OPTIMIZED EXTENSIONS OF ARBITRARY CUBATURES

by

MATTHEW W. FISTER

SAMEER B. MULANI, COMMITTEE CHAIR
VISHESH VIKAS
XIAOWEN WANG

A THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Aerospace Engineering and Mechanics
in the Graduate School of
The University of Alabama

TUSCALOOSA, ALABAMA

2021

ABSTRACT

An optimization of cubature (OOC) method is developed to refine computationally intensive numerical integrations by optimally extending an existing cubature. Typically, to increase the accuracy of numerical integrations using methods which cannot be nested, such as Gaussian quadrature, the integrand is evaluated over a larger, disjoint set of abscissas, without using the previous integrand evaluations. The developed OOC method adds any number of abscissas to the existing quadrature and reevaluates all associated weights. To optimize the abscissas and weights, a global optimization technique is used to minimize the sum of squared numerical integration error for a set of training functions, calculated as a function of the optimized weights and abscissas. The training functions must have a known integral over a hypercube domain. The abscissas are constrained to the domain and weights are constrained to be positive. The optimized weights and abscissas are then used to numerically integrate a function of over the domain. Additionally, a method for the optimization of weights, a variant of the OOC method in which only cubature weights are optimized, is presented and compared to traditional methods such as Gauss quadrature and Bayesian quadrature. The OOC method performs well on multivariate integrands, however, the optimization required by the OOC method may add significant computational expense. Therefore, the OOC method is determined to be best suited for computationally expensive, multivariate integrands for which the number of integrand evaluations is severely limited.

DEDICATION

To Sam.

ACKNOWLEDGMENTS

I am very lucky to have received as much support as I did while writing this thesis. I'd like to thank Dr. Sameer Mulani for his guidance throughout the entire process. I'd also like to thank Dr. Vikas and Dr. Wang for serving on my committee. Lastly, I owe sincere thanks to my parents and to my wife Sam, who have been supporting me for much longer than it took to write this thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Motivation

Numerical integration is a fundamental building block for the computational tools in a multitude of disciplines including engineering, mathematics, finance, physics, and biology, to name just a few. Of particular relevance to aerospace engineering are finite element (FE) theory, computational fluid dynamics, and uncertainty quantification (UQ), all of which utilize numerical integration. While oftentimes a simple quadrature rule will suffice, such as Gauss quadrature (GQ) in FE theory, integrations over multiple dimensions may quickly become prohibitively expensive due to the "curse of dimensionality", which states that the computational expense of a numerical integration increases exponentially with the number of dimensions of the domain of integration. These multiple integrals often arise in the context of UQ [1]. Computing a polynomial chaos expansion (PCE) surrogate model using non-intrusive spectral projection (NISP) requires the integration over all surrogate model inputs [2]–[6]. These methods are extensively applied in stochastic CFD [4], [6]–[13]. Estimation of the posterior distribution in Bayesian inference similarly requires integration over a domain with an order equal to the number of inputs [14]. As the defense industry continues to push towards a wider adoption of UQ in an effort to reduce margins and better predict performance [15], [16], the need for efficient numerical computation of multiple integrals increases as well.

One way to reduce the computational expense is to employ adaptive numerical integra-

tion techniques. Adaptive techniques in some way incorporate information about the integrand into the integration algorithm. For example, adaptive sampling will evaluate the integrand at sample points which are of particular interest. These may be due to high gradients, high uncertainty, or other metrics [17]–[21]. These adaptive techniques may reduce computational expense if the integrand can be accurately integrated using dense sampling points only where needed. Examples of iterative integration suitable for higher order integrals are Monte-Carlo (MC) methods [22], Quasi-Monte Carlo (QMC) methods [23], and Bayesian quadrature (BQ) methods [24]–[29]. These methods have the added benefit of a trivially calculated integration error estimate which can serve as a convergence criteria.

1.2 Goals

The goal of the present thesis is to present and analyze a novel method for optimization of cubature (OOC) as well as a method for the optimization of weights (OOW), a variant of the OOC method in which cubature weights are optimized for a predetermined set of cubature points. These two methods will be compared to Gauss quadrature and Bayesian quadrature through a range of benchmarking studies and a practical engineering application. Through these comparisons, the applicability and utility of the presented methods will be assessed.

1.3 Outline

This thesis is organized as follows. Chapter 2 will present a survey of the literature, including relevant numerical integration techniques such as Gauss quadrature, Clenshaw-Curtis quadrature, Smolyak sparse grids, Bayesian integration, and adaptive sampling. Chapter 3 will present the methods employed by the author in detail. Chapter 4 will present a comparison of the chosen methods through the use of specific benchmark integrand functions found in the literature. Chapter 5 will demonstrate the use of the novel method for adaptive integration by propagating uncertainty through a stochastic computational fluid

dynamics (CFD) model. Finally, Chapter 6 will present conclusions drawn from the presented material and identify areas of future work which will benefit the state-of-the-art applications of multivariate numerical integration to engineering problems.

CHAPTER 2

BACKGROUND

Numerical integration methods approximate the integral of a function as a weighted sum of function evaluations over some domain D.

$$I = \int_D f(\mathbf{x})\mathbf{dx} \approx \sum_{i=1}^{n} w_i f(\mathbf{x}_i) \tag{2.1}$$

Numerical integration will not be exact except in special cases. The associated error is defined as the signed difference between the exact integral and the numerical integral.

$$E = I - \sum_{i=1}^{n} w_i f(\mathbf{x}_i) \tag{2.2}$$

It is valuable to be able to approximate this error. Accurate error estimations allow for automatic quadrature, in which the numerical integration is iteratively refined until the error estimate is below a certain threshold.

Typically numerical integration methods, especially quadrature rules, are defined to calculate an integral over the hypercube domain, $\Omega^d := [-1, 1]^d$. Any definite integral over a hyperrectangle can be mapped to the hypercube domain using an affine transformation without introducing additional error. Integrals over domains which cannot simply be mapped to the hypercube domain are handled differently and are not discussed here.

In this chapter several classes of numerical integration methods are presented. First are quadrature rules, of which several types are mentioned. These methods are suitable for low-dimensional integrations. Sparse grids are then introduced, which serve to extend

the applicability of quadrature rules into integrations with higher number of dimensions. Lastly, Monte Carlo-type methods and Bayesian quadrature are presented for use in integrations with the highest number of dimensions.

2.1 Newton-Cotes Formulas

The Newton-Cotes formulas include some of the most well known quadratures, including the midpoint rule, trapezoid rule, and Simpson's rule [30]. These quadratures consist of equally-spaced sample points and thus, while simple to understand and implement, are typically less accurate than other common quadratures such as Gauss quadrature and Clenshaw-Curtis quadrature. Nevertheless, when integrand evaluations are only given at regular intervals, e.g. data collected from a data acquisition system of a given sample rate, then Newton-Cotes formulas are useful numerical integration methods.

A Newton-Cotes formula of degree N approximates the integral $\int_a^b f(x)dx$ by sampling $f(x)$ at N points equally spaced with a step size of $h = \frac{b-a}{N}$. For a closed Newton-Cotes formula, the endpoints will be sampled. For an open Newton-Cotes formula, the endpoints are not sampled and the outermost sampling points are a distance $h/2$ from the endpoints. The weights associated with each sampling point are derived by assuming the integrand can be approximated by a Lagrange interpolating polynomial.

$$f(x) \approx \sum_{i=0}^{N} f(x_i) L_i(x) \tag{2.3}$$

The Lagrange polynomial has the form

$$L_i(x) = \prod_{j \neq i} \frac{x - x_i}{x_j - x_i} \tag{2.4}$$

for the sample points x_i. The integral of each Lagrange polynomial does not depend on the integrand, thus the weights are known, $w_i = \int_a^b L_i(x)dx$. Through a transform of variables, the weights of the quadrature can be calculated a priori using the integral of the

Lagrange polynomial over the domain $[0, 1]$ and simply scaled to the domain $[a, b]$. The first several closed Newton-Cotes formulas are given in Table 2.1.

Table 2.1: Closed Newton-Cotes quadratures.

Name	N	h	Formula	Error Estimation
Trapezoid rule	1	$b-a$	$\frac{h}{2}(f_0+f_1)$	$-\frac{1}{12}h^3 f^{(2)}(\xi)$
Simpson's rule	2	$\frac{b-a}{2}$	$\frac{h}{3}(f_0+4f_1+f_2)$	$-\frac{1}{90}h^5 f^{(4)}(\xi)$
Simpson's 3/8 rule	3	$\frac{b-a}{3}$	$\frac{3h}{8}(f_0+3f_1+3f_2+f_3)$	$-\frac{3}{80}h^5 f^{(4)}(\xi)$
Boole's rule	4	$\frac{b-a}{4}$	$\frac{2h}{45}(7f_0+32f_1+12f_2+32f_3+7f_4)$	$-\frac{8}{945}h^7 f^{(6)}(\xi)$

Although a Newton-Cotes formula of any arbitrary degree can be computed, typically only low-order quadratures are used. This is due to the Runge phenomena, which can cause higher-order Newton-Cotes quadratures to become unstable. To avoid this problem, it is common to use a composite, low-order, Newton-Cotes rule where the domain $[a, b]$ is split into equal intervals and the low-order quadrature is applied to each interval and summed to equal the integral over the whole.

In the case where a function is particularly nonlinear only on some subintervals of the domain, it might be wasteful to use a composite rule which equally divides the domain. Smaller subdomains may be needed where the integrand is nonlinear, and may be wasted where it is linear. In response to this common problem, adaptive quadratures are often used. Adaptive quadratures sample the integrand more densely where there is some perceived increase in difficulty. This sense of difficulty may be determined algorithmically from the previously sampled values, using measures such as gradients or variability.

While quadrature rules such as the Newton-Cotes formulas are defined for 1D functions, they can be applied to multi-dimensional integrals using a tensor product formulation.

These tensor product quadratures are commonly called cubatures. They have the form:

$$\begin{aligned}\int_{a_d}^{b_d} \dots \int_{a_1}^{b_1} f(x_1, \dots, x_d) dx_1 \dots dx_d &= c \int_{-1}^{1} \dots \int_{-1}^{1} f(y_1, \dots, y_d) dy_1 \dots dy_d \\ &\approx c \sum_{i_1=1}^{n_1} \dots \sum_{i_d=1}^{n_d} \Big(\prod_{k=1}^{d} w_{i,k} \Big) f(y_{i_1}, \dots, y_{i_d}) \end{aligned} \tag{2.5}$$

Compositing and adaptive quadrature can also be extended to multi-dimensional quadratures in an analogous manner.

Although Newton-Cotes rules can be easily nested and composited, the curse of dimensionality prevents the uniformly spaced abscissas from scaling well at high dimensions. Additionally, the relatively low accuracy compared to non-uniformly spaced quadratures makes Newton-Cotes the best option only when equally-spaced sampling points is a rigid constraint. Therefore, Newton-Cotes quadratures are best suited for numerically integrated functions in low dimensions where function evaluations are only known at uniformly sampled points.

2.2 Gauss Quadrature

Perhaps the most widely known quadrature, Gauss quadrature, and specifically Gauss-Legendre quadrature, is the de facto standard against which all other quadrature rules are measured. Gauss quadrature exactly integrates a polynomial of order $2n-1$ over the domain $[-1, 1]$ using only n samples. In short this is because Gauss quadrature trades predetermined sample point locations for increased accuracy. To appreciate this trade off we look to the Legendre polynomials [30].

Assume the integrand $f(x)$ is a polynomial of degree $2n-1$. The n-order Legendre polynomial is given as $q_n(x)$. Using polynomial division, we can express the integrand as $f(x) = p(x)q_n(x) + r(x)$, where $p(x)$ is a polynomial of degree $n-1$ and $r(x)$ is a polynomial of degree $n-1$. Due to the orthogonality of Legendre polynomials with respect to the weight

measure $w(x) = 1$,

$$\langle q_i(x), q_j(x) \rangle_w = \int_{-1}^{1} q_i(x) q_j(x) w(x) dx = \frac{2}{2i+1} \delta_{ij} \tag{2.6}$$

Any polynomial of order $n-1$, such as $p(x)$, can be exactly represented as a weighted sum of Legendre polynomials up to the $n-1$ order, and therefore the integrand of $f(x)$ over $[-1, 1]$ reduces to

$$\int_{-1}^{1} f(x) dx = \int_{-1}^{1} r(x) dx \tag{2.7}$$

Since $r(x)$ is a polynomial of only order $n-1$, it now seems natural that the derived Gauss-Legendre quadrature would only need n points. Fortunately, $r(x)$ is known at exactly n points, the roots of the Legendre polynomial $q(x)$ (Figure 2.1). These are the n-point Gauss-Legendre quadrature abscissas.

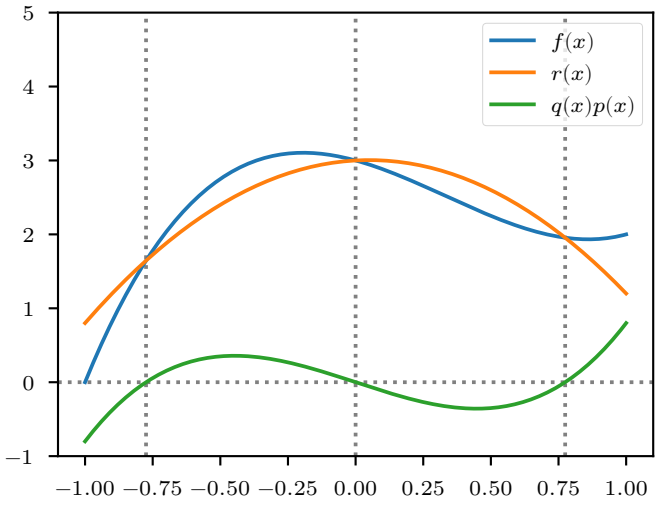


Figure 2.1: The integral of $f(x)$ over $[-1, 1]$ is equal to the integral of the lower-order polynomial $r(x)$ over $[-1, 1]$ due to the orthogonality of Legendre polynomials. At the roots of $q(x)p(x)$, $r(x) = f(x)$.

An entirely different approach to producing the n-point Gauss-Legendre quadrature is to solve a system of equations, knowing that polynomials up to order $2n-1$ must be integrated exactly with n points and n associated weights. This leads to the following system

Table 2.2: n-point Gaussian quadrature

#	Abscissas	Weights
1	0	2
2	± 0.5774	1
3	$0, \pm 0.7746$	0.8889, 0.5556
4	$\pm 0.3400, \pm 0.8611$	0.6521, 0.3479
5	$0, \pm 0.5385, \pm 0.9062$	0.5689, 0.4786, 0.2369

of equations.

$$\int_{-1}^{1} x^j dx = \sum_{i=1}^{n} w_i x_i^j, \ j = 0, 1, \ldots, 2n-1 \tag{2.8}$$

This gives $2n$ equations with $2n$ unknowns and can be solved exactly. However, in practice Gauss quadrature weights are either pre-tabulated or calculated using quicker methods such as those in [31].

From Table 2.2, it is clear that Gauss quadrature does not support nesting. While Gauss quadrature can be used as a basis for an adaptive quadrature, it does not make for an efficient composite rule because the endpoints are not sampled. However, Kronrod [32] developed an extension to Gauss quadrature which expands an n-point Gauss quadrature into a $2n-1$ point quadrature which can still integrate a $3n+1$ order polynomial exactly. While Kronrod's extension is useful for calculating an error estimate, it wasn't until Patterson [33] created a recursive Gauss quadrature extension that it became suitable for adaptive quadrature. However, even Patterson's extension poses problems, as not all Gauss quadratures have Patterson extensions and some have only a limited number [34]. A common basis for Patterson's extensions are the 3-point Gauss quadrature, which can be extended numerous times with accuracy equal to that of Kronrod's original extension.

For low-dimension integration where sample points can be unequally spaced, Gauss quadrature is perhaps the best quadrature available. However, for high dimensional problems, Gauss quadrature, like all other tensor product quadratures, suffers from the curse of dimensionality. It is not easily made adaptive and the Patterson extensions nearly doubles the number of function evaluations with each iteration.

