Some Improvements and Applications of Non-intrusive Polynomial Chaos Expansions

Jonathan Feinberg

Thesis submitted for the degree of Philosophiae Doctor
Department of Mathematics
Faculty of Mathematics and Natural Sciences
University of Oslo
Date July 20, 2015

Series of dissertations submitted to the
Faculty of Mathematics and Natural Sciences, University of Oslo
No. 1657

ISSN 1501-7710

Cover: Hanne Baadsgaard Utigard.
Printed in Norway: AIT Oslo AS.

Produced in co-operation with Akademika Publishing.
The thesis is produced by Akademika Publishing merely in connection with the thesis defence. Kindly direct all inquiries regarding the thesis to the copyright holder or the unit which grants the doctorate.

Preface

This thesis is submitted in partial fulfilment of the requirements for the degree of Philosophiae Doctor, to the Department of Mathematics, Faculty of Mathematics and Natural Sciences, University of Oslo. All research has been conducted at the Center for Biomedical Computing, at Simula Research Laboratory, during the period between Febuary 2011 and May 2015. I would like to express my gratitude to Simula and Statoil for excellent work condition and financial support.

There are also many individuals involved that I like to thank. Firstly we have Prof Arne Bang Huseby and Dr Stuart Clark from my advisory committee. You have both given excellent help with all my work and this thesis would not be possible without your expertise.

From Norwegian University of Science and Technology, I want to thank Vinzenz Eck, Prof Leif Rune Hellevik and Prof Victorian Prot. Our collaboration have been rewarding both in enjoyment and in academical output.

Additionally, I also want to thank Margrethe Gustavsen, Lynn Feinberg, Steven Moffat, Yapi Donatien Achou and Karoline Hagane. You have all been very helpful, each in your own way.

Lastly, I like to give special thanks to my advisor and mentor Prof Hans Petter Langtangen. You have been pivotal for my reaserch thorughout my doctorate, and your enthusiasm for the sciences and hard work is truely inspirational.

Jonathan Feinberg

Contents

List of Papers

This thesis consists of an introduction and the following three papers:

Paper I
A Novel Method for Sensitivity Quantification of Timing and Amplitude of Pressure and Flow Waves in the Arterial System

Published in *International Journal for Numerical Methods in Biomedical Engineering*, 2015. This paper is co-authored with Vinzenz Eck, Hans Petter Langtangen and Leif Rune Hellevik.

Paper II
Chaospy: An Open Source Tool for Designing Methods of Uncertainty Quantification

Submitted to *Journal of Computational Science*, 2015. This paper is co-authored with Hans Petter Langtangen.

Paper III
Multivariate Polynomial Chaos Expansions with Dependent Variables

Submitted to *SIAM Journal on Scientific Computing*, 2015. This paper is co-authored with Hans Petter Langtangen.

Introduction

The fundamental idea behind modelling is to try to create a mathematical description of some state-of-nature. Often the state-of-nature is governed by a set of equations given by some underlying physical process. Our goal is then to solve these equations for a quantity of interest to be able to reproduce the state-of-nature mathematically. However, along the way various errors are often introduced to the model. The size of those errors are then key to determining how correct the model is. We start by discussing what these various errors are.

Solving the governing equations will for more complicated problems not be a trivial task. There is simply not enough information to completly describe the nature we are modelling. Because of this partial lack of knowledge, assumptions have to be made to compensate in order to make the problem solvable. This will inevitably create a discrepancy between the state-of-nature and the model we use to describe it. We denote this discrepency as *model selection error*.

Another aspect of the problem with solving the governing equations is that it may be too complicated to be solved analytically. Assumptions are added for practical reasons to make it more feasible to solve the governing equations. One approach to compensate for this, is to solve the governing equations using a numerical solver. For most problems however, introducing a numerical solver, will effectively replace the analytical model solution with a numerical approximation. We denote the added discrepancy introduced by using a numerical solver as *numerical errors*.

Modelling a state-of-nature often ends up being a two-fold problem. So far we have described the part called the *forward problem*: We create a response from governing equations, either analytically or through a numerical solver, to predict the state-of-nature. However, the governing equations often have model parameters. To be able to create values from the response, we have to identify those parameters first. This is known as the *inverse problem*. We solve the inverse problem through the use of physical measurements: We create an objective function that quantifies the distance between the model response and the physical measurements. We then select the model parameters that minimize the objective function.

A common source of error when solving the inverse problem stems from the measurements. In some cases the measurements gathered only describe the behavior of the model indirectly. And in some cases model evaluations have to be inferred through added assumptions. In other cases the measurement

equipment is inaccurate. In any case, the result is *measuring errors*.

As a result of the various sources of errors, it is likely that the optimal model response created is not the correct one. Moreover, there are many combinations of the model parameters that will result in a feasable model response. This motivates us to abandon the deterministic assumption that each model parameter only has one single fixed value. Instead we can assume that the model parameters are unknown and have a probability density function. The density function describes the relative likelihood for a random variable to take a given value.

Given that we assume that the model parameters should have probability density function assigned to them, the next step is to determine what those densities should be. Determining model parameter densities can be done by either reformulating the inverse problem to identify the densities or by eliciting expert opinions. For insight into the probabilistic inverse problem, see the excellent book by Idier [1]. And for insight into expert elicitation, see the excellent book by O'Hagen et al. [2]. In this thesis however, the probability density functions are assumed to be known.

Given that the model parameters are assumed random, it follows that solving the governing equations will result in a random response. In other words, there exists a probability density function that describes the model response. With this density function available we have an approach for estimating the uncertainty in model response. And here lies the reason for performing uncertainty quantification: We can calculate statistical metrics for the response. For example, we can find out how far the response will deviate from a best estimate through the standard deviation operator, i.e. how stable is the response. Other statistical metrics include expected value, correlation, confidence intervals, (variance based) sensitivity to mention a few.

Problems that have random components are denoted as *probabilistic*. In contrast we can denote problems that are not probabilistic as *deterministic*. Most numerical solvers are constructed under a deterministic assumption and the model parameters are assumed fixed for the problem to be solvable. To account for uncertainty a deterministic numerical solver must be replaced with a probabilistic one. One such approach is the non-intrusive polynomial chaos expansion method [3]. This method is the foundation for this thesis. In the next section the polynomial chaos expansion method will be disussed in more detail.

2.1 Method Description

Let us consider a stochastic computational mechanics problem on the form

$$\mathcal{L}(\boldsymbol{x}, t, u, \boldsymbol{\eta}, \boldsymbol{\xi}) = 0, \tag{2.1}$$

where $\mathcal{L}$ is a possibly non-linear differential operator, $\boldsymbol{x}$ are spatial coordinates in a domain $\Omega \subseteq \mathbb{R}^3$, and $t \in \mathbb{R}$ is time. Let $u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})$ be the solution of (2.1). Appropriate boundary and initial conditions are assumed. Further, let $\boldsymbol{\eta} = (\eta_0, \ldots, \eta_{R-1})^T \in \mathbb{R}^R$ be all parameters known accurately without uncertainty and in contrast let $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_D)^T \in \mathbb{R}^D$ be the parameters which contain uncertainty. The uncertainty in $\boldsymbol{\xi}$ can sufficiently be described through a joint probability density function $p_{\boldsymbol{\xi}}$, which is assumed to be known. From this problem formulation, we define the objective to quantify the uncertainty in the solution u, given uncertainty in $\boldsymbol{\xi}$. More specifically, we want to compute the statistical properties of u, or in a goal functional involving u, at various space or time locations. A typical goal functional of interest are integrals of a flux or stress vector at parts of the boundary, but here we shall, with no lack of generality, assume that the interest is in the values of u itself at all space-time locations.

Ultimately, we seek the joint probability distribution p_u at all space-time points, but this is more information than we need to digest in practice. We therefore aim to address the following two questions: What is the expected value and the variance of u at a space-time point, given the uncertainty of $\boldsymbol{\xi}$? And how accurately can we compute these statistical measures? Mathematically, we aim to compute the response mean and variance:

$$\begin{aligned}
\mathbb{E}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})] &= \int_{\mathbb{R}^D} u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi}) p_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi} \\
\mathrm{Var}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})] &= \int_{\mathbb{R}^D} \left(u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi}) - \ \mathbb{E}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})]\right)^2 p_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi}.
\end{aligned} \tag{2.2}$$

Except for trivial examples, the complexity of deriving u is high and a numerical approach must be assumed. This implies that the deterministic solution of (2.1) is only possible for known values of $\boldsymbol{\xi}$. However, since $\boldsymbol{\xi}$ is assumed uncertain and can only be described through a probability density function, any single deterministic solution of the equation will be insufficient to represent u. We will assume that a numerical solution is numerically costly, so the number of such solutions should be kept low. We therefore need an approach that properly quantifies the uncertainty of u under the constraint that (2.1) can only be solved numerically, and only a few model evaluations are allowed.

The most popular approach to perform numerical uncertainty analysis is the Monte Carlo method [4]. The method can be summarized as follows: Create independent identically distributed pseudo-random samples $Q = \{Q_k\}_{I_K}$ whith $I_K = \{0, \ldots, K\}$ from the density $p_{\boldsymbol{\xi}}$, and calculate the mean and variance from

empirical measures:

$$\mathbb{E}[u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi})] \approx \widehat{\mathbb{E}}[u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi})] = \frac{1}{K+1}\sum_{k\in I_K} u(\boldsymbol{x},t;\boldsymbol{\eta},Q_k)$$

$$\mathrm{Var}[u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi})] \approx \widehat{\mathrm{Var}}[u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi})] = \frac{1}{K}\sum_{k\in I_K}\left(u(\boldsymbol{x},t;\boldsymbol{\eta},Q_k) - \widehat{\mathbb{E}}[u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi})]\right)^2.$$

The method's strengths include that it considers the simulator as a black box and assumes nothing about the shape of u, and that it scales very well with the number of uncertain parameters D. On the other hand, it requires a very large number of samples to converge. The method has an error which is in the order of magnitude of $\mathrm{Var}[u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi})]/(K+1)$. Since we have assumed that the cost of running the simulator for solving (2.1) is very high, it is seldom feasible to perform Monte Carlo integration on computational mechanics problems.

As a more efficient alternative to the Monte Carlo method, we have the polynomial approximation method. This method will be discussed next. Let us make the fundamental assumption that u is a smooth function of the random input parameters $\boldsymbol{\xi}$. Then we can effectively approximate u as a function of $\boldsymbol{\xi}$ through a polynomial approximation:

$$u(\boldsymbol{x},t;\boldsymbol{\eta},\boldsymbol{\xi}) \approx u_M(\boldsymbol{x},t;\boldsymbol{\xi}) = \sum_{n\in I_N} c_n(\boldsymbol{x},t)\boldsymbol{\Phi}_n(\boldsymbol{\xi}) \qquad I_N = \{0,\ldots,N\}, \tag{2.3}$$

where M is the polynomial order, $N+1$ is the number of evaluations, $\{c_n\}_{n\in I_N}$ are coefficients, and $\{\boldsymbol{\Phi}_n\}_{n\in I_N}$ are polynomials. The mean and variance can then be calculated cheaply either analytically or by performing Monte Carlo simulation on $\hat{u}$ as a proxy for u. At this point we must emphasize that even though we assume u to depend smoothly on $\boldsymbol{\xi}$, u does not need to vary smoothly with $\boldsymbol{x}$ and t. For example, with the deterministic solution of (2.1) for a given value of $\boldsymbol{\xi}$ can be assume to contain discontinuities. The polynomial approximation method only require smoothness in the mapping from $\boldsymbol{\xi}$ to u.

There are several polynomial approximation methods available. Both Taylor polynomials and Lagrange polynomials are examples. For our purpose, let us first just assume that the polynomials $\{\boldsymbol{\Phi}_n\}_{n\in I_N}$ are selected as fixed simple monomials. An example of monomials can be found in Table 2.1.

With the polynomials known and fixed, the coefficients $\{c_n(\boldsymbol{x},t)\}_{n\in I_N}$ must be calculated in such a way that u_M approximates u as well as possible. Though this is preferably done analytically, because of the limitations in u, this has to be estimated numerically as well:

$$u_M(\boldsymbol{x},t;\boldsymbol{\xi}) \approx \hat{u}_M(\boldsymbol{x},t;\boldsymbol{\xi}) = \sum_{n\in I_N} \hat{c}_n(\boldsymbol{x},t)\boldsymbol{\Phi}_n(\boldsymbol{\xi}), \tag{2.4}$$

where each $\hat{c}_n$ is an approximation of c_n.

One all-purpose method for estimating the coefficients $\{c_n(\boldsymbol{x},t)\}_{n\in I_N}$ is the *point collocation method* (PCM). It is defined as follows. We first select

Order	Index	Polynomial
0	$\boldsymbol{\Phi}_0$	1
1	$\boldsymbol{\Phi}_1 \quad \boldsymbol{\Phi}_2$	$\xi_0 \quad \xi_1$
2	$\boldsymbol{\Phi}_3 \quad \boldsymbol{\Phi}_4 \quad \boldsymbol{\Phi}_5$	$\xi_0^2 \quad \xi_0\xi_1 \quad \xi_1^2$
3	$\boldsymbol{\Phi}_6 \quad \boldsymbol{\Phi}_7 \quad \boldsymbol{\Phi}_8 \quad \boldsymbol{\Phi}_9$	$\xi_0^3 \quad \xi_0^2\xi_1 \quad \xi_0\xi_1^2 \quad \xi_1^3$
4	$\boldsymbol{\Phi}_{10} \quad \boldsymbol{\Phi}_{11} \quad \boldsymbol{\Phi}_{12} \quad \boldsymbol{\Phi}_{13} \quad \boldsymbol{\Phi}_{14}$	$\xi_0^4 \quad \xi_0^3\xi_1 \quad \xi_0^2\xi_1^2 \quad \xi_0\xi_1^3 \quad \xi_1^4$
$\vdots$	$\vdots$	$\vdots$

Table 2.1: Simple monic polynomial expansion sorted reversed and ordered lexicographically.

collocation nodes $\boldsymbol{Q} = \{\boldsymbol{Q}_k\}_{k\in I_K}$. The nodes can be selected somewhat arbitrarily, but will here be selected to be random samples drawn from $p_{\boldsymbol{\xi}}$. Given the samples, we evaluate the polynomials and model solver:

$$P(\boldsymbol{Q}) = \begin{bmatrix} \boldsymbol{\Phi}_0(\boldsymbol{Q}_0) & \cdots & \boldsymbol{\Phi}_N(\boldsymbol{Q}_0) \\ \vdots & & \vdots \\ \boldsymbol{\Phi}_0(\boldsymbol{Q}_K) & \cdots & \boldsymbol{\Phi}_N(\boldsymbol{Q}_K) \end{bmatrix} \qquad U(\boldsymbol{x}, t, \boldsymbol{Q}) = \begin{bmatrix} u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{Q}_0) \\ \vdots \\ u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{Q}_K) \end{bmatrix}.$$

The coefficients can be estimated using linear least squares minimization:

$$\hat{\boldsymbol{c}}(\boldsymbol{x}, t) = (\hat{c}_0(\boldsymbol{x}, t), \ldots, \hat{c}_N(\boldsymbol{x}, t))^T = \left(P(\boldsymbol{Q})^T P(\boldsymbol{Q})\right)^{-1} P(\boldsymbol{Q})^T U(\boldsymbol{x}, t, \boldsymbol{Q}).$$

Since u is evaluated numerically on a space-time mesh, $\hat{c}$ can be calculated for all or a subset of all mesh points. Alternatively, U may involve the values of one or more goal functionals of u.

Let us present a simple example to demonstrate the computation of the polynomial coefficients. We look at a decaying process in time, described by

$$u(t; c, I) = I \exp(-ct).$$

Let $\boldsymbol{\xi} = (c, I)^T$ be stochastically independent uncertain parameters with uniform distributions

$$c \sim \text{Uni}(0, 0.1) \qquad\qquad I \sim \text{Uni}(8, 10).$$

A relevant interval for t is then $[0, 10]$. Let the polynomials be defined as in Table 2.1. To determine the vector U and the matrix P, we evaluated the simulator u and the polynomials $\{\boldsymbol{\Phi}_n\}_{n\in I_N}$ respectively at $K+1$ collocation nodes $\{\boldsymbol{Q}_k\}_{k\in I_K}$ in the probability space spanned by the uncertain parameters $\boldsymbol{\xi} = (c, I)$. The coefficients $\hat{c}$ are thereafter computed by the linear least squares formula. Following the suggestion of Hosder [5] we select $K + 1 = 2\,(N + 1)$.

We may compare the price-performance of the polynomial approximation and the Monte Carlo integration (MCI) method. More precisely, we investigate

the integrated errors in the mean and variance,

$$\varepsilon_{\text{mean}} = \int_0^{10} |\mathbb{E}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})] - \mathbb{E}[\hat{u}(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})]|$$
$$\varepsilon_{\text{variance}} = \int_0^{10} |\text{Var}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})] - \text{Var}[\hat{u}(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})]| \, \mathrm{d}t,$$

where the exact mean and variance can be calculated analytically:

$$\mathbb{E}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})] = 90\frac{1 - e^{-0.1t}}{t}$$
$$\text{Var}[u(\boldsymbol{x}, t; \boldsymbol{\eta}, \boldsymbol{\xi})] = 1220\frac{1 - e^{-0.2t}}{3t} - \left(90\frac{1 - e^{-0.1t}}{t}\right)^2.$$

Figure 2.1 shows the error in mean and variance as a function of the number of samples for both Monte Carlo integration and the approximation using monic polynomials. (Note the "the number of samples" is the number of collocation nodes in the polynomial approximation.) Since Monte Carlo integration is a method based on random samples, an average of 10^4 re-runs is used instead of a single "unstable" Monte Carlo solution.

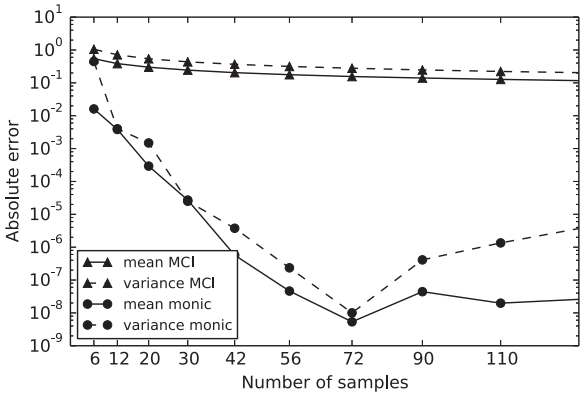

Figure 2.1: A comparison of the discrepancy between Monte Carlo integration (MCI) and point collocation method using monic polynomials (monic) in mean and variance.

The figure shows that the polynomial approximation converges much faster than Monte Carlo integration. However, as the order of the polynomial goes up, the convergence stops and starts to diverge. The method is not numerically stable.

One way to increase stability of convergence, is to use a particular polynomial approximation variant called polynomial chaos expansion. Here we exchange the

polynomials in Table 2.1 with polynomials orthogonal on a custom weighted function space $L^2_{\boldsymbol{\xi}}$. Let $L^2_{\boldsymbol{\xi}}$ be defined with inner product and norm defined with respect to the probability density $p_{\boldsymbol{\xi}}$:

$$\langle f, g \rangle = \mathbb{E}[f(\boldsymbol{\xi})g(\boldsymbol{\xi})] \qquad\qquad \|h\|_{\boldsymbol{\xi}} = \sqrt{\langle h, h \rangle_{\boldsymbol{\xi}}}.$$

We can then construct polynomials $\boldsymbol{\Phi}$ to be orthogonal:

$$\langle \boldsymbol{\Phi}_i, \boldsymbol{\Phi}_j \rangle_{\boldsymbol{\xi}} = 0 \qquad\qquad i \neq j. \tag{2.5}$$

The orthogonality gives us a few beneficial properties [3]. First, a linear least squares method using the above defined norm, leads to a diagonal linear system that can be solved analytically:

$$c_i(\boldsymbol{x}, t) = \frac{\langle u(\boldsymbol{x}, t, \boldsymbol{\eta}, \boldsymbol{\xi}), \boldsymbol{\Phi}_i(\boldsymbol{\xi}) \rangle_{\boldsymbol{\xi}}}{\|\boldsymbol{\Phi}_i(\boldsymbol{\xi})\|^2_{\boldsymbol{\xi}}}.$$

This is nothing but the well-known formula for the Fourier coefficients of generalized Fourier expansions of u. The formula implies that the c_i values become untangled: calculating a coefficient can be done irrespectively of the other coefficients or the size of the expansion. Together with the assumption that $\boldsymbol{\Phi}_0 = 1$ the orthogonality of the polynomials also imply that the estimation is simplified:

$$\mathbb{E}[\hat{u}_M(\boldsymbol{x}, t; \boldsymbol{\xi})] = c_0 \qquad\qquad \mathrm{Var}[\hat{u}_M(\boldsymbol{x}, t; \boldsymbol{\xi})] = \sum_{i=1}^{N} c_i^2 \|\boldsymbol{\Phi}_i\|^2_{\boldsymbol{\xi}}.$$

In practice, the orthogonality ensures numerical stability and the halt in convergence seen in Figure 2.1 is removed. We illustrate this fact in Figure 2.2.

For a given probability density p_ξ we need to find the associated orthogonal polynomials. For some standard distributions, the so-called Wiener-Askey scheme [6] provides the relevant polynomials. If the distribution does not exist in the Wiener-Askey scheme the expansion can be constructed analytically from methods like Gram-Schmidt orthogonalization. From a numerical perspective, however, any method that creates orthogonal polynomials from raw moments, is inherently ill-posed [7]. By raw moments we mean any expected value on the form:

$$\mathbb{E}\Big[\xi_1^{k_1} \cdots \xi_D^{k_D}\Big] = \int_{\mathbb{R}^D} \xi_1^{k_1} \cdots \xi_D^{k_D} p_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \, \mathrm{d}\boldsymbol{\xi},$$

where $(k_1, \cdots, k_D)$ are non-negative integers. Small changes in input, may in such ill-posed problems create large changes in the computed output. Therefore, orthogonal polynomials are in practice created from an alternative method known as discretized Stieltjes' method. Stieltjes' original method in one

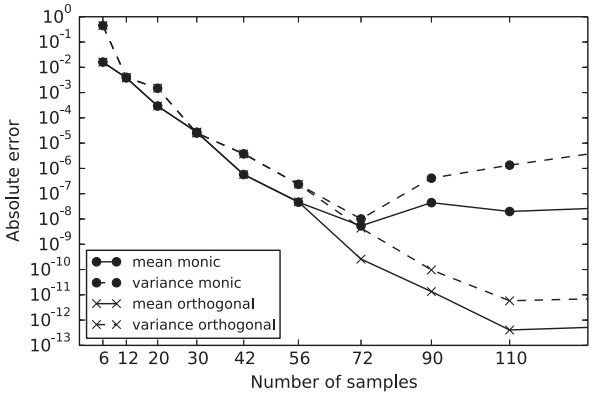

Figure 2.2: The error when approximating using monic (monic) and orthogonal (orthogonal) polynomials.

dimension is to construct orthogonal polynomials from a three terms recurrence (TTR) formula:

$$\Phi_{-1} = 0 \qquad \Phi_0 = 1 \qquad \Phi_{i+1} = (\xi - A_i)\Phi_i - B_i\Phi_{i-1}$$

where A_i and B_i are known coefficients defined

$$A_i = \frac{\langle \xi\Phi_i, \Phi_i \rangle_\xi}{\langle \Phi_i, \Phi_i \rangle_\xi} \qquad B_i = \frac{\langle \Phi_i, \Phi_i \rangle_\xi}{\langle \Phi_{i-1}, \Phi_{i-1} \rangle_\xi},$$

where $B_0 = \langle \Phi_0, \Phi_0 \rangle_\xi$. Stieltjes' method becomes discretized when the coefficients are estimated numerically. To extend the method into the multivariate case, we use a tensor product rule for combining one-dimensional polynomial expansions. For the polynomial to maintain the orthogonallity property, the method assumes that the random variables are stochastically independent. This may not seem as a strong restriction in computational mechanics as various input parameters are usually statistically independent.

