Uncertainty Quantification

Personal summary by Gilberto Lem

Summer 2018

Contents

1 Introduction

1.1 Uncertainty Quantification

Uncertainty Quantification determines how likely certain outcomes are if some aspects of the system are not exactly known.

The number of uncertain parameters is called **stochastic dimensionality**.

UQ in general is slow, because one needs to run the model several times to get good results. To speed up the process there are three options:

- Make the deterministic model faster
- Use surrogate model instead of expensive high-fidelity model (tradeoff for accuracy)
- Use clever UQ method with low amount of evaluation points (maybe tradeoff for accuracy)

1.2 Basics of Probability Theory and Statistics

A **probability space** is a triplet $(\Omega, \mathscr{F}, \mathscr{P})$,where:

- Ω is the sample space, the set of all possible single outcomes
- $\mathscr{F}$ (σ-algebra) is the combinatorial of all the elements of Ω
- $\mathscr{P}$ is the probability measure, $\mathscr{F} \to [0, 1]$

1.2.1 Univariate Concepts

A **random variable** is a function X which maps $\Omega \to \mathbb{R}$. It represents an outcome of a random phenomenon as a number.

We can represent the probability of obtaining $X = x$, being x a real number, as $f_X(x)$. If X is continuous, this function is called the **probability density function (PDF)**, if X is discrete, is called the **probability mass function (PMF).**

The value of f_X has to be always greater or equal than zero, and its accumulate value over all its doimain has to be 1.

Every random variable has an associated **Cumulative Distribution Function (CDF)**, represented by $F_X(x)$. It represents the probability that the random variable X has a value less or equal than a specified x, i.e. $F_X(x) = \mathscr{P}\{X \leq x\}$. This can be represented as an integral:

$$F_X(x) = \int_{-\infty}^{x} f_X(s)ds$$

The **expectation** of a continuous random variable X is defined as:

$$\mu := \mathbb{E}[X] = \int x f_X(x)dx$$

The **variance** of X is the expected value of the square of the variation of X with respect to its mean:

$$\sigma^2 := \mathrm{Var}(X) = \int (x - \mathbb{E}[X])^2 f_X(x) dx = \mathbb{E}[X^2] - \mathbb{E}[X]^2$$

If X is discrete, the **sample mean** and **sample variance** are defined:

$$\bar{X} = \frac{1}{n} \sum_i X_i, \qquad S^2 = \frac{1}{n-1} \sum_i (X_i - \bar{X})^2$$

1.2.2 Multivariate Concepts

X can be generalized to a n-dimensional random vector $\boldsymbol{X}$, which maps $\Omega \to \mathbb{R}^n$. $\boldsymbol{X}$ consists of n random variables.

The **covariance** of two random variables X and Y is

$$cov(X,Y) = \mathbb{E}[\,(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\,] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

Covariance is high when X and Y are "similar".

If you normalize this covariance you get the **Pearson correlation coefficient:**

$$\rho_{XY} = \frac{cov(X,Y)}{\mathrm{Var}(X)\mathrm{Var}(Y)}$$

which have values $[-1, 1]$.

Independence Let A and B denote two events. If the probability for A to occur is the same if B happens or if B does not happen, A and B are **independent.** If additionally, both events have the same distribution of probability, they are **independent and identically distributed (iid)** random variables.

1.2.3 Biased vs. unbiased estimators

An **estimator** is a procedure to construct estimates for a quantity q based on random samples. An **estimate** is a realization of the estimator. Example estimators are the mean and the variance.

An estimator is called **biased** if its expected value is not the same as the parameter to be estimated.

Correspondingly, an **unbiased** estimator is when the expected value of the estimator of a sample is equal to the real value of the parameter we are estimating.

For an unbiased mean, if we take the mean of a sample, we would expect the same as the mean of the population.