2.3 Clenshaw-Curtis Quadrature

Clenshaw-Curtis quadrature (CCQ) is another popular quadrature often mentioned alongside Gauss quadrature. While its accuracy, in theory, is less than Gauss quadrature, it often performs nearly as well [35] and has the added benefit of being a nestable quadrature. Clenshaw-Curtis quadrature is defined over the domain $[-1, 1]$, but like all quadratures can be easily mapped to an arbitrary domain $[a, b]$. The abscissas of an n-point CCQ are located at the nodes of the Chebyshev polynomials of the first kind 2.2.

$$x_i^n = \cos \frac{i\pi}{n-1},\ i = 0, 1, ..., n-1 \tag{2.9}$$

The Chebyshev polynomials [36] are orthogonal with respect to the weight measure $w(x) = \frac{1}{\sqrt{1-x^2}}$ over the domain $[-1, 1]$.

$$\langle T_i, T_j \rangle = \int_{-1}^{1} T_i(x) T_j(x) w(x) dx = \delta_{ij} \tag{2.10}$$

They have the property that $T_i(x) = \cos ix$. They can be easily derived through the recurrence relation

$$T_0(x) = 1,\ T_1(x) = x,\ T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x) \tag{2.11}$$

CCQ assumes that the integrand $f(x),\ -1 < x < 1$ can be approximated as

$$f(x) \approx \sum_{i=0}^{n}{}'' a_i T_i(x) \tag{2.12}$$

where $\sum''$ is a finite sum in which the first and last terms are halved. The coefficients a_i are calculated as

$$a_i = \frac{2}{n} \sum_{j=0}^{n}{}'' f\left(\cos \frac{\pi j}{n}\right) \cos \frac{\pi i j}{n} \tag{2.13}$$

The approximation of $f(x)$ is now known and the approximation can be integrated exactly [37].

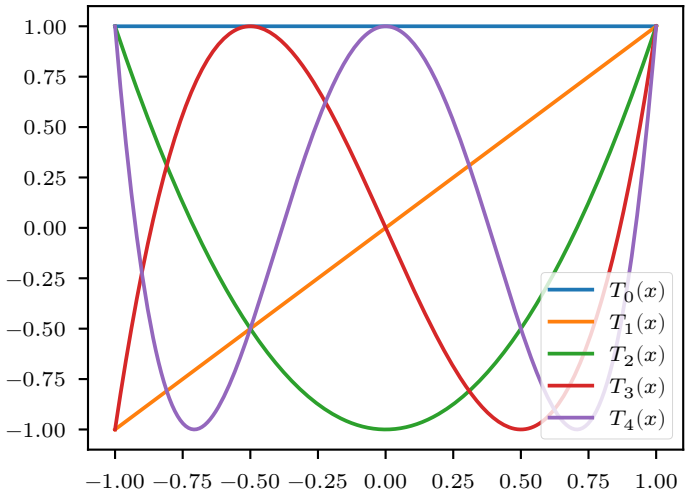


Figure 2.2: The first five Chebyshev polynomials of the first kind.

CCQ can exactly integrate a $n-1$ polynomial using only n points. CCQ has been extensively compared to Gauss quadrature and the results have shown that both achieve similar convergence rates up until some critical number of samples, after which the convergence rate of CCQ nearly halves [35].

CCQ is often used for mid-dimensional integrations due to both its comparable performance to Gauss quadrature and its ability to nest. This makes error estimation more efficient and makes CCQ suitable for sparse grids. Because CCQ samples the endpoints of the domain, it is also efficiently used in adaptive sub-dividing quadrature techniques.

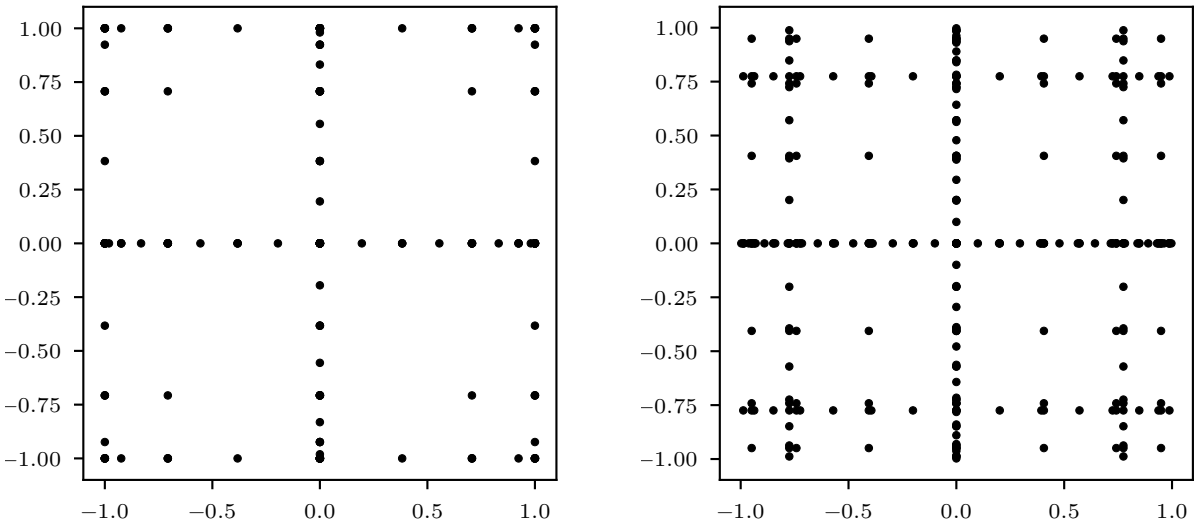
2.4 Sparse Grids

Smolyak sparse grids are a technique for generating a combination of quadrature rules in $\mathbb{R}^N$ for which the number of sampling points does not scale exponentially with N, thus alleviating the curse of dimensionality. Smolyak sparse grids have been employed extensively since Smolyak's 1963 paper, particularly for high-dimensional numerical integration [34].

Formally, Smolyak sparse grids can be expressed as

$$Q_l^d f := \sum_{l \le |k|_1 \le l+d-1} (-1)^{l+d-|k|_1-1} \binom{d-1}{|k|_1 - l} (Q_{k_1}^1 \otimes \cdots \otimes Q_{k_d}^1) f \tag{2.14}$$

where Q_l^d is a level l quadrature over the d-dimensional hypercube. A more intuitive sense of sparse grids can be obtained graphically. Figure 2.3 shows both nested and non-nested sparse grids.



(a) Clenshaw-Curtis

(b) Gauss-Legendre

Figure 2.3: Comparison of $l = 5$ sparse grids with a nested quadrature basis (Clenshaw-Curtis) and a non-nested quadrature basis (Gauss-Legendre).

Smolyak sparse grids can be built using many quadratures, and can even use different quadrature rules for different dimensions of the same sparse grid. However, the number of sampling points scales most efficiently when a single, nested quadrature rule is used to construct the entire grid. Common rules used for Smolyak sparse grid construction are trapezoidal rule, CCQ, and Gauss quadrature with Patterson extensions.
Smolyak sparse grids can also be used adaptively by iteratively dividing the domain in regions of interest. See [38], [39] for derivation and application. This makes Smolyak sparse

grids suitable for mid-dimensional problems.

While Smolyak sparse grids successfully extend the applicability of quadratures to higher dimensions for adequately smooth integrands, they do not completely lift the curse of dimensionality. For high-dimensional integrals the number of sampling points required by an accurate sparse grid may still by computationally infeasible.

2.5 Monte Carlo and Quasi-Monte Carlo Methods

MC methods are commonly used for integrations with the highest dimensions. They are easy to implement in an arbitrary number of dimensions, can provide an integration with any number of function evaluations over the domain, and have a straightforward statistical error estimation. However, the convergence rate is slow. Since their development at Los Alamos National Laboratory in the Manhattan Project era [40], Monte Carlo methods have been used extensively in engineering, physical sciences, computer graphics, finance, and artificial intelligence.

MC methods use an equally-weighted set of sampling points to approximate an integral, giving the form

$$Q(f) = \frac{1}{n} \sum_{i=1}^{n} f(x) \tag{2.15}$$

The abscissas may be randomly sampled from a simple uniform distribution, or from any other distribution. Because of this random sampling, the error estimate associated with MC integrations comes from a statistical measure.

$$E(f) = \frac{\sigma(f)}{\sqrt{n}},\ \sigma^2(f) \coloneqq I(f^2) - (I(f))^2 \tag{2.16}$$

Here $\sigma^2(f)$ is the variance of f and can be estimated using the same function evaluations used to approximate the integral. The convergence rate of MC methods is also statistical: $O(1/\sqrt{n})$. This slow convergence rate is what spurred the development of QMC methods in the 1950s.

MC methods can be used adaptively with algorithms for subdividing the domain, a technique called recursive stratified sampling. MC methods can also employ importance sampling, which assumes a priori knowledge of the integrand in an attempt to randomly sample where the contribution of the integrand is highest. Alternatively, MC methods can use adaptive sampling, which serve the same purpose but without assuming anything about the integrand. Adaptive sampling algorithms infer information about the integrand based upon the sampled values.

QMC methods take things a step further and forego the requirement that samples be random. QMC methods instead sample from low-discrepancy sequences which are designed to be better than random. For adequately smooth integrands, QMC methods can achieve much better convergence than traditional MC methods [23].

MC and QMC methods are good for high-dimensional integrals which are simply intractable using quadrature-based methods, even with the use of sparse grids. These methods are robust, easy to implement, and trivially parallelizable. The trade-off is that MC and QMC methods both suffer from relatively slow convergence.

2.6 Bayesian Quadrature

Bayesian quadrature is a form of model-based integration. It assumes a certain form for the integrand, typically a Gaussian process, and uses sampling points to fit the model form to the integrand, effectively creating a surrogate model of the integrand. The integral of the surrogate model can be computed much more quickly, often analytically, and approximates the integral of the true integrand. While this adds computational expense, it allows for lots of flexibility. Any method of sampling can be used. Many surrogate models can be used. Additionally, it provides a more detailed error estimate than any of the aforementioned numerical integration techniques. Using a Gaussian process as the surrogate model, one can calculate a distribution of integral values. Because any sampling method may be used, the curse of dimensionality may be lifted for sufficiently smooth integrands.

Adaptive sampling is a critically useful extension of Bayesian quadrature. Because Bayesian quadrature can accommodate any arbitrary set of sampling points, it is trivial to employ an informed sampling algorithm. Many have been suggested and used for surrogate modeling [17], [41]. These methods all calculate a metric which can be used to intelligently choose sampling points. These scoring metrics may take into account spacing, gradients, or variability. Several popular metrics have emerged. See [42] for a comparison of many.

The flexibility of Bayesian quadrature carries with it several difficulties. The model form assumed must be fit to the integrand based on the samples. For a Gaussian process, this process depends not only on the function samples, but also several hyperparameters. Tuning these hyperparameters is critical for good accuracy and additional algorithms must be used to handle this process [27]. Additionally, any model form will have limitations. While a priori knowledge of the integrand can be used as an advantage, lack of information or incorrect assumptions about the integrand may lead to a poor surrogate model approximation and thus a poor integration result. Convergence metrics of the surrogate model can be used to monitor this. However, it's important to note that this difficulty isn't necessarily unique to Bayesian quadrature. Lack of information, incorrect assumptions, and a difficult integrand can lead to poor results using any numerical integration method.

Because of its flexibility, Bayesian quadrature can be successfully applied to integrations from low to high number of dimensions. For low-dimensional problems, it may be preferable due to its more informative error estimation. Similarly, its probabilistic estimation of the integral may be useful for uncertainty propagation. At higher dimensions, Bayesian quadrature can alleviate the curse of dimensionality. Regardless of dimension, Bayesian quadrature can incorporate information about the integrand through the choice of surrogate model and hyperparameters. It can also be combined with any form of adaptive sampling. Difficulties arise when the surrogate model introduces model form error. The computational expense of fitting the surrogate model may make Bayesian quadrature relatively inefficient if the detailed distributions on the integration value and error estimation

are not of value.

CHAPTER 3

METHODS

> When good results are obtained in integrating a high-dimensional function, we should conclude first of all that an especially tractable integrand was tried and not that a generally successful method has been found. A secondary conclusion is that we might have made a very good choice in selecting an integration method to exploit whatever features of f made it tractable. - Art B. Owen, PhD., 1998

3.1 Optimization of Cubature

3.1.1 Algorithm

The OOC method attempts to optimize an extension for any cubature. The objective function for the optimization algorithm quantifies the performance of the extended cubature on a set of functions with known integrals, termed "training functions". The objective metric for a cubature is calculated as the weighted sum of the integration error when integrating the training functions using that cubature. The result of the optimization algorithm is a new cubature which may build upon an existing cubature and is determined to be optimal for a certain set of functions. The simple flow control of this algorithm is presented in Algorithm 1. A mathematical definition of the algorithm follows.

Consider a quadrature with p points in the d-dimensional hypercube domain Ω^d, defined as $\{(x_1, \ldots, x_d) \mid x_i \in [-1, 1] \text{ for all } i = 1, \ldots, d\}$. The cubature abscissas are $A = \{\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_p\}$, and the cubature weights are $W = \{w_1, w_2, \ldots, w_p\}$. All cubature weights

Algorithm 1 Optimization of cubature

```
procedure OOC(existing abscissas A, batch size m, iterations k, training functions T)
    for i = 1 → k do
        Read A
        Create initial guess for new abscissas A′
        Create initial guess for weights W
        Create objective function O(A′, W, A, T)
        Minimize O(A′, W, A, T) by optimizing W and A′
```

are real and positive. Assume the integrand $f(\vec{x})$ exists over Ω^d and has previously been evaluated at A. To improve the accuracy of the cubature, abscissas must be added and weights should be redistributed. The OOC method expands the cubature by adding m abscissas to A, forming a new set $A' = \{\vec{a}_1, ..., \vec{a}_q\}$ where $\vec{a}_{p+1}, ..., \vec{a}_q$ must be determined. Similarly, a new set of q weights must be determined. The undetermined abscissas and weights will be optimized to minimize the numerical integration error when integrating a set of n training functions, $T_j(x)$, over Ω^d. The objective function is $E = \sum_{j=1}^{n} e_j^2$, where the error terms e_j are calculated as follows.

$$e_j = \sum_{i=1}^{q} w_i T_j(\vec{x_i}) - \int_{\Omega^d} T_j(\vec{x}) dx, \;\; j = 1, ..., n \tag{3.1}$$

The training functions must have a known integral over the domain Ω^d. By default, the OOC algorithm uses a set of training functions consisting of a complete set of multivariate monomials with an order less than or equal to O. For example, given $O = 2$ and $d = 2$, the training set would be $\{1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\}$.

The undetermined abscissas are constrained to the domain Ω^d. The weights are constrained to be positive and less than the volume of the domain Ω^d. The objective function, E, must be minimized using a global optimization technique due to the presence of numerous local extrema in all but the simplest of cases. After the undetermined abscissas and weights have been optimized, the cubature extension is completely defined and should have improved accuracy compared to the previous cubature.

It should be noted that this method is not adaptive in the traditional sense. It does not depend upon the integrand evaluations. However, it allows for several methods of incorporating knowledge of the integrand to improve the numerical integration. The training functions can be any function for which a known integral over the domain exists. If the true integrand is known to have a certain form, specific training functions of a similar form, but with an easily calculated integral, could be used in place of the general monomial functions. The number of training functions can also be varied. In general, it is best to have enough training functions to avoid an underconstrained system of equations, which could pose issues for the optimization algorithm and result in negligibly-weighted sampling points. However, the exact number of training functions could be used to tailor the OOC method for a particular integrand, such as by either including or neglecting higher order training functions. Alternatively, a simple weighting function could be used to prioritize the training functions. An increase in the number of training functions will not increase the number of variables to optimize, but it will increase the nonlinearity of the cost function to be optimized, thereby increasing computational expense.

With the complexity comes several potential disadvantages. Most notably, the OOC algorithm inherently relies upon a global optimization algorithm. The OOC method can only be as reliable and accurate as the global optimization algorithm used. The optimization problem posed by the OOC algorithm is not easy. It is inherently nonlinear, with numerous local extrema. With poor parameter tuning, the optimization problem may be underconstrained or intractably nonlinear. At high dimensions and with a large number of sampling points, the optimization step of the algorithm consumes significant computational resources. The number of optimized variables equals $d * m + q$, where d is the number of dimensions, m is the number of abscissas to add, and q is the total number of abscissas. For a 5 dimensional integration, with an existing 2 point Gauss quadrature in each dimension, 38 variables must be optimized just to add a single sampling point to the quadrature! Because of this, the OOC algorithm is limited in its applicability. In practice, it is most

useful on extremely computationally expensive integrands (to justify the high cost of optimization), with a sparsely sampled domain consisting of fewer than 10 dimensions (to keep the optimization tractable). The OOC method is also useful when the integrand sampling points cannot be chosen precisely. This may occur when the sampling points have already been determined by some other method, or when the sampling points correspond to a physical test configuration which contains some measured deviation from the planned configuration.

3.1.2 Influential Parameters

Due to the relative complexity of the presented algorithm for OOC, there are numerous parameters which can affect the accuracy and efficiency of the algorithm. The most influential are presented here, along with rationales for the chosen defaults or guidelines for choosing values for specific problems.