With orthogonal polynomials, we can compute c_i directly from the formula, though with a numerical approximation of the integral:

$$\begin{aligned} c_i = \frac{\langle u, \boldsymbol{\Phi}_i \rangle_{\boldsymbol{\xi}}}{\|\boldsymbol{\Phi}_i\|^2_{\boldsymbol{\xi}}} &= \frac{1}{\|\boldsymbol{\Phi}_i\|^2_{\boldsymbol{\xi}}} \int_{\mathbb{R}^D} u(x,t;\boldsymbol{\eta},\boldsymbol{\xi})\boldsymbol{\Phi}_i(\boldsymbol{\xi}) p_{\boldsymbol{\xi}}(\boldsymbol{\xi})\, \mathrm{d}\boldsymbol{\xi} \\ &\approx \frac{1}{\|\boldsymbol{\Phi}_i\|^2_{\boldsymbol{\xi}}} \sum_{k=0}^{K} u(x,t;\boldsymbol{\eta},\boldsymbol{Q}_k)\boldsymbol{\Phi}_i(\boldsymbol{Q}_k) p_{\boldsymbol{\xi}}(\boldsymbol{Q}_k) W_k. \end{aligned} \tag{2.6}$$

Here, $\boldsymbol{Q}_k$ and W_k are respectively quadrature nodes and weights. The norm $\|\boldsymbol{\Phi}_i\|^2_{\boldsymbol{\xi}}$ is often numerically difficult to calculate and is instead estimated using

the numerically stable recurrence coefficients:

$$\begin{aligned}\|\Phi_i\|_\xi^2 = \langle \Phi_i, \Phi_i \rangle_\xi^2 &= \frac{\langle \Phi_i, \Phi_i \rangle_\xi}{\langle \Phi_{i-1}, \Phi_{i-1} \rangle_\xi} \langle \Phi_{i-1}, \Phi_{i-1} \rangle_\xi \\ &= \frac{\langle \Phi_i, \Phi_i \rangle_\xi}{\langle \Phi_{i-1}, \Phi_{i-1} \rangle_\xi} \cdots \frac{\langle \Phi_1, \Phi_1 \rangle_\xi}{\langle \Phi_0, \Phi_0 \rangle_\xi} \langle \Phi_0, \Phi_0 \rangle_\xi \\ &= B_i \cdots B_1 \cdot B_0 = \prod_{j=0}^{i} B_j.\end{aligned}$$

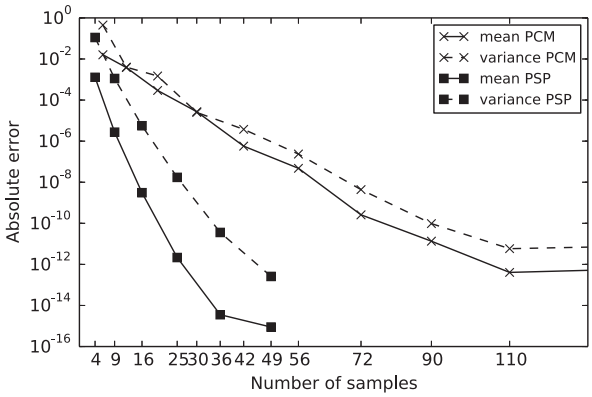

Figure 2.3: Comparing absolute error in aproximation when using point collocation nodes (PCM) and pseudo-spectral projection (PSP).

Figure 2.3 compares absolute error in approximation using pseudo-spectral projection (PSP) and our previously defined point collocation method. For the pseudo-spectral method Gauss-Legendre quadrature was used (which is optimal for moments of uniformly distributed variables). We can observe that in this example PSP is superior to PCM.

Note that the only component varying with the index i in (2.6) is the polynomial term $\mathbf{\Phi}_i$; The evaluations of u can be reused for every i, reducing the computational cost by not needing to reevaluate u for each coefficient. Note that our formerly described least squares approximation of c_i is based on a set of collocation nodes and no information on $p_{\boldsymbol{\xi}}$. The analytical Fourier coefficient formula for c_i is based on a least squares method (or Galerkin projection) involving an integral over the probability space that incorporates $p_{\boldsymbol{\xi}}$. The latter should therefore be more accurate than the former. However, when the integral is approximated by a quadrature rule, we are back to using $K+1$ collocation nodes, this time governed by the choice of quadrature rule, which depends on the probability density $p_{\boldsymbol{\xi}}$.

2.2 Summary of Papers

Three journal papers represents the main research of this thesis. The over all theme of the papers is non-intrusive polynomial chaos expansions. Paper I applies non-intrusive polynomial chaos expansion on a 1-dimensional flow problem modelling the arterial system in the human body. Paper II introduces a software toolbox that allows the user to perform non-intrusive polynomial chaos expansions. Paper III introduces a new approach in polynomial chaos expansions that aims to broaden the class of problems where polynomial chaos expansions are effective.

Paper I: A novel method for sensitivity quantification of timing and amplitude of pressure and flow waves in the arterial system

The human heart beats to keep us all alive. It is designed to pump blood periodically through the arteries and the capillaries, transporting nutrients to the body's organs. The blood propagates through the arteries as a coupled pressure and flow waves. However, on the journey the waves meet obstacles such as bifurcations, narrowing, etc. At these locations some waves are reflected and travel back towards the heart. We call these waves backward propagation waves. In a young and healthy person, when the major part of the reflected wave returns to the heart, the aortic valve is closed, causing no harm to the heart.

As we get older, the walls of the arteries often stiffens [8, 9]. As a result the wave speed of the blood flow increases, and the backward propagation wave reaches the heart earlier and with a higher amplitude. If the coupled wave reaches the heart before the valve has the chance to close, blood might enter the left ventricle and increase the load on the heart. This increased load is associated with high blood pressure [10] (hypertension), expansion of the ventricle [9, 11] (ventricular hypertrophy), and various heart diseases associated with these two conditions. Identifying the timing and the amplitude of the backward pressure wave is therefore important for understanding the mechanism of these diseases.

In this paper we present a model to study phenomena related to stiffening of the arterial walls. We construct a one-dimensional idealized mathematical model of the human systemic arterial tree. Through this model we can observe the change of amplitude and timing of the pressure wave at various key locations, and in particular how early the backward propagation reaches the heart.

One of the major challenges in such models is that the model parameters are uncertain in nature. This motivates solving the problem using uncertainty quantification. Each parameter will then have a probability distribution, and our goal is changed to finding statistical metrics instead of deterministic ones. In practice, we address the timing and amplitudes of the wave in the form of expected values, standard deviation and variance based sensitivity index. In other words, we are looking for an estimate under uncertainty, the stability of

the solution and a metric telling us which input parameter has the largest effect on the pressure timing and amplitude.

Solving the problem in practice, we use the Starfish software for the blood flow simulation [12]. To handle the uncertainty, the model is coupled with the Chaospy software [13]. Chaospy is a tailored software for performing uncertainty quantification using polynomial chaos expansions. The problem at hand is well suited for using polynomial chaos expansion for the following reasons. Even though one-dimensional distributed models are considered computationally cheap compared to other methods like the 3D fluid-structure interaction approach [14], the computational cost is high enough to be non-trivial. In addition, all metrics of interest can be expressed as simple functions of raw statistical moments of the model solver. Polynomial chaos expansions are well equipped to calculate these features.

In the paper we first outline the model and the tools for analysis before we describe four trials. In the first trial, we compare the output of the Starfish software, with a series of results from another established simulation model by Reymond et al. [15]. Key pressure and flow wave patterns from Starfish software are calculated and compared to the same patterns from the comparable model. The results of the first trial indicate that the numerical model implemented in Starfish performs on par with the model implemented by Reymond et al. which was validated in clinical trials.

In the second trial we evaluate the sensitivity of the backward pressure wave given the stiffening in different body compartments. We select 8 groups of vessels in the arterial network and give each location a random stiffness value. The distribution of this random value is selected based on data to represent the change in arterial stiffness over age. We then perform uncertainty quantification to determine which group has the most influence on the backward pressure timing and amplitude. From the resulting calculations we are able to identify the most sensitive location in the arterial network as the aortic arteries, followed by ascending aorta. The effects of the other locations on the wave were negligible.

In the third trial we assess the effects of ageing by using material parameters representative for both a young and elderly adult. With this we quantify the change in wave pattern.

In the fourth and last trial, we follow the result of the second trial and look at the sensitivity of the pressure wave in ascending aorta with respect to the stiffening of the aortic arteries. We determine which of the ten aortic arteries is the most sensitive.

Paper II: Chaospy: An Open Source Tool for Designing Methods of Uncertainty Quantification

The number of comprehensive toolboxes for performing stochastic analysis using non-intrusive polynomial chaos expansions are for most part limited [16], despite the increased popularity in the method in the recent years [17]. And considering the large number of mathematical details involved in implementing the methods, it is useful to assist users in implementing the methods with quality software. To

this end, the second paper in this thesis introduces a new software toolbox called Chaospy [13]. Chaospy aims at being a researcher's experimental foundry within the scope of uncertainty quantification using advanced Monte Carlo methods and non-intrusive polynomial chaos expansions.

Chaospy is a direct competitor to the following two software packages: the Dakota project [18] and the Opus Open Turns library [19]. The aim of the Dakota project, and to some extent the Opus Open Turns library are all encompassing software for doing uncertainty quantification using predefined state-of-the-art methods. They provide these functionalities through a black-box approach where the user's interaction with the underlying technology is limited to keep the interface simpler. In comparison Chaospy aims to assist scientists; it provides several black-box functionalities, but it is designed to be more modular. Through a compact and powerful Python interface, Chaospy allows for a very high level of customization. Through little effort it is possible to extend the software to interact with other software projects, like Pandas [20] and Scikit Learn [21] or any user define software that can be interfaced through Python.

From a technology point of view, Chaospy separates itself from Dakota and Turns by the heavy utilization of the Rosenblatt transformation [22]. The transformation allows for detailed control over the creation of arbitrary multivariate random variables, and is useful when generating random samples for Monte Carlo simulation [23], and in generalized polynomial chaos expansion [24]. In Chaospy the Rosenblatt transformation is extended to incorporate Copulas [25].

To minimize the amount of code needed to produce useful results, a collection of wrapper methods are used to automate various aspects of the underlying details, and is designed to approximate any missing features if possible. For example, if an inverse transformation is not provided by the user when constructing a random variable, the software will use a custom Newton's method for minimization to estimate those values. Likewise, any raw statistical moments are estimated using numerical integration techniques. It is therefore possible to perform a full uncertainty analysis with custom random variables using very little code. Note that Chaospy still provides a collection of 64 different distributions that all are defined without using any of these approximation methods.

Another building block in the Chaospy toolbox is the robust polynomial class, which is used in both the expansions and the model approximation in polynomial chaos expansion. The toolbox contains several methods for constructing orthogonal polynomial expansions, but allows the user to user define the polynomials. The software toolbox also provides a collection of general tools for manipulating the polynomials. These tools include basic arithmetic operators, variable substitution and creation of gradient functions to name a few. In addition, the class is constructed to be compatible with Numpy, the default toolbox for fast numerical calculation in Python [26], in terms of shaping and indexing.

Even though Chaospy is designed for Monte Carlo simulation and non-

intrusive polynomial chaos expansions, the modular nature of the software toolbox makes it usable beyond its scope. For example, when formulating the intrusive Galerkin method [27], custom code has to be written involving construction and manipulation of orthogonal polynomial expansions. With the tools Chaospy provide for arithmetic and expected value operators, the process can be greatly assisted.

One variant of polynomial chaos expansion is called pseudo-spectral projection, which is based on numerical integration. Chaospy provides a collection of tools for numerical integration. These methods include adaptive cubature [28] and optimal Gaussian quadrature [29] to mention a couple. In addition the users are free to create their own quadrature rule. Methods for linking custom methods together is provided.

Another variant of polynomial chaos expansion is the stochastic collocation method [30]. It is based on a linear regression fit between an orthogonal polynomial expansion and the model samples. Chaospy provides a few methods for this task. In addition it provides a wrapper to the Scikit-learn package that contains a large collection of state-of-the-art linear regression methods [21]. In addition to this, since the polynomial expansions can be manipulated directly, it is always possible to construct user defined regression methods as needed.

Irrespectively of the method used to construct a polynomial chaos expansion, the end result is still part of the powerful polynomial class mentioned above. Among the tools available to manipulate polynomials there exists multiple tools for performing stochastic analysis. These tools include mean, covariance, skewness, Sobol indices [31] to mention a few.

Paper III: Multivariate Polynomial Chaos Expansions with Dependent Variables

One of the major assumptions when creating a polynomial chaos expansion is that the model solution as a function of the random model parameters is smooth. A discontinuity in either the function itself or in its first derivative will result in the loss of the fast convergence property that the polynomial chaos expansions are known for. This is known as the so-called Gibbs phenomena [32].

In the third paper we introduce a new polynomial chaos expansion method that is designed to have fast convergence for models with Gibbs phenomena. This new method can be described as follows. First we search for auxiliary variables, or transformations of the uncertain model parameters, that can be used to create a re-parameterization of the model. The goal is to create a model with a solution that is smooth as a function of these auxiliary variables in the probability space. Given that these auxiliary variables are found, we can create a polynomial chaos expansion with polynomials orthogonal with respect to these variables. The resulting polynomial approximation created is then unaffected of the Gibbs phenomena present in the original parameterization.

One challenge with the new method is that the auxiliary variables created often are stochastically dependent. There are a few methods for addressing stochastically dependent variables in polynomial chaos expansions [33, 22].

However, these methods also involve re-parameterizations and will interfere with the smoothness property introduced by the proposed auxiliary variables. We therefore introduce a new type of multivariate polynomial chaos expansion where the polynomials are orthogonal on a weighted function space spanned by the stochastically dependent auxiliary variables. This results in a polynomial chaos expansion which maintains the smooth feature in the probability space created by the re-parameterization.

Creating orthogonal polynomials given stochastically dependent variables is a numerically ill-conditioned procedure [7]. To address this ill-conditioning we introduce a new methodology for constructing orthogonal polynomial expansions based upon the modified Cholesky decomposition method [34]. We illustrate how well the new construction method works by showing how much the orthogonality property holds as the polynomial order increases for a few stochastically independent random variables. We then compare the results with current best numerical method for creating orthogonal polynomial expansions: the discretized Stieltjes method [35]. The comparison shows that the two methods have comparable ability to create orthogonal polynomial expansions.

To illustrate the effectiveness of the new method, we introduce a case study involving diffusion in a multi-material/multi-domain model. We define the model with the following properties. Each layer's internal property and the boundary location between each layer are assumed to be uncertain and probability density functions are assigned to all of them. This model contains discontinuities in the first derivative as a result of the sharp change in material properties in the interface between each layer. Using classical implementations of polynomial chaos expansions, we show that the method has slow convergence properties, illustrating the Gibbs phenomena in the model.

The existence of a discontinuity in the first derivative in the multi-material/multi-domain model motivates the use of the new methodology. The paper demonstrates the steps involved in applying the new methodology to this model, including several proposals for re-parameterization and creation of auxiliary variables. The solution for each of the proposals is then compared to the classical polynomial chaos expansion implementation using both pseudo-spectral projection and point collocation. The comparison shows that the new methodology can achieve fast convergence where the classical polynomial chaos expansions can not. ¿¿¿¿¿¿¿ 6f4c6a5127aeaa533f03814ff5d13eeb5afd78d0

2.3 Future Work

The first paper explores the application of polynomial chaos expansions used on arterial flow system. The input is assumed to be stochastically independent. In practice however, it is not unreasonable to assume that the input parameters are stochstically dependent. There are two reasons for this. One, a strong predictor for the stiffness of each bloodvessel is a person's age. And two, the stiffness is usually not isolated physically at a stationary location. If one blood vessel is found to be highly stiff, it is reasonable to assume that the neighboring vessels

are stiff as well. In either case, this stiffness can be modelled by adding a positive correlation between the variables. This correlation can be introduced into the model through a Nataf [33] or Rosenblatt [22] transformation. In practice this can be approached using generalized polynomial chaos expansions [36]. It would be of interest to explore how large the positive correlation coefficients have to be for respectively a person's age and locational dependencies to have a significant effect on the model response.

The second paper introduces Chaospy, a new software toolbox for performing non-intrusive polynomial expansion. Since the topic of polynomial chaos expansions is an active reasearch field [17], there are many features that would be of interest to have included in the toolbox. Some of these features include: support for Karhunen-Loeve expansions [37] so that stochastic processes can be included, piecewise polynomial basis [38, 39], wavelet basis [40, 41] and multi-element polynomial chaos expansions [42, 43], and methods for addressing the so-called "curse of dimensionality" [44, 45, 46]. In addition, better support for adaptive methods, like the method implemented in Dakota [18] would also be of interest. Lastly, it is worth noting that the underlying code of Chaospy is written in Python programming language. To be able to address higher dimensional problems in practice, it is useful to increase the computational efficiency of the software by implementing the code using a computationally faster language like C or C++.

The last paper introduces a new method for addressing problems solved with polynomial chaos expansion with discontinuities in the probability space. Because of the discontinuities, the convergence property of the expansion is lost. We show that the new methodology can restore the convergence property on a one-dimensional diffusion problem. A natural next step would be to investigate how well the method works on two- and three dimensional cases. From there, it would be of interest to look into how well the method works on more real world practical applications. Another research direction of interest would be to look at the theoretical foundation of the new method. This paper demonstrates the method in practice, but only hints at a theoretical foundation.

Bibliography

[1] J. Idier, *Bayesian approach to inverse problems*. John Wiley and Sons, 2008.

[2] A. O'Hagan, C. E. Buck, A. Daneshkhah, J. R. Eiser, P. H. Garthwaite, D. J. Jenkinson, J. E. Oakley, and T. Rakow, *Uncertain Judgements: Eliciting Experts' Probabilities*. London ; Hoboken, NJ: Wiley, 1 edition ed., Sept. 2006.

[3] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, July 2010.

[4] G. Fishman, *Monte Carlo: concepts, algorithms, and applications*. Springer, 1996.

[5] S. Hosder, R. W. Walters, and M. Balch, "Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables," in *Proceedings of the 48th Structures, Structural Dynamics, and Materials Conference*, vol. 125, (Honolulu, HI), 2007.

[6] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2003.

[7] W. Gautschi, "Construction of Gauss-Christoffel quadrature formulas," *Mathematics of Computation*, vol. 22, p. 251, Apr. 1968.

[8] A. P. Avolio, F.-Q. Deng, W.-Q. Li, Y.-F. Luo, Z.-D. Huang, L. F. Xing, and M. F. O'rourke, "Effects of aging on arterial distensibility in populations with high and low prevalence of hypertension: comparison between urban and rural communities in China.," *Circulation*, vol. 71, no. 2, pp. 202–210, 1985.

[9] M. F. O'Rourke, "Arterial aging: pathophysiological principles," *Vascular Medicine*, vol. 12, no. 4, pp. 329–341, 2007.

[10] K. H. Pettersen, S. M. Bugenhagen, J. Nauman, D. A. Beard, and S. W. Omholt, "Arterial stiffening provides sufficient explanation for primary hypertension," *PLoS computational biology*, vol. 10, no. 5, p. e1003634, 2014.

[11] E. G. Lakatta and D. Levy, "Arterial and cardiac aging: major shareholders in cardiovascular disease enterprises part I: aging arteries: a set up for vascular disease," *Circulation*, vol. 107, no. 1, pp. 139–146, 2003.

[12] V. G. Eck, "Starfish: Stochastic Arterial Flow Simulations," Aug. 2014.

[13] J. Feinberg and H. P. Langtangen, "Chaospy Software Package for Uncertainty Quantification," 2014. *https://github.com/hplgit/chaospy*.

[14] L. Formaggia, A. M. Quarteroni, and A. Veneziani, *Cardiovascular Mathematics - Modelling and simulation of the circulatory system*. 1, Springer, 1 ed., 2009.

[15] P. Reymond, F. Merenda, F. Perren, D. Rfenacht, and N. Stergiopulos, "Validation of a one-dimensional model of the systemic arterial tree," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 297, no. 1, pp. H208–H222, 2009.

[16] R. C. Smith, *Uncertainty Quantification: Theory, Implementation, and Applications*. 1, SIAM, Dec. 2013.

[17] D. Xiu, "Fast numerical methods for stochastic computations: a review," *Communications in computational physics*, vol. 5, no. 2-4, pp. 242–272, 2009.

[18] M. S. Eldred, A. A. Giunta, B. G. van Bloemen Waanders, S. F. Wojtkiewicz, W. E. Hart, and M. P. Alleva, *DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 4.1 reference manual*. Sandia National Laboratories Albuquerque, NM, 2007.

[19] G. Andrianov, S. Burriel, S. Cambier, A. Dutfoy, I. Dutka-Malen, E. De Rocquigny, B. Sudret, P. Benjamin, R. Lebrun, and F. Mangeant, "Open TURNS, an open source initiative to Treat Uncertainties, Risks N Statistics in a structured industrial approach," in *Procedings ESREL2007 safety and reliability conference. Stavanger, Norway*, 2007.

[20] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 2012.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[22] M. Rosenblatt, "Remarks on a Multivariate Transformation," *Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 470–472, 1952.

[23] D. P. Kroese, T. Taimre, and Z. I. Botev, *Handbook of Monte Carlo Methods*, vol. 706. John Wiley & Sons, 2011.

[24] D. Xiu and G. E. Karniadakis, "Modeling uncertainty in flow simulations via generalized polynomial chaos," *Journal of Computational Physics*, vol. 187, no. 1, pp. 137–167, 2003.

[25] A. J. Lee, "Generating random binary deviates having fixed marginal distributions and specified degrees of association," *The American Statistician*, vol. 47, no. 3, pp. 209–215, 1993.

[26] S. van der Walt, S. C. Colbert, and G. Varoquaux, "The NumPy Array: A Structure for Efficient Numerical Computation," *Computing in Science & Engineering*, vol. 13, pp. 22–30, Mar. 2011.

[27] R. Ghanem and P. D. Spanos, *Stochastic finite elements: a spectral approach*. Courier Dover Publications, Aug. 2003.

[28] S. G. Johnson, "Cubature (Multi-dimensional integration)," 2013.

[29] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, vol. 23. Mathematics of Computation, 1967.

[30] S. S. Isukapalli, *Uncertainty analysis of transport-transformation models*. PhD thesis, Rutgers, The State University of New Jersey, 1999.

[31] I. M. Sobol, "On sensitivity estimation for nonlinear mathematical models," *Matematicheskoe Modelirovanie*, vol. 2, no. 1, pp. 112–118, 1990.

[32] E. Hewitt and R. E. Hewitt, "The Gibbs-Wilbraham phenomenon: an episode in Fourier analysis," *Archive for History of Exact Sciences*, vol. 21, no. 2, pp. 129–160, 1979.