In the following equations, $\bar{X}$ and S^2 represent unbiased estimators:

$$\mathbb{E}[\bar{X}] = \mu, \qquad \mathbb{E}[S^2] = \sigma^2$$

If we take the following definitions of $\bar{X}$ and S^2 :

$$\bar{X} = \frac{1}{n} \sum_i X_i \qquad S^2 = \frac{1}{n} \sum_i \left(X_i - \bar{X} \right)^2$$

$\bar{X}$ is unbiased but S^2 is biased. To correct for this, we multiply our definition of sample variance by $n/(n-1)$ to end up with:

$$S^2 = \frac{1}{n-1} \sum_i \left(X_i - \bar{X} \right)^2$$

Note that numpy's var function computes the biased version of the variance. When we use that funcion we have to correct it.

2 Sampling Methods

Sampling methods consist in select a subset of "individuals" from a statistical "population", to estimate characteristics of the whole population.

2.1 Monte Carlo

Monte Carlo Sampling consists in generating samples with a pseudo-random number generator from a given probability distribution. After that compute deterministic computation with these samples and aggregate the results.

One application of Monte Carlo Sampling is Monte Carlo Integration. It computes the value of an integral of a function over a domain via sampling of the function. The error of the approximation is given by the **Root Mean Squared Error:**

$$RMSE \coloneqq \sqrt{\mathbb{E}\left[(I - \hat{I}_f)^2\right]}$$

Or with the equivalent expression:

$$RMSE \approx \frac{\hat{\sigma}_f}{\sqrt{N}}$$

with

$$\hat{\sigma}_f = \frac{1}{N-1}\sum_i \left(f(U_i) - \hat{I}_f\right)^2$$

Monte Carlo Sampling is easy, robust and embarassingly parallel, but its convergence rate is $\mathcal{O}(\frac{1}{\sqrt{N}})$, although independent of the input dimension, is very slow.

So, how can we reduced the RMSE given by Monte Carlo? We could make our simulations faster, also we could increase N (which is not desirable because it would require more deterministic simulations), decrease $\hat{\sigma}_f$ (variance-reduction techniques) or improve the random number generation (quasi-montecarlo).

2.2 Variance-Reduction Techniques

2.2.1 Antithetic Sampling

We can use a different sampling (choosing of X values) to reduce the variance. This type of sampling assumes that a continuous PDF $f_X(x)$ is symmetric around c. This implies that the symmetric counterpart of x is $\tilde{x} = 2c - x$,and that $f_X(x) = f_X(\tilde{x})$.

Thus, we can sample $n/2$ times from $f_X(x)$ and the remaining $n/2$ samples are obtained via reflection.

For example for an uniform distribution in $[0,1]$,we would get $n/2$ values for X, and the rest of the $n/2$ values would be obtained with the relation $\tilde{U} = 1 - U$.

Taking into account the symmetry of the function helps reducing the variance of the sampling.

2.2.2 Stratified Sampling

This type of sampling divides the interval of sampling to prevent clustering in a particular region of the interval. Suppose our interval is $[x_i, x_f]$. We would choose a $\lambda \in (0,1)$ and make two intervals. We take λn samples in $[x_i, \lambda(x_f - x_i)]$ and $(1-\lambda)n$ samples in $[\lambda(x_f - x_i), x_f]$.

2.2.3 Control Variates

This does not reduce the variance through sampling, but through an auxilary function. We take as an illustratory example the estimation of the integral of $f(x)$ over a domain. Here we introduce a control function ϕ, which can be easily integrated in that domain.

$$I = \int f(x)dx = \int (f - \phi)dx + \int \phi dx$$

We know the exact integral for ϕ, so we would only need to sample the difference. If we look at the variance of the difference:

$$\text{Var}(f - \phi) = \text{Var}(f) + \text{Var}(\phi) - 2cov(f, \phi)$$

we note that if $cov(f, \phi)$ is high, then $\text{Var}(f - \phi) < \text{Var}(f)$, and we would have reduced the variance. So we only need to find a “similar function” to f.

