This is the preprint version of the following article:
Bingran Wang, Mark Sperry, Victor E. Gandarillas, and John T. Hwang. Accelerating model evaluations in uncertainty propagation on tensor grids using computational graph transformations. Aerospace Science and Technology, 2024.

Published article: https://doi.org/10.1016/j.ast.2023.108843
Preprint pdf: https://github.com/LSDOlab/lsdo_bib/blob/main/pdf/wang2024accelerating.pdf
Bibtex: https://github.com/LSDOlab/lsdo_bib/blob/main/bib/wang2024accelerating.bib

Accelerating model evaluations in uncertainty propagation on tensor grids using computational graph transformations

Bingran Wang*, Mark Sperry†, Victor E. Gandarillas‡, and John T. Hwang§

University of California San Diego, La Jolla, CA, 92093

Abstract

Methods such as non-intrusive polynomial chaos (NIPC), and stochastic collocation are frequently used for uncertainty propagation problems. Particularly for low-dimensional problems, these methods often use a tensor-product grid for sampling the space of uncertain inputs. A limitation of this approach is that it encounters a significant challenge: the number of sample points grows exponentially with the increase of uncertain inputs. Current strategies to mitigate computational costs abandon the tensor structure of sampling points, with the aim of reducing their overall count. Contrastingly, our investigation reveals that preserving the tensor structure of sample points can offer distinct advantages in specific scenarios. Notably, by manipulating the computational graph of the targeted model, it is feasible to avoid redundant evaluations at the operation level to significantly reduce the model evaluation cost on tensor-grid inputs. This paper presents a pioneering method: Accelerated Model Evaluations on Tensor grids using Computational graph transformations (AMTC). The core premise of AMTC lies in the strategic modification of the computational graph of the target model to algorithmically remove the repeated evaluations on the operation level. We implemented the AMTC method within the compiler of a new modeling language called the Computational System Design Language (CSDL). We demonstrate the effectiveness of AMTC by using it with the full-grid NIPC method to solve four low-dimensional UQ problems involving an analytical piston model, a multidisciplinary unmanned aerial vehicle design model, a multi-point air taxi mission analysis model, and a single-disciplinary rotor model, respectively. For three of the four test problems, AMTC reduces the model evaluation cost by between 50% and 90%, making the full-grid NIPC the most efficacious method to use among the UQ methods implemented.

*Ph.D Candidate, Department of Mechanical and Aerospace Engineering.
†Ph.D Student, Department of Mechanical and Aerospace Engineering.
‡Ph.D Candidate, Department of Mechanical and Aerospace Engineering.
§Assistant Professor, Department of Mechanical and Aerospace Engineering.

1 Introduction

In practical scenarios, the numerical models inherently fall short of achieving perfect fidelity with reality and consistently contend with inherent uncertainties. To address this challenge, uncertainty propagation, also known as forward uncertainty quantification (UQ), aims to undertake a rigorous examination and analysis of how a system's output is influenced by the uncertainties inherent in its inputs. The resultant assessments of uncertainty serve as a valuable foundation for facilitating informed decision-making and thorough risk assessment. The applications of UQ can be found in many scientific and engineering domains such as weather forecast [1, 2], structural analysis [3, 4] and aircraft design [5–7].

Uncertainties within the inputs can stem from diverse origins and are typically classified into two categories: aleatoric and epistemic. Aleatoric uncertainties, recognized as irreducible uncertainties, are intrinsic to the system and encompass factors such as fluctuations in operational conditions, mission prerequisites, and model parameters. These uncertainties can often be described within a probabilistic framework. In contrast, epistemic uncertainties constitute reducible uncertainties arising from knowledge gaps. These uncertainties may be induced by approximations utilized in computational models or numerical solution methods, and are often difficult to be described within a probabilistic framework. This paper focuses on UQ problems within a probabilistic formalism where uncertain inputs are continuous random variables with known probability density distributions. The goal is to estimate statistical moments such as the mean and variance of the Quantity of Interest (QoI), or more complex risk measures such as the probability of failure or the conditional value at risk.

For this type of UQ problem, the most common non-intrusive methods include the method of moments, polynomial chaos, kriging, and Monte Carlo. The method of moments analytically computes the statistical moments of the QoI, using the Taylor series approximation. It often only uses function evaluations and derivative information at one or a small number of input points [8]. The most commonly employed variants of this method utilize derivatives up to either first [9] or second order [10, 11] to ensure cost-effectiveness. While adept at efficiently estimating statistical moments, this approach grapples with challenges in estimating other kinds of risk measures. Moreover, it may lack accuracy when input variance is significant. In contrast, the Monte Carlo approach employs random sampling of uncertain inputs to compute the risk measures. Based on the law of large numbers, Monte Carlo's convergence rate is independent of the number of uncertain inputs, making it particularly attractive for solving high-dimensional problems. Recent strides in enhancing its efficiency encompass multi-fidelity [12, 13] and importance sampling [14] techniques. However, in the context of low-dimensional problems, the Monte Carlo method might demand substantially more model evaluations to achieve the same level of accuracy as the alternative UQ methodologies. Kriging, also known as Gaussian process regression, is a common statistical technique employed in various fields. In UQ, kriging constructs a surrogate response surface by leveraging input-output data pairs. The surrogate model replaces the original expensive function to allow a large number of model evaluations in order to perform reliability analysis [15, 16] or optimization under uncertainty [17]. For low-dimensional UQ problems, if the objective function is assumed to be smooth, the polynomial chaos-based techniques often stand out as the most efficacious option. The generalized polynomial chaos theory represents the QoI as orthogonal polynomials that are derived from the distributions of the uncertain inputs. Benefiting from the inherent smoothness of the random space, polynomial chaos-based methods exhibit rapid convergence rates facilitated by sampling or integration techniques. Common polynomial chaos-based methods include non-intrusive

polynomial chaos (NIPC) [18–20] and stochastic collocation (SC) [21, 22].

For both NIPC and SC methods, we can use a tensor-product grid to sample the input space, but this approach is often not preferred and is only used for very low-dimensional UQ problems, as the number of input points increases exponentially with the number of uncertain inputs. The commonly used methods choose to use a sparse grid to sample the input space or have the sample points randomly generated, in order to reduce the input points with a minimum loss of accuracy. However, our recent findings show that maintaining a tensor structure for the input points offers some unique advantages and for certain problems, the total model evaluation cost can be reduced in a different approach without changing the tensor structure of the input points.

When we evaluate a computational model on full-grid input points, if we view the computational model as a computational graph with only elementary operations, the current framework evaluates each operation the same number of times at the tensor-product input points of all the uncertain inputs. However, for each operation, the output data only has distinct values at the distinct input points in their dependent input space, and a large portion of the operations may not be dependent on all of the uncertain inputs. This means the current framework of NIPC and SC could create many wasteful evaluations on a large portion of the operations.

In this paper, we propose a computational graph transformation method called Accelerated Model evaluations on Tensor grids using Computational graph transformations (AMTC) that algorithmically modifies the computational graph to eliminate the unnecessary evaluations incurred by the current framework of integration-based NIPC and SC. AMTC reduces the model evaluation cost on full-grid quadrature points by partitioning the computational graph into sub-graphs using the operations' dependency information and evaluating each sub-graph on the distinct quadrature points. The proposed method is implemented in conjunction with a new algebraic modeling language, the Computational System Design Language (CSDL). We achieve cost reductions on UQ problems in a general and automatic way by analyzing and manipulating the computational graph that CSDL makes available.

This method has been applied to four different UQ problems with different computational graph structures. For three of the problems, we observed a significant acceleration in model evaluation time when solving the UQ problems using the full-grid NIPC method, which makes this UQ method the most efficient method among the implemented UQ methods for these problems. Generally, for a wide range of UQ problems that involve multi-point, multi-disciplinary, or other models that possess a sparse graph structure, we expect our method to significantly reduce the model evaluation time on tensor-grid input points, which can improve the time scaling for, but not limited to NIPC and SC methods.

This paper is organized as follows. Section 2 gives some background on PCE-related UQ methods, the current model evaluation framework, and how to view tensor-grid evaluations through computational graphs. Section 3 presents the details of the new computational graph transformation algorithm, AMTC, and how it is implemented in CSDL. Section 4 shows the numerical results of three UQ problems. Section 5 summarizes the work and offers concluding thoughts.

