Stochastic Finite Element Methods & Data-driven models in Engineering Applications

Lecture Notes

version date: 12.10.2022
(under constant development)

Contents

Chapter 1

Probability Theory

1.1 Probability space

Let us consider an experiment and denote the set of all possible outcomes of the experiment as Θ, called the *sample space*. The sample space is classified into three categories according to its cardinality, that is, *finite*, *countably infinite* (or *denumerable*) and *uncountable*. More specifically, Θ is said to be finite if the number of elements $\theta \in \Theta$ is finite, countably infinite if a bijection (one-to-one correspondence and onto) between every $\theta \in \Theta$ and the infinite set of natural numbers can be established and uncountable if it is not countable. For instance, for the experiment of rolling a die, Θ is finite since $\Theta = \{1, 2, 3, 4, 5, 6\}$. On the other hand, for the experiment of performing stress tests on concrete specimens to estimate their yield stress, the sample space is uncountable.

A non-empty collection of subsets $\mathscr{F}$ of Θ is called a *σ-algebra* on Θ, if it satisfies the following properties:

1. the empty set belongs to $\mathscr{F}$: $\emptyset \in \mathscr{F}$
2. $\mathscr{F}$ is closed under complements: $A \in \mathscr{F} \Rightarrow A^c \in \mathscr{F}$
3. $\mathscr{F}$ is closed under countable unions: $A_i \in \mathscr{F},\ i \in I,\ I =$ a countable set$\Rightarrow \cup_{i \in I} A_i \in \mathscr{F}$

Every element of $\mathscr{F}$ is called *event*, or *$\mathscr{F}$-measurable* subset of Θ. The pair $(\Theta, \mathscr{F})$ constitutes a *measurable space*.

Let us also define a real-valued function μ on $\mathscr{F}$ with the following property:

$$\mu\left(\cup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu\left(A_i\right) \text{ for } A_i \in \mathscr{F},\ A_i \cap A_j = \emptyset,\ i \neq j \tag{1.1}$$

The function μ is called a *measure* and the triple $(\Theta, \mathscr{F}, \mu)$ a *measure space*. Also, a measure with the property $\mu(\Theta) < \infty$ is called a *finite measure* and the scaled version of this measure, $\mu(A)/\mu(\Theta)$, takes values in the $[0, 1]$ domain. A set function $P : \mathscr{F} \to [0, 1]$ that satisfies the following properties:

1. $P(\Theta) = 1$
2. $P\left(\cup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P\left(A_i\right)$ for $A_i \in \mathscr{F},\ A_i \cap A_j = \emptyset,\ i \neq j$

is called a *probability measure* or simply *probability* and the triple $(\Theta, \mathscr{F}, P)$ constitutes the *probability space*.

1.2 Properties of Probability Measure

In this section some properties of the probability measure P, which are particularly useful for applications, are provided.

- $P(A) \leq P(B),\ A \subseteq B,\ A, B \in \mathscr{F}$
- $P(A) = 1 - P(A^c),\ A \in \mathscr{F}$
- $P(A \cup B) = P(A) + P(B) - P(A \cap B),\quad A, B \in \mathscr{F}$
- The above equation generalizes to the union of multiple events as

$$P(\cup_{i=1}^{n} A_i) = \sum_{i=1}^{n} P(A_i) - \sum_{i=2}^{n}\sum_{j=1}^{i-1} P(A_i \cap A_j) + \sum_{i=3}^{n}\sum_{j=2}^{i-1}\sum_{k=1}^{j-1} P(A_i \cap A_j \cap A_k) - \cdots + (-1)^{n+1} P\left(\cap_{q=1}^{n} A_q\right),\ A_i \in \mathscr{F}$$

- Given a partition of Θ, A_i such that $\cup_{i=1}^{n} A_i = \Theta,\ A_i \cap A_j = \emptyset,$ for $i \neq j$ and an event $B \in \mathscr{F}$

$$P(B) = \sum_{i=1}^{n} P(B \cap A_i)$$

- Let $(\Theta, \mathscr{F}, P)$ be a probability space and an event $B \in \mathscr{F}$. A new probability measure can be defined on $(\Theta, \mathscr{F})$ under the assumption that B has occurred. This new measure is referred to as the *conditional probability*. The probability of the event A occurring conditional on the event B is given by the following formula:

$$P(A|B) = \frac{P(A \cap B)}{P(B)},\quad A, B \in \mathscr{F} \text{ and } P(B) > 0$$

- Let $A_i \in \mathscr{F},\ i = 1, 2, \cdots, n$ be a partition of Θ.

  The *law of total probability* states:

$$P(B) = \sum_{i=1}^{n} P(B \cap A_i) = \sum_{i=1}^{n} P(B|A_i)P(A_i)$$

  The *Bayes formula* states:

$$P(A_j|B) = \frac{P(A_j)P(B|A_j)}{P(B)} = \frac{P(A_j)P(B|A_j)}{\sum_{i=1}^{n} P(A_i)P(B|A_i)}$$

1.3 Random Variables

A (continuous) *random variable* defined over a probability space $\{\Theta, \mathscr{F}, P\}$ is a function X with domain Θ and codomain $\mathbb{R}$, that is:

$$\begin{aligned} X :&\Theta \to \mathbb{R} \\ &\theta \mapsto X(\theta) \end{aligned}$$

The *cumulative distribution function (cdf)* of a random variable X is the function:

$$F(x) = P(X \leq x), \quad -\infty < x < \infty$$

which satisfies the following properties:

1. $0 \leq F(x) \leq 1$
2. F is a non-decreasing function of x
3. $F(-\infty) = 0$ and $F(\infty) = 1$
4. F is right continuous: $lim_{x \to x_0^+} F(x) = F(x_0)$ for every x

The derivative of the cdf is called the *probability density function (pdf)*

$$\begin{aligned} f(x) &= \frac{dF(x)}{dx} \\ &= \lim_{\Delta x \to 0} \frac{P(x \leq X \leq x + \Delta x)}{\Delta x} \end{aligned}$$

A pdf satisfies the normalization condition

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

Also, the following relations are equivalent

$$\begin{aligned} P(a \leq x \leq b) &= F(b) - F(a) \\ &= \int_a^b f(x)dx \end{aligned}$$

A d-dimensional vector whose coordinates are random variables is called a *random vector* $\boldsymbol{X}$:

$$\begin{aligned} \boldsymbol{X} :&\Theta \to \mathbb{R}^d \\ &\theta \mapsto \boldsymbol{X}(\theta) \end{aligned}$$

The definitions of the pdf and cdf are straightforwardly extended for random vectors. Let $\boldsymbol{X} = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^d$ be a random vector with d coordinates. Its *joint cumulative distribution function* is defined as:

$$F(x_1, x_2, \ldots, x_d) = P(X_1 \leq x_1, \ldots, X_d \leq x_d), \quad -\infty < x_1, \ldots, x_d < \infty$$

and the corresponding *joint probability density function* is:

$$f(x_1, x_2, \dots, x_d) = \frac{\partial^d}{\partial_1 \cdots \partial_d} F(x_1, \dots, x_d)$$

Also, the joint cdf can be obtained by appropriate integration of the joint pdf, that is:

$$F(x_1, x_2, \dots, x_d) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_d} f(u_1, \dots, u_d) du_1 \dots du_d, \quad -\infty < x_1, \dots, x_d < \infty$$

The *marginal cumulative distribution function* of the variable X_m, for $m = 1, ..., d$ is obtained by integrating out the rest of the variables in the random vector. More specifically,

$$\begin{aligned} F_{X_m}(x_m) &= P(X_m \leq x_m) \\ &= \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{x_m} \cdots \int_{-\infty}^{\infty} f(u_1, \dots, u_d) du_1 \dots du_d, \quad -\infty < x_m < \infty \end{aligned}$$

and the corresponding *marginal probability density function* as

$$f_{X_m}(x_m) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \dots, x_d) dx_1 \dots dx_{m-1} dx_{m+1} \dots dx_d$$

Now, let us consider a random vector $\boldsymbol{X} \in \mathbb{R}^d$ whose coordinates X_i satisfy the following relation:

$$F(x_1, \dots, x_d) = F_{X_1}(x_1) \dots F_{X_d}(x_d)$$

or, equivalently,

$$f(x_1, \dots, x_d) = f_{X_1}(x_1) \dots f_{X_d}(x_d)$$

Then, the X_i's are said to be *independent* random variables and the above equations are the necessary and sufficient conditions for *independence*. Intuitively, the notion of independence suggests that the value of a random variable X_i, obtained through an experiment, does not give any further knowledge about the value of another random variable X_j in the same experiment.

Lastly, given two random variables X, Y and their joint pdf f, the *conditional probability density function* of Y given X is defined by the following relation:

$$f_{Y|X} = \begin{cases} \dfrac{f(x,y)}{f_X(x)}, & 0 < f_X(x) < \infty \\ 0, & \text{otherwise} \end{cases}$$

1.4 Moments

Let us first define a linear operator acting on random variables defined over a probability space $(\Theta, \mathscr{F}, P)$. This operator, acting on a (continuous) random variable X, is defined as:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x f(x) dx$$

and is called *expectation operator*, or simply *expectation* of a random variable X. For random variables with finite expectations, it is straightforward to prove that the following properties of $\mathbb{E}$ hold:

1. $\mathbb{E}[X+Y] = \mathbb{E}[X] + \mathbb{E}[Y]$
2. $\mathbb{E}[aX] = a\mathbb{E}[X]$
3. $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$ iff X, Y independent

The expected value of a function of X, $g(X)$ given that X has a probability density function $f(x)$ is given by:

$$\mathbb{E}[g(X)] = \int_{-\infty}^{\infty} g(x)f(x)dx$$

The above relations extend to multidimensional cases, where f is replaced by the corresponding joint density.

We define the n-th *moment about the origin* of a random variable X as the expectation of X^n and denote it as μ_n, that is:

$$\mu_n = \mathbb{E}[X^n] = \int_{-\infty}^{\infty} x^n f(x)dx$$

For $n = 1$, μ_1, usually denoted with m, is called the *mean value* of X and for $n = 2$, μ_2 is the *mean square*.

Similarly, we define the n-th *moment about the mean* or n-th *central moment* of a random variable X as the expectation of $(X-m)^n$ and denote it as s_n, that is:

$$s_n = \mathbb{E}[(X-m)^n] = \int_{-\infty}^{\infty} (x-m)^n f(x)dx$$

The first central moment s_1 is zero and the second central moment s_2 is called the *variance*, usually denoted as σ^2, where $\sigma = \sqrt{s_2}$ is the *standard deviation.* The third and forth central moments are used in the definition of the *skewness* $\gamma = \frac{s_3}{\sigma^3}$ and *kurtosis* $\kappa = \frac{s_4}{\sigma^4}$. Skewness and kurtosis are both descriptors of the shape of a pdf, where the former measures the asymmetry of a pdf and the latter its 'tailedness'.

Some other important quantities that can be defined using the moments are:

- The *coefficient of variation* for a random variable X

$$c.o.v = \frac{\sigma}{m}$$

- The *covariance* between two random variables X, Y

$$Cov[X,Y] = \mathbb{E}[XY] - \mathbb{E}[X]E[Y]$$

  Note that in the case $X = Y$, $Cov[X,X] \doteq Var[X]$ where $Var[X]$ is the variance of X and

$$Var[X] = \mathbb{E}[X^2] - \mathbb{E}^2[X] = \mu_2 - m^2$$

- The *correlation coefficient*

$$\rho = \frac{\mathbb{E}[XY]}{\sqrt{Var[X]}\sqrt{Var[Y]}}$$

Lastly, we define the *conditional expectation* of a random random variable Y given the event $X = x$ has occurred, as

$$\begin{aligned}\mathbb{E}[Y|X = x] &= \int_{-\infty}^{\infty} yf(y|x)dy \\ &= \int_{-\infty}^{\infty} \frac{yf(x, y)}{f_X(x)}dy\end{aligned}$$

1.5 $\mathscr{L}_2$-Probability space

The random variables defined over a probability space $(\Theta, \mathscr{F}, P)$ satisfying the property:

$$\mathbb{E}[X^2] = \int_{-\infty}^{\infty} |x|^2 f(x)dx < \infty$$

are elements of a vector space, henceforth denoted as $\mathscr{L}_2$, defined over the field of real numbers $\mathbb{R}$. In $\mathscr{L}_2$ the addition operation corresponds to the usual addition of two real-valued random variables, and the operation of scalar multiplication corresponds to the usual multiplication of a real-valued random variable by a real number.

The vector space is also endowed with an inner-product structure induced be the expectation operator $E[\cdot]$:

$$\langle X, Y\rangle \doteq \mathbb{E}[XY]$$

One can easily verify from the definition of the expectation that for $X, Y, Z \in \mathscr{L}_2$ the following inner product properties are satisfied:

1. $\langle X, Y\rangle = \langle Y, X\rangle$
2. $\langle X, X\rangle \geq 0$ and $\langle X, X\rangle = 0$ iff $P(X = 0) = 1$ $(X \equiv 0)$
3. $\langle aX, Y\rangle = a\langle X, Y\rangle$ for $a \in \mathbb{R}$
4. $\langle X + Y, Z\rangle = \langle X, Z\rangle + \langle Y, Z\rangle$

The $\mathscr{L}_2$ space is also *complete*, which means that every *Cauchy sequence* in $\mathscr{L}_2$ converges in $\mathscr{L}_2$. As a consequence, $\mathscr{L}_2$ is a *Hilbert space* (complete inner product space) and, thus, there exists an orthonormal basis that spans $\mathscr{L}_2$. This particular conclusion will prove very useful in the subsequent formulations.

1.6 Additional literature

- *Introduction to Probability*, D. Bertsekas, J. Tsitsiklis, 2ed, 2008
- *Probability, Random Variables and Stochastic Processes*, A. Papoulis, S.U. Pillai, 4ed, 2002

Chapter 2

Stochastic Processes

2.1 Introduction

In probability theory, a stochastic process is a mathematical object which is defined as the collection of an infinite number of random variables. Their study originated from the need to model dynamic systems that seemed to randomly change over time. The motion of a particle immersed in a fluid is such an example. Each time the particle is immersed in the fluid, it will follow a different trajectory as a result of the random microscopic collisions with the molecules of the fluid. This particular example of a stochastic process is known as the *Brownian motion*, named after the botanist Robert Brown, who first observed it in 1827. Since then stochastic processes have found applications in various fields in science and engineering, including image and signal processing, earthquake prediction, weather forecasting and finance.

The theory of stochastic processes is an important field in mathematics combining elements of different branches such as probability theory, measure theory, set theory, Fourier analysis and real analysis. The aim of this chapter is to revisit the fundamental concepts on stochastic processes, that will be used throughout these notes.

2.2 Definitions

Let us recall, that a random variable was defined as a rule for assigning to every outcome θ of an experiment a number $X(\theta)$. In certain applications, however, the experiment 'evolves' with respect to some deterministic parameter t belonging in an interval I. For instance, this would be the case of an engineering system subjected to random dynamic loads over a time interval $I \subseteq \mathbb{R}^+$. Then, the response at a material point of this system would be described by a collection of random variables $\{X(t)\}$ indexed by the parameter $t \in I$, rather than a single random variable. This collection of random variables over the interval I is called a *stochastic process* and is denoted by $\{X(t), t \in I\}$ or X. In this regard, a stochastic process can be seen as a generalization of the concept of a random variable, in the sense that it assigns to every outcome θ of the experiment, a function $X(t, \theta)$, referred to as a *realization* or a *sample function.*

A stochastic process is called a *continuous-time real-valued stochastic process* if I is the real axis and each random variable in the collection takes on values in $\mathbb{R}$. If I is the set of integers, then the process is called *discrete-time real-valued stochastic process.* Furthermore, if the process takes on values in $\mathbb{R}^d$, $d > 1$, the the process is called $\mathbb{R}^d$-*valued stochastic* *process* and the notation $\{\boldsymbol{X}(t), t \in I\}$ or, simply, $\boldsymbol{X}$ will be used. Lastly, if $\boldsymbol{X}$ is indexed by some space coordinate $\boldsymbol{s} \in D \subseteq \mathbb{R}^n$ rather than time t, then $\{\boldsymbol{X}(\boldsymbol{s}), \boldsymbol{s} \in D\}$ is called a *random field*. The summarize the above we provide the following two definitions:

Let $\boldsymbol{X} : I \times \Theta \to \mathbb{R}^d$ be a vector-valued function of two arguments, $t \in I$ and $\theta \in \Theta$, where $I \subseteq \mathbb{R}$ and $(\Theta, \mathcal{F}, P)$ denotes a probability space. Then, $\boldsymbol{X}$ is said to be an $\mathbb{R}^d$-valued stochastic process, if $\boldsymbol{X}(t)$ is an $\mathbb{R}^d$-valued random variable on the probability space $(\Theta, \mathcal{F}, P)$ for each $t \in I$.

Similarly, let $\boldsymbol{X} : D \times \Theta \to \mathbb{R}^d$ be a vector-valued function of two arguments, $\boldsymbol{s} \in D$ and $\theta \in \Theta$, where $D \subseteq \mathbb{R}^n$ and $(\Theta, \mathcal{F}, P)$ denotes a probability space. Then, $\boldsymbol{X}$ is said to be an $\mathbb{R}^d$-valued random field, if $\boldsymbol{X}(\boldsymbol{s})$ is an $\mathbb{R}^d$-valued random variable on the probability space $(\Theta, \mathcal{F}, P)$ for each $\boldsymbol{s} \in D$.

2.3 Statistics of Stochastic Processes

Let $\{\boldsymbol{X}(t), t \geq 0\}$ be an $\mathbb{R}^d$-valued stochastic process defined over a probability space $(\Theta, \mathcal{F}, P)$. Let $n \geq 1$ be an integer, $t_i \geq 0$, $i = 1, ..., n$, be arbitrary distinct times and set $\mathcal{X}_n = (\boldsymbol{X}(t_1), ..., \boldsymbol{X}(t_n))$. The *finite dimensional distributions of order* n of $\boldsymbol{X}$ are the distributions of the random vectors $\mathcal{X}_n$:

$$F_n\left(\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}; t_1, ..., t_n\right) = P\left(\cap_{i=1}^n \{\boldsymbol{X}(t_i) \in \prod_{k=1}^d (-\infty, x_{i,k}]\}\right) \tag{2.1}$$

where $\boldsymbol{x}^{(i)} = (x_{i,1}, ..., x_{i,d}) \in \mathbb{R}^d$. For example, if X is a real-valued $(d = 1)$ stochastic process, its finite dimensional distributions are:

$$F_n\left(x_1, ..., x_n; t_1, ..., t_n\right) = P\left(X(t_1) \geq x_1, ..., X(t_n) \geq x_n\right) \tag{2.2}$$

Then, the corresponding *finite dimensional densities* of X can be derived as:

$$f_n\left(x_1, ..., x_n; t_1, ..., t_n\right) = \frac{\partial^n}{\partial x_1 \cdots \partial x_n} F_n\left(x_1, ..., x_n; t_1, ..., t_n\right) \tag{2.3}$$

A complete determination of the statistical properties of a stochastic process can only be attained through the knowledge of the function $F_n\left(x_1, ..., x_n; t_1, ..., t_n\right)$ for every x_i,t_i and n. However, in most applications, the available information on the process allow us to estimate at most its first and second order finite dimensional densities. From these, the following averages can be obtained as follows:

- The mean $m(t)$ of X is the expected value of each random variable $X(t)$:

  $$m(t) = \mathbb{E}\left[X(t)\right] = \int_{-\infty}^{\infty} x f(x; t) dx$$

  It should be mentioned that from every process X, a zero-mean process can be obtained, called *centered* process, simply by subtracting the mean, that is

  $$\tilde{X}(t) = X(t) - m(t)$$

- The autocorrelation $R_X(t_1, t_2)$ of X is the expected value of the product $X(t_1)X(t_2)$:

$$R_X(t_1, t_2) = \mathbb{E}[X(t_1)X(t_2)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x_1 x_2 f(x_1, x_2; t_1, t_2) dx_1 dx_2$$

  In the case $t_1 = t_2 = t$ the autocorrelation function gives the mean-square moment of each random variable $X(t)$, that is, $R_X(t, t) = \mathbb{E}[X^2(t)]$.

- The autocovariance $C_X(t_1, t_2)$ of X is the expectation of the product $(X(t_1) - m(t_1))$ $(X(t_2) - m(t_2))$, which is the covariance of the random variables $X(t_1)$ and $X(t_2)$:

$$\begin{aligned} C_X(t_1, t_2) &= \mathbb{E}\left[(X(t_1) - m(t_1))(X(t_2) - m(t_2))\right] \\ &= \int_{-\infty}^{\infty} (x_1 - m(t_1))(x_2 - m(t_2)) f(x_1, x_2; t_1, t_2) dx_1 dx_2 \end{aligned}$$

  After some algebra, it can be shown that:

$$C_X(t_1, t_2) = R_X(t_1, t_2) - m(t_1)m(t_2)$$

  and in the case $t_1 = t_2 = t$, $C(t, t)$ equals the variance of each random variable $X(t)$. Also, it is evident, that for zero mean processes $C_X(t_1, t_2) \equiv R_X(t_1, t_2)$.

- The correlation coefficient $\rho_X(t_1, t_2)$ is defined as the ratio:

$$\rho_X(t_1, t_2) = \frac{C_{X(t_1,t_2)}}{\sqrt{C_{X(t_1,t_1)}C_{X(t_2,t_2)}}}$$

- Correlation and covariance between two stochastic process X and Y, known as *cross-correlation* and *cross-covariance* are defined in a similar fashion:

$$R_{XY}(t_1, t_2) = \mathbb{E}\left[(X(t_1))(Y(t_2)))\right]$$

  and

$$C_{XY}(t_1, t_2) = \mathbb{E}\left[(X(t_1) - m_X(t_1))(Y(t_2) - m_Y(t_2))\right]$$

2.4 Classes of Stochastic Processes

2.4.1 Stationary processes

A stochastic process X is called *stationary in the strict sense* or *strictly stationary* if

$$f_n(x_1, ..., x_n; t_1, ..., t_n) = f_n(x_1, ..., x_n; t_1 + \tau, ..., t_n + \tau) \tag{2.4}$$

for any $n \geq 1$ and time shift τ. The above definition implies that a process is strictly stationary if its joint pdf does not change when shifted in time. This assumption is quite restrictive and rarely met in real life phenomena. A milder assumption is that of *wide-sense stationarity* or *weak stationarity.*

A stochastic process X is called *stationary in the wide sense* or *weakly stationary* if its mean is constant

$$\mathbb{E}\left[X(t)\right] = m \ (= cnst) \tag{2.5}$$

and its autocorrelation depends only on the time lag $\tau = t_1 - t_2$

$$R_X(t_1, t_2) = R(t_1 - t_2) = R(\tau) = \mathbb{E}\left[X(\tau)^2\right] \tag{2.6}$$

From equations (2.5) and (2.6) that the variance of a wide-sense stationary process is constant with respect to time

$$Var\left[X(t)\right] = C_X(t,t) = \sigma^2 \ (= cnst) \tag{2.7}$$

2.4.2 Ergodic processes

A stochastic process X is said to be *ergodic* if the ensemble averages are equal to time averages. In other words, if every statistical property of the process can be obtained from one realization (of sufficient length) of the process. Weaker ergodicity conditions are that of *ergodicity of the mean* and *ergodicity of the autocorrelation* expressed through equations (2.8) and (2.9), respectively :

$$m = \mathbb{E}\left[X(t)\right] = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} X(t)dt \tag{2.8}$$

$$R(\tau) = \mathbb{E}\left[X(t)X(t+\tau)\right] = \lim_{T\to\infty} \frac{1}{2T}\int_{-T}^{T} X(t)X(t+\tau)dt \tag{2.9}$$

Note that an ergodic process is always stationary, but the reverse is not always true, as shown in fig. 2.1.

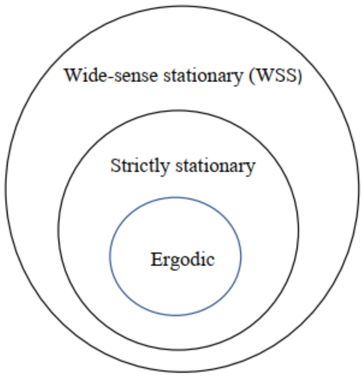


Figure 2.1: ergodic $\Rightarrow$ strictly stationary $\Rightarrow$ wide-sense stationary

2.4.3 Gaussian processes

A stochastic process X is said to be *Gaussian* if the joint pdf of the random variables $\{X(t_1), ..., X(t_n)\}$ is Gaussian for any n and t_i, $i = 1, ..., n$. A weakly-stationary Gaussian process possesses the favorable property of being completely defined by its mean value m, its standard deviation σ and its autocorrelation function. The pdf of the random variable $X(t)$ is given by

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp^{-\frac{(x-m)^2}{2\sigma^2}} \tag{2.10}$$

This implies that a weakly-stationary Gaussian process is also stationary in the strict sense.

2.4.4 Translation processes

Generally, when a stochastic process X is not Gaussian it is practically impossible to estimate the joint pdf of the random variables. However, a class of non-Gaussian stochastic processes with given marginal distribution and second moment information can be defined via a nonlinear marginal transformation (translation), g, of an underlying Gaussian field, $G(t)$, that is

$$X(t) = g\left(G(t)\right) = F^{-1} \circ \Phi\left(G\right) \tag{2.11}$$

where F is an arbitrary distribution with density f, Φ is the cdf of the Gaussian distribution $N(0,1)$ and G is a Gaussian process with $\mathbb{E}[G(t)] = 0$ and $\mathbb{E}[G(t)^2] = 1$. The marginal distribution of X is F and the finite dimensional density of order n of X is given by

$$f_n(x_1, ..., x_n) = \frac{1}{\sqrt{(2\pi)^n det(\boldsymbol{\rho})}} \prod_{p=1}^{n} \frac{f(x_p)}{\phi(y_p)} exp\left(-\frac{1}{2}\boldsymbol{y}^T\boldsymbol{\rho}^{-1}\boldsymbol{y}\right) \tag{2.12}$$

where, $\boldsymbol{\rho} = \mathbb{E}\left[G(t_p)G(t_q)\right]$, $y_p = \Phi^{-1} \circ F(x_p),\ p,q = 1, ..., n$, ϕ is the density of $N(0,1)$, and $\boldsymbol{y} = (y_1, ..., y_n)$.