2.2.4 Importance Sampling

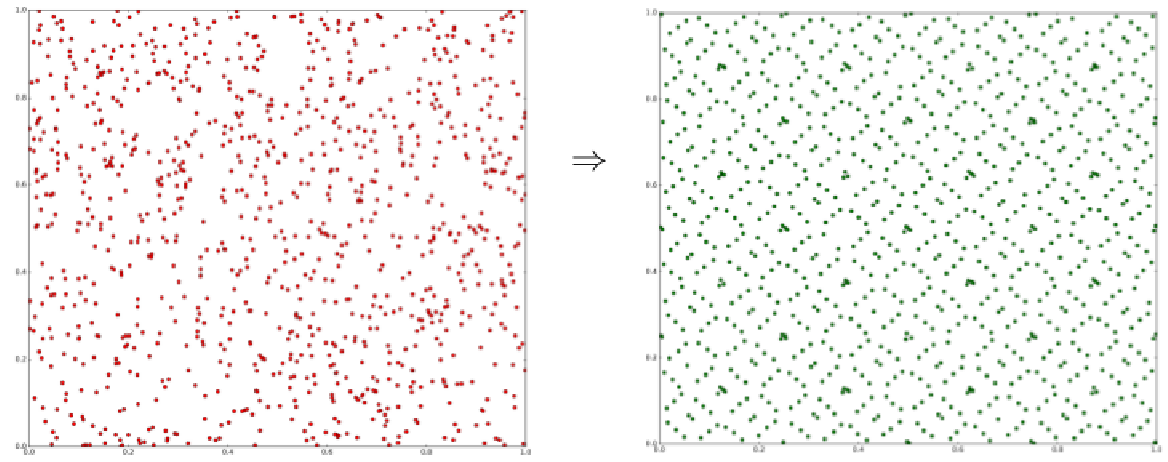
In a similar way to control variates, if we in Monte Carlo instead of sampling from the uniform distribution, sample from another function g_X, and f and g_X have similar shapes, then variance is reduced.

$$I = \int f(x)dx = \int \frac{f(x)}{g_X(x)} g_X(x)dx$$

In this case, the sampling would be done according to the g_X,and the function to evaluate would be $f(x)/g_x(x)$.

2.3 Quasi Monte-Carlo

Consists in alternative sampling techniques. Instead of using pseudo-random samples as in standard Monte-Carlo, which would result in a nonuniform sampling, we use deterministic samples. Examples are Fibonacci generators, latin hypercube sampling, Sobol’ sequences and Halton sequences.



In this course we will use techniques based on **low-discrepancy sequences**. For them, the error is $\mathcal{O}\left(\frac{\log(N)^d}{N}\right)$, where d is the dimension.

One example of low-discrepancy sequences are the Halton Sequences. They consist in choosing a prime number p and dividing the interval in p subintervals, and then each interval in other p subintervals, and so on until reaching the desired number of points.

In chaospy, it is very easy to generate QMC samples. If for example for generating 1000 Halton sequence-based samples on an uniform distribution:

```
distr = chaospy.Uniform()
samples = distr.sample(size=1000, rule='H')
```

3 Basic Interpolation and Quadrature

3.1 Interpolation

Interpolating consists in having some data points and estimating unknown values between those points. One way to do this is using polynomials, because they are analytically simple and computationally cheap.

Given a continuous function, and a grid where we know the value of the function, there exists a **unique** polynomial of degree N that satisfies the value of the function on those points. This polynomial can be expressed as a sum of **Lagrange cardinal polynomials**.

For a certain grid there is a unique polynomial, but we have freedom in choosing the grid. For defining the error of the interpolation, we can relate the error of our interpolation taking grid G with respect to the real function in terms of the error of an interpolation with the *best approximation polynomial* (with the same number of nodes).

$$Error_G \leq (1 + \Lambda_N(G))\,(Error_{\text{Best grid}})$$

Λ is the **Lebesgue constant** relative to the grid G, and contains all the information about the effect of the grid choice.

Lagrange interpolation on an uniform grid is not always a good idea, it can have strong variations near the limits of the interval (this is called the **Runge phenomenon**). Possible remedies for this is to use non-uniform grids (as Chebyscheff nodes) or different basis functions (maybe functions with local support, e.g. splines).

3.2 Quadrature

The idea of Quadrature is to approximate the definite integral of a function based on function evaluations. Quadrature uses rules that integrate polynomials exactly.

The general form of quadrature is a weighted sum of function values, however, as the function evaluation can be very computationally expensive, we need rules with small error of quadrature using few evaluations.

