified for each component model $\mathcal{M}_i$. In this way, the PyNetUQ library operates only in terms of component inputs and outputs.

As long as the fixed-point operator is a contraction, the fixed-point iterations generated by the Jacobi algorithm converge q-linearly [1]. This is rarely competitive with other solution methods for nonlinear equations (e.g. Newton's method). Thus relaxation methods [13–15] are typically used only where these types of solvers are not applicable.

In this work, we seek to improve the convergence rate by employing Anderson acceleration [16–18]. Anderson acceleration works by modifying the fixed-point-iteration updates in a way that is similar to the generalized minimum residual (GMRES) method for linear problems [19] and multisecant methods for nonlinear problems [17].

2.3 Implementation

For the results to follow, we exclusively use an overlapping domain decomposition technique with Dirichlet boundary conditions imposed on both sides of the interface. As an example, consider a system with two components, $\mathcal{M}_1$ and $\mathcal{M}_2$ which operate on the domains shown in Figure 2. The boundary conditions at $x_{L,1}$ and $x_{R,2}$ along with any other parameters comprise potential exogenous inputs u. The boundary conditions at $x_{R,1}$ and $x_{L,2}$ are endogenous inputs to $\mathcal{M}_1$ and $\mathcal{M}_2$ respectively.

For deterministic DD, these are simply the solution variables of the other component(s) evaluated at the interface locations. For NetUQ, these are the PCE representations of the solution variables of the other component(s) evaluated at the interface locations and sampled at the appropriate quadrature point.

In the PyNetUQ implementation, each component model is responsible for extracting its own endogenous boundary conditions from the outputs of each neighboring component. At each iteration, the library provides a dummy version of these neighbors' outputs populated with the appropriate information for each required point in probability space and executes the component model. These component model evaluations are all performed in parallel within each network iteration.

Iterations continue until either a convergence tolerance is met

$$\frac{\|\tilde{x}_{i+1}-\tilde{x}_i\|_2}{\|\tilde{x}_i\|_2} < \varepsilon \tag{7}$$

or until the step size fails to decrease for 10 consecutive iterations. In practice, $\|\tilde{x}_i\|_2$ will only approach zero in degenerate cases where the solution is trivial. For the results to follow, a convergence tolerance of $\varepsilon = 10^{-5}$ is used.

As will be seen, an important parameter for transient simulations is the output frequency. Since the PyNetUQ library couples adjacent components using these output files, the output frequency must be sufficient for time accuracy. Infrequent outputs result in temporally underresolved boundary conditions for neighboring components. Since each component manages its own timestepping scheme and output frequency, care must be taken to ensure that sufficiently resolved transient information is made available to the network.

3 Results

Since monolithic system models are difficult to construct, we have chosen simplistic example problems for which $\mathcal{M}$ is easily constructed or have used the deterministic domain decomposition method described in Section 2.2.1 as a substitute for the results to follow. In Section 3.1 we analyze the effect that different boundary conditions have on the network convergence for thermal and mechanical problems by focusing on simple 1D domains. In Section 3.2 we extend this discussion to the propagation of uncertainty in a set of increasingly complex thermal systems. The first is a 1D steady-state linear heat conduction problem, the second is a 1D transient heat conduction problem, and the third is a complex system transient thermal model with multiple sources on nonlinearity.

3.1 Boundary Condition Effects on Network Convergence

In this section, we apply the network approach to a set of thermal and mechanical problems to analyze the convergence behavior. Both problems are pseudo-1D in that they are mathematically one-dimensional but are analyzed using a 3D finite element mesh. Both problems are also linear and transient. In all cases, we will apply a fixed Dirichlet boundary condition to one side and vary the boundary condition applied to the other side in order to determine what effect, if any, the boundary condition has on the network convergence behavior.

3.1.1 Mechanical Network Results

For this first example, we analyze a linear elastic cantilevered beam with a Young's modulus of 10^6 and a Poisson's ratio of $1/4$. The beam is 10 units long with a constant square cross-section of unit area. One end of the beam ($x = 0$) is held fixed. Dirichlet and Neumann boundary conditions were applied to the other (free, $x = 10$) end of the beam in order to investigate the efficiency of the network for simple systems. The governing equations are given by

$$\begin{aligned}(EIu_{xx})_{xx} + mu_{tt} &= 0\\ u(0,t) &= 0,\\ u_x(0,t) &= 0\end{aligned} \tag{8}$$

with Young's modulus E, mass per unit length m, and second moment of inertia I. For this geometry, $I = 1/12$.

The Dirichlet boundary condition applied is a linear ramp from 0 displacement to the maximum tip displacement over 2 milliseconds, such that

$$u(L,t) = \frac{u_{max}t}{2\times 10^{-6}}. \tag{9}$$

The prescribed displacement is applied quasi-statically. The solution in this case is given by

$$u(x,t) = \frac{u(L,t)x^2(3L-x)}{2L^3} \tag{10}$$

The Neumann boundary condition applied is a constant load perpendicular to the beam, such that

$$(EIu_{xx}(L,T))_x = F, \tag{11}$$

with F being the prescribed load. The solution in this case is given by

$$u(x,t) = \frac{Fx^2(3L-x)}{6EI}. \tag{12}$$

Our quantities of interest are the tip and mid-point deflections at the final time step.

Two solution methods were contrasted: (1) a monolithic approach (labeled "Classic" in the Figures) and (2) the NetUQ approach described in Section 2.2.2. The monolithic case is the traditional FEM analysis of a cantilever beam using a single component meshing scheme. For the NetUQ approach, the domain is divided into two overlapping regions, $x \in [0, 5.5]$ and $x \in [4.5, 10]$. The PyNetUQ library is used to couple separate simulations of the two components. Convergence in this context represents the network's ability to approach the monolithic solution within prescribed solution parameters and will be the main metric when analyzing its performance and efficiency.

Boundary conditions play a significant role in determining network convergence behavior for the system. Figure 3 illustrates the difference in behavior of the midpoint and tip displacements at the final time step of each iteration of the network analysis of the transient mechanical model when using either Dirichlet or Neumann boundary conditions. Figure 3a imposes a linear-in-time displacement resulting in a unit tip displacement at the final time (2 milliseconds), whereas Figure 3b imposes a force necessary to create a maximum tip deflection of approximately one unit, Figures 3c and 3d double the imposed conditions from Figures 3a and 3b respectively to correspond to a tip deflection of two.