The number of training functions used in the optimized cost function can render the addition of sampling points either useless or impractical if too low or too high. For example, assume the existing abscissas consist of a three-point Gauss quadrature in one dimension. This quadrature can exactly integrate monomials up to the 5th order. If only six monomial training functions are chosen (0 to 5th order) in an attempt to add additional sampling points, the system of equations will not be able to improve upon the existing quadrature. This will result in overlapping sampling points, negligible sampling weights, or a combination of both. See Figure 3.1 for an example. On the other hand, if 20 training functions are used for the cost function in the same scenario, the cost function may become too nonlinear for a useful quadrature to be optimized. The ideal number of monomial training functions is approximately $2q - 1$, where q is the total number of abscissas. If the OOC method is used to repeatedly add abscissas, the number of training functions should be reduced slightly.

Although properly tuning the number of training functions mitigates and often prevents close-proximity sampling points, a spacing penalty can be introduced into the cost func-

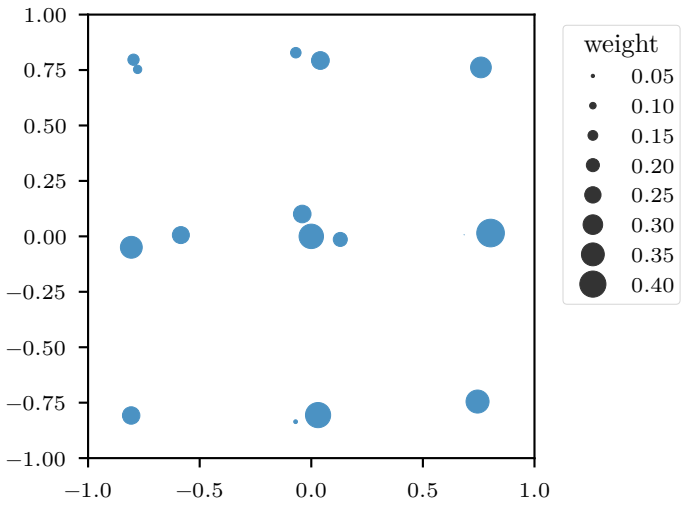


Figure 3.1: Crowded sampling points.

tion to effectively force new sampling points away from existing sampling points. This is done by adding a term to the cost function which increases exponentially as the minimum Euclidean distance between sampling points decreases. The threshold for the distance at which the spacing penalty increases sharply is a tunable parameter. This threshold distance should be decreased as the number of sampling points increases. A general rule of thumb is $\sqrt[d]{1/q}$. However, because the OOC algorithm is best suited for sparsely sampled, multivariate integrands, the penalty spacing is typically excluded and is thus not explored further.

Batching also has a significant effect on the accuracy of the OOC algorithm. In theory, adding multiple sampling points at once could improve the quality of the resulting quadrature. For example, adding two sampling points, one at a time, to a 1 point Gauss quadrature may result in an asymmetric, suboptimal 3 point quadrature, whereas adding both points at once should result in the optimal 3 point Gauss quadrature. However, adding multiple sampling points at once increases the number of variables to optimize, increasing the computational cost and difficulty of the optimization problem. This could result in decreased accuracy if the optimizer does not sufficiently converge. For example, consider Figure 3.2, in which a range of batch sizes are used to iteratively extend an initial

1-point Gauss quadrature. The optimized quadrature is then applied to a Genz Gaussian Peak integrand (described in detail in the next chapter) and the numerical integration error is plotted. The accuracy of the first quadrature extension generated by a given batch size (e.g. the first quadrature extension with a batch size of 4 would produce a 5-point quadrature), tends to increase as the batch size increases. The exception is the batch size of 10, for which the increased optimization difficulty outweighs any benefit of choosing all 10 integrand evaluations at once. However, only the batch size of 1 exhibits steady convergence. Very similar trends can be seen in Figure 3.3, in which the generated quadratures are applied to a 4th order polynomial integrand. Therefore, batching may be useful when generating a single quadrature extension, but can suffer from poor convergence when used iteratively.

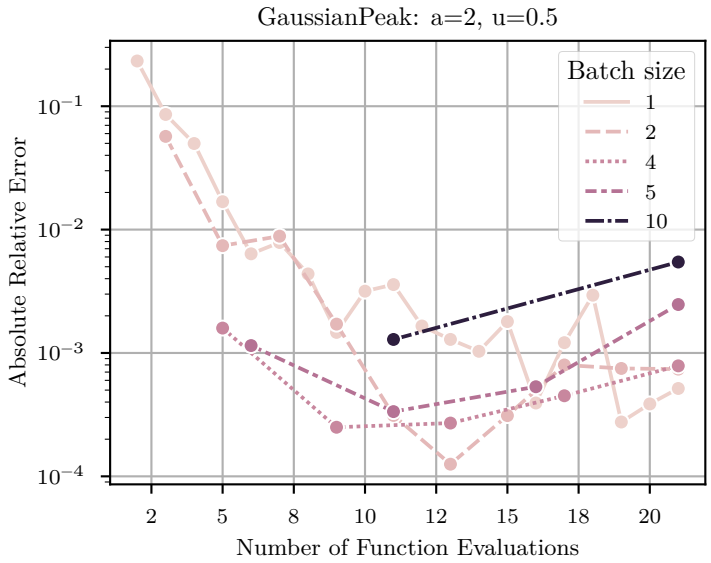


Figure 3.2: The effect of batch size on OOC numerical integration error when applied to a Genz Gaussian Peak integrand.

The choice of optimizer for the OOC algorithm also has a first order effect on the accuracy of the result. The cost function for the OOC algorithm may be extremely nonlinear and will almost certainly have multiple minima. Multiple extrema preclude the use of a local minimization algorithm. Instead, a global optimizer is used. The differential evolution al-

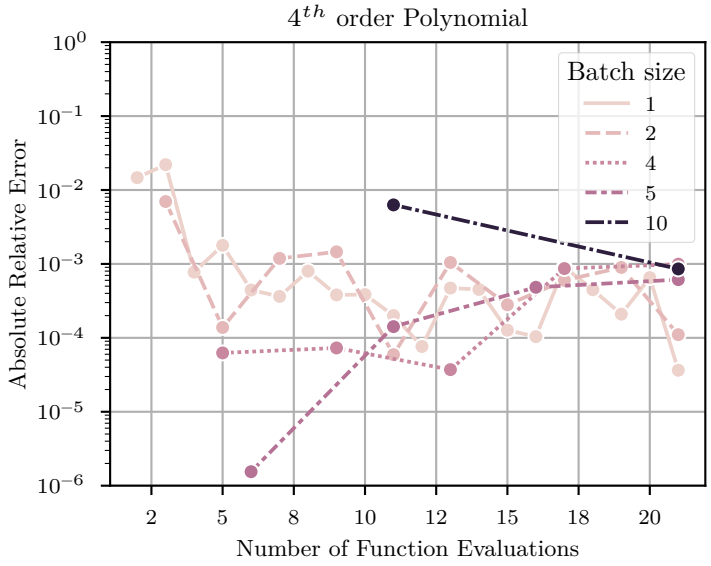


Figure 3.3: The effect of batch size on OOC numerical integration error when applied to a 4th order polynomial integrand.

gorithm [43] has shown to provide robust results compared to other algorithms. It has the added benefit of being easily parallelizable. Along with the choice of any optimization algorithm comes the choice of optimization parameters. Optimization algorithms are themselves complex algorithms with many tunable parameters. Other algorithms, such as dual annealing [44] and basin-hopping [45] may be suitable alternatives to differential evolution.

3.2 Optimization of Weights

3.2.1 Algorithm

To reduce the difficulty of the optimization required by the OOC algorithm, it is possible to choose sampling points based upon adaptive sampling alone. The weights for the new set of sampling points are then optimized just as in the OOC algorithm. This specific use of OOC, in which only weights are optimized, is termed the OOW method. Adaptive sampling is any sampling method which utilizes previously evaluated sampling points to inform the sampling process. Optimal sampling points are chosen based upon a scoring metric. Several scoring metrics are implemented for use with the OOW method.

The simplest scoring metric is "space-filling," which chooses sampling points which are furthest from existing sampling points. Latin-hypercube samples of potential sampling points are generated in the hypercube domain. The distance from each potential point to the nearest existing sampling point is calculated and recorded as that point's score. The potential point with the highest score is chosen as the next sampling point. While this method is simple, and often effective, it is not actually adaptive. It does not incorporate the previous function evaluations into the sampling process.

A similar approach is termed "range space-filling." Similarly to the space-filling approach, this method also locates the nearest neighbor for each potential point from a Latin-hypercube sample. The scoring metric in this case is the absolute difference between the function evaluation at the nearest neighbor and the predicted function evaluation at the potential point. The predicted function evaluation is calculated using a Gaussian process surrogate model, built using all previously evaluated points. The potential point with the highest scoring metric is chosen as the next sampling point. This method tends to resolve locations with high gradients, although the gradient is not a direct input to this method.

The "variance" scoring metric takes advantage of the easily computed variance associated with the Gaussian process surrogate model. Like the range space-filling method, a Gaussian process surrogate model is fit to the existing sampling points and function evaluations. Latin-hypercube samples are generated and the surrogate model variance is evaluated at each potential point. The point with the highest variance is chosen as the next sampling point.

The final scoring metric utilized here is the "weighted variance" metric. It is simply calculated by multiplying the space-filling score and variance score. The motivation for this metric is that while the variance metric most effectively reduces uncertainty in the surrogate model, it is the uncertainty in the integral of the surrogate model which directly affects the numerical integration accuracy. Similar to how a definite integral is proportional to both function value and domain size, the weighted variance metric is proportional to

both variance and nearest neighbor distance. Figure 3.4 illustrates how samples chosen with the weighted variance metric may differ from the variance metric.

When adaptive sampling is used to determine the location of additional sampling points, the optimization can be limited to only optimizing the weights of the cubature. This can ease the difficulty and computational expense of the global optimizer. It is also useful when the abscissas of a cubature are predetermined and cannot be changed or added to.

For the following studies in the next two chapters, the weighted variance sampling method will be used for both OOW and Bayesian quadrature. The other sampling metrics will not be explored further.

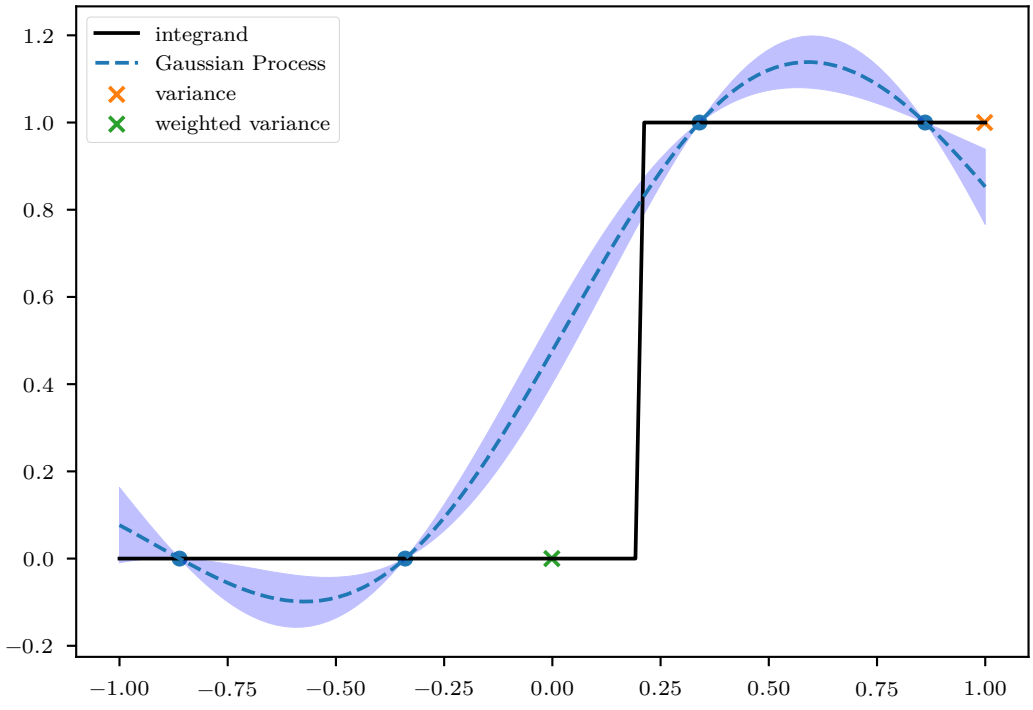


Figure 3.4: Variance and weighted variance adaptive sampling methods.

3.2.2 Influential Parameters

The most important factors when employing adaptive sampling with optimized weights are the choice of sampling metric, batching, and the number and type of training functions.

By default, the weighted variance scoring metric is used to determine sampling points. Different scoring metrics should be chosen based upon knowledge of the integrand. For example, an integrand with a step-like behavior at a point in the domain may benefit from the range space-filling approach.

Batching (adding multiple sampling points at once) is limited in the current implementation. To accommodate batching, the adaptive sampling algorithm will choose the first new sampling point, then add that point and its predicted function evaluation to the surrogate model, and repeat for any additional points. The downside of this method is that new function evaluations are not incorporated into the surrogate model during the generation of a batch of sampling points. Thus it is generally not recommended to add sampling points in a batch when using the adaptive sampling algorithm.

The same principles for choosing a set of training functions for OOC apply to the OOW method. The only difference is that the number of optimization variables is reduced for OOW. The number of training functions should be adjusted accordingly to avoid an underconstrained or overconstrained optimization problem.

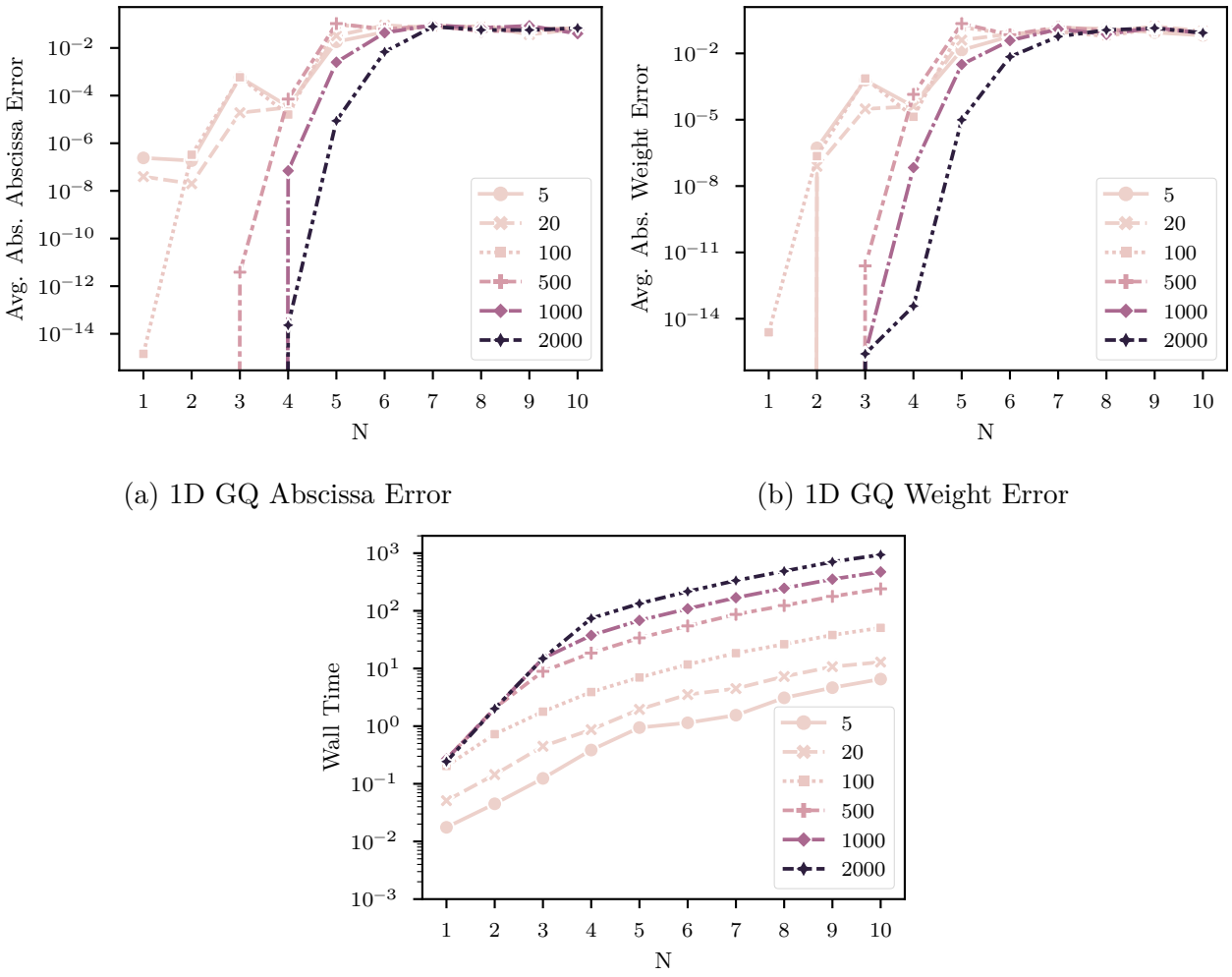
CHAPTER 4

BENCHMARKS

4.1 Method Verification

The derivation for the OOC algorithm is very similar to the approach to deriving Gauss quadrature presented in Equation 2.8. This provides an easy method for verification of the algorithm. The n-point Gauss quadrature will perfectly integrate any monomial up to $2n-1$ order. Therefore, if the OOC algorithm were to optimize a cubature for the set of training functions which consists of monomials up to order $2n-1$, the solution is the n-point Gauss quadrature. However, some error will be introduced due to difficulties in the optimization step of the algorithm. This error can be quantified by taking the average absolute difference between each optimized abscissa location and the known Gauss quadrature abscissa location. This error metric is termed the "abscissa error." Similarly, the error in the weights can be calculated by comparing the optimized weight for each point to the known Gauss quadrature weight for the point. This is termed the "weight error." The abscissa and weight error introduced when attempting to converge upon a 1-dimensional Gauss quadrature ranging from $n = 1$ to $n = 10$ is presented here. To study the effects of the number of optimization iterations on both accuracy and computational expense, the number of optimization iterations is varied from 5 to 2000. Afterwards, the convergence to Gauss quadratures in 2 and 3 dimensions is assessed.

