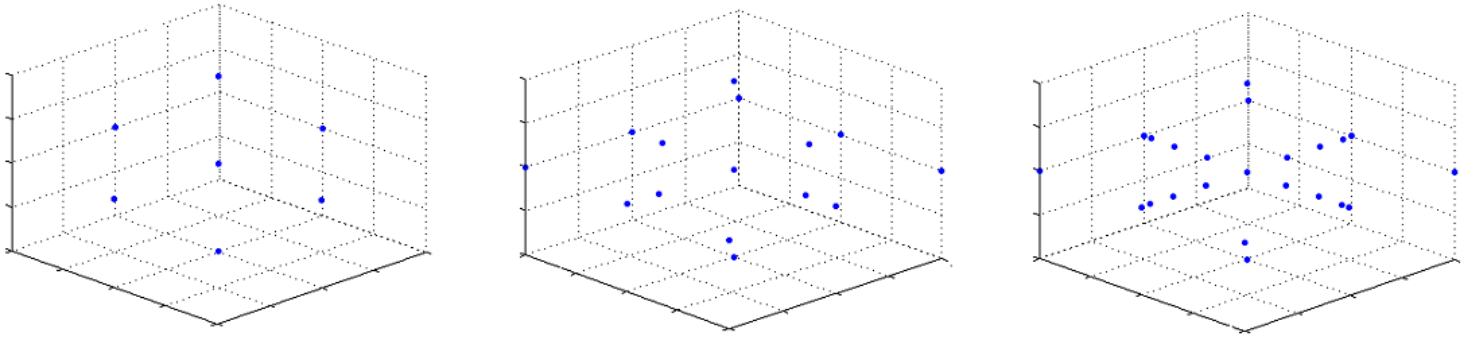
Master of Science Thesis



Dimension-adaptive sparse grid for industrial applications using Sobol variances

Heavy gas flow over a barrier

Desmedt S.G.L.

March 11, 2015

Aerodynamics

Faculty of Aerospace Engineering

TU Delft

Delft University of Technology

Dimension-adaptive sparse grid for industrial applications using Sobol variances

Heavy gas flow over a barrier

Master of Science Thesis

For obtaining the degree of Master of Science in Aerospace Engineering
at Delft University of Technology

Desmedt S.G.L.

March 11, 2015

Faculty of Aerospace Engineering · Delft University of Technology

Delft University of Technology

Copyright © Aerospace Engineering, Delft University of Technology
All rights reserved.

DELFT UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF AERODYNAMICS

The undersigned hereby certify that they have read and recommend to the Faculty of Aerospace Engineering for acceptance the thesis entitled **"Dimension-adaptive sparse grid for industrial applications using Sobol variances"** by **Desmedt S.G.L.** in fulfillment of the requirements for the degree of **Master of Science**.

Dated: March 11, 2015

Supervisors:

_____
Reader 1

_____
Reader 2

_____
Reader 3

_____
Reader 4

Preface

This project has been the first time I was responsible for the planning and execution of a project of this magnitude. There have been a few bumps along the road, but it has definitely been a great learning experience. I would like to thank my family and friends for their support, but I also want to take this opportunity to thank a few people in particular for their help.

First of all, I would like to thank my supervisor at TNO for the majority of the Msc. Thesis, Ir. Pejman Shoeibi Omrani. His advice, feedback, patience and practical experience have helped me out on many occasions. I also want to thank Dr. Richard Dwight, my supervisor at the Aerospace Engineering faculty, for helping me gain insight in the theoretical aspects of the work. Next, I would like to thank Tom O'Mahoney, who was my TNO supervisor for the first three months, for helping me get started and Andreas Mack for his help with the setup of the CFD computations. I would like to thank the kind people at the Heat Transfer and Fluid Dynamics department at TNO for providing me with a productive environment to work in, as well as the members of the Uncertainty Quantification group. I would also like to thank Bernard Vervaet, for his help in proofreading the thesis.

Finally, I want to thank my parents, who have been nothing but supportive throughout my studies in Delft. Without them, I would not have accomplished this.

Summary

The area of interest for this study is the field of uncertainty quantification in computational fluid dynamics. The goal is to contribute to a new method to perform uncertainty quantification analyses for industrial, computationally expensive CFD simulations.

To this end, a new adaptive grid refinement method is developed. The existing sparse grid procedure introduced by Smolyak [20] for high-dimensional parameter spaces is used with Clenshaw-Curtis quadrature rules. Starting from a low level grid, more points are added based on the values of the Sobol variances, which are estimated values. The Sobol variances provide an indication of the importance of each variable and interactions between variables.

The method is applied to an industrial atmospheric flow case, where a heavy gas is released upwind of a barrier. The quantity of interest is the effect distance, the distance from the barrier where the molar concentration drops below 1 percent, which is important for safety. The adaptive refinement algorithm is compared to a standard sparse grid and to the adaptive method by Gerstner & Griebel [18]. The results indicate that the new adaptive grid refinement method requires only 23 grid points, compared to the 69 for the standard sparse grid. The final grid is smaller than for the Gerstner & Griebel approach, which contains 31 grid points. The error of the new algorithm is slightly larger than the Gerstner & Griebel approach when comparing to a standard sparse grid. The method shows promise for use with industrial cases, where a significant reduction in the computational effort can be achieved compared to the standard sparse grid approach.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter will start by giving a general introduction to the field of uncertainty quantification and the goal of this study in Section 1.1. A literature review has been performed, which is summarized in Section 1.2. Finally, Section 1.3 will present the structure of the remainder of the report.

1.1 Uncertainty quantification

The area of interest for this study is the field of uncertainty quantification (UQ) in computational fluid dynamics. Uncertainty can be divided into two categories [3]:

- Epistemic uncertainty: A potential deficiency in any phase or activity of the modeling process that is due to lack of knowledge. This is also known as reducible uncertainty, since increasing the knowledge can reduce this type of uncertainty. An example would be a lack of experimental data.
- Aleatory uncertainty: The physical variation present in the system being analyzed or its environment. This is also known as irreducible or inherent uncertainty.

In CFD simulations, there are different sources of uncertainty: geometric uncertainties, model uncertainties, input uncertainties and more. This study will focus only on the uncertainty of input parameters and the effect on the model response. This is done by finding a probability distribution for the input parameters and then propagating this uncertainty through the model to obtain the statistics of the response.

As will be shown in the literature review, there are a number of methods available to perform UQ analysis, ranging from the traditional random sampling (the most well-known example

being the Monte Carlo method, see for example [4]), to structured sampling approaches, such as sparse grids [5]. One of the big challenges in the field is the curse of dimensionality [2], the exponentional increase in computational effort of conventional approaches with increasing dimension, which is why UQ methods in general strive to keep the required computational work to a minimum.

The goal of this study is to contribute a new method to perform input uncertainty quantification analysis. This new method can then be used for industrial, computationally intensive CFD simulations. We will look into reducing the number of computations required by developing an effective adaptive algorithm. An overview of the currently used methods in the field is given in Section 1.2. The conclusions from the literature review and the structure of the remainder of the report can be found in Section 1.3.

1.2 Literature review

As mentioned in the introduction (Section 1.1), one of the big challenges in the uncertainty quantification (UQ) field is to reduce the computational effort while achieving a high accuracy. This is due to the curse of dimensionality [2], a term coined by Bellman in 1961, stating that the amount of data required grows exponentially with increasing dimension. This becomes a problem especially for high-dimensional problems, but also for moderate- to low-dimensional problems if data sampling is time-consuming.

In the field of uncertainty quantification, there are two main approaches. The first approach are the (quasi-)random sampling methods. The idea here is to take a large number of samples and use these to obtain reliable statistical information.

The second approach uses deterministic sampling, sampling the data at certain points and use it to construct a surrogate model. An example of this type of approach is the use of sparse grids [5]. Once a surrogate model is constructed, it is then used to extract information. The surrogate model is much cheaper to evaluate, which can save a lot of time. The accuracy of the model depends on the method and amount of data used to construct the surrogate model and of course of the the complexity of the original system.

1.2.1 Random Sampling

As mentioned before, traditional approaches use (pseudo-)random sampling, the most known of which is the Monte Carlo (MC) method (see for example [4]). Once a large amount of data is collected, a statistical analysis is perfomed on the results. The advantage of such a method is that it is robust and reliable, given a sufficiently large sample, it is relatively straight-forward to implement, and used in many fields already. The disadvantage of these methods is that they generally require a large number of samples. Especially when dealing with computationally demanding cases, this is not feasible in practice. Even for low-dimensional problems, it is often not possible to take a sufficient amount of samples to perform a meaningful statistical

analysis.

There are techniques which reduce the number of samples required, such as for example Latin Hypercube Sampling (LHS) or importance sampling [22]. However, for industrial applications where cases are computationally expensive this type of approach is generally impractical.

1.2.2 Deterministic sampling

A different type of uncertainty quantification methods uses deterministic sampling. Instead of taking random samples, the computations are run at particular combinations of the input parameters. This data is typically used to construct a surrogate model, which is much cheaper to evaluate.

Polynomial Chaos

In the context of uncertainty quantification, the Polynomial Chaos (PC) framework is well-known. It based on the work by Wiener [6], which originally dealt with stochastic processes with Gaussian random variables. More recently, the generalized PC approach was introduced by Xiu [7].

Note that there are both intrusive and a non-intrusive PC methods. In short, the intrusive approach requires the governing equations to be rewritten. This means altering the source code which is used for the computations. One example of the intrusive approach can be found in [32]. This type of approach is only possible when the source code is available. On the other hand, the non-intrusive approach treats the governing equations constituting the model as a blackbox. The way to gain information about the system is by simply taking data samples, which means running simulations. The non-intrusive approach is much more common for engineering applications, for example [26], [7]. The reason for this is that it is much easier to run computations with a readily available commercial code than it is to rewrite the governing equations in the source code to include intrusive PC.

Polynomial chaos framework

Regardless of the distinction between intrusive and non-intrusive method, all Polynomial Chaos (PC) based methods operate on the same principle. An approximation to the actual model is constructed using an orthogonal set of polynomials which serve as basis functions for an N-dimensional parameter space.

A polynomial chaos expansion can be written as:

$$\mathbf{Y}(\mathbf{X}) := \sum_{j=0}^{\infty} a_j \Phi_j(\mathbf{X}) \tag{1.1}$$

where $\mathbf{Y}$ is the model response and $\mathbf{X}$ contains the input variables, both of which are affected by uncertainty. Equation 1.1 shows the model response as a spectral expansion in the vector

of independent basis random variables $\mathbf{X}$ with random polynomial basis $\{\Phi_j\}$. The solution is split into a deterministic part, coefficients a_j, and a stochastic part, the polynomial chaoses, $\Phi_j(\mathbf{X})$.

The original work by Wiener [6] uses Hermite polynomials as the basis functions to represent Gaussian random variables. Depending on which probability distribution is used for the random variables, different types of polynomials are proposed. An overview of these polynomial basis functions with corresponding probability distributions is found in the Wiener-Askey scheme [8]. For arbitrary probability functions, we refer to the work of Wan and Karniadakis [9]. It is worth noting that using a different type of polynomial than proposed in the Wiener-Askey scheme does not mean that the surrogate model will be inaccurate. Rather, the use of the proposed polynomials in combination with the corresponding input distribution is optimal, and generally converges faster towards the real model. More information on the PC framework can be found in Section 2.2.

The main disadvantage of PC-based methods is that if the model shows non-smooth or discontinuous behavior, polynomial approximation is inaccurate. In these cases, simply adding more data points does not improve the accuracy since the issue is inherent to using polynomial approximation. This effect is also known under the name Gibbs' phenomenon. One way to circumvent this problem would be to use linear interpolation or one of the methods mentioned in Section 1.2.2.

Quadrature

Quadrature rules are used to approximate definite integrals using an optimal number of points by rewriting the integral as the sum of the product of function values (which are determined by sampling the function) and a weight. By choosing a suitable quadrature rule, the convergence of the integral value is improved.
Any quadrature rule follows the following form:

$$\int_{-1}^{1} f(x)\beta(x)dx \approx \sum_{k=1}^{n} w_k f(x_k), \tag{1.2}$$

where $\beta(x)$ is a weight function that corresponds to a probability density function, x_k are the abscissae of the quadrature rule and w_k are the corresponding weights. Since $\beta(x)$ is a PDF, by definition $\int_{-1}^{1} \beta(x)dx = 1$. From this it follows that $\sum w_k = 1$ must be true. The number of abscissae depends on the quadrature rule and increases with increasing level.

One of the properties of a quadrature rule is whether or not the points are nested. Nestedness implies that some or all points from a lower level are re-used for higher levels, meaning that fewer points are required overall for higher level approximations. More information is found in [23].

The best known quadrature rules are the Gaussian quadrature rules. The classical Gaussian quadrature rules, such as Gauss-Legendre, Gauss-Hermite, etc. are non-nested. The abscissae and weights for these Gaussian rules are determined using for example the Golub-Welsch al-

gorithm [10]. More recently, there have been some extensions to existing Gaussian quadrature rules, such as the Gauss-Kronrod [12] and Gauss-Lobatto [11] rules.

There are alternatives to the Gaussian quadrature rules, for example the Newton-Cotes, Clenshaw-Curtis and Fejer quadrature rules. An important property of these rules is that they are nested and use the same abscissae regardless of input variable distribution. The weights can be computed to accomodate the use of different input distributions, for example see [15].

A recent paper by Trefethen [13] compares the convergence of Gauss quadrature with Clenshaw-Curtis quadrature and finds that they are almost equal in practice. If the Clenshaw-Curtis points are used, Lagrange polynomials can serve as the orthogonal basis. An example where this is applied is found in [14].

To summarize, the number of quadrature rules to choose from is extensive, depending on what input distribution is used. Different input distributions can taken into account by using a different othogonal set of polynomials [8], such as for the Gauss rules. Each Gaussian rule has different abscissae, meaning that in order to analyze multiple input distributions, additional samples would be required. On the other hand, for Clenshaw-Curtis this is more straightforward. Here, the input distribution is taken into account solely in the calculation of the weights [15].

Sparse grid vs. full tensor grid

For multidimensional problems, data samples are required in all dimensions. A relatively straightforward method would be to construct a tensor grid of quadrature rules. The disadvantage of this approach is that the number of computations required scaled exponentially with increasing dimension. In practice, it is important to limit the number of computations. For this reason, sparse grids are highly preferred. Sparse grids were first introduced by Smolyak [20]. Using univariate basis functions, a multidimensional function can be constructed by tensor product. The advantage of these grids is that they are more efficient than the traditional full tensor grids. The convergence of both types of grid is comparable, but the number of points used in a sparse grid is significantly lower [21].

Sparse grids are constructed using quadrature rules, which means a good approximation of the integral of the response can be found on these grids. The choice of quadrature rule is dependent on the application, but nested rules can save a lot of points so these are preferred for this study. A qualitative comparison of a tensor grid, a sparse grid and a random sampling grid is given in Figure 1.2.2.

Other methods

There are other methods available to construct surrogate models, some of these are mentioned here for completeness. One alternative to the Polynomial Chaos approach would be to use Kriging to construct a response surface. First, presented by Krige [27], it was originally used in geostatistics but has found uses in other fields as well. Kriging is essentially a different

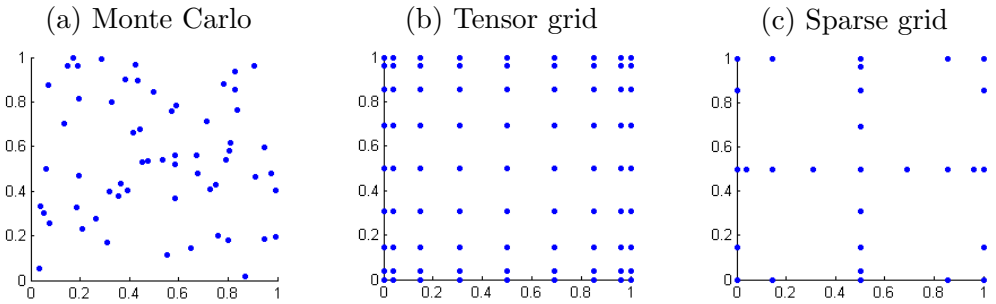


Figure 1.1: Qualitative comparison of a Monte Carlo random sampling approach to a tensor grid and a sparse grid constructed from quadrature formulas in 2 dimensions.

kind of interpolation, where the interpolated values are modeled by a Gaussian process[25]. The method does not require the use of a structured grid, so it is quite flexible. This method has already been used in the field of uncertainty quantification, for example [24] and [28].

For the interpolation of data there are still other options, such as radial basis functions [26]. However, the main idea is always the same: the model is sampled at certain points. This data is used to construct a surrogate model, which is used to analyse the statistics of the model. The closer the surrogate model represents the real model, the better the result of the statistical analysis.

1.2.3 Sensitivity Analysis

It is important to extract useful information from the results, such as which parameters have the most influence on the model response. This is why it is useful to perform a global sensitivity analysis. Sobol indices are used to quantify the relative importance of input variables on the response of a mathematical model. It is a variance-based method. This method is closely related to the ANOVA (ANalysis Of VAriance) decomposition [16].

This approach can be applied to Monte Carlo and other (pseudo-)random methods but Sobol indices can also be calculated analytically.

1.2.4 Adaptivity

One particular point of interest of this project is using an dimension-adaptive grid refinement algorithm to reduce the number of computations as much as possible. Since computational resources are limited, they should be used as efficiently as possible.

There are different approaches to perform adaptive grid refinement. One well-known example is the approach by Gerstner and Griebel [18], based on minimizing an error estimate. In this approach, the effect of adding sparse grid components on the error estimate is determined.

Only those components which introduce a large difference are added. This procedure of adding points goes on until a certain tolerance is reached. The study contains valuable information on how to structure an adaptive sparse grid code.

A different approach is described in [19]. The grid is enriched using a quasi-random approach (Latin Hypercube Sampling or quasi-Monte Carlo) and use is made of the ANOVA decomposition. An approximation of the PC expansion coefficients is then calculated.

The use of Sobol indices for a Stochastic Collocation (which is based on the PC framework) method has already been investigated in [17], where it was concluded that using Sobol indices as a post-processing procedure to the construction of the PC expansion provides an attractive route to adaptive grid refinement. This method would combine the use of sparse grids with a PC expansion. The use of Sobol indices would give us valuable information of the effect of inputs on the model response, which is then used to adaptively refine the grid.

1.3 Report structure

The aim of this project is to contribute a new method for input parameter uncertainty quantification for industrial CFD computations. This means that we want a structured method which provides reliable results and limits the required computational resources. Since computational resources are so limited, random sampling methods are impractical for the purpose of this project. Though the approach provides a reliable statistical analysis, the disadvantage of needing a large number of samples is so large that it is impractical. A deterministic sampling approach using sparse grids is chosen as a basis for the new method, since it is proven to be efficient. Using a sparse grid limits the number of points while maintaining a convergence which is similar to full tensor grids. Clenshaw-Curtis quadrature rules will be used to construct these grids. The sparse grids will be used in combination with the Polynomial Chaos framework to construct an accurate surrogate model, which can then be used to perform a cheap statistical analysis. This approach limits the number of simulations required and still provides reliable results, which is exactly what we are looking for. The method that will be developed and tested in this study will be a dimension-adaptive grid refinement method based on Sobol indices. They will provide an indication of which parameters have the most influence on the model response. This information can then be used to refine the grid.

The remainder of the report will provide an overview of the work performed throughout this project. First, Chapter 2 will provide the mathematical framework, going into more detail on how to construct sparse grids, compute the abscissae and weights to incoorporate different input prarameter distributions. After this, the approximation of the Sobol variances on a sparse grid is given, followed by an existing method by Gerstner and Griebel [18] and the new adaptive grid refinement method. Subsequently, Chapter 3 will verify the implementation of the different methods by using a set of test functions. The new adaptive grid refinement algorithm is applied to an industrial case, the results are presented in Chapter 4. Finally,

Chapter 5 will summarize the findings of the study and provide a few recommendations for the work if it is to be used in the future.

Chapter 2

Mathematical Framework

This chapter will provide the mathematical tools required for a new adaptive grid refinement method. First, a problem formulation is presented in Section 2.1, followed by a description of the Polynomial Chaos method in Section 2.2. In Section 2.3.1 the ANOVA decomposition is introduced. This is followed by a definition of the abscissae and weights of the Clenshaw-Curtis quadrature rule in Section 2.4. Sections 2.5 and 2.6 show how to construct sparse grids and how to perform integration and interpolation on these grids. Then, Section 2.7 shows how to approximate Sobol variances using sparse grids. Finally, Section 2.8 will describe two adaptive grid refinement approaches: the existing Gerstner and Griebel approach, and the new method based on Sobol variances.

2.1 Problem formulation

The problem under consideration is the quantification of the effect of parametric uncertainty on the response of the model. In other words, the uncertainty in the input parameters is propagated through the model to determine the distribution of the output.

Let us consider a physical model represented by a deterministic function $\mathbf{y} = \mathcal{M}(\mathbf{x})$, where $\mathbf{x} = [x_1, \ldots, x_N]^T \in \mathbb{R}^N$, $N \geq 1$ is the vector of input variables, and $\mathbf{y} = [y_1, \ldots, y_Q]^T \in \mathbb{R}^Q$, $Q \geq 1$ is the vector containing the quantities of interest of the model [19]. Vector $\mathbf{y}$ is also known as the model response. Now assume input vector $\mathbf{x}$ is affected by uncertainty, which is represented by a random vector $\mathbf{X}$ with prescribed joint probability density function (PDF) $f_{\mathbf{X}}(\mathbf{x})$. It is assumed that the components $\{X_1, \ldots, X_N\}$ are independent, such that $f_{\mathbf{X}}(\mathbf{x}) = \Pi_{i=1}^{N} f_{X_i}(x_i)$ is the marginal PDF of X_i. The model response can then be seen as the random vector $\mathbf{Y}$ defined by:

$$\mathbf{Y} = \mathcal{M}(\mathbf{X}) \tag{2.1}$$

where $\mathcal{M}$ can be seen as a differential operator which contains space and time derivatives.