The Neumann boundary condition converges more quickly for smaller tip deflections as shown in Figure 3b; however, for the larger deflections given in Figures 3c and 3d the network converges more quickly for the Dirichlet boundary condition. For the Dirichlet cases, the tip deflection is prescribed, forcing the network to initially simulate the largest deflection at all iterations; the network would then initially begin with the largest displacement variances and then slowly try to converge to the appropriate solution. For small deflections the network initially overestimates the center displacement and underestimates for larger deflections, whereas, the Neumann case always seems to initially underestimate the deflection of the component for similar tip displacements. (This is related to the boundary condition used at the component interfaces to initialize the network.) This will not impact the network extensively; however, for large deflections this could increase the number of iterations needed to achieve sufficient tolerance and thus slow the speed of convergence.

Results after imposing an extreme displacement with Dirichlet boundary condition of 5 are shown in Figure 3e. Analyzing even a slightly larger tip displacement with the Neumann boundary conditions presents a convergence error. In this case, the applied force results in a maximum tip displacement of ≈ 2.5. Figure 3f shows that although the system has satisfied the imposed convergence tolerance the network converges to a slightly different final tip displacement. This was due to the output frequency sensitivity. The network is not required to communicate information at every time step for each component within the PyNetUQ framework. Rather, each component has a defined output frequency which may be adjusted individually and linear interpolation is used when a neighboring component requires information at an intermediate time point. Infrequent outputs (especially early in time) can result in incorrect behavior for networks with Neumann boundary conditions. Although this same frequency can produce accurate results for the extreme Dirichlet condition, it is not sufficient for larger Neumann boundary conditions. This sensitivity for the Neumann condition highlights the influence the boundary conditions have on the network.

3.1.2 Thermal network results

In this section, we analyze the network convergence for a transient linear heat conduction application. The governing equations are given by

$$\begin{aligned} u_t - \alpha u_{xx} &= 0 \\ u(0,t) &= T_0 \\ u(L,t) &= T_L \\ u(x,0) &= T_{init}, \end{aligned} \tag{13}$$

with $T_{init} = 300$, $T_0 = 1000$, and $\alpha = 0.01$. T_L is considered a variable boundary condition. The series solution from separation of variables is

$$\begin{aligned} u(x,t) &= T_0 + \frac{x}{L}(T_L - T_0) + \sum_{n=1}^{\infty} B_n C_n(x,t) \\ B_n &= \frac{4}{n\pi}(T_{init} - T_0) - \frac{2}{n\pi}(-1)^n (T_0 - T_L) \\ C_n(x,t) &= \sin\left(\frac{n\pi x}{L}\right) \exp\left(-\frac{\alpha t}{L^2}(n\pi)^2\right), \end{aligned} \tag{14}$$

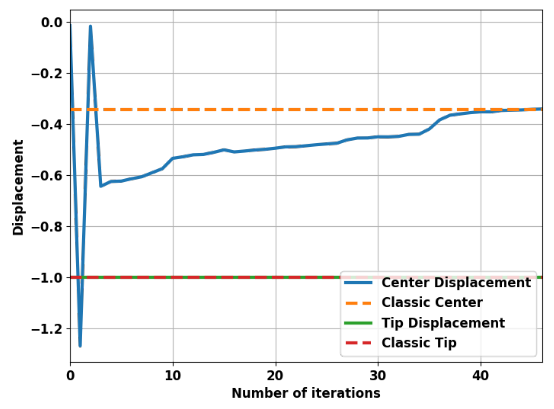
Once again, two solution methods are contrasted: (1) a monolithic approach (labeled “Classic” in the Figures) and (2) the NetUQ approach described in Section 2.2.2. The “Classic” approach is verified to agree with the analytical solution given by Eq. 14 to within the specified tolerances (10^-6 for this example).

The network can more easily solve the thermal case compared to the mechanical case; a smooth monotonic convergence is seen in Figure 4a. This is due to the different equations analyzed as the mechanical system requires the solution of a fourth order PDE; whereas, the thermal case is a second order PDE. Even if a larger temperature gradient is imposed, the system converges as quickly as the lower tip temperature condition as shown in Figure 4c.

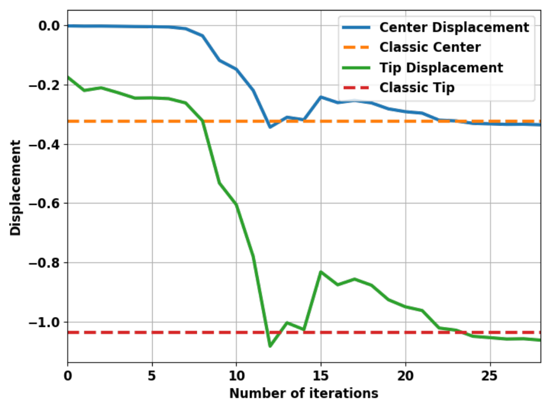
Varying the Dirichlet boundary conditions on the thermal case seemed to have negligible impacts on the network performance. Examining the Neumann boundary conditions revealed a similar lack of sensitivity. Figure 5 shows the results of applying either a negative (top) or positive (bottom) heat flux to the right-hand boundary. The network performed just as well when the fluxes varied from a positive to negative flux. Figures 5b and 5d show that the network converges to the monolithic solution in a similar manner as the Dirichlet boundary conditions; whereas in the mechanical case the convergence behavior is dependent on the boundary conditions imposed. Thus, we conclude that the network performance for thermal systems is not influenced by the type of boundary conditions imposed on the system.