[33] A. Nataf, "Dtermination des distributions de probabilites dont les marges sont donnes," *Comptes rendus de lacademie des sciences*, vol. 225, pp. 42–43, 1962.

[34] J. Gill and G. King, "What to Do When Your Hessian is Not Invertible – Alternatives to Model Respecification in Nonlinear Estimation," *Sociological Methods & Research*, vol. 33, no. 1, pp. 54–87, 2004.

[35] G. E. Forsythe, "Generation and use of orthogonal polynomials for data-fitting with a digital computer," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 2, p. 74, 1957.

[36] M. Eldred, C. Webster, and P. Constantine, "Evaluation of Non-Intrusive Approaches for Wiener-Askey Generalized Polynomial Chaos," in *NDA-4: Probabilistic Method Development*, (Schaumburg), American Institute of Aeronautics and Astronautics, Apr. 2008.

[37] M. Loeve, "Probability theory, Fourth edition," *Graduate texts in mathematics*, vol. 46, pp. 0–387, 1978.

[38] I. Babuka, R. Tempone, and G. E. Zouraris, "Galerkin finite element approximations of stochastic elliptic partial differential equations," *SIAM Journal on Numerical Analysis*, pp. 800–825, 2005.

[39] C. Schwab and R. Todor, "Sparse finite elements for stochastic elliptic problemshigher order moments," *Computing*, vol. 71, no. 1, pp. 43–63, 2003.

[40] O. P. Le Matre, H. N. Najm, R. G. Ghanem, and O. M. Knio, "Multi-resolution analysis of Wiener-type uncertainty propagation schemes," *Journal of Computational Physics*, vol. 197, no. 2, pp. 502–531, 2004.

[41] O. P. Le Matre, O. M. Knio, H. N. Najm, and R. G. Ghanem, "Uncertainty propagation using WienerHaar expansions," *Journal of Computational Physics*, vol. 197, no. 1, pp. 28–57, 2004.

[42] X. Wan and G. E. Karniadakis, "An adaptive multi-element generalized polynomial chaos method for stochastic differential equations," *Journal of Computational Physics*, vol. 209, no. 2, pp. 617–642, 2005.

[43] X. Wan and G. E. Karniadakis, "Multi-element generalized polynomial chaos for arbitrary probability measures," *SIAM Journal on Scientific Computing*, vol. 28, no. 3, pp. 901–928, 2006.

[44] P. Frauenfelder, C. Schwab, and R. Todor, "Finite elements for elliptic problems with stochastic coefficients," *Computer methods in applied mechanics and engineering*, vol. 194, no. 2, pp. 205–228, 2005.

[45] B. Ganapathysubramanian and N. Zabaras, "Sparse grid collocation schemes for stochastic natural convection problems," *Journal of Computational Physics*, vol. 225, no. 1, pp. 652–685, 2007.

[46] F. Nobile, R. Tempone, and C. G. Webster, "An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data," *SIAM Journal on Numerical Analysis*, vol. 46, no. 5, pp. 2411–2442, 2008.

I

Paper I

A novel method for sensitivity quantification of timing and amplitude of pressure and flow waves in the arterial system

A novel method for sensitivity quantification of timing and amplitude of pressure and flow waves in the arterial system

Vinzenz Eck, Jonathan Feinberg, Hans Petter Langtangen, Leif Rune Hellevik

In the field of computational hemodynamics, sensitivity quantification of pressure and flow wave dynamics has received little attention. This work presents a novel study of the sensitivity of pressure wave timing and amplitude in the arterial system with respect to arterial stiffness. Arterial pressure and flow waves were simulated with a one-dimensional distributed wave propagation model for compliant arterial networks. Sensitivity analysis of this model was based on a generalized polynomial chaos expansion combined with a stochastic collocation method. First-order statistical sensitivity indices were formulated to assess the effect of arterial stiffening on timing and amplitude of the pressure wave and backward propagating pressure wave (BPW) in the *ascending aorta*, at the maximum pressure and inflection point in the systolic phase. Only the stiffness of aortic arteries were found to significantly influence timing and amplitude of the BPW, whereas other large arteries in the systemic tree showed marginal impact. Furthermore, the *ascending aorta*, *aortic arch*, *thoracic aorta* and *infrarenal abdominal aorta* had the largest influence on amplitude, whereas only the *thoracic aorta* influenced timing. Our results showed that the non-intrusive polynomial chaos expansion is an efficient method to compute statistical sensitivity measures for wave propagation models. These sensitivities provide new knowledge in the relative importance of arterial stiffness at various locations in the arterial network. Moreover, they will significantly influence clinical data collection and effective composition of the arterial tree, for in-silico clinical studies.

sensitivity quantification, systemic arterial tree, wave propagation, arterial stiffening

3.1 Introduction

This paper presents a mathematical model for studying the phenomena of aging in the arterial system and related pathologies. The main contribution was to embed the model in a computational framework where simulation results can be judged within the context of significant statistical uncertainty and natural variation in physiological input data.

With increased age, the arterial wall stiffens and as a consequence,the pulse wave velocity rises [1, 2]. Primary hypertension [3] and left ventricular hypertrophy [2, 4] are only some of the pathologies believed to result from arterial stiffening. Pressure and flow at any location in the arterial tree are composed of two waves, one propagating from the heart to the periphery (forward) and another traveling from the periphery to the heart (backward) [5]. Arterial stiffening and rise in pulse wave velocity affect the wave propagation, particularly timing and amplitude of the backward propagating pressure wave (BPW). Knowledge of the impact on timing and amplitude plays a key role on the understanding how primary hypertension and left ventricular hypertrophy arise.

The arterial system has been investigated in many numerical simulation studies. Commonly three deterministic numerical model types have been employed: i) simplified lumped parameter models [6], ii) one-dimensional distributed models [7] and iii) computationally intensive 3D fluid-structure interaction approaches [8]. These arterial models differ in computational effort, level of detail, and necessary input/output parameters. One-dimensional distributed models are computational efficient and ideal to simulate wave propagation phenomena [9, 10, 11, 12, 13, 8, 14, 15, 16].

Some researchers have studied arterial aging with deterministic arterial models, e.g., [3, 17, 18]. The investigators [3, 17] used lumped parameter models which do not take wave propagation effects into account [6], while [18] applied a one-dimensional distributed model. Liang et al. [18] focused on ventricular-arterial coupling with simulations of the major systemic arteries. They described forward and backward waves in the *ascending aorta* and age-related changes. However, they neither assessed the changes in timing and amplitude in detail nor did they investigate the effect of increased wave reflections in the arterial tree.

To account for uncertainty in arterial models, a stochastic simulation approach is needed. Arterial models are commonly discretized with a finite difference method (FDM), finite volume method (FVM) or finite element method (FEM). In combination with stochastic methods FDM, FVM and FEM can be referred to as: stochastic finite difference (SFDM), stochastic finite volume (SFVM) and stochastic finite element method (SFEM), respectively. Some of the most common numerical methods for stochastic computing [19, 20, 21] which can be applied to FDM, FVM and FEM are: i) the perturbation method, ii) joint diagonalization method, iii) Monte Carlo method and its variants and iv) projection-based methods e.g., polynomial chaos expansion. The perturbation method is computationally undemanding, but can capture only second order moments and supports only small variations in input and output parameters (around 10%). The joint diagonalization method is commonly applied in the context of SFEM [20, 21] and has no limits on the range or type of random variables. In simple cases, the algebraic stochastic solution can be calculated. However, the computational costs are directly related to the size of the applied stochastic system. The Monte Carlo method is computationally demanding as its convergence rate is slow, although some

variants like the quasi-Monte Carlo method have an increased convergence rate. All Monte Carlo methods can handle large numbers of random input variables. The projection-based methods have a fast convergence rate, i.e., low computational costs. However, the methods are somewhat restricted in the maximum number of random input variables (around 20). The one-dimensional model in our study is discretized with FVM and combined with a projection-based method, namely a non-intrusive polynomial chaos expansion method. The limitations of the polynomial chaos expansion did not affect the calculations, as the number of stochastic input variables was manageable.

Recently, some efforts have been made to incorporate uncertainty quantification in one-dimensional models, to account for biological variation and measurement errors in arterial networks [22, 23, 24, 25, 26]. Most approaches for uncertainty quantification in one-dimensional models are based on Monte Carlo simulations [22, 23, 24] or a generalized polynomial chaos expansion method [25, 26]. The former [25] introduced the generalized polynomial chaos method to the domain of one-dimensional blood flow simulations. As a proof-of-concept by means of an analytic validation, they focused mostly on a small synthetic vascular network (with one bifurcation). The latter [26] focused on a global sensitivity analysis of a larger arterial network. They estimated the effect of uncertainty in most input parameters and linked the results to general potential physiological and pathological implications. However, biological variation of these parameters was not taken into account. An artificial variation was used instead to determine the sensitivity of pressure and flow patterns. In particular, to the best of our knowledge, sensitivity analysis of BPWs timing and amplitude, based on real physiological data has not been addressed in previous work.

This paper we present a computational framework STARFiSh (S), which combines a one-dimensional arterial wave propagation solver with a stochastic polynomial chaos method. First the deterministic one-dimensional arterial wave propagation solver was compared against the well established simulation model of Reymond et al. [16] (R), which has been validated against clinical data and reproduces physiological pressure and flow wave forms [16, 27]. Thereafter, aortic pressure simulated with parametric uncertainty, accounting for the biological variation in arterial stiffness, was compared to the model predictions simulated with (R). In addition, the effects of age-related variation in arterial stiffness on the dynamics of the BPW in the *ascending aorta* were assessed with deterministic simulations. The impact on BPW timing and amplitude was computed by a novel stochastic sensitivity analysis procedure. The new statistical analysis can identify arteries where uncertainties play a key role in the response data. Finally, the sensitivity of BPW timing and amplitude was studied with respect to stiffening in: i) groups of arteries, representing parts of the human body, and ii) individual aortic arteries.

3.2 Methods

Deterministic solver component

Governing equations.

The arterial system was regarded as one-dimensional in space, consisting of straight compliant blood vessels with a circular cross section, in accordance with the modeling framework introduced in [28]. The blood was assumed to be a Newtonian fluid with dynamic viscosity μ and density ρ. The physical behavior of pressure and flow waves in each vessel is governed by the principle of mass and momentum balance combined with a constitutive law for the elasticity of the arterial walls. Starting with the balance equations in integral form [29], the partial differential equations were derived:

$$\frac{\partial A}{\partial t} + \frac{\partial Q}{\partial x} = 0, \tag{3.1a}$$

$$\frac{\partial Q}{\partial t} + \delta \frac{\partial}{\partial x}\left(\frac{Q^2}{A}\right) + \frac{A}{\rho}\frac{\partial P}{\partial x} = -2\pi(\gamma+2)\frac{\mu}{\rho}\frac{Q}{A}, \tag{3.1b}$$

$$\text{with } \delta = \frac{(\gamma+2)}{(\gamma+1)}.$$

The primary variables, pressure $P = P(x,t)$, volumetric flow rate $Q = Q(x,t)$ and lumen area $A = A(x,t)$ are quantities averaged over cross sections of the vessel segment. Moreover, δ and γ are coefficients related to the velocity profile. The convective acceleration and viscous wall friction force (the second and fourth term in the conservation of momentum (3.1b)), both depend on the velocity profile in a local cross section. The local velocity profile $v(x,r,t)$ was approximated with a power law profile introduced by Hughes and Lubliner [30]:

$$v(x,r,t) = \bar{v}(x,t)\ \frac{\gamma+2}{\gamma}\ \left(1 - \left(\frac{r}{R}\right)^{\gamma}\right), \tag{3.2}$$

where R is the vessel radius, $\bar{v}(x,t)$ the mean (cross sectional) axial velocity and γ a coefficient for velocity profile bluntness. In this study, $\gamma = 2$ was set, resulting in a parabolic profile.

Compliant properties of the arterial wall.

Arterial blood vessels constrict and dilate due to changes in transmural pressure. The relation between cross-sectional area A and pressure P was based on the exponential pressure-diameter (P,d) relation established experimentally by Hayashi et al. [31]:

$$P(d) = P_s\ e^{\ \beta(d/d_s\ -\ 1)}, \tag{3.3}$$

where the diameter d_s and pressure P_s are values at a reference state. β is a material parameter accounting for the arterial wall stiffness. According to the

suggestions of Hayashi et al. [31], $P_s = 100$ mmHg was chosen and d_s (at P_s) calculated from radius data presented in Table 3.1.

Based on (3.3), an expression for the compliance C was derived, which is a measure for changes in cross-sectional area A due to actual pressure P:

$$C(P) = \frac{\partial A(P)}{\partial P} = \frac{2\ A_s}{\beta\ P}\ \left(1 + \frac{1}{\beta}\ \ln\left(\frac{P}{P_s}\right)\ \right), \tag{3.4}$$

where A_s is the area at reference pressure $P_s = 100$ mmHg. With (3.4), the system of governing equations (3.1) was closed. The primary unknowns were pressure $P = P(x,t)$ and volumetric flow rate $Q = Q(x,t)$.

The wave speed in the model is dependent on the actual P and A:

$$c^2 = \frac{A}{\rho\ C}. \tag{3.5}$$

An alternative expression for the wave speed as function of actual pressure was derived by including (3.3) and (3.4) in (3.5):

$$c^2 = \frac{P}{2\rho}\left(\beta + \ln\left(\frac{P}{P_s}\right)\right). \tag{3.6}$$

Numerical solution method.

The governing equations (3.1) were posed on a network of one-dimensional domains. Each domain modeled a part of the arterial tree as a straight vessel. A uniform grid was applied and the system of partial differential equations were solved with an explicit forward-backward MacCormack scheme [32], which is second order in space and time. The explicit scheme restricted the choice of both the global time step Δt and local grid spacing Δx_i in domain i according to the CFL stability condition $\Delta t \leq \Delta x_i / c_i$, where c_i is the local pulse wave velocity (3.5) in domain i. The discretization parameters Δt and Δx_i, were chosen in a way that the CFL stability condition was met for all domains i in the simulated arterial tree (55 domains) with a lower limit of five grid points per domain. Typically, the global time step was around $\Delta t = 0.346$ ms, the local Δx_i around 0.43 mm and the total number of grid points N_T around 1200. The computational time to solve the PDE-system (3.1) once for a single domain with 30 grid nodes was 0.138 ms on a standard desktop computer. The run-time of a simulation with the applied arterial network took 2 1/2 min for five heart cycles. The pressure and flow waves reached a steady state condition after the second heart beat.

Boundary conditions.

Each vessel in the arterial tree was subject to inflow and outflow conditions. The vessels were connected via bifurcations. At each bifurcation point, conservation of momentum and volumetric flow were imposed.

At the proximal boundary of the *ascending aorta* (Id 1) a volumetric flow was set as a Dirichlet boundary condition. At the distal boundaries three-element Windkessel (WK3) models were implemented:

$$\frac{dP}{dt} + \frac{P - P_v}{C_T R_c} = \frac{Q}{C_T R_c}(Z + R_c) + Z\frac{dQ}{dt}, \tag{3.7}$$

where P_v is the venous pressure, C_T the terminal compliance, Z the proximal and R_c the distal resistance of the peripheral bed. To achieve minimal reflections at high frequencies at the distal boundary, the proximal resistance (Z) was set equal to vessel impedance (Z_c) of the boundary vessel:

$$Z \;\equiv\; Z_c = \sqrt{\frac{\rho\; c(P)}{A(P)}}. \tag{3.8}$$

Values for the terminal compliance C_T and the total terminal resistance $R_T = R_c + Z$ are presented in Table 3.1. Distal boundary conditions were imposed with Riemann invariants in the same manner as proposed by [29].

Wave separation.

The BPW were extracted by means of wave separation, as suggested by Westerhof et al. [33]. This approach is based on the linearised and in-viscid form of the governing equations (3.1):

$$C\frac{\partial P}{\partial t} + \frac{\partial Q}{\partial x} = 0, \tag{3.9a}$$

$$\frac{\partial Q}{\partial t} + \frac{A}{\rho}\frac{\partial P}{\partial x} = 0. \tag{3.9b}$$

The pressure solutions $P(t)$, at an arbitrary point in the system is assumed to consist of a forward $P_f = P(x - ct)$ and backward $P_b = P(x + ct)$ traveling pressure wave. The same is applicable for the volumetric flow rate $Q(t)$:

$$P = P_f + P_b, \qquad Q = Q_f + Q_b. \tag{3.10}$$

With the characteristic vessel impedance $Z_c = P_f/Q_f = -P_b/Q_b$, a relation between volumetric flow rate and the forward and backward pressure component is stated:

$$Q = \frac{P_f}{Z_c} - \frac{P_b}{Z_c}. \tag{3.11}$$

Finally, the backward component of the pressure wave is deduced through algebraic elimination of (3.10) in (3.11):

$$P_b = \frac{P - Z_c Q}{2}. \tag{3.12}$$

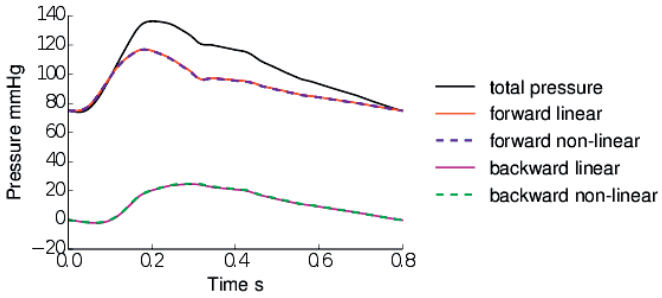

Figure 3.1: Wave splitting of the pressure wave in the *ascending aorta* with linear and non-linear methods.

The vessel impedance Z_c (3.8) applied in the linear wave splitting method, was calculated with the mean of the simulated pressure over time. Non-linear wave splitting equations can be derived, accounting for the pressure dependent characteristic impedance [34]. A comparison of pressure wave forms, calculated with linear and non-linear wave splitting, is presented in Figure 3.1. The linear wave splitting method was applied because the root mean-square error between the resulting backward waves was small (0.012).

Stochastic response modelling

The wave propagation model involves numerous input parameters that are subject to significant uncertainty. To assess the impact of such uncertainties, selected input parameters, in particular, the arterial stiffness, were described as random variables.

Our wave propagation model maps these stochastic variables to one or more response variables (e.g., BPW timing and amplitude). The goal of uncertainty quantification is to calculate the statistical properties of such response variables. Let $z = (z_1, \ldots, z_D)$ be a vector of continuous random input variables, and q a resulting scalar random response variable. The wave propagation model implies a mapping $\mathcal{M}$ between z and q: $q = \mathcal{M}(z)$. In practice, the mapping is computed by selecting a range of fixed values of the random input vector and subsequent use the deterministic wave propagation model to solve the governing equations (3.1) for pressure $P(x,t)$ and flow $Q(x,t)$. Finally, the desired response q is derived from the resulting Q and P.

Surrogate model.

The statistical properties of q can easily be computed by Monte Carlo simulation of $q = \mathcal{M}(z)$, but this approach requires a very large number of samples. More efficient methods try to replace the mapping $\mathcal{M}$ by a simple and cheap surrogate model. One such successful model is based on polynomial chaos expansions [35].

The fundamental idea is to assume that the mapping $\mathcal{M}$ is smooth, such that it can well be described by a fast-converging polynomial approximation. (Note that the assumption of smoothness relates to the mapping of z to q in probability space, not the smoothness of P and Q as functions of space and time.) The polynomial approximation takes the form:

$$q \approx \hat{q}_M = \sum_{n=0}^{N_p} c_n \Phi_n(z), \tag{3.13}$$

where M is the order of the polynomial, c_n unknown (Fourier) coefficients to be computed, Φ_n predefined orthogonal polynomials and $N_p + 1$ the number of terms related to M and the number of input variables D through the binomial coefficient:

$$N_p = \binom{M+D}{D}. \tag{3.14}$$

Fast spectral convergence of (3.13) demands the polynomials to be orthogonal with respect to the probability distribution of z. For the most common distributions, specific expressions for the polynomials can be found in the Wiener-Askey scheme [36].

The stochastic point collocation method [37] was used to calculate the Fourier coefficients c_n. The selection of collocation points in z space followed the suggestion of Hosder et al. [38] and used $K = 2\ N_p + 2$ samples from the Hammerslay sequence [39]. The coefficients c_n were then computed by a least-squares minimization of the deviation between (3.13) and simulated response at collection points. Multiple response parameters require repeated use of the least-squares procedure for each response.

Known c_n, (3.13) makes analytic expressions for lower-order moments of q, such as the expectation $\mathbb{E}[q]$, the variance $\mathrm{Var}[q]$ and standard deviation $SD = \sqrt{\mathrm{Var}[q]}$ conveniently available. These statistical quantities are of most interest in this study. Monte Carlo simulation of (3.13) can cheaply establish the full probability density of q, if desired.

The hope is that polynomial chaos expansions converge so fast that M can be kept low (say $2-4$), resulting in a modest number of evaluations (K) of the wave propagation model. This number is usually orders of magnitude smaller than what is required for a comparable accuracy of $\mathrm{Var}[q]$ by Monte Carlo simulation of the underlying flow model.

Sensitivity analysis.

To quantify the impact of each uncertain parameter z_i on the model response q, a variance-based sensitivity analysis was introduced. More precisely, a first-order sensitivity index (Si) [40] was used:

$$Si(z_d) = \frac{\mathrm{Var}\left[\mathbb{E}\left[q \mid z_d\right]\right]}{\mathrm{Var}\left[q\right]} \qquad\qquad d = 1, \ldots, D, \tag{3.15}$$

where $\mathrm{Var}\left[\mathbb{E}\left[q \mid z_d\right]\right]$ is the variance of the expected value of the model response *given* the random variable z_d. That is, this quantity measures the contribution of random parameter z_d to the global variance of the model response q. The advantage of first-order sensitivity indices is that they quantify the relative influence of the input variable z_d on a scale from 0 to 1.

Sensitivity analysis of timing and amplitude.

The sensitivity analysis was conducted for the BPW timing and amplitude in the *ascending aorta*. The occurrence time and amplitude of two distinctive points in the wave were used as model response: i) the maximum pressure and ii) the inflection point in the systolic phase of the wave (see Figure 3.9). These points were chosen as they are distinct, convenient to evaluate, and representative for the change in timing and amplitude (see also Figure 3.9). Both points were tracked in all deterministic simulations required for the uncertainty quantification procedure. Based on the BPW occurrence times and amplitudes, polynomial expansions (3.13) were calculated. Finally, first-order sensitivity indices (3.15) were derived for the timing (Si^T) and amplitude (Si^A) of both points. To the best of our knowledge, such specific sensitivity indices for BPW timing and amplitude have not been used before to quantify the impact of material parameters (arterial stiffness) in a one-dimensional simulation framework for blood flow.

Random variables.

To estimate the effect of changes in arterial wall stiffness, uncertainty was introduced in the stiffness coefficients β of the arterial wall model (3.4). In general, the variation of the stiffness coefficient (B) was defined as a product of a deterministic β value and a random variable z_d with a given probability distribution, e.g., as uniformly distributed random variable:

$$B = z_d \, \beta. \tag{3.16}$$

The random variable z_d reflects the relative variability of the deterministic variable β. This implies that the same random variable z_d can be applied in the variation equation (3.16) of a group of arteries or the hole network.

Numerical simulations

Arterial tree and initial conditions.