In Figure 4.1a, the effect of the maximum allowable number of optimization iterations on the abscissa error is plotted. The optimization iterations determines how many iterations



(a) 1D GQ Abscissa Error

(b) 1D GQ Weight Error

(c) Wall clock time.

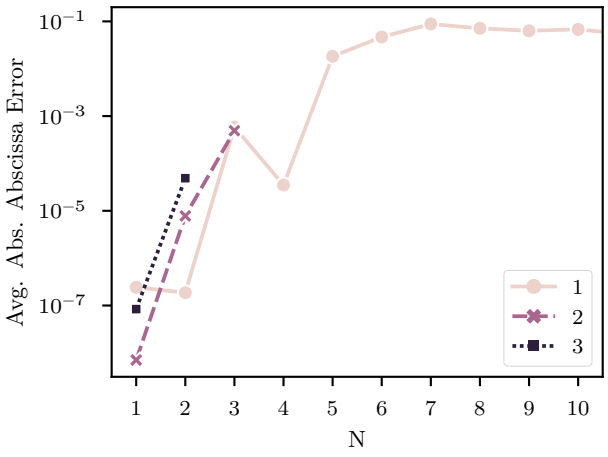
Figure 4.1: OOC recreation of 1D GQ.

of the differential evolution global optimization algorithm are computed. If the solution converges before reaching the maximum number of iterations then the actual number of iterations used may not equal the maximum number of iterations allowed. We have calculated the abscissa error for generating a one-dimensional Gauss quadrature ranging from 1 point to 10 points. For each n-point quadrature which is optimized, all n points are added simultaneously; no abscissas are added iteratively. As the number of points in the Gauss quadrature increases, the abscissa error increases exponentially. This increasing error is slower when the number of maximum optimization iterations is higher. With a maximum

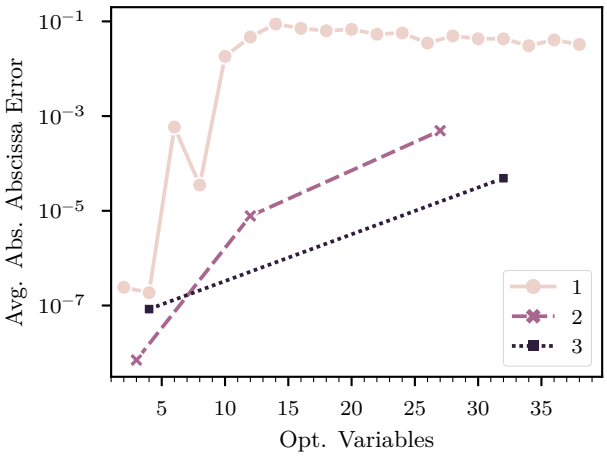
number of optimization iterations of 5, already at a one-point Gauss quadrature there is an error of approximately 10^{-7} (10^{-5}%). However, for a maximum number of iterations greater than 100, the OOC algorithm has converged upon a 1-point Gauss quadrature within machine precision. With a maximum number of optimization iterations of 2000, a perfectly converged Gauss quadrature of at least three points is computed. Approaching a 5-point Gauss quadrature, even with 2,000 optimization iterations the error approaches 10^{-3}%. For a seven-point Gauss quadrature in one dimension, the absolute error is greater than 1%, indicating that the global optimization step of the algorithm failed to converge. Similarly, for the weights, Figure 4.1b shows that while the number of optimization iterations does reduce the error, there is still significant error in the weights introduced. Complete failure to converge is apparent by at least a 7-point Gauss quadrature.

Importantly, Figure 4.1c shows that the wall clock time associated with the algorithm increases drastically with the number of optimization iterations. Therefore, one must find a balance between the allowable computational expense and the desired accuracy. Although the appropriate number of iterations is problem dependent, 2000 iterations has been successfully used for multidimensional integrations and may serve as a good general recommendation.

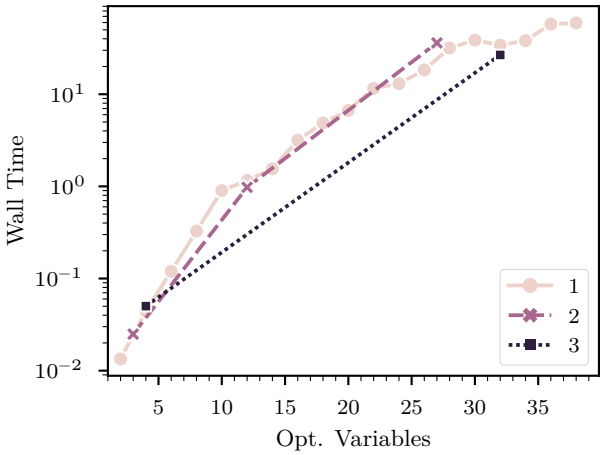
The abscissa and weight error have also been computed for multidimensional Gauss quadrature. For this study, the number of optimization iterations has been limited to 5. While this is insufficient for most optimization problems, it significantly reduced the computational expense of this study and is not likely to have an effect on the observed trends of interest. Figure 4.2 plots the same error metrics but for a 1-dimensional, 2-dimensional, and 3-dimensional Gauss quadrature. The average abscissa error appears to be a strong function of the number of points in the quadrature regardless of the number of dimensions. For example, a 2-point Gauss quadrature in three dimensions appears to be just as difficult as a 2-point Gauss quadrature in one dimension. This is despite the fact that the number of optimization variables increases linearly with the number of dimensions. In Figure 4.2b,



(a) ND GQ Abscissa Error vs N



(b) ND GQ Abscissa Error vs Opt. Variables



(c) Wall clock time.

Figure 4.2: OOC recreation of ND GQ.

we are plotting the average abscissa error against the number of optimization variables. Contrasted to Figure 4.2a, the number of optimization variables is not the first order effect on the average abscissa error. In Figure 4.2c, the wall clock time is measured in seconds versus the number of optimization variables on the x-axis. The wall clock time is indeed a strong function of the number of optimization variables. This is intuitive because the optimization is the most computationally expensive step in the algorithm and the domain space of the optimization grows exponentially with the number of optimization variables. This verification study shows that the OOC algorithm works as intended. Numerical er-

ror can be introduced into the computed cubature due to insufficient convergence during the global optimization step of the algorithm. With better optimization, either through greater computation resources or a more advanced global optimization algorithm, the OOC should improve as well. We have shown that the difficulty in convergence is a strong function of the number of abscissas in the optimized cubature. This is likely due to the increased nonlinearity of the optimization objective function (i.e. the sum of the squared integration error of all training functions) as the number of abscissas grows. Perhaps unintuitively, the number of optimization variables, although correlated to the number of abscissas, does not appear to be the primary factor affecting the optimization error. This implies that while the OOC algorithm may not be well suited for difficult 1-dimensional integrands, it may be well suited for higher-dimensional integrands.

4.2 Benchmark Integrands

To evaluate the performance of the presented methods, OOC and OOW, each method is applied to seven well-characterized integrand families. Each integrand family has a known integral over the unit hypercube domain which has been calculated analytically. All seven benchmark integrand families are highly nonlinear and contain functions of any number of dimensions, and thus can be extremely challenging for any integration method. The first six benchmark integrands are from Genz [46]. The final integrand is from Roos and Arnold [47]. These integrands have been widely used in the literature for testing integration algorithms [48]–[51]. The two-dimensional function within each integrand family has been plotted in Figure 4.3.

All of the Genz benchmark functions are defined in terms of parameters (Table 4.1). The d parameter is the number of dimensions. The a parameter affects the nonlinearity of the benchmark function. A higher value of a makes the function more nonlinear and thus more difficult to integrate. The second parameter, u, affects the location of some feature in the domain but does not significantly affect the difficulty of the integration. For example,

the peak in the Genz Gaussian Peak function is determined by the u parameter. Since all of these functions can be defined in any number of dimensions, the presented algorithms can be tested over a range of dimensionality but with consistent features in the test functions.

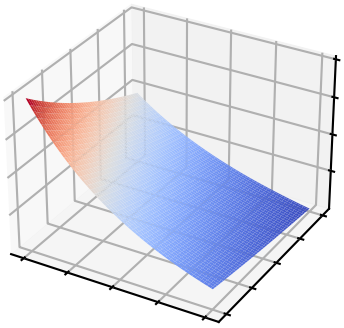
The benchmark integrands exhibit a range of characteristics: smooth (Genz Gaussian Peak, Figure 4.3b), continuous (Genz Continuous, Figure 4.3e), discontinuous (Genz Discontinuous, Figure 4.3d), and oscillatory (Genz Oscillatory, Figure 4.3c). Some have a single local extrema (Genz Corner Peak, Figure 4.3a) and others have multiple extrema (Roos Arnold, Figure 4.3g).

The presented algorithms, OOC and OOW, will be tested alongside two well-established methods. The first method is Gauss quadrature, which is perhaps the most well-known and widely used quadrature. However, Gauss quadrature is not adaptive, cannot be nested without significant losses of efficiency, and very quickly requires a high number of points for high numbers of dimensions. The second method is Bayesian quadrature. Specifically, a Gaussian process surrogate is sampled using the weighted variance metric and analytically integrated.

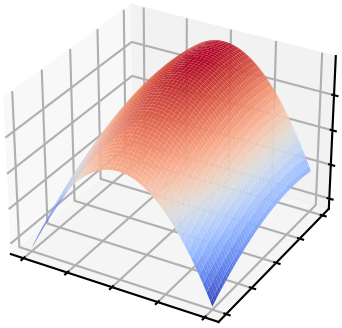
First, the four methods (OOC, OOW, Bayesian quadrature, and Gauss quadrature) will be applied to a 1-dimensional function of each of the 7 integrand families. Then the same four methods will be applied to two benchmark functions in 3 dimensions. Lastly, the OOC method will be compared to Gauss quadrature and Bayesian integration for a single 6-dimensional benchmark function.

Table 4.1: Benchmark integrand families

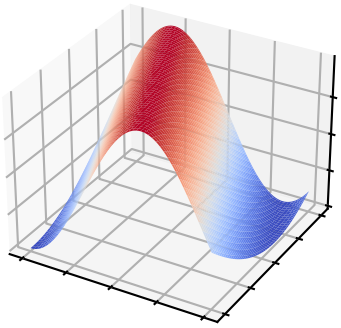
Family	Equation
Genz Corner Peak	$f(x) = \left(1 + \sum_{i=1}^{d} a_i x_i\right)^{-(d+1)}$
Genz Gaussian Peak	$f(x) = \exp\left(-\sum_{i=1}^{d} a_i^2 \left(x_i - u_i\right)^2\right)$
Genz Oscillatory	$f(x) = \cos\left(2\pi u_1 + \sum_{i=1}^{d} a_i x_i\right)$
Genz Discontinuous	$f(x) = \begin{cases} 0, & \text{if } x_1 > u_1 \text{ or } x_2 > u_2 \\ \exp\left(\sum_{i=1}^{d} a_i x_i\right), & \text{otherwise} \end{cases}$
Genz Continuous	$f(x) = \exp\left(-\sum_{i=1}^{d} a_i \left\lvert x_i - u_i\right\rvert\right)$
Genz Product Peak	$f(x) = \prod_{i=1}^{d} \frac{1}{a_i^{-2} + \left(x_i - u_i\right)^2}$
Roos Arnold	$f(x) = \prod_{i=1}^{d} \lvert 4x_i - 2\rvert$



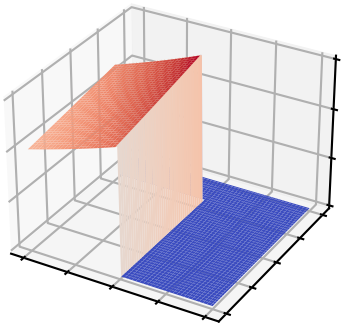
(a) Genz Corner Peak



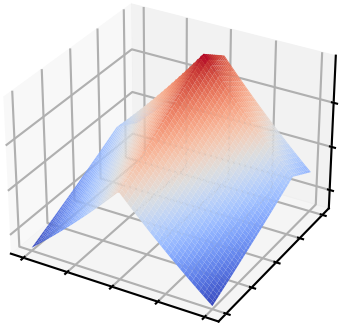
(b) Genz Gaussian Peak



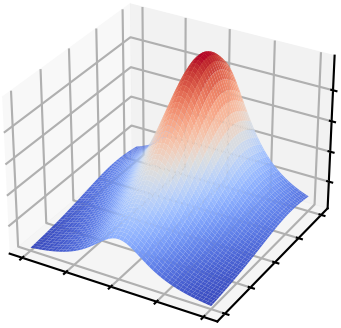
(c) Genz Oscillatory



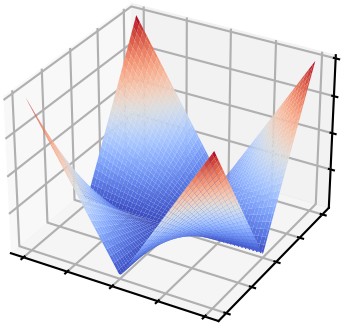
(d) Genz Discontinuous



(e) Genz Continuous



(f) Genz Product Peak



(g) Roos Arnold

Figure 4.3: Benchmark integrands.

4.3 Univariate Integration

The OOC algorithm which is benchmarked here was tuned based on lessons learned from the previous method verification study in Section 4.1. A complete set of monomials ranging in order from 0 to 30 is used as training functions. Using a high maximum training function order should prevent the optimization algorithm from choosing duplicate abscissas with negligible weights due to an underconstrained system of equations. The number of optimization iterations is capped at 200 to minimize the computational cost of this benchmarking study. Increasing the number of optimization iterations as the number of abscissas increases would improve performance but at a substantial computational expense.

Both optimization methods (OOC and OOW) begin with only a single existing abscissa, and associated function evaluation, at the center of the domain. Each optimization-based method iteratively adds a single point to the quadrature at a time (i.e. no batching). The Bayesian integration, labeled "BQ", also adds samples iteratively. Gauss quadrature, however, is not iterative. As a result, calculating a 1D integral using a 2-point, 4-point, and 6-point Gauss quadrature would require 12 total function evaluations. By contrast, applying a 2-, 4-, and 6-point quadrature using the OOC algorithm requires only 6 total function evaluations. Therefore, while Gauss quadrature exhibits the best convergence rate in Figures 4.4 - 4.10, the convergence rate with respect to the number of function evaluations is effectively much slower if multiple Gauss quadratures are used successively. For each plot, the error metric on the y-axis is the absolute value of the integration error expressed as a fraction of the true integral value (computed analytically).

For several of the 1D integrand benchmarks, the optimization-based methods were significantly outperformed by Gauss quadrature (Figures 4.4, 4.6, 4.7, and 4.8). Gaussian quadrature occasionally integrated the 1D integrands within machine precision after fewer than 16 samples, due to the analytical form of the integrands such as Genz Gaussian Peak, Genz Corner Peak, and Genz Oscillatory. Bayesian integration often performed as well as

Gauss quadrature (e.g. Figure 4.8), but occasionally exhibited convergence rates alongside the optimization-based methods (e.g. Figure 4.7).

With the Genz Oscillatory function in Figure 4.6, Gauss quadrature almost perfectly calculated the integral within 10 samples. Bayesian integration performs well but exhibits a slower rate of convergence compared to Gauss quadrature. The presented methods, optimized of cubature and OOW, have the worst rate of convergence and appear to suffer from failure to converge in the optimization step of the algorithm.

However, in Figures 4.5, 4.9, and 4.10, the optimization-based methods perform similarly, and occasionally outmatch, Gauss quadrature and Bayesian integration. The convergence of the optimization-based methods is typically not smooth. This is indicative of the stochastic nature of the optimization algorithms. Despite this, the optimization-based methods seem only slightly less reliable than Gauss quadrature for severely nonlinear integrands.