3.2 Uncertainty Quantification Results

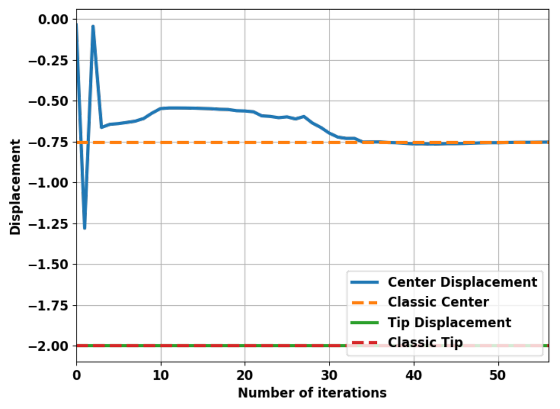
The results from the preceding section confirm that PyNetUQ provided the correct mean behavior and that the iterative solvers are (mostly) performing robustly. In this section, those results will be expanded upon to demonstrate that PyNetUQ is correctly propagating uncertainty through the system. We do this by examining both steady-state and transient results for linear heat conduction and a transient simulation of a complex thermal model which includes numerous



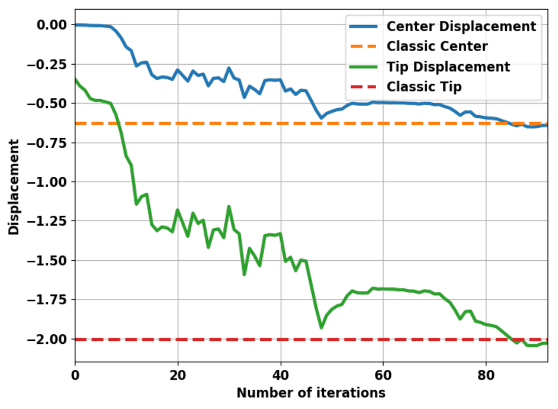
(a) Dirichlet condition with a final tip displacement of 1



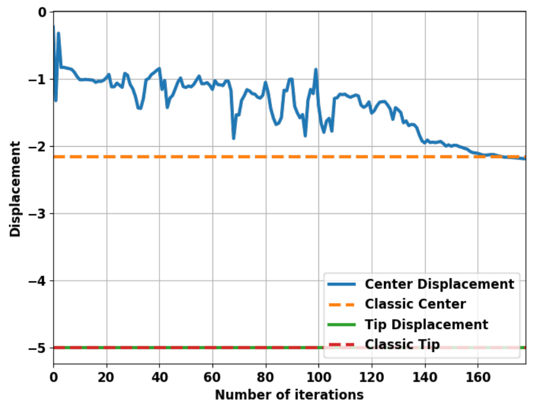
(b) Neumann condition with a final tip displacement of ≈ 1



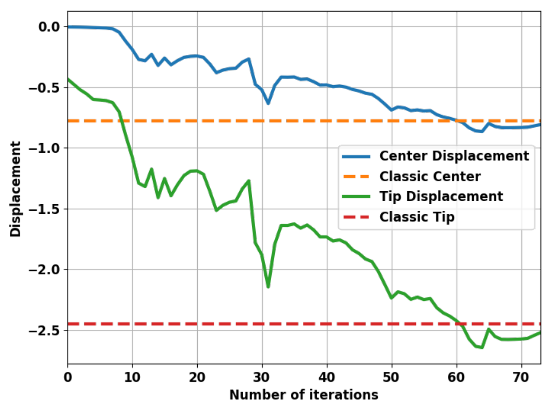
(c) Dirichlet condition with a final tip displacement of 2



(d) Neumann condition with a final tip displacement of ≈ 2



(e) Dirichlet condition with a final tip displacement of 5



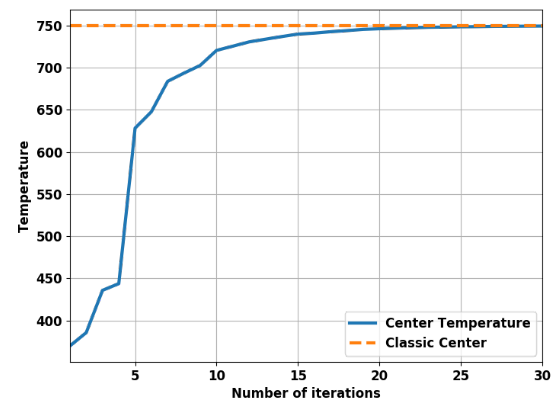
(f) Neumann condition with a final tip displacement of ≈ 2.5

Fig. 3: Dirichlet and Neumann boundary conditions for the mechanical case: (a) small Dirichlet condition, (b) small Neumann condition, (c) medium Dirichlet condition, (d) medium Neumann condition, (d) large Dirichlet Condition, (e) large Neumann Condition

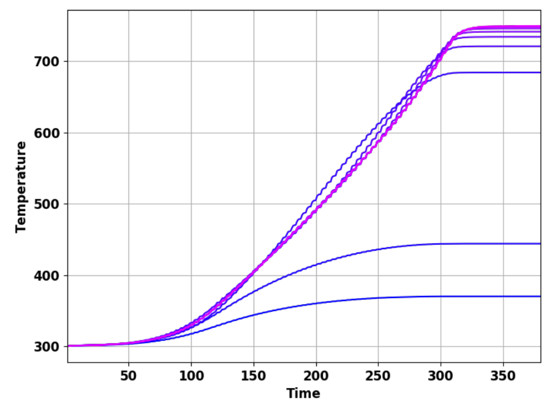
nonlinearities. In all cases, the uncertain input parameters are assumed to be Gaussian and PCE coefficients are determined through NISP using a full quadrature.