The simulated arterial tree consisted of the largest arteries in the systemic circuit (Figure 3.2a), which are generally said to stiffen due to aging [2]. The geometry, distensibility and terminal boundary parameters (Table 3.1) were taken from Reymond et al. [16], except for the terminal boundary parameters of the carotid and vertebral arteries, which were taken from Stergiopulos et al. [15]. The geometry and model parameters were considered as representative

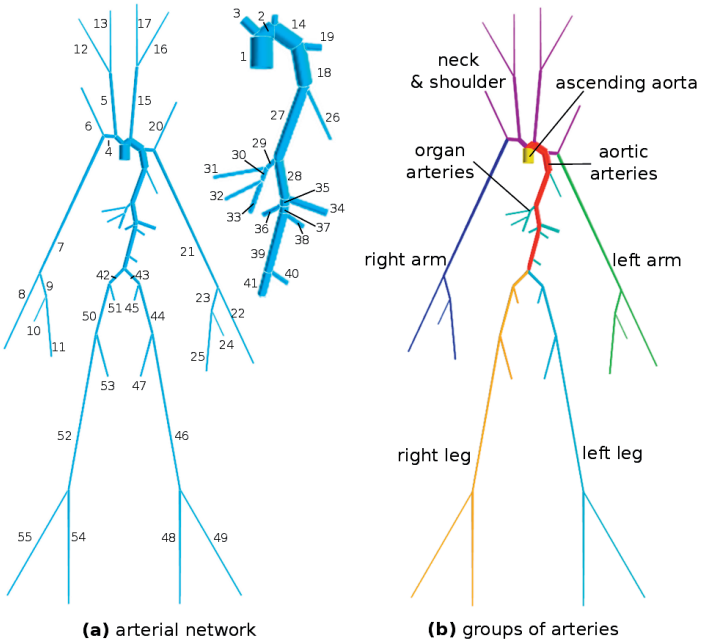

Figure 3.2: **(a)** representation of the arterial tree used in this study, **(b)** groups of arteries used in the non-deterministic simulations of Trial II.

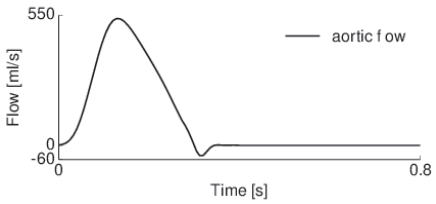

Figure 3.3: Physiological volumetric flow rate imposed as Dirichlet boundary condition at the proximal grid node of the *ascending aorta* (Id 1).

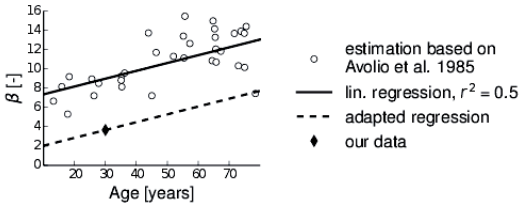

Figure 3.4: Estimated correlation of β_{30} and age from data in the *ascending aorta* published by [1].

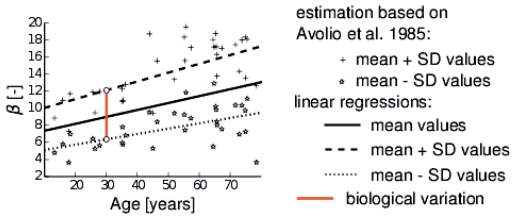

Figure 3.5: Estimated biological variation of β_{30} from data in the *ascending aorta* published by [1].

of a healthy young adult from a Western population [15, 16]. Therefore, a corresponding age of 30 was assumed and the stiffness parameters β_{30} (β at age 30) were calculated from distensibility $D_w(P_s) = C(P_s)/A_s$ (Table 3.1). The reference area A_s at pressure $P_s = 100$ mmHg was calculated from the average of proximal and distal radii presented in Table 3.1. The blood was assumed to have viscosity $\mu = 0.004$ Pa s and density $\rho = 1.050$ kg/m^3. At the *ascending aorta* (Id 1) a physiological flow rate (Figure 3.3), found in [16], with a heart rate of 0.8 s was imposed. All vessels were initialized with a pressure of 100 mmHg and a volumetric flow rate of 0 ml/s.

Variation of arterial stiffness.

The assumption was that the stiffness coefficients β_{30} of all arteries have the same relative variation (3.16). To evaluate the random variable z_d in (3.16) with respect to age-related stiffness changes, a deterministic counterpart to (3.16) was defined:

$$B = f\ \beta_{30}, \tag{3.17}$$

where f is a function describing the relative change in stiffness over age (α). The function f was estimated from data in the *ascending aorta* (mean values and standard deviations of pulse wave velocities c_{Av} with corresponding pressure P_{Av} and age α_{Av}) published by Avolio et al. [1]. First the corresponding $\beta_{Av}(c_{Av}, P_{Av}, \alpha_{Av})$ stiffness parameters of mean values in the published data were calculated based on (3.6) (see Fig 3.4). Secondly, a linear least-square regression was fitted for β_{Av} values over age and normalized with the β_{Av} at age 30, which led to:

$$f(\alpha) = (b\ \alpha + d) \tag{3.18}$$

where $b = 0.022$ and $d = 0.33$. The change in $B(\alpha)$ for a given age-range α_1 to α_2 was expressed by stochastic means with the introduction of a uniformly distributed random variable $z_d = u$ (as each age has the same probability):

$$u = \mathcal{U}\left(f(\alpha_1), f(\alpha_2)\right) \quad \Rightarrow \quad B = u\ \beta_{30}, \tag{3.19}$$

where $f(\alpha_1)$ and $f(\alpha_2)$ are the lower and upper bounds of the uniform distribution.

For the biological variation of β_{30} for a person at age 30, the stiffness parameters $\beta_{Av} \pm SD$ were additionally calculated, based on the standard deviations of the published data [1]. After fitting two regression curves for $\beta_{Av} \pm SD$, respectively, outliers were calculated for β_{Av} at age 30 (see Figure 3.5). The relative biological variation for β_{30} was assumed to be normally distributed. Thus, a normally distributed random variable was defined from the normalized β_{Av} and its outliers at age 30, and inserted into (3.16):

$$n = \mathcal{N}(\mu, \sigma) \quad \Rightarrow \quad B = n\ \beta_{30}, \tag{3.20}$$

where $\mu = 1.0$ and $\sigma = 0.31$.

Simulation trials.

For this study, four simulations trials were conducted. The geometry and boundary condition parameters were the same in all trials (Table 3.1). Only the arterial stiffness β was changed in the arterial wall models (3.3) for the different trials (overview in Table 3.2).

Trial I: Comparison of numerical models (S) and (R).

To test the predictions of our deterministic model, the simulation results were compared to those simulated with the one-dimensional code (R) of Reymond et al. [16]. The code (R) was first presented by Stergiopulos et al. [15], later enhanced and validated experimentally [16] and applied in a patient-specific investigation [27].

Our wave propagation model (S), and the code (R), have some distinct differences. First, the numerical time discretization is different: explicit (S) versus implicit (R). Second, (R) applies a visco-elastic constitutive law, while in (S) the law is non-linearly elastic. Third, the velocity profile in (R) is based on the Womersley theory, while (S) applies a power law profile. Fourth, all vessels have the same amount of grid points ($N_i = 5$) in (R), whereas (S) uses least five grid points in each vessel.

Simulations were conducted with the arterial tree geometry in Figure 3.2a and model parameters from Table 3.1, using both codes (S) and (R), to qualitatively compare the resulting pressure and flow waves by visual inspection. In addition, the discrepancy between resulting pressure waves was assessed quantitatively with a root-mean-square estimator (RMS), where P^S denotes the resulting pressure with (S) and P^R the pressure obtained with the (R):

$$\mathrm{RMS}(x,t) = \sqrt{\left(\frac{P^S - P^R}{P^R}\right)^2}. \tag{3.21}$$

This quantity defines the discrepancy between deterministic solvers (S) and (R) at one point in time and space. Finally, the mean value $\overline{\mathrm{RMS}}$ and standard

deviation SD(RMS) of all RMS in the middle of a vessel was calculated for all points in time of one heart cycle. A linear interpolation was used to match the solutions of (S) and (R) in time.

The biological variation of β_{30} in all arteries was assessed with non-deterministic simulations. The result was compared to simulations conducted with (R), using the elastic and the visco-elastic model approach for the arterial wall behavior. The same relative biological variation (3.20) with one normally distributed random variable was applied to β_{30} of all arteries. Simulations for a polynomial chaos expansion (3.13) with order $M = 3$ were conducted. For comparison, an SD interval was used, reaching from $\mathbb{E} - SD$ to $\mathbb{E} + SD$ at each point of the pressure wave.

Trial II: Sensitivity analysis with stiffening in groups of arteries.

In this trial, we estimated the sensitivity with respect to arterial stiffness in eight groups of arteries. The 55 vessels of the simulated arterial tree were divided into eight groups representing different parts of the body and topological togetherness (Figure 3.2b): *ascending aorta*, *aortic arteries*, *neck & shoulder arteries*, *organ arteries*, *left/right arm* and *left/right leg*. Even though the *ascending aorta* is one of the aortic arteries, it is treated as a separate group. This is because preliminary simulations [41] showed that changes in local compliance can have a significant influence on local pressure and flow wave forms. Eight random variables were defined, one for each group of arteries, based on (3.19), adapting changes in β_{30} on an interval corresponding to ages 19 to 75. Thus, all stiffness parameters of one group vary with the same relative variation variable z_d, i.e., all stiffness parameters of one group are always representative for the same age. Simulations were conducted for the polynomial chaos expansions, ranging in order from $M = 1$ to $M = 4$, for sensitivity indices Si^T and Si^A (3.15) both points and found that $M = 3$ gives sufficient accuracy. The evaluation of the polynomial chaos expansion (3.13) with order $M = 3$ required 330 deterministic simulations.

Trial III: Deterministic aging of the aortic arteries.

Based on the findings in Trial II, this focused on the impact of the aortic arteries. To quantify the change in BPW timing and amplitude in the *ascending aorta*, two deterministic simulations were conducted. The stiffness parameters β of the aortic arteries were set corresponding to a young adult at age 19 and an elderly adult at age 75, based on (3.17) and (3.18). The stiffness parameters for the remaining arteries were kept constant, as presented in Table 3.1. The wave patterns and some characteristic points of the waves were analyzed: the diastole (minimum pressure before the pulse), maximum pressure and inflection point in the systolic phase of the wave.

Trial IV: Sensitivity analysis with stiffening in the aortic arteries.

As in Trial III, the effect of changed arterial stiffness in the ten aortic arteries was investigated, albeit in this trial with a sensitivity analysis. Ten uniform distributed random variables (3.19) (one for each aortic artery) were defined with same age-interval, representing ages 19 to 75. The β parameters for the remaining arteries were set according to Table 3.1. For the calculation of the polynomial chaos expansion (3.13) with order $M = 3$, 572 collocation points (i.e., deterministic simulations) were used. As in Trial II, the Si^T and Si^A (3.15) at the inflection point and maximum pressure were evaluated.

3.3 Results

Trial I.

Qualitatively, the pressure and flow wave patterns of the simulations with (S) and (R) matched well (Figure 3.6), especially the rising slopes to systole (maximum) and the diastolic decay of the waves. The pressure and flow pattern of (S) were steadier whereas (R) revealed a wavy form, especially in the decay phase of the flow. In the volumetric flow wave pattern of *vertebral (Id 6)*, *subclavian B (Id 32)* and *radial (Id 22)* a second peak was observed in (S) but not in (R). Although the pressure waves of the *vertebral (Id 6)* matched well, the systolic value of volumetric flow had the highest discrepancy.

The quantified discrepancy between the simulation results in each vessel confirmed the findings (Figure 3.7 and Table 3.3). The $\overline{\text{RMS}}$ for all arteries close to the heart was under 3.0%. The $\overline{\text{RMS}}$ extended 4.3% only for the vessels in the arms and legs. The periphery vessels which terminated with a WK3 model, had d particularly higher relative discrepancy than the interior vessels. The comparison of biological variation in arterial stiffness with the different compliance models of (R) is presented in Figure 3.8, showing the pressure in the *ascending aorta*. The discrepancy between our model (S) and the visco-elastic model in (R) is qualitatively in the same order of magnitude as the elastic model in (R). The yellow area ($\mathbb{E} \pm SD$ interval) in Figure 3.8 states where the pressure will be within a standard deviation of the mean, for the defined biological variation of the stiffness parameters. Both predictions of (R), with the elastic and with the visco-elastic model, lay inside the yellow area.

Trial II.

The sensitivity indices (3.15) Si^T and Si^A are presented in Figure 3.10a for the maximum pressure and in Figure 3.10b for the inflection point. The highest sensitivity arose at both points in the *aortic arteries* group. At the maximum pressure, the sensitivity indices were $Si^T = 0.37$ and $Si^A = 0.73$ and at the inflection point $Si^T = 0.59$ and $Si^A = 0.50$. The *ascending aorta* showed a considerable influence (i.e., highest sensitivity index) only on the timing of the maximum pressure ($Si^T = 0.26$). The *neck and shoulder* group revealed

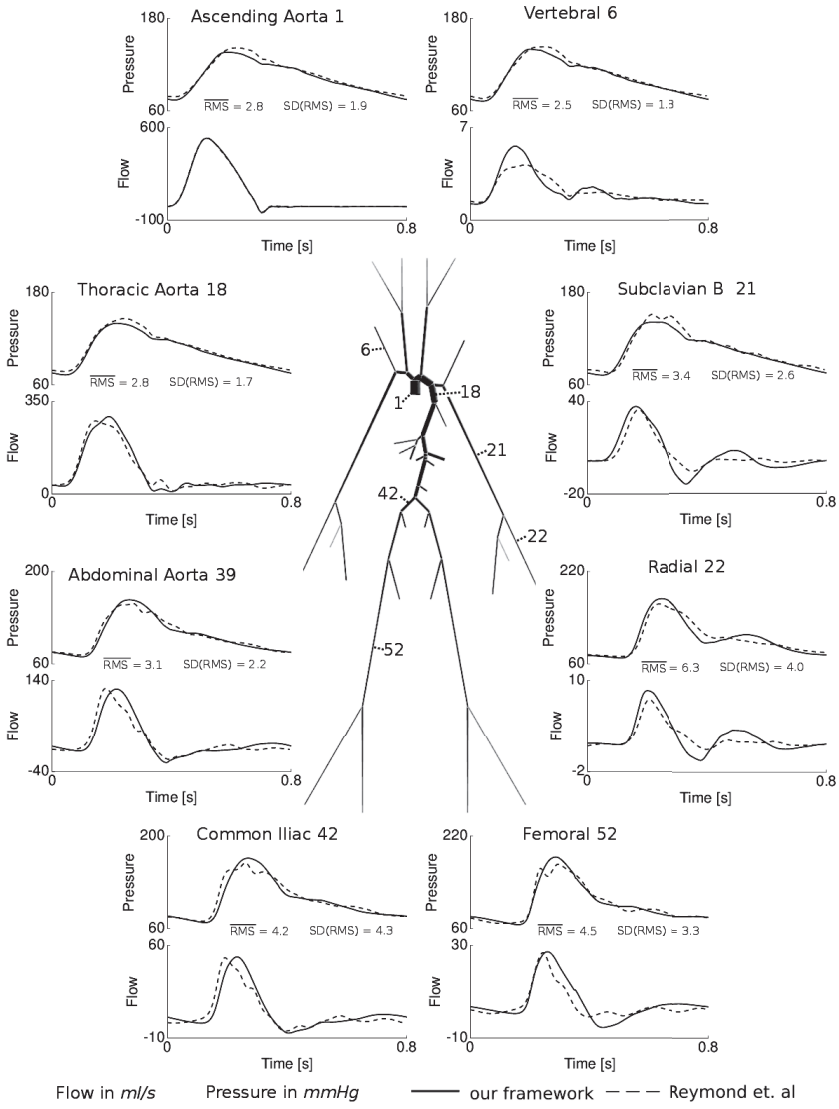

Figure 3.6: Comparison of pressure and flow waves at eight significant locations in the arterial tree at age 30 (using parameters presented in Table 3.1) simulated with our framework (continuous lines) and the code presented by Reymond et al. [16] (dashed lines).

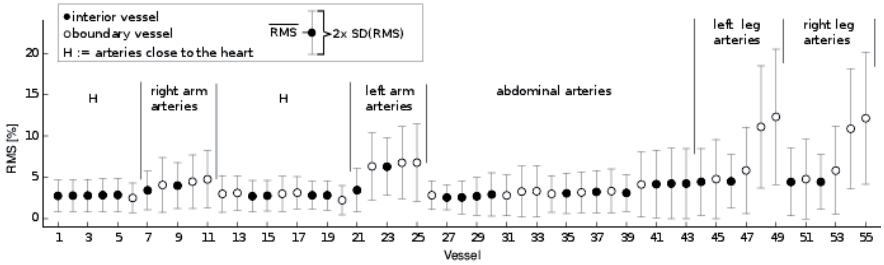

Figure 3.7: Quantitative comparison of both simulation approaches: $\overline{\text{RMS}}$ with SD(RMS) for all vessels in the simulated arterial tree. Unfilled dots represent the vessels at the boundary terminated with a WK3 model.

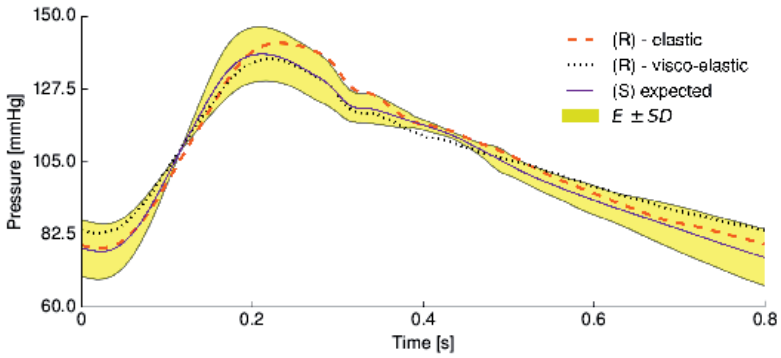

Figure 3.8: Pressure in the *ascending aorta* of non-deterministic simulations taking the biological variation of the arterial stiffness in the arterial tree into account and in comparison with deterministic simulations of the code (R), using different numerical material models for the arterial walls.

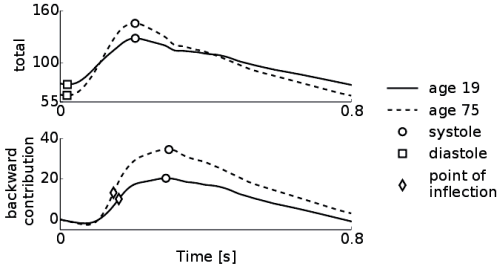

Figure 3.9: Pressure and BPW in the *ascending aorta* with stiffness parameter corresponding to age 19 (continuous) and age 75 (dashed).

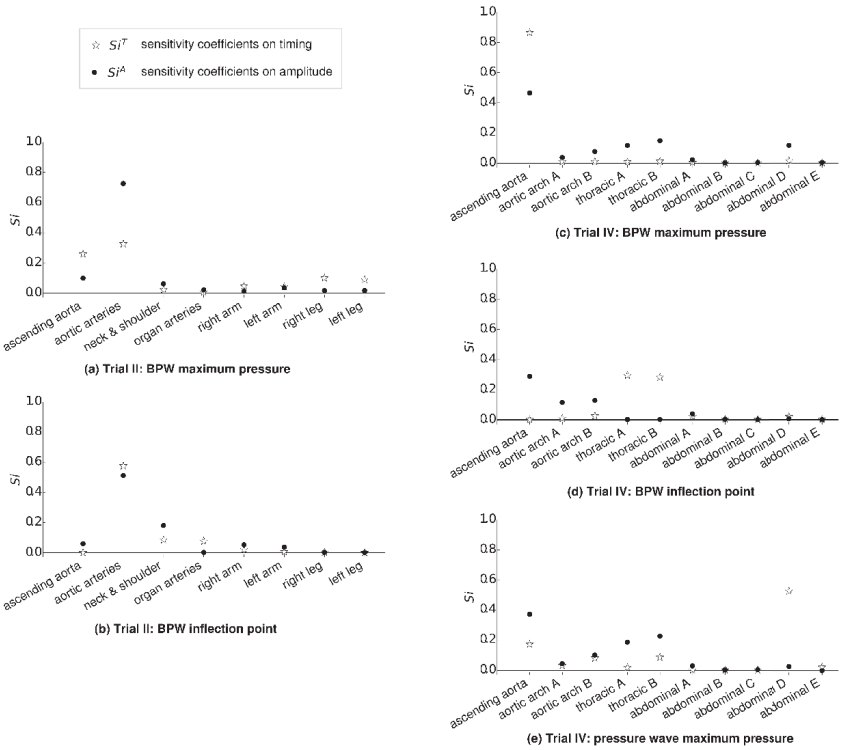

Figure 3.10: Timing (Si^T) and amplitude (Si^A) sensitivity of pressure waves in the *ascending aorta* with respect to stiffening in: groups of arteries (Trial II) **(a)** and **(b)** and aortic arteries (Trial IV) **(c)**–**(e)**.

marginal response, except for the amplitude of the inflection point. Beside for the timing of the maximum pressure ($Si^T = 0.1$), the sensitivity on the *right/left leg* was negligible. Both points showed no significant sensitivity to the *organ arteries* and the *right/left arm*.

Trial III.

The form of the pressure waves with β, corresponding to ages 19 and 75 were similar (Figure 3.9 up). The pressure wave widened with age and the rising slope steepened. The systole increased with $\Delta P = 17.19$ mmHg (Table 3.4), while the diastole decreased with $\Delta P = -12.75$ mmHg at age 75. The variation in timing was small for both the systole and diastole.

The BPW revealed a distinctly heightened amplitude at age 75 (Figure 3.9 down). The maximum pressure peak increased by $\Delta P = 10.56$ mmHg, corresponding to an increase of 66% from age 19 to 75. The occurrence time of the maximum pressure peak remained almost constant. At the inflection point, the amplitude changed insignificantly, but the occurrence time changed with $\Delta t = -0.015\ s$. This means, that the BPW arrives earlier at age 75; this corresponds to 1.9% of the heartbeat. The rising slope steepened about the same angle as the pressure wave ($\Delta\gamma = 20°$). The change in the diastole was not significant.

Trial IV.

The sensitivity indices (3.15) for timing Si^T and amplitude Si^A of the BPW maximum pressure, are presented in 3.10c. Stiffening in the *ascending aorta* induced the largest sensitivity indices of timing ($Si^T = 0.86$) and amplitude ($Si^A = 0.46$). All other vessels had no considerable sensitivity on the timing. A low but appreciable amplitude sensitivity showed stiffening in the *aortic arch B* ($S^A = 0.07$), as well as the *thoracic aorta A and B* ($Si^A = 0.12$ and $Si^A = 0.15$). Also, the *abdominal aorta D* revealed some considerable response on amplitude ($Si^A = 0.12$). The sensitivity with respect to all other abdominal vessels was negligible.

Si^T and Si^A (3.15) of BPW inflection point are shown in Figure 3.10d. The only evident sensitivity for amplitude revealed stiffening in the *ascending aorta* ($Si^A = 0.29$) and *aortic arch A and B* ($Si^A = 0.11$ and $Si^A = 0.13$). The timing sensitivity was only for *thoracic aorta A and B* larger than $Si^T = 0.03$.

Si^T and Si^A (3.15) at the maximum pressure of the pressure wave are presented in Figure 3.10e. The indices reflected the findings of the BPW maximum pressure point (Figure 3.10c), with timing sensitivity somewhat lower in the *ascending aorta* but higher in the *abdominal aorta D* ($Si^T = 0.53$).