However, the existence of a translation field with the desired characteristics is not always guaranteed. This is due to the fact that the marginal distribution $X(t)$ imposes constraints to its correlation structure. Therefore, the following compatibility equation must be satisfied between F and $R_X(\tau)$

$$R_X^T(\tau) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F^{-1} \circ \Phi\left(G_1\right) F^{-1} \circ \Phi\left(G_2\right) f(G_1, G_2; R_G(\tau)) dG_1 dG_2 \tag{2.13}$$

with $G_1 = G(t)$, $G_2 = G(t+\tau)$ and $f(G_1, G_2; R_G(\tau))$ denotes the joint density of G_1 and G_2. If F and $R_X(\tau)$ are proven to be incompatible then one has to resort to translation fields that match approximately the target marginal distribution and/or autocorrelation function.

2.5 Power spectral Density

Let X be a weakly stationary stochastic process in $L^2(\Theta, \mathcal{F}, P)$. It can be easily verified that the autocorrelation function $R_X(t_1, t_2)$ is positive definite, that is

$$\sum_{i=1}^{n}\sum_{j=1}^{n} R_X(t_i, t_j) c_i c_j \geq 0 \tag{2.14}$$

for all finite sequences of points $t_1, ..., t_n$ and all choices of real numbers $c_1, ..., c_n$.

Bochner's Theorem states that a continuous function $r : \mathbb{R} \to \mathbb{C}$ is positive definite if and only if it admits a representation of the form

$$r_X(\tau) = \int_{-\infty}^{\infty} e^{i\omega\tau} \mathcal{S}_X(\omega) d(\omega) \tag{2.15}$$

where $\mathcal{S}_X$ is a real-valued, positive and bounded function. The autocorrelation function R_X of a weakly stationary process satisfies the conditions of Bochner's theorem and therefore it has a representation of the form of eq. (2.15). In this context, the function $\mathcal{S}_X$ is called the *spectral density function* of X and the variable ω is the *angular frequency*. Under this prism, the autocorrelation and the spectral density functions are Fourier pairs that satisfy the following equations, known as Wiener-Khinchin relations:

$$R_X(\tau) = \int_{-\infty}^{\infty} e^{i\omega\tau} \mathcal{S}_X(\omega) d\omega \tag{2.16}$$

$$\mathcal{S}_X(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-i\omega\tau} R_X(\tau) d\tau \tag{2.17}$$

Using Euler's formula, $e^{i\omega\tau} = cos(\omega\tau) + isin(\omega\tau)$ and using the fact that $sin(\omega\tau)$ is odd, while $cos(\omega\tau)$, $R_X(\tau)$ and $\mathcal{S}_X(\omega)$ are even functions, then the integrals of the above equations can be simplified as:

$$R_X(\tau) = 2\int_{0}^{\infty} cos(\omega\tau) \mathcal{S}(\omega) d\omega \tag{2.18}$$

$$\mathcal{S}_X(\omega) = \frac{1}{\pi} \int_{0}^{\infty} cos(\omega\tau) R(\tau) d\tau \tag{2.19}$$

From eq. (2.18) one observes that the area under $\mathcal{S}_X(\omega)$ is $R_X(0)$. If $\mathbb{E}[X(t)] = 0$, this area is the variance of X. So, in essence, the spectral density of $\mathcal{S}_X$ tells us how the "energy" $\mathbb{E}[X^2(t)] = R_X(0)$ of the process X is distributed along the values of ω [1]. Now, since $\mathcal{S}_X(\omega)$ is an even function and negative values of ω lack any physical interpretation, instead of $\mathcal{S}_X(\omega)$, we prefer to use the one-sided spectral density $G_X(\omega)$ defined as:

$$G_X(\omega) = \begin{cases} 2\mathcal{S}_X(\omega), & \omega \geq 0 \\ 0, & \omega < 0 \end{cases} \tag{2.20}$$

For example, a particularly common correlation function used to describe wide-sense stationary processes is the Gaussian correlation function given by the relation:

$$R_X(\tau) = \sigma^2 e^{-\frac{\lfloor \tau \rfloor}{\lambda}} \tag{2.21}$$

where σ^2 is the variance of the process and λ is called the correlation length parameter. In this case, the corresponding two-sided power spectrum $\mathcal{S}_X(\omega)$ and one-sided power spectrum $G_X(\omega)$ are:

[1]This sentence will make more sense once we have talked about the Spectral Representation method.

$$\mathcal{S}_X(\omega) = \frac{\sigma^2\lambda}{\pi(\lambda^2\omega^2+1)}, \quad -\infty < \omega < \infty \tag{2.22}$$

$$G_X(\omega) = \frac{2\sigma^2\lambda}{\pi(\lambda^2\omega^2+1)}, \quad \omega \geq 0 \tag{2.23}$$

$$\tag{2.24}$$

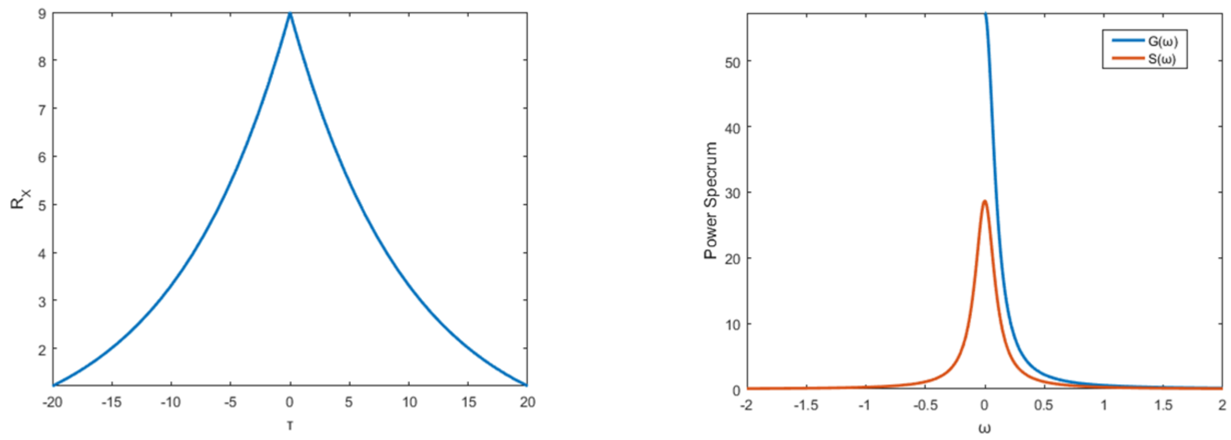


Figure 2.2: Gaussian correlation function and corresponding power spectra for $\sigma = 3$ and $\lambda = 10$

2.6 Additional literature

- *Stochastic Calculus*, M. Grigoriu, 2002
- *Probability, Random Variables and Stochastic Processes*, A. Papoulis, S.U. Pillai, 4ed, 2002

Chapter 3

Series Expansions of Stochastic Processes

3.1 Intuition

In the previous chapters, we saw that the complete probabilistic description of a random variable is given by its corresponding probability density function. On the other hand, for a random process $\{X(t), t \in I\}$, or just $X(t)$, we need all joint density functions of all the combinations of the random variables that compose the process, in order to fully determine it. Since a random process involves an infinite number of random variables, it becomes evident that we cannot know all of its joint pdf's. Therefore most applications of engineering or scientific interest, which include stochastic processes, are mathematically and computationally intractable.

In this section, we will explore ways to represent a stochastic process as a series of a finite number of random variables. To achieve this goal, we will rely on the additional structure (stationarity, ergodicity or L^2-integrability) that the process may possess. Ultimately, we will end up with a good approximation of the original process that would also allow us to perform computations.

3.2 The Karhunen-Loève series expansion

The first series expansion method that we will discuss is the Karhunen-Loève expansion. This method applies to stationary processes [1] and exploits the autocorrelation structure of the process. The Karhunen-Loève expansion of a zero-mean random process $X(t)$ is based on the spectral decomposition of its covariance function[2] defined as

$$C_X(t,s) = \sigma_X(t) \cdot \sigma_X(s) \cdot \rho(t,s) \tag{3.1}$$

where ρ is the correlation coefficient. By definition, $C_X(t,s)$ is bounded, symmetric and non-negative definite. We can associate to C_X a linear operator acting on functions $f(t)$ as follows:

[1] it can be extended to the case of non-stationary processes

[2] autocorrelation and autocovariance are the same thing here, since we talk about zero-mean processes

$$[T_{C_X} f](t) = \int_{\mathscr{D}} C_X(t,s) f(s) ds \tag{3.2}$$

where $\mathscr{D}$ is the domain in which the stochastic processes is defined. Then, Mercer's theorem states that C_X has the following representation or eigendecomposition:

$$C_X(t,s) = \sum_{n=1}^{\infty} \lambda_n \varphi_n(t) \varphi_n(s) \tag{3.3}$$

where φ_n and λ_n are orthogonal deterministic eigenfunctions and eigenvalues of the covariance function C_X, respectively, derived from the solution of the eigenvalue problem

$$\int_{\mathscr{D}} C_X(t,s) \cdot \varphi_n(s) \mathrm{d}s = \lambda_n \cdot \varphi_n(t) \tag{3.4}$$

This eigenvalue problem in also known in the literature as the homogeneous Fredholm integral equation of the second kind.

So the first thing in the Karhunen-Loève expansion is to obtain the eigenvalues and eigenfunctions of the linear operator T_{C_X} by solving eq.(3.4). In practice, an analytical solution of eq.(3.4) can only be attained in a few cases, such as the case of the Gaussian correlation function $C_X(t,s) = \sigma^2 e^{-\frac{|t-s|}{\lambda}} = \sigma^2 e^{-\frac{|\tau|}{\lambda}} = C_X(\tau)$ [3]. In the general case, a numerical solution to eq.(3.4) is the only viable option.

The eigenfunctions form a complete orthogonal set satisfying the equation:

$$\int_{\mathscr{D}} \varphi_k(t) \varphi_l(t) \mathrm{d}t = \delta_{kl} \tag{3.5}$$

where δ_{kl} is the Kronecker-delta function. Then, the process $X(t)$ can be represented as follows :

$$X(t) = \sum_{i=1}^{\infty} \sqrt{\lambda_n} \cdot \varphi_n(t) \cdot \xi_n, \quad t \in \mathscr{D} \tag{3.6}$$

where $\xi_n := \xi_n(\theta)$ is a set of uncorrelated random variables with mean $\mathbb{E}\big[\xi_n(\theta)\big] = 0$ and covariance function $\mathbb{E}\big[\xi_k(\theta) \cdot \xi_l(\theta)\big] = \delta_{kl}$ which can be expressed as

$$\xi_n = \frac{1}{\sqrt{\lambda_n}} \int_{\mathscr{D}} X(t) \cdot \varphi_n(t) \mathrm{d}t \tag{3.7}$$

Equation (3.6) is known to converge in the mean square sense for any distribution of $X(t)$. The KL expansion of a Gaussian process has the property that $\xi_n(\theta)$ are independent standard normal variables, or in other words, they follow the $\mathcal{N}(0,1)$ distribution. For practical implementation, the infinite series of eq. (3.6) is truncated after a finite number of terms, M, giving the approximation

$$X(t) \approx \hat{X}(t) = \sum_{n=1}^{M} \sqrt{\lambda_n} \cdot \varphi_n(t) \cdot \xi_n \tag{3.8}$$

[3]See textbook "Stochastic Finite Element Methods", pp. 32-34

The corresponding covariance function is then approximated by

$$\hat{C}_X(t,s) = \sum_{n=1}^{M} \lambda_n \varphi_n(t) \varphi_n(s) \tag{3.9}$$

This truncated series is optimal in the mean square since the eigenvalues λ_n of eq. (3.8) are converging fast to zero (Fig.3.1). Thus, the choice of the covariance eigenfunction basis $\{\varphi_n(t)\}$ is optimal in the sense that the mean square error resulting from a truncation after the M-th term is minimized.

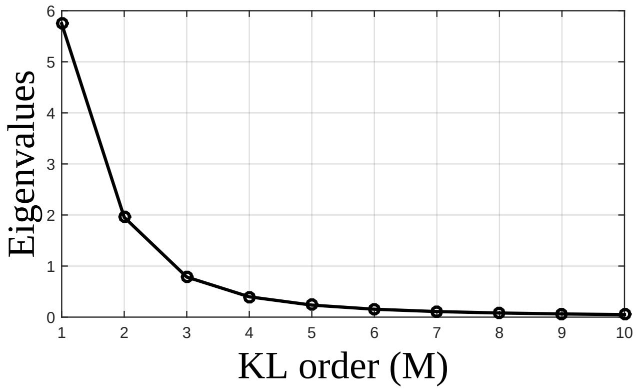


Figure 3.1: Decaying eigenvalues from the solution of the Fredholm integral of the second kind for $M = 10$.

The variance error e_{var} after truncating the expansion in M terms can be easily computed as

$$e_{var} = \mathrm{Var}\big[X(t,\theta) - \hat{X}(t,\theta)\big] = \sigma_X^2 - \sum_{n=1}^{M} \lambda_n \cdot \varphi_n^2(t) \tag{3.10}$$

The righthand side of the above equation means that the KL expansion always under-represents the true variance of the field.

Now, let's discuss how eq. (3.8) can be utilized in order to generate realizations from a zero-mean Gaussian process $X(t)$, given the eigenvalues $\{\lambda_n\}_{n=1}^{M}$ and eigenfunctions $\{\phi_n\}_{n=1}^{M}$.

1. To generate the j-th realization, we draw a random value for each ξ_n, $n = 1, ...M$ from the standard Gaussian distribution $\mathcal{N}(0,1)$ and obtain $\xi_1^j, ..., \xi_M^j$

2. We insert these values to eq. (3.8) in order to obtain the j-th realization:

$$\hat{X}^j(t) = \sum_{n=1}^{M} \sqrt{\lambda_n} \cdot \varphi_n(t) \cdot \xi_n^j \tag{3.11}$$

3. To generate additional realizations, we simply draw new random values for ξ_n, $n = 1, ...M$, each from $\mathcal{N}(0,1)$

Another interesting case that will be examined here, is the application of KL expansion method to the modeling of Lognormal fields. Lognormal fields are frequently used in order to model phenomena, which cannot admit negative values, e.g. material properties. Let us recall that if Y is a random variable such that $Y\ Lognormal(m, s^2)$, then there is an underlying Gaussian random variable $Y^{Gaussian}\ \mathcal{N}(\mu, \sigma^2)$ that satisfies the relation:

$$Y = exp(Y^{Gaussian}) \tag{3.12}$$

The parameters μ and σ of the underlying Gaussian are given from the following relations:

$$\mu = ln\left(\frac{m^2}{\sqrt{s^2 + m^2}}\right) \tag{3.13}$$

$$\sigma = \sqrt{ln\left(\frac{s^2}{m^2} + 1\right)} \tag{3.14}$$

Therefore, if $X(t)$ is a stationary lognormal field with mean m and variance σ^2, we can find an underlying Gaussian field $X^{Gaussian}(t)$ with μ and σ^2. Then, we represent $X^{Gaussian}(t)$ using the KL series expansion and we obtain:

$$X = exp(X^{Gaussian}(t)) \approx exp(\sum_{n=1}^{M} \sqrt{\lambda_n}\phi_n(t)\xi_n) \tag{3.15}$$

Lastly, for random processes where the analytical solution of the Fredholm integral equation cannot be obtained, a numerical solution is necessary. One major category of such solution schemes are the expansion methods such as the Galerkin, the collocation and the Rayleigh-Ritz methods. Galerkin methods are essentially error minimization schemes with respect to some residual calculated over the entire domain of the solution. Assuming that each eigenfunction $\varphi_n(t)$ of $C_X(t, s)$ may be represented by its expansion over a polynomial basis $\{h_i(\cdot)\}$, defined in the solution space, as:

$$\varphi_n(t) = \sum_{i=1}^{\infty} d_i^n \cdot h_i(t) \tag{3.16}$$

where d_i^n are unknown coefficients to be estimated, the Galerkin procedure targets to an optimal approximation of the eigenfunctions $\varphi_n(\cdot)$ after truncating the above series in N terms and computing the residual as:

$$\varepsilon_N(t) = \sum_{i=1}^{N} d_i^n \cdot \left[\int_{\mathcal{D}} C_X(t, s) \cdot h_i(s)\mathrm{d}s - \lambda_j \cdot h_i(t)\right] \tag{3.17}$$

Requiring the residual to be orthogonal to the space spanned by the same basis we get

$$< \varepsilon_N, h_j >:= \int_{\mathcal{D}} \varepsilon_N(t) \cdot h_j(t)\mathrm{d}t = 0, \quad j = 1, \ldots, N \tag{3.18}$$

which leads to the following matrix eigenvalue equation:

$$\mathbf{C} \cdot \mathbf{D} = \mathbf{\Lambda} \cdot \mathbf{B} \cdot \mathbf{D} \tag{3.19}$$

where

$$\mathbf{B}_{ij} = \int_{\mathscr{D}} h_i(t) \cdot h_j(t) dt \tag{3.20}$$

$$\mathbf{C}_{ij} = \int_{\mathscr{D}} \int_{\mathscr{D}} C_X(t,s) \cdot h_i(s) \mathrm{d}s \tag{3.21}$$

$$\mathbf{D}_{ij} = d_i^j \tag{3.22}$$

$$\mathbf{\Lambda}_{ij} = \delta_{ij} \cdot \lambda_j \tag{3.23}$$

where $\mathbf{C}, \mathbf{D}, \mathbf{B}$ and $\mathbf{\Lambda}$ are $N \times N$-dimensional matrices. This generalized algebraic eigenvalue problem of eq. (3.19) can be solved for $\mathbf{D}$ and $\mathbf{\Lambda}$ and with backsubstitution we can estimate the eigenfunctions of the covariance kernel. This solution scheme can be implemented using piecewise polynomials for the basis $\{h_i(\cdot)\}$ of the expansion.

3.3 The Spectral Representation series expansion

The spectral representation method was proposed by Shinozuka and Deodatis in 1991 and it is a method for generating sample functions that are ergodic in the mean value and autocorrelation. Its main property is that it expands the stochastic field onto a series of trigonometric functions with random phase angles. For a zero-mean, one-dimensional stationary stochastic process $X(t)$ with autocorrelation function $R_X(t,s)$ and two-sided power spectral function $S_X(\omega)$ it was proven that

$$X(t) = \sqrt{2} \cdot \sum_{k=0}^{N-1} \left(2S_X(\omega_k) \cdot \Delta\omega\right)^{\frac{1}{2}} \cdot \cos(\omega_k \cdot t + \Phi_k) \tag{3.24}$$

as $N \to \infty$. In the above equation, $\Delta\omega = \frac{\omega_u}{N}$, with ω_u being an upper cutoff frequency beyond which the power spectrum may assumed to be zero for either mathematical or physical purposes. Also, $\omega_k = k\Delta\omega$ for $k = 1, ...$ are the frequencies of the power-spectrum $(2S_X)$, $(2S_X(\omega_k)$ are the values of the spectrum at each frequency ω_k and Φ_k are the random phase angles, which are random variables following the uniform distribution $\mathscr{U}\,[0, 2\pi]$.

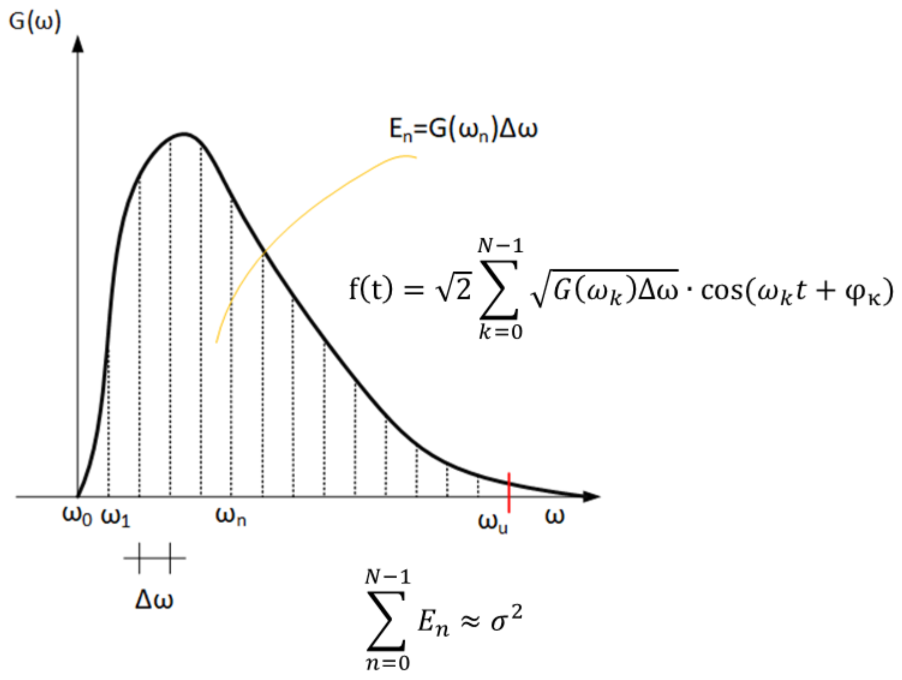
For practical applications, eq. (3.24) can be utilized as a simulation algorithm in order to generate realizations $f(t)$ of the process $X(t)$. First, a number N is considered, which will be the number of random terms in the expansion. Then, to generate a realization of the process, we draw N values for the random phase angles $\{\Phi_k\}_{k=0}^{N-1}$ from the $\mathscr{U}[0, 2\pi]$ distribution, denoted as $\{\phi_k\}_{k=0}^{N-1}$, and evaluate the following relation

$$f(t) = \sqrt{2} \cdot \sum_{k=0}^{N-1} A_k \cdot \cos(\omega_k \cdot t + \phi_k) \tag{3.25}$$

where,

$$
\begin{aligned}
A_k &= \left(2S_X(\omega_k)\cdot\Delta\omega\right)^{\frac{1}{2}} \quad \text{for } k=0,1,\ldots,N-1 \\
\omega_k &= k\cdot\Delta\omega \\
\Delta\omega &= \frac{\omega_u}{N} \\
A_0 &= 0 \quad \text{or } S_X(\omega_0=0)=0 \\
\Phi_k &\sim \mathcal{U}[0,2\pi], \ \text{independent}
\end{aligned}
\tag{3.26}
$$

The following figure will help us to better understand the procedure.



G(ω)
E$_n$=G(ω$_n$)Δω
$f(t) = \sqrt{2}\sum_{k=0}^{N-1}\sqrt{G(\omega_k)\Delta\omega}\cdot\cos(\omega_k t + \varphi_\kappa)$
ω$_0$ ω$_1$
ω$_n$
ω$_u$
ω
Δω
$\sum_{n=0}^{N-1} E_n \approx \sigma^2$

Figure 3.2: Illustration of the method

Some remarks:

- Note that the simulated process is asymptotically Gaussian as N becomes large due to the central limit theorem
- The simulated process is periodic with period:

$$T_0 = \frac{2\pi}{\Delta\omega} \tag{3.27}$$

- The coefficient A_0 is chosen zero such that the temporal mean value averaged over the whole simulation time $T_0 = \frac{2\pi}{\Delta\omega}$ of the generated stochastic process $\hat{X}(t,\theta)$ remains zero in each generated sample. This is because if some power spectral contribution is added at $\omega = 0$, a random variable term is always present, shifting the temporal (sample) average apart from being zero.
- Instead of writing $2S_X(\omega_k)$ in eq. (3.24) we can simply write $G_X(\omega_k)$

- As mentioned, ω_u in eq.(3.26) corresponds to the upper-cut off frequency after which the power spectrum becomes practically zero. In order to estimate this frequency we use the following criterion:

$$\int_0^{\omega_u} S_X(\omega)\mathrm{d}\omega = (1-\epsilon)\int_0^{\infty} S_X(\omega)\mathrm{d}\omega \tag{3.28}$$

where $\epsilon \ll 1$ is the 'admissible relative error'. The target autocorrelation function $R_{\hat{X}}(\tau)$ is given by

$$R_{\hat{X}}(\tau) = \int_{-\omega_u}^{\omega_u} S_X(\omega)e^{i\omega\tau}\mathrm{d}\omega = \int_0^{\omega_u} 2S_X(\omega)\cos\omega\tau\mathrm{d}\omega \tag{3.29}$$

The difference between these two functions

$$\epsilon^*(\tau) = R_X(\tau) - R_{\hat{X}}(\tau) = \int_{\omega_u}^{\infty} 2S_X(\omega)\cos(\omega\tau)\mathrm{d}\omega \tag{3.30}$$

corresponds to the mean square simulation error due to the truncation of the spectral density function for $|\omega| \geqslant \omega_u$, which is termed 'truncation error'.

- It must be mentioned that the step Δt of the generated sample functions must satisfy the following condition in order to avoid aliasing.

$$\Delta t \leq \frac{\pi}{\omega_u} \tag{3.31}$$

- The simulations generated with this simulation algorithm satisfy the following ergodic property:

$$\langle f(t)\rangle = \frac{1}{T}\int_0^T f(t)dt = \mathbb{E}[X(t)] = 0, \quad \text{either when } T = T_0 \text{ or as } T \to \infty \tag{3.32}$$

$$\langle f(t)f(t+\tau)\rangle = \frac{1}{T}\int_0^T f(t)f(t+\tau)dt = R_X(\tau), \quad \text{either when } T = T_0 \text{ or as } T \to \infty \tag{3.33}$$

3.4 The Polynomial Chaos series expansion

The third series expansion that we will examine is the Polynomial Chaos. In order for this expansion to apply we only require for the process to be L^2-integrable. To better illustrate the method, let us first consider a random variable X, which belongs to the Hilbert space L^2 ($\mathbb{E}[X^2] < \infty$). Thus, X can be written as a series expansion with respect to set of orthogonal basis functions $\{\Psi_i\left(\{\xi_j\}_{j=1}^M\right)\}_{i\in\mathbb{N}}$, which constitute the Hilbert basis of the space such that:

$$X = \sum_{i \in \mathbb{N}} \hat{c}_i \Psi_i(\boldsymbol{\xi}) \tag{3.34}$$

where $\Psi_i(\boldsymbol{\xi}) := \Psi_i\left(\{\xi_j\}_{j=1}^M\right)$ are the multivariate orthogonal polynomials, which form the so-called polynomial chaos basis defined by means of M random variables $\{\xi_j\}_{j=1}^M$. The coefficients $\hat{c}_i \in \mathbb{R}$ are the coordinates of the random variable X with respect to the chosen basis. The type of polynomials is chosen according to the probability measure of the input random variable X. For instance, if X is a Gaussian random variable, the univariate Hermite polynomials are chosen because they produce the optimal basis. These are given by the following recurrent relation:

$$\Psi_n(\xi) = (-1)^n e^{\frac{\xi^2}{2}} \frac{\partial^n}{\partial \xi^n} e^{-\frac{\xi^2}{2}} \tag{3.35}$$

where ξ is a standard normal variable. It should be mentioned, that the classic Polynomial Chaos series expansion refers to the basis vectors formed by the Hermite polynomials. For other types of random variables, different families of polynomials were proven to produce optimal bases, leading to the so-called *generalized Polynomial Chaos*. For example, the Legendre-Chaos is preferred for modeling uniform random variables and the Laguerre-chaos for random variables following the gamma distribution. In the frame of this course, we will only focus on the Hermite-chaos (aka Wiener chaos).