3.2.1 Quasi-1D steady-state thermal

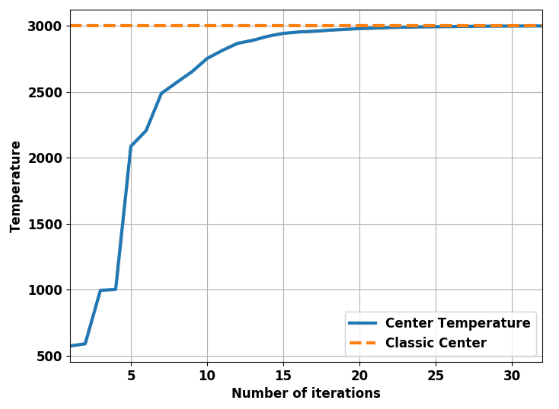
First, consider the 1D steady-state linear heat transfer problem defined by



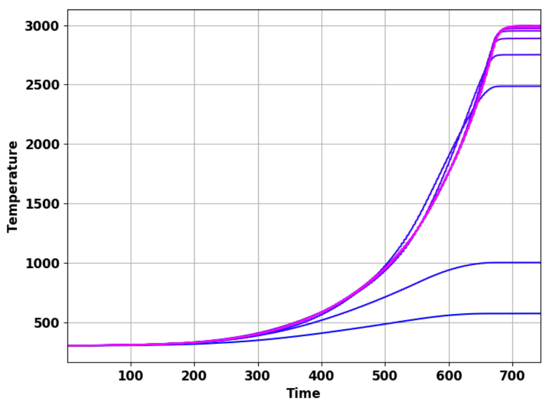
(a) Midpoint temperature at the final time, $U(L/2, 500)$



(b) Transient evolution of the midpoint temperature $U(L/2,t)$ at different iterations



(c) Midpoint temperature at the final time, $U(L/2, 500)$



(d) Transient evolution of the midpoint temperature $U(L/2,t)$ at different iterations

Fig. 4: Network convergence results for a Dirichlet boundary condition for the thermal case with $T_R = 500$ for (a) and (b) and $T_R = 5500$ for (c) and (d). For the transient plots, (b) and (d), dark blue lines are early iterations and later iterations approach a magenta color.

$$\begin{aligned} u_{xx} &= 0 \\ u(0) &= T_0 \\ u(L) &= T_L. \end{aligned} \tag{15}$$

The resulting solution is

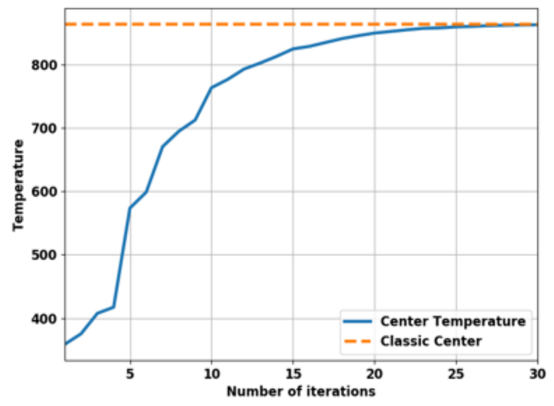
$$u(x) = T_0 + \frac{x}{L}(T_L - T_0). \tag{16}$$

Let $T_0 \sim \mathcal{N}(\mu = 1000, \sigma^2 = 40^2)$ and $T_L \sim \mathcal{N}(\mu = 500, \sigma^2 = 20^2)$. Our quantity of interest is the temperature at the midpoint of the slab,

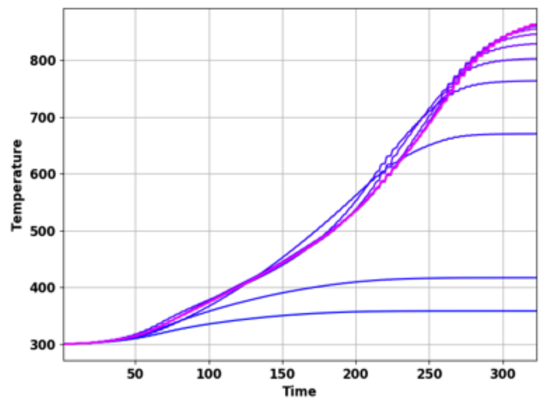
$$u(L/2) \sim \mathcal{N}\left(\mu = 750, \sigma^2 = \frac{40^2 + 20^2}{4}\right). \tag{17}$$

A coarse hex mesh (Figure 6a) is used to solve Eq. 15 numerically 3 different ways. First, the monolithic problem in which the heat equation is solved concurrently on all 3 blocks. Next, the domain is decomposed into two components. The first component consisting of the purple and gray blocks and the second component consisting of the orange and gray blocks. The gray block is the overlap region shared by both components. The decomposed problem is then solved using both the deterministic domain decomposition and NetUQ methods. All forward propagation steps are computed using SIERRA:Aria [20].

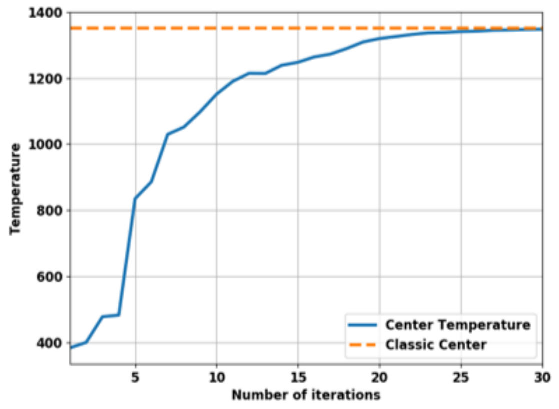
For the monolithic and deterministic domain decomposition approaches, a 3rd order Gauss-Hermite polynomial chaos is used to represent the uncertainty in the quantity of interest (the centerpoint temperature). The coefficients of the PCE are generated through NISP using a full quadrature with 4 points per stochastic dimension. For the NetUQ approach, both components use a 3rd order Gauss-Hermite polynomial chaos to represent the solution uncertainty at the endogenous



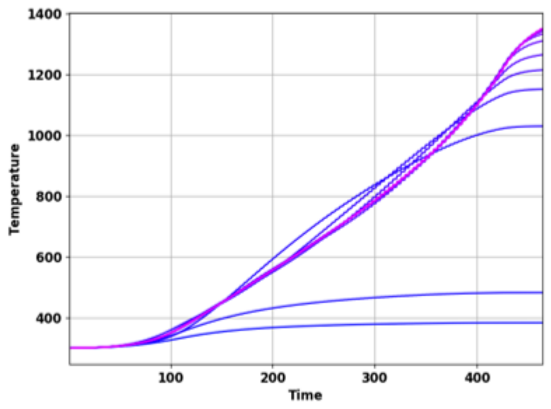
(a) Midpoint temperature at the final time, $U(L/2,500)$



(b) Transient evolution of the midpoint temperature $U(L/2,t)$ at different iterations

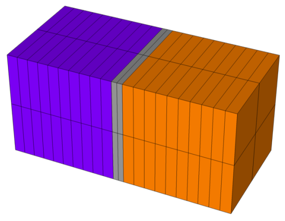


(c) Midpoint temperature at the final time, $U(L/2,500)$

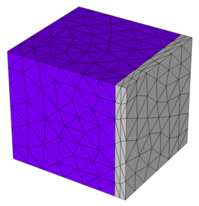


(d) Transient evolution of the midpoint temperature $U(L/2,t)$ at different iterations

Fig. 5: Network convergence results for a Neumann boundary condition for the thermal case with (a)(b) a negative flux applied to the right boundary and (c)(d) a positive flux applied to the right boundary. For the transient plots, (b) and (d), dark blue lines are early iterations and later iterations approach a magenta color.