3.4 Discussion

In this study, we have introduced a novel computational framework and demonstrated how it can be used to quantify the sensitivity of BPW timing and

amplitude with respect to arterial stiffness. With this method, the BPW timing and amplitude sensitivity in the *ascending aorta* was assessed with respect to changed arterial stiffness due to aging. The sensitivity quantification showed that the timing and amplitude had the highest sensitivity ($Si^T = 0.59$ and $Si^A = 0.73$) to arterial stiffness in the *aortic arteries* group. The aortic arteries are the largest and most compliant vessels in the arterial tree; thus, it may be expected that these vessels are most influential. Furthermore, the *ascending aorta*, *aortic arch*, *thoracic aorta* and *abdominal aorta D* were found to have the most influence on the BPW amplitude. The *thoracic aorta* was mainly influential on the timing. Deterministic simulations representing healthy adults at ages 19 and 75 showed that the BPW amplitude increased by 66% because of arterial stiffening.

The deterministic solution of our framework revealed quantitatively good agreement with results obtained with code (R) from [16] and in the non-linear high amplitude phases, the agreement of both pressure and flow patterns was especially good. The $\overline{\text{RMS}}$ of the pressure waves of (S) and (R) was less than 4.2% and the SD(RMS) less than 3% for most arteries. However, a lower diastolic pressure was observed in our simulations which might result from different material models for the arterial walls. The *tibial* arteries in the legs showed a somewhat high relative discrepancy ($\overline{\text{RMS}}$ = 10-12%), which might come from the different implementation of WK3 models. Interestingly, the results from our solver with an elastic compliance model were in better agreement with the visco-elastic model of (R), than the elastic model of (R) [16]. This finding might indicate that visco-elasticity is not strictly imperative to resemble physiological pressure and flow waves in one-dimensional simulation models.

We compared the impact of biological variation in stiffness parameters to both the elastic and visco-elastic compliance model of (R) (Figure 3.8). The results revealed that the uncertainty of arterial stiffness with respect to biological variation has at least the same impact on the prediction of one-dimensional models as the different arterial wall models. This indicates that to succeed in predictive modeling for real clinical applications uncertainty quantification is needed as much as model development.

The variation in stiffness parameters was estimated from corresponding β^A parameters, calculated with the data given by [1]. These β^A parameters were higher than the β_{30} of the considered geometry and model data (Table 3.1). This discrepancy arises due to higher pulse wave velocities in Asian populations as compared to Western populations. However, the rate of change in pulse wave velocities over age is reported to be equivalent [42].

The rate of change in stiffness over age was estimated from measurements in the *ascending aorta* (Figure 3.4) and applied to all arteries in the arterial tree. Thus, the arteries in arms and legs stiffened over age with the same rate as the aortic arteries, which is somehow not physiological, as the peripheral arteries were found not to stiffen as much as the aorta [1]. However, we believe that this simplification did not influence the sensitivity analysis, as our results revealed that the sensitivity indices were marginal with respect to stiffening in the arm

and leg arteries (Figure 3.10a and b).

The largest arteries in the arterial tree are stiffening with age causing inreased aortic blood pressure. Several investigators [4, 43, 2] claim that the stiffening actually effects the BPW timing and amplitude, which results in an increased aortic blood pressure. Our results are in accordance with this hypothesis (see Figure 3.9).

The contribution of the largest arteries in the arterial tree to the changes of BPW was assessed by dividing the arterial tree into eight groups (Figure 3.2b) and by performing an uncertainty quantification. The stiffness of the *aortic arteries* group was found to be by far the most influential with respect to both timing ($Si^T = 0.59$) and amplitude ($Si^A = 0.73$) (Figure 3.10c and d). The groups of peripheral arteries such as *left/right leg* and *left/right arm* and the *organ arteries* had marginal influence on the timing and amplitude, although the rate of change in arterial stiffness was overestimated.

As the aortic arteries seemed to be the most influential arteries, the change in pressure and BPW in the *ascending aorta* was quantified with deterministic simulations, taking only stiffening in the aortic arteries into account. The results showed remarkable changes in BPW, with an increase of 66% maximum amplitude and earlier arrival (Figure 3.9). Steepening of systolic wave flank could be found in both the pressure and its backward component. The diastole of the pressure wave decreases with age, which is in agreement with findings of others [44]. However, an alternation in wave form pattern could not be observed, e.g., a second peak after the maximum pressure [43], leading to the conclusion that different or additional mechanisms led to this phenomenon. The change in BPW in timing was best described by the inflection point, whereas the change in amplitude was best represented by the maximum pressure.

The sensitivity of the BPW to stiffening in individual aortic arteries (Figure 3.10c-e) was assessed. The amplitude was mostly affected by changes in the *ascending aorta*, *aortic arch*, *thoracic aorta* and *abdominal aorta D*. Our uncertainty quantification revealed that the *thoracic aorta* had the largest influence on the timing, evaluated at the inflection point (Figure 3.10d). Also the *abdominal aorta D* (infrarenal abdominal aorta) showed some sensitivity to the pressure wave timing. Stiffening in the *ascending aorta* had a higher influence on timing of the maximum pressure in the backward component than on the systole in total pressure (Figure 3.10e).

Banding experiments on pigs showed that increased local stiffness of the *ascending aorta* and *aortic arch* resulted in isolated systolic hypertension, i.e., increased maximum pressure [45]. BPW timing and amplitude can have a crucial impact on the heart load. If a major part of the backward component arrives earlier in the systole at the heart, the aortic valve is still open, and the BPW poses a greater load to the left ventricle. The *thoracic aorta* and *abdominal aorta D* are common locations for an aneurysm, implicating local stiffening in this region. Even after surgery to insert a graft, the vessel wall does not return to its original elastic state but remains stiff. Thus, the altered local stiffening directly influences the left ventricular pressure, as the BPW timing and amplitude is affected.

Due to its efficiency and straight-forward application, the stochastic model presented in this paper (the polynomial chaos expansion) may have great potential. However, for studies with a greater number of random variables, the method can become less efficient and one might consider instead a quasi-Monte Carlo or joint diagonalization method [20, 21].

In conclusion, we have compared our model against the well established model of Reymond et al. [16] and found that the predicted pressure and flow pattern agree well. The discrepancy of both models was in about the same magnitude as the impact of stiffness parameter with biological variation. This underlines the need for uncertainty quantification in the application of such wave propagation models for clinical applications. The sensitivity of BPW timing and amplitude with respect to arterial stiffness was successfully assessed with a novel method for sensitivity quantificaion. Of all arterial the groups *aortic arteries* had the largest influence (i.e., highest sensitivity index) on the BPW timing and amplitude in the *ascending aorta*. Local changes of arterial stiffness in the *ascending aorta*, *aortic arch*, *thoracic aorta* and *abdominal aorta D* (infrarenal), influenced the BPW amplitude, while the other aortic arteries revealed no significant impact. The BPW timing revealed high sensitivity to local changes in arterial stiffness in the *thoracic aorta*. Furthermore, the maximum pressure of the BPW increased by 66% from age 19 to 75 and the backward propagating pressure wave arrived earlier at age 75.

We thank the research group of N. Stergiopulos, Laboratory of Hemodynamics and Cardiovascular Technology (LHCT) at EPFL Lausanne, Switzerland, for the support and access to their code.

Table 3.1: Geometry, distensibility, terminal resistance and compliance of the arterial network

Vessel Name	Id	Length mm	Proximal Radius mm	Distal Radius mm	Distensibility D_w mmHg10e^{-3}	β_{30} -	Terminal Resistance R_T mmHg s m^{-1}	Terminal Compliance C_T ml mmHg10e^{-5}
Aorta Arteries								
Ascending Aorta	1	40	14.7	14.4	5.42	3.66		
Aortic Arch A	2	20	12.55	12	4.90	4.08		
Aortic Arch B	14	39	10.7	10.4	4.48	4.46		
Thoracic Aorta A	18	52	10	9.45	4.26	4.69		
Thoracic Aorta B	27	104	8.25	6.45	3.60	5.56		
Abdominal Aorta A	28	53	6.1	6.1	3.22	6.21		
Abdominal Aorta B	35	20	5.75	5.65	3.09	6.47		
Abdominal Aorta C	37	20	5.9	5.9	3.16	6.33		
Abdominal Aorta D	39	106	5.8	5.5	3.07	6.51		
Abdominal Aorta E	41	20	5.4	5.2	2.96	6.76		
Neck and Shoulder Arteries (left/right)								
Brachiocephalic	3	34	10.1	9	4.22	4.74		
Subclavian A	19/4	34	5.5/5.75	4.25/4.5	2.81/2.90	7.12/6.90		
Common Carotid	15/5	139/94	6/6.75	3/3.5	2.68/2.93	7.46/6.83		
Internal Carotid	16/12	178	2.65/2.85	2.05/2.15	1.82/1.89	10.99/10.58	104.26*	178.44*
External Carotid	17/13	41	2.35/2.5	2.15/2.25	1.77/1.83	11.30/10.93	104.26*	178.44*
Vertebral	20/6	148/149	1.85	1.4	1.46	13.7	45.08*	412.70*
Arm Arteries (left/right)								
Subclavian B, axillary, brachial	21/7	422	4.05	2.35	2.19	9.13		
Radial	22/8	235	1.75/1.85	1.4/1.55	1.43/1.49	13.99/13.42	39.7	469.76
Ulnar A	23/9	67	2.15/1.85	2.15/1.7	1.72/1.53	11.63/13.07		
Interosseous	24/10	79	0.9/1.05	0.9	1.03/1.08	19.42/18.42	633.8	29.42
Ulnar B	25/11	171	2.05/1.6	1.85/1.4	1.62/1.39	12.35/14.39	39.7	469.76
Organ Arteries (left/right)								
Intercostals	26	80	6.3	4.75	3.04	6.58	10.5	178.44
Celiac A	29	20	3.9	3.45	2.38	8.40		
Celiac B	30	20	2.6	2.45	1.90	10.53		
Hepatic	31	66	2.7	2.2	1.87	10.70	27.3	683.29
Gastric	32	71	1.6	1.5	1.42	14.08	40.7	458.48
Splenic	33	63	2.1	1.95	1.66	12.05	17.4	1069.12
Superior Mesenteric	34	59	3.95	3.55	2.41	8.30	7	2667.11
Renal	36/38	32	2.6	2.6	1.93	10.36	8.5	2195.01
Inferior Mesenteric	40	50	2.35	1.6	1.64	12.20	51.7	360.53
Leg Arteries (left/right)								
Common Iliac	43/42	59	3.95	3.5	2.39	8.37		
External Iliac	44/50	144	3.2	3.05	2.15	9.30		
Inner Iliac	45/51	50	2	2	1.65	12.12	59.7	312.55
Femoral	46/52	443	2.6	1.9	1.77	11.30		
Deepfemoral	47/53	126	2	1.85	1.61	12.42	35.9	520.00
Posterior Tibial	48/54	321	1.55	1.4	1.38	14.49	35.9	520.00
Anterior Tibial	49/55	343	1.3	1.15	1.24	16.13	42	443.71

Given arterial radii and distensibility are assumed for reference pressure $P_s = 100$ mmHg.
All values are taken from [16], except ($\star$) total terminal resistance R_T and terminal compliance C_T of carotid arteries are taken from Stergiopulos et al. [15]
β_{30}: arterial stiffness parameter calculated from given distensibility and the average of distal/proximal area

Table 3.2: Stiffness parameter in Trials I-IV.

trial	case	simulation type	random variables	distribution	β-values
I	comparison	d	–	–	β_{30}
	biological variation	s	1	$n = \mathcal{N}(1, 0.3)$	$B = n\ \beta_{30}$ (3.20)
II	groups of arteries	s	8	$u = \mathcal{U}\,(f(\alpha_1), f(\alpha_2))$ (3.19) $\alpha_1 = 19,\ \alpha_2 = 75$	$B = u_j\ \beta_{30}$ (3.19) for each group j
III	deterministic aging	d	–	–	$B = f(\alpha)\ \beta_{30}$ (3.17) for aortic arteries others: β_{30}
IV	aortic arteries	s	11	$u = \mathcal{U}\,(f(\alpha_1), f(\alpha_2))$ (3.19) $\alpha_1 = 19,\ \alpha_2 = 75$	$B = u_i\ \beta_{30}$ (3.19) for each aortic artery i others: β_{30}

d: deterministic, s: stochastic, $\mathcal{N}$: normal distribution, $\mathcal{U}$: uniform distribution
$f(\alpha)$: β-age relation function (3.18), β_{30}: values presented in Table 3.1

Table 3.3: Statistics of the $\overline{\text{RMS}}$ and SD(RMS), showing mean and extreme values of the artery groups in Figure 3.6 (Trial I)

	$\overline{\text{RMS}}$			SD(RMS)		
	mean	max	min	mean	max	min
arteries close to the heart	2.81	3.14	2.23	1.93	2.22	1.69
right arm	4.07	4.76	2.34	3.24	3.48	2.34
left arm	6.32	6.78	3.45	4.04	4.70	2.59
abdominal arteries	3.14	4.24	2.56	2.52	4.25	1.50
right leg	5.30	12.15	4.44	5.07	7.99	3.33
left leg	5.31	12.32	4.45	5.01	8.18	3.22

Table 3.4: Distinctive points of the pressure wave and BPW in the *ascending aorta* at ages 19 and 75, presented in Figure 3.9.

		systole / maximum		diastole / minimum		inflection point	
	age	amplitude	time	amplitude	time	amplitude	time
pressure wave	19	128.29	0.2066	75.22	0.0201	97.93	0.1048
	75	145.48	0.2059	62.47	0.0187	119.64	0.1289
backward	19	20.54	0.2915	-1.84	0.0672	10.19	0.1613
pressure wave	75	31.10	0.2840	-3.69	0.0780	11.81	0.1464

Bibliography

[1] A. P. Avolio, F.-Q. Deng, W.-Q. Li, Y.-F. Luo, Z.-D. Huang, L. F. Xing, and M. F. O'rourke, "Effects of aging on arterial distensibility in populations with high and low prevalence of hypertension: comparison between urban and rural communities in China.," *Circulation*, vol. 71, no. 2, pp. 202–210, 1985.

[2] M. F. O'Rourke, "Arterial aging: pathophysiological principles," *Vascular Medicine*, vol. 12, no. 4, pp. 329–341, 2007.

[3] K. H. Pettersen, S. M. Bugenhagen, J. Nauman, D. A. Beard, and S. W. Omholt, "Arterial stiffening provides sufficient explanation for primary hypertension," *PLoS computational biology*, vol. 10, no. 5, p. e1003634, 2014.

[4] E. G. Lakatta and D. Levy, "Arterial and cardiac aging: major shareholders in cardiovascular disease enterprises part I: aging arteries: a set up for vascular disease," *Circulation*, vol. 107, no. 1, pp. 139–146, 2003.

[5] P. Segers, J. Mynard, L. Taelman, S. Vermeersch, and A. Swillens, "Wave reflection: myth or reality?," *Artery Research*, vol. 6, no. 1, pp. 7–11, 2012.

[6] Y. Shi, P. Lawford, R. Hose, and others, "Review of zero-D and 1-D models of blood flow in the cardiovascular system," *Biomed. Eng. Online*, vol. 10, no. 1, p. 33, 2011.

[7] F. N. van de Vosse and N. Stergiopulos, "Pulse wave propagation in the arterial tree," *Annual Review of Fluid Mechanics*, vol. 43, pp. 467–499, 2011.

[8] L. Formaggia, A. M. Quarteroni, and A. Veneziani, *Cardiovascular Mathematics - Modelling and simulation of the circulatory system.* 1, Springer, 1 ed., 2009.

[9] Y. Huo and G. S. Kassab, "A hybrid one-dimensional/Womersley model of pulsatile blood flow in the entire coronary arterial tree," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 292, no. 6, pp. H2623–H2633, 2007.

[10] J. J. Wang and K. H. Parker, "Wave propagation in a model of the arterial circulation," *Journal of Biomechanics*, vol. 37, no. 4, pp. 457–470, 2004.

[11] D. Bessems, M. Rutten, and F. Van De Vosse, "A wave propagation model of blood flow in large vessels using an approximate velocity profile function," *Journal of Fluid Mechanics*, vol. 580, pp. 145–168, 2007.

[12] D. Bessems, *On the propagation of pressure and flow waves through the patient specific arterial system*. PhD, Technische Universiteit Eindhoven, Netherlands, 2007.

[13] S. J. Sherwin, V. Franke, J. Peir, and K. Parker, "One-dimensional modelling of a vascular network in space-time variables," *Journal of Engineering Mathematics*, vol. 47, no. 3-4, pp. 217–250, 2003.

[14] L. Formaggia, F. Nobile, A. Quarteroni, and A. Veneziani, "Multiscale modelling of the circulatory system: a preliminary analysis," *Computing and Visualization in Science*, vol. 2, no. 2-3, pp. 75–83, 1999.

[15] N. Stergiopulos, D. F. Young, and T. R. Rogge, "Computer simulation of arterial flow with applications to arterial and aortic stenoses," *Journal of Biomechanics*, vol. 25, no. 12, pp. 1477–1488, 1992.

[16] P. Reymond, F. Merenda, F. Perren, D. Rfenacht, and N. Stergiopulos, "Validation of a one-dimensional model of the systemic arterial tree," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 297, no. 1, pp. H208–H222, 2009.

[17] P. Segers, N. Stergiopulos, and N. Westerhof, "Quantification of the contribution of cardiac and arterial remodeling to hypertension," *Hypertension*, vol. 36, no. 5, pp. 760–765, 2000.

[18] F. Y. Liang, S. Takagi, R. Himeno, and H. Liu, "Biomechanical characterization of ventriculararterial coupling during aging: a multi-scale model study," *Journal of Biomechanics*, vol. 42, no. 6, pp. 692–704, 2009.

[19] D. Xiu, "Fast numerical methods for stochastic computations: a review," *Communications in computational physics*, vol. 5, no. 2-4, pp. 242–272, 2009.

[20] C. F. Li, Y. T. Feng, and D. R. J. Owen, "Explicit solution to the stochastic system of linear algebraic equations $(1 A 1+ 2 A 2++ m A m) x= b$," *Computer Methods in Applied Mechanics and Engineering*, vol. 195, no. 44, pp. 6560–6576, 2006.

[21] C. F. Li, S. Adhikari, S. Cen, Y. T. Feng, and D. R. J. Owen, "A joint diagonalisation approach for linear stochastic systems," *Computers & Structures*, vol. 88, no. 19, pp. 1137–1148, 2010.

[22] W. Huberts, C. de Jonge, W. P. M. van der Linden, M. A. Inda, K. Passera, J. H. M. Tordoir, F. N. van de Vosse, and E. M. H. Bosboom, "A sensitivity analysis of a personalized pulse wave propagation model for arteriovenous fistula surgery. Part B: Identification of possible generic model parameters," *Medical Engineering & Physics*, vol. 35, no. 6, pp. 827–837, 2013.

[23] C. A. D. Leguy, E. M. H. Bosboom, H. Gelderblom, A. P. G. Hoeks, and F. N. van de Vosse, "Estimation of distributed arterial mechanical properties using a wave propagation model in a reverse way," *Medical engineering & physics*, vol. 32, no. 9, pp. 957–967, 2010.

[24] C. A. D. Leguy, E. M. H. Bosboom, A. S. Z. Belloum, A. P. G. Hoeks, and F. N. van de Vosse, "Global sensitivity analysis of a wave propagation model for arm arteries," *Medical Engineering & Physics*, vol. 33, no. 8, pp. 1008–1016, 2011.

[25] D. Xiu and S. J. Sherwin, "Parametric uncertainty analysis of pulse wave propagation in a model of a human arterial network," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1385–1407, 2007.

[26] P. Chen, A. Quarteroni, and G. Rozza, "Simulation-based uncertainty quantification of human arterial network hemodynamics," *International Journal for Numerical Methods in Biomedical Engineering*, vol. 29, no. 6, pp. 698–721, 2013.

[27] P. Reymond, Y. Bohraus, F. Perren, F. Lazeyras, and N. Stergiopulos, "Validation of a patient-specific one-dimensional model of the systemic arterial tree," *American Journal of Physiology-Heart and Circulatory Physiology*, vol. 301, no. 3, pp. H1173–H1182, 2011.

[28] A. C. L. Barnard, W. A. Hunt, W. P. Timlake, and E. Varley, "A theory of fluid flow in compliant tubes," *Biophysical Journal*, vol. 6, no. 6, pp. 717–724, 1966.

[29] P. R. Leinan, "Biomechanical modeling of fetal veins: The umbilical vein and ductus venosus bifurcation," in *8th International Fluid Dynamics in the Oil & Gas, Metallurgical and Process Industries*, Tapir academic press, 2012.

[30] T. J. R. Hughes and J. Lubliner, "On the one-dimensional theory of blood flow in the larger vessels," *Mathematical Biosciences*, vol. 18, no. 1, pp. 161–170, 1973.

[31] K. Hayashi, H. Handa, S. Nagasawa, A. Okumura, and K. Moritake, "Stiffness and elastic behavior of human intracranial and extracranial arteries," *Journal of Biomechanics*, vol. 13, no. 2, pp. 175–184, 1980.

[32] L. R. Hellevik, J. Vierendeels, T. Kiserud, N. Stergiopulos, F. Irgens, E. Dick, K. Riemslagh, and P. Verdonck, "An assessment of ductus venosus

tapering and wave transmission from the fetal heart," *Biomechanics and Modeling in Mechanobiology*, vol. 8, no. 6, pp. 509–517, 2009.

[33] N. Westerhof, P. Sipkema, G. C. Van Den Bos, and G. Elzinga, "Forward and backward waves in the arterial system," *Cardiovascular Research*, vol. 6, no. 6, pp. 648–656, 1972.

[34] F. Pythoud, *Analysis of wave reflections in the arterial systems*. PhD thesis, Ecole Polytechnique Federale de Lausanne, 1996.

[35] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, July 2010.

[36] D. Xiu and G. E. Karniadakis, "The Wiener-Askey polynomial chaos for stochastic differential equations," *SIAM Journal on Scientific Computing*, vol. 24, no. 2, pp. 619–644, 2003.

[37] S. S. Isukapalli, *Uncertainty analysis of transport-transformation models*. PhD thesis, Rutgers, The State University of New Jersey, 1999.

[38] S. Hosder, R. W. Walters, and M. Balch, "Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables," in *Proceedings of the 48th Structures, Structural Dynamics, and Materials Conference*, vol. 125, (Honolulu, HI), 2007.

[39] J. M. Hammersley, "Monte Carlo methods for solving multivariable problems," *Annals of the New York Academy of Sciences*, vol. 86, no. 3, pp. 844–874, 1960.

[40] I. M. Sobol, "On sensitivity estimation for nonlinear mathematical models," *Matematicheskoe Modelirovanie*, vol. 2, no. 1, pp. 112–118, 1990.

[41] V. G. Eck, J. Feinberg, H. P. Langtangen, and L. R. Hellevik, "Assessment of Statistical Variability in material parameters for 1D wave propagation in arterial networks," in *Proceedings of the 3rd International Conference on Computational*
Mathematical Biomedical Engineering, 2013.

[42] A. P. Avolio, S.-G. Chen, R.-P. Wang, C.-L. Zhang, M.-F. Li, and M. F. O'Rourke, "Effects of aging on changing arterial compliance and left ventricular load in a northern Chinese urban community.," *Circulation*, vol. 68, no. 1, pp. 50–58, 1983.