To apply this method and express the random variable in this series expansion requires the evaluation of the coefficients $\hat{c}_i$. But before we do this, let us first address the case where $X = X(t)$ is a stochastic process. In this case, the coefficients are $\hat{c}_i = \hat{c}_i(t)$, that is, real functions with respect to time. Then,

$$X(t) = \sum_{i \in \mathbb{N}} \hat{c}_i(t) \Psi_i(\boldsymbol{\xi}) \tag{3.36}$$

Subsequently, we truncate the above series after a finite number P of terms, which gives us the approximation:

$$X(t) \approx \sum_{i=0}^{P-1} \hat{c}_i(t) \Psi_i(\boldsymbol{\xi}) \tag{3.37}$$

It is of great significance that the polynomial basis $\{\Psi_i\}$ of Hermite-Chaos forms a complete orthonormal basis, satisfying the following properties:

$$\begin{aligned} &\Psi_0 = 1 \\ &\mathbb{E}[\Psi_i] = 0, \quad i > 0 \\ &\mathbb{E}[\Psi_i \Psi_j] = \langle \Psi_i \Psi_j \rangle = \langle \Psi_i^2 \rangle \delta_{ij} \end{aligned} \tag{3.38}$$

where, $\langle \cdot, \cdot \rangle$ denotes the inner product:

$$\langle \Psi_i(\boldsymbol{\xi}), \Psi_j(\boldsymbol{\xi}) \rangle = \int \Psi_i(\boldsymbol{\xi}) \Psi_j(\boldsymbol{\xi}) d\mu(\boldsymbol{\xi}) \tag{3.39}$$

with $d\mu$ being the Gaussian measure:

$$d\mu(\boldsymbol{\xi}) = \frac{1}{\sqrt{(2\pi)^M}} e^{-\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi}} d\boldsymbol{\xi} \tag{3.40}$$

We will provide an example of a stochastic ordinary differential equation[4] to demonstrate the use of the Polynomial chaos:

Example We consider the ode

$$\frac{dy(t)}{dt} = -ky, \quad y(0) = \hat{y} \tag{3.41}$$

where the decay rate coefficient k is considered to be a random variable $k(\theta)$ with a certain probability density function $f(k)$ and mean value $\bar{k}$.

Since k is a random variable we can write

$$k = \sum_{i=0}^{P-1} k_i \Psi_i(\xi) \tag{3.42}$$

and for the unknown random process $y(t)$, we can write

$$y(t) = \sum_{i=0}^{P-1} y_i(t) \Psi_i(\xi) \tag{3.43}$$

By substituting these two expansions into the ODE, we obtain

$$\sum_{i=0}^{P-1} \frac{dy_i(t)}{dt} \Psi_i(\xi) = -\sum_{i=0}^{P-1}\sum_{j=0}^{P-1} \Psi_i(\xi)\Psi_j(\xi) k_i y_j(t) \tag{3.44}$$

Now, we will convert the above equation into a system of linear and deterministic ODEs using the Galerkin projection technique. In this regard, we take the inner products of the RHS and LHS of the equation with $\Psi_l(\xi)$, for $l = 0, ..., P-1$ and we get

$$\langle \sum_{i=0}^{P-1} \frac{dy_i(t)}{dt} \Psi_i(\xi), \Psi_l(\xi) \rangle = \langle -\sum_{i=0}^{P-1}\sum_{j=0}^{P-1} \Psi_i(\xi)\Psi_j(\xi) k_i y_j(t), \Psi_l(\xi) \rangle, \quad \text{for } l = 0, ..., P-1 \tag{3.45}$$

Now we will exploit the linearity of the inner product and the orthogonality of the basis functions, which gives us the following set of equations:

$$\frac{dy_l(t)}{dt} = -\frac{1}{\langle \Psi_l^2 \rangle} \sum_{i=0}^{P-1}\sum_{j=0}^{P-1} \epsilon_{ijl} k_i y_j(t), \quad \text{for } l = 0, ..., P-1 \tag{3.46}$$

with $\epsilon_{ijl} = \langle \Psi_i \Psi_j \Psi_l \rangle$. We can solve the above system of deterministic ODEs using any conventional solver in order to obtain the coefficients of the PC expansion $y_i(t)$. So, with this

[4]In some more rigorous mathematical texts, the term stochastic ode is reserved only for ode's driven by white-noise or a semi-martingale in general

approach we essentially managed to convert the stochastic ODE into a set of deterministic ODEs, which is solvable with standard numerical tools. The one-dimensional Hermite polynomials are given in table 3.1

In the example given above we used the univariate Hermite polynomials. The reason why we used univariate is because the system had only one random variable, k. If the system had two random variables k_1 and k_2 we would use the bivariate polynomials given in table 3.2, and so forth, depending on the number of random variables in the system we study. In later chapters, we will talk about *stochastic collocation* and the *Spectral Stochastic Finite Element Method*. In the frame of these methods we will deal with n-dimensional random vectors, where we will use the n-variate Hermite polynomials. These are given by the following recurrent relation:

$$H_n(\xi_{i_1}, ..., \xi_{i_n}) = e^{\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi}}(-1)^n \frac{\partial^n}{\partial_{i_1}, ..., \partial_{i_n}} e^{-\frac{1}{2}\boldsymbol{\xi}^T\boldsymbol{\xi}} \tag{3.47}$$

Then, Y can be represented as:

$$Y = c_0 H_0 + \sum_{i_1=1}^{\infty} c_{i_1} H_1(\xi_{i_1}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1} c_{i_1 i_2} H_2(\xi_{i_1}, \xi_{i_2}) + \sum_{i_1=1}^{\infty}\sum_{i_2=1}^{i_1}\sum_{i_3=1}^{i_2} c_{i_1 i_2 i_3} H_3(\xi_{i_1}, \xi_{i_2}, \xi_{i_3}) + ... \tag{3.48}$$

The above equation can be written more conveniently as:

$$Y = \sum_{i=0}^{\infty} \hat{c}_i \Psi_i(\boldsymbol{\xi}) \tag{3.49}$$

with $\Psi_i(\mathbf{x})$ being on an one-to-one correspondence with $H_n(x_{i_1}, ..., x_{i_n})$ [5]. As mentioned, the polynomial basis $\{\Psi\}$ of Hermite-Chaos forms a complete orthonormal basis, satisfying the properties of eq: (3.38).

The one-dimensional and two-dimensional Hermite polynomial chaoses are given in tables 3.1 and 3.2, respectively.

Table 3.1: One-Dimensional Polynomial Chaoses

j	order of polynomial chaos	i^{th} Polynomial Chaos Ψ_i	$\langle\Psi_i^2\rangle$
0	p=0	1	1
1	p=1	ξ	1
2	p=2	$\xi^2 - 1$	2
3	p=3	$\xi^3 - 3\xi$	6

[5] it's just a rearrangement of the terms in eq. (3.48) which simplifies the notation

Table 3.2: Two-Dimensional Polynomial Chaoses

j	order of polynomial chaos	i^{th} Polynomial Chaos Ψ_i	$\langle\Psi_i^2\rangle$
0	p=0	1	1
1	p=1	ξ_1	1
2		ξ_2	1
3	p=2	$\xi_1^2 - 1$	2
4		$\xi_1\xi_2$	1
5		$\xi_2^2 - 1$	2
6	p=3	$\xi_1^3 - 3\xi_1$	6
7		$\xi_1^2\xi_2 - \xi_2$	2
8		$\xi_1\xi_2^2 - \xi_1$	2
9		$\xi_2^3 - 3\xi_2$	6

3.5 Additional literature

- B. Sudret and A. Der Kiureghian, *Stochastic Finite Element Methods and Reliability, A State-of-the-Art Report* (available online)
- R. Ghanem and P. Spanos, *Stochastic Finite Elements: A Spectral Approach*, 1991
- M. Shinozuka and G. Deodatis, *Simulation of Stochastic Processes by Spectral Representation*, Appl. Mech. Rev, 1991

Chapter 4

The Monte Carlo simulation

4.1 The Stochastic Finite Element Method

Most problems in physics and engineering are described by (systems of) ordinary differential equations or partial differential equations. For instance, the heat flow along a uniform rod is given by the equation:

$$\frac{\partial T}{\partial t} = \frac{k}{c\rho}\frac{\partial^2 T}{\partial x^2} \tag{4.1}$$

where $T := T(x)$ is the temperature field, k is the thermal conductivity of the material, c is the specific heat capacity and ρ is the material density.

Another example is from structural dynamics, where the equations of motion of 2-D linear elasticity can be written as:

$$\frac{E}{2(1+\nu)}\nabla^2 u + \frac{E}{2(1-\nu)}\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = \rho\frac{\partial u^2}{\partial t^2} \tag{4.2}$$

$$\frac{E}{2(1+\nu)}\nabla^2 v + \frac{E}{2(1-\nu)}\frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) = \rho\frac{\partial v^2}{\partial t^2} \tag{4.3}$$

where $\boldsymbol{u} = (u, v)$ is the displacement vector field, E is the modulus of elasticity, ν is the Poisson ratio and ρ is the material mass density.

Both equations involve some parameters such as k, c, ρ in eq. (4.1) and E, ν in eq. (4.2). In this section we will focus on problems where the parameters are considered to be stochastic. For instance, the modulus of elasticity E can be modeled as a random variable or, more generally, as a random field $E := E(\boldsymbol{x})$. This implies that the solution $\boldsymbol{u}(\boldsymbol{x}, t)$ will also be a random field. In this regard, we can define now the Stochastic Finite Element Method as **the set of numerical and mathematical methods developed to solve systems of PDEs that involve random parameters**.

For the purposes of illustration, in the continuation of this chapter we will consider the simplest case, that of linear static (or steady-state) problems governed by parametrized pde's of the form:

$$\begin{aligned} \mathscr{N}\left(\boldsymbol{u}\left(\boldsymbol{x};\boldsymbol{\xi}\right)\right) &= f(\boldsymbol{x};\boldsymbol{\xi}), \quad \boldsymbol{x}\in\Omega, \boldsymbol{\xi}\in\Xi \\ \mathscr{B}(\boldsymbol{u}(\boldsymbol{x};\boldsymbol{\theta}) &= b(\boldsymbol{x};\boldsymbol{\xi}), \quad \boldsymbol{x}\in\partial\Omega, \boldsymbol{\xi}\in\Xi \end{aligned} \tag{4.4}$$

where $\boldsymbol{u}(\boldsymbol{x};\boldsymbol{\xi})$ is the field of interest (scalar or vector field), $\mathscr{N}$ is a linear differential operator that involves spatial derivatives, and $f(\boldsymbol{x};\boldsymbol{\xi})$ is a source field. Furthermore, $\mathscr{B}$ is the operator for the boundary conditions defined on the boundary $\partial\Omega$ of the domain Ω and $\boldsymbol{\xi} \in \Xi$ is a vector of uncertain parameters that include randomness in the system parameters, loading or boundary conditions. Here we use the notation $(\cdot;\boldsymbol{\xi})$ to state explicitly dependence on the stochastic quantities. Since analytic tools to solve such equations apply only for specific cases, the most generic way to solve them is using numerical methods, the finite element method in particular, as it is the most versatile approach. A reminder of the general idea of FEM is given in the Appendix. Eventually, FEM will produce a linear system of equations of the form:

$$\boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{U}(\boldsymbol{\xi}) = \boldsymbol{F}(\boldsymbol{\xi}) \tag{4.5}$$

with $\boldsymbol{U} \in \mathbb{R}^n$ denoting the unknown vector of nodal values of the response. This will be an $n \times n$ linear system of equations with n being the number of degrees of freedom from the finite element discretization.

4.2 Classical Monte Carlo Integration

Before showing how simulation techniques can be employed to solve stochastic systems of the form of eq. (4.5), we first need to develop their properties in some detail. This is more easily accomplished by looking at the generic problem of evaluating the integral

$$\mathbb{E}_f[h(X)] = \int_{\mathscr{D}} h(x)f(x)dx \tag{4.6}$$

A natural way to approximate the above integral, is to generate values $\{x_i\}_{i=1}^m$ from the density f and compute the empirical average

$$\bar{h}_m = \frac{1}{m}\sum_{i=1}^{m} h(x_i) \tag{4.7}$$

By the Strong Law of large numbers, $\bar{h}_m$ converges almost surely to $\mathbb{E}_f[h(X)]$ as $m \to \infty$. Moreover, when h^2 has a finite expectation under f, the speed of convergence of $\bar{h}_m$ can be estimated from the sample's variance

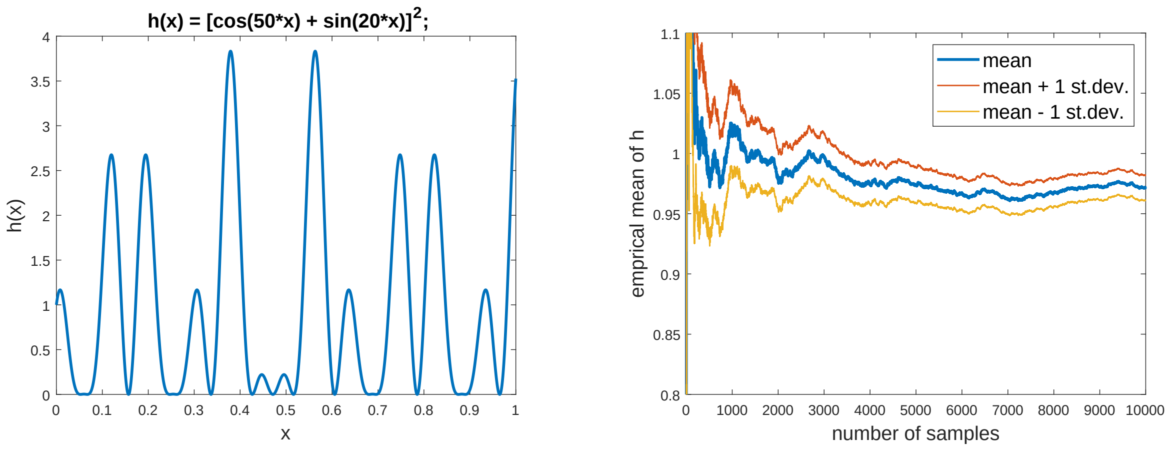
$$v_m = \frac{1}{m^2}\sum_{i=1}^{m}[h(x_i) - \bar{h}_m]^2 \tag{4.8}$$

because, for m large, the quantity

$$\frac{\bar{h}_m - \mathbb{E}_f[h(X)]}{\sqrt{v_m}} \tag{4.9}$$

is approximately distributed as a $\mathscr{N}(0,1)$ variable. This allows us to construct a convergence test and confidence bounds on the approximation of $\mathbb{E}_f[h(X)]$.

Example Let us consider the function $h(x) = [cos(50x) + sin(20x)]^2$ and integrate it over the $[0,1]$ domain. To calculate the integral we generate samples $\{x_i\}_{i=1}^N$ from the uniform distribution $\mathcal{U}[0,1]$ and approximate $\int_0^1 h(x)dx$ with $\frac{1}{N}\sum_{i=1}^{N} h(x_i)$. The exact value of the integral is 0.965.



(a) function $h(x)$ (b) Mean $\pm$ one standard error

Figure 4.1: Example of Monte Carlo integration

As an exercise, you can try to evaluate the integral $h(x)f(x)$, where $f(x)$ is the pdf of the standard normal distribution.

Lastly, we would like to mention that there are more elaborate techniques to assess the convergence of estimators (based on the Central Limit Theorem, Cauchy priors, etc.), however, in practical applications, a convergence plot such as the one in 4.1b is satisfactory.

4.3 The Monte Carlo simulation

Now let us return to our initial problem of solving eq. (4.5). Suppose that we have already identified the sources of randomness in our system, that is, the random variables, ξ_i and we know their distributions, as well as their joint distribution. An important note here concerns the case where the system under consideration involves random fields, which as we said are a family of infinite random variables. In such problems the series expansion methods help us reduce the random variables in the system to a finite number. Since we are dealing with stochastic systems, it is evident the quantities of interest $\boldsymbol{U}$ will be random variables[1]. Therefore, our aim is to find the probability density functions of these quantities.

The (brute force or crude) **Monte Carlo simulation** for solving eq. (4.5) consists in generating a large number N_{sim} of samples $\boldsymbol{\xi}_{(i)}$, $i = 1, .., N_{sim}$ from their probability distribution and for each of these samples to solve the deterministic problem

$$\boldsymbol{K}(\boldsymbol{\xi}_{(i)})\boldsymbol{U}(\boldsymbol{\xi}_{(i)}) = \boldsymbol{F}(\boldsymbol{\xi}_{(i)}) \tag{4.10}$$

The next step is to collect the N_{sim} response vectors $\boldsymbol{U}_i := \boldsymbol{U}(\boldsymbol{\xi}_{(i)})$ and perform a statistical post-processing in order to extract useful information such as mean value, variance, histogram, empirical pdf/cdf, etc. For instance,

[1]or random fields if the problem was dynamic

$$\mathbb{E}[\boldsymbol{U}] \approx \frac{1}{N_{sim}} \sum_{j=1}^{N_{sim}} \boldsymbol{U}_i \tag{4.11}$$

In a similar manner, higher order moments of response quantities can be estimated and finally, the entire pdf can be constructed from the statistical post-process of the N_{sim} samples.

Pros

- The Monte Carlo simulation is applicable to every problem type (linear/nolinear, static/dynamic, etc.).
- The computational cost of the method remains invariant with respect to the number of random variables involved.
- It's very easy to implement.

Cons

- The cost of performing a large number of simulations may be prohibitive for complex systems (imagine if each simulation takes from several minutes to several hours).

Closing this section, we briefly discuss ways to overcome the only limitation that MCS has, namely, its computational cost. The first approach is to exploit parallel programming architectures in order to run multiple simulations simultaneously. It is evident that MCS is "embarassingly" parallel, which renders this approach straightforward to implement. However, it has specific hardware requirements or access to supercomputers that are not always available to the researcher. An alternative approach, would be to substitute our complex model with a model that mimics the behavior of the original and is easier to solve. We call such models **surrogates** or **meta-models**. In most cases, the construction of a meta-model relies on the some information we have about the system in the form of data, hence, they are usually called **data-driven** models.

4.4 Additional literature

Chapter 5

Data-driven modeling

5.1 Introduction

The great advances in computational technology over the last two decades enabled the use of high-fidelity physics-based mathematical models to describe complex physical systems, arising in all areas of engineering and science. These detailed models are better able to describe the underlying physics of the problem and thus provide accurate representations of the system's behavior. Nevertheless, for many applications such as nonlinear and/or dynamic systems with high-dimensionality, the computational cost can still be considerably high. This cost becomes prohibitive in the case of parametrized systems, which arise in the fields of stochastic analysis, sensitivity analysis, parameter inference or optimization, where a large number of repeated simulations is required. To circumvent this problem, metamodels (or surrogate models) may be employed in order to replace the original model by a function that emulates the complex system's behavior at significantly smaller cost per model evaluation.

To describe the problem in a more concrete setting, let as consider a computational model $\mathbb{M}$, which takes M-dimensional vectors $\mathbf{x}$ as inputs and maps them to $\mathbb{R}^N$:

$$\mathbb{M} : \mathbf{x} \in \mathfrak{D} \subset \mathbb{R}^M \mapsto \mathbf{y} \in \mathbb{R}^n \tag{5.1}$$

A surrogate model $\tilde{\mathbb{M}}$ can then be defined as:

$$\tilde{\mathbb{M}} : \mathbf{x} \in \mathfrak{D} \subset \mathbb{R}^M \mapsto \mathbf{y} \in \mathbb{R}^n \tag{5.2}$$

such that

$$\tilde{\mathbb{M}} \approx \mathbb{M} \tag{5.3}$$

The construction of the surrogate model can be based on an assumption about the functional shape of $\mathbb{M}$ and/or some information about the system's behavior obtained either from experimental measurements or from limited runs of the original model. The latter are usually referred to as synthetic data. Further, $\mathbb{M}$ is considered as a black box, in the sense that the inner mechanisms of the model are unknown and only the output $\mathbf{y} = \mathbb{M}(\mathbf{x})$ is accessible. As an example, a finite element model constitutes a surrogate model, since it approximates the solution of the system's governing equations that cannot be solved analytically.

To make things more clear, we will consider a simple example of a surrogate model. Let's suppose that we have a model $\mathbb{M}$ that takes as input an $x \in \mathbb{R}$ and outputs a $y \in \mathbb{R}$. Let us also suppose that we know nothing about the model's mechanisms with regards to how it processes x and gives us y, but instead we have a set of measurements given in the form of pairs (x_i, y_i), $i = 1, ..., N$. Then, we can construct a metamodel by assuming a functional shape for the model, i.e. we can assume a linear model of the form $y = ax + b$, with a and b being the metamodel's parameters. Then, the goal is to calibrate the a, b parameters in order to have a good agreement between the model and the surrogate. In this particular example, the parameters that give the best fit (in the least-squares approach) are:

$$\hat{a} = \bar{y} - \hat{b}\bar{x} \tag{5.4}$$

$$\hat{b} = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{5.5}$$

with $\bar{x}, \bar{y}$ being the average of x_i and y_i, respectively. Obviously, we could assume more elaborate functions such as higher-order polynomials or neural networks, which we will see later on, that involve more parameters and allow us to capture more complex model behaviors. The general premise, however, is the same: we use data (experimental or synthetic) to calibrate the parameters in our metamodel.

Since, the construction of a (meta-)model depends on data, we refer to this approach as **data-driven modeling**, and the derived models are called **data-driven models**. Nowadays, the term data-driven is associated to Machine Learning (although not exclusively), implying the utilization of computer automated methods to build our models.

We will say a few words about Machine Learning (ML), just to get the general idea. ML is an application of artificial intelligence that involves the study of algorithms and statistical models that computer systems use to perform a specific task without using explicit instructions, relying on pattern recognition and inference instead. The process of learning begins with observations or data, such as examples, direct experience, or instructions, in order to look for patterns in data and make better decisions in the future based on the examples that the user provided. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly. ML algorithms can be broadly classified into supervised or unsupervised. The algorithms in the first category are able a build a mathematical model from a set of labeled data, that is, a set which contains both the inputs and the desired outputs. Then, the system can provide targets (predictions) for any new input after sufficient training. In contrast, unsupervised ML algorithms are used in order to construct a mathematical model from a set of data which contains only inputs and no desired output labels. Unsupervised learning is used for discovering patterns in the data and grouping the inputs into categories (feature learning). Other classes of ML algorithms include semi-supervised learning algorithms, active learning algorithms and reinforcement learning algorithms.

It becomes evident from the above that ML algorithms are perfect candidates for constructing surrogate models of complex physical systems. The data set that will be used to train the algorithm may come either from a limited number of runs of the original complex model (simulation-based approach) or some experiments. In this chapter, we will discuss several Machine Learning algorithms and how to apply them in the context of computational mechanics in order to construct accurate and cheap-to-evaluate metamodels.

5.2 Artificial Neural Networks

5.2.1 Feed-forward fully connected deep NNs

In this section, we will discuss about the most common type of neural networks, that is, feed-forward fully-connected neural networks are presented in a concise manner. NNs are information-processing mathematical models inspired by the biological neural networks that constitute the human brain. As its original counterpart, they are able to learn from observational data, that is, by considering examples without being programmed with any task-specific rules. The basic component of an NN is the artificial neuron. An artificial neuron, denoted with j is a processing unit which performs the following operations:

1. It receives an input signal x_i from the synapse i
2. It multiplies the signal by the synaptic weight w_{ji}
3. It sums all input signals x_i with their respective weights w_{ji}, for all the synapses $i = 1, ..., n$ and adds a bias term b_j.
4. It processes the sum of the input signals through an activation function $\varphi(\cdot)$, for example the sigmoid or the hyperbolic tangent function, and outputs the result y_j.

In mathematical terms, the neuron j can be described by the equation:

$$y_k = \varphi(\sum_{i=1}^{n} w_{ji}x_i + b_j) \tag{5.6}$$

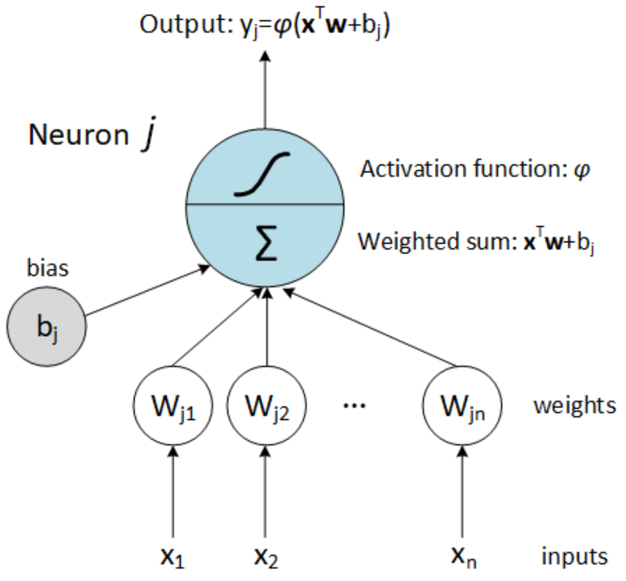

Output: $y_j=\varphi(\mathbf{x}^T\mathbf{w}+b_j)$
Neuron j
Activation function: φ
Weighted sum: $\mathbf{x}^T\mathbf{w}+b_j$
bias
b_j
W_{j1}
W_{j2}
...
W_{jn}
weights
x_1
x_2
x_n
inputs

Figure 5.1: Nonlinear model of a neuron, labeled j

A schematic representation of the computational model of a nonlinear neuron is depicted in fig. 5.1. Some typical examples of activation functions are:

- The *ReLU* function

$$\varphi(S) = max(0, S) \tag{5.7}$$

- The *sigmoid* function

$$\varphi(S) = \frac{e^S}{1 + e^S} \tag{5.8}$$

- The *tanh* function

$$\varphi(S) = \frac{e^S - e^{-S}}{e^S + e^{-S}} \tag{5.9}$$

In this context, an NN is an oriented graph with neurons being the nodes of the graph and the synapses being the oriented edges. The synaptic weights are calibrated through a training process based on observational data. Depending on the interconnection of neurons, different types of neural networks arise. Among them, the most popular and widely applied type is the feed-forward neural network (FFNN), also known as a multilayer perceptron. In terms of the architecture, an FFNN consists of the input layer, the hidden layer(s) and the output layer. NNs with more than one hidden layer are referred to as deep neural networks. In terms of connectivity, in FFNN neurons from a layer can only be connected with neurons from the next layer towards the output layer. This means that the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if they exist) and to the output nodes. An example of an FFNN network architecture is given in figure 5.2.