(a)



(b)

Fig. 6: Meshes used for steady-state verification problems.

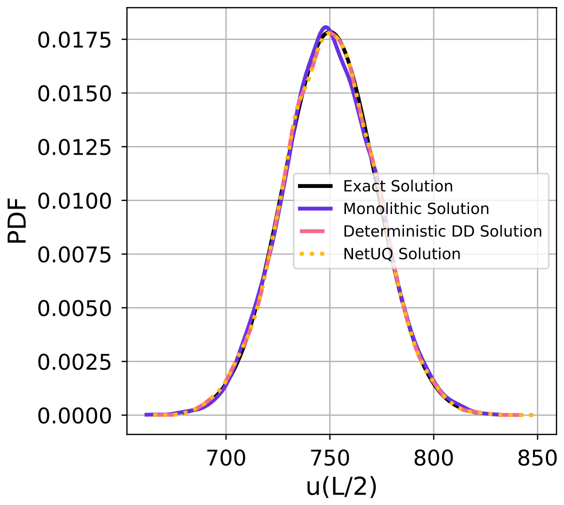
nodes. Since the QoI lies within the overlap region, a unique PCE for the QoI may be extracted from each component. However, since both components use the same order PCE these predictions are identical (within the convergence tolerance) as shown in Figure 7a.

To demonstrate the flexibility of the overlapping domain decomposition approaches, the mesh for the left (purple+gray) component is changed to a tet mesh (Figure 6b). The only nodes that are consistent between the hex and tet meshes are the ones at the 8 corners of the overlap region. Admittedly, for a 1D problem that's more than enough to uniquely specify the temperature throughout the overlap region, so this is a fairly weak test, but it does test the spatial interpolation between meshes. The results for this case are shown in Figure 7b.

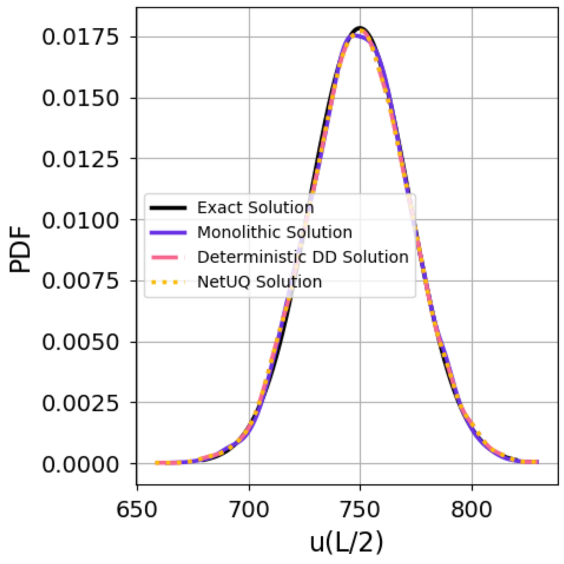
3.2.2 Transient Linear Heat Conduction

Consider now the analogous transient thermal problem. This is the same problem previously described in Section 3.1.2. Once again, let $T_0 \sim \mathcal{N}(\mu = 1000,\, \sigma^2 = 40^2)$ and $T_L \sim \mathcal{N}(\mu = 500,\, \sigma^2 = 20^2)$. Our quantity of interest is now the time evolution of the temperature at the midpoint of the slab, $u(L/2,t)$. The solution is given by

$$u(L/2,t) \sim \mathcal{N}\left(\mu = \mu(t),\, \sigma^2 = \sigma(t)^2\right), \qquad (18)$$



(a) Consistent meshes in the overlap region



(b) Inconsistent meshes in the overlap region

Fig. 7: QoI output PDFs from solving the steady-state linear heat conduction problem four different ways.

with

$$\mu(t) = 750 + \sum_{n=1}^{\infty} B_n C_n(L/2,t), \tag{19}$$

and

$$\begin{aligned}\sigma(t) &= \sqrt{\sigma_{T_0}^2 + \sigma_{T_L}^2} \\ \sigma_{T_0} &= 40\left(\frac{1}{2} - \sum_{n=1}^{\infty}\left(\frac{4}{n\pi} - \frac{2}{n\pi}(-1)^n\right)C_n(L/2,t)\right) \\ \sigma_{T_L} &= 20\left(\frac{1}{2} + \sum_{n=1}^{\infty}\frac{2}{n\pi}(-1)^n C_n(L/2,t)\right).\end{aligned} \tag{20}$$

As a demonstration, we solve this transient heat conduction problem with the more complicated network of components shown in Figure 8. The solution, given by Eq. 18 only varies in the x direction despite the higher-dimensional mesh and more complex network of components. We also adjust the time steps for the two components to be different to test the temporal interpolation.