2 Background

2.1 Polynomial chaos expansion

Wiener initially introduced the concept of polynomial chaos expansion (PCE), utilizing Hermite polynomials to model the response of a system affected by Gaussian uncertain inputs [23]. This foundational concept was later advanced into the generalized polynomial chaos (gPC) method

by Xiu and Karniadakis [24]. Within the framework of gPC, model responses are represented through polynomial series of uncertain inputs. These polynomials are meticulously chosen to exhibit orthogonality with respect to the probability distribution of the uncertain inputs, thereby ensuring exponential convergence.

We address a general problem in uncertainty quantification (UQ) involving a function defined as follows:

$$f = \mathcal{F}(u), \tag{1}$$

where $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ represents the model evaluation function, $u \in \mathbb{R}^d$ denotes the input vector, and $f \in \mathbb{R}$ represents a scalar output. The uncertainties associated with the inputs are expressed as a stochastic vector denoted by $U := [U_1, \ldots, U_d]$, under the assumption that these random variables are mutually independent. The stochastic input vector adheres to the probability distribution $\rho(u)$ with its support defined by Γ. Our focus centers on the evaluation of risk measures of the output random variable, $f(U)$.

In gPC, the stochastic output, $f(U)$, is defined as an infinite series of orthogonal polynomials with respect to the uncertain input variables:

$$f(U) = \sum_{i=0}^{\infty} \alpha_i \Phi_i(U), \tag{2}$$

where $\Phi_i(U)$ are the PCE basis functions that are generated based on $\rho(U)$, and α_i are the corresponding weights that need to be determined.

Choosing the PCE basis functions is important to ensure the rapid convergence of PCE-based methods. These PCE basis functions have to satisfy the orthogonality property,

$$\langle \Phi_i(U), \Phi_j(U) \rangle = \delta_{ij}, \tag{3}$$

where δ_{ij} is Kronecker delta and the inner product is defined as

$$\langle \Phi_i(U), \Phi_j(U) \rangle = \int_{\Gamma} \Phi_i(u) \Phi_j(u) \rho(u) du. \tag{4}$$

In one-dimensional problems, we can directly use the univariate orthogonal polynomials as PCE basis functions. In Table 1, we show the univariate orthogonal polynomials for common types of continuous random variables. In multi-dimensional problems, if the input variables are mutually independent, the PCE basis functions can be formed as a tensor-product of the univariate orthogonal polynomials corresponding to each uncertain input. To elucidate this, we introduce the concept of multi-index notation and define a p-tuple as

$$i' = (i_1, \ldots, i_p) \in \mathbb{N}_0^p, \tag{5}$$

with its magnitude determined as

$$|i'| = i_1 + \ldots + i_p. \tag{6}$$

Employing this notation, we denote the collection of univariate PCE basis functions for variable u_k as $\{\phi_i(u_k)\}_{i=0}^p$. Then, the multivariate PCE basis with an upper limit on the total degree of p can be expressed as

$$\Phi_{\boldsymbol{i'}}(u) = \phi_{i_1}(u_1) \ldots \phi_{i_d}(u_d), \quad |i'| \leq p. \tag{7}$$

Table 1: Orthogonal polynomials for common types of continuous random variables

Distribution	Orthogonal polynomials	Support range
Normal	Hermite	$(-\infty, \infty)$
Uniform	Legendre	$[-1, 1]$
Exponential	Laguerre	$[0, \infty)$
Beta	Jacobi	$(-1, 1)$
Gamma	Generalized Laguerre	$[0, \infty)$

In practice, one may use the PCE basis functions in (7) to truncate the infinite series in (2), resulting in:

$$f(U) \approx \sum_{i=0}^{q} \alpha_i \Phi_i(U). \tag{8}$$

The resultant number of PCE basis functions, $q+1$, satisfies:

$$q + 1 = \frac{(d+p)!}{d!p!}. \tag{9}$$

Once the PCE coefficients are estimated, it becomes straightforward to compute statistical moments such as the mean and standard deviation of the model output. These calculations can be expressed as follows:

$$\mu_f = \alpha_0, \tag{10}$$

$$\sigma_f = \sum_{i=1}^{d} \alpha_i^2. \tag{11}$$

For risk measures like the probability of failure, the truncated PCE model can be treated as a surrogate model, and methods like Monte Carlo can be applied on the surrogate model to compute the desired risk measure.

2.1.1 Non-intrusive polynomial chaos method

The non-intrusive polynomial chaos (NIPC) method solves the PCE coefficients α_i in (8) by either integration or regression. The integration approach uses the orthogonal property in (3) and projects the model output onto each basis function:

$$\alpha_i = \frac{\langle f(U), \Phi_i \rangle}{\langle \Phi_i^2 \rangle} = \frac{1}{\langle \Phi_i^2 \rangle} \int_u f(u) \Phi_i(u) \rho(u) du. \tag{12}$$

This requires solving a multi-dimensional integration problem, which is often estimated by using the Gauss quadrature approach:

$$\begin{aligned} \alpha_i &= \frac{1}{\langle \Phi_i^2 \rangle} \int_{u_1} \dots \int_{u_d} f(u_1, \dots, u_d) \Phi_k(u_1, \dots, u_d) \rho(u_1, \dots, u_d) du_1 \dots du_d \\ &\approx \frac{1}{\langle \Phi_i^2 \rangle} \sum_{i_1=0}^{k} \dots \sum_{i_d=0}^{k} w_1^{(i_1)} \dots w_d^{(i_d)} f(u_1^{(i_1)}, \dots, u_d^{(i_d)}) \Phi_i(u_1^{(i_1)}, \dots, u_d^{(i_d)}), \end{aligned} \tag{13}$$

where $(u_i^{(1)}, \ldots, u_i^{(k)})$ and $(w_i^{(1)}, \ldots, w_i^{(k)})$ are the nodes and weights for the 1D Gauss quadrature rule in the u_i dimension, k is the number of quadrature points used in each dimension. The 1D quadrature rule with k nodes is chosen such that the quadrature rule can exactly integrate 1D polynomials up to $(2k-1)$th order. In the multi-dimensional case, the Gauss quadrature rule forms a tensor product of the 1D quadrature rule, which requires us to evaluate the model output at the tensor-product quadrature points, defined as

$$\boldsymbol{u} = \boldsymbol{u}_1^k \times \cdots \times \boldsymbol{u}_d^k, \tag{14}$$

where

$$\boldsymbol{u}_i^k := \{u_i^{(j)}\}_{j=1}^k. \tag{15}$$

The total number of quadrature points is thus k^d, and this number increases exponentially with respect to the number of uncertain input variables. For high-dimensional problems, one way to mitigate the exponential growth of quadrature points is to use the Smolyak sparse grid approach [25], which drops the higher-order cross terms in the quadrature points with minimal loss of accuracy. The sparse grid quadrature points can be expressed as:

$$\boldsymbol{u} = \bigcup_{\ell-d+1 \le |i| \le \ell} \left(\boldsymbol{u}_1^{i_1} \times \cdots \times \boldsymbol{u}_d^{i_d} \right), \tag{16}$$

where ℓ is the level of construction. The existing variations of sparse-grid methods can be found in [26]. Another way to reduce the total number of quadrature points is to use a designed quadrature approach. In [27], Keshavarzzadeh et al. formulate an optimization problem to optimize for the designed quadrature points and the corresponding weights that ensure the exact integration on a polynomial subspace. The number of designed quadrature points can be significantly smaller than the full-grid quadrature points for solving the high-dimensional integration problems for the same level of accuracy.

The regression approach first generates n sample points for U and evaluates the model for each sample point. The coefficients are determined by solving a linear least-squares problem:

$$\begin{bmatrix} \Phi_1(u^{(1)}) & \ldots & \Phi_q(u^{(1)}) \\ \vdots & & \vdots \\ \Phi_1(u^{(n)}) & \ldots & \Phi_q(u^{(n)}) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_q \end{bmatrix} = \begin{bmatrix} f(u^{(1)}) \\ \vdots \\ f(u^{(n)}) \end{bmatrix}. \tag{17}$$

The rule of thumb is that the number of samples n needs to be 2-3 times the number of coefficients q. This means the cost of the regression-based NIPC method can be reduced by using the sparse PCE basis, resulting in fewer coefficients to be estimated. Recent advances focus on adaptive basis selection and adaptive sampling[28–30].

2.2 Model evaluation cost for current non-intrusive UQ methods

If we consider only the non-adaptive approaches, the current framework for non-intrusive UQ methods includes three steps:

1. Generate sample/quadrature points based on the distribution of uncertain inputs.
2. Evaluate the target model for each of the sample points.