2.2 Polynomial Chaos

Recall the problem formulation (2.1):

$$\mathbf{Y} = \mathcal{M}(\mathbf{X}) \tag{2.2}$$

where $\mathbf{Y}$ is the model response, $\mathcal{L}$ is a (possibly nonlinear) differential operator which contains space and time derivatives and $\mathbf{X}$ contains the input variables, which are affected by uncertainty. The polynomial chaos expansion of $\mathbf{Y}(\mathbf{X})$ is then:

$$\mathbf{Y}(\mathbf{X}) := \sum_{j=0}^{\infty} a_j \Phi_j(\mathbf{X}) \tag{2.3}$$

which is a spectral expansion in the vector of independent basis random variables $\mathbf{X}$ with random polynomial basis $\{\Phi_j\}$. The solution is split into a deterministic part, coefficients a_j, and a stochastic part, the polynomial chaoses, $\Phi_j(\mathbf{X})$. In practice, the number of terms in (2.3) is truncated:

$$\mathbf{Y}(\mathbf{X}) := \sum_{j=0}^{N} a_j \Phi_j(\mathbf{X}) \tag{2.4}$$

The polynomial basis $\{\Phi_j(\mathbf{X})\}_{j=0}^{N}$ of the Polynomial Chaos expansion in (2.3) satisfies the following orthogonality condition:

$$\langle \Phi_j(\mathbf{X}), \Phi_k(\mathbf{X}) \rangle = \langle \Phi_j^2(\mathbf{X}) \rangle \delta_{jk} \tag{2.5}$$

where $j, k = 0, 1, \ldots, N$, δ_{jk} is the Kronecker delta and $\langle , \rangle$ denotes the inner product.

$$\langle \Phi_j(\mathbf{X}), \Phi_k(\mathbf{X}) \rangle = \int_{S_\mathbf{X}} \Phi_j(\mathbf{X}) \Phi_k(\mathbf{X}) w(\mathbf{X}) d\mathbf{X} \tag{2.6}$$

where $w(\mathbf{X})$ is the weighting function corresponding to the polynomials $\{\Phi_j(\mathbf{X})\}_{j=0}^{N}$ and $S_\mathbf{X}$ is the support of $\mathbf{X}$.

The term Polynomial Chaos (PC) originates from the work of Wiener [6], using Hermite polynomials as an orthogonal basis to represent stochastic processes with Gaussian random variables. Note that there are two main categories of PC-based methods: intrusive and non-intrusive. In this context, intrusive implies rewriting the governing equations of the model, in other words rewriting the code of the simulation software. Non-intrusive methods treat the model as a black box and only use deterministic samples to obtain information on the system. Non-intrusive methods PC-based methods are more common since they can be used with commercial software, see for example [30],[26]. There are also cases of intrusive PC being implemented, but this requires the source code to be available. An example of intrusive PC being implemented can be found in [32]. For the rest of this thesis, unless explicitly stated otherwise, only non-intrusive PC is considered.

Choosing the optimal polynomials for common weighting functions leads to the Wiener-Askey scheme. To name a few examples for continuous weighting functions [7],[8]:

- Hermite polynomials are associated with the Gaussian distribution.
- Legendre polynomials to the uniform distribution.
- Laguerre polynomials with the Gamma distribution.
- Jacobi polynomials to the Beta distribution.

The nodes and weights are chosen from the Gauss quadratures corresponding to the polynomial basis, for more information on this see the Golub-Welsch algorithm [10].

For input distributions which are not in the Wiener-Askey scheme it is still possible to construct a orthogonal polynomial basis numerically as shown in [9]. Alternatively, Gram-Schmidt orthogonalization has also been used [31]. In this case, the nodes and weights can be determined using the Stieltjes procedure combined with the Lanczos algorithm, as shown in [9]. Once the orthogonal polynomial basis and the corresponding weights are known, it is possible to determine the coefficients a_j from (2.4) by taking samples at the collocation points defined by the appropriate quadrature rule. Note that for higher-dimensional problems, a tensor grid is formed from the 1-D rules. This leads to the following system:

$$\begin{bmatrix} \Phi_0(X_0) & \Phi_1(X_0) & \dots & \Phi_N(X_0) \\ \Phi_0(X_1) & \Phi_1(X_1) & \dots & \Phi_N(X_1) \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_0(X_N) & \Phi_1(X_N) & \dots & \Phi_N(X_N) \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} \mathbf{Y}(X_0) \\ \mathbf{Y}(X_1) \\ \vdots \\ \mathbf{Y}(X_N) \end{pmatrix} \tag{2.7}$$

Since the only unknowns are the coefficients a_j for $j = 0, \dots, N$, they can be extracted from (2.7). The coefficients are only approximations of the real polynomial expansion coefficients. The more accurate the approximation using the basis polynomials, the closer the coefficients will be to the actual values. The solution $\mathbf{Y}$ from (2.1) is now fully defined; the mean $\mu_{\mathbf{Y}}$ and variance $\sigma^2_{\mathbf{Y}}$ are then [26]:

$$\mu_{\mathbf{Y}} = a_0 \tag{2.8}$$

$$\sigma^2_{\mathbf{Y}} = \sum_{j=1}^{N} a_j^2 \Phi_j^2(\mathbf{X}). \tag{2.9}$$

2.3 Global Sensitivity of Functions: Sobol Indices

Consider a multivariate real function $f : \mathbb{R}^N \to \mathbb{R}$. Sobol indices are one measure of the *global* sensitivity of the function with respect to its arguments $\mathbf{x}$. Usually sensitivity measures are *local*, that is, they concern themselves only with the linearized behaviour of f function at a reference point $\mathbf{x}_0 \in \mathbb{R}^N$. For instance local sensitivities might be defined as:

$$S_i := \sigma_i \left. \frac{\partial f}{\partial x_i} \right|_{\mathbf{x}_0} \qquad i \in \{1, \dots, N\}, \tag{2.10}$$

where σ_i is a measure of how much variable x_i is expected to vary. This definition ignores the behavour of f away from $\mathbf{x}_0$. For example $f(x) = 10^{10}x^2$ at $x_0 = 0$ is considered to be "insensitive" to x, and $f(x) = H(x)$, (with $H(\cdot)$ the Heaviside function) is "insensitive" to x everywhere (except possibly at the origin). Furthermore S provides no information on the effect of interactions of multiple variables on f.

Global sensitivity measures attempt to address these limitations. The first step is to specify what is meant by "global". In the case of variance-based sensitivity indices (of which Sobol indices are an exmaple) this is achieved by defining a probability density function (pdf) for each input variable, specifying the range of that variable which is of interest:

$$\rho(x_1), \ldots, \rho(x_N),$$

with the corresponding random-variables denoted $\mathbf{X} = (X_1, \ldots, X_N)$. These are comparable in purpose to σ_i in the local case. To continue the derivation of Sobol indices, the Analysis of Variance (ANOVA) decomposition must be introduced.

2.3.1 ANOVA Decomposition

Assume that $f(\mathbf{x})$ is square-integrable[1] with respect to the metric generated by $\mathbf{X}$. Furthmore let $X_1, \ldots, X_N \sim \mathcal{U}(0,1)$ be independently uniformly distributed on $[0,1]$. Any input space can be transformed onto this unit hypercube, so there is no loss of generality. Then $f(\mathbf{X})$ is a random-variable with finite variance, which we represent in the form:

$$f(\mathbf{X}) = f_\emptyset + \sum_{s=1}^{N} \sum_{1 \le i_1 < \cdots < i_s \le N} f_{i_1 \ldots i_s}(X_{i_1}, \ldots, X_{i_s}). \tag{2.11}$$

Or in long-hand:

$$\begin{aligned} f(\mathbf{X}) = {} & f_\emptyset \\ & + f_1(X_1) + \cdots + f_N(X_N) \\ & + f_{12}(X_1, X_2) + f_{13}(X_1, X_3) + \cdots + f_{N-1N}(X_{N-1}, X_N) \\ & + f_{123}(X_1, X_2, X_3) + \ldots \\ & + \ldots \\ & + f_{1\ldots N}(X_1, \ldots, X_N) \end{aligned}$$

The most convenient form is the third form:

$$f(\mathbf{X}) = \sum_{u \subseteq \mathcal{U}} f_u(X_u) \tag{2.12}$$

where u is a *multi-index*, $\mathcal{U} = \{1, 2, \ldots, d\}$, and the sum is over all subsets of $\mathcal{U}$. Now X_u is the set of random-variables whose indices lie in u, and f_u is the component function only dependent on X_u. If it is true that – in physical and engineering models – low-order

[1] Incidentally, a much weaker condition on f than that required by (2.10).

interactions of variables have the main effect on the output, and if this is captured by the decomposition above, then we should be able to truncate the sum without substancial loss of fidelity. Compare this to sparse grids, in which high polynomial order interactions of multiple variables are also preferentially eliminated.

This formula is called an ANOVA decomposition if:

$$\int f_u(x_{u_1}, \ldots, x_{u_s}) d\rho(x_i) = 0 \quad \text{for } i \in u. \tag{2.13}$$

This implies:

$$\mathbb{E} f_u := \int f_u(x_u) d\rho(x_u) = 0 \quad \text{for } u \neq \emptyset, \tag{2.14}$$

i.e. all f_u have zero mean, with the exception of $f_\emptyset$, and

$$\text{cov}(f_u, f_v) = \int f_u(x_u) f_v(x_v) d\rho(x_u \cup x_v) = 0 \quad \text{for } u \neq v, \tag{2.15}$$

i.e. f_u, f_v are orthogonal. Let u' is the complement of u, so that $\{u \cup u'\} = \mathcal{U}$ and $\{u \cap u'\} = \emptyset$. These properties are satisfied when the component functions f_u are defined as:

$$\begin{aligned} f_\emptyset &= \int f(\mathbf{x}) d\rho(\mathbf{x}), \\ f_u &= \int f(\mathbf{x}) d\rho(x_{u'}) - \sum_{w \subset u} f_w(x_w) \quad \text{for } u \neq \emptyset, \end{aligned}$$

which can be rewritten in terms of conditional expectations:

$$\begin{aligned} f_\emptyset &= \mathbb{E} f, \\ f_i &= \mathbb{E}(f|X_i) - f_0, \\ f_{ij} &= \mathbb{E}(f|X_i, X_j) - f_i - f_j - f_0, \\ &\ldots \end{aligned}$$

at which point the terms can be interpreted. It is evident that f_i captures the effect of varying X_i alone, with all other variables integrated out. And f_{ij} captures the effect of varying X_i and X_j simoultaneously, *minus* the effect of their individual variations. And so on.

The variances of these terms are therefore our desired sensitivity measures:

$$D_u := \text{var}(f_u) = \int f_u^2 d\rho(x_u), \tag{2.16}$$

which implies that all Sobol variances are non-negative. A little algebra shows that D_u simplifies to:

$$\begin{aligned} D_u &:= \int \left(\int f(x) d\rho(x_{u'}) \right)^2 d\rho(x_u) - \sum_{w \subset u} \int \left(f_w(x_w) \right)^2 d\rho(x_u), \\ &= \int \left(\int f(x) d\rho(x_{u'}) \right)^2 d\rho(x_u) - \sum_{w \subset u} D_w, \end{aligned} \tag{2.17}$$

which is a readily computable expression for D_u, and allows computation in order of increasing interaction, first D_i, then D_{ij}, then D_{ijk} etc. As well as $D_u \geq 0$ we have

$$D := \text{var}(f) = \sum_{u \subseteq \mathcal{U}} D_u,$$

i.e. the variance of f has been decomposed into the effects due to individual combinations of variables. This property suggests the definition of Sobol *indices*, which are just normalized Sobol variances:

$$S_u := \frac{D_u}{D}.$$

2.4 Clenshaw-Curtis Quadrature

As mentioned in the literature review (Section 1.2), there are many quadrature rules with corresponding basis functions which can be used. For this project, the Clenshaw-Curtis quadrature rule is used in combination with Lagrange polynomials.

Quadrature rules are typically used in combination with polynomial basis functions, see [8]. However, it can be convenient to use linear basis functions when dealing with non-smooth functions. This is why Section 2.4.2 will use polynomial basis functions, and Section 2.4.3 will discuss the use of linear hat basis functions. But first, a definition of the abscissae is given in Section 2.4.1.

2.4.1 Abscissae

The Clenshaw-Curtis quadrature rule is used to approximate an integral as:

$$\int_{-1}^{1} f(x)\beta(x)dx \approx \sum_{k=1}^{n} w_k f(x_k), \tag{2.18}$$

where $\beta(x)$ is a weighting function that corresponds to a probability density function, x_k are the abscissae of the Clenshaw-Curtis rule, and w_k are the corresponding weights. Since $\beta(x)$ is a PDF, by definition $\int_{-1}^{1} \beta(x)dx = 1$. The abscissae of the n-point Clenshaw-Curtis rule in the reference interval $[-1, 1]$ are defined as [15]:

$$x_k := \begin{cases} 0 & \text{if } l = 1, \\ cos\left(\frac{(k-1)\pi}{n-1}\right), k = 1, ..., n & \text{if } l > 1. \end{cases} \tag{2.19}$$

where $n = 2^{l-1} + 1$. The computation of the weights is discussed in the following two sections.

2.4.2 Weights for Polynomial Basis Functions

This section will deal with the computation of the weights for polynomial basis functions. First, a uniform input parameter distribution is considered, where each point is equally important. This distribution can be utilized if there is no additional information on the input. It is possible to obtain weights corresponding to other distributions.

We will use an algorithm from the work of Sommariva [15]. It uses a discrete cosine transform to compute the weights of Clenshaw-Curtis and Féjer quadrature rules for general weight functions β. Sommariva shows that the Clenshaw-Curtis weights can be computed as long as the weighted moments of the Chebyshev polynomials are known. These are defined as:

$$\gamma_k := \int_{-1}^{1} T_k(x)\beta(x)dx, \tag{2.20}$$

where $\beta(x)$ is the weighting function and $T_k(x)$ the Chebyshev polynomials, which can be found using [1]:

$$\begin{cases} T_0(x) = 1, \\ T_1(x) = x, \\ T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x). \end{cases} \tag{2.21}$$

The first few Chebyshev polynomials are given here:

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ T_5(x) &= 16x^5 - 20x^3 + 5x \end{aligned}$$

Since an analytical expression for the Chebyshev polynomials is available, only the weighting function $\beta(x)$ is required to compute the weighted moments for the Sommariva algorithm. Two common weighting functions will be used in this thesis: the uniform distribution and the truncated normal distribution.

Uniform Distribution

For the uniform distribution, the weighting function is set $\beta(x) = \frac{1}{2}$ for $x \in [-1, 1]$, which simplifies equation 2.20 to the integrals of the Chebyshev polynomials. Note that other works dealing with Clenshaw-Curtis quadrature rules tend to use $\beta(x) = 1$ for the uniform distribution, for example [33]. The weights are the same up to a constant, in this case $\frac{1}{2}$. The reason for choosing $\beta(x) = \frac{1}{2}$ here is to ensure that $\int_{-1}^{1} \beta(x)dx = 1$. In other words $\beta(x)$ is the probability density function of x. It is straightforward to compute the weighted moments

as:

$$\begin{aligned}
\gamma_0 &= \int_{-1}^{1} T_0(x)dx = \int_{-1}^{1} dx = 2 \\
\gamma_1 &= \int_{-1}^{1} T_1(x)dx = \int_{-1}^{1} xdx = 0 \\
\gamma_2 &= \int_{-1}^{1} T_2(x)dx = \int_{-1}^{1} 2x^2 - 1dx = -\frac{2}{3} \\
\gamma_3 &= \int_{-1}^{1} T_3(x)dx = 0 \\
\gamma_4 &= \int_{-1}^{1} T_4(x)dx = -\frac{2}{15} \\
&\vdots
\end{aligned} \tag{2.22}$$

or in general:

$$\gamma_k = \begin{cases} 2 & \text{if k is 0} \\ 0 & \text{if k is odd} \\ -\frac{2}{2^k-1} & \text{if k is even} \end{cases} \tag{2.23}$$

Using the vector of γ_k as input for the Sommariva algorithm, the weights for the uniform distribution are then calculated.

Truncated Normal Distribution

For the second distribution considered, the truncated normal distribution, the weighting function is set to $\beta(x) = \mathcal{N}_T(x)$ such that:

$$\int_{-1}^{1} f(x)\mathcal{N}_T(x)dx = \frac{1}{\int_{-1}^{1}\mathcal{N}(x)dx}\int_{-1}^{1} f(x)\mathcal{N}(x)dx \tag{2.24}$$

where $f(x)$ can be any function and $\mathcal{N}(x)$ is the normal distribution:

$$\mathcal{N}(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.25}$$

It is convenient to rewrite the integral by splitting up the terms of the Chebyshev polynomial. Each of these integral can be computed separately and subsequently summed up to obtain the actual results. For example:

$$\gamma_2 = \int_{-1}^{1} T_2(x)\mathcal{N}_T(x)dx = \int_{-1}^{1}(2x^2-1)\mathcal{N}_T(x) \tag{2.26}$$

$$= 2\int_{-1}^{1} x^2\mathcal{N}_T(x) - \int_{-1}^{1}\mathcal{N}_T(x) \tag{2.27}$$

This implies that to compute the modified moments γ_k, the integrals of the form:

$$I_k = \int_{-1}^{1} x^k\mathcal{N}(x) \tag{2.28}$$

are required for $k = 0, ..., k$. For this project, the normal distribution is truncated at the 95% range, which corresponds to approximately $\mu \pm 2\sigma$, though it is possible to choose any value between 0 and 1. It is trivial to derive (2.28) from (2.24) since $\int_{-1}^{1} \mathcal{N}(x)dx$ is known to be 0.95; it is also by definition equal to I_0. Note that the truncated normal distribution is a symmetric function and the Chebyshev polynomials alternate between even and odd functions. It follows that half of the integrals will be of odd functions, which by definition are equal to zero. This is confirmed when computing the integrals. Performing partial integration yields the following recursive expression:

$$\begin{aligned} I_0 &= 0.95 \\ I_k &= 0 \qquad\qquad\qquad\qquad \text{if k is odd} \\ I_k &= -\frac{\sqrt{2}\sigma}{\sqrt{\pi}} e^{\frac{1}{2\sigma^2}} + (n-1)\sigma^2 I_{n-2} \quad \text{if k is even.} \end{aligned} \tag{2.29}$$

The expression above can also be written analytically, but it is shorter and more convenient to write it in a recursive way. Note that μ has been set to zero in equation (2.29) since $\mu = 0$ when dealing with the range $[-1, 1]$.

Finally, the weighted moments γ_k can be calculated by combining (2.29) with the correct coefficients for each Chebyshev polynomial. The first few are given below. Recall that:

$$\int_{-1}^{1} T_k(x)\mathcal{N}_T(x)dx = \frac{1}{I_0}\int_{-1}^{1} T_k(x)\mathcal{N}(x)dx, \tag{2.30}$$

such that the weighted moments γ_k become:

$$\begin{aligned} \gamma_0 &= \int_{-1}^{1} T_0(x)\mathcal{N}_T(x)dx = \frac{I_0}{I_0} = 1, \\ \gamma_1 &= \int_{-1}^{1} T_1(x)\mathcal{N}_T(x)dx = 0, \\ \gamma_2 &= \int_{-1}^{1} T_2(x)\mathcal{N}_T(x)dx = \frac{2I_2 - I_0}{I_0}, \\ \gamma_3 &= \int_{-1}^{1} T_3(x)\mathcal{N}_T(x)dx = 0, \\ \gamma_4 &= \int_{-1}^{1} T_4(x)\mathcal{N}_T(x)dx = \frac{4I_3 - 3I_1}{I_0}, \\ &\vdots \end{aligned} \tag{2.31}$$

Using this vector containing γ_k for $k = 0, ..., N$, the Sommariva algorithm can be used to obtain the weights for the truncated normal distribution.

2.4.3 Weights for Linear Hat Basis Functions

Instead of using polynomials as the basis functions, it is also possible to use linear hat functions. When dealing with a function which shows non-smooth behavior, it is convenient to

have this option available. The convergence using linear hat functions is not spectral, which implies it will in general be slower than for polynomials, which is why it is only used as a backup. The definition of the $i-$th linear hat function is given by:

$$H_i(x) = \begin{cases} ax + b & \text{if } x \in [x_{i-1}, x_i] \\ cx + d & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

where H are the linear hat function, n is the number of nodes and x_i are nodes, in this case Clenshaw-Curtis abscissae. H is equal to 1 at the i-th node. Coefficients a, b, c and d can

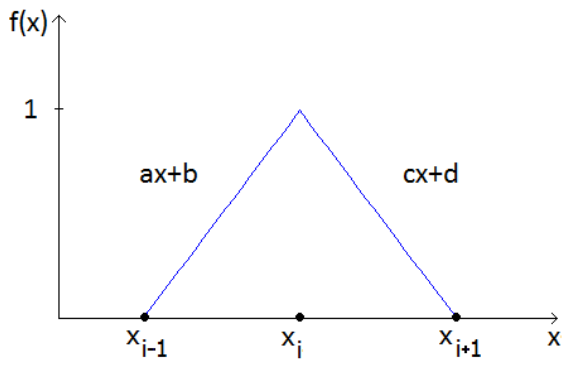


Figure 2.1: Linear hat function.

be determined from the geometry of the hat function:

$$a = \frac{1}{x_i - x_{i-1}} \tag{2.33}$$

$$b = \frac{x_{i-1}}{x_{i-1} - x_i} \tag{2.34}$$

$$c = \frac{1}{x_i - x_{i+1}} \tag{2.35}$$

$$d = \frac{x_{i+1}}{x_{i+1} - x_i}. \tag{2.36}$$

We are trying to approximate $f(x)$ by linear hat functions:

$$f(x) \approx \sum_{i=1}^{n} f(x_i) H_i(x). \tag{2.37}$$

Taking the integral $f(x)$ multiplied by some weighting function $\beta(x)$:

$$\int_{-1}^{1} f(x)\beta(x)dx \approx \sum_{i=1}^{n} f(x_i) \int_{-1}^{1} H_i(x)\beta(x)dx = \sum_{i=1}^{n} f(x_i) w_i. \tag{2.38}$$

The calculation of the weights is done by computing the following integral:

$$w_i = \int_{-1}^{1} H_i(x)\beta(x)dx = \int_{x_{i-1}}^{x_{i+1}} H_i(x)\beta(x)dx, \tag{2.39}$$

since $H_i(x)$ is 0 outside $[x_{i-1}, x_{i+1}]$. Splitting this up into two parts for convenience gives:

$$w_i = \int_{x_{i-1}}^{x_i} (ax+b)\beta(x)dx + \int_{x_i}^{x_{i+1}} (cx+d)\beta(x)dx, \tag{2.40}$$

where a, b, c and d are given in (2.33).

