EINDHOVEN UNIVERSITY OF TECHNOLOGY
CENTRUM WISKUNDE & INFORMATICA

Non-Intrusive, High-Dimensional Uncertainty Quantification for the Robust Simulation of Fluid Flows

Author:
Laurent van den Bos

Supervisors:
dr. ir. Jeroen Witteveen
prof. dr. ir. Barry Koren

14TH DECEMBER 2015

TU/e Technische Universiteit Eindhoven University of Technology

CWI

Abstract

Uncertainty Quantification is the field of mathematics that focuses on the propagation and influence of uncertainties on models. Mostly complex numerical models are considered with uncertain parameters or uncertain model properties.

Several methods exist to model the uncertain parameters of numerical models. Stochastic Collocation is a method that samples the random variables of the input parameters using a deterministic procedure and then interpolates or integrates the unknown quantity of interest using the samples. Because moments of the distribution of the unknown quantity are essentially integrals of the quantity, the main focus will be on calculating integrals.

Calculating an integral using samples can be done efficiently using a quadrature or cubature rule. Both sample the space of integration in a deterministic way and several algorithms to determine the samples exist, each with its own advantages and disadvantages. In the one-dimensional case a method is proposed that has all relevant advantages (positive weights, nested points and dependency on the input distribution). The principle of the introduced quadrature rule can also be applied to a multi-dimensional setting.

However, if negative weights are allowed in the multi-dimensional case a cubature rule can be generated that has a very small number of points compared to the methods described in literature. The new method uses the fact that the tensor product of several quadrature rules has many points with the same weight that can be considered as one group.

The cubature rule is applied to the Genz test functions to compare the accuracy with already known methods. If the cubature rule is applied to C_∞-functions, the number of points required to reach a certain error is a factor less than the number conventional methods need. However, if the cubature rule is applied to C_0-functions or to discontinuous functions, results are not good.

Because no stringent upper bound of the number of points of this cubature rule can be determined, it is not possible to create a general multi-level method and balance all the relevant errors. Therefore, a new method is proposed that balances the errors for each sample point. It consists of determining the cost function of a cubature rule (which is essentially the sum of all the cost of all the points) and then minimizing this cost function given an maximum error bound. The solution can be found using either the Lagrange multipliers method or Karush-Kuhn-Tucker conditions.

The new proposed method is applied to two different test cases to test the quality of the method. The first test case is the lid-driven cavity flow problem and is used to compare the new UQ method to conventional UQ methods. The second case is simulating the flow around an airplane with uncertain geometry and uncertain flow conditions.

Introduction

In physics, chemistry, or financial mathematics one is often looking for a model to predict behavior of a system, for instance a fluid flow, a chemical reaction, or a stock exchange. Such a model can be seen as a black box: you plug in certain input values and (possibly after some time) receive the desired output. Calculating the output from the input can be very easy or extremely hard. For example, simulating fluid flow using the Navier–Stokes equations is a model for which it can be hard to determine the flow given the flow properties but simulating a falling point mass with one equation of motion is very easy.

After the model has been deduced exactly as a formula or programmed as a simulation such that it can be evaluated, it can be run for several input values and checked against reference values found in literature or measured by experiments. The prescribed input can be plugged into the model. If there is no reference value one can get good insight how the original system will behave in this case without creating the system itself.

In practice however, a model is not perfect. Two errors can be considered. Firstly the model itself can be wrong. For example, modeling a falling ball as a point mass with just one equation ignores motion of the ball *around* its center of gravity. Secondly, the input cannot be physically realized exactly. An example of this is a turbulence model, where mathematical stability is not the same as physical stability and little perturbations in the input have much impact.

Assuming uncertainty in the input parameters of a model and determining the propagation of the uncertainties through the model is the main topic of this report. The focus will be on computational fluid dynamics with uncertain fluid parameters such as the viscosity, force of the fluid flow, boundary conditions, etc.

The uncertainties in the model are briefly considered and are incorporated in the uncertainty quantification model. The emphasis will however be on the propagation of input parameters, assuming that the model is correct.

Set-up

Uncertainty Quantification can be modeled in various ways. Therefore, in Chapter 1 several methods will be discussed briefly including their advantages and disadvantages. Moreover, the goals and quantities of interest are outlined.

The method that is going to be used is called Stochastic Collocation and the quantities of interest can be written as integrals. Therefore, a main challenge will be constructing integration methods. These will be covered in Chapter 2 and Chapter 3. Chapter 2 will cover one-dimensional integrals and Chapter 3 will cover multi-dimensional integrals.

Both chapters contain discussions about well-known quadrature and cubature rules. For the one-dimensional case, variants of the Gaussian quadrature rule and the Clenshaw–Curtis quadrature rule will be discussed the most thoroughly. In the multi-dimensional case, the tensor cubature rule and

the Smolyak sparse grid will be the most important. In both cases a new quadrature or cubature rule is introduced based on the removal of points of a quadrature or cubature rule, which is called a reduction step. The resulting quadrature (or cubature) rule is therefore called a reduced quadrature (or cubature) rule.

In Chapter 4 the model uncertainties are taken into account and a method is proposed to deal with those uncertainties point-wise in a quadrature or cubature rule.

The implementation of these methods is non-trivial, which will be discussed in Chapter 5. All relevant algorithms that cannot be implemented straightforwardly in one's favorite programming language will be discussed and hints and tips will be given to improve the implementations even further.

Finally, all methods will be applied to two test cases in Chapter 6. Moreover, several integrals of test functions of the proposed cubature rule will be checked with Smolyak and tensor cubature rules.

This report will be mainly about quadrature and cubature rules and this notion will be used before it is even introduced formally (otherwise the beginning of this report would be a bit formal). A *quadrature rule* is one-dimensional and the well-known way to estimate an integral. A *cubature rule* is actually the same as a quadrature rule, but then multi-dimensional. The distinction is made clearly because the deduction of cubature rules is much more cumbersome than the deduction of quadrature rules.

Acknowledgments

This report is my master thesis of the master Industrial and Applied Mathematics of Eindhoven University of Technology (TU/e). The work has been supported by the Centrum Wiskunde & Informatica by providing me a workplace.

During the research, two persons were supervising my project. One took care of the daily supervision and one was responsible for the supervision from the university.

The daily supervisor was Jeroen Witteveen. He has been of great importance for the result in this report by providing references, checking results and keeping a global overview. Moreover, he gave the unique possibility to let me give a presentation representing the CWI and the TU/e in the UMRIDA workshop, an international workshop in Delft with emphasis on Uncertainty Quantification. Unfortunately Jeroen passed away in July 2015, a great and very tragic loss.

The other supervisor is Barry Koren, who is a professor at the Eindhoven University of Technology. He is the global supervisor of the research, took care of the project as a whole and kept a global overview. Besides this, he could also answer all relevant questions. After the loss of Jeroen he took the responsibility of the supervision on its own and made sure the project would succeed successfully. He also deserves a great thanks from me.

Contents

Chapter 1

Stochastic Expansion Methods

Let (Ω, Σ, P) be a probability space with Ω the set of possible events, Σ the σ-algebra on Ω and P the probability measure. Let X be a Banach space and let $u : \Omega \times X \mapsto \mathbb{R}$ be a random field. The set-up will be as follows. u is the quantity of interest with Ω the set of uncertain parameters and X a spatial parameter. Time-dependencies and more certain parameters can be modeled by increasing the dimension of X, with a little abuse of notation. The main problem is to determine the statistical properties of the random field $u(\omega, \mathbf{x})$, like the probability density function (PDF) and the moments.

Three methods are briefly discussed. The first method is the well-known Monte Carlo method, which is based on creating random samples of the space Ω. The other two methods are both based on expressing the unknown quantity in known functions.

After the methods have been outlined, this chapter is concluded with a small discussion about the methods and the main goals of the project are outlined.

1.1 Monte Carlo

Monte Carlo (MC) methods are methods that use samples that represent the distribution. The principle is to create a finite number of sample points $\omega_1, \omega_2, \ldots, \omega_N$ from the space Ω and evaluate $u(\omega_k, \mathbf{x})$ for each sample. Statistics can be applied to $\{u(\omega_k, \mathbf{x})\}_{k=1}^N$ to determine statistical properties like the PDF, moments, and extrema. For example, the mean can be estimated with the well-known formula

$$\mathbb{E}(u) \approx \frac{1}{N} \sum_{k=1}^{N} u(\omega_k, \mathbf{x}). \tag{1.1}$$

Here, $\mathbb{E}(u)$ is a function from X to $\mathbb{R}$.

The absolute error of (1.1) has a convergence rate of $\mathcal{O}(\frac{1}{\sqrt{N}})$, i.e., decreasing the error by a factor 10 requires 100 times more samples. Although this is too expensive in many cases, the method is still used much today because the convergence rate does not depend on the dimensionality of the problem (i.e., the number of uncertain parameters). Quantifying the uncertainties using this method of a model with 3 input parameters is as expensive as quantifying the uncertainties of a model with 300 input parameters.

Because of the bad convergence rate, several improvements have been made to the MC method. For example, the Quasi Monte Carlo (QMC) method is the same as the MC method described above, besides the fact that the sampling strategy is not completely random anymore and constrained by conditions. Lattice Hypercube sampling is such a method and demands that the sampling points can

be placed in a lattice square (or hypercube in multi-dimensional settings) with at most one sample in each row and column. (Ye, 1998; Caflisch, 1998)

The Multilevel Monte Carlo (MLMC) method does not improve the Monte Carlo method itself but improves the evaluation of the discrete model u if it is a numerical model. To this end, multiple coarse and fine numerical grids are generated. A large set of samples is being calculated using a coarse grid and a small set of samples on a fine grid. Balancing the numerical error and the convergence rate of (1.1) should improve the convergence of the MC method as a whole.

Combining both improvements is also possible, which is called the Multilevel Quasi Monte Carlo (MLQMC) method. Constructive ways of generating samples and creating the multilevel structure have been developed.

1.2 Polynomial Chaos

The Polynomial Chaos (PC) method is based on the fact that any square-integrable random variable (RV) $y : \Omega \mapsto \mathbb{R}$ can be expressed as (Najm, 2009)

$$y(\omega) = a_0\Gamma_0 + \sum_{i_1=1}^{\infty} a_{i_1}\Gamma_1(\xi_{i_1}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} a_{i_1 i_2}\Gamma_2(\xi_{i_1},\xi_{i_2}) + \dots,$$

where $\{\xi_i(\omega)\}_i \subset \Xi$ is a set of independent identically distributed RVs on Ω (most of the times standard Gaussian). Γ_k is the Wiener PC of order k. The expression above can be rewritten more compactly as

$$y(\omega) = \sum_{j=0}^{\infty} \alpha_j \Psi_j(\xi_1, \xi_2, \dots),$$

where $\Psi_j : \Omega \mapsto \mathbb{R}$ polynomials (products of Hermite polynomials if all $\xi_i(\omega)$ are standard Gaussian) and $\alpha_j \in \mathbb{R}$.

If y is a random field and depends on (for example) a spatial parameter $\mathbf{x} \in X$, the spatial parameter can be incorporated into the expression using the notation introduced above:

$$y(\omega, \mathbf{x}) = \sum_{j=0}^{\infty} \alpha_j(\mathbf{x})\Psi_j(\xi_1, \xi_2, \dots).$$

In a practical context, the sum and the dimension of Ω are both truncated such that

$$y(\omega, \mathbf{x}) \approx \sum_{j=0}^{N} \alpha_j(\mathbf{x})\Psi_j(\xi_1, \xi_2, \dots, \xi_d).$$

The method can be used for UQ by choosing a basis $\boldsymbol{\xi}$ and a set of orthogonal polynomials $\{\Psi_j\}$ with respect to the $L_2(\Omega)$-inner product, which allows to determine the truncated PC representation of an RV $u \in L_2(\Omega)$ by orthogonal projection:

$$u(\omega) \approx \sum_{j=0}^{P} u_j \Psi_j(\boldsymbol{\xi}(\omega)), \tag{1.2}$$

where

$$u_j = \frac{\langle u, \Psi_j \rangle}{\langle \Psi_j, \Psi_j \rangle},$$

with $\langle f, g \rangle$ the L_2-inner product and $\xi \in \Xi$. The method shows many similarities with the Finite Element Method, because it also solves the problem by expressing the unknown in a known and finite basis. The problem has been translated to determining the coefficients $\{u_j\}_j$, which can be done in several ways.

1.2.1 Intrusive methods

If the original model is a set of partial differential equations, it might be possible to determine a weak formulation by the Galerkin projection method (Van Kan et al., 2014). Because now both the numerical model and the UQ model are expanded in a known basis, they can be combined into one big set of governing equations. Note that this only works if the original model is governed by a set of equations that can be written using a weak formulation or other expansion method.

This is called an *intrusive* method. Its biggest advantage is that the resulting set of equations can be used to directly solve for u_j. Its biggest disadvantage is however that the set of equations of the original model is being changed in a non-trivial way, hence simulation code written for the original model must be changed too in a non-trivial way. Because of this disadvantage, intrusive methods are not discussed in this report. More information about such methods can be found in Ghanem and Spanos (1991).

1.2.2 Non-intrusive methods

The second way to proceed is by using a collocation or sampling procedure. First, a set of samples of ξ is chosen (say $\{\xi_k\}_{k=1}^N$). For each sample point, the value of u_j is determined deterministically just as in the MC method and finally u is constructed using (1.2). This is called a *non-intrusive* method and also the Monte Carlo method discussed above is such a method. The biggest advantage of this method is that many algorithms that sample the (for example) standard Gaussian distribution can be used here including all the improvements made to that. Moreover, because u is evaluated at several fixed sample values, existing simulation codes can be reused.

1.3 Stochastic Collocation

The Stochastic Collocation (SC) method is a technique similar to the PC method because it also expresses the unknown in a finite set of functions. However, this time not necessarily a basis is chosen. It is customary to choose the Lagrange interpolation polynomials. Let $\{L_k\}_{k=1}^N$ be these polynomials with respect to a point set $\{\omega_k\}_{k=1}^N$, then u is approximated using

$$u(\omega, \mathbf{x}) \approx \sum_{k=1}^{N} u(\omega_k, \mathbf{x}) L_k(\omega). \tag{1.3}$$

The functions are chosen such that they depend on a set of samples $\{\omega_k\}_{k=1}^N$, such that the estimation above depends on N function evaluations of u (hence the method is non-intrusive). If the interpolation points are chosen well and u is sufficiently smooth the convergence is exponential (Eldred and Burkardt, 2009).

The biggest advantage of choosing Lagrange interpolation polynomials is that calculating (high) moments of u boils down to evaluating a quadrature or a (tensor) cubature rule. This can be seen as follows. First, fix N quadrature rule points $\{\omega_k\}_{k=1}^N$. Let $\{L_k(\omega)\}$ be the Lagrange interpolation

polynomials with respect to these points and calculate u with (1.3). Calculating $\mathbb{E}(u)$ yields:

$$\begin{aligned}\mathbb{E}(u) &= \int_\Omega u \, \mathrm{d}P(\omega) \\ &\approx \int_\Omega \sum_{k=1}^{N} u(\omega_k, \mathbf{x}) L_k(\omega) \, \mathrm{d}P(\omega) \\ &= \sum_{k=1}^{N} u(\omega_k, \mathbf{x}) \int_\Omega L_k(\omega) \, \mathrm{d}P(\omega) \\ &= \sum_{k=1}^{N} u(\omega_k, \mathbf{x}) w_k, \end{aligned} \tag{1.4}$$

if the weights are chosen such that $w_k = \int_\Omega L_k(\omega) \, \mathrm{d}P(\omega)$ and all the integrals exist. For higher moments, the same quadrature rule can then be re-used.

This can be generalized even further by skipping the interpolation procedure and choosing the sample points such that the quadrature rule of (1.4) can be evaluated as accurately as possible. In high-dimensional problems, less points are needed because lower bounds of the necessary number of points of interpolation schemes are more stringent than those of cubature rules.

1.4 Assumptions and goals

All three methods described above have their advantages and disadvantages, and there is not one method that can be used in all cases for all conditions. Moreover, many more intrusive and non-intrusive methods exist (sometimes based on completely different principles) that do not have the issues of the presented methods but have their own subtleties (Najm, 2009).

To select the "best" method, first several assumptions of the current research are stated to decide what "best" means. These assumptions are based on the Computational Fluid Dynamics (CFD) cases that will be solved using the method. From this, the method that is going to be used is chosen and the main goals are outlined.

1.4.1 Assumptions

The focus will be on non-intrusive methods and it is assumed that u can be evaluated for fixed values of the input parameters. Moreover, all parameters are real-valued and the domain of u is compact (i.e., u can be evaluated for a range of values).

Furthermore, it is assumed that the d uncertain parameters are independent, real-valued, and have an explicit probability density function (PDF) and cumulative density function (CDF). This means that Ω and ω can be written as $\Omega_1 \times \Omega_2 \times \cdots \times \Omega_d$ and $(\omega_1, \omega_2, \ldots, \omega_d)$ respectively and $\omega \in \Omega \subset \mathbb{R}^d$.

The number of uncertain parameters of the CFD cases studied in this report is smaller than 25. This means that the dimension of Ω is bounded by 25*.

Finally, calculating the expectation, variance, PDF, and other statistical properties of u is of more interest than the calculation of u itself. So the emphasis will be on calculating integral quantities of u and not on interpolation polynomials of u. In the one-dimensional case, quadrature rules are often based on interpolating u, so this emphasis will be the most important in the multi-dimensional case.

*Examples of models that depend on too many uncertain parameters are porous media models where each grain is uncertain and traffic simulations where discrete event simulation is used (such that the arrival of each car is uncertain). The number of grains or cars is typically larger than 25.

1.4.2 Goals

Concluding, Stochastic Collocation is the method that is going to be applied in this report. The strength of the PC method is that it can calculate u very precisely, but the current interest is in the statistical properties of u which are most of the time integrals of u and not u itself. Moreover, the SC method converges often faster than the PC method (Eldred and Burkardt, 2009). The MC method is a good method for high dimensional problems, but the CFD simulations that are going to be considered are not of high dimensionality.

The main problem of the current research is to study different deterministic algorithms that generate quadrature or cubature rule sample points for the Stochastic Collocation method and improve these methods. The improvements made are focused on lowering the number of required points that are necessary to reach a certain accuracy, because each point is equal to one simulation.

Chapter 2

Quadrature rules

The space of possible input parameters is multi-dimensional because there is more than one uncertain input parameter. However, many algorithms that generate quadrature rules for multi-dimensional spaces use one-dimensional quadrature rules. Therefore a clear distinction is made in this report. As already stated, a *quadrature rule* is one-dimensional. A multi-dimensional quadrature rule is called a *cubature rule*. The focus of this chapter will be on quadrature rules. The construction of cubature rules from quadrature rules and the direct construction of cubature rules will be the topic of the next chapter.

Calculating moments of a function is the same as calculating an integral. Let u be a function as above and p be a PDF, then the problem becomes determining the integral

$$\mathbb{E}(u) = \mathscr{I}u = \int_a^b u(\omega)p(\omega)\,\mathrm{d}\omega.$$

as exact as possible for any $a, b \in \mathbb{R} \cup \{-\infty, \infty\}$ and $a < b$ with respect to the Lebesgue measure. The spatial parameter has been dropped because that can be incorporated by calculating multiple integrals for each spatial point. If P is the cumulative density function (CDF, so $P(y) = \int_{-\infty}^{y} p(\omega)\,\mathrm{d}\omega$), then the problem can be written as a Stieltjes-integral

$$\mathscr{I}u = \int_a^b u(y)\,\mathrm{d}P(y).$$

Both notations will be used without clarification. Moreover, the measure induced by the CDF is used as integration measure for all L_p spaces hereafter (unless stated otherwise).

This chapter is constructed as follows. First, general properties of quadrature rules are given. The relevance of the properties for the SC method are also discussed using examples. Several well-known quadrature rules are discussed including their properties. Finally, a construction is proposed that combines all relevant properties of the discussed quadrature rules. This construction is of importance, because it can be generalized to a multi-dimensional setting.

It is customary to denote the quadrature rule points using the letter x, such that the points become the set $\{x_k\}$. To make notation a bit less cumbersome, the integral is denoted as

$$\mathscr{I}u = \int_a^b u(x)p(x)\,\mathrm{d}x,$$

and the bound variable of the integral becomes x.

2.1 General properties and nomenclature

A quadrature rule is a linear functional $\mathscr{Q}_N : L_1([a,b]) \mapsto \mathbb{R}$ constructed with a set of N distinct points $\{x_k\}_{k=1}^N$ and a set of weights $\{w_k\}_{k=1}^N$, defined as follows:

$$\mathscr{Q}_N u := \sum_{k=1}^{N} u(x_k) w_k.$$

If the points $\{x_k\}$ and the weights $\{w_k\}$ are chosen well, this functional can be used to estimate $\mathscr{I}u$. $\mathscr{Q}_N$ could also be called an N-point quadrature rule, because most algorithms that generate a quadrature rule have N as argument (i.e., they can generate quadrature rules with any number of points)*.

Any quadrature rule discussed hereafter fits in this framework. It is required in the current setting that $\{x_k\}_{k=1}^N \subset [a,b]$, otherwise the quadrature rule could evaluate the function u at values where it is not defined.

To measure the accuracy or quality of a quadrature rule, the notion of the degree of a quadrature rule is introduced.

Definition 1 (Degree of a quadrature rule)**.** *Let $\mathscr{Q}_N$ be an N-point quadrature rule. Then this quadrature rule has degree $K \in \mathbb{N}$ if*

$$\mathscr{Q}_N p = \mathscr{I} p, \ \forall p \in \mathbb{P}(K),$$

and

$$\exists p \in \mathbb{P}(K+1) : \mathscr{Q}_N p \neq \mathscr{I} p.$$

Here, $\mathbb{P}(K)$ is the space of all polynomials of degree less or equal than K.

Because $\mathscr{Q}_N$ and $\mathscr{I}$ are both linear, the following corollary is trivial.

Corollary 2. *Let $\mathscr{Q}_N$ be an N-point quadrature rule. Then this quadrature rule has degree $K \in \mathbb{N}$ if and only if*

$$\mathscr{I}(x \mapsto x^j) = \mathscr{Q}_N(x \mapsto x^j),$$

for all $j = 0, \ldots, K$ and

$$\mathscr{I}(x \mapsto x^{K+1}) \neq \mathscr{Q}_N(x \mapsto x^{K+1}).$$

If the quadrature rule is applied on a general function u, the following result is of interest.

Theorem 3. *Let $\mathscr{Q}_N$ be a quadrature rule of degree K. Then for any function $u \in C^{(K+1)}([a,b])$,*

$$\mathscr{I}u - \mathscr{Q}_N u = \frac{C}{(K+1)!}\left(\mathscr{I}x^{K+1} - \mathscr{Q}_N x^{K+1}\right),$$

with $C = u^{(K+1)}(\xi)$, for a certain $\xi \in [a,b]$.

Proof. Start by rewriting u in its Taylor series:

$$u(x) = \sum_{i=1}^{K} u^{(i)}(y)\frac{(x-y)^i}{i!} + \frac{(x-y)^{K+1}}{(K+1)!}u^{(K+1)}(\xi),$$

*Note that according to the current notation, it might seem that $\mathscr{Q}_N$ depends on N only, but that is of course not the case as the points and weights are still necessary. $\mathscr{Q}_N$ is chosen just to make the notation less cumbersome (instead of the formally correct $\mathscr{Q}_{\{x_k\}_{k=1}^N,\{w_k\}_{k=1}^N}$) and making the notation more compact. Hereafter, the reader can safely assume that $\mathscr{Q}_N$ is an N-point quadrature rule with points $\{x_k\}_{k=1}^N$ and weights $\{w_k\}_{k=1}^N$.

for a fixed $y \in [a,b]$ and a certain $\xi \in [a,b]$ with $u^{(K+1)} = \frac{\partial^{K+1} u}{\partial x^{K+1}}$. Substituting this into $\mathscr{I}$ becomes

$$\begin{aligned}\mathscr{I}u(x) &= \mathscr{I}\left(\sum_{i=1}^{K} u^{(i)}(y)\frac{(x-y)^i}{i!} + \frac{(x-y)^{K+1}}{(K+1)!}u^{(K+1)}(\xi)\right) \\ &= \sum_{i=1}^{K} u^{(i)}(y)\mathscr{I}\frac{(x-y)^i}{i!} + \mathscr{I}\frac{(x-y)^{K+1}}{(K+1)!}u^{(K+1)}(\xi). \end{aligned} \tag{2.1}$$

Substituting the Taylor series into the quadrature rule is

$$\begin{aligned}\mathscr{Q}_N u(x) &= \mathscr{Q}_N\left(\sum_{i=1}^{K} u^{(i)}(y)\frac{(x-y)^i}{i!} + \frac{(x-y)^{K+1}}{(K+1)!}u^{(K+1)}(\xi)\right) \\ &= \sum_{i=1}^{K} u^{(i)}(y)\mathscr{Q}_N\frac{(x-y)^i}{i!} + \mathscr{Q}_N\frac{(x-y)^{K+1}}{(K+1)!}u^{(K+1)}(\xi) \\ &= \sum_{i=1}^{K} u^{(i)}(y)\mathscr{I}\frac{(x-y)^i}{i!} + \mathscr{Q}_N\frac{(x-y)^{K+1}}{(K+1)!}u^{(K+1)}(\xi). \end{aligned} \tag{2.2}$$

Choosing $y = 0$ and computing the difference between (2.1) and (2.2) yields the desired result. □

Any distinct set of N points in $[a,b]$ can be used to construct a quadrature rule of degree $N-1$. This can be seen by considering the following linear system:

$$\begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_N^0 \\ x_1^1 & x_2^1 & \cdots & x_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{N-1} & x_2^{N-1} & \cdots & x_N^{N-1} \end{pmatrix}\begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} \mathscr{I}x^0 \\ \mathscr{I}x^1 \\ \vdots \\ \mathscr{I}x^{N-1} \end{pmatrix}. \tag{2.3}$$

The matrix on the left-hand side of the equation is the *Vandermonde-matrix* of the quadrature rule. The Vandermonde-matrix is non-singular if the quadrature points are distinct, hence the linear system has a solution and there exist weights such that the quadrature rule is of degree $N-1$.

Although interpolating u using the set $\{x_k\}$ is not the main topic of this report, the following result is also worth mentioning. The proof is omitted.

Theorem 4. *Let $u \in C^N([a,b],\mathbb{R})$, let $\hat{u}$ be the Lagrange interpolating polynomial of u using the nodes $\{x_k\}_{k=1}^N$, then for any $x \in [a,b]$*

$$u(x) - \hat{u}(x) = \frac{K(x)}{N!}u^{(N)}(\xi),$$

with $K(x) = \prod_{j=1}^{N}(x - x_j)$ and a certain $\xi \in [a,b]$. Trivially, $u(x_k) = \hat{u}(x_k)$.

Several properties are of interest of a quadrature rule. Because no applications of quadrature rules have been discussed so far, each property also contains an example explaining why the property might be important.

2.1.1 Positive weights

The positivity of the weights (i.e., $w_k > 0$ for all k) is an important property of a quadrature rule because without it, one might get negative valued integrals for positive valued functions. In automated systems where nonphysical answers are not desired, this might lead to problems as can be seen in the following example.

Example 5. *Suppose a chemical company measures the average temperature of a pipe and if this becomes above a certain threshold, it closes down the facility to let the system cool down. To measure the average temperature, a formula of the form* $\frac{1}{L}\int_{pipe} T(x)\,\mathrm{d}x$ *is used, where* L *is the length of the pipe and* T *the temperature. The integral is calculated using measurement points on the pipe at the location of quadrature rule points. Some of the points have negative weights.*

Due to a small explosion in the pipe, a sudden temperature increase happens. Unfortunately, this happens at the location of a quadrature point with a negative weight. The real average temperature now increases, but the value calculated by the quadrature rule decreases due to negative weight. The facility should close now automatically, but it does not because it cannot notice the sudden temperature increase.

When the function u is not polynomial, positivity of u does not automatically imply positivity of the estimated value of the integral if there are negative weights. In a paper which appeared in 1957, V. Tchakaloff proved that for any K there exists an $(K+1)$-point quadrature rule of degree K with positive weight (Davis, 1967)[†]. The result of Tchakaloff also holds for cubature rules and is known as the Tchakaloff bound.

Note that if positivity of the weights is necessary, it is not possible to choose a set of quadrature points in $[a,b]$ and find the weights using (2.3) because the weights might become negative.

2.1.2 Stability

Closely related to positivity of the weights is the stability of the quadrature rule. A quadrature rule is called stable if

$$\lim_{N\to\infty} \mathcal{Q}_N u = \mathcal{I} u,$$

for analytic functions u (i.e., the Taylor series exists).

Closely related to the stability is the operator norm of the quadrature rule, which can be calculated as follows

$$\begin{aligned}\|\mathcal{Q}_N u\| &= \left\| \sum_{k=1}^{N} u(x_k) w_k \right\| \\ &\leq \sum_{k=1}^{N} \|u(x_k)\|\,\|w_k\| \\ &\leq \sum_{k=1}^{N} |w_k|\,\|u(x_k)\|,\end{aligned}$$

so $\|\mathcal{Q}_N\| = \sum_{k=1}^{N} |w_k|$, independent of the chosen norm. The quadrature rule is stable if $\mathcal{Q}_N$ is of degree $N-1$ and

$$\lim_{N\to\infty} \|\mathcal{Q}_N\| < \infty.$$

Note that a quadrature rule with positive weights that integrates a constant function is by definition stable, because then $\sum_{k=1}^{N} |w_k| = \sum_{k=1}^{N} w_k = b-a$, which is the smallest value possible.

2.1.3 Nested

Nested quadrature rules are quadrature rules for which a strictly increasing sequence $(N_k)_{k=1}^{\infty}$ exists such that the quadrature points of $\mathcal{Q}_{N_j}$ are also quadrature points of $\mathcal{Q}_{N_{j+1}}$. The property is not important in general, but is handy as can be seen in the following example.

[†]The original paper of V. Tchakaloff is not freely available on the internet and both the CWI and the TU/e do not have access rights to the published version of the paper unfortunately.

Example 6. *Suppose a researcher is using a non-nested quadrature rule and applies it to a UQ problem. Unfortunately, after the simulations have run for two weeks, it turns out that the resulting estimation is not precise enough and he needs to do more simulations. Because his used quadrature rule is not nested, he cannot use the simulations he has done so far for his new result so he loses a lot of time.*

A quadrature rule is called finitely nested if there does not exists a sequence $(N_k)_{k=1}^{\infty}$ as above but there does exist such a sequence $(N_k)_{k=1}^{K}$ of finite length.