3. Post-process the output points to compute the desired quantities of the output.

Step one generates the input points based on the distribution of uncertain inputs. For methods like regression-based NIPC, Monte Carlo and kriging, the input points are often generated randomly (e.g. random sampling, Latin hypercube sampling). For integration-based NIPC and SC methods, they can be generated using the full/sparse-grid strategy. In step two, the target model is repeatedly evaluated at each of the input points. For some models that support vectorized input, the input points can be vectorized and evaluated at the same time. However, most of the time, the target model needs to be run in a for-loop by evaluating one input point at a time. In step three, the model output points are used to estimate the PCE coefficients for NIPC, form Lagrange interpolation functions for SC, or construct Gaussian process model for kriging, before computing the desired quantity for the output.

Under this framework, the model evaluation cost for any UQ method can be approximated as:

$$\text{cost} \approx nO(\mathcal{F}(u)), \tag{18}$$

where n is the number of sample points and $O(\mathcal{F}(u))$ is the model evaluation cost for one sample point. When we use the full-grid approach for integration-based NIPC or SC to generate the input points, the number of input points increases exponentially with the dimension of uncertainty inputs as $n = k^d$. Under the current framework to evaluate the model, the evaluation cost follows Eq. 18 and thus increases exponentially as the number of uncertain inputs, thus suffers the curse of dimensionality.

The current methods (e.g. sparse-grid and designed quadrature) to overcome the curse of dimensionality all focus on reducing the number of input points, n, by breaking the full-tensor structure of the input points, so that the input points do not increase exponentially as the number of the uncertain inputs, but none tries to break the linear relationship in Eq. 18 to reduce the evaluation cost on tensor-grid input points by using a computational graph transformation approach.

2.3 Viewing tensor-grid evaluations via computational graph

Any computer program that describes a computational model is a computational process that executes a sequence of elementary functions and operations. A computational process can be described by a computational graph. A computational graph is a directed acyclic graph in which the nodes represent the operations and variables, and the directed edges represent how operations and variables are connected to each other. Such a graph is also bipartite because each operation is connected to an output variable, and likewise, variables are connected to operations (as arguments). For example, the computer program that evaluates a function $f = cos(u_1) + exp(-u_2)$ can be decomposed into the following set of fundamental operations:

$$\begin{aligned}
\xi_1 &= \varphi_1(u_1) = cos(u_1); \\
\xi_2 &= \varphi_2(u_2) = -u_2; \\
\xi_3 &= \varphi_3(\xi_2) = exp(\xi_2); \\
f &= \varphi_4(\xi_1, \xi_3) = \xi_1 + \xi_3,
\end{aligned} \tag{19}$$

The computational graph for this model is shown in Fig. 1 with the blue nodes representing the variables and the orange nodes representing the operations. A common use of computational graphs is in automatic differentiation (AD) [31, 32] to automatically compute the total derivatives. AD

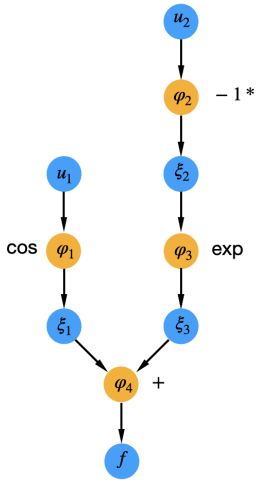


Figure 1: Computational graph of function $f = cos(u_1) + exp(-u_2)$

views the computer program as a sequence of operations and functions and repeatedly applies the chain rule to compute the total derivative. Another use of the computational graph is to modify the computational graph to reduce the time of memory cost of the model. In Tensorflow [33], a graph optimization method is implemented to reduce memory usage and evaluation costs for neural network models. It modifies the computational graph through constant folding, arithmetic simplification, and optimization.

If we view the model as a computational graph, in the current UQ framework, each operation and function is evaluated the same number of times as the number of input points. However, this could create many unnecessary evaluations when we evaluate the model on tensor-grid input points. For example, consider a UQ problem involving the simple function we showed before, given by

$$f = \cos(u_1) + \exp(-u_2). \tag{20}$$

In this problem, U_1 and U_2 are independent uncertain inputs, and we aim to compute the random variable $f(U_1, U_2)$. The computational process that describes this function is in (19). If we choose the full-grid quadrature points with k quadrature points in each dimension, we need to evaluate this function at a total of k^2 quadrature points which are

$$\mathbf{u} = \begin{Bmatrix} (u_1^{(1)}, u_2^{(1)}) & \dots & (u_1^{(k)}, u_2^{(1)}) \\ \vdots & \ddots & \vdots \\ (u_1^{(1)}, u_2^{(k)}) & \dots & (u_1^{(k)}, u_2^{(k)}) \end{Bmatrix}. \tag{21}$$

For both inputs u_1 and u_2, if we view their input points as vectors, they are in a tensor-product

form between the quadrature points in its dimension and a vector of ones:

$$
\begin{aligned}
u_1 &= \mathrm{vec}\left(\begin{bmatrix} u_1^{(1)} & \dots & u_1^{(k)} \\ \vdots & \ddots & \vdots \\ u_1^{(1)} & \dots & u_1^{(k)} \end{bmatrix}\right) = \mathrm{vec}\left([u_1^{(1)}, \dots u_1^{(k)}] \otimes [1, \dots, 1]\right), \\
u_2 &= \mathrm{vec}\left(\begin{bmatrix} u_2^{(1)} & \dots & u_2^{(1)} \\ \vdots & \ddots & \vdots \\ u_2^{(k)} & \dots & u_2^{(k)} \end{bmatrix}\right) = \mathrm{vec}\left([1, \dots, 1] \otimes [u_2^{(1)}, \dots u_2^{(k)}]\right).
\end{aligned} \tag{22}
$$

Even though both vectors have a size of k^2, there are only k distinct values, which are the values of the k quadrature points in its dimension. In the current framework, we can view the model evaluation step as propagating the input vectors in (22) through the model's computational graph, which is shown in Fig. 1. When we view the propagation of input vectors from an operational level, the current framework creates many repeated evaluations. For example, when we evaluate the first operation node in the computational graph, $\xi_1 = \cos(u_1)$, we will evaluate this operation for k^2 points, since there are k^2 input points for u_1. However, out of the k^2 evaluations, there are at most k distinct evaluations and thus the current framework creates $k^2 - k$ wasteful repeated evaluations in this case. Similarly, for $\xi_2 = -1 * u_2$ and $\xi_3 = \exp(\xi_2)$, both of the operations' outputs are only dependent on u_2, and they will also have at most k distinct values to be evaluated for. The current framework also creates $k^2 - k$ repeated evaluations for these operations. In contrast, for the last operation node, $f = \xi_1 + \xi_3$, there are k^2 distinct values to be evaluated as f is dependent on both u_1 and u_2.

In practical UQ problems, we could have a large computational graph with a number of uncertain inputs, but some of the inputs may only affect a small part of the computational graph. For example, for a multidisciplinary model, we may have uncertain inputs coming from different disciplines. When we view the computational graph for the whole model, there may be a small portion of operations that are dependent on some uncertain inputs. In this case, we could reduce the total computational cost on the tensor-grid inputs by eliminating the repeated evaluation cost for each operation node but the output data still carry the necessary results to compute the results of the QoI.

The logic here is simple and straightforward. One can manually modify the size of the input and state variables in the model to achieve this reduction if the model has a simple graph structure. However, for a complicated model with a large number of uncertainties, it becomes impractical to manually change the code to perform only necessary evaluations and match the shape of input for each node in the computational process.

3 Methodology

Here, we present the *Accelerated Model evaluations on Tensor grids using Computational graph transformations* (AMTC) method. The goal of our method is to enable automatic generation of a modified computational graph, given a computer program, that eliminates all wasteful evaluations due to tensor-product sampling within a NIPC or SC algorithm. The modified computational graph leverages the graph structure of the target model and enables performing the minimal number of evaluations for each operation. To achieve this, we modify the size of input data that are passed into the operations.