[43] M. F. ORourke and W. W. Nichols, "Aortic diameter, aortic stiffness, and wave reflection increase with age and isolated systolic hypertension," *Hypertension*, vol. 45, no. 4, pp. 652–658, 2005.

[44] S. S. Franklin, W. Gustin, N. D. Wong, M. G. Larson, M. A. Weber, W. B. Kannel, and D. Levy, "Hemodynamic patterns of age-related changes in blood pressure The Framingham Heart Study," *Circulation*, vol. 96, no. 1, pp. 308–315, 1997.

[45] C. V. Ioannou, D. R. Morel, A. N. Katsamouris, S. Katranitsa, I. Startchik, A. Kalangos, N. Westerhof, and N. Stergiopulos, "Left ventricular hypertrophy induced by reduced aortic compliance," *Journal of Vascular Research*, vol. 46, no. 5, pp. 417–425, 2009.

II

Paper II

Chaospy: An Open Source Tool for Designing Methods of Uncertainty Quantification

Chaospy: An Open Source Tool for Designing Methods of Uncertainty Quantification

Jonathan Feinberg, Hans Petter Langtangen

The paper describes the philosophy, design, functionality, and usage of the Python software toolbox Chaospy for performing uncertainty quantification via polynomial chaos expansions and Monte Carlo simulation. The paper compares Chaospy to similar packages and demonstrates a stronger focus on defining reusable software building blocks that can easily be assembled to construct new, tailored algorithms for uncertainty quantification. For example, a Chaospy user can in a few lines of high-level computer code define custom distributions, polynomials, integration rules, sampling schemes, and statistical metrics for uncertainty analysis. In addition, the software introduces some novel methodological advances, like a framework for computing Rosenblatt transformations and a new approach for creating polynomial chaos expansions with dependent stochastic variables.

4.1 Introduction

We consider a computational science problem in space $\boldsymbol{x}$ and time t where the aim is to quantify the uncertainty in some response Y, computed by a forward model f, which depends on uncertain input parameters $\boldsymbol{Q}$:

$$Y = f(\boldsymbol{x}, t, \boldsymbol{Q}). \tag{4.1}$$

We treat $\boldsymbol{Q}$ as a vector of model parameters, and Y is normally computed as some grid function in space and time. The uncertainty in this problem stems from the parameters $\boldsymbol{Q}$, which are assumed to have a known joint probability density function $p_{\boldsymbol{Q}}$. The challenge is that we want to quantify the uncertainty in Y, but nothing is known about its density p_Y. The goal is then to either build the density p_Y or relevant descriptive properties of Y using the density $p_{\boldsymbol{Q}}$ and the forward model f. For all practical purposes this must be done by a numerical procedure.

In this paper, we focus on two approaches to numerically quantify uncertainty: Monte Carlo simulation and non-intrusive global polynomial chaos expansions. For a review of the former, there is a very useful book by Rubinstein, Reuven and Kroese [1], while for the latter, we refer to the excellent book by Xiu [2]. Note that other methods for performing uncertainty quantification also exist, such as perturbation methods, moment equations, and operator based

methods. These methods are all discussed in [2], but are less general and less widely applicable than the two addressed in this paper.

The number of toolboxes available to perform Monte Carlo simulation is vastly larger than the number of toolboxes for non-intrusive polynomial chaos expansion. As far as the authors know, there are only a few viable options for the latter class of methods: *The Dakota Project* (referred to as Dakota) [3], the *Opus Open Turns library* (referred to as Turns) [4], *Uncertenty Quantification Toolkit* [5], and *MIT Uncertenty Quantification Library* [6]. In this paper we will focus on the former two: Dakota and Turns. Both packages consist of libraries with extensive sets of tools, where Monte Carlo simulation and non-intrusive polynomial chaos expansion are just two tools available among several others.

It is worth noting that both Dakota and Turns can be used from two perspectives: as a user and as a developer. Both packages are open source projects with comprehensive developer manuals. As such, they both allow anyone to extend the software with any functionality one sees fit. However, these extension features are not targeting the common user and require a deeper understanding of both coding practice and the underlying design of the library. In our opinion, the threshold for a common user to extend the library is normally out of reach. Consequently, we are in this paper only considering Dakota and Turns from the point of view of the common user.

Dakota requires the forward model f to be wrapped in a stand-alone callable executable. The common approach is then to link this executable to the analysis software through a configuration file. The technical steps are somewhat cumbersome, but has their advantage in that already built and installed simulation software can be used without writing a line of code.

Alternative to this direct approach is to interact with an application programming interface (API). This approach requires the user to know how to program in the supported languages, but this also has clear benefits as an interface through a programming language allows for a deeper level of integration between the user's model and the UQ tools. Also, exposing the software's internal components through an API allows a higher detailed control over the tools and how they can be combined in statistical algorithms. This feature is attractive to scientists who would like the possibility to experiment with new or non-standard methods in ways not thought of before. This approach is used by the Turns software (using the languages Python or R) and is supported in Dakota through a library mode (using C++).

For example, consider bootstrapping [7], a popular method for measuring the stability of any parameter estimation. Neither Dakota nor Turns support bootstrapping directly. However, since Turns exposes some of the inner components to the user, a programmer can combine these to implement a custom bootstrapping technique.

This paper describes a new, third alternative open source software package called Chaospy [8]. Like Dakota and Turns, it is a toolbox for analysing uncertainty using advanced Monte Carlo simulation and non-intrusive polynomial chaos expansions. However, unlike the others, it aims to assist

scientists in constructing tailored statistical methods by combining a lot of fundamental and advanced building blocks. Chaospy builds upon the same philosophy as Turns in that it offers flexibility to the user, but takes it significantly further. In Chaospy, it is possible to gain detailed control and add user defined functionality to all of the following: random variable generation, polynomial construction, sampling schemes, numerical integration rules, response evaluation, and point collocation. The software is designed from the ground up in Python to be modular and easy to experiment with. The number of lines of code to achieve a full uncertainty analysis is amazingly low. It is also very easy to compare a range of methods in a given problem. Standard statistical methods are easily accessible through a few lines of R or Pandas [9] code, and one may think of Chaospy as a tool similar to R or Pandas, just tailored to polynomial chaos expansion and Monte Carlo simulation.

Although Chaospy is designed with a large focus on modularity, flexibility, and customization, the toolbox comes with a wide range of pre-defined statistical methods. Within the scope of Monte Carlo sampling and non-intrusive polynomial chaos expansion, Chaospy has a competitive collection of methods, comparable to both Dakota and Turns. It also offers some novel features regarding statistical methods, first and foremostly a flexible framework for defining and handling input distributions, including *dependent* stochastic variables. Detailed comparisons of features in the three packages appear throughout the paper.

The paper is structured as follows. We start in Section 4.2 with a quick demonstration of how the software can be used to perform uncertainty quantification in a simple physical problem. Section 4.3 addresses probability distributions and the theory relevant to perform Monte Carlo simulation. Section 4.4 concerns non-intrusive polynomial chaos expansions, while conclusions and topics for further work appear in Section 4.5.

4.2 A Glimpse of Chaospy in Action

To demonstrate how Chaospy is used to solve an uncertainty quantification problem, we consider a simple physical example of (scaled) exponential decay with an uncertain, piecewise constant coefficient:

$$u'(x) = -c(x)u(x), \quad u(0) = u_0,\ c(x) = \begin{cases} c_0, & x < 0.5 \\ c_1, & 0.5 \le x < 0.7 \\ c_2, & x \ge 0.7 \end{cases} \tag{4.2}$$

Such a model arises in many contexts, but we may here think of $u(x)$ as the porosity at depth x in geological layers and c_i as a (scaled) compaction constant in layer number i. For simplicity, we consider only three layers with three uncertain constants c_0, c_1, and c_2.

The model can easily be evaluated by solving the differential equation problem, here by a 2nd-order Runge-Kutta method on a mesh `x`, coded in

Python as:

```
def model(x, u0, c0, c1, c2):
    def c(x):
        if   x < 0.5:           return c0
        elif 0.5 <= x < 0.7:    return c1
        else:                   return c2

    N = len(x)
    u = np.zeros(N)

    u[0] = u0
    for n in xrange(N-1):
        dx = x[n+1] - x[n]
        K1 = -dx*u[n]*c(x[n])
        K2 = -dx*u[n] + K1/2*c(x[n]+dx/2)
        u[n+1] = u[n] + K1 + K2
    return u
```

Alternatively, the model can be implemented in some external software in another programming language. This software can either be run as a stand-alone application, where the Python function `model` runs the application and communicates with it through input and output files, or the `model` function can communicate with the external software through function calls if a Python wrapper has been made for the software (there are numerous technologies available for creating Python wrappers for C, C++, and Fortran software).

The Chaospy package may be loaded by

```
import chaospy as cp
```

Each of the uncertain parameters must be assigned a probability density, and we assume that c_0, c_1, and c_2 are stochastically independent:

```
c0 = cp.Normal(0.5, 0.15)
c1 = cp.Uniform(0.5, 2.5)
c2 = cp.Uniform(0.03, 0.07)
# Joint probability distribution
distribution = cp.J(c0, c1, c2)
```

The sample points (c_0, c_1, c_2) in probability space, where the model is to be evaluated, can be chosen in many ways. Here we specify a third-order Gaussian Quadrature scheme tailored to the joint distribution:

```
nodes, weights = cp.generate_quadrature(
    order=3, domain=distribution, rule="Gaussian")
```

The next step is to evaluate the computational model at these sample points (object `nodes`):

```
x = np.linspace(0, 1, 101)
samples = [model(x, u0, node[0], node[1], node[2])
           for node in nodes.T]
```

Now, `samples` contains a list of arrays, each array containing u values at the 101 `x` values for one combination (c_0, c_1, c_2) of the input parameters.

To create a polynomial chaos expansion, we must generate orthogonal polynomials corresponding to the joint distribution. We choose polynomials of the same order as specified in the quadrature rule, computed by the widely used three-term recurrence relation (`ttr`):

```
polynomials = cp.orth_ttr(order=3, dist=distribution)
```

To create an approximate solver (or surrogate model), we join the polynomial chaos expansion, the quadrature nodes and weights, and the model samples:

```
model_approx = cp.fit_quadrature(
               polynomials, nodes, weights, samples)
```

The `model_approx` object can now cheaply evaluate the model at a point (c_0, c_1, c_2) in probability space for all x points in the `x` array. Built-in tools can be used to derive statistical information about the model response:

```
mean = cp.E(model_approx, distribution)
deviation = cp.Std(model_approx, distribution)
```

The `mean` and `deviation` objects are arrays containing the mean value and standard deviation at each point in `x`. A graphical illustration is shown in Figure 4.1.

The accuracy of the estimation is comparable to what Dakota and Turns can provide. Figure 4.2 shows that the estimation error in the three software toolboxes are almost indistinguishable. The error is calculated as the absolute difference between the true value and the estimated value integrated over the depth x:

$$\varepsilon_{\mathbb{E}} = \int_0^1 |\mathbb{E}(u) - \mathbb{E}(u_{\text{approx}})| \, \mathrm{d}x \qquad \varepsilon_{\mathbb{V}} = \int_0^1 |\mathbb{V}(u) - \mathbb{V}(u_{\text{approx}})| \, \mathrm{d}x$$

Both the point collocation method and the pseudo-spectral projection method are included. The former is calculated using two times the random collocation nodes as the number of polynomials, and the latter using Gaussian quadrature integration with quadrature order equal to polynomial order. Note that Turns does not support pseudo-spectral projection, and is therefore only compared using point collocation.

4.3 Modelling Random Variables

Rosenblatt Transformation

Numerical methods for uncertainty quantification need to generate pseudo-random realizations

$$\{\boldsymbol{Q}_k\}_{k \in I_K} \qquad I_K = \{1, \ldots, K\},$$

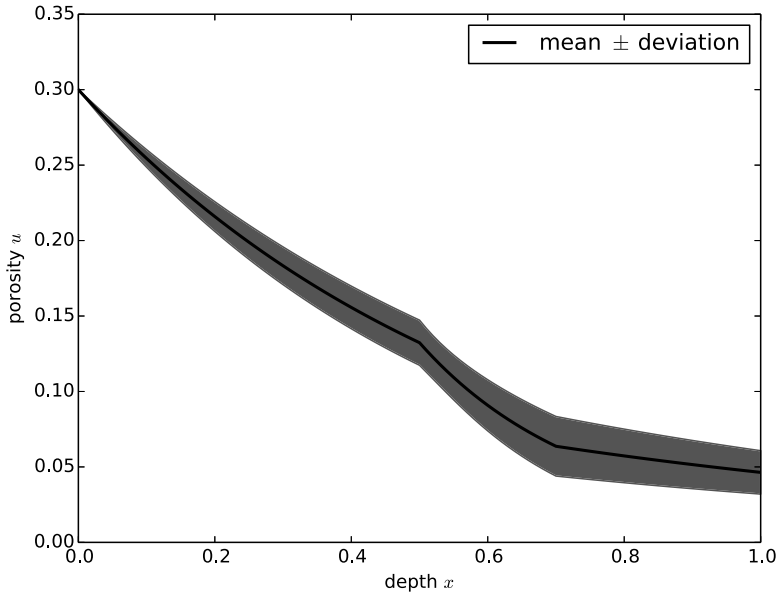

Figure 4.1: Solution of a simple stochastic differential equation with uncertain coefficients.

from the density $p_{\boldsymbol{Q}}$. Each $\boldsymbol{Q} \in \{\boldsymbol{Q}_k\}_{k \in I_K}$ is multivariate with the number of dimensions $D > 1$. Generating realizations from a given density $p_{\boldsymbol{Q}}$ is often non-trivial, at least when D is large. A very common assumption made in uncertainty quantification is that each dimension in $\boldsymbol{Q}$ consists of stochastically independent components. Stochastic independence allows for a joint sampling scheme to be reduced to a series of univariate samplings, drastically reducing the complexity of generating a sample $\boldsymbol{Q}$.

Unfortunately, the assumption of independence does not always hold in practice. We have examples from many research fields where stochastic dependence must be assumed, including modelling of climate [10], iron-ore minerals [11], finance [12], and ion channel densities in detailed neuroscience models [13]. There also exists examples where introducing dependent random variables is beneficial for the modelling process, even though the original input was stochastically independent [14]. In any cases, modelling of stochastically dependent variables are required to perform uncertainty quantification adequately. A strong feature of Chaospy is its support for stochastic dependence.

All random samples are in Chaospy generated using Rosenblatt transformations $T_{\boldsymbol{Q}}$ [15]. It allows for a random variable $\boldsymbol{U}$, generated uniformly

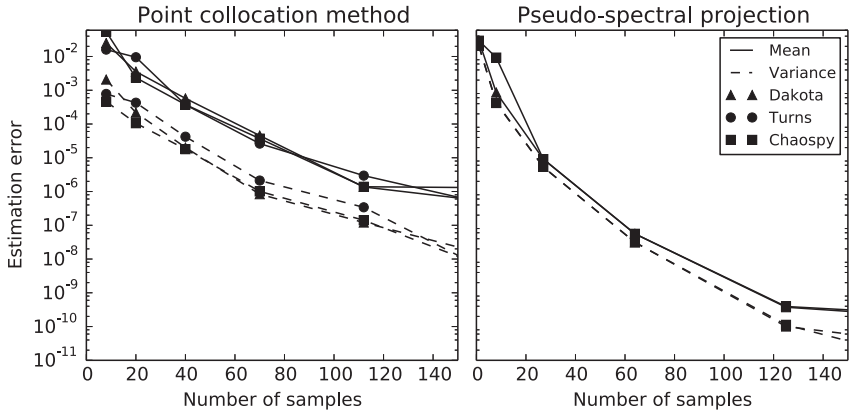

Figure 4.2: The error in estimates of the mean and variance, computed by Dakota, Turns, and Chaospy using point collocation and pseudo-spectral projection, is almost identical.

on a unit hypercube $[0,1]^D$, to be transformed into $\boldsymbol{Q} = T_{\boldsymbol{Q}}^{-1}(\boldsymbol{U})$, which behaves as if it were drawn from the density $p_{\boldsymbol{Q}}$. It is easy to generate pseudo-random samples from a uniform distribution, and the Rosenblatt transformation can then be used as a method for generating samples from arbitrary densities.

The Rosenblatt transformation can be derived as follows. Consider a probability decomposition, for example for a bivariate random variable $\boldsymbol{Q} = (Q_0, Q_1)$:

$$p_{Q_0,Q_1}(q_0, q_1) = p_{Q_0}(q_0) p_{Q_1|Q_0}(q_1 \mid q_0), \tag{4.3}$$

were p_{Q_0} is an marginal density function, and $p_{Q_1|Q_0}$ is a conditional density. For the multivariate case, the density decomposition will have the form

$$p_{\boldsymbol{Q}}(\boldsymbol{q}) = \prod_{d=0}^{D-1} p_{Q'_d}(q'_d), \tag{4.4}$$

where

$$Q'_d = Q_d \mid Q_0, \ldots, Q_{d-1} \qquad q'_d = q_d \mid q_0, \ldots, q_{d-1} \tag{4.5}$$

denotes that Q_d and q_d are dependent on all components with lower indices. A forward Rosenblatt transformation can then be defined as

$$T_{\boldsymbol{Q}}(\boldsymbol{q}) = (F_{Q'_0}(q'_0), \ldots, F_{Q'_{D-1}}(q'_{D-1})), \tag{4.6}$$

where $F_{Q'_d}$ is the cumulative distribution function:

$$F_{Q'_d}(q'_d) = \int_{-\infty}^{q_d} p_{Q'_d}(r \mid q_0, \ldots, q_{d-1}) \, \mathrm{d}r. \tag{4.7}$$

This transformation is bijective, so it is always possible to define the inverse Rosenblatt transformation $T_{\boldsymbol{Q}}^{-1}$ in a similar fashion.

Numerical Estimation of Inverse Rosenblatt Transformations

To implement the Rosenblatt transformation in practice, we need to identify the inverse transform $T_{\boldsymbol{Q}}^{-1}$. Unfortunately, $T_{\boldsymbol{Q}}$ is often non-linear without a closed-form formula, making analytical calculations of the transformation's inverse difficult. In the scenario where we do not have a symbolic representation of the inverse transformation, a numerical scheme has to be employed. To the authors' knowledge, there are no standards for defining such a numerical scheme. The following paragraphs therefore describe our proposed method for calculating the inverse transformation numerically.

The problem of calculating the inverse transformation $T_{\boldsymbol{Q}}^{-1}$ can, by decomposing the definition of the forward Rosenblatt transformation in (4.6), be reformulated as

$$F_{Q_d'}^{-1}(u \mid q_0, \ldots, q_{d-1}) = \left\{ r : F_{Q_d'}(r \mid q_0, \ldots, q_{d-1}) = u \right\} \qquad d = 0, \ldots, D-1.$$

In other words, the challenge of calculating the inverse transformation can be reformulated as a series of one dimensional root-finding problems. In Chaospy, these roots are found by employing a Newton-Raphson scheme. However, to ensure convergence, the scheme is coupled with a bisection method. The bisection method is applicable here since the problem is one-dimensional and the functions of interest are by definition monotone. When the Newton-Raphson method fails to converge at an increment, a bisection step gives the Newton-Raphson a new start location away from the previous location. This algorithm ensures fast and reliable convergence towards the root.

The Newton-Raphson-bisection hybrid method is implemented as follows. The initial values are the lower and upper bounds $[lo_0, up_0]$. If $p_{Q_d'}$ is unbound, the interval is selected such that it approximately covers the density. For example for a standard normal random variable, which is unbound, the interval $[-7.5, 7.5]$ will approximately cover the whole density with an error about 10^{-14}. The algorithm starts with a Newton-Raphson increment, using the initial value $r_0 = (up_0 - lo_0)u + lo_0$:

$$r_{k+1} = r_k - \frac{F_{Q_d'}(r_k \mid q_0, \ldots, q_{d-1}) - u}{p_{Q_d'}(r_k \mid q_0, \ldots, q_{d-1})}, \tag{4.8}$$

where the density $p_{Q_d'}$ can be approximated using finite differences. If the new value does not fall in the interval $[lo_k, up_k]$, this proposed value is rejected, and is instead replaced with a bisection increment:

$$r_{k+1} = \frac{up_k + lo_k}{2}. \tag{4.9}$$

In either case, the bounds are updated according to

$$(lo_{k+1}, up_{k+1}) = \begin{cases} (lo_k, r_{k+1}) & F_{Q'_d}(r_{k+1} \mid q_0, \ldots, q_{d-1}) > u \\ (r_{k+1}, up_k) & F_{Q'_d}(r_{k+1} \mid q_0, \ldots, q_{d-1}) < u \end{cases} \tag{4.10}$$

The algorithm repeats the steps in (4.8), (4.9) and (4.10), until the residual $|F_{Q'_d}(r_k \mid q_0, \ldots, q_{d-1}) - u|$ is sufficiently small.

The described algorithm overcomes one of the challenges of implementing Rosenblatt transformations in practice: how to calculate the inverse transformation. Another challenge is how to construct a transformation in the first place. This is the topic of the next section.

Constructing Distributions

The backbone of distributions in Chaospy is the Rosenblatt transformation $T_{\boldsymbol{Q}}$. The method, as described in the previous section, assumes that $p_{\boldsymbol{Q}}$ is known to be able to perform the transformation and its inverse. In practice, however, we first need to construct $p_{\boldsymbol{Q}}$, before the transformation can be used. This can be a challenging task, but in Chaospy a lot of effort has been put into constructing novel tools for making the process as flexible and painless as possible. In essence, users can create their own custom multivariate distributions using a new methodology as described next.

Following the definition in (4.6), each Rosenblatt transformation consists of a collection of conditional distributions. We express all conditionality through distribution parameters. For example, the location parameter of a normal distribution can be set to be uniformly distributed, say on $[-1, 1]$. The following interactive Python code defines a normal variable with a normally distributed mean:

```
>>> uniform = cp.Uniform(lo=-1, up=1)
>>> normal = cp.Normal(mu=uniform, sigma=0.1)
```

We now have two stochastic variables, `uniform` and `normal`, whose joint bivariate distribution can be constructed through the `cp.J` function:

```
>>> joint = cp.J(uniform, normal)
```

The software will, from this minimal formulation, try to sort out the dependency ordering and construct the full Rosenblatt transformation. The only requirement is that a decomposition as in (4.4) is in fact possible. The result is a fully functioning forward and inverse Rosenblatt transformation. The following code evaluates the forward transformation (the density) at $(1, 0.9)$, the inverse transformation at $(0.4, 0.6)$, and draws a random sample from the joint distribution:

```
>>> print joint.fwd([1, 0.9])
[ 1.          0.15865525]
>>> print joint.inv([0.4, 0.6])
[-0.2        -0.17466529]
>>> print joint.sample()
[-0.05992158 -0.07456064]
```

Distributions in higher dimensions are trivially obtained by including more arguments to the `cp.J` function.

As an alternative to the explicit formulation of dependency through distribution parameters, it is also possible to construct dependencies implicitly through arithmetic operators. For example, it is possible to recreate the example above using addition of stochastic variables instead of letting a distribution parameter be stochastic. More precisely, we have a uniform variable on $[-1, 1]$ and a normally distributed variable with location at $x = 0$. Adding the uniform variable to the normal variable creates a new normal variable with stochastic location:

```
>>> uniform = Uniform(lo=-1, up=1)
>>> normal0 = Normal(mu=0, scale=0.1)
>>> normal = normal0 + uniform
>>> joint = J(uniform, normal)
```

As before, the software automatically sorts the dependency ordering from the context. Here, since the uniform variable is present as first argument, the software recognises the second argument as a normal distribution, conditioned on the uniform distribution, and not the other way around.