Cubature rules are often generated using quadrature rules by combining several quadrature rules of different numbers of points. There are less points in the resulting cubature rule if the original quadrature rule is nested. This effect will be studied more thoroughly in the next chapter.

2.2 Gaussian quadrature rules

The Gaussian quadrature rule is a quadrature rule depending on the distribution with positive weights. The quadrature rule exists for all numbers of points, although the definition does not include a constructive algorithm.

Reconsider $\mathbb{P}(N)$, the space with all polynomials of degree less or equal N. It is equipped with the following well-known inner product for functions $f, g \in \mathbb{P}_N([a,b])$:

$$\langle f, g \rangle = \int_a^b f(x)g(x)\,\mathrm{d}P(x),$$

which is based on the L_2-inner product of two real function on the measure space (Ω, Σ, P) introduced in the previous chapter.

Let the polynomials $f_0, f_1, \ldots, f_{N-1}, f_N$ form an orthogonal basis of $\mathbb{P}(N)$ with respect to this inner product. Without loss of generality, it is assumed that the degree of f_0 is 0 (i.e., a constant), the degree of f_1 is 1 and in general that the degree of f_k is k. Note that $f_0, f_1, \ldots, f_K$ is then an orthogonal basis of $\mathbb{P}(K)$. The Gaussian quadrature points are defined as the roots of f_N. The weights are such that (2.3) holds.

The Gaussian quadrature rule has a high degree, as can be seen in the next theorem.

Theorem 7. *Let $\mathcal{Q}_N$ be an N-point Gaussian quadrature rule. Then the degree of this quadrature rule is $2N-1$.*

Proof. Let $q(x) \in \mathbb{P}(2N-1)$ be a polynomial of degree $2N-1$. By division with remainder q can be written as $q(x) = l(x)f_N(x) + r(x)$, with l and r polynomials of degree $N-1$. The calculation can be done using the extended algorithm of Euclid. Because $l \in \mathbb{P}(N-1)$, and $f_0, f_1, \ldots, f_{N-1}$ form a basis of that space, l can be written as

$$l = \sum_{j=0}^{N-1} \langle f_j, l \rangle f_j.$$

All f_k are orthogonal, hence

$$\begin{aligned}\int_a^b q(x)\,\mathrm{d}P(x) &= \int_a^b \big(l(x)f_N(x)+r(x)\big)\,\mathrm{d}P(x)\\ &= \int_a^b l(x)f_N(x)\,\mathrm{d}P(x)+\int_a^b r(x)\,\mathrm{d}P(x)\\ &= \sum_{j=0}^{N-1}\big\langle f_j,l\big\rangle\int_a^b f_j(x)f_N(x)\,\mathrm{d}P(x)+\int_a^b r(x)\,\mathrm{d}P(x)\\ &= \sum_{j=0}^{N-1}\big\langle f_j,l\big\rangle\big\langle f_j,f_N\big\rangle+\int_a^b r(x)\,\mathrm{d}P(x)\\ &= \int_a^b r(x)\,\mathrm{d}P(x).\end{aligned}$$

Moreover, if x_k is a Gaussian quadrature point, $q(x_k)=l(x_k)f_N(x_k)+r(x_k)=r(x_k)$. So finally

$$\begin{aligned}\int_a^b q(x)\,\mathrm{d}P(x) &= \int_a^b r(x)\,\mathrm{d}P(x)\\ &= \sum_{k=1}^{N} r(x_k)w_k\\ &= \sum_{k=1}^{N} q(x_k)w_k.\end{aligned}$$

□

The Gaussian quadrature rule has positive weights. Because it has been argued that this is such an important property, this statement is formalized as a theorem.

Theorem 8. *Let $\{x_k\}_{k=1}^N$ be N Gaussian quadrature points with $\{w_k\}_{k=1}^N$ the N weights. Then $w_k \geq 0$ for all $k=1,\ldots,N$.*

Proof. Fix $k\in\{1,\ldots,N\}$. Let q_k be the Lagrange interpolating polynomial such that $q_k(x_j)=\delta_{k,j}$, i.e.,

$$q_k(x)=\prod_{\substack{j\in\{1,\ldots,N\}\\ j\neq k}}\frac{x-x_j}{x_k-x_j},$$

which is a polynomial of degree $N-1$. Consider $q_k^2(x)$, which is a polynomial of degree $2N-2$ with $q_k^2(x_j)=\delta_{k,j}$. The Gaussian quadrature rule is of degree $2N-1$ and this polynomial is non-negative for all real x (and not all-zero), hence

$$\begin{aligned}0 &< \int_a^b q_k^2(x)\,\mathrm{d}P(x)\\ &= \sum_{j=1}^{N} q_k^2(x_j)w_j\\ &= q_k^2(x_k)w_k\\ &= w_k,\end{aligned}$$

which concludes the proof. □

For some $P(x)$, the polynomials f_k have special names which are used without reference. The Gauss quadrature rule is then called Gauss–*something* quadrature rule, where the name of the polynomial is filled in. Four polynomials are of special interest for the current story:

- Legendre polynomials are the polynomials that are orthogonal when $p(x) = 1$, i.e., when just calculating an integral or using a uniform distribution;
- Jacobi polynomials are the orthogonal polynomials of the $\beta(a, b)$-distribution;
- Laguerre polynomials are the orthogonal polynomials of the exponential distribution;
- Hermite polynomials are the orthogonal polynomials of the Gaussian distribution (or the normal distribution, not to be confused with the quadrature rule that is being discussed).

The Gaussian quadrature rule is not necessarily nested. All the distributions in the list above do have that the generated Gaussian quadrature rules are non-nested.

2.2.1 Gauss–Kronrod quadrature

Kronrod (1965) introduced an extension to an N-point Gauss quadrature rule that adds $N+1$ points and is of degree $3N+1$. The construction works for all Gauss quadrature rules, but Kronrod proved its correctness and existence for the Gauss–Legendre quadrature rule. Let $f_0, \ldots, f_N$ be the orthogonal polynomials from above. Suppose there exists a polynomial κ of degree $N+1$ with $N+1$ distinct roots such that $\left\langle \kappa(x)x^k, f_N \right\rangle = 0$ for all $k = 0, \ldots, N$. Then the zeros of the polynomial $f(x)\kappa(x)$ are the Gauss–Kronrod quadrature points. The weights are again defined by (2.3). Note that the Gauss–Kronrod quadrature rule makes the Gauss quadrature rule finitely nested. It can be used to calculate error estimates of the integral, by calculating the estimation using both the Gauss and Gauss–Kronrod quadrature rule and comparing the results.

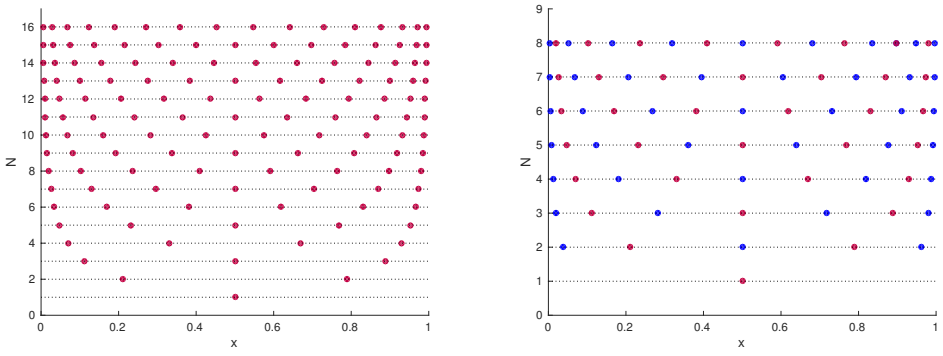
The degree of $3N+1$ is remarkable, because the quadrature rule consists of $2N+1$ points (i.e., the degree is higher than the number of points).

Theorem 9. *The* $(2N+1)$*-point Gauss–Kronrod quadrature rule has degree* $3N+1$.

Proof. Let $\{x_k\}$ be the Gauss–Kronrod quadrature points. Let $q \in \mathbb{P}(3N+1)$ be a polynomial of degree $3N+1$. Then q can be written as $q(x) = l(x)\kappa(x)f_N(x) + r(x)$ by division with remainder. The degree of l is N and the degree of r is $2N$. It can be proved that $q(x_k) = r(x_k)$, so

$$\begin{aligned}
\int_a^b q(x)\,\mathrm{d}P(x) &= \int_a^b \big(l(x)\kappa(x)f_N(x) + r(x)\big)\,\mathrm{d}P(x) \\
&= \int_a^b l(x)\kappa(x)f_N(x)\,\mathrm{d}P(x) + \int_a^b r(x)\,\mathrm{d}P(x) \\
&= \sum_{k=1}^{N} \langle f_k, l\rangle \int_a^b f_k(x)\kappa(x)f_N(x)\,\mathrm{d}P(x) + \int_a^b r(x)\,\mathrm{d}P(x) \\
&= \int_a^b r(x)\,\mathrm{d}P(x) \\
&= \sum_{k=1}^{N} r(x_k)w_k \\
&= \sum_{k=1}^{N} q(x_k)w_k.
\end{aligned}$$

□



(a) *(b)*

Figure 2.1: The Gauss–Legendre quadrature rule (on the left) with the Kronrod extension (on the right, blue are the added points, red is just a Gauss quadrature rule) for several numbers of points. The number of points of the quadrature rule is displayed on the y-axis for the different quadrature rules.

For general functions $P(x)$, the described polynomial κ might not exist or has imaginary roots if it does (Gauss–Hermite rules are such a case). However, if it does exist, no better quadrature rule extension exists (Kronrod, 1965; Laurie, 1997).

Patterson (1968) improved this idea further by studying extensions of general quadrature rules of the uniform distributions. He noticed (and proved) that any such quadrature rule can be improved with $N+1$ points to a quadrature rule of degree $3N+1$. Furthermore, he discusses in his paper how these points can be determined. His algorithm can be extended to general symmetric distributions (i.e., the PDF of the distribution is symmetric around a certain point) and reuses the orthogonal polynomials $f_0, \ldots, f_N$ introduced above. Generalizations of this principle, where less than N points are added, might exist as well even for cases where the Gauss–Kronrod quadrature rule does not exist (e.g., Genz and Keister (1996) studied the Gauss–Hermite case).

2.2.2 Examples

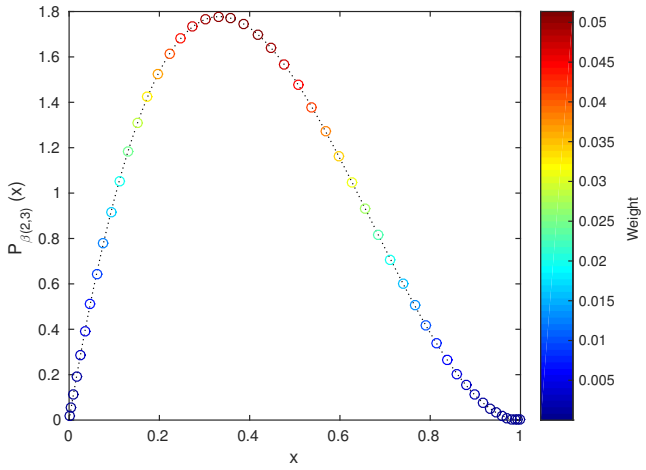
Using the algorithm above together with the Kronrod extension, several Gauss quadrature rules can be constructed. Two results are presented here.

The first one is the Gauss–Legendre quadrature rule and Gauss–Legendre–Kronrod quadrature rule (see Figure 2.1). This distribution is considered to show the non-nesting of the points of the Gauss quadrature rule and the principle of the Gauss–Kronrod quadrature rule.

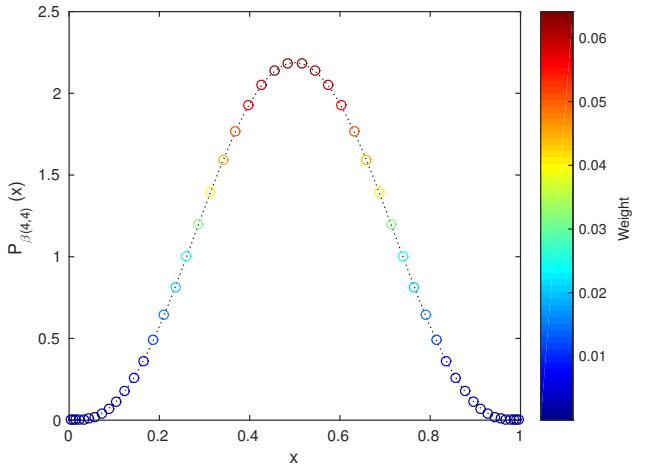
The second one is the Gauss–Jacobi quadrature rule of the β-distribution (see Figure 2.2). The points are plotted on the PDF and colored according to their weights such that the dependence on the distribution in both the placement of the points and the size of the weights is shown.

2.3 Other notable quadrature rules

Although Gaussian quadrature rules will play an important role in the deduction of the quadrature rule extension in the next chapter, several other methods exist as well. Two will be discussed in this section. One is of historical interest and is called the Newton–Cotes formula, which will be briefly discussed for sake of completeness. The other is the Clenshaw–Curtis quadrature rule, which has the



(a) $\beta(2,3)$



(b) $\beta(4,4)$

Figure 2.2: The Gauss–Jacobi quadrature rule for two different β-distributions on $[0,1]$. The PDF is dotted in black.

nice property of nested points and will prove to be a good counterexample later on when several cubature rules are being compared.

2.3.1 Newton–Cotes formulas

Newton–Cotes formulas have the same shape as general quadrature rules, with the condition that the points are equally spaced. They first appeared in a letter from I. Newton to G. Leibniz in 1676 and later in 1722 in the book by R. Cotes. Cotes created in his book one of the first tables with quadrature rule points and weights.

The original formulas did only consider plain integrals (i.e., the uniform distribution). The simplest case of the Newton–Cotes formula is the midpoint rule:

$$\int_a^b f(x)\,\mathrm{d}x \approx (b-a) f\left(\frac{1}{2}(a+b)\right).$$

This is a so-called *open* method, which means that the function is not evaluated at the endpoints a and b.

Using two points, the trapezium formula is recovered:

$$\int_a^b f(x)\,\mathrm{d}x \approx \frac{b-a}{2}\left(f(a)+f(b)\right),$$

which is a *closed* method. Its name comes from the shape that is used to estimate the integral.

The closed three- and four-point rules are called the Simpson's rule and Simpson's 3/8 rule respectively. The last one is of special interest, because that is used in the well-known 4^{th} order explicit Runge–Kutta method.

Although the methods are historically interesting and are still used today for time integration methods, they are not of interest for the current discussion due to their relatively high inaccuracy, lack of positive weights and lack of dependency on the distribution.

2.3.2 Clenshaw–Curtis

The Clenshaw–Curtis quadrature rule (Clenshaw and Curtis, 1960) is a quadrature rule with fixed points, independent of the distribution. It has positive weights if the uniform distribution is considered. The quadrature points of this quadrature rule are the extrema of the Chebyshev polynomials defined on $[-1,1]$ and therefore it could be seen as estimating the integrand $f(x)$ using these polynomials, which are defined by the following recurrence relation:

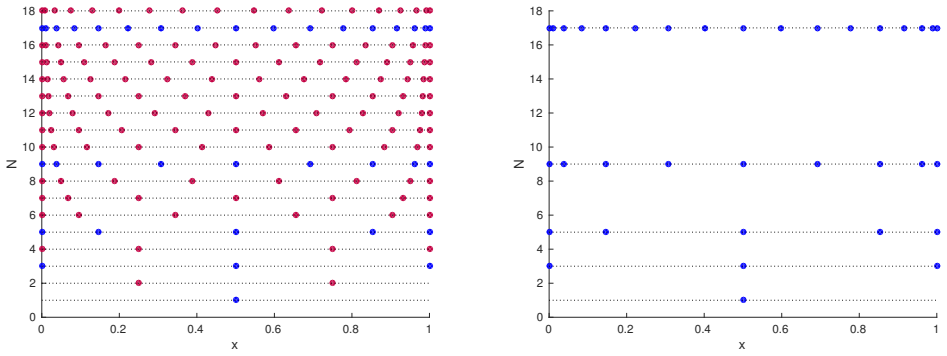
$$T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x),$$

where $T_0(x) = 1$ and $T_1(x) = x$. They have the property that $T_k(\cos\vartheta) = \cos(k\vartheta)$ and using this the quadrature rule can be rewritten as

$$\int_{-1}^{1} f(x)\,\mathrm{d}x = \int_0^{\pi} f(\cos\vartheta)\sin\vartheta\,\mathrm{d}\vartheta.$$

The N extrema (and hence the N quadrature points) of T_N are given by

$$x_k = -\cos\left(\frac{\pi(k-1)}{N-1}\right),$$



(a) *(b)*

Figure 2.3: The Clenshaw–Curtis quadrature rule for several numbers of points. The number of points is shown on the y-axis. On the left, all quadrature rules are displayed and the blue points are nested. On the right, only the blue levels are shown and it can be seen that indeed these are nested.

where $k = 1, \ldots, N$.

The Clenshaw–Curtis quadrature points are nested by choosing the sequence $N_k = 2^k + 1$. Here, k is often called the *level* of the quadrature rule. Moreover, the points are the optimal choice to create an interpolation polynomial because it can be proved that for this set of points the interpolation error $\max_x |K(x)|$ (see Theorem 4) equals 2^{-N} and is the minimal value possible (Gil et al., 2007, Theorem 3.4).

Unfortunately, the Clenshaw–Curtis quadrature points do not depend on the distribution and many points are closer to the boundary of the interval than to the center of the interval. Therefore, if distributions which have a PDF which is small near the boundary (the normal distribution and $\beta(a, b)$-distribution for $a, b > 1$ are two notable examples) are used, the weights in the region close to the center will become large (and maybe negative) while weights close to the boundary will become small. This behavior is unwanted and attempts have been made to overcome this (Evans and Webster, 1999).

2.3.3 Example

The points of the Gaussian quadrature rule depend on the distribution, so many examples could be considered. The Clenshaw–Curtis quadrature rule points do not depend on the distribution and are calculated using a formula. Therefore, there is only one example (see Figure 2.3). This example shows the quadrature rule for a varying number of points and also shows that the quadrature rule is indeed nested.

2.4 Reduced quadrature rules

Both the Gaussian and the Clenshaw–Curtis quadrature rule have nice properties, but the Gaussian quadrature rule is not nested and the Clenshaw–Curtis quadrature rule does not guarantee positive valued weights for general distribution functions. Existence of the Gauss–Kronrod quadrature rule for general distributions is not guaranteed.

Therefore, a quadrature rule is wanted that has positive weights, is nested and can be applied to a large class of distributions. To determine whether such a quadrature rule can be found, an N-

point quadrature rule with positive weights is constructed and the $N \times N$ Vandermonde-matrix V is reconsidered. Recall that $V\mathbf{w} = \mathbf{m}$, with $\mathbf{w}$ the weights and $\mathbf{m}$ the values of the integrals. Because a point must be removed, it is likely that the quadrature rule will be of smaller degree. Therefore, the last row of V and last element of $\mathbf{m}$ is removed, which essentially reduces the degree of the quadrature rule. If a point can be removed such that the weights remain positive, then this is equal to the removal of the respective column of V. The resulting matrix without the last row and the column of the removed point is then the Vandermonde-matrix of the new quadrature rule. The question remains which column can be removed.

There are N columns of V in an $N-1$ dimensional space. The columns do span the space because the Vandermonde-matrix is non-singular, hence they form a basis. $\mathbf{m}$ can be expressed in this basis. Moreover, $\mathbf{m}$ has a coordinate vector with respect to this basis with only positive elements (namely the weights). It is trivial to proof that one vector can be removed such that the others still span $\mathbf{m}$. The main question is however if one vector can be removed such that the others span $\mathbf{m}$ and $\mathbf{m}$ still has only positive elements in its coordinate vector.

This main question can be answered using Carathéodory's theorem. The proof of the theorem is constructive and can be used to remove a point from a quadrature rule. The removal of a point is called a *reduction step* and the resulting quadrature rule is therefore called a *reduced quadrature rule*.

2.4.1 Carathéodory's Theorem

Consider a quadrature rule with positive weights. The central theorem of this section is the following.

Theorem 10. *Let $\{x_k\}_{k=1}^N$ be N quadrature rule points and $\{w_k\}_{k=1}^N$ the N weights. Assume that the quadrature rule is (at least) of degree $N-1$. Then there exists a quadrature point x_j such that the quadrature rule consisting of the other $N-1$ points is of degree $N-2$ and has positive weights.*

To prove this theorem, Carathéodory's theorem can be used. Carathéodory's theorem is actually a theorem of convex optimization. However, the problem can be rewritten into finding a different basis of a cone, which can be written as an optimization problem.

Theorem 11 (Carathéodory's theorem)**.** *Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N, \mathbf{v}_{N+1}$ be $N+1$ vectors spanning an N-dimensional space. Let $\mathbf{v} = \sum_{k=1}^{N+1} \lambda_k \mathbf{v}_k$ with $\lambda_k \geq 0$. Then there exist $\beta_k \geq 0$ such that $\mathbf{v} = \sum_{k \in I} \beta_k \mathbf{v}_k$ and $I \subset \{1, \ldots, N+1\}$ with $|I| \leq N$.*

Proof. Because $\mathbf{v}_1, \ldots, \mathbf{v}_{N+1}$ are $N+1$ vectors in an N-dimensional space, they must be linearly dependent. So there are c_k not all equal to zero such that

$$\sum_{k=1}^{N+1} c_k \mathbf{v}_k = 0.$$

So for any $\alpha \in \mathbb{R}$, it is true that

$$\begin{aligned} \mathbf{v} &= \sum_{k=1}^{N+1} \lambda_k \mathbf{v}_k - \alpha \sum_{k=1}^{N+1} c_k \mathbf{v}_k \\ &= \sum_{k=1}^{N+1} (\lambda_k - \alpha c_k) \mathbf{v}_k. \end{aligned}$$

Without loss of generality, it is assumed that at least one $c_k > 0$. Then the following choice is well-defined.

$$\alpha = \min_{k=1,\ldots,N} \left\{ \frac{\lambda_k}{c_k} : c_k > 0 \right\} =: \frac{\lambda_{k_0}}{c_{k_0}}.$$

Hence α is the minimal value of the set $\{\frac{\lambda_k}{c_k} \mid c_k > 0, k = 1, \ldots, N\}$ and k_0 is such that $\alpha = \frac{\lambda_{k_0}}{c_{k_0}}$.

Choosing $\beta_k = \lambda_k - \alpha c_k$, it is true that $\beta_{k_0} = 0$ so with $I = \{1, 2, \ldots, k_0 - 1, k_0 + 1, \ldots, N\}$ the following holds:

$$\mathbf{v} = \sum_{k \in I} \beta_k \mathbf{v}_k. \qquad \square$$

Now proving Theorem 10 can be done by reconsidering (2.3) and removing the last equation, i.e.,

$$\begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_N^0 \\ x_1^1 & x_2^1 & \cdots & x_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{N-2} & x_2^{N-2} & \cdots & x_N^{N-2} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} \mathscr{I}x^0 \\ \mathscr{I}x^1 \\ \vdots \\ \mathscr{I}x^{N-2} \end{pmatrix}.$$

This matrix is $N \times (N-1)$ and the Vandermonde-matrix is non-singular, hence the N columns of the matrix span an $(N-1)$-dimensional space. Hence Carathéodory's theorem can be applied, one weight and point can be removed such that the other weights remain positive.

Carathéodory's theorem outlines a constructive way to generate the nested quadrature rule. The key consideration is that the numbers c_k form a null vector of the matrix above, so an algorithm based on the proof can be created (see Algorithm 1).

Remark 12. *Note that the elements β_k in the proof of the Carathéodory's theorem are not unique in general. If there are both negative and positive β_k, then applying the proof with $-\beta_k$ yields a different choice of β_k. For the quadrature rule this means that the constructed nested rule is not unique and multiplying the null vector with -1 does also yield a correct null vector if the null vector contains both positive and negative elements.*

The numbers $\mathscr{I}x^k$ are not used in this algorithm. These numbers decay or increase rapidly depending on a and b, so it is a nice property that these are not used. However, a null vector has to be determined. Because this is a subtle procedure (and straightforward methods do not work for general quadrature rules), this will be discussed later.

2.4.2 Selection strategies

If a quadrature rule is given, its reduced quadrature rule is not necessarily unique: if $\mathbf{c}$ is a null vector then so is $-\mathbf{c}$ and if $\mathbf{c}$ contains both positive and negative elements both null vectors can be used to remove points. Each null vector will remove a different point, so it is possible to choose a point such that certain criteria are met.

There does not exist one perfect criterion, because this choice depends on the application of the quadrature rule. Several criteria are discussed here. Globally, the criteria can be fitted in two groups: criteria that use the distribution and criteria that do not. Each criteria compares two quadrature rules and selects the best one. In the next subsection, all criteria are compared and also disadvantages are discussed.

If the distribution is not taken into account, the following options can be considered.

- A quadrature rule can be used to construct interpolation polynomials. Therefore, it is possible to select the quadrature rule such that the interpolation error of Theorem 4 is the smallest.

Algorithm 1 Nested reduced quadrature rule

Input: Let $\{x_k\}_{k=1}^N$ and $\{w_k\}_{k=1}^N$ be an N-point quadrature rule with positive weights and at least of degree $N-1$.

Output: k_0 and new weights $\{w_k^*\}_{k=1}^N$ such that $w_{k_0}^* = 0$ and all weights are non-negative.

1: Construct the Vandermonde-matrix:

$$\begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_N^0 \\ x_1^1 & x_2^1 & \cdots & x_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{N-2} & x_2^{N-2} & \cdots & x_N^{N-2} \end{pmatrix}.$$

2: Determine a null vector $\mathbf{c}$ of the matrix above. Let c_k be the k^{th} coefficient of the null vector.

3: Determine k_0 such that

$$\frac{w_{k_0}}{c_{k_0}} = \min_{k=1,\ldots,N} \left\{ \frac{w_k}{c_k} : c_k > 0 \right\}.$$

4: Return k_0 and new weights $\{w_k^*\}_{k=1}^N$ such that $w_k^* = w_k - \alpha c_k$.

- If the removed point had a large weight, it might be possible that other weights get a large value in the smaller quadrature rule. To overcome this, the quadrature rule can be chosen that has the smallest maximum weight.
- If it is known to what type of functions the quadrature rules are applied, the quadrature rule can be selected that integrates a set of functions of that type the most accurately. This option is not discussed in this section, because no applications are considered.

If the distribution is taken into account, it can be used to construct a few other criteria.

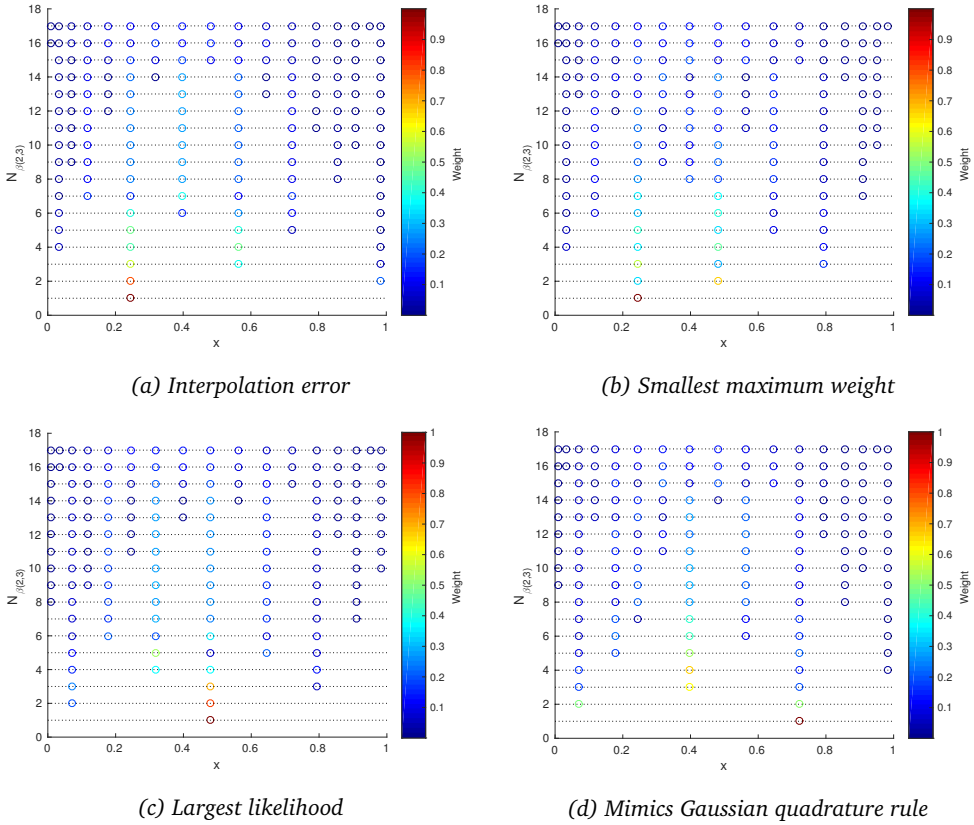
- The (statistical) likelihood of the two quadrature rules can be determined and the quadrature rule with the largest likelihood can be considered as the "best" representation of the distribution.
- If the distribution and the original quadrature rule are symmetric, the quadrature rule can be constructed such that it also is symmetric. Note that it is necessary to remove two points then (so four different quadrature rules have to be considered).
- Each time a Gaussian quadrature can be generated that can be considered as the "perfect" quadrature rule. The quadrature rule that is the closest to this Gaussian quadrature rule can be considered to be better.

It might be possible that there is no choice at all, because all elements of $\mathbf{c}$ are (non-)negative. Examples are discussed of all criteria above.

2.4.3 Examples

Two distributions are considered: the $\beta(2,3)$-distribution and the uniform distribution.