Table 2: Operations dependency information for function $f = cos(u_1) + exp(-u_2)$

$D(\varphi_i, u_j)$	u_1	u_2
φ_1	1	0
φ_2	0	1
φ_3	0	1
φ_4	1	1

In the UQ context, when we evaluate the computational model on full-grid quadrature points, we are forming a tensor product grid from the quadrature points in each uncertain input dimension. If we use k quadrature points in each dimension, we end up having k^d number of input points for d uncertain inputs. For each dimension, these k quadrature points are determined based on the probability distribution of that uncertain input. When we view the target model monolithically, the output is dependent on all of the d uncertain inputs. In the d-dimensional uncertain input space, we have a total of k^d quadrature points, and it is required for us to evaluate the model output on all k^d input points to gather all of the necessary information to determine the effect of the uncertain inputs on the model output for UQ purpose. However, when we view it as a computational graph and consider the output of each operation in the graph, not all of the operations' outputs are dependent on d uncertain inputs. For an output dependent on d' uncertain inputs with $d' < d$, in its uncertain input space, there are only $k^{d'}$ number of quadrature points, and accordingly this output only needs to be computed $k^{d'}$ times to compute the necessary information for UQ purpose. Drawing from this rationale, we want to partition the model's computational graph into discrete sub-graphs. These sub-graphs are chosen such that the operations within each are dependent on the same uncertain inputs and thus need to be evaluated on the same quadrature points. We achieve this by generating the dependency information for each operation, which shows whether the operation is dependent on an uncertain input. The dependency information is stored as an *influence matrix*, written as $D(\varphi_i, u_j)$, whose (i, j)th entry is 1 if the ith operation depends on the jth uncertain input and 0 otherwise. For example, the dependency information for the simple function in (16) is shown in Tab.2.

From the dependency information, the operations that have the same dependency information on all of the uncertain inputs can be grouped together to form a sub-computational graph to be evaluated on the same quadrature points. Each sub-computational graph should be evaluated the same number of times as the number of quadrature points in its uncertain input space. However, the output of one sub-graph may be the input of another sub-graph, since they have different uncertain input spaces, it is required to extend the uncertain input space of that output to match the uncertain input space of the second sub-graph's output. This requires us to perform a deliberate tensor-algebra operation. Fortunately, we can use the Einstein summation (Einsum) operation to achieve that. The Einsum operation can be used for compactly performing summations involving indices that appear in multiple tensors/matrices. In this case, we can treat the output of the sub-graph as a tensor and use the Einsum operation to perform a summation on specific indices between the output tensor and a tensor of ones, in order to extend its uncertain input space to a desired higher-order input space. For example, consider an addition operation in the computational graph

$$f = \xi_1 + \xi_2$$

with $\xi_1(u_1) \in \mathbb{R}^{k_{u_1}}$ and $\xi_2(u_2) \in \mathbb{R}^{k_{u_2}}$. In this case, ξ_1 is only dependent on u_1 with a size of k_{u_1},

while ξ_2 is only dependent on u_2 with a size of k_{u_2}. The values in ξ_1 and ξ_2 correspond to their evaluations on the quadrature points in u_1 and u_2 dimensions, respectively. The output of this operation, f is dependent on both u_1 and u_2 and its vector should have a size of $\mathbb{R}^{k_{u_1}k_{u_2}}$. What we need to do here is to insert operations to modify the sizes of the inputs so that they have the same size as the output f is supposed to be. Thus we evaluate the addition operation, it results in the correct vector for it. We show a demonstration of how to extend the input size in a Python implementation using NumPy's Einstein summation function followed by a reshape operation. The specific code in Python would be

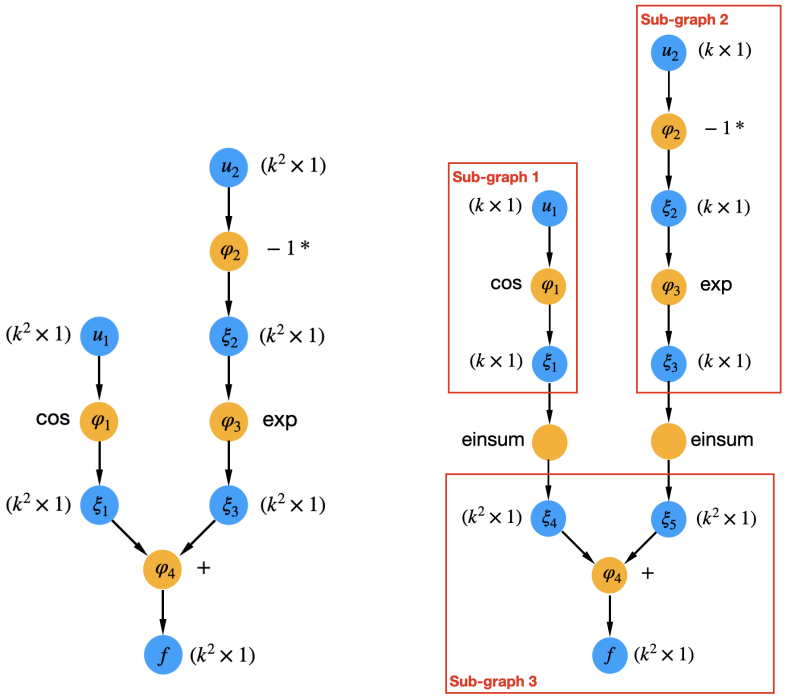
```
import numpy as np
original_shape_xi_1 = (k_u1,1) #Original size for xi_1
original_shape_xi_2 = (k_u2,1) #Original size for xi_2
modified_shape = (k_u1*k_u2, 1) #Target size for xi_1 and xi_2
xi_1 = np.einsum('i...,p...->pi...', xi_1, np.ones((k_u2, 1)) #(k_u1,1) -> (k_u1,
                                                    k_u2,1)
xi_2 = np.einsum('i...,p...->ip...', xi_2, np.ones((k_u1, 1)) #(k_u2,1) -> (k_u1,
                                                    k_u2,1)
xi_1 = np.reshape(xi_1, modified_shape) #(k_u1,k_u2,1) ->(k_u1*k_u2, 1)
xi_2 = np.reshape(xi_2, modified_shape) #(k_u1,k_u2,1) ->(k_u1*k_u2, 1).
```

The code above transforms both inputs into the same size of the output with the correct order of values in the vectors. In fact, for any dimension and size of the input, we can always use the Einstein summation functions followed by reshape functions to transform the input data to the correct size we need. In our method, we insert the *einsum* operation nodes for any connections between the sub-graphs we partitioned. Each *einsum* operation node has an Einstein summation function followed by a reshape function with the right arguments to ensure the correct data flow between the sub-graphs. The outline for the AMTC method is shown in Alg.1.

Algorithm 1 Accelerated Model evaluations on Tensor grids using Computational graph transformations (AMTC) algorithm

1: Specify a computational graph for a computational model
2: Run Kahn's algorithm[34] to determine the correct order for the operations to be evaluated
3: Generate the operations' dependency information and store it as an influence matrix
4: Group the operations that share the same dependency information and form the sub-graphs
5: Insert the *einsum* operation nodes with the right argument between the sub-graphs
6: Evaluate the modified computational graph with the correct input points

To demonstrate how our algorithm modifies the computational graph and reduces the evaluation costs, we show a comparison of computational graphs with and without using AMTC in Fig. 2. Assuming we evaluate the function in (20) at full-grid quadrature points of the uncertain inputs, this figure shows the computational graphs with and without using the AMTC method. The size of the data flow is also labeled in the graphs and the partitioned sub-graphs are indicated on the modified computational graph. In this case, without using the AMTC method, both inputs u_1 and u_2 have the size of $k^2 \times 1$, and thus each operation in the computational graph is evaluated for k^2 number of times. However, with the AMTC method, the computational graph is partitioned into three sub-graphs. The operations in sub-graph 1 are evaluated on the k quadrature points in u_1 dimension while operations in sub-graph 2 are evaluated on the k quadrature points in u_2



(a) Computational graph without using AMTC

(b) Computational graph using AMTC

Figure 2: Computational graphs with data size for full-grid quadrature points evaluation on $f = cos(u_1) + exp(-u_2)$

dimension. Two *einsum* operations are inserted connecting sub-graph 1 and sub-graph 2 to sub-graph 3, so that the outputs of the first two sub-graphs are modified into the correct size and sub-graph 3 can be evaluated k^2 times to gather the same output values as the current method.

3.1 Implementation in CSDL

The AMTC method can only be implemented from the data access layer of a software package where we have access to the computational graph and data structure of a computer program. This is possible thanks to the Computational System Design Language (CSDL) package[1] [35]. CSDL is an embedded domain-specific language targeting large-scale multidisciplinary design analysis and optimization problems. With CSDL, the user defines the model as a sequence of operations by using a software interface that is highly expressive and natural (CSDL code resembles regular Python).