From observing the performance of the presented methods on the seven 1-dimensional benchmark integrands, it appears that the presented methods performed poorly on smooth integrands when compared to well-established Gauss quadrature and even the iterative and adaptive Bayesian quadrature. However, several of the more difficult integrands were integrated to a similar accuracy using all four methods. This OOC method is tailored for higher dimensional integrals so it is no surprise that 1-dimensional performance is not stellar.

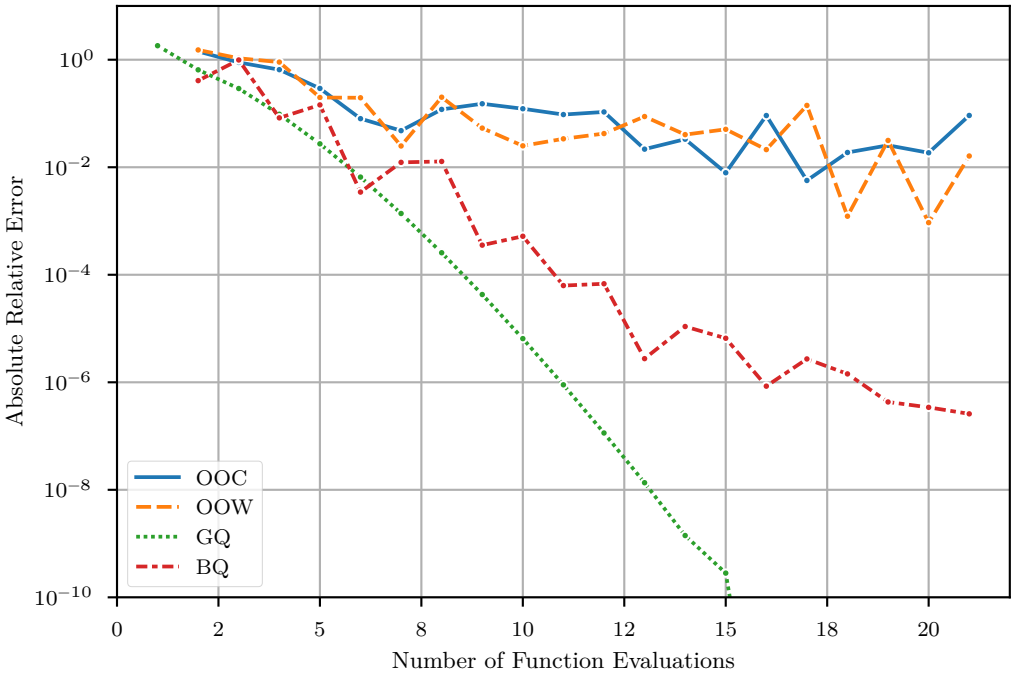


Figure 4.4: Integration error of the 1D Genz Gaussian peak using four methods.

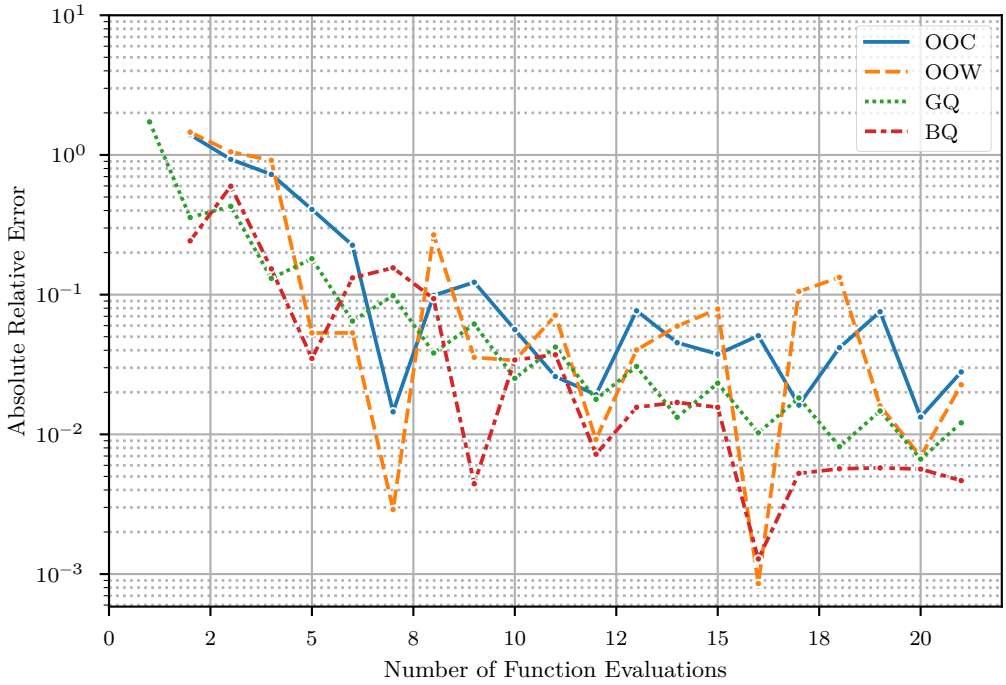


Figure 4.5: Integration error of the 1D Genz Continuous function using four methods.

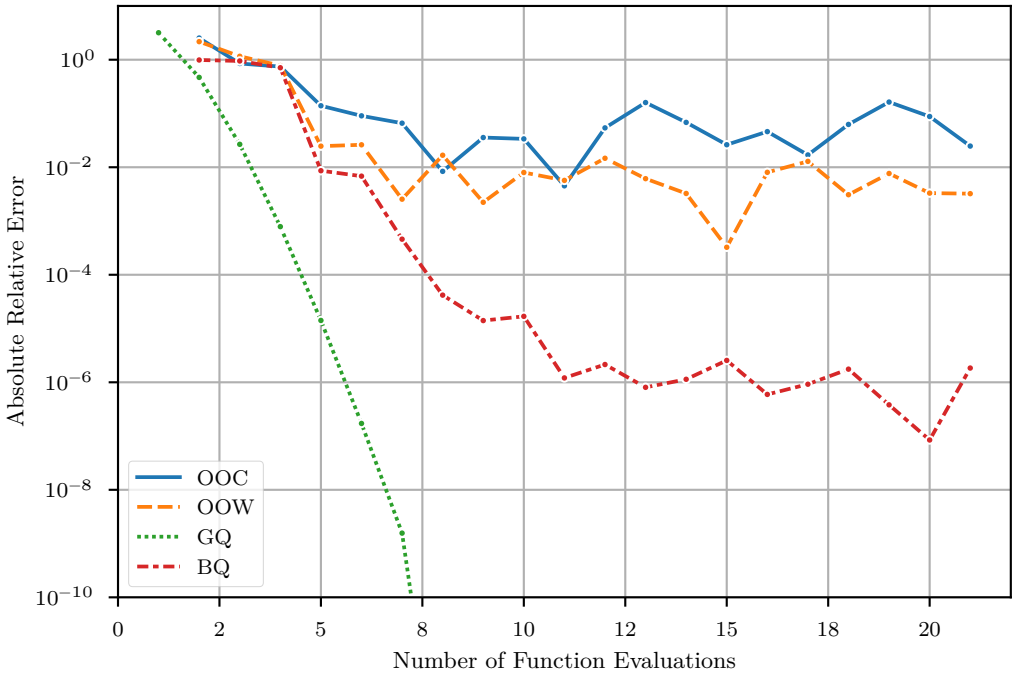


Figure 4.6: Integration error of the 1D Genz Oscillatory function using four methods.

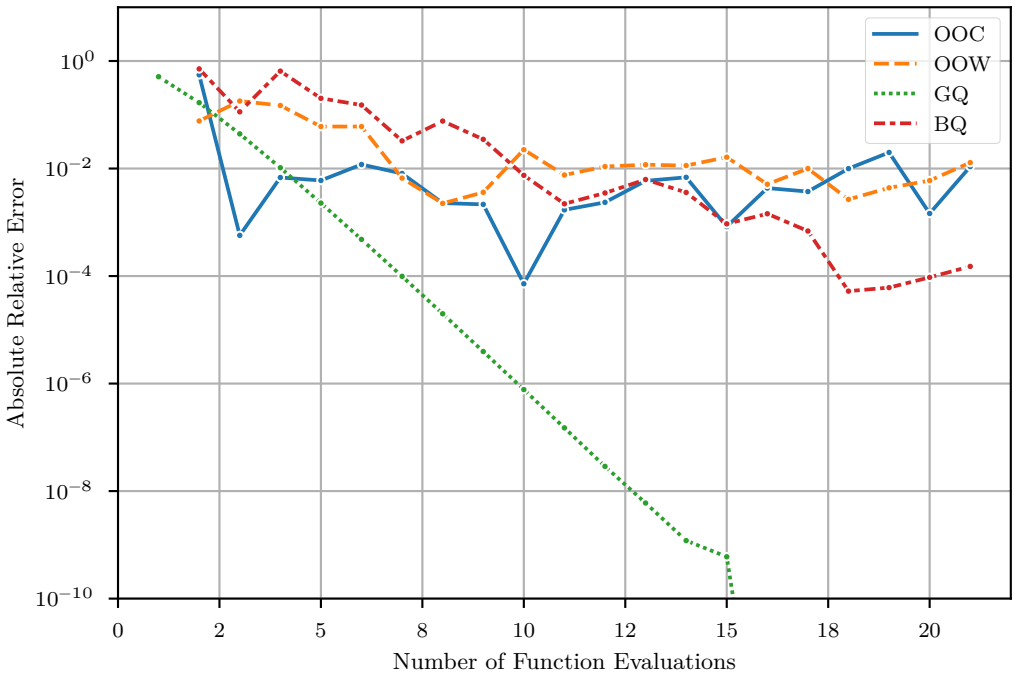


Figure 4.7: Integration error of the 1D Genz Corner Peak function using four methods.

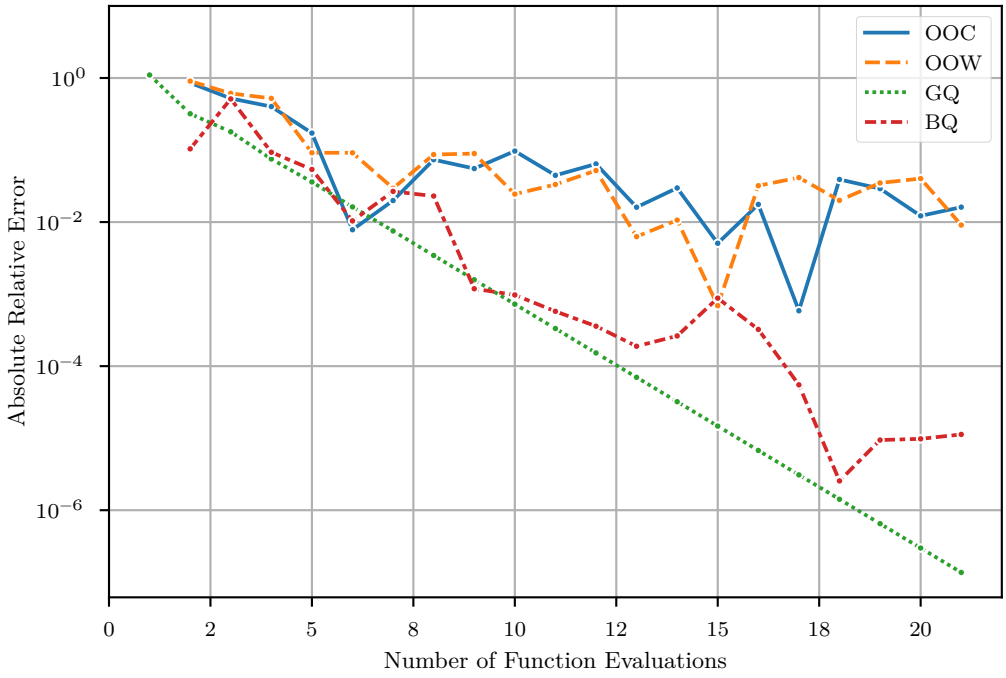


Figure 4.8: Integration error of the 1D Genz Product Peak function using four methods.

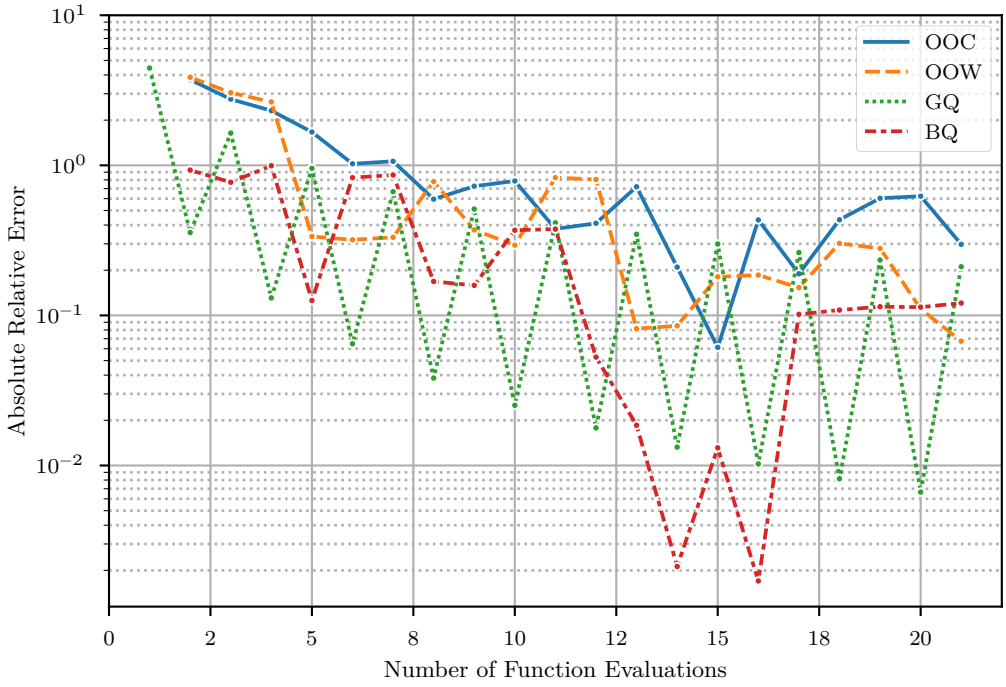


Figure 4.9: Integration error of the 1D Genz Discontinuous function using four methods.

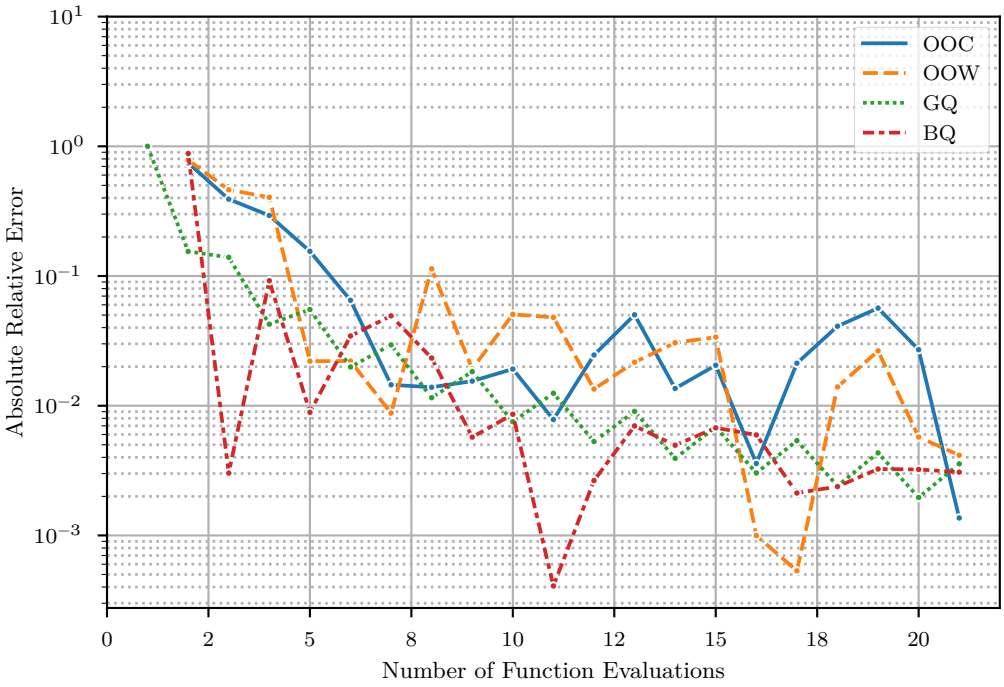


Figure 4.10: Integration error of the 1D Roos Arnold function using four methods.

4.4 Multivariate integration

This section addresses the performance of the optimization-based algorithms on 3-dimensional integrands. The same four methods are compared in each of these plots, however only two different 3-dimensional integrands are benchmarked: Genz Oscillatory and Genz Gaussian Peak. In these two benchmarks, the OOC algorithm performed poorly compared to Gauss quadrature and Bayesian integration in one dimension. However, in 3 dimensions the results are generally much better.

Figure 4.11 shows that Gauss quadrature has the fastest rate of convergence for the 3-dimensional Genz Oscillatory integrand. The other three methods appear to stop converging, or significantly slow down, after about 10 function evaluations. The OOW method appears to have consistently slow convergence from the start. The OOC method initially keeps up with Bayesian integration but suffers from lack of convergence, probably due to optimization difficulties, after about 10 function evaluations. This is a common problem seen in several of the benchmark results. After a certain number of optimization variables is achieved, the optimization algorithm can no longer converge within the maximum number of optimization iterations allowed. This can be remedied by increasing the maximum number of optimization iterations, effectively extending the number of function evaluations over which the optimization-based algorithms could maintain a certain rate of convergence.

In Figure 4.12, the four methods are assessed against the Genz Gaussian Peak integrand in 3 dimensions. The top performer is Gauss quadrature but OOC is a close second. The Bayesian integration appears to have difficulty converging until about 10 sample points, after which the accuracy is similar to that of the OOC and Gauss quadrature. The worst method by far is the OOW using the weighted variance sampling metric. While Gauss quadrature has the best performance it is not an iterative method. Therefore, to go from a 2-point Gauss quadrature in each dimension, which corresponds to 8 function evaluations on the x-axis, to a 3-point Gauss quadrature in each dimension, all 27 points of the

3-point Gauss quadrature would have to be evaluated. No points within the 2-point Gauss quadrature could be reused. On the other hand, the OOC algorithm is completely iterative and only added one point at each time. Therefore the convergence of the algorithm could be monitored and an estimated error could be calculated while the algorithm is running. This gives the advantage to the OOC algorithm because it can calculate an estimated integration error after each sample and still achieve similar accuracy to that Gauss quadrature.

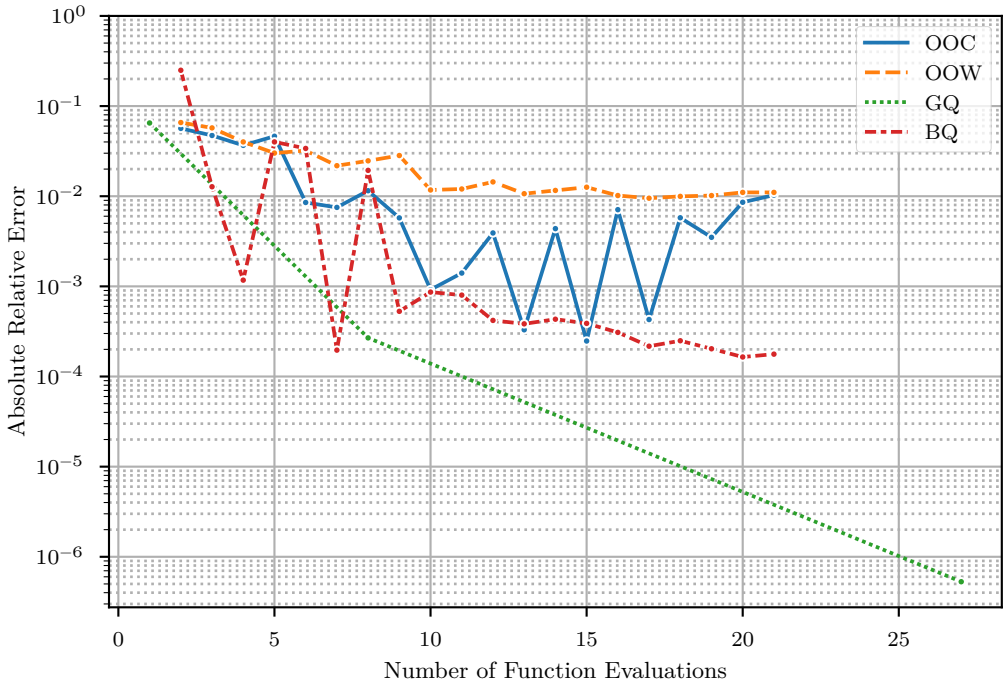


Figure 4.11: Integration of 3D Genz Oscillatory integrand.

Finally, the last of our benchmark integrands is the Genz Gaussian Peak integrand in 6 dimensions. For this benchmark only three methods are assessed, Gauss quadrature, OOC, and OOW. Bayesian integration is not evaluated for this comparison. A comparison of the 1D and 3D benchmarks suggested that the OOC method may perform best, relative to Gauss quadrature, at higher dimensions. This trend is confirmed in this 6D benchmark shown in Figure 4.13. Unlike the previous benchmarks in which only Gauss quadratures

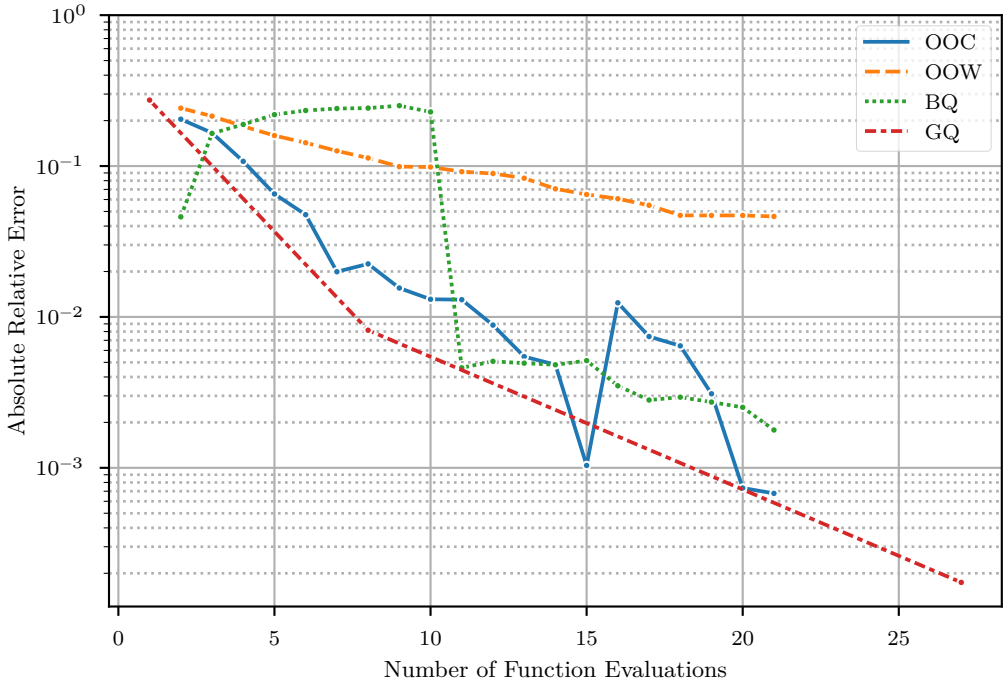


Figure 4.12: Integration of 3D Gaussian Peak integrand.

with identical number of points in each dimension were considered, this 6D benchmark plots all possible Gauss quadratures which contain fewer than 20 abscissas.

The optimized cubature outperformed all possible Gauss quadratures after approximately 10 function evaluations. Because the optimized cubature is iterative, achieving accuracy similar or better than Gauss quadrature for this benchmark integrand is a very encouraging result. In Figure 4.14, the six-dimensional samples of the optimized cubature are plotted. Each plot within the corner grid shows the location of all abscissas in two dimensions. On the diagonal is a histogram over a single dimension. The weight of each abscissa is indicated by the color of the point. The OOC algorithm prioritized sampling the 6-dimensional integrand closer to the edges of the domain than the center. This can be seen in the histograms on the diagonal. Although these six-dimensional sample points appear close together when squashed into two dimensions, the minimum distance between any two sample points is 1.0. Due to the sparsity of the domain sampling, any grouping of

sampling points was easily avoided.

While the optimized cubature is not the preferred method for low dimensional integrals, this 6-dimensional benchmark indicates that the optimized cubature can be an effective iterative integration method which can allow the user to both estimate integration error, automatically choose the number of sampling points, and still maintain accuracy similar to that of Gauss quadrature. Because the optimization step of the algorithm can become computationally expensive, the OOC method is best suited for computationally expensive integrands (such as large simulations or even physical test measurements) for which the number of evaluations is kept below 20. The OOW method, on the other hand, exhibited relatively slow convergence in both 1-dimensional and multi-dimensional integration. Therefore, the OOW method is only recommended when the cubature sampling points are predetermined. Even then, Bayesian integration should be considered.

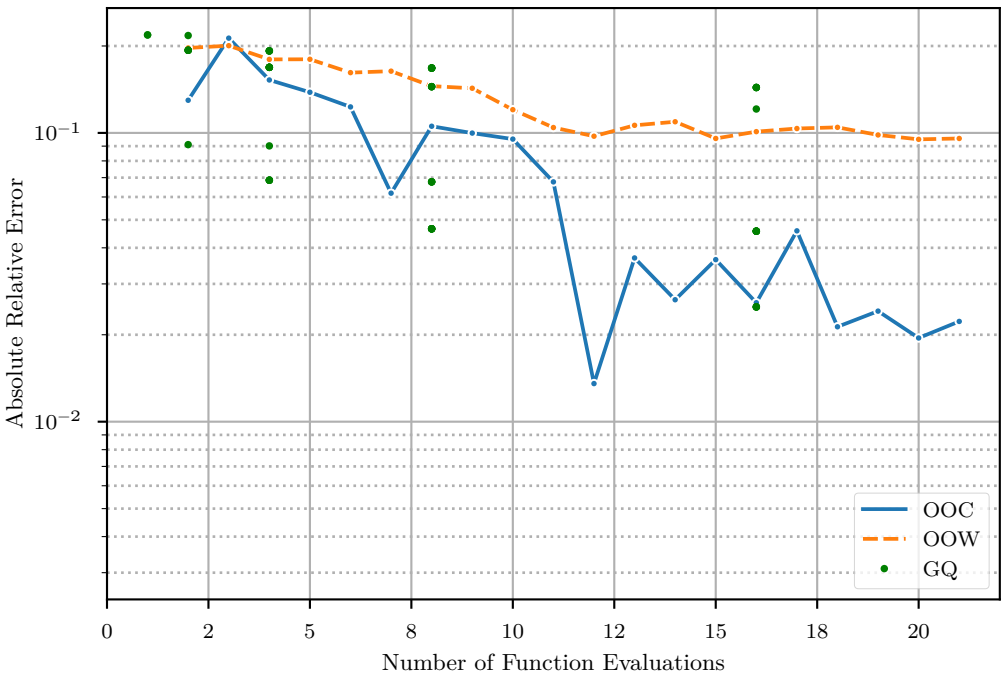


Figure 4.13: Integration of 6D Gaussian Peak integrand.

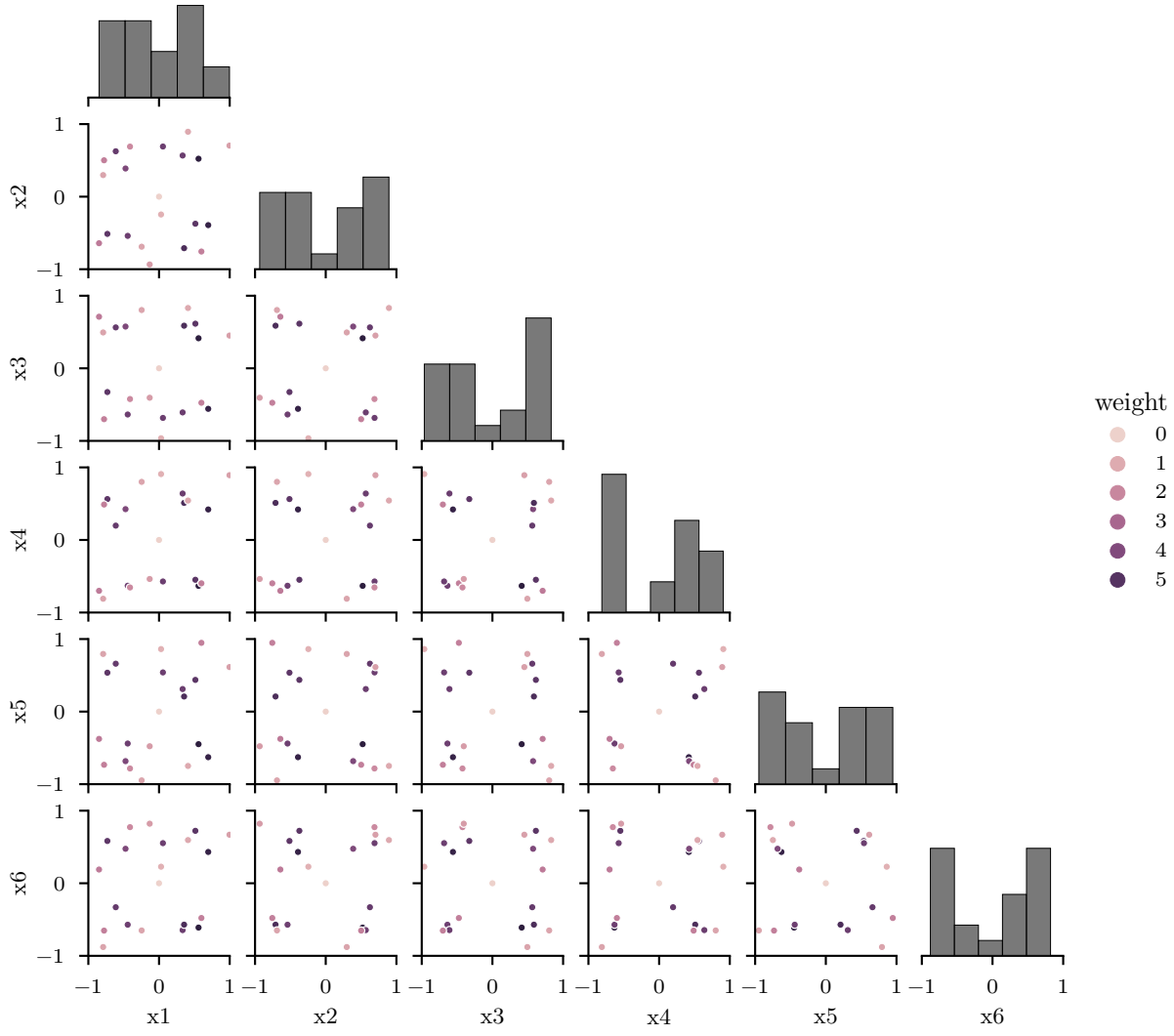


Figure 4.14: 6D Genz Gaussian Peak integrand samples.

CHAPTER 5

ENGINEERING APPLICATION

5.1 Model

The OOC algorithm has been shown to perform well on nonlinear, multivariate integrands, particularly at cubatures ranging up to 100 samples and at least 6 dimensions. These benchmarking studies aimed to simulate the challenges posed by real-world problems today, but to properly assess the algorithm's performance and demonstrate its use, we must apply it to an actual engineering problem. For this example, we turn to a foundational aerodynamics problem: supersonic flow over a wedge. Supersonic flow over a wedge is a relevant example of the highly nonlinear, computationally expensive problems in CFD and has been used to assess other algorithms relevant to uncertainty propagation, such as PCE [52].

The model of supersonic flow over a wedge, illustrated in Figure 5.1, is an inviscid, steady-state, two-dimensional, simulation of a calorically-perfect gas entering a channel at supersonic velocity and passing over a wedge of height y. An attached shock wave forms at the base of the wedge and may reflect off of the channel ceiling, before finally exiting at a zero pressure gradient outlet. The inlet conditions are defined by a normalized inlet pressure p_i, normalized inlet velocity U_i, and normalized inlet temperature T_i. The outlet pressure p_o is observed at the midpoint of the outlet. The mean and variance of p_o will be calculated by propagating uncertainty in four stochastic input variables.

The inlet conditions, p_i, U_i, and T_i, and the wedge height y are all stochastic input vari-

ables. They are assumed to have uniform distributions over the ranges defined in Table 5.1. The nominal solution is shown in Figure 5.2. To illustrate the sensitivity of the midpoint outlet pressure p_o to the input variables, p_o was recorded as each input variable was sampled at 10 regular intervals within its distribution (all other variables remaining at their nominal values). The effect of each input variable is plotted in Figure 5.3. Clearly the most sensitive and nonlinear response is with respect to the wedge height y. The inlet velocity U_i has a less impactful nonlinear effect, whereas the inlet pressure and temperature have a minimal, linear effect. Due to the nonlinearity of the oblique shock relations, nonlinear interactions between wedge height and inlet velocity are expected. Figures 5.2 and 5.3d indicate that the shock discontinuity is not fully resolved. Further mesh refinement or use of a higher-order solver scheme would result in a better-resolved discontinuity, thus increasing the nonlinearity and difficulty of the uncertainty propagation.