The $\beta(2,3)$-distribution is a non-symmetric distribution. The criteria to optimize the interpolation error, the weights, and the likelihood and the criterion to select the quadrature rule that mimics a



(a) Interpolation error

(b) Smallest maximum weight

(c) Largest likelihood

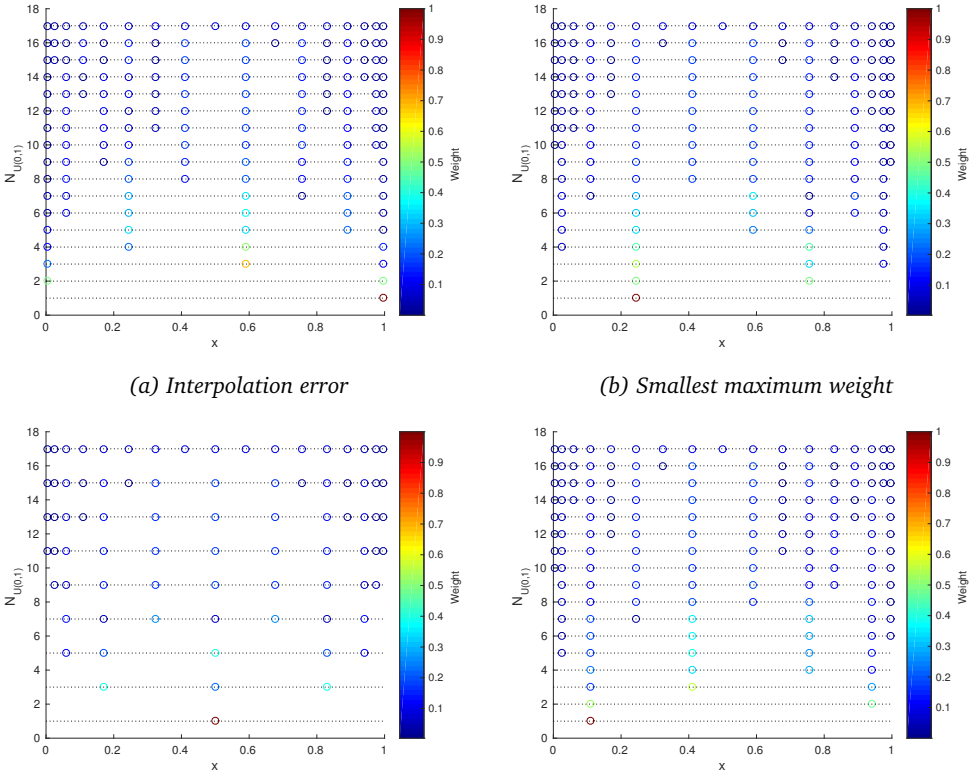
(d) Mimics Gaussian quadrature rule

Figure 2.4: The reduced Gauss–Hermite quadrature rule for several strategies described in the text. The number of points is shown on the y-axis. The distribution is each time a $\beta(2,3)$-distribution.

Gaussian quadrature rule are all possible (see Figure 2.4). Note that the expectation of a $\beta(2,3)$-distribution equals $\frac{2}{5}$, such that mimicking a Gaussian turns out to be a bad strategy because only a few points are close to the expectation. The other strategies produce pretty acceptable results.

The uniform distribution is a symmetric distribution. Because the PDF is constant, likelihood optimization is impossible. However, the 17-point Gaussian quadrature rule is symmetric, such that the quadrature rule can be kept symmetric by removing 2 points each step (see Figure 2.5). In this case, optimizing the interpolation error gives the worst result. Only two points at the end of the interval are kept, which is undesired if integration routines are applied (especially if the quadrature rule is used for a cubature rule, which is discussed in the next chapter). Also mimicking a Gaussian quadrature rule results in bad results because of this reason.

Concluding, if optimizing for the likelihood is possible, it is probably the best method. If the original distribution is symmetric, then the quadrature rule should be kept symmetric.



(a) Interpolation error *(b) Smallest maximum weight*

(c) Symmetric quadrature rule *(d) Mimics Gaussian quadrature rule*

Figure 2.5: The reduced Gauss–Legendre quadrature rule for several strategies described in the text. The number of points is shown on the y-axis.

Chapter 3

Cubature rules

Generating multi-dimensional quadrature rules (cubature rules hereafter) can be done by considering the space of all polynomials, formalizing the requirements of a general quadrature rule, and solving the system. However, solving high dimensional polynomial systems is not a trivial task and proving existence and uniqueness of solutions is not easy.

To overcome this situation, several algorithms have been constructed that create cubature rules from quadrature rules. Three variants will be discussed. The first two are known from literature and are the tensor product (which is actually the straightforward generalization of the quadrature rule) and the Smolyak sparse cubature rule. The third algorithm is a proposed algorithm and is an extension of the introduced reduced quadrature rule. Many other variants exist when the degree of a cubature rule should be optimized against the number of points, see Cools (1992) for a thorough survey.

Before all the methods will be discussed, some notation must be introduced. Because the spaces are now high-dimensional, polynomials are multi-variate and notation is not trivial anymore.

3.1 Nomenclature

The already introduced notation remains the same, so Ω is still the set of possible events. d is the dimension of Ω (i.e., the number of uncertain parameters). It is assumed that all input parameters are independent, such that $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_d$ with Ω_k one-dimensional. Because integration is a linear functional and all input parameters are independent, without loss of generality it can be assumed that $\Omega = [0,1]^d$.

The CDF of the distribution of the uncertain input parameters is again denoted as P. Because the input parameters are mutually independent, $P(x_1, x_2, \ldots, x_d)$ can be written as $P(x_1, x_2, \ldots, x_d) = P_1(x_1) \cdot P_2(x_2) \cdot \cdots \cdot P_d(x_d)$, where P_k is the CDF of the k^{th} input parameter.

The space of interest is the d-dimensional space of polynomials of degree K, denoted by $\mathbb{P}(K,d)$. The space of absolute integrable functions is likewise denoted as $L_1([0,1]^d)$. A function $u \in \mathbb{P}(K,d)$ or $u \in L_1([0,1]^d)$ is then a function of $\mathbf{x} = \left(x^{(1)}, x^{(2)}, \ldots, x^{(d)}\right)$ to $\mathbb{R}$. Monomials are denoted as $\mathbf{x}^{\boldsymbol{\alpha}}$, with*$\boldsymbol{\alpha} \in \mathbb{N}^d$, so

$$\mathbf{x}^{\boldsymbol{\alpha}} = \left(x^{(1)}\right)^{\alpha_1} \left(x^{(2)}\right)^{\alpha_2} \cdots \left(x^{(d)}\right)^{\alpha_d}.$$

Note that $\mathbf{x}_k$ will be used to denote the k^{th} cubature rule point.

*It is assumed that $0 \in \mathbb{N}$. The space $\mathbb{N}_+$ is defined as $\mathbb{N} \setminus \{0\}$.

The *degree* of a monomial is defined as the 1-norm of $\boldsymbol{\alpha}$ (i.e., $\|\boldsymbol{\alpha}\|_1 = \sum_{k=1}^{d} \alpha_k$). The *total degree* of a monomial is defined as the ∞-norm of $\boldsymbol{\alpha}$ (i.e., $\|\boldsymbol{\alpha}\|_\infty = \max_{k=1}^{d} \alpha_k$).

Any multi-variate polynomial can be written as a linear combination of monomials. The (total) degree of a polynomial is the maximum of the (total) degree of its monomials. Because a polynomial is a finite linear combination, this number is also finite. The 0-polynomial has degree 0, so it is considered as a constant polynomial.

The space of d-variate polynomials of degree K has dimension $\binom{K+d}{d}$.

3.1.1 Cubature rules

The main focus is on the calculation of moments, so on the estimation of integrals. The integral that is being calculated is defined for $u \in L_1([0,1]^d)$ as follows:

$$\mathcal{I}u = \underbrace{\int_{[0,1]} \int_{[0,1]} \cdots \int_{[0,1]}}_{d \text{ integrals}} u(x^{(1)}, x^{(2)}, \ldots, x^{(d)}) \, \mathrm{d}P_1(x^{(1)}) \, \mathrm{d}P_2(x^{(2)}) \cdots \, \mathrm{d}P_d(x^{(d)}),$$

which will also be written in short as

$$\mathcal{I}u = \int_\Omega u(\mathbf{x}) \, \mathrm{d}P(\mathbf{x}) = \int_{[0,1]^d} u(\mathbf{x}) \, \mathrm{d}P(\mathbf{x}).$$

A clear distinction is made between $\mathscr{I}$ and $\mathcal{I}$, which are the one- and multi-dimensional integral respectively.

A cubature rule consists of cubature rule points $\{\mathbf{x}_k\}_{k=1}^N$ and weights $\{w_k\}_{k=1}^N$ to define the linear functional

$$\mathcal{Q}_N u := \sum_{k=1}^{N} u(\mathbf{x}_k) w_k,$$

where the same notational difference exists between $\mathscr{Q}_N$ and $\mathcal{Q}_N$.

The notion of degree can be extended to the multi-variate case.

Definition 13 (Degree of a cubature rule)**.** *Let $\mathcal{Q}_N$ be a d-dimensional cubature rule. Then this cubature rule is of degree K if for all $p \in \mathbb{P}(K,d)$, it holds that*

$$\mathcal{Q}_N p = \mathcal{I} p,$$

and $\exists p \in \mathbb{P}(K+1,d)$ such that

$$\mathcal{Q}_N p \neq \mathcal{I} p.$$

Due to the finite dimensionality of $\mathbb{P}(K,d)$ and the linearity of $\mathcal{Q}_N$ and $\mathcal{I}$, it is enough to prove the statement above for a basis of $\mathbb{P}(K,d)$ which can be all monomials of degree less or equal than K.

With the notion of the cubature rule, it is also possible to generalize the integration error for general functions. Here, again Taylor series are used.

Theorem 14. *Let $\mathcal{Q}_N$ be any d-dimensional cubature rule of degree K. Let $u(\mathbf{x}) \in C^{K+1}([0,1]^d)$ be a function defined on $[0,1]^d$. Then*

$$\mathcal{I}u - \mathcal{Q}_N u = \sum_{\|\beta\|_1 = K+1} \left(\mathcal{I}\left(\mathbf{x} \mapsto R_\beta(\mathbf{x}) \mathbf{x}^\beta\right) - \mathcal{Q}_N\left(\mathbf{x} \mapsto R_\beta(\mathbf{x}) \mathbf{x}^\beta\right) \right),$$

where R_β is an absolutely bounded function (i.e., $\|R_\beta\|_\infty \leq C_\beta$).

Proof. Rewrite u in its Taylor series using multi-index notation, so for any $\mathbf{y}$

$$\begin{aligned} u(\mathbf{x}) &= \sum_{\|\alpha\|_1 \geq 0} \frac{(\mathbf{x}-\mathbf{y})^\alpha}{\alpha!} (\partial^\alpha u)(\mathbf{y}) \\ &= \sum_{\|\alpha\|_1 \leq K} \frac{(\mathbf{x}-\mathbf{y})^\alpha}{\alpha!} (\partial^\alpha u)(\mathbf{y}) + \sum_{\|\beta\|_1 = K+1} R_\beta(\mathbf{x})(\mathbf{x}-\mathbf{y})^\beta, \end{aligned}$$

with

$$R_\beta(\mathbf{x}) = \frac{\|\beta\|_1}{\beta!} \int_0^1 (1-t)^{\|\beta\|_1 - 1} \partial^\beta u(\mathbf{y} + t(\mathbf{x}-\mathbf{y})) \, \mathrm{d}t.$$

Note that

$$\left|R_\beta(\mathbf{x})\right| \leq \frac{1}{\beta!} \max_{\|\alpha\|_1 = \|\beta\|_1} \max_{\mathbf{a} \in [0,1]^d} |\partial^\alpha u(\mathbf{a})| = C_\beta,$$

so R_β is an absolutely bounded function.

Substituting the Taylor series in the integral of u yields

$$\begin{aligned} \mathcal{I}u &= \mathcal{I}\left(\sum_{\|\alpha\|_1 \leq K} \frac{(\mathbf{x}-\mathbf{y})^\alpha}{\alpha!} (\partial^\alpha u)(\mathbf{y}) + \sum_{\|\beta\|_1 = K+1} R_\beta(\mathbf{x})(\mathbf{x}-\mathbf{y})^\beta \right) \\ &= \sum_{\|\alpha\|_1 \leq K} (\partial^\alpha u)(\mathbf{y}) \mathcal{I}\left(\frac{(\mathbf{x}-\mathbf{y})^\alpha}{\alpha!} \right) + \sum_{\|\beta\|_1 = K+1} \mathcal{I}\left(R_\beta(\mathbf{x})(\mathbf{x}-\mathbf{y})^\beta \right), \end{aligned}$$

where functions are functions of $\mathbf{x}$.

Substituting the Taylor series in the cubature rule yields

$$\begin{aligned} \mathcal{Q}_N u &= \mathcal{Q}_N \left(\sum_{\|\alpha\|_1 \leq K} \frac{(\mathbf{x}_k-\mathbf{y})^\alpha}{\alpha!} (\partial^\alpha u)(\mathbf{y}) + \sum_{\|\beta\|_1 = K+1} R_\beta(\mathbf{x}_k)(\mathbf{x}_k-\mathbf{y})^\beta \right) \\ &= \sum_{\|\alpha\|_1 \leq K} (\partial^\alpha u)(\mathbf{y}) \mathcal{I}\left(\frac{(\mathbf{x}-\mathbf{y})^\alpha}{\alpha!} \right) + \sum_{\|\beta\|_1 = K+1} \mathcal{Q}_N \left(R_\beta(\mathbf{x}_k)(\mathbf{x}_k-\mathbf{y})^\beta \right). \end{aligned}$$

Choosing $\mathbf{y} = 0$ and computing the difference yields the result. □

Due to this theorem, stability can be introduced in the same way as in the one-dimensional case. Also the other properties (like positive weights) can be extended to the current multi-dimensional setting.

3.1.2 Generalized Vandermonde-matrix

In the one-dimensional case, it was proved that any set of N distinct points can be used to generate a quadrature rule of degree $N-1$. This was proved using the system from (2.3), which contained the Vandermonde-matrix

$$\begin{pmatrix} x_1^0 & x_2^0 & \cdots & x_N^0 \\ x_1^1 & x_2^1 & \cdots & x_N^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{N-1} & x_2^{N-1} & \cdots & x_N^{N-1} \end{pmatrix},$$

which is non-singular if all the quadrature points are distinct.

In the multi-dimensional case, such a matrix also exists but can become singular even if all the points are distinct. In this section, first the Generalized Vandermonde-matrix is introduced and non-singularity is studied.

The matrix can be constructed straightforwardly. Suppose N cubature rule points are given. Then if N monomials $m_1(\mathbf{x}), m_2(\mathbf{x}), \ldots, m_N(\mathbf{x})$ are given, points $\{\mathbf{x}_k\}$ and weights $\{w_k\}$ of a cubature rule of degree high enough solve the following equations:

$$\sum_{k=1}^{N} m_j(\mathbf{x}_k) w_k = \mathcal{I} m_j, \quad \forall j = 1, \ldots, N,$$

which can be written as the linear system:

$$\begin{pmatrix} m_1(\mathbf{x}_1) & m_1(\mathbf{x}_2) & \ldots & m_1(\mathbf{x}_N) \\ m_2(\mathbf{x}_1) & m_2(\mathbf{x}_2) & \ldots & m_2(\mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ m_N(\mathbf{x}_1) & m_N(\mathbf{x}_2) & \ldots & m_N(\mathbf{x}_N) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} = \begin{pmatrix} \mathcal{I} m_1 \\ \mathcal{I} m_2 \\ \vdots \\ \mathcal{I} m_N \end{pmatrix}. \tag{3.1}$$

The Generalized Vandermonde-matrix (GV-matrix) is defined as the matrix in the system above. The "normal" Vandermonde matrix is now a special case of this matrix, with $m_k(x) = x^k$, which is obviously only defined in the one-dimensional case.

The GV-matrix is singular for specific choices of m_k and points. For example, let $d = 2$ with $\mathbf{x} = (x, y)$ and $m_k(\mathbf{x})$ be all monomials of degree 2 or less (so the monomials are 1, x, y, x^2, xy and y^2). Then choosing the quadrature points $\mathbf{x}_k$ to be 6 distinct points on the line $y = x$ yields that the second and the third row (monomials x and y) are equal, just as the fourth and the sixth row (monomials x^2 and y^2).

3.1.3 Tensor products

Probably the easiest cubature rule consists of taking the tensor product of d quadrature rules:

$$\mathcal{Q}_{N^d} = \bigotimes_{k=1}^{d} \mathcal{Q}_N u.$$

With the next two lemmas, it can be proved that the GV-matrix of this cubature rule is non-singular.

Lemma 15. *Let $A_1, A_2, \ldots, A_d$ be d non-singular square matrices. Then all these matrices are non-singular if and only if $\bigotimes_{k=1}^{d} A_k$ is non-singular.*

Proof. By induction, it suffices to proof that $A \otimes B$ is non-singular if and only if both A and B are non-singular. So let A and B be non-singular and let A be $N \times N$ and let B be $M \times M$. Then

$$\begin{aligned} A \otimes B &= (A \circ I_M) \otimes (I_N \circ B) \\ &= (A \otimes I_M) \circ (I_N \otimes B). \end{aligned}$$

By decomposing the M-dimensional space I_M spans, I_M can be written as $\bigoplus_{k=1}^{M} I_1$. Because the tensor product preserves direct sums, $A \otimes I_M$ can be written as $(A \otimes I_1) \oplus \cdots \oplus (A \otimes I_1) = A \oplus \cdots \oplus A$, hence $\det(A \otimes I_M) = \det(A)^M$. The same can be done for $I_N \circ B$, hence

$$\det(A \otimes B) = \det(A)^M \det(B)^N,$$

which is non-zero if and only if both $\det(A)$ and $\det(B)$ are non-zero. □

Lemma 16. *Let $A_1, A_2, \ldots, A_d \in \mathbb{R}^{N\times N}$ be d $N \times N$ non-singular matrices. Then*

$$\left(\bigotimes_{k=1}^{d} A_k\right)^{-1} = \bigotimes_{k=1}^{d} A_k^{-1}.$$

Proof. By the mixed-product property, it is true for tensors v, w, y, z that $(v \otimes w) \circ (y \otimes z) = (v \circ y) \otimes (w \circ z)$, so

$$\begin{aligned} I &= \bigotimes_{k=1}^{d} (A_k \circ A_k^{-1}) \\ &= \left(\bigotimes_{k=1}^{d} A_k\right) \circ \left(\bigotimes_{k=1}^{d} A_k^{-1}\right). \end{aligned}$$

□

Let V be the Vandermonde matrix of the one-dimensional quadrature rule. The GV-matrix of the tensor cubature rule is then equal to $\bigotimes_{k=1}^{d} V$ and because V is non-singular, $\bigotimes_{k=1}^{d} V$ is non-singular.

The relevance of the non-singularity is not discussed now, but this result will be used later when the reduced cubature rule is introduced.

3.2 Smolyak sparse cubature rule

The Smolyak sparse cubature rule (Smolyak, 1963) is a so-called sparse cubature rule using a sparse grid. Sparse grids are grids which are used to interpolate or integrate high-dimensional functions and often consist of (much) less points than normal tensor grids. The Smolyak cubature rule is a sparse grid technique that combines smaller tensor grids to form a sparse grid. The rule has been developed to interpolate robustly, but it turned out that the grid can also be used very well for numerical integration.

The setup of the method can be described in several ways (see Kaarnioja (2013) for a thorough survey). In this report it is based on the demand that the cubature rule should be of degree K. All proofs will be omitted, but references are provided.

Definition 17. *Let $\mathcal{Q}_{N_k}$ be an N_k-point cubature rule on the interval $[0,1]$ and assume that N_k is an increasing sequence for $k = 1, 2, \ldots$. Let $\Delta_k = \mathcal{Q}_{N_k} - \mathcal{Q}_{N_{k-1}}$ and $\Delta_1 = \mathcal{Q}_1$. Then*

$$\mathcal{S}_K = \sum_{\substack{\|\alpha\|_1 \leq K \\ \alpha \in \mathbb{N}^d}} \bigotimes_{k=1}^{d} \Delta_{\alpha_k} \tag{3.2}$$

is the Smolyak cubature rule.

In the definition above it is assumed that the same quadrature rule is used in each dimension. It is possible to extend the definition such that this is not required, but that is not relevant for the current story.

Remark 18. *Note that a tensor product rule could be written as*

$$\mathcal{Q}_K = \sum_{\substack{\|\alpha\|_\infty \leq K \\ \alpha \in \mathbb{N}^d}} \bigotimes_{k=1}^{d} \Delta_{\alpha_k},$$

such that the Smolyak cubature rule could be seen as demanding that all polynomials of degree K are integrated accurately.

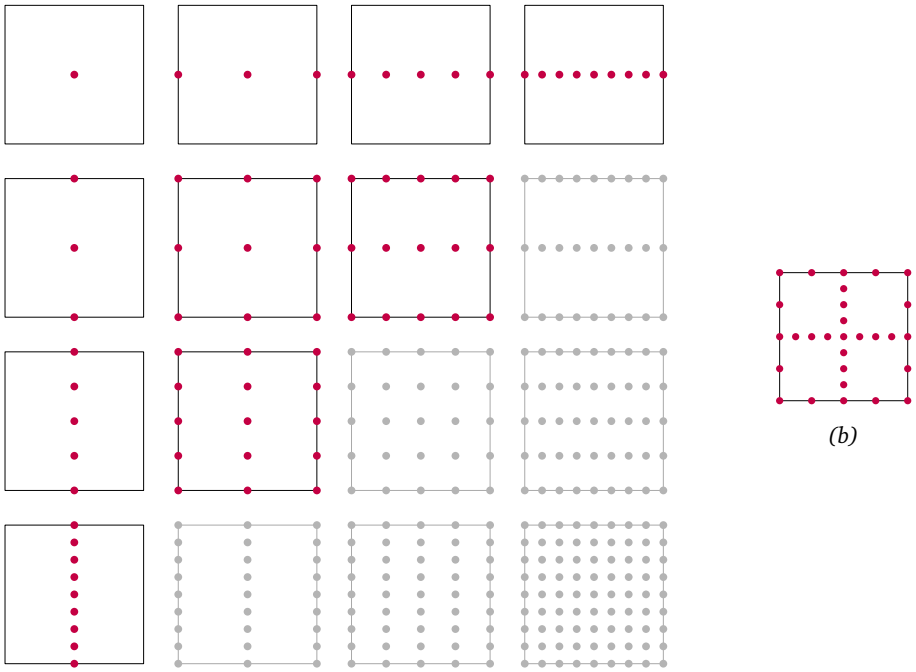


Figure 3.1: The creation of a Smolyak sparse grid. Combining the (dark) red cubature points of the left figure, which are essentially straightforward tensor products, yields the resulting mesh of the right figure. These are less points than the well-known tensor product (lower right of left figure) if the original quadrature rule is nested.

Note that the number of points is not known beforehand, which is why the parameter of $\mathcal{S}$ is changed into K.

Smolyak cubature rules can be generated for each number of dimensions. If the original quadrature rule (hence the tensor cubature rules) are nested, then less points are needed (see Figure 3.1 for a sketch).

By writing out the complete addition and subtraction of (3.2), the Smolyak cubature rule can be rewritten as (Wasilkowski and Wozniakowski, 1995)

$$\mathcal{S}_K = \sum_{\substack{K-d+1 \leq \|\alpha\|_1 \leq K \\ \alpha \in \mathbb{N}^d}} (-1)^{K-\|\alpha\|_1} \binom{d-1}{K-\|\alpha\|_1} \bigotimes_{k=1}^{d} \mathcal{Q}_{N_k}.$$

Visually, this can be seen as adding all grids on the anti-diagonals of Figure 3.1 together and repeatedly adding and subtracting the results. Each grid is then added or subtracted once.

Note that the nesting of the original quadrature rule is of importance, which will be shown using some examples. The Smolyak cubature rule does not have guaranteed positive weights if the quadrature rule has positive weights.

3.2.1 Error estimates

Novak and Ritter (1999) have proved some bounds for general multi-variate cubature rule formulas, comparable to the ones that have been deduced in the one-dimensional case.

Theorem 19 (Novak and Ritter, 1999). *Let $N_{min}(l,d)$ be the minimal amount of points that is necessary to create a cubature rule (not necessary a Smolyak cubature rule) of degree l. Then*

$$\dim \mathbb{P}(\lfloor l/2 \rfloor, d) \leq N_{min}(l,d) \leq \dim \mathbb{P}(l,d),$$

with $\dim \mathbb{P}(l,d) = \binom{l+d}{d}$.

Let $\{\mathcal{Q}_{N_k}\}$ be a family of quadrature rules of N_k points and of degree $N_k - 1$. Let N_k be such that

$$\begin{aligned} N_1 &= 1 \\ N_2 &\geq 3 \\ N_{i+1} - N_i &\geq N_i - N_{i-1}. \end{aligned} \tag{3.3}$$

Theorem 20 (Novak and Ritter, 1999). *Assume* (3.3) *and let* $N_0 = -1$. *Define* $l(N,d)$ *by*

$$l(N,d) = (N_{\sigma-1}+1)(d-(\tau+1)) + (N_\sigma+1)(\tau+1) - 1,$$

where

$$q = \sigma d + \tau$$

for some $\sigma \in \mathbb{N}$ *and* $\tau \in \{0,\ldots,d-1\}$. *Then* $\mathcal{S}_K$ *is of degree* $l(K,d)$.

Although this theorem is a sharp bound, the following result is more useful.

Corollary 21 (Novak and Ritter, 1999). $\mathcal{S}_K$ *has at least degree* $2(K-d)+1$.

3.2.2 Examples

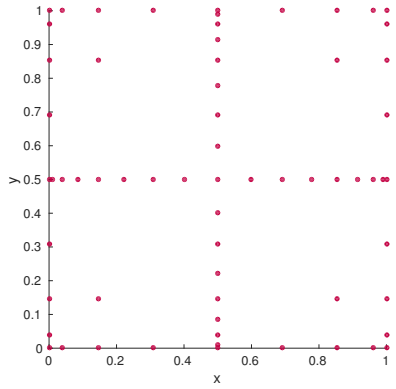
The Smolyak cubature rule adds smaller grids together to create one large grid. If the original quadrature rule is nested then the resulting Smolyak sparse grid has less points. Therefore, four cases are considered (see Figure 3.2):

- The Clenshaw–Curtis quadrature rule is nested and therefore the Smolyak sparse grid only contains 65 points (see Figure 3.2a);
- Gauss–Legendre quadrature rule is not nested and yields therefore many points in the Smolyak sparse grid (see Figure 3.2b);
- If a non-uniform distribution is considered, the Clenshaw–Curtis quadrature rule points remain fixed and a Gaussian quadrature rule does yield many points (see Figure 3.2d). Using a set of reduced Gauss–Jacobi quadrature rules does yield just 65 points and the grid is dependent on the distribution (see Figure 3.2c). The null vector of the reduced quadrature rules is chosen such that the likelihood of the quadrature rule is the largest.

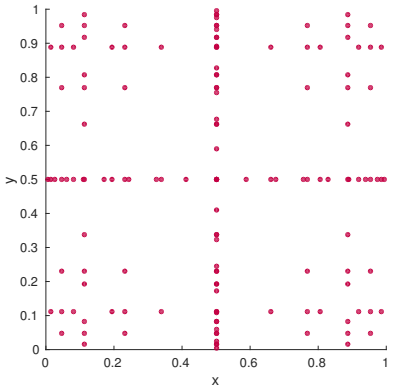
3.3 Reduced cubature rules

Recall reduced quadrature rules of Section 2.4. The same construction can be made in the current multi-dimensional case. However, the GV-matrix could be singular and by combining this with the fact that tensor products have many points with the same weights multiple points can be removed during each reduction step.

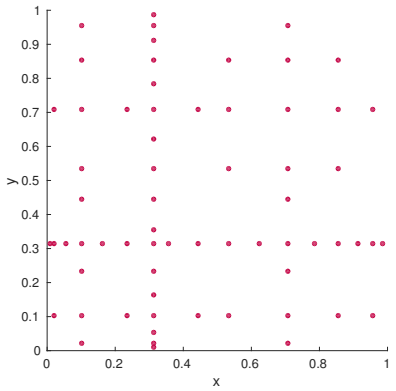
This section is built as follows. First, the principle of reduced quadrature rules is generalized to a multi-dimensional setting, which is just good bookkeeping. The requirement that the weights should



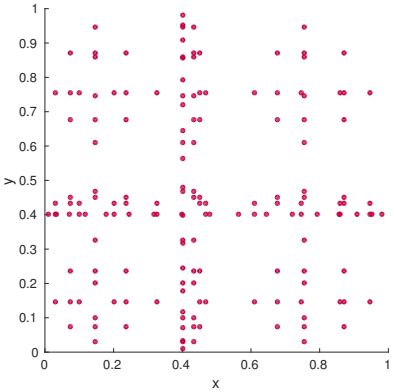
(a) Clenshaw–Curtis (65 points)



(b) Gauss–Legendre ($U(0,1)$, 142 points)



(c) Red. Gauss–Jacobi ($\beta(2,3)$, 65 points)



(d) Gauss–Jacobi ($\beta(2,3)$, 161 points)

Figure 3.2: Four Smolyak sparse grid of degree 9. Each figure is generated with a different quadrature rule.

remain positive is dropped. Then weight groups are introduced, which are essentially sets of points with the same weight. These sets will be useful when optimizing the cubature rule. It turns out that employing these groups results in a GV-matrix with a low rank, that can be used to remove a large number of points. Just like before, some examples of the new cubature rules are shown.

Note that in this section no implementation details are provided. The implementation of the algorithm is not trivial however, which will be discussed in the next chapter.

3.3.1 Carathéodory's theorem

Recall Carathéodory's theorem (see Theorem 10), which states that a linear combination of $d+1$ vectors with positive elements in a d-dimensional space can be rewritten as a combination of d vectors. This theorem was applied to the Vandermonde-matrix to construct a nested quadrature rule, which was called a reduction step. In the current case, the theorem can be applied to the GV-matrix to construct a nested cubature rule.

Let $\mathcal{Q}_{N^d}$ be an N^d-point tensor cubature rule of total degree $N-1$ with positive weights. Let $m_1, m_2, \ldots, m_{\binom{N+d}{d}}$ be a monomial basis of $\mathbb{P}(N,d)$. Then the first $\binom{N+d}{d}$ equations of (3.1) are given by

$$\begin{pmatrix} m_1(\mathbf{x}_1) & m_1(\mathbf{x}_2) & \ldots & m_1(\mathbf{x}_{N^d}) \\ m_2(\mathbf{x}_1) & m_2(\mathbf{x}_2) & \ldots & m_2(\mathbf{x}_{N^d}) \\ \vdots & \vdots & \ddots & \vdots \\ m_{\binom{N+d}{d}}(\mathbf{x}_1) & m_{\binom{N+d}{d}}(\mathbf{x}_2) & \ldots & m_{\binom{N+d}{d}}(\mathbf{x}_{N^d}) \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{\binom{N+d}{d}} \\ \vdots \\ w_{N^d} \end{pmatrix} = \begin{pmatrix} \mathcal{I}m_1 \\ \mathcal{I}m_2 \\ \vdots \\ \mathcal{I}m_{\binom{N+d}{d}} \end{pmatrix}.$$

The previous idea of Carathéodory's theorem is determining a null vector of the matrix above. In this case not one, but at least $N^d - \binom{N+d}{d}$ independent null vectors can be determined. Then repeatedly each null vector can be used to remove one point and weight of the cubature rule and hence one column of the matrix above until the matrix is square. Then the cubature rule has $\binom{N+d}{d}$ points and still has positive weights.