Uniform Distribution

The weighting function is set to $\beta(x) = \frac{1}{2}$, as was done in Section 2.4.2. In this case it becomes straightforward to evaluate (2.40). Alternatively from the geometry of the linear hat function it is clear that we are simply looking for the area of a triangle, for which the formula is $1/2 \times base \times height$. From the definition of the hat function (2.32) the base is known to be from x_{i-1} to x_{i+1}, and the height $h = 1$ such that:

$$w_i = \frac{(x_{i+1} - x_{i-1})}{2}. \tag{2.41}$$

Truncated Normal Distribution

For the truncated normal input distribution, (2.40) now includes the truncated normal distribution. Splitting up the terms yields:

$$w_i = a\int_{x_{i-1}}^{x_i} x\mathcal{N}_T(x)dx + b\int_{x_{i-1}}^{x_i} \mathcal{N}_T(x)dx + c\int_{x_i}^{x_{i+1}} x\mathcal{N}_T(x)dx + d\int_{x_i}^{x_{i+1}} \mathcal{N}_T(x)dx, \tag{2.42}$$

where the fact that $H_i(x)$ is 0 outside $[x_{i-1}, x_{i+1}]$ has been used. Recalling from (2.24):

$$\int_{-1}^{1} f(x)\mathcal{N}_T(x)dx = \frac{1}{\int_{-1}^{1}\mathcal{N}(x)dx}\int_{-1}^{1} f(x)\mathcal{N}(x)dx, \tag{2.43}$$

where $f(x)$ can be any function and $\mathcal{N}(x)$ is the normal distribution given in (2.24). The choice is again made to truncate at the 95% range. The integrals in (2.42) can then be split up into two categories: $\int_{x_s}^{x_{s+1}} x\mathcal{N}(x)dx$ and $b\int_{x_s}^{x_{s+1}} \mathcal{N}(x)dx$. To compute the integrals of the second kind, use the well-known fact that:

$$\int_{-\infty}^{x} \mathcal{N}(x) = \frac{1}{2}\left(1 + erf\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right), \tag{2.44}$$

such that:

$$\int_{x_s}^{x_{s+1}} \mathcal{N}(x)dx = \int_{-\infty}^{x_{s+1}} \mathcal{N}(x)dx - \int_{-\infty}^{x_s} \mathcal{N}(x)dx \tag{2.45}$$

$$= \frac{1}{2}\left(1 + erf\left(\frac{x_{s+1}-\mu}{\sigma\sqrt{2}}\right)\right) - \frac{1}{2}\left(1 + erf\left(\frac{x_s-\mu}{\sigma\sqrt{2}}\right)\right). \tag{2.46}$$

For the integrals of the first kind, it is actually deceptively easy to compute the integral using partial integration. Substituting the $\mathcal{N}(x)$ notation by its definition:

$$\int_{x_s}^{x_{s+1}} x\mathcal{N}(x)dx = \int_{x_s}^{x_{s+1}} x\frac{1}{\sigma\sqrt{2\pi}}e^{\frac{(x-\mu)^2}{2\sigma^2}}dx = \left[\frac{\sigma}{\sqrt{2\pi}}e^{-\frac{1}{2\sigma^2}x^2}\right]_{x_s}^{x_{s+1}}, \tag{2.47}$$

where the fact that $\mu = 0$ has been used. The weights in (2.42) can now be determined by combining the above with (2.33).

2.5 Sparse Grid Construction

The sparse grid method is a numerical discretization technique which was first introduced by Smolyak [20]. It uses tensor products of 1-dimensional quadrature rules to construct an n-dimensional grid which can be used for multivariate integration. The difference with the sampling grid of the PC method is that specific tensor product combinations are used to construct a grid, instead of making a full tensor grid. Using the notation of Gerstner and Griebel [18] to describe the numerical integation of functions $f^{(d)}(x)$ from a function class $\mathcal{F}$ over the d-dimensional hypercube $\Omega = [-1, 1]^d$:

$$If^{(d)} := \int_{\Omega} f^{(d)}(x)dx \tag{2.48}$$

by a sequence of $n_l^{(d)}$-point quadrature formulas with level $l \in \mathbb{N}$ and $n_l^{(d)} < n_l + 1^{(d)}$:

$$Q_l f^{(d)} := \sum_{i=1}^{n_l^{(d)}} w_{li} f^{(d)}(x_{li}) \tag{2.49}$$

using weights w_{li} and abscissae x_{li}. The construction of the sparse grid begins with a series of 1-dimensional quadrature formulas for a univariate function $f^{(1)}$:

$$Q_l f^{(1)} := \sum_{i=1}^{n_l^{(1)}} w_{li} f^{(1)}(x_{li}) \tag{2.50}$$

The difference formulas are defined by:

$$\Delta Q_k f^{(1)} := (Q_k - Q_{k-1}) f^{(1)} \text{ with} \tag{2.51}$$

$$Q_0 f^{(1)} := 0 \tag{2.52}$$

For index set $\mathcal{X} \in \mathbb{N}^d$, the conventional sparse grid quadrature method for d-dimensional functions $f^{(d)}$ for a given level $l \in \mathbb{N}$ is:

$$Q_{\mathcal{X}} f^{(d)} := \sum_{\mathbf{k} \in \mathcal{X}} (\Delta Q_{k_1} \otimes \cdots \otimes \Delta Q_{k_d}) f^{(d)} \tag{2.53}$$

For a standard sparse grid, all possible tensor product combinations of one-dimensional quadrature formulas are considered which satisfy the criterion $\mathcal{X} = \{\mathbf{k} : \sum k_i \leq l + d - 1, i = 1, \ldots, d\}$, such that the index set $\mathcal{X}$ describes a unit simplex. Here, k_i indicate the multi-indices in the admissible index set. To give a visual representation of the sparse grid construction the tensor products can be plotted as blocks, Figure 2.5 shows an example of a level 3 simplex in 2D and 3D. Each of the blocks in the simplex represents one sparse grid component, which is a tensor product of 1-d quadrature rules. A visual representation of the 2-dimensional tensor product of a 1-point rule with a 3-point rule is shown in Figure 2.5. Adding all the components corresponding to the blocks in the simplex produces a conventional sparse grid, as shown in Figure 2.4 for level $l = 3$. The Smolyak procedure can be used with any quadrature rule, be it nested or non-nested. Each of the previous figures has used a nested rule.

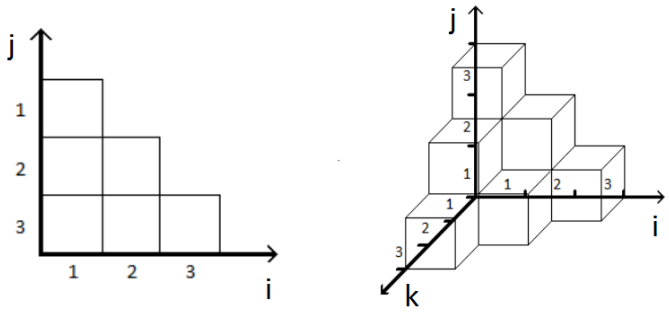


Figure 2.2: A level 3 sparse grid simplex in 2 (left) and 3 (right) dimensions.

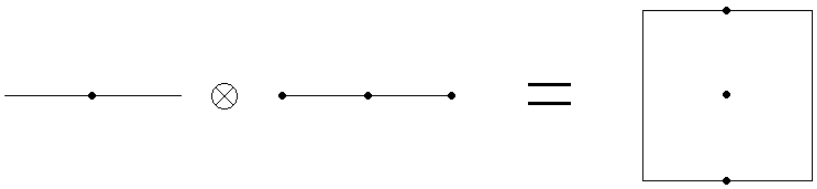


Figure 2.3: Tensor product of a 1-point rule and a 3-point rule, resulting in a 2D grid containing 3 points.

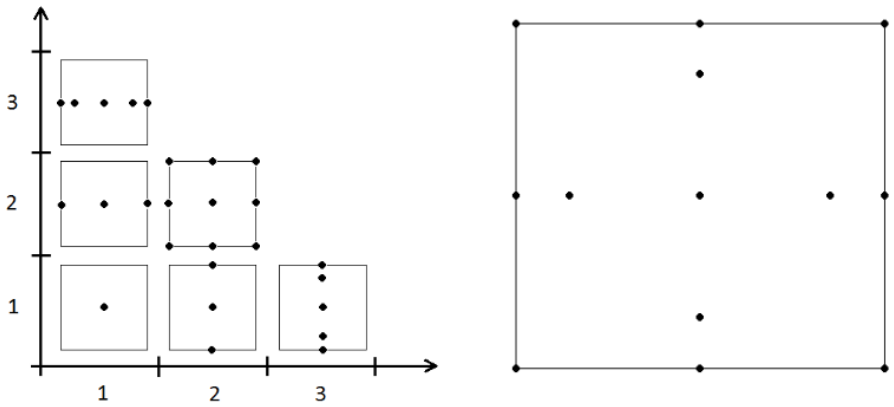


Figure 2.4: Sparse grid components in the 2-dimensional level 3 simplex (left) and the resulting sparse grid (right). In this figure the grid is constructed using nested quadrature rules.

2.6 Interpolation on Sparse Grids

This section will go over two types of interpolation on sparse grids: interpolation using Lagrange polynomials in Section 2.6.1 and using linear hat functions in Section 2.6.2.

2.6.1 Lagrange Polynomial Basis Functions

The i-th Lagrange polynomial associated with abscissae x_k is defined as [29]:

$$L_i(x) = \frac{\Pi_{j \neq i}(x - x_j)}{\Pi_{j \neq i}(x_i - x_j)} \tag{2.54}$$

x_i and x_i are sample points. Essentially, each Lagrange polynomial has value 1 at one of the abscissae and has value 0 at all the others. Below, Lagrange polynomials through 3 points ($x = -1, 0$ and 1) are shown. By multiplying the Lagrange polynomials with the

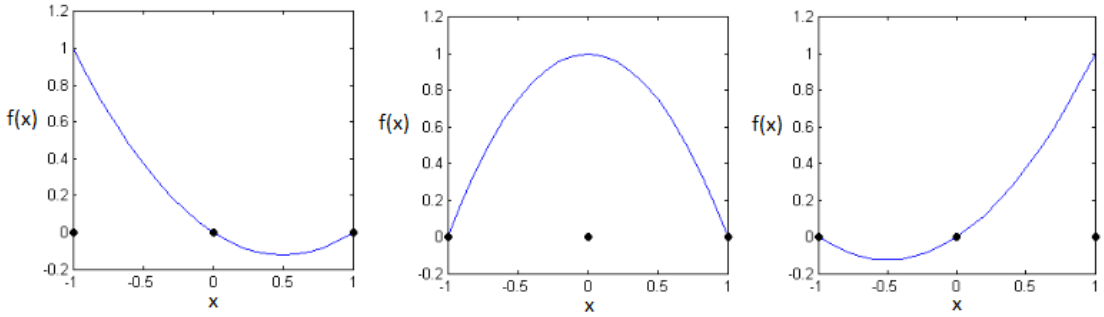


Figure 2.5: Lagrange polynomials going through abscissae $x = -1, 0$ and 1.

corresponding function values, and adding them together, an interpolating polynomial is obtained which passes through all the function values. The order of the resulting polynomial is equal to $n - 1$.

$$f(x) \approx P(x) = \sum_{i=1}^{n} f(x_i) L_i(x) \tag{2.55}$$

As an example, Figure 2.6 shows the polynomial going through $(-1, 3)$, $(0, 1)$ and $(1, 2)$ as the sum of the three Lagrange polynomials multiplied by their respective function values.

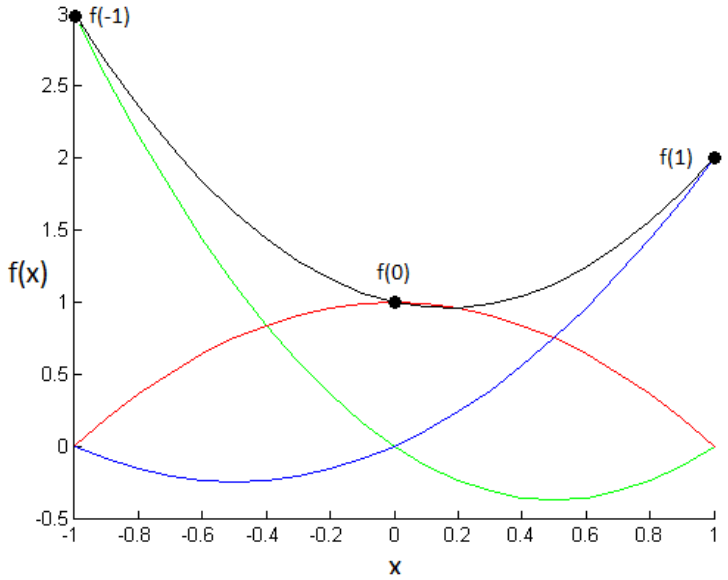


Figure 2.6: Polynomial interpolation through $(-1, 3)$, $(0, 1)$ and $(1, 2)$ as the sum of 3 Lagrange polynomials multiplied by the respective function values.

2.6.2 Linear Hat Basis Functions

The i-th linear hat function with associated abscissa ξ_i is given by:

$$H_i(x) = \begin{cases} ax + b & \text{if } x \in [x_{i-1}, x_i] \\ cx + d & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases} \tag{2.56}$$

where n is the number of sample points and ξ are the sample points. Put simply, it connects the center points x_{k_i} with its two neighbours through a straight line. It is similar to the Lagrange polynomial in the fact that it has value 1 at one of the abscissae and 0 at all the others. Because of this, the formula for the approximation of a function using linear hat functions has the same form as equation 2.55:

$$f(x) \approx \sum_{i=1}^{n} f(x_i) H_i(x) \tag{2.57}$$

To illustrate the use of linear hat functions, they are plotted through $x = -1, 0$ and 1 in Figure 2.7. The linear interpolation through $(-1, 3)$, $(0, 1)$ and $(1, 2)$ is shown in Figure 2.8. This can be compared to Figure 2.6, which shows the Lagrange polynomial interpolation through the same points.

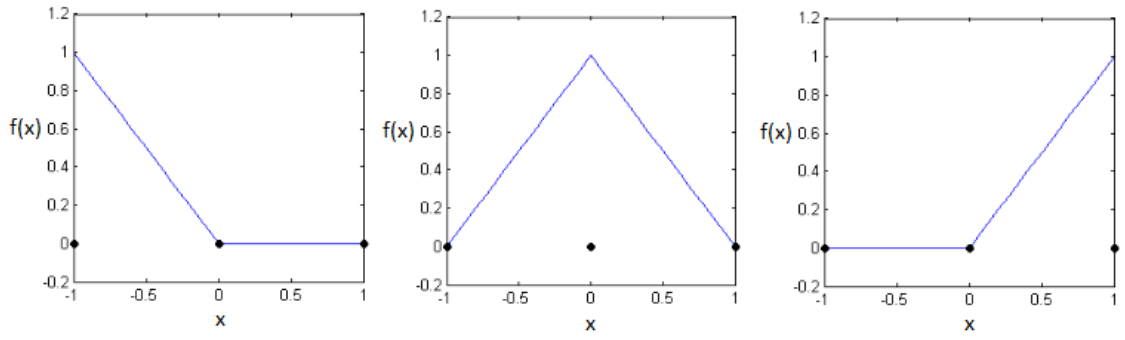


Figure 2.7: Linear hat functions going through abscissae $x = -1, 0$ and 1.

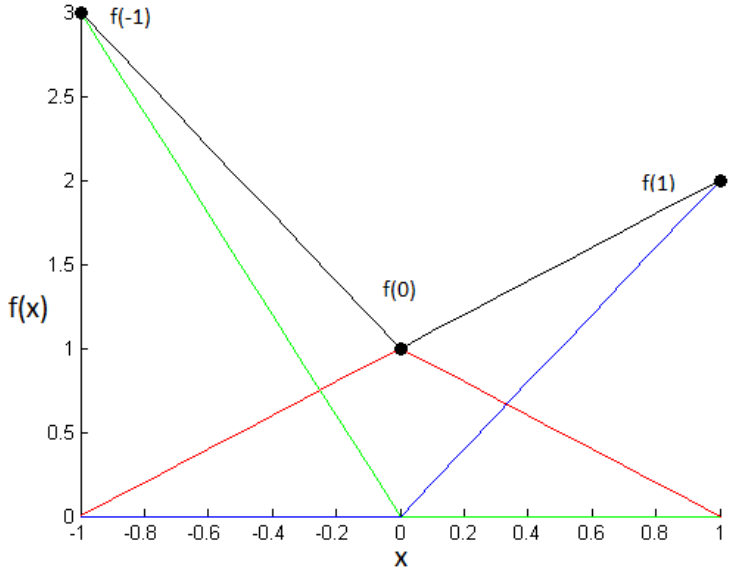


Figure 2.8: Linear interpolation through $(-1, 3)$, $(0, 1)$ and $(1, 2)$ as the sum of 3 linear hat functions multiplied by the respective function values.

2.6.3 Sparse Grid Interpolation

Both 1-dimensional interpolation methods methods can be extended to n dimensions. The procedure is similar to the extension of 1-D quadrature rules to n dimensions shown in Section 2.5. The notation below only shows the usage of Lagrange polynomials. To do the same thing for linear interpolation, the Lagrange polynomials are replaced by linear hat functions.

The goal is to find an approximation of a function $f^{(d)}(x)$ using the 1-D interpolation defined in the previous subsections:

$$P_l f^{(1)} := \sum_{i=1}^{n_l^{(1)}} f^{(1)}(x_i) L_i(x) \tag{2.58}$$

where P_l denotes that this is an approximation of f and n_l is the number of points in the quadrature rule. The other symbols are defined in Section 2.6.1. Now define the difference formulas as:

$$\Delta P_k f^{(1)} := (P_k - P_{k-1}) f^{(1)} \text{ with} \tag{2.59}$$

$$P_0 f^{(1)} := 0 \tag{2.60}$$

For $\mathcal{X} \in \mathbb{N}^d$, the sparse grid interpolation for d-dimensional functions $f^{(d)}$ for a given level $l \in \mathbb{N}$ is:

$$P_l f^{(d)} := \sum_{\mathbf{k} \in \mathcal{X}} (\Delta P_{k_1} \otimes \cdots \otimes \Delta P_{k_d}) f^{(d)} \tag{2.61}$$

For a standard sparse grid, all possible tensor product combinations of one-dimensional quadrature formulas are considered which satisfy the criterion $\mathcal{X} = \{\mathbf{k} : \sum k_i \leq l + d - 1, i = 1, \ldots, d\}$, such that the index set $\mathcal{X}$ describes a unit simplex.

2.7 Numerical Approximation of Sobol Variances

To compute the Sobol variances on a sparse grid, the following integral is evaluated, recalling from (2.17):

$$D_u := \int \left(\int f(x) d\rho(x_{u'}) \right)^2 d\rho(x_u) - \sum_{w \subset u} D_w, \tag{2.62}$$

It was already shown in (2.61) that function $f(x)$ can be approximated by means of interpolation:

$$f(x) \approx P_l f^{(d)} := \sum_{\mathbf{k} \in \mathcal{X}} (\Delta P_{k_1} \otimes \cdots \otimes \Delta P_{k_d}) f^{(d)} \tag{2.63}$$

For a standard sparse grid, all possible tensor product combinations of one-dimensional quadrature formulas are considered which satisfy the criterion $\mathcal{X} = \{\mathbf{k} : |k|_1 \leq l + d - 1\}$,

such that the index set describes a unit simplex. In (2.53) the approximation of an integral of a function using a standard sparse grid has been shown to be:

$$\int f(x)dx \approx Q_l f^{(d)} := \sum_{\mathbf{k}\in\mathcal{X}} (\Delta Q_{k_1} \otimes \cdots \otimes \Delta Q_{k_d}) f^{(d)} \tag{2.64}$$

Recall the difference formulas:

$$\Delta Q_k f^{(1)} := (Q_k - Q_{k-1}) f^{(1)} \text{ with} \tag{2.65}$$

$$Q_0 f^{(1)} := 0, \tag{2.66}$$

and:

$$Q_k f^{(1)} := \sum_{i=1}^{n_k^{(1)}} w_{k_i} f^{(1)}(x_{k_i}) \tag{2.67}$$

It was shown in Section 2.4.1 that it is possible to compute the weights such that it includes an input weighting distribution. To solve (2.62), first consider the inner integral $\int f(x)d\rho(x_{u'})$. Using the information above, this can be rewritten:

$$\int f(x)d\rho(x_{u'}) \approx \int P_l f d\rho(x_{u'}) \tag{2.68}$$

$$= \int \sum_{\mathbf{k}} (\Delta P_{k_1} \otimes \cdots \otimes \Delta P_{k_d}) f d\rho(x_{u'}) \tag{2.69}$$

$$= \int \sum_{\mathbf{k_u}} \sum_{\mathbf{k'_u}} (\Delta P_{k_{u_1}} \otimes \cdots \otimes \Delta P_{k_{un}}) \otimes (\Delta P_{k_{u'_1}} \otimes \cdots \otimes \Delta P_{k_{u'_p}}) f d\rho(x_{u'}) \tag{2.70}$$

where $\mathbf{k}$ has been split up into contribution from u and u'. The subscripts of f have been dropped for simplicity. The above can be rewritten by taking into account:

$$\int f(x)dx \approx \int P_l f dx \text{ and} \tag{2.71}$$

$$\int f(x)dx \approx Q_l f \tag{2.72}$$

$$\tag{2.73}$$

such that:

$$\int f(x)d\rho(x_{u'}) \approx \sum_{\mathbf{k_u}} \sum_{\mathbf{k'_u}} (\Delta P_{k_{u_1}} \otimes \cdots \otimes \Delta P_{k_{un}}) f \otimes (\Delta Q_{k_{u'_1}} \otimes \cdots \otimes \Delta Q_{k_{u'_p}}) f \tag{2.74}$$

$$= P_l f_u \otimes Q_l f_{u'} \tag{2.75}$$

Combining this info with (2.62) gives [17]:

$$D_u \approx \int \left(P_l f_u \otimes Q_l f_{u'}\right)^2 d\rho(x_u) - \sum_{w\subset u} D_w \tag{2.76}$$

$$= (Q_l f_{u'})^2 \otimes Q_l f_u^2 - \sum_{w\subset u} D_w \tag{2.77}$$

The Sobol variances are calculated in order of increasing cardinality, so all D_w are always known from previous calculations.