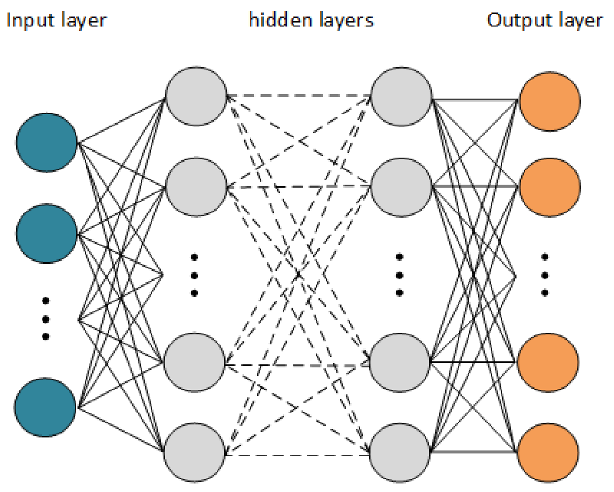


Figure 5.2: Example of a generic feedforward neural network

For a specific choice of network architecture, to train the network a set of N_{tr} labeled data $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}\}_{1 \leq k \leq N_{tr}}$ is first provided. Next, a loss function is specified, such as the mean square error, which is given by the expression:

$$MSE = \sum_{k=1}^{N_{tr}} \|\bar{\boldsymbol{y}}_k - \hat{\boldsymbol{y}}_k\|_2^2 \in \mathbb{R} \tag{5.10}$$

where, $\bar{\boldsymbol{y}}_k$ is the target output for the input $\bar{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{y}}_k$ is the respective network's output. Then, the training of the network consists in finding the optimal weights $\boldsymbol{w} = (w_{ij})$ and biases $\boldsymbol{b}$ that minimize $MSE = MSE(\boldsymbol{w}, \boldsymbol{b})$. Some commonly preferred algorithms to solve this (non-convex) optimization problem are the stochastic gradient descent algorithm, the Levenberg-Marquardt algorithm, Adam, stochastic gradient descent. We won't go into details with respect to these solvers, as they are already available in most neural network packages (Tensorflow, PyTorch, Matlab Deep Learning Toolbox, etc.).

Based on the above, FFNNs essentially establish a non-linear map from the space of the input data to the space of the output data. Their powerful approximation properties are well-established from numerous applications, as well as from a theoretical standpoint, due to the universal function approximation theorem which states that:*A FFNN with one hidden layer, that contains a finite number of neurons and has non-constant, bounded and continuous activation functions, can approximate any continuous function defined on a compact subset of* $\mathbb{R}^m$.

Even though the aforementioned theorem proves that FFNN are universal function approximators, yet, they do not provide any guidelines for selecting the exact network architecture, nor the number of samples required to train the network. In addition, to identify the network parameters only heuristics can be employed to solve the non-convex optimization problem. As a consequence, the optimal network architecture and parameters are achieved in practice via a trial-and-error process which can be quite cumbersome (if not intractable) for large-scale problems.

Example 1.

Consider the following neural network

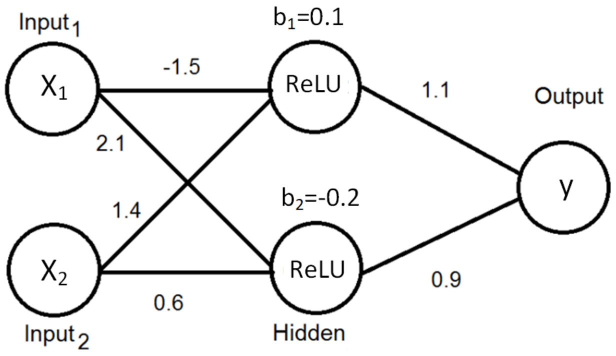


Figure 5.3: An example of a fully-connected feed-forward neural network

In this case

$$y = 1.1max(0, -1.5x_1 + 1.4x_2 + 0.1) + 0.9max(0, 2.1x_1 + 0.6x_2 - 0.2) \tag{5.11}$$

Example 2: Construction of a surrogate

As a more practical example to demonstrate the usage of NNs as surrogates in the service of the Monte Carlo simulation, let us consider the following: We want to estimate the ultimate compressive strength, P_u, of concrete specimens.

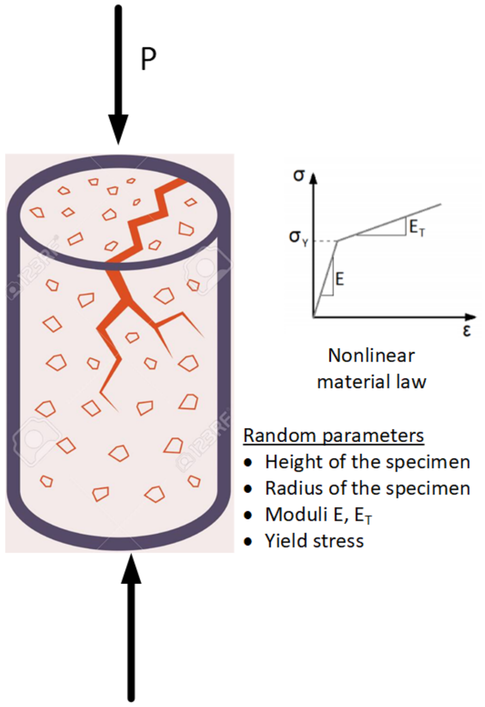


Figure 5.4: An example of a fully-connected feed-forward neural network

In this problem, we consider the following as random variables

1. The height of the specimen, denoted with the r.v. ξ_1
2. The radius of the specimen, denoted with the r.v. ξ_2
3. The moduli of elasticity, denoted with the r.v. ξ_3 and ξ_4
4. The yield stress, denoted with the r.v. ξ_5

In this problem we have 5 random variables, written as a random vector $\boldsymbol{\xi} = (\xi_1, ..., \xi_5)$. It is obvious that the ultimate strength P_u of the specimen will be a random variable, so we need to estimate its probabilistic characteristics. In the context of brute force Monte Carlo, we would generate N_{sim} realizations of the random vector $\boldsymbol{\xi}$, that is $\{\boldsymbol{\xi}_i\}_{i=1}^{N_{sim}}$, solve the problem N_{sim} times and statistically process the results $\{(P_u)_i\}_{i=1}^{N_{sim}}$. However, this is a computationally intensive process.

Instead, we could perform only N_{tr} simulations of the model, with $N_{tr} \ll N_{sim}$, and then build a neural network using as input the 5-tuples $\{(\xi_1)_i, ..., (\xi_5)_i\}_{i=1}^{N_{tr}}$ and output the values $\{(P_u)_i\}_{i=1}^{N_{tr}}$. After successfully training the network, we can use it to obtain the values of P_u for any new realization of $\boldsymbol{\xi}$ at minimum cost.

As a closing remark, we would like to state that FFNNs face difficulties when the dimensionality of the input space and/or output space becomes too large. This "curse of dimensionality" problem however can be tackled with other network architectures, as we will see later on.

5.2.2 Convolutional Neural Networks

Convolutional neural networks (CNN) is another class of deep neural networks commonly applied to tasks such as pattern recognition in images and videos, image segmentation and classification, time-series analysis and more. The main advantage they have over fully connected feedforward NNs is that they can handle data with high dimensionality. In CNNs the input is a tensor (which is viewed as nD-array), such as a set of images, which can be given by the 4D-matrix: (number of images) $\times$ (image height) $\times$ (image width) $\times$ (input channels). Imagine the dimensionality of the input space if you wanted to analyse a set of images of resolution 1024×768. CNNs can handle such problems using some layers of specific type such as convolutional layers and pooling layers. So, in general, a CNN can be built by stacking together a set of dense [1], convolutional, pooling and normalization layers.

Convolutional layers

Convolutional layers take as input an n-D array $\boldsymbol{M}$ and apply a filter $\mathscr{F}$ (a.k.a. kernel) of specified size to the elements of $\boldsymbol{M}$ in a moving window fashion. This process is schematically depicted in figure 5.5. Essentially, the objective of the convolution operation is to extract the most important features from the input and use them to encode it. To better clarify this process, let us consider a $2-D$ array $\boldsymbol{M} = [m_{ij}]$ and its encoded version $\boldsymbol{M}^{enc} = [\mu_{ij}]$, called feature map, which is obtained after applying a filter $\boldsymbol{W} = [w_{ij}]$ of size $f_h \times f_w$, moving with vertical stride s_v and horizontal stride s_h. The element μ_{ij} of $\boldsymbol{M}^{enc}$ is given by the equation:

$$\mu_{ij} = \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} m_{i'j'} \cdot w_{uv} + b \quad \text{with} \begin{cases} i' = i \times s_v + u \\ j' = j \times s_h + v \end{cases} \tag{5.12}$$

where b is the bias term and w_{uv} is the element of the filter $\boldsymbol{W}$ that gives the connection weight between elements of $\boldsymbol{M}^{enc}$ and the elements of $\boldsymbol{M}$ within the respective window.

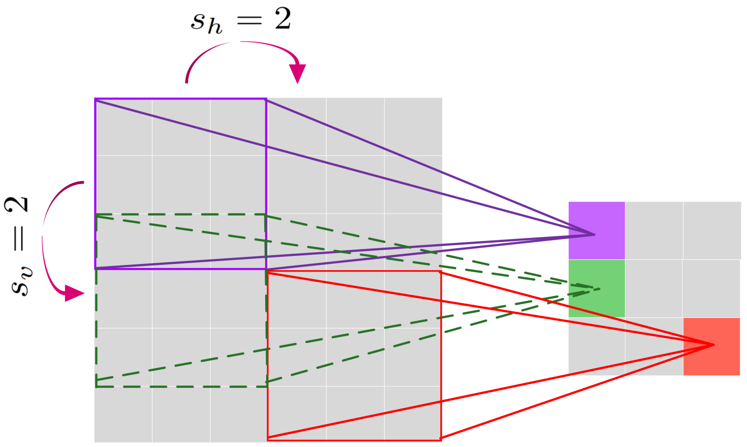


Figure 5.5: Schematic representation of a 2-D convolutional filter with strides $s_h = 2$ and $s_v = 2$.

This layer architecture is significantly more economical than that of a fully connected layer since the parameters involved are only the $f_h \times f_w$ elements of the filter w_{ij} and the bias term

[1]different term for fully connected

b. The filter parameters do not require to be manually defined, instead the convolutional layer will automatically learn the most appropriate filter for the task. Also, a convolutional layer can have multiple filters, in which case it outputs one feature map $\boldsymbol{M}_k^{enc}$ per each filter k. This enables it to detect multiple features anywhere in its inputs. Additionally, several convolutional layers can be stacked in order to build deep architectures which allow the network to concentrate on small low-level features in this first layer and progressively assemble them into larger higher-level features in the subsequent layers. In this more general case, the element μ_{ijk} at the q-th convolutional layer, corresponding to row i, column j of the k feature map $\boldsymbol{M}_k^{enc}$ is obtained as:

$$\mu_{ijk} = \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} m_{i'j'k'} \cdot w_{uvk'k} + b_k \quad \text{with} \begin{cases} i' = i \times s_v + u \\ j' = j \times s_h + v \end{cases} \tag{5.13}$$

where now $f_{n'}$ is the number of feature maps in the previous layer (layer $q-1$), $m_{i'j'k'}$ the value located in row i', column j' of the $q-1$ layer's feature map k' and b_k is the bias term for the k-th feature map (in layer q). Also, $w_{uvk'k}$ is the connection weight between the values in feature map k of layer q and its input located at row u, column v at the window of the k' feature map. To simplify the notation, the application of several convolutional layers, with multiple filters each, to an array $\boldsymbol{M}$ will be expressed as

$$\boldsymbol{M}^{enc} = ConvNN(\boldsymbol{M}) \tag{5.14}$$

with $ConvNN(\cdot)$ denoting the mapping from the initial input space to its encoded representation.

Depending on the application, the convolutional filters can either be one, two or three dimensional with the difference between them being the way they slide across the data. For instance, if the focus is on processing time series data, then 1-D convolutional filters, such as the one depicted in figure 5.6, can be used to scan the data only in the time axis.

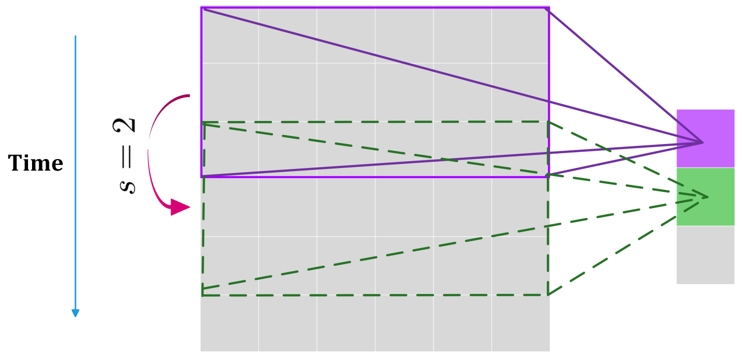


Figure 5.6: Schematic representation of an 1-D convolutional filter with stride $s = 2$.

Pooling layers

Aside of convolutional and dense layers, another important layer type often employed in CNNs are the pooling layes. Pooling layers are quite similar to convolutional layers in the sense that they downsample the input in order to decrease its size, however, they do not involve any trainable parameters. Their goal is to reduce the computational load, the memory usage, and the number of parameters. The latter is particularly useful since it also limits the risk of overfitting. Each neuron in a pooling layer is linked to a limited number of neurons in the previous layer, located within a small window. The window's size and stride are user defined.

Common types of pooling layers include the max pooling layer and the average pooling layer. The first outputs the maximum value from the portion of the input covered by the filter and all other inputs are neglected. Accordingly, average pooling layers return the average from the portion of the input. Aside from its dimensionality reduction properties, the pooling operation can be useful for extracting dominant features of the input such as translational, rotational and scale invariance. Nevertheless, caution should be exercised regarding the usage of pooling layers because the corresponding accuracy loss might outweigh the benefits they provide.

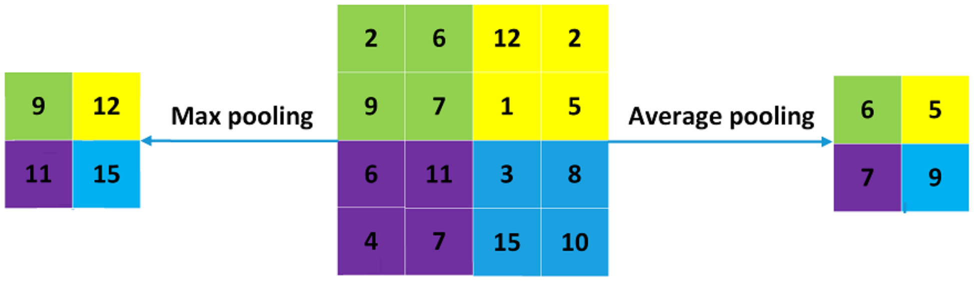


Figure 5.7: Examples of pooling.

Now, let us give two examples of CNNs, in order to better illustrate their application

Example 1: CNN for vehicle classification

Given a set of images depicting various vehicles (car, truck, van, bus, ..., bicycle), we can train a CNN that classifies them into their respective categories. In fig. 5.8 the generic architecture of such a CNN is illustrated. For example, if we give a training input image depicting a car, the respective output will be the vector $[1, 0, 0, ..., 0]$. We experiment with different combinations of convolutional, pooling and dense layers until we have obtained an adequate level of accuracy. Once we have trained the CNN, we can feed any other image of a vehicle to the CNN and it will give as an output vector of coefficients in the $[0, 1]$ range that will be used to classify the vehicle. For instance, if the output vector is $[0.18, 0.67, 0.12, ..., 0.01]$ then we can infer that the image depicted a truck.

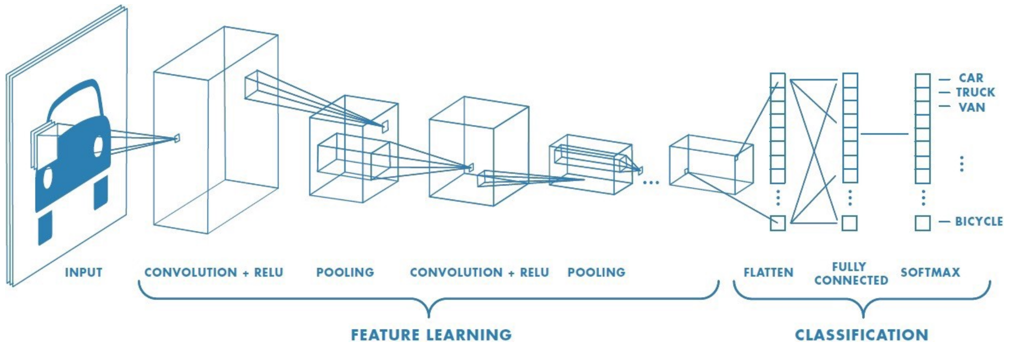


Figure 5.8: Example of a CNN that classifies car types

5.2.3 Autoencoders

The autoencoder (AE) concept was introduced in 1986 and it is regarded as a neural network that learns from an unlabeled data set in an unsupervised manner. The aim of an autoencoder is to learn a reduced representation for a set of data, referred to as encoding, and then to learn how to reconstruct the original input from the encoded input with the minimum possible error. The latter part of the AE is called decoder.

In particular, let $\boldsymbol{X}$ be a subset of $\mathbb{R}^d$ with $\boldsymbol{x} \in \boldsymbol{X}$ denoting an element of the set. Then, the AE's encoder and decoder are defined as transition maps ϕ, ψ such that:

$$\phi : \boldsymbol{X} \subseteq \mathbb{R}^d \to \boldsymbol{H} \subseteq \mathbb{R}^l \tag{5.15}$$

$$\psi : \boldsymbol{H} \subseteq \mathbb{R}^l \to \boldsymbol{X} \subseteq \mathbb{R}^d \tag{5.16}$$

$$\phi, \psi = \underset{\phi,\psi}{\arg\min} \, \|\boldsymbol{X} - (\psi \circ \phi)\boldsymbol{X}\|^2 \tag{5.17}$$

with the dimension l typically being much smaller than d.

Now, let us consider the simplest case, where the encoder has only one hidden layer. It takes an input $\boldsymbol{x} \in \mathbb{R}^d$ and sends it to $\boldsymbol{h} = \phi(\boldsymbol{x}) \in \mathbb{R}^l$, which in this case can also be written as

$$\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{5.18}$$

with σ being an activation function (eg. $tanh$, $ReLU$, etc), $\boldsymbol{W}$ a weight matrix and $\boldsymbol{b}$ a bias vector. The image $\boldsymbol{h}$ of $\boldsymbol{x}$ is the latent or encoded representation of $\boldsymbol{x}$ and $\boldsymbol{H}$ is the latent or feature space.

The decoder's task is to establish the inverse mapping ψ that will reconstruct the input $\boldsymbol{x}$, given its latent representation $\boldsymbol{h}$. Again, considering an one-hidden layer, the reconstructed point $\tilde{\boldsymbol{x}} = \psi(\boldsymbol{h})$ is given by

$$\tilde{\boldsymbol{x}} = \tilde{\sigma}(\tilde{\boldsymbol{W}}\boldsymbol{h} + \tilde{\boldsymbol{b}}) \tag{5.19}$$

where $\tilde{\sigma}$, $\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{b}}$ may be unrelated to those of encoder. Also, the network's architecture selected for the encoder can be different than the decoder's and the number of hidden layer's can be greater than one, leading to the so-called deep autoencoders. The general concept and architecture of an autoencoder is schematically presented in figure 5.9.

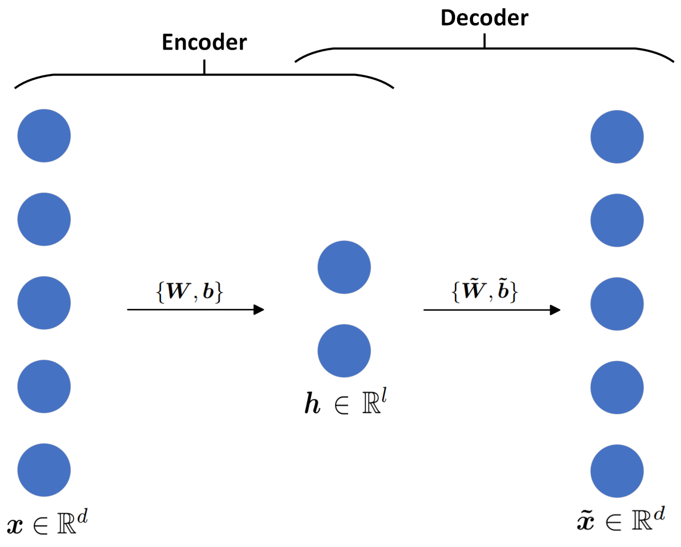

Encoder
Decoder
$\{\boldsymbol{W}, \boldsymbol{b}\}$
$\{\tilde{\boldsymbol{W}}, \tilde{\boldsymbol{b}}\}$
$\boldsymbol{h} \in \mathbb{R}^l$
$\boldsymbol{x} \in \mathbb{R}^d$
$\tilde{\boldsymbol{x}} \in \mathbb{R}^d$

Figure 5.9: Schematic representation of a basic autoencoder

In the context of AEs, the loss function becomes the reconstruction error between the input points $\boldsymbol{x}_i$ and their respective output $\tilde{\boldsymbol{x}}_i$. It is usually expressed as the mean-square error:

$$\mathscr{L} = \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{x}_i - \tilde{\boldsymbol{x}}_i||_2^2 \tag{5.20}$$

with $\|\cdot\|_2$ denoting the L^2-norm and N being the number of points in the training data set. It should be explicitly mentioned that even though the minimization of the reconstruction error implies that the encoder and decoder are trained jointly, however, they can be used separately.

5.2.4 Convolutional Autoencoders

Despite their powerful dimensionality reduction properties, the classic AEs face significant challenges when dealing with very high-dimensional inputs, due to the fact that the number of trainable parameters increases drastically with an increase in the input's dimensionality. In addition, AEs are not capable of capturing the spatial features of the input (e.g. when dealing with images) nor the sequential information in the input (e.g. when dealing with sequence data).

To remedy these issues, a new type of autoencoders was recently emerged, that of convolutional autoencoders (CAEs) Similarly to AEs, CAEs also consist of an encoder and a decoder that are trained to minimize the loss function of eq. (5.20), but they are built from different layer types. Specifically, in CAEs the encoder part is built using a combination of convolutional layers, fully connected layers, pooling layers and normalization layers, while the decoder is built from deconvolutional layers and unpooling layers along with fully connected and normalization layers. Intuitively, CAEs can be viewed as extensions of ordinary AEs in the same way that convolutional neural networks are extensions of fully-connected neural networks. We have already discussed about convolutional and pooling layers, so in this section we review their reverse operations, namely, deconvolution and unpooling.

Deconvolutional Layers

A deconvolutional layer performs the reverse operation of convolution, called deconvolution, and it is used to construct decoding layers. Their function is to multiply each input value by a filter elementwise. For instance, a 2D $f_h \times f_w$ deconvolution filter maps an 1×1 spatial region of the input to an $f_h \times f_w$ region of the output. Thus, the filters learned in the deconvolutional layers create a base used for the reconstruction of the inputs' shape, taking into consideration the required shape of the output. As before, a deconvolutional layer can have multiple filters, while several deconvolutional layers can be stacked for building deep architectures for CAEs. The decoding procedure can be represented as:

$$\boldsymbol{M} = DeconvNN(\boldsymbol{M}^{enc}) \tag{5.21}$$

Unpooling layers

Unpooling layers perform the reverse operation of pooling and their aim is to reconstruct the original size of each rectangular patch. During the max pooling operation, a matrix is created which records the location of the maximum values selected during pooling. This matrix is then employed in the unpooling operation in order to place each value back to its original pooled location, while setting all other values to zero. In the case of average unpooling, it assigns the same mean value to all elements of the output window. A schematic representation of max pooling, average pooling and unpooling is given in figure 5.10.

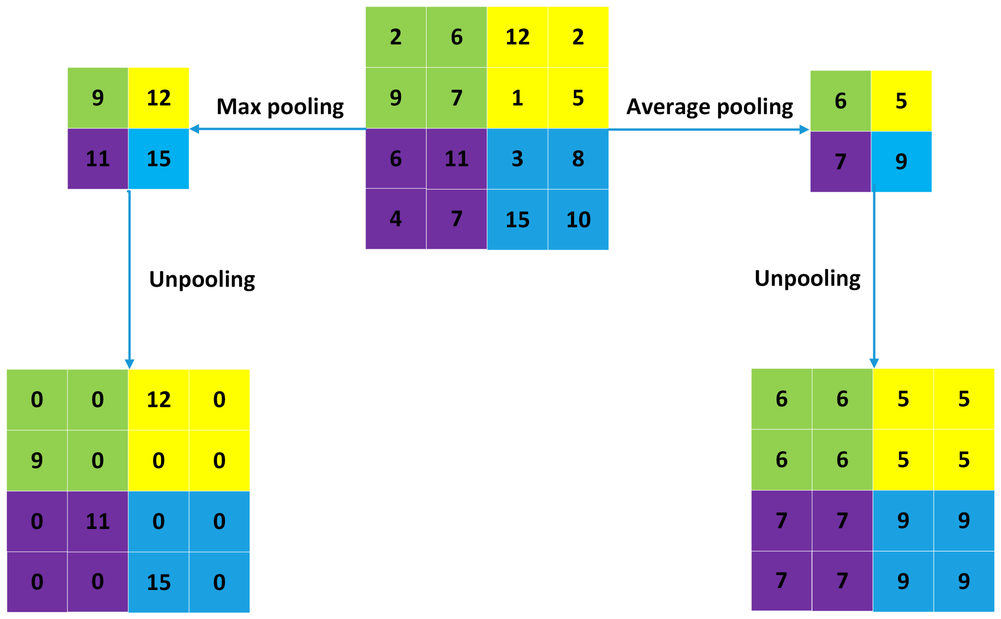


Figure 5.10: Examples of pooling and unpooling.