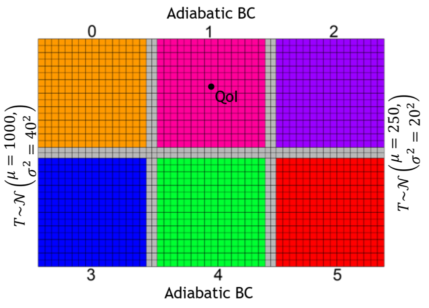
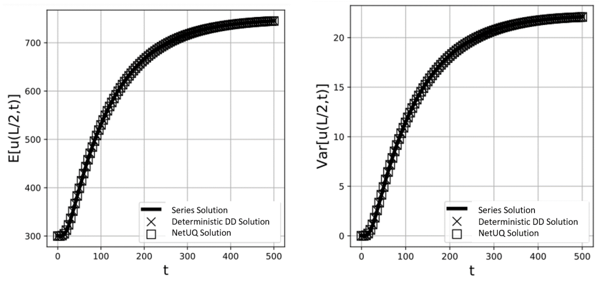

Adiabatic BC
0
1
2
QoI
$T \sim \mathcal{N}\left(\substack{\mu=1000,\\ \sigma^2=40^2}\right)$
$T \sim \mathcal{N}\left(\substack{\mu=250,\\ \sigma^2=20^2}\right)$
3
4
5
Adiabatic BC

Fig. 8: Network of 6 overlapping components.



(a) Expected Value (b) Variance

Fig. 9: Time evolution of the stochastic representation of the midpoint temperature, $u(L/2,t)$.

Figures 9a and 9b illustrate that the results for this problem are consistent with those generated with the decomposition shown in Figure 6a as well as the analytical series solution. This demonstrates the validity of the method for larger numbers of components that interact with each other in more complex ways.

3.2.3 U-Bomb

As a final demonstration, a large-scale exemplar problem was selected for evaluating the PyNetUQ capability. The PyNetUQ framework was applied to an abnormal thermal simulation using an unclassified exemplar bomb (U-Bomb) geometry shown in Figure 10 [21]. The U-Bomb is examined in two configurations: a simplified low-fidelity configuration and a detailed configuration with enhanced geometric fidelity. The low-fidelity configuration is represented by a network of 4 components (red, green, blue, and orange). The detailed U-Bomb geometry is divided into 7 components with the forward-most (green) component having been subdivided further into 4 separate components as shown in Figure 11. An uncertain radiative boundary condition is applied to the exterior surface to approximate a fully engulfing hydrocarbon fuel fire environment. The radiative boundary condition is assumed to follow a normal distribution $\mathcal{N}(\mu = 1000, \sigma^2 = 40^2)$.

The outer case is constructed of aluminum 7075. Components 1, 2, and 3 are potted with PMDI foam. An organic material decomposition model [22] is used for these blocks. Components 1a, 1b, and 1c are constructed of stainless steel 304L. All other unnamed internal potted components are constructed of either stainless steel 304L or a representative laminate with anisotropic thermal conductivity representative of a printed circuit board. The thermal properties of the aluminum and steel are treated as nonlinear functions of temperature.

Eight different QoIs are defined for the U-Bomb application corresponding to temperature-time histories at different locations of interest within the system. The locations of the different QoIs are shown in Figure 12. The minimum temperature is selected for the blocks colored purple while the maximum temperature is selected for blocks colored orange.

A workflow was developed in which a PyNetUQ model was constructed for the low-fidelity configuration and then updated to include the added geometric fidelity of the detailed configuration. The components were all meshed independently with no requirements for mesh consistency at the component interfaces (although approximate geometric consistency is required). Component 0 remained unchanged between configurations. The network was initially solved using the low-fidelity configuration. The low-fidelity configuration solution was then used to initialize the network for the detailed configuration. This procedure significantly reduced the number of iterations required to achieve convergence for the detailed configuration.

Due to the lack of a conformal mesh for these scenarios, no monolithic simulation is attempted for validation. Instead, the network model is analyzed using both the deterministic domain decomposition and NetUQ methods implemented in the PyNetUQ library. The results of this comparison are shown in Figure 12. The results agree very well including the variability. Where the solutions differ, it is expected that the NetUQ solution is more accurate due to the ability to apply a tighter convergence tolerance to the NetUQ simulation owing to its more rapid convergence relative to the deterministic domain decomposition method.

The faster convergence of the NetUQ method in this case is due primarily to the fact that Anderson acceleration is only implemented for the NetUQ method in the PyNetUQ library. Unfortunately, this lack of consistency within the library makes apples-to-apples timing comparisons impossible. However, we can assert that the primary advantage of the NetUQ approach relative to the deterministic domain decomposition approach is in terms of flexibility rather than performance. The NetUQ approach allows for unique uncertainty representations for each component which may allow for computational savings if e.g. a simple Gaussian is sufficient for capturing the response for a linear or nearly linear component. For this application, the same uncertainty representation was used for all components to be consistent with the results from the deterministic domain decomposition method.

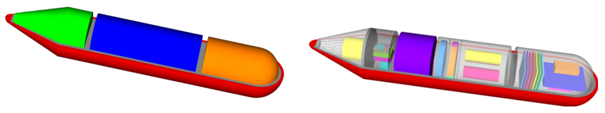

(a) Low-Fidelity (b) Detailed

Fig. 10: Unclassified bomb geometry. The U-Bomb is examined in two configurations: a simplified low-fidelity configuration (a) and a detailed configuration with enhanced geometric fidelity (b). The detailed U-Bomb geometry is divided into 7 components for network analysis.

The differences in QoI responses shown in Figure 12 are due to the thermal paths to and through the components, differences in thermal diffusivities of the blocks, and using block minimums versus maximums. The large temperature difference between QoIs seen in Figure 12a is largely due to the fact that the purple block being composed of a low-conductivity material, fiberite. As a result, the location of the minimum temperature of that block (which is surrounded by low-conductivity PMDI foam encapsulant) is extremely well-insulated. In contrast, the orange block is in intimate contact with the metallic case such that its maximum temperature more closely tracks the imposed boundary condition. Temperatures of the orange and purple blocks in Figure 12b are much more similar, with the difference being primarily attributed to the orange block using the maximum temperature and the purple block using the minimum. Both blocks shown in Figure 12c are fully suspended in foam encapsulant causing more muted responses. The difference in those blocks' responses is mainly due to a difference in thermal diffusivity. The component in Figure 12d is the orange block in Figure 12a, with QoIs now being internal blocks within the component. Note that the thermal responses of the orange blocks in Figures 12a and 12d are nearly identical due to the orange block in Figure 12d being in contact with the external face of the component. In order for heat to reach the purple block within Figure 12d, it primarily travels along a path through contacting blocks within the component which significantly delays heat reaching the purple block compared to the orange block that resides at the start of the path.