In the literature review in Section 1.2 it was already mentioned that the convergence of Clenshaw-Curtis integration has been shown to be close to Gaussian integration, which can approximate a polynomial P of degree $2N-1$ with N sampling points [13]. If there are N samples of a function f, the approximation of the function goes up to a degree of maximum $2N-1$. To now approximate f^2 (of degree $2(2N-1) = 4N-2$), an additional number of samples is required. These samples are generated by interpolating the available data on a sparse grid of a higher level. The increase in number of points for each increasing level is:

$$N_{l+1} = 2N_l - 1, \tag{2.78}$$

which follows from (2.19). Here, subscript l denotes the current level. If the refined grid is a sparse grid one level higher than the current one, it could approximate f^2 up to degree:

$$2N_{l+1} - 1 = 2(2N_l - 1) - 1 \tag{2.79}$$

$$= 4N_l - 3, \tag{2.80}$$

which is not enough to integrate the polynomial approximation of f^2 exactly. Following the same principle, if the sparse is refined by 2 levels it is guaranteed to always integrate the polynomial approximation of f^2 exactly. All of this only holds when dealing with polynomial basis functions.

For the linear basis functions, the function f is approximated by piecewise linear basis functions. This means that f^2 consists of piecewise quadratic functions. By considering a grid of one level higher than the current sparse grid, these piecewise quadratic functions can be determined and used to integrate the integral of the piecewise quadratic approximation of f^2. The piecewise quadratic approximation was not implemented, meaning that the integrals are not evaluated exactly when using linear basis functions. One way to still get a reasonable approximation is to make a grid of a much higher level, using piecewise linear functions as usual. The disadvantage of this is that the calculations may show Sobol variances to be negative or inaccurate. However, it can still pose problems by interfering with the adaptive scheme, which selects refinements based on the values of the Sobol variances. This is a serious drawback of having to use linear integration to determine the Sobol variances, and if this method is developed further, this is definitely one of the things that needs to be implemented.

2.8 Adaptive Grid Refinement

This section will give a description of two adaptive grid refinement methods. For clarity, first a list of definitions for frequently used terms is given here. Most of these definitions are taken from [18]:

> Definition 1 A multi-index represents one combination of 1D quadrature rules to form a N-dimensional grid component. For example: multi-index $(1, 3)$ would represent a 2-dimensional grid component formed by the tensor product $Q_1 \otimes Q_3$.

DEFINITION 2 An index set is a set of multi-indices that represent the components of the sparse grid. An index set $\mathcal{X}$ is admissible if it satisfies the following condition for generalized sparse grid construction for all multi-indices $\mathbf{k} \in \mathcal{X}$:

$$\mathbf{k} - \mathbf{e}_j \in X \text{ for } 1 \leq j \leq d, k_j > 1, \tag{2.81}$$

where $\mathbf{k}$ is a multi-index, $\mathbf{e}_j$ is the j-th unit vector and d is the number of dimensions of the problem. In other words, any admissible index set for each $\mathbf{k}$ contains all indices which have smaller entries than $\mathbf{k}$ in at least one dimension.

DEFINITION 3 The forward neighbours of a multi-index $\mathbf{k}$ is defined as the d multi-indices $\{\mathbf{k} + \mathbf{e}_j, 1 \leq j \leq d\}$. By extension, the forward neighbours of a grid are all multi-indices which satisfy this definition, and are not yet part of the index set $\mathcal{X}$. As an example consider a level 2, 2-dimensional sparse grid. The index set for this grid is: $\{(1,1),(2,1),(1,2)\}$. The set of forward neighbours for this grid would be: $\{(3,1),(2,2),(1,3)\}$. This is depicted in Figure 2.8.1, where the sparse grid is shown in dark gray, and the forward neighbours in light grey.

DEFINITION 4 The backward neighbours of a multi-index are defined as all multi-indices $\{\mathbf{k} - \mathbf{e}_j, 1 \leq j \leq d\}$.

DEFINITION 5 A parent interaction is an interaction which contains exactly $(i-1)$ of the indices of the current interaction, where i indicates the order of the current interaction. As an example, for third order interaction $(1,2,4)$ there are exactly three parent interactions: $(1,2)$, $(1,4)$ and $(2,4)$.

The next section will present a description of the adaptive refinement approach by Gerstner & Griebel [18] and the Sobol adaptive refinement method.

2.8.1 Gerstner and Griebel Adaptivity

The Gerstner and Griebel adaptive approach is explained in detail in [18], which the information contained in this section is based on. This approach divides the index set $\mathcal{I}$ into two disjoint sets, called *active* and *old* index set. The active index set $\mathcal{A}$ contains the forward neighbours of the old index set $\mathcal{I}$. For each of the multi-indices in $\mathcal{A}$, an error indicator is calculated:

$$g_j = \left| I_{\mathcal{I}} - I_{\mathcal{I}+\mathcal{A}_j} \right|, \tag{2.82}$$

where g is the error indicator, I indicates the computed value of the integral and j indicates a multi-index in $\mathcal{A}$. In other words, the error indicator calculates the difference between the calculated value of the integral with the old index set, the value of the integral calculated with the old index set plus one of the multi-indices in the active index set. This is done for each entry in $\mathcal{A}$. The global error estimate η is then the sum of all g_j:

$$\eta = \sum_{j=1}^{m} g_j, \tag{2.83}$$

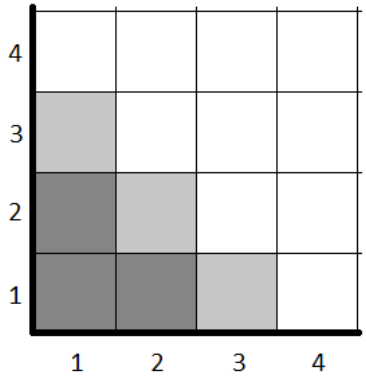


Figure 2.9: Diagram showing the forward neighbours of a level 2, 2-dimensional sparse grid. The multi-indices of the sparse grid are shown in dark grey, while the three forward neighbours are light grey.

where m is the number of multi-indices in $\mathcal{A}$. For each iterative refinement step, the following actions are taken:

- The index with the largest associated error indicator is selected from the active index set and put into the old index set.
- Its associated error estimate is subtracted from the global error estimate η.
- At the same time the new admissible forward neighbours are added to the active index set. The global error estimate is recalculated by determining the error estimators of these new forward neighbours.
- The value of the integral is updated.
- If the global error estimate falls below a given threshold, the computation is stopped and the computed integral value is returned. If this is not the case, the cycle goes back to the first step.

The computed integral value from the Gerstner & Griebel algorithm was compared to a high-level full sparse grid, and yielded results which were the same up to an accuracy of 10^{-8} or better.

Algorithm 2.8.1: GERSTNER&GRIEBEL(TOL)

$\mathbf{i} := (1, \ldots, 1)$
$\mathcal{O} := \emptyset$
$\mathcal{A} := \mathbf{i}$
$r := \Delta_{\mathbf{i}} f$
while $(\eta > TOL)$
do
- select $\mathbf{i}$ from $\mathcal{A}$ with largest g_i
- $\mathcal{A} := \mathcal{A} \setminus \{\mathbf{i}\}$
- $\mathcal{O} := \mathcal{O} \cup \{\mathbf{i}\}$
- $\eta := \eta - g_i$
- **for** $k := 1, \ldots, d$
  - **do**
    - $\mathbf{j} := \mathbf{i} + \mathbf{e}_k$
    - **if** $\mathbf{j} - \mathbf{q} \in \mathcal{O}$ for all$q = 1, \ldots, d$
      - **then**
        - $\mathcal{A} := \mathcal{A} \cup \{\mathbf{j}\}$
        - $s := \Delta_{\mathbf{j}} f$
        - $r := r + s$
        - $\eta = \eta + g_{\mathbf{j}}$

return (r)

2.8.2 Sobol Adaptive Refinement Algorithm

The goal of this algorithm is to find a grid refinement based on the information contained in the Sobol variances of the current grid. Since the values obtained are just approximations of the real values, they can change between refinement steps. This algorithm does not look forward (like the Gerstner & Griebel approach, and also does not use the values from previous refinement steps; for each step the refinements are chosen based on the current grid. The algorithm is explained in words first:

First, the Sobol adaptive refinement algorithm starts with a level 2 sparse grid, with the number of dimensions d equal to number of the dimensions of the problem under consideration. The range of all parameters is normalized to $[0, 1]$ in the code, so this has to be translated to the range of the problem parameters. From this, an input value matrix can constructed and the simulations are run. The relevant data is extracted from the simulation results, which are then used to calculate the mean, total variance and first order Sobol variances. Note that, since this is only a level 2 grid, any Sobol variances higher than the first order are automatically zero. The only way to approximate the higher order Sobol variances is by adding points in the appropriate directions, which means a mechanism is required to add the appropriate higher order interactions.

Next, the Sobol variances are sorted based on their magnitude, from high to low. A cut-off value is used to determine which directions to refine. The sum of the first N Sobol variances

is such that:

$$\sum_{r=1}^{N} D_r \geq C D_{tot}, \tag{2.84}$$

where r indicates the rank of the Sobol variance according to magnitude, C is the cutoff value and D_{tot} is the total variance. In parallel, a list of valid forward neighbours of the current grid is calculated. The refinement of the grid is added in the directions corresponding to the N selected Sobol variances by selecting the appropriate multi-indices out of the list of forward neighbours. This is done as follows: for D_1, the refinement occurs only in the direction of the first parameter (which corresponds to multi-indices of the form $(x, 1, \ldots, 1)$) where the x can be any value > 1.

For a second order interaction, the refinement occurs on the plane spanned by 2 corresponding parameters. For example take $D_{1,2}$, the corresponding forward neighbours must be of the form $(x, y, 1, \ldots, 1)$, where x and y can be any number > 1. The principle extends to N-th order interactions. Which interactions are refined depends on the cut-off value. Since the calculated values of the Sobol variances are only approximations, their exact value can change between refinement steps.

Using these guidelines on their own would mean that higher-order interactions are never introduced. Since only the first order Sobol variances are available on a level $l = 2$ sparse grid, refinements would only occur in the directions of single parameters. To remedy this, a higher-order interaction is included in the refinement if it satisfies two conditions:

- All of its parent interactions are being refined.
- It is the first time this higher order interaction is being refined.

The first condition is based on the assumption that higher order interactions are more likely to be significant if their parent interactions are. In some cases this may not be true, in which case a significant higher order interaction can be neglected. This is where the importance of the cut-off value becomes clear. The second condition is necessary since we only want to add a higher order interaction once. After that, the corresponding Sobol variance can be calculated. Whether or not more refinements are added in this direction then depends on the ranking of the Sobol variances in the next refinement cycle.

The Sobol variances are only approximations of the real values, which means they will change with each step until the value converges. For high a cut-off value, only interactions which appear to be of small significance are not refined. Even if the initial approximation of the values is different from the real value, it is likely that the total error will be limited. As such, the cut-off value plays an important role in the performance of the algorithm: the higher the value, the closer the refined grid will look like a standard sparse grid. On the other hand, the lower this value is set, the higher the chance significant interactions are ignored. A good starting point is to set the cut-off to 0.95 as it appears to include the most significant interactions while still reducing the number of points of the refined grid when comparing to a standard sparse grid.

Finally, a mechanism is in place to check that the parent refinements of each interaction are being refined as well. If this would not be the case, there is a risk of limiting the search space. It is not known exactly how much this limited search space could affect the performance of the algorithm, so the choice was made to include this condition. It also helps conserve the simplex structure of a standard sparse grid. The algorithm is summarized here:

Algorithm 2.8.2: SOBOLADAPTIVEREFINEMENT(C)

$\mathbf{i} := [(1,1,\ldots,1),(2,1,\ldots,1),(1,2,\ldots,1),\ldots,(1,1,\ldots,2)]$
while $(continue = true)$
do
- $data := f(\mathbf{i})$
- $D := Sobol\_variances(\mathbf{i}, data)$
- $var := D_{tot}$
- $D := \mathrm{sort(D)}$
- $D_c := \{\}$
- $j = 1$
- **while** $(sum(D_c) < C.var)$
  - **do**
    - $D_c = D_c + D(j)$
    - $j = j + 1$
- $N_f := forwardneighbours(\mathbf{i})$
- $\mathbf{r} := refinements(N_f, D_c)$
- $\mathbf{i} := \mathbf{i} + \mathbf{r}$

The advantages of this method are as follows:

- Conserves a grid structure which resembles the standard sparse grid, which is already better then traditional tensor product grids and Monte Carlo simulations.
- Reduced grid size when compared to a standard sparse grid.
- Intuitive refinement criterion based on Sobol variances, which indicate which parameters have the largest influence on the result.
- A low number of refinement steps (and thus simulations) is generally sufficient to gain a good approximation of the solution for problems of low to moderate dimension. This means gaining valuable information of the problem at hand at a limited computational cost.

Some disadvantages include:

- There is generally no guarantee that the method converges to the true value for a high number of refinement steps. The only exception to this is when the cut-off value is set

so high that all interactions are included, in which case the algorithm simply produces a standard sparse grid.

- There is a risk that a significant interaction is missed because the value of one or multiple of its parent interactions is so small that it is not included in the refinements. The chance of this happening increases with dimensions, simply because there are more potentially important interactions.
- Information from previous refinement steps is not currently included when deciding which refinements to add. Taking into account previous values could mean that no more points are added in a certain direction when the value has converged to a given threshold. This was not included since the search space could be limited, and the effect of this is unknown. The overall effect of this is almost negligible, since in practice only a low number of refinement steps will be taken.

Chapter 3

Verification

This chapter is dedicated mostly to the verification of the implementation of integration and interpolation on sparse grids, as well as two adaptive grid refinement methods. First, the test functions which are used for the integration and interpolation testing are defined in 3.1. Sections 3.2 and 3.3 will show the performance of the sparse grid method to perform integration and interpolation for these six functions. Finally, Section 2.8 will explain the new Sobol adaptive refinement method as well as the existing method by Gerstner & Griebel [18]. The section is finalized with a simple application example.

3.1 Test functions

A set of six test functions defined by Genz [39] will be used to verify the implementation of the integration and interpolation in the code. They are defined as follows:

$$f_1(x) = \cos\left(2\pi w_1 + \sum_{i=1}^{d} a_i \mathbf{x}_i\right), \qquad \text{"Oscillatory"}, \tag{3.1}$$

$$f_2(x) = \prod_{i=1}^{d} \frac{1}{a_i^2 + (\mathbf{x}_i - w_i)^2}, \qquad \text{"Product Peak"}, \tag{3.2}$$

$$f_3(x) = \frac{1}{\left(1 + \sum_{i=1}^{d} a_i \mathbf{x}_i\right)^{d+1}}, \qquad \text{"Corner Peak"}, \tag{3.3}$$

$$f_4(x) = \exp\left(-\sum_{i=1}^{d} a_i^2 (\mathbf{x}_i - w_i)^2\right), \qquad \text{"Gaussian"}, \tag{3.4}$$

$$f_5(x) = \exp\left(-\sum_{i=1}^{d} a_i|\mathbf{x}_i - w_i|\right), \qquad \text{"Continous"}, \tag{3.5}$$

$$f_6(x) = \begin{cases} 0 & \text{if } \mathbf{x}_1 > w_1 \text{ and } x_2 > w_2, \\ \exp\left(\sum_{i=1}^{d} a_i \mathbf{x}_i\right) & \text{otherwise,} \end{cases} \qquad \text{"Discontinous"}. \tag{3.6}$$

where d is the number of dimensions, $\mathbf{x}$ is the input vector of length d, and a_i and w_i are both set to constants for the following computations:

$$a_i = \frac{9}{d} \qquad \text{for } i = 1, \ldots, d, \tag{3.7}$$

$$w_i = 0.5 \qquad \text{for } i = 1, \ldots, d. \tag{3.8}$$

It is also possible to define vectors $\mathbf{a}$ and $\mathbf{w}$ such that the entries are not constant, but since the objective is simply to verify the implementation of the code, this is deemed unnecessary.

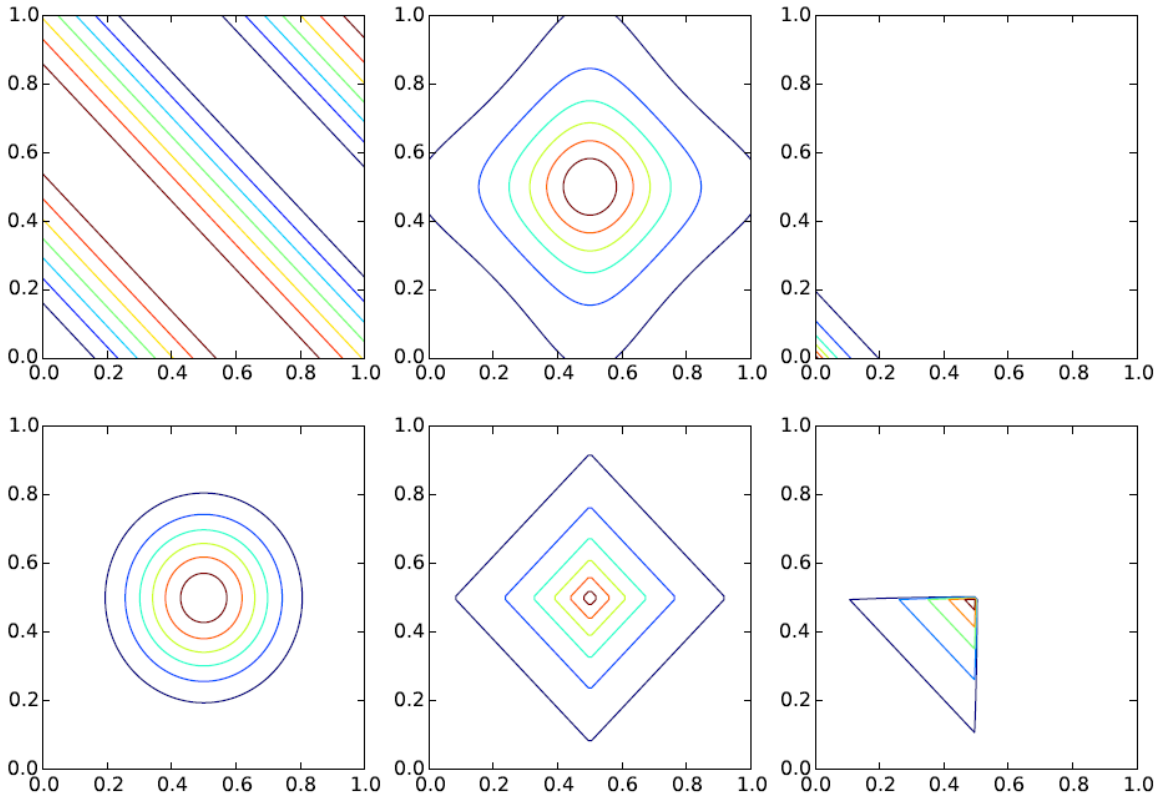


Figure 3.1: Qualitative contour plots of the six Genz functions in 2 dimensions. On the top: Genz function 1 to 3, on the bottom: Genz function 4 to 6. Blue indicates a low value, red indicates a high value.

A qualitative contour diagram of the six Genz functions is given in Figure 3.1 For five out of six Genz functions, an analytical solution for the integral $\int_0^1 f_i(x)dx$ can be found. The only function for which an analytical solution is not known is Genz function $f_3(x)$. The others are

given here:

$$\int_0^1 f_1(x)dx = 2^d \cos\left(2\pi w_1 + 0.5\sum_{i=1}^{d} a_i\right) \prod_{i=1}^{d} \frac{\sin(0.5a_i)}{a_i}, \tag{3.9}$$

$$\int_0^1 f_2(x)dx = \prod_{i=1}^{d} a_i \left(\arctan(a_i(1-w_i)) + \arctan(a_i w_i)\right), \tag{3.10}$$

$$\int_0^1 f_4(x)dx = \prod_{i=1}^{d} \frac{\sqrt{\pi}}{2a_i} \left(\operatorname{erfc}(-a_i w_i) - \operatorname{erfc}(a_i(1-w_i))\right), \tag{3.11}$$

$$\int_0^1 f_5(x)dx = \prod_{i=1}^{d} \frac{1}{a_i} \left(2 - \exp(-a_i w_i) - \exp(a_i(w_i-1))\right), \tag{3.12}$$

$$\int_0^1 f_6(x)dx = \begin{cases} \prod_{i=1}^{2} \frac{\exp(a_i w_i)-1}{a_i} & \text{if } d < 3, \\ \prod_{i=1}^{2} \frac{\exp(a_i w_i)-1}{a_i} \prod_{i=3}^{d} \frac{\exp(a_i)-1}{a_i} & \text{otherwise} \end{cases}. \tag{3.13}$$

In the equations above, erfc denotes the complementary error function. All other symbols are the same as previously defined. The given analytical solutions are used in the following section to track the convergence of the polynomial and linear integration. Since the domain is $[0,1]^d$, calculating the integral is in this case the same as calculating the mean. The integral of the third Genz function is estimated by a Monte Carlo simulation of $N = 100$ million samples. An indication of the accuracy of the Monte Carlo method is given by calculating the standard error of the estimated mean:

$$\epsilon_{std} = \frac{\sigma}{\sqrt{N}} \tag{3.14}$$

where σ is the standard deviation and can be calculated from the integral form for each of the test functions. The standard error can be seen as the standard deviation of the estimated sample mean with respect to the real mean. As a criterion for the accuracy we take $2\epsilon_{std}$, such that there is roughly a 95% chance that the error of the Monte Carlo estimated mean to the integral value is less than or equal to $2\epsilon_{std}$. The same procedure can be followed for the integration using non-uniform weights, since the analytical solutions of the integrals are only available for a uniform input distribution but we still want to check the accuracy of the integration method.