Table 5.1: Stochastic input variables for supersonic flow over a wedge.

Variable	Distribution
p_i	Uniform[0.95, 1.05]
U_i	Uniform[0.95, 1.05]
T_i	Uniform[4.75, 5.25]
y	Uniform[0.115, 0.125]

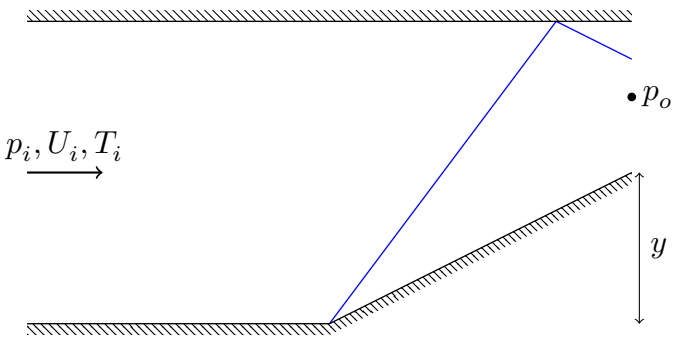


Figure 5.1: Illustration of supersonic flow over a wedge and the stochastic inputs and outputs.

The mean and variance of the midpoint outlet pressure p_o are calculated using the OOC algorithm. To do so, the mean and variance are expressed in their integral form. The mean

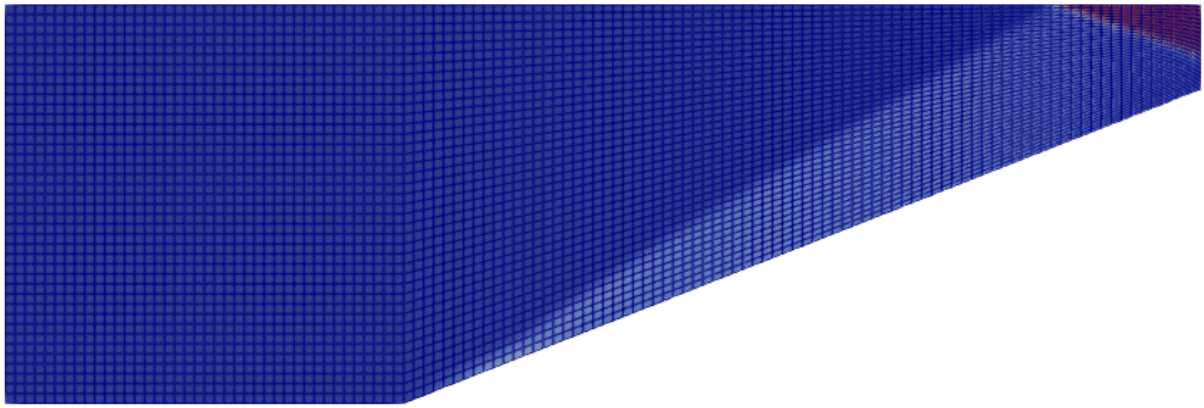
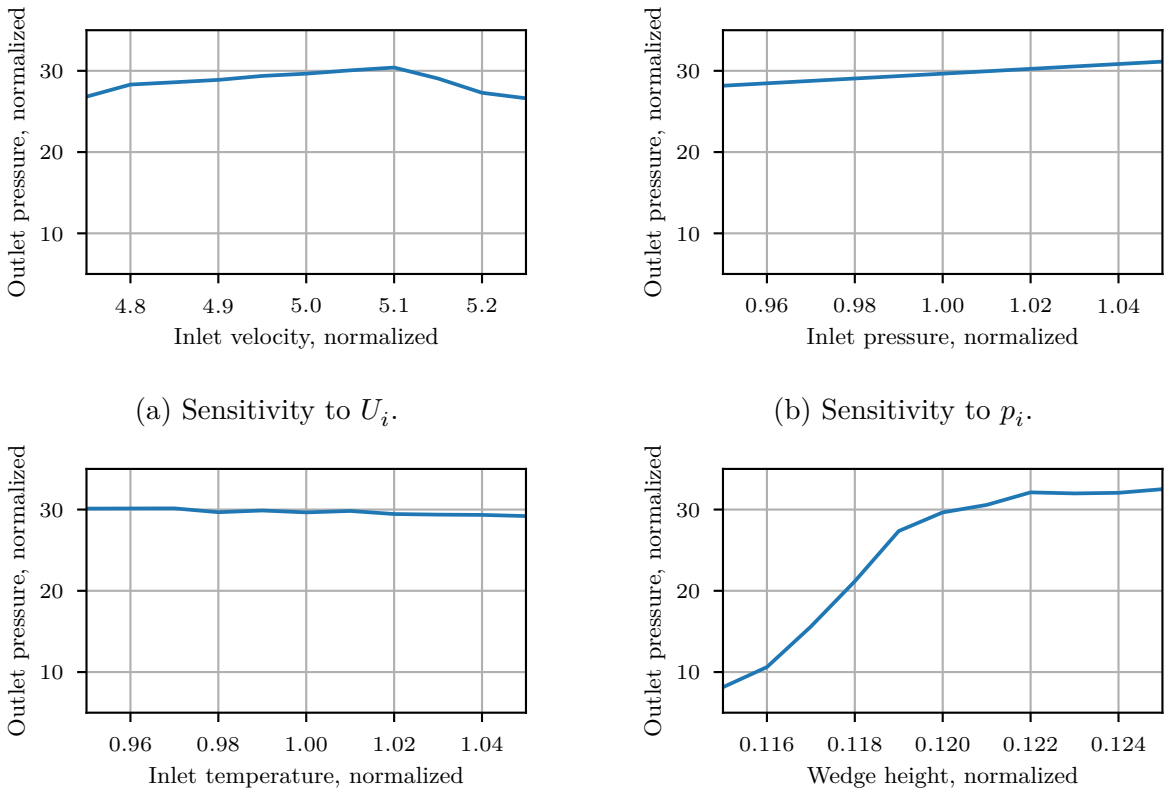


Figure 5.2: Nominal evaluation of the supersonic flow over a wedge deterministic model.

is calculated as the expectation, $\mathrm{E}[p_o] = \int_D p_o f(x) dx$, where D is the input domain. The probability density function simplifies to $f(x) = 1/D$ due to the uniform distributions of the stochastic input variables. The variance is calculated similarly as $\mathrm{Var}[p_o] = \mathrm{E}[(p_o - \mu)^2]$. The OOC algorithm was initialized with only the nominal evaluation of the CFD simulation. The algorithm optimized cubatures ranging up to 35 points, with no batching (i.e. adding one point at a time), 2000 optimization iterations per cubature, and using a complete set of monomial training functions up to 5th order. For comparison, Gauss quadrature is also used to calculate the mean.

Finally, a least-squares point-collocation PCE is generated using the function evaluations chosen by the OOC algorithm. This comparison serves two purposes. Firstly, it demonstrates the convergence rate of the OOC when applied side-by-side with PCE, a widely used approach for uncertainty propagation [53]. Second, it demonstrates that the function evaluations chosen by the OOC algorithm can be utilized for more than just integration. Here it will be demonstrated that additional methods such as PCE can be built upon results from the OOC.



(a) Sensitivity to U_i.

(b) Sensitivity to p_i.

(c) Sensitivity to T_i.

(d) Sensitivity to y.

Figure 5.3: Wedge sensitivity to the stochastic input variables.

5.2 Results

The calculated expectation of the outlet pressure for a given number of function evaluations is plotted in Figure 5.4. Optimized cubature achieves a convergence rate similar to that of Gauss quadrature. Critically, while the Gauss quadrature required knowledge of the integrand in order to determine in which dimension to allocate the most samples (which has a significant effect), the OOC performed without any knowledge of the integrand. Further still, the OOC algorithm offers finer control over the number of function evaluations. Due to its iterative nature, the convergence of the OOC could be monitored as the function is evaluated. Error estimates for Gauss quadrature could not be calculated without using multiple cubatures, significantly increasing the number of function evaluations.

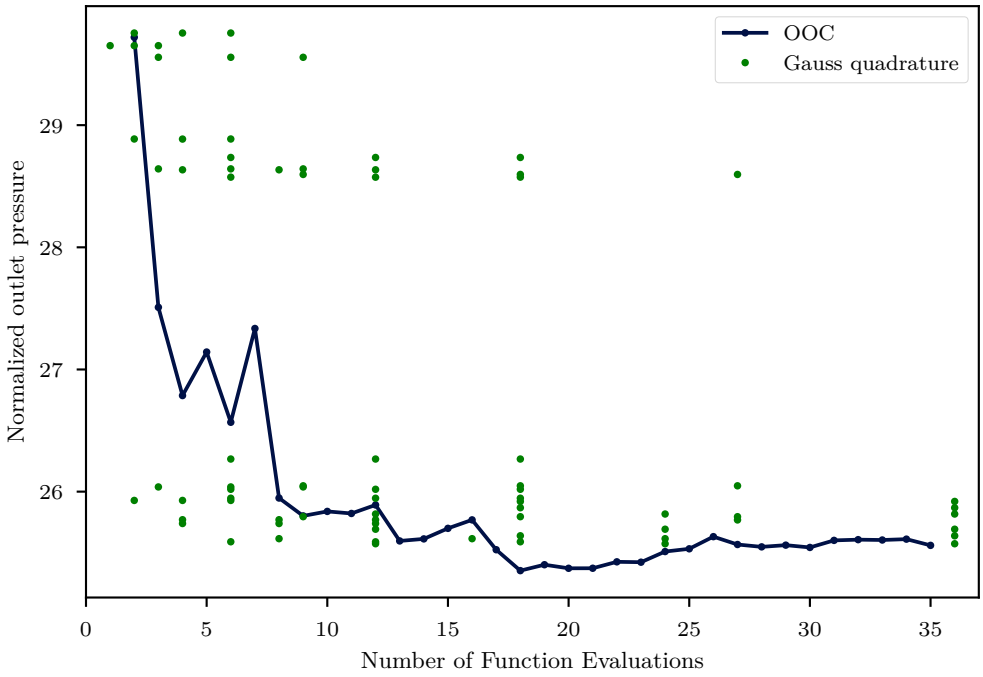


Figure 5.4: Mean of normalized outlet pressure.

All 35 samples in the final optimized cubature, normalized to the 4-dimensional hypercube domain Ω^4, are plotted in Figure 5.5. A pattern can be seen, illustrating the optimized cubature's preference for sampling the center, midpoints, and corners of the hypercube domain. Despite the appearance of clumping when plotted in just two dimensions, no clumping occurred. The average normalized distance between an abscissa and its nearest neighbor is 0.717. The minimum distance between any two points is 0.104.

In Figures 5.6 and 5.7, the optimized cubature algorithm is compared to results using a non-intrusive, point collocation PCE. Multiple PCEs of orders ranging from 0 to 3 are computed. A PCE of order n requires at minimum $\frac{(n+4)!}{4!n!}$ function evaluations for this 4-dimensional problem, and the accuracy will improve as more function evaluations are added above the minimum.

Convergence to the mean outlet pressure, plotted in Figure 5.6, is similar for the optimized cubature and 2nd order PCE. With 35 function evaluations, only a single construction of

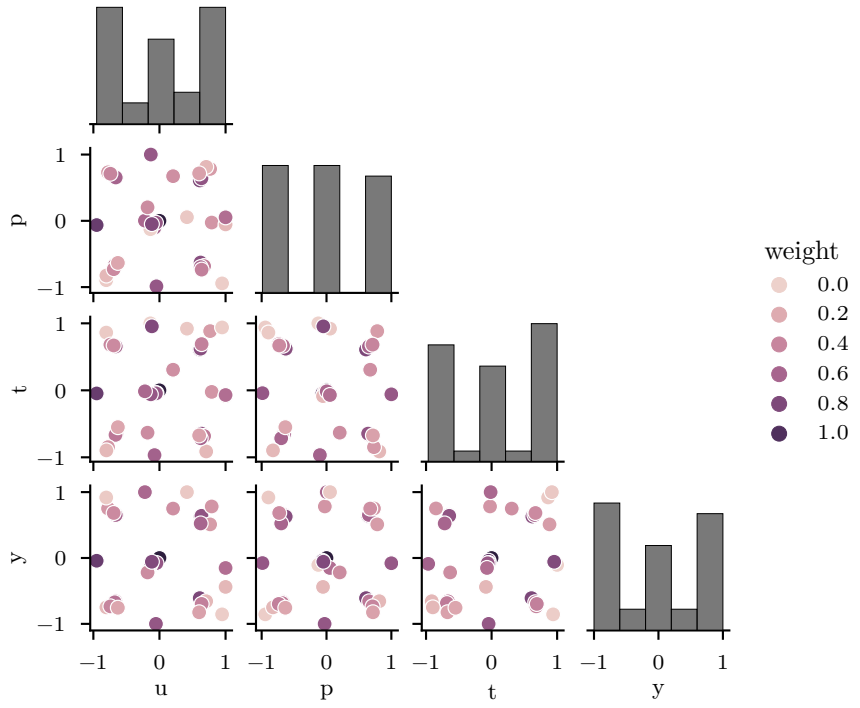


Figure 5.5: Stochastic CFD input variable evaluations, normalized onto the 4-dimensional hypercube domain Ω^4.

the 3rd order PCE is possible. Although difficult to see in the figure, it agrees well with the optimized cubature results. Both 0th order and 1st order PCE do not contain enough higher order terms to effectively capture the relation between the outlet pressure and the four stochastic inputs.

In Figure 5.7, it is clear that each PCE requires more than just the minimum number of function evaluations to converge upon the variance. For example, a 2nd order PCE requires only 15 function evaluations, yet the calculated variance is wildly inaccurate until approximately 20 evaluations. For this reason, the 3rd order PCE is not plotted, as it would require more than the available 35 evaluations to converge. The OOC appears to converge at a rate similar to, if not better than, the constructed PCEs.

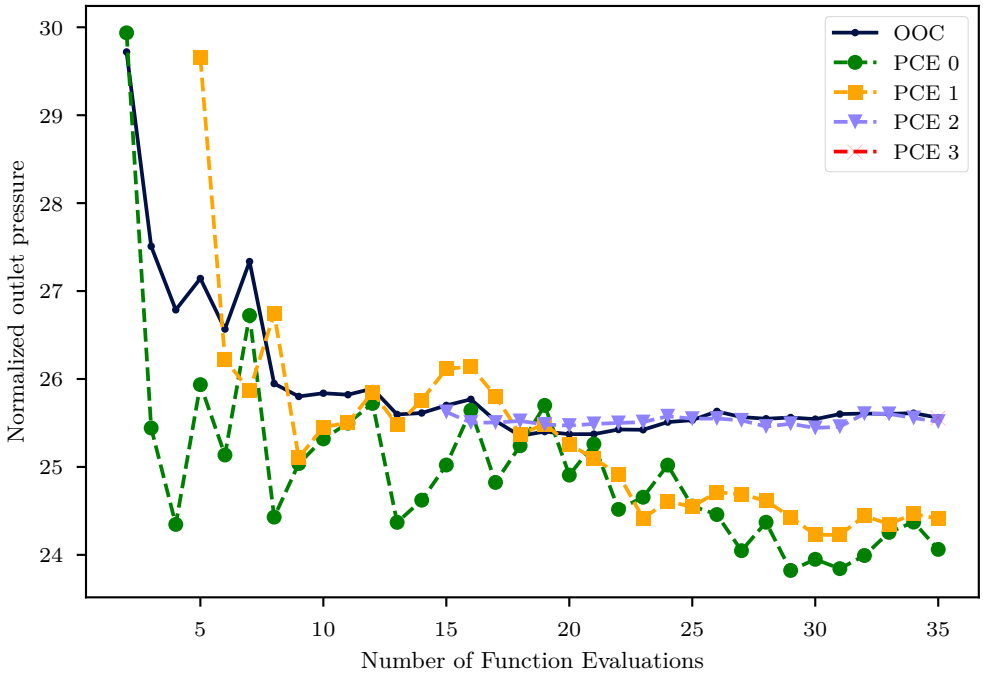


Figure 5.6: Convergence to the outlet pressure mean value.

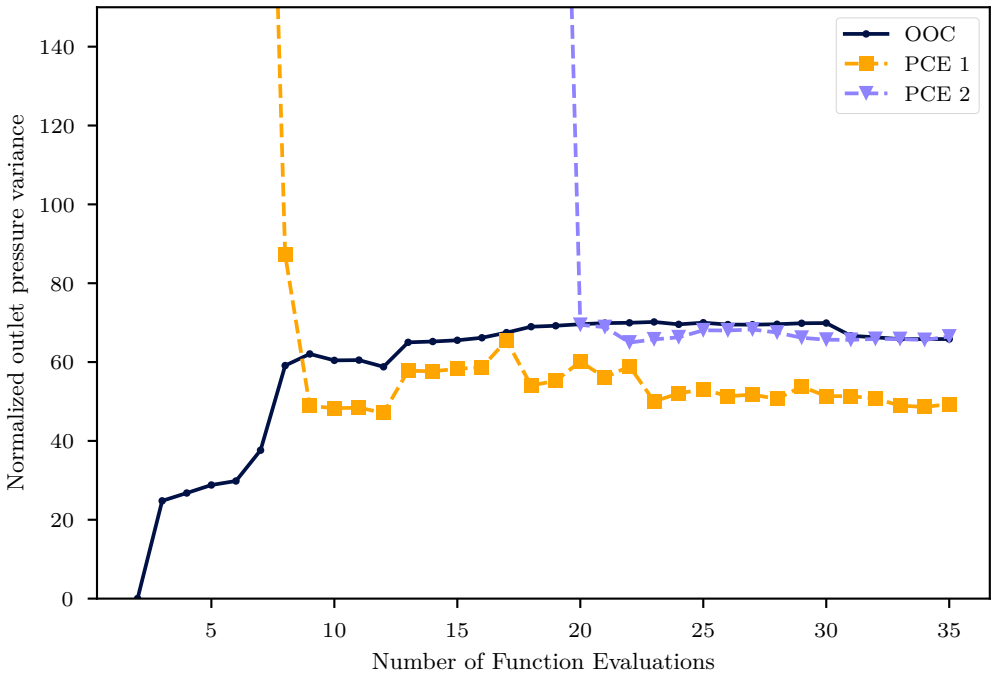


Figure 5.7: Convergence to the outlet pressure variance.

CHAPTER 6

CONCLUSIONS

A method has been presented for optimally extending existing cubatures, enabling the efficient use of all available integrand evaluations. The optimized cubature extensions can be applied iteratively and to any arbitrary existing cubature. The presented method offers granular control over the number of integrand evaluations and requires no knowledge of the integrand.

The method has been applied to numerous benchmarking studies for 1-, 3-, and 6-dimensional integrals, which demonstrated convergence rates similar to Gauss quadrature as the number of dimensions increased. Additionally, 4-dimensional uncertainty propagation through a CFD simulation was performed using the algorithm. Results compared favorably to both Gauss quadrature and PCE.

Given these results, the presented method is shown to be an accurate and efficient method for computing multivariate integrals using cubatures up to 35 points. This performance indicates that the presented method is best suited for highly computationally-expensive integrands, for which the number of evaluations is limited. Furthermore, the presented method may be useful when the integrand evaluations cannot be precisely predetermined, such as when a fixed set of integrand evaluations are available or when cubature abscissas are measured (e.g. abscissas corresponding to measured values in a physical experiment).

Future work may expand the presented method to perform efficient integrations with respect to various weighting functions corresponding to the Wiener-Askey scheme. Future work may also include an analysis of any patterns present in the optimized cubature ex-

tensions, such that a deterministic approach may be taken instead of the computationally expensive optimization-based approach.

REFERENCES

[1] R. G. Ghanem and P. D. Spanos, "Representation of stochastic processes," in *Stochastic Finite Elements: A Spectral Approach*, R. G. Ghanem and P. D. Spanos, Eds., New York, NY: Springer New York, 1991, pp. 15–65, ISBN: 978-1-4612-3094-6. DOI: 10.1007/978-1-4612-3094-6_2.

[2] M. Thapa, S. Mulani, and R. Walters, "Polynomial chaos for uncertainty quantification: Past, present, and future," in *Uncertainty Quantification: Advances in Research and Applications*, 2019, pp. 1–61, ISBN: 978-1-5361-4862-6.

[3] D. Xiu and G. Karniadakis, "The wiener–askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, Jan. 1, 2002, ISSN: 1064-8275. DOI: 10.1137/S1064827501387826.

[4] D. Xiu and G. E. Karniadakis, "Modeling uncertainty in flow simulations via generalized polynomial chaos," *Journal of Computational Physics*, vol. 187, no. 1, pp. 137–167, May 1, 2003, ISSN: 0021-9991. DOI: 10.1016/S0021-9991(03)00092-5.

[5] N. Wiener, "The homogeneous chaos," *American Journal of Mathematics*, vol. 60, no. 4, pp. 897–936, 1938, ISSN: 0002-9327. DOI: 10.2307/2371268.

[6] D. Xiu, D. Lucor, C.-H. Su, and G. E. Karniadakis, "Stochastic modeling of flow-structure interactions using generalized polynomial chaos," *Journal of Fluids Engineering*, vol. 124, no. 1, pp. 51–59, Mar. 1, 2002, ISSN: 0098-2202. DOI: 10.1115/1.1436089.

[7] O. P. Le Maitre, O. M. Knio, H. N. Najm, and R. G. Ghanem, "A stochastic projection method for fluid flow: I. basic formulation," *Journal of Computational Physics*, vol. 173, no. 2, pp. 481–511, Nov. 1, 2001, ISSN: 0021-9991. DOI: 10.1006/jcph.2001.6889.

[8] O. P. Le Maitre, M. T. Reagan, H. N. Najm, R. G. Ghanem, and O. M. Knio, "A stochastic projection method for fluid flow: II. random process," *Journal of Computational Physics*, vol. 181, no. 1, pp. 9–44, Sep. 1, 2002, ISSN: 0021-9991. DOI: 10.1006/jcph.2002.7104.

[9] O. Le Maitre, M. Reagan, B. Debusschere, H. Najm, R. Ghanem, and O. Knio, "Natural convection in a closed cavity under stochastic non-boussinesq conditions," *SIAM Journal on Scientific Computing*, vol. 26, no. 2, pp. 375–394, Jan. 1, 2004, ISSN: 1064-8275. DOI: 10.1137/S1064827503422853.

[10] L. Mathelin, M. Y. Hussaini, and T. A. Zang, "Stochastic approaches to uncertainty quantification in CFD simulations," *Numerical Algorithms*, vol. 38, no. 1, pp. 209–236, Mar. 1, 2005, ISSN: 1572-9265. DOI: 10.1007/BF02810624.

[11] L. Mathelin, M. Y. Hussaini, T. A. Zang, and F. Bataille, "Uncertainty propagation for a turbulent, compressible nozzle flow using stochastic methods," *AIAA Journal*, vol. 42, no. 8, pp. 1669–1676, Aug. 1, 2004, ISSN: 0001-1452. DOI: 10.2514/1.5674.

[12] O. M. Knio and O. P. Le Maître, "Uncertainty propagation in CFD using polynomial chaos decomposition," *Fluid Dynamics Research*, Recent Topics in Computational Fluid Dynamics, vol. 38, no. 9, pp. 616–640, Sep. 1, 2006, ISSN: 0169-5983. DOI: 10.1016/j.fluiddyn.2005.12.003.

[13] H. N. Najm, "Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics," *Annual Review of Fluid Mechanics*, vol. 41, no. 1, pp. 35–52, Dec. 19, 2008, ISSN: 0066-4189. DOI: 10.1146/annurev.fluid.010908.165248.

[14] W. Bolstad, "Bayesian inference," in *Understanding Computational Bayesian Statistics*, John Wiley & Sons, Ltd, 2012, pp. 47–60, ISBN: 978-0-470-56737-1. DOI: 10.1002/9780470567371.ch3.

[15] E. National Academies of Sciences and Medicine, Division on Engineering and Physical Sciences, Laboratory Assessments Board, and Army Research Laboratory Technical Assessment Board, *2015-2016 Assessment of the Army Research Laboratory*. Washington, D.C., United States: National Academies Press, 2017, ISBN: 978-0-309-45437-7.

[16] J. J. Ramsey, "Survey of existing uncertainty quantification capabilities for army relevant problems," US Army Research Laboratory Aberdeen Proving Ground United States, ARL-TR-8218, Nov. 27, 2017.

[17] H. Wang and J. Li, "Adaptive gaussian process approximation for bayesian inference with expensive likelihood functions," *Neural Computation*, vol. 30, no. 11, pp. 3072–3094, Sep. 14, 2018, ISSN: 0899-7667. DOI: 10.1162/neco_a_01127.

[18] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, Jun. 2008, ISSN: 1532-4435.

[19] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–423, 1989, ISSN: 0883-4237.

[20] A. Gorodetsky and Y. Marzouk, "Mercer kernels and integrated variance experimental design: Connections between gaussian process regression and polynomial approx-

imation," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 4, no. 1, pp. 796–828, Jan. 1, 2016. DOI: 10.1137/15M1017119.

[21] D. J. C. MacKay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, pp. 590–604, Jul. 1992, ISSN: 0899-7667.

[22] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970, ISSN: 0006-3444. DOI: 10.2307/2334940.

[23] J. Dick, F. Y. Kuo, and I. H. Sloan, "High-dimensional integration: The quasi-monte carlo way*†," *Acta Numerica*, vol. 22, pp. 133–288, May 2013, ISSN: 0962-4929, 1474-0508. DOI: 10.1017/S0962492913000044.

[24] A. O'Hagan, "Bayes–hermite quadrature," *Journal of Statistical Planning and Inference*, vol. 29, no. 3, pp. 245–260, Nov. 1, 1991, ISSN: 0378-3758. DOI: 10.1016/0378-3758(91)90002-V.

[25] T. Gunter, M. A. Osborne, R. Garnett, P. Hennig, and S. J. Roberts, "Sampling for inference in probabilistic models with fast bayesian quadrature," *arXiv:1411.0439 [stat]*, Nov. 3, 2014. arXiv: 1411.0439.

[26] M. Kennedy, "Bayesian quadrature with non-normal approximating functions," *Statistics and Computing*, vol. 8, no. 4, pp. 365–375, Dec. 1, 1998, ISSN: 1573-1375. DOI: 10.1023/A:1008832824006.

[27] M. A. Osborne, D. Duvenaud, R. Garnett, C. E. Rasmussen, S. J. Roberts, and Z. Ghahramani, "Active learning of model evidence using bayesian quadrature," in *Advances in Neural Information Processing Systems*, vol. 1, 2012, pp. 46–54, ISBN: 978-1-62748-003-1.

[28] C. E. Rasmussen, "Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals," *Bayesian Statistics 7*, pp. 651–659, 2003.

[29] C. E. Rasmussen and Z. Ghahramani, "Bayesian monte carlo," in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, ser. NIPS'02, Cambridge, MA, USA: MIT Press, 2002, pp. 505–512.

[30] M. B. Allen III and E. L. Isaacson, "Numerical integration," in *Numerical Analysis for Applied Science*, John Wiley & Sons, Ltd, 2011, pp. 313–348, ISBN: 978-1-118-03312-8. DOI: 10.1002/9781118033128.ch6.

[31] A. Glaser, X. Liu, and V. Rokhlin, "A fast algorithm for the calculation of the roots of special functions," *SIAM Journal on Scientific Computing*, vol. 29, no. 4, pp. 1420–1438, Jan. 1, 2007, ISSN: 1064-8275. DOI: 10.1137/06067016X.

[32] A. S. Kronrod, *Nodes and weights of quadrature formulas, sixteen-place tables: Authorized translation from the Russian.* New York: Consultants Bureau, 1965, OCLC: 221908705.

[33] T. N. L. Patterson, "The optimum addition of points to quadrature formulae," *Mathematics of Computation*, vol. 22, no. 104, 847–s31, 1968, ISSN: 0025-5718. DOI: 10.2307/2004583.

[34] T. Gerstner and M. Griebel, "Numerical integration using sparse grids," *Numerical Algorithms*, vol. 18, no. 3, p. 209, Jan. 1, 1998, ISSN: 1572-9265. DOI: 10.1023/A:1019129717644.

[35] J. A. C. Weideman and L. N. Trefethen, "The kink phenomenon in fejér and clenshaw–curtis quadrature," *Numerische Mathematik*, vol. 107, no. 4, pp. 707–727, Oct. 1, 2007, ISSN: 0945-3245. DOI: 10.1007/s00211-007-0101-2.

[36] E. L. Ortiz and T. J. Rivlin, "Another look at the chebyshev polynomials," *The American Mathematical Monthly*, vol. 90, no. 1, pp. 3–10, 1983, ISSN: 0002-9890. DOI: 10.2307/2975684.

[37] C. W. Clenshaw and A. R. Curtis, "A method for numerical integration on an automatic computer," *Numer. Math.*, vol. 2, no. 1, pp. 197–205, Dec. 1960, ISSN: 0029-599X. DOI: 10.1007/BF01386223.

[38] H.-J. Bungartz and M. Griebel, "Sparse grids," *Acta Numerica*, vol. 13, pp. 147–269, May 2004, ISSN: 1474-0508, 0962-4929. DOI: 10.1017/S0962492904000182.

[39] T. Bonk, "A new algorithm for multi-dimensional adaptive numerical quadrature," in *Adaptive Methods — Algorithms, Theory and Applications: Proceedings of the Ninth GAMM-Seminar Kiel, January 22–24, 1993*, ser. Notes on Numerical Fluid Mechanics (NNFM), W. Hackbusch and G. Wittum, Eds., Wiesbaden: Vieweg+Teubner Verlag, 1994, pp. 54–68, ISBN: 978-3-663-14246-1. DOI: 10.1007/978-3-663-14246-1_4.

[40] N. Metropolis and S. Ulam, "The monte carlo method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949, ISSN: 0162-1459. DOI: 10.2307/2280232.

[41] M. S. Ebeida, S. A. Mitchell, L. P. Swiler, V. J. Romero, and A. A. Rushdi, "POF-darts: Geometric adaptive sampling for probability of failure," *Reliability Engineering & System Safety*, vol. 155, pp. 64–77, Nov. 1, 2016, ISSN: 0951-8320. DOI: 10.1016/j.ress.2016.05.001.

[42] D. Maljovec, B. Wang, A. Kupresanin, G. Johannesson, V. Pascucci, and P.-T. Bremer, "Adaptive sampling with topological scores," *International Journal for Uncer-*

tainty Quantification, vol. 3, no. 2, 2013, ISSN: 2152-5080, 2152-5099. DOI: 10.1615/Int.J.UncertaintyQuantification.2012003955.

[43] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1, 1997, ISSN: 0925-5001. DOI: 10.1023/A:1008202821328.

[44] Y. Xiang, D. Y. Sun, W. Fan, and X. G. Gong, "Generalized simulated annealing algorithm and its application to the thomson model," *Physics Letters A*, vol. 233, no. 3, pp. 216–220, Aug. 25, 1997, ISSN: 0375-9601. DOI: 10.1016/S0375-9601(97)00474-X.

[45] D. J. Wales and J. P. K. Doye, "Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms," *The Journal of Physical Chemistry A*, vol. 101, no. 28, pp. 5111–5116, Jul. 1, 1997, Publisher: American Chemical Society, ISSN: 1089-5639. DOI: 10.1021/jp970984n.

[46] A. Genz, "Testing multidimensional integration routines," in *Proc. Of International Conference on Tools, Methods and Languages for Scientific and Engineering Computation*, event-place: Paris, France, New York, NY, USA: Elsevier North-Holland, Inc., 1984, pp. 81–94, ISBN: 978-0-444-87570-9.

[47] P. Roos and L. Arnold, "Numerische experimente zur mehrdimensionalen quadratur," Springer, 1963.

[48] E. Novak and K. Ritter, "High dimensional integration of smooth functions over cubes," *Numerische Mathematik*, vol. 75, no. 1, pp. 79–97, Nov. 1, 1996, ISSN: 0945-3245. DOI: 10.1007/s002110050231.

[49] A. Klimke and B. Wohlmuth, "Algorithm 847: Spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB," *ACM Trans. Math. Softw.*, vol. 31, no. 4, pp. 561–579, Dec. 2005, ISSN: 0098-3500. DOI: 10.1145/1114268.1114275.

[50] P. R. Conrad and Y. M. Marzouk, "Adaptive smolyak pseudospectral approximations," *SIAM Journal on Scientific Computing*, vol. 35, no. 6, A2643–A2670, Jan. 1, 2013, Publisher: Society for Industrial and Applied Mathematics, ISSN: 1064-8275. DOI: 10.1137/120890715.

[51] A. B. Owen, "The dimension distribution and quadrature test functions," *Statistica Sinica*, vol. 13, no. 1, pp. 1–17, 2003, Publisher: Institute of Statistical Science, Academia Sinica, ISSN: 1017-0405.

[52] S. Hosder, R. Walters, and R. Perez, "A non-intrusive polynomial chaos method for uncertainty propagation in CFD simulations," in *44th AIAA Aerospace Sciences*

Meeting and Exhibit, ser. Aerospace Sciences Meetings, 0 vols., American Institute of Aeronautics and Astronautics, Jan. 9, 2006. DOI: 10.2514/6.2006-891.

[53] R. Walters and L. Huyse, "Uncertainty analysis for fluid mechanics with applications," p. 50, Feb. 1, 2002.