We show a demonstration graph for the implementation of AMTC on CSDL in Fig. 3. The CSDL package has a unique three-stage compiler, which includes front-end, middle-end, and back-end. Using the front-end, the user defines the model and specifies the problem that needs to be solved. Next, the middle-end constructs a computational graph for that model, and the AMTC acts as a graph transformation method to generate the modified computational graph that is more efficient to evaluate. Finally, at the back-end, it creates an executable script based on the modified computational graph using an automatic code generation approach.

[1]CSDL software repository: `https://lsdolab.github.io/csdl/`

Below is a coarse outline of how CSDL and our proposed method interact:

1. User defines the model and identifies the uncertain inputs.
2. Generate the quadrature points according to the distributions of the input variables.
3. Generate the computational graph from the numerical model the user defines.
4. Modify the computational graph as in Alg. 1.
5. Generate an executable back-end.
6. Post-process the output data to calculate the desired quantities of the output.

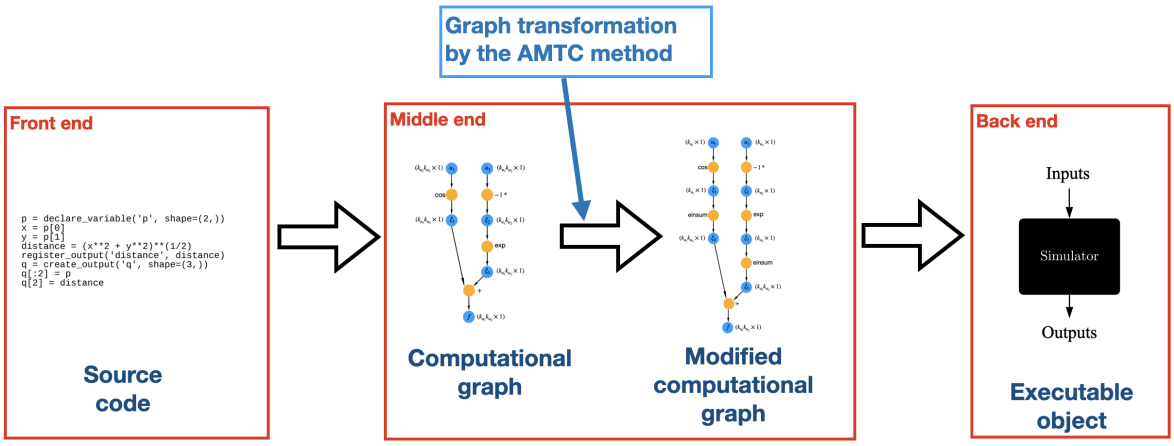


Figure 3: Demonstration for the implementation of AMTC on CSDL

4 Numerical Results

We investigate the performance of AMTC by using it with the full-grid NIPC method on three low-dimensional UQ problems involving different computational models. The AMTC method has been implemented in the middle-end of CSDL compiler as a graph transformation method, and all of the computational models involved in the test problems are built in CSDL in order to use AMTC to accelerate its tensor-grid evaluations.

4.1 Analytical piston model

We first consider a UQ problem involving an analytical non-linear model of the cycle time of a piston. This problem is adapted from [36]. The piston cycle time C in seconds is expressed as

$$C = 2\pi\sqrt{\frac{M}{k + S^2\frac{P_0V_0T_a}{T_0V^2}}}, \tag{23}$$

with

$$V = \frac{S}{2k}\left(\sqrt{A^2 + 4k\frac{P_0V_0}{T_0}T_a} - A\right) \text{ and } A = P_0S + 19.62M - \frac{kV_0}{S}. \tag{24}$$

In this problem, three of the input parameters are normal random variables, and the rest of them are deterministic variables. The values and descriptions for all of the variables are shown in Table 3.

Input parameters	Range	Description
M	$N(50, 10)$	Piston weight (kg)
S	$N(0.01, 0.002)$	Piston surface area (m^2)
V_0	$N(0.005, 0.001)$	Initial gas volume (m^3)
k	3000	Spring coefficient (N/m)
P_0	100000	Atmospheric pressure (N/m^2)
T_a	293	Ambient temperature (K)
T_0	350	Filling gas temperature (K)

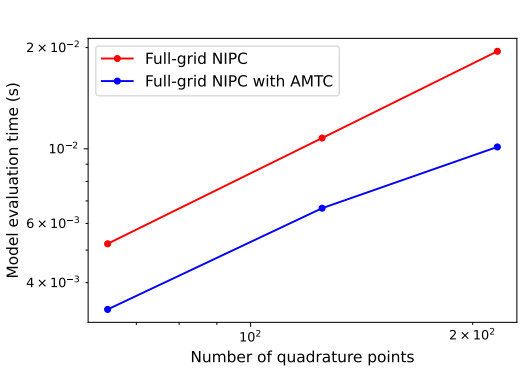
Table 3: Input parameters and ranges for the piston problem

The objective of this problem is to compute the expectation value of the piston cycle time, namely, $\mathbb{E}[C]$.

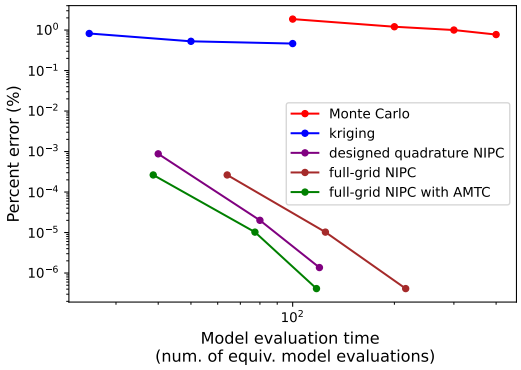
This UQ problem and those that follow are all solved using five methods: integration-based NIPC method using the full-grid quadrature points (full-grid NIPC), full-grid NIPC with the AMTC method (full-grid NIPC with AMTC); integration-based NIPC method using the designed quadrature method (designed quadrature NIPC); kriging and the Monte Carlo method. The kriging method is implemented in its basic form without using adaptive sampling and hyperparameter tuning. The sample points are generated by random sampling, and the kriging surrogate model is trained using the surrogate modelling toolbox in [37].

In our numerical experiments, the full-grid NIPC and the full-grid NIPC with AMTC methods always generate the same results. Fig. 4a shows the model evaluation time in terms of the number of equivalent model evaluations for the full-grid NIPC method with and without AMTC. The results show that the AMTC method brought a consistent 50%-60% reduction in evaluation time. In Fig. 4b, the convergence plots of these five methods are compared for the UQ result. The percentage errors are calculated with respect to the results of the full-grid NIPC using 400 quadrature points. For this UQ problem, the NIPC methods completely outperform the Monte Carlo and kriging methods. This is because this UQ problem is fairly low-dimensional (only three uncertain inputs), and the function is generally smooth. In this kind of UQ problem, PCE-related methods are often the most efficient choices. Among the NIPC methods, the designed quadrature NIPC performs better than the full-grid NIPC, as the designed quadrature NIPC requires a smaller number of quadrature points to achieve the same level of accuracy on integration. However, we see a dramatic speed-up after applying the AMTC method to the full-grid NIPC; the total model evaluation time is reduced by almost an order of magnitude, making the full-grid NIPC with AMTC the most efficient method among other methods.

The major speed-up that AMTC brought for full-grid NIPC can be explained by the sparsity of the influence matrix that describes the dependency information of the computational model in this UQ problem. We show the number of dependent operations for each uncertain input in this computational model in Tab. 4. From this table, we observe that out of the 65 operations in the computational graph, 50% of the operations depend on the uncertain input M and 60% of the operations depend on uncertain inputs S and V_0. This means, a large portion of the operations are not dependent on all of the uncertain inputs and a significant number of repeated evaluations



(a) Comparison of the full-grid NIPC methods with and without applying the AMTC method



(b) Convergence of the UQ results for the five methods

Figure 4: UQ results on the piston model

are eliminated from the operational level with the AMTC method. As this computational model only comprises basic arithmetic operations, the repeated evaluations AMTC eliminated significantly reduces the overall model evaluation time, making the full-grid NIPC the most efficient UQ method.

Uncertain input	No. of dependent operations	Total operations
M	31	65
S	43	65
V_0	45	65

Table 4: Number of dependent operations for each uncertain input in the piston model

4.2 Low-fidelity multidisciplinary model

The second UQ problem we consider involves a low-fidelity multidisciplinary model that computes the total energy stored for a laser-beam-powered unmanned aerial vehicle (UAV) cruising around a ground station for one cycle while being charged by the laser beam. The UQ problem aims to compute the expectation of the total energy stored under three uncertain inputs. We show the mission plot in Fig. 5 and the input parameters in Tab. 5. The computational model involves five disciplines, comprising beam propagation, weights, aerodynamics, and performance models. Further details can be found in [38].

The performance comparison of the full-grid NIPC method with and without AMTC is shown in Fig. 6a, and the convergence results for all of the UQ methods are shown in Fig. 6b. We observe that the AMTC method provided approximately a 90% speed-up in the model evaluation time for the full-grid NIPC method. As a result, full-grid NIPC with AMTC is more efficient to use than all of the other UQ methods we implemented. This is not surprising to see, as in the multidisciplinary model, the parameter uncertainties typically come from different disciplines and there may exist some uncertain inputs that only affect a small portion of the operations. In this scenario, the acceleration provided by the AMTC method can be extremely significant when evaluating on the full-grid quadrature points. From the dependency information shown in Tab.6, the uncertain input

η affects only less than 10% of the operations in the computational model, while the other two operations affect roughly 50% of the operations.

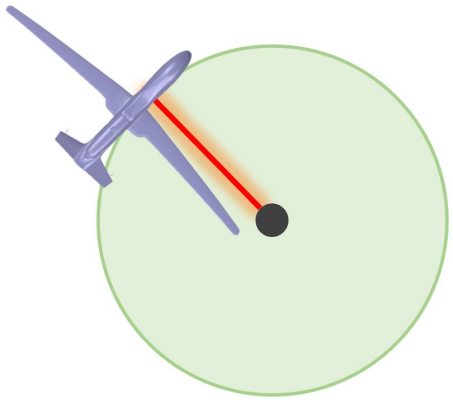


Figure 5: A circular cruise mission around a ground station

Input parameters	Range	Description
V	$N(100, 20)$	Velocity (m/s)
h	$N(10,000, 2,000)$	Altitude(m)
η	$N(0.2, 0.03)$	Atmospheric extinction

Table 5: Input parameters and ranges for the multidisciplinary model

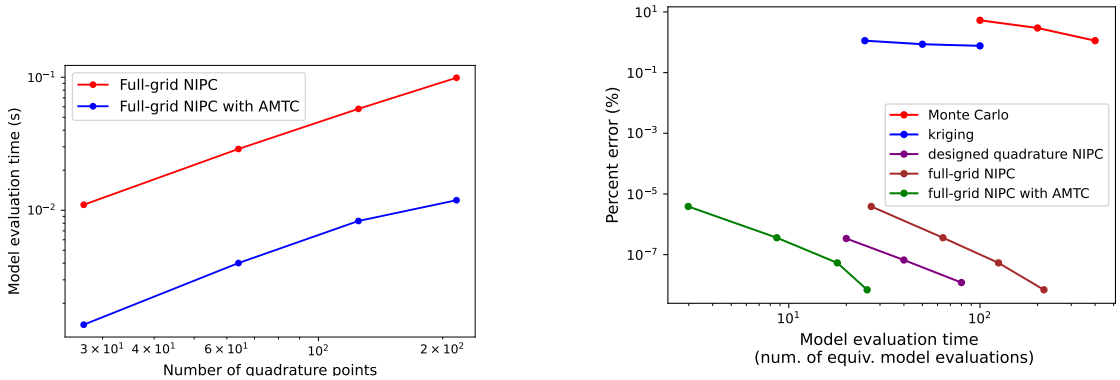
Uncertain input	No. of dependent operations	Total operations
V	162	285
h	141	285
η	21	285

Table 6: Number of dependent operations for each uncertain input in the multidisciplinary model

4.3 Medium-fidelity multi-point model

The third UQ problem we consider is a multi-point mission analysis problem involving an electric vertical takeoff and landing (eVTOL) aircraft. The problem is adapted from [39]. The UQ problem aims to compute the expectation of the total energy consumption of a lift-plus-cruise eVTOL aircraft concept (Fig.7) in a two-segment mission including climb and cruise. Two uncertain inputs are considered in this problem, which are the flight speeds at two mission segments. The details of the properties for the climb and cruise segment are presented in Table 7. For each flight segment, we perform a single-point analysis halfway through the stage and use the vortex-lattice method (VLM), an incompressible and inviscid aerodynamic analysis method to compute the lift and drag forces. The details of the computational model can be found in [40].

We show the performance comparison of the full-grid NIPC method with and without AMTC in Fig. 8a and the UQ convergence plot comparing the five UQ methods in Fig. 8b. For this problem, the AMTC provided a 70-90% percent reduction in evaluation time and the reduction



(a) Comparison of the full-grid NIPC methods with and without applying the AMTC method

(b) Convergence of the UQ results for the five methods

Figure 6: UQ results on the multi-disciplinary model

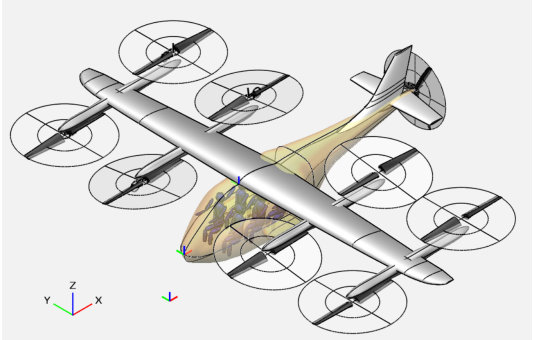
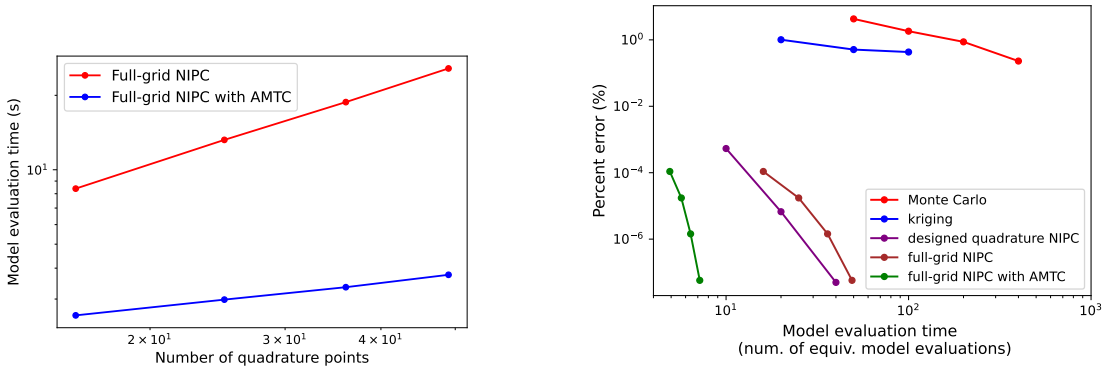


Figure 7: Representation of the eVTOL aircraft [39], credit to NASA

	Climb segment	Cruise segment
Initial altitude (ft)	6,000	10,000
Final altitude (ft)	10,000	10,000
Flight path angle γ (deg)	20	0
Range R (nmi)	R_{climb}	$37.5 - R_{\text{climb}}$
Speed V (Mach)	$N(0.3, 0.03)$	$N(0.5, 0.05)$

Table 7: Properties for climb and cruise segments of the flight mission



(a) Comparison of the full-grid NIPC methods with and without applying the AMTC method

(b) Convergence of the UQ results for the five methods

Figure 8: UQ results on the multi-point model

got more significant as we increase the number of quadrature points. As a result, the full-grid NIPC with AMTC method is significantly more efficient to use than the other UQ methods. The dependency information in Tab. 8 shows roughly 40% of the operations depend on each uncertain input. The sparsity of the influence matrix is present because, for this multi-point problem, we have two uncertain inputs each only affecting the aerodynamic analysis at one point. Although the output of the model is based on the results from both aerodynamic analyses, each aerodynamic analysis is only evaluated on the quadrature points in one-dimensional input space by using the AMTC method. This dramatically accelerated its model evaluation time on full-grid quadrature points. In fact, the performance of AMTC can be more significant for many multi-point problems involving multi-point analyses and multiple uncertain inputs.

We recognize that on this test problem, given the straightforward computational graph structure, our method's performance could be replicated by manually conducting two distinct sets of evaluations with the VLM solver and subsequently integrating these evaluations using tensor operations. However, in a practical design workflow, where UQ problems might undergo frequent reformulations, our method can automatically achieve this reduction in computational cost, sparing users the effort of manual implementation.

Uncertain input	No. of dependent operations	Total operations
V_{climb}	605	1505
V_{cruise}	604	1505

Table 8: Number of dependent operations for each uncertain input in the multi-point model

4.4 Blade element momentum rotor model

The fourth UQ problem we consider is a rotor aerodynamic analysis involving a blade element momentum model. The description of the model can be found in [41]. In this UQ problem, we aim to compute the expectation of the total torque generated by the rotor, given two uncertain inputs, rotational rotor speed, and axial free-stream velocity. The description of the uncertain inputs is shown in Tab. 9.

The performance comparison of the full-grid NIPC method with and without AMTC is shown in Fig. 9a. The convergence results for all of the UQ methods are shown in Fig. 9b, and the dependency information is shown in Tab. 10. In Tab. 10, we observe that the influence matrix here is also sparse, as roughly 55% of the operations in the computational model are dependent on each of the uncertain inputs. However, we observe little acceleration for the full-grid NIPC using AMTC in Fig. 9a. As a result, the full-grid NIPC is outperformed by the designed quadrature method for the UQ results in Fig. 9b. This is because the dependency information is not the only factor that affects the AMTC method's performance. The other factor is each operation's evaluation time. In this model, there exists an implicit operation which also counts as one operation node but its evaluation time takes more than 95% of the model evaluation time. Since the AMTC method does not reduce the number of model evaluations on this implicit operation, the reduction in model evaluation time by AMTC becomes insignificant.

Input parameters	Range	Description
Ω	$N(100, 10)$	Rotational rotor speed (rad/s)
V_x	$N(50, 5)$	Axial free-stream velocity(m/s)

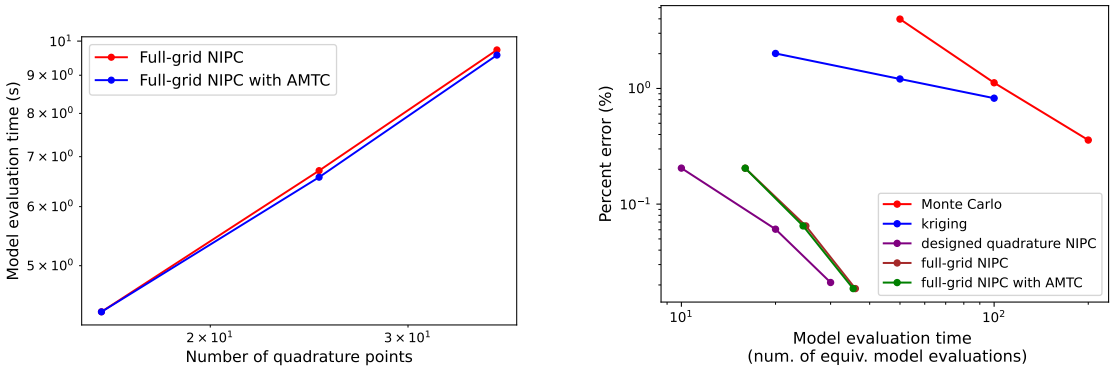
Table 9: Input parameters and ranges for the rotor model

Uncertain input	No. of dependent operations	Total operations
Ω	398	708
V_x	423	708

Table 10: Number of dependent operations for each uncertain input in the rotor model

5 Conclusion

This paper introduces a new method, Accelerated Model evaluations on Tensor grids using Computational graph transformations (AMTC), designed to notably reduce the model evaluation cost on tensor grids of input points via computational graph transformation. It achieves the reduction of model evaluation cost by using the computational graph of the model and algorithmically removing the repeated evaluations on each data node so that it generates the model evaluations



(a) Comparison of the full-grid NIPC methods with and without applying the AMTC method

(b) Convergence of the UQ results for the five methods

Figure 9: UQ results on the rotor model

on the full-grid input points in an efficient way. This method can be used with the integration-based non-intrusive polynomial chaos (NIPC) and stochastic collocation (SC) methods to solve forward uncertainty quantification problems. To generate the numerical results in this paper, we implemented AMTC as a graph transformation method within the three-stage compiler for the Computational System Design Language, so that the modeling language and its compiler automate the process of detecting and capitalizing the model structure, achieving the UQ cost reduction in a general way. We demonstrated the performance of this method on four test problems, which include an analytical function model, a multidisciplinary model, a multi-point model, and a single-disciplinary model. For the first three problems, AMTC significantly reduces the model evaluation cost for the full-grid NIPC method, showcasing a speed-up exceeding 50%. Evidently, this positions AMTC as the optimal choice for achieving unparalleled efficiency within UQ methodologies. However, we did not see this level of performance for the fourth problem as the AMTC does not reduce the number of evaluations on the most cost-dominant operation in the computational graph. Even though we do not expect this method to achieve a significant reduction of computational cost for every UQ problem, AMTC's application holds promise across a broad range of problems that involve multi-disciplinary models, multi-point models, and other models with sparse computational graph structures. These are the realms in which AMTC is poised to markedly enhance the scalability of NIPC and SC methods.

One limitation of this method is that this method assumes a full-grid structure for the input points. Looking ahead, a key direction for future work lies in extending AMTC's functionality to encompass partially tensor-structured input points. Enabling users to determine the optimal tensor structure based on an assessment of the computational graph structure could lead to even lower model evaluation costs via AMTC. Another promising direction is to enable optimization under uncertainty (OUU) with AMTC. As common OUU frameworks perform a UQ analysis at each optimization iteration, the speedup achieved by AMTC for the UQ analysis is vital to solve certain OUU problems with high-fidelity computational models,

Acknowledgments

The material presented in this paper is, in part, based upon work supported by NASA under award No. 80NSSC21M0070 and DARPA under grant No. D23AP00028-00.

References

[1] Susan Joslyn and Sonia Savelli. "Communicating forecast uncertainty: Public perception of weather forecast uncertainty". In: *Meteorological Applications* 17.2 (2010), pp. 180–195. DOI: 10.1002/met.190.

[2] F. Pappenberger, K. J. Beven, N. M. Hunter, P. D. Bates, B. T. Gouweleeuw, J. Thielen, and A. P. J. de Roo. "Cascading model uncertainty from medium range weather forecasts (10 days) through a rainfall-runoff model to flood inundation predictions within the European Flood Forecasting System (EFFS)". In: *Hydrology and Earth System Sciences* 9.4 (2005), pp. 381–393. DOI: 10.5194/hess-9-381-2005.

[3] Hua-Ping Wan, Zhu Mao, Michael D Todd, and Wei-Xin Ren. "Analytical uncertainty quantification for modal frequencies with structural parameter uncertainty using a Gaussian process metamodel". In: *Engineering Structures* 75 (2014), pp. 577–589. DOI: 10.1016/j.engstruct.2014.06.028.

[4] Zhen Hu, Sankaran Mahadevan, and Dan Ao. "Uncertainty aggregation and reduction in structure–material performance prediction". In: *Computational Mechanics* 61.1 (2018), pp. 237–257. DOI: 10.1007/s00466-017-1448-6.

[5] Leo WT Ng and Karen E Willcox. "Monte Carlo information-reuse approach to aircraft conceptual design optimization under uncertainty". In: *Journal of Aircraft* 53.2 (2016), pp. 427–438. DOI: 10.2514/1.C033352.

[6] Bingran Wang, Nicholas C Orndorff, Mark Sperry, and John T Hwang. "High-dimensional uncertainty quantification using graph-accelerated non-intrusive polynomial chaos and active subspace methods". In: *AIAA AVIATION 2023 Forum*. 2023, p. 4264. DOI: 10.2514/6.2023-4264.

[7] Daejin Lim, Hyeongseok Kim, and Kwanjung Yee. "Uncertainty propagation in flight performance of multirotor with parametric and model uncertainties". In: *Aerospace Science and Technology* 122 (2022), p. 107398. DOI: 10.1016/j.ast.2022.107398.

[8] Jeffrey M Wooldridge. "Applications of generalized method of moments estimation". In: *Journal of Economic perspectives* 15.4 (2001), pp. 87–100. DOI: 10.1257/jep.15.4.87.

[9] KB Fragkos, EM Papoutsis-Kiachagias, and KC Giannakoglou. "pFOSM: An efficient algorithm for aerodynamic robust design based on continuous adjoint and matrix-vector products". In: *Computers & Fluids* 181 (2019), pp. 57–66. DOI: 10.1016/j.compfluid.2019.01.016.

[10] Jiaqi Luo and Feng Liu. "Statistical evaluation of performance impact of manufacturing variability by an adjoint method". In: *Aerospace Science and Technology* 77 (2018), pp. 471–484. DOI: 10.1016/j.ast.2018.03.030.

[11] Jiaqi Luo, Yao Zheng, and Feng Liu. "Optimal tolerance allocation in blade manufacturing by sensitivity-based performance impact evaluation". In: *Journal of Propulsion and Power* 36.4 (2020), pp. 632–638. DOI: 10.2514/1.B37718.

[12] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Optimal model management for multifidelity Monte Carlo estimation". In: *SIAM Journal on Scientific Computing* 38.5 (2016), A3163–A3194. DOI: 10.1137/15M1046472.

[13] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. "Survey of multifidelity methods in uncertainty propagation, inference, and optimization". In: *Siam Review* 60.3 (2018), pp. 550–591. DOI: 10.1137/16M1082469.

[14] Armin Tabandeh, Gaofeng Jia, and Paolo Gardoni. "A review and assessment of importance sampling methods for reliability analysis". In: *Structural Safety* 97 (2022), p. 102216. DOI: 10.1016/j.strusafe.2022.102216.

[15] Irfan Kaymaz. "Application of kriging method to structural reliability problems". In: *Structural safety* 27.2 (2005), pp. 133–151. DOI: 10.1016/j.strusafe.2004.09.001.

[16] Zhen Hu and Sankaran Mahadevan. "A single-loop kriging surrogate modeling for time-dependent reliability analysis". In: *Journal of Mechanical Design* 138.6 (2016), p. 061406. DOI: 10.1115/1.4033428.

[17] Markus P Rumpfkeil. "Optimizations under uncertainty using gradients, Hessians, and surrogate models". In: *AIAA journal* 51.2 (2013), pp. 444–451. DOI: 10.2514/1.J051847.

[18] Serhat Hosder, Robert Walters, and Rafael Perez. "A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations". In: *44th AIAA aerospace sciences meeting and exhibit*. 2006, p. 891. DOI: 10.2514/6.2006-891.

[19] Brandon A Jones, Alireza Doostan, and George H Born. "Nonlinear propagation of orbit uncertainty using non-intrusive polynomial chaos". In: *Journal of Guidance, Control, and Dynamics* 36.2 (2013), pp. 430–444. DOI: 10.2514/1.57599.

[20] Vahid Keshavarzzadeh, Felipe Fernandez, and Daniel A Tortorelli. "Topology optimization under uncertainty via non-intrusive polynomial chaos expansion". In: *Computer Methods in Applied Mechanics and Engineering* 318 (2017), pp. 120–147. DOI: 10.1016/j.cma.2017.01.019.

[21] Dongbin Xiu and Jan S Hesthaven. "High-order collocation methods for differential equations with random inputs". In: *SIAM Journal on Scientific Computing* 27.3 (2005), pp. 1118–1139. DOI: 10.1137/040615201.

[22] Ivo Babuška, Fabio Nobile, and Raúl Tempone. "A stochastic collocation method for elliptic partial differential equations with random input data". In: *SIAM Journal on Numerical Analysis* 45.3 (2007), pp. 1005–1034. DOI: 10.1137/050645142.

[23] Norbert Wiener. "The homogeneous chaos". In: *American Journal of Mathematics* 60.4 (1938), pp. 897–936. DOI: 10.2307/2371268.

[24] Dongbin Xiu and George Em Karniadakis. "The Wiener–Askey polynomial chaos for stochastic differential equations". In: *SIAM journal on scientific computing* 24.2 (2002), pp. 619–644. DOI: 10.1137/S1064827501387826.

[25] Fabio Nobile, Raúl Tempone, and Clayton G Webster. "A sparse grid stochastic collocation method for partial differential equations with random input data". In: *SIAM Journal on Numerical Analysis* 46.5 (2008), pp. 2309–2345. DOI: 10.1137/060663660.

[26] Thomas Gerstner and Michael Griebel. "Numerical integration using sparse grids". In: *Numerical algorithms* 18.3-4 (1998), pp. 209–232. DOI: 10.1023/A:1019129717644.

[27] Vahid Keshavarzzadeh, Robert M Kirby, and Akil Narayan. "Numerical integration in multiple dimensions with designed quadrature". In: *SIAM Journal on Scientific Computing* 40.4 (2018), A2033–A2061. DOI: 10.1137/17M1137875.

[28] Géraud Blatman and Bruno Sudret. "Sparse polynomial chaos expansions and adaptive stochastic finite elements using a regression approach". In: *Comptes rendus mécanique* 336.6 (2008), pp. 518–523. DOI: 10.1016/j.crme.2008.02.013.

[29] Mishal Thapa, Sameer B Mulani, and Robert W Walters. "Adaptive weighted least-squares polynomial chaos expansion with basis adaptivity and sequential adaptive sampling". In: *Computer Methods in Applied Mechanics and Engineering* 360 (2020), p. 112759. DOI: 10.1016/j.cma.2019.112759.

[30] Nora Lüthen, Stefano Marelli, and Bruno Sudret. "Sparse polynomial chaos expansions: Literature survey and benchmark". In: *SIAM/ASA Journal on Uncertainty Quantification* 9.2 (2021), pp. 593–649. DOI: 10.1137/20M1315774.

[31] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic Differentiation in Machine Learning: a Survey". In: *Journal of Machine Learning Research* 18.153 (2018), pp. 1–43. URL: http://jmlr.org/papers/v18/17-468.html.

[32] Mark Sperry, Kavish Kondap, and John T Hwang. "Automatic adjoint sensitivity analysis of models for large-scale multidisciplinary design optimization". In: *AIAA AVIATION 2023 Forum*. 2023, p. 3721. DOI: 10.2514/6.2023-3721.

[33] Martín Abadi. "TensorFlow: learning functions at scale". In: *Proceedings of the 21st ACM SIGPLAN international conference on functional programming*. 2016, pp. 1–1. DOI: 10.1145/2951913.2976746.

[34] Arthur B Kahn. "Topological sorting of large networks". In: *Communications of the ACM* 5.11 (1962), pp. 558–562. DOI: 10.1145/368996.369025.

[35] Victor Gandarillas, Anugrah Jo Joshy, Mark Z. Sperry, Alexander K. Ivanov, and John T Hwang. "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis". In: *Structural and Multidisciplinary Optimization* ((under revision)).

[36] Einat Neumann Ben-Ari and David M Steinberg. "Modeling data from computer experiments: an empirical comparison of kriging with MARS and projection pursuit regression". In: *Quality Engineering* 19.4 (2007), pp. 327–338. DOI: 10.1080/08982110701580930.

[37] Mohamed Amine Bouhlel, John T. Hwang, Nathalie Bartoli, Rémi Lafage, Joseph Morlier, and Joaquim R. R. A. Martins. "A Python surrogate modeling framework with derivatives". In: *Advances in Engineering Software* (2019), p. 102662. ISSN: 0965-9978. DOI: 10.1016/j.advengsoft.2019.03.005.

[38] Bingran Wang, Nicholas C Orndorff, and John T Hwang. "Optimally tensor-structured quadrature rule for uncertainty quantification". In: *AIAA SCITECH 2023 Forum*. 2023, p. 0741. DOI: 10.2514/6.2023-0741.

[39] Christopher Silva, Wayne R Johnson, Eduardo Solis, Michael D Patterson, and Kevin R Antcliff. "VTOL urban air mobility concept vehicles for technology development". In: *2018 Aviation Technology, Integration, and Operations Conference*. 2018, p. 3847. DOI: 10.2514/6.2018-3847.

[40] Bingran Wang, Mark Sperry, Victor E Gandarillas, and John T Hwang. "Efficient uncertainty propagation through computational graph modification and automatic code generation". In: *AIAA AVIATION 2022 Forum*. 2022, p. 3997. DOI: 10.2514/6.2022-3997.

[41] Marius L Ruh and John T Hwang. "Fast and Robust Computation of Optimal Rotor Designs Using Blade Element Momentum Theory". In: *AIAA Journal* 61.9 (2023), pp. 4096–4111. DOI: 10.2514/1.J062611.