3.2 Verification of Integration

3.2.1 Uniform Input distribution

The integrals of the test functions defined in the previous section were computed using Clenshaw-Curtis sparse grids of levels $l = 1, \ldots, 9$ and dimensions $d = 2, 3, 4, 6$ and 8, applying both linear and polynomial weights. For each of these, the relative error is plotted as a function of the grid size. The relative error is defined as:

$$\epsilon = \left| \frac{I_{approx} - I_{exact}}{I_{approx_{l=1}} - I_{exact}} \right|, \tag{3.15}$$

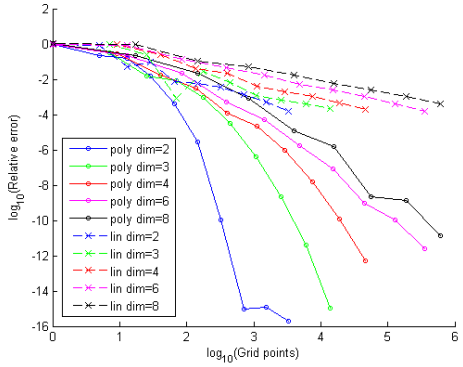
where I_{approx} is the approximate integral value obtained with the sparse grid code, and I_{exact} is the analytical solution of the integral.

For $\int_0^1 f_3(x)dx$, the exact solution is not known. Instead of using I_{exact}, I_{MC} from a Monte Carlo simulation, using $N = 10^8$ samples, is used as the reference. An indication of the Monte Carlo method is given by calculating $2\epsilon_{std}$, where ϵ_{std} is given by (3.14). These are put in Table 3.1. From Figure 3.2 the following observations can be made:

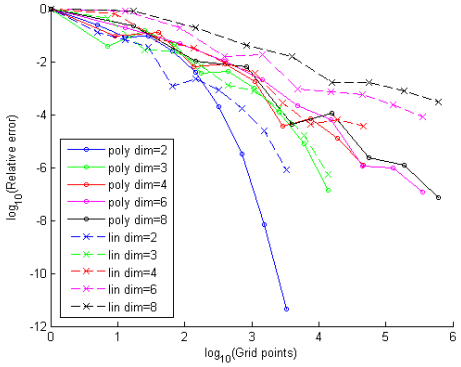
- The convergence rate generally decreases with increasing dimensions, for all functions. Linear integration is less affected by this effect than polynomial integration.
- For the first Genz function the polynomial integration clearly outperforms its linear counterpart. There is clearly spectral convergence for the polynomial integration, while the convergence of the linear integration stabilizes at a steady value.
- For the second Genz function, there is spectral convergence for $d = 2, 3$, but not yet for higher dimensions. The linear integration seems to converge faster than for the first Genz function, but it is still outperformed by the polynomial integration, which in turn is slower than for the first Genz function. The difference between linear and polynomial integration is small for $d = 3$ and 4.
- The polynomial integration seems to perform slightly better than its linear counterpart for the third Genz function. Only the convergence of polynomial integration for $d = 2$ and to a lesser extent 3 is decent, for all other cases the convergence is poor. The error ϵ never comes close to $2\epsilon_{std}$ as tabulated in Table 3.1.
- For the fourth Genz function a similar trend to the second Genz function is found. For $d = 2, 3$ the polynomial integration the convergence is spectral, for higher dimensions this is not (yet) the case. The linear integration is outperformed by the polynomial integration for each dimensions.
- For the fifth Genz function the linear integration outperforms the polynomial integration. The difference between both increases with dimensions as the linear integration converges faster with increasing level, while the polynomial integration converges slower.
- The sixth Genz function also has a better convergence rate for linear integration, though the difference is very small. Both types of integration seem to be ineffective for this type of function. This is because the number of points around the discontinuity is quite small, meaning it is hard to resolve the jump. For the linear integration especially this affects the convergence rate.

For practical cases, the computational resources are much more limited than for the simple test functions used here. It is convenient to have an idea of the number of simulations required to reach a certain accuracy. To that end, the grid points required to reach an error lower than 10^{-2} for each function is given in Table 3.2. The third and sixth Genz function are omitted since convergence generally does not reach a value of 10^{-2} for these functions for an acceptable l. It is worth noting that it looks like, in some cases, the minimum grid size which satisfy the convergence requirement is higher for smaller dimensions. This is because

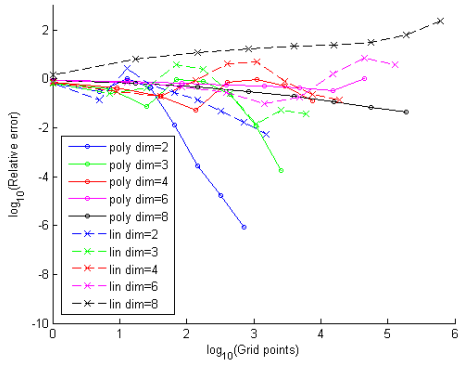
(a) First Genz function.



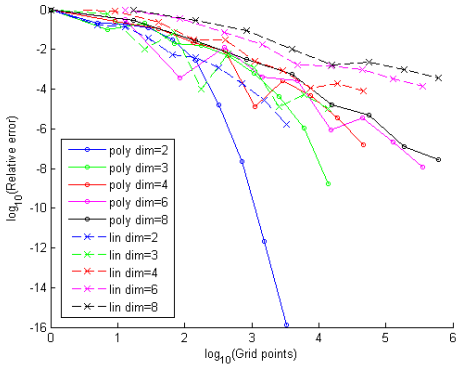
(b) Second Genz function.



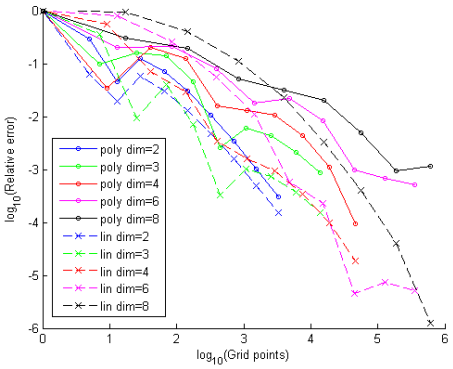
(c) Third Genz function.



(d) Fourth Genz function.



(e) Fifth Genz function.



(f) Sixth Genz function.

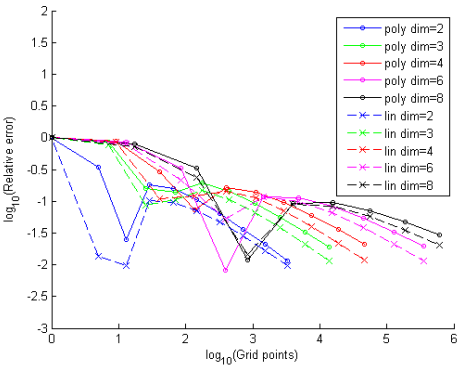


Figure 3.2: Convergence of linear and polynomial integration of the different Genz functions for dimension $d = 2, 3, 4, 6, 8$ and level $l = 1, \ldots, 9$ for a uniform input parameter distribution.

Table 3.1: Accuracy of the Monte Carlo simulation for the third Genz function assuming a uniform input parameter distribution.

Dimensions	$2\epsilon_{std}$
2	9.2e-06
3	2.4e-06
4	5.3e-07
6	1.9e-06
8	1.5e-07

Table 3.2: Minimum grid sizes to reach an error below 10^{-2} for the first, second, fourth and fifth Genz function.

(a) Polynomial integration of f_1.

Dimensions	Number of grid points	Error
2	65	3.9e-04
3	69	8.1e-03
4	137	3.1e-03
6	389	4.8e-04
8	849	8.0e-04

(b) Linear integration of f_1.

Dimensions	Number of grid points	Error
2	65	7.3e-03
3	441	6.2e-03
4	1105	3.9e-03
6	4865	4.9e-03
8	15713	5.8e-03

(c) Polynomial integration of f_2.

Dimensions	Number of grid points	Error
2	145	4.0e-03
3	177	3.6e-03
4	137	6.3e-03
6	1457	2.1e-03
8	849	6.6e-03

(d) Linear integration of f_2.

Dimensions	Number of grid points	Error
2	65	1.1e-03
3	177	6.9e-03
4	401	9.0e-03
6	4865	9.5e-04
8	15713	1.7e-03

(e) Polynomial integration of f_4.

Dimensions	Number of grid points	Error
2	145	3.0e-03
3	441	5.0e-03
4	401	8.7e-03
6	1457	4.0e-04
8	849	3.0e-03

(f) Linear integration of f_4.

Dimensions	Number of grid points	Error
2	65	5.4e-03
3	177	9.9e-03
4	1105	2.6e-03
6	4865	1.6e-03
8	15713	1.7e-03

(g) Polynomial integration of f_5.

Dimensions	Number of grid points	Error
2	705	3.4e-03
3	441	2.6e-03
4	7537	4.3e-03
6	15121	8.4e-03
8	15713	3.3e-03

(h) Linear integration of f_5.

Dimensions	Number of grid points	Error
2	321	4.7e-03
3	177	7.1e-03
4	401	3.7e-03
6	4865	5.7e-04
8	9017	1.9e-03

the number of points in a grid increases significantly with each level. In a few cases, the convergence jumps from for example 10^{-2} to 10^{-4}, while for others the jump is much smaller. Since the grid sizes shown are the values for which the error goes below 10^{-2}, it is possible that the grid of a lower level was barely too high to quilify for this requirement. These values are simply given as an indication of what to expect in terms of computational effort for a problem of given dimensions.

The convergence for the third and sixth Genz function never reaches the requirement of $< 10^{-2}$, which is why there are no tables for these functions.

3.2.2 Truncated Normal Input distribution

The analytical solutions of the integrals are only available for a uniform weighting function $\rho(x) = 1$, which means there is no exact solution available. However, a Monte Carlo simulation can be used to estimate the integral.

$$\epsilon = \left| \frac{I_{approx} - I_{MC}}{I_{approx_{l=1}} - I_{MC}} \right|, \tag{3.16}$$

where the denominator is introduced for the sole purpose of easily comparing the convergence for different dimensions by making sure ϵ always starts at 1. The integrals were again computed using Clenshaw-Curtis sparse grids of levels $l = 1, \ldots, 9$ and dimensions $d = 2, 3, 4, 6$ and 8, applying both linear and polynomial weights.

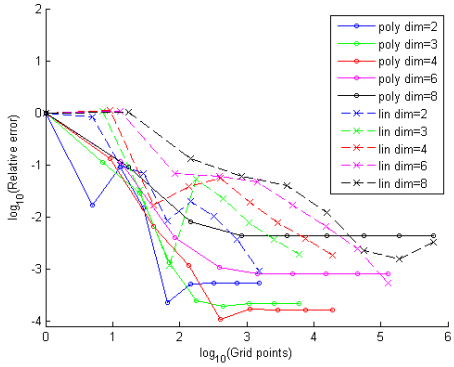
For each of the Genz functions, the reference solution is calculated using a Monte Carlo simulation of $N = 10^8$ samples. An indication of the accuracy of the results obtained using Monte Carlo is the standard error defined in (3.14):

$$\epsilon_{std} = \frac{\sigma}{\sqrt{N}} \tag{3.17}$$

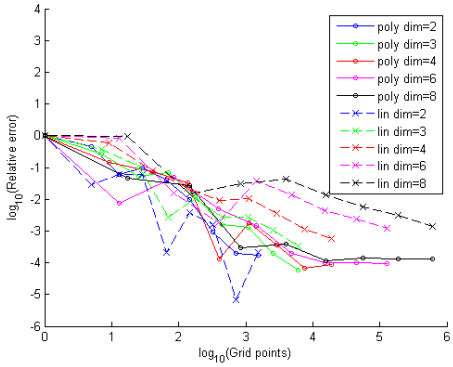
In the tables below, $2\epsilon_{std}$ is given for dimensions for each test function.
From Figure 3.3 the following observations can be made:

- The convergence rate generally decreases with increasing dimensions, for all functions.
- Since the reference solutions are obtained via Monte Carlo. In some of the figures, the error ϵ stops going down once it reaches a certain value; this is because the reference result is only accurate up to a certain order of magnitude. The accuracy of the Monte Carlo results is estimated by calculating 2 ϵ_{std} for each function for different dimensions. These values are given in Table 3.3. In almost all cases, the value where the error stays constant is below $2\epsilon_{std}$, meaning that it is within the expected 95% range of the result. In the cases where this effect occurs at a value higher than $2\epsilon_{std}$, it is still close enough to that value (i.e.: the same order of magnitude) to assume that the integration method is valid.
- For the first Genz function the polynomial integration outperforms its linear counterpart. The polynomial integration quickly reaches a point where the error is of the same

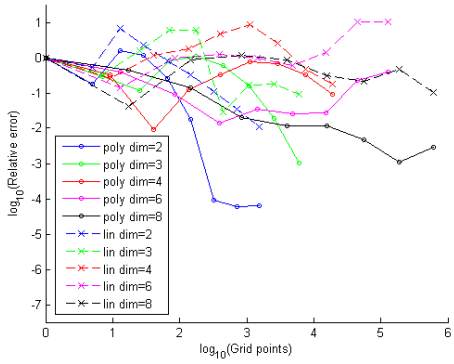
(a) First Genz function.



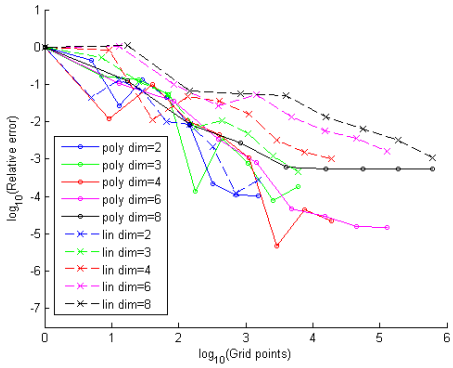
(b) Second Genz function.



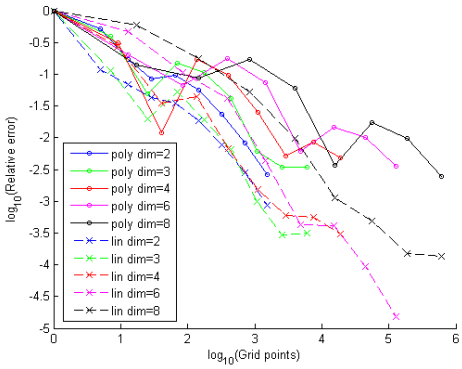
(c) Third Genz function.



(d) Fourth Genz function.



(e) Fifth Genz function.



(f) Sixth Genz function.

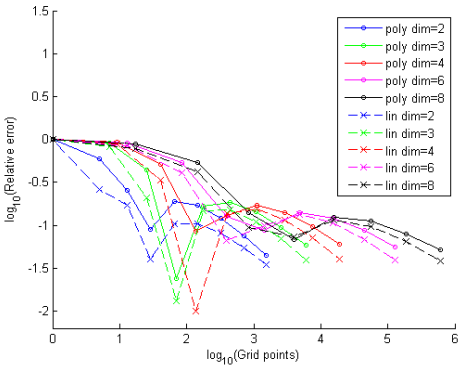


Figure 3.3: Convergence of linear and polynomial integration of the different Genz functions for dimension $d = 2, 3, 4, 6, 8$ and level $l = 1, \ldots, 9$ for a truncated normal input parameter distribution.

Table 3.3: Accuracy of the Monte Carlo simulation for each Genz function assuming a truncated normal input parameter distribution.

(a) First Genz function.

Dimensions	$2\epsilon_{std}$
2	7.0e-05
3	6.8e-05
4	6.5e-05
6	6.0e-05
8	5.5e-05

(b) Second Genz function.

Dimensions	$2\epsilon_{std}$
2	1.0e-02
3	1.5e-02
4	1.2e-02
6	1.8e-03
8	7.4e-05

(c) Third Genz function.

Dimensions	$2\epsilon_{std}$
2	8.3e-04
3	8.4e-04
4	8.6e-03
6	7.8e-02
8	5.6e-02

(d) Fourth Genz function.

Dimensions	$2\epsilon_{std}$
2	8.3e-05
3	7.7e-05
4	7.3e-05
6	6.8e-05
8	6.4e-05

(e) Fifth Genz function.

Dimensions	$2\epsilon_{std}$
2	4.6e-05
3	5.2e-05
4	4.8e-05
6	2.7e-05
8	NaN

(f) Sixth Genz function.

Dimensions	$2\epsilon_{std}$
2	NaN
3	NaN
4	1.9e-05
6	6.7e-05
8	8.6e-05

magnitude as the accuracy of the Monte Carlo results. For all dimensions, the convergence rate of the polynomial integration is pretty close, the linear integration seems to be more affected by dimension.

- For the second Genz function, The polynomial and linear integration are very similar (with a few outliers) for dimensions $d = 2$ and 3. For higher dimensions, there is a clear difference between the two, where the polynomial integration performs better with increasing dimension. It takes much longer to reach the point where the error approaches the accuracy of the Monte Carlo method compared to the first Genz function.

- The polynomial integration seems to perform better than its linear counterpart for the third Genz function. Only the convergence of polynomial integration for $d = 2$ and to a lesser extent 3 is decent, for all other cases the convergence is very poor. The error ϵ never comes close to $2\epsilon_{std}$ except for the polynomial integration for $d = 2$.

- The results for the fourth Genz function are similar to those of the second Genz function. For $d = 2, 3$ the polynomial and linear integration are close, but for higher dimensions the polynomial integration performs better.

- For the fifth Genz function the linear integration outperforms the polynomial integra-

tion. The difference between both increases with dimension as the error for the linear integration becomes smaller. The polynomial integration never goes below an error of roughly $10^{-2.5}$, for any number of dimensions or grid points.

- The linear integration outperforms the polynomial integration for the sixth Genz function, though the difference is small. Both types of integration are ineffective for this type of function. This is because the number of points around the discontinuity is quite small, meaning it is hard to resolve the jump. For the linear integration especially this affects the convergence rate.

3.3 Verification of Interpolation

To quantify the convergence of the linear and polynomial interpolation, the exact function values at N random locations are compared to the interpolated values. The following measure β is calculated:

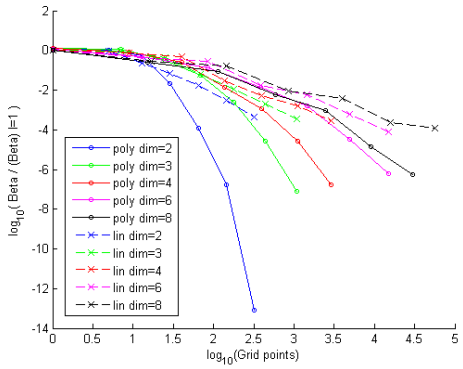
$$\beta = \frac{1}{N} \sum_{i=1}^{N} \left(f_{approx}(\mathbf{x}_i) - f_{exact}(\mathbf{x}_i) \right)^2 , \tag{3.18}$$

where N is the number of samples, i denotes a random sample withing the parameter space and $\mathbf{x}$ is the input vector.

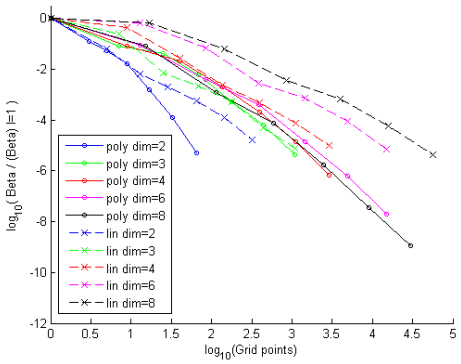
We are investigating the evolution of this value for increasing level and dimensions. To normalize the results for different levels and dimensions, $\beta/(\beta)_{level=1}$. The smaller this value becomes, the better the convergence of the interpolation. It is plotted for the first, second, fourth and fifth Genz functions. The third and sixth Genz functions are omitted since it was already indicated in Section 3.2.1 that the convergence for these two functions is poor. In all the figures, a total sample size of $N = 10^4$ was used to compute the β values. The level was limited to 7 to save time when running the code. To get more accurate results, a larger sample size would be optimal. However, since we are only interested in the trends, and not necessarily in the exact value, $N = 10^4$ is sufficient. The following observations can be made:

- For all functions, both linear and polynomial interpolation converge slower with increasing dimension.
- For the first Genz function, the difference between polynomial and linear interpolation is quite clear, in favor of polynomials. The polynomial interpolation shows spectral convergence.
- The difference between both interpolation methods for the second Genz function is quite small for lower dimensions, but increases with dimension where polynomial interpolation performs better.
- The fourth Genz function has the same trends as the second Genz function. Both interpolation methods are close for $d = 2, 3, 4$, but for $d = 6, 8$ polynomial interpolation is better than linear interpolation.

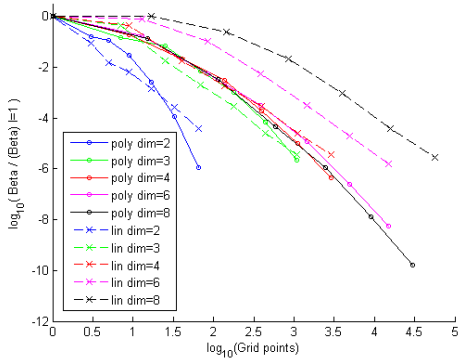
(a) First Genz function.



(b) Second Genz function.



(c) Fourth Genz function.



(d) Fifth Genz function.

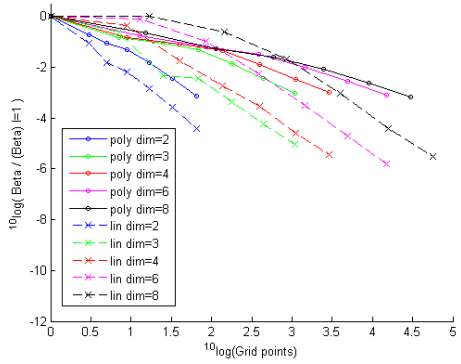


Figure 3.4: Convergence of linear and polynomial interpolation of the different Genz functions for $d = 2, 3, 4, 6, 8$ and level $l = 1, \ldots, 7$.

- The convergence of interpolation for the fifth Genz function is not good for either interpolation method, though linear interpolation outperforms its polynomial counterpart. The difference between the two increases with dimension.

3.4 Adaptive Grid Refinement: Test case

This section will apply both adaptive refinement algorithms to a test case and compare the results to a standard sparse grid. The goal is to show the difference between the two schemes, and also to indicate the difference between polynomial and linear integration.

3.4.1 Moody diagram

The Moody diagram (or Moody chart) plots the friction factor as a function of Reynolds number, inner pipe diameter and roughness. For low Reynolds number, the flow through a pipe is laminar, but at a certain point it will transition to turbulent flow, which is indicated by a discontinuity in the friction factor. Transition does not actually occur at one point, but is shown as the grey area in Figure 3.5. However, for the calculations the assumption is made that the jump occurs at a Reynolds number of 2100. The following explicit formula is known [35]:

$$f = \begin{cases} \frac{64}{Re_D} & \text{if } Re_D < 2100, \\ \left(-1.8 \log_{10} \left[\frac{6.9}{Re_D} + \left(\frac{k/D}{3.7}\right)^{1.11}\right]\right)^{-2} & \text{otherwise,} \end{cases} \tag{3.19}$$

where f is the Darcy friction factor, Re_D denotes the Reynolds number, D is the inner diameter of the pipe and k is the pipe roughness. The Reynolds number is defined as:

$$Re_D = \frac{UD}{\nu}, \tag{3.20}$$

where the subscript D denotes the fact that we are dealing with pipes. U is the flow velocity and ν is the kinematic viscosity. The region where $Re_D < 2100$ corresponds to a laminar flow. At $Re_D = 2100$ the flow transitions into the turbulent regime. The Moody diagram shows the friction factor as a function of Reynolds number for different values of pipe roughness, as shown in Figure 3.5 [36].

It is possible to treat this as a engineering problem with uncertain inputs. Consider ϵ and U as uncertain input parameters where:

$$\epsilon_{min} = 0.001, \tag{3.21}$$
$$\epsilon_{max} = 0.04, \tag{3.22}$$
$$U_{min} = 0.01, \tag{3.23}$$
$$U_{max} = 0.5, \tag{3.24}$$

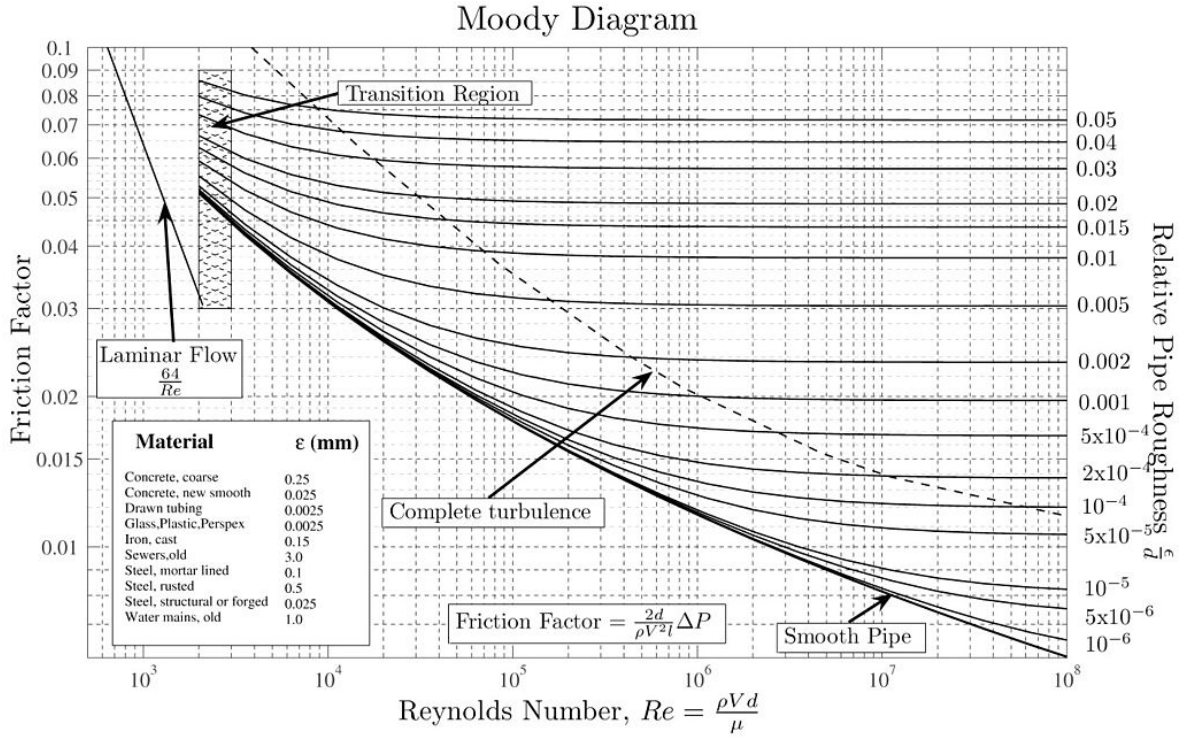


Figure 3.5: The friction factor f as a function of Reynolds number Re for different values of pipe roughness.

where ϵ is the ratio of roughness and inner diameter of the pipe k/D. The other variables are set constant:

$$\nu = 10^{-6}, \tag{3.25}$$
$$D = 0.025. \tag{3.26}$$

Using the above information, the range of Reynolds numbers can be found:

$$(Re_D)_{min} = 250, \tag{3.27}$$
$$(Re_D)_{max} = 12500 \tag{3.28}$$

The two variables ϵ and U will be analyzed for two input parameter distributions: a uniform distribution and a truncated normal distribution. The results are given in the following two sections.

Standard sparse grid results

The evolution of the Sobol variances on a standard sparse grid are given in Tables 3.4 and 3.5 for level $l = 2, 3, 4, 5$ and 6 for the uniform and truncated normal input parameter distribution respectively.

Table 3.4: Evolution of the Sobol variances on the standard sparse grid for the Moody diagram test case assuming a uniform input parameter distribution. Indices 0 and 1 correspond to the Reynold number Re and the ratio of roughness and inner pipe diameter ϵ respectively.

Polynomial basis functions

Variable	Level 2	Level 3	Level 4	Level 5	Level 6
Grid points	5	13	29	65	145
Mean	8.708e-02	5.386e-02	5.476e-02	5.620e-02	5.683e-02
Variance	4.432e-03	2.076e-03	6.693e-04	5.062e-04	4.968e-04
D_0	4.342e-03	2.002e-03	6.055e-04	4.363e-04	4.243e-04
D_1	9.016e-05	6.283e-05	4.532e-05	5.446e-05	5.900e-05
$D_{0,1}$	0	1.086e-05	1.851e-05	1.549e-05	1.346e-05

Linear basis functions

Variable	Level 2	Level 3	Level 4	Level 5	Level 6
Grid points	5	13	29	65	145
Mean	1.033e-01	6.214e-02	5.708e-02	5.679e-02	5.698e-02
Variance	4.383e-03	1.860-03	7.361e-04	5.484e-04	5.049e-04
D_0	4.162e-03	1.770e-03	6.780e-04	4.805e-04	4.246e-04
D_1	-3.560e-04	-9.289-05	1.026e-04	7.663e-05	4.808e-05
$D_{0,1}$	5.672e-04	1.834e-04	-4.452e-05	-8.779e-06	3.215e-05

Table 3.5: Evolution of the Sobol variances on the standard sparse grid for the Moody diagram test case assuming a truncated normal input parameter distribution. Indices 0 and 1 correspond to the Reynold number Re and the ratio of roughness and inner pipe diameter ϵ respectively.

Polynomial basis functions

Variable	Level 2	Level 3	Level 4	Level 5	Level 6
Grid points	5	13	29	65	145
Mean	7.389e-02	4.744e-02	5.354e-02	5.460e-02	5.502e-02
Variance	2.625e-03	1.193e-03	2.283e-04	1.852e-04	1.793e-04
D_0	2.571e-03	1.147e-03	1.873e-04	1.397e-04	1.328e-04
D_1	5.343e-05	4.225e-05	3.325e-05	4.021e-05	4.239e-05
$D_{0,1}$	0	3.815e-06	7.733e-06	5.298e-06	4.103e-06

Linear basis functions

Variable	Level 2	Level 3	Level 4	Level 5	Level 6
Grid points	5	13	29	65	145
Mean	9.023e-02	5.362e-02	5.414e-02	5.473e-02	5.504e-02
Variance	2.743e-03	6.602e-04	2.356e-04	1.834e-04	1.710e-04
D_0	2.504e-03	5.880e-04	1.993e-04	1.395e-04	1.125e-04
D_1	-4.586e-04	-1.021e-04	1.043e-04	5.718e-05	2.629e-05
$D_{0,1}$	6.914e-06	1.742e-04	-6.811e-05	-1.327e-05	3.220e-05

Table 3.6: Evolution of the Sobol variances on the Sobol adaptive sparse grid for the Moody diagram test case using polynomial integration. Indices 0 and 1 correspond to the Reynold number Re and the ratio of roughness and inner pipe diameter ϵ respectively.

Uniform input distribution

Variable	Level 2	Level 3	Level 4	Level 5	Level 6
Grid points	5	7	11	25	45
Mean	8.708e-02	5.379e-02	5.460e-02	5.614e-02	5.678e-02
Variance	4.432e-03	2.087e-03	6.935e-04	5.109e-04	4.991e-04
D_0	4.342e-03	1.997e-03	6.034e-04	4.373e-04	4.251e-04
D_1	9.016e-05	9.016e-05	9.016e-05	6.283e-05	6.288e-05
$D_{0,1}$	0	0	0	1.081e-05	1.113e-05

Truncated normal distribution

Variable	Level 2	Level 3	Level 4	Level 5	Level 6
Grid points	5	7	11	25	45
Mean	7.389e-02	4.740e-02	5.347e-02	5.457e-02	5.501e-02
Variance	2.624e-03	1.203e-03	2.409e-04	1.860e-04	1.792e-04
D_0	2.571e-03	1.150e-03	1.875e-04	1.400e-04	1.330e-04
D_1	5.343e-05	5.343e-05	5.343e-05	4.225e-05	4.224e-05
$D_{0,1}$	0	0	0	3.777e-06	3.875e-06

Sobol adaptive sparse grid results

The Sobol adaptive sparse grid algorithm is explained in Section 2.8.2, the cut-off value is set to 0.95. When looking at the linear approximation for the standard sparse grid, the accuracy of the Sobol variances was poor. Due to this, only polynomial approximation was considered for the adaptive algorithm. The results can be found in Table 3.6.

Comparison

As is clear from (3.19), there will be a discontinuity in the friction factor values since the flow transitions from the laminar to the turbulent regime. To investigate the influence of the discontinuity, both polynomial and linear integration and interpolation are used to plot the response surfaces and to calculate the Sobol variances for two input parameter distributions. As mentioned in Section 2.7, for polynomial integration the data is interpolated to a sparse grid of 2 levels above the initial grid to ensure the integrals approximating the Sobol variances are exact. For the linear integration, the grid is interpolated to a grid of 5 levels above the initial grid.

The first important observation from the standard sparse grid results is that the linear approximation of the Sobol variances seems to be poor. By definition, none of the Sobol variances should have a negative value, yet this is the case for some of them. In Section 2.7, it was shown that the Sobol variances are calculated as follows (eq. (2.62)):

$$D_u := \int \left(\int f(x) d\rho(x_{u'}) \right)^2 d\rho(x_u) - \sum_{w \subset u} D_w. \tag{3.29}$$

Each Sobol variance is determined by calculating an integral and then subtracting the approprioate Sobol variances of a higher order. Because the integral to approximate the Sobol variances is not solved exactly with the linear basis functions, this introduces an error for each of the integral values. When subtracting the higher order Sobol variances, the error associated with each of them is also subtracted, which affects the overall results in a negative way. The algorithm uses these values to make decisions on where to refine the grid, so the performance of the refinement algoritm is heavily tied to the accuracy of the computed Sobol variances. Because of the accuracy of the Sobol variances, the Sobol adaptive refinement algorithm will only be used with polynomial approximation. In order to evaulate the integrals exactly, piecewise quadratic approximation needs to be implemented.

For both input distributions, the polynomial approximation on the standard sparse grid and the Sobol adaptive sparse grid can be compared. The final grids for both input distributions turn out to be the same in this case. The level 6 standard sparse grid and the adaptive sparse grid are shown in Figure 3.6. The final adaptive grid contains 45 grid points compared to 145 in the standard sparse grid, which is a reduction of 69%. The results in Tables 3.4, 3.5 and 3.6 indicate that the mean for the adaptive grid is always close to the results on the standard sparse grid, with a maximum deviation in the order of 1%. For the Sobol variances there are some small differences in values but the relative importance of each variable shows little difference, meaning that the accuracy of the results is affected only slightly. Since the linear approximation method provides irregular results, it is hard to quantify the effect of

the discontinuity in the solution on the values obtained for mean and Sobol variances. On the other hand, the refinement algorithm is working properly, showing a significant grid size reduction at the cost of a minimal reduction in accuracy.

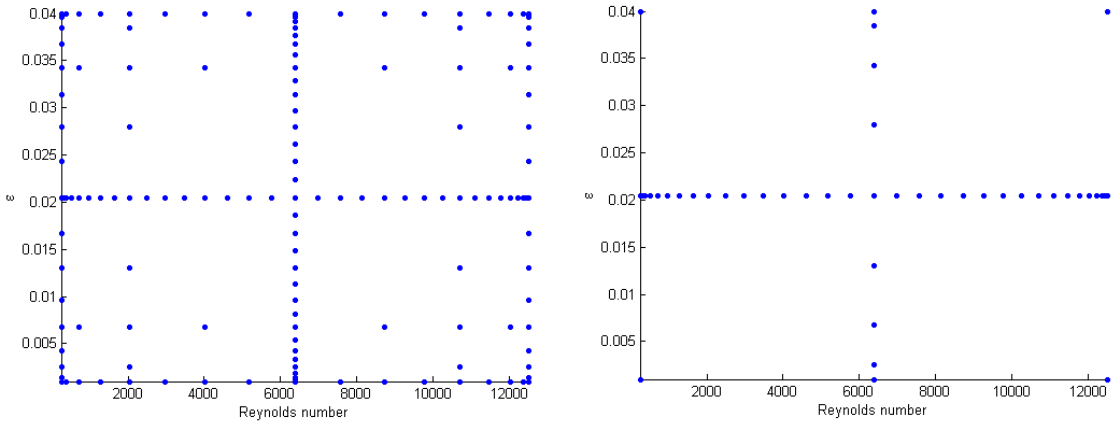


Figure 3.6: A comparison of the level 6 standard sparse grid (145 points) to the result of the adaptive refinement algorithm (45 points) on the right.

Chapter 4

Application: Heavy gas release

This chapter introduces an industrial case, for which the performance of the standard sparse grid, the Gerstner & Griebel adaptive sparse grid and the new Sobol adaptive sparse grid are compared. Section 4.1 will present the specifics of the case and its applications. Section 4.2 presents the results for the three different methods. Finally, Section 4.3 will compare and discuss the differences between the methods.

4.1 Heavy Gas Release

The case that will be studied is a release of a heavy gas - in this case propane - over a barrier located downstream of the release point. The quantity of interest is the effect distance: the distance from the barrier where the molar concentration propane gas drops below 1%, which is roughly half of the lower explosion limit (LEL) [37].

The real application of this case is in external safety. For example: A truck transporting a heavy gas is involved in an accident, which causes the gas being transported to be released into the atmosphere. After a while, the gas may start spilling over the barrier next to road. It is important to know the perimeter where the gas concentration is higher than a given value, since people near the area are in potential danger. If there are buildings near the highway, it is important to know whether evacuation is necessary or not. This is also where the input uncertainty plays a role. The results may change drastically with different input or combinations thereof. A parameter study was already performed prior to this study, from which the three most important input parameter were chosen. Since the computational resources are limited, a simplified model is used to investigate the behavior of the heavy gas. In this model, the geometry is completely fixed. Some of the parameter relating to the model are presented here:

- The barrier height is fixed at $4m$ height.

- The gas is released 60m upstream of the barrier.
- The gas is released at at a height of 1m and the release point has a diameter of 1.243m.
- The wind direction is set to 0 degrees.
- The atmospheric boundary layer velocity U_{ABL} has a range of 3 to 7m/s.
- The gas release velocity U_{rel} has a range of 18 to 22m/s.
- The gas release temperature T_{rel} has a range of 270 to 310K.

Here the wind velocity profile at the boundary of the domain is characterized by the atmospheric boundary layer velocity. The input uncertainty of the three input variables U_{ABL}, U_{rel} and T_{rel} is modelled using a uniform and a truncated normal distribution. The topology of the area surrounding the gas release is neglected. In reality this would influence the flow of the gas greatly, but it is too complicated to consider different landscapes when we are interested in a simple model. A diagram representing the computational domain is given in Figure 4.1. The gas release point lies in a symmetry plane, effectively cutting the computational

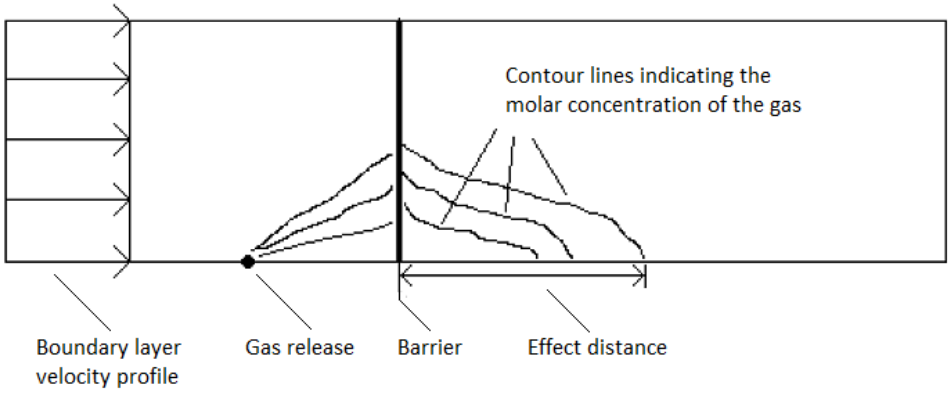


Figure 4.1: Top view diagram of the release of propane gas flowing over a barrier.

domain to half without loss of accuracy. The ground is modelled as a closed wall, the other four boundaries are open so the gas is allowed to exit the domain. The problem is modelled with ANSYS^{TM}/Fluent version 14.5 [38] using steady-state RANS equations. The turbulence model used in the computations is the standard k-ϵ two equation model where 3 constants are specified: $C_\mu = 0.09$, $C_{1\epsilon} = 1.44$ and $C_{2\epsilon} = 1.92$. The turbulent Prandtl number is set to 1. The mesh contains roughly 1 million mixed cells with a higher density of cells near the gas release point and close to the ground. The mesh is shown in Figure 4.2.

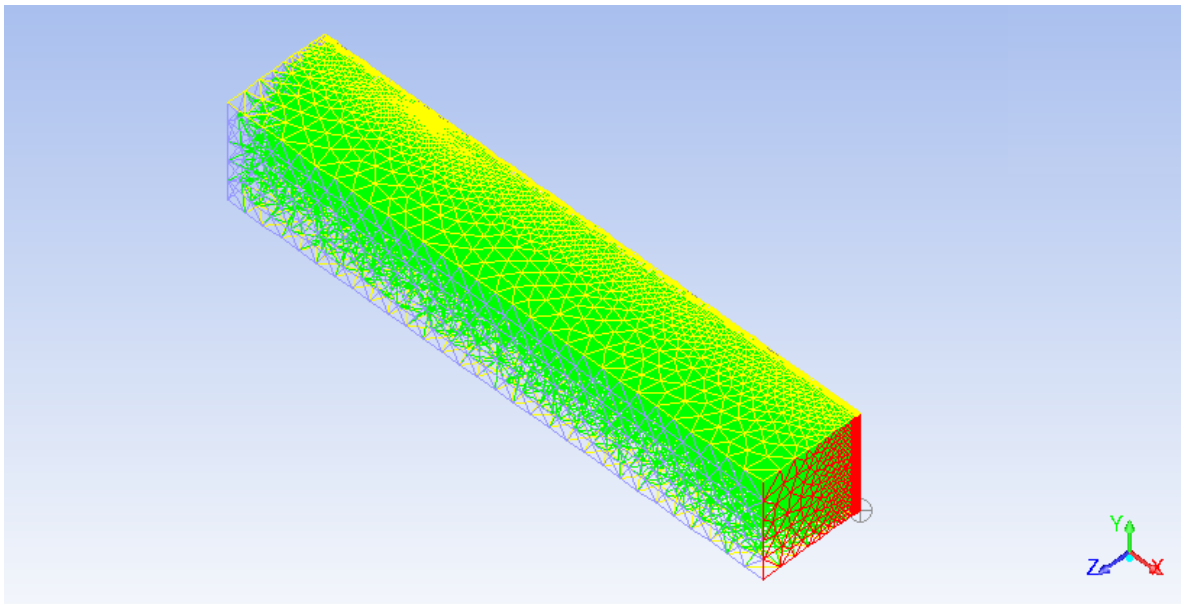


Figure 4.2: The mesh over the entire domain in ANSYS^{TM}/Fluent version 14.5 [38].

4.2 Results

This section presents the results of the simulations. All simulations were run until the solution converged to 10^{-6}. One computation took anywhere between 12 and 24 hours on one core. A total of 69 simulations was performed, corresponding to a standard sparse grid of level $l = 4$ in 3 dimensions. A table containing the input matrix and corresponding effect distances can be found in Appendix B. The results for the complete data set, as well as two adaptive methods, are presented here.

4.2.1 Standard sparse grid

A level 4 sparse grid in 3 dimensions was constructed, containing 69 sample points. Each sample point represents a combination of input values within their specified ranges. Using this data, information on the system is obtained. The evolution of the sparse grid from level 1 to level 4 for the heavy gas release is shown in Figure 4.3. The Sobol variances are computed assuming a uniform and a truncated normal distribution for the three input variables. Their values are given for levels $l = 2, 3$ and 4 in Table 4.1. The level 1 grid is omitted since it contains only 1 point. To compute the Sobol variances of order n, a grid of level $l = n + 1$ is required. This explains why the second order Sobol variances are only available for a level 3 sparse grid, the third order Sobol variances for the level 4 sparse grid and so on.

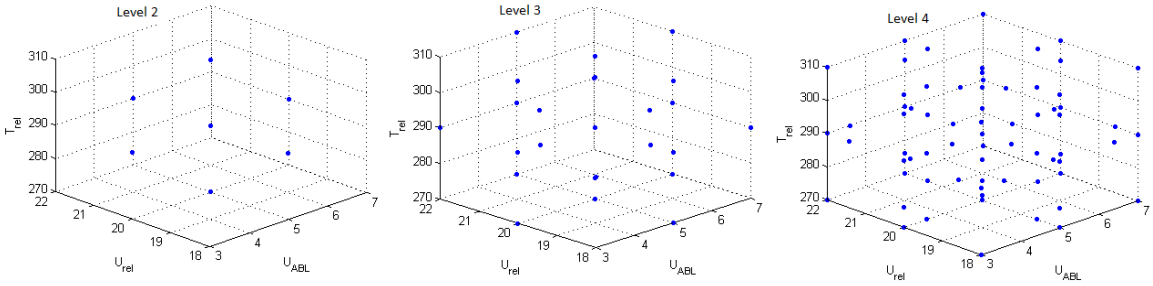


Figure 4.3: Evolution of the standard sparse grid for the heavy gas release. Going from left to right: grid points for level $l = 2, 3$ and 4.

Table 4.1: Sobol variances calculated on a standard sparse grid. Indices $0, 1$ and 2 correspond to the atmospheric boundary layer velocity U_{ABL}, the propane release velocity U_{rel} and the propane release temperature T_{rel} respectively.

Uniform input parameter distribution.

Variable	Level 2	Level 3	Level 4
Grid points	7	25	69
Mean	184.7	183.1	182.8
Variance	446.2	363.9	346.6
D_0	375.9	261.3	253.0
D_1	60.18	73.59	75.73
D_2	10.15	0.5088	0.9390
$D_{0,1}$	0	3.826	2.417
$D_{0,2}$	0	24.86	13.24
$D_{1,2}$	0	0.002394	0.1490
$D_{0,1,2}$	0	0	1.096

Normal input parameter distribution.

Variable	Level 2	Level 3	Level 4
Grid points	7	25	69
Mean	182.8	181.3	180.9
Variance	264.4	166.3	158.6
D_0	222.7	114.6	111.0
D_1	35.66	40.30	40.56
D_2	6.014	1.391	2.560
$D_{0,1}$	0	1.343	0.8433
$D_{0,2}$	0	8.651	3.536
$D_{1,2}$	0	0.0008375	0.03190
$D_{0,1,2}$	0	0	0.09199

4.2.2 Gerstner and Griebel adaptive sparse grid results

The results of the Gerstner and Griebel with each step is given in Table 4.2. To summarize, this approach uses two index sets: the 'old' and the 'active' index set. The 'old' index set is a list of all the blocks currently in the grid. The 'active' index set contains the forward neighbouring blocks of the 'old' index set. For each block in the 'active' index set, an error estimate is calculated. In each step, the block with the largest error estimate is moved from the 'active' to the 'old' index set, and the new forward neighbours with corresponding error estimates are determined. The table shows for each step the number of points in the 'old' and 'active' index sets, the mean calculated using the grid based on the 'old' index set. The last column shows which block is added to the 'old' index set in the next iteration.

The data set available is limited to a level 4, 3D standard sparse grid, so any multi-indices outside the level 4 simplex are not available. The algorithm starts with one block: the multi-index $(1, 1, 1)$. The adaptive algorithm then begins by refining in the direction of the first parameter, until it reaches the multi-index $(4, 1, 1)$. Note that the forward neighbour $(5, 1, 1)$ is not a part of the data set available to us. Since the error estimate tends to become smaller with each refinement, it is assumed that the error estimate for this forward block $(5, 1, 1)$ is small enough that the algorithm would not refine further in this direction. Following this assumption, multi-index $(5, 1, 1)$ is considered to be part of the 'active' index set (and thus contributes to the number of grid points), but can never become part of the 'old' index set. This is why all the following entries for the 'active' index set are followed by a *. The two alternatives to this assumption would be to either perform more simulations, or to stop the algorithm. Since the assumption seems realistic, the algorithm is continued under the mentioned restrictions. In the last step one multi-index is moved from the 'active' to the 'old' set, but new forward neighbouring blocks are not considered. This is why the number of grid points stays constant (indicated by **).

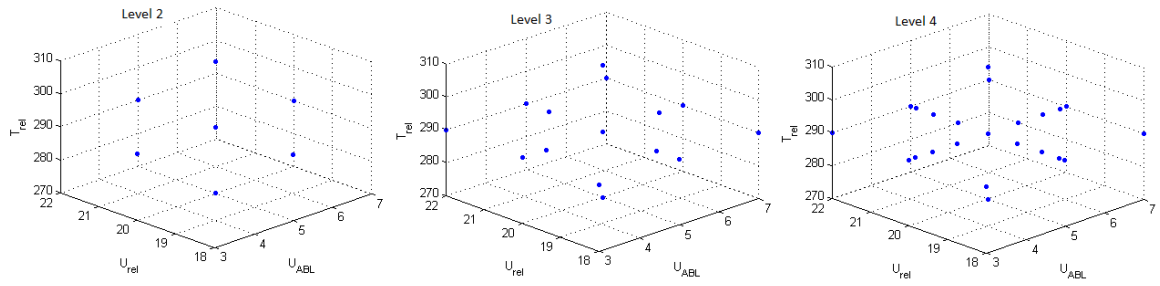


Figure 4.4: Evolution of the Sobol adaptive sparse grid for the heavy gas release. Going from left to right: grid points for level $l = 2, 3$ and 4.

Table 4.2: Evolution of the mean following the Gerstner & Griebel adaptive approach [18]. The * and ** indicate that refinement in the direction of the first parameter is not possible, since the data set is limited to a level 4 sparse grid. To investigate the evolution of the algorithm, the assumption is made that the refinement in this direction is already sufficient.

Uniform input parameter distribution.

Grid points 'old' (+'active') index set	Mean	Add multi-index to 'old' index set
1 (7)	180.04	(2, 1, 1)
3 (9)	184.65	(3, 1, 1)
5 (13)	182.44	(4, 1, 1)
9 (21*)	182.22	(1, 1, 2)
11 (27*)	182.41	(2, 1, 2)
15 (31*)	183.00	(3, 1, 2)
19 (31**)	182.81	/

Normal input parameter distribution.

Grid points 'old' (+'active') index set	Mean	Add multi-index to 'old' index set
1 (7)	180.04	(2, 1, 1)
3 (9)	182.77	(3, 1, 1)
5 (13)	181.02	(4, 1, 1)
9 (21*)	180.68	(1, 1, 2)
11 (27*)	180.79	(2, 1, 2)
15 (31*)	180.99	(3, 1, 2)
19 (31**)	180.91	/

4.2.3 Sobol adaptive sparse grid algorithm

The evolution of the Sobol adaptive sparse grid from level 2 to level 4 for the heavy gas release is shown in Figure 4.4. The values of the Sobol variances on the adaptive grid are given in Table 4.3. The adaptive algorithm starts for a standard level 2 sparse grid, which is why the results are exactly the same as before. From the initial grid, D_0 and D_1 are the most important contributions to the total variance. The cutoff value has been set to 0.95, meaning that D_2 is considered insignificant. As a result of this, the refinement of the grid only occurs in the directions of U_{ABL} and U_{rel}. Because of this, $D_{0,2}$, $D_{1,2}$ and $D_{0,1,2}$ will always be 0 in this case. The second refinement only adds points in the direction of U_{ABL} and U_{rel}. (4)

Table 4.3: Evolution of the Sobol variances on the adaptive sparse grid. Indices $0, 1$ and 2 correspond to the atmospheric boundary layer velocity U_{ABL}, the propane release velocity U_{rel} and the propane release temperature T_{rel} respectively.

Uniform input parameter distribution.

Variable	Level 2	Level 3	Level 4
Grid points	7	15	23
Mean	184.7	182.5	182.4
Variance	446.2	309.6	312.5
D_0	375.9	222.6	225.6
D_1	60.18	72.71	72.80
D_2	10.15	10.15	10.15
$D_{0,1}$	0	3.826	4.013
$D_{0,2}$	0	0	0
$D_{1,2}$	0	0	0
$D_{0,1,2}$	0	0	0

Normal input parameter distribution.

Variable	Level 2	Level 3	Level 4
Grid points	7	15	23
Mean	182.8	181.1	180.8
Variance	264.4	150.3	151.1
D_0	222.7	102.9	103.4
D_1	35.66	40.01	40.08
D_2	6.014	6.014	6.014
$D_{0,1}$	0	1.343	1.546
$D_{0,2}$	0	0	0
$D_{1,2}$	0	0	0
$D_{0,1,2}$	0	0	0

4.3 Discussion

This section will discuss the results of the simulations to see which parameters are important. Then, the performance of the two adaptive grid refinement methods is compared to a standard sparse grid and to each other.

4.3.1 Flow characteristics

The Sobol indices computed on the standard sparse grid indicate that U_{ABL} and U_{rel} are the two most important parameters. Before further discussing the results of the different methods, we will look at response surface plots to investigate local behavior. This can provide additional information on the response, since Sobol variances by definition are global measures. Since

T_{rel} seems to have the least effect, the variation of the effect distance with respect to U_{ABL} and U_{rel} is plotted in Figure 4.5 for constant values of T_{rel}. The following trends become clear from these plots:

- The effect distance increases with increasing U_{rel} and with decreasing U_{ABL} and T_{rel}. U_{ABL} is the most important parameter followed by U_{rel}.
- T_{rel} does not have a big effect on its own, but there is an interaction between U_{ABL} and T_{rel}. When both values are low, the effect distance increases even more than just the separate contributions.
- The variation of effect distance with U_{ABL} comes mainly from values between 3 and $4m/s$, where the gradient becomes much steeper.
- A low temperature seems to add to above effect, but it is still limited to the same range of $U_{ABL} = 3 - 4m/s$.
- The maximum effect distance occurs for $U_{ABL} = 3m/s$, $U_{rel} = 22m/s$ and $T_{rel} = 270K$. The minimum effect distance occurs for $U_{ABL} = 7m/s$, $U_{rel} = 18m/s$ and $T_{rel} = 310K$. This is in line with the first two trends.

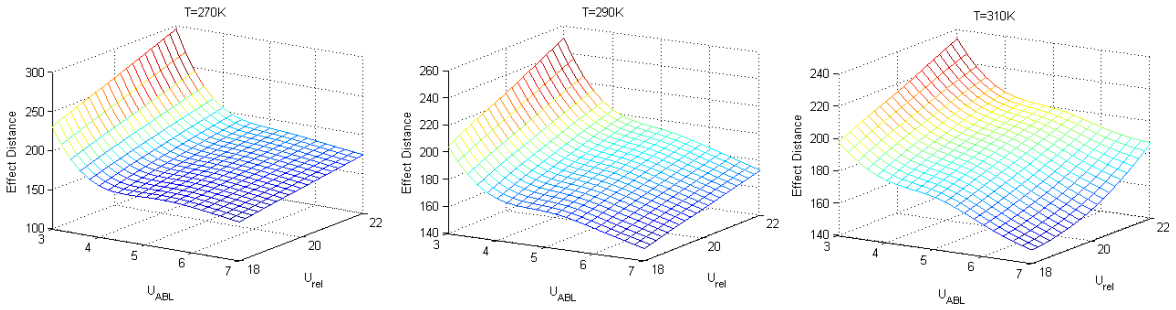
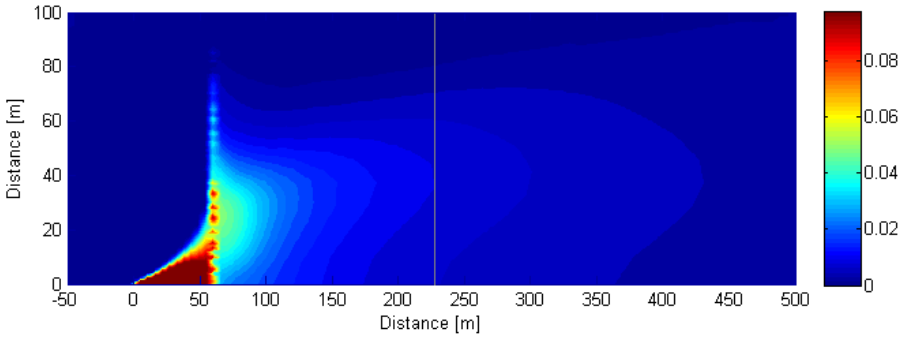


Figure 4.5: Effect distance as a function of U_{ABL} and U_{rel} for $T_{rel} = 270K$, $290K$ and $310K$.

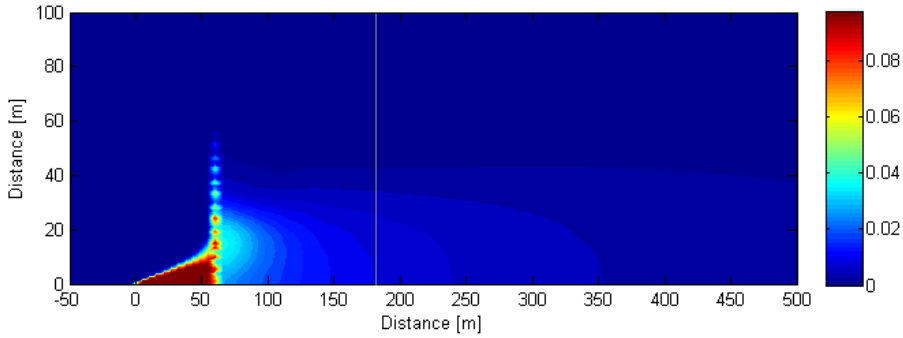
The behavior of the gas in front of upstream of the barrier is very important, since it strongly affects the profile of the propane concentration behind the barrier. This in turn affects the effect distance. In order to examine this, the molar concentration profile in the entire domain can be plotted for different inputs. From these plots, it is confirmed that the boundary layer velocity U_{ABL} has by far the largest influence on the effect distance, as was indicated by the Sobol variances. To illustrate this, the molar concentration is plotted for 3 values of U_{ABL}: 3, 5 and $7m/s$ in Figure 4.6. The other 2 parameters are kept at their mean values. Plots for other input values do not add any new information, and are omitted for this reason.

The wind velocity profile is dependent on the boundary layer velocity, ranging from 3 to $7m/s$. Once the gas is released, it will start mixing with the air. As a result of this, the velocity of the air-propane mixture will go down from the original release temperature and get closer to the surrounding air velocity. The value of U_{ABL} will thus strongly affect the velocity of the propane cloud when it reaches the barrier.

(a) $U_{ABL} = 3m/s$.



(b) $U_{ABL} = 5m/s$.



(c) $U_{ABL} = 7m/s$.

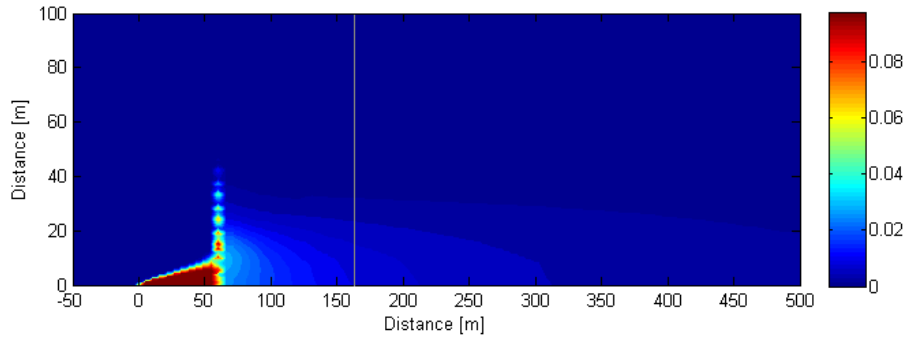


Figure 4.6: Molar concentration of propane gas over the computational domain at a height of $1m$ for different values of U_{ABL}. T_{rel} and U_{rel} are kept constant at $290K$ and $20m/s$ respectively. The thin gray lines indicate the effect distance.

The release velocity U_{rel} will also affect the velocity of the propane cloud prior to reaching the barrier. For higher values of U_{rel}, the propane cloud will reach the barrier at higher velocities, though the effect is not as significant as for U_{ABL}.

Finally, the propane gas has a higher density than air and will gravitate to the ground. Of course, this effect depends on the concentration of the propane gas. As the air and propane gas start mixing, this effect will become less strong. The release temperature of the gas ranges between 270 and 310 Kelvin. As a result of this, the density of the gas will vary, going down when the temperature increases. In this specific case, the variation in density is very small, so that the effect of the release temperature is negligible.

The reason the velocity of the cloud is so important is because of how it affects the buildup of gas near the barrier. If the propane cloud is moving at a high speed, it will flow over the barrier more easily, and the tangential velocity component due to impact is less important. If the cloud is moving at a relatively low speed, the momentum of the gas cloud is much lower and the buildup of gas becomes much larger. This effect is clearly visible in Figure 4.6. When this happens, the tangential velocity component becomes much more significant and the cloud of gas will flow over the barrier at a point further away from the symmetry plane. This in turn affects the propane concentration contour profiles downstream of the barrier, which is also visible in the figure.

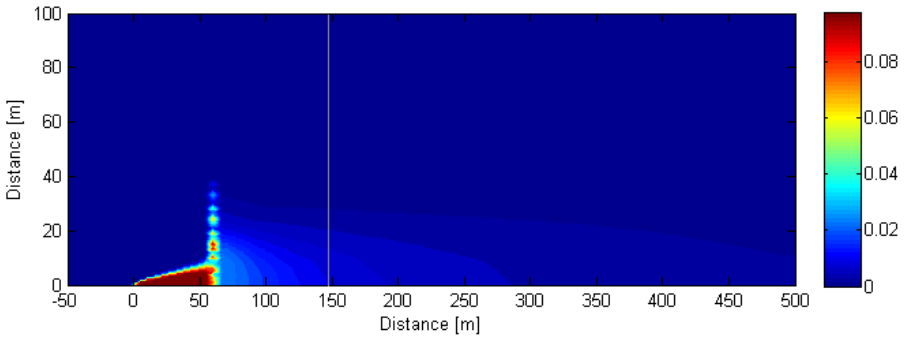
An interesting observation is that this also increases the effect distance. Data was only collected in the plane $z = 1m$, so a 3-dimensional representation of the cloud is not available, but it is certain that the effect distance measurements are greatly affected by the height at which it is measured. The highest effect distances are found when the gas cloud velocity upstream of the barrier is low. This means that the velocity downstream of the barrier will be low as well. A likely explanation is that for these lower velocities, the effect of the wake downstream of the barrier is smaller. Because of this, the gas will be closer to the ground. For higher gas cloud velocities, the wake has a much larger effect and the gas is more spread out in the z-direction. Since the molar concentrations are only measured in a plane at a height of $1m$, the effect distances measured are much closer to the barrier.

Finally, for completeness, the propane concentration contour plots are given in Figure 4.7.

4.3.2 Comparison

Let's continue by comparing the mean computed with the three methods. The data is taken from Tables 4.1, 4.2 and 4.3. Both adaptive methods get a good estimation of the mean with a significant reduction in the grid points. For this case, the Gerstner & Griebel approach gives almost exactly the same results as the standard sparse grid. The Sobol adaptive approach gives a slightly different value, but since it differs only a fraction of a percent it is not significant. The results for both input distributions follow the same trend for this case, although the values differ. The variance is smaller when assuming a normal input distribution. This is expected since a large part of the variation of the effect distance occurs near the boundaries of the domain.

(a) $U_{ABL} = 3m/s$, $U_{rel} = 22m/s$, $T_{rel} = 270K$.



(b)

$U_{ABL} = 7m/s$, $U_{rel} = 18m/s$, $T_{rel} = 310K$.

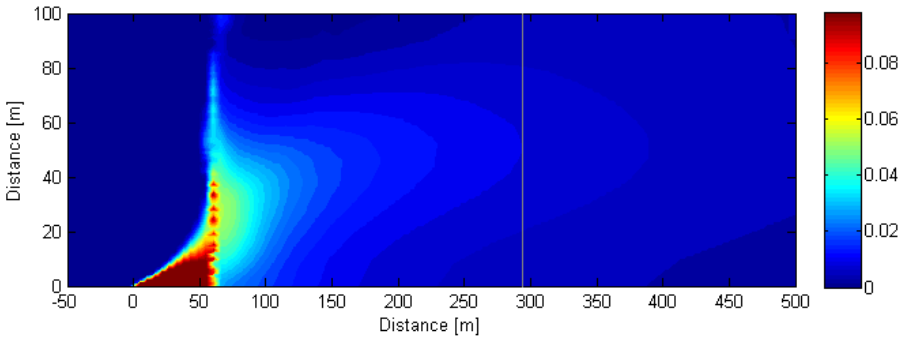


Figure 4.7: Molar concentration of propane gas over the computational domain at a height of $1m$ for minimum (a) and maximum (b) effect distance, which are indicated by the thin grey lines.

The main difference between the two adaptive grid refinement methods is that the Sobol adaptive method does not look forward, the only information used to determine the refinements is the Sobol variances on the current grid. Because it uses a cutoff value, convergence is not strictly guaranteed for a high number of refinement steps. The Gerstner & Griebel approach does look forward, and chooses its refinements based on an error indicator. This method will converge given enough refinement steps, but since it has to look forward, this comes at a cost of requiring extra data samples. Finally, the grids from the three methods

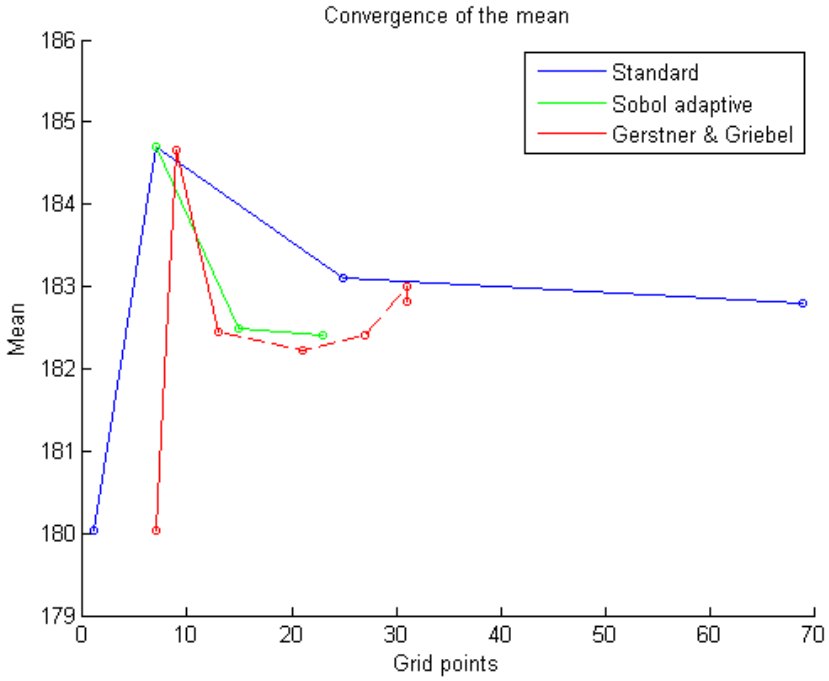


Figure 4.8: Convergence of the mean for the standard sparse grid, Sobol adaptive sparse grid and the Gerstner & Griebel adaptive sparse grid, assuming a uniform input parameter distribution.

are compared in block diagrams in Figure 4.9. For the first two diagrams, the grey blocks represent the multi-indices that are a part of the final grid. For the third diagram, the grey blocks indicate the multi-indices which are a part of the 'old' index set, using which the mean is calculated. The red lined blocks are a part of the 'active' index set, for which the error indicator is calculated, but which are not added to the grid. The pink block is the multi-index $(5, 1, 1)$. Strictly speaking, it is not part of the data set. The reasons why it was included anyways are explained in Section 4.2.2.

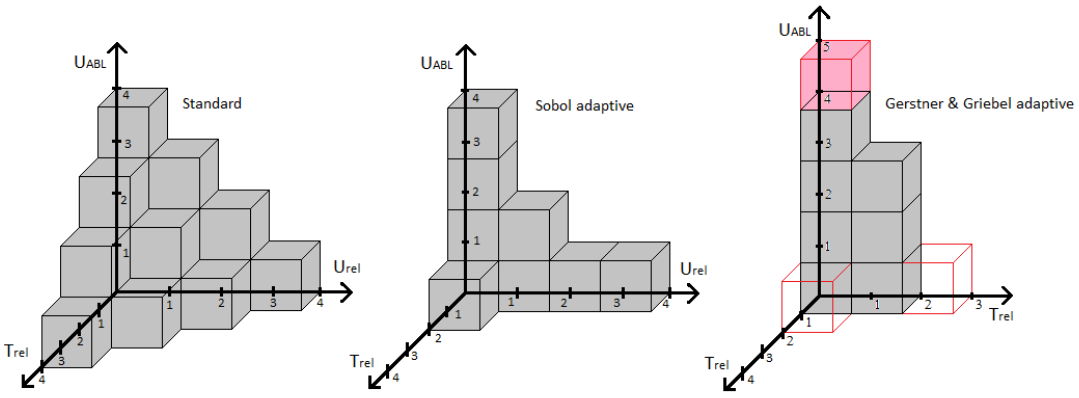


Figure 4.9: Block diagram representation of the final grid for the heavy gas release. From left to right: standard sparse grid, Sobol adaptive sparse grid and Gerstner & Griebel adaptive sparse grid. For the third diagram, the grey blocks represents the 'old' index set, while the red blocks belong to the 'active' index set. Also note that the U_{rel} and T_{rel} axes are switched for convenience.

Chapter 5

Conclusions and Recommendations

5.1 Conclusions

The goal of this study was to contribute to a new method to perform uncertainty quantification analysis, specifically for industrial, computationally intensive CFD simulations. To that end, an adaptive grid refinement algorithm was proposed and applied to a real-life industrial application.

First, a code was developed which can perform interpolation and integration using polynomial and linear hat basis functions, based on the mathematical framework introduced in Chapter 2. The quadrature weights for two input parameter distributions were implemented. This implementation was verified in Chapter 3, by using a set of test functions. It has become clear that the polynomial integration and interpolation generally outperforms its linear counterpart, since the polynomial method will show spectral convergence. In case there is a discontinuity, the performance of polynomial basis functions will deteriorate sginificantly. This is due to the effect known as the Gibbs' phenomenon, which means the discontinuity can not be resolved using polynomial, which introduces oscillations in the approximation. The linear approximation method is not affected by this effect, but since the convergence is generally quite slow it is only used as a backup in case polynomial approximation is inaccurate. The weights for both input parameter distributions were verified by comparing the integral results to the exact values where available, and to Monte Carlo results otherwise. The accuracy of the results obtained from Monte Carlo simulations is dependent on the number of samples. The effect of this is estimated by calculating the standard error, for all test functions the results of the code were sufficiently close to the Monte Carlo results so we can conclude that the implementation is correct.

The new adaptive grid refinement is based on the approximation of Sobol variances and was introduced in Section 2.8. Based on the values of the Sobol variances, more grid points are

added in the appropriate directions. In Chapter 4, the method is compared to a standard sparse grid approach and a different grid refinement approach by Gerstner and Griebel [18]. For the application considered, the values obtained with the Gerstner and Griebel approach were shown to be closer to the sparse grid approach. However, the Sobol refinement method used only 23 samples, compared to 31 in the Gerstner and Griebel method, and 69 for a standard sparse grid.

The advantages of the Sobol adaptive grid refinement method are that the refinements are based on an intuitive measure, the Sobol variances, which indicate which interactions are the most significant. Based on the result in Chapter 4, we can say that this method provides a good approximation of the response, while significantly reducing the computational cost. It seems especially suited for problems of low to moderate dimensions where the computational resources are limited.

The main disadvantage of the method is that convergence to the real response is not guaranteed for a high number of refinement steps. Since the method is intended for industrial applications, this disadvantage is marginal. It will still provide a good estimation of the response at a low computational cost.

5.2 Recommendations

This section will provide a few recommendation for the method moving forward:

- As already mentioned in the previous section, a problem was encountered when calculating Sobol variances using linear hat basis functions. In order to evaluate the integrals exactly, piecewise quadratic integration needs to be implemented. For now, the problem is partly circumvented by applying a refinement to the grid and using the linear integration method.
- The current algorithm will not converge to the exact solution when increasing the number of grid points. This is because of the introduction of a cut-off value, which limits the directions in which points are added. Although the effect of this seems insignificant for industrial applications, since the number of computation is limited, it would still be worth looking into. If the algorithm has this property, it can be more easily compared to other methods, and will be more reliable when more computational resources are available.
- Right now, the grid refinement is based solely on the Sobol variances on the current grid. It would be interesting to look into a way to incorporate information from previous steps into the decision making process. This could provide the user with an additional reduction of grid points. Depending on the approach, it could also provide a solution for the second recommendation.
- Only two input parameter distributions have been implemented. However, a similar approach to what is described in 2.4 can be used to compute the quadrature weights

for other distributions.

- It can be convenient to use different distributions for different input parameters. Right now, this option is not available; one distribution is chosen and it is then applied to all input parameters.
- More applications are required to test the Sobol adaptive refinement grid approach. The results presented in Chapter ch:application indicate that this method is promising, but to be sure of this it should be applied to a large range of industrial cases. An investigation on the effect of the cut-off value on the result for the mean and variance for different cases would provide a better view of the accuracy of the method.

Bibliography

[1] P. L. Chebyshev, "Théorie des mécanismes connus sous le nom de paraléllogrammes", Imprimerie de l'Académie Impériale des sciences, 1853.

[2] R. E. Bellman, "Adaptive control processes: a guided tour", Princeton University Press, 1961.

[3] W. L. Oberkampf and S. Ferson, "Model validation under both aleatory and epistemic uncertainty", from "Proceedings of NATO RVO-AVT-147 symposion on Computational Uncertainty", Athens, Greece, 2007.

[4] A. Cunha Jr. et al, "Uncertainty Quantification through the Monte Carlo method in a cloud computing setting.", Computer Physics Communications (Computer Physics Communications, Vol. 185): p.13551363, Elsevier, 2014.

[5] T. Gerstner, M. Griebel, "Sparse Grids", Encyclopedia of Quantitative Finance , R. Cont (ed.), Wiley, 2008.

[6] N. Wiener, "The Homogeneous Chaos", American Journal of Mathematics, Vol. 60, No. 4: page 897 - 936, 1938.

[7] D. Xiu, "Numerical Methods for Stochastic Computation: A Spectral Method Approach", Princeton University Press, 2010.

[8] D. Xiu and G. E. Karniadakis, "The Wiener-Askey Polynomial Chaos for Stochastic Differential Equations", Journal of Scientific Computing, Vol. 24, No. 2: pages 619-644, 2002.

[9] X. Wan and G. E. Karniadakis, "Beyond Wiener-Askey Expansions: Handling Arbitrary PDFs", Journal of Scientific Computing, Vol. 27, Nos. 1-3: pages 619-644, 2006.

[10] G. H. Golub and J. H. Welsch, "Calculation of Gauss Quadrature Rules", 1968.

[11] W. Gautschi, "Generalized Gauss-Radau and Gauss-Lobatto Formulae", BIT Numerical Mathematics, Vol. 44, Issue 4: pages 711-720, 2004.

[12] D. P. Laurie, "Calculation of Gauss-Kronrod Quadrature Rules", Mathematics of Computation, Vol. 66, No. 219: pages 1133-1145, 1997.

[13] L.N. Trefethen, "Is Gauss Quadrature Better than Clenshaw-Curtis?", SIAM Rev., Vol. 50, No. 1: pages 67-87, 2008.

[14] I. Babuska et al., "A stochastic collocation method for elliptic partial differential equations with random input data", SIAM Journal of Numerical Analysis, Vol. 45, No. 2: page 1005-1034, 2007.

[15] A. Sommariva,"Fast Construction of Fejér and Clenshaw-Curtis rules for general weight functions", Computers & Mathematics with Applications, Vol. 65, Issue 4: page 682-693, 2013.

[16] I. M. Sobol, "Global Sensitivity Indices for Nonlinear Mathematical Models. Review", Willmot Magazine, 2001.

[17] G. Tang et al., "Global Sensitivity Analysis for Stochastic Collocation Expansion," paper AIAA-2010-2922 in Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (12th AIAA Non-Deterministic Approaches Conference), Orlando, FL, Apr 12-15, 2010.

[18] T. Gerstner and M. Griebel, "Dimension-Adaptive Tensor-Product Quadrature", Department of Applied Mathematics, University of Bonn, 2003.

[19] G. Blatman and B. Sudret, "Efficient computation of global sensitivity indices using sparse polynomial chaos expansions", Reliability Engineering and System Safety, Vol. 95: page 1216-1229, 2010.

[20] S. A. Smolyak, "Quadrature and interpolation formulas for tensor products of certain classes of functions", Dokl. Akad. Nauk. SSSR 4, page 240 - 243, 1963.

[21] H. J. Bungartz and M. Griebel, "Sparse Grids", Acta Numerica: p1-123, 2004.

[22] R. Srinivasanm "Importance sampling - Application in Communications and Detection", Springer-Verlag, New York, 2002.

[23] J. Burkhardt, "1D Quadrature rules for Sparse Grids", Department of Scientific Computing, Florida State University, 2014.

[24] R.P. Dwight and Z.-H. Han, "Efficient Uncertainty Quantification using Gradient-Enhanced Kriging", AIAA Conference on Non-Deterministic Approaches, Palm Springs, AIAA-2009-2276, 2009.

[25] D.R. Jones, M. Schonlau and W.J. Welsch, "Efficient global optimization of expensive black-box functions", Journal of Global Optimization, Vol. 13: page 455492, 1998.

[26] G.J.A. Loeven, "Efficient uncertainty quantification in computational fluid dynamics", 2010.

[27] "Prof. D.G. Krige: Biography", National Academy of Engineering, retrieved September 3, 2014, from: http://www.nae.edu/MembersSection/Directory20412/31144.aspx

[28] J.H.S. De Baar, T.P. Scholcz, C.V. Verhoosel, R.P. Dwight, A.H. van Zuijlen and H. Bijl, "Efficient Uncertainty Quantification with Gradient-Enhanced Kriging: Applications in FSI", European Congress on Computational Methods in Applied Sciences and Engineering, Vienna, Austria, September 10-14 2012.

[29] J. Burkardt, "The Sparse Grid Interpolant", Department of Scientific Computing, Florida State university, September 13 2012.

[30] J.A.S Witteveen, "Efficient and Robust Uncertainty Quantification for Computational fluid Dynamics and Fluid-Structure Interaction", 2009.

[31] J.A.S. Witteveen, H. Bijl, "Modeling arbitrary uncertainties using Gram-Schmidt polynomial chaos", 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, Nevada, AIAA-2006-896, 2006.

[32] C.Lacor, C. Dinescu, C. Hirsh, S. Smirnov, "Implementation of Intrusive Polynomial Chaos in CFD Codes and Application to 3D Navier-Stokes", from Uncertainty Quantification in Computational Fluid Dynamics, pp.193-223, 2013.

[33] J. Waldvogel, "Fast Construction of the Fejér and Clenshaw-Curtis Quadrature Rules", BIT Numerical Mathematics, Vol. 43, No. 1, pp. 001-018, 2003.

[34] J. Anderson, "Fundamentals of Aerodynamics", 4th Edition (International Edition), McGraw-Hill Book Company, 2007.

[35] F. M. White, "Viscous Fluid Flow", 3rd Edition (International Edition), McGraw-Hill Book Company, 2006.

[36] L. F. Moody, "Friction factors for pipe flow", Transactions of the ASME 66 (8): 671684, 1944.

[37] Mathesongas, "Pure gases - Propane", retrieved from https://www.mathesongas.com/pdfs/products/Propane-Pure-Gas.pdf on 24/09/2014.

[38] ANSYS ^{TM} Fluent, Version 14.5, Released 2012, www.ansys.com/.

[39] A. Genz, "Testing multidimensional integration routines", Tools, Methods, and Languages for Scientific and Engineering Computation, Elsevier North-Holland, pages 81-94, 1984.

Appendix A

Sommariva Algorithm

Below is the Matlab code used to calculate the weights for polynomial integration as detailed in section 2.4. First, the weighted Chebyshev polynomials are determined, and then the corresponding weights are calculated using a discrete cosine transform. The procedure is treated in more detail in [15].

A.1 Matlab code

```
function [ w, x ] = Sommariva_CC( L, distr )
% This function will calculate the polynomial weights for a uniform or
% normal distribution (95% at boundaries, implement flexibility later).
% The algorithm used is taken from the paper by Sommariva:
% 'Fast Contruction of Fejer and Clenshaw-Curtis rules for general
% weight functions', 2012.

n=2^(L-1)+1;
V=zeros(2^(L-1)+1);
% Used for the normal and truncated normal distributions.
confidence=0.95;
if strcmp(distr,'uniform')==true

    % integrals of Chebyshev polynomials
    for i=1:n/2
        g(i)=2/(1-(2*i)^2);
    end
%     g

    for i=1:n
```

```
        if i==1
            g_k(i)=2;
        elseif mod(i,2)==0
            g_k(i)=0;
        else
            g_k(i)=g((i-1)/2);
        end
    end

%     g_k
    moms=g_k/2;

elseif strcmp(distr,'normal_trunc')==true
    % confidence interval boundaries are at +- sig* sigma
    sig=erfinv(confidence)*sqrt(2);
    sigma=1/sig;
    % constants, easier for use later on
    c_1=1/(sigma*sqrt(2*pi));
    c_2=1/(2*sigma*sigma);

    % T contains the indices of the Chebyshev polynomials, these are used
    % later on for the computation of the weights.
    % for example N=4: T0=1, T1=x, T2= 2x^2-1, T3=4x^3-3x
    %
    % Then T becomes [ 0 0 0 1;
    %                  0 0 1 0;
    %                  0 2 0 1;
    %                  4 0 -3 0]

    T=zeros(n,n);
    T(1,1)=1;
    T(2,2)=1;
    for i=3:9
        for j=1:i
            T(i,j+1)=T(i-1,j)*2;

        end
        T(i,:)=T(i,:)-T(i-2,:);
    end
    T=T(1:n,1:n);

    % I contains the integrals of N(x), x*N(x), x^2*N(x), x^3*N(x), ...
    % Note that, since N(x) is symmetric and x^N for uneven N is
    % antisymmetric, the integral of x^N*N(x) becomes zero in these cases.
    I=zeros(length(V(1,:)),1);

    I(1)=confidence; % I0
```

```
    for k=2:length(V(1,:))

        if mod(k,2) == 0
            % integral is zero for odd functions (I1,I3,I5,...)
            I(k)=0;
        else
            % Matlab doesn't handle index k=0 so k-1 in the expression
            % becomes (k-1)-1 since we start from k=1 instead of k=0
            I(k)=-c_1/c_2*exp(-c_2)+((k-1)-1)/(2*c_2)*I((k-1)-1);

        end

    end
    %scaling

    I=I/confidence;

    % The moments are equal to the integrals of the Chebyshev polynomials
    % multiplied with the Normal distribution.
    % As an example: I4=integral of T4*N(x)= integral of (4x^3-3x)*N(x)
    % this can be rewritten as 4 * integral of x^3*N(x) -3 * integral of
    % x*N(x). Which is the same as multiplying the 4th row of the T matrix
    % with the I vector.

    moms=T*I;

else
    disp('Wrong input, invalid distribution type')
end

% Sommariva algorithm, see paper: 'Fast Contruction of Fejer and
% Clenshaw-Curtis rules for general weight functions', 2012.
% Computes the weights and abscissa locations, input required are the
% moments computed above.

if L==1
    %disp('something')
    w=[1];
    x=[0];
else

momscc(1:n-2)=1/2*(moms(1:n-2)-moms(3:n));
% disp(length(momscc));
theta=(1:n-2)'*pi/(n-1);

xx=cos(theta);
% length(xx);
```

```
w=((2*sin(theta)/(n-1)).*dst(momscc')).(1-xx.^2);
w1=(2*sum(moms)-moms(1)-moms(end))/(2*(n-1));
wn=moms(1)-w1-sum(w);
x=[1;xx;-1]; w=[w1;w;wn];



for i=1:length(x)
    if abs(x(i))<1.0e-10
        x(i)=0;
    end
end
end
end
```

Appendix B

Heavy gas release: results

The tables below contains the results of the simulations the barrier case. There are a total of 69 simulations, corresponding to a standard sparse grid of level $l = 4$ in 3 dimensions.

Table B.1: Input values and results of the barrier case.

$U_{ABL}[m/s]$	$U_{rel}[m/s]$	$T_{rel}[m/s]$	Effect distance $[m]$
5	20	290	180.04
3	20	290	226.67
7	20	290	161.04
5	18	290	166.23
5	22	290	193.10
5	20	270	175.09
5	20	310	186.11
3.585786	20	290	198.50
6.414214	20	290	167.14
3	18	290	204.35
7	18	290	149.08
3	22	290	250.17
7	22	290	172.94
5	18.58579	290	170.28
5	21.41421	290	189.29
3	20	270	262.36
7	20	270	162.29
3	20	310	215.01
7	20	310	159.20
5	18	270	160.90

Table B.2: Input values and results of the barrier case.

$U_{ABL}[m/s]$	$U_{rel}[m/s]$	$T_{rel}[m/s]$	Effect distance $[m]$
5	22	270	188.37
5	18	310	172.13
5	18	310	199.47
5	20	275.8579	176.06
5	20	304.1421	184.59
3.152241	20	290	217.83
4.234633	20	290	184.57
5.765367	20	290	173.81
6.847759	20	290	162.67
3.585786	18	290	179.47
6.414214	18	290	154.59
3.585786	22	290	215.46
6.414214	22	290	179.48
3	18.58579	290	211.5
7	18.58579	290	152.78
3	21.41421	290	243.35
7	21.41421	290	169.32
5	18.15224	290	167.26
5	19.23463	290	174.91
5	20.76537	290	185.23
5	21.84776	290	192.10
3.585786	20	270	206.16
6.414214	20	270	167.00
3.585786	20	310	200.23
6.414214	20	310	166.62
3	18	270	230.94
7	18	270	149.6
3	22	270	294.59
7	22	270	174.77
3	18	310	197.47
7	18	310	148.36
3	22	310	233.67
7	22	310	186.56
5	18.58579	270	165.05
5	21.41421	270	184.48

Table B.3: Input values and results of the barrier case.

$U_{ABL}[m/s]$	$U_{rel}[m/s]$	$T_{rel}[m/s]$	Effect distance $[m]$
5	18.58579	310	176.28
5	21.41421	310	195.67
3	20	275.8579	249.58
7	20	275.8579	161.98
3	20	304.1421	217.17
7	20	304.1421	159.80
5	18	275.8579	161.58
5	22	275.8579	189.49
5	18	304.1421	170.56
5	22	304.1421	197.64
5	20	271.5224	175.36
5	20	282.3463	177.45
5	20	297.6537	182.60
5	20	308.4776	185.72