The construction can also be applied to GV-matrices of other cubature rules (e.g., the Smolyak cubature rule). This is however only a matter of bookkeeping, so details are not provided here.

3.3.2 Weight groups

In general, a tensor cubature rule can be constructed using a tensor product of many different quadrature rules. However, if the same quadrature rule is used several times, the resulting tensor cubature rule will have several points with the same weight. This will also happen if a symmetric quadrature rule is used. Because both situations happen often and because the proposed cubature rule can employ this, the construction is optimized in this case and it can be proved that indeed (possible many) more points can be removed.

The principle is as follows. If a null vector of the GV-matrix is such that if two weights are equal also the elements of the null vector are equal, then removing a weight with the null vector also immediately sets all the same weights to zero, without reducing the degree of the cubature rule. It is however not trivial to see that such a null vector exists and if it does, how it can be found. In this section an approach is outlined to determine such a null vector if it exists. In the next section, it is studied under what conditions such a null vector exists.

To determine the null vector, the notion of *weight groups* is introduced[†]. A weight group is a set of points that have the same weight. Each weight group has obviously one unique value of the weight of all points. All weight groups together form a partition of the set of cubature points.

The GV-matrix V of the tensor cubature rule is transformed into matrix A to incorporate for the weight groups. Matrix A has the following two defining properties:

1. Every null vector has a corresponds to a null vector of V (but not necessarily vice versa),
2. Using a null vector to remove points removes all points in one weight group immediately.

Constructing the matrix A can be done as follows. Let N_U be the number of unique weights (hence the number of weight groups) and let W_k be the k^{th} weight group. Then requiring that a null vector should remove all points of one weight group at the same time is requiring that all columns of those points have the same element of that null vector. This can be formalized by summing those columns and then determining a null vector. So consider the following columns:

$$\begin{pmatrix} m_1(\mathbf{x}_1)\mathbb{1}(\mathbf{x}_1 \in W_k) + m_1(\mathbf{x}_2)\mathbb{1}(\mathbf{x}_2 \in W_k) + \cdots + m_1(\mathbf{x}_{N^d})\mathbb{1}(\mathbf{x}_{N^d} \in W_k) \\ m_2(\mathbf{x}_1)\mathbb{1}(\mathbf{x}_1 \in W_k) + m_2(\mathbf{x}_2)\mathbb{1}(\mathbf{x}_2 \in W_k) + \cdots + m_2(\mathbf{x}_{N^d})\mathbb{1}(\mathbf{x}_{N^d} \in W_k) \\ \vdots \\ m_{\binom{N+d}{d}}(\mathbf{x}_1)\mathbb{1}(\mathbf{x}_1 \in W_k) + m_{\binom{N+d}{d}}(\mathbf{x}_2)\mathbb{1}(\mathbf{x}_2 \in W_k) + \cdots + m_{\binom{N+d}{d}}(\mathbf{x}_{N^d})\mathbb{1}(\mathbf{x}_{N^d} \in W_k) \end{pmatrix},$$

for $k = 1, \ldots, N_U$ and let A be the $\binom{N+d}{d} \times N_U$-matrix with these columns. A will be called the *reduced generalized Vandermonde-matrix* or just RGV-matrix. Then a null vector of the matrix A can be transformed in a null vector of V in a trivial way. Let P be an $N^d \times N_U$ binary matrix that maps a vector with the unique weights back to the weights themselves (i.e., it contains one 1 on each row and zeros elsewhere).

The following lemma is then trivial and could even be used to define A itself.

Lemma 22. *Let $\mathbf{a}_k$ be the k^{th} row of A and $\mathbf{v}_k$ be the k^{th} row of V. Then $\mathbf{a}_k^{\mathrm{T}} = P^{\mathrm{T}}\mathbf{v}_k^{\mathrm{T}}$.*

With this, it can actually be proved that the null vectors of A and V relate.

Theorem 23. *Let $\mathbf{c}_A$ be a null vector of the matrix A. Then $\mathbf{c}_V = P\mathbf{c}_A$ is a null vector of V.*

Proof. If $\mathbf{a}_1, \ldots, \mathbf{a}_{\binom{N+d}{d}}$ are the rows of A and $\mathbf{c}_A$ is a null vector of A, then $\mathbf{a}_k^{\mathrm{T}} \cdot \mathbf{c}_A = 0$ for all $k = 1, \ldots, \binom{N+d}{d}$. Replacing $\mathbf{a}_k^{\mathrm{T}}$ by $P^{\mathrm{T}}\mathbf{v}_k^{\mathrm{T}}$ yields $P^{\mathrm{T}}\mathbf{v}_k^{\mathrm{T}} \cdot \mathbf{c}_A = 0$. So finally $\mathbf{v}_k^{\mathrm{T}} \cdot P\mathbf{c}_A = \mathbf{v}_k^{\mathrm{T}} \cdot \mathbf{c}_V = 0$. □

Because the construction above plays a central role in the remainder of the report and the notation is cumbersome, a small example is discussed.

Example

In this section, the procedure described above is applied to a two-dimensional tensor cubature rule, based on two 3-point Clenshaw–Curtis quadrature rules. This quadrature rule of degree 3. Let $\{x_k\} = \{0, 0, 0, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1, 1, 1\}$, $\{y_k\} = \{0, \frac{1}{2}, 1, 0, \frac{1}{2}, 1, 0, \frac{1}{2}, 1\}$ and $\{w_k\} = \{\frac{1}{36}, \frac{1}{9}, \frac{1}{36}, \frac{1}{9}, \frac{4}{9}, \frac{1}{9}, \frac{1}{36}, \frac{1}{9}, \frac{1}{36}\}$ be the tensor cubature rule points and weights. Then

$$\mathbf{w} = \left(\frac{1}{36}, \frac{1}{9}, \frac{1}{36}, \frac{1}{9}, \frac{4}{9}, \frac{1}{9}, \frac{1}{36}, \frac{1}{9}, \frac{1}{36}\right)^{\mathrm{T}}$$

[†]A weight group has nothing to do with the mathematical algebraic structure from group theory. The name has been chosen because a weight group *groups* points with the same *weight* together. Note that a weight group contains points and not weights.

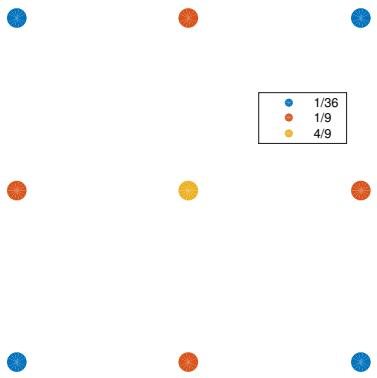


Figure 3.3: The partitions of a two-dimensional Clenshaw–Curtis tensor cubature rule (sketched without axis for clarity). There are three unique weights, so three unique partitions. The legend shows the weights of the points.

is the a vector with all the weights. There are three unique weights, which will be stored in the vector $\mathbf{w}_u$. Hence

$$\mathbf{w}_u = \left(\frac{1}{36}, \frac{1}{9}, \frac{4}{9}\right)^{\mathrm{T}}.$$

Partitioning $\{x_k\}$ yields three weight groups

$$\{x_k\} = \{0,0,1,1\} \cup \{0, \frac{1}{2}, \frac{1}{2}, 1\} \cup \{\frac{1}{2}\}$$

and partitioning $\{y_k\}$ becomes

$$\{y_k\} = \{0,1,0,1\} \cup \{\frac{1}{2}, 0, 1, \frac{1}{2}\} \cup \{\frac{1}{2}\}.$$

So there are three weight groups, containing either 4 points or 1 point.

Geometrically, the 4 points in the corners have the same weights, the 4 points on the axis have the same weight and the point in the middle has a unique weight (see Figure 3.3). Using these partitions, the columns of A can be constructed easily, so

$$A = \begin{pmatrix} 1+1+1+1 & 1+1+1+1 & 1 \\ x_1+x_3+x_7+x_9 & x_2+x_4+x_6+x_8 & x_5 \\ y_1+y_3+y_7+y_9 & y_2+y_4+y_6+y_8 & y_5 \\ \vdots & \vdots & \vdots \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 4 & 1 \\ 2 & 2 & \frac{1}{2} \\ 2 & 2 & \frac{1}{2} \\ \vdots & \vdots & \vdots \end{pmatrix}.$$

The second and third row are the same. Dependency of the rows is a key point to proving the existence of a non-trivial null space of A.

The null space of A consists of only one non-zero vector (or multiples of it):

$$\mathbf{c}_A = \left(\sqrt{21}, -2\sqrt{21}, 4\sqrt{21}\right)^{\mathrm{T}},$$

which can be transformed into the following vector for the null space of V (here the ordering of $\mathbf{w}_u$ is the same as the ordering of $\mathbf{w}$)

$$\mathbf{c}_V = \left(\sqrt{21}, -2\sqrt{21}, \sqrt{21}, -2\sqrt{21}, 4\sqrt{21}, -2\sqrt{21}, \sqrt{21}, -2\sqrt{21}, \sqrt{21}\right)^{\mathrm{T}}.$$

Now determining α as in the one-dimensional case or choosing to remove the largest number of points (it does not matter in this case), the cubature rule becomes $\{x_k\} = \{1, 1, 0, 0, \frac{1}{2}\}$, $\{y_k\} = \{1, 0, 1, 0, \frac{1}{2}\}$ and $\{w_k\} = \{\frac{1}{12}, \frac{1}{12}, \frac{1}{12}, \frac{1}{12}, \frac{2}{3}\}$. Geometrically, the red points have been removed (see Figure 3.3). This is a cubature rule of 5 points and of degree 3. It integrates all polynomials in a $\binom{3+2}{2} = 10$ dimensional space.

The number of weight groups

Introducing the notion of weight groups raises the question what the number of weight groups actually is. For certain general cases this number can be deduced exactly.

First, the case of a d-dimensional tensor cubature rule is studied, created with one N-point quadrature rule. This case is formalized using a theorem. The other cases studied are special cases of this theorem.

Theorem 24. *Let $\mathcal{Q}_{N^d}$ be a d-dimensional tensor cubature rule created with one N-point quadrature rule with N_U unique weights. Then the tensor cubature rule has at most $\binom{N_U-1+d}{d}$ unique weights.*

Proof. Number the N_U weights using the indices $1, \ldots, N_U$. Then the weights of the tensor product can be written as products of these weights, hence as sequences of these weights of length d. Two weights are equal if both sequences are built using the same numbers, so sorting the sequences yields the same sequence. The number of sorted sequences with these conditions equals $\binom{N_U-1+d}{d}$ (see Appendix A, Lemma 35 for a proof). Inequality happens if a product of multiple weights equals a product of multiple different weights. □

3.3.3 The row space of A

The matrix A will consist of more rows than columns in general. Because of this, the existence of a (non-trivial) null vector of A is not guaranteed. However, it turns out that if the same quadrature rule is used multiple times or if quadrature rules are symmetric, the matrix A has several dependent rows. This was also the case in the example of the previous section. If this fact is combined with the number of unique rows, it can be proved that a null vector exists.

If the requirement is dropped that the weights should remain positive, each null vector can hopefully be used to remove the weight group with the largest number of points.

These statements are formalized in several theorems. Because the proofs of these theorems are cumbersome, these are collected in Appendix A for sake of clarity.

Theorem 25 (Reduced tensor cubature rule of Gaussian quadrature rules)**.** *Let A be the $\binom{2N-1+d}{d} \times U$ RGV-matrix of a d-dimensional tensor cubature rule generated with N-point Gaussian quadrature rules (possibly created with different distributions). Then the dimension of the row space of A is bounded by $\binom{2N-1+d}{d} - N\binom{N+d-1}{d-1}$.*

Proof. See Appendix A, page 90. □

The dependent rows of the theorem above are all rows based on a monomial power $\boldsymbol{\alpha}$ with $N \leq \|\boldsymbol{\alpha}\|_\infty \leq 2N-1$.

Theorem 26 (Reduced tensor cubature rule of the same quadrature rule)**.** *Let A be the RGV-matrix of a d-dimensional tensor cubature rule generated by one N-point quadrature rule at least of degree $N-1$. Then the dimension of the row space of A is bounded by $1+\sum_{l=1}^{K} p(l,d)$, where $p(l,d)$ is the restricted partition number[‡] and K the degree of the cubature rule.*

Proof. See Appendix A, page 91. □

The theorem above can be used to prove that all rows based on a monomial power $\boldsymbol{\alpha}$ are equal to other rows which have a monomial power that is a permutation of $\boldsymbol{\alpha}$.

Theorem 27 (Reduced tensor cubature rule of symmetric quadrature rules)**.** *Let A be the RGV-matrix of a d-dimensional tensor cubature rule generated by d symmetric N-point quadrature rules at least of degree $N-1$. Then the dimension of the row space of A is bounded by $\binom{\lfloor \frac{N}{2} \rfloor + d}{d}$.*

Proof. See Appendix A, page 92. □

Combining all theorems yields the best results, i.e., determining the reduced cubature rule using weight grouping of a tensor cubature rule created with one symmetric N-point Gaussian quadrature rule. In Appendix B several cases are considered and the number of points is compared to results found in Novak and Ritter (1999) for the Smolyak cubature rule.

A lower bound to the number of null vectors is the difference between the number of columns and rows of the RGV-matrix. Therefore, quadrature rules that have many equal weights do not yield good results. It is well known that the Gauss–Chebyshev quadrature rule (i.e., $p(x) = \frac{1}{\sqrt{1-x^2}}$) yields the only Gaussian quadrature rule with equal weights.

3.3.4 Number of points

An upper bound of the number of points of a reduced tensor cubature rule is $\mathcal{O}(K^d)$, where K is the degree of the cubature rule and d is the dimension. This asymptotic upper bound cannot be improved, i.e., no cubature rule exists with an asymptotically smaller number of points (Hinrichs and Novak, 2007, Theorem 2).

An upper bound of the number of points of the Smolyak cubature rule is $\mathcal{O}(K^d(\log K)^{d-1})$ (Novak and Ritter, 1999). This upper bound equals $\mathcal{O}(K^{d+\varepsilon})$ for any $\varepsilon > 0$ and is theoretically worse than the upper bound of the tensor cubature rule. However, the constant hidden in the big-$\mathcal{O}$ notation is much smaller in the case of the Smolyak cubature rule than in the case of the tensor cubature rule. For practical situations (i.e., d and K are not too large) does the Smolyak cubature rule need less points than the tensor grid having the same degree.

A reduced tensor cubature rule still has the upper bound $\mathcal{O}(K^d)$, but lowers the hidden constant of the big-$\mathcal{O}$ notation because points are removed. The number of points of the reduced tensor cubature rule depends of the number of non-trivial null vectors of the RGV-matrix, which depends on the number of independent rows and columns of the RGV-matrix. The number of independent rows can be determined using Theorem 25, 26, and 27. No result has been found to determine the

[‡]There are several definitions of the restricted partition number. Here, it is the number of compositions of the number l with at most d summands.

number of independent columns of the RGV-matrix, so (sharp) estimations of the constant could not be found.

In practice it turns out that the constant of $\mathcal{O}(K^d)$ becomes so small that the number of points of the reduced tensor cubature rule is (in some cases much) smaller than the number of points of the Smolyak cubature rule. Because no stringent upper bound can be determined, the number of points has been calculated by determining the cubature rule itself and is compared with the Smolyak cubature rule. Results can be found in Appendix B.

Concluding, the number of points of the reduced tensor cubature rule will be less in cases were the same quadrature rule is used multiple times and symmetric quadrature rules are being used. The case of only different non-symmetric quadrature rules yields the trivial $\dim \mathbb{P}(K, d)$ upper bound, and therefore only yields less points than the Smolyak cubature rule for high K.

3.3.5 Initial grid

In the previous section two different reduction steps have been introduced. The first version, which was actually the extension of the one-dimensional version, removed the weights one by one and allowed to keep the weights positive. The second one introduced the principle of weight grouping and allowed to remove many points.

The question that remains is what initial grid produces the best results.

In the first case (i.e., without weight grouping) it is necessary that the cubature rule has positive weights. The procedure will only be able to remove points if the number of points of the initial cubature rule is larger than the dimension of the function space that is being considered. Tensor grids work well because of their nice properties and have more points than the dimension of the function space. The Smolyak cubature rule does however integrate all functions from a function space with a larger dimension than the number of points of the grid, such that a reduced Smolyak cubature rule is the same as a plain Smolyak cubature rule. However, this is only the case if nested quadrature rules are used.

If weight grouping is applied the initial grid needs to be a tensor grid, otherwise Theorems 26 and 27 cannot be used. If the tensor grid is created with Gaussian quadrature rules, it is also possible to apply Theorem 25 and the best results can be achieved. Moreover, symmetric quadrature rules of odd length will perform better than symmetric quadrature rules of even length. This is because an odd-length symmetric quadrature has more weight groups (two instead of one) and therefore yields an RGV-matrix of more columns, which yields a better result.

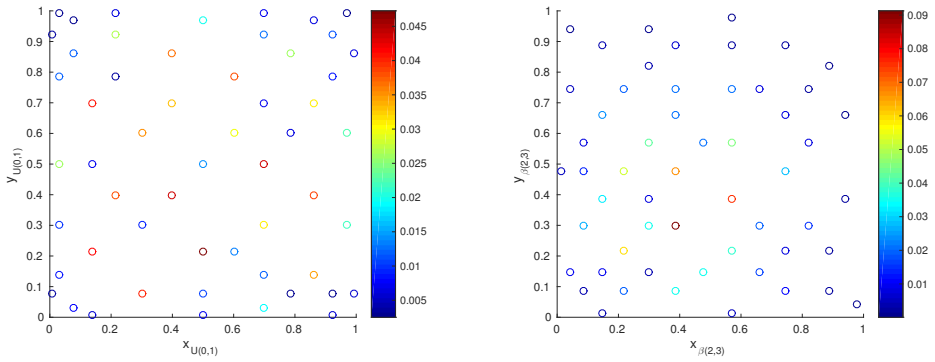
3.3.6 Examples

Two different versions of the reduced cubature rule have been introduced: the version with and without weight grouping. For both cases examples are stated.

Without weight grouping

Just as in the one-dimensional case there can exist multiple null vectors that can be used in each reduction step. The criteria discussed for the one-dimensional case can also be applied to the current case. However, requiring that the cubature rule remains symmetric is the same as requiring that one weight group should be removed. If also the criteria that did not yield good results are ignored (those were optimizing the interpolation error and mimicking a Gaussian quadrature rule) only two criteria are left: maximizing the likelihood and selecting the cubature rule with the largest maximum weight.

If the distribution is not constant, the likelihood of all cubature rule points can be calculated and the point can be removed that keeps the likelihood the largest. If this is not possible, the cubature rule can be chosen that has the smallest maximum weight (see Figure 3.4).



(a) $U(0,1)$ *(b)* $\beta(2,3)$

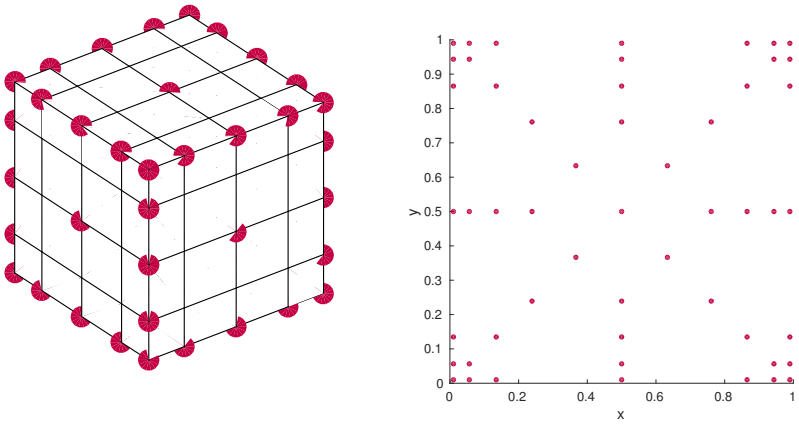
Figure 3.4: Reduced 15×15 *tensor cubature rule for two distributions: the uniform- and the* β*-distribution. Both are created with Gaussian quadrature rules and points were removed until 55 points (which is* $\dim \mathbb{P}(9,2)$*) were left such that the resulting cubature rule has degree 9.*

With weight grouping

The reduced tensor cubature rule with weight grouping works best for high-dimensional settings because the higher the dimension, the more points will have equal weights (see Appendix B for examples). However, a 10-dimensional cubature rule cannot be visualized easily, hence the current examples are two low-dimensional examples for the Gauss–Legendre quadrature rule (see Figure 3.5).

The two-dimensional grid is a reduced tensor cubature rule created with the 11-point Gauss–Legendre rule. The desired degree of the cubature rule was fixed at 11, such that points would be removed. The tensor cubature rule of the 6-point Gauss–Legendre rule yields less points, but is not interesting in this case because no points are being removed then.

The three-dimensional case already performs better than the plain tensor grid and Smolyak cubature rule for the current case. The reduced tensor cubature rule created with 5-point Gauss–Legendre quadrature rules consists of 77 points. The tensor cubature rule would need $5^3 = 125$ points for that and Smolyak cubature rule needs 177 points. This comparison is a bit unfair however, because the Smolyak cubature rule yields the best results for high dimensionality.



(a) 3D *(b) 2D*

Figure 3.5: Reduced tensor cubature rule with weight grouping in a three-dimensional and two-dimensional setting. Both were generated from Gauss–Legendre quadrature rules. The left one (the cube) also contains all the points inside the cube on the vertices (77 points in total) and is of degree 9. The right one (the grid) has 53 points and is of degree 11.

Chapter 4

Point-wise error estimates

Reconsider the problem of estimating the integral of an unknown function u using a cubature rule:

$$\int_\Omega u(\mathbf{x})\,\mathrm{d}\mathbf{x} \approx \sum_{k=1}^{N} u(\mathbf{x}_k) w_k.$$

In the previous chapters several strategies have been outlined to pick the cubature rule points such that the resulting cubature rule has a high degree. The numerical error made in the calculation of u was neglected. In this chapter, this assumption is dropped and the numerical error of u is incorporated into quadrature and cubature rules.

Incorporating the numerical error in the Monte Carlo method is a well-studied problem and mostly multiple levels are considered, each with its own numerical error. This is called the Multi-Level Monte Carlo (MLMC) method. Recently the approach of multiple levels as in the MLMC method has been applied to the Stochastic Collocation method to balance the error made by the cubature rule and the numerical method to estimate u (H. et al., 2013; Teckentrup et al., 2014). These methods are applicable when many estimations exist for the model that is being considered. The estimations required for the numerical models are generalized for many complicated models (Vasseur, 2010, considered the Navier–Stokes equations), but the estimations for the quadrature or cubature rule cannot be used in the current case because there is no bound of the number of points found.

Without the bound, it is possible to just use the estimates on the numerical model to improve the situation. This method will likely not be as good as the full multi-level SC method, but at least a small improvement should be noticed. In this chapter, such a method is proposed. The principle of the method is that multiple levels are not necessary and that the adaption is done *after* the calculation of the cubature rule points.

As in the previous chapters, $\mathcal{I}$ denotes the integral

$$\mathcal{I}u = \int_\Omega u(\mathbf{x})\,\mathrm{d}\mathbf{x},$$

which is estimated using an N-point cubature rule $\mathcal{Q}_N$ formed as follows:

$$\mathcal{Q}_N = \sum_{k=1}^{N} u(\mathbf{x}_k) w_k.$$

The main extension in this chapter is that u is estimated by a numerical model. Let $\hat{u}$ be an estimation of the real u. Let

$$\varepsilon = \|\hat{u} - u\|_2,$$

and assume that ε can be determined or somehow estimated as well.

The error in $\mathcal{Q}_N$ can then be determined as follows:

$$\begin{aligned}\|\mathcal{Q}_N\hat{u} - \mathcal{Q}_N u\|_2 &\leq \|\mathcal{Q}_N\|\|\hat{u} - u\|_2 \\ &\leq \|\mathcal{Q}_N\|\varepsilon.\end{aligned}$$

Recall that the norm $\|\mathcal{Q}_N\|$ is closely related to the stability of the cubature rule. This operator norm will play an essential role in the remainder of this chapter, which is built as follows. First, the global method is explained. Then the notion of computational cost is introduced, which will be optimized then. Finally, all formulas are put together in an example using estimates found in literature.

4.1 Cost minimization

Let $C(\mathbf{x}, \varepsilon)$ be the cost it takes to calculate $u(\mathbf{x})$ with accuracy ε. It is assumed that if ε decreases, the cost to reach that accuracy increases. Hence calculating a more accurate estimation of u also requires more computational cost. Mathematically this means that $\varepsilon \mapsto C(\mathbf{x}, \varepsilon)$ is a decreasing function for any $\mathbf{x}$.

Evaluating a cubature rule is calculating $u(\mathbf{x})$ multiple times, so the cost of evaluating an N-point cubature rule can be written as $\sum_{k=1}^{N} C(\mathbf{x}_k, \varepsilon_k)$. Here, ε_k is the error of the k^{th} summand.

The most trivial approach here could be to choose ε_k to be a fixed constant, without considering anything else. Implementations are easy then. Intuitively however, it would be more logical to deduce $u(\mathbf{x}_k)$ very accurately if its weight is very high and less accurately if its weight is small (i.e., close to zero). This is also the case and can be made more rigorous.

Therefore, let ε_b be the desired error of the cubature rule, so it is desired that $\|\mathcal{Q}_N\hat{u} - \mathcal{Q}_N u\|_2 \leq \varepsilon_b$. This however cannot be guaranteed, because u itself is unknown. However, it is true that

$$\begin{aligned}\|\mathcal{Q}_N\hat{u}\|_2 &= \left\| \sum_{k=1}^{N} \hat{u}(\mathbf{x}_k) w_k \right\|_2 \\ &\leq \sum_{k=1}^{N} \|\hat{u}(\mathbf{x}_k)\|_2 |w_k|,\end{aligned}$$

so

$$\|\mathcal{Q}_N\hat{u} - \mathcal{Q}_N u\|_2 \leq \sum_{k=1}^{N} |w_k|\varepsilon_k.$$

Because ε_k can be chosen, this is the quantity that is going to be optimized.

Putting everything together yields the following minimization problem

$$\arg\min_{\varepsilon_k} \left\{ \sum_{k=1}^{N} C(\mathbf{x}_k, \varepsilon_k) \mid \sum_{k=1}^{N} |w_k|\varepsilon_k \leq \varepsilon_b \right\},$$

where w_k and ε_b are given.

Fixing $\mathbf{x}_k$ and considering $C(\mathbf{x}_k, \varepsilon)$ as a function of ε, it is assumed that this function is invertible, decreasing, continuous, continuously differentiable and the derivative is also invertible*. These assumptions make the minimization problem above solvable and are not very stringent. To solve the problem, the method of Lagrange multipliers is used and the next lemma proves the last requirement of that method (the other requirements have been assumed).

*An example of a function that is invertible, decreasing, continuous and continuously differentiable is $y(x) = -x^3$ on the domain $[-1, 1]$. The derivative $y'(x) = 3x^2$ is however *not* invertible on this domain.

Lemma 28. *Let ε_k such that $\sum_{k=1}^N |w_k|\varepsilon_k < \varepsilon_b$. Then there exists an ε_k^* such that $\sum_{k=1}^N C(\mathbf{x}_k, \varepsilon_k^*) < \sum_{k=1}^N C(\mathbf{x}_k, \varepsilon_k)$ and $\sum_{k=1}^N |w_k|\varepsilon_k^* \leq \varepsilon_b$.*

Proof. Let $\varepsilon_1^* = \varepsilon_1 + \varepsilon_b - \sum |w_k|\varepsilon_k$. Then $\varepsilon_1^* > \varepsilon_1$, hence $C(\mathbf{x}_1, \varepsilon_1^*) < C(\mathbf{x}_1, \varepsilon_1)$ and $\sum_{k=1}^N |w_k|\varepsilon_k^* = \varepsilon_b$, so the solution is improved. □

Using this, the minimization problem can be rewritten as

$$\arg\min_{\varepsilon_k} \left\{ \sum_{k=1}^N C(\mathbf{x}_k, \varepsilon_k) \mid \sum_{k=1}^N |w_k|\varepsilon_k = \varepsilon_b \right\},$$

which can be solved using the method of Lagrange multipliers.

Lagrange multipliers can be used to solve a large set of problems. It can be formulated as follows. Let $\mathbf{p} \in \mathbb{R}^n$, then $\mathbf{p}$ maximizes a (possible multi-dimensional) function $f(\mathbf{p})$ with (possible multi-dimensional) constraint $g(\mathbf{p}) = 0$ if

$$\begin{cases} g(\mathbf{p}) = 0, \\ \nabla f(\mathbf{p}) - \lambda \nabla g(\mathbf{p}) = 0, \end{cases}$$

with $\lambda \in \mathbb{R}$ the so-called Lagrange multiplier. This procedure only works if f is continuous and continuously differentiable.

In the current case, $f(\boldsymbol{\varepsilon}) = -\sum_{k=1}^N C(\mathbf{x}_k, \varepsilon_k)$ is chosen, with $\boldsymbol{\varepsilon}$ the vector containing all ε_k. Let $g(\boldsymbol{\varepsilon}) = \sum_{k=1}^N |w_k|\varepsilon_k - \varepsilon_b$. Then replacing f and g in the equations above yields the system

$$\begin{cases} \displaystyle\sum_{k=1}^N |w_k|\varepsilon_k - \varepsilon_b = 0, \\ \displaystyle -\frac{\partial C}{\partial \varepsilon}(\mathbf{x}_j, \varepsilon_j) - \lambda |w_j| = 0 \quad \forall j = 1, \ldots, N, \end{cases}$$

which is a partially uncoupled system. The solution of this system can be written as

$$\varepsilon_k = \left(\frac{\partial C}{\partial \varepsilon} \right)^{-1} (\mathbf{x}_k, -\lambda |w_k|),$$

with λ such that

$$\sum_{k=1}^N |w_k|\varepsilon_k - \varepsilon_b = 0.$$

$\frac{\partial C}{\partial \varepsilon}$ should be considered as a function of ε. It is always true that $\lambda \geq 0$, because $\frac{\partial C}{\partial \varepsilon} \leq 0$.