Gaussian Quadrature uses not only the weights of the quadrature as degrees of freedom, but also the nodes on the grid. If we make a quadrature considering $N + 1$ points (quadrature of degree N), this technique can integrate exactly polynomials of degree up to $2N + 1$. Gaussian quadrature has the following form:

$$\int f(x)w(x)\mathrm{d}x = \sum_{i=0}^{N} w_i f(x_i) + \epsilon$$

Again, $\epsilon = 0$ for polynomials up to degree $2N + 1$.

Note that we are integrating not only f, but a product with w. This is a weight function, and this makes it is useful for UQ when we take w as a probabilty distribution. This would be equivalent to taking the expected value of $f(x)$ with respect to the probability distribution $w(x)$. On the right side, w_i corresponds to the weights given to each point evaluation, and x_i correspond to the nodes.

Getting the nodes We can find the nodes if we find a family of polinomials $p_i(x)$ that is orthogonal with respect to $w(x)$ (i.e. $\int p_i(x)p_j(x)w(x)dx = \delta_{ij}$). Then the nodes are simply the roots of the polynomial p_{N+1}. For w constant (as in the uniform probability distribution) our orthogonal family is the **Legendre polynomials**. For w gaussian (as in the normal distribution) our family is the **Hermite polynomials**.

Getting the weights The weights can be computed with the Lagrange cardinal polynomials evaluated at the nodes, integrated with respect to the weight function (at the end of the day we are integrating a $(2N+1)$-degree interpolation of $f(x)$).

4 Polynomial Chaos

Forward UQ consists on having a stochastic input that goes into a model f, and a stochastic output which we want to characterize in the cheapest way possible. Montecarlo methods (and its improved versions) have slow convergence and need many samples. Another idea is to approximate our model f with a series of polynomials:

$$f(t,\omega) \approx \sum_{n=0}^{N-1} \hat{f}_n(t)\phi_n(\omega)$$

where t represent the independent variable(s) of our model, ω is the stochastic input variable (with distribution $\rho(\omega)$), $\hat{f}_n(t)$ are the coefficients of the polynomials and $\phi_n(\omega)$ are orthogonal polynomials of degree n. Note that the dependency on the independent variable is comprised only by the coefficients, and the dependency on the stochastic input variable is only in the polynomials.

The orthogonal polynomials $\phi_n(\omega)$ would be chosen according to the input distribution of the stochastic variable ($\rho(\omega)$). Polynomials which are orthogonal with respect to the inner product defined with $\rho(\omega)$. For example for a uniform distribution we would choose Legendre polynomials, for a normal distribution Hermite polynomials, and so on. Computation of other families of polynomials can be made with the Stieltjes' **three-term recursion relation**, which is satisfied by all orthogonal polynomials, and can be gotten with a numerically stable method, or instead we could use other schemes as a Gram-Schmidt algorithm.

Once we have $\hat{f}_n(t)$, we can evaluate statistical moments of our resulting function $f(t,\omega) \approx \sum_{n=0}^{N-1} \hat{f}_n(t)\phi_n(\omega)$ pretty easily.

- **Expectation value:** $\mathbb{E}[f(t,\omega)] = \hat{f}_0(t)$ (just like in fourier series)
- **Variance:** $\mathrm{Var}[f(t,\omega)] = \sum_{n=1}^{N-1} \hat{f}_n^2(t)$ (all the mixed terms vanish because of the orthogonality of our polynomials)

4.1 Computing the coefficients: The Pseudo Spectral Approach

As our polynomials represent an orthonormal basis, we could compute our coefficients with a simple inner product with respect to $\rho(\omega)$:

$$\hat{f}_n(t) = \langle f(t,\omega), \phi_n(\omega)\rangle_\rho = \int f(t,\omega)\phi_n(\omega)\rho(\omega)d\omega$$

This way of getting the coefficients is called the **pseudo-spectral approach**.

