CRANFIELD UNIVERSITY

MARINA MENSHIKOVA

UNCERTAINTY ESTIMATION USING THE MOMENTS METHOD FACILITATED BY AUTOMATIC DIFFERENTIATION IN MATLAB

DEFENCE COLLEGE OF MANAGEMENT AND TECHNOLOGY

PhD

CRANFIELD UNIVERSITY

DEFENCE COLLEGE OF MANAGEMENT AND TECHNOLOGY

DEPARTMENT OF ENGINEERING SYSTEMS AND MANAGEMENT

PhD THESIS

Academic Year 2009-2010

Marina Menshikova

Uncertainty Estimation Using the Moments Method Facilitated by Automatic Differentiation in Matlab

Supervisor: Dr. S. Forth

January 2010

Abstract

Computational models have long been used to predict the performance of some baseline design given its design parameters. Given inconsistencies in manufacturing, the manufactured product always deviates from the baseline design. There is currently much interest in both evaluating the effects of variability in design parameters on a design's performance (uncertainty estimation), and robust optimization of the baseline design such that near optimal performance is obtained despite variability in design parameters. Traditionally, uncertainty analysis is performed by expensive Monte-Carlo methods. This work considers the alternative moments method for uncertainty propagation and its implementation in Matlab.

In computational design it is assumed a computational model gives a sufficiently accurate approximation to a design's performance. As such it can be used for estimating statistical moments (expectation, variance, etc.) of the design due to known statistical variation of the model's parameters, e.g., by the Monte Carlo approach. In the moments method we further assume the model is sufficiently differentiable that a Taylor series approximation to a model may be constructed, and the moments of the Taylor series may be taken analytically to yield approximations to the model's moments.

In this thesis we generalise techniques considered within the engineering community and design and document associated software to generate arbitrary order Taylor series approximations to arbitrary order statistical moments of computational models

implemented in Matlab; Taylor series coefficients are calculated using automatic differentiation. This approach is found to be more efficient than a standard Monte Carlo method for the small-scale model test problems we consider. Previously Christianson and Cox (2005) have indicated that the moments method will be non-convergent in the presence of complex poles of the computational model and suggested a partitioning method to overcome this problem. We implement a version of the partitioning method and demonstrate that it does result in convergence of the moments method. Additionally, we consider, what we term, the branch detection problem in order to ascertain if our Taylor series approximation might only be valid piecewise.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The traditional engineering approach to problem solving is to logically design and implement systems, coupled with continual assessment of performance and failure modes. This leads to future improvements of the original design. With the development of computers and technical software, engineers became able to simulate complex models of engineering systems. This allowed them to determine weak points of the system via simulation, then eliminate or ameliorate them, so reducing the costs of implementation, or ownership, or costs associated with failure.

While facing design problems, engineers normally systematise the process into several steps. One of these classifications is described by Ayyub, [1].

- First, one should identify the problem.
- By identifying the problem, one must define the objectives. All the known and unknown variables must be stated.
- The next step is to develop possible solutions of the problem and evaluate them, choosing the most appropriate one. While doing that, one should consider the associated uncertainties and be able to assess the array of possible outcomes.

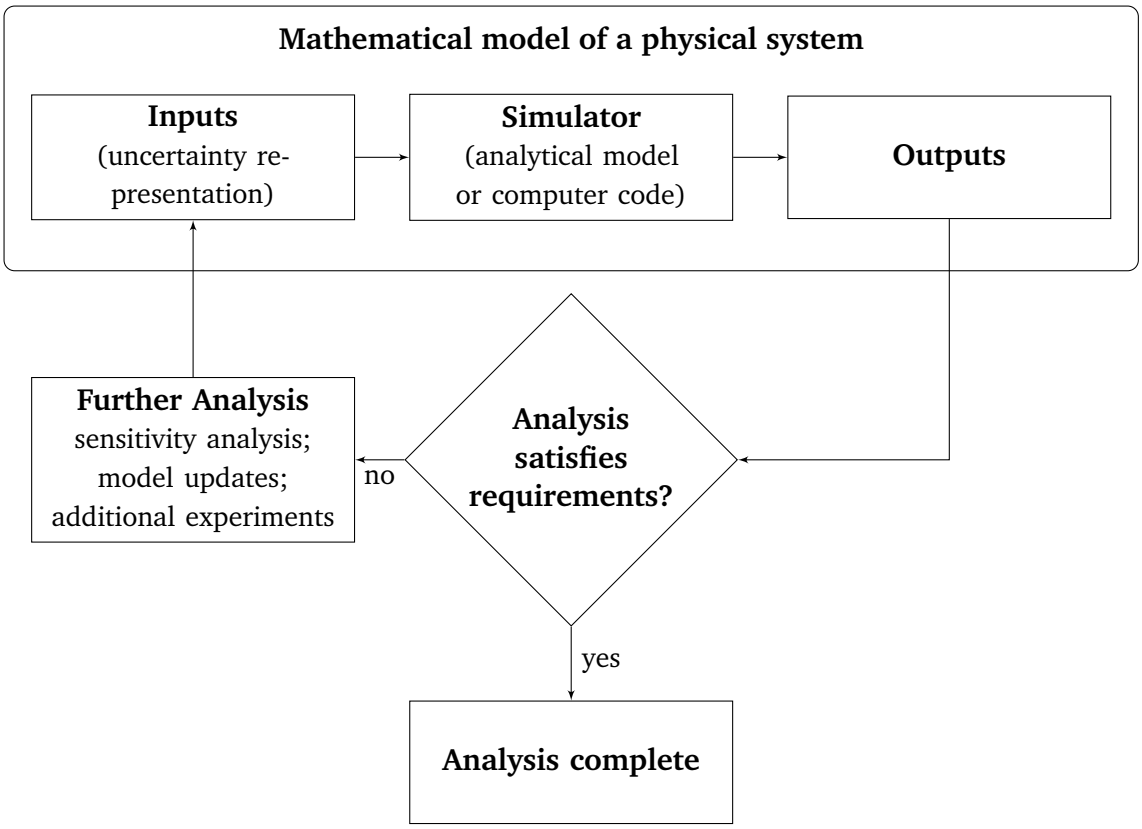


Figure 1.1: Propagation of uncertainties in mathematical modelling according to Oberkampf et al. [2].

- Finally, the best alternative gets implemented.

Probability and statistics play a very important role in ensuring that every task is handled properly.

In the paper by Oberkampf et al. [2] the engineering design problem process is described via the diagram, see Figure 1.1.

Engineering decision problems are subdivided into single- or multiple-objective problems. Multidisciplinary design optimisation is a methodology for the design of systems in which strong interactions between disciplines motivates designers to simultaneously manipulate variables in several disciplines. It involves the coordination of analysis techniques from several disciplines to realise more effective solutions during the design and optimisation of complex systems.

The inputs of such analysis for a given problem (single-/multiple-objective) are often assumed to be precisely known. And the studies of such cases are called **deter-**

ministic analysis. However, a major difficulty arises when solving problems where the inputs are uncertain.

If we look, for example, at the simulation of an aeroplane wing during flight, engineers must know how much deformation is likely to occur. This can be calculated using a finite element analysis. However, to use such an analysis, there are a number of inputs required, such as the elastic modulus and surface roughness of the wing materials. While these are known for specific materials, they are likely to vary slightly as a result of manufacturing processes. To account for that, all calculations should include some measure of the uncertainty. Ultimately, this will allow for confidence to be placed in the figures provided by the software, and to determine if the values fall below the accepted factor of safety.

This example of the aeroplane wing over-simplifies the scenario. There are of course many other important factors that need to be considered and determined to ensure that the wing is optimally designed. In order to ensure that wing satisfies its purpose in an optimal way the following parameters must be incorporated. The primary objective of the wing is to achieve lift. The amount of lift determines the load that the plane can carry. However, lift is fundamentally interlinked with factors like wing shape, cruising speed and air density. The idea of wing design is to test the smallest number of different wing shapes and to obtain a balance between lift and weight while minimising the drag of the aeroplane and maintaining structural integrity of the wing. (Drag is the force that resists the aeroplane's forward motion.) With small drag, the size of the engines can also be reduced enabling weight reduction so diminishing the required lift and thrust. Smaller engines consume less fuel, thus the fuel tank size can also be reduced, so reducing the required wing volume and thus changing its weight. One can vary all these parameters to obtain an optimal aeroplane design.

1.2 Uncertainties

In the process of developing the model for some given problem, decisions about which aspects of it to include and which to exclude must be made. There are many other factors due to which some other aspects of the system might also be unknown, for example, due to conflicting information, human errors, etc. Therefore, the uncertainty level of the model is growing.

Uncertainties in engineering models are normally assigned to ambiguity in describing and defining the parameters of the system and their relations. The sources of such components are normally classified as cognitive and noncognitive, [1]. **The cognitive types of uncertainty**, which are also known as subjective, or epistemic uncertainties, represent a mind-based reflection of the reality, which is subjective and imprecise. Cognitive sources can be, for example, environmental consequences of projects, the current state of existing structures, non comprehensive understanding of the complex processes, skills and experience of construction workers and engineers, and other human factors, as well as the defining relations between the parameters, especially in the case of complex systems. In other words, these are the uncertainties based on lack of knowledge. Probability and statistics do not properly model the uncertainties arising from such sources. By increasing knowledge in a subject one can reduce or even eliminate completely from the problem this type of uncertainty. **Noncognitive sources of uncertainty** are generally those which can be dealt with by using the theories of probability and statistics. These can be: uncertainties due to limited information for estimating their characteristics - statistical uncertainties; uncertainties due to idealization and/or simplification of assumptions in modelling - model uncertainties; and physical unpredictable and/or random behaviour. These uncertainties are also called statistical, or aleatory. The noncognitive uncertainties are the ones considered in this work.

1.3 Errors

The process of engineering modelling inevitably leads to the creation of a number of errors as one moves from one step to the next.

1. Firstly, no mathematical model can perfectly describe the real world. Thus the transition from the real world into mathematical model causes some errors known as *the modelling errors*.

2. The next step of modelling is to convert, typically, the continuous mathematical model into something that can be interpreted by a computer, in other words adapting the mathematical model for numerical evaluation. This usually involves discretisation. An example of this can be seen when structures are simulated in finite element packages as meshing takes place. The solution is normally obtained a number of times with smaller and smaller mesh sizes such that *the discretisation error* is reduced and convergence is observed.

3. Another possible source of error is in the computer realisation. This error is called *the round-off error*. It results from the numeric limitations of the computer, i.e. rounding errors, machine precision, etc.

4. The final source of error is human implementation error. This ranges from the incorrect choice of algorithm, poor execution of the mathematical model or simply the use of bad practices in the coding of the computer program.

In this thesis we only consider uncertainties due to statistical variation of model parameters, and work under the assumptions that

- there are no cognitive inputs,
- the modelling errors are negligible,
- there are no discretisation errors,

- the computer code is implemented perfectly.

For modelling uncertain systems, uncertain parameters may be considered as random variables that acquire some appointed distribution of values instead of a single value. These can be distributions such as those provided by statistical books (e.g. normal distribution, etc.), or even unknown distributions that can only be approximated. Furthermore, there are a lot of probabilistic methods, such as stochastic finite-element methods, reliability methods, probabilistic engineering mechanics, and others, which have been developed and are used for these cases.

It must also be appreciated that a crucial role in engineering analyses is played by the computer code realisation of the engineering model. Usually, this part is computationally expensive. Despite a rapid growth of computing speed and power, the complexity of codes seems to increase as well. While the idea of this work deals with supplying a design variable's vector - input - X and receiving the response vector - output - Y, the present solution is acceptable for small problem size and becomes inappropriate as size increases.

1.4 Literature review

For many applications it is sufficient to represent a model outputs' uncertainties in terms of their expectation and variance. Such an approach is sufficient if we require estimates of expected cost and performance. It is not appropriate however when considering rare events such as probability of failure.

In mathematical terms, let us represent a model as a function $y = g(x)$, where $g : \mathbb{R}^n \longrightarrow \mathbb{R}^m$. In deterministic problems, y is evaluated for given $x \in \mathbb{R}^n$. In stochastic problems, x is taken from a random distribution F_x, and one must calculate the expectation μ, variance σ_x^2 and perhaps higher order moments of F_y, the probability density function of y. There are several possible ways to perform this tasks.

1.4.1 Simulation methods

The most straight forward approach for computing moments is to use a Monte Carlo (MC) simulation, e.g. [3, 4], etc. It is based on the use of random numbers and statistics for investigating a problem and the basic algorithm proceeds as follows.

- A random number generator provides a set of values $\mathbf{x} = \{x_1, \ldots, x_N\}$ from the given distribution $F_{\mathbf{x}}$.
- The model $y_i = g(x_i)$ is evaluated for all $x_i \in \mathbf{x}$, $i = \{1, \ldots, N\}$.
- The computation of the mean and the variance of F_y based on all y_i, $i = 1, \ldots, N$ is performed. The mean and variance of the distribution F_y is then approximated as the mean and variance of the sample y_i, $i = 1, \ldots, N$.

One of the problems one can face while using Monte Carlo simulation techniques is that the randomly generated points may not be well spread on the relatively large design, or probability, space. To maximize the accuracy of the resulting measures the number of simulation evaluations N should be increased. The convergence rate of Monte Carlo method is only $O(N^{-1/2})$. By raising the number of evaluations one will improve the accuracy but, obviously, increase the computation time.

The Monte Carlo method is one of the most commonly used ones, if not as a main technique, then as a technique for comparison.

1.4.2 Integration methods

Analytically, statistical moments are defined by an integral. For example, the expectation μ_g of the function $g(\mathbf{x})$ can be written as

$$\mu_g = \int_{-\infty}^{+\infty} g(t) f_{\mathbf{x}}(t) dt, \tag{1.1}$$

where $f_{\mathbf{x}}$ is a probability density function (pdf). More details on this matter are given in Chapter 2. However, on inspection of (1.1) one's first instinct might be to apply a

numerical quadrature rule to approximate the integral.

For example, in the paper by Brookes and Wise, [5], the authors use the trapezoidal rule - one of the simplest numerical integration methods with error $O(N^{-3})$, where N is the number of subintervals. Thus, to improve the accuracy of the approximation it is necessary to increase the number of subintervals. The convergence order can be improved by choosing a different quadrature scheme. For integrating (1.1) Guassian quadrature [6, 7], or its variant Gauss-Hermite quadrature [8], are known to perform best.

Gaussian quadrature is defined by

$$I = \int_{\mathscr{D}} g(t)W(t)dt \approx \sum_{i=1}^{N} w_i g(x_i), \tag{1.2}$$

where $\mathscr{D} \subset \mathbb{R}$, and W is a so-called weight function such that $\int_{\mathscr{D}} W(t)dt = 1$. When a vector of random variables $\mathbf{x} = (x_1, \ldots, x_n)$, with every variable x_i defined over $\mathscr{D}_i \subset \mathbb{R}$, $i = 1, \ldots, n$, and distributed with pdf $f_{x_i}(x_i)$, and further assuming the statistical independence of elements of $\mathbf{x}$, the first statistical moment, the expectation, becomes

$$\begin{aligned} \mu_g &= M_1 = \int_{\mathscr{D}_1 \times \ldots \mathscr{D}_n} g(t_1, \ldots, t_n) f_{x_1}(t_1) \ldots f_{x_n}(t_n) dt_1 \ldots dt_n \\ &\approx \sum_{i_1=1}^{N_1} \cdots \sum_{i_n=1}^{N_n} w_{i_1} \ldots w_{i_n} g(x_{i_1}, \ldots, x_{i_n}). \end{aligned} \tag{1.3}$$

In the same way higher order moments can be estimated:

$$\begin{aligned} M_k &= \int_{\mathscr{D}_1 \times \ldots \mathscr{D}_n} \left[g(t_1, \ldots, t_n)\right]^k f_{x_1}(t_1) \ldots f_{x_n}(t_n) dt_1 \ldots dt_n \\ &\approx \sum_{i_1=1}^{N_1} \cdots \sum_{i_n=1}^{N_n} w_{i_1} \ldots w_{i_n} \left[g(x_{i_1}, \ldots, x_{i_n})\right]^k, \end{aligned} \tag{1.4}$$

where k is the order of the considered statistical moment M. The requirement for independent inputs can be relaxed, but then the component pdfs f_{x_i}, $i = 1, \ldots n$, must be replaced by the joint pdf $f_{\mathbf{x}} = f_{\mathbf{x}}(x_1, \ldots, x_n)$. The application of the quadrature

rule to multiple integrals to obtain (1.4) is possible due to Fubini's theorem, [9], that allows one to change the order of the integral from multiple to repeated single-dimensional ones.

The use of quadrature for approximating statistical moments is not only highly dependent on the convergence order of the particular rule, it also can turn out to be very computationally expensive as the values of the function $g(\mathbf{x})$ are computed $N_1 \times \cdots \times N_n$ times. This demand for computer resources can become a major problem as the dimension of the model increase.

There are a number of improved quadrature methods. For example, in the papers by Padulo et al., [7] and [10], the authors suggest using the Sigma-Point technique that relies on reduced quadrature rule. The efficiency of this method is justified by comparison with low order moments method based on first order Taylor series; the Sigma-Point approach produces the same, or higher, order accurate results for the expectation and the variance as the moments method. It also does not require derivative computations, thus it can handle even discontinuous functions as well as functions dependent on discrete variables.

1.4.3 Stochastic differential equations and polynomial chaos

Another possible way to deal with uncertainties is by using stochastic differential equations, [3]. While building a mathematical model of the physical reality one directly incorporates the uncertainties into the associated equations. This approach can be very efficient and inexpensive, but relies heavily on the engineers' skills and knowledge. For a simple first order stochastic pde the Karhunen-Loeve expansion is often used, [11, 12]. Many of the stochastic pde solvers are based on the polynomial chaos expansion [13]. According to Wiener, who first introduced this concept, polynomial chaos is a method that uses polynomials as a basis for representing stochastic processes. Another way to tackle stochastic pde problems is to use the stochastic finite element method [12]. There are many publications on the subject in the recent years.

However, stochastic pde solvers appear underdeveloped, therefore the technique is not yet practical for practicing engineers.

1.4.4 Moments method

A further alternative to these approaches is the moments method. It is relatively easy to perform, provided derivatives can be computed, and may produce results with less consumption of computational time than methods like Monte Carlo. It gives an approximation to the moments of the distribution, expectation and variance in particular. The idea of the moments method is to approximate the distribution of the output function y in terms of its derivatives by using a Taylor series. Moments of the Taylor approximation then yield approximation of the first, the second and possibly higher order statistical moments. Any improvements in the accuracy of this method require computation of higher order derivatives, which becomes possible due to new developments in the field of automatic differentiation.

In this work we consider the moments method in more details. There are only a few papers that consider this subject, although the results of those papers were already summarised in the books by Keane and Nair [3] (p.337-338) and Papoulis and Pillai [14] (p.150), and are widely used for engineering modelling.

The basic approach for the moments method is considered in the conference paper of Ghate and Giles [4], where the authors derived the moments method based on a single-variable Taylor series with no given assumptions for the input distribution. Therefore, the resulting formulas are obtained in general univariate form. Based on the first and second order Taylor expansions there are first and second order moments methods presented, respectively. They also mentioned that the second order Taylor approximation does not guarantee second order of accuracy for the variance approximation. The details of this statement are considered in more detail in Section 4.1.1. As test examples trigonometric functions, $\sin x$ and $\cos x$, were chosen. Further analysis of the results of their experiments is considered as we shall see in Chapter 4.

Although Ghate's work [4] has been reviewed first, that paper was fully grounded on the more general and comprehensive work of Putko et. al. [15]. They first used the moments method for uncertainty propagation in aerospace design.

The authors did not make any restrictions on the dimension of the problem, but assumed random, statistically independent and normally distributed inputs. This allowed them to simplify algebraic computations by neglecting the inputs' covariance matrix and skewness (which are zero for normal distribution) terms. With these assumptions, the authors developed the first and second order moments methods by using multivariate Taylor series of first and second order.

As a demonstration example they apply these methods to estimate the influence of uncertainty in CFD input parameters. The analysis is implemented with the quasi 1D Euler equation and boundary conditions (stagnation enthalpy, inlet entropy, outlet static pressure) describing subsonic flow through a variable area nozzle. They consider two different behaviours of the moments method depending on what parameters were defined to be the random input variables. By setting geometric shape parameters to be statistically independent random input variables they predict the Mach number, compare the results with CFD solutions and Monte Carlo simulation analysis, and find good agreement between the first order moments method and Monte Carlo simulation for the outputs mean and variance. Though if the free-stream Mach number and back pressure are reassigned to be independent random input, the output function is more nonlinear in their neighbourhood. In this case the second order moments method produces better agreement with Monte Carlo simulation.

The main problem all described works were facing is computing higher order derivatives, which are required for higher order moments methods, but which are difficult to obtain without using AD tools.

The most general and advanced case was considered in the conference paper by Bruce Christianson and Maurice Cox [16]. The authors subdivide the task and look at the following cases: linear and nonlinear output functions, correlated/uncorrelated

and single/multiple inputs. They also point out some directions for future research related to convergence of the Taylor series and singularities of the output functions. We consider their analysis and results in more details in Chapter 4.

1.5 Thesis outline

Due to development of automatic differentiation (AD) tools for different programming languages, it has became possible to obtain the higher order derivatives required for computing Taylor coefficients. Thus, in this work we develop a higher order moments method to those considered in the published sources [3, 4, 15, 10], generalise it upto an arbitrary order of the Taylor expansion, implement it not only for first and second statistical moments but also extend it for skewness (third) and kurtosis (fourth) with the possibility of computing even further moments. The computer implementation makes use of AD in Matlab [17].

In Chapter 2 of this thesis we provide the mathematical background required for adequate understanding of the whole work. Subjects related to statistical distributions, Taylor series and the basics of the combinatorial theory are considered in details.

Chapter 3 introduces the fundamentals of automatic differentiation theory and its implementation. We also demonstrate its use by applying it to numerical integration schemes and overloading the quadrature routine in Matlab `quad`. The results are tested and compared to those in published work.

Following from this, we familiarise the reader with the moments method by repeating the results obtained in both [4] and [15], confirmed by [7] and [10]. The ways of deriving the moments method are subdivided into two approaches, and the differences between are presented. Particular attention is paid to the convergence of the method and implementing the partitioning approach advised by Christianson and Cox [16] for the cases when the radius of convergence for Taylor series is finite. Va-

rious examples are given to demonstrate the validity of the moments method as well as the partitioning approach itself. All of this work can be found in Chapter 4.

The implementation of the moments method in Matlab is closely considered in Chapter 5 with a detailed description of every step of the developed algorithm.

Chapter 6 deals with changes in control-flow. In other words, we consider what happens if the model function contains branches, and thus the derivatives computation varies depending on the values of the input parameters and their changes. The results of the analysis are implemented in the branch detection package for Matlab.

The final chapter, Chapter 7, summarises all the experimentation carried out throughout this body of research. In addition, potential ideas for further work are also proposed.

Because the details of all the computations performed are extensive, it was decided to gather these together in the Appendices section.

Chapter 2

Statistical and mathematical background

In order to prepare the reader for the next chapters, here we present various definitions of probability and the statistical background. The references used as a main source of information for this subject were the books of Ayyub and McCuen [1], Drew and Wampold [18], Barr and Zehna [19], Hines and Montgomery [20], Ingram [21].

2.1 Distributions and PDFs

A **random variable** is a function which maps from a possibility space into a set of numbers. Random variables can be classified into two types, discrete and continuous. A continuous random variable is a random variable that takes on an infinite number of values; furthermore, between any two specified values the random variable assumes a value. A discrete random variable is a random variable that takes numerical values from a finite or countably infinite range. Depending on the type of the random variable we are dealing with, the function describing the behaviour of the distribution is called the probability density function (pdf) or probability mass function (pmf). Since we are not considering here discrete cases, from now on we refer only to continuous random variables and their associated distributions, and pdfs.

Probability density functions specify how the values x of a random variable X are distributed. Such functions are said to give the **distribution** of X. The pdf, $f(X)$, for

a random variable is a function that assigns a probability density to each value of the random variable. Specifically, the probability that the random variable X lies within the interval $[x_1, x_2]$ is given by

$$P\left(x_1 \leq X \leq x_2\right) = \int_{x_1}^{x_2} f(t)dt. \tag{2.1}$$

Any pdf satisfies the following two conditions:

$$f(X) \geq 0 \quad \text{for all values of } X, \text{ and} \tag{2.2}$$

$$\int_{-\infty}^{\infty} f(t)dt = 1. \tag{2.3}$$

In Table 2.1 we introduce pdfs for several distributions commonly used in science and engineering.

Distribution	Density function
Laplace Distribution	$\frac{1}{2b}\exp\left(-\frac{\lvert X-\mu_X\rvert}{b}\right)$, where $b>0$ - scale parameter
Log Normal Distribution	$\frac{\exp\left(-\frac{(\ln X-\mu_X)^2}{2\sigma_X^2}\right)}{X\sigma_X\sqrt{2\pi}}$
Normal Distribution	$\frac{1}{\sigma_X\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{X-\mu_X}{\sigma_X}\right)^2\right)$
Student's t-Distribution	$\frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\nu\pi}\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{X^2}{\nu}\right)^{-\frac{\nu+1}{2}}$, where $\nu = n-1$, n - number of independent random variables, and $\Gamma(z) = \int_0^{+\infty} t^{z-1}e^{-t}dt$ is the Gamma function

Table 2.1: Probability density functions, where μ_X and σ_X are mean and standard deviation, respectively.

2.1.1 Expectation

The **mathematical expectation**, $E(X)$, or μ_X, is one of the measures of location of a distribution - it defines its centre. Or, in terms of random variables, it defines measure of average. The intuitive way to describe such an average mathematically is known as the arithmetic expectation and written as

$$E(X) = \frac{1}{n}\sum_{i=1}^{n} x_i,$$

where $X = \{x_1, \ldots, x_n\}$ is a discrete random variable. For a continuously distributed random variable X with pdf $f(X)$ the expected value $\mu_X = E(X)$ is defined by

$$E(X) = \int_{-\infty}^{+\infty} t\, f(t) dt. \tag{2.4}$$

Similarly, the expected value of an arbitrary function of X, $G(X)$, with respect to the probability density function of X, f(X), is given by

$$E(G(X)) = \int_{-\infty}^{+\infty} G(t)\, f(t) dt. \tag{2.5}$$

Some useful properties of the expectation are all readily obtained from (2.3) and (2.5):

$$E(a) = a, \tag{2.6}$$

where a is a constant. It means that if a random variable assumes only one value a with probability 1, then the expectation is also only a.

$$E(aX) = aE(X). \tag{2.7}$$

$$E(X + a) = E(X) + a. \tag{2.8}$$

$$E(X + Y) = E(X) + E(Y). \tag{2.9}$$

We will use these properties when deriving formulas for the moments method in further sections.

2.1.2 Variance

The **variance** σ_X^2 is a measure of dispersion of a distribution and is defined as follows:

$$\sigma_X^2 = E\left[(X-\mu_X)^2\right]. \tag{2.10}$$

Using (2.5) we can see that

$$\begin{aligned}\sigma_X^2 &= \int_{-\infty}^{+\infty}(t-\mu_X)^2 f(t)dt = \int_{-\infty}^{+\infty}(t^2-2\mu_X t+\mu_X^2)f(t)dt \\ &= \int_{-\infty}^{+\infty}t^2 f(t)dt - 2\mu_X\int_{-\infty}^{+\infty}tf(t)dt + \mu_X^2\int_{-\infty}^{+\infty}f(t)dt \\ &= E(X^2)-2\mu_X E(X)+\mu_X^2 = E(X^2)-E(X)^2. \end{aligned}\tag{2.11}$$

The positive square root of the variance is called the **standard deviation** of X and is denoted by

$$\sigma_X = \sqrt{E\left[(X-\mu_X)^2\right]}. \tag{2.12}$$

2.1.3 Moments of the distribution

For any positive integer p, X**'s** p**-th central moment** is defined as

$$E\left[(X-\mu_X)^p\right] = \int_{-\infty}^{+\infty}(t-\mu_X)^p f(t)dt. \tag{2.13}$$

We can see from the definition of mean that $E\left[(X-\mu_X)^1\right]=0$ and that the variance $\sigma_X^2 = E\left[(X-\mu_X)^2\right]$ is given by the second central moment ($p=2$).

Skewness

The third central moment $E\left[(X-\mu_X)^3\right]$ measures the symmetry of the distribution of X about its mean. In some references $E\left[(X-\mu_X)^3\right]$ is termed the skewness. But we will follow the notation of Ingram [21] and define the skewness $S(x)$ as a third normalized moment (later called simply the third moment):

$$S(X) = \frac{E\left[(X-\mu_X)^3\right]}{\sigma_X^3}. \tag{2.14}$$

If the distribution is symmetric about μ_X then necessarily $S(X)$ is zero. When $S(X)$ is negative the distribution is **skewed to the left**; positive $S(X)$ indicates X's distribution is **skewed to the right**. Skewness on the left or right implies a long tail on the left or right, respectively. Table 2.2 lists the skewness for a number of commonly used distributions.

Distribution	Skewness $S(X)$
Laplace Distribution	0
Log Normal Distribution	$\sqrt{e^{\sigma^2}-1}\left(2+e^{\sigma^2}\right)$
Normal Distribution	0
Student's t-Distribution	0

Table 2.2: Skewness for some distributions.

Kurtosis

The **kurtosis** $K(X)$ can be thought of as the degree of "peakedness" of the probability distribution of a real-valued random variable X. In a similar way to the skewness case, we define kurtosis as a normalized form of the fourth central moment

$$K(X) = \frac{E\left[(X-\mu_X)^4\right]}{\sigma_X^4}. \tag{2.15}$$

To determine the "peakedness" of a distribution using $K(x)$ the normal distribution is used as a standard. For a normal distribution, $K(x) = 3$, therefore an **excess kurtosis** γ is defined

$$\gamma = \frac{E\left[(x-\mu_x)^4\right]}{\sigma^4} - 3 = K(x) - 3.$$

The reader should note that in some references the term "kurtosis" is used for the excess kurtosis γ or even for the fourth central moment, $E\left[(X-\mu_X)^4\right]$.

Distributions with $\gamma = 0$ are called **mesokurtic**. The normal distribution is mesokurtic, regardless of the value of its parameters. A few other well-known distributions can be mesokurtic, depending on parameter values. For example, the binomial distribution is mesokurtic for $p = \frac{1}{2} \pm \sqrt{\frac{1}{12}}$. If $\gamma < 0$, then the distribution is "less-peaked" than the normal distribution, and is called **platykurtic**. An example of a platykurtic distribution is the Bernoulli distribution with $p = \frac{1}{2}$. A distribution that is "more peaked" than the normal distribution, $\gamma > 0$, is termed **leptokurtic**. Examples of leptokurtic distributions include the Laplace distribution and the logistic distribution.

The kurtosis for some commonly used distributions is listed in Table 2.3.

Distribution	Kurtosis $K(X)$
Laplace Distribution	6
Log Normal Distribution	$e^{4\sigma^2} + 2e^{3\sigma^2} + 3e^{2\sigma^2} - 3$
Normal Distribution	3
Student's t-Distribution	$\frac{3(n-2)}{n-4}$

Table 2.3: Kurtosis for some distributions.

Note that using the binomial expansion,

$$E\left[(X-\mu_X)^p\right] = E\left[\sum_{i=0}^{p}\binom{p}{i}X^p(-\mu_X)^{p-i}\right] = \sum_{i=0}^{p}\binom{p}{i}E(X^p)(-\mu_X)^{p-i}, \qquad (2.16)$$

where $\binom{p}{i} = \frac{p!}{i!(p-i)!}$. Therefore, similar to the variance computation (2.11), we may rewrite (2.14) and (2.15) in the alternative forms

$$S(X) = \frac{E(X^3) - 3\mu_X E(X^2) + 2\mu_X^3}{\sigma^3}, \tag{2.17}$$

$$K(X) = \frac{E(X^4) - 4\mu_X E(X^3) + 6\mu_X^2 E(X^2) - 3\mu_X^4}{\sigma^4}. \tag{2.18}$$

In many natural processes, random variation conforms to a particular probability distribution known as the normal distribution, which is the most commonly observed probability distribution and is described more fully in the next section.

2.1.4 Normal distribution

The general formula for the probability density function of the normal distribution is

$$f(X) = \frac{\exp\left(\frac{-(X-\mu)^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}}, \tag{2.19}$$

where μ is the location parameter and σ is the scale parameter. The case where $\mu = 0$ and $\sigma = 1$ is called the standard normal distribution. The pdf for the standard normal distribution is

$$f(X) = \frac{\exp\left(-\frac{X^2}{2}\right)}{\sqrt{2\pi}}. \tag{2.20}$$

The shape of the normal distribution resembles a bell (see Fig.1), so it is often referred to as the “bell curve”. The normal distribution is symmetric; unimodal (of one peak); and satisfies the condition

$$\int_{-\infty}^{+\infty} f(t)dt = 1 \tag{2.21}$$

and it extends from $-\infty$ to $+\infty$.

Any normal distribution can be completely described by its mean and variance. With both parameters known one can get information about every point in the data set.

2.1.5 Two or more random variables

It is common to deal with two or more random variables at the same time when solving engineering problems. Let X and Y both be continuous random variables. Now the distribution of X and Y determine their separate statistics. In [14] it is called **marginal statistics**. However, to compute their **bivariate**, or **joint**, **statistics** we need to consider the relations between these two random variables.

In other words, to deal with X and Y separately we use the pdfs $f_X(X)$ and $f_Y(Y)$, but to take them both into account in the same problem, we require the joint pdf $f(X,Y)$. If the random variables X and Y are independent, then

$$f(X,Y) = f_X(X)f_Y(Y). \tag{2.22}$$

Thus the probability that the random variables X and Y are within the intervals $[x_1, x_2]$ and $[y_1, y_2]$ respectively is

$$P\left(x_1 \leq X \leq x_2; y_1 \leq Y \leq y_2\right) = \int_{x_1}^{x_2}\int_{y_1}^{y_2} f(t_X, t_Y)dt_X dt_Y. \tag{2.23}$$

When integrating the joint pdf $f(X,Y)$ over $\mathbb{R}^2$, the probability always equals 1, i.e.

$$P\left(-\infty \leq X \leq +\infty; -\infty \leq Y \leq +\infty\right) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(t_X, t_Y)dt_X dt_Y = 1. \tag{2.24}$$

If $Z = g(X,Y)$ is another random variable, the expected value of it is given by

$$\mu_Z = E(Z) = E\big(g(X,Y)\big) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g(t_X, t_Y)f(t_X, t_Y)dt_X dt_Y. \tag{2.25}$$

The second moment of two random variables is called the **covariance** and is defined as

$$C_{XY} = E\big((X - \mu_X)(Y - \mu_Y)\big). \tag{2.26}$$

Analogously to (2.11), it can be rewritten as

$$C_{XY} = E(XY) - E(X)E(Y). \tag{2.27}$$

When $X = Y$ the covariance becomes $C_{XX} = E(X^2) - E(X)^2 = \sigma_X^2$.

The covariance matrix contains variances on the diagonal and covariances below and above the diagonal. Since $C_{XY} = C_{YX}$, the covariance matrix is symmetric:

$$C = \begin{pmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{pmatrix} = \begin{pmatrix} \sigma_X^2 & C_{XY} \\ C_{XY} & \sigma_Y^2 \end{pmatrix}. \tag{2.28}$$

The **correlation coefficient** ρ_{XY} is defined as the ratio

$$\rho_{XY} = \frac{C_{XY}}{\sigma_X \sigma_Y}. \tag{2.29}$$

It lies in the range between -1 and 1 and is also known as a *normalised covariance* with respect to the standard deviations σ_X, σ_Y.

Two random variables X and Y are called **independent** if their covariance C_{XY} is equal to 0:

$$C = \begin{pmatrix} \sigma_X^2 & 0 \\ 0 & \sigma_Y^2 \end{pmatrix}. \tag{2.30}$$

Therefore the correlation coefficient ρ_{XY} is also 0. Independent random variables are also often called **uncorrelated**, implying $\rho_{XY} = 0$. However, in practice the absence of correlation does not necessarily indicate the absence of covariance. Thus in this work we are careful in distinguishing the definitions of uncorrelated and independent

variables.

For independent variables $E(XY) = E(X)E(Y) = \mu_X \mu_Y$ and the covariance matrix includes only diagonal elements.

Extending these definitions for multiple random variables $\mathbf{X} = (X_1, \ldots, X_n)$, the expectation is defined as

$$E(Z) = E\big(g(\mathbf{X})\big) = \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} g(t_1, \ldots, t_n) f(t_1, \ldots, t_n) dt_1 \ldots dt_n. \tag{2.31}$$

Then the covariance matrix is

$$C = \begin{pmatrix} \sigma^2_{X_1} & C_{X_1X_2} & \ldots & C_{X_1X_n} \\ C_{X_2X_1} & \sigma^2_{X_2} & \ldots & C_{X_2X_n} \\ \vdots & \vdots & \ddots & \vdots \\ C_{X_nX_1} & C_{X_nX_2} & \ldots & \sigma^2_{X_n} \end{pmatrix}. \tag{2.32}$$

Higher order moments are defined in a similar manner. For example, the third order moment is an $(n \times n \times n)$ array M^3 with the entries given as

$$M^3_{X_iX_jX_k} = E\big((X_i - \mu_{X_i})(X_j - \mu_{X_j})(X_k - \mu_{X_k})\big), \tag{2.33}$$

where $i = 1, \ldots, n$, $j = 1, \ldots, n$, $k = 1, \ldots, n$. When $i = j = k$, the corresponding third order moment becomes skewness as defined in (2.14):

$$M^3_{X_i} = E\big((X_i - \mu_{X_i})^3\big).$$

2.2 Taylor series

In this section we introduce the main definitions regarding power series including Taylor series. This information is essential for understanding the idea of the moments method, which is based on using the Taylor approximation, and consequently the

convergence issues that arise together with the definition of the series. The main source for this material are the books by Adams [22] and Spiegel [23].

A series of the form

$$\sum_{i=0}^{\infty} a_i (x-c)^i = a_0 + a_1(x-c) + a_2(x-c)^2 + \dots \tag{2.34}$$

is called a **power series** about the point c; a_0, a_1, a_2, ... are the coefficients of the power series.

Depending on the value of x, the power series (2.34) may or may not converge. For values of x, for which the series converges, the sum (2.34) defines a function of x. The point c is called the **centre of convergence** of power series, meaning the series (2.34) definitely converges at that point.

For any power series (2.34), the series converges either

- only at point c, or
- everywhere on the space of real numbers, or
- everywhere if $|x-c| < r$, where r is a positive real number, and diverges if $|x-c| > r$. In this case the power series may or may not converge at the endpoints $c-r$ and $c+r$. The number r is called the **radius of convergence**.

Therefore, the **interval of convergence** of (2.34) has one of the following forms:

- the isolated point c, $r = 0$,
- the entire space of real numbers $\mathbb{R}$, $r = \infty$,
- the finite interval: $[c-r, c+r]$, $[c-r, c+r)$, $(c-r, c+r]$, or $(c-r, c+r)$.

When the power series has the radius of convergence $r > 0$, then the sum of the series defines a function g on the interval of convergence $(c-r, c+r)$, and the coefficients of (2.34) $a_i = \dfrac{g^{(i)}(c)}{i!}$ for $i = 0, 1, 2, \dots$.

If $g(x)$ is continuously differentiable at point $x = c$, then the series

$$g(x) = \sum_{i=0}^{\infty} \frac{1}{i!} g^{(i)}(c)(x-c)^i = g(c) + g'(c)(x-c) + \frac{1}{2!} g''(c)(x-c)^2 + \dots \quad (2.35)$$

is known as the **Taylor series** of function g about $x = c$.

For the Taylor series (2.35) the radius of convergence is the distance from the point c to the nearest singularity of the function g. A point at which the function g is undefined is a called **singular point**, or **singularity**.

If the function has the form

$$g(x) = a_{p_1}(x-c)^{p_1} + \dots + a_1(x-c) + a_0 + a_{-1}(x-c)^{-1} + \dots + a_{-p_2}(x-c)^{-p_2}, \quad (2.36)$$

then $x = c$ is called a **pole** of order p_2.

When the $(p+1)^{st}$ derivative of the function g exists, and the Taylor polynomial of degree p is constructed about $x = c$, then the **remainder** for Taylor expansion can be written as

$$R_{p+1} = \int_c^x \frac{1}{p!}(x-t)^p g^{p+1}(t)\,dt. \quad (2.37)$$

A Taylor series allows close approximations to an arbitrary differentiable function on an interval by using partial sums of the series - polynomials. However, power series are not well suited for periodic functions, since polynomials are not periodic, [22]. The function g is called **periodic** with period T if $g(t+T) = g(t)$ for all $t \in \mathbb{R}$. This may influence the results for moments method based on Taylor approximation used with periodic functions. Thus, periodic functions are worth particular attention when testing methods considered in this work.

2.3 Combinatorics

To deal with the sets of indices used in Chapter 5 the basics of combinatorics are essential. In this section the main definitions of combinations and permutations are

given. We also stress the attention on the difference between them, which allows us to determine the coefficients in Chapter 5 correctly.

2.3.1 Permutations

Definition 2.3.1. A ***permutation*** is an ordered arrangement of things.
There are two types of permutations:

- permutations with repetitions;
- permutations with no repetitions.

Example 2.3.1. What are the ways of choosing all ordered sets of 2 elements from the set of 3 elements $A = \{1, 2, 3\}$? If the repetitions are allowed, meaning every element of A can be chosen more than once, the permutations *with* repetitions are

$$\begin{array}{ccc} (1,1) & (1,2) & (1,3) \\ (2,1) & (2,2) & (2,3) \\ (3,1) & (3,2) & (3,3) \end{array} \tag{2.38}$$

On the other hand, when the repetitions are not used, the permutations *without* repetitions are written below.

$$\begin{array}{ccc} & (1,2) & (1,3) \\ (2,1) & & (2,3) \\ (3,1) & (3,2) & \end{array} \tag{2.39}$$

Let the set $A = \{a_1, \ldots, a_n\}$ contain n elements. An *r-permutation with repetition* of a set A is the number of ways to choose r elements from A with repetition allowed, [24]. Analogously, an *r-permutation without repetition* of a set A is the number of ways to choose r elements from A when repetitions are not allowed.

To calculate the permutations with repetitions, the number of choices for every i^{th} position, $i = 1, \ldots, r$, is n, thus

$$\underbrace{n \times n \times \cdots \times n}_{r\text{-times}} = n^r. \tag{2.40}$$

For permutations without repetitions however the number of choices for the i^{th} position decreases by $i - 1$,

$$n \times (n-1) \times (n-2) \times \cdots \times (n-r+1) = \frac{n!}{(n-r)!}. \tag{2.41}$$

In the Example 2.3.1, when $n = 3$, $r = 2$, the number of permutations with repetitions is $3^2 = 9$, and the number of permutations with no repetitions is $\frac{3!}{(3-2)!} = 6$.

2.3.2 Combinations

Definition 2.3.2. A ***combination*** is an unordered permutation. In other words, r-combination of a set A is a subset of size r, [24]. Similarly to permutations, there are combinations with and without repetitions.

Example 2.3.2. We can alter all permutations with repetitions in example 2.3.1 so that the order does not matter,

$$\begin{array}{ccc} (1,1) & (1,2) & (1,3) \\ & (2,2) & (2,3) \\ & & (3,3) \end{array} \tag{2.42}$$

and obtain all combinations *with* repetitions. But when repetitions are not allowed, the combinations *without* repetitions are

$$\begin{array}{cc} (1,2) & (1,3) \\ & (2,3) \end{array} \tag{2.43}$$

In this case the permutations have three times as many possibilities.

To compute the number of all combinations without repetitions all permutations without repetition are required to be reduced by the number of ways the objects could be ordered:

$$\left(\begin{array}{c} n \\ r \end{array} \right) = \frac{n!}{(n-r)!} \times \frac{1}{r!} = \frac{n!}{(n-r)!\, r!}. \tag{2.44}$$

In the example above we found three ways to choose two elements without repetitions from the set $A = \{1, 2, 3\}$. Using (2.44), $\binom{n}{r} = \binom{3}{2} = \frac{3!}{2!\ 1!} = 3$.

To explain the formula for computing the number of combinations with repetitions the example with ice cream scoops is often used, [24].

Example 2.3.3. How many different triple-scoop ice cream cones are possible giving that there are n flavours available? Several scoops of the same flavour are permitted, and the cones with reordered scoops are considered to be the same.

$$\underbrace{|\, a_1 \,|\, a_2 \,|\, a_3 \,|\, \ldots \,|\, a_n \,|}_{n \text{ ice cream containers}}$$

We put one star in the i^{th} container every time that the i^{th} flavour appears in the cone. For example,

$$|\ \underbrace{\star}_{a_1}\ |\ \underbrace{\star\,\star}_{a_2,\, a_2}\ |\ \ |\ \ldots\ |\ \ |$$

corresponds to a cone with one scoop of a_1 and two scoops of a_2. This way to order the desired flavours we skip containers with the ice cream we don't want and scoop from those we like, aiming for three scoops in total. Moving from 1^{st} container to n^{th} requires $(n-1)$ steps, scooping corresponds to 3 steps. Thus the number of all possible ice cream cones with three scoops is the number of all possible variations of skips-and-scoops, when there is $n-1$ skips and 3 scoops. That is to say that the number of such variations is equal to the number of ways to choose r distinct positions for the scoops in a string of $n+r-1$ skips and scoops, which is the number of r-combinations without repetitions of a set with $n+r-1$ elements.

Therefore, the number of all r-combinations with repetitions is

$$\left(\begin{array}{c} n+r-1 \\ r \end{array} \right) = \frac{(n+r-1)!}{\big((n+r-1)-r\big)!\; r!} = \frac{(r+n-1)!}{(n-1)!\; r!}. \tag{2.45}$$

Now when we considered all the necessary statistical and mathematical background required throughout the work, we can get to introducing the automatic differentiation techniques.

Chapter 3

Introduction to automatic differentiation

Automatic differentiation (AD) is the scientific field concerned with calculating derivatives of a function defined by a computer program efficiently and accurately with minimal programmer intervention. The solution of many mathematical problems requires knowledge of the gradient, Jacobian or Hessian matrices of given functions. AD techniques provide the automatic computation of derivatives of any such general function, based on use of the chain-rule for evaluating derivatives with respect to the input function's arguments. AD differentiates computer coded functions of any complexity, assuming that the composition of the elementary functions forming the main function is finite.

There are many sources available describing automatic differentiation tools in full details. For the introduction in this thesis the book of Griewank [25], and articles by Verma, [26], and Forth, [17], in particular, were used.

There are two basic modes of computing derivatives with AD: forward and reverse. They are both considered on the following example.

Consider the function, which computes $g(x) = x^2 + ax + b$, where a and b are some constants. An AD tool internally breaks this function into a form known as an ***evaluation trace*** (or ***code list***) [25] such as

`function` $y = g(x)$

$v_1 = x * x$;

$v_2 = a * x$;

$v_3 = v_1 + v_2$;

$y = v_3 + b$;

`end`

where v_1, v_2 and v_3 are called ***intermediate variables***.

The forward mode of AD propagates derivatives throughout the computation using the chain rule in step with each intermediate variable computation.

`function` $(y, \nabla y) = \nabla g(x, \nabla x)$

$v_1 = x * x$;

$\nabla v_1 = 2 * x * \nabla x$;

$v_2 = a * x$;

$\nabla v_2 = a * \nabla x$;

$v_3 = v_1 + v_2$;

$\nabla v_3 = \nabla v_1 + \nabla v_2$;

$y = v_3 + b$;

$\nabla y = \nabla v_3$;

`end`

where ∇x, ∇v_i, and ∇y are ***directional derivatives***.

We now can systematise the derivative calculation into the matrix form

$$\begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 2x & -1 & 0 & 0 & 0 \\ a & 0 & -1 & 0 & 0 \\ 0 & 1 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} * \begin{bmatrix} \nabla x \\ \nabla v_1 \\ \nabla v_2 \\ \nabla v_3 \\ \nabla y \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

or, in other words, $J \nabla X = R$, where the matrix J is called ***the extended Jacobian***.

Setting $x = 2$, $a = 3$, $b = 4$, and $\nabla x = \frac{dx}{dx} = 1$, we get results

$v_1 = x * x = 2 * 2 = 4$;

$\nabla v_1 = 2 * x * \nabla x = 2 * 2 * 1 = 4;$

$v_2 = a * x = 3 * 2 = 6;$

$\nabla v_2 = a * \nabla x = 3 * 1 = 3;$

$v_3 = v_1 + v_2 = 10;$

$\nabla v_3 = \nabla v_1 + \nabla v_2 = 7;$

$y = v_3 + b = 10 + 4 = 14;$

$\nabla y = \nabla v_3 = 7.$

The reverse mode of AD calculates the derivatives backwards through the computation. Therefore it is more complicated and requires a double run through the computer code, once forward, once in reverse. The entire information from the first run, when the function values are computed, needs to be stored for the derivative computation on the second run. Hence, it causes the memory cost to increase. In terms of matrices, reverse computation of derivatives for the same example is determined by solving the linear system $J^T A = P$, where $P^T = [0\ 0\ 0\ 0\ -1]$, and A is a vector of adjoint variables $\bar{y}$, $\bar{v_i}$, $\bar{x}$ that are defined as

$$\begin{aligned} \bar{y} &= \frac{\partial g}{\partial y}, \\ \bar{v_i} &\equiv \frac{\partial g}{\partial v_i}, \\ \bar{x} &= \frac{\partial g}{\partial x}. \end{aligned}$$

Hence, the function calculating derivatives by using the reverse mode of AD is

`function` $(\bar{x}) = \bar{g}(x, \bar{y})$

$v_1 = x * x;$

$v_2 = a * x;$

$v_3 = v_1 + v_2;$

$y = v_3 + b;$

% Reverse mode for derivatives computation

$\bar{v_3} = \bar{y}$

$\bar{v_2} = \bar{v_3}$

$\bar{v_1} = \bar{v_3}$

$\bar{x} = 2x\bar{v_1} + a\bar{v_2}$

`end`

where $\bar{y} = 1$.

The main reasons for using AD tools are to calculate derivatives faster and more accurately than, for example, the finite differences approximation. The improvements in AD accuracy are normally compared to finite differences methods, which incur truncation errors, while AD computes the derivatives up to the machine precision.

3.1 Automatic differentiation of quadrature

As an example of automatic differentiation we investigate AD for quadrature algorithms. We consider the automatic differentiation of prototypical methods in order to give conditions for their convergence and expected errors and to make the implementation of quadrature in AD packages efficient. The background information on numerical integration can be found, for example, in [27] or [28].

The integral

$$I = \int_a^b g(x)dx \tag{3.1}$$

has the well-known derivatives

$$\frac{\partial I}{\partial a} = -g(a), \tag{3.2}$$

$$\frac{\partial I}{\partial b} = g(b). \tag{3.3}$$

Such derivatives may be used directly in an AD package for a language such as Matlab which has its own quadrature routines: `quad` and `quad4`. We will show that the direct, automatic differentiation of an arbitrary quadrature scheme leads to the same

results (3.2), (3.3) up to the order of accuracy of the quadrature.

The simplest quadrature is the rectangle rule.

3.1.1 Rectangle Rule

The composite left rectangle rule for approximating (3.1) is written

$$I_R = h\sum_{i=0}^{n-1} g(x_i), \tag{3.4}$$

where $h = \frac{b-a}{n}$, and $x_i = a + ih = b - \frac{n-i}{n}(b-a)$. Differentiating (3.4) with respect to a gives

$$\begin{aligned}
\frac{\partial I_R}{\partial a} &= -\frac{1}{n}\sum_{i=0}^{n-1} g(x_i) + h\sum_{i=0}^{n-1} g'(x_i)\frac{n-i}{n} \\
&= -\frac{1}{n}\sum_{i=0}^{n-1}\Big(g(x_i) - hg'(x_i)(n-i)\Big) \\
&= -\frac{1}{n}\sum_{i=0}^{n-1}\Big(g(x_i) - g'(x_i)(b-x_i)\Big) \\
&= -\frac{1}{b-a}h\sum_{i=0}^{n-1} g(x_i) + h\sum_{i=0}^{n-1} g'(x_i)\frac{b-x_i}{b-a}.
\end{aligned} \tag{3.5}$$

Let us define $\Theta_a(x) = \frac{b-x}{b-a}$, this simplifies (3.5) to,

$$\frac{\partial I_R}{\partial a} = -\frac{1}{b-a}h\sum_{i=0}^{n-1} g(x_i) + h\sum_{i=0}^{n-1} g'(x_i)\Theta_a(x_i). \tag{3.6}$$

Now consider the rectangle rule applied to the two integrals

$$\int_a^b g(x)dx = h\sum_{i=0}^{n-1} g(x_i) + E_g^R \tag{3.7}$$

and

$$\int_a^b g'(x)\Theta_a(x)dx = h\sum_{i=0}^{n-1} g'(x_i)\Theta_a(x_i) + E_{g'\Theta}^R, \tag{3.8}$$

where E_g and $E_{g'\Theta}$ are the truncation errors associated with the two approximate integrals. We see that we may rewrite (3.6) as

$$\frac{\partial I_R}{\partial a} = -\frac{1}{b-a}\int_a^b g(x)dx + \frac{1}{b-a}E_g^R + \int_a^b g'(x)\Theta_a(x)dx - E_{g'\Theta}^R. \tag{3.9}$$

Defining

$$E_{g,a}^R = \frac{1}{b-a}E_g^R - E_{g'\Theta}^R, \tag{3.10}$$

and using integration by parts for the integral of (3.8),

$$\begin{aligned}\int_a^b g'(x)\Theta_a(x)dx &= \int_a^b g'(x)\frac{b-x}{b-a}dx \\ &= \left(g(x)\frac{b-x}{b-a}\right)\Big|_a^b + \frac{1}{b-a}\int_a^b g(x)dx \\ &= -g(a) + \frac{1}{b-a}\int_a^b g(x)dx, \end{aligned}\tag{3.11}$$

we get

$$\begin{aligned}\frac{\partial I_R}{\partial a} &= -\frac{1}{b-a}\int_a^b g(x)dx - g(a) + \frac{1}{b-a}\int_a^b g(x)dx + E_{g,a}^R \\ &= -g(a) + E_{g,a}^R. \end{aligned}\tag{3.12}$$

Comparison with (3.2) shows that (3.12) has truncation error $E_{g,a}^R = \frac{1}{b-a}E_g^R - E_{g'\Theta}^R$.

Similarly we find,

$$\frac{\partial I_R}{\partial b} = g(b) + E_{g,b}^R, \tag{3.13}$$

where $E_{g,b}^R = -\frac{1}{b-a}E_g + E_{g'\Theta} = -E_{g,a}$.

Now let us consider the truncation error in more detail. The truncation error for the rectangle rule is

$$E_g^R = \frac{g'(\eta^*)h(b-a)}{2},$$

for some $\eta^* \in [a,b]$, providing g' is continuous on $[a,b]$.

Similarly for $E^R_{g'\Theta}$, using $g \to g'(x)\Theta(x)$, we require $\frac{d}{dx}\Big(g'(x)\Theta(x)\Big)$ and hence g'' to be continuous, and obtain

$$\begin{aligned} E^R_{g'\Theta} &= -\frac{g''(\eta^{**})\frac{b-\eta^{**}}{b-a}(b-a)h}{2} - \frac{g'(\eta^{**})h}{2} \\ &= -\frac{g''(\eta^{**})(b-\eta^{**})h}{2} - \frac{g'(\eta^{**})h}{2}, \end{aligned} \tag{3.14}$$

for some $\eta^{**} \in [a,b]$. Therefore,

$$\begin{aligned} E^R_{g,a} = -E^R_{g,b} &= \frac{1}{b-a}E^R_g - E^R_{g'\Theta} \\ &= \frac{g'(\eta^{*})h}{2} + \frac{g'(\eta^{**})h}{2} + \frac{g''(\eta^{**})(\eta^{**}-b)}{2} \\ &= \frac{h}{2}\Big(g'(\eta^{*}) + g'(\eta^{**}) + g''(\eta^{**})(\eta^{**}-b)\Big) = O(h). \end{aligned} \tag{3.15}$$

So we see that direct automatic differentiation of the rectangle rule gives derivatives with respect to the end points which are correct to the same order of accuracy as the original quadrature rule within truncation error $O(h)$. As might be expected we require one degree higher differentiability of the function, i.e. continuous second derivatives instead of first, to ensure convergence of the derivative of the integral.

3.1.2 General Form

Using the same approach as for the rectangle rule to the general composite form for approximating the integral (3.1)

$$I = \sum_{i=0}^{n-1}\sum_{j=1}^{n} \omega_j g(x_i + \nu_j h), \tag{3.16}$$

we get the following derivatives of I with respect to a and b

$$\frac{\partial I}{\partial a} = -\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{n} \omega_j g(x_i+\nu_j h) + h\sum_{i=0}^{n-1}\sum_{j=1}^{n} \omega_j g'(x_i+\nu_j h)\left(\frac{n-i}{n} - \nu_j\frac{1}{n}\right)$$

$$
\begin{aligned}
&= -\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{n}\omega_j g(x_i+\nu_j h)+\frac{h}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{n}\omega_j g'(x_i+\nu_j h)(n-i-\nu_j)\\
&= -\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{n}\Big(\omega_j g(x_i+\nu_j h)+h\omega_j g'(x_i+\nu_j h)(-n+i+\nu_j)\Big)\\
&= -\frac{1}{n}\sum_{i=0}^{n-1}\sum_{j=1}^{n}\Big(\omega_j g(x_i+\nu_j h)+\omega_j g'(x_i+\nu_j h)(x_i-b+h\nu_j)\Big)\\
&= -\sum_{i=0}^{n-1}\sum_{j=1}^{n}h\omega_j g(x_i+\nu_j h)\frac{1}{b-a}-\sum_{i=0}^{n-1}\sum_{j=1}^{n}h\omega_j g'(x_i+\nu_j h)\frac{x_i+\nu_j h-b}{b-a},
\end{aligned}
$$

where ω_i and ν_i are the weights. Providing that g and $g'\Theta$ have continuous derivatives,

$$
\begin{aligned}
\frac{\partial I}{\partial a} &= -\frac{1}{b-a}\int_a^b g(x)dx-\int_a^b g'(x)\Theta(x)dx+\frac{1}{b-a}E_g-E_{g'\Theta}\\
&= -g(a)+E_{g,a},
\end{aligned}
\tag{3.17}
$$

where E_g, $E_{g'\Theta}$ are defined as the truncation errors for the quadrature applied to $\int_a^b g(x)dx$ and $\int_a^b g'(x)\Theta(x)dx$ respectively.

Similarly,

$$
\frac{\partial I}{\partial b} = g(b)+E_{g,b}, \tag{3.18}
$$

where

$$
E_{g,b} = -E_{g,a}. \tag{3.19}
$$

To guarantee convergence of the differentiated numerical integration scheme, the function must be one more degree continuously differentiable than is required for convergence of the quadrature scheme itself.

3.1.3 Results

We have differentiated the Matlab numerical quadrature `quad`. The developed tool is called `quadMAD` and has the same parameters as the standard Matlab integration routine. The function is available from the accompanying CD.

Consider the integration of the function $g(x) = 1 + e^{-x}\cos px$, where $p = 4$ over the fixed interval $[a, b] = [0, 1]$ applying various quadrature rules and estimating the truncation error.

The analytical result of the integral is

$$\begin{aligned}\int_0^1 g(x)dx &= \int_0^1 \left(1 + e^{-x}\cos 4x\right)dx \\ &= \left(x + \frac{4\sin 4x}{17e^x} - \frac{\cos 4x}{17e^x}\right)\Bigg|_0^1 \\ &= 1 + \frac{4\sin 4}{17e} - \frac{\cos 4}{17e} + \frac{1}{17} = 1.00745963139791\ldots\end{aligned}$$

and analytic derivatives w.r.t. a and b are

$$\begin{aligned}\frac{\partial}{\partial a}\left(\int_a^b g(x)dx\right) &= -g(a) = -2, \\ \frac{\partial}{\partial b}\left(\int_a^b g(x)dx\right) &= g(b) = 0.75953795003142\ldots\end{aligned}$$

N	$\frac{\partial I}{\partial a}$	error $\epsilon_N = \frac{\partial I}{\partial a} + g(a)$	$\frac{\epsilon_N}{\epsilon_{2N}}$
2	-2.362212	-0.36221	1.59402
4	-2.227232	-0.22723	1.79387
8	-2.126673	-0.12667	1.89996
16	-2.066667	-0.06667	1.95113
32	-2.034170	-0.03417	1.97629
64	-2.017294	-0.01729	1.98736

Table 3.1: Error in the derivative $\frac{\partial I}{\partial a}$ of the integral $I = \int_a^b \left(1 + e^{-x}\cos 4x\right) dx$ for $a = 0$ and $b = 1$ when differentiating the rectangle rule.

From (3.12) and (3.15) the error in the computation of $\frac{\partial I}{\partial a}$ behaves as for rectangular rule, $O(h)$. Consequently, the error on a mesh of N points should asymptotically

be twice that of a mesh of $2N$ points. In Table 3.1 we see that $\frac{\epsilon_N}{\epsilon_{2N}}$ is approaching 2 as predicted.

Similarly, we can show that the results obtained by differentiating the midpoint rule are accurate to order $O(h^2)$; and consequently $\frac{\epsilon_N}{\epsilon_{2N}} \to 4$. Results confirming this behaviour are shown in Table 3.2.

For Simpson's rule the order of convergence for the integral's derivatives is $O(h^4)$, and so $\frac{\epsilon_N}{\epsilon_{2N}} \to 16$ as $N \to \infty$ as shown in Table 3.3.

N	$\frac{\partial I}{\partial a}$	error $\epsilon_N = \frac{\partial I}{\partial a} + g(a)$	$\frac{\epsilon_N}{\epsilon_{2N}}$
2	-2.09225	-9.2252 x 10^{-2}	3.53266
4	-2.02611	-2.6114 x 10^{-2}	3.92061
8	-2.00666	-6.6607 x 10^{-3}	3.98200
16	-2.00167	-1.6727 x 10^{-3}	3.99574
32	-2.00042	-4.1862 x 10^{-4}	3.99904
64	-2.00003	-1.0468 x 10^{-4}	3.99957

Table 3.2: Numerical results for differentiating midpoint rule.

N	$\frac{\partial I}{\partial a}$	error $\epsilon_N = \frac{\partial I}{\partial a} + g(a)$	$\frac{\epsilon_N}{\epsilon_{2N}}$
2	-1.99553388	4.4661 x 10^{-3}	22.44948
4	-1.99980106	1.9894 x 10^{-4}	17.49692
8	-1.99998863	1.1370 x 10^{-5}	16.36796
16	-1.99999931	6.9465 x 10^{-7}	16.09178
32	-1.99999996	4.3168 x 10^{-8}	16.02257
64	-1.99999999	2.6942 x 10^{-9}	16.00642

Table 3.3: Numerical results for differentiating Simpson's rule.

3.2 Psychometric models using automatic differentiation

This example is taken from the article of Cudeck [29]. Following the main steps of the paper, but replacing derivative calculations by using `quadMAD` we aim to repeat the results generated in Cudeck's article.

Consider the problem of computing the tetrachoric correlation, ρ, for dichotomous variables, y_1 and y_2, assuming a bivariate normal distribution for the corresponding latent variables. The estimated tetrachoric correlation, $\hat{\rho}$, is the root of

$$h(\rho) = (p_1 p_2 - p_{00}) + I(\rho) = 0. \tag{3.20}$$

The algorithm cycles through three phases

$$I(\rho) = \frac{\rho}{2} \sum \omega_q g\left(\frac{1}{2}\rho(\nu_q + 1)\right) \quad \text{- quadrature rule,} \tag{3.21}$$

$$h(\rho) = p_1 p_2 - p_{00} + I(\rho), \tag{3.22}$$

$$\rho^{k+1} = \rho^k - h(\rho^k)/h'(\rho^k). \tag{3.23}$$

Equation (3.23) is Newton's method.

The initial value for ρ_0 is

$$\rho^0 = \cos\left[\pi\left(1 + \sqrt{p_a p_d / p_b p_c}\right)^{-1/2}\right], \tag{3.24}$$

where $p_a = p_{00}, p_b, p_c$, and p_d are the simple joint proportions of the 2 x 2 table. It is important to note that $p_a = p_{00}$, not $p_d = p_{00}$, as it was given in Cudeck's paper, since it crucially influences computational results. The marginal probabilities are $p_1 = p_a + p_c$, $p_2 = p_a + p_b$; and z_1, z_2 are normal deviates. The function g is defined as following

$$g(y) = \frac{\exp(t(y))}{2\pi\sqrt{1 - y^2}}, \tag{3.25}$$

where

$$t(y) = \frac{-\left(z_1^2 + z_2^2 - 2z_1 z_2 y\right)}{2(1-y^2)}, \tag{3.26}$$

and standard error estimation $se(\rho)$ is defined by Hamdan's formula,

$$se(\rho) = \frac{1}{\sqrt{n}g(\hat{\rho})}\left(\frac{1}{p_a} + \frac{1}{p_b} + \frac{1}{p_c} + \frac{1}{p_d}\right)^{-1/2}. \tag{3.27}$$

Here, another error was made in the original article, which showed $\sqrt{ng(\hat{\rho})}$ in the denominator of (3.27).

In an AD implementation of the process when $h(\rho)$ is computed, $h'(\rho)$ is simultaneously available simplifying the implementation of Newton's method.

	Inoculated	Not Inoculated	Totals
Army in India			
Escaped	10,798	109,034	119,832
Cases	84	1,475	1,559
Totals	10,882	110,509	121,391
Ladysmith Garrison			
Escaped	1,670	9,040	10,710
Cases	35	1,489	1,524
Totals	1,705	10,529	12,234

Table 3.4: Two samples of British military personnel, classified by inoculation (yes or no) and disease status (yes or no).

Considering the first example of Cudeck's paper [29], Army in India, p_a, p_b, p_c and p_d are defined in Table 3.5.

The army in India table gave

Cudeck's result	$\hat{\rho} = 0.100$	$se(\hat{\rho}) = 0.02$
Our result	$\hat{\rho} = 0.0994$	$se(\hat{\rho}) = 0.0186$

The difference can be explained by the rough rounding of Cudeck's results. The Ladysmith Garrison produced

	Inoculated	Not Inoculated	Totals
Army in India			
Escaped	$10{,}798 = p_a$	$109{,}034 = p_b$	$119{,}832 = p_2$
Cases	$84 = p_c$	$1{,}475 = p_d$	1,559
Totals	$10{,}882 = p_1$	110,509	121,391

Table 3.5: Defining the parameters p_a,p_b,p_c and p_d for Cudeck's test case.

$$\begin{array}{lll} \text{Cudeck's result} & \hat{\rho} = 0.447 & se(\hat{\rho}) = 0.03 \\ \text{Our result} & \hat{\rho} = 0.415 & se(\hat{\rho}) = 0.0189 \end{array}$$

Though there is a significant difference between results which does not look like a problem of rounding, there are still doubts about the absolute reliability of Cudeck's article and his results. It might be just another mistake. Similarly, we run the same program for data from [30] for the final test. The 2 x 2 table gives $p_a = 0.3$, $p_b = 0.1$, $p_c = 0.2$, and $p_d = 0.4$.

$$\begin{array}{ll} \text{Bonett's result} & \hat{\rho} = 0.6071 \\ \text{Our result} & \hat{\rho} = 0.6055 \end{array}$$

This difference was expected and can be a result of simple substitution of quadrature rule for computation of (3.21).

3.3 Conclusions

In this chapter we introduced the general definitions of the AD theory and applied them to the differentiation of the numerical integration scheme. The algorithm for numerical integration differentiation is implemented in Matlab as `quadMAD` routine, that is available from the CD accompanying this work.

Chapter 4

Taylor series for moments estimation

We assume $\mathbf{x}$ is a random vector with n independent random components $\mathbf{x} = \{x_1, \ldots, x_n\}$ of known probability distribution, i.e. mean $\mu_{\mathbf{x}} = \{\mu_{x_1}, \ldots, \mu_{x_n}\}$, standard deviation $\sigma_{\mathbf{x}} = \{\sigma_{x_1}, \ldots, \sigma_{x_n}\}$, and higher order moments are known. Further, let g be any smooth function that represents our model.

The idea of the moments method is to approximate the distribution of $y = g(\mathbf{x})$ in terms of its derivatives by using Taylor approximations of the statistical moments. There are several ways of doing it.

The most straight forward and well-known ([4],[15], etc) approach is to apply the expectation operator to the Taylor expansion of the function $g(\mathbf{x})$. This way one obtains the expectation for the function g. By squaring it and subtracting from the expectation of the squared Taylor approximation of the same function $g(\mathbf{x})$ one satisfies the definition of the variance (2.11). Depending on the prescribed input distributions and properties of the output function, the first and second order moments methods may become inaccurate. To improve the precision, higher order approximations are required. Therefore, higher order derivatives must be evaluated. This can be done by using automatic differentiation. Although it sounds simple, the mathematical implementation of such method faces computational difficulties.

On the other hand, one can compute the expectation the same way as it is done in the first approach, but for the variance avoid squaring the Taylor series by squaring

the function instead, and rewriting the expectation of $g(\mathbf{x})$ as the expectation of $g^2(\mathbf{x})$.

In this chapter we consider both ways of computing statistical moments for the function $g(\mathbf{x})$ in details, validate already published results, generalise the approach, and investigate convergence issues of the method.

4.1 First approach

4.1.1 Single variable case

For the simplification of the analytical calculation process we first consider the case where $n = 1$. One must then obtain the expectation μ_g and variance σ_g of the known output function $g(x)$ for given μ_x and σ_x of the scalar input x.

The Taylor expansion of $y = g(x)$ at μ_x is

$$g(x) = g + \frac{\partial g}{\partial x}(x - \mu_x) + \frac{1}{2!}\frac{\partial^2 g}{\partial x^2}(x - \mu_x)^2 + \ldots, \tag{4.1}$$

where all evaluations of function g and its derivatives are made at μ_x.

Using the definition (2.4) and properties (2.6)-(2.9) the first moment of $g(x)$ can be approximated by

$$\mu_g = E\big(g(x)\big) = E\big(g\big) + \frac{\partial g}{\partial x}E(x - \mu_x) + \frac{1}{2!}\frac{\partial^2 g}{\partial x^2}E\big((x - \mu_x)^2\big) + \ldots \tag{4.2}$$

Then, since

$$E(x - \mu_x) = E(x) - E(\mu_x) = \mu_x - \mu_x = 0, \tag{4.3}$$

the moments method for expectation gives

$$\mu_g = g + \frac{1}{2!}\frac{\partial^2 g}{\partial x^2}\sigma_x^2 + \ldots \tag{4.4}$$

According to [4], ***the i^{th} order moments method*** is the method that corresponds

to a derivation of the moments approximation based on an i^{th} order Taylor series. Therefore, the first order moments method approximation for expectation is

$$\mu_g = g + O(\sigma_x^2), \tag{4.5}$$

and second order moments method is

$$\mu_g = g + \frac{1}{2!}\frac{\partial^2 g}{\partial x^2}\sigma_x^2 + O(\sigma_x^3). \tag{4.6}$$

These results are identical to the ones obtained by Ghate and Giles [4].

We now can use (4.6) and the definition of the variance (2.11) to obtain the second order moments method approximation for variance as in [4]. First, we square the second order Taylor approximation of the function $g(x)$:

$$\begin{aligned} g^2(x) \approx\ & g^2 + 2g\frac{\partial g}{\partial x}(x-\mu_x) + \left(g\frac{\partial^2 g}{\partial x^2} + \left(\frac{\partial g}{\partial x}\right)^2\right)(x-\mu_x)^2 \\ & + \frac{\partial g}{\partial x}\frac{\partial^2 g}{\partial x^2}(x-\mu_x)^3 + \frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\right)^2(x-\mu_x)^4 \end{aligned} \tag{4.7}$$

By taking the expectation of (4.7), as we did for (4.2), we get:

$$\begin{aligned} E\left(g^2(x)\right) =\ & g^2 + \left(g\frac{\partial^2 g}{\partial x^2} + \left(\frac{\partial g}{\partial x}\right)^2\right)\sigma_x^2 \\ & + \frac{\partial g}{\partial x}\frac{\partial^2 g}{\partial x^2}S(x)\sigma_x^3 + \frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\right)^2 K(x)\sigma_x^4, \end{aligned} \tag{4.8}$$

where $S(x)$ is skewness, and $K(x)$ is kurtosis, as defined in (2.14)-(2.15). Knowing (2.11) and using μ_g from (4.6), the second order moments method approximation for the variance is

$$\sigma_g^2 = \left(\frac{\partial g}{\partial x}\right)^2\sigma_x^2 + \frac{\partial^2 g}{\partial x^2}\frac{\partial g}{\partial x}S(x)\sigma_x^3 + \frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\right)^2\big(K(x)-1\big)\sigma_x^4. \tag{4.9}$$

The order of error for the variance in (4.9) can be demonstrated by taking a 3^{rd} order

Taylor expansion. Subtracting from

$$\begin{aligned}\tilde{E}\left(g^2(x)\right) &= g^2+\left(g\frac{\partial^2 g}{\partial x^2}+\left(\frac{\partial g}{\partial x}\right)^2\right)\sigma_x^2 \\ &+\left(\frac{1}{3}g\frac{\partial^3 g}{\partial x^3}+\frac{\partial g}{\partial x}\frac{\partial^2 g}{\partial x^2}\right)S(x)\sigma_x^3 \\ &+\left(\frac{1}{3}\frac{\partial g}{\partial x}\frac{\partial^3 g}{\partial x^3}+\frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\right)^2\right)K(x)\sigma_x^4 \\ &+\frac{1}{6}\frac{\partial^2 g}{\partial x^2}\frac{\partial^3 g}{\partial x^3}M_5(x)\sigma_x^5+\frac{1}{36}\left(\frac{\partial^3 g}{\partial x^3}\right)^2 M_6(x)\sigma_x^6, \qquad (4.10)\end{aligned}$$

where M_5 and $M_6(x)$ are the fifth and the sixth moments, respectively, as defined in (2.13), the squared third order moments method for the expectation is

$$\begin{aligned}\tilde{\mu}_g^2 &= g^2+g\frac{\partial^2 g}{\partial x^2}\sigma_x^2+\frac{1}{3}g\frac{\partial^3 g}{\partial x^3}S(x)\sigma_x^3+\frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\right)^2\sigma_x^4 \\ &+\frac{1}{6}\frac{\partial^2 g}{\partial x^2}\frac{\partial^3 g}{\partial x^3}S(x)\sigma_x^5+\frac{1}{36}\left(\frac{\partial^3 g}{\partial x^3}\right)^2 S^2(x)\sigma_x^6. \qquad (4.11)\end{aligned}$$

The third order moments method for variance then is

$$\begin{aligned}\tilde{\sigma}_g^2 &= \left(\frac{\partial g}{\partial x}\right)^2\sigma_x^2+\frac{\partial g}{\partial x}\frac{\partial^2 g}{\partial x^2}S(x)\sigma_x^3+\frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\right)^2\left(K(x)-1\right)\sigma_x^4 \\ &+\frac{1}{3}\frac{\partial g}{\partial x}\frac{\partial^3 g}{\partial x^3}K(x)\sigma_x^4+O(\sigma_x^5). \qquad (4.12)\end{aligned}$$

When we compare (4.9) and (4.12), the missing terms in (4.9) at 4^{th} order are revealed:

$$|\sigma_g^2-\tilde{\sigma}_g^2|=\frac{1}{3}\frac{\partial g}{\partial x}\frac{\partial^3 g}{\partial x^3}K(x)\sigma_x^4. \qquad (4.13)$$

Thus (4.9) is only 3^{rd} order accurate, since the 4^{th} order term is incomplete.

It proves that by following this approach, and despite the fact that an i^{th} order Taylor approximation gives an i^{th} order moments method for expectation with error $O(\sigma^{i+1})$, an i^{th} order Taylor series does not produce an i^{th} order error for variance. In particular, the order of error for (4.9) is $O(\sigma_x^4)$.

In the case when the function has more than one output $\mathbf{y} = \big(g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})\big)$, one must compute the covariance matrix

$$C_y = \begin{pmatrix} \sigma^2_{g_1} & C_{g_1 g_2} & \ldots & C_{g_1 g_m} \\ C_{g_2 g_1} & \sigma^2_{g_2} & \ldots & C_{g_2 g_m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{g_m g_1} & C_{g_m g_2} & \ldots & \sigma^2_{g_m} \end{pmatrix}, \tag{4.14}$$

where the diagonal elements can be computed using (4.12). Since

$$C_{g_i g_j} = E\big((g_i - \mu_{g_i})(g_j - \mu_{g_j})\big) = E(g_i g_j) - E(g_i)E(g_j), \tag{4.15}$$

we now need to approximate $E(g_i g_j)$ instead of $E(g_i^2)$ as for the variance. Using the second order Taylor series for both g_i and g_j, the Taylor approximation for $g_i g_j$ becomes

$$\begin{aligned} g_i g_j \approx\ & g_i g_j + g_i \frac{\partial g_j}{\partial x}(x - \mu_x) + g_j \frac{\partial g_i}{\partial x}(x - \mu_x) \\ & + g_i \frac{1}{2!}\frac{\partial^2 g_j}{\partial x^2}(x - \mu_x)^2 + g_j \frac{1}{2!}\frac{\partial^2 g_i}{\partial x^2}(x - \mu_x)^2 \\ & + \frac{\partial g_i}{\partial x}\frac{\partial g_j}{\partial x}(x - \mu_x)^2 + \left(\frac{1}{2!}\right)^2 \frac{\partial^2 g_i}{\partial x^2}\frac{\partial^2 g_j}{\partial x^2}(x - \mu_x)^4 \\ & + \frac{1}{2!}\frac{\partial g_i}{\partial x}\frac{\partial^2 g_j}{\partial x^2}(x - \mu_x)^3 + \frac{1}{2!}\frac{\partial g_j}{\partial x}\frac{\partial^2 g_i}{\partial x^2}(x - \mu_x)^3. \end{aligned} \tag{4.16}$$

By applying the expectation operator to (4.16) we get $E(g_i g_j)$ for (4.15):

$$\begin{aligned} E(g_i g_j) =\ & g_i(\mu_x) g_j(\mu_x) + \left(\frac{1}{2} g_i \frac{\partial^2 g_j}{\partial x^2} + \frac{1}{2} g_j \frac{\partial^2 g_i}{\partial x^2} + \frac{\partial g_i}{\partial x}\frac{\partial g_j}{\partial x}\right)\sigma_x^2 \\ & + \frac{1}{2}\left(\frac{\partial g_i}{\partial x}\frac{\partial^2 g_j}{\partial x^2} + \frac{\partial g_j}{\partial x}\frac{\partial^2 g_i}{\partial x^2}\right) S(x)\sigma_x^3 \\ & + \frac{1}{4}\frac{\partial^2 g_i}{\partial x^2}\frac{\partial^2 g_j}{\partial x^2} K(x)\sigma_x^4. \end{aligned} \tag{4.17}$$

Thus the covariance $C_{g_i g_j}$ is

$$\begin{aligned} C_{g_i g_j} &= \frac{\partial g_i}{\partial x}\frac{\partial g_j}{\partial x}\sigma_x^2 + \frac{1}{2}\left(\frac{\partial g_i}{\partial x}\frac{\partial^2 g_j}{\partial x^2} + \frac{\partial g_j}{\partial x}\frac{\partial^2 g_i}{\partial x^2}\right) S(x)\sigma_x^3 \\ &\quad + \frac{1}{4}\frac{\partial^2 g_i}{\partial x^2}\frac{\partial^2 g_j}{\partial x^2}(K(x)-1)\sigma_x^4. \end{aligned} \tag{4.18}$$

It is obvious that the formula for the covariance (4.18) is the same as the one for the variance (4.9) when $i = j$.

Similarly to the moments method comparison for the variance when using the second and the third order Taylor approximations to reveal the missing terms of the 4^{th} order, we use the third order Taylor series to compute the difference for the covariance. Therefore, by subtracting from

$$\begin{aligned} \tilde{E}(g_i g_j) &= g_i(\mu_x)g_j(\mu_x) + \left(\frac{1}{2}g_i\frac{\partial^2 g_j}{\partial x^2} + \frac{1}{2}g_j\frac{\partial^2 g_i}{\partial x^2} + \frac{\partial g_i}{\partial x}\frac{\partial g_j}{\partial x}\right)\sigma_x^2 \\ &\quad + \frac{1}{6}\left(g_i\frac{\partial^3 g_j}{\partial x^3} + g_j\frac{\partial^3 g_i}{\partial x^3}\right) S(x)\sigma_x^3 \\ &\quad + \frac{1}{2}\left(\frac{\partial g_i}{\partial x}\frac{\partial^2 g_j}{\partial x^2} + \frac{\partial g_j}{\partial x}\frac{\partial^2 g_i}{\partial x^2}\right) S(x)\sigma_x^3 \\ &\quad + \frac{1}{6}\left(\frac{\partial g_i}{\partial x}\frac{\partial^3 g_j}{\partial x^3} + \frac{\partial g_j}{\partial x}\frac{\partial^3 g_i}{\partial x^3}\right) K(x)\sigma_x^4 \\ &\quad + \frac{1}{4}\frac{\partial^2 g_i}{\partial x^2}\frac{\partial^2 g_j}{\partial x^2}K(x)\sigma_x^4 + \frac{1}{36}\frac{\partial^3 g_i}{\partial x^3}\frac{\partial^3 g_j}{\partial x^3}M_6\sigma_x^6 \\ &\quad + \frac{1}{12}\left(\frac{\partial^2 g_i}{\partial x^2}\frac{\partial^3 g_j}{\partial x^3} + \frac{\partial^2 g_j}{\partial x^2}\frac{\partial^3 g_i}{\partial x^3}\right) M_5(x)\sigma_x^5 \end{aligned} \tag{4.19}$$

the multiplied $\tilde{E}(g_i)$ and $\tilde{E}(g_j)$ that are also based on the third order Taylor series, the third order moments method for the covariance then becomes

$$\begin{aligned} \tilde{C}_{g_i g_j} &= \frac{\partial g_i}{\partial x}\frac{\partial g_j}{\partial x}\sigma_x^2 + \frac{1}{2}\left(\frac{\partial g_i}{\partial x}\frac{\partial^2 g_j}{\partial x^2} + \frac{\partial g_j}{\partial x}\frac{\partial^2 g_i}{\partial x^2}\right) S(x)\sigma_x^3 \\ &\quad + \frac{1}{4}\frac{\partial^2 g_i}{\partial x^2}\frac{\partial^2 g_j}{\partial x^2}(K(x)-1)\sigma_x^4 \\ &\quad + \frac{1}{6}\left(\frac{\partial g_i}{\partial x}\frac{\partial^3 g_j}{\partial x^3} + \frac{\partial g_j}{\partial x}\frac{\partial^3 g_i}{\partial x^3}\right) K(x)\sigma_x^4 + O(\sigma_x^5). \end{aligned} \tag{4.20}$$

Hence the missing term of the 4^{th} order for the covariance is

$$|C_{g_i g_j} - \tilde{C}_{g_i g_j}| = \frac{1}{6}\left(\frac{\partial g_i}{\partial x}\frac{\partial^3 g_j}{\partial x^3} + \frac{\partial g_j}{\partial x}\frac{\partial^3 g_i}{\partial x^3}\right) K(x)\sigma_x^4. \tag{4.21}$$

4.1.2 Multiple variable case

In the case of $\mathbf{x} \in \mathbb{R}^n$, the Taylor series expansion for the output function $g(\mathbf{x})$ about the expectation $\mu_{\mathbf{x}}$ is

$$\begin{aligned} g(\mathbf{x}) \;=\; & g + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(x_i - \mu_{x_i}) \\ & + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\ & + \frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k}(x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \\ & + \dots \end{aligned} \tag{4.22}$$

In a similar manner to the single variable case, for vector inputs the expectation is given by:

$$\begin{aligned} \mu_g \;=\; & E\big(g(\mathbf{x})\big) = E\big(g\big) + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i} E(x_i - \mu_{x_i}) \\ & + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j} E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})\big) \\ & + \frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k} E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k})\big) \\ & + O(\sigma_{\mathbf{x}}^4). \end{aligned} \tag{4.23}$$

After taking into account the assumption of uncorrelated inputs

$$E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})\big) = \begin{cases} \sigma_{x_i}^2, & \text{if i = j,} \\ 0, & \text{otherwise,} \end{cases} \tag{4.24}$$

and applying the same sequence of actions as for the single variable case, the second order moment approximations to the expectation and variance are

$$\mu_g = g + \frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2 + O(\sigma_{\mathbf{x}}^3), \tag{4.25}$$

and

$$\begin{aligned}\sigma_g^2 &= \sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2\sigma_{x_i}^2 + \sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial g}{\partial x_i}S(x_i)\sigma_{x_i}^3 \\ &+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2\sigma_{x_i}^2\sigma_{x_j}^2 + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^3 g}{\partial x_i^2 \partial x_j}\frac{\partial g}{\partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2 \\ &+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\left(K(x_i)-1\right)\sigma_{x_i}^4 + O(\sigma_{\mathbf{x}}^4),\end{aligned} \tag{4.26}$$

respectively. When the function $\mathbf{y} = g(\mathbf{x}) = \big(g_1(\mathbf{x}),\ldots,g_m(\mathbf{x})\big)$ is a vector function, the second order moment is defined as a covariance matrix

$$C_y = \begin{pmatrix} \sigma_{g_1}^2 & C_{g_1g_2} & \ldots & C_{g_1g_m} \\ C_{g_2g_1} & \sigma_{g_2}^2 & \ldots & C_{g_2g_m} \\ \vdots & \vdots & \ddots & \vdots \\ C_{g_mg_1} & C_{g_mg_2} & \ldots & \sigma_{g_m}^2 \end{pmatrix}, \tag{4.27}$$

where $C_{g_pg_q} = E(g_pg_q) - E(g_p)E(g_q)$. The second order moment approximation for the covariance is

$$\begin{aligned}C_{g_pg_q} &= \sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2}S(x_i)\sigma_{x_i}^3 \\ &+\frac{1}{2}\sum_{i=1}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^2 g_p}{\partial x_i^2}S(x_i)\sigma_{x_i}^3 + \frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\big(K(x_i)-1\big)\sigma_{x_i}^4 \\ &+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2,\end{aligned} \tag{4.28}$$

where $p,q = 1,\ldots,m$. Now the variance approximation (4.26) is a partial case

of (4.28) when $p = q$.

The incompleteness of the fourth order term in both (4.26) and (4.28) can be demonstrated by considering the third order moments approximations. The details of these computations are given in Appendix A. In summary, the moments method based on the third order Taylor series for the covariance and its partial case, variance, is

$$\begin{aligned}
\tilde{C}_{q_p g_q} &= \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2} S(x_i)\sigma_{x_i}^3 \\
&+ \frac{1}{2}\sum_{i=1}^{n} \frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial g_q}{\partial x_i} S(x_i)\sigma_{x_i}^3 + \frac{1}{6}\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3} K(x_i)\sigma_{x_i}^4 \\
&+ \frac{1}{6}\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i^3} K(x_i)\sigma_{x_i}^4 + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&+ \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&+ \frac{1}{4}\sum_{i=1}^{n} \frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\big(K(x_i)-1\big)\sigma_{x_i}^4
\end{aligned} \tag{4.29}$$

and

$$\begin{aligned}
\tilde{\sigma}_g^2 &= \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i}\right)^2 \sigma_{x_i}^2 + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2} S(x_i)\sigma_{x_i}^3 \\
&+ \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2 \sigma_{x_i}^2\sigma_{x_j}^2 + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&+ \frac{1}{4}\sum_{i=1}^{n} \left(\frac{\partial^2 g}{\partial x_i^2}\right)^2 \big(K(x_i)-1\big)\sigma_{x_i}^4 + \frac{1}{3}\sum_{i=1}^{n} \frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i^3} K(x_i)\sigma_{x_i}^4
\end{aligned} \tag{4.30}$$

respectively. Thus the errors in second order approximations are

$$\begin{aligned}
|C_{g_p g_q} - \tilde{C}_{g_p g_q}| &= \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&+ \frac{1}{6}\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3} K(x_i)\sigma_{x_i}^4 + \frac{1}{6}\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i^3} K(x_i)\sigma_{x_i}^4,
\end{aligned} \tag{4.31}$$

and

$$|\sigma_g^2 - \tilde{\sigma}_g^2| = \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\partial g}{\partial x_i} \frac{\partial^3 g}{\partial x_i \partial x_j^2} \sigma_{x_i}^2 \sigma_{x_j}^2 + \frac{1}{3} \sum_{i=1}^{n} \frac{\partial g}{\partial x_i} \frac{\partial^3 g}{\partial x_i^3} K(x_i) \sigma_{x_i}^4 \tag{4.32}$$

for the covariance and the variance respectively.

Performing the analyses of this sort becomes more and more algebraically challenging with increasing the order of Taylor series. For previous authors, for whom higher order derivatives of the function were difficult to obtain, the issue of deriving higher order moments method did not have a priority. With the current rapid improvements in automatic differentiation techniques, we can now acquire higher order terms. Therefore, the second approach to compute moments based on Taylor approximation we suggest in this work becomes necessary.

4.2 Second approach

To confirm the results we can use a slightly different approach. To evaluate the moments approximation for function g, one can simply substitute the squared function g^2 on the place of g in (4.22) instead of squaring the Taylor series of g for computing $E(g^2)$. Thus the Taylor approximation for g^2 can be rewritten as following

$$\begin{aligned} g^2(\mathbf{x}) \;=\; & g^2(\mu_{\mathbf{x}}) + \sum_{i=1}^{n} 2g \frac{\partial g}{\partial x_i} (x_i - \mu_{x_i}) \\ & + \frac{1}{2!} \sum_{i=1}^{n} \sum_{j=1}^{n} 2 \left(\frac{\partial g}{\partial x_i} \frac{\partial g}{\partial x_j} + g \frac{\partial^2 g}{\partial x_i \partial x_j} \right) (x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\ & + \frac{1}{3!} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 2 \left(\frac{\partial g}{\partial x_i} \frac{\partial^2 g}{\partial x_j \partial x_k} + \frac{\partial g}{\partial x_j} \frac{\partial^2 g}{\partial x_i \partial x_k} \right. \\ & \qquad \left. + \frac{\partial g}{\partial x_k} \frac{\partial^2 g}{\partial x_i \partial x_j} + g \frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k} \right) \times \\ & \qquad \times (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \;+\; \ldots \end{aligned} \tag{4.33}$$

Using this approach the i^{th} order Taylor series produces i^{th} order moments method with error $O(\sigma^{i+1})$ for all statistical moments approximations.

4.2.1 Uncorrelated inputs

Applying the expectation operator to (4.33) and assuming uncorrelated input condition (4.24) we get

$$\begin{aligned} E(g^2(\mathbf{x})) &= g^2 + \frac{1}{2!}\sum_{i=1}^{n} 2\left(\left(\frac{\partial g}{\partial x_i}\right)^2 + g\frac{\partial^2 g}{\partial x_i^2}\right) E\left((x_i - \mu_{x_i})^2\right) \\ &\quad + \frac{1}{3!}\sum_{i=1}^{n} 2\left(3\frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2} E\left((x_i - \mu_{x_i})^3\right) + g\frac{\partial^3 g}{\partial x_i^3} E\left((x_i - \mu_{x_i})^3\right)\right) \\ &\quad + \dots \end{aligned}$$

That is equivalent to

$$\begin{aligned} \mu_{g^2} &= g^2 + \sum_{i=1}^{n}\left(g\frac{\partial^2 g}{\partial x_i^2} + \left(\frac{\partial g}{\partial x_i}\right)^2\right)\sigma_{x_i}^2 \\ &\quad + \sum_{i=1}^{n}\left(\frac{1}{3}g\frac{\partial^3 g}{\partial x_i^3} + \frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2}\right) S(x_i)\sigma_{x_i}^3 + \dots \end{aligned} \tag{4.34}$$

To compute the 3^{rd} order approximation for the variance using (2.11) we require the squared 3^{rd} order approximation of expectation. The 3^{rd} order approximation for expectation is

$$\mu_g = g + \frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2 + \frac{1}{3!}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3} S(x_i)\sigma_{x_i}^3 + O(\sigma_{\mathbf{x}}^4). \tag{4.35}$$

After squaring (4.35) and neglecting 4^{th} and higher order terms, we get

$$\mu_g^2 = g^2 + g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{3}g\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3} S(x_i)\sigma_{x_i}^3. \tag{4.36}$$

Subtracting (4.36) from the 3^{rd} order Taylor series for the squared function (4.33) gives

$$\sigma_g^2 = \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i} \right)^2 \sigma_{x_i}^2 + \sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i^2} \frac{\partial g}{\partial x_i} S(x_i) \sigma_{x_i}^3. \tag{4.37}$$

Using this approach, the resulting moments method has the order of error $O(\sigma_{\mathbf{x}}^{p+1})$, where p is the order of Taylor series. This way we can get the results from the section 4.1.2, verify them and reveal if there are possibly any missing terms. Following the same steps, the 4^{th} order approximation for the variance is

$$\begin{aligned} \sigma_g^2 \;=\; & E(g^2) - E(g)^2 = \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i} \right)^2 \sigma_{x_i}^2 + \sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i^2} \frac{\partial g}{\partial x_i} S(x_i) \sigma_{x_i}^3 \\ & + \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\partial^3 g}{\partial x_i \partial x_j^2} \frac{\partial g}{\partial x_i} \sigma_{x_i}^2 \sigma_{x_j}^2 + \frac{1}{2} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(\frac{\partial^2 g}{\partial x_i \partial x_j} \right)^2 \sigma_{x_i}^2 \sigma_{x_j}^2 \\ & + \frac{1}{3} \sum_{i=1}^{n} \frac{\partial^3 g}{\partial x_i^3} \frac{\partial g}{\partial x_i} K(x_i) \sigma_{x_i}^4 + \frac{1}{4} \sum_{i=1}^{n} \left(\frac{\partial^2 g}{\partial x_i^2} \right)^2 \big(K(x_i) - 1\big) \sigma_{x_i}^4 \\ & + O(\sigma_{\mathbf{x}}^5). \end{aligned} \tag{4.38}$$

The details of these calculations are given in Appendix A. When comparing the variance representation obtained in Section 4.1.2 and (4.38) we reveal that the third order approximation (4.30) has the order $O(\sigma_{\mathbf{x}}^5)$, as there are no derivatives of the fourth order in (4.38):

$$|\sigma_g^2 - \tilde{\sigma_g^2}| = O(\sigma_{\mathbf{x}}^5). \tag{4.39}$$

Currently, the only record of generalised moments method with no assumptions on the input distributions in the case of independent inputs variables is available in the paper by Christianson and Cox [16], where it is written as

$$\begin{aligned} E\big((g - E(g))^2\big) \;=\; & \sum_{i=1}^{n} \big(g^{(i)}\big)^2 + 2S_i g^{(i)} g^{(ii)} + (K_i - 1)\big(g^{(ii)}\big)^2 + 2K_i g^{(i)} g^{(iii)} \\ & + \sum_{i<j}^{n} \big(g^{(ij)}\big)^2 + 2g^{(i)} g^{(ijj)} + 2g^{(iij)} g^{(j)} + \dots \end{aligned} \tag{4.40}$$

Since $y^{(i)}$, $y^{(ij)}$, $y^{(ijk)}$, etc. are declared as Taylor coefficients, the substitution

$$
\begin{aligned}
y^{(i)} &= \frac{\partial g}{\partial x_i}\sigma_{x_i}, && (4.41)\\
y^{(ij)} &= \frac{1}{2!}\frac{\partial^2 g}{\partial x_i \partial x_j}\sigma_{x_i}\sigma_{x_j}, && (4.42)\\
y^{(ijk)} &= \frac{1}{3!}\frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k}\sigma_{x_i}\sigma_{x_j}\sigma_{x_k}, && (4.43)\\
&\text{etc.}
\end{aligned}
$$

into (4.40) leads to our formula (4.38).

In a similar manner we compute the moments method for the covariance of the vector function $g(\mathbf{x}) = \big(g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})\big)$. Based on the 4^{th} order Taylor approximation, the second order moments method for the covariance is

$$
\begin{aligned}
C_{g_p g_q} &= E(g_p g_q) - E(g_p)E(g_q)\\
&= \sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2} + \frac{\partial^2 g_q}{\partial x_i^2}\frac{\partial g_q}{\partial x_i}\right)S(x_i)\sigma_{x_i}^3\\
&\quad + \frac{1}{6}\sum_{i=1}^{n}\left(\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3} + \frac{\partial^3 g_p}{\partial x_i^3}\frac{\partial g_q}{\partial x_i}\right)K(x_i)\sigma_{x_i}^4\\
&\quad + \frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\big(K(x_i)-1\big)\sigma_{x_i}^4\\
&\quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i \partial x_j^2} + \frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j} + \right.\\
&\qquad\qquad \left. + \frac{\partial^3 g_p}{\partial x_i \partial x_j^2}\frac{\partial g_q}{\partial x_i}\right)\sigma_{x_i}^2\sigma_{x_j}^2. \qquad (4.44)
\end{aligned}
$$

The comparison of the moments method for the covariance (4.29) based on the third order Taylor series obtained using the first approach and the fourth order moments method (4.44) using the second approach again demonstrates the completeness of (4.29):

$$
|C_{q_p g_q} - \tilde{C}_{g_p g_q}| = O(\sigma_{\mathbf{x}}^5).
$$

4.2.2 Correlated inputs

Whenever the condition for uncorrelated inputs (4.24) does not hold, in other words, the input variables are dependent, the input variance vector must be substituted by the covariance matrix (2.32). The same thing happens to the higher order statistical moments of the input variables. From now on they are defined by $\underbrace{k \times k \times \cdots \times k}_{k-times}$ arrays

$$\{M^k\} = \left\{M^k_{i_1,\ldots,i_k}\right\} = \left\{E\left((x_{i_1} - \mu_{x_{i_1}})\ldots(x_{i_k} - \mu_{x_{i_k}})\right)\right\}, \tag{4.45}$$

where k is the order of the moment, $i_j = 1,\ldots,n$ for any $j = 1,\ldots,k$. Therefore, applying the expectation operator to (4.22), the expectation of the function $g(\mathbf{x})$ becomes

$$\begin{aligned} \mu_g &= E\big(g(\mathbf{x})\big) = g(\mu_{\mathbf{x}}) + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 g}{\partial x_i \partial x_j}M^2_{i,j} \\ &\quad + \frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k}M^3_{i,j,k} + \ldots \end{aligned} \tag{4.46}$$

Analogously to (4.37) and (4.38), the formulation of the variance can be obtained.

4.3 Convergence of the method

It is important to be able to determine the correct order of the Taylor series required to obtain the accurate approximation of the moments for the given function g. As we will see in this section, if the function $g(x)$ has finite radius of convergence then using too many Taylor terms can cause the series to diverge, and too few may not be enough for convergence.

Here we demonstrate the significance of Taylor order choice by considering the test function from Christianson's paper [16] in detail.

Example 4.3.1. The function

$$g(x) = \frac{1}{1+x^2} \tag{4.47}$$

is always finite, and has a Taylor series for any finite $x \in \mathbb{R}$. The Taylor polynomial of $g(x)$ about $x = 0$ is given by

$$g(x) = 1 - x^2 + x^4 - x^6 + x^8 - \dots \tag{4.48}$$

For $|x| \geq 1$ the series does not converge. Based on that fact, the approximation of the statistical moments using Taylor series can not be trusted either. Table 4.1 illustrates the divergence of the expectation approximation for the function (4.47) when the number of Taylor terms p is increasing.

Quadrature	p	μ_g
3.1243	2	-2.2204e-16
	4	3
	6	-12
	8	93
	10	-852

Table 4.1: Comparison of the expectation for the function $g = \frac{1}{1+x^2}$ with $x \in N(0,1)$, computed by using the quadrature rule, with the expectation approximation based on p^{th} order Taylor expansion.

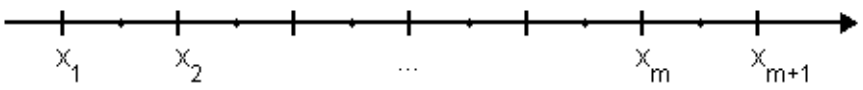
4.3.1 Partitioning approach

To deal with the finite radius of convergence caused by imaginary poles of the function (4.47), Christianson and Cox [16], citing personal communication with Harley Flanders, suggested the so-called *partitioning approach*. Although the idea of the method was published, it has to the best of our knowledge never been implemented.

For simplification, in this work we consider the partitioning approach for the one dimensional case, meaning a single input variable results in a single output.

To start with, we subdivide the space of input parameter into subintervals in the way that only a finite number of subintervals have an associated non-negligible pro-

bability. Probabilities $P(x < x_1)$ and $P(x > x_{m+1})$ are sufficiently small as to be ignored.



When subdividing, it is important to make sure that the distance from the centre of each partition $\bar{x}_i = \dfrac{x_{i+1} - x_i}{2}$, for all $i = 1, \ldots, m$, to any pole of the function g is larger than the length of the corresponding subinterval. This is because the distance between the centre point $\bar{x}_i$ and the pole of the function g is the radius of convergence r_i of the function around $\bar{x}_i$. Consequently, the smaller the length of the i^{th} subinterval in comparison to r_i, the faster the function g converges.

Now for every subinterval we compute the expectation around its centre, and accumulate all the results into the final approximation of the expectation.

Example 4.3.2. To demonstrate the partitioning approach in details, consider the function

$$g(x) = \frac{1}{1 + a^2 x^2}, \tag{4.49}$$

which corresponds to that of Example 4.3.1 for $a = 1$. See Figure 4.1.

The distribution for the input x is set to be normal with expectation $\mu = 0$ and variance $\sigma^2 = 1$. The poles for such function are $\mathring{x} = \pm i/a$, and the radius of convergence is the distance from the point about which the Taylor series is expanded to the nearest pole,

$$r = \sqrt{Re(\mathring{x} - c)^2 + Im(\mathring{x} - c)^2}. \tag{4.50}$$

When $c = 0$, the radius of convergence $r = \dfrac{1}{a}$.

Applying the partitioning approach, we first subdivide the x-space $[-10, 10]$ into subintervals, assuming that the probability outside that region is negligible. The size of the subintervals must be sufficiently small to guarantee the convergence of all

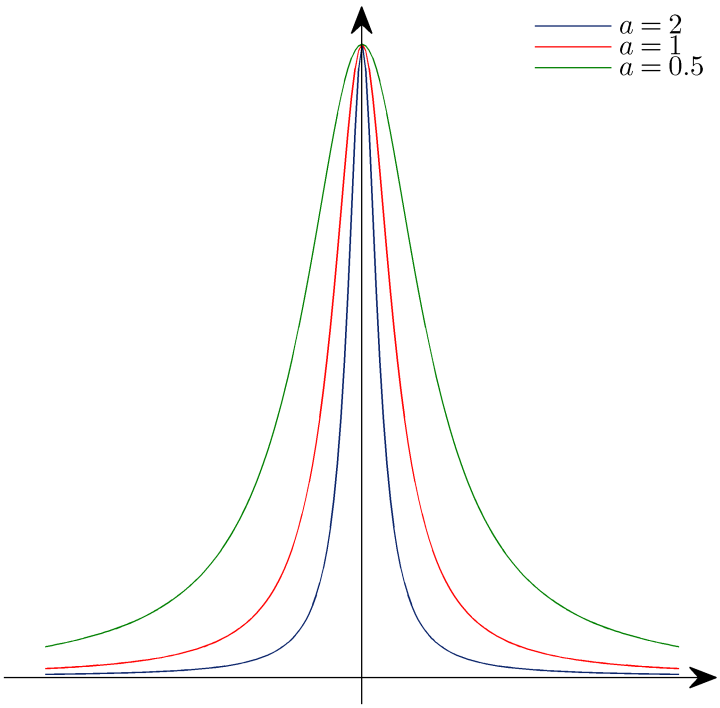


Figure 4.1: Graphical representation of the function $g(x) = \dfrac{1}{1+a^2x^2}$.

Taylor series used in the expectation operator.

For example, allow us to set $a = 1$, thus the radius of convergence about $x = 0$ is $r = 1$. When the number of equal partitions is $m = 5$, the set of centre points $\bar{x}_i$ is $\{-8, -4, 0, 4, 8\}$ and the length of the subintervals is $l = 4$. To obtain the convergence on every interval $[x_i, x_{i+1}]$, $i = 1, \ldots, 5$, the radius of convergence around $\bar{x}_i$, denoted as r_i, must create a circle of convergence containing the i^{th} subinterval completely. It means, the radius r_i must be not smaller than a half of the length l:

$$r_i \geqslant \frac{l}{2} = 2. \tag{4.51}$$

By Pythagoras' rule $r_i = \sqrt{Re(\mathring{x} - \bar{x}_i)^2 + Im(\mathring{x} - \bar{x}_i)^2}$, then since $Im(\bar{x}_i) = 0$,

$$r_i^2 = Re(\mathring{x} - \bar{x}_i)^2 + Im(\mathring{x})^2 \geqslant 4. \tag{4.52}$$

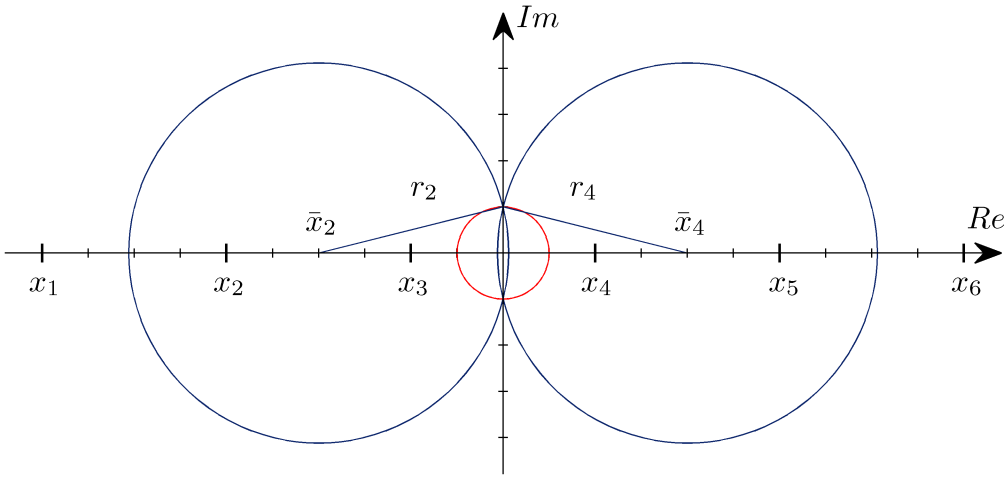


Figure 4.2: Partitioning approach for the function $g(x) = \frac{1}{1+x^2}$. For the intervals $[x_2, x_3]$ and $[x_4, x_5]$ the radius of convergence exceeds the interval half width. For the interval $[x_3, x_4]$ it does not.

In this particular case $Re(\mathring{x}) = 0$, $Im(\mathring{x}) = 1$, therefore $\bar{x}_i^2 \geqslant 3$:

$$\bar{x}_i \in (-\infty, -\sqrt{3}] \cup [\sqrt{3}, +\infty) \approx (-\infty, -1.7321] \cup [1.7321, +\infty), \tag{4.53}$$

which is true for $\bar{x}_i$ when $i = 1, 2, 4, 5$, but not $i = 3$. For the interval $[x_3, x_4]$, the radius $r_3 = 1 < \frac{l}{2} = 2$. See Figure 4.2.

On the other hand, if $m = 10$, $l = 2$, and $\{\bar{x}_i\} = \{-9, -7, \ldots, -1, 1, \ldots, 7, 9\}$, then

$$r_i = \sqrt{Re(\mathring{x} - \bar{x}_i)^2 + Im(\mathring{x})^2} \geqslant 1. \tag{4.54}$$

Hence the condition for the centre of i^{th} subinterval is $\bar{x}_i \in \mathbb{R}\backslash\{0\}$, which is valid for all $\bar{x}_i$, $i = 1, \ldots, 10$. See Figure 4.3.

In Tables 4.2 and 4.3 results for the function (4.49) when $a = 1$ are given. Table 4.2 is created for odd m. One can see that there is no convergence of the results for $m = 5$ and $m = 7$, but increasing the number of subintervals m sorts the problem out and improves computations. Already for $m = 9$ the expectation computed using partitioning approach starts to slowly converge to the expectation determined

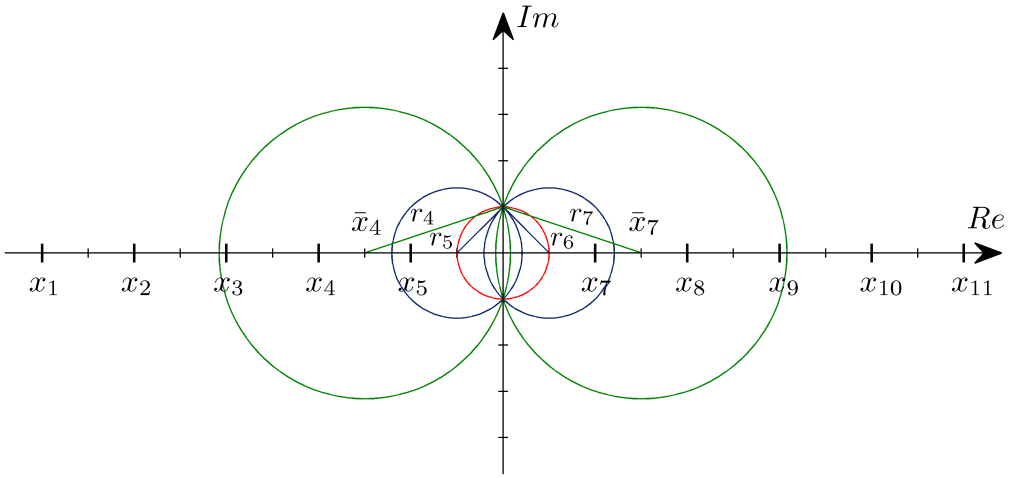


Figure 4.3: Partitioning approach for the function $g(x) = \frac{1}{1+x^2}$. The circles of convergence for intervals $[x_4, x_5]$, $[x_5, x_6]$, $[x_6, x_7]$, $[x_7, x_8]$ contain the respective intervals entirely. Analogously, the same is valid for the remaining intervals, i.e. $[x_1, x_2]$, $[x_2, x_3]$, $[x_3, x_4]$, $[x_8, x_9]$, $[x_9, x_{10}]$, and $[x_{10}, x_{11}]$.

by using quadrature. Results in Table 4.3 though, constructed for even m, are always convergent. In other words, for this particular example we can always observe the convergence for even m. However, for odd m to obtain convergent results, one must ensure that the length of partitions is less than double the radius of convergence at the point $\mu_x = 0$ for the function $g(x)$.

Similarly, generalising the convergence condition for any function $g(x)$, one must ensure that when subdividing the input variable space, the length of all partitions l_i is at most twice the distance from the centre point $\bar{x}_i$ to the imaginary pole $\mathring{x}$ to avoid the dependence on the function's pole location.

a	Quadrature	p	m	Partitioning approach	Error
1	0.655679542418802	4	5	1.574576910429300	9.1890e-001
			7	0.915851509969488	2.6017e-001
			9	0.733340863691872	7.7661e-002
			11	0.682121596314070	2.6442e-002
			13	0.666045646602571	1.0366e-002
			15	0.660295154363698	4.6156e-003
		6	5	-1.728559443582948	2.3842e+000
			7	0.279528048487026	3.7615e-001
			9	0.585316849121551	7.0363e-002
			11	0.639573355262032	1.6106e-002
			13	0.651306902163210	4.3726e-003
			15	0.654304468395577	1.3751e-003
		8	5	7.573160834964647	6.9175e+000
			7	1.244977734186994	5.8930e-001
			9	0.723992702236952	6.8313e-002
			11	0.666346854125417	1.0667e-002
			13	0.657794060055278	2.1145e-003
			15	0.656182552544450	5.0301e-004
		10	5	-20.855246952065400	2.1511e+001
			7	-0.316879359582553	9.7256e-001
			9	0.586402749475115	6.9277e-002
			11	0.648382951039756	7.2966e-003
			13	0.654636733510136	1.0428e-003
			15	0.655490192944368	1.8935e-004
		12	5	70.710291197465594	7.0055e+001
			7	2.318806159771613	1.6631e+000
			9	0.728196490486604	7.2517e-002
			11	0.660824232783849	5.1447e-003
			13	0.656207078137631	5.2754e-004
			15	0.655751417616606	7.1875e-005

Table 4.2: Comparison of the results for the function $g = \frac{1}{1+a^2x^2}$ for $a = 1$ on the interval $[-10, 10]$, $x \in N(0, 1)$, p is the order of Taylor series, m is the *odd* number of subintervals.

a	Quadrature	p	m	Partitioning approach	Error
1	0.655679542418802	4	4	0.668310940693024	1.2631e-002
			8	0.700203273677766	4.4524e-002
			16	0.654230866057646	1.4487e-003
			32	0.655654197292376	2.5345e-005
			64	0.655679515086225	2.7333e-008
			100	0.655679540610278	1.8085e-009
		6	4	0.755677244881720	9.9998e-002
			8	0.657679466340704	1.9999e-003
			16	0.655379806212782	2.9974e-004
			32	0.655682548158247	3.0057e-006
			64	0.655679542643579	2.2478e-010
			100	0.655679542420513	1.7104e-012
		8	4	0.748666564416940	9.2987e-002
			8	0.645773548303815	9.9060e-003
			16	0.655812462822065	1.3292e-004
			32	0.655679313961582	2.2846e-007
			64	0.655679542404234	1.4569e-011
			100	0.655679542418793	9.5479e-015
		10	4	0.699584919623698	4.3905e-002
			8	0.653115686016219	2.5639e-003
			16	0.655668013596968	1.1529e-005
			32	0.655679552092654	9.6739e-009
			64	0.655679542419680	8.7785e-013
			100	0.655679542418795	7.6605e-015
		12	4	0.653402873399573	2.2767e-003
			8	0.657785894068247	2.1064e-003
			16	0.655674408215680	5.1342e-006
			32	0.655679542530159	1.1136e-010
			64	0.655679542418759	4.3188e-014
			100	0.655679542418795	7.7716e-015

Table 4.3: Comparison of the results for the function $g = \frac{1}{1+a^2x^2}$ for $a = 1$ on the interval $[-10, 10]$, $x \in N(0, 1)$, p is the order of Taylor series, m is the *even* number of subintervals.

Algorithm 4.3.1. *Partitioning approach*

- Partition the distribution of the input into subintervals: $[x_1, x_2, \ldots, x_{m+1}]$, m is a number of subintervals, let $\bar{x}_i = \dfrac{x_{i+1} - x_i}{2}$ be the centres of the subintervals.
- For every subinterval $[x_i, x_{i+1}]$ compute the expectation $E_i(g)$ separately by constructing the Taylor approximation for function the g around $\bar{x}_i$.
- Sum up all $E_i(g)$:

$$E(g) = \sum_{i=1}^{m} E_i(g). \tag{4.55}$$

In other words, using the definition of the Taylor series (2.35) with the remainder (2.37) around the centre point $\bar{x}_i$

$$g(x) = \sum_{j=0}^{p} \frac{1}{j!} \frac{\partial^j g}{\partial x^j}(\bar{x}_i)(x - \bar{x}_i)^j + R_{p+1}, \tag{4.56}$$

the expectation of the function becomes

$$\begin{aligned} E(g) &= \int_{-\infty}^{+\infty} g(x) f(x) dx \\ &\cong \sum_{i=1}^{m} \int_{x_i}^{x_{i+1}} \sum_{j=0}^{p} \frac{1}{j!} \frac{\partial^j g}{\partial x^j}(\bar{x}_i)(x - \bar{x}_i)^j f(x) dx \\ &= \sum_{i=1}^{m} \sum_{j=0}^{p} \frac{1}{j!} \frac{\partial^j g}{\partial x^j}(\bar{x}_i) \int_{x_i}^{x_{i+1}} (x - \bar{x}_i)^j f(x) dx. \end{aligned} \tag{4.57}$$

Tables 4.2-4.7 show the results of comparing the partitioning method and numerical quadrature for the function (4.49) depending on the choice of parameter a, as well as the number of subintervals m and the order of the Taylor series approximation. The tables for odd m (i.e. Tables 4.2, 4.4, and 4.6) indicate the change in convergence when the chosen number m provides the sufficiently small subintervals.

To select the smallest odd m for which the result is convergent one can use the

relation (4.51), where $l = \dfrac{x_{m+1} - x_1}{m}$. Thus the following must hold

$$m \geqslant \frac{x_{m+1} - x_1}{2 \min_{i=1,\dots,m} r_i}. \tag{4.58}$$

Since the closest point from $\mathbb{R}$ to the imaginary pole $\mathring{x}$ is $x = Re(\mathring{x})$, the smallest r_i is

$$\min_{i=1,\dots,m} r_i = Im(\mathring{x}). \tag{4.59}$$

Therefore,

$$m \geqslant \frac{x_{m+1} - x_1}{2\, Im(\mathring{x})}. \tag{4.60}$$

Considering again the case with the function (4.49) on the interval $[-10, 10]$ when $a = \dfrac{1}{2}$, the poles are $\mathring{x} = \pm 2i$. Hence m must be not smaller than 5. Similarly, when $a = 2$, the number of subintervals $m \geqslant 20$. The corresponding tables confirm it. However, in Table 4.6 a slow convergence is observed even for $m = 19$. One can explain this fact by taking into account that the proof of convergence we present here does not prove the divergence as well. It means that the convergence condition provides accurate results. But if convergence condition does not hold, the convergence of the resuls still might be observed.

One can see that the smaller a is, the larger the radius of convergence of the function g is. Therefore, when $a \to 0$ the expectation of g as $a \to 0$ is

$$E\left[g(x)\right] = \int_{-\infty}^{+\infty} \lim_{a \to 0} \frac{1}{1 + a^2 x^2} N(0,1) dx = \int_{-\infty}^{+\infty} N(0,1) dx = 1 \tag{4.61}$$

due to the integral property of the normal distribution (2.21). The radius of convergence approaches infinity.

a	Quadrature	p	m	Partitioning approach	Error
0.5	0.842738458576110	4	3	0.930395279647364	8.7657e-002
			5	0.873501971537361	3.0764e-002
			7	0.850330253158174	7.5918e-003
			9	0.845016935722775	2.2785e-003
			11	0.843531559136205	7.9310e-004
			13	0.843015374911965	2.7692e-004
		6	3	0.727396332061593	1.1534e-001
			5	0.821941179407435	2.0797e-002
			7	0.840098709075598	2.6397e-003
			9	0.842308916407815	4.2954e-004
			11	0.842634226775850	1.0423e-004
			13	0.842704889381763	3.3569e-005
		8	3	1.028217895798460	1.8548e-001
			5	0.858250117758056	1.5512e-002
			7	0.843813031783956	1.0746e-003
			9	0.842840624919204	1.0217e-004
			11	0.842751885907989	1.3427e-005
			13	0.842741220872408	2.7623e-006
		10	3	0.502154798599100	3.4058e-001
			5	0.830481150937550	1.2257e-002
			7	0.842287687589311	4.5077e-004
			9	0.842710877356778	2.7581e-005
			11	0.842736034286216	2.4243e-006
			13	0.842738181693419	2.7688e-007
		12	3	1.396221437233889	5.5348e-001
			5	0.852835356310040	1.0097e-002
			7	0.842932306264022	1.9385e-004
			9	0.842745743474252	7.2849e-006
			11	0.842738927383430	4.6881e-007
			13	0.842738499071885	4.0496e-008

Table 4.4: Comparison of the results for the function $g = \frac{1}{1+a^2x^2}$ for $a = 0.5$ on the interval $[-10, 10]$, $x \in N(0,1)$, p is the order of Taylor series, m is the *odd* number of subintervals.

a	Quadrature	p	m	Partitioning approach	Error
0.5	0.842738458576110	4	4	0.927672822957566	8.4934e-002
			8	0.840831832970346	1.9066e-003
			16	0.842636077791186	1.0238e-004
			32	0.842737980799533	4.7778e-007
			64	0.842738451037514	7.5386e-009
			100	0.842738458055547	5.2056e-010
		6	4	0.847244014761711	4.5056e-003
			8	0.841738152582329	1.0003e-003
			16	0.842746392825047	7.9342e-006
			32	0.842738462422852	3.8467e-009
			64	0.842738458588409	1.2299e-011
			100	0.842738458576458	3.4761e-013
		8	4	0.823406287923583	1.9332e-002
			8	0.843035689566326	2.9723e-004
			16	0.842738037287821	4.2129e-007
			32	0.842738458498839	7.7271e-011
			64	0.842738458576093	1.6986e-014
			100	0.842738458576109	9.9920e-016
		10	4	0.837521080046281	5.2174e-003
			8	0.842729620219046	8.8384e-006
			16	0.842738463801623	5.2255e-009
			32	0.842738458579393	3.2833e-012
			64	0.842738458576110	3.3307e-016
			100	0.842738458576109	7.7716e-016
		12	4	0.846465018425702	3.7266e-003
			8	0.842723419268354	1.5039e-005
			16	0.842738460334232	1.7581e-009
			32	0.842738458575984	1.2590e-013
			64	0.842738458576110	5.5511e-016
			100	0.842738458576109	7.7716e-016

Table 4.5: Comparison of the results for the function $g = \frac{1}{1+a^2x^2}$ for $a = 0.5$ on the interval $[-10, 10]$, $x \in N(0,1)$, p is the order of Taylor series, m is the *even* number of subintervals.

a	Quadrature	p	m	Partitioning approach	Error
2	0.438182228226860	4	11	1.239257618408874	8.0108e-001
			13	0.775295093552342	3.3711e-001
			15	0.592741346077874	1.5456e-001
			17	0.514335571966556	7.6153e-002
			19	0.478092937347386	3.9911e-002
			21	0.460261731608494	2.2080e-002
		6	11	-1.482063338100518	1.9202e+000
			13	-0.148966981657659	5.8715e-001
			15	0.233766539902159	2.0442e-001
			17	0.359185014676319	7.8997e-002
			19	0.404940961029994	3.3241e-002
			21	0.423172608030217	1.5010e-002
		8	11	5.395692286199993	4.9575e+000
			13	1.532844840177975	1.0947e+000
			15	0.726276650096417	2.8809e-001
			17	0.525338525522329	8.7156e-002
			19	0.467704123139761	2.9522e-002
			21	0.449161301127540	1.0979e-002
		10	11	-12.983846962087647	1.3422e+001
			13	-1.696275163261215	2.1345e+000
			15	0.014446462867118	4.2374e-001
			17	0.338063726650696	1.0012e-001
			19	0.410970655724911	2.7212e-002
			21	0.429883664098910	8.2986e-003
		12	11	37.976720642833413	3.7539e+001
			13	4.730568019580059	4.2924e+000
			15	1.080243222567627	6.4206e-001
			17	0.556580566063321	1.1840e-001
			19	0.463994602597341	2.5812e-002
			21	0.444635865749501	6.4536e-003

Table 4.6: Comparison of the results for the function $g = \frac{1}{1+a^2x^2}$ for $a = 2$ on the interval $[-10, 10]$, $x \in N(0, 1)$, p is the order of Taylor series, m is the *odd* number of subintervals.

a	Quadrature	p	m	Partitioning approach	Error
2	0.438182228226860	4	4	0.252133481135809	1.8605e-001
			8	0.444510639679177	6.3284e-003
			16	0.458311796334639	2.0130e-002
			32	0.437273930907543	9.0830e-004
			64	0.438172468021303	9.7602e-006
			100	0.438182190481077	3.7746e-008
		6	4	0.346965565889904	9.1217e-002
			8	0.489915356554571	5.1733e-002
			16	0.437499886180888	6.8234e-004
			32	0.438112387096965	6.9841e-005
			64	0.438183569446338	1.3412e-006
			100	0.438182233183903	4.9570e-009
		8	4	0.421511187058821	1.6671e-002
			8	0.485732306153244	4.7550e-002
			16	0.433358363189148	4.8239e-003
			32	0.438242060033157	5.9832e-005
			64	0.438182113427429	1.1480e-007
			100	0.438182227777993	4.4887e-010
		10	4	0.473599030173376	3.5417e-002
			8	0.460339664078416	2.2157e-002
			16	0.437341392721561	8.4084e-004
			32	0.438173573280758	8.6549e-006
			64	0.438182234265902	6.0390e-009
			100	0.438182228254417	2.7556e-011
		12	4	0.502648087318382	6.4466e-002
			8	0.435794316934630	2.3879e-003
			16	0.439299794059562	1.1176e-003
			32	0.438180650624674	1.5776e-006
			64	0.438182228145466	8.1395e-011
			100	0.438182228224600	2.2607e-012

Table 4.7: Comparison of the results for the function $g = \frac{1}{1+a^2x^2}$ for $a = 2$ on the interval $[-10, 10]$, $x \in N(0, 1)$, p is the order of Taylor series, m is the *even* number of subintervals.

Example 4.3.3. Consider a more complicated case of a function with poles,

$$g(x) = \frac{1}{1 + bx + a^2x^2}, \tag{4.62}$$

where $a, b \in \mathbb{R}$. The poles are $\mathring{x} = \dfrac{-b \pm \sqrt{b^2 - 4a^2}}{2a^2}$, and the radius of convergence about the point $x = c$ is $r = \sqrt{Re(\mathring{x} - c)^2 + Im(\mathring{x} - c)^2}$.

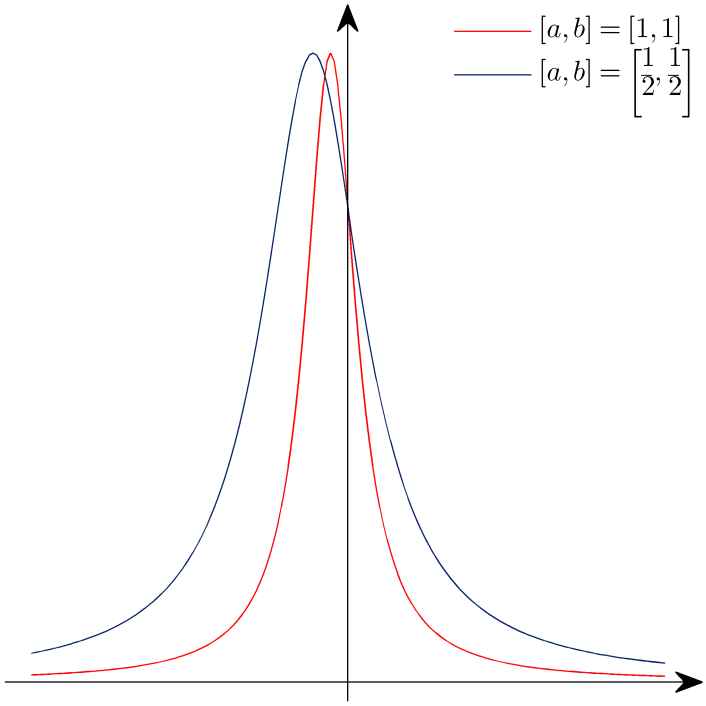


Figure 4.4: Graphical representation of the function $g(x) = \dfrac{1}{1 + bx + a^2x^2}$.

If $[a, b] = [1, 1]$, the poles are $\mathring{x} = -\dfrac{1}{2} \pm i\dfrac{\sqrt{3}}{2}$. Therefore, the radius of convergence about the origin is $r = 1$. When $[a, b] = \left[\dfrac{1}{2}, \dfrac{1}{2}\right]$, the poles are $\mathring{x} = -1 \pm i\sqrt{3}$, the radius of convergence about the origin is $r = 2$. For these two cases the function is plotted in Figure 4.4. The partitioning approach is demonstrated by Figure 4.5.

Tables 4.8 and 4.9 show the influence of the size of the partitioning for the interval $[-10, 10]$ on the result of computations. The assumption for m to be odd or even as in Example 4.3.2 is not applicable anymore, unless the function is considered on an interval symmetric around $Re\mathring{x}$, in which case for even m one can always observe the

convergence of the results. However, the previously obtained general condition (4.60) for choosing m to guarantee convergence always holds. Therefore, on the interval $[x_1, x_{m+1}] = [-10, 10]$ results for function (4.62) are expected to be convergent for $m \geqslant \dfrac{20}{\sqrt{3}} \approx 11.54 > 11$ when $[a, b] = [1, 1]$, and for $m \geqslant \dfrac{20}{2\sqrt{3}} \approx 5.77 > 5$ when $[a, b] = \left[\dfrac{1}{2}, \dfrac{1}{2}\right]$.

In Tables 4.8 and 4.9 for insufficient partitioning size divergence is observed: the results converge for $m \geqslant 12$ and $m \geqslant 6$. Though insignificant fluctuation might occur for higher Taylor order and/or larger number of subdivisions. These slight changes are likely due to accumulated machine errors.

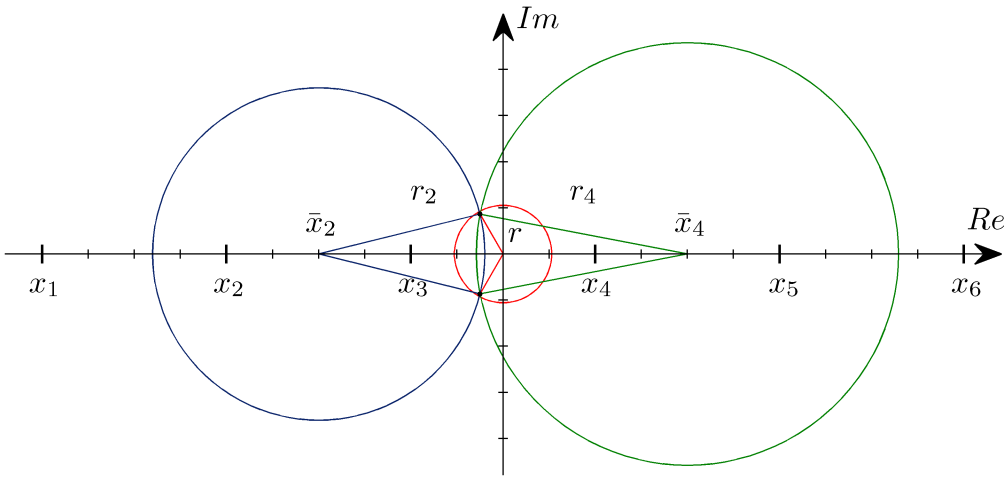


Figure 4.5: Partitioning approach for the function $g(x) = \frac{1}{1+x+x^2}$. On the interval $[x_3, x_4]$ function diverges since the radius of convergence $r_3 = r$ does not create the circle of convergence that would cover the corresponding interval.

On the other hand, the function (4.62) is symmetric (see Figure 4.4) around its maxima at $x = -\dfrac{a^2}{2b}$. Therefore, it is logical to consider the function on the interval symmetric around the maxima point as well. Hence the interval $[-10, 10]$ is transformed into $[-10 - 0.5, 10 - 0.5] = [-10.5, 9.5]$ for $[a, b] = [1, 1]$, and into $[-10 - 0.25, 10 - 0.25] = [-10.25, 9.75]$ for $[a, b] = [0.5, 0.5]$. Now one expects the convergent results for all even m, and for odd m only if $m > 11$ or $m > 5$, respectively. Tables 4.10-4.13 demonstrate the predicted behaviour.

$[a,b]$	Quadrature	p	m	Partitioning approach	Error
$[1,1]$	0.762826343373264	4	10	0.698211061281502	6.4615e-002
			11	0.730336267522099	3.2490e-002
			12	0.729602861197111	3.3223e-002
			13	0.752521270533596	1.0305e-002
			25	0.762350652674096	4.7569e-004
			50	0.762825992214978	3.5116e-007
		6	10	0.763623552319840	7.9721e-004
			11	0.771756377732378	8.9300e-003
			12	0.784381312732329	2.1555e-002
			13	0.764755250233345	1.9289e-003
			25	0.762880316839346	5.3973e-005
			50	0.762826379544399	3.6171e-008
		8	10	0.803541044538194	4.0715e-002
			11	0.771162239018585	8.3359e-003
			12	0.755990625794147	6.8357e-003
			13	0.764492770039113	1.6664e-003
			25	0.762823999344900	2.3440e-006
			50	0.762826340308498	3.0648e-009
		10	10	0.728396286104334	3.4430e-002
			11	0.753170532823711	9.6558e-003
			12	0.760460066334792	2.3663e-003
			13	0.761466498552278	1.3598e-003
			25	0.762825916450474	4.2692e-007
			50	0.762826343559841	1.8658e-010
		12	10	0.763763556764697	9.3721e-004
			11	0.765648429784355	2.8221e-003
			12	0.768431530728445	5.6052e-003
			13	0.763066901848099	2.4056e-004
			25	0.762826486802571	1.4343e-007
			50	0.762826343364356	8.9083e-012

Table 4.8: Comparison of the results for the function $g = \frac{1}{1+bx+a^2x^2}$ for $a = 1$, $b = 1$, on the interval $[-10,10]$, $x \in N(0,1)$, p is the order of Taylor series, m is the number of subintervals.

$[a,b]$	Quadrature	p	m	Partitioning approach	Error
$[0.5,0.5]$	0.935403999437389	4	4	0.954221358048837	1.8817e-002
			5	0.895604534833013	3.9799e-002
			6	0.864859451345101	7.0545e-002
			7	0.928118980590299	7.2850e-003
			25	0.935406802387182	2.8029e-006
			50	0.935404013467034	1.4030e-008
		6	4	0.681031580386082	2.5437e-001
			5	0.947139745575656	1.1736e-002
			6	0.974929717374682	3.9526e-002
			7	0.936257433918614	8.5343e-004
			25	0.935403752619203	2.4682e-007
			50	0.935403999350188	8.7201e-011
		8	4	1.006672665535302	7.1269e-002
			5	0.946961858945584	1.1558e-002
			6	0.926892454033017	8.5115e-003
			7	0.936240980608903	8.3698e-004
			25	0.935404016531876	1.7094e-008
			50	0.935403999437682	2.9265e-013
		10	4	1.163408584116051	2.2800e-001
			5	0.919157829076105	1.6246e-002
			6	0.927080618603155	8.3234e-003
			7	0.934839392919981	5.6461e-004
			25	0.935403998524944	9.1245e-010
			50	0.935403999437386	2.6645e-015
		12	4	0.823664564832794	1.1174e-001
			5	0.941515532937906	6.1115e-003
			6	0.947959137937857	1.2555e-002
			7	0.935488382462601	8.4383e-005
			25	0.935403999476326	3.8937e-011
			50	0.935403999437388	1.4433e-015

Table 4.9: Comparison of the results for the function $g = \frac{1}{1+bx+a^2x^2}$ for $a = 0.5$, $b = 0.5$, on the interval $[-10,10]$, $x \in N(0,1)$, p is the order of Taylor series, m is the number of subintervals.

$[a,b]$	Quadrature	p	m	Partitioning approach	Error
$[1,1]$	0.762826343373264	4	8	0.827962509668094	6.5136e-002
			10	0.795105621158395	3.2279e-002
			12	0.773560716128379	1.0734e-002
			14	0.764290888518840	1.4645e-003
			24	0.762220658944241	6.0568e-004
			48	0.762825123462876	1.2199e-006
		6	8	0.782101271063672	1.9275e-002
			10	0.758913943921443	3.9124e-003
			12	0.757234966982609	5.5914e-003
			14	0.759834148971611	2.9922e-003
			24	0.762876325421314	4.9982e-005
			48	0.762826497297243	1.5392e-007
		8	8	0.751186586283063	1.1640e-002
			10	0.755819075522386	7.0073e-003
			12	0.761511152502485	1.3152e-003
			14	0.763071122818182	2.4478e-004
			24	0.762831094510658	4.7511e-006
			48	0.762826329990353	1.3383e-008
		10	8	0.752426839925630	1.0400e-002
			10	0.762830834757915	4.4914e-006
			12	0.763759820744462	9.3348e-004
			14	0.763128157889076	3.0181e-004
			24	0.762824222215708	2.1212e-006
			48	0.762826344174237	8.0097e-010
		12	8	0.762579937390889	2.4641e-004
			10	0.764483524340984	1.6572e-003
			12	0.763010595833514	1.8425e-004
			14	0.762755827864436	7.0516e-005
			24	0.762826650294834	3.0692e-007
			48	0.762826343339104	3.4160e-011

Table 4.10: Comparison of the results for the function $g = \frac{1}{1+bx+a^2x^2}$ for $a = 1$, $b = 1$, on the interval $[-10.5, 9.5]$, $x \in N(0,1)$, p is the order of Taylor series, m is the *even* number of subintervals.

$[a,b]$	Quadrature	p	m	Partitioning approach	Error
$[1,1]$	0.762826343373264	4	7	1.485203506894798	7.2238e-001
			9	0.971994528548443	2.0917e-001
			11	0.832968521769950	7.0142e-002
			13	0.789787002591585	2.6961e-002
			15	0.774531180436274	1.1705e-002
			17	0.768455837104585	5.6295e-003
		6	7	-0.638202390836922	1.4010e+000
			9	0.509552991276576	2.5327e-001
			11	0.705168630194878	5.7658e-002
			13	0.747071355190383	1.5755e-002
			15	0.757851715835312	4.9746e-003
			17	0.761051164894892	1.7752e-003
		8	7	3.704047155574289	2.9412e+000
			9	1.090770708480060	3.2794e-001
			11	0.813460088444060	5.0634e-002
			13	0.772870547169571	1.0044e-002
			15	0.765242346218336	2.4160e-003
			17	0.763500378442864	6.7404e-004
		10	7	-5.733954157612054	6.4968e+000
			9	0.318771114331293	4.4406e-001
			11	0.716609776954344	4.6217e-002
			13	0.756235707752046	6.5906e-003
			15	0.761629261844706	1.1971e-003
			17	0.762563166713559	2.6318e-004
		12	7	15.619207765699979	1.4856e+001
			9	1.383148177907660	6.2032e-001
			11	0.806287144570025	4.3461e-002
			13	0.767278314140032	4.4520e-003
			15	0.763434216990932	6.0787e-004
			17	0.762930287447307	1.0394e-004

Table 4.11: Comparison of the results for the function $g = \frac{1}{1+bx+a^2x^2}$ for $a = 1$, $b = 1$, on the interval $[-10.5, 9.5]$, $x \in N(0,1)$, p is the order of Taylor series, m is the *odd* number of subintervals.

$[a,b]$	Quadrature	p	m	Partitioning approach	Error
$[0.5,0.5]$	0.935403999437389	4	2	1.069344546267774	1.3394e-001
			4	1.042269083111542	1.0687e-001
			6	0.895354955299605	4.0049e-002
			8	0.924605395595643	1.0799e-002
			24	0.935406102533304	2.1031e-006
			48	0.935404017322437	1.7885e-008
		6	2	1.327726139384357	3.9232e-001
			4	0.815966562636295	1.1944e-001
			6	0.938321286721154	2.9173e-003
			8	0.940919180949776	5.5152e-003
			24	0.935403904039433	9.5398e-008
			48	0.935403999315657	1.2173e-010
		8	2	1.323044099732943	3.8764e-001
			4	0.873678739328930	6.1725e-002
			6	0.949169808439427	1.3766e-002
			8	0.933568258933700	1.8357e-003
			24	0.935404001396319	1.9589e-009
			48	0.935403999437920	5.3046e-013
		10	2	1.167649560846725	2.3225e-001
			4	1.030996927940963	9.5593e-002
			6	0.924968924352961	1.0435e-002
			8	0.935758844937587	3.5485e-004
			24	0.935403999479191	4.1802e-011
			48	0.935403999437382	7.3275e-015
		12	2	0.987359504420506	5.1956e-002
			4	0.989834673535083	5.4431e-002
			6	0.937454800335328	2.0508e-003
			8	0.935449196498683	4.5197e-005
			24	0.935403999432914	4.4752e-012
			48	0.935403999437389	3.3307e-016

Table 4.12: Comparison of the results for the function $g = \frac{1}{1+bx+a^2x^2}$ for $a = 0.5$, $b = 0.5$, on the interval $[-10.25, 9.75]$, $x \in N(0,1)$, p is the order of Taylor series, m is the *even* number of subintervals.

$[a,b]$	Quadrature	p	m	Partitioning approach	Error
$[0.5,0.5]$	0.935403999437389	4	3	0.875912849935714	5.9491e-002
			5	0.910539481055221	2.4865e-002
			7	0.929429591115569	5.9744e-003
			9	0.934069172323019	1.3348e-003
			11	0.934715878551792	6.8812e-004
			13	0.934928344125073	4.7566e-004
		6	3	1.157293141199919	2.2189e-001
			5	0.974536440839733	3.9132e-002
			7	0.940477508422501	5.0735e-003
			9	0.936071679991049	6.6768e-004
			11	0.935501692047711	9.7693e-005
			13	0.935453734413980	4.9735e-005
		8	3	0.523996297142851	4.1141e-001
			5	0.905652751886234	2.9751e-002
			7	0.933521061025043	1.8829e-003
			9	0.935225581348432	1.7842e-004
			11	0.935395983151581	8.0163e-006
			13	0.935404018209513	1.8772e-008
		10	3	1.319266029531639	3.8386e-001
			5	0.942832823073624	7.4288e-003
			7	0.935537709674896	1.3371e-004
			9	0.935413869136310	9.8697e-006
			11	0.935403793397566	2.0604e-007
			13	0.935403119330794	8.8011e-007
		12	3	1.706924328635746	7.7152e-001
			5	0.950533229152008	1.5129e-002
			7	0.935734553039233	3.3055e-004
			9	0.935416989883433	1.2990e-005
			11	0.935404465928422	4.6649e-007
			13	0.935404161780014	1.6234e-007

Table 4.13: Comparison of the results for the function $g = \dfrac{1}{1+bx+a^2x^2}$ for $a = 0.5$, $b = 0.5$, on the interval $[-10.25, 9.75]$, $x \in N(0,1)$, p is the order of Taylor series, m is the *odd* number of subintervals.

Example 4.3.4. Now consider a function with several pairs of poles. Take

$$g(x) = \frac{1}{1 + dx + c^2x^2 + b^3x^3 + a^4x^4}. \tag{4.63}$$

with coefficients $[a^4, b^3, c^2, d] = \left[\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, -\frac{1}{2}\right]$. We get two sets of poles $\mathring{x}_1 = \frac{1}{2} \pm i\frac{\sqrt{3}}{2}$ and $\mathring{x}_2 = -1 \pm i\sqrt{3}$. As we approximate the function with Taylor series about $x = 0$, the distance between x and the poles is $\mathring{r}_1 = 1$ and $\mathring{r}_2 = 2$, respectively. Thus the radius of convergence for the function (4.63) is

$$r = \min_{i=1,2}(\mathring{r}_i) = 1.$$

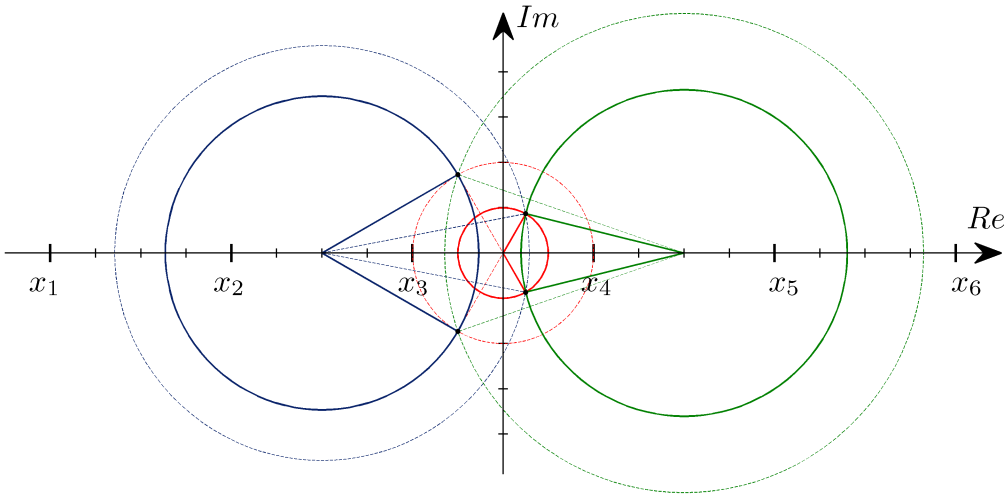


Figure 4.6: Partitioning approach. Illustration of the improved radius of convergence for the function g.

Analogously to the examples 4.3.2 and 4.3.3, we use the condition (4.60) for selecting the number of partitions m. However this time we are dealing with two sets of poles. Hence, to guarantee the convergence the choice of m must depend on both of them:

$$m \geqslant \max_{i=1,2} \frac{x_{m+1} - x_1}{2\, Im(\mathring{x}_i)}. \tag{4.64}$$

For the considered values of parameters, $m \geqslant \max_{i=1,2} m_i > \max\{3, 6\} = 6$. Table 4.14

shows the influence of the decision made for m: the results are convergent for all $m \geqslant 7$.

$[a^4, b^3, c^2, d]$	Quadrature	p	m	Partitioning approach	Error
$\left[\frac{1}{4}, \frac{1}{4}, \frac{3}{4}, -\frac{1}{2}\right]$	0.6659675265755	4	4	0.8298049209129	1.6384e-001
			5	0.5636764390915	1.0229e-001
			6	0.6393657765981	2.6602e-002
			7	0.6491838674496	1.6784e-002
			8	0.6509597778278	1.5008e-002
			9	0.6634528205650	2.5147e-003
		6	4	0.6489171748634	1.7050e-002
			5	0.7740029907115	1.0804e-001
			6	0.6440333080138	2.1934e-002
			7	0.6746093693735	8.6418e-003
			8	0.6748551329045	8.8876e-003
			9	0.6668814504819	9.1392e-004
		8	4	0.4986578240164	1.6731e-001
			5	0.6602788632495	5.6887e-003
			6	0.7021951513609	3.6228e-002
			7	0.6668503593796	8.8283e-004
			8	0.6622649933331	3.7025e-003
			9	0.6660067140612	3.9187e-005
		10	4	0.6452946503425	2.0673e-002
			5	0.5294102222152	1.3656e-001
			6	0.6502785721216	1.5689e-002
			7	0.6624579747187	3.5096e-003
			8	0.6667333178217	7.6579e-004
			9	0.6657947308223	1.7280e-004
		12	4	0.8617786172175	1.9581e-001
			5	0.8432932130027	1.7733e-001
			6	0.6536576261412	1.2310e-002
			7	0.6678395704270	1.8720e-003

Continued on the next page...

...continued from the previous page

$[a^4, b^3, c^2, d]$	Quadrature	p	m	Partitioning approach	Error
			8	0.6664536507282	4.8612e-004
			9	0.6660247374852	5.7211e-005

Table 4.14: Comparison of the results for the function $g = \frac{1}{1+dx+c^2x^2+b^3x^3+a^4x^4}$ for $a^4 = \frac{1}{4}$, $b^3 = \frac{1}{4}$, $c^2 = \frac{3}{4}$, $d = -\frac{1}{2}$, on the interval $[-6,6]$, p is the order of Taylor series, m is the number of subintervals.

An option of adjusting the interval $[x_1, x_{m+1}]$ to be able to assume the convergence for all even m for examples with more than one set of imaginary poles is not applicable any longer.

4.3.2 Gaussian function

Some might not be completely convinced that the divergent results for the function (4.49) are caused by its poles and their positioning about the radius of convergence recreated around the centre point of every partition. Does the function shape have any influence on the choice of the number of partitions required to get accurate results? Is the "peakedness" of the function the main cause of convergence inconsistencies when the larger number of partitions required to approximate its sharp segments is used?

To answer these questions we consider here another "bell curved" function, this time with no poles - the Gaussian function itself.

The function of the form

$$g(x) = ae^{-\frac{(x-b)^2}{2c^2}}, \tag{4.65}$$

where a, b, $c \in \mathbb{R}_+$,is called a **Gaussian function**. In statistic, this function is known as the probability density function for the normal distribution when $a = \frac{1}{\sigma\sqrt{2\pi}}$, $b = \mu$, and $c = \sigma$.

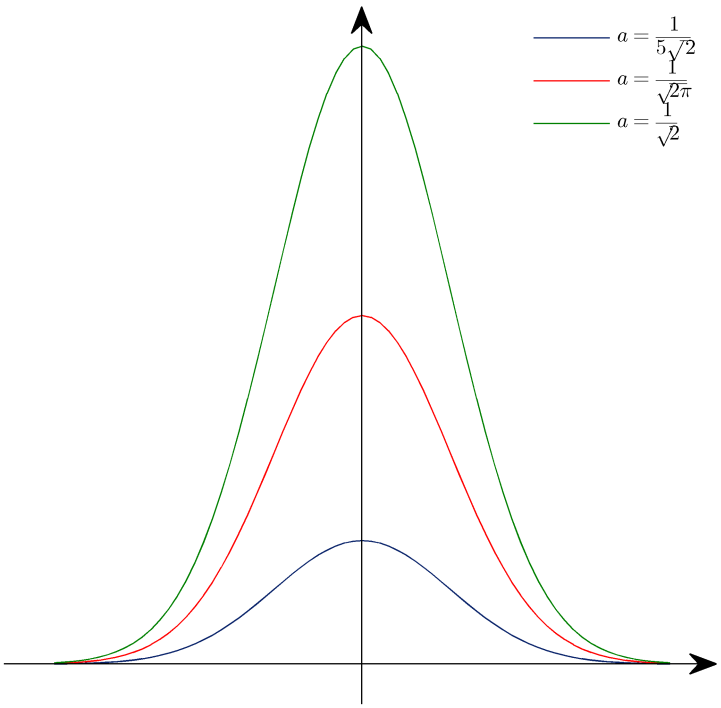


Figure 4.7: Gaussian function "bell curves" with variation of the parameter a.

The integral of the Gaussian function on the real plane $\mathbb{R}$ is called the **Gaussian integral** and can be evaluated exactly using a polar coordinate transformation, even though the indefinite integral $\int ae^{-\frac{(x-b)^2}{2c^2}}dx$ does not have elementary functions representation.

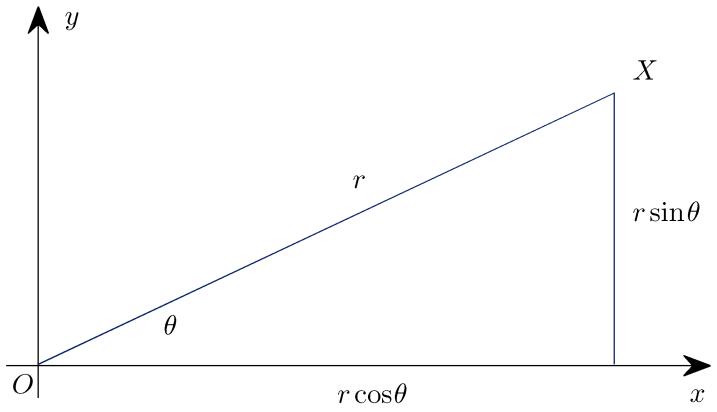
Theorem 4.3.1. *The Gaussian integral can be evaluated analytically and*

$$I = \int_{-\infty}^{+\infty} ae^{-\frac{(x-b)^2}{2c^2}}dx = ac\sqrt{2\pi}. \tag{4.66}$$

Proof: In polar coordinates the position of any point X in the space is defined by the pair $[r, \theta]$. Here, r is the distance from the **pole** O, an origin of the polar system, to the point X. r is known as the **radius**. And θ is the angle that OX makes with the **polar axis**, a ray extending from O horizontally to the right.

Take the integral

$$I = \int_{-\infty}^{+\infty} ae^{-\frac{(x-b)^2}{2c^2}}dx, \tag{4.67}$$



and square it to give

$$\begin{aligned} I^2 &= \int_{-\infty}^{+\infty} a e^{-\frac{(x-b)^2}{2c^2}} dx \times \int_{-\infty}^{+\infty} a e^{-\frac{(y-b)^2}{2c^2}} dy \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} a^2 e^{-\frac{1}{2c^2}\left((x-b)^2+(y-b)^2\right)} dxdy \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} a^2 e^{-\frac{1}{2c^2}\left(\tilde{x}^2+\tilde{y}^2\right)} d\tilde{x}d\tilde{y}, \end{aligned} \tag{4.68}$$

when $\tilde{x} = x-b$, $\tilde{y} = y-b$. We transform to polar coordinates, $\tilde{x} = r\cos\theta$, $\tilde{y} = r\sin\theta$, thus $\tilde{x}^2+\tilde{y}^2 = r^2$ on the plane $\mathbb{R}$, and $dxdy = rdrd\theta$. The radius r is always positive, and the angle θ is changing from 0 to 2π. Therefore,

$$\begin{aligned} I^2 &= a^2 \int_0^{2\pi} \int_0^{+\infty} e^{-\frac{1}{2c^2}r^2} r dr d\theta \\ &= a^2 \left(\theta \int_0^{+\infty} e^{-\frac{1}{2c^2}r^2} r dr \right)_0^{2\pi} = 2\pi a^2 \int_0^{+\infty} e^{-\frac{1}{2c^2}r^2} r dr \\ &= 2\pi a^2 \int_0^{+\infty} -c^2 e^{-\frac{1}{2c^2}r^2} d\left(-\frac{1}{2c^2}r^2\right) = -2\pi a^2 c^2 \left(e^{-\frac{1}{2c^2}r^2}\right)_0^{+\infty} \\ &\approx -2\pi a^2 c^2 (0-1) = 2\pi a^2 c^2. \end{aligned} \tag{4.69}$$

As a result, we get that $I = ac\sqrt{2\pi}$. □

Since the product of several Gaussian functions is still a Gaussian function, we can compute the expectation of (4.65) analytically. Assuming normal distribution $N(0,1)$

and coefficients for the function $b = 0$, $c = 1$, when varying a, the expectation μ_g is

$$\mu_g = E(g) = a\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{+\infty} e^{-x^2}dx. \tag{4.70}$$

It is again the Gaussian integral with coefficients $[\tilde{a}, \tilde{b}, \tilde{c}] = \left[\frac{a}{\sqrt{2\pi}}, 0, \frac{1}{\sqrt{2}}\right]$. Hence,

$$\mu_g = \frac{a}{\sqrt{2\pi}}\frac{1}{\sqrt{2}}\sqrt{2\pi} = \frac{a}{\sqrt{2}}. \tag{4.71}$$

We now can compare the exact solution with the results produced by the partitioning approach. Tables 4.15, 4.16, and 4.17 illustrate the convergence of the partitioning approach with $a = \frac{1}{\sqrt{2}}$, $\frac{1}{5\sqrt{2}}$, and $\frac{1}{\sqrt{2\pi}}$, respectively.

It certainly is important to choose a large enough number of partitions to embrace all more or less abrupt changes in behaviour of a function to get accurate results. However, insufficient partitioning does not cause the divergence of the computations as we observed for the function with poles.

$[a,b,c]$	Analytical solution	p	m	Partitioning approach	Error
$\left[\frac{1}{\sqrt{2}},0,1\right]$	$\frac{1}{2}=0.5000$	4	5	0.504704314460094	4.7043e-003
			10	0.499939634379945	6.0366e-005
			20	0.499999342477803	6.5752e-007
			40	0.499999989486882	1.0513e-008
			80	0.499999999834786	1.6521e-010
		6	5	0.499503149265141	4.9685e-004
			10	0.500002491554100	2.4916e-006
			20	0.500000003972523	3.9725e-009
			40	0.500000000016043	1.6043e-011
			80	0.500000000000063	6.3116e-014
		8	5	0.500030758876817	3.0759e-005
			10	0.499999903487286	9.6513e-008
			20	0.499999999981460	1.8540e-011
			40	0.499999999999981	1.9151e-014
			80	0.500000000000000	1.6653e-016
		10	5	0.499998599219173	1.4008e-006
			10	0.500000003259818	3.2598e-009
			20	0.500000000000071	7.0777e-014
			40	0.500000000000000	2.2204e-016
			80	0.500000000000000	1.6653e-016
		12	5	0.500000139345288	1.3935e-007
			10	0.499999999907990	9.2010e-011
			20	0.500000000000000	3.3307e-016
			40	0.500000000000000	2.2204e-016
			80	0.500000000000000	1.6653e-016

Table 4.15: Comparison of the results for the function $g(x)=ae^{-\frac{(x-b)^2}{2c^2}}$ on the interval $[-6,6]$, p is the order of Taylor series, m - the number of subintervals.

$[a,b,c]$	Analytical solution	p	m	Partitioning approach	Error
$\left[\frac{1}{5\sqrt{2}},0,1\right]$	$\frac{1}{10}=0.1000$	4	5	0.100940862892019	9.4086e-004
			10	0.099987926875989	1.2073e-005
			20	0.099999868495561	1.3150e-007
			40	0.099999997897377	2.1026e-009
			80	0.099999999966957	3.3043e-011
		6	5	0.099900629853028	9.9370e-005
			10	0.100000498310820	4.9831e-007
			20	0.100000000794505	7.9450e-010
			40	0.100000000003209	3.2088e-012
			80	0.100000000000013	1.2546e-014
		8	5	0.100006151775363	6.1518e-006
			10	0.099999980697457	1.9303e-008
			20	0.099999999996292	3.7079e-012
			40	0.099999999999996	3.8025e-015
			80	0.100000000000000	9.7145e-017
		10	5	0.099999719843835	2.8016e-007
			10	0.100000000651964	6.5196e-010
			20	0.100000000000014	1.4169e-014
			40	0.100000000000000	2.7756e-017
			80	0.100000000000000	1.1102e-016
		12	5	0.100000027869057	2.7869e-008
			10	0.099999999981598	1.8402e-011
			20	0.100000000000000	4.1633e-017
			40	0.100000000000000	2.7756e-017
			80	0.100000000000000	1.1102e-016

Table 4.16: Comparison of the results for the function $g(x)=ae^{-\frac{(x-b)^2}{2c^2}}$ on the interval $[-6,6]$, p is the order of Taylor series, m - the number of subintervals.

$[a,b,c]$	Analytical solution	p	m	Partitioning approach	Error
$\left[\frac{1}{\sqrt{2\pi}},0,1\right]$	0.2820947917739	4	5	0.2847489169900	2.6541e-003
			10	0.2820607341198	3.4058e-005
			20	0.2820944208067	3.7097e-007
			40	0.2820947858425	5.9314e-009
			80	0.2820947916807	9.3212e-011
		6	5	0.2818144737647	2.8032e-004
			10	0.2820961974827	1.4057e-006
			20	0.2820947940151	2.2413e-009
			40	0.2820947917829	9.0516e-012
			80	0.2820947917739	3.5416e-014
		8	5	0.2821121456118	1.7354e-005
			10	0.2820947373224	5.4451e-008
			20	0.2820947917634	1.0460e-011
			40	0.2820947917739	1.0658e-014
			80	0.2820947917739	1.1102e-016
		10	5	0.2820940014679	7.9031e-007
			10	0.2820947936130	1.8392e-009
			20	0.2820947917739	3.9913e-014
			40	0.2820947917739	0.0000e+000
			80	0.2820947917739	1.6653e-016
		12	5	0.2820948703910	7.8617e-008
			10	0.2820947917220	5.1911e-011
			20	0.2820947917739	1.6653e-016
			40	0.2820947917739	0.0000e+000
			80	0.2820947917739	1.6653e-016

Table 4.17: Comparison of the results for the function $g(x) = ae^{-\frac{(x-b)^2}{2c^2}}$ on the interval $[-6,6]$, p is the order of Taylor series, m - the number of subintervals.

4.3.3 Error estimation

For the error analysis of the partitioning approach we require the $(p+1)^{th}$ Taylor term

$$R_i = \int_{x_i}^{x_{i+1}} \frac{1}{(p+1)!} g^{(p+1)}(\xi_i)(x-\bar{x}_i)^{p+1} f(x)dx, \tag{4.72}$$

where $x_i \leq \xi_i \leq x_{i+1}$. Assuming x is normally distributed with $\mu_x = 0$ and $\sigma_x^2 = 1$, then

$$f(x) = N(0,1) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \tag{4.73}$$

Therefore, on each interval

$$\begin{aligned} R_i &= \frac{g^{(p+1)}(\xi_i)}{(p+1)!} \int_{x_i}^{x_{i+1}} (x-\bar{x}_i)^{p+1} f(x)dx \\ &= \frac{1}{\sqrt{2\pi}} \frac{g^{(p+1)}(\xi_i)}{(p+1)!} \int_{x_i}^{x_{i+1}} (x-\bar{x}_i)^{p+1} e^{-x^2/2} dx. \end{aligned} \tag{4.74}$$

The Taylor series for the exponential function in (4.74) at the interval centre point $\bar{x}_i$ is

$$e^{-x^2/2} = e^{-\bar{x}_i^2/2} - \bar{x}_i e^{-\bar{x}_i^2/2}(x-\bar{x}_i) + \ldots. \tag{4.75}$$

Because of that, (4.74) becomes

$$\begin{aligned} R_i &= \frac{e^{-\bar{x}_i^2/2}}{\sqrt{2\pi}} \frac{g^{(p+1)}(\xi_i)}{(p+1)!} \int_{x_i}^{x_{i+1}} \big(1 - \bar{x}_i(x-\bar{x}_i) + \ldots\big)(x-\bar{x}_i)^{p+1} dx \\ &\approx \frac{e^{-\bar{x}_i^2/2}}{\sqrt{2\pi}} \frac{g^{(p+1)}(\xi_i)}{(p+1)!} \int_{x_i}^{x_{i+1}} (x-\bar{x}_i)^{p+1} dx \\ &- \frac{e^{-\bar{x}_i^2/2}}{\sqrt{2\pi}} \frac{g^{(p+1)}(\xi_i)}{(p+1)!} \bar{x}_i \int_{x_i}^{x_{i+1}} (x-\bar{x}_i)^{p+2} dx. \end{aligned} \tag{4.76}$$

Let $C_i = \dfrac{e^{-\bar{x}_i^2/2}}{\sqrt{2\pi}} \dfrac{g^{(p+1)}(\xi_i)}{(p+1)!}$ and $h_i = x_{i+1} - x_i$, then

$$R_i \approx C_i \left[\frac{(x-\bar{x}_i)^{p+2}}{p+2} \right]_{x_i}^{x_{i+1}} - C_i \bar{x}_i \left[\frac{(x-\bar{x}_i)^{p+3}}{p+3} \right]_{x_i}^{x_{i+1}}$$

$$= \frac{C_i}{p+2}\left(\left(\frac{h_i}{2}\right)^{p+2} - \left(-\frac{h_i}{2}\right)^{p+2}\right) - \frac{C_i\bar{x}_i}{p+3}\left(\left(\frac{h_i}{2}\right)^{p+3} - \left(-\frac{h_i}{2}\right)^{p+3}\right). \quad (4.77)$$

That is equivalent to

$$R_i \approx \begin{cases} \frac{2}{p+2}C_i\left(\frac{h_i}{2}\right)^{p+2}, & \text{if } p \text{ is odd;} \\ -\frac{2}{p+3}C_i\bar{x}_i\left(\frac{h_i}{2}\right)^{p+3}, & \text{otherwise.} \end{cases} \quad (4.78)$$

Taking into account that $R = \sum_{i=1}^{m} R_i$, we consider cases for odd and even order of Taylor series separately.

For odd p

$$|R| \le \sum_{i=1}^{m}|R_i| = \frac{2}{p+2}\sum_{i=1}^{m}\left(\frac{h_i}{2}\right)^{p+2}|C_i|. \quad (4.79)$$

Assuming $h = \max\limits_{1\le i\le m} h_i \approx \frac{x_{m+1}-x_1}{m}$, then $m \approx \frac{x_{m+1}-x_1}{h}$. As a result,

$$|R| \le \frac{2}{p+2}\left(\frac{h}{2}\right)^{p+2}\frac{x_{m+1}-x_1}{h}\max_{1\le i\le m}|C_i| = \frac{1}{p+2}\left(\frac{h}{2}\right)^{p+1}(x_{m+1}-x_1)\max_{1\le i\le m}|C_i| = O(h^{p+1}). \quad (4.80)$$

Similarly, when p is even,

$$|R| \le \frac{2}{p+3}\sum_{i=1}^{m}\left(\frac{h_i}{2}\right)^{p+3}|C_i|\bar{x}_i \le \frac{2}{p+3}\left(\frac{h}{2}\right)^{p+3}\left(\frac{x_{m+1}-x_1}{h}\right)\left(\frac{x_{m+1}-x_1}{2}\right)\max_{1\le i\le m}|C_i| = \frac{1}{2(p+3)}\left(\frac{h}{2}\right)^{p+2}(x_{m+1}-x_1)^2\max_{1\le i\le m}|C_i| = O(h^{p+2}). \quad (4.81)$$

Since $h \propto \frac{1}{m}$, the resulting relation for the remaining term R is

$$R = \begin{cases} O\left(\left(\frac{1}{m}\right)^{p+1}\right), & \text{if } p \text{ is odd;} \\ O\left(\left(\frac{1}{m}\right)^{p+2}\right), & \text{otherwise.} \end{cases} \tag{4.82}$$

4.4 Convergence test

The demonstration of the order of convergence summarised by (4.82) can be performed on the results for the function (4.47). To do so, we plot the number of subdivisions m against the error in computations using a logarithmic scale.

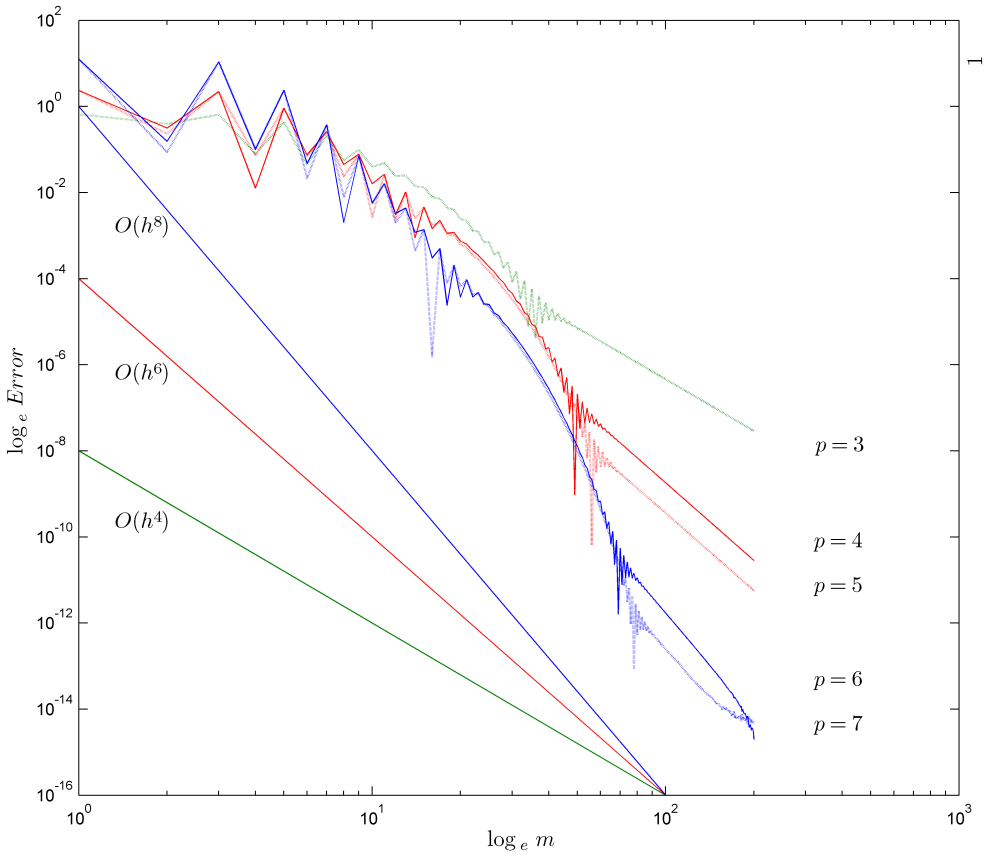


Figure 4.8: Error estimation for the function $g(x) = \frac{1}{1+x^2}$.

The estimated error on Figure 4.8 is compared to the corresponding so-called

convergence lines. One can see that the results for computations using Taylor order $p = 3$ are convergent as $O(h^4)$. Analogously, for $p = 4$ and $p = 5$ the convergence order is $O(h^6)$, and for $p = 6$, $p = 7$ the order of the error is $O(h^8)$, that corresponds to the theory derived before in Section 4.3.3. The order of convergence estimated by (4.82)is only true for large m, therefore the nonlinear behaviour on Figure 4.8 can be explained by insufficient choice of m for which the error estimation formula can not be applied.

4.5 Uncertainty propagation test cases

As an initial check we verify the results of Section 4.1 using previously published test cases, like the one in the paper by Ghate and Giles [4]. By repeating their experiments we can demonstrate the difference between the results.

Example 4.5.1. For the given output function $y = \cos x$ we assess the performance of the moments method for the scalar normally distributed input with $\mu_x = 0$ comparing it to the Monte Carlo simulation results. The values of the standard deviation σ_x are increasing from 0 to $\pi/8$. A sample size for each Monte Carlo simulation is taken to be 100,000.

The second order Taylor expansion provides second order of accuracy for the moments method for the expectation, but only first order of accuracy for the variance.

The moments method for the expectation (4.6) completely agrees with the result of Ghate's paper [4]. Assuming that $g(x) = \cos x$ and $\mu_x = 0$, we get

$$\mu_g = \cos(0) - \frac{1}{2}\cos(0)\sigma_x^2 = 1 - \frac{1}{2}\sigma_x^2.$$

To obtain second order convergence for the variance, the third order Taylor approximation is required. The second order moments method for the variance, i.e. the moments method obtained from the second order Taylor approximation, according

to [4], is

$$\sigma_g^2 = \left(\frac{\partial g}{\partial x}\sigma_x\right)^2 + \frac{\partial^2 g}{\partial x^2}\frac{\partial g}{\partial x}S(x)\sigma_x^3 + \frac{1}{4}\left(\frac{\partial^2 g}{\partial x^2}\sigma_g^2\right)^2\big(K(x)-1\big)$$

and differs from (4.12) by one term.

Moreover, by choosing the trigonometric functions like $\sin x$ or $\cos x$ and considering their expansion in $\mu_x = 0$ one might easily miss out the distinction because of the properties of the chosen functions - the important terms providing higher order of accuracy in (4.12) vanish for $\mu_x = 0$, as it does for example for $g(x) = \cos x$:

$$\begin{aligned}\sigma_g^2 &= \sin^2(0)\sigma_x^2 + \sin(0)\cos(0)S(x)\sigma_x^3 \\ &\quad + \frac{1}{4}\cos^2(0)\big(K(x)-1\big)\sigma_x^4 - \frac{1}{3}\sin^2(0)K(x)\sigma_x^4 \\ &= \frac{1}{4}\big(K(x)-1\big)\sigma_x^4.\end{aligned}$$

However, the slight shifting of μ_x away from the zero-point results in an immediate numerical change, as can be seen in Table 4.18.

μ_x	σ_x	First approach		Numerical
		MM(2)	MM(3)	quadrature
0	1	0.500000	0.500000	0.199788
	0.5	0.031250	0.031250	0.024465
	0.1	0.000050	0.000050	0.000049
0.1	1	0.500050	0.499950	0.199812
	0.5	0.031272	0.031266	0.024482
	0.1	0.000051	0.000051	0.000051

Table 4.18: Comparison of the second and third order moments methods using first approach for computing first two statistical moments for the function $g = \cos x$.

The size of the standard deviation of the input x also influences the accuracy of the approximation produced by the moments method. In other words, the smaller the

standard deviation σ_x, the less Taylor terms are required to get a better precision for the statistical moments approximation (see Figure 4.9). Figures 4.10-4.13 show that

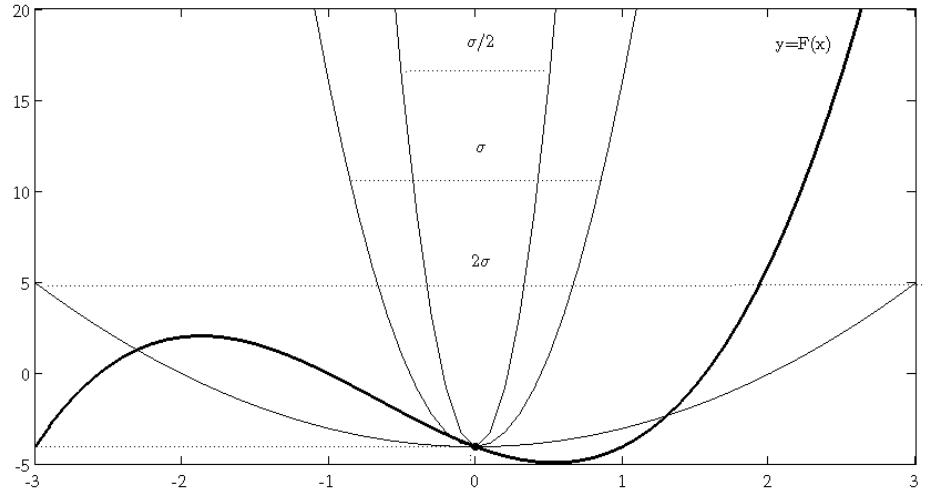


Figure 4.9: The influence of the choice of the standard deviation on the accuracy of the statistical moments approximation using moments method.

for smaller standard deviation even lower order moments method produces accurate result. As an accuracy measuring results to compare to we consider the moments obtained by numerical integration in Matlab, using adaptive Simpson quadrature.

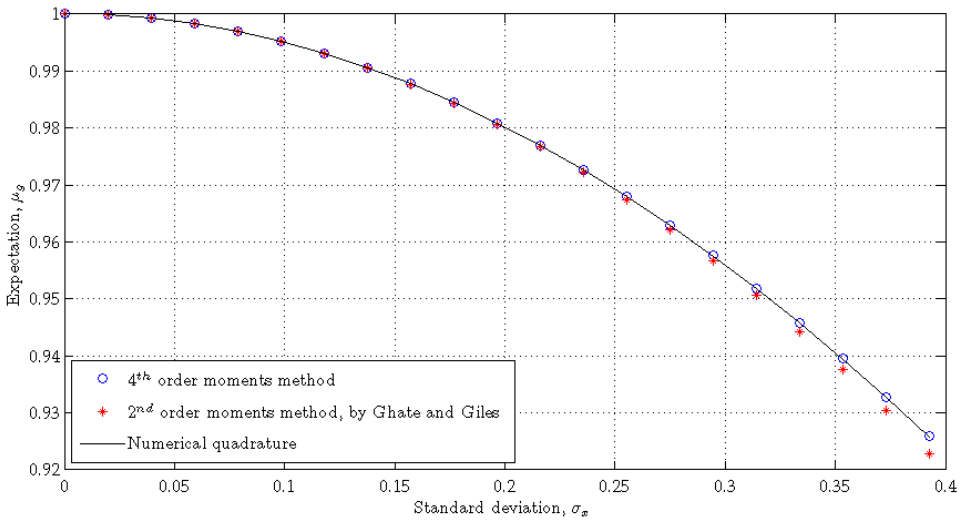


Figure 4.10: The prediction of the expectation μ_g for the function $g(x) = \cos x$ with increasing the standard deviation σ_x, when $\mu_x = 0$.

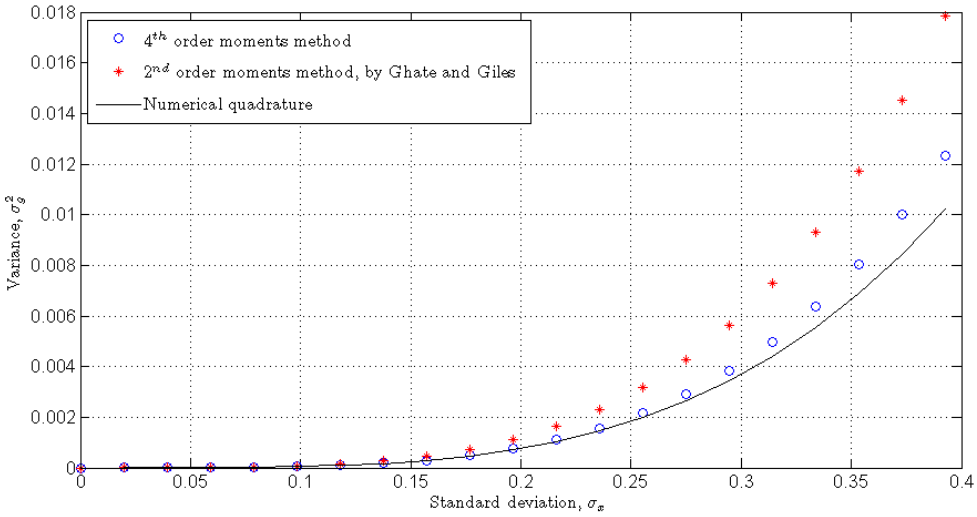


Figure 4.11: The prediction of the variance μ_g for the function $g(x) = \cos x$ with increasing the standard deviation σ_x, when $\mu_x = 0$.

Tables 4.19 and 4.20 compare the Monte Carlo and moments method results and their efficiency* to the numerical integration result.

μ_x	σ_x	ϵ_{MC}	CPU_{MC} time(s)	$\epsilon_{MM(10)}$	$\text{CPU}_{MM(10)}$ time(s)	$\epsilon_{MM(12)}$	$\text{CPU}_{MM(12)}$ time(s)
0	1	1.2459e-03	1.20	2.0382e-05	0.17	1.3197e-06	0.81
		5.6610e-04	1.20				
		2.2133e-03	1.20				
	0.5	2.8314e-04	1.20	2.9275e-09	0.20	8.2257e-09	0.79
		3.2411e-04	1.20				
		3.2354e-04	1.23				
	0.1	1.1720e-05	1.20	6.3297e-09	0.17	6.3297e-09	0.81
		1.8564e-05	1.20				
		1.3492e-05	1.20				
0.01	1	3.8715e-05	1.95	2.0282e-05	0.18	1.4181e-06	0.80
		0.0023e-03	1.25				
		2.6983e-03	1.34				
	0.5	4.4582e-04	1.23	3.7320e-09	0.17	9.0299e-09	0.81
		1.5999e-04	1.20				

Continued on the next page...

*CPU time was obtained using Intel Core 2 Duo 2.66GHz machine with 2GB RAM

...continued from the previous page

μ_x	σ_x	ϵ_{MC}	CPU_{MC} time(s)	$\epsilon_{MM(10)}$	$\text{CPU}_{MM(10)}$ time(s)	$\epsilon_{MM(12)}$	$\text{CPU}_{MM(12)}$ time(s)
		3.4590e-04	1.22				
	0.1	2.9989e-05	1.22	3.0484e-07	0.19	3.0483e-07	0.77
		1.6012e-05	1.23				
		1.6621e-05	1.23				

Table 4.19: Comparison of the MC simulation and MM(p) performances for computing the expectation of the function $g = \cos x$, $N = 100,000$, where ϵ is an error of the method to the quadrature.

μ_x	σ_x	ϵ_{MC}	CPU_{MC} time(s)	$\epsilon_{MM(10)}$	$\text{CPU}_{MM(10)}$ time(s)	$\epsilon_{MM(12)}$	$\text{CPU}_{MM(12)}$ time(s)
0	1	8.5913e-04	1.97	3.4310e-06	1.13	1.0108e-08	8.95
		1.6655e-03	2.00				
		4.7669e-04	2.02				
	0.5	3.1033e-05	2.03	1.0122e-05	1.13	7.1951e-07	9.00
		2.8447e-04	2.06				
		1.0168e-04	2.03				
	0.1	2.0719e-07	1.98	7.1942e-09	1.13	7.1942e-09	9.00
		1.5657e-07	2.00				
		1.4829e-07	2.02				
0.01	1	1.7371e-03	2.00	3.4303e-02	1.12	1.0106e-02	9.09
		1.8694e-03	2.01				
		7.6781e-04	1.98				
	0.5	1.3615e-04	1.98	1.0121e-05	1.14	7.1800e-07	9.01
		7.2823e-05	2.01				
		6.9334e-06	1.97				
	0.1	1.4544e-06	2.06	2.9067e-07	1.13	2.9067e-07	9.05
		2.8487e-07	2.06				
		8.3846e-07	2.02				

Table 4.20: Comparison of the MC simulation and MM(p) performances for computing the variance of the function $g = \cos x$, $N = 100,000$, where ϵ is an error of the method to the quadrature.

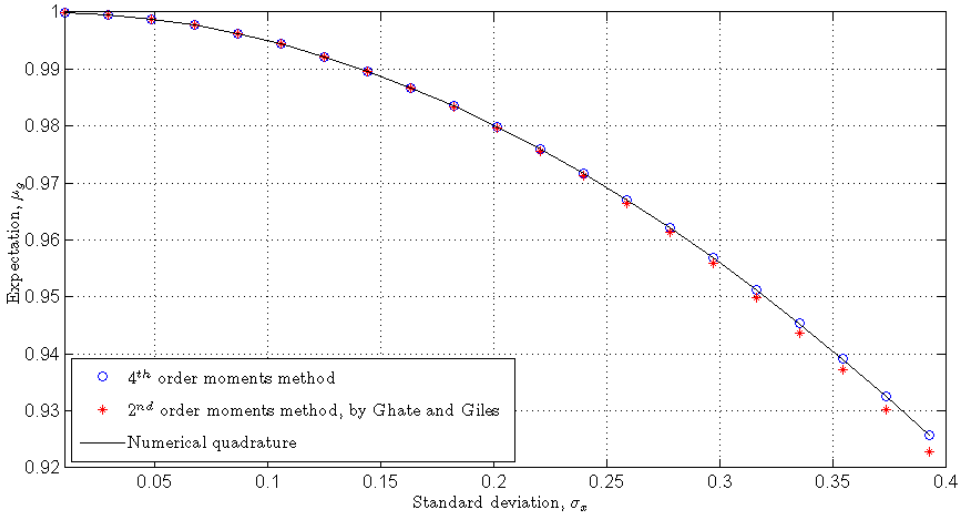


Figure 4.12: The prediction of the expectation μ_g for the function $g(x) = \cos x$ with increasing the standard deviation σ_x, when $\mu_x = 0.01$.

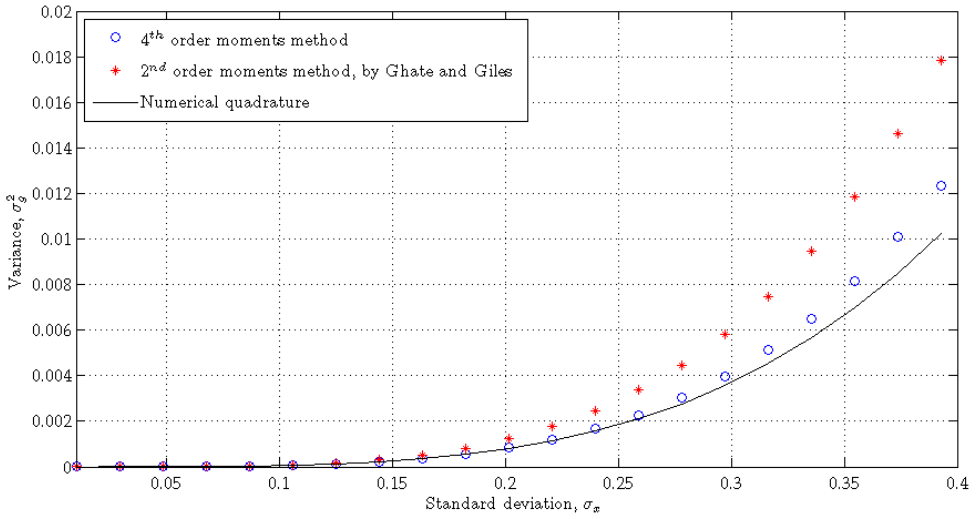


Figure 4.13: The prediction of the variance μ_g for the function $g(x) = \cos x$ with increasing the standard deviation σ_x, when $\mu_x = 0.01$.

The comparison of the moments method results for the expectation and the variance with the Monte Carlo simulation results for an increasing number of iterations N can be seen in Figures 4.14-4.17.

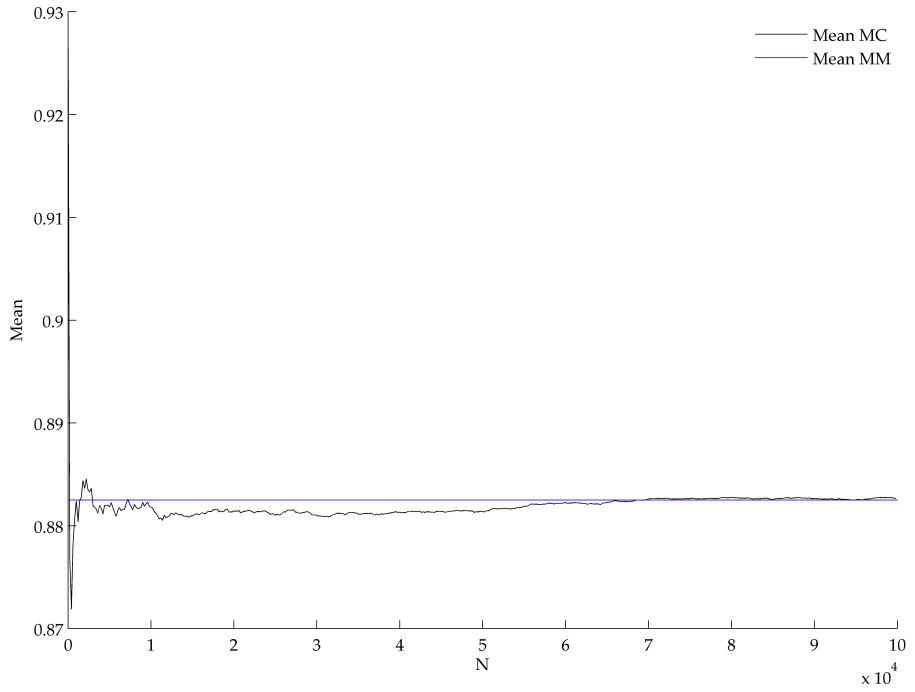


Figure 4.14: Comparison of the Monte Carlo results with the moments method of the order 12 for the function $g(x) = \cos x$, when $\mu_x = 0$ and $\sigma_x = 0.5$, as the number of iterations for MC increases.

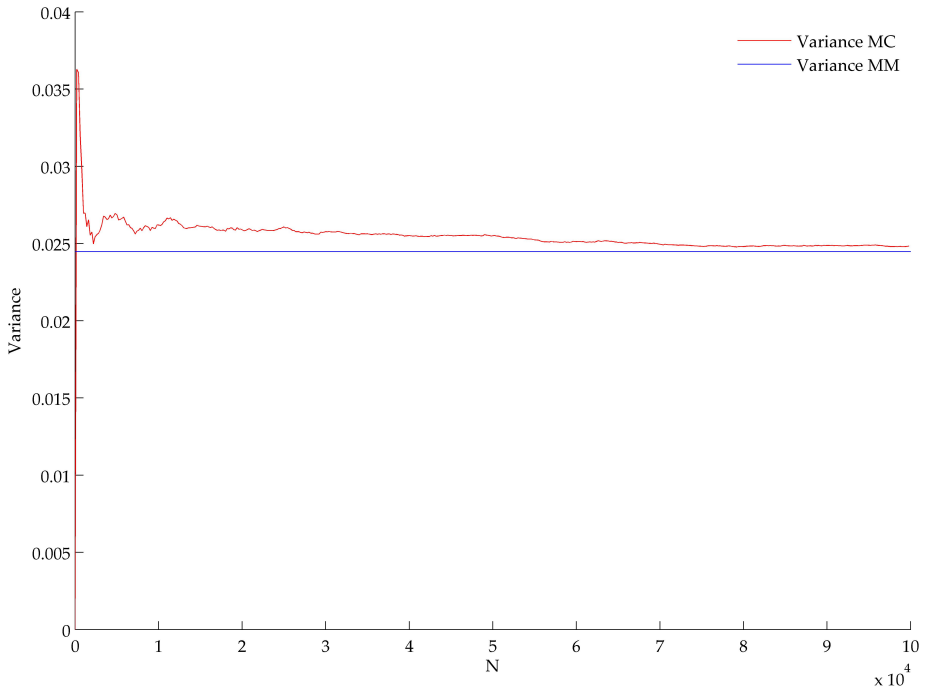


Figure 4.15: Comparison of the Monte Carlo results with the moments method of the order 12 for the function $g(x) = \cos x$, when $\mu_x = 0$ and $\sigma_x = 0.5$, as the number of iterations for MC increases.

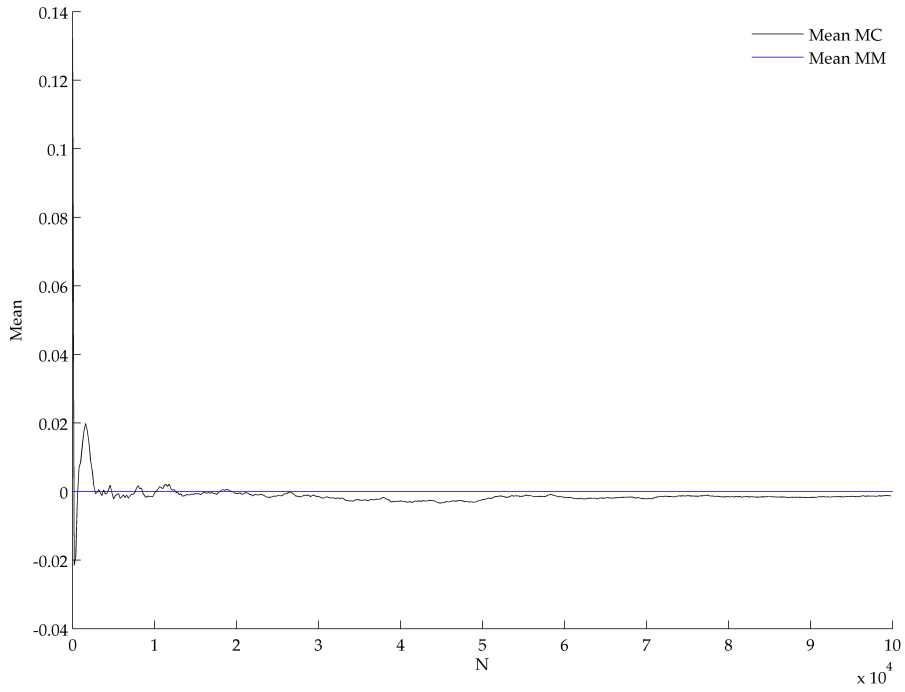


Figure 4.16: Comparison of the Monte Carlo results with the moments method of the order 12 for the function $g(x) = \sin x$, when $\mu_x = 0$ and $\sigma_x = 0.5$, as the number of iterations for MC increases.

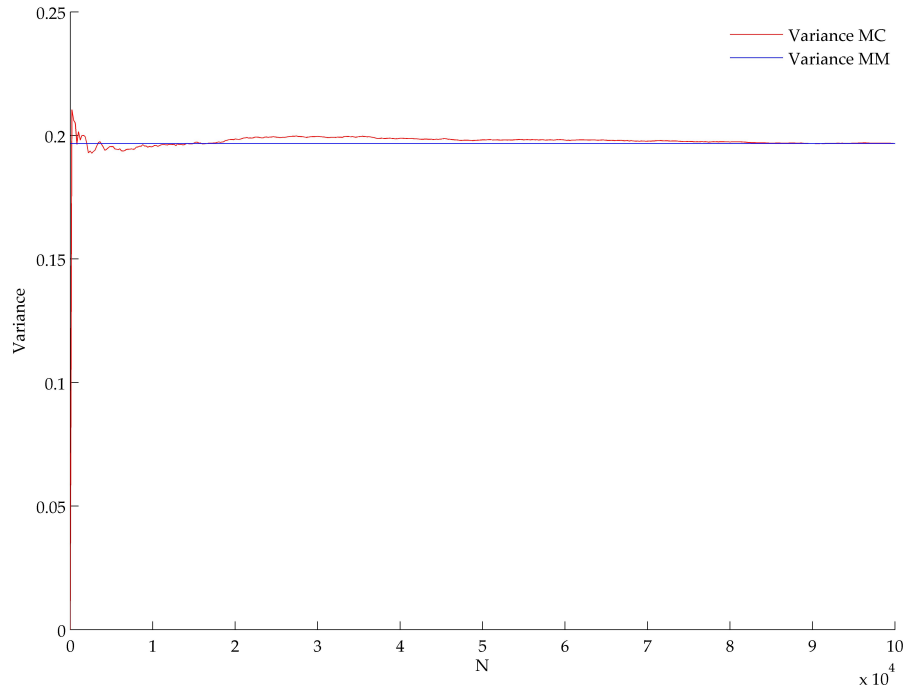


Figure 4.17: Comparison of the Monte Carlo results with the moments method of the order 12 for the function $g(x) = \sin x$, when $\mu_x = 0$ and $\sigma_x = 0.5$, as the number of iterations for MC increases.

The resulting formula for the variance (4.38) computed based on fourth order Taylor approximation matches the one published in the paper by Padulo et al. [7]. However, when testing the moments method in [7] the authors ignore those terms with the derivatives higher than third order. Due to AD tool development in Matlab we do not stop on low order derivatives and are ambitious to use the moments method for arbitrary order p of the Taylor series.

Example 4.5.2. In this example we consider the same functions as tested in [7] to show the importance of the extended approximations for moments:

$$g_1(\mathbf{x}) = \sin(x_1 - 0.21)\sin(x_2 - 0.21); \tag{4.83}$$

$$g_2(\mathbf{x}) = 0.5x_1^2 - 1.5x_1 + 0.7x_2^2 - 1.2x_2 + 1.05; \tag{4.84}$$

$$g_3(\mathbf{x}) = 1.7x_1^3 + 1.3x_1^2 - 2.4x_1 - 0.5x_2^3 + 3.2x_2^2 - 1.6x_2 + 0.1x_1^2x_2 - 0.2x_1x_2^2 + 1.6x_1x_2 - 12.8; \tag{4.85}$$

$$g_4(\mathbf{x}) = 0.4x_1^2 + 0.7x_1 + 0.5x_2^3 - 1.1x_2^2 - 0.9x_2 - 1.3x_2^2x_1. \tag{4.86}$$

The input variables are always normally distributed about the mean $\mu_{\mathbf{x}} = (\mu_{x_1}, \mu_{x_2}) = (1, 1)$.

In Tables 4.21 and 4.22 we compute the expectation and the variance using the moments method based on the second approach. As the distribution is assumed to be normal, the odd order p moments method has the same order of convergence as the one with the order $(p-1)$. One can see that the more non-linear the function is, the more Taylor terms are required to obtain an accurate approximation of statistical moments. For the polynomial functions g_2, g_3 and g_4 we can obtain the exact values of the expectation and the variance if the correct Taylor order is chosen. For example, the g_2 is two times continuously differentiable. Thus to obtain the exact expectation the second order Taylor expansion is required. However, for computing the exact variance, the Taylor order must be equal to four. Since the errors in Tables 4.21 and 4.22 are based on comparison to the Monte Carlo method, where the number of

iterations is $N = 10^6$, the non-zero entries in corresponding to exact values columns in fact demonstrate MC errors.

It is also often the case that the errors for those moments computed based on smaller values of standard deviation have a better order of accuracy. If this condition does not hold (according to the data in Tables 4.21 and 4.22), it still can be obtained by repeating the MC simulation a number of times, as MC is based on the random number generator and thus results vary from run to run.

Test function	$\sigma_{\mathbf{x}}$	Difference between MM and MC ($N = 10^6$)		
		$p = 4$	$p = 6$	$p = 8$
g_1	0.05	4.974106e-04	4.962071e-04	4.962073e-04
	0.2	2.928814e-03	1.999219e-03	1.997759e-03
	0.5	1.357095e-03	1.673573e-03	1.720264e-03
g_2	0.05	*3.653767e-06*	*3.653767e-06*	*3.653767e-06*
	0.2	*2.461048e-04*	*2.461048e-04*	*2.461048e-04*
	0.5	*1.205029e-04*	*1.205029e-04*	*1.205029e-04*
g_3	0.05	*4.127466e-04*	*4.127466e-04*	*4.127466e-04*
	0.2	*1.443245e-03*	*1.443245e-03*	*1.443245e-03*
	0.5	*4.553740e-03*	*4.553740e-03*	*4.553740e-03*
g_4	0.05	*7.195205e-05*	*7.195205e-05*	*7.195205e-05*
	0.2	*1.292212e-03*	*1.292212e-03*	*1.292212e-03*
	0.5	*2.041427e-05*	*2.041427e-05*	*2.041427e-05*

Table 4.21: Expectation estimation.

4.6 Conclusions

In this chapter we have derived the moments method for the expectation by standard application of the expectation operator to the Taylor series of the required order. For the variance we have introduced two different approaches for obtaining the moments method: one is based on squaring the Taylor series of the function g when computing

Test function	σ_x	Error of MM with respect to MC ($N = 10^6$)		
		$p = 4$	$p = 6$	$p = 8$
g_1	0.05	1.025592e-04	1.015295e-04	1.015295e-04
	0.2	1.913518e-02	9.367547e-02	9.389113e-02
	0.5	8.343380e-04	9.907470e-04	1.024517e-03
g_2	0.05	*6.173291e-07*	*6.173291e-07*	*6.173291e-07*
	0.2	*2.270066e-05*	*2.270066e-05*	*2.270066e-05*
	0.5	*2.345744e-04*	*2.345744e-04*	*2.345744e-04*
g_3	0.05	1.082186e-04	*1.098100e-04*	*1.098100e-04*
	0.2	3.418763e-03	*7.268363e-03*	*7.268363e-03*
	0.5	7.430478e-01	*6.648347e-02*	*6.648347e-02*
g_4	0.05	1.872665e-06	*1.685477e-06*	*1.685477e-06*
	0.2	1.236548e-03	*1.004868e-03*	*1.004868e-03*
	0.5	9.802677e-05	*9.821395e-05*	*9.821395e-05*

Table 4.22: Variance estimation.

the $E(g^2)$ term in (2.11), while the other one uses the Taylor approximation of the squared function g^2 instead. Cases with single and multiple inputs are considered. Furthermore, both correlated and uncorrelated types of multiple inputs are examined.

Since the main idea behind the moments method is the use of the Taylor series, the convergence issue is raised. For polynomial functions the order of Taylor series required to obtain an accurate moments approximation is precisely known and is the order of differentiability of the function. For other continuously differentiable functions the accuracy can be improved by increasing the Taylor series order, unless the function g has finite radius of convergence, or by reducing the standard deviation values for the input variables, if applicable. Finite radius of convergence may cause the divergence of the Taylor series. The alternative, *partitioning approach*, to deal with divergent series is developed and demonstrated on a number of examples.

We have also tested some of the functions previously used in related papers (i.e. [7, 4]) to verify the results as well as to make evident the advantages of using higher

order moments methods.

Chapter 5

Algorithm and software for moments estimation

To implement the moments computation upto an arbitrary order in Matlab we are using the approach based on the approximation of the expectation as described in Section 4.2:

$$\mu_g = g + \frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2 + \frac{1}{3!}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^3 + \ldots. \tag{5.1}$$

By substituting the function g^i in the place of g in (5.1) and exploiting the relation (2.13) for computing higher order moments

$$M_i(g) = \frac{E\big((g-\mu_g)^i\big)}{\sigma_g^i}, \tag{5.2}$$

where i is the order of the moment, arbitrary order moments can be obtained.

For example, to compute the variance,

$$\sigma_g^2 = M_2(g) = \mu_{g^2} - \mu_g^2, \tag{5.3}$$

one requires a tool that computes the expectation of the function g given that the expectation, the variance, and the distribution type for all input parameters are known. Our tool for doing this is known as `MADMean`, where `MAD` stands for Matlab Automa-

tic Differentiation. This routine then can approximate both the expectation of the function g and the expectation of the function g^2. When subtracting the squared expectation of g from the expectation of g^2, one gets the variance σ_g^2.

Analogously, `MADMean` can be used for computing the expectation of g^i, $i = 3, 4, \ldots$, necessary to acquire moments of order i.

Thus by developing `MADMean` we simultaneously solve the problem of approximating higher order moments as well. What do we need to successfully implement the `MADMean` tool?

5.1 Uncorrelated case

The p^{th} order moments method for the expectation is based on the consecutive summation of p terms obtained by applying the expectation operator to the p^{th} order Taylor series:

$$\begin{aligned}\mu_g &= E\big(g(\mathbf{x})\big) = E\big(g\big) + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i} E(x_i - \mu_{x_i}) \\ &+ \frac{1}{2!} \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j} E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})\big) \\ &+ \frac{1}{3!} \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k} E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k})\big) \\ &+ \ldots. \end{aligned} \tag{5.4}$$

The k^{th} term T_k of the moments approximation in (4.23) is

$$T_k = \frac{1}{k!} \sum_{i_1=1}^{n} \cdots \sum_{i_k=1}^{n} \frac{\partial^k g}{\partial x_{i_1} \ldots \partial x_{i_k}} E\big((x_{i_1} - \mu_{x_{i_1}}) \ldots (x_{i_k} - \mu_{x_{i_k}})\big). \tag{5.5}$$

Of course, every term contains partial derivatives. Therefore we must have an efficient approach to access them. Associated with each partial derivative $\frac{\partial^k g}{\partial x_{i_1} \ldots \partial x_{i_k}}$ in (5.5) is what we refer to as an *index vector* of k^{th} order $(i_1, \ldots, i_k)$, $k = 1, \ldots, p$, specifying

the partial input variables for that term. Here, p is the order of the requested Taylor approximation. Every single term in the summation of (5.5) can be written as

$$T_k^c = \frac{\partial^k g}{\partial x_{i_1}^{q_1} \dots \partial x_{i_l}^{q_j}} E(x_{i_1} - \mu_{x_{i_1}})^{q_1} \dots E(x_{i_l} - \mu_{x_{i_l}})^{q_j}, \tag{5.6}$$

with $c = 1, \dots, n^k$ - running for all possible permutations of indeces in the index vector; $q_1 + \dots + q_j = k$, $1 \le l \le k$, where q_i is termed the *multiplicity*.

Here we consider uncorrelated inputs,

$$E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})\big) = \begin{cases} \sigma_{x_i}^2, & \text{if i = j,} \\ 0, & \text{otherwise.} \end{cases} \tag{5.7}$$

Hence some of the terms in (5.4) disappear. The index vector, where $i_j \neq i_l$, for any $j, l = 1, \dots, k$, $j \neq l$, meaning i_j appears in the index vector uniquely, corresponds to a *zero term* of the moments approximation. In other words, when any $q_i = 1$, $i = 1, \dots, j$, the respective term of (5.6) is a zero term.

To identify the derivatives included in all non-zero terms of the moments approximation, we collect all possible index vectors and omit those corresponding to zero terms.

Example 5.1.1. Consider the function of three input arguments $g(\mathbf{x}) = g(x_1, x_2, x_3)$. To get the 4^{th} order moments method for the expectation one requires partial derivatives of g upto the order $p = 4$. Let us take a closer look at some examples of index vectors for the 4^{th} term of the expectation approximation

$$T_4 = \frac{1}{4!} \sum_{i_1=1}^{3} \sum_{i_2=1}^{3} \sum_{i_3=1}^{3} \sum_{i_4=1}^{3} \frac{\partial^4 g}{\partial x_{i_1} \partial x_{i_2} \partial x_{i_3} \partial x_{i_4}} E\big((x_{i_1} - \mu_{x_{i_1}})(x_{i_2} - \mu_{x_{i_2}})(x_{i_3} - \mu_{x_{i_3}})(x_{i_4} - \mu_{x_{i_4}})\big).$$

For the index vector $(i_1\ i_2\ i_3\ i_4) = (1\ 1\ 2\ 2)$ we have a term

$$T_4^6 = \frac{\partial^4 g}{\partial x_1 \partial x_1 \partial x_2 \partial x_2} E(x_1 - \mu_{x_1}) E(x_1 - \mu_{x_1}) E(x_2 - \mu_{x_2}) E(x_2 - \mu_{x_2})$$

$$= \frac{\partial^4 g}{\partial x_1^2 \partial x_2^2} E(x_1 - \mu_{x_1})^2 E(x_2 - \mu_{x_2})^2.$$

The term does not satisfy the zero term definition, hence we must retain it for the moments approximation. In contrast, index vector (1 1 1 2) produces a zero term

$$T_4^2 = \frac{\partial^4 g}{\partial x_1^3 \partial x_2} E(x_1 - \mu_{x_1})^3 E(x_2 - \mu_{x_2}),$$

thus needs to be eliminated, since $E(x_2 - \mu_{x_2}) = 0$.

Table 5.1 shows the relations between the index vectors and terms of the moments approximation for $n = 3$ and $k = 4$. □

Ordered combinations of indices	Corresponding terms	Classification
1 1 1 1	$\frac{\partial^4 g}{\partial x_1^4} E(x_1 - \mu_{x_1})^4$	non-zero
1 1 1 2	$\frac{\partial^4 g}{\partial x_1^3 \partial x_2} E(x_1 - \mu_{x_1})^3 E(x_2 - \mu_{x_2})$	zero
1 1 1 3	$\frac{\partial^4 g}{\partial x_1^3 \partial x_3} E(x_1 - \mu_{x_1})^3 E(x_3 - \mu_{x_3})$	zero
1 1 2 2	$\frac{\partial^4 g}{\partial x_1^2 \partial x_2^2} E(x_1 - \mu_{x_1})^2 E(x_2 - \mu_{x_2})^2$	non-zero
1 1 2 3	$\frac{\partial^4 g}{\partial x_1^2 \partial x_2 \partial x_3} E(x_1 - \mu_{x_1})^2 E(x_2 - \mu_{x_2}) E(x_3 - \mu_{x_3})$	zero
1 1 3 3	$\frac{\partial^4 g}{\partial x_1^2 \partial x_3^2} E(x_1 - \mu_{x_1})^2 E(x_3 - \mu_{x_3})^2$	non-zero
1 2 2 2	$\frac{\partial^4 g}{\partial x_1 \partial x_2^3} E(x_1 - \mu_{x_1}) E(x_2 - \mu_{x_2})^3$	zero
1 2 2 3	$\frac{\partial^4 g}{\partial x_1 \partial x_2^2 \partial x_3} E(x_1 - \mu_{x_1}) E(x_2 - \mu_{x_2})^2 E(x_3 - \mu_{x_3})$	zero
1 2 3 3	$\frac{\partial^4 g}{\partial x_1 \partial x_2 \partial x_3^2} E(x_1 - \mu_{x_1}) E(x_2 - \mu_{x_2}) E(x_3 - \mu_{x_3})^2$	zero
1 3 3 3	$\frac{\partial^4 g}{\partial x_1 \partial x_3^3} E(x_1 - \mu_{x_1}) E(x_3 - \mu_{x_3})^3$	zero

Continued on the next page...

...continued from the previous page

Ordered combinations of indices	Corresponding terms	Classification
2 2 2 2	$\frac{\partial^4 g}{\partial x_2^4} E(x_2 - \mu_{x_2})^4$	non-zero
2 2 2 3	$\frac{\partial^4 g}{\partial x_2^3 \partial x_3} E(x_2 - \mu_{x_2})^3 E(x_3 - \mu_{x_3})$	zero
2 2 3 3	$\frac{\partial^4 g}{\partial x_2^2 \partial x_3^2} E(x_2 - \mu_{x_2})^2 E(x_3 - \mu_{x_3})^2$	non-zero
2 3 3 3	$\frac{\partial^4 g}{\partial x_2 \partial x_3^3} E(x_2 - \mu_{x_2}) E(x_3 - \mu_{x_3})^3$	zero
3 3 3 3	$\frac{\partial^4 g}{\partial x_3^4} E(x_3 - \mu_{x_3})^4$	non-zero

Table 5.1: All terms of 4-th order for $g(\mathbf{x}) = g(x_1, x_2, x_3)$.

Practically, it is more efficient to deal with ordered combinations of indices instead of permutations. In this case, we apply the check $\frac{(n+k-1)!}{k!(n-1)!}$ times only instead of n^k, and then multiply the corresponding successful index vectors terms by the number of its entries in the moments approximation.

One can summarise the algorithm for identifying the non-zero terms by manipulating index vectors and formulate it in general terms.

Algorithm

Let $\Lambda = \emptyset$ be an empty set; $\Theta = \{\theta^c = (\theta_1^c, \ldots, \theta_i^c),\ c = 1, \ldots, \frac{(n+k-1)!}{k!(n-1)!}\}$ - the set of all ordered combinations, combinations with repetitions, of k indeces from 1 to n.

For every θ^c we form the *difference vector* $\Delta\theta^c = (1,\ \theta_2^c - \theta_1^c,\ \ldots,\ \theta_k^c - \theta_{k-1}^c,\ 1)$. The differences between neighbouring elements θ_{i+1}^c and θ_i^c of the index vector

can either be zero, indicating that i-th index appears in θ^c more than once, or non-zero, signifying a change. Therefore, two successive non-zeroes in a row specify that the index vector corresponds to a zero term.

The operation of adding ones at the beginning and at the end of the vector we call *padding*. It is necessary for detecting the changes of indices on the edges of θ^c.

If $\Delta\theta^c$ does not contain two successive non-zeroes then add θ^c to the set Λ.

```
for k = 2 : p
    for c = 1 : (n+k-1)!/k!(n-1)!
        check = TRUE;
        Δθ^c = (1, θ^c_2 − θ^c_1, ..., θ^c_k − θ^c_{k−1}, 1);
        for i = 1 : k
            if Δθ^c_i ≠ 0 and Δθ^c_{i+1} ≠ 0 then
                check = FALSE;
            end
        end
        if check then
            Λ = {Λ; θ^c}
        end
    end
end
```

Here the parameter *check* is an indicator of logical type, the value of which indicates whether the respective index vector corresponds to a non-zero term and needs to be saved in Λ. The number of indices k runs from 2 to p since here we are interested only in partial derivatives of order $\geqslant 2$.

Example 5.1.2. As before consider the function $g(\mathbf{x}) = g(x_1, x_2, x_3)$ which is dependent on three input variables. Initially the set of non-zero terms is empty: $\Lambda = 0$. Applying the algorithm for this function, we will get the set of all indices corresponding to non-zero terms.

When $k = 2$, the number of all ordered combinations of indices is

$$\frac{(n+k-1)!}{k!(n-1)!} = \frac{(3+2-1)!}{2!(3-1)!} = 6.$$

And the set of all indices is

$$\Theta = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 2 & 2 \\ 2 & 3 \\ 3 & 3 \end{bmatrix}.$$

Now for all $c = 1 : 6$ we form padded difference vectors $\Delta\theta^c$ and check if $\Delta\theta^c$ defines a non-zero term in (5.1). Thus,

$$\Delta\Theta = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}.$$

When $c = 1$, the given $\theta^1 = (1\ 1)$ corresponds to $T_2^1 = \dfrac{\partial^2 g}{\partial x_1^2}\sigma_{x_1}^2$. The corresponding padded difference vector $\Delta\theta^1 = (1\ 0\ 1)$. It does not contain any successive non-zeros, therefore we update our Λ:

$$\Lambda = [\ 1\ 1\ ].$$

If $c = 2$, $\theta^2 = (1\ 2)$ corresponds to $T_2^2 = \dfrac{\partial^2 g}{\partial x_1 \partial x_2} E(x_1 - \mu_{x_1}) E(x_2 - \mu_{x_2})$, where both $E(x_1 - \mu_{x_1})$ and $E(x_2 - \mu_{x_2})$ are zeros as in (4.22). Hence θ^2 is expected to be

excluded. When checking its padded difference vector $\Delta\theta^2 = (1\ 1\ 1)$ we see three consecutive non-zeros. Therefore this term is indeed zero and so at this step Λ remains unchanged.

By proceeding with the same algorithm for all index vectors when $k = 2$, Λ becomes

$$\Lambda = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}.$$

Similarly, let $k = 4$. The set of all $\dfrac{(n+k-1)!}{k!(n-1)!} = \dfrac{(3+4-1)!}{4!(3-1)!} = 15$ ordered combinations is

$$\Theta = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 3 \\ 1 & 1 & 2 & 2 \\ \vdots & \vdots & \vdots & \vdots \\ 3 & 3 & 3 & 3 \end{bmatrix}.$$

The padded difference vectors for this Θ are

$$\Delta\Theta = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 2 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Thus when $c = 1$, $\theta^1 = (1\ 1\ 1\ 1)$ corresponds to $T_4^1 = \dfrac{\partial^4 g}{\partial x_1^4} K(x_1)\sigma_{x_1}^4$. It is a non-zero term, therefore we expect $\Delta\theta^1$ not to contain any successive non-zeros. Since $\Delta\theta^1 = (1\ 0\ 0\ 0\ 1)$ satisfies the condition of identification of non-zero terms, we add θ^1 to Λ. The same happens for $c = 4$, $\theta^4 = (1\ 1\ 2\ 2)$ that defines the term $T_4^4 = \frac{\partial^4 g}{\partial x_1^2 \partial x_2^2}\sigma_{x_1}^2\sigma_{x_2}^2$. It is a non-zero term, and the padded difference vector $\Delta\theta^4 = (1\ 0\ 1\ 0\ 1)$ confirms this. The terms coinciding with index vectors in between, i.e. $c = 2$, $c = 3$, are zeroes

and need to be eliminated. As a final result, the set of all indices that create non-zero approximation terms are

$$\Lambda = \begin{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \\ 3 & 3 & 3 \end{bmatrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & 2 \\ 1 & 1 & 3 & 3 \\ 2 & 2 & 2 & 2 \\ 2 & 2 & 3 & 3 \\ 3 & 3 & 3 & 3 \end{bmatrix} \end{bmatrix}. \square$$

Now we know which coefficients are required for non-zero terms of (5.4) and because we represent those coefficients as ordered arrangements, permutations, we also want to know the number of times the corresponding element occurs in the approximation. For example, the single index vector (1 1 2 2) represents (1 1 2 2), (1 2 1 2), (1 2 2 1), (2 1 1 2), (2 1 2 1), (2 2 1 1). This problem, classified as a problem of combinatorial algebra, applies to every index vector of Λ.

Let us assume that every vector entry λ of the size k from Λ is a vector constructed from the elements of the set $A = \{a_1, \ldots, a_n\} = \{1, \ldots, n\}$, with n still a dimension of the problem. Every element $a_i = i$, $i = 1, \ldots, n$, occurs in λ $k_i \geqslant 0$ times, $\sum_{i=1}^{n} k_i = k$. What is the number of all possible arrangements of r elements of the set A, taking into account that each i^{th} element is repeated exactly k_i times? In [24] such arrangements are refered to as permutations with limited repetitions.

Example 5.1.3. Let $A = \{1, 2\}$, $k_1 = 2$, $k_2 = 2$. We first create a new set $\tilde{A}$ where every element of A is repeated corresponding number of times: $\widetilde{A} = \{1, 1, 2, 2\}$. The

number of $(k_1 + k_2)$ permutations without repetitions for elements of $\widetilde{A}$ is defined by (2.41): $(k_1 + k_2)! = 4!$. However, in this case both $\widetilde{a}_1 = 1$ and $\widetilde{a}_2 = 1$ are considered to be distinct elements of $\widetilde{A}$. We account for $(\widetilde{a}_1\ \widetilde{a}_2)$, but need to avoid $(\widetilde{a}_2\ \widetilde{a}_1)$ since in principle $(\widetilde{a}_1\ \widetilde{a}_2) = (\widetilde{a}_2\ \widetilde{a}_1) = (1\ 1)$. There are $k_1! = 2!$ ways to compute all possible k_1-permutations without repetitions for them. Thus

$$\frac{(k_1 + k_2)!}{k_1!} = \frac{4!}{2!} \tag{5.8}$$

defines the number of (k_1+k_2)-permutations of all elements of $\widetilde{A}$ when repetitions are not allowed and $\widetilde{a}_1$, $\widetilde{a}_2$ are considered to be the same. Analogously, for $\widetilde{a}_3 = \widetilde{a}_4 = 2$ we reduce (5.8) by all possible k_2-permutations without repetitions:

$$\frac{(k_1 + k_2)!}{k_1!\ k_2!} = \frac{4!}{2!\ 2!} = \frac{4 \times 3 \times 2}{2 \times 2} = 6. \tag{5.9}$$

Generalising the simple case considered in Example 5.1.3 for $A = \{a_1, \ldots, a_n\}$ the number of all permutations of the set A, where i^{th} element is repeated k_i times, $i = 1, \ldots, n$, is

$$\frac{(k_1 + k_2 + \cdots + k_n)!}{k_1!\ k_2!\ \ldots\ k_n!}. \tag{5.10}$$

5.1.1 Programming the algorithm

With all the details cleared now, the final algorithm for the implementation using the programming language is summarised in this section.

The input arguments required to perform the expectation approximation using the moments method are the modeling function $g(\mathbf{x})$, the order of Taylor series expansion p desired for the moments method approximation, the expectation $\mu_{\mathbf{x}} = (\mu_{x_1}, \ldots, \mu_{x_n})$, the standard deviation $\sigma_{\mathbf{x}} = (\sigma_{x_1}, \ldots, \sigma_{x_n})$, where n is the number of input variables for the modeling function g, and the reference to the function that

computes the statistical moments for the input variables $x_1, \ldots, x_n$.

First we compute the derivatives of the function g upto order p:

Δg = cell(1,$p+1$);

[Δg{:}] = MADHigherDerivs(@g,$\mu_{\mathbf{x}}$,[],extraargs{:});

Then we initialise the computation of the expectation by computing it using the first order moments method:

μ_g = $g(\mu_{\mathbf{x}})$

Depending on the order of the Taylor series expansion requested, the following holds:

% for every i^{th} element of the vector function g

for i = 1:dimension(g)

% compute Taylor coefficients

for j = 2:p

% compute non-zero indices Λ of order j

Λ = [...];

sp = 0; *% spare element*

for every $\lambda \in \Lambda$

% compute the number of distinct elements in every λ

% $N_{\Delta\theta}$ is the number of non-zero elements in corresponding $\Delta\theta$

$\tilde{k}$ = $N_{\Delta\theta}$ - 1;

% compute the number of repetitions of every index in λ

$k_1^\lambda, \ldots, k_{\tilde{k}}^\lambda$ = ...;

% compute corresponding k_l^{th} moments of the given distribution

$M_{k_1}^\lambda, \ldots, M_{k_{\tilde{k}}}^\lambda$ = ...;

l = length(λ);

$$sp = sp + \frac{(k_1^\lambda + \cdots + k_{\tilde{k}}^\lambda)!}{k_1^\lambda! \ldots k_{\tilde{k}}^\lambda!} * \Delta g(\lambda) * \left(M_{k_1}^\lambda * \cdots * M_{k_{\tilde{k}}}^\lambda\right) * \left(\sigma_{\lambda(1)} * \cdots * \sigma_{\lambda(l)}\right);$$

```
            end
            μ_g(i) = μ_g(i) + 1/j! * sp;
        end
    end
```

The most computationaly expensive step of this algorithm currently is the one that computes the derivatives in Matlab, `MADHigherDerivs`. The produced derivatives are stored in cell arrays, where the array corresponding to the p^{th} order derivatives is of the size $m \times n^p$, m is the dimension of the function $g(\mathbf{x})$, and n is the dimension of the vector of input variables $\mathbf{x} = (x_1, \ldots, x_n)$.

5.2 Correlated case

Assuming that the modeling function $g(\mathbf{x})$ as well as the statistical moments up to an order p for the input variables $\mathbf{x} = (x_1, \ldots, x_n)$ are given, the implementation of the correlated case of the moments method becomes very straight forward as there is no longer any explicitly zero terms. With no such assumption, we must consider every element of the approximation as the one that has possible non-zero impact on the accuracy of the moments computations. Thus the algorithm for computing the expectation approximation of order p is

1. compute all required derivatives upto an order p;
2. initialise the expectation computation by computing the moments method of order 1:
   $$\mu_g = g(\mu_{\mathbf{x}});$$
3. for all p arrays of derivatives, update the expectation of the modeling function as
   $$\mu_g = \mu_g + \frac{1}{j!} \sum \left(\partial^j g \times \text{corresponding moment(s)} \right),$$
   where $j = 2, \ldots, p$, and the sum is over all derivatives of the order j.

5.3 Comparison of the algorithms

The algorithm for uncorrelated inputs is in fact a partial case of the one for correlated inputs. A simple restructuring of the vectors with statistical moments for $\mathbf{x}$ into the arrays of the corresponding dimensions with zeros in place of the correlations allows us to use the more general algorithm for computing the statistical moments of the function $g(\mathbf{x})$, where all elements of $\mathbf{x}$ are statistically independent.

The anticipation however would be that the memory required to run the algorithm increases as the space for storing the large arrays with the values for the statistical moments increases exponentially. In other words, to store a $1 \times n$ vector of values in Matlab there are $8 \times n$ bytes in use, so to store the accordingly modified n^j array of values, where j is the order of the statistical moment considered, $8 \times n^j$ bytes is required. Thus, for example, for a simple problem with only two input variables, to store their statistical moments up to the order 16 one would use 1Mb of the memory instead of 256 bytes when keeping them in vector form. Table 5.2 demonstrates the change in memory requirements when one switches from the vector representation of the statistical moments to the multidimensional array form.

Number of input variables	The order of Taylor expansion	Bytes in vector form	Bytes in multidimensional array form
2	15	240	524272 ≈ 0.50Mb
	16	256	1048560 ≈ 1.00Mb
3	10	240	708576 ≈ 0.68Mb
	11	264	2125752 ≈ 2.03Mb
4	8	256	699040 ≈ 0.67Mb
	9	288	2796192 ≈ 2.67Mb
5	7	280	781240 ≈ 0.75Mb
	8	320	3906240 ≈ 3.73Mb
10	4	320	88880 ≈ 0.08Mb
	6	480	8888880 ≈ 8.48Mb

Continued on the next page...

...continued from the previous page

Number of input variables	The order of Taylor expansion	Bytes in vector form	Bytes in multidimensional array form
	8	640	888888880 ≈ 847.71Mb

Table 5.2: The memory requirements for storing the statistical moments of input variables. (1Mb = 1048576 Bytes)

Consequently, the running time increases as well, since the computational loop expands. Even though we still deal with highly sparse cell arrays of large dimensions that store the derivatives, when computing the moments using the uncorrelated case we access only those derivatives that potentially produce the non-zero affect on achieving the final result. For the correlated case however we force the program to run through all derivatives as there is no way of knowing which one would influence the computation.

Example 5.3.1. For the function $g(\mathbf{x}) = g(x_1, x_2, x_3)$ calculation of the expectation using 4^{th} order moments method requires only 6 derivatives of the order 4 that correspond to the non-zero terms:

$$\left\{ \frac{\partial^4 g}{\partial x_1^4}, \frac{\partial^4 g}{\partial x_1^2 \partial x_2^2}, \frac{\partial^4 g}{\partial x_1^2 \partial x_3^2}, \frac{\partial^4 g}{\partial x_2^4}, \frac{\partial^4 g}{\partial x_2^2 \partial x_3^2}, \frac{\partial^4}{\partial x_3^4} \right\}$$

We do not even need to repeatedly accumulate these derivatives through all possible combinations of indices when they occur, since we simply compute their coefficients by using (5.10). Instead of accessing those derivatives directly when knowing their indices as it is done in the case of independent inputs, the correlated algorithm calls for them $3^4 = 81$ times. It is not difficult to calculate how many times the loop is increased by when increasing the order of the moments method as well as the dimension of the problem.

5.4 Test

To demonstrate how the newly developed software functions, in this section we consider several examples.

Example 5.4.1. As a simplest test, let us consider a polynomial function of a single input variable

$$g(x) = x^3 + 2x^2 - 3x - 4. \tag{5.11}$$

The analytical results for this function with the $\mu_x = 0$ and $\sigma_x = 1$, when x is normally distributed are

$$\begin{cases} \mu_g &= -2.0000; \\ \sigma_g^2 &= 14.0000. \end{cases}$$

The function is 3 times continuously differentiable, thus to compute accurate expectation the Taylor series of the third order is required. Using `MADMoments` function we get:

```
>> [Eg,Vg] = MADMoments(Func,3,0,1,'Normal')
Eg = -2.0000
Vg = 5.0000
```

Even though the third order Taylor approximation produces the exact result for the expectation, for the variance the expectation of the squared function g^2 is required to be computed. This necessitates the sixth order Taylor series for the exact variance approximation. Our approach makes use of the arbitrary order Taylor series giving accurate results, limited only by the efficiency of the underlying AD tool.

```
>> [Eg,Vg] = MADMoments(Func,6,0,1,'Normal')
Eg = -2.0000
Vg = 14.0000
```

The accuracy of the result depends not only on the Taylor order, but the input's standard deviation as well. For sufficiently small standard deviation fewer Taylor terms

are required to get higher order accuracy in moments approximation. Analytically, when $\mu_x = 0$, $\sigma_x = 0.5$, the first two statistical moments for the function g are

$$\begin{cases} \mu_g &= -3.5000; \\ \sigma_g^2 &= 1.8594. \end{cases}$$

The results using the third order moments method produces the following results:

```
>> [Eg,Vg] = MADMoments(Func,3,0,0.5,'Normal')
Eg = -3.5000
Vg = 2.0000
```

As well as when for $\mu_x = 0$ and $\sigma_x = 0.05$ analytical integration results in

$$\begin{cases} \mu_g &= -3.99500; \\ \sigma_g^2 &= 0.02244, \end{cases}$$

and the moments method simulation produces

```
>> [Eg,Vg] = MADMoments(Func,3,0,0.05,'Normal')
Eg = -3.99500
Vg = 0.02247
```

To obtain these results for such simple example using the third order moments method, the computer takes virtually no run time (0.01 s and 0.03 s for the third and the sixth order moments method, respectively). For comparison in Table 5.3 we provide the CPU time* required to get similar results using the Monte Carlo method.

N	Analytical expectation	MC expectation error	Analytical variance	MC variance error	CPU time (s)
100,000	-2.000000	0.012851	14.000000	0.905540	1.94
		0.023412		0.693269	1.92
		0.009497		0.552186	1.91
1,000,000		0.000295		0.029438	21.52
		0.001907		0.007149	21.53

Continued on the next page...

*CPU time was obtained using Intel Core 2 Duo 2.66GHz machine with 2GB RAM

...continued from the previous page

N	Analytical expectation	MC expectation error	Analytical variance	MC variance error	CPU time (s)
		0.003163		0.209460	21.70
10,000,000		0.001865		0.022720	217.61
		0.000902		0.010637	218.73
		0.001081		0.003364	216.58

Table 5.3: The performance of the Monte Carlo method for the function $g(x) = x^3 + 2x^2 - 3x - 4$, when $\mu_x = 0$, $\sigma_x = 1$, and N is a number of MC iterations.

One can see that even after over 3 mins of running the Monte Carlo method with 10,000,000 iterations the approximation may still be considered insufficiently accurate depending on the purposes of this computation.

Example 5.4.2. Let us now consider a vector function $g(\mathbf{x})$ of the multiple input variables

$$\begin{aligned} g_1(\mathbf{x}) &= x_1 + x_2 x_3 - 2; \\ g_2(\mathbf{x}) &= x_1 x_2 + x_3 + 1; \\ g_3(\mathbf{x}) &= x_1 x_2 x_3. \end{aligned}$$

It is a polynomial of the third order, thus we can obtain the exact values for the statistical moments when choosing the right Taylor order. To compute the expectation and the variance accurately, we require sixth order Taylor expansion. The input variables are assumed to be statistically independent distributed around $\mu_{\mathbf{x}} = (0.1, 0.1, 0.1)$ using normal distribution with the standard deviations $\sigma_{\mathbf{x}} = (\sigma_{x_1}, \sigma_{x_2}, \sigma_{x_3}) = (1.0, 0.5, 0.2)$. We can show the change in CPU requirements when running the moments method through the uncorrelated and correlated modes. The first one in 1.04 s[†] produces

```
>> Ex = [0.1,0.1,0.1];% the vector of expectations
```

[†]CPU time was obtained using Intel Core 2 Duo 2.66GHz machine with 2GB RAM

```
>> SDx = [1,0.5,0.2]; % the vector of standard deviations
>> [Eg,Vg] = MADMoments(Func,6,Ex,SDx,'normal')
Eg =
    -1.89000000000000
     1.11000000000000
     0.00100000000000
Vg =
    1.01290000000000   0.10650000000000   0.01129000000000
    0.10650000000000   0.30250000000000   0.02665000000000
    0.01129000000000   0.02665000000000   0.01312900000000
```

For the comparison, the results obtained by MC simulation are

$$\mu_g = \begin{pmatrix} -1.89424208317729 \\ 1.11134506559027 \\ 0.00030327304238 \end{pmatrix},$$

$$\sigma_g^2 = \begin{pmatrix} 1.01466584179149 & 0.10608962403285 & 0.01110260074542 \\ 0.10608962403285 & 0.29975838193223 & 0.02651170606261 \\ 0.01110260074542 & 0.02651170606261 & 0.01319065605791 \end{pmatrix}.$$

Providing that the higher order moments for the input variables with joint normal distribution are known and given when calling the correlated case of the moments method implementation, the software obtains identical results to the one for uncorrelated case within the average of 1 s[‡].

Here one must keep in mind, that the general representation of the moments method implementation does not compute statistical moments for the input variables, and requires them to be provided as input parameters when calling the `MADMomets`. It is also more difficult to obtain the general form for the statistical moments of the known joint distribution. Thus the use of the correlated mode of the `MADMoments` tool is limited by the data provided.

[‡]CPU time was obtained using Intel Core 2 Duo 2.66GHz machine with 2GB RAM

By increasing the number of the input variables for the function $g(\mathbf{x})$ preserving its complexity, an average CPU time for the moments method implementations resembles the similar behaviour, as shown in Table 5.4. Thus when deciding which

n	Average CPU time(s)	
	correlated	uncorrelated
3	1.00	1.04
4	1.03	1.12
5	1.23	1.32
6	1.40	1.56
7	2.04	2.54
8	3.27	4.24
9	5.86	7.12
10	10.39	12.61

Table 5.4: CPU time comparison with increasing the number of input variables n for the cubic polynomial vector function g.

implementation mode to use for independent input variables case one must consider the data format available, as well as take into account the complexity of the problem and available memory.

5.5 Conclusions

In this chapter we considered the details essential for the implementation of the moments method using Matlab. The generalised version of the algorithm deals with both dependent and independent types of input variables. However, it requires the manipulation of the form of the representation of the data for the independent inputs to transform it from the intuitive vector form to cell arrays. This is coupled with an increase in the memory and run time requirements. The algorithm for uncorrelated inputs is also implemented separately. It is more complicated in its implementation, but compensates in the overall computational efficiency. Therefore, if the input va-

riables are independent, this approach performs better.

The software that implements both of these techniques is available on the accompanying CD.

Chapter 6

Branch detection

So far we did not put any conditions onto the mathematical model represented by computer code in Matlab. Except, in order to be able to compute accurate derivatives, the function has to be continuously differentiable. However, in practice very often the mathematical model may contain *branches* that can cause inadequate results even though on every branch the function is still continuous. How can this happen?

Let us assume that the function $g^+(x)$ holds for all $x > 0$, while for $x < 0$ it is described by g^-, and if $x = 0$ the model triggers g^0:

$$y = \begin{cases} g^+, & \text{if } x > 0; \\ g^0, & \text{if } x = 0; \\ g^- & \text{otherwise.} \end{cases} \tag{6.1}$$

When building the Taylor series, we compute the derivatives of the function only once. However, in process the active variables may change, and so may the function. How substantial these changes are can only be decided by the model developer. Even so the reality is that the changes of the function cause the changes in Taylor coefficients, thus the expected results of the simulation are actually differ from those produced by `MADMoments` software. Moreover, the Taylor series is only valid for one interval associated with the particular branch. Thus assuming a single representation of the Taylor series for the function that contains branches results in an unreliable outcome

of the `MADMoments`. Thus when computing the expectation for the function (6.1) one must keep in mind that

$$\int_{-\infty}^{+\infty} y(t)f(t)dt = \int_{-\infty}^{0} g^{-}(t)f(t)dt + \int_{0}^{+\infty} g^{+}(t)f(t)dt. \tag{6.2}$$

It means that for computing the correct expectation of the function (6.1) Taylor series for both g^+ and g^- are required. However, using the moments method approach, as described in Chapter 4, the Taylor series is only constructed for that branch of the function the expectation of the input parameters belong to. The derivatives and Taylor coefficients are computed just once.

The ideal solution for this kind of problem would be the ability to compute the derivatives on the whole space of input variables and then call for the correct ones whenever it is required. However implementation of this technique in Matlab might only be possible using *source transformation*. It goes beyond the bounds of our work: limited to the overloaded AD implementation means we are unable to prevent branching of the code. But we still can signal to the user if there are potential risks in his model by testing it thoroughly for the presence of branches and collecting detailed data about their location in the code. Based on the information we provide he can make a decision on the importance of the branches and possibly change the model by eliminating them or run the computation several times and combine the results manually.

In this chapter of our work we define branch detection problem when dealing with computer coded models and implement it as a part of the tool for statistical moments computation.

6.1 Motivation

To motivate studying branches in the computer code, we consider several examples from different areas of applications.

Example 6.1.1 (Sparsity estimation for Jacobian). A function's ***Jacobian*** is the matrix of all first-order partial derivatives of the function of m equations $\mathbf{g} = \mathbf{g}(x_1, \ldots, x_n)$:

$$J = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix}.$$

Let us consider protopypical some function $\mathbf{g}$ that contains a branch.

```
function g = g(x1, x2, x3)
if x1 > 0
    g = [x1; x2x3]
else
    g = [x1x2; x3]
end
```

Analytically computing the Jacobian for such a function we get

$$J = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & x_3 & x_2 \end{bmatrix}, & \text{if } x_1 > 0; \\ \begin{bmatrix} x_2 & x_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, & \text{if } x_1 \leqslant 0. \end{cases} \tag{6.3}$$

Therefore, the corresponding sparsity pattern S_J for (6.3) actually also has two possible evaluations:

$$S_J = \begin{cases} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, & \text{if } x_1 > 0; \\ \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, & \text{if } x_1 \leqslant 0. \end{cases} \tag{6.4}$$

It is clear that the sparsity pattern for both branches differ from each other. Whenever one refers to the Jacobian or its sparsity matrix now, it is important to know whether or not the correct values for the triggered branch were used.

Example 6.1.2. One of the ways to implement the reverse mode for the AD requires

recording mathematical operations used to describe the function. For the function

```
function g = g(x1, x2, x3)
    if x1 > 0
        g = x1 + x2 × x3
    else
        g = x1 + x2 + x3
    end
```

there are two *tapes* of operations

$$\begin{cases} \begin{array}{c|c|c|c} \times & x_2 & x_3 & \text{v} \\ + & \text{v} & x_1 & \text{g} \end{array}, & x_1 > 0; \\ \\ \begin{array}{c|c|c|c} + & x_1 & x_2 & \text{v} \\ + & \text{v} & x_3 & \text{g} \end{array}, & x_1 \leqslant 0. \end{cases} \tag{6.5}$$

Here, the first column corresponds to the operation types, the second and third ones store the operation's arguments, and the last column contains the result of the operation.

One can see that the branch detection problem is raised not only for computating Taylor series coefficients. When disregarding information about the structure of the mathematical model, one risks analysing the results by executing the incorrect branch of the function. Needless to say, the cases when such mistakes happen can cause critical errors in application.

6.2 Dealing with branches

Summing up all of the above, one can define ***branch detection*** as a detection of data-dependent control flow.

To tackle the problem of branch detection in Matlab, we overload the main comparison functions and introduce a new global variable that is assigned for storing all

the collected information about the branches in the code. However, not every branch in the code leads to the problems with derivatives computation. Thus when overloading comparison functions, we only overload those that deal with active variables, meaning the comparison functions of the class `fmad`.

6.2.1 Global variable `MADBRANCHES`

The draw back of any overloading operation is the fact that the code and therefore the functionality of the function get modified. By overloading fundamental functions of the certain class one has to take into account the impact on the whole library. Indeed one has to respect the interface of the function, such as the number of input parameters, the type of the arguments and the returned values. Therefore, the simple mechanism that allows to alter the behaviour of the function by adding to it the required functionality is needed. When calling for function only arguments and global variables can access and influence the function return. Avoiding any changes in input arguments, the only solution is to use a global variable.

In the case of the comparison functions, we do not want them to produce the branch related information by default every time they are used. Supplementary functionality achieved via using global variables may influence the computational expense. And clearly it is not always necessary to conduct the branch detection. There are plenty of tasks for automatic differentiation tool when branching can be neglected.

The global variable, we call it `MADBRANCHES`, gathers the information required for the user to make an adequate decision about the importance of the branches in the function code. In Matlab we define `MADBRANCHES` as a structured variable that contains four fields

`.switch`

The `switch` component is of the logical data type. It can possess the value of `TRUE` or `FALSE` and its purpose is to indicate whether or not the branch

detection is required. Due to this field, the overridden functions do not actually trigger the modified areas of the code unless it is required to do so, i.e. `MADBRANCHES.switch = TRUE`.

`.new`

This field is an actual branch tape. In other words, it is a vector of values of logical type that corresponds to the comparison results: whenever the comparison holds true, the value recorded on the tape is `TRUE`, otherwise it is `FALSE`. For example,

$c = x > y$;

```
MADBRANCHES.new = [MADBRANCHES.new; c(:)];
```

This tape allows the user to analyse the behaviour of the function when hitting the branch.

`.info{:, 1:5}`

It contains all the information about the hit branches that can be collected using the Matlab instruction `dbstack('-completenames')`. `dbstack` returns `file` - the file in which the function appears, `name` - the name of the function within the file, and `line` - the line number within the function. The first parameter of `.info` corresponds to the sequence number of the branch, the second one accommodates five strings of the information: file name, function name, line number, operation type and the number of times the branch is hit.

`.old`

Whenever it is necessary to consider the results of several different function runs, this field allows us to store the previous branch tape available for comparison and analysis.

Clearly, the global variable `MADBRANCHES` does not exist by default. The function is required to determine the user's request for branch detection as well as the structure of `MADBRANCHES` preparing it for future use. We call this function `MADSetBranches`.

Branch operation type	Corresponding operation commands
BranchDetection	'on' – turns the branch detection on;
	'off' – switches the branch detection off, but does not clear the global branch variable;
	'clear' – clears the global branch variable and turns the detection off;
	'again' – records the information about the previously detected branches by copying the branch tape into MADBRANCHES.old.

Table 6.1: The combination of the branch operation type and corresponding operation command as used in MADSetBranches.

It has two inputs, branch operation type and branch operation command. Table 6.1 contains permitted values for these two inputs. The algorithm for MADSetBranches written in pseudo-code is presented here. The functioning Matlab code can be found on the CD supplied with this work. As an output it creates or removes structured MADBRANCHES.

```
function MADSetBranches(operation_type, operation_command)
    global MADBRANCHES;
    if operation_type == 'BranchDetection'
        if operation_command == 'on' then
            % creates a field to store the new branch tape
            MADBRANCHES.new = [];
            % creates a field to store the branches information
            MADBRANCHES.info = [];
            % creates a field to count the number of branches in the code
            MADBRANCHES.i = 1;
            % creates table for sorted information about branches
            MADBRANCHES.info{1,1} = 'File Name';
            MADBRANCHES.info{1,2} = 'Function Name';
```

```
        MADBRANCHES.info{1,3} = 'Line Number';
        MADBRANCHES.info{1,4} = 'Operation Type';
        MADBRANCHES.info{1,5} = 'Number of Hits';
        % activates the use of the branch detection
        MADBRANCHES.switch = true;
    end
    if operation_command == 'off' then
        % deactivates the use of the branch detection
        MADBRANCHES.switch = false;
    end
    if operation_command == 'clear' then
        % clears the global variable
        clear global MADBRANCHES;
    end
    if operation_command == 'again' then
        % copies branch tape of the previous run for further comparison
        MADBRANCHES.old = MADBRANCHES.new;
        % repeats the processes as for operation_command = 'on'
        MADSetBranches('BranchDetection','on');
    end
end
```

Although the function `MADSetBranches` has the ability to activate the branch detection process, it does not have the functionality for obtaining and recording information about the branches.

6.2.2 Overloading of comparison functions

Whenever `MADSetBranches` receives input operation type `'on'` or `'again'`, it sets up the structured global variable. Yet the structure is not filled with the information: it is empty but ready to accept the data. We now need to establish the connection between the comparison functions and variable `MADBRANCHES`.

There are six comparison functions in `fmad`.

Comparison:	==	>=	>	<=	<	~=
Function name in Matlab:	eq.m	ge.m	gt.m	le.m	lt.m	ne.m

Let us consider the equality function `eq.m` in detail and demonstrate the changes made by overriding.

```
function y = eq(a,b)
    y = getvalue(a)== getvalue(b);
    MADStoreBranches(a,b,y,'eq');
```

In this function either a, or b, or both a and b are active, i.e. have derivatives. When comparing two variables of `fmad` type, only their values are compared, not the values of their derivatives. The function `getvalue` returns the value of the variable. Function `MADStoreBranches` updates the information on branch detection stored in `MADBRANCHES`.

```
function MADStoreBranches(a,b,y,op)
    global MADBRANCHES;
    % checks if the branch detection is switched on
    if MADBRANCHES.switch
        % adds comparison outputs to the branch tape
        MADBRANCHES.new = [MADBRANCHES.new; y];
        % get information of the function call stack
        function_info = dbstack('-completenames');
        % number of branch locations detected
        i = MADBRANCHES.i;
```

```
% detector if this branch was detected before
check = FALSE;
for all j - branches detected by now
    if current branch has been seen before, i.e. matches the branch j
        % increase the number of hits for j^th branch by 1
        MADBRANCHES.info{j,5} = MADBRANCHES.info{j,5}+1;
        % note that this location has been seen before
        check = TRUE;
    end
    % if the location has not been seen before, save it as a new one
    if check == FALSE
        % update the number of branches detected in the code
        i = i + 1
        % record the branch information into the specialised fields
        MADBRANCHES.info{i,1} = function_info(3).file;
        MADBRANCHES.info{i,2} = function_info(3).name;
        MADBRANCHES.info{i,3} = function_info(3).line;
        MADBRANCHES.info{i,4} = op;
        MADBRANCHES.info{i,5} = 1;
        MADBRANCHES.i = i;
    end
end
end
```

Thus we obtain detailed information about branches which is kept as a set of values in MADBRANCHES. However, the structure of the global variable is multidimensional, and as a result not easily accessible.

6.2.3 Accessing branch data

To be able to analyse and compare the information stored in `MADBRANCHES` without getting into the details of the whole branch detection implementation a specified routine for allocating different sets of data is developed. The options for the output include

- the newly recorded branch tape - `MADBRANCHES.new`;
- previously recorded branch tapes - `MADBRANCHES.old`;
- all recorded branch tapes, new and old ones, simultaneously;
- detailed information about the locations, types and frequencies of the branches occurred in the computer coded model - `MADBRANCHES.info`.

The function to access all of these outputs is called `MADGetBranches`. It contains a single argument that indicates the type of the information required:

`'Branches'` – provides most recently recorded branch tape;

`'OldBranches'` – reproduce previously recorded branch tapes;

`'AllBranches'` – prints out all recorded branch tapes;

`'HTML'` – creates HTML page with the table containing file and function names, line numbers, types of branches and the number of their hits.

As it is clear where the branch tapes are coming from and it is easy to systematise the output, there is no command in Matlab for the straight forward presentation of the data in HTML format. Therefore, to implement that option in the routine we create the `.html` file and fill it in with the required HTML code.

6.3 Tests

To demonstrate the functionality of the branch detection algorithms implemented in Matlab, we consider here several coded functions with active variables - variables with derivatives. The branches in Matlab are usually given by the executive statements with certain conditions holding. These are `if` and `while`. We do not consider branches caused by `case` since `case` should only be used for comparing discrete and hence non-differentiable values.

Example 6.3.1. Let us consider the function that contains both `if` and `while`.

```
function y = func(a,b,c)
    v = a + b;
    if v > 2
        y = 2 * c;
    else
        y = v. * c;
    end
    while a > 1
        y = y + 1;
        a = a./2;
    end
```

We first give values to the arguments of the function. Let them be

```
>> a = 1;
>> b = 2;
>> c = -0.5;
```

The function with these inputs gets the value $y = -1$:

```
>> y = func(a,b,c)
y =
    -1
```

We now want to differentiate the function with respect to a, b, and c. In `fmad` format in Matlab the variables are then given as

```
>> a = fmad(1,[1,0,0]);
>> b = fmad(2,[0,1,0]);
>> c = fmad(-0.5,[0,0,1]);
```

When running the function for these inputs, we now can also obtain the derivatives at the specified points:

```
>> y = func(a,b,c);
>> yvalue = getvalue(y)
yvalue =
    -1
>> yderivs = getderivs(y)
yderivs(:,:,1) =
    0
yderivs(:,:,2) =
    0
yderivs(:,:,3) =
    2
```

However these are the derivatives computed with respect to the branches triggered in the code. Certainly, once we change input values the derivatives change as well. But when we are dealing with a case like constructing Taylor series, a discontinuity of the function and its derivatives results in sharp changes of the Taylor coefficients. Thus let us test the function for the purpose of detecting possibile discontinuities in the code.

```
>> MADSetBranches('BranchDetection','on');
>> y = func(a,b,c);
```

At this point we can call for data about branches in two ways.

```
>> branches = MADGetBranches('Branches')
```

```
branches =
     1
     0
```

We can see that there are two branches being hit: on the first one the condition is fulfilled, thus the value of the branch is `TRUE`, the second one does not hold and the value recorded on the tape is `FALSE`. Instead of looking through the code searching for branches we call

```
>> MADGetBranches('HTML')
```

that produces the html table as in figure 6.1. It is documented that the branches are

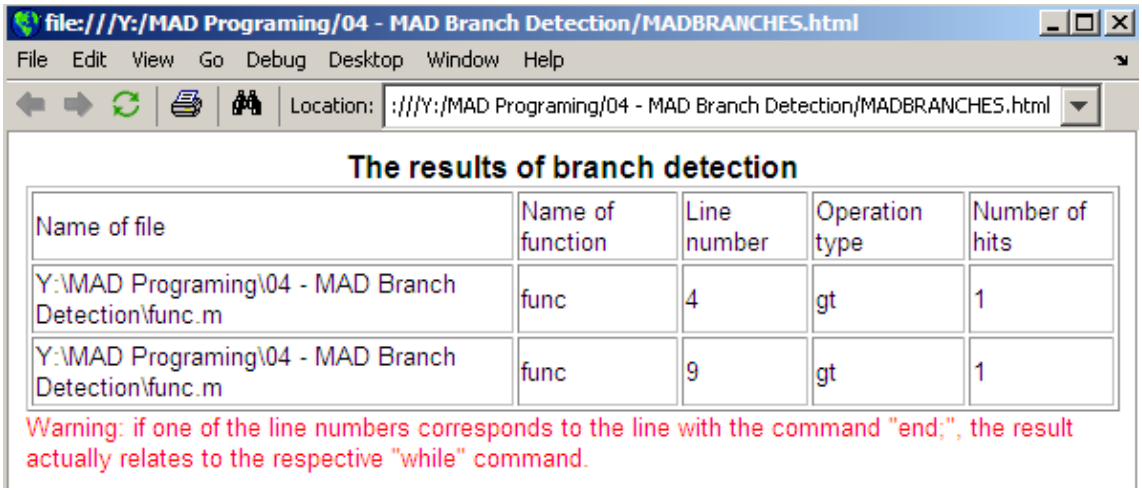


Figure 6.1: The summary of branch detection results in html form.

on the lines 4 and 9 of the function `func.m`, both of them are for the operation of comparison `gt`, greater, and are triggered only once.

The necessity of the warning on the created HTML page is caused by Matlab compilation of code containing the `while` command. Let us change the values of the input parameters:

```
>> a = fmad(12,[1,0,0]);
>> b = fmad(-11,[0,1,0]);
>> c = fmad(1,[0,0,1]);
```

The branch tape for this function differs from the previous one.

```
>> MADSetBranches('BranchDetection','again');
```

```
>> y = func(a,b,c);
>> branches = MADGetBranches('AllBranches')
branches =
    [2x1 double] [6x1 double]
```

Now the first cell array of `branches` contains the old branch tape, when the second one corresponds to the new one.

```
>> branches{1}
ans =
     1
     0
>> branches{2}
ans =
     0
     1
     1
     1
     1
     0
```

The change in number of elements in the tapes is due to the multiple hits of the `while` command this time. However, when calling for the HTML data display

```
>> MADGetBranches('HTML')
```

the three branches listed are potentially confusing (see figure 6.2). When debugging the program in Matlab, we do not get the third branch record. Instead, the number of hits on the line 9 is 5. But the compiler assigns the comparison repetitions on that line to the end of the cycle. Thus we produce the notice at the bottom of the HTML document that warns the user about this peculiarity of the Matlab compiler. In fact, to get the correct number of hits for the `while` cycle one must add together the result for the line with the `while` and the line with the corresponding `end`.

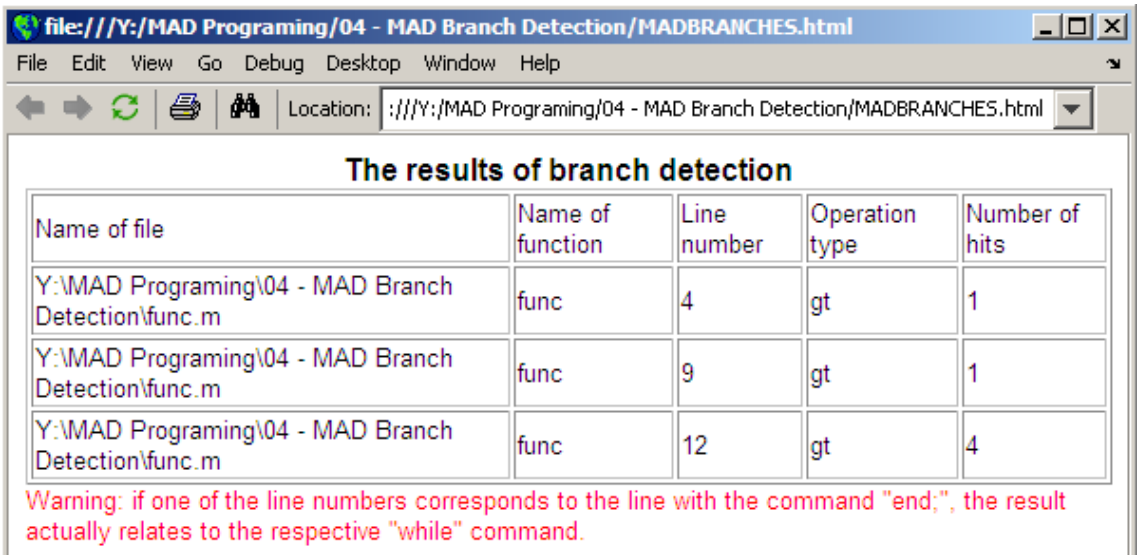


Figure 6.2: The summary of branch detection results in html form.

6.4 Branch detecting in MADMoments

We have integrated the branch detection tool into our package for computing statistical moments MADMoments. MADSetBranches, MADGetBranches and MADStoreBranches are combined in our MADBranchDetection package, and together with overloaded comparison functions are available as an update for the fmad tool, we now only need to add an extra option for the MADMoments function to activate branch detection whenever it is required and displaying the outcome of the test as a HTML document. It makes sense to limit the output options for branch detection used in MADMoments to avoid the risk of overdoing the complexity of the statistical moments package. When the users receive warning information with the necessary summary about the presence of the branches in his/her code, it is upto them to test the code using the other options provided for branch detection beyond MADMoments.

Thus, to call MADMoments one must add the combination of two extra arguments - branch operation type and corresponding operation command, as described in Table 6.1.

Example 6.4.1. Let us consider a function that contains active branch:

```
function y = Func(x1, x2, x3)
```

```
if x(3) > 0
    y = x(3)*(sin(x(1))+cos(x(2)));
else
    y = sin(x(1))-cos(x(2));
end
```

We assume the statistical independency of the input variables. When running the moments method for such function we expect different results depending on the value of x_3.

```
>> Ex = [0.1,0.2,-1]; % the expectation of the input variable x
>> SDx = [0.1,0.1,0.3]; % the standard deviation of the input variable x
>> p = 8; % the order of the Taylor series expansion
>> [Ey,Vy] = MADMoments(@Func,p,Ex,SDx,'normal',...
                        ... 'BranchDetection','on')
The branch detection is switched on.
Please, make sure you use one of on/clear/off/again commands
for every new run of the function!
Ey = -0.875842979987688
Vy = 0.010239170380712
```

The program also produces the branch report as shown in Figure 6.3.

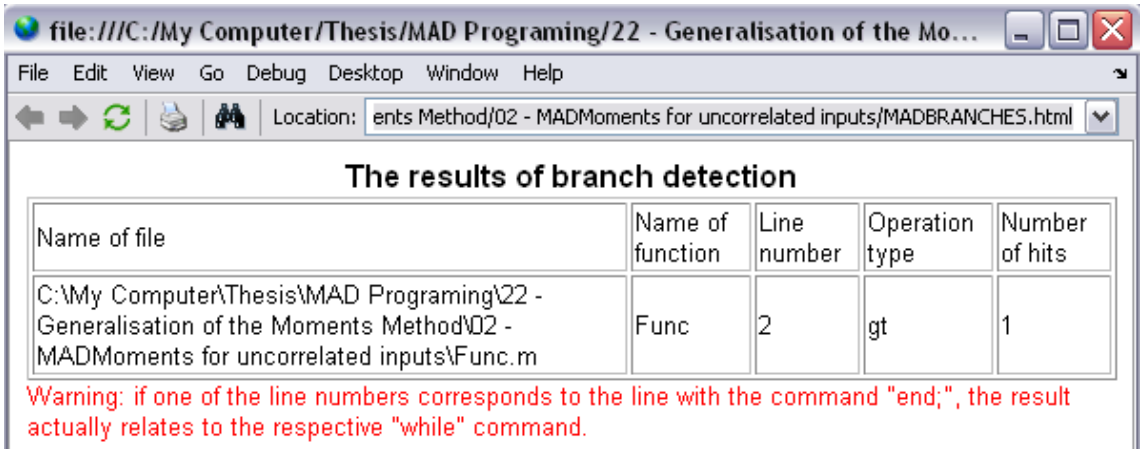


Figure 6.3: The summary of branch detection results in html form.

By tracing the branch and changing the value of x_3 accordingly

```
>> Ex = [0.1,0.2,1]; % the expectation of the input variable x
```

we obtain the results based on the Taylor series expansion for the other branch of the function:

```
>> [Ey,Vy] = MADMoments(@Func,p,Ex,SDx,'normal')
Ey = 1.074513970795767
Vy = 0.115059310430860
```

6.5 Conclusions

Branches in modeling functions can be the cause of serious errors in the final results. Without being able to detect and analyse them, the user may unknownly obtain non-reliable outcomes of the simulation.

In this chapter, we developed the tool that helps the user discover and collect the information on branches in the computer coded model that may have a potentially harmful affect on the result of the modeling. Our tool can function as a part of the moments computation package as well as on its own when required.

The developed software is available on the accompanying CD as an independent tool as well as an integrated option in the `MADMoments` package.

Chapter 7

Conclusion and future work

7.1 Summary of results

In this work we gave detailed consideration to one of the tools for the uncertainty estimation - the moments method. Due to the rapid improvements in the field of automatic differentiation, it was deemed appropriate to investigate the moments method in order to expand its capability and as a result its area of applicability. The technique is based on the use of the Taylor series of the function described by the computational model. Thus, due to the inability (or limited ability) to differentiate computer coded modelling function, the importance of the moments method has not been fully appreciated until recently.

Since the moments method implementation requires the understanding and the use of the automatic differentiation tool, we started the work by taking a closer look at the AD background and made a modest advance in this area by developing the numerical quadrature for Matlab, `quadMAD`. We have also proved that applying an AD tool to determine sensitivities with respect to the limits of integration for an arbitrary quadrature scheme gives approximations to the analytic derivatives with order of accuracy commensurate with that of the underlying quadrature provided the integrand is sufficiently differentiable. The software has been tested on the previously published example of [29] proving itself to be a more accurate tool for the task.

In Chapter 4 we tried two different approaches to automatically apply the moments method for statistical moments of order greater than two. The first, which for the variance involves squaring the function's Taylor series and then taking the expectation, gets algebraically very complicated with increasing Taylor series order. The second, in which we determine the Taylor series of the square of the function and then take the expectation, allows us to compute the moments approximations up to an arbitrary order of the Taylor expansion. Such a technique allows the user to adapt the order of approximation based on the particular properties of the function defined by the computational model.

While developing the moments method implementation in Matlab we have managed to take into account not only independent inputs for the scalar modelling function, as by previous authors [3], [4], [15], etc., but extended it in Chapter 4 for vector functions with correlated input variables: the user also has the freedom to choose the required distribution from either those that are well-known and published or those specially developed for their particular modelling problem.

As a result of the conducted work, the main constraints on the efficiency of the moments method currently are those related to the higher order derivatives computations required. Our present Matlab implementation is unable to take advantage of the symmetry of higher order derivatives, e.g. $\partial^2 g/\partial x_1 \partial x_2 = \partial^2 g/\partial x_2 \partial x_1$ as has been done using other AD tools in C++, [25]. Presently we have not taken advantage of reverse mode for cases when the number of output variables is small compared to the number of inputs. It also is unable to deal with sparsity in the derivatives' representation. Thus the CPU time required to differentiate the function upto any requested order takes up the major share of the overall running time. Depending on the size of the modelling problem, the storage of the derivatives can also excessively demand large amounts of memory. As a result our run time and memory costs are higher than strictly necessary for an AD-based technique. Nevertheless, when comparing the moments method implemented in Matlab to the most common tool for such tasks, Monte

Carlo simulation, the use of `MADMoments` is undoubtedly the way to go for small problems, those with only a handful of input variables. The examples in Chapters 4 and 5 compare the MC and moments methods. The moments method allows us to obtain accurate results as the properties of the function can be accounted for by an adequate choice of the Taylor order. The moments method outperforms Monte Carlo not only in run time, but also in terms of accuracy as well. When increasing the number of input parameters in the modelling function or the function's computational complexity, the computational expenses for both methods increases as well. Computing variance using moments method approximations for non-linear functions, can often be less efficient than Monte Carlo simulations, as MC gets similarly accurate results with less run time. However, the MC results are obtained using a random number generator, thus vary from run to run, and therefore the precision varies as well. With higher run time requirements the moments method results are still more consistent, thus more reliable. Moreover, with future development of AD tools in Matlab, we expect the moments method to decidedly break forward, ahead of MC method, since as it was already mentioned, today the derivative computation is the main run time consumer.

There are many restrictions on the use of the Taylor series that cause corresponding problems when implementing the moments method. Discontinuous functions require multiple evaluations of the Taylor coefficients which is currently not possible in Matlab. However we considered one such case, that when a function's discontinuity is represented in the computer code as a branch. Even though we are still unable to deliver the moments approximation that accounts for the changes in control flow, in Chapter 6 we have developed a tool for detecting and collecting the information about branches. This leaves it upto a user to decide whether or not the discovered branches influence expected results.

Convergence issues are considered in Chapter 4. As a tool for dealing with loss of convergence caused by imaginary poles, the partitioning approach, suggested by Christianson and Cox [16], was developed. Presently, it has only been developed for

the one-dimensional case, thus can not yet be used for general problems.

The experiments presented in this thesis can easily be repeated using the `MADMoments` tool available on the accompanying CD and the necessary guidance notes throughout the work.

Different parts of this thesis were presented at several conferences and workshops, such as Fifth, Sixth and Seventh European AD Workshops, Fifth International AD Conference and First African Conference on Computational Mechanics. We have not yet published any articles on the subjects considered in this work. However, we hope to do so shortly after the thesis submission.

7.2 Future work

While working on this thesis some issues and questions have been raised. In this section we summarise them as possible options for future work.

When the modelling function has fewer outputs than inputs, the forward mode of the automatic differentiation is no longer considered efficient, as the number of the derivatives for computing Taylor coefficients grows and consequently so do the corresponding costs. According to Neidinger [31], the cost of the forward AD mode roughly can be estimated as $3n + 1$ function evaluations, where n is the number of inputs. However, the reverse AD mode cost does not depend on the the number of inputs, but the number of outputs instead, which is sufficiently small in comparison to the inputs. Thus, one of the possible ways to improve the efficiency of the moments method for such cases would be to implement the reverse mode for derivatives computations.

We also still do not know how to choose the order of the Taylor series to obtain the desired accuracy of the approximation. Corliss et al., [32, 33], used Taylor series for solving ordinary differential equations, dealing with the singularities issues and automating the choice of Taylor order. This analysis might be adapted for use in

the moments computation. This should allow us to improve the moments method's efficiency, as one would be able to reduce the order of the derivatives to compute, and thus to reduce the computation time and memory requirements.

There are many difficulties coupled with the use of Taylor series. In this work we successfully dealt with convergence issues associated with imaginary poles. However, the partitioning approach that takes care of such problems is so far considered only for simple one-dimensional problems. Can this approach be extended to higher dimensional inputs?

Branches in computer coded modelling functions can also raise problems with the results' reliability. Even though we developed a tool that can identify and locate potential problems, the neat way to deal with branches would be the ability to compute and store Taylor coefficients for all possible changes in control flow.

Due to all the issues associated with Taylor series, the use of moments method is often questioned, as in [7, 10] for example. What are the reasonable alternatives? Would the use of Gauss-Hermite quadrature methods be more fruitful for computing the statistical moments of the function when the input variables are normally distributed, as suggested by [8]?

When developing and testing the software for computing statistical moments, we have not had access to any engineering data. Tests related to real world problems are still needed.

Bibliography

[1] Bilal M. Ayyub and Richard H. McCuen. *Probability, statistics, and reliability for engineers*. CRC Press, 1997.

[2] William L. Oberkampf, Jon C. Helton, Cliff A. Joslyn, Steven F. Wojtkiewicz, and Scott Ferson. Challenge problems: Uncertainty in system response given uncertain parameters. *Reliability Engineering and System Safety*, 85:11–19, 2004.

[3] Andy J. Keane and Prasanth B. Nair. *Computational approaches for aerospace design. The pursuit of excellence*. John Wiley and Sons, 2005.

[4] D. Ghate and M.B. Giles. *Inexpensive Monte Carlo uncertainty analysis*, pages 203–210. Tata McGraw-Hill, New Delhi, 2006.

[5] Michael C. Brooks and William R. Wise. Quantifying uncertainty due to random errors for moment analysis of breakthrough curves. *Journal of Hydrology*, 303:165–175, 2005.

[6] Bruno Sadret and Imane Cherradi. Quadrature method for finite element reliability analysis. In *9th International Conference on Applications of Statistics and Probability in Civil Engineering*, 2003.

[7] Mattia Padulo, Michele Sergio Campobasso, and Marin Dimitrov Guenov. Comparative analysis of uncertainty propagation methods for robust engineering design. In *International Conference on Engineering Design*. ICED'07, 2007.

[8] Jr Richard V. Field, Thomas L. Paez, and John R. Red-Horse. Utilizing gauss-hermit quadrature to evaluate uncertainty in dynamic system response. In *Proceedings of the 17th International Modal Analysis Conference*, volume 3727(2), pages 1856–1861, 1999.

[9] L.D. Kudryavtsev. *Encyclopedia of Mathematics*. Kluwer Academic Publishers, 2001.

[10] Mattia Padulo, Shaun A. Forth, and Marin D. Guenov. Robust aircraft conceptual design using automatic differentiation in matlab. *Lecture Notes in Computational Science and Engineering*, 64:271–280, 2008.

[11] M.M.R. Williams. Polynomial chaos functions and stochastic differential equations. *Annals of Nuclear Energy*, 33:774–785, 2006.

[12] George Stefanou. The stochastic finite element method: Past, present and future. *Comput. Methods Appl. Mech. Engrg.*, 198:1031–1051, 2009.

[13] T. Lovett, F. Ponci, and A. Monti. A polynomial chaos approach to measurement uncertainty. In *International Workshop in Advanced Methods for Uncertainty Estimation in Measurement, AMUEM 2005*, 2005.

[14] Athanasios Papoulis and S. Unnikrishna Pillai. *Probability, random variables, and stochastic processes*. McGraw-Hill, 4th edition, 2002.

[15] Michele M. Putko, Perry. A. Newman, Arthur C. Taylor III, and Lawrence. L. Green. Approach for uncertainty propagation and robust design in CFD using sensitivity derivatives. In *15th AIAA Computational Fluid Dynamics Conference*, AIAA, pages 2001–2528, Anaheim, CA, 2001.

[16] Bruce Christianson and Maurice Cox. Automatic propagation of uncertainties. In H. M. B, editor, *Automatic Differentiation: Applications, Theory, and Implemen-*

tations, Lecture Notes in Computational Science and Engineering, pages 47–58. Springer, 2005.

[17] Shaun A. Forth. An efficient overloaded implementation of forward mode automatic differentiation in MATLAB. *ACM Transactions on Mathematical Software*, 32(2):195–222, jun 2006.

[18] Bruce E. Wampold and Clifford J. Drew. *Theory and application of statistics*. McGraw-Hill, 1990.

[19] Donald R. Barr and Peter W. Zehna. *Probability: Modelling uncertainty*. Addison-Wesley, 1983.

[20] William W. Hines and Douglas C. Montgomery. *Probability and statistics in engineering and management science*. John Wiley and Sons, 3rd edition, 1990.

[21] Olkin Ingram, Leon J. Gleser, and Cyrus Derman. *Probability models and applications*. Macmillan College Publishing Company, 2nd edition, 1994.

[22] Robert A. Adams. *Calculus, a complete course*. Addison-Wesley, 4th edition, 1999.

[23] Murray R. Spiegel. *Schaum's outline of theory and problems of complex variables with an introduction to conformal mapping and its applications*. Schaum publishing company, 1964.

[24] Albert Meyer and Radhika Nagpal. Course notes 9: Permutations and combinations, 2002. http://ocw.mit.edu/NR/rdonlyres/Electrical-Engineering-and-Computer-Science/6-042JMathematics-for-Computer-ScienceFall2002/41E2F2B7-03BB-4956-9A5B-3BBCAE245073/0/ln9.pdf Last accessed on 05 August 2009.

[25] Andreas Griewank. *Evaluating derivatives. Principles and techniques of algorithmic differentiation*. SIAM, 2000.

[26] Arun Verma. An introduction to automatic differentiation. *Current Science*, 78(7):804–807, 2000.

[27] S.D. Conte and Carl de Boor. *Elementary Numerical Analysis - An Algorithmic Approach*. McGraw-Hill, 3rd edition, 1981.

[28] G.M. Phillips and P.J. Taylor. *Theory and applications on numerical analysis*. Academic Press, 1989.

[29] Robert Cudeck. Fitting psychometric models with methods based on automatic differentiation. *Psychometrica*, 70(4):599–617, 2005.

[30] Douglas G. Bonnet. Comments, conjectures and conclusions: A new algorithm for the tetrachoric correlation. *Journal of Statistical Computation and Simulation*, 76(8):737–740, 2006.

[31] Richard D. Neidinger. Introduction to automatic differentiation and Matlab object-oriented programming. Davidson College, October 15, 2009.

[32] George Corliss and Y.F. Chang. Solving ordinary differential equations using taylor series. *ACM Transactions on Mathematical Software*, 8(2):114–144, 1982.

[33] George Corliss and Gabriela Kirlinger. On implicit taylor series methods for stiff odes. In *Proceedings of SCAN'91: International Symposium on Computer Arithmetic and Scientific Computing*, 1991.

Appendix A

Calculations for expectation and variance

This section of the work includes all necessary details for understanding the resulting formulas in Chapter 4 when deriving the statistical moments using first approach. Section 4.1.1 follows well explained steps when obtaining the expectation and the variance using Taylor series of a single variable. However, the extension of the method for the Taylor series with multiple variables requires more insight.

When $\mathbf{x} \in \mathbb{R}^n$, the Taylor series expansion of the function $g(\mathbf{x})$ about $\mu_{\mathbf{x}}$ is

$$\begin{aligned} g(\mathbf{x}) &= g + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(x_i - \mu_{x_i}) + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\ &\quad + \frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k}(x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \\ &\quad + \dots \end{aligned} \tag{A.1}$$

Applying the expectation operator to (A.1)we get

$$\begin{aligned} \mu_g &= E\big(g(\mathbf{x})\big) = E\big(g\big) + \sum_{i=1}^{n} \frac{\partial g}{\partial x_i} E(x_i - \mu_{x_i}) \\ &\quad + \frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j} E\big((x_i - \mu_{x_i})(x_j - \mu_{x_j})\big) \end{aligned}$$

$$+\frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^3 g}{\partial x_i \partial x_j \partial x_k}E\big((x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\big)$$
$$+\dots \tag{A.2}$$

In Section 4.1.2 we assume statistical independence of the variables, meaning

$$E\big((x_i-\mu_{x_i})(x_j-\mu_{x_j})\big)=\begin{cases}\sigma^2_{x_i}, & \text{if i = j,}\\ E(x_i-\mu_{x_i})E(x_j-\mu_{x_j})=0, & \text{otherwise,}\end{cases} \tag{A.3}$$

thus taking into account that $E(x_i-\mu_{x_i})=0$, (A.2) becomes

$$\mu_g = g+\frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma^2_{x_i}+\frac{1}{3!}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma^3_{x_i}+\dots \tag{A.4}$$

A.1 First approach

We are now trying to compute variance using the 2^{nd} order moments method according to Ghate and Giles, [4], thus only the 2^{nd} order expectation approximation is required:

$$\mu_g = g+\frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma^2_{x_i}+O(\sigma^3_{\mathbf{x}}). \tag{A.5}$$

Taking into account that the variance is $\sigma^2_g = E(g^2)-E(g)^2$, we need to square the expectation approximation as well as Taylor series of the function g before applying the expectation operator to it.

The squared approximation of (A.5) is

$$\begin{aligned}E(g)^2 &= g^2+g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma^2_{x_i}+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\sigma^4_{x_i}\\ &\quad+\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}\sigma^2_{x_i}\sigma^2_{x_j}.\end{aligned} \tag{A.6}$$

When squaring the Taylor series of 2^{nd} order for function g we get

$$\begin{aligned} g^2(\mathbf{x}) &= g^2(\mu_{\mathbf{x}}) + \left(\sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(x_i - \mu_{x_i})\right)^2 \\ &\quad + \left(\frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right)^2 \\ &\quad + 2 \times g(\mu_{\mathbf{x}}) \sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(x_i - \mu_{x_i}) \\ &\quad + 2 \times \frac{1}{2!} g \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\ &\quad + 2 \times \left(\sum_{i=1}^{n} \frac{\partial g}{\partial x_i}(x_i - \mu_{x_i})\right) \\ &\quad\quad \times \left(\frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right). \end{aligned} \tag{A.7}$$

We now apply the expectation operator to (A.7) simplifying some of the notations on the way due to the assumed input variables independency (A.3). Thus,

$$\begin{aligned} E(g^2) &= E\big(g^2(\mu_{\mathbf{x}})\big) + \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i}\right)^2 E\big((x_i - \mu_{x_i})^2\big) \\ &\quad + \left(\frac{1}{2!}\right)^2 \sum_{i=1}^{n} \left(\frac{\partial^2 g}{\partial x_i^2}\right)^2 E\big((x_i - \mu_{x_i})^4\big) \\ &\quad + \left(\frac{1}{2!}\right)^2 \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2} E\big((x_i - \mu_{x_i})^2 (x_j - \mu_{x_j})^2\big) \\ &\quad + 2 \times \left(\frac{1}{2!}\right)^2 \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2 E\big((x_i - \mu_{x_i})^2 (x_j - \mu_{x_j})^2\big) \\ &\quad + 2 \times \frac{1}{2!} g \sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i^2} E\big((x_i - \mu_{x_i})^2\big) \\ &\quad + 2 \times \frac{1}{2!} \sum_{i=1}^{n} \frac{\partial^2 g}{\partial x_i^2}\frac{\partial g}{\partial x_i} E\big((x_i - \mu_{x_i})^3\big) \\ &= g^2(\mu_{\mathbf{x}}) + \sum_{i=1}^{n} \left(\frac{\partial g}{\partial x_i}\right)^2 \sigma_{x_i}^2 + \frac{1}{4}\sum_{i=1}^{n} \left(\frac{\partial^2 g}{\partial x_i^2}\right)^2 K(x_i)\sigma_{x_i}^4 \end{aligned}$$

$$+\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i\partial x_j}\right)^2\sigma_{x_i}^2\sigma_{x_j}^2$$
$$+g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2+\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial g}{\partial x_i}S(x_i)\sigma_{x_i}^3 \tag{A.8}$$

By subtracting (A.6) from (A.8) we get the 2^{nd} order moments method for the variance:

$$\begin{aligned}\sigma_g^2 &= E(g^2)-E(g)^2\\ &= \sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2\sigma_{x_i}^2+\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial g}{\partial x_i}S(x_i)\sigma_{x_i}^3\\ &\quad+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i\partial x_j}\right)^2\sigma_{x_i}^2\sigma_{x_j}^2\\ &\quad+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\big(K(x_i)-1\big)\sigma_{x_i}^4\end{aligned} \tag{A.9}$$

Assuming that $\mathbf{x}\in\mathbb{R}$ it is easy to see that both (A.5) and (A.9) are transformed into the 2^{nd} order moments method for single variable case (4.6) and (4.9), respectively.

Analogously, we now obtain the 3^{rd} order moments method. For expectation it is

$$\mu_g = g+\frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2+\frac{1}{3!}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^3+O(\sigma_{\mathbf{x}}^4) \tag{A.10}$$

as follows from (A.4). To get $E(g)^2$ we square (A.10):

$$\begin{aligned}E(g)^2 &= g^2(\mu_{\mathbf{x}})+\left(\frac{1}{2!}\right)^2\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\sigma_{x_i}^4\\ &\quad+\left(\frac{1}{2!}\right)^2\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2\\ &\quad+\left(\frac{1}{3!}\right)^2\sum_{i=1}^{n}\left(\frac{\partial^3 g}{\partial x_i^3}\right)^2S(x_i)^2\sigma_{x_i}^6\\ &\quad+\left(\frac{1}{3!}\right)^2\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^3 g}{\partial x_i^3}\frac{\partial^3 g}{\partial x_j^3}S(x_i)S(x_j)\sigma_{x_i}^3\sigma_{x_j}^3\end{aligned}$$

$$
\begin{aligned}
&+g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2+2\times\frac{1}{3!}g\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^3\\
&+\frac{1}{3!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^5\\
&+\frac{1}{3!}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^3 g}{\partial x_j^3}S(x_j)\sigma_{x_i}^2\sigma_{x_j}^3. \qquad \text{(A.11)}
\end{aligned}
$$

The squared Taylor series of order 3 is

$$
\begin{aligned}
g^2(\mathbf{x}) \;=\;& g^2(\mu_{\mathbf{x}})+\left(\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}(x_i-\mu_{x_i})\right)^2\\
&+\left(\frac{1}{2!}\right)^2\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 g}{\partial x_i\partial x_j}(x_i-\mu_{x_i})(x_j-\mu_{x_j})\right)^2\\
&+\left(\frac{1}{3!}\right)^2\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_k}(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\right)^2\\
&+2\times\frac{1}{2!}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 g}{\partial x_i\partial x_j}(x_i-\mu_{x_i})(x_j-\mu_{x_j})\right)\;\times\\
&\quad\times\frac{1}{3!}\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_k}(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\right)\\
&+2\times g\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}(x_i-\mu_{x_i})\\
&+2\times\frac{1}{2!}g\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 g}{\partial x_i\partial x_j}(x_i-\mu_{x_i})(x_j-\mu_{x_j})\\
&+2\times\frac{1}{3!}g\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_k}(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\\
&+2\times\frac{1}{2!}\left(\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}(x_i-\mu_{x_i})\right)\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\partial^2 g}{\partial x_i\partial x_j}(x_i-\mu_{x_i})(x_j-\mu_{x_j})\right)\\
&+2\times\frac{1}{3!}\left(\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}(x_i-\mu_{x_i})\right)\;\times\\
&\quad\times\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_k}(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\right) \qquad \text{(A.12)}
\end{aligned}
$$

As in (A.8), we again assume statistical independence of the input variables and apply

expectation operator to (A.12) simplifying some of the notations straight away. For simplicity we also neglect terms of order higher than $\sigma_{\mathbf{x}}^4$. Therefore,

$$
\begin{aligned}
E(g^2) &= g^2(\mathbf{x}) + \sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2 E\big((x_i-\mu_{x_i})^2\big) \\
&\quad + \left(\frac{1}{2!}\right)^2 \sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2 E\big((x_i-\mu_{x_i})^4\big) \\
&\quad + 2\times\left(\frac{1}{2!}\right)^2 \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2 E\big((x_i-\mu_{x_i})^2(x_j-\mu_{x_j})^2\big) \\
&\quad + \left(\frac{1}{2!}\right)^2 \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2} E\big((x_i-\mu_{x_i})^2(x_j-\mu_{x_j})^2\big) \\
&\quad + g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2} E\big((x_i-\mu_{x_i})^2\big) + \frac{1}{3} g\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3} E\big((x_i-\mu_{x_i})^3\big) \\
&\quad + \sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2} E\big((x_i-\mu_{x_i})^3\big) + \frac{1}{3}\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i^3} E\big((x_i-\mu_{x_i})^4\big) \\
&\quad + 2\times 3\frac{1}{3!}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i \partial x_j^2} E\big((x_i-\mu_{x_i})^2(x_j-\mu_{x_j})^2\big) \\
&= g^2(\mathbf{x}) + \sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2 \sigma_{x_i}^2 + \frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2 K(x_i)\sigma_{x_i}^4 \\
&\quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2 \sigma_{x_i}^2\sigma_{x_j}^2 \\
&\quad + \frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&\quad + g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2 + \frac{1}{3} g\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3} S(x_i)\sigma_{x_i}^3 \\
&\quad + \sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2} S(x_i)\sigma_{x_i}^3 + \frac{1}{3}\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i^3} K(x_i)\sigma_{x_i}^4 \\
&\quad + \sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2. \qquad \text{(A.13)}
\end{aligned}
$$

After neglecting $O(\sigma_{\mathbf{x}}^5)$ and higher order terms in (A.11) and subtracting it from (A.13)

the approximation of variance takes the form

$$\begin{aligned}\tilde{\sigma}_g^2 &= \sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2\sigma_{x_i}^2+\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2}S(x_i)\sigma_{x_i}^3\\ &+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i\partial x_j}\right)^2\sigma_{x_i}^2\sigma_{x_j}^2+\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2\\ &+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\big(K(x_i)-1\big)\sigma_{x_i}^4+\frac{1}{3}\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i^3}K(x_i)\sigma_{x_i}^4. \qquad \text{(A.14)}\end{aligned}$$

Again, assuming $\mathbf{x}\in\mathbb{R}$ the equation (A.14) transforms into (4.26). The comparison of (A.9) and (A.14) also reveals several missing terms of order $O(\sigma_{\mathbf{x}}^4)$

$$|\sigma_g^2-\tilde{\sigma}_g^2|=\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{3}\sum_{i=1}^{n}\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_i^3}K(x_i)\sigma_{x_i}^4. \qquad \text{(A.15)}$$

However, in case when the function $g(\mathbf{x})$ is in fact a vector function $\mathbf{y}=g(\mathbf{x})$, its second statistical moment is defined as the covariance

$$C_y=\begin{pmatrix}\sigma_{g_1}^2 & C_{g_1g_2} & \dots & C_{g_1g_m}\\ C_{g_2g_1} & \sigma_{g_2}^2 & \dots & C_{g_2g_m}\\ \vdots & \vdots & \ddots & \vdots\\ C_{g_mg_1} & C_{g_mg_2} & \dots & \sigma_{g_m}^2\end{pmatrix}, \qquad \text{(A.16)}$$

where $C_{g_pg_q}=E(g_pg_q)-E(g_p)E(g_q)$.

Using second order Taylor approximation for g_p and g_q, we now compute the

product of these functions:

$$
\begin{aligned}
g_p g_q \;=\;& g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}}) + g_p \sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}(x_i - \mu_{x_i}) \\
&+ \frac{1}{2!} g_p \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\
&+ g_q \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i}) + \left(\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{j=1}^{n} \frac{\partial g_q}{\partial x_j}(x_j - \mu_{x_j})\right)
\end{aligned}
$$

$$
\begin{aligned}
&+ \frac{1}{2!}\left(\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^2 g_q}{\partial x_j \partial x_k}(x_j - \mu_{x_j})(x_k - \mu_{x_k})\right) \\
&+ \frac{1}{2!} g_q \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\
&+ \frac{1}{2!}\left(\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^2 g_p}{\partial x_j \partial x_k}(x_j - \mu_{x_j})(x_k - \mu_{x_k})\right) \\
&+ \left(\frac{1}{2!}\right)^2 \left(\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right) \times \\
&\qquad \times \left(\sum_{k=1}^{n}\sum_{l=1}^{n} \frac{\partial^2 g_q}{\partial x_k \partial x_l}(x_k - \mu_{x_k})(x_l - \mu_{x_l})\right). \qquad \text{(A.17)}
\end{aligned}
$$

Applying the expectation operator to (A.17) and taking into account the independence of input variables condition (A.3), we get

$$
\begin{aligned}
E(g_p g_q) \;=\;& g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}}) + \frac{1}{2} g_p \sum_{i=1}^{n} \frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2 \\
&+ \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2} S(x_i)\sigma_{x_i}^3 \\
&+ \frac{1}{2} g_q \sum_{i=1}^{n} \frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}\frac{\partial^2 g_p}{\partial x_i^2} S(x_i)\sigma_{x_i}^3 \\
&+ \frac{1}{4}\sum_{i=1}^{n} \frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2} K(x_i)\sigma_{x_i}^4 + \frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\partial^2 g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&+ 2 \times \frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2. \qquad \text{(A.18)}
\end{aligned}
$$

Taking into account that

$$\begin{aligned} E(g_p)E(g_q) &= g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}}) + \frac{1}{2}g_p\sum_{i=1}^{n}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2 + \frac{1}{2}g_q\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2 \\ &\quad + \frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^4 + \frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2, \end{aligned} \tag{A.19}$$

where $E(g)$ is the approximation based on the 2^{nd} order Taylor series of the function g, the 2^{nd} order moments method for the covariance is

$$\begin{aligned} C_{g_p g_q} &= \sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2}S(x_i)\sigma_{x_i}^3 \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^2 g_p}{\partial x_i^2}S(x_i)\sigma_{x_i}^3 + \frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\big(K(x_i)-1\big)\sigma_{x_i}^4 \\ &\quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2. \end{aligned} \tag{A.20}$$

It is easy to see that when $p = q$, (A.20) is equivalent to the 2^{nd} order moments method for the variance (A.9).

To demonstrate the error in the term of the 4^{th} order, we also compute the cova-

riance based on the 3^{rd} order Taylor series. First of all,

$$
\begin{aligned}
g_p g_q \;=\;& g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}}) + g_p \sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}(x_i - \mu_{x_i}) \\
&+ \frac{1}{2!} g_p \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_q}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j}) \\
&+ \frac{1}{3!} g_p \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^3 g_q}{\partial x_i \partial x_j \partial x_k}(x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \\
&+ g_q \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i}) + \left(\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}(x_i - \mu_{x_i})\right) \\
&+ \frac{1}{2!}\left(\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_q}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right) \\
&+ \frac{1}{3!}\left(\sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} \frac{\partial^3 g_q}{\partial x_i \partial x_j \partial x_k}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right) \\
&+ \frac{1}{2!} g_q \left(\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right) \\
&+ \frac{1}{2!}\left(\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i}(x_i - \mu_{x_i})\right)\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right) \\
&+ \left(\frac{1}{2!}\right)^2 \left(\sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j}(x_i - \mu_{x_i})(x_j - \mu_{x_j})\right) \times
\end{aligned}
\tag{A.21}
$$

$$
\begin{aligned}
&\times \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 g_q}{\partial x_i \partial x_j} (x_i - \mu_{x_i})(x_j - \mu_{x_j}) \right) \\
&+ \frac{1}{2!} \frac{1}{3!} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 g_p}{\partial x_i \partial x_j} (x_i - \mu_{x_i})(x_j - \mu_{x_j}) \right) \times \\
&\quad \times \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^3 g_q}{\partial x_i \partial x_j \partial x_k} (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \right) \\
&+ \frac{1}{3!} g_q \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_k} (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \right) \\
&+ \frac{1}{3!} \left(\sum_{i=1}^{n} \frac{\partial g_q}{\partial x_i} (x_i - \mu_{x_i}) \right) \times \\
&\quad \times \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_k} (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \right) \\
&+ \frac{1}{2!} \frac{1}{3!} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial^2 g_q}{\partial x_i \partial x_j} (x_i - \mu_{x_i})(x_j - \mu_{x_j}) \right) \times \\
&\quad \times \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_k} (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \right) \\
&\left(\frac{1}{3!} \right)^2 \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_k} (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \right) \times \\
&\quad \times \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} \frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_k} (x_i - \mu_{x_i})(x_j - \mu_{x_j})(x_k - \mu_{x_k}) \right). \qquad \text{(A.22)}
\end{aligned}
$$

When calculating the expectation of (A.22) under the condition of the independent inputs (A.3), we also omit the terms that are higher than the order 4 to avoid the formula looking too bulky, as we do not require those terms for our task anyway:

$$
\begin{aligned}
\tilde{E}(g_p g_q) \;=\; & g_p(\mu_{\mathbf{x}}) g_q(\mu_{\mathbf{x}}) + \frac{1}{2!} g_p \sum_{i=1}^{n} \frac{\partial^2 g_q}{\partial x_i^2} \sigma_{x_i}^2 \\
& + \frac{1}{3!} g_p \sum_{i=1}^{n} \frac{\partial^3 g_q}{\partial x_i^3} S(x_i) \sigma_{x_i}^3 + \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i} \frac{\partial g_q}{\partial x_j} \sigma_{x_i}^2 \\
& + \frac{1}{2!} \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i} \frac{\partial^2 g_q}{\partial x_i^2} S(x_i) \sigma_{x_i}^3 + \frac{1}{3!} \sum_{i=1}^{n} \frac{\partial g_p}{\partial x_i} \frac{\partial^3 g_q}{\partial x_i^3} K(x_i) \sigma_{x_i}^4 \\
& + 3 \times \frac{1}{3!} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \frac{\partial g_p}{\partial x_i} \frac{\partial^3 g_q}{\partial x_i \partial x_j^2} \sigma_{x_i}^2 \sigma_{x_j}^2 + \frac{1}{2!} g_q \sum_{i=1}^{n} \frac{\partial^2 g_p}{\partial x_i^2} \sigma_{x_i}^2
\end{aligned}
$$

$$
\begin{aligned}
&+\frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial g_q}{\partial x_i}S(x_i)\sigma_{x_i}^3+\frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}K(x_i)\sigma_{x_i}^4\\
&+\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+2\times\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i\partial x_j}\frac{\partial^2 g_q}{\partial x_i\partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2\\
&+\frac{1}{3!}g_q\sum_{i=1}^{n}\frac{\partial^3 g_p}{\partial x_i^3}S(x_i)\sigma_{x_i}^3+\frac{1}{3!}\sum_{i=1}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i^3}K(x_i)\sigma_{x_i}^4\\
&+3\times\frac{1}{3!}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2. \qquad \text{(A.23)}
\end{aligned}
$$

The product of the expectation approximations of the third order for two different functions g_p and g_q after dismissing the terms of order > 4 is

$$
\begin{aligned}
\tilde{E}(g_p)\tilde{E}(g_q) &= g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}})+\frac{1}{2!}g_p\sum_{i=1}^{n}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2+\frac{1}{3!}g_p\sum_{i=1}^{n}\frac{\partial^3 g_q}{\partial x_i^3}\sigma_{x_i}^3S(x_i)\sigma_{x_i}^3\\
&\quad+\frac{1}{2!}g_q\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2+\left(\frac{1}{2!}\right)^2\left(\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2\right)\left(\sum_{i=1}^{n}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2\right)\\
&\quad+\frac{1}{3!}g_q\sum_{i=1}^{n}\frac{\partial^3 g_p}{\partial x_i^3}S(x_i)\sigma_{x_i}^3\\
&= g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}})+\frac{1}{2}g_p\sum_{i=1}^{n}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2+\frac{1}{2}g_q\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2\\
&\quad+\frac{1}{6}g_p\sum_{i=1}^{n}\frac{\partial^3 g_q}{\partial x_i^3}S(x_i)\sigma_{x_i}^3+\frac{1}{6}g_q\sum_{i=1}^{n}\frac{\partial^3 g_p}{\partial x_i^3}S(x_i)\sigma_{x_i}^3\\
&\quad+\frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^4+\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2. \qquad \text{(A.24)}
\end{aligned}
$$

The third order moments method for the variance is then

$$
\begin{aligned}
\tilde{C}_{q_p g_q} &= \sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2\\
&\quad+\frac{1}{2}\sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2}S(x_i)\sigma_{x_i}^3+\frac{1}{2}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial g_q}{\partial x_i}S(x_i)\sigma_{x_i}^3\\
&\quad+\frac{1}{6}\sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3}K(x_i)\sigma_{x_i}^4+\frac{1}{6}\sum_{i=1}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i^3}K(x_i)\sigma_{x_i}^4
\end{aligned}
$$

$$+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2$$
$$+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\big(K(x_i)-1\big)\sigma_{x_i}^4. \quad \text{(A.25)}$$

Again, when $p = q$ the covariance moments method (A.25) is transformed into the moments method for the variance (A.14).

The error in the fourth order term after comparing the second and the third order moments methods for the covariance is

$$\begin{aligned}|C_{g_p g_q}-\tilde{C}_{g_p g_q}| \;=\;& \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i \partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2\\ &+\frac{1}{6}\sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3}K(x_i)\sigma_{x_i}^4+\frac{1}{6}\sum_{i=1}^{n}\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i^3}K(x_i)\sigma_{x_i}^4. \quad \text{(A.26)}\end{aligned}$$

In the similar manner one can show that the moments method based on 4^{th} order Taylor series produces the approximation of the variance with the 2^{nd} order of accuracy.

A.2 Second approach

The idea of this technique differs from the first approach as we do not square Taylor series any longer. This time we square the function itself. In other words, instead of squaring Taylor approximation for the function g to obtain the approximation of the function g^2, we expand g^2 using Taylor series.

Let us assume that $y(\mathbf{x}) = g^2(\mathbf{x})$. Therefore, Taylor approximation of (A.1) for g^2 can be written as following

$$y(\mathbf{x}) \;=\; g^2(\mathbf{x}) = g^2(\mu_{\mathbf{x}})+\sum_{i=1}^{n}2g\frac{\partial g}{\partial x_i}(x_i-\mu_{x_i})$$

$$
\begin{aligned}
&+\frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n}2\left(\frac{\partial g}{\partial x_i}\frac{\partial g}{\partial x_j}+g\frac{\partial^2 g}{\partial x_i\partial x_j}\right)(x_i-\mu_{x_i})(x_j-\mu_{x_j})\\
&+\frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}2\left(\frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_j\partial x_k}+\frac{\partial g}{\partial x_j}\frac{\partial^2 g}{\partial x_i\partial x_k}\right.\\
&\qquad\left.+\frac{\partial g}{\partial x_k}\frac{\partial^2 g}{\partial x_i\partial x_j}+g\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_k}\right)\times\\
&\qquad\times(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\\
&+\frac{1}{4!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}2\left(\frac{\partial^3 g}{\partial x_i\partial x_k\partial x_l}\frac{\partial g}{\partial x_j}+\frac{\partial^2 g}{\partial x_i\partial x_k}\frac{\partial^2 g}{\partial x_j\partial x_l}\right.\\
&\qquad+\frac{\partial^2 g}{\partial x_i\partial x_l}\frac{\partial^2 g}{\partial x_j\partial x_k}+\frac{\partial g}{\partial x_i}\frac{\partial^3 g}{\partial x_j\partial x_k\partial x_l}\\
&\qquad+\frac{\partial^2 g}{\partial x_k\partial x_l}\frac{\partial^2 g}{\partial x_i\partial x_j}+\frac{\partial g}{\partial x_k}\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_l}\\
&\qquad\left.+\frac{\partial g}{\partial x_l}\frac{\partial^3 g}{\partial x_i\partial x_j\partial x_k}+g\frac{\partial^4 g}{\partial x_i\partial x_j\partial x_k\partial x_l}\right)\times\\
&\qquad\times(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})(x_l-\mu_{x_l})+\ldots
\end{aligned}
\tag{A.27}
$$

Applying expectation operator to (A.27) under the assumption of statistical independency we get $E(g^2)$

$$
\begin{aligned}
E(g^2) &= g^2(\mu_{\mathbf{x}})+\frac{1}{2!}\sum_{i=1}^{n}2\left(\left(\frac{\partial g}{\partial x_i}\right)^2+g\frac{\partial^2 g}{\partial x_i^2}\right)E\left((x_i-\mu_{x_i})^2\right)\\
&\quad+\frac{1}{3!}\sum_{i=1}^{n}2\times\left(3\frac{\partial g}{\partial x_i}\frac{\partial^2 g}{\partial x_i^2}+g\frac{\partial^3 g}{\partial x_i^3}\right)E\left((x_i-\mu_{x_i})^3\right)\\
&\quad+\frac{1}{4!}\sum_{i=1}^{n}2\times\left(4\frac{\partial^3 g}{\partial x_i^3}\frac{\partial g}{\partial x_i}+3\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2+g\frac{\partial^4 g}{\partial x_i^4}\right)E\left((x_i-\mu_{x_i})^4\right)\\
&\quad+\frac{1}{4!}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}2\times 3\left(2\frac{\partial^3 g}{\partial x_i\partial x_j^2}\frac{\partial g}{\partial x_i}+2\frac{\partial^3 g}{\partial x_i^2\partial x_j}\frac{\partial g}{\partial x_j}+2\left(\frac{\partial^2 g}{\partial x_i\partial x_j}\right)^2\right.\\
&\qquad\left.+\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}+g\frac{\partial^4 g}{\partial x_i^2\partial x_j^2}\right)\times\\
&\qquad\times E\left((x_i-\mu_{x_i})^2(x_j-\mu_{x_j})^2\right)\\
&= g^2(\mu_{\mathbf{x}})+\sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2\sigma_{x_i}^2+g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial g}{\partial x_i}S(x_i)\sigma_{x_i}^3+\frac{1}{3}g\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^3\\
&+\frac{1}{3}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}\frac{\partial g}{\partial x_i}K(x_i)\sigma_{x_i}^4+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2K(x_i)\sigma_{x_i}^4\\
&+\frac{1}{12}g\sum_{i=1}^{n}\frac{\partial^4 g}{\partial x_i^4}K(x_i)\sigma_{x_i}^4+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\partial^3 g}{\partial x_i\partial x_j^2}\frac{\partial g}{\partial x_i}\sigma_{x_i}^2\sigma_{x_j}^2\\
&+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\partial^3 g}{\partial x_i^2\partial x_j}\frac{\partial g}{\partial x_j}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i\partial x_j}\right)^2\sigma_{x_i}^2\sigma_{x_j}^2\\
&+\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{4}g\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\partial^4 g}{\partial x_i^2\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2. \qquad \text{(A.28)}
\end{aligned}
$$

To obtain the variance we also require squared expectation approximation based on 4^{th} order Taylor series:

$$
\begin{aligned}
\mu_g^2 \;=\; E(g)^2=\Bigg[&g(\mu_{\mathbf{x}})+\frac{1}{2!}\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2+\frac{1}{3!}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^3\\
&+\frac{1}{4!}\sum_{i=1}^{n}\frac{\partial^4 g}{\partial x_i^4}K(x_i)\sigma_{x_i}^4+\frac{1}{4!}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}3\frac{\partial^4 g}{\partial x_i^2\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2\Bigg]^2. \qquad \text{(A.29)}
\end{aligned}
$$

Dismissing terms of greater than order 4 the squared expectation approximation becomes

$$
\begin{aligned}
E(g)^2 \;=\; &g^2+g\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\sigma_{x_i}^2+\frac{1}{3}g\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}S(x_i)\sigma_{x_i}^3\\
&+\frac{1}{4}\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial^2 g}{\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{4}g\sum_{i=1}^{n}\sum_{\substack{j=1\\j\neq i}}^{n}\frac{\partial^4 g}{\partial x_i^2\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2\\
&+\frac{1}{12}g\sum_{i=1}^{n}\frac{\partial^4 g}{\partial x_i^4}K(x_i)\sigma_{x_i}^4+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\sigma_{x_i}^4. \qquad \text{(A.30)}
\end{aligned}
$$

Thus, the 4^{th} order variance approximation is

$$
\sigma_g^2 \;=\; E(g^2)-E(g)^2=\sum_{i=1}^{n}\left(\frac{\partial g}{\partial x_i}\right)^2\sigma_{x_i}^2+\sum_{i=1}^{n}\frac{\partial^2 g}{\partial x_i^2}\frac{\partial g}{\partial x_i}S(x_i)\sigma_{x_i}^3
$$

$$+\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^3 g}{\partial x_i \partial x_j^2}\frac{\partial g}{\partial x_i}\sigma_{x_i}^2\sigma_{x_j}^2+\frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial^2 g}{\partial x_i \partial x_j}\right)^2\sigma_{x_i}^2\sigma_{x_j}^2$$
$$+\frac{1}{3}\sum_{i=1}^{n}\frac{\partial^3 g}{\partial x_i^3}\frac{\partial g}{\partial x_i}K(x_i)\sigma_{x_i}^4+\frac{1}{4}\sum_{i=1}^{n}\left(\frac{\partial^2 g}{\partial x_i^2}\right)^2\big(K(x_i)-1\big)\sigma_{x_i}^4. \qquad (A.31)$$

Since there are no derivatives of the fourth order involved in (A.31), we expect the approximation (A.14) obtained via the first approach using the third order Taylor series to be complete upto the order 4. By comparing them it is easy to see that this condition holds as expected:

$$|\sigma_g^2-\tilde{\sigma_g^2}|=0. \qquad (A.32)$$

Similarly as before, we now compute the fourth order moments method for the covariance when $\mathbf{y}(\mathbf{x})=\big(g_1(\mathbf{x}),\ldots,g_m(\mathbf{x})\big)$. Thus the Taylor series for the function $g_p(\mathbf{x})g_q(\mathbf{x})$ around $\mu_{\mathbf{x}}$ is

$$\begin{aligned}
g_p(\mathbf{x})g_q(\mathbf{x}) \;=\;& g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}})+\sum_{i=1}^{n}\left(g_p\frac{\partial g_q}{\partial x_i}+g_q\frac{\partial g_p}{\partial x_i}\right)(x_i-\mu_{x_i})\\
&+\frac{1}{2!}\sum_{i=1}^{n}\sum_{j=1}^{n}\left(g_p\frac{\partial^2 g_q}{\partial x_i\partial x_j}+\frac{\partial g_p}{\partial x_j}\frac{\partial g_q}{\partial x_i}+\frac{\partial g_q}{\partial x_j}\frac{\partial g_p}{\partial x_i}+g_q\frac{\partial^2 g_p}{\partial x_i\partial x_j}\right)\times\\
&\qquad\times(x_i-\mu_{x_i})(x_j-\mu_{x_j})\\
&+\frac{1}{3!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\left(g_p\frac{\partial^3 g_q}{\partial x_i\partial x_j\partial x_k}+\frac{\partial g_p}{\partial x_k}\frac{\partial^2 g_q}{\partial x_i\partial x_j}+\frac{\partial g_p}{\partial x_j}\frac{\partial^2 g_q}{\partial x_i\partial x_k}\right.\\
&\qquad+\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_j\partial x_k}+\frac{\partial g_q}{\partial x_i}\frac{\partial^2 g_p}{\partial x_j\partial x_k}\\
&\qquad\left.+\frac{\partial g_q}{\partial x_j}\frac{\partial^2 g_p}{\partial x_i\partial x_k}+\frac{\partial g_q}{\partial x_k}\frac{\partial^2 g_p}{\partial x_i\partial x_j}+g_q\frac{\partial^3 g_p}{\partial x_i\partial x_j\partial x_k}\right)\times\\
&\qquad\times(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})\\
&+\frac{1}{4!}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{l=1}^{n}\left(g_p\frac{\partial^4 g_q}{\partial x_i\partial x_j\partial x_k\partial x_l}+\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_j\partial x_k\partial x_l}\right.\\
&\qquad+\frac{\partial g_p}{\partial x_j}\frac{\partial^3 g_q}{\partial x_i\partial x_k\partial x_l}+\frac{\partial g_p}{\partial x_k}\frac{\partial^3 g_q}{\partial x_i\partial x_j\partial x_l}\\
&\qquad\left.+\frac{\partial g_p}{\partial x_l}\frac{\partial^3 g_q}{\partial x_i\partial x_j\partial x_k}+\frac{\partial^2 g_p}{\partial x_i\partial x_j}\frac{\partial^2 g_q}{\partial x_k\partial x_l}\right.
\end{aligned}$$

$$
\begin{aligned}
&+\frac{\partial^2 g_p}{\partial x_i \partial x_k}\frac{\partial^2 g_q}{\partial x_j \partial x_l}+\frac{\partial^2 g_p}{\partial x_i \partial x_l}\frac{\partial^2 g_q}{\partial x_j \partial x_k}\\
&+\frac{\partial^2 g_p}{\partial x_j \partial x_k}\frac{\partial^2 g_q}{\partial x_i \partial x_l}+\frac{\partial^2 g_p}{\partial x_j \partial x_l}\frac{\partial^2 g_q}{\partial x_i \partial x_k}\\
&+\frac{\partial^2 g_p}{\partial x_k \partial x_l}\frac{\partial^2 g_q}{\partial x_i \partial x_j}+\frac{\partial^3 g_p}{\partial x_j \partial x_k \partial x_l}\frac{\partial g_q}{\partial x_i}\\
&+\frac{\partial^3 g_p}{\partial x_i \partial x_k \partial x_l}\frac{\partial g_q}{\partial x_j}+\frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_l}\frac{\partial g_q}{\partial x_k}\\
&+\frac{\partial^3 g_p}{\partial x_i \partial x_j \partial x_k}\frac{\partial g_q}{\partial x_l}+g_q\frac{\partial^4 g_p}{\partial x_i \partial x_j \partial x_k \partial x_l}\Bigg)\times\\
&\times(x_i-\mu_{x_i})(x_j-\mu_{x_j})(x_k-\mu_{x_k})(x_l-\mu_{x_l}). \qquad \text{(A.33)}
\end{aligned}
$$

The expectation of (A.33) under the assumption of the statistical independency for the input variables (A.3) is

$$
\begin{aligned}
E(g_p g_q) \;=\;& g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}})+\frac{1}{2!}\sum_{i=1}^{n}\left(g_p\frac{\partial^2 g_q}{\partial x_i^2}+2\frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}+g_q\frac{\partial^2 g_p}{\partial x_i^2}\right)\sigma_{x_i}^2\\
&+\frac{1}{3!}\sum_{i=1}^{n}\left(g_p\frac{\partial^3 g_q}{\partial x_i^3}+3\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2}+3\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial g_q}{\partial x_i}+g_q\frac{\partial^3 g_p}{\partial x_i^3}\right)S(x_i)\sigma_{x_i}^3\\
&+\frac{1}{4!}\sum_{i=1}^{n}\left(g_p\frac{\partial^4 g_q}{\partial x_i^4}+4\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3}+6\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\right.\\
&\qquad\left.+4\frac{\partial g_q}{\partial x_i}\frac{\partial^3 g_p}{\partial x_i^3}+g_q\frac{\partial^4 g_p}{\partial x_i^4}\right)K(x_i)\sigma_{x_i}^4\\
&+3\times\frac{1}{4!}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(g_p\frac{\partial^4 g_q}{\partial x_i^2 \partial x_j^2}+2\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i \partial x_j^2}+2\frac{\partial g_p}{\partial x_j}\frac{\partial^3 g_q}{\partial x_i^2 \partial x_j}\right.\\
&\qquad+\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_j^2}+4\frac{\partial^2 g_p}{\partial x_i \partial x_j}\frac{\partial^2 g_q}{\partial x_i \partial x_j}+\frac{\partial^2 g_p}{\partial x_j^2}\frac{\partial^2 g_q}{\partial x_i^2}\\
&\qquad\left.+2\frac{\partial^3 g_p}{\partial x_i \partial x_j^2}\frac{\partial g_q}{\partial x_i}+2\frac{\partial^3 g_p}{\partial x_i^2 \partial x_j}\frac{\partial g_q}{\partial x_j}+g_q\frac{\partial^4 g_p}{\partial x_i^2 \partial x_j^2}\right)\times\\
&\times\sigma_{x_i}^2\sigma_{x_j}^2. \qquad \text{(A.34)}
\end{aligned}
$$

When multiplying the expectation approximation of the 4^{th} order for the function g_p by the one of the same order for g_q and neglecting the terms of order higher than 4,

we get

$$
\begin{aligned}
E(g_p)E(g_q) &= g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}}) + \frac{1}{2!}g_p\sum_{i=1}^{n}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2 + \frac{1}{3!}g_p\sum_{i=1}^{n}\frac{\partial^3 g_q}{\partial x_i^3}S(x_i)\sigma_{x_i}^3 \\
&\quad + \frac{1}{4!}g_p\sum_{i=1}^{n}\frac{\partial^4 g_q}{\partial x_i^4}K(x_i)\sigma_{x_i}^4 + 3\times\frac{1}{4!}g_p\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\frac{\partial^4 g_q}{\partial x_i^2\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&\quad + \frac{1}{2!}g_q\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2 + \left(\frac{1}{2!}\right)\left(\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\sigma_{x_i}^2\right)\left(\sum_{i=1}^{n}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^2\right) \\
&\quad + \frac{1}{3!}g_q\sum_{i=1}^{n}\frac{\partial^3 g_p}{\partial x_i^3}S(x_i)\sigma_{x_i}^3 + \frac{1}{4!}g_q\sum_{i=1}^{n}\frac{\partial^4 g_p}{\partial x_i^4}K(x_i)\sigma_{x_i}^4 \\
&\quad + 3\times\frac{1}{4!}g_q\sum_{i=1}^{n}\frac{\partial^4 g_p}{\partial x_i^2\partial x_j^2}\sigma_{x_i}^2\sigma_{x_j}^2 \\
&= g_p(\mu_{\mathbf{x}})g_q(\mu_{\mathbf{x}}) + \frac{1}{2!}\sum_{i=1}^{n}\left(g_p\frac{\partial^2 g_q}{\partial x_i^2} + g_q\frac{\partial^2 g_p}{\partial x_i^2}\right)\sigma_{x_i}^2 \\
&\quad + \frac{1}{3!}\sum_{i=1}^{n}\left(g_p\frac{\partial^3 g_q}{\partial x_i^3} + g_q\frac{\partial^3 g_p}{\partial x_i^3}\right)S(x_i)\sigma_{x_i}^3 \\
&\quad + \frac{1}{4!}\sum_{i=1}^{n}\left(g_p\frac{\partial^4 g_q}{\partial x_i^4} + g_q\frac{\partial^4 g_p}{\partial x_i^4}\right)K(x_i)\sigma_{x_i}^4 + \frac{1}{4!}\sum_{i=1}^{n}6\times\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}\sigma_{x_i}^4 \\
&\quad + 3\times\frac{1}{4!}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(g_p\frac{\partial^4 g_q}{\partial x_i^2\partial x_j^2} + 2\times\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_j^2} + g_q\frac{\partial^4 g_p}{\partial x_i^2\partial x_j^2}\right)\times \\
&\qquad \times\sigma_{x_i}^2\sigma_{x_j}^2. \qquad\qquad (A.35)
\end{aligned}
$$

The fourth order moments method for the covariance is then

$$
\begin{aligned}
C_{g_pg_q} &= E(g_pg_q) - E(g_p)E(g_q) \\
&= \sum_{i=1}^{n}\frac{\partial g_p}{\partial x_i}\frac{\partial g_q}{\partial x_i}\sigma_{x_i}^2 + \frac{1}{2}\sum_{i=1}^{n}\left(\frac{\partial g_p}{\partial x_i}\frac{\partial^2 g_q}{\partial x_i^2} + \frac{\partial^2 g_q}{\partial x_i^2}\frac{\partial g_q}{\partial x_i}\right)S(x_i)\sigma_{x_i}^3 \\
&\quad + \frac{1}{6}\sum_{i=1}^{n}\left(\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i^3} + \frac{\partial^3 g_p}{\partial x_i^3}\frac{\partial g_q}{\partial x_i}\right)K(x_i)\sigma_{x_i}^4 \\
&\quad + \frac{1}{4}\sum_{i=1}^{n}\frac{\partial^2 g_p}{\partial x_i^2}\frac{\partial^2 g_q}{\partial x_i^2}(K(x_i)-1)\sigma_{x_i}^4 \\
&\quad + \frac{1}{2}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}\left(\frac{\partial g_p}{\partial x_i}\frac{\partial^3 g_q}{\partial x_i\partial x_j^2} + \frac{\partial^2 g_p}{\partial x_i\partial x_j}\frac{\partial^2 g_q}{\partial x_i\partial x_j} +\right.
\end{aligned}
$$

$$+ \frac{\partial^3 g_p}{\partial x_i \partial x_j^2} \frac{\partial g_q}{\partial x_i} \Bigg) \sigma_{x_i}^2 \sigma_{x_j}^2 . \tag{A.36}$$