Remark 29. *The bigger the quantity $\|\mathcal{Q}_N\| = \sum_{k=1}^N |w_k|$ is, the smaller ε_k must be to solve the minimization problem. So cubature rules with positive weights need less simulation time per point. However, the applications that are being considered do have long simulation times, such that minimizing the number of simulations is still more important than keeping the weights positive. It might be possible to balance $\sum_{k=1}^N |w_k|$ and the number of points such that less simulation time is needed, but this has not been studied.*

4.2 Lower bounds

In practice, it might be impossible to determine a numerical solution for the problem for all ε. The accuracy of a numerical solution depends on many factors (i.e., the mesh size, the numerical method, errors in boundary conditions or geometry, etc.) and the used computer also has a finite machine precision. If the approach above is applied to a cubature rule it is possible that the solution $\boldsymbol{\varepsilon}$ contains ε_k that cannot be realized by the numerical simulation. In this section the introduced approach is extended with lower bounds of ε_k.

To incorporate this into the described optimization procedure, the minimization problem is extended with an inequality:

$$\arg\min_{\varepsilon_k} \left\{ \sum_{k=1}^{N} C(\mathbf{x}_k, \varepsilon_k) \mid \sum_{k=1}^{N} |w_k| \varepsilon_k = \varepsilon_b, \varepsilon_k \geq \varepsilon_m \right\},$$

hence the Lagrange multiplier method must also be extended. Such a method exists and the conditions of that method are called the Karush–Kuhn–Tucker (KKT) conditions.

These conditions can be stated as follows. Let $\mathbf{p} \in \mathbb{R}^N$, then $\mathbf{p}$ maximizes a function $f(\mathbf{p})$ with constrains $g(\mathbf{p}) = 0$ and (possible multiple) $h_j(\mathbf{p}) \leq 0$, if

$$\begin{cases} \nabla f(\mathbf{p}) - \lambda \nabla g(\mathbf{p}) - \sum_{j=1}^{N} \mu_j \nabla h_j(\mathbf{p}) = 0, & \\ g(\mathbf{p}) = 0, & \\ h_j(\mathbf{p}) \leq 0 & \forall j = 1, \ldots, N, \\ \mu_j \geq 0 & \forall j = 1, \ldots, N, \\ \mu_j \cdot h_j(\mathbf{p}) = 0 & \forall j = 1, \ldots, N. \end{cases}$$

Geometrically, it can be seen as $N+1$ regions. One region has that $g(\mathbf{p}) = 0$ and another N regions have $h_j(\mathbf{p}) \leq 0$. The requirement $h_j(\mathbf{p}) \leq 0$ is encoded such that if the maximum of f has $h_j(\mathbf{p}) \leq 0$, h_j does not play a role so $\mu_j = 0$. If the maximum of f has $h_j(\mathbf{p}) > 0$, μ_j is non-zero and the solution of the minimization problem is on the boundary of the region with $h_j(\mathbf{p}) \leq 0$ (see Figure 4.1 for a sketch with one region). This is encoded in the last equation.

f and g are not altered and $h_j(\boldsymbol{\varepsilon}) = \varepsilon_m - \varepsilon_j$, such that $h_j(\boldsymbol{\varepsilon}) \leq 0$ implies $\varepsilon_m \leq \varepsilon_j$. Note that $\nabla h_j(\boldsymbol{\varepsilon}) = -\mathbf{e}_j$. Replacing f, g and h in the conditions above and writing the first equation coordinate-wise yields

$$\begin{cases} -\frac{\partial C}{\partial \varepsilon}(\mathbf{x}_j, \varepsilon_j) - \lambda |w_j| + \mu_j = 0 & \forall j = 1, \ldots, N, \\ \sum_{k=1}^{N} |w_k| \varepsilon_k - \varepsilon_b = 0, & \\ \varepsilon_m - \varepsilon_j \leq 0 & \forall j = 1, \ldots, N, \\ \mu_j \geq 0 & \forall j = 1, \ldots, N, \\ \mu_j (\varepsilon_m - \varepsilon_j) = 0 & \forall j = 1, \ldots, N. \end{cases}$$

In general the KKT conditions are necessary but not sufficient for the minimization problem, but if $\frac{\partial C}{\partial \varepsilon}$ is a smooth function and C is decreasing the conditions are sufficient. Details are not discussed.

Solving this set of equations is less trivial than solving the set of the Lagrange multiplier method. However, all equations except for the third ($\varepsilon_m - \varepsilon_j \leq 0$) can be solved using the Lagrange multiplier

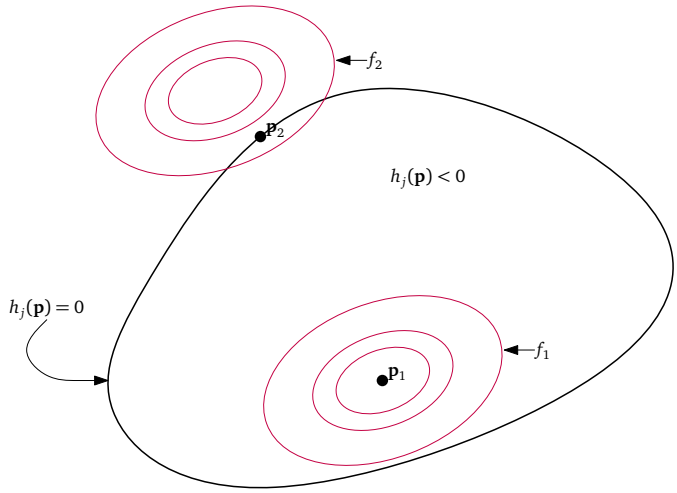


Figure 4.1: Either f_1 or f_2 is maximized (the red ellipses are contour lines) with $h_j(\mathbf{p}) \leq 0$ (the shape). If the maximum value of the function is in the region then $h_j(\mathbf{p}) < 0$ and $\mu_j = 0$, which is the case for f_1. If it is not in the region then $h_j(\mathbf{p}) = 0$ and $\mu_j > 0$, which is the case for f_2. So $\mu_j \cdot h_j = 0$.

method with $\mu_j \equiv 0$. Then considering this as a loop invariant, a simple algorithm can be stated that fulfills the KKT conditions (see Algorithm 2).

Algorithm 2 Determining ε_k using the KKT-conditions

Input: Weights $\{w_j\}_{j=1}^N$, upper bound ε_b, lower bound ε_m.
Output: $\{\varepsilon_j\}_{j=1}^N$ that minimizes the cost function as stated in the text.

1: Solve ε_j with the Lagrange multiplier method, let $\mu_j \equiv 0$.
2: **while** $\exists j : \varepsilon_m - \varepsilon_j > 0$ **do**
▷ *Loop invariant: all equations are fulfilled except $\varepsilon_m - \varepsilon_j \leq 0$*
3: Determine I and $\{j_k\}_{k \in I}$ with $\varepsilon_{j_k} < \varepsilon_m$
4: Set $\varepsilon_{j_k} = \varepsilon_m$ and $\mu_{j_k} = C'(\varepsilon_m) + \lambda |w_{j_k}|$ for all j_k.
▷ *Note that $\frac{\partial C}{\partial \varepsilon}(\mathbf{x}_{j_k}, \varepsilon_{j_k}) + \lambda |w_{j_k}| = 0$ and $\frac{\partial C}{\partial \varepsilon}(\mathbf{x}_{j_k}, \varepsilon_{j_k}) < \frac{\partial C}{\partial \varepsilon}(\mathbf{x}_{j_k}, \varepsilon_m)$ because $\varepsilon_{j_k} < \varepsilon_m$.*
5: Redetermine ε_i with $i \notin I$ such that $\sum_{k=1}^N |w_k| \varepsilon_k - \varepsilon_b = 0$.
6: **end while**

Note that it is not guaranteed that a solution exists, because the lower bound might be too stringent. The algorithm does not yield a correct solution then (and does not even quit). However, it can be checked easily whether a solution exists by considering ε_k as constants solving $\sum_{k=1}^N \varepsilon_k |w_k| = \varepsilon_b$ and checking whether they fulfill the lower bound.

Proving the correctness of the stated algorithm is easy and can be done formally using the loop invariant.

4.3 Example

The theory above can be applied to general numerical methods. In the field of Finite Element Methods it is typical that the error of the method is proportional to h^ρ, where h is the mesh size (lower is

better, typically measured by the size of one cell) and ρ related to the rate of convergence. So the cost function C of this method can be written as

$$C(\mathbf{x}, \varepsilon) = \varepsilon^\gamma,$$

with $\gamma = -\rho < 0$. Hereafter $\mathbf{x}$ is omitted because C is a function of ε only. Note that C does fulfill all requirements of Lagrange multiplier method. The KKT conditions are not applied, because it is non-trivial to apply these analytically.

Calculating the inverse of C' with $C'(\varepsilon) = \gamma\varepsilon^{\gamma-1}$ yields

$$(C')^{-1}(-\lambda|w_k|) = \left(\frac{-\lambda|w_k|}{\gamma}\right)^{\frac{1}{\gamma-1}} \overset{\lambda\geqq 0}{=} \lambda^{\frac{1}{\gamma-1}} \left(\frac{-|w_k|}{\gamma}\right)^{\frac{1}{\gamma-1}},$$

so $\varepsilon_j = K\left(\frac{-|w_j|}{\gamma}\right)^{\frac{1}{\gamma-1}}$ with K a scaling parameter such that $\sum_{k=1}^N |w_k|\varepsilon_k - \varepsilon_b = 0$, hence

$$K = \frac{\varepsilon_b}{\sum_{k=1}^N |w_k|(-|w_k|/\gamma)^{\frac{1}{\gamma-1}}}.$$

Comparing this approach to the approach with ε_k constant yields that the total cost C^* of the cubature rule in this case is (all sums are abbreviated for clarity, each sum is a sum from $k = 1$ to N)

$$\begin{aligned}
C_1^*(\varepsilon_b) &= \sum \varepsilon^\gamma \\
&= \sum K^\gamma \left(\frac{-|w_k|}{\gamma}\right)^{\frac{\gamma}{\gamma-1}} \\
&= \varepsilon_b^\gamma \frac{\sum(-|w_k|/\gamma)^{\frac{\gamma}{\gamma-1}}}{\left(\sum |w_k|(-|w_k|/\gamma)^{\frac{1}{\gamma-1}}\right)^\gamma} \\
&= \varepsilon_b^\gamma \frac{\sum |w_k|^{\frac{\gamma}{\gamma-1}}}{\left(\sum |w_k|^{\frac{\gamma}{\gamma-1}}\right)^\gamma} \\
&= \varepsilon_b^\gamma \left(\sum |w_k|^{\frac{\gamma}{\gamma-1}}\right)^{1-\gamma},
\end{aligned}$$

and if ε_k is constant (i.e., $\varepsilon_k = \frac{\varepsilon_b}{\sum|w_k|}$), then

$$\begin{aligned}
C_2^*(\varepsilon_b) &= \sum \frac{\varepsilon_b^\gamma}{\left(\sum |w_k|\right)^\gamma} \\
&= \varepsilon_b^\gamma \sum \left(\sum |w_k|\right)^{-\gamma} \\
&= \varepsilon_b^\gamma N \left(\sum |w_k|\right)^{-\gamma}.
\end{aligned}$$

Note that C_1^* is bounded from below because

$$\lim_{\gamma\to-\infty} \frac{C_1^*(\varepsilon_b)}{C_2^*(\varepsilon_b)} = \frac{1}{N}\left(\sum_{k=1}^N |w_k|\right)\left(\prod_{k=1}^N |w_k|^{-|w_k|}\right)^{\frac{1}{\sum_{k=1}^N |w_k|}},$$

which denotes the maximum possible improvement as percentage of the original cost.

Quadrature rule	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
3-point Clenshaw–Curtis	1	1	1	1
5-point Clenshaw–Curtis	0.93	0.92	0.92	0.92
7-point Clenshaw–Curtis	0.94	0.94	0.94	0.94
3-point Gauss–Legendre	1.11	1.15	1.17	1.18
5-point Gauss–Legendre	1.09	1.12	1.14	1.15
7-point Gauss–Legendre	1.08	1.10	1.12	1.13
3-point Gauss–Hermite	1	1	1	1
5-point Gauss–Hermite	0.80	0.78	0.77	0.77
7-point Gauss–Hermite	0.68	0.65	0.65	0.64

Table 4.1: The optimized cost per quadrature point scaled with the cost of the Clenshaw–Curtis quadrature rule. The error bound was $\varepsilon_b = 10^{-3}$.

4.3.1 Cubature rules

A cubature rule with weights that are all equal cannot be optimized using the method above, because the ε_k will also be constant. Cubature rules with varying weights can be improved better if the weights vary much, i.e., the largest absolute weight is much larger than the minimum absolute weight and many different values of the weights exists.

Three different cubature rules have been considered in this report: the tensor cubature rule, the Smolyak cubature rule and the reduced tensor cubature rule. For each cubature rule, the quantities above can be calculated and compared to see which cubature rule allows for the largest improvement and which cubature rule needs the least simulation time to reach a certain error.

Because $\sum_{k=1}^{N} w_k = 1$ for cubature rules with positive weights, a clear distinction between cubature rules with and without positive weights is made.

Positive weights

If a cubature rule has positive weights, $|w_k| = w_k$ and $\sum_{k=1}^{N} |w_k| = 1$. Therefore $C_2^*(\varepsilon_b) = \varepsilon_b^\gamma N$.

Let $C_1^*(\varepsilon_b)$ be the optimized cost of a quadrature rule with quadrature rule weights $\{w_k\}$. It is easy to see that the cost $\hat{C}_1^*(\varepsilon_b)$ of the d-dimensional tensor cubature rule created using this quadrature rule equals

$$\hat{C}_1^*(\varepsilon_b) = \left(C_1^*(\varepsilon_b)\right)^d.$$

The value $C_1^*(\varepsilon_b)$ can be calculated for various quadrature rules (see Table 4.1). Quadrature rules that have weights that do not vary much (such as the 7-point Gauss–Legendre, which has weights between 0.065 and 0.209) cannot be optimized as well as quadrature rules that have weights that do vary much (such as the 7-point Gauss–Hermite quadrature rule, which has weights between 0.005 and 0.457).

Reduced quadrature rules can have almost any value of C_1, depending on the initial quadrature rule. If the original quadrature rule consists of many points and a large number of points is removed, the quadrature rule can be optimized to have the smallest cost possible. Details are not discussed.

Cubature rule	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$
5-dim. Smolyak, deg. 3	1	1	1	1
5-dim. Smolyak, deg. 5	2.20	5.76	15.65	43.15
5-dim. Smolyak, deg. 7	4.07	19.69	98.28	495.73
5-dim. red. tensor, deg. 3	1.27	1.64	2.14	2.79
5-dim. red. tensor, deg. 5	0.95	1.57	2.94	5.77
5-dim. red. tensor, deg. 7	0.66	0.51	0.40	0.32
10-dim. red. tensor, deg. 3	2.53	7.04	20.42	60.33
10-dim. red. tensor, deg. 5	8.29	82.84	873.66	9413.73
10-dim. red. tensor, deg. 7	23.97	977.87	46639.36	2369496.58

Table 4.2: The optimized cost per cubature points scaled with the cost of the 5-dimensional Smolyak cubature rule of degree 3. All Smolyak cubature rules are created with Clenshaw–Curtis quadrature rules. The initial grid of the reduced tensor cubature rules are Gauss–Legendre tensor cubature rules.

Possible negative weights

The Smolyak cubature rule and reduced tensor cubature rules with weight grouping yield a small number of points but have negative weights. So the value of $C_2^*(\varepsilon_b)$ depends on the weights because $\sum |w_k| \neq \sum w_k$.

Again, the value $C_1^*(\varepsilon_b)$ can be calculated for various cubature rules (see Table 4.2).

The weights of the Smolyak cubature rule are relatively close to each other, e.g., the 5-dimensional Smolyak cubature rule of degree 7 has absolute weights between 0.004 and 0.50). The weights of the reduced tensor cubature rule with weight grouping are less predictable. The weights vary less in the 5-dimensional case, e.g., the reduced tensor cubature rule of degree 7 has absolute weights between 0.33 and 0.001. However, the weights vary a lot more in the 10-dimensional case. The reduced tensor cubature rule of degree 7 has absolute weights between 0.008 and 14.9. The larger γ is, the better that difference can be used.

Chapter 5

Numerical implementations

The algorithms and strategies which can be used to generate quadrature and cubature rules that have been discussed so far can perform theoretically very well. Implementing the algorithms on a computer with a finite precision arithmetic is not trivial, because many numerical instabilities might occur. An option would be to implement all algorithms symbolically, but the running time increases then. In this chapter it is discussed how quadrature and cubature rules can be constructed with a computer. Only the cases where the theoretical algorithms cannot be implemented straightforwardly are discussed.

First, the implementations of quadrature rules are discussed. Then the algorithms for the cubature rules are discussed. Reduced quadrature and cubature rules with and without weight grouping will play a central role in this discussion.

5.1 Quadrature rules

Calculating a Gaussian quadrature rule of N points could be done by starting with the basis $f_0 = 1$, $f_1 = x$, $f_2 = x^2$, etc., applying a Gram–Schmidt procedure to make determine orthogonal polynomials and then determine the zeros of f_N, where N is the number of points. This works quite well for small N, but determining all zeros of high degree polynomials is not a trivial task anymore. Moreover, this naive procedure does not provide a way to determine the weights accurately. Therefore, many algorithms have been developed to construct the polynomials efficiently and determine the points and weights as exactly as possible. The algorithm that is being discussed here is the algorithm of Golub (Golub and Welsch, 1969).

Generating the points and weights of the Clenshaw–Curtis quadrature rule can be done using Fourier transformations. Details are not provided here.

5.1.1 Algorithm of Golub

The main consideration of this algorithm is that the orthogonal polynomials $f_0, f_1, \ldots, f_N$ that are a basis of $\mathbb{P}(N)$ satisfy the three-term recurrence relationship

$$f_j(x) = (a_j x + b_j) f_{j-1}(x) - c_j f_{j-2}(x),$$

with $a_j > 0$, b_j and $c_j > 0$ constants and conditions $f_{-1} \equiv 0$ and $f_0 \equiv 1$. This relationship can be rewritten as the matrix equation

$$x\begin{pmatrix} f_0(x) \\ f_1(x) \\ \vdots \\ \vdots \\ f_{N-2}(x) \\ f_{N-1}(x) \end{pmatrix} = \begin{pmatrix} -b_1/a_1 & 1/a_1 & & & & \\ c_2/a_2 & -b_2/a_2 & 1/a_2 & & & \\ & c_3/a_3 & -b_3/a_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_N/a_N & -b_N/a_N \end{pmatrix}\begin{pmatrix} f_0(x) \\ f_1(x) \\ \vdots \\ \vdots \\ f_{N-2}(x) \\ f_{N-1}(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ f_N(x)/a_N \end{pmatrix}.$$

This system can be rewritten as $x\mathbf{f}(x) = T\mathbf{f}(x) + (1/a_N)f_N(x)\mathbf{e}_N$, where T is the matrix above. Hence $f_N(x) = 0$ if and only if $x\mathbf{f}(x) = T\mathbf{f}(x)$, which is an eigenvalue problem.

Now if T is not symmetric, it can be written as

$$DTD^{-1} = J = \begin{pmatrix} \alpha_1 & \beta_1 & & & & \\ \beta_1 & \alpha_2 & \beta_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & \beta_{N-2} & \alpha_{N-1} & \beta_{N-1} \\ & & & & \beta_{N-1} & \alpha_N \end{pmatrix},$$

where D is diagonal,

$$\alpha_i = -\frac{\beta_i}{\alpha_i},$$

and

$$\beta_i = \sqrt{\frac{c_{i+1}}{a_i a_{i+1}}}.$$

If T is symmetric nothing has to be done and $J = T$.

Golub and Welsch now note that it is true that

$$w_j(\mathbf{f}(x_k))^{\mathrm{T}}(\mathbf{f}(x_k)) = 1,$$

for an eigenvalue x_k of the matrix T. Now if x_k are calculated such that $J\mathbf{g}_k = x_k\mathbf{g}_k$ with $\|\mathbf{g}_k\| = 1$, then by this relationship $w_k = g_{k,1}/f_0^2(x_k)$, where $g_{k,1}$ is the first element of the eigenvector $\mathbf{g}_k$.

The coefficients a_j, b_j, and c_j are tabulated for many distributions. However, in the paper where the algorithm of Golub is explained, also a method is explained to determine these coefficients. The method will explained here briefly and requires the first $2N+1$ moments of the distribution, i.e., $\mathscr{I}x^k$ for $k = 0, 1, \ldots, 2N$. The first set is to construct M as the Hankel matrix of the list of these moments (if m is the list, then $M_{i,j} = m_{i+j-1}$). The Hankel matrix is positive definite, so applying a Cholesky decomposition yields $M = R^{\mathrm{T}}R$ with R upper diagonal. Then

$$\alpha_j = \frac{R_{j,j+1}}{R_{j,j}} - \frac{R_{j-1,j}}{R_{j-1,j-1}},$$

for $j = 1, 2, \ldots, N$, and

$$\beta_j = \frac{R_{j+1,j+1}}{R_{j,j}},$$

for $j = 1, 2, \ldots, N-1$ and $R_{0,0} = 1$ and $R_{0,1} = 0$.

Proving the correctness of this construction is not trivial and is outside of the scope of this report. The original paper also explains an efficient method that calculates the eigenvalues in this case using a modified QR decomposition (with time complexity $\mathcal{O}(N^2)$ instead of $\mathcal{O}(N^3)$), but due to the large computational power currently available this method is not necessary for the applications that are being considered in this report.

Note that the sequence $\left(\mathscr{I} x^k\right)_{k=1}^{\infty}$ decreases fast (almost exponentially) for well-known distributions. Because of this, it might be necessary to implement the algorithm of Golub using variable-precision arithmetic and then convert the matrix R back to finite precision before calculating the eigenvalues and eigenvectors*.

Extension to Gauss–Kronrod

The algorithm above can be extended to an algorithm to generate Gauss–Kronrod quadrature rules. This is done by extending the recurrence coefficients defined above and creating a $(2N+1)\times(2N+1)$-matrix. The Gauss–Kronrod quadrature rule has however not been applied in the current research and this extension is not trivial, so it is not discussed here. A complete overview and deduction of this extension, including pseudo code to generate the new matrix, can be found in Laurie (1997).

5.1.2 Reduced quadrature rules

Recall the proposed one-dimensional reduced quadrature rule. With that construction it is possible to generate a nested quadrature rule inside a bigger one, keeping the weights positive. Two steps of the algorithm are of interest. The first one is determining the null space of matrix V (which is an $(N-1) \times N$-matrix, where N is the number of points) and the second one is updating the weights using the null vector. The other steps of the algorithm can be implemented straightforwardly without introducing numerical instabilities.

Determining the null space

The null space can be determined in various ways. The biggest problem when determining the null space is that the elements of V are close to each other because all quadrature and cubature rules are defined on the interval $[0, 1]$. If a 20-point quadrature rule is considered, the matrix V is 19×20 and the last row consists of the monomial x^{19}, which becomes very small compared to 1 (which is x^0, the first row). For example, the middle point $\frac{1}{2}$ has $\left(\frac{1}{2}\right)^{20} \approx 9.5 \times 10^{-7}$. If the degree is increased even more, this number only gets smaller. Note that if the point $\frac{1}{8}$ is in a quadrature rule of degree 20, the last row already contains numbers that are smaller than a typical machine precision (which is approximately 10^{-17}).

The default way to numerically calculate the null space of a matrix is determing a singular value decomposition. A singular value decomposition calculates the null space of a matrix A by decomposing the matrix into $A = U\Sigma V^{\mathrm{T}}$, where U and V orthogonal and Σ a diagonal matrix with singular

*Mathematica can be used for this. In MATLAB the Symbolic Math Toolbox is required. The moments can be calculated symbolically using the function `sym` or with variable-precision arithmetic `vpa`. Note that the symbolic performance of MATLAB is less than that of Mathematica and calculating eigenvalues and eigenvector symbolically is a very computational expensive procedure for both software packages. It is advised to execute the procedure using variable-precision arithmetic such that the eigenvalues are not determined symbolically.

values on its diagonal. Zeroes will appear on the diagonal if A has a null vector and these elements can be used to construct the null vectors. It is known beforehand that at least one null vector exists, so numerical tolerances are not necessary.

This single null vector can also be determined using one QR factorization. A QR factorization is writing the matrix V as follows

$$V = \begin{pmatrix} Q_1 \mid Q_2 \end{pmatrix} \begin{pmatrix} R \\ 0 \end{pmatrix},$$

where Q_1 is orthonormal and Q_2 consists of orthonormal columns. R is an upper diagonal matrix. This method is implemented in many numerical software tool kits and can be calculated efficiently using Householder transformations, which are essentially linear reflections. The QR decomposition can then be determined by repeatedly reflecting the column basis of V until a orthonormal basis is recovered (which is Q_1). Continuing the process yields Q_2. To determine the null space of a matrix A, a QR factorization of A^{T} can be determined and Q_2 will then contain all null vectors.

Both methods are suitable to determine the null vector of the non-square matrix that is being considered and both result in a numerical error of the same order. However, determining a singular value decomposition using the numerical software used for this project (see Appendix D for a list of the used software) took much more time than determining a QR factorization, which is why the QR factorization has been selected as the preferred method in this case.

Updating the weights

If the null vector is determined, it can be used to remove a weight. For that, the following quantity has to be determined:

$$\alpha = \min_{k=1,\dots,N} \left\{ \frac{w_k}{c_k} : c_k > 0 \right\} =: \frac{w_{k_0}}{c_{k_0}}.$$

It is possible that $\mathbf{c}$ has elements very close to 0. If this is the case, w_k/c_k is a large number. To account for this, it is possible to determine α such that

$$\alpha = \min_{k=1,\dots,N} \left\{ \frac{w_k}{c_k} : c_k > \varepsilon w_k \right\} =: \frac{w_{k_0}}{c_{k_0}},$$

where ε is a small number, but large enough to make sure that α does not become very large. It might be possible that weights become negative after the reduction step, but if ε is small enough it should be possible to remove these weights without introducing a large error. Weights that are smaller than the numerical error introduced by QR decomposition can always be considered to be equal to 0.

5.2 Smolyak cubature rule

The best way to construct the Smolyak cubature rule is with the formula

$$\mathcal{S}_K = \sum_{\substack{K-d+1 \le \|\boldsymbol{\alpha}\|_1 \le K \\ \boldsymbol{\alpha} \in \mathbb{N}^d}} (-1)^{K-\|\boldsymbol{\alpha}\|_1} \binom{d-1}{K-\|\boldsymbol{\alpha}\|_1} \bigotimes_{k=1}^{d} \mathcal{Q}_{N_k},$$

i.e., by combining the red tensor grids from Figure 3.1 (see page 32) and adjusting the weights with the factor $(-1)^{K-\|\boldsymbol{\alpha}\|_1}$ as above.

Note that the implementation must keep track of points that are in two or more tensor grids if the algorithm is executed iteratively. A hash map, binary tree or "trie" can be used for this. Details are not discussed.

The generation of all sequences $\boldsymbol{\alpha}$ with $K-d+1 \leq \|\boldsymbol{\alpha}\|_1 \leq K$ might also be an issue. Because such algorithms are also necessary for the creation of reduced cubature rules, an efficient example is discussed. This problem is split in two. First, for each $k = K-d+1,\ldots,K$ all compositions of the number k are generated. These compositions are sequences $\boldsymbol{\alpha}$ with $\|\boldsymbol{\alpha}\|_1 = k$. Then all possible sequences $\boldsymbol{\alpha}$ can be generated by considering all permutations of these compositions. For both parts an efficient algorithm exists.

5.2.1 Compositions

Generating all compositions of a number n can be done using a straightforward recursive algorithm, but due to Kelleher a much more efficient way exists (Kelleher, 2005; Kelleher and O'Sullivan, 2014). The basic principle of the algorithm is by starting with the composition $1+1+1+\cdots+1$ and then keep adding the last numbers (see Algorithm 3).

Algorithm 3 Composition generation of the number n

Input: A number $n \in \mathbb{N}_+$.
Output: All compositions of the number n. Here, $1+1+1+\cdots+1$ is the first composition and n is the last.

```
1: Let a = {0, 0, ..., 0} of length n + 1, zero-based
2: Let k = 1 and a(1) = n.
3: while k ≠ 0 do
4:     Let x = a(k − 1) + 1
5:     Let y = a(k) − 1
6:     Let k = k − 1
7:     while x ≤ y do
8:         Let a(k) = x
9:         Let y = y − x
10:        Let k = k + 1
11:    end while
12:    Let a(k) = x + y
13:    Yield the numbers a(0, ..., k)
14: end while
```

In this algorithm the keyword *Yield* is used, which effectively means that at that point, the provided set of numbers can be processed. In this way, generating each composition from the previous one requires constant overhead. Almost all modern programming languages provide constructs which do this and this is mostly called a *generator*.

The algorithm can be improved even further by noting that many modifications to the current composition are done in the last two numbers, so adding a separate loop just to cover that case saves loop iterations of the while-loop in the current algorithm.

5.2.2 Permutations

Generating all permutations of a sequence $\mathbf{s} = (s_1, s_2, \ldots, s_n)$ of length n can be done in a straightforward way by recursively permuting the sequence. This, however, is a very slow approach and is not

efficient if several elements of $\mathbf{s}$ are the same. There is a more efficient algorithm due to Williams. The algorithm can iterate through all permutations with constant overhead (Williams, 2009).

The procedure is very short. Let $\mathbf{s}$ be the sequence as above. Let i be the length of the longest non-increasing sequence (i.e., $s_1 \geq s_2 \geq s_3 \geq \cdots \geq s_i < s_{i+1}$). Let $\sigma(\mathbf{s})$ be all permutations of $\mathbf{s}$. Then calculating the next permutation in $\sigma(\mathbf{s})$ can be done by calculating $\mathbf{t}$ as follows

$$\mathbf{t} = \begin{cases} (s_{i+1}, s_1, s_2, \ldots, s_i, s_{i+2}, \ldots, s_n) & \text{if } i \leq n-2 \text{ and } s_{i+2} > s_i, \\ (s_{i+2}, s_1, s_2, \ldots, s_{i+1}, s_{i+3}, \ldots, s_n) & \text{if } i \leq n-2 \text{ and } s_{i+2} \leq s_i, \\ (s_n, s_1, s_2, \ldots, s_{n-1}) & \text{otherwise,} \end{cases}$$

and i must be updated such that $i := 1$ if $t_1 < t_2$ and $i := i + 1$ otherwise (i.e., i is the length of the longest non-increasing sequence of $\mathbf{t}$).

5.3 Reduced cubature rules

Determining reduced cubature rules without weight grouping can be done in a similar way as in the one-dimensional case. The GV-matrix can be constructed efficiently using the algorithms of Williams and Kelleher and O'Sullivan.