Another favorable feature in Chaospy is that multiple transformations can be stacked on top of each other. For example, consider the example of a multivariate log-normal random variable $\boldsymbol{Q}$ with three dependent components. (Let us ignore for a moment the fact that Chaospy already offers such a distribution.) Trying to decompose this distribution is a very cumbersome task if performed manually. However, this process can be drastically simplified through variable transformations, for which Chaospy has strong support. A log-normal distribution, for example, can be expressed as

$$\boldsymbol{Q} = e^{\boldsymbol{Z}L+\boldsymbol{b}},$$

where $\boldsymbol{Z}$ are standard normal variables, and L and $\boldsymbol{b}$ are predefined matrix and vector, respectively. To implement this particular transformation, we only have to write

```
>>> Z = cp.J(cp.Normal(0,1), cp.Normal(0,1), cp.Normal(0,1))
>>> Q = e**(Z*L + b)
```

The resulting distribution is fully functional multivariate log-normal, assuming L and $\boldsymbol{b}$ are properly defined.

One obvious prerequisite for using univariate distributions to create conditionals and multivariate distributions, is the availability of univariate

distributions. Since the univariate distribution is the fundamental building block, Chaospy offers a large collection of 64 univariate distributions. They are all listed in Table 4.1. The list also shows that Dakota's support is limited to 11 distributions, and Turns has a collection of 26 distributions.

Distribution	D	T	C	**Distribution**	D	T	C
Alpha	n	n	y	Anglit	n	n	y
Arcsinus	n	n	y	Beta	y	y	y
Brandford	n	n	y	Burr	n	y	y
Cauchy	n	n	y	Chi	n	y	y
Chi-Square	n	y	y	Double Gamma	n	n	y
Double Weibull	n	n	y	Epanechnikov	n	y	y
Erlang	n	n	y	Exponential	y	y	y
Exponential Power	n	n	y	Exponential Weibull	n	n	y
Birnbaum-Sanders	n	n	y	Fisher-Snedecor	n	y	y
Fisk/Log-Logistic	n	n	y	Folded Cauchy	n	n	y
Folded Normal	n	n	y	Frechet	y	n	y
Gamma	y	y	y	Gen. Exponential	n	n	y
Gen. Extreme Value	n	n	y	Gen. Gamma	n	n	y
Gen. Half-Logistic	n	n	y	Gilbrat	n	n	y
Truncated Gumbel	n	n	y	Gumbel	y	y	y
Hypergeometric Secant	n	n	y	Inverse-Normal	n	y	n
Kumaraswamy	n	n	y	Laplace	n	y	y
Levy	n	n	y	Log-Gamma	n	n	y
Log-Laplace	n	n	y	Log-Normal	y	y	y
Log-Uniform	y	y	y	Logistic	n	y	y
Lomax	n	n	y	Maxwell	n	n	y
Mielke's Beta-Kappa	n	n	y	Nakagami	n	n	y
Non-Central Chi-Squared	n	y	y	Non-Central Student-T	n	y	y
Non-central F	n	n	y	Normal	y	y	y
Pareto (First kind)	n	n	y	Power Log-Normal	n	n	y
Power Normal	n	n	y	Raised Cosine	n	n	y
Rayleigh	n	y	y	Reciprocal	n	n	y
Rice	n	y	n	Right-skewed Gumbel	n	n	y
Student-T	n	y	y	Trapezoidal	n	y	n
Triangle	y	y	y	Truncated Exponential	n	n	y
Truncated Normal	n	y	y	Tukey-Lamdba	n	n	y
Uniform	y	y	y	Wald	n	n	y
Weibull	y	y	y	Wigner	n	n	y
Wrapped Cauchy	n	n	y	Zipf-Mandelbrot	n	y	n

Table 4.1: List of supported continuous distributions in software. The titles 'D', 'T' and 'C' represents Dakota, Turns and Chaospy respectively. The elements 'y' and 'n' represent the answers 'yes' and 'no' indicating if the distribution is supported or not.

The Chaospy software supports in addition custom distributions through the function `cp.constructor`. To illustrate its use, consider the simple example of a uniform random variable on the interval $[lo, up]$. The minimal input to create such a distribution is

```
>>> Uniform = cp.constructor(
...     cdf=lambda self,x,lo,up: (x-lo)/(up-lo),
...     bnd=lambda self,x,lo,up: (lo,up) )
>>> uniform = Uniform(lo=-1, up=1)
```

Here, the two provided arguments are a cumulative distribution function (`cdf`), and a boundary interval function (`bnd`), respectively. The `cp.constructor` function also takes several optional arguments to provide extra functionality. For example, the inverse of the cumulative distribution function – the point percentile function – can be provided through the `ppf` keyword. If this function is not provided, the software will automatically approximate it using the method described in Section 4.3.

Copulas

Dakota and Turns do not support the Rosenblatt transformation applied to multivariate distributions with dependencies. Instead, the two packages model dependencies using copulas [16]. A copula consists of stochastically independent multivariate distributions made dependent using a parameterized function g. Since the Rosenblatt transformation is general purpose, it is possible to construct any copula directly. However, this can quickly become a very cumbersome task since each copula must be decomposed individually for each combination of independent distributions and parameterization of g. To simplify the user's efforts, Chaospy has dedicated constructors that can reformulate a copula coupling into a Rosenblatt transformation. This is done following the work of Lee [17] and approximated using finite differences. The implementation is based of the software toolbox RoseDist [18]. In practice, this approach allow copulas to be defined in a Rosenblatt transformation setting. For example, to construct a bivariate normal distribution with a Clayton copula in Chaospy, we do the following:

```
>>> joint = cp.J(cp.Normal(0,1), cp.Normal(0,1))
>>> clayton = cp.Clayton(joint, theta=2)
```

A list of supported copulas are listed in Table 4.2. It shows that Turns supports 7 methods, Chaospy 6, while Dakota offers 1 method.

Variance Reduction Techniques

As noted in the beginning of Section 4.3, by generating samples $\{\boldsymbol{Q}_k\}_{k\in I_K}$ and evaluating the response function f, it is possible to draw inference upon Y without knowledge about p_Y, through Monte Carlo simulation. Unfortunately,

Supported Copulas	Dakota	Turns	Chaospy
Ali-Mikhail-Haq	no	yes	yes
Clayton	no	yes	yes
Farlie-Gumbel-Morgenstein	no	yes	no
Frank	no	yes	yes
Gumbel	no	yes	yes
Joe	no	no	yes
Minimum	no	yes	no
Normal/Nataf	yes	yes	yes

Table 4.2: The list of supported copulas in the various software packages.

the number of samples K to achieve reasonable accuracy can often be very high, so if f is assumed to be computationally expensive, the number of samples needed frequently make Monte Carlo simulation infeasible for practical applications. As a way to mitigate this problem, it is possible to modify $\{\boldsymbol{Q}_k\}_{k\in I_K}$ from traditional pseudo-random samples, so that the accuracy increases. Schemes that select non-traditional samples for $\{\boldsymbol{Q}_k\}_{k\in I_K}$ to increase accuracy are known as *variance reduction techniques*. A list of such techniques are presented in Table 4.3, and it shows that Dakota, Turns and Chaospy support 4, 7, and 7 variance reduction techniques, respectively.

One of the more popular variance reduction technique is the *quasi-Monte Carlo scheme* [1]. The method consists of selecting the samples $\{\boldsymbol{Q}_k\}_{k\in I_K}$ to be a low-discrepancy sequence instead of pseudo-random samples. The idea is that samples placed with a given distance from each other increase the coverage over the sample space, requiring fewer samples to reach a given accuracy. For example, if standard Monte Carlo requires 10^6 samples for a given accuracy, quasi-Monte Carlo can often get away with only 10^3. Note that this would break some of the statistical properties of the samples [19].

Most of the theory on quasi-Monte Carlo methods focuses on generating samples on the unit hypercube $[0,1]^N$. The option to generate samples directly on to other distributions exists, but is often very limited. To the authors' knowledge, the only viable method for including most quasi-Monte Carlo methods into the vast majority of non-standard probability distributions, is through the Rosenblatt transformation. Since Chaospy is built around the Rosenblatt transformation, it has the novel feature of supporting quasi-Monte Carlo methods for all probability distributions. Turns and Dakota only support Rosenblatt transformations for independent variables and the Normal copula.

Sometimes the quasi-Monte Carlo method is infeasible because the forward model is too computationally costly. The next section describes polynomial chaos expansions, which often require far fewer samples than the quasi-Monte Carlo method for the same amount of accuracy.

Quasi-Monte Carlo Scheme	Dakota	Turns	Chaospy
Faure sequence [20]	no	yes	no
Halton sequence [21]	yes	yes	yes
Hammersley sequence [22]	yes	yes	yes
Haselgrove sequence [23]	no	yes	no
Korobov latice [24]	no	no	yes
Niederreiter sequence [25]	no	yes	no
Sobol sequence [26]	no	yes	yes
Other Methods	Dakota	Turns	Chaospy
Antithetic variables [1]	no	no	yes
Importance sampling [1]	yes	yes	yes
Latin Hypercube sampling [27]	yes	limited	yes

Table 4.3: The different sampling schemes available.

4.4 Polynomial Chaos Expansions

Polynomial chaos expansions represent a collection of methods that can be considered a subset of polynomial approximation methods, but particularly designed for uncertainty quantification. A general polynomial approximation can be defined as

$$\hat{f}(\boldsymbol{x}, t, \boldsymbol{Q}) = \sum_{n \in I_N} c_n(\boldsymbol{x}, t)\boldsymbol{\Phi}_n(\boldsymbol{Q}) \qquad I_N = \{0, \dots, N\}, \tag{4.11}$$

where $\{c_n\}_{n \in I_N}$ are coefficients (often known as Fourier coefficients) and $\{\boldsymbol{\Phi}_n\}_{n \in I_N}$ are polynomials. If $\hat{f}$ is a good approximation of f, it is possible to either infer statistical properties of $\hat{f}$ analytically or through cheap numerical computations where $\hat{f}$ is used as a surrogate for f.

A polynomial chaos expansion is defined as a polynomial approximation, as in (4.11), where the polynomials $\{\boldsymbol{\Phi}_n\}_{n \in I_N}$ are orthogonal on a custom weighted function space L_Q:

$$\langle \boldsymbol{\Phi}_n, \boldsymbol{\Phi}_m \rangle = \mathbb{E}[\boldsymbol{\Phi}_n(\boldsymbol{Q})\boldsymbol{\Phi}_m(\boldsymbol{Q})] = \int \cdots \int \boldsymbol{\Phi}_n(\boldsymbol{q})\boldsymbol{\Phi}_m(\boldsymbol{q}) p_{\boldsymbol{Q}}(\boldsymbol{q}) \, \mathrm{d}\boldsymbol{q} = 0 \quad n \neq m. \tag{4.12}$$

As a side note, it is worth noting that in parallel with polynomial chaos expansions, there also exists an alternative collocation method based on multivariate Lagrange polynomials [28]. This method is supported by Dakota and Chaospy, but not Turns.

To generate a polynomial chaos expansion, we must first calculate the polynomials $\{\boldsymbol{\Phi}_n\}_{n \in I_N}$ such that the orthogonality property in (4.12) is satisfied. This will be the topic of Section 4.4 In Section 4.4 we show how to estimate the coefficients $\{c_n\}_{n \in I_N}$. Last, in Section 4.4, tools used to quantify uncertainty in polynomial chaos expansions will be discussed.

Orthogonal Polynomials Construction

From (4.12) it follows that the orthogonality property is not in general transferable between distributions, since a new set of polynomials has to be constructed for each $p_{\boldsymbol{Q}}$. The easiest approach to construct orthogonal polynomials is to identify the probability density $p_{\boldsymbol{Q}}$ in the so-called Askey-Wilson scheme [29]. The polynomials can then be picked from a list, or be built from analytical components. The continuous distributions supported in the scheme include the standard normal, gamma, beta, and uniform distributions respectively through the Hermite, Laguerre, Jacobi, and Legendre polynomial expansion. All the three mentioned software toolboxes support these expansions. RIP: Note about Stieltjes-Wigert/Lognormal variables

Moving beyond the standard collection of the Askey-Wilson scheme, it is possible to create custom orthogonal polynomials, both analytically and numerically. Unfortunately, most methods involving finite precision arithmetics are ill-posed, making a numerical approach quite a challenge [30]. This section explores the various approaches for constructing polynomial expansions. A full list of methods is found in Table 4.4. It shows that Dakota, Turns and Chaospy support 4, 3 and 5 orthogonalisation methods, respectively.

Orthogonalization Method	Dakota	Turns	Chaospy
Askey-Wilson Scheme [29]	yes	yes	yes
Bertran recursion [31]	no	no	yes
Cholesky Decomposition [14]	no	no	yes
Discretized Stieltjes [32]	yes	no	yes
Modified Chebyshev [32]	yes	yes	no
Modified Gram-Schmidt [32]	yes	yes	yes

Table 4.4: Methods for generating expansions of orthogonal polynomials.

Looking beyond an analytical approach, the most popular method for constructing orthogonal polynomials is the discretized Stieltjes procedure [33]. As far as the authors know, it is the only truly numerically stable method for orthogonal polynomial construction. It is based upon one-dimensional recursion coefficients that are estimated using numerical integration. Unfortunately, the method is only applicable in the multivariate case if the components of $p_{\boldsymbol{Q}}$ are stochastically independent.

Generalized Polynomial Chaos Expansions One approach to model densities with stochastically dependent components numerically, is to reformulate the uncertainty problem as a set of independent components through generalised polynomial chaos expansion [34]. As described in detail in Section 4.3, a Rosenblatt transformation allows for the mapping between any domain and the unit hypercube $[0,1]^D$. With a double transformation we can

reformulate the response function f as

$$f(\boldsymbol{x}, t, \boldsymbol{Q}) = f(\boldsymbol{x}, t, T_{\boldsymbol{Q}}^{-1}(T_{\boldsymbol{R}}(\boldsymbol{R}))) \approx \hat{f}(\boldsymbol{x}, t, \boldsymbol{R}) = \sum_{n \in I_N} c_n(\boldsymbol{x}, t) \boldsymbol{\Phi}_n(\boldsymbol{R}),$$

where $\boldsymbol{R}$ is any random variable drawn from $p_{\boldsymbol{R}}$, which for simplicity is chosen to consists of independent components. Also, $\{\boldsymbol{\Phi}_n\}_{n \in I_N}$ is constructed to be orthogonal with respect to $L_{\boldsymbol{R}}$, not $L_{\boldsymbol{Q}}$. In any case, $\boldsymbol{R}$ is either selected from the Askey-Wilson scheme, or calculated using the discretized Stieltjes procedure. We remark that the accuracy of the approximation deteriorate if the transformation composition $T_{\boldsymbol{Q}}^{-1} \circ T_{\boldsymbol{R}}$ is not smooth [34]. Dakota, Turns, and Chaospy all support generalized polynomial chaos expansions for independent stochastic variables and the Normal/Nataf copula listed in Table 4.2. Since Chaospy has the Rosenblatt transformation underlying the computational framework, generalized polynomial chaos expansions are in fact available for all densities.

The Direct Multivariate Approach Given that both the density $p_{\boldsymbol{Q}}$ has stochastically dependent components, and the transformation composition $T_{\boldsymbol{Q}}^{-1} \circ T_{\boldsymbol{R}}$ is not smooth, it is still possible to generate orthogonal polynomials numerically. As noted above, most methods are numerically unstable, and the accuracy in the orthogonality can deteriorate with polynomial order, but the methods can still be useful [14]. In Table 4.4, only Chaospy's implementation of Bertran's recursion method [31], Cholesky decomposition [35] and modified Gram-Schmidt orthogonalization [32] support construction of orthogonal polynomials for multivariate dependent densities directly.

Custom Polynomial Expansions In the most extreme cases, an automated numerical method is insufficient. Instead, a polynomial expansion has to be constructed manually. User-defined expansions can be created conveniently, as demonstrated in the next example involving a second-order Hermite polynomial expansion, orthogonal with respect to the normal density [29]:

$$\{\boldsymbol{\Phi}_n\}_{n \in I_6} = \left\{1, Q_0, Q_1, Q_0^2 - 1, Q_0 Q_1, Q_1^2 - 1\right\}$$

The relevant Chaospy code for creating this polynomial expansion looks like

```
>>> q0, q1 = cp.variable(2)
>>> phi = cp.Poly([1, q0, q1, q0**2-1, q0*q1, q1**2-1])
>>> print phi
[1, q0, q1, q0^2-1, q0q1, -1+q1^2]
```

Chaospy contains a collection of tools to manipulate and create polynomials, see Table 4.5.

One thing worth noting is that polynomial chaos expansions suffers from the curse of dimensionality: The number of terms grows exponentially with the number of dimensions [36]. As a result, Chaospy does not support neither high

Function	Description
`all`	Test all coefficients for non-zero
`any`	Test any coefficients for non-zero
`around`	Round to a given decimal
`asfloat`	Set coefficients type as float
`asint`	Set coefficient type as int
`basis`	Create monomial basis
`cumprod`	Cumulative product
`cumsum`	Cumulative sum
`cutoff`	Truncate polynomial order
`decompose`	Convert from series to sequence
`diag`	Construct or extract diagonal
`differential`	Differential operator
`dot`	Dot-product
`flatten`	Flatten an array
`gradient`	Gradient (or Jacobian) operator
`hessian`	Hessian operator
`inner`	Inner product
`mean`	Average
`order`	Extract polynomial order
`outer`	Outer product
`prod`	Product
`repeat`	Repeat polynomials
`reshape`	Reshape axes
`roll`	Roll polynomials
`rollaxis`	Roll axis
`rolldim`	Roll the dimension
`std`	Empirical standard deviation
`substitute`	Variable substitution
`sum`	Sum along an axis
`swapaxes`	Interchange two axes
`swapdim`	Swap the dimensions
`trace`	Sum along the diagonal
`transpose`	Transpose the coefficients
`tril`	Extract lower triangle of coefficients
`tricu`	Extract cross-diagonal upper triangle
`var`	Empirical variance
`variable`	Simple polynomial constructor

Table 4.5: List of tools for creating and manipulating polynomials.

dimensional nor infinite dimensional problems (random fields). One approach to address such problems with polynomial chaos expansion is to first reduce the number of dimension through techniques like Karhunen-Loeve expansions [37].

If software implementations of such methods can be provided, the user can easily extend Chaospy to high and infinite dimensional problems.

Chaospy includes operators such as the expectation operator $\mathbb{E}$. This is a helpful tool to ensure that the constructed polynomials are orthogonal, as defined in (4.12). To verify that two elements in `phi` are indeed orthogonal under the standard bivariate normal distribution, one writes

```
>>> dist = cp.J(cp.Normal(0,1), cp.Normal(0,1))
>>> print cp.E(phi[3]*phi[5], dist)
0.0
```

More details of operators used to perform uncertainty analysis are given in Section 4.4.

Calculating Coefficients

There are several methodologies for estimating the coefficients $\{c_n\}_{n \in I_N}$, typically categorized either as non-intrusive or intrusive, where non-intrusive means that the computational procedures only requires evaluation of f (i.e., software for f can be reused as a black box). Intrusive methods need to incorporate information about the underlying forward model in the computation of the coefficients. In case of forward models based on differential equations, one performs a Galerkin formulation for the coefficients in probability space, leading effectively to a D-dimensional differential equation problem in this space [38]. Back et al. [39] demonstrated that the computational cost of such an intrusive Galerkin method in some cases was higher than some non-intrusive methods. None of the three toolboxes discussed in this paper have support for intrusive methods.

Within the realm of non-intrusive methods, there are in principle two viable methodologies available: pseudo-spectral projection [40] and the point collocation method [41]. The former applies a numerical integration scheme to estimate Fourier coefficients, while the latter solves a linear system arising from a statistical regression formulation. Dakota and Chaospy support both methodologies, while Turns only supports point collocation. We shall now discuss the practical, generic implementation of these two methods in Chaospy.

Integration Methods

The pseudo-spectral projection method is based on a standard least squares minimization in the weighted function space $L_{\boldsymbol{Q}}$. Since the polynomials are orthogonal in this space, the associated linear system is diagonal, which allows a closed-form expression for the Fourier coefficients. The expression involves high-dimensional integrals in $L_{\boldsymbol{Q}}$. Numerical integration is then required,

$$c_n = \frac{\mathbb{E}[Y\boldsymbol{\Phi}_n]}{\mathbb{E}[\boldsymbol{\Phi}_n^2]} = \frac{1}{\mathbb{E}[\boldsymbol{\Phi}_n^2]} \int \cdots \int p_{\boldsymbol{Q}}(\boldsymbol{q}) f(\boldsymbol{x}, t, \boldsymbol{q}) \boldsymbol{\Phi}_n(\boldsymbol{q}) \, \mathrm{d}\boldsymbol{q} \tag{4.13}$$

$$\approx \frac{1}{\mathbb{E}[\boldsymbol{\Phi}_n^2]} \sum_{k \in I_K} w_k p_{\boldsymbol{Q}}(\boldsymbol{q}_k) f(\boldsymbol{x}, t, \boldsymbol{q}_k) \boldsymbol{\Phi}_n(\boldsymbol{q}_k) \qquad I_K = \{0, \ldots, K-1\},$$

where w_k are weights and $\boldsymbol{q}_k$ nodes in a quadrature scheme. Note that f is only evaluated for the nodes $\boldsymbol{q}_k$, and these evaluations can be made once. Thereafter, one can experiment with the polynomial order since any c_n depends on the same evaluations of f.

Table 4.6 shows the various quadrature schemes offered by Dakota and Chaospy (recall that Turns does not support pseudo-spectral projection). All techniques for generating nodes and weights in Chaospy are available through the `cp.generate_quadrature` function. Suppose we want to generate optimal Gaussian quadrature nodes for the normal distribution. We then write

```
>>> nodes, weights = cp.generate_quadrature(
...             3, cp.Normal(0, 1), rule="Gaussian")
>>> print nodes
[[-2.33441422 -0.74196378  0.74196378  2.33441422]]
>>> print weights
[ 0.04587585  0.45412415  0.45412415  0.04587585]
```

Node and Weight Generators	Dakota	Turns	Chaospy
Clenshaw-Curtis quadrature [42]	yes	no	yes
Cubature rules [43]	yes	no	no
Gauss-Legendre quadrature [44]	yes	no	yes
Gauss-Patterson quadrature [45]	yes	no	yes
Genz-Keister quadrature [46]	yes	no	yes
Leja quadrature [47]	no	no	yes
Monte Carlo integration [1]	yes	no	yes
Optimal Gaussian quadrature [44]	yes	no	yes

Table 4.6: Various numerical integration strategies implemented in the three software toolboxes.

Most quadrature schemes are designed for univariate problems. To extend a univariate scheme to the multivariate case, integration rules along each axis can be combined using a tensor product. Unfortunately, such a product suffers from the curse of dimensionality and becomes a very costly integration procedure for large D. In higher-dimensional problems one can replace the full tensor product by a Smolyak sparse grid [48]. The method works by taking multiple lower order tensor product rules and joining them together. If the rule is nested, i.e., the same samples found at a low order are also included at higher order, the number

of evaluations can be further reduced. Another feature is to add anisotropy such that some dimensions are sampled more than others [49]. In addition to the tensor product rules, there are a few native multivariate cubature rules that allow for low order multivariate integration [43]. Both Dakota and Chaospy also support the Smolyak sparse grid and anisotropy.

Chaospy has support for construction of custom integration rules defined by the user. The `cp.rule_generator` function can be used to join a list of univariate rules using tensor grid or Smolyak sparse grid. For example, consider the trapezoid rule:

```
>>> def trapezoid(n):
...     X = np.linspace(0, 1, n+1)
...     W = np.ones(n+1)/n
...     W[0] *= 0.5; W[-1] *= 0.5
...     return X, W
...
>>> nodes, weights = trapezoid(2)
>>> print nodes
[ 0.   0.5  1. ]
>>> print weights
[ 0.25  0.5   0.25]
```

The `cp.rule_generator` function takes positional arguments, each representing a univariate rule. To generate a rule for the multivariate case, with the same one-dimensional rule along two axes, we do the following:

```
>>> mvtrapezoid = cp.rule_generator(trapezoid, trapezoid)
>>> nodes, weights = mvtrapezoid(2, sparse=True)
>>> print nodes
[[ 0.   0.5  1.   0.   0.   1. ]
 [ 0.   0.   0.   0.5  1.   1. ]]
>>> print weights
[ 0.     0.25   0.125  0.25   0.125  0.25 ]
```

Software for constructing and executing a general-purpose integration scheme is useful for several computational components in uncertainty quantification. For example, in Section 4.4 when constructing orthogonal polynomials using raw statistical moments, or calculating discretized Stieltjes' recurrence coefficients, numerical integration is relevant. Like the `ppf` function noted in Section 4.3, the moments and recurrence coefficients can be added directly into each distribution. However, when these are not available, Chaospy will automatically estimate missing information by quadrature rules, using the `cp.generate_quadrature` function described above.

To compute the Fourier coefficients and the polynomial chaos expansion, we use the `cp.fit_quadrature` function. It takes four arguments: the set of orthogonal polynomials, quadrature nodes, quadrature weights, and the user's function for evaluating the forward model (to be executed at the quadrature nodes). Note that in the case of the discretized Stieltjes method discussed in Section 4.4, the nominator $\mathbb{E}\left[\mathbf{\Phi}_n^2\right]$ in (4.13) can be calculated more accurately using recurrence coefficients [32]. Special numerical features like this can be added by including optional arguments in `cp.fit_quadrature`.

Point Collocation

The other non-intrusive approach to estimate the coefficients $\{c_k\}_{k\in I_K}$ is the point collocation method. One way of formulating the method is to require the polynomial expansion to equal the model evaluations at a set of collocation nodes $\{\boldsymbol{q}_k\}_{k\in I_K}$, resulting in an over-determined set of linear equations for the Fourier coefficients:

$$\begin{bmatrix} \Phi_0(\boldsymbol{q}_0) & \cdots & \Phi_N(\boldsymbol{q}_0) \\ \vdots & & \vdots \\ \Phi_0(\boldsymbol{q}_{K-1}) & \cdots & \Phi_N(\boldsymbol{q}_{K-1}) \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ c_N \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{q}_0) \\ \vdots \\ f(\boldsymbol{q}_{K-1}) \end{bmatrix}, \tag{4.14}$$

Unlike pseudo spectral projection, the locations of the collocation nodes are not required to follow any integration rule. Hosder [41] showed that the solution using Hammersley samples from quasi-Monte Carlo samples resulted in more stable results than using conventional pseudo-random samples. In other words, well placed collocation nodes might increase the accuracy. In Chaospy these collocation nodes can be selected from integration rules or from pseudo-random samples from Monte Carlo simulation, as discussed in Section 4.3. In addition, the software accepts user defined strategies for choosing the sampling points. Turns also allows for user-defined points, while Dakota has its predefined strategies.

The obvious way to solve the over-determined system in (4.14) is to use least squares minimization, which resembles the standard statistical linear regression approach of fitting a polynomial to a set of data points. However, from a numerical point of view, this might not be the best strategy. If the numerical stability of the solution is low, it might be prudent to use Tikhonov regularization [50], or if the problem is so large that the number of coefficients is very high, it might be useful to force some of the coefficients to be zero through least angle regression [51]. Being able to run and compare alternative methods is important in many problems to see if numerical stability is a potential problem. Table 4.7 lists the regression methods offered by Dakota, Turns, and Chaospy.

Generating a polynomial chaos expansion using linear regression is done using Chaospy's `cp.fit_regression` function. It takes the same arguments as `cp.fit_quadrature`, except that quadrature weights are omitted, and optional arguments define the rule used to optimize (4.14).

Model Evaluations

Irrespectively of the method used to estimate the coefficients c_k, the user is left with the job to evaluate the forward model (response function) f, which is normally by far the most computing-intensive part in uncertainty quantification. Chaospy does not impose any restriction on the simulation code used to compute the forward model. The only requirement is that the user can provide an array of values of f at the quadrature or collocation nodes. Chaospy users will usually wrap any complex simulation code for f in a Python function `f(q)`, where `q` is

Regression Schemes	Dakota	Turns	Chaospy
Basis Pursuit [52]	yes	no	no
Bayesian Auto. Relevance Determination [53]	no	no	yes
Bayesian ridge [54]	no	no	yes
Elastic Net [55]	yes	no	yes
Forward Stagewise [56]	no	yes	no
Least Absolute Shrinkage & Selection [51]	yes	yes	yes
Least Angle & Shrinkage with AIC/BIC [57]	no	no	yes
Least Squares Minimization	yes	yes	yes
Orthogonal matching pursuit [58]	yes	no	yes
Singular Value Decomposition	no	yes	no
Tikhonov Regularization [50]	no	no	yes

Table 4.7: List of supported regression methods for estimating Fourier coefficients

a node in probability space (i.e., `q` contains values of the uncertain parameters in the problem). For example, for pseudo-spectral projection, samples of f can be created as

```
samples = [f(node) for node in nodes.T]
```

or perhaps done in parallel if f is time consuming to evaluate:

```
import multiprocessing as mp
pool = mp.Pool(mp.cpu_count())
samples = pool.map(f, nodes.T)
```

The evaluation of all the f values can also be done in parallel with MPI in a distributed way on a cluster using the Python module like `mpi4py`. Both Dakota and Turns support parallel evaluation of f values, but the feature is embeded into the code, potentially limiting the customization options of the parallelization.

Extension of polynomial expansions

There is much literature that extends on the theory of polynomial chaos expansion [36]. For example, Isukapalli showed that the accuracy of a polynomial expansion could be increased by using partial derivatives of the model response [59]. This theory is only directly supported by Dakota. In Turns and Chaospy the support is indirect by allowing the user to add the feature manually.

To be able to incorporate partial derivatives of the response, the partial derivative of the polynomial expansion must be available as well. In both Turns and Chaospy, the derivative of a polynomial can be generated easily. This derivative can then be added to the expansion, allowing us to incorporate

Isukapalli's theory in practice. This is just an example on how manipulation of the polynomial expansions and model approximations can overcome the lack of support for a particular feature from the literature.

To be able to support many current and possible future extensions of polynomial chaos, a large collection of tools for manipulating polynomials must be available. In Dakota, no such tools exist from a user perspective. In Turns, there is support for some arithmetic operators in addition to the derivative. In Chaospy, however, the polynomial generated for the model response is of the same type as the polynomials generated in Sections 4.4 and 4.4, and the rich set of manipulations of polynomials is then available for $\hat{f}$ as well.

Beyond the analytical tools for statistical analysis of $\hat{f}$, either from the toolbox or custom ones by the user, there are many statistical metrics that cannot easily be expressed as simple closed-form formulas. Such metrics include confidence intervals, sensitivity indices, p-values in hypothesis testing, to mention a few. In those scenarios, it makes sense to perform a secondary uncertainty analysis through Monte Carlo simulation. Evaluating the approximation $\hat{f}$ is normally computationally much cheaper than evaluating the full forward model f, thus allowing a large number of Monte Carlo samples within a cheap computational budget. This type of secondary simulations are done automatically in the background in Dakota and Turns, while Chaospy does not feature automated tools for secondary Monte Carlo simulation. Instead, Chaospy allows for simple and computationally cheap generation of pseudo-random samples, as described in Section 4.3, such that the user can easily put together a tailored Monte Carlo simulation to meet the needs at hand. Within a few lines of Python code, the samples can be analyzed with the standard Numpy and the Scipy libraries [60] or with more specialized statistical libraries like Pandas [9], Scikit-learn [61], Scikit-statsmodel [62], and Python's interface to the rich R environment for statistical computing. For example, for the specific $\hat{f}$ function illustrated above, the following code computes a 90 percent confidence interval, based on 10^5 pseudo-random samples and Numpy's functionality for finding percentiles in discrete data:

```
>>> q_samples = cp.Normal(0,1).sample(10**5)
>>> samples = f_approx(*q_samples)
>>> p05, p95 = np.percentile(samples, [5, 95], axis=-1)
>>> print p05[:3]
[ 1.         1.0000004  1.0000016]
>>> print p95[:3]
[ 1.          1.00038886  1.00155544]
```

Since the type of statistical analysis of $\hat{f}$ often strongly depends on the physical problem at hand, we believe that the ability to quickly compose custom solutions by putting together basic building blocks is very useful in uncertainty quantification. This is yet another example of the need for a package with a strong focus on easy customization.

Descriptive Tools

The last step in uncertainty quantification based on polynomial chaos expansions is to quantify the uncertainty. In polynomial chaos expansion this is done by using the uncertainty in the model approximation `f_approx` as a substiute for the uncertainty in the model f. For the most popular statistical metrics, like mean, variance, correlation, a polynomial chaos expansion allows for analytical analysis, which is easy to calculate and has high accuracy. This property is reflected in all the three toolboxes. To calculate the expected value, variance and correlation of a simple (here univariate) polynomial approximation `f_approx`, with a normally distributed ξ_0 variable, we can with Chaospy write

```
>>> f_approx = fit_quadrature(orth, nodes, weights, samples)
>>> print f_approx
[q0, q0^2, q0^3]
>>> dist = Normal(0,1)
>>> print E(f_approx, dist)
[ 0.  1.  0.]
>>> print Var(f_approx, dist)
[  1.   2.  15.]
>>> print Corr(f_approx, dist)
[[ 1.          0.          0.77459667]
 [ 0.          1.          0.        ]
 [ 0.77459667  0.          1.        ]]
```

A list of supported analytical metrics is listed in Table 4.8.

Method	Dakota	Turns	Chaospy
Covariance/Correlation	yes	yes	yes
Expected value	yes	yes	yes
Conditional expectation	no	no	yes
Kurtosis	yes	yes	yes
Sensitivity index	yes	yes	yes
Skewness	yes	yes	yes
Variance	yes	yes	yes

Table 4.8: List of common statistical operators that can be used for analytical evaluation of polynomials.

4.5 Conclusion and Further Work

Until now there have only been a few real software alternatives for implementing non-intrusive polynomial chaos expansions. Two of the more popular implementations, Dakota and Turns, are both high-quality software that can be applied to a large array of problems. The present paper has introduced a new alternative: Chaospy. Its aim is to be an experimental foundry for scientists. Besides featuring a vast library of state-of-the-art tools, Chaospy allows for a

high degree of customization in a user-friendly way. Within a few lines of high-level Python code, the user can play around with custom distributions, custom polynomials, custom integration schemes, custom sampling schemes, and custom statistical analysis of the result. Throughout the text we have compared the built-in functionality of the three packages, and Chaospy do very well in this comparison, which is summarized in table 4.9. But the primary advantage of the package is the strong emphasis on offering well-designed software building blocks, with a high abstraction level, that can easily be combined to create tailored uncertainty quantification algorithms for new problems.

Although the primary aim of the software is to construct polynomial chaos expansions, the software is also a state-of-the-art toolbox for performing Monte Carlo simulation, either directly on the forward model or in combination with polynomial chaos expansions. Variance reduction techniques are included to speed up the convergence, and because Chaospy is based on Rosenblatt transformations, efficient quasi-Monte Carlo sampling is available for any distribution. Another novel feature of Chaospy is the ability to handle *stochastically dependent* input variables through a new mathematical technique.

Feature	**Dakota**	**Turns**	**Chaospy**
Distributions	11	26	64
Copulas	1	7	6
Sampling schemes	4	7.5	7
Orthogonal polynomial schemes	4	3	5
Numerical integration strategies	7	0	7
Regression methods	5	4	8
Analytical metrics	6	6	7

Table 4.9: A summary of the various features in Dakota, Turns and Chaospy.

Acknowledgement

The work is supported by funding from Statoil ASA through the Simula School of Research and Innovation, and by a Center of Excellence grant from the Research Council of Norway through the Center for Biomedical Computing.

Thanks also go to Vinzenz Gregor Eck, Stuart Clark, Karoline Hagane, Samwell Tarly, and a random distribution of unnamed bug fixers for their contributions.

Bibliography

[1] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method*. Wiley-Interscience, 2 ed., Dec. 2007.

[2] D. Xiu, *Numerical Methods for Stochastic Computations: A Spectral Method Approach*. Princeton University Press, July 2010.

[3] M. S. Eldred, A. A. Giunta, B. G. van Bloemen Waanders, S. F. Wojtkiewicz, W. E. Hart, and M. P. Alleva, *DAKOTA, a multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 4.1 reference manual*. Sandia National Laboratories Albuquerque, NM, 2007.

[4] G. Andrianov, S. Burriel, S. Cambier, A. Dutfoy, I. Dutka-Malen, E. De Rocquigny, B. Sudret, P. Benjamin, R. Lebrun, and F. Mangeant, "Open TURNS, an open source initiative to Treat Uncertainties, Risks N Statistics in a structured industrial approach," in *Procedings ESREL2007 safety and reliability conference. Stavanger, Norway*, 2007.

[5] B. J. Debusschere, H. N. Najm, P. P. Pbay, O. M. Knio, R. G. Ghanem, and O. P. Le Matre, "Numerical challenges in the use of polynomial chaos representations for stochastic processes," *SIAM Journal on Scientific Computing*, vol. 26, no. 2, pp. 698–719, 2004.

[6] P. R. Conrad and Y. M. Marzouk, "Adaptive Smolyak pseudospectral approximations," *SIAM Journal on Scientific Computing*, vol. 35, no. 6, pp. A2643–A2670, 2013.

[7] B. Efron, "Bootstrap methods: another look at the jackknife," *Annals of Statistics*, pp. 1–26, 1979.

[8] J. Feinberg and H. P. Langtangen, "Chaospy Software Package for Uncertainty Quantification," 2014. *https://github.com/hplgit/chaospy*.

[9] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, 2012.

[10] P. Laux, G. Jackel, R. M. Tingem, and H. Kunstmann, "Impact of climate change on agricultural productivity under rainfed conditions in CameroonA

method to improve attainable crop yields by planting date adaptations," *Agricultural and Forest Meteorology*, vol. 150, no. 9, pp. 1258–1271, 2010.

[11] R. C. Boardman and J. E. Vanna, "A review of the application of copulas to improve modelling of non-bigaussian bivariate relationships (with an example using geological data)," in *International Congress on Modelling and Simulation*, 2011.

[12] J. Dobric and F. Schmid, "A goodness of fit test for copulas based on Rosenblatt's transformation," *Computational Statistics & Data Analysis*, vol. 51, no. 9, pp. 4633–4642, 2007.

[13] P. Achard and E. De Schutter, "Complex parameter landscape for a complex neuron model," *PLoS Computational Biology*, vol. 2, July 2006.

[14] J. Feinberg and H. P. Langtangen, "Multivariate Polynomial Chaos with Dependent Variables," 2015. Unbublished journal article URL: *http://bit.ly/1Bkp72S*.

[15] M. Rosenblatt, "Remarks on a Multivariate Transformation," *Annals of Mathematical Statistics*, vol. 23, no. 3, pp. 470–472, 1952.

[16] R. B. Nelsen, *An introduction to copulas*. Springer, 1999.

[17] A. J. Lee, "Generating random binary deviates having fixed marginal distributions and specified degrees of association," *The American Statistician*, vol. 47, no. 3, pp. 209–215, 1993.

[18] J. Feinberg and S. Clark, "RoseDist: Generalized Tool for Simulating with Non-Standard Probability Distributions," in *International Congress on Modelling and Simulation* (J. Piantadosi, R. S. Anderssen, and J. Boland, eds.), (Adelaide, Australia), pp. 367–372, Dec. 2013. http://www.mssanz.org.au/modsim2013/A7/feinberg.pdf.

[19] S. Tezuka, *Uniform random numbers: Theory and practice*. Kluwer Academic Publishers Boston, 1995.

[20] S. Galanti and A. Jung, "Low-discrepancy sequences: Monte Carlo simulation of option prices," *Journal of Derivatives*, vol. 5, no. 1, pp. 63–83, 1997.

[21] J. H. Halton, "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals," *Numerische Mathematik*, vol. 2, no. 1, pp. 84–90, 1960.

[22] J. M. Hammersley, "Monte Carlo methods for solving multivariable problems," *Annals of the New York Academy of Sciences*, vol. 86, no. 3, pp. 844–874, 1960.

[23] C. B. Haselgrove, "A method for numerical integration," *Mathematics of Computation*, pp. 323–337, 1961.

[24] N. M. Korobov, "The approximate calculation of multiple integrals using number theoretic methods," *Doklady Academii Nauk SSSR*, vol. 115, pp. 1062–1065, 1957.

[25] H. Niederreiter, "Point sets and sequences with small discrepancy," *Monatshefte fur Mathematik*, vol. 104, no. 4, pp. 273–337, 1987.

[26] I. M. Sobol, "On the distribution of points in a cube and the approximate evaluation of integrals," *USSR Computational Mathematics and Mathematical Physics*, vol. 7, no. 4, pp. 86–112, 1967.

[27] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[28] D. Xiu and J. S. Hesthaven, "High-order collocation methods for differential equations with random inputs," *SIAM Journal on Scientific Computing*, vol. 27, p. 1118, 2005.

[29] R. Askey and J. A. Wilson, *Some basic hypergeometric orthogonal polynomials that generalize Jacobi polynomials*. Amer Mathematical Society, 1985.

[30] W. Gautschi, "Construction of Gauss-Christoffel quadrature formulas," *Mathematics of Computation*, vol. 22, p. 251, Apr. 1968.

[31] M. Bertran, "Note on Orthogonal Polynomials in v-Variables," *SIAM Journal on Mathematical Analysis*, vol. 6, no. 2, pp. 250–257, 1975.

[32] W. Gautschi, *Orthogonal Polynomials: Computation and Approximation*. Oxford University Press, USA, June 2004.

[33] T. J. Stieltjes, "Quelques recherches sur la thorie des quadratures dites mcaniques," *Ann. Sci. cole Norm. Sup. (3)*, vol. 1, pp. 409–426, 1884.

[34] D. Xiu, D. Lucor, C. H. Su, and G. E. Karniadakis, "Stochastic modeling of flow-structure interactions using generalized polynomial chaos," *Journal of Fluids Engineering*, vol. 124, p. 51, 2002.

[35] J. Feinberg and H. P. Langtangen, "Uncertainty Quantification of Diffusion in Layered Media by a New Method Based on Polynomial Chaos Expansion," in *Seventh National Conference on Computational Mechanics MekIT'13* (H. I. Andersson and B. Skallerud, eds.), (Norwegian University of Science and Technology), Akademika Publishing, May 2013.

[36] D. Xiu, "Fast numerical methods for stochastic computations: a review," *Communications in computational physics*, vol. 5, no. 2-4, pp. 242–272, 2009.

[37] S. Sakamoto and R. Ghanem, "Polynomial chaos decomposition for the simulation of non-Gaussian nonstationary stochastic processes," *Journal of Engineering Mechanics*, vol. 128, no. 2, pp. 190–201, 2002.

[38] R. Ghanem and P. D. Spanos, *Stochastic finite elements: a spectral approach*. Courier Dover Publications, Aug. 2003.

[39] J. Back, F. Nobile, L. Tamellini, and R. Tempone, "Stochastic spectral Galerkin and collocation methods for PDEs with random coefficients: a numerical comparison," in *Spectral and High Order Methods for Partial Differential Equations*, pp. 43–62, Springer, 2011.

[40] D. Gottlieb and S. A. Orszag, *Numerical Analysis of Spectral Methods: Theory and Applications*. Society for Industrial and Applied Mathematics, 1977.

[41] S. Hosder, R. W. Walters, and M. Balch, "Efficient sampling for non-intrusive polynomial chaos applications with multiple uncertain input variables," in *Proceedings of the 48th Structures, Structural Dynamics, and Materials Conference*, vol. 125, (Honolulu, HI), 2007.

[42] C. W. Clenshaw and A. R. Curtis, "A method for numerical integration on an automatic computer," *Numerische Mathematik*, vol. 2, no. 1, pp. 197–205, 1960.

[43] A. H. Stroud, *Approximate calculation of multiple integrals*, vol. 431. Prentice-Hall Englewood Cliffs, NJ, 1971.

[44] G. H. Golub and J. H. Welsch, *Calculation of Gauss quadrature rules*, vol. 23. Mathematics of Computation, 1967.

[45] T. Patterson, "The optimum addition of points to quadrature formulae," *Mathematics of Computation*, vol. 22, no. 104, pp. 847–856, 1968.

[46] A. Genz and B. Keister, "Fully Symmetric Interpolatory Rules for Multiple Integrals over Infinite Regions," *Journal of Computational Applied Mathematics*, vol. 72, 1996.

[47] A. Naraya and J. Jakeman, "Adaptive Leja sparse grid construction for stochastic collocation and high-dimensional approximation," *arXiv e-print*, 2014. https://cfwebprod.sandia.gov/cfdocs/CompResearch/docs/1404.5663v1.pdf.

[48] S. A. Smolyak, "Quadrature and interpolation formulas for tensor products of certain classes of functions," in *Doklady Akademii Nauk SSSR*, vol. 4, p. 123, 1963.

[49] J. Burkardt, "The "combining coefficient" for anisotropic sparse grids," tech. rep., Virginia Tech. 125, 2009.

[50] R. M. Rifkin and R. A. Lippert, "Notes on regularized least squares," *Journal of Linear Algebra*, vol. 18, pp. 281–288, June 2009.

[51] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[52] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[53] D. P. Wipf and S. S. Nagarajan, "A new view of automatic relevance determination," in *Advances in Neural Information Processing Systems*, pp. 1625–1632, 2007.

[54] D. J. C. MacKay, "Bayesian interpolation," *Neural computation*, vol. 4, no. 3, pp. 415–447, 1992.

[55] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[56] T. Hastie, J. Taylor, R. Tibshirani, and G. Walther, "Forward stagewise regression and the monotone lasso," *Electronic Journal of Statistics*, vol. 1, pp. 1–29, 2007.

[57] H. Zou, T. Hastie, and R. Tibshirani, "On the degrees of freedom of the lasso," *Annals of Statistics*, vol. 35, no. 5, pp. 2173–2192, 2007.

[58] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[59] S. S. Isukapalli, *Uncertainty analysis of transport-transformation models*. PhD thesis, Rutgers, The State University of New Jersey, 1999.

[60] E. Jones, T. Oliphant, and P. Peterson, "SciPy: Open source scientific tools for Python," *AIP Publishing*, 2001. URL: http://scipy.org/.

[61] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[62] S. Seabold and J. Perktold, "Statsmodels: econometric and statistical modeling with Python," in *Proceedings of the 9th Python in Science Conference*, pp. 57–61, 2010.

III