That inner product is computed through an integral, and as f can be expensive or even a black box, we rather use quadrature to solve it. Gaussian quadrature is specially adequate for this. We just need to get the nodes and weights (that depend solely on the probability distribution ρ) and then we would only have to evaluate f once at every node. This would yield:

$$\hat{f}_n = \sum_{k=0}^{K-1} w_k f(t, \omega_k) \phi_n(\omega_k)$$

Note that the integral is with respect to ω, and thus the grid ω_k that we get as the roots of the polynomials, represent values of our stochastic variable ω.

Choosing maximum order of polynomials and quadrature

Consider that the number of nodes to evaluate the integral (order of the quadrature) is completely independent of the number of polynomials used to represent our model (order of the interpolation):

- Using K nodes to evaluate the quadrature would be as integrating exactly a polynomial of degree up to $2K-1$, this polynomial being an interpolation of $f(t, \omega)\phi_n(\omega)$ with respect to ω.
- Independent of that, we have the polynomial of degree $N - 1$ that interpolates $f(t, \omega)$ with respect to ω.

Choose K: It may be very expensive to evaluate $f(t, \omega_k)$, but once these values are known, with them we can compute $\hat{f}_n$, and after that we can evaluate our interpolation of $f(t, \omega)$ wherever we want with low computational effort. Thus, our K determines the computational effort.

Choose N: On the other side, generally $\hat{f}_n(t)$ decay exponentially with respect to n, so few coefficients (low N) are sufficient. As a rule of thumb we can use $N \approx K/2$.

4.2 Computing the Coefficients: The Stochastic Galerkin Method

The quadrature we use in the pseudo-spectral approach introduces an additional error. The **Stochastic Galerkin Method** consists in *not* solving the integrals. Instead, it treats the polynomial expansion as an Ansatz to the model (it requires access to the model equations or code), together with another polynomial chaos expansion but for the uncertain input, and takes advantage of the orthogonality of our polynomial basis to get a transformed model, which can be solved to get the coefficients we look for.

In the example of the linear damped oscillator, which is this:

$$\begin{gathered} \tfrac{dx}{dt} = v \\ \tfrac{dv}{dt} = f \cos(\omega_0 t) - cv - kx \end{gathered}$$

with $x(0) = x_0$ and $v(0) = v_0$, if we have an uncertainty in the damping coefficient c, our Stochastic Galerkin approach would be the following:

- Make a polynomial chaos expansion of our stochastic input: $c = c(\omega) = \sum_n c_n \phi_n(\omega)$
- Make a polynomial chaos expansion of our model: $x(t, \omega) = \sum_n \hat{x}_n(t)\phi_n(\omega)$, $v(t, \omega) = \sum_n \hat{v}_n(t)\phi_n(\omega)$
- Insert the expansions to the ODEs and the Initial Conditions
- Make an inner product of all the equations with ϕ_j to take advantage of orthogonality
- Make analytic work, and transform the model. From 2 coupled ODEs we would get $2N$ coupled ODEs (N is the size of our polynomial (sub)space).
- Solve the new model for $\hat{x}_j$ and $\hat{v}_j$
- Get the statistical properties of our approximation in the same way as we did in the pseudospectral approach.

4.3 Comparison

	Pseudo-Spectral	**Stochastic Galerkin**
Getting Coefficients	Quadrature	System of Diff. Equations
Error Sources	- Series truncation of the model - Quadrature error	- Series truncation of the model
Requisites	Output of the model	Equations or code of the model
Implementation	Reusable	Different for each model
Summary	Easier	More Accurate

4.4 Multivariate Polynomial Chaos Expansion

Polynomial expansion can be generalized for d dimensions. Now instead of n we have a vector $\boldsymbol{n} = (n_1, ..., n_d) \in \mathbb{N}^d$, and our polynomials are product of one-dimensional polynomials (orthogonal with respect to the combination of all their indices). Generally for this approach one chooses a given N as a global order, and then we have a finite number of combinations of $(n_1, ..., n_d)$,such that $n_1 + ... + n_d \leq N$. This number of combinations is notated by $P := \begin{pmatrix} d+N \\ d \end{pmatrix}$. The computational cost, which is K is only indirectly related to P.

5 Sparse Grids

When you have more than one dimension, the evaluation of the quadratures for the Pseudo Spectral Approach in gPCE becomes extremely expensive. In the standard approach (full tensor-grid approach), we have to evaluate $M = K^d$ points, where K is the number of quadrature points in one direction.

Sparse Grids (SG) techniques consist in not using all the points as in the standard approach, but to discard the components that do not contribute much to the overall solution.

5.1 Method

Imagine $\mathcal{U}^{(i)}$ is a linear operator (in our case a quadrature scheme) along the dimension i. Then our whole quadrature, in d dimensions would be:

$$\mathcal{U}^{(\boldsymbol{d})} = \mathcal{U}^{(1)} \otimes ... \otimes \mathcal{U}^{(d)}$$

As $\mathcal{U}^{(i)}$ is generally available only theoretically, we can approximate it with $\mathcal{U}_k^{(i)}$,where $\mathcal{U}_k^{(i)} \to \mathcal{U}^{(i)}$ for $k \to \infty$.

Then we could define a telescoping sum:

$$\mathcal{U}^{(i)} = \mathcal{U}_0^{(i)} + (\mathcal{U}_1^{(i)} - \mathcal{U}_0^{(i)}) + (\mathcal{U}_2^{(i)} - \mathcal{U}_1^{(i)}) + ... = \sum_{k=0}^{\infty} \Delta_k^{(i)}$$

(setting $\Delta_{-1}^{(i)} = 0$). In d-dimensions, this would be:

$$\mathcal{U}^{(\boldsymbol{d})} = \sum_{k_1=0}^{\infty} \Delta_{k_1}^{(1)} \otimes ... \otimes \sum_{k_d=0}^{\infty} \Delta_{k_d}^{(d)}$$

The product of sums can be viewed as a sum of products, where each term would have a multiindex $\boldsymbol{k}$:

$$\mathcal{U}^{(\boldsymbol{d})} = \sum_{|\boldsymbol{k}|_1=0}^{\infty} \Delta_{k_1}^{(1)} \otimes ... \otimes \Delta_{k_d}^{(d)}$$

where $|\boldsymbol{k}|_1$ is the L1-norm of the multiindex: $|\boldsymbol{k}|_1 = \sum_{i=1}^{d} |k_i|$.

So, this is an infinite sum of products of differences of linear operators. For taking the terms that contribute significantly, we will truncate the sum. For this we would choose those instances of $\boldsymbol{k}$ that contribute significantly to the overall solution. If we want a specific level L of truncation, this set of significant multiindices is

$$\mathcal{K} = \left\{ \boldsymbol{k} \in \mathbb{N}^d \colon |\boldsymbol{k}|_1 \leq L + d - 1 \right\},$$

5.2 Further comments

- If a grid with a certain level includes points from the grid at a lower level, the grid is said to be **nested**. Otherwise it is not nested.

- The magnitude of the savings acheived with this method depends on the dimension d and on the SG parametric choices (w/o nesting, w/o boundary points, growth over grid levels). In a 10 dimensional example using nested Clenshaw-Curtis points, with 9 points in each dimension, the number of points in SG is 1.6×10^3, versus 3×10^9 of the full tensor grid.

- This technique is used in low- to mid-size dimensionality (4 to 20). It is worth noting that although we discussed the usage of SG for quadrature, the technique is not restricted to this. In UQ, it could be used to make a sparse interpolation, just taking into account the most importants point into a grid to make the interpolation. This is an alternative for using in polynomial chaos expansion.

6 Sensitivity Analysis

Sensitivity Analysis attempts to determine which parameters contribute the most to the output. In this context, which stochastic parameters contribute the most to the stochastic output, which would be useful to:

- **Evaluate robustness** of the underlying model with respect to a certain parameter i.e. that the outputs are similar for similar inputs.
- **Dimensionality reduction** of the stochastic model. With this we would set deterministic values for insensitive parameters.
- Finding regimes in the space of input parameters that **optimize** responses or uncertainties

6.1 Local vs. Global Sensitivity Analysis

Local SA aims to assess the sensitivity of **the output** with respect to the inputs. This is generally made using **gradients** (of outputs w.r.t. inputs). It is a useful technique for stochastic dimensionality reduction (**one input at a time**) for before doing the forward uncertainty propagation, but it may be expensive.

Global SA analizes, instead of the output, a measure of **uncertainty of the output** (e.g. variance). It quantifies the contribution of **every input** to the output uncertainty, and relies only on properties of the model and not on experimental data.

6.2 Variance-based Global Sensitivity Analysis

ANOVA decomposition consists in defining our model $f(t,\boldsymbol{\omega})$ as a sum of functions, each one with dependencies only on one combination of the elements of $\boldsymbol{\omega}$:

$$f(t,\boldsymbol{\omega}) = f_0(t) + \sum_{i=1}^{d} f_i(t,\omega_i) + \sum_{1\le i<j\le d} f_{ij}(t,\omega_i,\omega_j) + \sum_{1\le i_1<\ldots<i_s\le d} f_{i_1\ldots i_s}(t,\omega_{i_1},\ldots,\omega_{i_s}) + \ldots + f_{12\ldots d}(t,\boldsymbol{\omega})$$

We can make this decomposition unique if we impose orthogonality between the different elememts $f_{i_1\ldots i_m}$. These terms can be computed with a process *similar* to a Gram-Schmidt orthogonalization, and each one requires an integration over a domain that has less dimensions than our domain Ω (e.g. for $f_{i_1\ldots i_m}$ we would integrate on a domain that does not include the dimensions 1 to m, sort of like a projection to the remaining dimensions).

The total variance of f would be get with the following expression:

$$\mathrm{Var}[f(t,\boldsymbol{\omega})] = \sum_{1\le i_1<\ldots<i_s\le d} D_{i_1\ldots i_s}(t),$$

where

$$D_{i_1\ldots i_s}(t) = \int_{\Gamma^s} f^2_{i_1\ldots i_s}(t,\omega_{i_1},\ldots,\omega_{i_s})d\omega_{i_1}\ldots d\omega_{i_s}$$

We can use this variance decomposition to asses the contribution of each term to the total variance. This is what **Sobol' indices** try to do.

Local Sobol' indices measure the contribution of an individual input or a specific combination of inputs:

$$S_{i_1\ldots i_s}(t) = \frac{D_{i_1\ldots i_s}(t)}{\mathrm{Var}[f(t,\boldsymbol{\omega})]}$$

They always have to be less or equal than one. On the other side, **total Sobol' indices** measure the individual contribution **and** the interactions between inputs:

$$S_i(t) = \sum_{i\in\{i_1\ldots i_s\}} S_{i_1\ldots i_s}(t)$$

The sum of total sobol indices can be greater than one because one considers multiple times the same variable.

6.3 Computing the $f_{i_1\ldots i_s}$ terms

As we mentioned, these terms are computed through integrals. The standard approach to do this would be through Monte Carlo sampling. However this is computationally expensive. Instead, we can take advantage of the fact that using gPC we could get a decomposition of $f(t,\boldsymbol{\omega})$ in an orthogonal basis, which would provide a unique ANOVA decomposition. If we have the following gPCE:

$$f(t,\boldsymbol{\omega}) \approx \sum_{|\boldsymbol{n}|_1=0}^{N-1} \hat{f}_{\boldsymbol{n}}(t)\phi_{\boldsymbol{n}}(\boldsymbol{\omega}),$$

we already know how to compute the variance of the terms:

$$D_i(t) = \sum_{\boldsymbol{n}\in A_i} \hat{f}^2_{\boldsymbol{n}}(t)$$

where

$$A_i = \left\{\boldsymbol{n}\in\mathbb{N}^d : \boldsymbol{n}_i > 0\right\}$$

i.e. the set of $\boldsymbol{n}$ vectors where the i-th component of $\boldsymbol{n}$ is not zero.

Then the Sobol' indices are simply:

$$S_i(t) = \frac{D_i(t)}{\mathrm{Var}[f(t,\boldsymbol{\omega})]}$$

Note that Sobol' indices are time dependent. The sensitivity of the output to a variable depends on the time of the output.

7 Random Fields

8 Software

9 Bayesian Inverse Problems