If weight-grouping is used, typically a large number of points is removed. For example, calculating the reduced Gauss–Legendre tensor cubature rule of degree 13 yields only 38 353 points, although there are 282 475 249 tensor cubature points that need to be considered. If all these points have to be stored in the memory of a computer and each number would occupy one byte, this would add up to 2.6 GB. The cubature rule of 38 353 points would occupy 38 KB. Hence if high-dimensional cases are considered, using all tensor cubature points just as in the one-dimensional case is not possible and groups of points have to be considered, i.e., the RGV-matrix A should be created column-wise without creating each individual point.

In this section, two algorithms are proposed. The first algorithm is the most general algorithm that removes the points in a tensor grid group wise. The second algorithm is a special, improved version of this algorithm that can be used for RGV-matrices of very low rank (i.e., just one symmetric quadrature rule was used to create the tensor cubature rule).

The idea of the first algorithm is to construct A column-wise and if a non-square matrix is obtained then a null vector is calculated using QR factorization (see Algorithm 4). This idea works because A has in general many more columns than rows after all dependent rows are removed.

The problem left to solve is to construct the columns of A. The columns must be constructed such that the column representing the largest weight group is generated first and the column representing the smallest weight group is generated last. The number of weight groups and the number of points are typically large, such that generating all weight groups or constructing all points is not an option.

Therefore a new concept is introduced: weight-structure sets. Weight-structure sets are sets of weight groups and can be used to generate all weight groups with the same number of points, i.e., they contain weight groups that have the same structure.

5.3.1 Weight-structure set

Let a 4-dimensional tensor cubature rule created with one 5-point non-symmetric quadrature rule be given. Let the points (x_1, x_1, x_3, x_4) and (x_3, x_2, x_2, x_5) be given. It is easy to see that both points do not have equal weights. The first point has 11 other points (all permutations) with the same weight, which is the same as the second point. So both points are in a weight group consisting of 12 points, but those are not the same weight groups. Somehow, both weight groups do have the same structure but are not the same.

Algorithm 4 Calculating reduced tensor cubature rules

Input: Dimension d, the quadrature rules, their properties and in which dimension they should be used.

Output: $\{\mathbf{x}_k\}_{k=1}^N$ and $\{w_k\}_{k=1}^N$ that forms the reduced tensor cubature rule generated with the quadrature rules. Weight grouping is applied.

Let M be an empty matrix. Let $\mathbf{v}$ be an empty sequence.
for $i = 1, \ldots, N_U$, where N_U is the number of weight groups **do**
▷ *Iterate through the weight groups* descending *in size (i.e., the largest first).*
  Calculate $\mathbf{a}_i$, the i^{th} column of A and determine w_i, its weight.
  Append $\mathbf{a}_i$ to M.
  Append w_i to $\mathbf{v}$.
  if M has more columns than rows **then**
  ▷ *M has a null vector that can be calculated efficiently with QR factorization*
    Let $\mathbf{c}$ be the null vector of M. Note that $\mathbf{c}$ is also a null vector of A, when padded with 0's.
    Use $\mathbf{c}$ to remove the largest weight group and update $\mathbf{v}$ in the usual way.
    Remove all weights that equal zero from V and their respective columns from M.
  end if
end for
Construct $\{\mathbf{x}_k\}_{k=1}^N$ and $\{w_k\}_{k=1}^N$

A weight-structure set uses this principle and is defined as follows (see Figure 5.1 for a sketch).

Definition 30 (Weight-structure group)**.** *Two weight groups are in the same weight-structure set if and only if both weight groups contain the same number of points. All weight-structure sets partition the set of all weight groups.*

Assume a d-dimensional tensor cubature rule is generated with just one quadrature rule. Moreover, assume that there are not three quadrature rule points with the same weight. The key consideration is that, given a point and its weight, the number of points with that weight equals

$$\binom{d}{(u_1, u_2, \ldots, u_K)} 2^S.$$

Figure 5.1: A sketch of weight groups in weight-structure sets. Each weight-structure set only contains weight groups with the same number of points (visualized by the height of a weight group).

Here $(u_1, u_2, \ldots, u_K)$ is a sequence depending on the cubature rule. K is the number of unique elements in the tensor cubature rule point and u_k counts the k^{th} unique element in the tensor cubature rule point. S is the number of elements of the point that can be replaced without changing the weight. So the number of points in a weight group can be determined by counting the unique elements of the coordinate of a point and counting all elements that can be replaced. The first part incorporates for permutation and the second part incorporates for symmetric quadrature rules.

Generating all weight groups (hence all weights) can be done by creating all possible sequences $(u_1, u_2, \ldots, u_K)$ and all possible S and filling in points that fulfill that requirement. All permutations and possible replacements due to symmetric quadrature rules can then be used to iterate through the weight group.

If multiple different quadrature rules are used, the tensor cubature rule can be considered as the tensor product of smaller tensor cubature rules.

Because there are not many weight-structure sets (a few hundred for high-dimensional cases) all possible weight-structure sets can be investigated and sorted by the number of points of their weight groups. Then it is possible to iterate through the weight groups and points and generate all columns of A in the right order. The other steps of the algorithm are outlined in Algorithm 4.

5.3.2 Well-conditioned RGV-matrix

If the RGV-matrix of the tensor cubature rule has a very small number of rows, then it is possible to directly determine the new weights of the cubature rule without removing the points groupwise. Whether this is possible depends on the condition number of the RGV-matrix, which will increase rapidly if the degree of the cubature rule increases due to the structure of the RGV-matrix. This condition number is small if one symmetric quadrature rule is used multiple times and the degree of the cubature rule is smaller than 10.

To incorporate this in the algorithm, the columns of A must be determined ascending in size of their weight group, i.e., first the column with respect to the smallest weight group should be constructed and so on. When A becomes a square matrix a QR factorization can be performed to check whether A is non-singular and if that is the case the weights can be constructed directly. The biggest advantage of this algorithm is that not all weight-structure groups have to be considered anymore and only a few matrix operations are necessary.

The complete procedure is outlined in Algorithm 5. Although the description of the algorithm looks differently than the previous algorithm, it is easy to change an implementation of Algorithm 4 to an implementation of this algorithm by determining the weight-structure sets in the ascending order and implementing a small check if M is a square matrix.

5.3.3 Final remarks

If the next remarks are also taken into account the implementation can be made even faster.

- Theorem 25, 26, and 27 still apply and can be used to make the size of the relevant columns much shorter. It is advised to first generate all relevant monomial powers and save those such that each column can be constructed very efficiently. This is especially relevant for the second algorithm as the number of rows directly influences the result.

- If all symmetric quadrature rules are scaled such that they are symmetric around zero and Theorem 26 is implemented, then switching a quadrature rule point with the other one with the same weight does not change the column of the GV-matrix. So generating the column of A can be done by considering a much smaller number of points and then just multiplying with 2^S, where S is as above.

Algorithm 5 Direct construction of reduced tensor cubature rule

Input: Dimension d, the quadrature rules, their properties and in which dimension they should be used.

Assumption: the number of rows of the RGV-matrix is such that the linear system of Step 12 of this algorithm is solvable.

Output: $\{\mathbf{x}_k\}_{k=1}^N$ and $\{w_k\}_{k=1}^N$ that forms the reduced cubature rule of the tensor cubature rule generated with the quadrature rule. Weight grouping is applied.

1: Let M be an empty matrix. Let $\mathbf{v}$ be an empty sequence.
2: **for** $i = 1, \ldots, N_U$, where N_U is the number of weight groups **do**
▷ *Iterate through the weight groups* ascending *in size (i.e., the smallest first).*
3: Calculate $\mathbf{a}_i$, the i^{th} column of A
4: Append $\mathbf{a}_i$ to M.
5: **if** M is square **then**
6: Determine a QR factorization of M.
7: **if** M is singular **then**
8: Determine a null vector $\mathbf{c}$ of M.
▷ *Each column of M has one element in* $\mathbf{c}$
9: Remove the column of the largest weight group with non-zero element in $\mathbf{c}$.
10: **else**
11: Determine $\mathbf{m}$, the vector with moments of the distribution.
12: Solve $M\mathbf{w} = \mathbf{m}$.
13: *Go to* 17.
14: **end if**
15: **end if**
16: **end for**
17: Construct $\{\mathbf{x}_k\}_{k=1}^N$ ($\{w_k\}$ is already constructed during Step 12)

- Scaling each column of A such that its norm equals 1 might make the QR factorization much more stable. Remember to scale the null vector back using the scaling. It depends on the specifics of the quadrature rule whether this improves the situation.
- Reconstructing the cubature rule in the end from the columns and weights that are left can be done using the same loops that are used constructing the matrix. The same efficient algorithms apply.
- The sum of the weights should always be equal to the size of the integration volume (typically 1). So during the algorithm, the sum should remain less than the volume (because only a part of the weight is being considered), but larger than 0.

Combining all these optimizations allows one to generate up to 25-dimensional cubature rules efficiently[†].

[†]At the time of writing, a 25-dimensional reduced Gauss–Legendre tensor cubature rule of degree 9 can be generated in less than 20 seconds on an average desktop computer.

Chapter 6

Results and Applications

In this chapter the proposed cubature rule is applied to multiple problems and compared with the tensor cubature rule and the Smolyak sparse cubature rule. The first problem considered will be about integration, without considering UQ. The other two problems will be quantifying uncertainties of a computational fluid dynamics problem.

This chapter is built as follows.

In the first section the cubature rules will be used to integrate the Genz test functions. These functions are test functions with hard properties that can be used to test for convergence and accuracy of integration methods.

The second and third section will contain applications of the cubature rule to two UQ cases. In the second section the lid-driven cavity flow problem with uncertain fluid properties will be studied. In the last section the main advantage of the method is shown, i.e., high accuracy for a moderately high-dimensional problem. A three-dimensional utility aircraft is considered with seven uncertain parameters. Because the conventional methods require a large number of simulations, only the results of the new reduced cubature rule are discussed.

6.1 Genz test functions

To test the quality of a cubature rule, several functions have been developed by Genz (Genz, 1984). Each function has a different specific property or attribute, of which the effect can be enlarged by a parameter **a**. A shape parameter **u** can be used to transform the function without changing the property (see Table 6.1 for all functions and their relevant attributes).

Reducing a cubature rule optimizes for polynomial accuracy, i.e., the higher the degree the better. Hence if the cubature rule is applied to a function that cannot be expressed into polynomials, it will not work well. This can also be seen in the results that have been generated using the test functions.

Unfortunately, it was impossible to determine the exact value of the integral of f_3, so the results of that function are omitted (see Appendix C for details about the calculations).

6.1.1 Family attribute

A way to measure the influence of the attribute of the test functions, is by choosing the parameters **a** and **u**. Two random vectors **a** and **u** are created, that are used to calculate the exact and the approximated value of the integral. Because **a** and **u** are chosen randomly, this is done 100 times to average the result. All results were created using 5-dimensional input spaces.

The integral is estimated using cubature rules of varying degrees (see Figure 6.1).

Integrand Family	Attribute
$f_1(\mathbf{x}) = \cos\left(2\pi u_1 + \sum_{i=1}^{n} a_i x_i\right)$	Oscillatory
$f_2(\mathbf{x}) = \prod_{i=1}^{n} \left(a_i^{-2} + (x_i - u_i)^2\right)^{-1}$	Product Peak
$f_3(\mathbf{x}) = \left(1 + \sum_{i=1}^{n} a_i x_i\right)^{-(n+1)}$	Corner Peak
$f_4(\mathbf{x}) = \exp\left(-\sum_{i=1}^{n} a_i^2 (x_i - u_i)^2\right)$	Gaussian
$f_5(\mathbf{x}) = \exp\left(-\sum_{i=1}^{n} a_i \lvert x_i - u_i\rvert\right)$	C_0 function
$f_6(\mathbf{x}) = \begin{cases} 0 & \text{if } x_1 > u_1 \text{ or } x_2 > u_2 \\ \exp\left(\sum_{i=1}^{n} a_i x_i\right) & \text{otherwise} \end{cases}$	Discontinuous

Table 6.1: The test functions from Genz (Genz, 1984). All functions are from a certain integrand family and depend on the parameters **a** *and* **u**. *The parameter* **u** *is a parameter that does not affect the difficulty of the integral. The parameter* **a** *determines the degree to which the family attribute is present.*

The results for f_1, f_2, and f_4 are pretty good. A reduced tensor cubature rule compares very well with Smolyak cubature rule, and for small c all cubature rules can integrate the test function very accurately. This is because those functions can be expressed in a Taylor series. Note that the Taylor series of f_1 converges uniformly, such that estimating the integral with a tensor cubature rule works very well.

f_5 and f_6 cannot be expressed in a Taylor series because they are not differentiable. Therefore does the reduced tensor cubature rule due to its small number of points perform very bad compared to both the tensor cubature rule and the Smolyak cubature rule. For f_6 the result is the worst of all cubature rules, because of the discontinuity.

6.1.2 Accuracy

Although a reduced tensor cubature rule compares very well to the Smolyak cubature rule in the previous case, this is not important in practice. If the cubature rule is applied to UQ, the number of simulations required to reach a certain accuracy needs to be as small as possible. Therefore, the integration error is determined using all cubature rules of varying degrees. **a** and **u** are again chosen randomly, with $\|\mathbf{a}\| = 2\frac{1}{2}$. Not only can these results be used to check which cubature rule needs the least points for a certain accuracy, but they also show whether the cubature rules are stable and converge (see Figure 6.2).

Again, the results for f_1, f_2, and f_4 are the best. Note that the seemingly bad result of f_1 for high numbers of points is a result of numerical cancellation. The reduced tensor cubature rule needs a small fraction of the points of the Smolyak cubature rule.

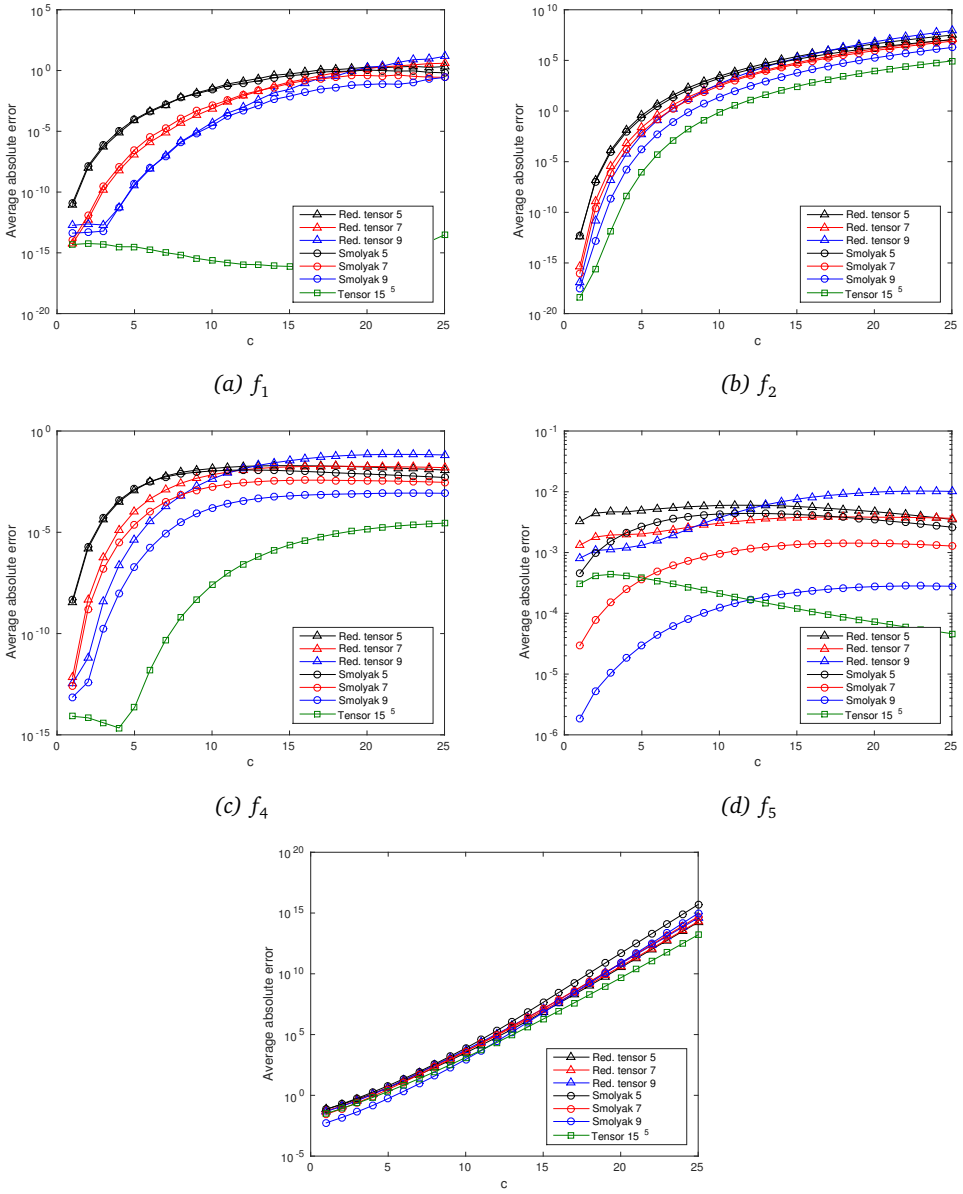
The results of f_5 and f_6 are bad if the reduced tensor cubature rule is applied. Because it is optimized for polynomial accuracy and those functions cannot be expressed in polynomials, the convergence is bad for f_5 and it looks like there is not even convergence for f_6.

6.1.3 Curse of Dimensionality

If the cubature rules are applied to UQ, the integrals considered will typically be high-dimensional. The so-called curse of dimensionality prescribes that the number of points increases rapidly as the dimension increases.

To test which cubature rules needs the highest number of points in high-dimensional cases, the number of points is determined that is needed to integrate the test functions with an absolute error of 10^{-8} (see Figure 6.3). The results of f_5 and f_6 are omitted because the cubature rules do not converge.

The reduced cubature rule (with weight grouping) needs the same number of points as the tensor cubature rule in low-dimensional cases because no points can be removed. However, for high dimensional cases (i.e., $d \geq 3$) points are removed and less points are needed than both the Smolyak cubature rule and the tensor cubature rule. The difference in the number of points in high-dimensional cases becomes smaller because all cubature rules do have the same asymptotic growth of the number of points.

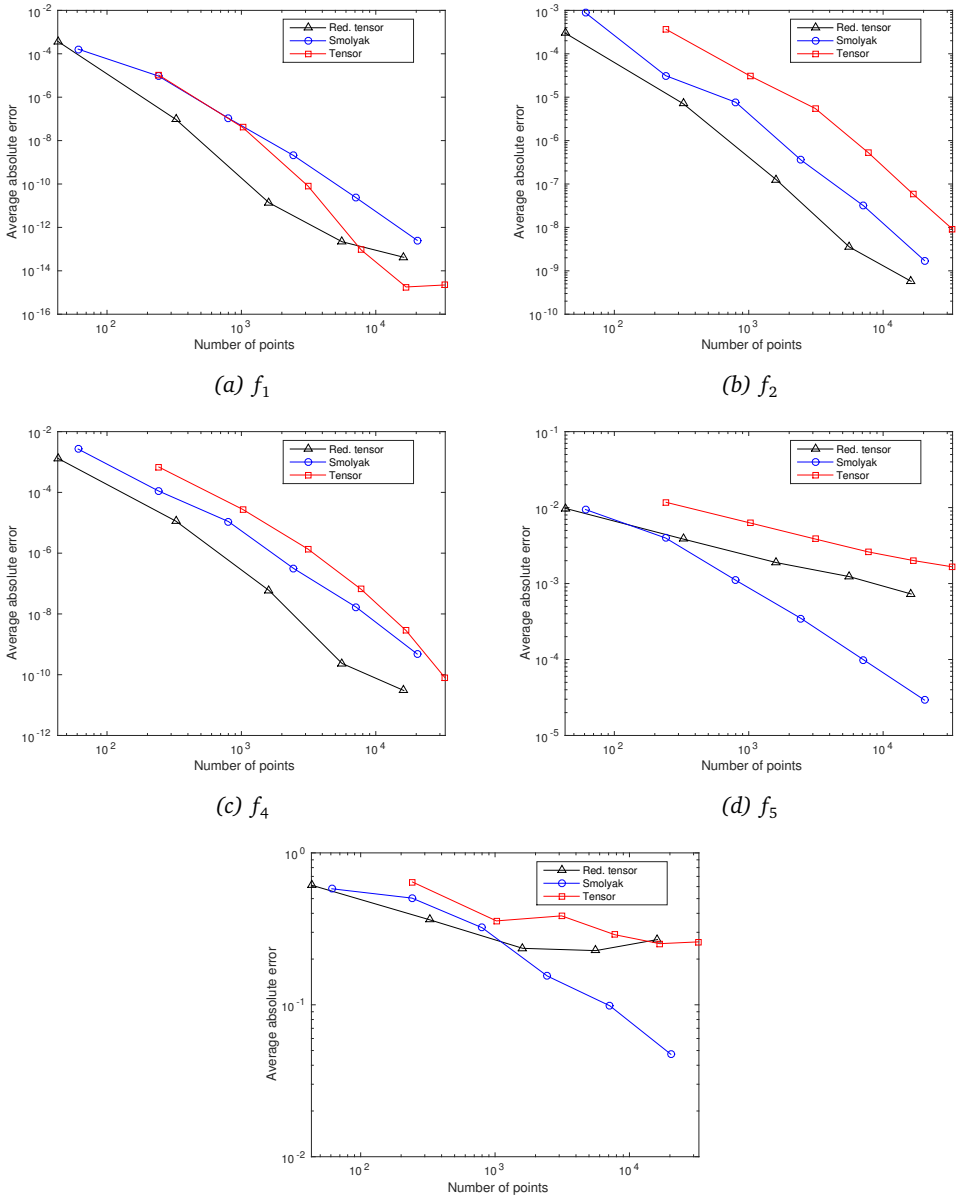


(a) f_1 (b) f_2

(c) f_4 (d) f_5

(e) f_6

Figure 6.1: The average absolute error versus the norm of the shape vector **a** *of the Genz test functions (denoted as c). The Smolyak cubature rule was chosen such that its polynomial degree matched the degree of the reduced tensor cubature rule.*

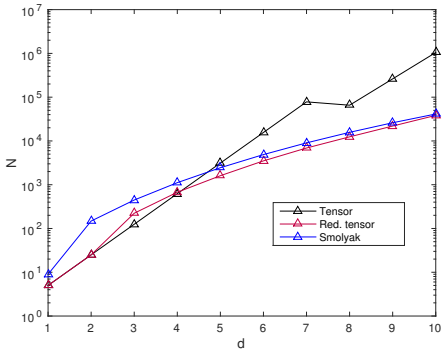


(a) f_1 (b) f_2

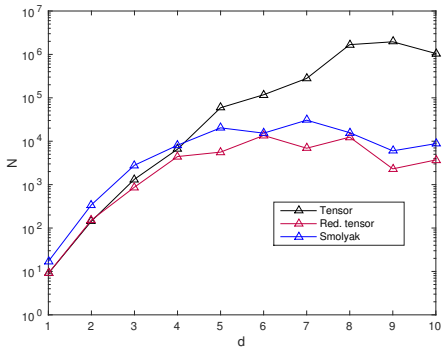
(c) f_4 (d) f_5

(e) f_6

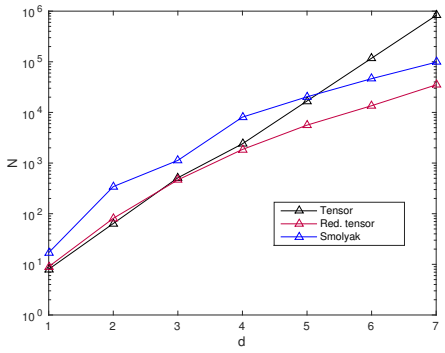
Figure 6.2: The accuracy of several cubature rules versus the number of points that are in the cubature rule. All integrals are 5-dimensional.



(a) f_1



(b) f_2



(c) f_4

Figure 6.3: The number of points needed to reach an absolute integration error of 10^{-8} versus the dimension. The result of f_5 and f_6 did not converge.

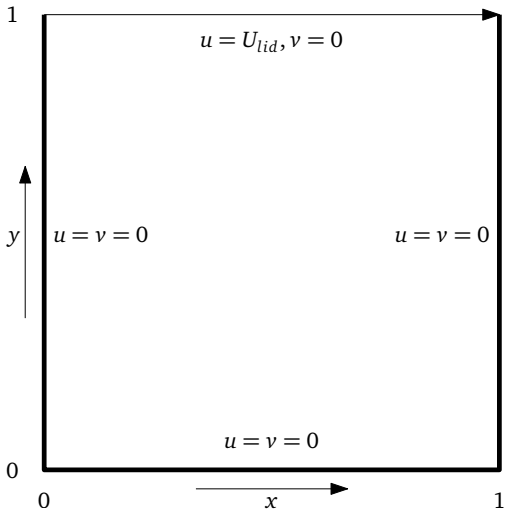


Figure 6.4: The geometry and boundary conditions of the lid-driven cavity flow.

6.2 Lid-driven cavity flow

The lid-driven cavity flow is a well-known CFD test problem, of which much test data is available. One simulation does not take much time, so three methods can be compared.

This section is built as follows. First, the problem is described including the uncertain parameters. To solve the problem, the Lattice Boltzmann method has been implemented. Then three UQ methods are compared, namely the straightforward Monte Carlo method, Stochastic Collocation with a Smolyak Sparse grid and Stochastic Collocation with the reduced tensor cubature rule.

6.2.1 Problem description

The lid-driven cavity flow is a 2D CFD problem where a certain incompressible fluid is contained in a square box with sides of unit length. On top of the box a lid is placed which is moved to the right with unit velocity. This velocity is a boundary condition (see Figure 6.4 for a sketch of the geometry and Figure 6.6 for a sample solution). The boundary condition at the corners (i.e., at $(0, 1)$ and $(1, 1)$) is $u = v = 0$.

The deterministic problem is solved using the Lattice Boltzmann method. This method is fairly new and does not solve the flow equations (i.e., the Navier–Stokes equations) directly, but instead simulates the Boltzmann equations using a distribution function defined on a grid. It can be seen as a hybrid method that solves a flow equation and also simulates a particle model. The implementation used for the current case is a straightforward D2Q9 BGK-model using Zou–He boundary conditions (Zou and He, 1997).

Reference data for several values of the Reynolds number can be found in literature (Ghia et al., 1982). The results from the implementation compare well with the data provided (see Figure 6.5).

Uncertainties

The current problem is adapted to the UQ setting by imposing two uncertainties on the flow parameters. The uncertain parameters including their distributions are given below.

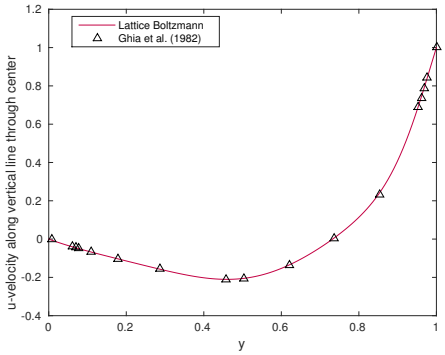


Figure 6.5: The u-component of the flow velocity along the vertical line through the geometrical center of the cavity at $Re = 100$.

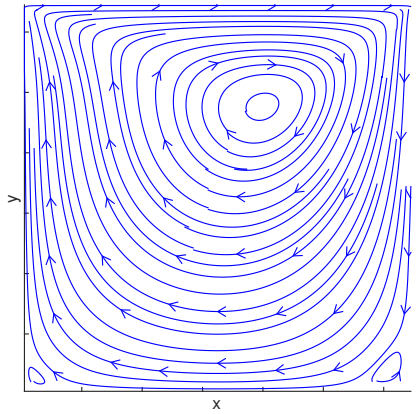


Figure 6.6: Streamlines for the lid-driven cavity flow problem at $Re = 100$.

Property	Distribution
ν (viscosity)	$\beta(2,3)$ with range $(0.05, 1.05)$
u_{lid} (speed of the lid)	$\mathcal{N}$ with mean 1 and standard deviation 0.1

6.2.2 Monte Carlo method

To make a good comparison between the three methods considered, the Monte Carlo method is applied to get a good solution. To determine the number of samples needed, the calculations are done twice, once with 50 and once with 100 samples. The differences are negligible, hence 50 samples are enough.

Calculating the mean with a Monte Carlo method is trivial, but calculating the standard deviation is less trivial because numerical errors may occur if the naive formula is used. Therefore, the so-called "online" algorithm is used. Let the mean μ and variance ν be equal to zero. Then for each calculated sample $\mathbf{x}_k$ for $k = 1, \ldots, 50$, the mean and variance are updated as follows:

$$\mu = \mu + \frac{\delta}{k},$$
$$\nu = \nu + \delta(\mathbf{x}_k - \mu),$$

with $\delta = \mathbf{x}_k - \mu$ calculated a priori each step and the mean calculated before the variance. After all samples have been processed, $\frac{\nu}{49}$ is the sample variance (and the square root is the standard deviation).

6.2.3 Stochastic Collocation

Stochastic Collocation is applied with two different cubature rules.

The first one is the Smolyak sparse cubature rule generated with one-dimensional reduced Gaussian quadrature rules.

The second one is the reduced Gaussian tensor cubature rule. Calculating a reduced tensor cubature rule yields just a tensor cubature rule created with two Gaussian quadrature rules, because it is a two-dimensional problem. Therefore, a tensor grid with twice the number of points is used as initial grid to test the removal of the points. This yields more points but, as a consequence, allows for a better comparison between the methods.

To be able to compare the methods visually, the u-component of the flow velocity is used at $x = \frac{1}{2}$ (just as in Ghia et al. (1982)). The results for the three methods compare very well (see Figure 6.7).

To test the convergence of the method, the norm of the difference of two consecutive grids was taken and plotted against the number of points (see Figure 6.8). Both the mean and the variance of the components of the flow and the pressure converge nicely. Note that here the samples of the largest set were reused, such that each time the initial grid was a 6×6 tensor grid.

6.2.4 Flow lines

Because this is a CFD problem, also flow lines can be drawn of the mean flow (see Figure 6.10a). Flow lines of the variance or standard deviation cannot be drawn easily, because the distribution is multi-variate. For this, the covariance can be used.

The covariance of two random variables X and Y is the following quantity

$$\mathbb{C}(X, Y) = \mathbb{E}\big((X - \mathbb{E}X)(Y - \mathbb{E}Y)\big),$$

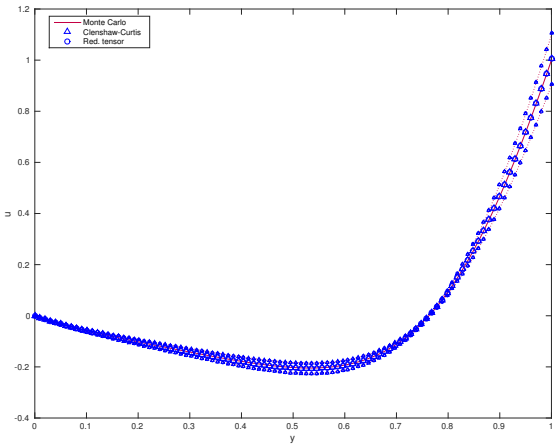
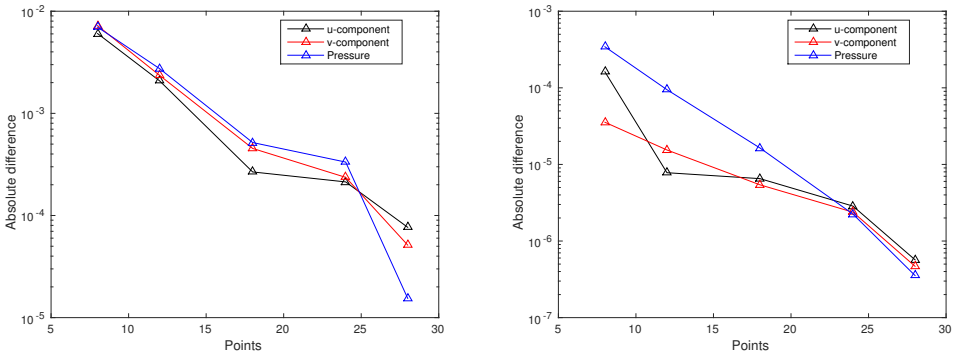


Figure 6.7: The mean and the standard deviation of the u-component of the flow velocity, calculated using different methods. The three lines are the mean, the mean plus twice standard deviation and mean minus twice standard deviation.



(a) Mean *(b) Variance*

Figure 6.8: The convergence of the mean and the variance of the SC method with a reduced tensor cubature rule. The values were calculated by taking the norm of the differences.

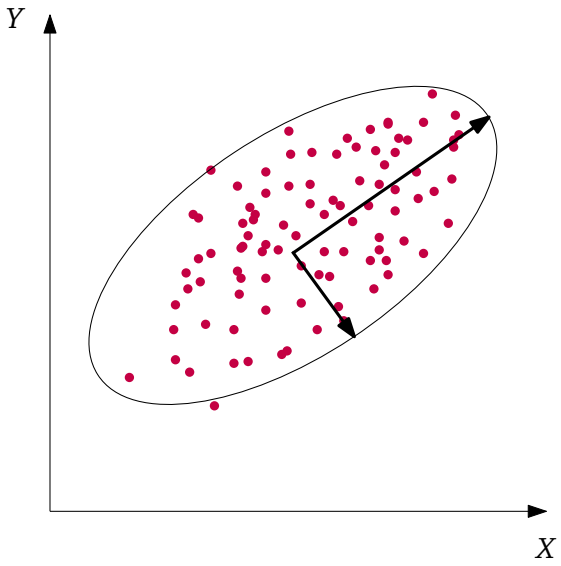


Figure 6.9: The eigenvectors of the covariance matrix, plotted as black arrows. Here the red points are the sample points and the ellipse is the variance of the distribution. The lengths of the arrows are the eigenvalues of the covariance matrix.

which is closely related to the second moment because

$$\mathbb{C}(X,Y) = \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y).$$

Moreover, note that $\mathbb{C}(X,X) = \mathbb{V}(X)$ (where $\mathbb{V}$ denotes the variance).

The covariance matrix Σ with respect to random variables X and Y is the matrix

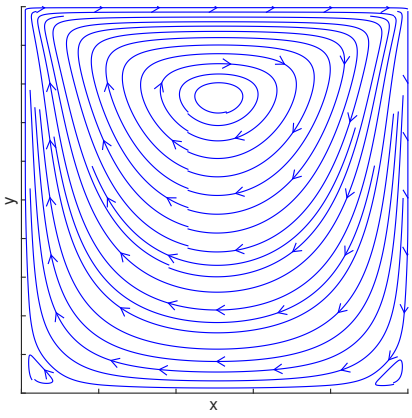
$$\Sigma = \begin{pmatrix} \mathbb{C}(X,X) & \mathbb{C}(X,Y) \\ \mathbb{C}(Y,X) & \mathbb{C}(Y,Y) \end{pmatrix}.$$

Σ is positive-semidefinite and symmetric. Its definition can be extended to multiple random variables $X_1, X_2, X_3, \ldots$.

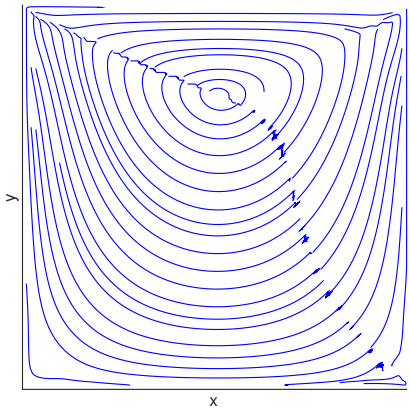
The eigenvectors of Σ are the quantities of interest here, because they allow to transform the covariance matrix into a diagonal matrix (i.e., "removing" the covariance between X and Y). The diagonal matrix then consists of the eigenvalues, so the eigenvector with the largest eigenvalue is the direction in which the distribution varies most. The eigenvector with the second largest eigenvalue is the direction in which the distribution varies most perpendicularly to the first eigenvector, and so on (see Figure 6.9 for a sketch). For each point in the cavity, the covariance matrix of the velocity components u and v can be determined and the direction of the largest variance (see Figure 6.10b) and the second largest variance (see Figure 6.10c) can be determined. Apparently, the largest variance is in the direction of the flow.

6.3 Twin-engine utility aircraft

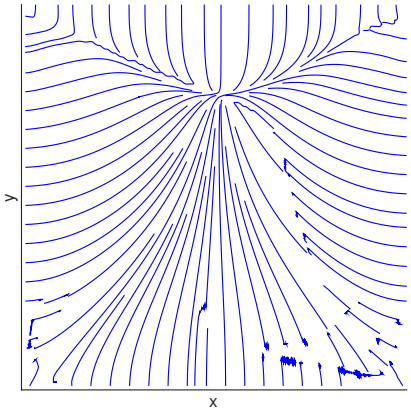
To show off the possibilities the new method provides (i.e., high accuracy with a relatively small number of points) a 3D CFD test case with 7 uncertain parameters is considered. Only the proposed cubature rule is used because the conventional methods need too many points.



(a) Mean



(b) Largest eigenvector of $\mathbb{C}\mathrm{ov}(u, v)$



(c) Smallest eigenvector of $\mathbb{C}\mathrm{ov}(u, v)$

Figure 6.10: The streamlines for the flow of the lid-driven cavity flow problem and the direction of maximum variance including its perpendicular direction.

First the geometry is introduced including the uncertain parameters, which are uncertainties in the boundary conditions and in the geometry. Then it is described how the deterministic problem is solved. Finally, UQ is applied and the results are presented.

Not all properties of the fluid and the geometry are presented to keep the story as clear as possible. All information about the fluid properties and used software can be found in Appendix D.

6.3.1 Problem overview

Before the problem can be described, new nomenclature must be introduced, explaining some relevant quantities used in the field of aircraft aerodynamics.

Nomenclature

When investigating the aerodynamics of an airplane, a relevant question is whether the airplane will produce sufficient lift to keep itself flying. For studying the aerodynamics of an airplane several coefficients may be considered.

The first one is the *pressure coefficient*. The pressure coefficient is a coefficient that is a ratio between the actual pressure along the airplane and the so-called *free-stream* pressure. Free-stream means that the quantity should be measured as if the airplane (which is an obstacle) is not there ("free" stream). It is sometimes called far-field pressure, because it can be modeled by measuring the quantity far from the airplane and considering the air flow as unperturbed there. The pressure coefficient differs for each point along the airplane. Because the geometry is chosen to be uncertain, the pressure coefficient itself cannot be used for the current UQ case, but it can be used to calculate for each simulation the *lift coefficient* and *drag coefficient*.

The lift coefficient is a coefficient that determines the lift force of the airplane and essentially tells whether the airplane goes up or down. It consists of the net upward (with respect to the line of flight) aerodynamic force, scaled by the fluid density, fluid velocity squared, and a projected area of the airplane. If it is positive, the airplane produces positive lift.

The drag coefficient is the same as the lift coefficient, but then measured in the line of flight of the airplane. It can be seen as a measurement of resistance of the airplane, the larger this coefficient is, the larger the aerodynamic drag of the airplane is.

The side-force coefficient is the sum of all other forces, i.e., all forces from the side of the airplane. This coefficient is scaled in a similar way.

Finally, the *moment coefficient* measures the moment around an axis through the center of gravity pointing in side direction. Because the center of gravity of the airplane considered is not known it is impossible to determine this coefficient, so it is only mentioned here for sake of completeness.

Besides these coefficients, a few properties of an airplane also need to be introduced. Four uncertain parameters are about uncertainties in the geometry of the wing of the airplane, in particular the geometry of the cross section of the wing in flight direction (the so-called airfoil). The *chord* of an airfoil is the length of the straight line connecting an airfoil's leading and trailing edge. The *camber* is the curve through the middle of the airfoil. Finally, the *leading edge radius* is the radius of an airfoil at its leading edge (see Figure 6.11 for a sketch of these quantities). These three quantities determine to a large extent the lifting capabilities of the airplane.

Geometry

Knowing all these quantities, the geometry of the airplane can be introduced (see Figure 6.12). Unfortunately, it was not possible to get the geometry of a real airplane because these are not freely available. However, the meshing tool used to generate the meshes for the numerical simulations also

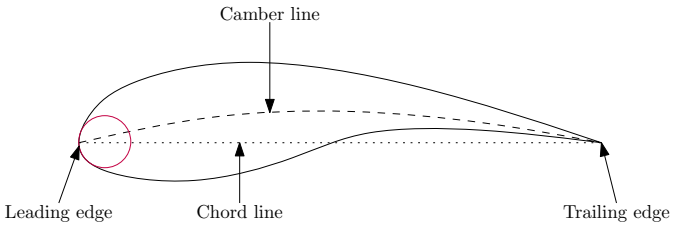


Figure 6.11: A sketched airfoil; the dotted line is the chord, the dashed curve is the camber and the radius of the red circle is the leading edge radius.

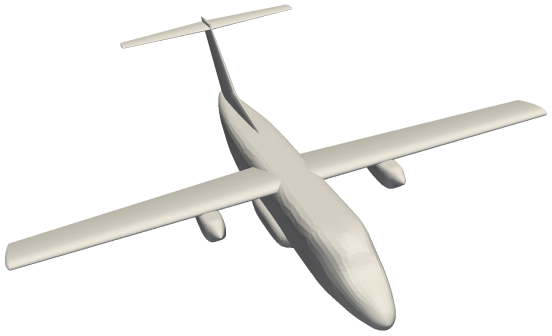


Figure 6.12: The geometry of the twin-engine utility aircraft. The two engines also should have rotor blades, but these are not included in the geometry due to their complex properties.

had several example airplanes. One of those is called the "twin-engine utility aircraft" and that one was used for the current simulations.

Geometrical uncertainties

Seven uncertainties are considered. Three are geometrical uncertainties and are assumed to be normally distributed. The mean is the base geometry and the standard deviation is defined to 5%. See the table below for more information.

Property	**Distribution**
Leading edge radius	$\mathcal{N}$ with 5% standard deviation
Maximum camber as percentage of the chord	$\mathcal{N}$ with 5% standard deviation
Distance of maximum camber from leading edge	$\mathcal{N}$ with 5% standard deviation

These uncertainties are chosen such that the 4-digit NACA airfoil can be used to model these. The 4-digit NACA airfoil is an airfoil defined by three parameters (that form 4 digits), which are essentially the three parameters from the table above.

The NACA airfoil without camber (so chord and camber line overlap) can be defined as a function y of x with the formula

$$y(x) = 5tc\left(0.2969\sqrt{\frac{x}{c}} - 0.1260\frac{x}{c} - 0.3537\left(\frac{x}{c}\right)^2 + 0.2843\left(\frac{x}{c}\right)^3 - 0.1015\left(\frac{x}{c}\right)^4\right),$$

where c is the chord length and t is the maximum thickness as a fraction of the chord. Filling in any x between 0 and c gives the half thickness at the given value of x.

Remark 31. *Officially, this is not the definition of the NACA 4-digit airfoil, because then the 0.3537 should be replaced by 0.3516. That airfoil however has an open trailing edge, which is unwanted during numerical simulations.*

The equation of a cambered 4-digit NACA airfoil is not defined as a function of x, but as a function from $\mathbb{R}^2$ to $\mathbb{R}^2$. Let $x \in [0, c]$ and determine $y(x)$. Then both $(x, y(x))$ and $(x, -y(x))$ are a coordinate on the NACA without chord. These coordinates are transformed into the coordinates

$$\begin{aligned} x_U &= x - y(x)\sin\vartheta, & y_U &= y_c(x) + y(x)\cos\vartheta, \\ x_L &= x + y(x)\sin\vartheta, & y_L &= y_c(x) - y(x)\cos\vartheta, \end{aligned}$$

such that (x_L, y_L) is a coordinate on the lower surface and (x_U, y_U) on the upper. ϑ is defined as $\arctan\left(\frac{\mathrm{d}y_c}{\mathrm{d}x}\right)$ and y_c is the formula of the chord, defined as

$$y_c(x) = \begin{cases} m\dfrac{x}{p^2}\left(2p - \dfrac{x}{c}\right), & \text{if } 0 \le x \le pc, \\ m\dfrac{c-x}{(1-p)^2}\left(1 + \dfrac{x}{c} - 2p\right), & \text{if } pc \le x \le c, \end{cases}$$

where m is the size of maximum camber and p is the location of maximum camber as percentage of the chord.

The leading edge radius of a 4-digit NACA airfoil can be calculated using the formula

$$r = 1.1019t^2.$$

The four digits that officially define a NACA airfoil are $100m$, $10p$, and $100t$, with m, p, and t rounded off. The base geometry of the airfoil that was used to generate the wing of the airplane is the NACA2412, i.e., $m = 0.02$, $p = 0.4$, and $t = 12\%$.

Operational uncertainties

Operational uncertainties are parameters of the numerical simulation itself (such as viscosity). Four uncertainties of this kind are considered. All have the same distribution, which is according to industrial literature (Num, 2015).

Property	Distribution
Angle of incidence	$\beta(4,4)$ with range $2.31° \pm 5\%$
Side-slip angle	$\beta(4,4)$ with range $0° \pm 0.5°$
Mach number	$\beta(4,4)$ with range $0.72 \pm 5\%$
Free-stream pressure	$\beta(4,4)$ with range $101\,325\ N/m^2 \pm 5\%$

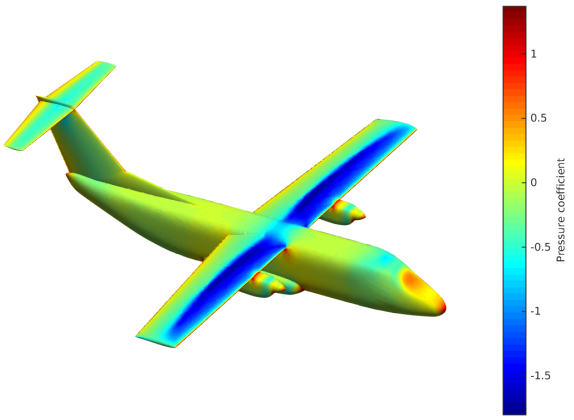


Figure 6.13: An example solution of the pressure coefficient on the airplane. The uncertain inputs are fixed on their respective expected values.

The *angle of incidence* and *side-slip angle* are not introduced yet and together they define the direction of the wind with respect to the airplane. The angle of incidence (or angle of attack) is the angle of the wind in up and down direction. The side-slip angle measures the angle of the wind from the side of the airplane. The wind blows straight onto the airplane if both angles are zero.

6.3.2 Discrete problem

Solving the discrete problem was done using SU^2 (Palacios et al., 2014), `sumo` and `TetGen` (see Figure 6.13 for an example solution).

The programs `sumo` and `TetGen` are responsible for the creation of the mesh. The program `sumo` is an application that can be used to model three-dimensional airplanes. The airplane considered here is an example provided freely with the software. Besides modeling the surfaces, it can also create surface meshes, which are actually the input for `TetGen`. The boundary conditions can be provided on a far-field, which is actually a big ball around the airplane. `TetGen` generates volume meshes from the surface meshes. All meshes are Delaunay triangulations and parameters such as the maximal allowed volume for each tetrahedron can be changed.

SU^2 is an open-source computational analysis tool designed for the simulation and optimization of airplanes and airfoils. It has been coded at Stanford University and is freely available. Many features (such as optimization, engines, elastic boundary conditions) have been implemented and many example test cases are available.

More details, including all the relevant parameters of the mentioned programs, can be found in Appendix D.

6.3.3 Uncertainty Quantification

Being able to solve the deterministic problem makes it possible to start doing Uncertainty Quantification with the seven uncertain parameters. For the latter purpose, a reduced tensor cubature rule with weight grouping of degree 9 is generated. To minimize the required time of all the simulations, the optimizations of Chapter 4 have been implemented. Moreover the simulations were executed in

parallel such that two simulations were able to run at the same time. The whole set of 1 293 simulations took two weeks to complete. If the same set of simulations was done using a Smolyak sparse grid of the same degree, 2 465 simulations were necessary. If a tensor grid was used, then 78 125 simulation were needed.

Lift, drag, and side-force coefficients

The lift, drag, and side-force coefficients are integral quantities, i.e., they do not differ along the body of the airplane but are a fixed constant depending on the whole geometry and fluid properties. Therefore it is possible to determine the PDF and CDF of these quantities and determine the moments using the cubature rule.

The PDF and CDF of the coefficients can be determined using regression techniques. This can be done as follows. Fix K and create all monomials of degree less or equal K. Then fill in all the data and put this in an $N \times \binom{K+d}{d}$ matrix Ψ, where N is the total number of data points (so cubature rule points in this case). K must be chosen such that $\binom{K+d}{d} \ll N$, because then the following system becomes well-conditioned and solvable using the method of least-squares:

$$\Psi \mathbf{v} = \mathbf{b},$$

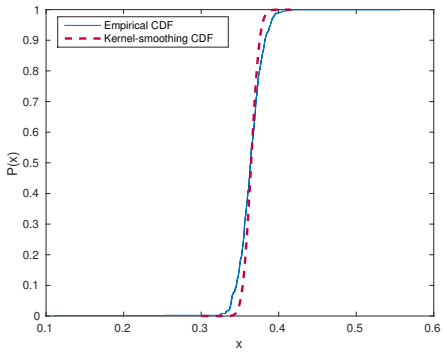
where $\mathbf{b}$ is the vector with realizations (evaluating u at the cubature rule points) of the data points. Solving this yields a polynomial f of degree K with coefficients $\mathbf{v}$, which is an estimation of u. With this estimation, Monte Carlo can be used to create a large number of samples and these can be used to estimate the CDF and PDF (see Figure 6.14). For these estimations of the PDF and CDF 250 000 Monte Carlo samples were drawn and the kernel smoothing estimate of the PDF and CDF was determined on 1 000 points. Note that the sample points are drawn deterministically (i.e., they form not a true set of samples of the distribution), such that the empirical CDF does not truly represent the CDF.

The comparison of the estimated CDF with the empirical CDF yields the worst result for the CDF of the drag coefficient. The result is the best for the side-force coefficient. The PDF is determined for several polynomial degrees K, such that convergence can be determined. For the lift and drag coefficient, it seems that the result is converged. For the side-force coefficient this is not the case. This is probably because the PDF has a very small variance.

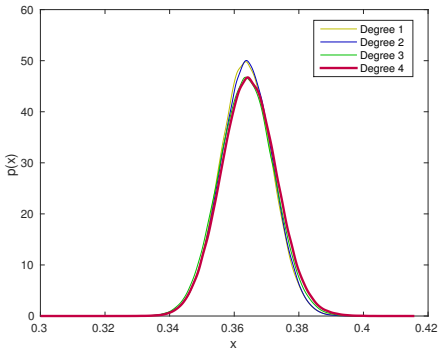
With this estimation, there are two ways to calculate the moments of the coefficients. The preferred method is to apply the cubature rule directly to the data. The simulation points are chosen such that they are the actual cubature points, so the evaluation of the moments in this way should yield the most accurate estimation. The second way is to use the 4^{th} degree estimation of f and integrate this function. f is a high-dimensional polynomial and calculating the fourth moment is then equal to integrating a polynomial of degree 16 in 7 dimensions. Reduced cubature rules with weight grouping can be used here. The coefficients determined with the first method are the best and denoted as c_l, c_d, and c_{sf}. The estimations using f are denoted as $\hat{c}_l$, $\hat{c}_d$, and $\hat{c}_{\text{sf}}$ (see Table 6.2). The results compare very well, so the PDF is actually a pretty good estimation of the real PDF. Note that here the results of the side-force coefficient are again the worst.

Pressure coefficient

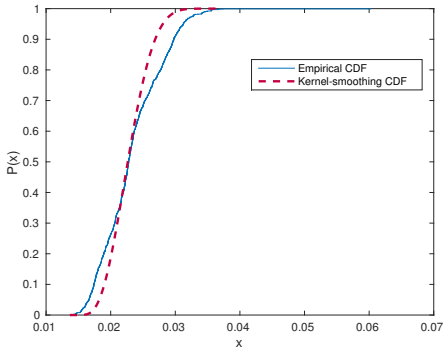
The pressure coefficient differs along the body of the airplane, hence determining one list of values for the airplane is not possible. It is however possible to calculate central moments that the method determines at each location of the airplane. For this the geometrical uncertainties cannot be taken into account, so these are neglected for now (see Figure 6.15).



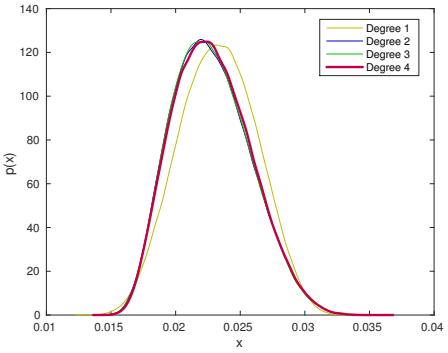
(a) CDF of c_l



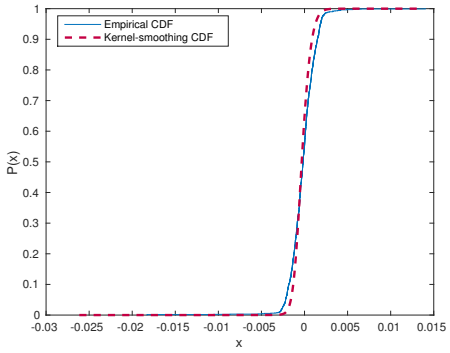
(b) PDF of c_l



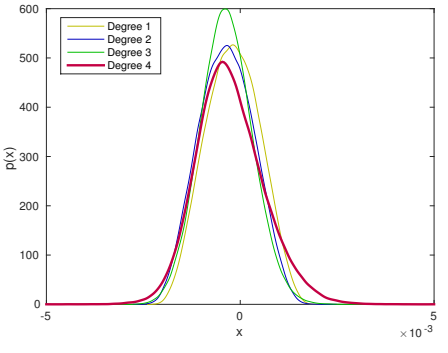
(c) CDF of c_d



(d) PDF of c_d



(e) CDF of c_{sf}

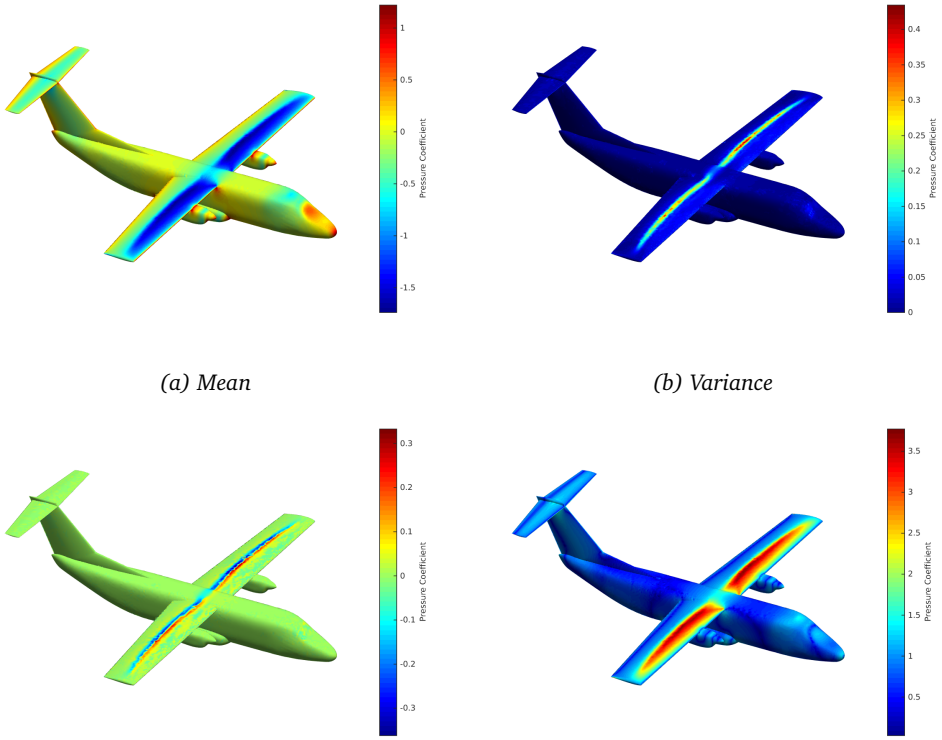


(f) PDF of c_{sf}

Figure 6.14: The CDF and PDF of the three coefficients mentioned in the text. The empirical CDF is generated using the exact values on the cubature points. The different degrees of the estimation of the PDF are the degrees of the interpolation polynomial. The kernel smoothing CDF in the left figures corresponds to an estimation of degree 4.

#	$\mathbf{c}_l$	$\hat{\mathbf{c}}_l$	$\mathbf{c}_d$	$\hat{\mathbf{c}}_d$	$\mathbf{c}_{sf}$	$\hat{\mathbf{c}}_{sf}$
0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
1	0.3640	0.3645	0.0226	0.0228	-0.0015	-0.0002
2	0.1326	0.1330	0.0005	0.0005	0.0000	0.0000
3	0.0483	0.0485	0.0000	0.0000		
4	0.0176	0.0177				

Table 6.2: The first four non-central moments determined either using the cubature rule directly on the results (without hat) or using a high-degree cubature rule on the least-squares estimation (with hat).



(a) Mean *(b) Variance*

(c) Third moment *(d) Fourth moment*

Figure 6.15: The first four central moments of the pressure coefficient on the airplane. Of the k^{th} moment the k^{th} root is taken such that the units are equal.

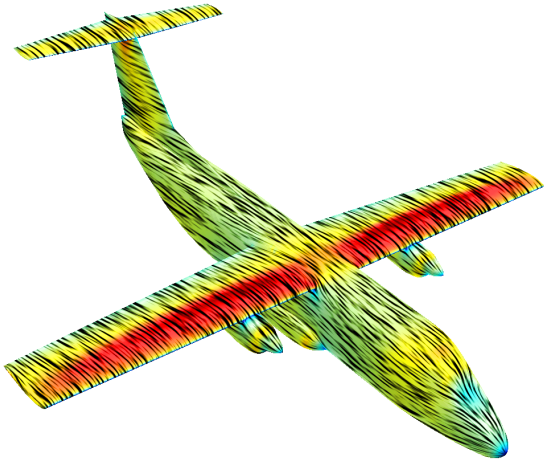
Flow lines

In the CFD case of the lid-driven cavity flow, flow lines were drawn for the mean flow and for the direction of the largest variance. The same can be done for the airplane. Note that only the flow on the airplane is known, so creating flow lines as in the case of the lid-driven cavity flow is not possible.

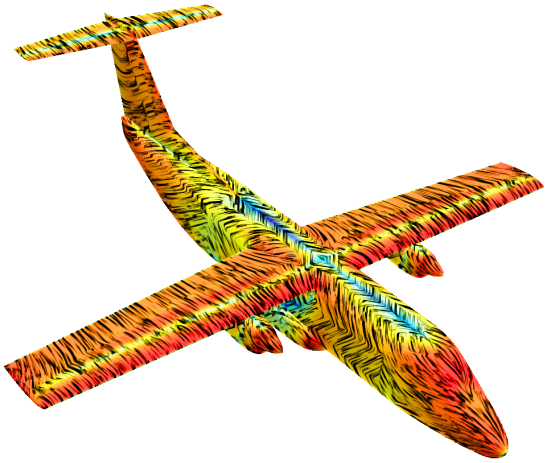
A technique to visualize the flow is line integral convolution (LIC). A randomized pattern of droplets is created along the body of the airplane which is evolved along the side of the airplane, leaving traces of ink behind. The amount of ink the droplet loses is the result of a convolution with a distribution (mostly Gaussian). Implementing LIC on a non-trivial surface (like an airplane) is hard, so for the current visualizations the SurfaceLIC plugin of the visualization program Paraview was used (see Appendix D). The contrast has been enhanced to create black flow lines (see Figure 6.16a for the mean flow).

Just as in the previous case, it is possible to determine the covariance matrix for each point on the airplane. This matrix is 3×3 and the eigenvector with the largest eigenvalue is the direction of the largest variance (see Figure 6.16b). The magnitude of the vector is given on a logarithmic scale to properly show differences. Note that on the wings the direction of largest variance is in the direction of the flow. This is not the case on the fuselage, where apparently the direction of the flow is more uncertain. The direction is least uncertain on top of the fuselage and below the wings (the blue regions).

Remark 32. *For the pressure coefficient, also the third and the fourth moment (skewness and kurtosis) are determined. Theoretically, it is also possible to define a co-skewness-matrix and co-kurtosis-matrix to determine the direction of the largest skewness and kurtosis, but this matrix is not uniquely defined and interpretations of this vary so much that this has not been done.*



(a) Mean



(b) Direction of largest variance (logarithmic scale)

Figure 6.16: The mean flow along the airplane and the direction of largest variance along the airplane. The colors are the magnitudes of the respective vectors.

Chapter 7

Conclusion and Discussion

The overall goal of this research was to improve the Stochastic Collocation method such that the accuracy remained the same but the computational time decreased. Several concepts of Uncertainty Quantification have been discussed and a new integration routine has been proposed. In this chapter the report is concluded and the general results are briefly discussed. This chapter is concluded with options for future work.

7.1 Uncertainty Quantification

The main goal was to apply UQ to CFD problems. The chosen method was the Stochastic Collocation method because it results in robust estimations and the collocation points are chosen deterministically.

In literature (and in the Introduction), this method is described slightly different from how it is applied in the current setting. SC is typically not evaluating at quadrature or cubature rule points, but evaluating at interpolation points such that an interpolant of u is determined. Then this interpolant can be integrated to determine the moments of the distribution, the PDF or other quantities of interest. The method applied here might therefore not be called Stochastic Collocation. However, the method is a collocation procedure because collocation just means the placement of points and it is stochastic due to the dependency on the distribution.

The original Stochastic Collocation procedure is not applied because there is a stringent lower bound of the number of points that can be used. If the interpolation function is a function from a d-dimensional space, the minimum number of points that are necessary to describe such a function is d. This lower bound cannot be realized in the high-dimensional setting of UQ because then many instabilities occur in the interpolant.

One way to overcome this problem is interpolating with a Smolyak sparse grid. This is originally also the main purpose of the Smolyak sparse grid, because it consists of several tensor grids and interpolation on tensor grids is easy ánd stable. As shown, the Smolyak sparse grid can also be used to integrate up to a high degree, which is the current focus.

The minimum number of points necessary for a d-dimensional integration routine is not fixed. There is a lower bound (see Theorem 19 on page 33) but no cubature rule exists that uses the number of samples of the lower bound.

If one likes to use both interpolation and integration, it is advised to use the one-dimensional reduced Gaussian quadrature rule and generate a Smolyak sparse grid with it. The Smolyak sparse grid can be used to interpolate and due to the dependence on the distribution of the quadrature rule, it can also be used for integration. A Gaussian quadrature rule is not nested and therefore creates a lot of points in the Smolyak sparse grid.

If one is only interested in moments, it is advised to directly generate a reduced tensor cubature rule. Interpolation may be done using regression techniques, but only in low-dimensional function spaces.

7.2 Integration techniques

Because in this report the main interest was to calculate the moments of the unknowns, the main focus was on integration techniques. Several one- and multi-dimensional approaches have been discussed.

7.2.1 One-dimensional case

In the one-dimensional case, there is one method that yields the smallest number of points with the highest degree: the Gaussian quadrature rule. However, the quadrature rule points of this method are not nested. The points of the Clenshaw–Curtis quadrature rule are nested, but has the undesired property that the placement of the points is fixed and therefore does not depend on the distribution of the function.

The proposed quadrature rule does have both properties and can generally be applied to any quadrature rule, but reducing a Gaussian quadrature rule is preferred because of the high degree and the positive weights. However, this quadrature rule is only finitely nested, i.e., if the simulations are started from bottom to top and points are being added until some convergence criterion is met, it may happen that there is no convergence when the highest level (the Gaussian quadrature rule) is reached. This can be overcome by repeatedly applying a Gauss–Kronrod–Patterson quadrature rule, but there is no guarantee that this quadrature rule exists.

7.2.2 Multi-dimensional case

In the multi-dimensional case, the method was compared with the tensor grid method and the Smolyak sparse grid method. Because the number of points in a tensor grid increases rapidly if the dimension increases, this is not the desired method.

As already noted, the Smolyak sparse grid method is originally an interpolation routine and not an integration routine. However, it turns out that the degree of the Smolyak cubature rule is larger than the trivial degree (i.e., less than $\binom{N+d}{d}$ points to integrate $\mathbb{P}(N,d)$). Hence this grid can be used for integration and interpolation.

The cubature rule introduced in this report compares very well to the Smolyak cubature rule if it is applied to Genz test functions. Moreover, also a comparison was made of the number of points that are necessary to integrate up to a certain degree. For this however, in many cases the exact number of points of the Smolyak cubature rule could not be determined, because it involves the creation of cubature rules of many points. For small degrees, the upper bound was checked and it was verified that it is indeed the same as the real number of points. Unfortunately, no stringent upper bound could be deduced for the reduced tensor cubature rule. A lower bound can be deduced by determining the number of points in the weight groups related to the first columns of A.

If the Smolyak cubature rule is used to integrate general continuous functions, there is a result about the convergence of the integral depending on the degree of the cubature rule (Kaarnioja, 2013). The convergence is faster the regular the integrand is. Such a result can be used to determine a stringent integration error if the cubature rule is applied to general continuous functions. In the current report, only a general result is discussed using Taylor series without using any properties of the new cubature rule.

In the one-dimensional case, it was mentioned that positive weights are a very important property. In the multi-dimensional case this is still true, but both the Smolyak cubature rule and the reduced

tensor cubature rule in the form that has been discussed do not have positive weights. It is possible however to generate a cubature rule with positive weights from a large tensor grid in the same way as the one-dimensional case, but this is computationally expensive and it yields an unstable cubature rule.

7.3 Future work

A lot of work has been done in this research on the new cubature rule, but many open ends and unsolved problems remain.

First of all, an upper bound of the number of points of the reduced cubature rule could not be determined. This is closely related to the number of independent vectors in the null space of A which is related to the structure of the independent and dependent columns of A. It would be nice if this structure could be found and hence a good prediction could be made about the number of points without determining these.

If a reduced cubature rule is determined numerically, many QR factorizations have to be done and each time they introduce small numerical errors. These are small, because the resulting cubature rule is pretty accurate, but still numerical errors persist. These numerical errors have not been quantified in the current research.

Several interesting properties have been discussed about one-dimensional cubature rules, but one of the most important properties (positive weights) has been ignored during the deduction of the reduced cubature rule. Only the largest number of points is selected and that group is removed, which yields negative weights. The strategy of the one-dimensional case (choosing α) also works in the multi-dimensional case, but unfortunately makes the resulting cubature rule very unstable. It would be nice if a solution existed to this problem.

If the cubature rule is applied to UQ, the moments can be determined very accurately but the PDF and CDF must be determined using regression techniques due to the unstructured grid. There is however also a method to estimate a PDF from the moments maximizing the entropy of the unknown moments (the so-called maximum entropy distribution or just “maxent” distribution). It has been tried to determine the PDF using this method, but convergence of the method seems to be an issue. More research on the subject of maximum entropy distributions might result in an algorithm that uses the accurately estimated moments to reconstruct the PDF.

Chapter 8

Bibliography

R.E. Caflisch. Monte Carlo and quasi-Monte Carlo methods. *Acta Numerica*, 7(1):1–49, 1998.

C.W. Clenshaw and A.R. Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2:197–205, 1960.

R. Cools. A survey of methods for constructing cubature formulae. In T. O. Espelid and A. Genz, editors, *Numerical Integration*, pages 1–24. Kluwer Academic Publisher, 1992.

P.J. Davis. A construction of nonnegative approximate quadratures. *Mathematics of Computation*, 21: 578–582, 1967.

M.S. Eldred and J. Burkardt. Comparison of non-instrusive polynomial chaos and stochastic collocation methods for uncertainty quantification. In *47th AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2009.

G.A. Evans and J.R. Webster. A comparison of some methods for the evaluation of highly oscillatory integrals. *Journal of Computational and Applied Mathematics*, 112(1–2):55–69, 1999.

A. Genz. Testing multidimensional integration routines. In *Proc. of International Conference on Tools, Methods and Languages for Scientific and Engineering Computation*, pages 81–94, New York, NY, USA, 1984. Elsevier North–Holland, Inc.

A. Genz and B.D. Keister. Fully symmetric interpolatory rules for multiple integrals over infinite regions with Gaussian weight. *Journal of Computational Applied Mathematics*, 71:299–309, 1996.

R.G. Ghanem and P.D. Spanos. *Stochastic Finite Elements: A Spectral Approach*. Springer New York, 1991.

U. Ghia, K.N. Ghia, and C.T. Shin. High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411, 1982.

A. Gil, J. Segura, and N.M. Temme. Chebyshev expansion. In *Numerical Methods for Special Functions*, chapter 3, pages 51–86. Society for Industrial and Applied Mathematics, 2007.

G.H. Golub and J.H. Welsch. Calculation of Gauss quadrature rules. *Mathematics of Computation*, 23 (106):221–230+s1–s10, 1969.

Harbrecht H., Peters M., and M. Siebenmorgen. On multilevel quadrature for elliptic stochastic partial differential equations. In J. Garcke and M. Griebel, editors, *Sparse Grids and Applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 161–179. Springer Berlin Heidelberg, 2013.

A. Hinrichs and E. Novak. Cubature formulas for symmetric measures in higher dimensions with few points. *Mathematics of Computation*, 76(259):1357–1372, 2007.

V. Kaarnioja. Smolyak quadrature. Master's thesis, University of Helsinki, 2013.

J. van Kan, A. Segal, and F.J. Vermolen. *Numerical Methods in Scientific Computing*. Delft Academic Press / VSSD, 2014.

J. Kelleher. *Encoding Partitions as Ascending Compositions*. PhD thesis, University College Vork, December 2005.

J. Kelleher and B. O'Sullivan. Generating all partitions: A comparison of two encodings. *arXiv:0909.2331v2 [cs.DS]*, 2014.

A.S. Kronrod. *Nodes and weights of quadrature formulas: sixteen-place tables*. Consultants Bureau, 1965.

D.P. Laurie. Calculation of Gauss–Kronrod quadrature rules. *Mathematics of Computation*, 66:1133–1145, 1997.

H.N. Najm. Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. *Annual Review of Fluid Mechanics*, 41(1):35–52, 2009.

E. Novak and K. Ritter. Simple cubature formulas with high polynomial exactness. *Constructive Approximation*, 15(4):499–522, 1999.

BC-02, Basic Challenge of UMRIDA Project test case descriptions. Numeca, 2015.

F. Palacios, J. Alonso, K. Duraisamy, Colonno M., J. Hicken, A. Aranake, A. Campos, S. Copeland, T. Economon, A. Lonkar, T. Lukaczyk, and T. Taylor. Stanford university unstructured (SU2): Analysis and design technology for turbulent flows. In *51st AIAA Aerospace Sciences Meeting including The New Horizons Forum and Aerospace Exposition*. American Institute of Aeronautics and Astronautics, 2014.

T.N.L. Patterson. The optimal addition of points to quadrature formulae. *Mathematics of Computation*, 22(104):847–856, 1968.

S. Smolyak. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Soviet Mathematics, Doklady*, 4:240–243, 1963.

A.L. Teckentrup, P. Jantsch, C.G. Webster, and M. Gunzburger. A multilevel stochastic collocation method for partial differential equations with random input data. *arXiv:1404.2647 [math.NA]*, 2014.

A. Vasseur. Higher derivatives estimate for the 3d Navier–Stokes equation. *Annales de l'Institut Henri Poincaré*, 27(5):1189–1204, 2010.

G.W. Wasilkowski and H. Wozniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. *Journal of Complexity*, 11(1):1–56, 1995.

A. Williams. Loopless generation of multiset permutations by prefix shifts. *SODA 2009, Symposium on Discrete Algorithms*, 2009.

K.Q. Ye. Orthogonal column latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association*, 93(444):1430–1439, 1998.

Q. Zou and X. He. On pressure and velocity boundary conditions for the Lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, 1997.

Appendices

Appendix A

Proofs of cubature rule theorems

Three central theorems are formulated about the RGV-matrix of the tensor cubature rule, generated with either Gaussian, symmetric or equal quadrature rules. In this appendix these theorems will be proved.

A.1 Combinatorics

There are several lemmas which are useful in the proofs of the theorems, containing some combinatorics. The lemmas are listed below.

All lemmas have the same structure and are about counting the number of sequences. Each lemma consists of the restrictions on this sequence and the number of such sequences that exist. So let $n_k \in \mathbb{N}^d$.

Lemma 33. *If*

- $\|n_k\|_1 \leq B$, *where* $B \in \mathbb{N}_+$,

then there exist $\binom{B+d}{d}$ *such sequences.*

Proof. Let s be a sequence of $B+d$ ones. Pick d times a 1 from that sequence. Sum the numbers between these ones, which forms the sequence n_k. Each sequence n_k can also be written as such a sequence of ones.

For example, if $B = 5$ and $d = 2$, then $s = (1,1,1,1,1,1,1)$. Choosing the ones at position 2 and 5 gives $s = (\underbrace{1}_{1}, \mathbf{1}, \underbrace{1,1}_{2}, \mathbf{1}, 1, 1)$, so $\{n_k\} = \{1, 2\}$.

The ones can be chosen in $\binom{B+d}{d}$ ways, because the order does not matter. □

Lemma 34. *If*

- $\|n_k\|_1 \leq B_1$, *where* $B_1 \in \mathbb{N}_+$,
- $\|n_k\|_\infty \leq B_2$, *where* $B_1 > B_2 > 0$ *and* $B_2 \in \mathbb{N}$,

then there exist $\binom{B_1+d}{d} - \binom{B_1-B_2+d-1}{d-1}(B_1 - B_2)$ *such sequences.*

Proof. Without the second condition, the number of sequences equals $\binom{B_1+d}{d}$. The result follows after counting all the sequences that have at least one element larger than B_2.

Consider all the sequences with sum equal to $B_1 - B_2$ of length d. There are $\binom{B_1-B_2+d-1}{d-1}$ such sequences. Adding individual values to elements, the sum can be lifted to B_1. There are $B_1 - B_2$ possible ways to do this. All sequences with at least one element larger than B_2 can be written as any sequence with all elements smaller than B_2 plus one of these sequences. □

Lemma 35. *If*

- $\|n_k\|_\infty \leq B$, *where* $B \in \mathbb{N}_+$,
- n_k *is weakly increasing,*

then there exist $\binom{B+d}{d}$ *such sequences.*

Proof. Let s be a sequence of $B + d$ ones and change d times a 1 into a 0. The k^{th} element of $\{n_k\}$ is determined by summing the ones of s to the k^{th} 0. Furthermore, all sequences which validate the requirements can be constructed in this way. There are $\binom{B+d}{d}$ possible choices of zeros. □

Finally, there is also a case where an explicit bound does not exist (or is not found yet).

Lemma 36. *If*

- $\|n_k\|_1 \leq B$, *where* $B > 0$ *and* $B \in \mathbb{N}$,
- n_k *is weakly increasing,*

then there exist $1 + \sum_{l=1}^{B} p(l,d)$ *such numbers, where* p *is the restricted partition function*[*].

Proof. If the demand was that $\|n_k\|_1 = B$, then this is the same as looking for all compositions of the number B with at most d summands because n_k has length d, which is $p(B,d)$ (sequence A026820 in OEIS[†]). Demanding that $\|n_k\|_1 \leq B$ is then the same as counting all compositions of the numbers $0, 1, \ldots, B$. The number of partitions of the number 0 is 1 and for the other cases the value is the restricted partition number $p(l,d)$. □

A.2 Proofs

A.2.1 Proof of Theorem 25

Proof of Theorem 25 on page 38. First, the statement is proved in the one-dimensional case. Then using this, it is proved for the multi-dimensional setting.

In the one-dimensional case, the plain $N \times N$ Vandermonde-matrix can be constructed for the Gaussian quadrature rule. This matrix is non-singular and is constructed using the monomials of degree less or equal $N - 1$. The row space of the matrix is a basis of $\mathbb{R}^N$. If the row is constructed using a monomial of larger degree, this row is in $\mathbb{R}^N$ and thus can be expressed in the rows of the Vandermonde-matrix.

[*]There are several definitions of the restricted partition number. Here, it is the number of compositions of the number l with at most d summands.

[†]Online Encyclopedia of Integer Sequences, see `https://oeis.org/A026820` (retrieved November 5th, 2015).

In the multi-dimensional case things are less trivial, because the matrix is not square anymore. Therefore, let $\alpha \in \mathbb{N}^d$ be given and let $\mathbf{a}$ be the row vector with respect to this monomial. Assume that there is a k such that $\alpha_k \geq N$. Hence $\mathbf{a}$ can be written as

$$\begin{aligned}\mathbf{a} &= \left(\sum_{k_1} \mathbf{x}_{k_1}^{\alpha}, \ldots, \sum_{k_M} \mathbf{x}_{k_M}^{\alpha} \right) \\ &= \left(\sum_{k_1} (x_{k_1}^{(k)})^N \mathbf{x}_{k_1}^{\beta}, \ldots, \sum_{k_M} (x_{k_M}^{(k)})^N \mathbf{x}_{k_M}^{\beta} \right),\end{aligned}$$

where β is chosen such that it is equal to α with the exception that $\beta_k = \alpha_k - N$ (which is non-negative due to the assumptions). Recall that $x_{k_j}^{(k)}$ is the k_j^{th} quadrature point of the Gaussian quadrature rule in the k^{th} dimension.

Now $\{x_1^{(k)}, \ldots, x_M^{(k)}\}$ is a Gaussian quadrature rule. Hence it is proved that

$$\begin{aligned}\left((x_1^{(k)})^N, \ldots, (x_M^{(k)})^N \right) &= \sum_{l=1}^{N-1} c_l \left((x_1^{(k)})^l, \ldots, (x_M^{(k)})^l \right) \\ &= \left(\sum_{l=1}^{N-1} c_l (x_1^{(k)})^l, \ldots, \sum_{l=1}^{N-1} c_l (x_M^{(k)})^l \right).\end{aligned}$$

Replacing this in the equation above, yields

$$\begin{aligned}\mathbf{a} &= \left(\sum_{k_1} (x_{k_1}^{(k)})^N \mathbf{x}_{k_1}^{\beta}, \ldots, \sum_{k_M} (x_{k_M}^{(k)})^N \mathbf{x}_{k_M}^{\beta} \right) \\ &= \left(\sum_{k_1} \sum_{l=1}^{N-1} c_l (x_{k_1}^{(k)})^l \mathbf{x}_{k_1}^{\beta}, \ldots, \sum_{k_M} \sum_{l=1}^{N-1} c_l (x_{k_M}^{(k)})^l \mathbf{x}_{k_M}^{\beta} \right) \\ &= \sum_{l=1}^{N-1} c_l \left(\sum_{k_1} (x_{k_1}^{(k)})^l \mathbf{x}_{k_1}^{\beta}, \ldots, \sum_{k_M} (x_{k_M}^{(k)})^l \mathbf{x}_{k_M}^{\beta} \right),\end{aligned}$$

which can be written as $\mathbf{a} = \sum_{l=1}^{N-1} c_l \mathbf{a}_l$, where $\mathbf{a}_l$ are rows whose monomials have smaller degree than $\mathbf{a}$, hence $\mathbf{a}$ is dependent of rows of smaller degree.

Concluding, only the rows created with a monomial power α such that $\|\alpha\|_1 \leq N-1$ are independent. All rows with $\|\alpha\|_1 \leq 2N-1$ and $\|\alpha\|_\infty > N-1$ are then dependent. Lemma 34 concludes the proof. □

A.2.2 Proof of Theorem 26

Proof of Theorem 26 on page 39. Fix $\alpha_1, \alpha_2 \in \mathbb{N}^d$. Let $\mathbf{a}_1$ be the row of α_1:

$$\mathbf{a}_1 = \left(\sum_{k_1} \mathbf{x}_{k_1}^{\alpha_1}, \ldots, \sum_{k_M} \mathbf{x}_{k_M}^{\alpha_1} \right),$$

and let $\mathbf{a}_2$ be the row of α_2. Let any cubature point $\mathbf{x}$ be given.

Let $\mathbf{y}$ be equal to $\mathbf{x}$, except that x_{d_1} and x_{d_2} are switched. Hence if

$$\mathbf{x} = (x_1, x_2, \ldots, x_{d_1-1}, x_{d_1}, x_{d_1+1}, \ldots, x_{d_2}, \ldots, x_d),$$

then

$$\mathbf{y} = (x_1, x_2, \ldots, x_{d_1-1}, x_{d_2}, x_{d_1+1}, \ldots, x_{d_1}, \ldots, x_d).$$

Because the same quadrature rule is chosen in the d_1^{th} and d_2^{th} direction, both $\mathbf{x}$ and $\mathbf{y}$ have the same weight. Two cases can be considered.

- If $x_{d_1} = x_{d_2}$, then $\mathbf{x} = \mathbf{y}$, hence $\mathbf{x}^{\alpha_1}$, $\mathbf{y}^{\alpha_1}$, $\mathbf{x}^{\alpha_2}$ and $\mathbf{y}^{\alpha_2}$ are all equal.
- If $x_{d_1} \neq x_{d_2}$, then $\mathbf{x}^{\alpha_1} = \mathbf{y}^{\alpha_2}$ and $\mathbf{x}^{\alpha_2} = \mathbf{y}^{\alpha_1}$.

So $\mathbf{x}^{\alpha_1} + \mathbf{y}^{\alpha_1} = \mathbf{x}^{\alpha_2} + \mathbf{y}^{\alpha_2}$. This is true for all such points, hence $\mathbf{a}_1 = \mathbf{a}_2$.

Concluding, each row generated with a monomial power that is the permutation of the monomial power of another row is the same. This is the same as demanding that the monomial power must be weakly increasing, hence Lemma 36 concludes the proof. □

A.2.3 Proof of Theorem 27

Proof of Theorem 27 on page 39. Let k be given and assume the k^{th} quadrature rule is symmetric around the point p.

First rewrite the k^{th} quadrature rule as points around p, so $\{x^{(k)}\} = \{p + y^{(k)}\}$. Then $y^{(k)}$ and $-y^{(k)}$ are the same quadrature rule. Hence the tensor cubature rule points $(x^{(1)}, x^{(2)}, \ldots, p + y^{(k)}, \ldots, x^{(d)})$ and $(x^{(1)}, x^{(2)}, \ldots, p - y^{(k)}, \ldots, x^{(d)})$ have the same weight. Let $\mathbf{a}$ be the row of the monomial power α, where $\alpha_k = q$ with q odd. Let $\mathbf{x}$ be given and construct $\mathbf{y}$.

Then

$$\begin{aligned}(p + y_k)^q + (p - y_k)^q &= \sum_{l=0}^{q} \binom{q}{l} p^l y_k^{q-l} + \sum_{l=0}^{q} \binom{q}{l} p^l (-1)^{q-l} y_k^{q-l} \\ &= \sum_{l=0}^{q} \binom{q}{l} p^l (1 + (-1)^{q-l}) y_k^{q-l}.\end{aligned}$$

But this last formula does not contain a factor y_l^q anymore, because $1 + (-1)^{q-k} = 0$ for $k = 0$. Hence $y_k^q = \sum_{l=0}^{q-1} c_l y_k^l$. So the row can be expressed in rows with smaller degree, which means that the row is dependent.

Concluding, only the rows with even powers are relevant. If N is even, it could be seen as generating all rows with monomial degree smaller or equal than $\frac{N}{2}$ and doubling the result. If N is odd, the monomial degree becomes $\frac{N-1}{2}$, because N itself is odd. □

Appendix B

Reduced tensor cubature rule: tables

This appendix contains tables with some results for reduced tensor cubature rules with weight grouping and shows that the proved theorems are indeed true. Each time the numbers are compared with the numbers from Novak and Ritter (1999). These numbers are officially upper bounds, but it is verified for some cases that these numbers are indeed stringent. Moreover, the differences are so large that it does not matter that it is an upper bound, although this has not been verified formally.

Each time the value of $N(K,d)$ is determined, which is the number of points that are necessary to integrate all functions in $\mathbb{P}(K,d)$ accurately, using either the Smolyak cubature rule or the reduced tensor cubature rule with weight grouping (N_S or N_R).

B.1 General case

Following Theorem 25 on page 38 and the discussion of Section 3.3.4 on page 39, an upper bound can be determined that should be better than the upper bound for Smolyak for high K. This can indeed be verified.

K	$N_S(K,5)$	$N_R(K,5)$	$N_S(K,10)$	$N_R(K,10)$
3	11	26	21	176
5	71	147	241	2 343
7	351	512	2 001	16 588
9	1 471	1 372	13 441	82 368
11	5 502	3 108	77 505	322 686
13	18 932	6 258	397 825	1 063 986
15	61 112	11 544	1 862 145	3 074 280
17	187 552	19 899	8 085 505	7 998 705

B.2 Same and non-symmetric

If the same non-symmetric quadrature rule is used, a large improvement over the theoretical upper bound is noticed. The open spots in the 10-dimensional cases could not be determined without

numerical cancellation in the algorithm.

K	$N_S(K,5)$	$N_R(K,5)$	$N_S(K,10)$	$N_R(K,10)$
3	11	53	21	1 053
5	71	73	241	1 233
7	351	255	2 001	2 185
9	1 471	605	13 441	6 085
11	5 502	1 572		
13	18 932	2 967		

B.3 Same and symmetric

If only symmetric quadrature rules are used, an even better improvement is noticed. In some cases, only a quarter of the number of points of the Smolyak cubature rule is necessary. Each result was generated using an Gauss quadrature rule with an odd, as small as possible number of points.

K	$N_S(K,5)$	$N_R(K,5)$	$N_S(K,10)$	$N_R(K,10)$
3	11	11	21	21
5	61	43	221	201
7	241	125	1 581	1 361
9	801	325	8 801	3 705
11	2 433	807	41 265	12 489
13	6 993	1 607	171 425	38 353
15	19 313	3 289	652 065	100 361
17	51 713	5 609	2 320 385	433 585

Appendix C

Genz test functions

In this chapter all values of the integrals of the test functions of Genz and Keister (1996) are shown. In the original implementation of Genz, a different set of test functions was used so those results could not be used in the current case.

C.1 Test function 1: oscillatory

The first test function is given by

$$f_1(\mathbf{x}) = \cos\left(2\pi u_1 + \sum_{i=1}^{d} a_i x_i\right).$$

The following exact value of the integral can be calculated:

$$\int_{[0,1]^d} f_1(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{2^d}{\prod_{i=1}^{d} a_i} \cos\left(\frac{1}{2}\sum_{i=1}^{d} a_i + 2\pi u_1\right) \prod_{i=1}^{d} \sin\left(\frac{a_i}{2}\right).$$

C.2 Test function 2: product peak

The second test function is given by

$$f_2(\mathbf{x}) = \prod_{i=1}^{d} \left(a_i^{-2} + (x_i - u_i)^2\right)^{-1}.$$

The following exact value of the integral can be calculated:

$$\int_{[0,1]^d} f_2(\mathbf{x})\,\mathrm{d}\mathbf{x} = \prod_{i=1}^{d} a_i \left(\tan^{-1}(a_i u_i) + \tan^{-1}(a_i - a_i u_i\right).$$

C.3 Test function 3: corner peak

The third test function is given by

$$f_3(\mathbf{x}) = \left(1 + \sum_{i=1}^{d} a_i x_i\right)^{-(d+1)}.$$

The exact value of this integral could not be found for general n.

C.4 Test function 4: Gaussian

The fourth test function is given by

$$f_4(\mathbf{x}) = \exp\left(-\sum_{i=1}^{d} a_i^2 (x_i - u_i)^2\right).$$

The integral cannot be stated explicitly in elementary functions because it is a Gaussian function, but it can be expressed using error functions as follows:

$$\int_{[0,1]^d} f_4(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{\pi^{d/2} \prod_{i=1}^{d} (\mathrm{erf}(a_i u_i) + \mathrm{erf}(a_i - a_i u_i))}{2^d a_1 a_2 a_3},$$

where

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,\mathrm{d}t.$$

C.5 Test function 5: C_0 function

The fifth test function is given by

$$f_5(\mathbf{x}) = \exp\left(-\sum_{i=1}^{d} a_i |x_i - u_i|\right).$$

The integral can be calculated accurately if it is used that

$$\int_0^1 f_5(x)\,\mathrm{d}x_i = \int_0^{u_i} f_5(x)\,\mathrm{d}x_i + \int_{u_i}^1 f_5(x)\,\mathrm{d}x_i,$$

which is easy to calculate because the absolute value can be neglected. The d-dimensional integral then contains 2^d integrals that can be calculated easily.

C.6 Test function 6: discontinuous function

The last test function can be seen as integrating the function

$$f_6^*(\mathbf{x}) = \exp\left(\sum_{i=1}^{d} a_i x_i\right)$$

over the region $[0,1]^d$ under the condition that $x_1 < u_1$ and $x_2 < u_2$.

The value of this integral for $d \geq 2$ equals

$$\int_{[0,1]^d} f_6(\mathbf{x})\,\mathrm{d}\mathbf{x} = \frac{1}{\prod_{i=1}^{d} a_i} (e^{a_1 u_1} - 1)(e^{a_2 u_2} - 1) \prod_{i=3}^{d} (e^{a_i} - 1)$$

Appendix D

Specification of airplane simulations

This appendix contains all fluid parameters and version numbers of the used programs to make the UQ simulations of the airplane as reproducible as possible.

D.1 List of used software

The following list is not a complete list of all used software, but it does contain the most relevant ones. All simulations were performed on a Linux system. The websites were accessed on October 1st, 2015.

Software	Version	Website
SU[2]	4.0.0	`http://su2.stanford.edu/`
dwfSumo	2.7.5	`http://www.larosterna.com/sumo.html`
TetGen	1.5.0	`http://wias-berlin.de/software/tetgen/`
MATLAB	2014b	`http://nl.mathworks.com/products/matlab/`
Paraview	4.3.1 64bit	`http://www.paraview.org/`
SurfaceLIC*	2.0	
gcc	4.8.3	`https://gcc.gnu.org/`

It might not be possible to get the exact same version as gcc as stated in the table above as it is smarter to take the version that is provided with your distribution.

With this, a `Makefile` was generated that creates a surface mesh from a geometry description, then creates a volume mesh and then performs the simulation. A program was developed that executed this Makefile for all samples points and collected the relevant results. The simulations were performed in parallel (i.e., two at the same time).

*SurfaceLIC is an officially supported plugin of Paraview. Details can be found on the wiki of Paraview: `http://www.paraview.org/Wiki/ParaView/Line_Integral_Convolution`.

D.2 Numerical properties

The following properties are relevant for the numerical properties of the solver. Most (if not all) of these properties are options of SU^2.

Property	**Value**
Number of simulations	1293
Number of uncertain parameters	7
Number of processes used	2
Equation to solve	Euler equations
Method for spatial gradients	Green-Gauss
CFL number	Adaptive in range $[0.95, 100]$
Multi-Grid levels	2
Multi-Grid cycle	W Cycle
Convective numerical method	JST
Slope limiter	Venkatakrishnan
Time discretization	Implicit Euler
Convergence criterion	ρ-residual
Boundary conditions	Euler boundary conditions

D.3 Fluid properties

Some fluid properties are uncertain, but some other define the properties of the fluid (which is air). Most (if not all) of these properties are options of SU^2.

Property	**Value**
Mach number	*uncertain*
Angle of attack	*uncertain*
Side-slip angle	*uncertain*
Free-stream pressure	*uncertain*
Free-stream temperature	273.15 K
Ratio of specific heats (γ)	1.4
Gas constant	287.87 J / (kg $\cdot$ K)
Reference origin	*Not used*