Based on the above, the CAE's architecture consists of convolutional/deconvolutional, pooling/unpooling and dense layers and is typically used for dimensionality reduction and reconstruction purposes. In practice, the CAE's encoder uses a number of convolutional layers to compress the input and once the desirable level of reduction has been achieved, the encoded matrix is flattened into a vector. Then, a dense layer is employed to map this vector to its latent representation. In the reverse direction, the decoder starts by taking the latent representation and transforming it into a vector through a denser layer. Subsequently, the input reconstruction is achieved by the deconvolutional layers. In accordance to eq. (5.20) the loss for CAEs becomes:

$$\mathscr{L} = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{M}_i - \tilde{\boldsymbol{M}}_i||_2^2 \tag{5.22}$$

where $\boldsymbol{M}_i$ denotes the input arrays used for training and $\tilde{\boldsymbol{M}}_i = DeconvNN(ConvNN(\boldsymbol{M}_i))$ the corresponding CAE's output. In figure 5.11, a schematic representation of a deep CAE is presented.

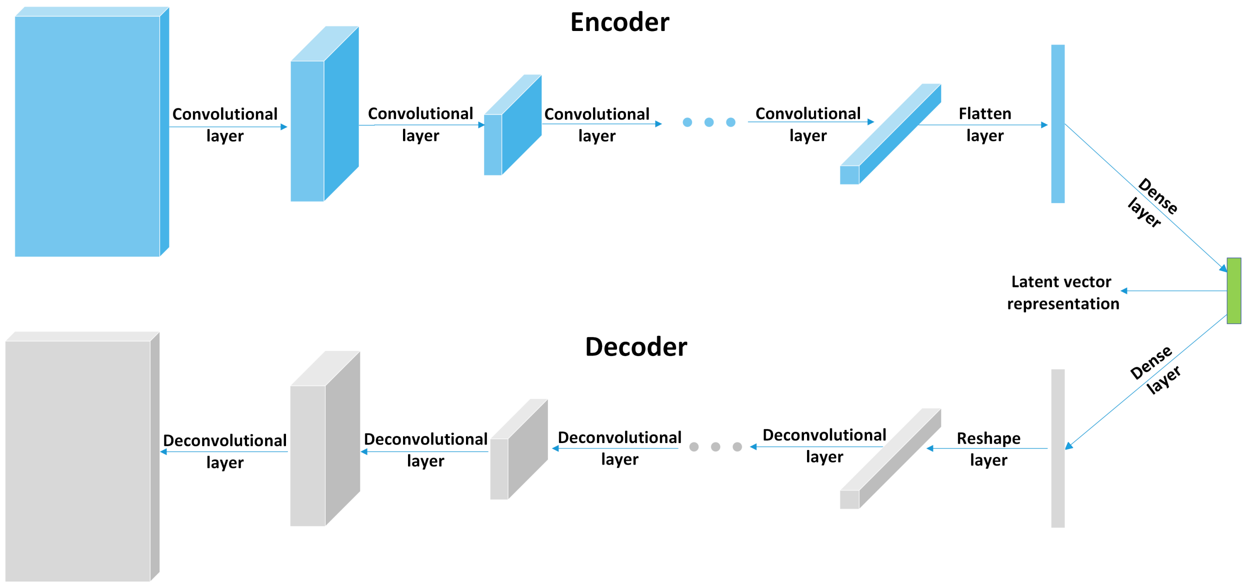


Figure 5.11: Schematic representation of a deep convolutional autoencoder.

Example: Surrogate based on FFNN and CAEs

Let us consider the 1D Burgers' equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2 u}{\partial^2 x} \tag{5.23}$$

where $u = u(x,t)$. The initial conditions were taken as $u(x,0) = -sin(\pi x)$ with $x \in [-1,1]$ and the boundary conditions $u(\pm 1,t) = 0$ with $t \in [0,5]$. In this model, ν is considered a random variable following the uniform distribution between $[0,1]$. In order to obtain exact solutions of eq. (5.23), a finite difference scheme is employed in both time and space domains, using a time step of $\Delta t = 0.0505$ sec and spatial discretization $\Delta x = 0.0101$ m, leading to 100 and 200 time and spatial points, respectively.

The first step to construct a data-driven metamodel is the generation of a sufficient number of training samples. To this purpose, Burgers' equation is solved numerically for $N = 100$ values of ν within the range $[0, 1]$. Subsequently, these solution snapshots are stored in a 3-D matrix $\mathbf{S} = [\boldsymbol{U}_1, \boldsymbol{U}_2, ..., \boldsymbol{U}_N] \in \mathbb{R}^{100\times200\times100}$, where $\boldsymbol{U}_i \in \mathbb{R}^{200\times100}$ is the velocity matrix of the i-th solution of equation (5.23).

Then, a CAE is trained over this data set and an encoded data matrix $\mathbf{S}^e = [\boldsymbol{z}_1, \boldsymbol{z}_2, ..., \boldsymbol{z}_N]$ is obtained, where each column $\boldsymbol{z}_i$ is the 8×1 latent vector representation of the solution matrix $\boldsymbol{U}_i$. The selected CAE's architecture is presented in figure 5.12.

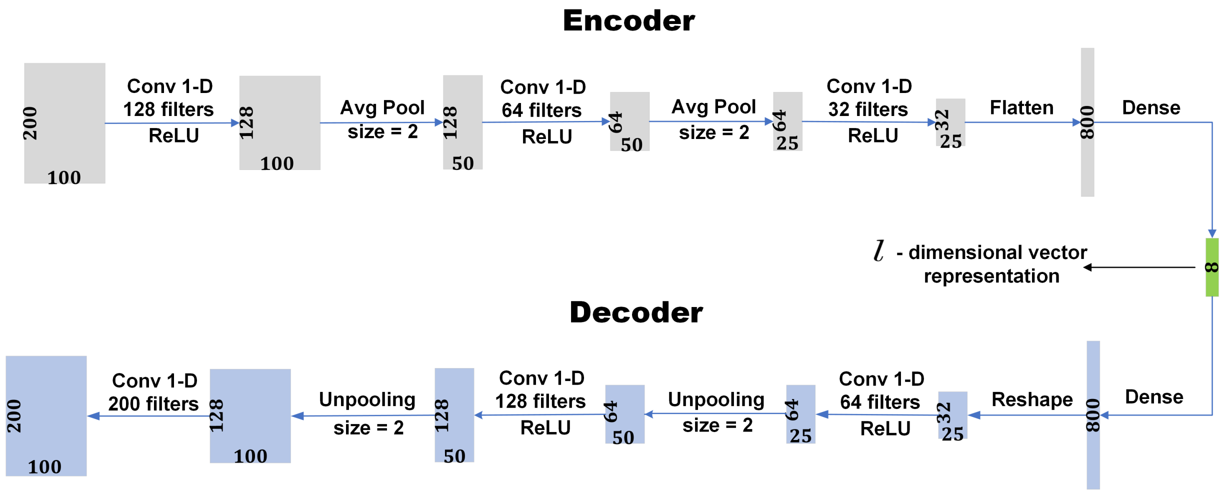


Figure 5.12: CAE architecture for the solution of Burgers' equation

The next step of the training procedure is the training of the FFNN in order to establish the mapping from the problem's parameters ν_i to the encoded vector representations $\boldsymbol{z}_i$. Using the FFNN and the decoder of the autoencoder, we can obtain the solution to the PDE at new values of ν at negligible cost. The idea of this surrogate modeling strategy is given schematically in the following figures:

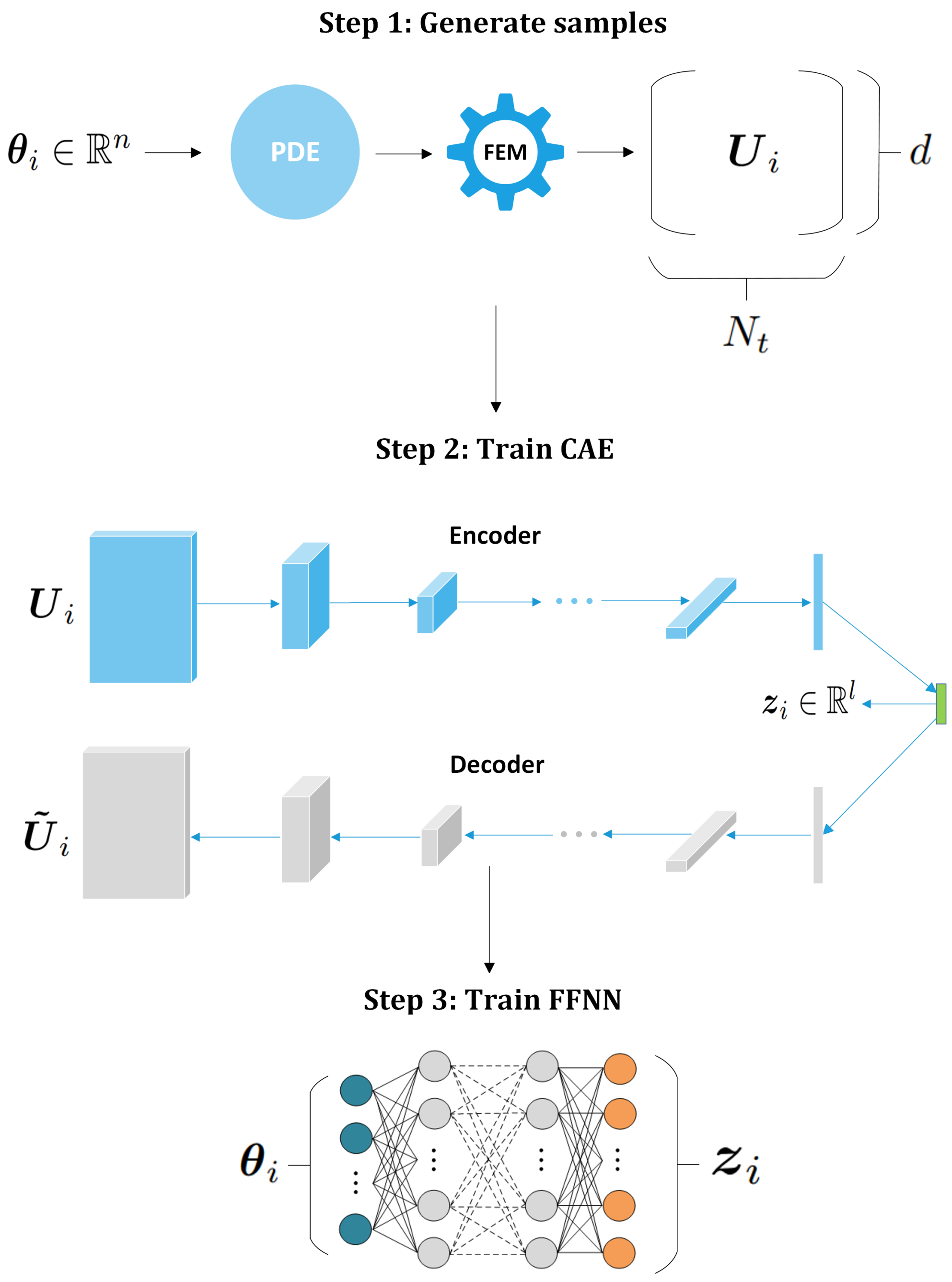


Figure 5.13: Offline phase of the proposed surrogate modeling method

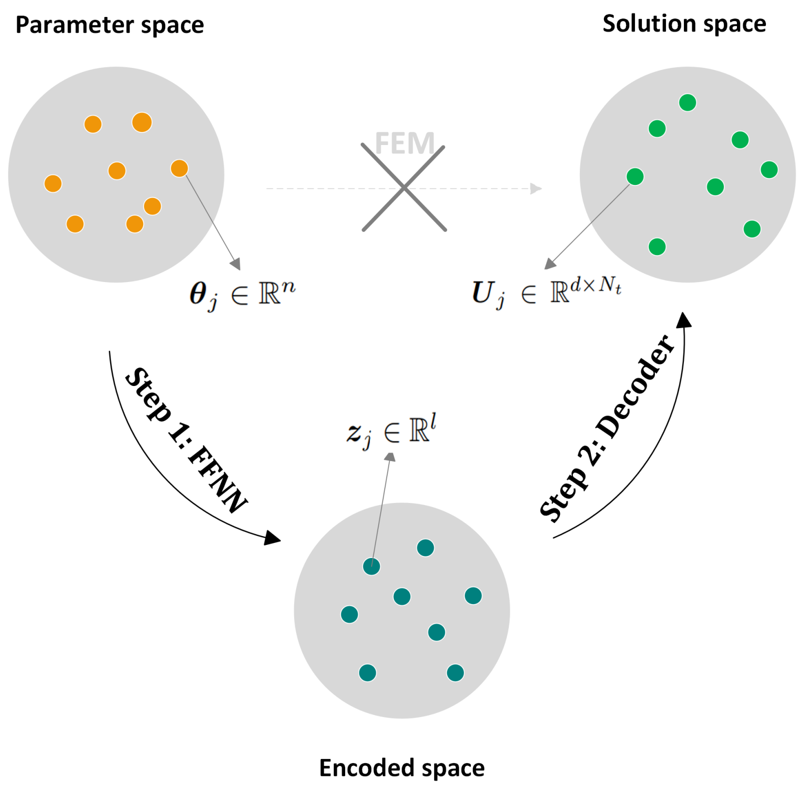
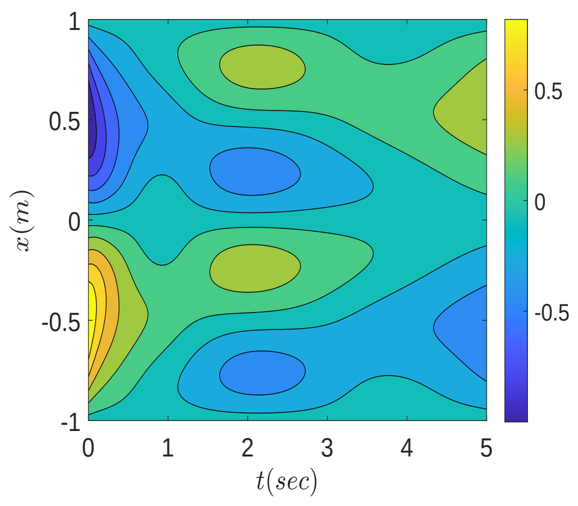
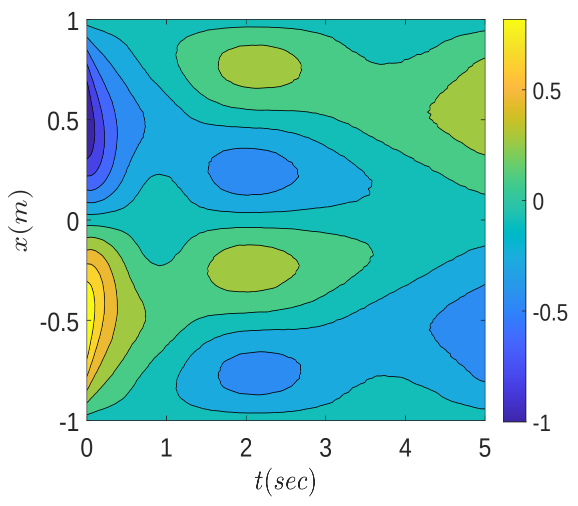


Figure 5.14: Online phase of the proposed surrogate modeling method

We now demonstrate some results to verify the accuracy of the proposed data-driven surrogate.



(a) exact model



(b) surrogate model

Figure 5.15: Solution profile $u(x,t)$ for $\nu = 0.2$ predicted by (a) the exact model and (b) the surrogate model

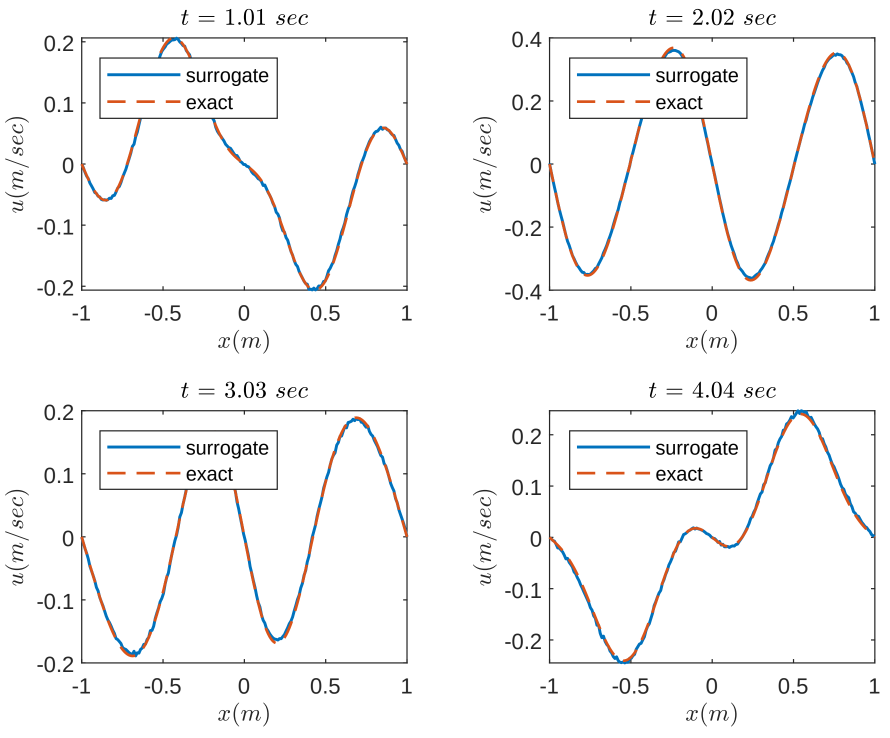


Figure 5.16: Solution profiles $u(x,t)$ at specific time instants for $\nu = 0.2$

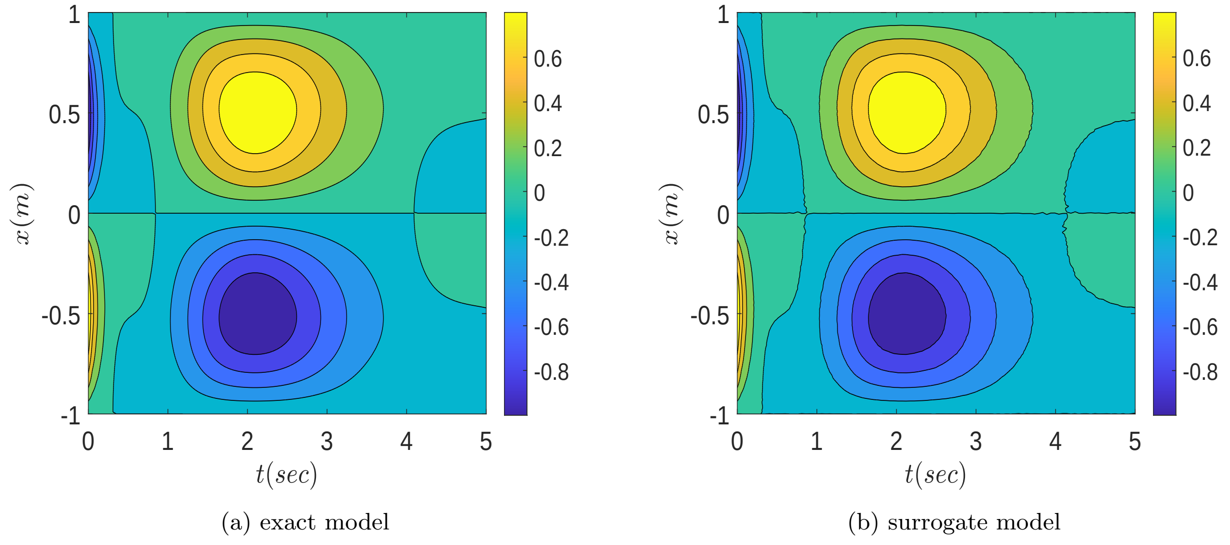


Figure 5.17: Solution profile $u(x,t)$ for $\nu = 0.8$ predicted by (a) the exact model and (b) the surrogate model

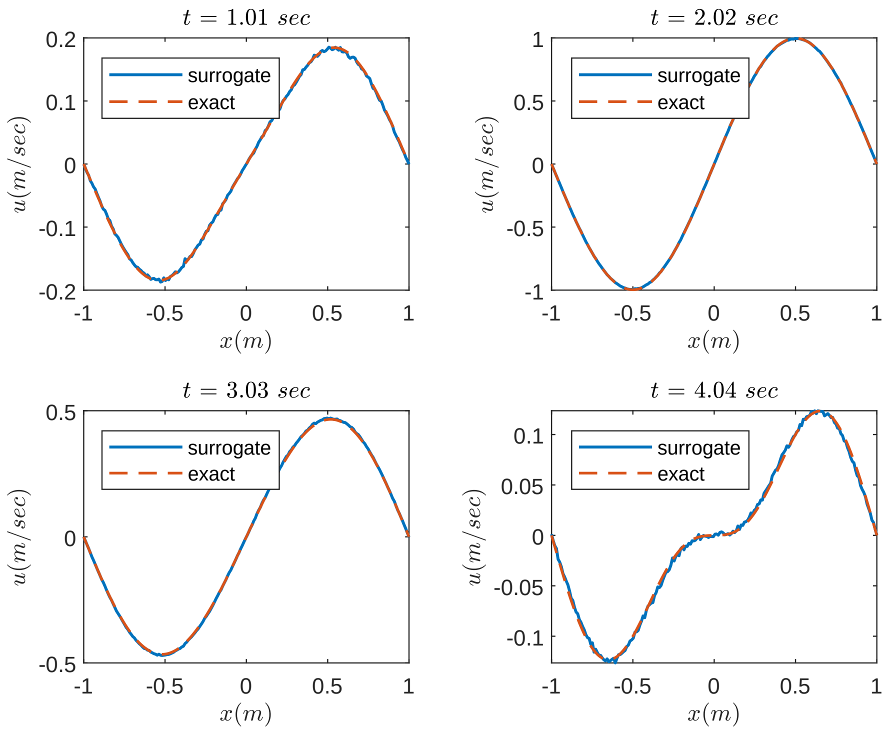
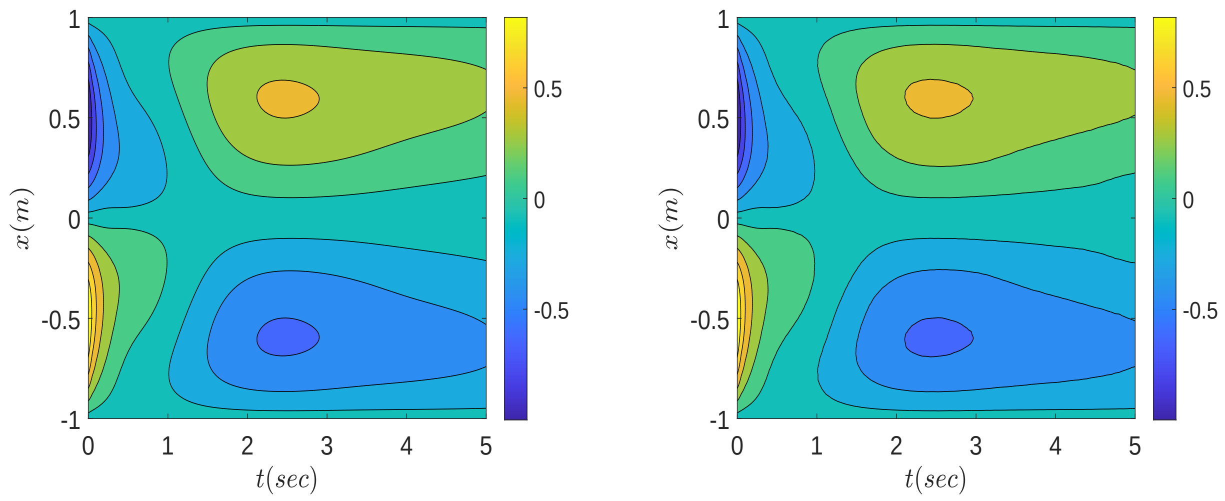


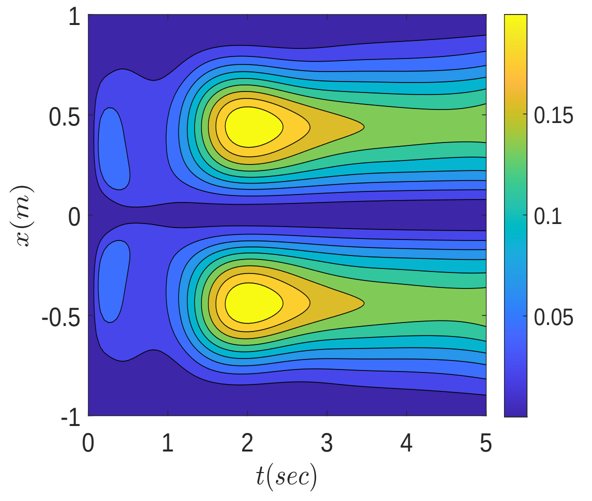
Figure 5.18: Solution profiles $u(x,t)$ at specific time instants for $\nu = 0.8$



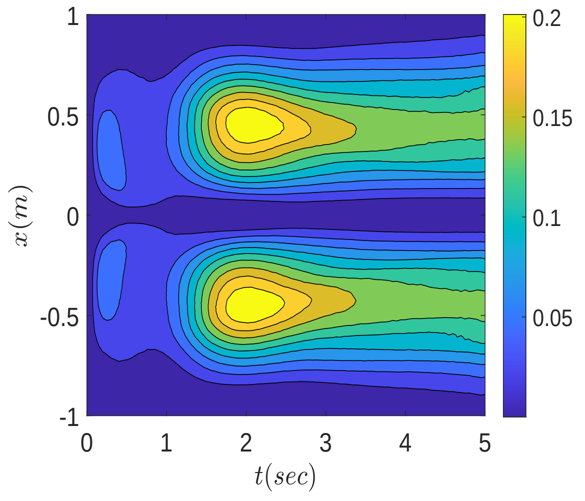
(a) exact model

(b) surrogate model

Figure 5.19: Mean value of $u(x,t)$ predicted by (a) the exact model and (b) the surrogate model



(a) exact model



(b) surrogate model

Figure 5.20: Variance of $u(x,t)$ predicted by (a) the exact model and (b) the surrogate model

5.3 Radial Basis function interpolation

Radial basis function methods can provide excellent interpolants for a large number of poorly distributed data points. Its ability to handle scarce and unevenly distributed data along with its algebraic simplicity renders this approach particularly popular in the field of machine learning. The implementation steps of this method are illustrated below.

Let $f(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$, be a real function of d variables, and let the values $f(\mathbf{x}_i)$, $i = 1, ..., N$, at the specified locations $\mathbf{x}_i$, $i = 1, ..., N$, be given. The interpolation problem consists in constructing a function $p(\mathbf{x})$, $\mathbf{x} \in \mathbb{R}^d$ such that:

$$p(\mathbf{x}_i) = f(\mathbf{x}_i), \quad \text{for } i = 1, ..., N \tag{5.24}$$

An N-dimensional vector space $\mathcal{A}$ of real functions from $\mathbb{R}^d$ to $\mathbb{R}$ is considered, which is spanned by a basis $a_j \in \mathcal{A}$, $j = 1, ..., N$. Then, the interpolant will take the following form:

$$p(\mathbf{x}) = \sum_{j=1}^{N} \lambda_j a_j(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d \tag{5.25}$$

Then, eq. (5.24) will provide the following linear system of equations

$$\sum_{j=1}^{n} \lambda_j a_j(\mathbf{x}_i) = f(\mathbf{x}_i), \quad i = 1, ..., N \tag{5.26}$$

that will determine the coefficients λ_j. These coefficients will be uniquely defined, provided that the matrix of the system is nonsingular. The matrix of the system is $N \times N$ matrix $\mathbf{\Phi}$ that has the elements $\Phi_{ij} = a_j(\mathbf{x}_i)$, $1 \leq i, j \leq N$, of the system is nonsingular. Unfortunately, the nonsingularity of Φ can not be avoided in the general case, unless specific basis functions are chosen.

In the RBF interpolation, we consider a special family of functions ϕ called radial functions, which satisfy the property $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$. We will use these functions to construct the basis vectors $\{a_j\}_{j=1}^N$ but before we do this let us first give some examples of such functions:

- Gaussian: $\phi(r) = e^{-(\epsilon r)^2}$
- Multiquadric: $\phi(r) = \sqrt{1 + (\epsilon r)^2}$
- Inverse multiquadric: $\phi(r) = \frac{1}{\sqrt{1+(\epsilon r)^2}}$
- Polyharmonic splines of odd degree: $\phi(r) = r^k, \quad k = 1, 3, 5, ...$
- Polyharmonic splines of even degree: $\phi(r) = r^{k-1} ln(r^r), \quad k = 2, 4, 6, ...$

The first three examples require the tuning of a shape parameter ϵ, while polyharmonic splines do not have such a dependence.

For a specific choice of ϕ, the basis functions are given as:

$$a_j(\mathbf{x}) = \phi(\|\mathbf{x} - \mathbf{x}_j\|), \quad j = 1, ..., N \tag{5.27}$$

and the linear system of eq. (5.26) becomes

$$\Phi \boldsymbol{\lambda} = \mathbf{f} \tag{5.28}$$

where $\boldsymbol{\lambda}$ and $\mathbf{f}$ are the vectors in $\mathbb{R}^N$ with components λ_i and $f(\mathbf{x}_i)$, $i = 1, ..., N$. Also, the matrix Φ has elements

$$\Phi_{ij} = \phi(\|\mathbf{x}_i - \mathbf{x}_j\|), \;\; 1 \leq i, j \leq N \tag{5.29}$$

Therefore, Φ is a symmetric matrix and in the Gaussian and inverse multiquadric case it is guaranteed to be positive definite, as well.

Solving the system of eq. (5.28) will provide the coefficients $\boldsymbol{\lambda}$ and the interpolant p becomes

$$p(\mathbf{x}) = \sum_{i=1}^{N} \lambda_i \phi(\|\mathbf{x} - \mathbf{x}_i\|), \quad \mathbf{x} \in \mathbb{R}^d \tag{5.30}$$

5.4 Principal Component Analysis and Principal Orthogonal Decomposition

In this section we will discuss about another data-driven modeling technique, which in the fields of machine-learning and statistics is known as Principal Component Analysis (PCA), while its application in computational mechanics is termed Principal Orthogonal Decomposition (POD)[2]. There are several ways to describe the method, but the exposition

[2]Also, the Karhunen-Loéve series representation is essentially the same method as PCA

in this section will focus on the application of PCA for **dimensionality reduction** in large data sets and, especially, PDEs.

Let $\boldsymbol{X} = [\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(N)}]$ be an $n \times N$ matrix, comprised of N observations. Then, we can consider the $\boldsymbol{X}\boldsymbol{X}^T$ symmetric matrix, which we view as the empirical sample covariance matrix of the data set. Since, $\boldsymbol{X}\boldsymbol{X}^T$ is symmetric we can perform an eigendecomposition and we can write

$$\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T \tag{5.31}$$

where $\boldsymbol{W} = [\boldsymbol{w}^{(1)}, ..., \boldsymbol{w}^{(n)}]$ is set of orthogonal eigenvectors and $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues. We usually rearrange the eigenvectors in $\boldsymbol{W}$ according to their eigenvalues (in decreasing order). The advantage that this approach offers us is that we can now dismiss all eigenvectors, whose eigenvalues are deemed too low to to affect the accuracy of the representation. For instance, we can dismiss all eigenvectors that have $\lambda < 0.05\lambda_{max}$. By keeping only the n_{red} (the subscript 'red' stands for reduced) most significant eigenvectors, we obtain a reduced representation of a vector $\boldsymbol{x} \in \mathbb{R}^n$ by writing it as $\boldsymbol{x} \approx c_1\boldsymbol{w}^{(1)} + ... + c_{n_{red}}\boldsymbol{w}^{(n_{red})}$ and considering the reduced representation $\boldsymbol{x}_{red} = [c_1, ..., c_{n_{red}}]^T \in \mathbb{R}^{n_{red}}$.

Example

Let's consider the vectors $\boldsymbol{x}_1 = [1, 1, 0.02]^T$, $\boldsymbol{x}_2 = [2, 0.5, -0.02]^T$, $\boldsymbol{x}_3 = [3, 1, 0.01]^T$ and $\boldsymbol{x}_4 = [1, 3, 0.01]^T$. We see that all four vectors approximately lie on the $x - y$ plane. Next, let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4] \in \mathbb{R}^{3\times 4}$. Performing the eigenvalue analysis of $\boldsymbol{X}\boldsymbol{X}^T$ we find:

$$\boldsymbol{W} = \begin{bmatrix} -0.7836 & -0.6212 & 0.0017 \\ -0.6212 & 0.7836 & -0.0056 \\ -0.0022 & 0.0055 & 1.0000 \end{bmatrix} \text{ and } \boldsymbol{\Lambda} = \begin{bmatrix} 21.3419 & 0 & 0 \\ 0 & 4.9084 & 0 \\ 0 & 0 & 7.51e-04 \end{bmatrix} \tag{5.32}$$

We see that the third eigenvalue is very small compared to the other two, so we can dismiss it and consider

$$\boldsymbol{W}_{red} = \begin{bmatrix} -0.7836 & -0.6212 \\ -0.6212 & 0.7836 \\ -0.0022 & 0.0055 \end{bmatrix} \text{ and } \boldsymbol{\Lambda}_{red} = \begin{bmatrix} 21.3419 & 0 \\ 0 & 4.9084 \end{bmatrix} \tag{5.33}$$

Therefore, we get $\boldsymbol{X}\boldsymbol{X}^T \approx \boldsymbol{W}_{red}\boldsymbol{\Lambda}_{red}\boldsymbol{W}_{red}^T$. Also, we the original data $\boldsymbol{x}^{(i)} \in \mathbb{R}^3$ can be represented as a vector $\boldsymbol{x}_{red}^{(i)}$ in $\mathbb{R}^2$. It is easy to check that $\boldsymbol{x}^{(1)} \approx -1.4049\boldsymbol{w}^{(1)} + 0.1625\boldsymbol{w}^{(2)} = [1, 1, 0.004]^T$, thus $\boldsymbol{x}_{red}^{(i)} = [-1.4049, 0.1625]^T$.

Principal Orthogonal Decomposition

Now let's us revisit eq. (4.5)

$$\boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{U}(\boldsymbol{\xi}) = \boldsymbol{F}(\boldsymbol{\xi}) \tag{5.34}$$

with $\boldsymbol{U} \in \mathbb{R}^n$ denoting the unknown vector of nodal values of the response. As mentioned, this will be an $n \times n$ linear system of equations with n being the number of degrees of freedom from the finite element discretization and $\boldsymbol{\xi}$ denotes the random parameters in this equation.

In the context of MCS, we generate N_{sim} samples $\{\boldsymbol{\xi}^{(1)}, ..., \boldsymbol{\xi}^{(N_{sim})}\}$ and solve N_{sim} times the $n \times n$ linear system. However, if n is large (order of 10^4 and above) the computational cost will be significant. In such cases we perform $M \ll N_{sim}$ analyses and build the $n \times M$ matrix $\mathbf{V} = [\boldsymbol{u}^{(1)}, ..., \boldsymbol{u}^{(M)}]$ which will be our data set. Next, we perform an eigenvalue analysis on $\mathbf{V}\mathbf{V}^T$. Let's suppose that $\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_K$, are the K eigenvectors corresponding to the K largest eigenvalues ($K << n$). Then, these eigenvectors form a basis of a 'good' subspace of the solution space and thus every solution can be represented in this new and lower-dimensional basis. In other words, every $\boldsymbol{U}$ can be approximated as

$$\boldsymbol{U} \approx \boldsymbol{U}_{red} = \sum_{i=1}^{K} v_i \boldsymbol{\phi}_i = [\boldsymbol{\phi}_1, ..., \boldsymbol{\phi}_K]\boldsymbol{v} = \boldsymbol{\Phi}\boldsymbol{v} \tag{5.35}$$

Inserting the reduced representation of $\boldsymbol{U}$ in eq. (5.34) we get

$$\begin{aligned} \boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{U}(\boldsymbol{\xi}) &= \boldsymbol{F}(\boldsymbol{\xi}) \Rightarrow \\ \boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{\Phi}\boldsymbol{v}(\boldsymbol{\xi}) &= \boldsymbol{F}(\boldsymbol{\xi}) \Rightarrow \\ \boldsymbol{\Phi}^T\boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{\Phi}\boldsymbol{v} &= \boldsymbol{\Phi}^T\boldsymbol{F}(\boldsymbol{\xi}) \Rightarrow \\ \boldsymbol{K}_{red}(\boldsymbol{\xi})\boldsymbol{v} &= \boldsymbol{F}_{red}(\boldsymbol{\xi}) \end{aligned} \tag{5.36}$$

The last system of equations is now $n_{red} \times n_{red}$, with $n_{red} \ll n$, therefore, the inversion of $\boldsymbol{K}_{red}$ will be far cheaper and faster than the inversion of the original matrix $\boldsymbol{K}$. Thus, solving eq. (5.36) for $\boldsymbol{v}$, we get $\boldsymbol{v} = \boldsymbol{K}_{red}^{-1}\boldsymbol{F}_{red}$. Then, we can then send $\boldsymbol{v}$ back to our original space $\mathbb{R}^n$ using the linear transformation $\boldsymbol{U} = \boldsymbol{\Phi}\boldsymbol{v}$.

Different variations of POD/PCA

1. Some authors prefer to remove the mean vector $\bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}^{(i)}$ from the data set $\boldsymbol{X}$ and consider $\bar{\boldsymbol{X}} = [\boldsymbol{x}^{(1)} - \bar{\boldsymbol{x}}, ..., \boldsymbol{x}^{(N)} - \bar{\boldsymbol{x}}]$.

2. In some applications, instead of viewing $\boldsymbol{X}\boldsymbol{X}^T$ as our correlation matrix, we consider a symmetric positive definite kernel $R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)})$, such as

$$R(\boldsymbol{x}^{(i)}, \boldsymbol{x}^{(j)}) = exp(-\frac{\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|}{l}) \tag{5.37}$$

   and perform the eigendecomposition on $R \in \mathbb{R}^{N \times N}$. This is what we did in the Karhunen-Loéve series expansion.

3. Performing eigendecomposition on matrices of large dimension is a particularly costly operation. Its complexity is $\mathcal{O}(n^{2.376})$. Now, if $n > N$, instead of the matrix $\boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{n \times n}$, we can find the eigenvalues and eigenvectors of $\boldsymbol{X}^T\boldsymbol{X} \in \mathbb{R}^{N \times N}$. Then, these are related to those of $\boldsymbol{X}\boldsymbol{X}^T$ as follows:

   Assume $\boldsymbol{v}\mathbb{R}^N$ is an eigenvector of $\boldsymbol{X}^T\boldsymbol{X}$ with eigenvalue $\lambda \neq 0$. Then,

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{v} = \lambda\boldsymbol{v} \Rightarrow$$
$$\boldsymbol{X}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{v}) = \lambda(\boldsymbol{X}\boldsymbol{v})$$

therefore $\boldsymbol{X}\boldsymbol{v} \in \mathbb{R}^n$ is an eigenvector of $\boldsymbol{X}\boldsymbol{X}^T$ with same eigenvalue λ.

5.5 Stochastic Collocation

Stochastic collocation (SC) is a data-driven modeling approach, primarily employed for uncertainty quantification. To carry out SC, one needs only a reliable deterministic simulation code that can be repetitively solved at different parameter values, just like the Monte-Carlo simulation.

Specifically, SC is a sampling-based method. The term "collocation" originates from the deterministic numerical methods for differential equations, where one seeks to satisfy the governing continuous equations discretely at a set of collocation points. This is to the contrary of the Galerkin method, where one seeks to satisfy the governing equation in a weak form. To illustrate the idea, let us consider the simple linear static problem

$$\boldsymbol{K}(\boldsymbol{\xi})\boldsymbol{U}(\boldsymbol{\xi}) = \boldsymbol{F}(\boldsymbol{\xi}) \tag{5.38}$$

with $\boldsymbol{U} \in \mathbb{R}^n$ denoting the unknown vector of nodal values of the response. Also, $\boldsymbol{\xi} = \{\xi_1, ..., \xi_d\}$ are the uncertain parameters and $\Xi \subseteq \mathbb{R}^d$ their domain. In uncertainty quantification computations, we are primarily interested in the solution dependence in the parameter space, meaning that we search for the mapping

$$\boldsymbol{U}(\boldsymbol{\xi}) : \Xi \to \mathbb{R}^n \tag{5.39}$$

In stochastic collocation, we seek to enforce the equation (5.38) at a discrete set of nodes, which are called collocation points. Let $\Theta_M = \{\boldsymbol{\xi}^{(1)}, ..., \boldsymbol{\xi}^{(M)}\}$ be a set of prescribed nodes in the random space, with M being the number of nodes (samples). Then, in SC we enforce eq. (5.38) at the node $\boldsymbol{\xi}^{(j)}$ for all $j = 1, ..., M$, by solving

$$\boldsymbol{K}(\boldsymbol{\xi}^{(j)})\boldsymbol{U}(\boldsymbol{\xi}^{(j)}) = \boldsymbol{F}(\boldsymbol{\xi}^{(j)}) \tag{5.40}$$

Just like in MCS, the result of solving eq. (5.40) is an ensemble of deterministic solutions $\boldsymbol{U}^{(j)}$, $j = 1, ..., M$. However, there are some differences between MCS and SC. In MCS, the nodal set Θ_M is randomly generated according to the distribution of $\boldsymbol{\xi}$, while in SC we typically prefer a structured mesh that explores the totality of $\boldsymbol{\Xi}$. The major difference, however, between SC and the classical MC sampling methods is that in SC, the goal is to construct an accurate approximation to the solution response function using the samples. This is a stronger goal than estimating the solution statistics. Knowing the response function(surface) of the solution allows us to immediately derive all of the statistical information of the solution. The converse does not hold, namely, knowing the solution statistics does not allow us to create the solution response function. To this end, SC can be classified as strong approximation methods, whereas the traditional MC methods are weak approximation methods.

Definition (stochastic collocation)
Let $\Theta_M = \{\boldsymbol{\xi}^{(1)}, ..., \boldsymbol{\xi}^{(M)}\} \subset \Xi$ be a set of prescribed nodes in the random space Ξ and $\{u^{(j)}\}_{j=1}^{M}$ be the corresponding solutions of eq. (5.40). Then find $\boldsymbol{w}(\boldsymbol{\xi}) \approx \boldsymbol{u}(\boldsymbol{\xi})$ such that it is an approximation to the true solution $\boldsymbol{u}(\boldsymbol{\xi})$ in the sense that $\|\boldsymbol{w}(\boldsymbol{\xi}) - \boldsymbol{u}(\boldsymbol{\xi})\|$ is sufficiently small[3].

Now, we <u>assume</u> that $\boldsymbol{w}(\boldsymbol{\xi})$ belongs to a vector subspace of L^2, and this vector subspace is the one spanned by a finite number of polynomials in the Polynomial Chaos. We assume that this subspace is the space of polynomial of degree up to p. Then, if $dim\Xi = d$, we can write

$$\boldsymbol{w}(\boldsymbol{\xi}) = \sum_{i=0}^{P-1} \boldsymbol{c}_i \Psi_i(\boldsymbol{\xi}) \tag{5.41}$$

where $P = \sum_{q=0}^{p} \binom{d+q-1}{q}$ and $\boldsymbol{c}_i \in \mathbb{R}^n$ are the unknown vectors of coefficients in the above expansion. So, the goal is to find these coefficients using the data set of solutions $\{u^{(j)}\}_{j=1}^{M}$ that we already have. There are three approaches to do so: Stochastic Collocation via Interpolation, Stochastic Collocation via Regression and Stochastic Collocation via Pseudo Projection. We will focus on Stochastic Collocation via Regression. The idea is the following.

For every sample $\boldsymbol{\xi}^{(j)}$ we know its corresponding solution $\boldsymbol{u}^{(j)}$. Also, we know that $\boldsymbol{w}(\boldsymbol{\xi}^{(j)}) = \sum_{i=0}^{P-1} \boldsymbol{c}_i \Psi_i(\boldsymbol{\xi}^{(j)})$. Then, we seek these vectors of coefficients $\boldsymbol{c}_i$, $i = 0, ..., P-1$ such that $\|\boldsymbol{w}(\boldsymbol{\xi}) - \boldsymbol{u}(\boldsymbol{\xi})\|_2$. These are given by solving

$$\boldsymbol{AC} = \boldsymbol{F} \tag{5.42}$$

where

- $\boldsymbol{A}$ is the $M \times P$ matrix with entries $a_{ij} = \Psi_{j-1}(\boldsymbol{\xi}^{(i)})$, $i = 1, ..., M$ and $j = 1, ..., P$

- $\boldsymbol{F} = [\boldsymbol{u}^{(1)}, ..., \boldsymbol{u}^{(M)}]^T$ is the $M \times n$ matrix built from the solution data set
- $\boldsymbol{C}$ is the $P \times n$ matrix of the coefficients in the expansion, that is $\boldsymbol{C} = [\boldsymbol{c}_0, ..., \boldsymbol{c}_{P-1}]^T$.

Then we find the matrix $\boldsymbol{C}$ that minimizes $\|\boldsymbol{w}(\boldsymbol{\xi}) - \boldsymbol{u}(\boldsymbol{\xi})\|_2$ (known as the least-squares solution) by the formula

$$\boldsymbol{C} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{F} = \boldsymbol{A}^\dagger \boldsymbol{F} \tag{5.43}$$

where $\boldsymbol{A}^\dagger$ denotes the (Moore-Penrose) pseudo-inverse of $\boldsymbol{A}$.

As a closing remark, having obtained the matrix of coefficients $\boldsymbol{C}$, then we can use the formula of eq. (5.41) to get the solution vector for every other new sample $\boldsymbol{\xi}_{new}$.

5.6 Gaussian Process regression/Kriging

As a general term, regression belongs to the more general concept of supervised learning. It is concerned with the prediction of continuous continuities based on a discrete set of

[3]Typically, we employ L^p-norms and most commonly the L^2-norm, leading to mean-square approximation

labeled data. In this section, we will describe Gaussian Process (GP) methods for regression problems.

Let us denote our training set of observations as $\mathscr{D}\{(\boldsymbol{x}_i, y_i)|i = 1, ..., n\}$, where $\boldsymbol{x}$ denotes an input vector (covariates) of dimension D and y denotes a scalar output or target (dependent variable). We write the inputs $\boldsymbol{x}_i$ of $\mathscr{D}$ as column vectors and form the $D \times n$ matrix $\boldsymbol{X}$, called the design matrix. Also, the targets are collected in the vector $\boldsymbol{y}$, so we can write $\mathscr{D} = (\boldsymbol{X}, \boldsymbol{y})$. We are interested in making inferences about the relationship between inputs and targets, which in this setting is viewed as a conditional distribution of the targets given the inputs.

5.6.1 Linear Regression model with Gaussian noise

At this point, we will recall the standard linear regression model with Gaussian noise:

$$f(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{w}, \qquad y = f(\boldsymbol{x}) + \epsilon \tag{5.44}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ is a vector of weights (parameters) of the linear model, f is the function value and y is the target value [4].We have also assumed that the observed values y differ from the function values $f(\boldsymbol{x})$ by additive noise ϵ, and we will further assume that this noise follows an independent, identically distributed Gaussian distribution with zero mean and variance σ_n^2

$$\epsilon \sim \mathcal{N}(0, \sigma_n^2) \tag{5.45}$$

Now, we consider the following likelihood function, namely, the probability density of the observations given the parameters

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) &= \prod_{i=1}^{n} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_n} exp\left(-\frac{\left(y_i - \boldsymbol{x}_i^T\boldsymbol{w}\right)^2}{2\sigma_n^2}\right) \\ &= \frac{1}{(2\pi\sigma_n^2)^{n/2}} exp\left(-\frac{1}{2\sigma_n^2}|\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}|^2\right) = \mathcal{N}(\boldsymbol{X}^T\boldsymbol{w}, \sigma_n^2\boldsymbol{I}) \end{aligned} \tag{5.46}$$

where $|\cdot|$ denotes the Euclidean length of a vector. Next, we specify a prior [5] over the parameters, which is a probability distribution that expresses our beliefs about the parameters before we look at the observations. Typically, we put a zero mean Gaussian prior with covariance matrix Σ_p on the weights

$$\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \Sigma_p) \tag{5.47}$$

Inference in the Bayesian linear model is based on the posterior distribution over the weights, computed by the Bayes' rule, which states that

$$posterior = \frac{likelihood \times prior}{marginal\ likelihood} \tag{5.48}$$

[4]Often a bias weight or offset is included, but as this can be implemented by augmenting the input vector $\boldsymbol{x}$ with an additional element whose value is always one, we do not explicitly include it in our notation.

[5]prior, likelihood these terms come from the Bayesian formalism, which we will see in later chapters

or,

$$p(\boldsymbol{w}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{X})} \tag{5.49}$$

where the normalizing constant in the denominator, called the marginal likelihood is given by

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w} \tag{5.50}$$

The posterior in eq. (5.49) combines the likelihood and the prior, and captures everything we know about the parameters. Writing only the terms from the likelihood and prior which depend on the weights, we obtain

$$\begin{aligned} p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) &\propto exp\left(-\frac{1}{2\sigma_n^2}\left(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\right)^T\left(\boldsymbol{y} - \boldsymbol{X}^T\boldsymbol{w}\right)\right) exp\left(-\frac{1}{2}\boldsymbol{w}^T\Sigma_p^{-1}\boldsymbol{w}\right) \\ &\propto exp\left(-\frac{1}{2}\left(\boldsymbol{w} - \bar{\boldsymbol{w}}\right)^T\left(\frac{1}{\sigma_n^2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_p^{-1}\right)\left(\boldsymbol{w} - \bar{\boldsymbol{w}}\right)\right) \end{aligned} \tag{5.51}$$

where $\bar{\boldsymbol{w}} = \sigma_n^{-2}(\sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_p^{-1})^{-1}\boldsymbol{X}\boldsymbol{y}$ and we recognize the form of the posterior distribution as Gaussian with mean $\bar{\boldsymbol{w}}$ and covariance matrix $\boldsymbol{A}^{-1}$

$$p(\boldsymbol{w}|\boldsymbol{X}, y) \sim \mathcal{N}(\bar{\boldsymbol{w}}, \boldsymbol{A}^{-1}) \tag{5.52}$$

with $\boldsymbol{A} = \sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^T + \Sigma_p^{-1}$.

Now that we know the posterior distribution $p(\boldsymbol{w}|\boldsymbol{X}, y)$, we can make predictions of a new test case $\boldsymbol{x}_\star$ by averaging over all possible parameter values, weighted by their posterior probability. In other words, the predictive distribution for $f_\star := f(\boldsymbol{x}_\star)$ at $\boldsymbol{x}_\star$ is given by averaging the output of all possible linear models w.r.t. the Gaussian posterior

$$\begin{aligned} p(f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y}) &= \int p(f_\star|\boldsymbol{x}_\star, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w} \\ &= \mathcal{N}(\frac{1}{\sigma_n^2}\boldsymbol{x}_\star^T\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{y}, \boldsymbol{x}_\star^T\boldsymbol{A}^{-1}\boldsymbol{x}_\star) \end{aligned} \tag{5.53}$$

In essence, the above equation tells us that for each new design point $\boldsymbol{x}_\star$ the output $f_\star$ will be a random variable with the pdf shown in eq. (5.53). Nevertheless, we can also derive a deterministic relation by taking $f_\star$ to be the most probable value of $p(f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y})$, namely, the mode of the pdf, which in the Gaussian case is the same as its mean value $\frac{1}{\sigma_n^2}\boldsymbol{x}_\star^T\boldsymbol{A}^{-1}\boldsymbol{X}\boldsymbol{y}$.

Example

Example of a linear model $f(x) = w_1 + w_2x$ with intercept w_1 and slope parameter w_2.

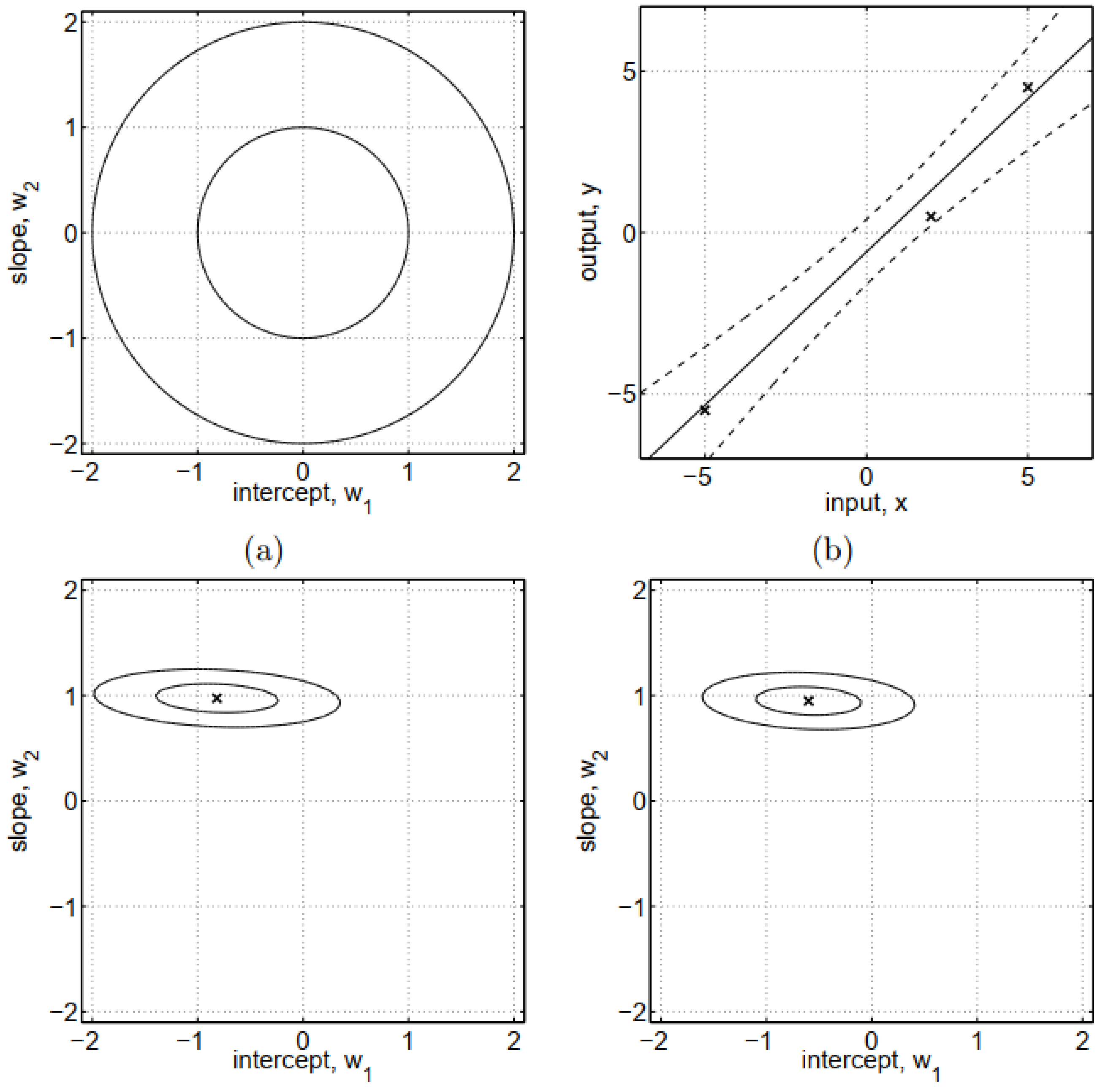


Figure 5.21: Panel (a) shows the contours of the prior distribution $p(\boldsymbol{w}) \sim \mathcal{N}(\mathbf{0}, I)$. Panel (b) shows three training points marked by crosses. Panel (c) shows contours of the likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$, assuming a noise level of $\sigma_n = 1$; note that the slope is much more "well determined" than the intercept. Panel (d) shows the posterior $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$; comparing the maximum of the posterior to the likelihood, we see that the intercept has been shrunk towards zero whereas the more "well determined" slope is almost unchanged. All contour plots give the 1 and 2 standard deviation equi-probability contours. Superimposed on the data in panel (b) are the predictive mean plus/minus two standard deviations of the predictive distribution $p(f_\star|\boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y})$

5.6.2 More general formulation

Instead of considering the additive noise ϵ in eq. (5.44), we can assume a model of the form:

$$y(\boldsymbol{x}) = f(\boldsymbol{x}) + z(\boldsymbol{x}) \tag{5.54}$$

where f is the regression model in $\boldsymbol{x}$, with the difference that now $\{f(\boldsymbol{x}_i)\}_{i=1}^n = \{\boldsymbol{w}^T s(\boldsymbol{x}_i)\}_{i=1}^n$ for some function s and $z(\boldsymbol{x})$ is a **zero-mean random process** with covariance function defined as:

$$C_z(\boldsymbol{x_1}, \boldsymbol{x_2}) = \mathbb{E}[z(\boldsymbol{x_1})z(\boldsymbol{x_2})] \tag{5.55}$$

we can consider the process $z(\boldsymbol{x})$ to be weakly stationary and we can denote $R_z(\tau)$, $\tau = d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|$ its normalized correlation function:

$$\sigma_z^2 R_z(\tau) = C_z(\boldsymbol{x}_1, \boldsymbol{x}_2) \tag{5.56}$$

where σ_z^2 is the variance of the process. A common choice for the correlation model in $\mathbb{R}$ is the exponential kernel we have seen in previous sections $exp(-\frac{|x_1 - x_2|}{\rho})$, ρ being the correlation length parameter. However, more general correlation models can be chosen from the Matérn class of functions:

$$R_z(d) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(\sqrt{2\nu}\frac{d}{\rho}\right)^{\nu} K_\nu \left(\sqrt{2\nu}\frac{d}{\rho}\right) \tag{5.57}$$

Here, Γ is the gamma function, K_ν is the modified Bessel function of the second kind and ν, ρ are non-negative parameters that control, respectively, the smoothness and the correlation length of the stochastic process. The likelihood function now becomes

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}, \rho) = \frac{1}{(2\pi\sigma_z^2)^{n/2}} \frac{1}{\sqrt{det(\boldsymbol{R})}} exp\left(-\frac{(\boldsymbol{y} - \boldsymbol{f})^T \boldsymbol{R}^{-1} (\boldsymbol{y} - \boldsymbol{f})}{2\sigma_z^2}\right) \tag{5.58}$$

Here, $\boldsymbol{R}$ depends on ρ and is the $n \times n$ matrix of correlations of the n design points with $R_{ij} = R(d(\boldsymbol{x}_i, \boldsymbol{x}_j))$, $\boldsymbol{y} = \{y_i\}_{i=1}^n$ and $\boldsymbol{f} = \{\boldsymbol{w}^T s(\boldsymbol{x}_i)\}_{i=1}^n$.

After some operations, we can evaluate the posterior $p(\boldsymbol{w}, \sigma_z^2, \rho|\boldsymbol{X}, \boldsymbol{y})$ by appropriately adjusting eq. (5.51). We omit these calculations and instead present here the final prediction formula:

$$\hat{y}(\boldsymbol{x}_\star) = f(\boldsymbol{x}_\star) + \boldsymbol{r}^T(\boldsymbol{x}_\star)\boldsymbol{R}^{-1}(\boldsymbol{y} - \boldsymbol{f}) \tag{5.59}$$

where $\hat{y}(\boldsymbol{x}_\star)$ is the GP's prediction at the design point $\boldsymbol{x}_\star$ and $\boldsymbol{r}^T(\boldsymbol{x}_\star) = [R(d(\boldsymbol{x}_1, \boldsymbol{x}_\star)), ..., R(d(\boldsymbol{x}_n, \boldsymbol{x}_\star))]$ is the vector of correlations between the design point $\boldsymbol{x}_\star$ and the observation points $\boldsymbol{x}$.

5.7 Additional literature

1. Hands-on Machine Learning with Scikit-Learn, Keras and Tensorflow, Aurelion Geron, 2nd edition

2. Pattern Recognition and Machine Learning, Christopher M. Bishop, Springer, 2006

Chapter 6

Spectral Stochastic Finite Element Method

The spectral stochastic finite element method is an extension of the deterministic finite element method for problems involving random material properties and\or loading conditions. For instance, if a material property, such as the modulus of elasticity is considered to be of stochastic nature, that is, $E = E(\theta)$, with θ denoting the random outcome, then the system's response will also be stochastic, $\mathbf{U} = \mathbf{U}(\theta)$. This implies that both E and $\mathbf{U}$ are elements of the $\mathscr{L}^2(\Theta, \mathscr{F}, \mathscr{P})$ probability space, and thus they can be projected onto appropriate bases of this space.

6.1 SSFEM for linear systems

In the SSFEM approach, it is preferred to use the Karhunen-Loéve expansion when the stochastic input parameters are Gaussian random fields and the Polynomial Chaos expansion for the system's response. More particularly, the truncated KL expansion for a random field $a(\mathbf{x}, \theta)$, describing a stochastic input parameter, with given covariance function $C(\mathbf{x}_1, \mathbf{x}_2)$ is defined as

$$\hat{a}(\mathbf{x}, \theta) = a_0(\mathbf{x}) + \sum_{j=1}^{M} \sqrt{\lambda_j} \xi_j(\theta) \varphi_j(\mathbf{x}) \tag{6.1}$$

where, λ_j are the eigenvalues of C, $\varphi_j(\mathbf{x})$ are the associated eigenfunctions and $\xi_j(\theta)$ are independent standard normal variables.

Using conventional notation, the equation of equilibrium of a linear static system of size $N \times N$ reads

$$\boldsymbol{K}\mathbf{U} = \mathbf{F} \tag{6.2}$$

In the above equation, $\mathbf{U}$ is the $N \times 1$ vector of the unknown displacements, resulting from the finite element discretization, $\mathbf{F}$ the corresponding loading vector and $\boldsymbol{K}$ the $N \times N$ global stiffness matrix, which is obtained after assembling the element stiffness matrices $\boldsymbol{K}^e$:

$$\boldsymbol{K}^e = \int_{\Omega_e} \boldsymbol{B}^T \boldsymbol{D} \boldsymbol{B} d\Omega_e \tag{6.3}$$

where, $\boldsymbol{D}$ and $\boldsymbol{B}$ stand for the constitutive and the deformation matrix, respectively.

Let us consider now the case the modulus of elasticity E is a Gaussian random field. Then, the constitutive matrix in point $\mathbf{x}$ can be written as:

$$\boldsymbol{D}(\mathbf{x}, \theta) \equiv E(\mathbf{x}, \theta)\boldsymbol{D}_0 \tag{6.4}$$

with $\boldsymbol{D}_0$ being a constant matrix. The Karhunen-Loéve expansion of E reads:

$$E(\mathbf{x}, \theta) = E_0(\mathbf{x}) + \sum_{j=1}^{M} \sqrt{\lambda_j}\xi_j(\theta)\varphi_j(\mathbf{x}) \tag{6.5}$$

where, $E_0(\mathbf{x})$ is the mean value of the field in point $\mathbf{x}$. Using eqs. (6.4) and (6.5), eq. (6.3) takes the form:

$$\boldsymbol{K}^e(\theta) = \boldsymbol{K}_0^e + \sum_{j=1}^{\infty} \boldsymbol{K}_j^e \xi_j(\theta) \tag{6.6}$$

In the above equation, $\boldsymbol{K}_0^e$ is the element's mean stiffness matrix and $\boldsymbol{K}_j^e$ are deterministic matrices obtained by:

$$\boldsymbol{K}_j^e = \sqrt{\lambda_j} \int_{\Omega_e} \varphi_j(\mathbf{x}) \boldsymbol{B}^T \boldsymbol{D}_0 \boldsymbol{B} d\Omega_e \tag{6.7}$$

Assembling the above element contributions in a manner similar to the deterministic case, the equation of equilibrium for the stochastic case becomes

$$\Big(\boldsymbol{K}_0 + \sum_{j=1}^{M} \boldsymbol{K}_j \xi_j(\theta)\Big)\mathbf{U}(\theta) = \mathbf{F} \tag{6.8}$$

$\boldsymbol{K}_0$ is the mean global stiffness matrix and $\boldsymbol{K}_j$ are deterministic matrices obtained by the assembly of $\boldsymbol{K}_j^e$.

Next, the system's response is projected in a PC basis as follows

$$\mathbf{U}(\theta) = \sum_{k=0}^{P-1} \mathbf{U}_k \Psi_k(\boldsymbol{\xi}) \tag{6.9}$$

where, $\{\Psi_k(\boldsymbol{\xi})\}_{k=0}^{P-1} = \{\Psi_k((\xi_1(\theta), ..., \xi_M(\theta)\}_{k=0}^{P-1}$ is the PC basis, consisting of the M-dimensional orthogonal Hermite polynomials of order p. The order of P in Eq. (6.9) is determined by the following formula

$$P = \sum_{q=0}^{p} \binom{M+q-1}{q} \tag{6.10}$$

By denoting $\xi_0(\theta) \equiv 1$, eq. (6.8) can be more compactly written as

$$\left(\sum_{j=0}^{M} \boldsymbol{K}_j \xi_j(\theta)\right)\left(\sum_{k=0}^{P-1} \mathbf{U}_k \Psi_k(\theta)\right) = \mathbf{F} \tag{6.11}$$

with the following residual due to the truncation error

$$\epsilon_{M,P} = \sum_{j=0}^{M}\sum_{k=0}^{P-1} \boldsymbol{K}_j \mathbf{U}_k \xi_j(\theta)\Psi_k(\theta) - \mathbf{F} \tag{6.12}$$

Galerkin minimization of the residual in the mean square sense leads to

$$\mathbb{E}[\epsilon_{M,P} \cdot \Psi_l] = 0 \quad l = 0, 1, ..., P-1 \tag{6.13}$$

Introducing the notation

$$c_{jkl} = \mathbb{E}[\xi_j \Psi_k \Psi_l] \tag{6.14}$$

$$\mathbf{F}_l = \mathbb{E}[\Psi_l \mathbf{F}] \tag{6.15}$$

eq. (6.11) can be rewritten as

$$\sum_{j=0}^{M}\sum_{k=0}^{P-1} c_{jkl}\boldsymbol{K}_j \mathbf{U}_k = \mathbf{F}_l \tag{6.16}$$

and denoting

$$\boldsymbol{K}_{kl} = \sum_{j=0}^{M} c_{jkl}\boldsymbol{K}_j \tag{6.17}$$

eq. (6.16) rewrites as

$$\sum_{k=0}^{P-1} \boldsymbol{K}_{kl}\mathbf{U}_k = \mathbf{F}_l \tag{6.18}$$

In the above equation, each $\mathbf{U}_k$ is a N-dimensional vector and each $\boldsymbol{K}_{kl}$ is a matrix of size $N \times N$. The P different equations can be cast in a matrix form of size $NP \times NP$ as follows:

$$\bar{\boldsymbol{K}}\bar{\mathbf{U}} = \bar{\mathbf{F}} \tag{6.19}$$

where,

$$\bar{\boldsymbol{K}} = \begin{bmatrix} \sum_{j=0}^{M} c_{j,0,0}\boldsymbol{K}_j & \sum_{j=0}^{M} c_{j,1,0}\boldsymbol{K}_j & \cdots & \sum_{j=0}^{M} c_{j,P-1,0}\boldsymbol{K}_j \\ \sum_{j=0}^{M} c_{j,0,1}\boldsymbol{K}_j & \sum_{j=0}^{M} c_{j,1,1}\boldsymbol{K}_j & \cdots & \sum_{j=0}^{M} c_{j,P-1,1}\boldsymbol{K}_j \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{j=0}^{M} c_{j,0,P-1}\boldsymbol{K}_j & \sum_{j=0}^{M} c_{j,1,P-1}\boldsymbol{K}_j & \cdots & \sum_{j=0}^{M} c_{j,P-1,P-1}\boldsymbol{K}_j \end{bmatrix} \tag{6.20}$$

and

$$\bar{\mathbf{U}} = [\mathbf{U}_0, \mathbf{U}_1, ..., \mathbf{U}_{P-1}]^T \tag{6.21}$$

$$\bar{\mathbf{F}} = [\mathbf{F}_0, \mathbf{F}_1, ..., \mathbf{F}_{P-1}]^T \tag{6.22}$$

It should be noted here that, in the case that $\mathbf{F}$ is deterministic

$$\mathbf{F}_k = \{\mathbf{0}\} \quad k > 0 \tag{6.23}$$

Solving the augmented system of equations in (6.19), will give the PC expansion coefficients $\mathbf{U}_k$, which, in turn, will produce the final solution

$$\mathbf{U}(\theta) = \sum_{k=0}^{P-1} \mathbf{U}_k \Psi_k(\theta) \tag{6.24}$$

An easy example

Consider the rod shown in the figure below. Use the SSFEM to find the mean value and standard deviation of the response u using polynomials of degree (a) $p = 1$ and (b) $p = 2$.

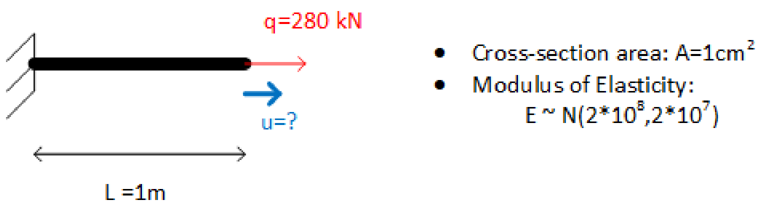


Figure 6.1: Rod with random E

Solution: The equation of equilibrium for the rod reads $K \cdot u = q$, with $K = \frac{EA}{L}$. Since $E \sim N(2 \cdot 10^8, 2 \cdot 10^7)$, then it can be written as:

$$E = 2 \cdot 10^8 + 2 \cdot 10^7 \xi \quad (= E_0 + \sigma\xi) \tag{6.25}$$

Consequently,

$$\begin{aligned} K &= \frac{EA}{L} = \frac{E_0 A}{L} + \sigma\xi \frac{E_0 A}{L} = K_0 + K_1 \xi \\ &= 20000 + 2000\xi \end{aligned}$$

For $M = 1$ 'KL-terms' in this example and for polynomials of degree $p = 2$, we have $P = 3$ terms in the PC expansion of the unknown response u. So, $u = u_0\Psi_0 + u_1\Psi_1 + u_2\Psi_2$, where

$$\begin{aligned} p = 0 &: \ \Psi_0 = 1 \\ p = 1 &: \ \Psi_1 = \xi \\ p = 2 &: \ \Psi_2 = \xi^2 - 1 \end{aligned}$$

The next step is to compute the coefficients $c_{ijk} = \mathbb{E}[\xi_i \Psi_j \Psi_k]$ using the formula

$$c_{ijk} = \int_{-\infty}^{\infty} \xi_i \Psi_j \Psi_k \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} d\xi \tag{6.26}$$

For example,

$$c_{000} = \int_{-\infty}^{\infty} 1 \cdot 1 \cdot 1 \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} d\xi = 1$$
$$c_{011} = \int_{-\infty}^{\infty} 1 \cdot \xi \cdot \xi \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} d\xi = 1$$
$$c_{112} = \int_{-\infty}^{\infty} \xi \cdot \xi \cdot (\xi^2 - 1) \frac{1}{\sqrt{2\pi}} e^{-\frac{\xi^2}{2}} d\xi = 2$$

Knowing the coefficients c_{ijk} it is straightforward to compute the 3×3 augmented matrix $\bar{\boldsymbol{K}}$. We also need the augmented force vector $\bar{\boldsymbol{F}} = [q_1, q_2, q_3]^T$, where $q_k = \mathbb{E}[q \cdot \Psi_k]$, $k = 1, 2, 3$. It is easy to verify that $q_1 = 280, q_2 = q_3 = 0$.

Eventually, the augmented system becomes:

$$\begin{bmatrix} K_{00} & K_{01} & K_{02} \\ K_{10} & K_{11} & K_{12} \\ K_{20} & K_{21} & K_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} \tag{6.27}$$

and solving this system will return the coefficients u_0, u_1, u_2. The solution is

$$\begin{aligned} u &= u_0\Psi_0 + u_1\Psi_1 + u_2\Psi_2 \\ &= 0.01414 \cdot 1 - 0.001443\xi + 0.000144(\xi^2 - 1) \end{aligned} \tag{6.28}$$

Now, we can compute:

$$\mathbb{E}[u] = u_0\mathbb{E}[\Psi_0] + u_1\mathbb{E}[\Psi_1] + u_2\mathbb{E}[\Psi_2] = u_0 = 0.01414 \tag{6.29}$$

and

$$Var[u] = \mathbb{E}[u^2] - (\mathbb{E}[u])^2 = \cdots = u_1^2\mathbb{E}[\Psi_1^2] + u_2^2\mathbb{E}[\Psi_2^2] = 2.125 \cdot 10^{-6} \tag{6.30}$$

Also, we can now generate a very large number of samples for ξ from $N(0, 1)$ and through eq. (6.28) find the corresponding values of u. This allows us to produce a histogram, or an empirical pdf, for u at negligible cost.

6.2 SSFEM for linear systems with non-Gaussian parameters

In the formulation presented in the previous section, Gaussian random fields were considered for the modeling of the system's random parameters. This assumption facilitates computations, however, it can be applied only to problems with small coefficient of variation (~15-20%), since it can be physically inconsistent in cases where the random parameters are not allowed to take negative values. Such is the case of the modulus of elasticity, where negative values of E would result in a displacement vector of infinite variance, and hence the solution would not belong to the $\mathscr{L}^2(\Theta, \mathscr{F}, \mathscr{P})$ probability space. As a consequence the PCE approximation is not convergent in the $\mathscr{L}^2$ sense. An approach to overcome this drawback is to use the PCE to represent the material property instead of the KL expansion, and will be briefly illustrated here.

The main idea in this approach is to identify an underlying Gaussian field for the corresponding non-Gaussian field. Then, all the equations in the previous section remain valid under some minor modifications. More specifically, the coefficients c_{jkl} in eq. (6.14) will now be given by the relation:

$$c_{jkl} = \mathbb{E}[\Psi_j \Psi_k \Psi_l] \tag{6.31}$$

and the upper bound in the summations in the previous equations will become P instead of M.

To further illustrate this process, let us consider the case where E is represented by a lognormal random field, typically used in most applications. In this case, the field can be represented by the exponential of an underlying Gaussian field, denoted as $E^G(\mathbf{x}, \theta)$, which has the following KL expansion:

$$E^G(\mathbf{x}, \theta) = E_0^G(\mathbf{x}) + \sum_{j=1}^{M} \sqrt{\lambda_j} \xi_j(\theta) \varphi_j(\mathbf{x}) \tag{6.32}$$

Then, $E\left(\mathbf{x}, \theta\right) = exp(E^G(\mathbf{x}, \theta)\big)$ and can be represented using the PCE as:

$$E\left(\mathbf{x}, \theta\right) = exp(E^G(\mathbf{x}, \theta)\big) = \sum_{j=0}^{P-1} e_j(\mathbf{x}) \Psi_j(\theta) \tag{6.33}$$

The values of the coefficients $e_j(\mathbf{x})$ can be obtained using the orthogonality property of the Ψ_j variables, that is

$$e_j(\mathbf{x}) = \frac{\mathbb{E}\left[\Psi_j E\left(\mathbf{x}, \theta\right)\right]}{\mathbb{E}\left[\Psi_j^2\right]} \tag{6.34}$$

For the calculation of the numerator in the above equation, closed form relations can be found in the literature.

6.3 Additional literature

Chapter 7

Bayesian Inference

In the previous chapters we discussed about methods, non-intrusive (Monte Carlo simulation) and intrusive (SSFEM), capable of solving stochastic systems and quantifying the uncertainty on the system's response. In this chapter, we present a different class of stochastic problems. The aim now is to update our initial beliefs on the problem's stochastic parameters based on available experimental measurements of the system under investigation. The process for solving this type of problems is called **Bayesian (Parameter) Inference** or **Bayesian Updating**.

Bayesian inference is very useful for engineering applications because there many types of **modeling** and **parametric** uncertainty. For instance, material properties are difficult to determine to a precise level for some material such as concrete - parametric uncertainty. On the other hand, assuming an elastic-perfectly plastic behavior for the material is a modeling error that can be treated as modeling uncertainty. Bayesian updating addresses both these two types of problems:

1. **parametric identification** for a prescribed mathematical model with uncertain parameters, based on system response measurements
2. **model updating**, that is, selection of a class of parametric models based on system response measurements

The idea of Bayesian updating is similar to our thinking process. We have a perception of different people and matters based on our experience, i.e. data. When a new event happens, i.e. new data is obtained, it modifies our perception. In other words, our perception is not only determined by the latest piece of information but it also depends on the original perception. In Bayesian analysis, the original perception is regarded as the prior information and the new piece of information is utilized to update our perception or mathematical model.

7.1 Basic Concepts

Let us use A and B to denote two events. The **conditional probability** of event A given the occurrence of event B is given by

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{7.1}$$

if $P(B) > 0$, else $P(A|B)$ is meaningless. If the conditional probability $P(A|B)$ is large, there are three possibilities:

1. Event A is a consequence of event B, or opposite.
2. There is higher level event C that induces both events A and B. For example, in Athens, the probability of a heavily traffic hammed day, provided good ambient air quality is high. Obviously, there is no direct reason-consequence between these two events. Note that vehicle emission is not the main source for ambient pollutants in Athens, even though it affects the air quality at a street level. However, a good ambient air quality day and a heavily traffic jammed day are consequences of a heavily raining day as the precipitation washes out pollutants from the air, and a heavily raining day triggers a heavily traffic jammed day.
3. There are more than one higher level or intermediate events in the reason-consequence tree for events A and B. This is a combination/extension of cases 1 and 2.

The **law of total probability** states that if the sample space A is subdivided into N (finite or countably-infinite) mutually exclusive events $A_1, A_2, ..., A_N$, the the probability of another event B is given by:

$$P(B) = \sum_{i=1}^{N} P(B \cap A_i) = \sum_{i=1}^{N} P(B|A_i)P(A_i) \tag{7.2}$$

The continuous analogy of the law of total probability takes the form

$$P(B) = \int_{-\infty}^{\infty} P(B|X = x)p(x)dx \tag{7.3}$$

where $p(x)$ is the probability density function that describes the random variable X. Furthermore, it can also be applied to random variables Y and X as

$$p(y) = \int_{-\infty}^{\infty} p(y|x)p(x)dx \tag{7.4}$$

where $p(y|x)p(x) = p(x, y)$ is the joint pdf of X and Y. Therefore, $p(y)$ is simply the *marginal* pdf of Y.

We now proceed with **Bayes' theorem** is several forms.

- **Bayes' Theorem** for discrete events:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{7.5}$$

and if $A_1, ..., A_N$ is a partition of the event A into N mutually exclusive events then

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{i=1}^{N} P(A_i)P(B|A_i)}, \qquad for\ i = 1, ..., N \tag{7.6}$$

- **Bayes' Theorem** for Continuous-valued Parameters by discrete events:

$$p(\boldsymbol{x}|A) = \frac{P(A|\boldsymbol{x})p(\boldsymbol{x})}{P(A)} \tag{7.7}$$

  where $\boldsymbol{X} = (X_1, ..., X_N)$ are the continuous-valued uncertain paramaters, whose pdf we want to update and $\boldsymbol{x} = (x_1, ..., x_N)$ their values. This form of Bayes' theorem is applicable to the identification of continuous-valued uncertain parameters with observation of discrete events. We call $p(\boldsymbol{x}|A)$ the **updated pdf** or **posterior pdf** of the parameter vector $\boldsymbol{x}$, while $p(\boldsymbol{x})$ is the **prior pdf**.

- **Bayes' Theorem** for Discrete events by Continuous-valued Parameters:
  The reciprocal of the previous equation can be obtained easily for the updated probability of an event with measurement of continuous-valued variables:

$$P(A|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|A)P(A)}{p(\boldsymbol{x})} \tag{7.8}$$

  This form can be used to update the probability of events given the measurement of continuous-valued variables.

- **Bayes' theorem** between Continuous-valued parameters
  This is the most interesting case, because in most engineering applications, the updating concern is on continuous-valued uncertain parameters with measurements of continuous-valued variables. If we use $\boldsymbol{X}$ to represent the vector of of uncertain parameters for identification and the vector $\boldsymbol{\Psi}$ for the measurements of the system, then the Bayes' theorem gives the expression for the updated pdf of the parameter vector $\boldsymbol{X}$ as

$$p(\boldsymbol{x}|\boldsymbol{\psi}) = \frac{p(\boldsymbol{\psi}|\boldsymbol{x})p(\boldsymbol{x})}{p(\boldsymbol{\psi})} \tag{7.9}$$

7.2 Bayesian Inference

In science and engineering, there are various uncertain parameters necessary to be determined for modeling and other purposes. The Bayes' theorem offers the possibility of inferencing uncertain models/systems from their measurements. As we mentioned, there are two levels of system identification. The first is **parametric identification**, in which a class of mathematical models for a particular physical phenomenon or system is given with unknown parameters to be identified. The second level deals with the selection of a suitable class of mathematical models for parametric identification. This is significantly more difficult but more important than the first level since parametric identification results will not be meaningful unless we have already chosen a suitable class of models. In this course, our focus will be on parametric identification.

Let us use $\mathcal{D}$ to denote the measured data of a system and consider it as the vector $\boldsymbol{\psi}$ in eq. (7.9). Then, the updated/posterior probability density function of the parameters $\boldsymbol{X} = (X_1, ..., X_N)$ is:

$$p(\boldsymbol{x}|\mathscr{D}, \mathscr{C}) = \frac{p(\mathscr{D}|\boldsymbol{x}, \mathscr{C})p(\boldsymbol{x}|\mathscr{C})}{\int_{\boldsymbol{\Theta}} p(\mathscr{D}|\boldsymbol{x}, \mathscr{C})p(\boldsymbol{x}|\mathscr{C})d\boldsymbol{x}} \tag{7.10}$$

where $\mathscr{C}$ denotes the class of probabilistic and physical models/rules used for the problem of concern. Since our focus is on parametric identification we could completely omit it from the above equation and write it more simply as:

$$p(\boldsymbol{x}|\mathscr{D}) = \frac{p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x})}{\int_{\boldsymbol{\Theta}} p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}} \tag{7.11}$$

The denominator in the above equation is a constant $k_0 = \int_{\boldsymbol{\Theta}} p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$, which makes sure that $p(\boldsymbol{x}|\mathscr{D})$ integrates to 1, thus being a pdf. The integration in the denominator is on the entirety of the parametric space $\boldsymbol{\Theta}$, where $\boldsymbol{x}$ is supported. The density function $p(\mathscr{D}|\boldsymbol{x})$ is called **likelihood function** and it reflects how likely it is to observe the measurements from the particular set of parameters. The key of Bayesian Updating is on this likelihood function. Also, it should be mentioned that the prior distribution $p(\boldsymbol{x})$ is commonly based on previous knowledge or user's judgment. Other choice's for priors include *improper priors* or *conjugate prior distributions*, but these are not necessary in practice.

Example: Updating of a single random variable

Let X be a random variable with standard normal prior, i.e. $p(x) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{x^2}{2}\right)$. A measurement $\mathscr{D} := d = 2$ of X is made, resulting in a value of 2. The measurement d is associated with an additive error ϵ, which is normal distributed with zero mean and standard deviation $\sigma_\epsilon = 0.5$. Let p_ϵ be the $\mathscr{N}(0, 0.5)$ normal pdf of ϵ. This means that $X - d = X - 2 = \epsilon$ and the likelihood function is given by the equation:

$$p(d|x) = p_\epsilon(x-2) = \frac{1}{0.5\sqrt{2\pi}} exp\left(-\frac{(x-2)^2}{2 \cdot 0.5^2}\right) \tag{7.12}$$

Now, it is straightforward to find the posterior $p(x|\mathscr{D})$ using analytical (or numerical) integration on eq. (7.11).

The results are shown in figure 7.1. The posterior is a normal distribution with mean 1.6 and standard deviation 0.45.

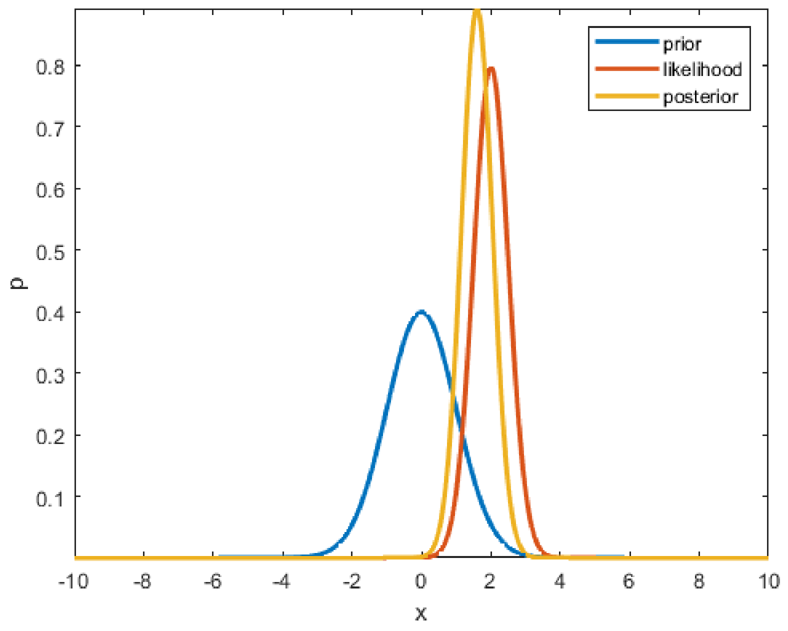


Figure 7.1: Baysian update of an RV

7.3 Bayesian Inference for mechanical models

In this section we will discuss, how the Bayesian framework can be applied to mechanical models with uncertain parameters. Let us denote $\mathscr{M}$ the mechanical model that has some uncertain parameters given by the random vector $\boldsymbol{X}$, such that $\mathscr{M} := \mathscr{M}(\boldsymbol{X})$. We use $\mathscr{D} = \{d_1, ..., d_N\}$ to denote the available N measurements we have about the model's response. As an example, we can consider $\mathscr{D}$ to consist of measurements of deformation for given values of $\boldsymbol{X}$. Let $\mathscr{M}_i(\boldsymbol{X})$ denote the model prediction corresponding to measurement d_i.

We assume that the discrepancy/error between the observed deformation d_i and the model prediction $\mathscr{M}_i(\boldsymbol{X})$ is a random variable ϵ_i, described by a pdf $p_{\epsilon_i}(\cdot)$. The following relation holds: $d_i - \mathscr{M}_i(\boldsymbol{X}) = \epsilon_i$. Therefore, the likelihood function describing the observation becomes

$$p_i(d_i|\boldsymbol{x}) = p_{\epsilon_i}(d_i - \mathscr{M}_i(\boldsymbol{x})) \tag{7.13}$$

Most commonly, ϵ_i is assumed to follow a zero-mean Gaussian distribution with a prescribed standard deviation.

As a side note, we could also consider multiplicative errors of the form $d_i = \mathscr{M}_i(\boldsymbol{X}) \cdot \epsilon_i$, in which case the likelihood functions would be

$$p_i(d_i|\boldsymbol{x}) = p_{\epsilon_i}\left(\frac{d_i}{\mathscr{M}_i(\boldsymbol{x})}\right) \tag{7.14}$$

In the general case, where $\mathscr{D}$ consists of N observations $d_1, ..., d_N$, the likelihood functions becomes

$$p(\mathscr{D}|\boldsymbol{x}) = \prod_{i=1}^{N} p_i(d_i|\boldsymbol{x}) \tag{7.15}$$

The above equation is valid if we assume that the errors are statistically independent, which is most often the case. Otherwise, the combined likelihood must be formulated as a function of the joint pdf of all ϵ_i.

Returning to eq. (7.11) which we need in order to evaluate the posterior pdf of the parameters, we notice the main difficulty in this setting. In particular, we don't have an analytic relation for the model's output $\mathscr{M}_i(\boldsymbol{x})$, $i = 1, ...N$. This fact suggests that we cannot perform the integration of eq. (7.11) (analytically of numerically) and we need another approach to evaluate the posterior.

7.4 Sampling Algorithms

In this section, we will discuss another approach for obtaining the posterior pdf of the parameters, in cases where a direct evaluation of eq. (7.11) is not feasible. We will present two sampling algorithms, namely the Acceptance-Rejection sampling algorithm and the Metropolis-Hastings algorithm, which have the property of collecting samples directly from the posterior distribution. Then, given a large number of samples, we can produce the (empirical) posterior pdf.

7.4.1 Acceptance-Rejection sampling

By looking eq. (7.11), we notice that we know the posterior distribution up to a proportionality constant $\frac{1}{k_0}$,

$$\begin{aligned} p(\boldsymbol{x}|\mathscr{D}) &= \frac{p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x})}{\int_{\boldsymbol{\Theta}} p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}} = \frac{1}{k_0} p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x}) \Rightarrow \\ p(\boldsymbol{x}|\mathscr{D}) &\propto p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x}) = p^{uns}(\boldsymbol{x}|\mathscr{D}) \end{aligned} \tag{7.16}$$

Acceptance-rejection-sampling (ARS) is a method for drawing random samples directly from a **target distribution** even though we only know the **unscaled target**. In Bayesian statistics, the **posterior distribution** is the target and all we know is the unscaled target which is **the prior times the likelihood**. This gives the shape of the target, but not the scale factor needed to make it an exact density. The ARS algorithm allows us to draw a random sample from the target distribution, when we only know its shape.

The ARS agorithm works by reshaping a random sample drawn from an easily sampled **candidate distribution** (aka **proposal** or **starting distribution**) into a random sample from the target by only accepting some of the candidate values into the final sample. Let the unscaled posterior $p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x})$ be the unscaled target, and let the easily sampled candidate density be $g_0(\boldsymbol{x})$. It is of significant importance that the candidate density **dominates** the unscaled target. That means that we can find a number M such that

$$Mg_0(\boldsymbol{x}) \geq p(\mathscr{D}|\boldsymbol{x})p(\boldsymbol{x}) \tag{7.17}$$

for all values of $\boldsymbol{x}$. ARS works by only accepting some of the candidates into the final sample. We can think of this as cutting out part of the scaled up candidate density in order to give it the shape of the target density. We must start with the scaled up candidate density being greater than the target for all values of $\boldsymbol{x}$. We can cut part of the candidate density out, but we can't put any part back in. The ARS algorithm proceeds as follows:

1. Draw a random value of $\boldsymbol{x}$ from the candidate density $g_0(\boldsymbol{x})$.
2. Calculate the weight for that $\boldsymbol{x}$ value as the ratio of the target to the scaled up candidate density

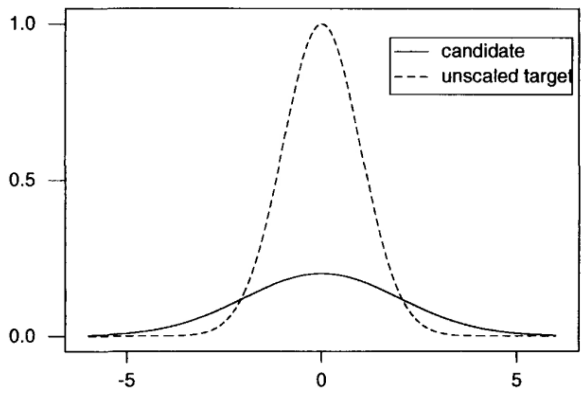
$$w(\boldsymbol{x}) = \frac{p(\mathcal{D}|\boldsymbol{x})p(\boldsymbol{x})}{Mg_0(\boldsymbol{x})} \tag{7.18}$$

   for that value. The weight will always be between 0 and 1.
3. Draw (independently) u from the uniform (0,1) distribution.
4. If $u < w(\boldsymbol{x})$ accept $\boldsymbol{x}$.
5. Otherwise reject $\boldsymbol{x}$ and return to step 1.

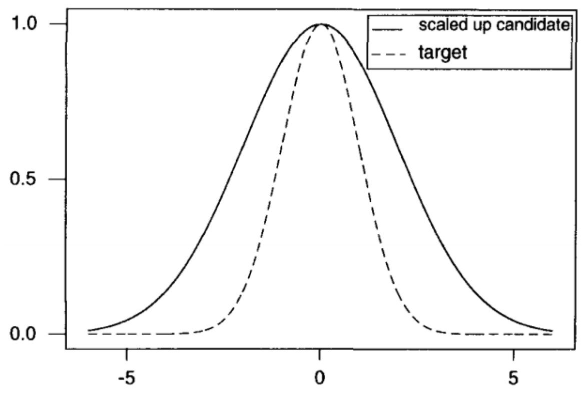
An accepted $\boldsymbol{x}$ is a random draw from the exact posterior distribution $p(\boldsymbol{x}|\mathcal{D})$, even though we only knew the unscaled version of the posterior. By repeating the steps of the ARS algorithm we can produce a large number of samples from the posterior and thus approximate the distribution.

As an exercise, you can test the ARS algorithm in the example of the previous section and see how close the results are to the analytical solution. (Hint: You can use the prior as the candidate density and $M = max(p(\mathcal{D}|\boldsymbol{x}))$.

Lastly, we close this section with some important things about the candidate function. The first thing to consider is that the ARS algorithm does not work if M is not finite. This may happen if the candidate distribution has lighter tails than the target. From the graph of the unscaled target and the candidate, we cannot tell which one has heavier tails since both are going to zero. To resolve this, we take logarithms of both the candidate density and the target and graph them in the same figure. Usually, heavy-tailed candidate distributions such as *Student's t* with low degrees of freedom are recommended as candidate distributions. An easier alternative (but not as efficient) would be to consider the prior as the candidate density.

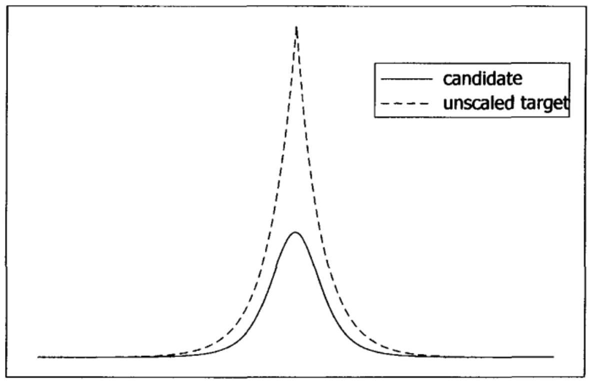


(a) Unscaled candidate density and unscaled target

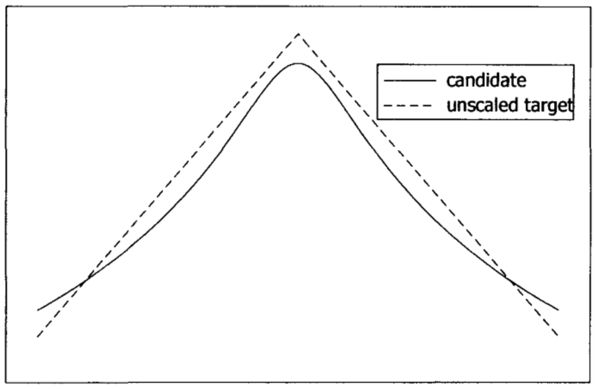


(b) Scaled up candidate density and unscaled target

Figure 7.2: Scaling the candidate density



(a) Unscaled candidate density and unscaled target



(b) Logarithms of unscaled candidate density and unscaled target

Figure 7.3: Scaling the candidate density

The main drawback of ARS is that it can lead to a lot of samples being rejected in cases where we have a high-dimensional likelihood function (many measurements) or the unscaled target is concentrated in a certain region (like a spike). For this type of problems, a different approach is employed, namely Markov Chain Monte Carlo methods such as the Metropolis-Hastings algorithm that we will present in the next section.

7.4.2 Markov Chain Monte Carlo

To overcome the drawbacks of ARS algorithms and efficiently draw samples from the posterior distribution, the Markov Chain Monte Carlo (MCMC) technique can be used. The MCMC approach is simply the Monte Carlo approach applied to Markov processes - namely, it is sampling from a distribution defined via a stochastic process known as a Markov Process. To better illustrate the difference between MC and MCMC let us revisit the use of MC to evaluate integrals of the form $\int g(x)f(x)dx$ with $f(x)$ being a probability distribution. Then, we draw N samples $\{x_1, ..., x_N\}$ from the distribution of $f(x)$ and the integral is approximately equal to $\sum_{i=1^N} g(x_i)/N$. In essence, we consider N **independent** random variables $\{X_1, ..., X_N\}$, from which we draw our samples. But these random variables need not be independent in order to accurately approximate integrals. MCMC constructs a **dependent** sequence of random variables that can be used to approximate the integrals.

To initiate the discussion, we recall the definition of a Markov chain which is a sequence of random variables $\{X_1, X_2, ...\}$ which satisfies

$$P(X_{n+1} \in B | X_1, ..., X_n) = P(X_{n+1} \in B | X_n) \tag{7.19}$$

i.e. the future, given the past and present, depends only on the present. From eq. (7.19) we can argue that the probabilistic properties of the chain are completely determined by (i) the initial distribution for X_0 and (ii) the **transition distribution** of X_{n+1}, given X_n.

Example 1

Let $U_1, U_2, ...$ be independent, identically distributed random variables following the $\mathcal{U}[-1, 1]$ distribution. Set $X_0 = 0$ and let $X_n = \sum_{i=1}^{n} U_i = X_{n-1} + U_n$. The initial distribution is $P(X_0 = 0) = 1$. The transition distribution is determined by

$$X_n = \begin{cases} X_{n-1} - 1 & \text{with probability } 1/2 \\ X_{n-1} + 1 & \text{with probability } 1/2 \end{cases} \tag{7.20}$$

The random walk is a simple example of a Markov process but it is a very important one!

Example 2

Consider modeling the weather as a Markov process. Suppose the states $S = \{sunny, cloudy, rainy\}$ and the transitions defined by the **transition matrix**:

$$P = \begin{bmatrix} 0.7 & 0.2 & 0.1 \\ 0.6 & 0.2 & 0.2 \\ 0.5 & 0.2 & 0.3 \end{bmatrix} \tag{7.21}$$

If today ($n = 0$) is a sunny day with probability 1, then our initial distribution (initial state) is $p_0 = \{1, 0, 0\}$. The weather prediction for tomorrow is given by

$$p_1 = p_0 P = \{0.7, 0.2, 0.1\} \tag{7.22}$$

and in n days from now

$$p_n = p_{n-1} P = p_0 P^n \tag{7.23}$$

This was an example of a **homogeneous** Markov Chain, because the transition matrix P does not vary with respect to n. In contrast to example 1, though, in this case we had a **finite-state** Markov chain.

Some terminology:

- A state A is **recurrent** if a chain starting in A will eventually return to A with probability 1. This state is **non-null** if the expected time to return is finite. A chain is called **recurrent** if each state is recurrent.

- A Markov chain is **irreducible** if there is a positive probability that a chain starting in a state A can reach any other state B.
- A Markov chain is **aperiodic** if, for a starting state A, there is no constraint on the times at which the chain can return to A.
- An irreducible, aperiodic Markov chain with all states being non-null recurrent is called **ergodic**.

Some important facts from Limit theory:

- A distribution f is a **stationary** distribution of the chain if $X_0 \sim f$ implies $X_n \sim f$ for all n.
- An ergodic Markov chain has at most one stationary distribution.
- If the chain is ergodic, then

$$\lim_{n\to\infty} P(X_{m+n} \in B | X_m \in A) = \int_B f(x)dx \tag{7.24}$$

  for all A, B, m

- More importantly, if $\phi(x)$ is an integrable function, then

$$\frac{1}{n}\sum_{t=1}^{n} \phi(x_t) \to \int \phi(x)f(x)dx, \text{ with probability 1} \tag{7.25}$$

  This is a version of the famous **ergodic theorem**.

The question arising at this point is why use MCMC instead of MC. In certain applications, such us in Bayesian inference, we want to generate samples from the target distribution $p(x|\mathscr{D})$, or equivalently, we want to generate random variables with distribution $p(x|\mathscr{D})$. But this is not possible to do exactly. Instead, MCMC is designed to construct an ergodic Markov chain with $p(x|\mathscr{D})$ as its stationary distribution. Asymptotically, the chain will resemble samples from $p(x|\mathscr{D})$. In particular, by the ergodic theorem, expectations with respect to $p(x|\mathscr{D})$ can be approximated by averages.

A vary famous algorithm for constructing and simulating a suitable Markov chain having $p(x|\mathscr{D})$ as its stationary distribution, is the Metropolis-Hastings (MH) algorithm. The implementation steps of the MH algorithm are summarized below:

1. Start with an initial value of parameters $\boldsymbol{x}_0$
2. Select an arbitrary candidate density function $g(\cdot|\cdot)$. The purpose of this function is to propose the next candidate sample at each MCMC step. In this notation, the first placeholder refers to the variables of the density function and the second to its parameters. A typical choice is a Gaussian distribution with the previously accepted sample as mean value and a pre-selected standard deviation.

3. Select a burn-in period N_{burn}. During this period the samples acquired from the MH algorithm are discarded. This is important in order to ensure that the Markov Chain has converged to its stationary distribution, meaning that the samples are drawn from the posterior distribution.

4. Set i=1

5. While $i \leq N_{burn} + N_{samples}$ ($N_{samples}$ being the required number of samples)

   (a) Generate a candidate sample $\boldsymbol{x}'$ from $g(\boldsymbol{x}'|\boldsymbol{x}_{i-1})$

   (b) Evaluate the model response $\boldsymbol{M}(\boldsymbol{x}')$ through a model evaluation and calculate the likelihood $p(\mathcal{D}|\boldsymbol{x}')$

   (c) Calculate the acceptance probability $\alpha(\boldsymbol{x}_{i-1}, \boldsymbol{x}') = min\left[1, \frac{p^{uns}(\boldsymbol{x}'|\mathcal{D})g(\boldsymbol{x}'|\boldsymbol{x}_{i-1})}{p^{uns}(\boldsymbol{x}_{i-1}|\mathcal{D})g(\boldsymbol{x}_{i-1}|\boldsymbol{x}')}\right]$

   (d) Draw v from the uniform distribution $\mathcal{U}(0, 1)$

   (e) If $v < \alpha(\boldsymbol{x}_{i-1}, \boldsymbol{x}')$ then let $\boldsymbol{x}_i = \boldsymbol{x}'$ and set $i = i + 1$, else let $\boldsymbol{x}_i = \boldsymbol{x}_{i-1}$

6. Discard the first N_{burn} samples.

The most important issue that needs to be addressed here is the selection of the proposal distribution. This choice is not easy and the performance of the algorithm depends upon it. In the algorithm presented above we considered a classical strategy for $g(\boldsymbol{x}|\boldsymbol{y})$, which is to consider a symmetric distribution (e.g. Gaussian), centered at the previous accepted sample $\boldsymbol{y}$ and a prescribed variance.

It can be proven, that the sequence of random variables $X_n,\ n = 1, 2, ...$, or, in other words, the stochastic process $\{X_n : n \geq 1\}$ is an ergodic Markov process that has $p(\boldsymbol{x}|\mathcal{D})$ as its stationary distribution. So, provided that we run the simulation "long enough", we should be able the get arbitrarily good approximations.

Several more sophisticated variations of the classic MH algorithm can be found in the literature, such as the adaptive MH, the transitional MCMC, Hamiltonian MCMC which are more efficient.

7.5 Additional literature

- Understanding Computational Bayesian Statistics, W.M.Bolstad, Wiley
- Bayesian Method for Structural Dynamics and Civil Engineering, K-V. Yuen, Wiley
- Monte Carlo Statistical Methods, C. Robert and G. Casella, Springer
- