Author version

Published as:

Wang, B., Orndorff, N. C., & Hwang, J. T. (2023). Optimally tensor-structured quadrature rule for uncertainty quantification. In AIAA SCITECH 2023 Forum (Paper No. AIAA 2023-0741). https://doi.org/10.2514/6.2023-0741

BibTeX for this reference:

https://lsdolab.github.io/lsdo-publications/publication/wang2023optimally/

```
@inproceedings{wang2023optimally,
  author = {Bingran Wang and Nicholas C. Orndorff and John T. Hwang},
  title = {Optimally tensor-structured quadrature rule for uncertainty quantification},
  booktitle = {AIAA SCITECH 2023 Forum},
  year = {2023},
  doi = {10.2514/6.2023-0741},
  url = {https://doi.org/10.2514/6.2023-0741}
}
```

Optimally tensor-structured quadrature rule for uncertainty quantification

Bingran Wang*, Nicholas C. Orndorff†, and John T. Hwang‡
University of California, San Diego, La Jolla, CA, 92093

Non-intrusive polynomial chaos (NIPC) is an efficient method for solving forward uncertainty quantification (UQ) problems. If using the integration-based approach, NIPC results in multi-dimensional integration problems that are often solved using the numerical quadrature approach. For the numerical quadrature rule, the full-grid Gauss quadrature method maintains a fully tensorial structure of univariate Gauss quadrature rule, and the number of quadrature points increases exponentially with the number of dimensions, while for the designed quadrature method, the quadrature rule does not maintain a tensorial structure and the number of quadrature points increases at a slower rate than the Gauss quadrature method. Recently, a new method called accelerated tensor-grid evaluations (ATE) was proposed to reduce the cost of model evaluations on the tensor-grid inputs. The ATE method modifies the computational graph of the model in the middle-end of a three-stage compiler so that repeated evaluations on the operation level are eliminated. With ATE, the full-grid NIPC method can be the most efficient UQ method on problems involving sparse computational graph structures. However, for many high-dimensional problems, there may exist a partially tensorial structure for the quadrature rule that is optimal to use with ATE. In this paper, we propose a new method called optimally tensor-structured quadrature rule. This method is based on the designed quadrature idea but finds the quadrature rule that has the optimal tensorial structure so that the cost of model evaluations is minimized while achieving a specific level of accuracy with the integration method. The proposed method also generates the same quadrature rule as full-grid Gauss quadrature or designed quadrature method when the optimal tensorial structure is a fully tensorial or non-tensorial structure, respectively. This method has been tested on a 4D uncertainty quantification problem that involves a low-fidelity multi-disciplinary model of a laser-beam powered UAV. On this problem, the NIPC method using the optimally tensor-structured quadrature points with ATE is at least 50% more efficient than the existing UQ methods we compared with, while achieving the same level of accuracy.

I. Introduction

Uncertainty quantification (UQ) is a broad research area that focuses on quantifying uncertainties, reducing uncertainties, and predicting the response uncertainties in both computational and real-world applications. In engineering, many design and simulation problems are subject to uncertainties and the applications of UQ can be found in many disciplines such as fluids [1, 2], structures [3, 4], control [5], hydrology [6, 7], and machine learning [8]. UQ problems can be broadly divided into two categories: forward problems and inverse problems. The forward UQ problem is an uncertainty propagation problem in which we want to determine the response uncertainties when we are given the probability distributions for uncertain inputs to the model. For the inverse problems, we are often given the observation data for the model responses and the problem is to determine the uncertainties in the parameters or inputs so that the model responses best match the observation data.

In this paper, we consider using non-intrusive UQ methods to solve uncertainty propagation problems. One particularly efficient method is the non-intrusive polynomial chaos (NIPC) method. The NIPC method approximates the model response as a linear combination of the polynomial chaos expansion (PCE) basis functions which are determined based on the distributions of the random inputs. The coefficients for PCE basis functions can be determined by using the integration or regression approach. The integration approach projects the model response onto each basis and uses the orthogonality of the basis functions to extract each PCE coefficient. This results in a multi-dimensional

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

integration problem. The integration problems can be solved by using the full-grid Gauss quadrature or the designed quadrature method. The full-grid Gauss quadrature method formulates the quadrature rule in a tensor product of univariate quadrature rules. This approach is easy to use but the number of quadrature points increases exponentially with the number of dimensions. The designed quadrature method solves an optimization problem to find approximate solutions of the moment-matching equations. The designed quadrature method does not maintain a tensorial structure for the quadrature points and the number of quadrature points increases at a much slower rate than the full-grid Gauss quadrature rule.

A method called accelerated tensor-grid evaluations (ATE) was proposed recently to reduce the model evaluation cost on tensor-grid inputs. This method takes advantage of the tensor-structured inputs and eliminates the repeated evaluations on the operation level by modifying the computational graph from the middle-end of a three-stage compiler for the mathematical model description. The ATE method works well with the full-grid Gauss quadrature approach and can significantly reduce its model evaluation cost for problems with sparse computational graph structures. With the ATE method, the model evaluation cost on the quadrature rule is no longer only related to the number of quadrature points. The cost now depends on the graph structure of the computational model, the tensorial structure of quadrature points, and the number of quadrature points in each dimension. However, there is no method that determines the optimal quadrature rule when the ATE method is available.

In this paper, we propose a new method to determine the quadrature rule with the optimal tensorial structure so that the cost of model evaluations is minimized while achieving a given level of accuracy for the integration problem. In the cases when the non-tensorial structure or the fully tensorial structure is the optimal tensorial structure, our method generates the same quadrature rule as the full-grid Gauss quadrature or the designed quadrature rule, respectively. This method is applied to an UQ problem that involves a low-fidelity multidisciplinary model describing a laser-beam-powered UAV. For this problem, our method finds a quadrature rule that has a partially tensorial structure. This quadrature rule is used with NIPC method for the UQ problem, and it requires at least 50% less of model evaluation cost than the other UQ methods to achieve the same level of accuracy on the UQ result.

This paper is organized as follows. In Section II, we give some background on the uncertainty propagation problems, polynomial chaos theory, the NIPC method and the ATE method. In Section III, we show the motivation of this work. In Section IV, we present the algorithm of a new method called optimally tensor-structured quadrature rule. In Section V, we show the numerical results of a 4D UQ problem involves a laser-beam powered UAV. The conclusion is made in Section VI.

II. Background

A. Uncertainty propagation

The forward uncertainty quantification problem aims to propagate uncertain inputs through the computational model and determine their effects on the quantity of interest (QoI). The uncertainties in the inputs can be broadly divided into two categories: aleatoric uncertainties and epistemic uncertainties. Aleatoric uncertainties are also known as irreducible uncertainties as they are inherent to the system and cannot be avoided. For example, variations in the operating conditions, mission requirements, and model parameters are aleatoric uncertainties, and they usually can be defined in a probabilistic framework. Epistemic uncertainties are reducible uncertainties as they typically come from the lack of knowledge by the engineers. For example, the errors that come from the approximations used in the computational model and the errors associated with the numerical solution methods are epistemic uncertainties, and they are difficult to be defined in a probabilistic framework. In this paper, we consider an uncertainty propagation problem within the probabilistic formalism in which the uncertain inputs are continuous random variables with known probability density distributions. In this problem, we assume the QoI becomes a continuous random variable, which can be characterized by its statistical moments like expectation and standard deviation. Other quantities like the probability of failure, conditional value at risk (CVaR), or the probability density distribution of the QoI can also be the objective of the uncertainty propagation problem.

The uncertainty propagation problem is also an essential sub-problem in the optimization under uncertainty (OUU) [9] problems. OUU problems incorporate the uncertainties into the optimization problems so that the optimized design is more robust and reliable in the presence of variability in the conditions and parameters. For a deterministic engineering design optimization problem, the optimizer optimizes an objective function with respect to the design

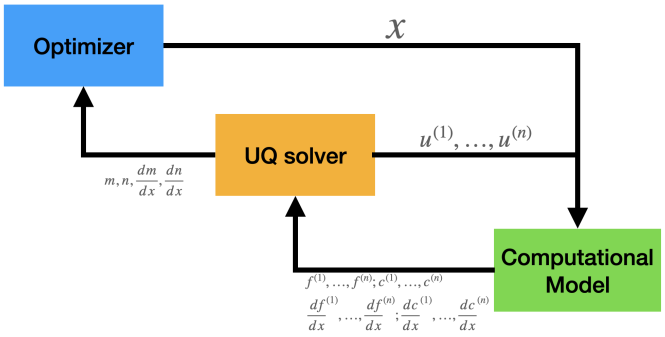


Fig. 1 Typical computational framework for optimization under uncertainties

variables subject to constraints, and a standard form is:

$$\begin{aligned} \min_{x} \quad & \mathcal{F}(x) \\ \text{s.t.} \quad & C(x) < 0, \end{aligned} \tag{1}$$

where x is the vector of design variables, $\mathcal{F}$ is the objective function and C is the vector-valued constraint function. In OUU problems, it is possible that the output of both objective and constraint functions become random variables. Depending on the ways of treating the constraints, OUU problems can be broadly divided into two categories: robust design optimization (RDO) [10, 11] and reliability-based design optimization (RBDO) [12]. The RDO problems consider the effects of uncertainty in the objective and constraints through statistical moments, e.g., mean plus standard deviation. On the other hand, the RBDO problems add probabilistic constraints based on the probability of failure. A standard form for RDO is:

$$\begin{aligned} \min_{x} \quad & \mathcal{M}(x) := \mathbb{E}[\mathcal{F}(x,U)] + \alpha_1 \mathbb{S}[\mathcal{F}(x,U)] \\ \text{s.t.} \quad & \mathcal{N}(x) := \mathbb{E}[C(x,U)] + \alpha_2 \mathbb{S}[C(x,U)] < 0, \end{aligned} \tag{2}$$

where U are the uncertain inputs ; $\mathcal{M}$ and $\mathcal{N}$ are the new objective and constraint functions respectively that consist of the statistical moments of the original functions, while α_1 and α_2 are the corresponding tunable parameters. A standard form for RBDO is:

$$\begin{aligned} \min_{x} \quad & \mathcal{M}(x) := \mathbb{E}[\mathcal{F}(x,U)] + \alpha \mathbb{S}[\mathcal{F}(x,U)] \\ \text{s.t.} \quad & \mathcal{N}(x) := \mathbb{P}[C(x,U) < 0] > p, \end{aligned} \tag{3}$$

where $\mathbb{P}[\cdot]$ is the probability of the statement being true, and p is the reliability constraint vector. The computational framework for solving the OUU problems typically involves two nested for-loops of a UQ solver inside an optimizer, a demonstration of which is shown in Fig. 1. Under this framework, at each optimization iteration, we are solving an uncertainty propagation problem to calculate the objective and constraint function, and also its derivatives if we are using a gradient-based optimizer. In this case, the scalability and the accuracy of the UQ method is extremely crucial for solving the large-scale OUU problems.

To solve the uncertainty propagation problems, common non-intrusive methods include the method of moments, polynomial chaos, and Monte Carlo. The method of moments uses the evaluations of the QoI and its derivatives at a fixed point to construct a Taylor series approximation of the QoI with respect to the uncertain inputs. The statistical moments of the QoI can be analytically calculated from the Taylor series approximation, and the cost of this method can be as cheap as two model evaluations [13] if using the adjoint method to compute the derivatives. However, this method can only compute the QoI's statistical moments and may lack accuracy when the variance of the uncertain inputs is significant. The Monte Carlo method randomly samples the space of uncertain inputs and uses the QoI evaluations at all of the sample points to compute the desired quantities of the QoI. This method is intuitive and easy to implement, and due to the law of large numbers, the convergence rate is independent of the number of certain inputs. However, the Monte Carlo method generally exhibits slow convergence and is commonly used for large-scale UQ problems and with multi-fidelity methods as in [14] and [15]. The polynomial chaos method represents the QoI using the orthogonal polynomials. It exploits the smoothness in the random space to construct sampling or solution techniques with a fast convergence rate for UQ problems. Common polynomial chaos-based methods include non-intrusive polynomial chaos [16–18] and stochastic collocation methods [19, 20].

B. Polynomial chaos

Polynomial chaos or polynomial chaos expansion (PCE) dates back to Wiener, who used Hermite polynomials to construct a physical theory for Gaussian random processes [21]. Ghanem extended this idea to use Hermite polynomials as orthogonal basis to represent the random processes [22]. However, this method suffers from convergence issues for problems with non-Gaussian uncertain inputs. This difficulty is solved by Xiu and Karniadakis in [23], by proposing the generalized polynomial chaos (gPC) method. The gPC method uses different orthogonal polynomials for uncertain inputs with different distributions to achieve optimal convergence for the UQ problem.

Consider an uncertainty propagation problem:

$$F = \mathcal{F}(U), \tag{4}$$

where $U = (U_1, \ldots, U_d) \in \mathbb{R}^d$ are a set of mutually independent random variables with probability density distributions $\rho(u)$ and support Γ, $F \in \mathbb{R}$ is the QoI we are interested and $\mathcal{F} : \mathbb{R}^d \to \mathbb{R}$ is the model evaluation function. Using the gPC method, the QoI can be represented as a linear combination of orthogonal polynomials as

$$F(U) = \sum_{i=0}^{\infty} \alpha_i \Phi_i(U), \tag{5}$$

where Φ_i are the PCE basis functions, and α_i are the corresponding PCE coefficients. In practice, 5 is typically truncated to polynomials order up to p, resulting into q PCE terms:

$$F(U) \approx \sum_{i=0}^{q} \alpha_i \Phi_i(U), \tag{6}$$

with

$$q + 1 = \frac{(d+p)!}{d!p!}. \tag{7}$$

The PCE basis functions satisfy the orthogonal property as

$$\left\langle \Phi_i(U), \Phi_j(U) \right\rangle = 0, \quad \text{if } i \neq j, \tag{8}$$

where the inner product is defined as

$$\left\langle \Phi_i(U), \Phi_j(U) \right\rangle = \int_{\Gamma} \Phi_i(u)\Phi_j(u)\rho(u)du. \tag{9}$$

In the one-dimensional case, the PCE basis functions are nivariate orthogonal polynomials that are chosen based on the distribution of the uncertain input. For common types of continuous random variables, their corresponding polynomials and support ranges are shown in Table 1. In multi-dimensional cases, the PCE basis functions are the tensor products of the univariate orthogonal polynomials. For example, for case with two uncertain inputs, the first six PCE basis functions are

$$\begin{aligned}
\Phi_0(U) &= \phi_0(U_1)\phi_0(U_2) \\
\Phi_1(U) &= \phi_1(U_1)\phi_0(U_2) \\
\Phi_2(U) &= \phi_0(U_1)\phi_1(U_2) \\
\Phi_3(U) &= \phi_2(U_1)\phi_0(U_2) \\
\Phi_4(U) &= \phi_1(U_1)\phi_1(U_2) \\
\Phi_5(U) &= \phi_0(U_1)\phi_2(U_2),
\end{aligned}$$

where $\phi_i(U_j)$ is the i-th univariate orthogonal polynomial that correspond to the j-th uncertain input. The support range for them is $\Gamma = \Gamma_1 \times \Gamma_2$, where Γ_i is the support of $\rho(u_i)$. If the uncertain inputs are dependent, the PCE basis functions can be numerically generated based on the joint probability distribution of the uncertain inputs [24].

With the gPC model in 6, the statistical moments of the QoI can be analytically computed as

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

$$\sigma_f = \sum_{i=0}^{d} \alpha_i^2 \left\langle \Phi_i^2 \right\rangle. \tag{11}$$

For other risk measures like probability of failure or CVaR, the gPC model can be used as a cheap-to-evaluate surrogate model and one can apply sampling methods on it to approximate the target risk measure.

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	$[0, \infty)$

C. Non-intrusive polynomial chaos

The non-intrusive polynomial chaos method solves the PCE coefficients α_i in (6) by using the integration or regression approach. The integration approach makes use of the orthogonal property in (8) and projects the QoI onto each basis function, which results in an integration problem as

$$\begin{aligned}\alpha_i &= \frac{\left\langle F(U), \Phi_j \right\rangle}{\left\langle \Phi_i^2 \right\rangle} \\ &= \frac{1}{\left\langle \Phi_i^2 \right\rangle} \int_\Gamma f(u)\Phi_i(u)\rho(u)du \\ &= \frac{1}{\left\langle \Phi_i^2 \right\rangle} \int_{\Gamma_1} \dots \int_{\Gamma_d} f(u_1, \dots, u_n)\Phi_i(u_1, \dots, u_n)\rho(u_1, \dots, u_n)du_1 \dots du_n.\end{aligned} \tag{12}$$

The regression approach computes the PCE coefficients by solving a linear least-squares problem as

$$\begin{bmatrix} \Phi_1(u^{(1)}) & \dots & \Phi_q(u^{(1)}) \\ \vdots & & \vdots \\ \Phi_1(u^{(n)}) & \dots & \Phi_q(u^{(n)}) \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_q \end{bmatrix} = \begin{bmatrix} f(u^{(1)}) \\ \vdots \\ f(u^{(n)}) \end{bmatrix}, \tag{13}$$

where $(u^{(1)}, \dots, u^{(n)})$ represents a set of n number of sample points generated from the distributions of the uncertain inputs. The rule of thumb is that the number of samples n needs to be 2-3 times the number of coefficients. 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. A recent review paper for sparse PCE is presented in [25]. In [26], Eldred and Burkardt showed that for UQ problems with a small number of uncertain inputs, integration-based NIPC and regression-based NIPC methods have similar performance, and they both demonstrate significantly higher convergence rates than the Monte Carlo method.

D. Numerical quadrature

This paper mainly considers using the integration-based NIPC method to solve multi-dimensional UQ problems. In this case, the integration-based NIPC method essentially solves the integration problems as in (12). We write the integration problem as

$$I(f) = \int_\Gamma f(u)\rho(u)du,$$

where $u = (u_1, \dots, u_d) \in \mathbb{R}^d$. This integration problem is often solved by using the numerical quadrature approach, which approximates the integral as the weighted sum of function evaluations at certain input points:

$$I(f) \approx \sum_{i=1}^{n} w^{(i)} f(u^{(i)}), \tag{14}$$

where $u^{(i)} \in \Gamma$ and $w^{(i)} > 0$ are the nodes and weights, respectively, that are to be determined by the quadrature rule.

The goal of the quadrature rule is to accurately approximate a large set of functions with a small number of function evaluations. This is typically achieved by enforcing the equality conditions for all of the functions f in a subspace Π of polynomials as

$$\int_\Gamma f(u)\rho(u)du = \sum_{i=1}^{n} w^{(i)} f(u^{(i)}) \quad \text{for all} \quad f \in \Pi_r. \tag{15}$$

We consider the cross-polynomial subspace with the total polynomial order of r, which is defined as

$$\Pi_r = \text{span}\{u^\alpha \mid \alpha \in \Lambda_r\}, \tag{16}$$

where

$$\alpha = (\alpha_1, \ldots, \alpha_d), \quad u^\alpha = \prod_{j=1}^{d} (u_j)^{\alpha_j} \tag{17}$$

and

$$\Lambda_r = \left\{ \alpha \mid \sum_{i=1}^{d} \alpha_i \leq r \right\}. \tag{18}$$

In many integration problems, the evaluation functions are smooth and can be well approximated by the polynomials within the cross-polynomial subspace. In this case, the quadrature rule that enforces (15) can be very effective.

1. Gauss quadrature rule

One way to generate the quadrature points is to use the Gauss quadrature rule [27]. In one-dimensional case, the quadrature points $u^{(1)}, \ldots, u^{(k)}$ are the roots of the orthogonal polynomial $p_k(u)$ with degree k. The orthogonal polynomials here are defined in the same way as the PCE basis functions and they also satisfy the orthogonal properties as

$$\left\langle p_i, p_j \right\rangle = \int_\Gamma p_i(u) p_j(u) \rho(u) du = 0, \quad \text{if } i \neq j. \tag{19}$$

The weights of the Gauss quadrature rule, $w^{(1)}, \ldots, w^{(k)}$ can be computed by solving the moment matching equations:

$$\sum_{i=1}^{k} p_j\left(u^{(i)}\right) w^{(i)} = \begin{cases} \sqrt{b_0} & \text{if } j = 0 \\ 0 & \text{for } j = 1, \ldots, k-1, \end{cases} \tag{20}$$

where

$$b_0 = \left\langle p_0, p_0 \right\rangle. \tag{21}$$

The univariate Gauss quadrature rule with k number of nodes and weights can exactly integrate the univariate polynomials up to $2k-1$ degrees.

For multi-dimensional integrations, the Gauss quadrature rule extends the univariate rule to a tensor structure, assuming using the k number of quadrature points in each dimension, the integral is approximated as

$$I \approx \sum_{i_1=0}^{k} \cdots \sum_{i_d=0}^{k} w_1^{(i_1)} \ldots w_d^{(i_d)} f(u_1^{(i_1)}, \ldots, u_d^{(i_d)}) \rho(u_1^{(i_1)}, \ldots, u_n^{(i_d)}), \tag{22}$$

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, respectively. In this way, the quadrature rule can still exactly integrate the cross-polynomials with total order up to $2k-1$. However, this approach suffers from the curse of dimensionality, as it requires function evaluations on the full-grid quadrature points and the total number of quadrature points increases exponentially as the number of dimensions, $n = k^d$. We represent the full-grid quadrature points as

$$\boldsymbol{u} = \boldsymbol{u}_d^k \times \cdots \times \boldsymbol{u}_1^k, \tag{23}$$

where

$$\boldsymbol{u}_i^k := \{u_i^{(j)}\}_{j=1}^{k}. \tag{24}$$

To mitigate the curse of dimensionality the Gauss quadrature rule suffers from, we may use the Smolyak sparse grid method [28]. This method drops the higher-order cross terms in the full-grid Gauss quadrature points to reduce the number of quadrature points while maintaining minimal loss on the accuracy of the quadrature rule. The sparse grid quadrature points can be expressed as:

$$\boldsymbol{u} = \bigcup_{\ell-d+1 \le |\mathbf{i}| \le \ell} \left(\boldsymbol{u}_1^{i_1} \times \cdots \times \boldsymbol{u}_d^{i_n} \right), \tag{25}$$

where ℓ is the level of construction.

2. Designed quadrature rule

A more effective quadrature rule for high-dimensional integration problems is to use the designed quadrature method which was proposed by Keshavarzzadeh et al. [29]. The core idea of the designed quadrature method is to find the nodes and weights that satisfy the multi-variate moment matching equations, which are defined as

$$\sum_{i=1}^{n} \pi_\alpha \left(u^{(i)} \right) w^{(i)} = \begin{cases} \sqrt{b_0} & \text{if } \alpha = 0 \\ 0 & \text{for } \alpha \in \Lambda_k \setminus 0, \end{cases} \tag{26}$$

where

$$\pi_\alpha = \prod_{i=1}^{d} p_{\alpha_i}^{(i)}(u_i) \tag{27}$$

and $p_j^{(i)}$ is the j-th orthogonal polynomials in the u_i dimension. If the nodes and weights of the quadrature rule satisfy the equations in (26), then the quadrature rules can exactly integrate cross-term polynomials up to the order of $(2k-1)$.

We denote the quadrature points and the weights as $\boldsymbol{u} := (u^{(1)}, \ldots, u^{(n)})$ and the weights $\boldsymbol{w} := (w^{(1)}, \ldots, w^{(n)})$ respectively. Then, the moment matching equations in (26) can be written as residual equations:

$$\mathcal{R}(\boldsymbol{u}, \boldsymbol{w}) = 0, \tag{28}$$

where $\mathcal{R} : \mathbb{R}^{d \times n} \times \mathbb{R}^n \to \mathbb{R}^n$.

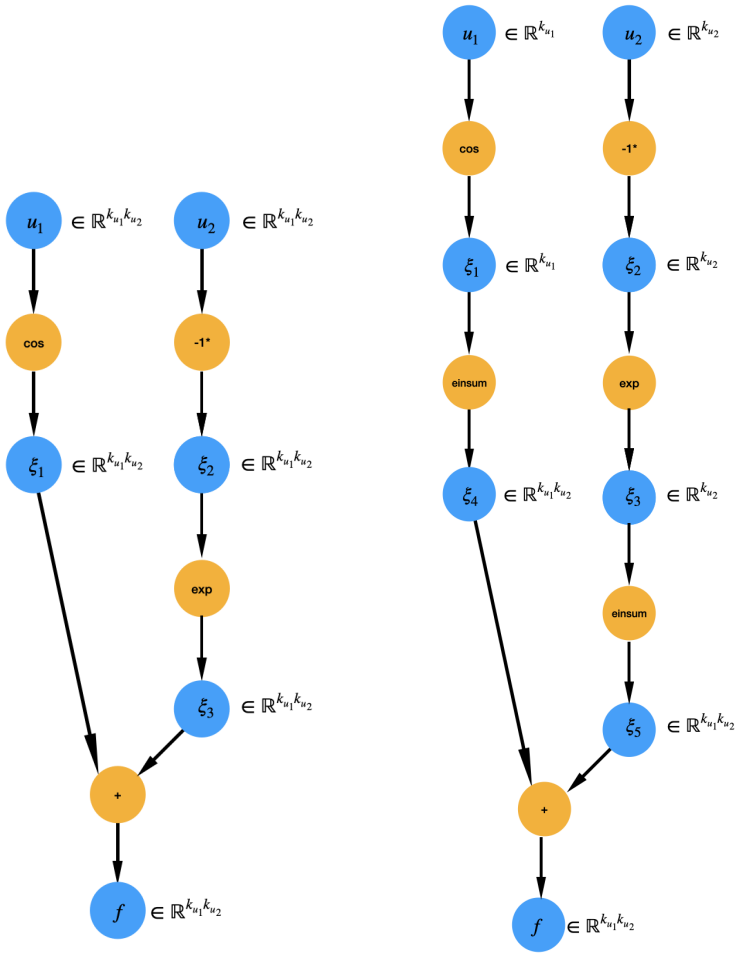
The designed quadrature method approximates the solution of the moment matching equations by solving an optimization problem formulated as

$$\begin{aligned} &\min_{\boldsymbol{u},\boldsymbol{w}} \|r\|_2 \\ &\text{subject to } u^{(j)} \in \Gamma, \quad j = 1, \ldots, n, \\ &\qquad\qquad w^{(j)} > 0, \quad j = 1, \ldots, n, \end{aligned} \tag{29}$$

in which it minimizes the norm of the residual equations and constrains the nodes to be within the support range and weights to be positive. This method has been tested to be more effective than the spars-grid Gauss quadrature rule for many high-dimensional integration problems, and the number of quadrature points required is chosen as $n = \eta \frac{(2d)^{k-1}}{(k-1)!}$, where $\eta \in [0.5, 0.9]$ is a tuning parameter.

E. Accelerated tensor-grid evaluations

Recently, a new method called accelerated tensor-grid evaluations (ATE) was proposed in [30]. The ATE method acts as a middle-end algorithm to modify the computational graph so that the model evaluation cost on the tensor-grid inputs can be reduced, this method has been implemented in a new modeling language called the Computational System Design Language (CSDL) [31]. The ATE method takes advantage of the tensor structure of the quadrature points and reduces the evaluation cost on the operation level. For example, if we want to evaluate a model on the full-grid Gauss quadrature points as in (23), there is a total of k^d number of quadrature points, but for each input dimension, there are only k distinct values. In this case, when we view the model as a computational graph of elementary functions and operations, the output of each elementary operation or function only has distinct values on the quadrature points within the space of the dependent inputs (the inputs that the output is dependent on). The ATE method modifies the



(a) Computational graph before applying ATE

(b) Computational graph after applying ATE

Fig. 2 Computational graphs with data size for full-grid quadrature points evaluations on $f = cos(u_1)+exp(-u_2)$

computational graph so that each operation only performs evaluation on the distinct values by using the dependency information of the graph. For example, for a simple function

$$f = cos(u_1) + exp(-u_2), \tag{30}$$

suppose we need to evaluate this function on full-grid quadrature points with k_1 and k_2 number of points in u_1 and u_2 dimensions, respectively. The quadrature points are given as

$$\boldsymbol{u} = \begin{Bmatrix} (u_1^{(1)}, u_2^{(1)}) & \dots & (u_1^{(k_1)}, u_2^{(1)}) \\ \vdots & \ddots & \vdots \\ (u_1^{(1)}, u_2^{(k_2)}) & \dots & (u_1^{(k_1)}, u_2^{(k_2)}) \end{Bmatrix}. \tag{31}$$

In Fig.2 we show the computational graphs before and after applying the ATE method for these function evaluations. In this case, after applying the ATE method, we are evaluating the ξ_1, ξ_2 and ξ_3 for fewer number times as they are only dependent on u_1 or u_2, and two Einstein summation nodes are added to the graph so that the data size matches between operations. The ATE method can significantly accelerate the model evaluation cost of the full-grid NIPC method on some multi-disciplinary and multi-point UQ problems, and in these cases, the full-grid NIPC with ATE can be the most efficient UQ approach.

III. Motivation

If we do not use the ATE method to accelerate the model evaluations, the quadrature rule with n quadrature points simply evaluates the entire model n times. If we assume the model evaluation cost on each quadrature point is roughly the same, the total evaluation cost can be approximated as:

$$\text{Cost} \approx nO\left(\mathcal{F}(u)\right), \tag{32}$$

where $O(\cdot)$ is the evaluation cost of a function or an operation. In this situation, the only way to reduce the model evaluation cost of NIPC is to reduce the number of quadrature points, n. For integration problems, the full-grid Gauss quadrature rule requires $n = k^d$ quadrature points to exactly integrate the polynomials up to $(2k-1)$ th order. For the designed quadrature method, it requires $n = \eta\frac{(2d)^{k-1}}{(k-1)!}$ quadrature points that can highly accurately integrate polynomials up to the same order. The designed quadrature method vastly outperforms the full-grid Gauss quadrature in multi-dimensional problems, as the required number of quadrature points of designed quadrature increases at a much lower rate with the number of dimensions than the full-grid Gauss quadrature rule.

If we use the ATE method to accelerate the model evaluations, for many UQ problems, the model evaluations on the full-grid Gauss quadrature method can be greatly reduced without reducing n. The performance of the ATE method depends on the sparsity of the uncertain inputs in the computational graph. Here, we refer to sparsity as the portion of operations that are dependent on uncertain inputs. For example, if most of the operations in the computational graph are dependent on an uncertain input, we call this uncertain input a dense uncertain input in the computational graph. Now, we have two efficient quadrature rules to use for the integration-based NIPC method. On one side, we have the full-grid quadrature rule, which maintains a fully tensorial structure for the quadrature points. Although the number of quadrature points increases exponentially with the dimensions, its tensorial structure can be utilized by the ATE method and the evaluation cost can be greatly reduced for computational models with sparse uncertain inputs in its computational graphs. On the other side, we have the designed quadrature rule. This method maintains a non-tensorial structure for the quadrature points, meaning the model evaluations cannot be accelerated by the ATE method. However, the number of quadrature points in the designed quadrature rule is often smaller than the full-grid Gauss quadrature rule, making it the more efficient approach when most of the uncertain inputs are dense in the computational graph.

The motivation of this work is that, for some computational models, when the ATE method is available, there may exist a partially tensorial structure that is more efficient to use than the fully tensorial and non-tensorial structures, but there is no method to determine if such a sturcture exists. In many UQ problems with a medium or large number of uncertain inputs, one typically has some sparse uncertain inputs in the computational graph, while the other uncertain inputs are dense. In this case, neither the full-grid Gauss quadrature rule nor the designed quadrature rule would be the optimal choice to use in the NIPC method. The intuition is that, for the space of dense inputs, there is no need to maintain a tensorial structure and we can use the designed quadrature method which results in fewer quadrature points while ensuring the same level of accuracy. For the space of sparse inputs, we want to maintain a tensorial structure between them, so that their sparsity in the computational graph can be taken advantage of by the ATE method. As a result, for certain UQ problems with complex models, there may exist a set of partially tensor-structured quadrature rules that achieves the same level of accuracy but results in the lowest model evaluation cost when ATE is available.

IV. Methodology

The objective of this paper is to propose a method to find the optimal tensorial structure and determine the quadrature points given a computational graph of the model, so that the model evaluation cost is minimized while maintaining the same level of accuracy of the quadrature rule. For a quadrature rule, we use the maximum order of polynomials that the quadrature rule can accurately integrate as a measure of accuracy. For the same level of accuracy, if we maintain a non-tensorial structure for the quadrature points, the optimal method is the designed quadrature method. In this case, we write the quadrature points as

$$\boldsymbol{u} = \boldsymbol{u}^n_{\{1,\dots,d\}}. \tag{33}$$

The nodes and weights in the designed quadrature method are determined at the same time by solving the optimization problem in (29), and the number of quadrature points is chosen as

$$n = \eta\frac{(2d)^{k-1}}{(k-1)!}, \quad \eta \in [0.5, 0.9]. \tag{34}$$

The designed quadrature rule can accurately integrate polynomials up to $(2k-1)$ th order.

If we maintain a fully tensorial structure for the quadrature points, the optimal method is the full-grid Gauss quadrature method in which we write the quadrature points as

$$\boldsymbol{u} = \boldsymbol{u}^k_{\{1\}} \times \cdots \times \boldsymbol{u}^k_{\{d\}}, \tag{35}$$

where $\boldsymbol{u}^k_{\{i\}}$ are the the 1D Gauss quadrature points in u_i dimension. The sets of 1D Gauss quadrature points in each dimension are determined separately so that the nodes and weights can exactly integrate the univariate polynomials up to $(2k-1)$ th order. By maintaining a fully tensorial structure of the 1D Gauss quadrature points and weights, the full-grid quadrature rule can exactly integrate cross-polynomials up to $(2k-1)$ th order, but it requires

$$n = k^d \tag{36}$$

number of quadrature points. In fact, in each dimension, the 1D Gauss quadrature points and weights correspond to global minimum of the optimization problem in (29). If we assume the optimization problem in (29) always finds the global minimum, applying the full-grid Gauss quadrature rule is equivalent to applying the designed quadrature rule using k points in each dimension and then formulating a fully tensorial structure between the 1D solutions.

We can also maintain a partially tensorial structure for the quadrature points and make sure the quadrature rule accurately integrates all of the polynomials to a certain order. For example, for a 4D integration problem, we can maintain a partially tensorial structure like

$$\boldsymbol{u} = \boldsymbol{u}^{n_1}_{\{1,2\}} \times \boldsymbol{u}^{n_2}_{\{3,4\}}, \tag{37}$$

in which we maintain a tensorial structure between the space of u_1, u_2 and the space of u_3, u_4. If the quadrature points in each space are chosen such that the quadrature points can accurately integrate polynomials up the same order like $(2k-1)$, then the partially tensorial quadrature rule can accurately integrate the polynomials to the same order in the space of u. In fact, the number of combinations of tensorial structures for the quadrature points can be represented by the Bell number from combinatorial mathematics, which can be written as

$$B_d = \sum_{i=0}^{d} \left\{ \begin{array}{c} i \\ d \end{array} \right\}. \tag{38}$$

For example, for $d = 3$, $B_3 = 5$ and the five options of the tensorial structure are:

$$\begin{aligned}
&\boldsymbol{u} = \boldsymbol{u}^{n}_{\{1,2,3\}}, \\
&\boldsymbol{u} = \boldsymbol{u}^{n_1}_{\{1,2\}} \times \boldsymbol{u}^{n_2}_{\{3\}}, \\
&\boldsymbol{u} = \boldsymbol{u}^{n_1}_{\{1,3\}} \times \boldsymbol{u}^{n_2}_{\{2\}}, \\
&\boldsymbol{u} = \boldsymbol{u}^{n_1}_{\{2,3\}} \times \boldsymbol{u}^{n_2}_{\{1\}}, \\
&\boldsymbol{u} = \boldsymbol{u}^{n_1}_{\{1\}} \times \boldsymbol{u}^{n_2}_{\{2\}} \times \boldsymbol{u}^{n_3}_{\{3\}}.
\end{aligned} \tag{39}$$

Once we generate all of the tensorial structure options for quadrature points, we need to decide on the required number of quadrature points for each option so that the quadrature rule can achieve a specific level of accuracy. In our method, for a specific tensorial structure of the quadrature rule, we apply the designed quadrature method in each space to solve for the nodes and weights such that they can accurately integrate the polynomials in that space up to an order, e.g. $(2k-1)$. The number of quadrature points used in the original designed quadrature method is in (34). This number is only practical for high-dimensional integration problems, but in our method, we will often face a low-dimensional space to apply the designed quadrature method. In our method, to achieve k level of accuracy (accurately integrate polynomials up to $(2k-1)$ th order in the integration space), we choose the number of designed quadrature points to be

$$n = \begin{cases} \frac{k^d}{d!} & \text{for } d \leq 2 \\ 0.9\frac{(2d)^{k-1}}{(k-1)!} & \text{for } d > 2, \end{cases} \tag{40}$$

where for dimension greater than two, we use the same rule as in (34) with $\eta = 0.9$, while for 1D and 2D case, we choose the number as $n = \frac{k^d}{d!}$. In this way, in the 1D space, it chooses k quadrature points to achieve k level of accuracy, which matches the Gauss quadrature rule. Therefore, for the 1D space, we can avoid solving the optimization problem and use the 1D Gauss quadrature rule, as they are the exact solution of the designed quadrature method.

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

After determining the number of points for each tensorial structure option, we need to predict the evaluation cost for each option with the ATE method. To predict that, we need the dependency and cost information of each operation in the computational graph. Note that the way ATE method works is to only evaluate on the distinct values for each operation, and this number of distinct values is dependent on the distinct values in the space of the dependent random inputs (the random inputs that the operation is dependent on). For a computational model $f = \mathcal{F}(u)$, we represent the data and operations nodes as ξ_i and φ_j, respectively, with l number of operations in the computational graph. For example, for the function

$$f = cos(u_1) + exp(-u_2), \tag{41}$$

the computational graph of it can be described as

$$\begin{aligned} \xi_1 &= \varphi_1(u_1) = cos(u_1); \\ \xi_2 &= \varphi_2(u_2) = -u_2; \\ \xi_3 &= \varphi_3(\xi_2) = exp(\xi_2); \\ f &= \varphi_4(\xi_2, \xi_3) = \xi_2 + \xi_3, \end{aligned} \tag{42}$$

with $l = 4$ number of operations. From the computational graph, we generate the dependency information which stores whether the output of one operation is dependent on one input as a binary number, written as $D(\varphi_i, u_j)$. The dependency information can be viewed in a matrix. For example, the dependency matrix of the function (41) is shown in Tab. 2. We also need to store the information for each operation's evaluation cost on one quadrature point, this information can be represented either by using the total flops of that operation or the evaluation time. With all of this information, we can estimate the evaluation cost of each tensorial structure for the quadrature rule to achieve k-level of accuracy. For the non-tensorial structure, there is hardly any repeated evaluation that can be reduced by the ATE method. If we assume the operations' cost on all of the quadrature points is roughly the same, the total model evaluation cost can be approximated as:

$$\text{Cost} \approx nO\left(\mathcal{F}(u^{(1)})\right) \approx n\sum_{i=1}^{l} O(\varphi_i), \tag{43}$$

where n is chosen as in (40). If we maintain a full-grid structure with k quadrature points in each dimension, the evaluation cost without the ATE method is

$$\text{Cost} \approx k^d O\left(\mathcal{F}(u^{(1)})\right) \approx k^d \sum_{i=1}^{l} O(\varphi_i). \tag{44}$$

However, with the ATE method, each operation in the graph is evaluated only for distinct values. In the full-grid case, if the output of an operation is dependent on $\tilde{d}$ number of uncertain inputs, there are only $k^{\tilde{d}}$ quadrature points in that space, and that operation will be evaluated for $k^{\tilde{d}}$ number of times in the ATE method. In this case, we omit the cost of the Einstein summation nodes added to the computational graph by the ATE method, the evaluation cost can be estimated as

$$\text{Cost} \approx \sum_{i=1}^{l} k^{\sum_{j=1}^{d} D(\varphi_i, u_j)} O(\varphi_i). \tag{45}$$

Using the same idea, we can also estimate the model evaluation cost for any tensorial structure option of the quadrature points. For example, if we have a partially tensorial structure for the quadrature points, such as

$$\boldsymbol{u} = \boldsymbol{u}_{\{1,2\}}^{n_1} \times \boldsymbol{u}_{\{3\}}^{n_2}. \tag{46}$$

In our method, the n_1 and n_2 are chosen from (40) with $d = 1$ and $d = 2$ respectively. In this case, there are n_1 quadrature points in the space of u_1 and u_2, and the n_2 quadrature points in the space of u_3. With the ATE method, for the operations that are dependent on only u_1, only u_2 or only u_1 and u_2, they will be evaluated n_1 times. Operations dependent on only u_3, will be evaluated n_2 times. Operations not dependent on any input are evaluated once. The rest of the operations are evaluated $n_1 n_2$ times. The model evaluation's cost for this tensor-structured quadrature rule can be estimated as

$$\text{Cost} \approx \sum_{i=1}^{l} n_1^{D(\varphi_i,u_1)D(\varphi_i,u_2)} n_2^{D(\varphi_i,u_3)} O(\varphi_i). \tag{47}$$

Our method is described in Alg.1. Given the computational graph of the model in the integration problem and a specific level of accuracy for the quadrature rule, k. We first generate the dependency information for all of the operations and evaluate the model on one input point to store the evaluation time for each operation. For the integration problem, we generate all of the tensorial structure options for the quadrature rule and compute the required number of quadrature points for each option. Then we estimate the model evaluation cost of each option when using the ATE method. The optimal tensorial structure option is chosen to have the lowest estimated evaluation cost. For the optimal tensorial structure chosen, we use the designed quadrature method to compute the nodes and weights in each space to achieve the k level of accuracy. For the 1D space, we directly use the 1D Gauss quadrature rule as they are the exact solutions of the designed quadrature method in 1d. We then formulate the quadrature rule for the whole input space using the optimal tensorial structure. Subsequently, we evaluate the computational model on the quadrature points we computed and we use the ATE method to reduce the model evaluation cost. Finally, the integral is computed using the model evaluations on the quadrature points and the weights from the quadrature rule.

For an integration problem with the computational graph information available, our method finds an optimal tensorial structure of the quadrature rule to achieve a specified level of accuracy. The optimal tensorial structure can be a non-tensorial structure, fully-tensorial structure, or a partially tensorial structure. For a non-tensorial structure, the quadrature rule generated from our method is the same as the designed quadrature method, while for a fully-tensorial structure, our method generates the full-grid Gauss quadrature rule. For a partially tensorial structure, our method generates a unique quadrature rule, which is different from the existing methods. The unique quadrature rule is tailored for the given computational model, and is often more efficient than the existing methods when the ATE method is available.

Algorithm 1 Optimally tensor-structured quadrature rule for integration problems

1: Specify a computational graph for a numerical model and the inputs to be integrated over, u
2: Specify the level of accuracy of the quadrature rule, k
3: Generate the dependency information of all the operations from the computational graph
4: Run model evaluation on a single quadrature point and store the evaluation time for each operation
5: Generate all of the tensorial structure options for the quadrature rule
6: Determine the number of quadrature points for each option
7: Predict the evaluation cost of each option with the ATE method
8: Select the optimal tensorial structure option
9: Compute the quadrature points and weights in each space using the designed quadrature method
10: Formulate the quadrature rule using the optimal tensorial structure
11: Evaluate the model with ATE on the quadrature points
12: Compute the integral using the model evaluations and their corresponding weights

V. Numerical Results

A. Test problem

We tested our problem on a 4D UQ problem with a low-fidelity multidisciplinary model. The model computes the total energy stored for a cruise mission of a laser-beam-powered UAV circling around a ground station once while being charged by the laser beam from the ground station. The mission is demonstrated in Fig. 3. We consider four random inputs in the model, which are flight velocity, flight altitude, payload weight, and atmospheric extinction coefficient. Their distributions along with the critical parameters of this problem are shown in Tab.3.

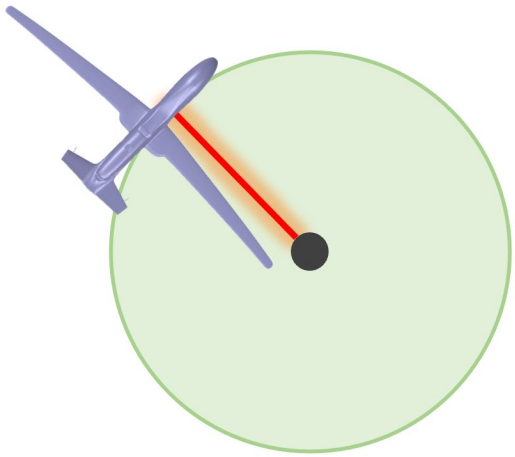


Fig. 3 A circular cruise mission around a ground station

Table 3 Critical parameters in the UQ problem

Airfoil	NACA 2412
Wing area (m^2)	11.45
Aspect ratio	19
Payload weight W_p (kg)	$N(90, 30)$
Atmospheric extinction coefficient σ	$N(0.2, 0.1)$
Flight altitude h (m)	$N(10,000, 2000)$
Radius of the flight circle r (m)	1000
Flight velocity v (m/s)	$N(100, 20)$
Laser beam power transmitted $P_{\text{tra}}(kw)$	50

B. Modelling

The computational model for this problem consists of five disciplines: atmosphere, laser beam, weights, aerodynamics, and performance. The structure of this multidisciplinary model is shown in Fig.4. For the atmosphere model, we trained a surrogate model that takes the inputs of velocity and altitude and outputs the density of air and dynamics pressure. For the laser beam model, we used the Beers-Lambert law for optical power transfer [32] to compute the power received by the aircraft as

$$P_{\text{rec}} = \eta P_{\text{tra}}, \quad \eta = e^{-\sigma R}, \tag{48}$$

where P_{rec} and P_{tra} are the power received and transmitted, respectively, σ is the atmospheric extinction coefficient describing the weather condition and R is the distance between the aircraft and the power source. For the weights model, we give specific weights for the motor, battery, and payloads. The aircraft component weights including wing, fuselage, and tail are computed using the simple statistical equations for general aviation weights from [33]. The weights model outputs the total weight of the aircraft, W. The aerodynamic model takes inputs of total weights and flight velocity. It computes the total drag force on the wing to maintain a steady state flight. The lift coefficient and drag coefficient are computed by fitting a linear model and a quadratic polar model, respectively, from airfoil data. The performance model

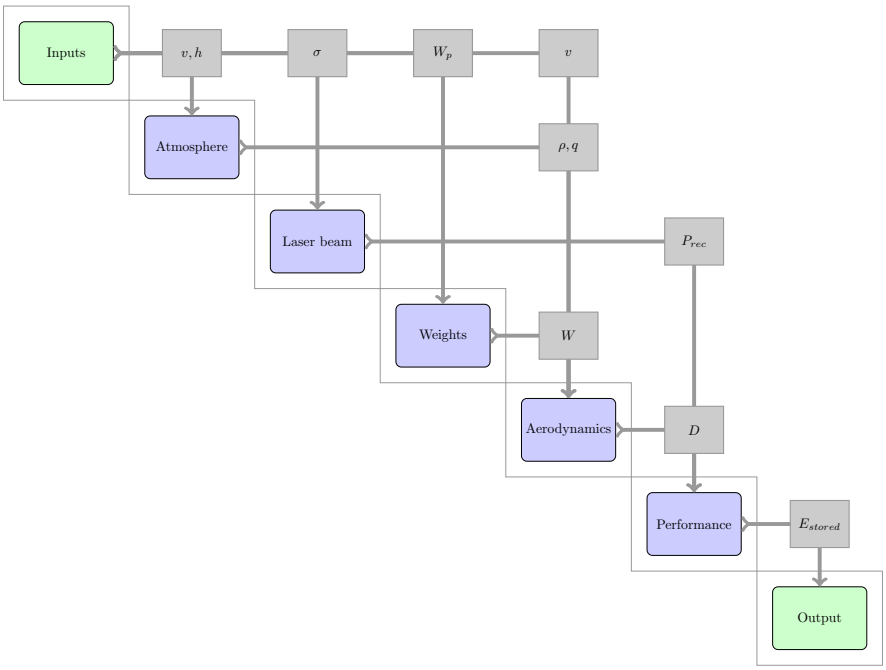


Fig. 4 Data flow of the multidisciplinary model used in the UQ problem

takes the input of flight velocity and total drag force and computes the total energy stored during the mission as

$$\begin{aligned} n &= \sqrt{\left(\frac{v}{rg}\right)^2 + 1}, \\ P_{\text{req}} &= \frac{D}{n}, \\ E_{\text{stored}} &= (P_{\text{rec}} - P_{\text{req}})t, \end{aligned} \tag{49}$$

where n is the turn load, r is the radius of the mission, g is the gravitational acceleration, D is the total drag force, P_{req} is the required power to maintain the steady flight state and t is the mission time.

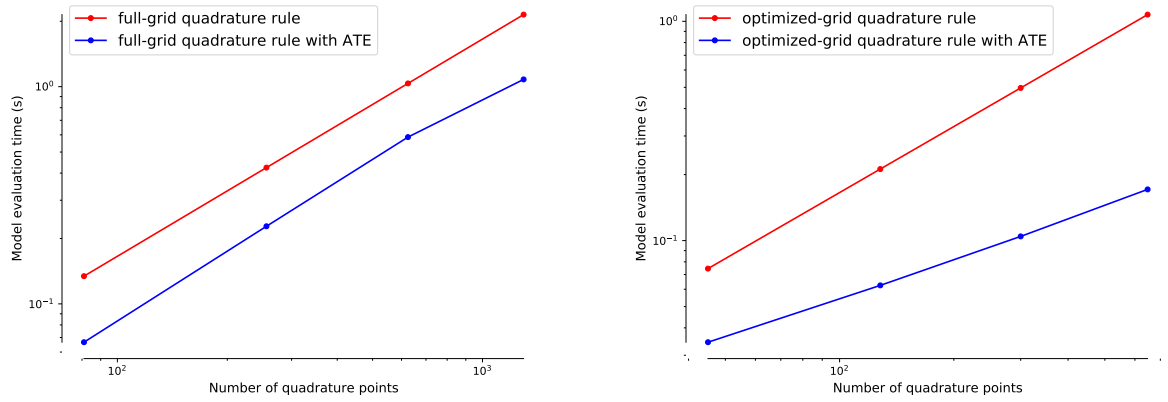
C. Performance

The UQ problem is solved using three UQ methods: integration-based NIPC, kriging and the Monte Carlo (MC) method. For the integration problem introduced from the integration-based NIPC, we apply three numerical quadrature approaches to solve it, which are the full-grid Gauss quadrature rule (full-grid), designed quadrature, and the optimally tensor-structured quadrature rule (optimized-grid). We consider two scenarios for the full-grid NIPC method and optimized-grid NIPC method: with the ATE method and without the ATE method.

In Fig.5, it shows the model evaluation time versus the number of quadrature points before and after applying the ATE method for both the full-grid and optimized-grid quadrature rules. For the full-grid quadrature rule, we observe a consistent 40% reduction in the model evaluation time when using the ATE method. For the optimally tensor-structured quadrature rule, the reduction when using ATE is even more significant and increases to 90% with the largest number of quadrature points we tested.

In Fig. 6, we show the convergence plots for all seven ways to solve this UQ problem. From the convergence plots, the Monte Carlo and kriging methods have the slowest convergence rate compared to the NIPC methods. For the NIPC methods without using ATE, the designed quadrature method has the best performance, since it requires the lowest number of quadrature points to achieve the same level of accuracy. The designed quadrature method also outperforms the full-grid NIPC method when ATE is available. However, for the optimized-grid NIPC, the ATE method significantly reduced its model evaluation cost, making the optimized-grid NIPC method the most efficient UQ approach for this

model. We observe that the optimized-grid NIPC requires at least 50% less model evaluation cost to achieve the same level of accuracy compared to all the UQ methods we implemented.



(a) Full-grid Gauss quadrature rule **(b) Optimally tensor-structured quadrature rule**

Fig. 5 Comparison of the tensor-structured quadrature rules before and after applying the ATE method

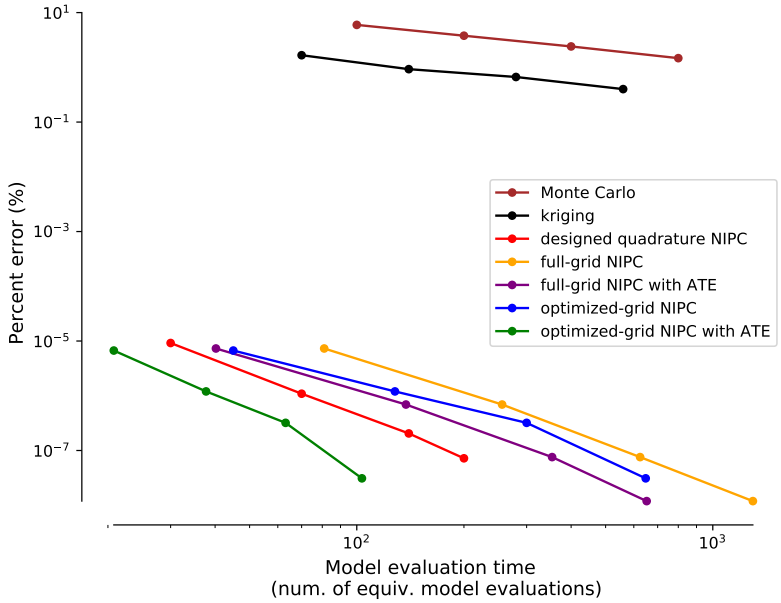


Fig. 6 Convergence plots for the seven UQ methods

D. Analysis

To understand why the optimized-grid NIPC is so efficient on this problem, we can examine the optimal tensorial structure our method found. The optimal tensorial structure found in our method is a partially tensorial structure with the quadrature points described as:

$$\boldsymbol{u} = \boldsymbol{u}^{n_1}_{\{h,v\}} \times \boldsymbol{u}^{n_2}_{\{P_w\}} \times \boldsymbol{u}^{n_3}_{\{\sigma\}}, \tag{50}$$

Table 4 Number of dependent operations for each random input on the computational graph

	h	v	P_w	σ
No. of dependent operations	162	141	53	21
Total operations	285			

in which it maintains a tensorial structure between the space of h and v, the space of P_w, and the space of σ. The dependency information on this computational graph can be summarized by counting the number of dependent operations on each random input, which is shown in Tab.4. The table shows that both h and v are relatively dense inputs in the computational graph with around half of the operations dependent on them, while the P_w and σ are the sparse inputs with less than 25% of the operations dependent on them. The optimized tensorial structure in (50) combined the space of h and v so that the number of quadrature points is reduced in that space, while maintaining the tensorial structure with the space of two sparse inputs so that the sparsity structure of the graph structure can be taken advantage of by the ATE method. As a result, this set of optimized-grid quadrature points forms a more efficient quadrature rule than the existing quadrature rules on this integration problem introduced by NIPC.

VI. Conclusion

In this paper, we proposed a new method that finds optimal tensor-structure quadrature rules that result in the lowest model evaluation cost when using the ATE method. This method is based on the idea of the designed quadrature method but determines the optimal tensorial structure based on the computational graph information. We used a UQ problem with a low-fidelity multidisciplinary model involving a laser-beam-powered UAV to test our method. The optimally tensor-structured quadrature rule on this problem maintains a partially tensorial structure, which is different from the existing quadrature rules. For this quadrature rule, the model evaluation cost can be significantly reduced with the ATE method. When applied to the integration-based NIPC method to solve this UQ problem, this method is at least 50% more efficient than all of the UQ methods we compared.

We do not expect this method to generate a new quadrature rule for every problem. For some problems with dense computation graph structures, when maintaining the non-tensorial structure is optimal, our method will generate the same quadrature rule as the designed quadrature method. For some problems with sparse computational graph structures, when maintaining a fully tensorial structure is the optimal choice, our method will generate the same quadrature rule as the full-grid Gauss quadrature rule. For many UQ problems with several random inputs and complex computational graphs, this method may find a partially tensor-structured quadrature rule that can achieve a given level of accuracy with the lowest model evaluation cost using the ATE method.

A limitation of our method is that our method needs to generate all of the tensor-structure options for the quadrature rule and predicts the cost for each of them in order to find the optimal option. The number of options increases significantly with the number of dimensions as in (38), so it will become intractable for integration problems with more than 10 dimensions. An interesting topic of future work is to find a more efficient method for finding the optimal tensorial structure for the quadrature rule.

VII. Acknowledgement

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

References

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

[2] Montomoli, F., Carnevale, M., D'Ammaro, A., Massini, M., and Salvadori, S., *Uncertainty quantification in computational fluid dynamics and aircraft engines*, Springer, 2015.

[3] Wan, H.-P., Mao, Z., Todd, M. D., and Ren, W.-X., "Analytical uncertainty quantification for modal frequencies with structural parameter uncertainty using a Gaussian process metamodel," *Engineering Structures*, Vol. 75, 2014, pp. 577–589.

[4] Hu, Z., Mahadevan, S., and Ao, D., "Uncertainty aggregation and reduction in structure–material performance prediction," *Computational Mechanics*, Vol. 61, No. 1, 2018, pp. 237–257.

[5] Albi, G., Pareschi, L., and Zanella, M., "Uncertainty quantification in control problems for flocking models," *Mathematical problems in Engineering*, Vol. 2015, 2015.

[6] Lall, U., Josset, L., and Russo, T., "A snapshot of the world's groundwater challenges," *Annual Review of Environment and Resources*, Vol. 45, 2020, pp. 171–194.

[7] Zhang, J., Man, J., Lin, G., Wu, L., and Zeng, L., "Inverse modeling of hydrologic systems with adaptive multifidelity Markov chain Monte Carlo simulations," *Water Resources Research*, Vol. 54, No. 7, 2018, pp. 4867–4886.

[8] Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., Fieguth, P., Cao, X., Khosravi, A., Acharya, U. R., et al., "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information Fusion*, Vol. 76, 2021, pp. 243–297.

[9] Yao, W., Chen, X., Luo, W., Van Tooren, M., and Guo, J., "Review of uncertainty-based multidisciplinary design optimization methods for aerospace vehicles," *Progress in Aerospace Sciences*, Vol. 47, No. 6, 2011, pp. 450–479.

[10] Beyer, H.-G., and Sendhoff, B., "Robust optimization–a comprehensive survey," *Computer methods in applied mechanics and engineering*, Vol. 196, No. 33-34, 2007, pp. 3190–3218.

[11] Park, G.-J., Lee, T.-H., Lee, K. H., and Hwang, K.-H., "Robust design: an overview," *AIAA journal*, Vol. 44, No. 1, 2006, pp. 181–191.

[12] Tu, J., Choi, K. K., and Park, Y. H., "A new study on reliability-based design optimization," 1999.

[13] Fragkos, K., Papoutsis-Kiachagias, E., and Giannakoglou, K., "pFOSM: An efficient algorithm for aerodynamic robust design based on continuous adjoint and matrix-vector products," *Computers & Fluids*, Vol. 181, 2019, pp. 57–66.

[14] Peherstorfer, B., Willcox, K., and Gunzburger, M., "Optimal model management for multifidelity Monte Carlo estimation," *SIAM Journal on Scientific Computing*, Vol. 38, No. 5, 2016, pp. A3163–A3194.

[15] Peherstorfer, B., Willcox, K., and Gunzburger, M., "Survey of multifidelity methods in uncertainty propagation, inference, and optimization," *Siam Review*, Vol. 60, No. 3, 2018, pp. 550–591.

[16] Hosder, S., Walters, R., and Perez, R., "A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations," *44th AIAA aerospace sciences meeting and exhibit*, 2006, p. 891.

[17] Jones, B. A., Doostan, A., and Born, G. H., "Nonlinear propagation of orbit uncertainty using non-intrusive polynomial chaos," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 430–444.

[18] Keshavarzzadeh, V., Fernandez, F., and Tortorelli, D. A., "Topology optimization under uncertainty via non-intrusive polynomial chaos expansion," *Computer Methods in Applied Mechanics and Engineering*, Vol. 318, 2017, pp. 120–147.

[19] Xiu, D., and Hesthaven, J. S., "High-order collocation methods for differential equations with random inputs," *SIAM Journal on Scientific Computing*, Vol. 27, No. 3, 2005, pp. 1118–1139.

[20] Babuška, I., Nobile, F., and Tempone, R., "A stochastic collocation method for elliptic partial differential equations with random input data," *SIAM Journal on Numerical Analysis*, Vol. 45, No. 3, 2007, pp. 1005–1034.

[21] Wiener, N., "The homogeneous chaos," *American Journal of Mathematics*, Vol. 60, No. 4, 1938, pp. 897–936.

[22] Ghanem, R., "Ingredients for a general purpose stochastic finite elements implementation," *Computer methods in applied mechanics and engineering*, Vol. 168, No. 1-4, 1999, pp. 19–34.

[23] Xiu, D., and Karniadakis, G. E., "The Wiener–Askey polynomial chaos for stochastic differential equations," *SIAM journal on scientific computing*, Vol. 24, No. 2, 2002, pp. 619–644.

[24] Lee, D., and Rahman, S., "Practical uncertainty quantification analysis involving statistically dependent random variables," *Applied Mathematical Modelling*, Vol. 84, 2020, pp. 324–356.

[25] Luthen, N., Marelli, S., and Sudret, B., "Sparse polynomial chaos expansions: Literature survey and benchmark," *SIAM/ASA Journal on Uncertainty Quantification*, Vol. 9, No. 2, 2021, pp. 593–649.

[26] Eldred, M., and Burkardt, J., "Comparison of non-intrusive polynomial chaos and stochastic collocation methods for uncertainty quantification," *47th AIAA aerospace sciences meeting including the new horizons forum and aerospace exposition*, 2009, p. 976.

[27] Golub, G. H., and Welsch, J. H., "Calculation of Gauss quadrature rules," *Mathematics of computation*, Vol. 23, No. 106, 1969, pp. 221–230.

[28] Smolyak, S. A., "Quadrature and interpolation formulas for tensor products of certain classes of functions," *Doklady Akademii Nauk*, Vol. 148, Russian Academy of Sciences, 1963, pp. 1042–1045.

[29] Keshavarzzadeh, V., Kirby, R. M., and Narayan, A., "Numerical integration in multiple dimensions with designed quadrature," *SIAM Journal on Scientific Computing*, Vol. 40, No. 4, 2018, pp. A2033–A2061.

[30] Wang, B., Sperry, M., Gandarillas, V. E., and Hwang, J. T., "Efficient uncertainty propagation through computational graph modification and automatic code generation," *AIAA AVIATION 2022 Forum*, 2022, p. 3997.

[31] Gandarillas, V., Joshy, A. J., Sperry, M. Z., Ivanov, A. K., and Hwang, J. T., "A graph-based methodology for constructing computational models that automates adjoint-based sensitivity analysis," *Structural and Multidisciplinary Optimization*, (submitted).

[32] Kim, I. I., McArthur, B., and Korevaar, E. J., "Comparison of laser beam propagation at 785 nm and 1550 nm in fog and haze for optical wireless communications," *Optical wireless communications III*, Vol. 4214, Spie, 2001, pp. 26–37.

[33] Raymer, D., *Aircraft design: a conceptual approach*, American Institute of Aeronautics and Astronautics, Inc., 2012.