The magnitude of variability found for all responses largely tracks with the temperature reached. Due to nonlinearities within the model including internal radiation enclosures, temperature dependent thermal properties, and decomposing foam, it is expected that the QoI response distributions are non-Gaussian. Unfortunately the quadrature results were no longer available at the time of documentation, so this expected non-linear distribution propagation could not be confirmed.

4 Conclusions

A network uncertainty quantification technique was implemented and evaluated using a number of example applications. This involved the creation of the the open source PyNetUQ library to facilitate the creation of network models of surrogate problems of interest. These problems spanned

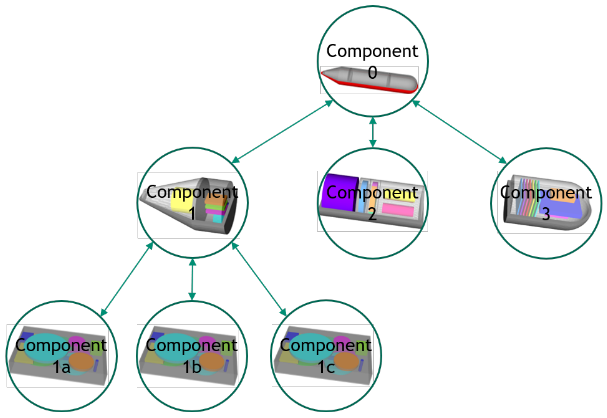


Fig. 11: Graph representation of detailed U-Bomb geometry

a wide range of complex physics, including thermal and mechanical simulations at a range of scales. The library itself represents a flexible tool for performing network-based uncertainty propagation and is integrated with a number of production codes. Additionally, it is easily extensible to be compatible with new physics applications.

Pseudo-1d steady-state and transient thermal problems were used to verify the network solver implementation and transient thermal and mechanical problems were used to evaluate the solver performance. For steady-state thermal transient thermal problems the network behaves robustly, but for transient solid mechanics problems the behavior is more complicated. For solid mechanics problems with fixed-displacement boundary conditions, the network converges robustly, but for fixed load boundary conditions, the accuracy of the converged solution is dependent on the load applied and the output frequency of the network components. Infrequent outputs and large loads result in inaccuracies due to the linear interpolation in time used between components. This difficulty may be overcome by increasing the output frequency during time periods with large temporal solution variations, particularly at early times.

Additionally, we applied the approach to a complex system exemplar model, the U-Bomb. This demonstrated the applicability of the approach to complex system-scale problems and highlights some advantages and challenges of the approach. The network approach essentially decouples the model development pipelines for different components and subassemblies. This results in increased agility and the potential to incorporate UQ into the fast-moving design cycle. Individual component geometries and models may be easily updated in a modular way, reducing dependencies and enabling an analyst access to a parallel model development paradigm. The network approach also directly ties plentiful component and subassembly validation evidence to full-system model predictions. This approach would result in fewer simplifications and a more robust credibility package.

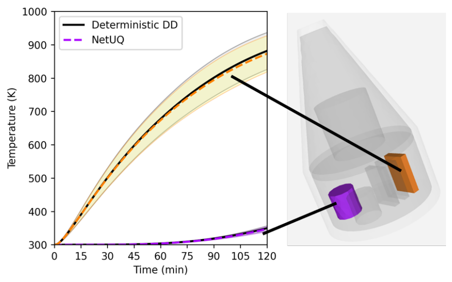
Unfortunately, the modularity and flexibility come at a cost. Network uncertainty quantification techniques are considerably slower (in terms of compute cost) than monolithic techniques. The advantage lies exclusively in the modularity and the potential to speed up model development at the expense of slower model execution. In many applications, this is a trade well worth making. Additionally, reduced-order and surrogate models provide an avenue to avoid this sacrifice in model execution speed [23–27].

Expensive component models may be seamlessly replaced by inexpensive surrogates within the network architecture without sacrificing accuracy. These surrogates may either be trained through a bottom-up approach without prior knowledge of the intercomponent boundary conditions or by utilizing partially converged network results.

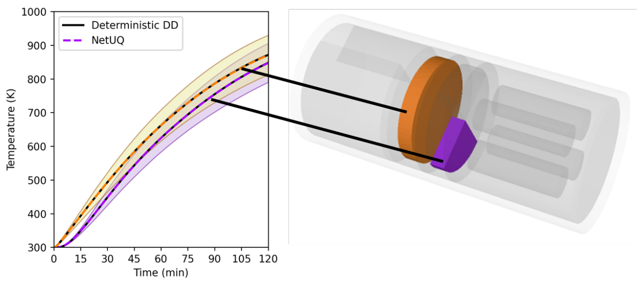
The network uncertainty propagation techniques are also inherently iterative. This is not itself a problem, but does pose additional challenges when operating within existing high performance computing (HPC) infrastructure. The current HPC paradigm assumes jobs to be independent and optimizes for high throughput (at the expense of latency). The iterative nature of these network UQ techniques introduces dependencies between jobs, and long queue times can be compounded. This can be circumvented by either packaging all of the jobs together when submitting to the HPC system or eschewing the HPC systems for alternative low-latency resources.

Acknowledgements

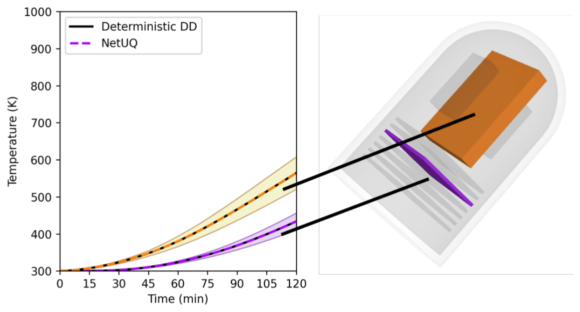
This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. This work was funded by the Advanced Simulation and Computing program and the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and 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-NA-0003525.



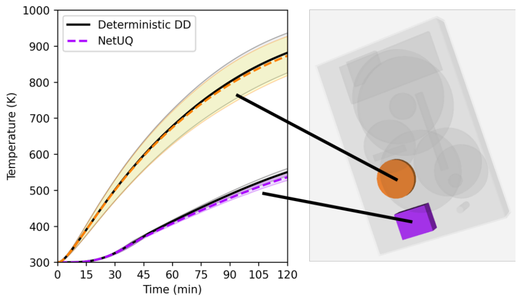
(a)



(b)



(c)



(d)

Fig. 12: Average response and 1-σ uncertainty bounds for 8 different quantities of interest for the U-Bomb problem computed using both the deterministic domain decomposition and NetUQ approaches. For each component, the minimum temperature over the purple block provides one quantity of interest, while the maximum temperature over the orange block provides another.

References

[1] Carlberg, K., Guzzetti, S., Khalil, M., and Sargsyan, K., 2019. "The network uncertainty quantification method for propagating uncertainties in component-based systems". *arXiv:1908.11476 [math.NA]*.

[2] Liao, Q., and Willcox, K., 2015. "A domain decomposition approach for uncertainty analysis". *SIAM Journal on Scientific Computing,* **37**(1), pp. A103–A133.

[3] Martin, J. D., and Simpson, T. W., 2006. "A methodology to manage system-level uncertainty during conceptual design". *Journal of Mechanical Design,* **128**(4), pp. 959–968.

[4] Rojas, E., and Tencer, J., 2021. "Performance of iterative network uncertainty quantification for multicomponent system qualification". In ASME International Mechanical Engineering Congress and Exposition, Vol. 85697, American Society of Mechanical Engineers, p. V013T14A031.

[5] Le Maître, O., and Knio, O. M., 2010. *Spectral methods for uncertainty quantification: with applications to computational fluid dynamics*. Springer Science & Business Media.

[6] Ghanem, R. G., and Spanos, P. D., 2003. *Stochastic finite elements: a spectral approach*. Courier Corporation.

[7] Xiu, D., and Karniadakis, G. E., 2002. "The Wiener–Askey polynomial chaos for stochastic differential equations". *SIAM Journal on Scientific Computing,* **24**(2), pp. 619–644.

[8] Xiu, D., 2010. "Numerical methods for stochastic computations". In *Numerical Methods for Stochastic Computations*. Princeton University Press.

[9] Smith, B. F., 1997. "Domain decomposition methods for partial differential equations". In *Parallel Numerical Algorithms*. Springer, pp. 225–243.

[10] Chan, T. F., and Mathew, T. P., 1994. "Domain decomposition algorithms". *Acta Numerica,* **3**, pp. 61–143.

[11] Toselli, A., and Widlund, O., 2004. *Domain decomposition methods-algorithms and theory*, Vol. 34. Springer Science & Business Media.

[12] Sargsyan, K., Safta, C., Johnston, K., Khalil, M., Chowdhary, K. S., Rai, P., Casey, T. A., Boll, L. D., Zeng, X., and Debusschere, B., 2021. UQTk version 3.1.1 user manual sand2021-3655. Tech. rep., Sandia National Laboratories.

[13] Yang, X. I., and Mittal, R., 2014. "Acceleration of the Jacobi iterative method by factors exceeding 100 using scheduled relaxation". *Journal of Computational Physics,* **274**, pp. 695–708.

[14] Pratapa, P. P., Suryanarayana, P., and Pask, J. E., 2016. "Anderson acceleration of the Jacobi iterative method: An efficient alternative to Krylov methods for large, sparse linear systems". *Journal of Computational Physics,* **306**, pp. 43–54.

[15] Eyert, V., 1996. "A comparative study on methods for convergence acceleration of iterative vector sequences". *Journal of Computational Physics,* **124**(2), pp. 271–285.

[16] Anderson, D. G., 1965. “Iterative procedures for nonlinear integral equations”. *Journal of the ACM (JACM),* **12**(4), pp. 547–560.

[17] Fang, H.-r., and Saad, Y., 2009. “Two classes of multisecant methods for nonlinear acceleration”. *Numerical Linear Algebra with Applications,* **16**(3), pp. 197–221.

[18] Toth, A., and Kelley, C., 2015. “Convergence analysis for Anderson acceleration”. *SIAM Journal on Numerical Analysis,* **53**(2), pp. 805–819.

[19] Walker, H. F., and Ni, P., 2011. “Anderson acceleration for fixed-point iterations”. *SIAM Journal on Numerical Analysis,* **49**(4), pp. 1715–1735.

[20] SIERRA Thermal/Fluid Development Team, 2019. SIERRA Multimechanics Module: Aria User Manual - Version 4.52 SAND2019-3786. Tech. rep., Sandia National Laboratories.

[21] Schroeder, B., Hetzler, A., Mills, B., and Shelton, J., 2018. An effort towards a consistent VVUQ approach for thermal systems analyses SAND2018-5411 PE. Tech. rep., Sandia National Laboratories.

[22] Scott, S. N., Dodd, A. B., Larsen, M. E., Suo-Anttila, J. M., and Erickson, K. L., 2016. “Validation of heat transfer, thermal decomposition, and container pressurization of polyurethane foam using mean value and latin hypercube sampling approaches”. *Fire Technology,* **52**(1), pp. 121–147.

[23] Benner, P., Gugercin, S., and Willcox, K., 2015. “A survey of projection-based model reduction methods for parametric dynamical systems”. *SIAM Review,* **57**(4), pp. 483–531.

[24] Peherstorfer, B., Willcox, K., and Gunzburger, M., 2018. “Survey of multifidelity methods in uncertainty propagation, inference, and optimization”. *Siam Review,* **60**(3), pp. 550–591.

[25] Brunini, V., Parish, E. J., Tencer, J., and Rizzi, F., 2022. “Projection-based model reduction for coupled conduction—enclosure radiation systems”. *Journal of Heat Transfer,* **144**(6).

[26] Gelsomino, F., and Rozza, G., 2011. “Comparison and combination of reduced-order modelling techniques in 3d parametrized heat transfer problems”. *Mathematical and Computer Modelling of Dynamical Systems,* **17**(4), pp. 371–394.

[27] Brunton, S. L., and Kutz, J. N., 2022. *Data-driven science and engineering: Machine learning, dynamical systems, and control.* Cambridge University Press.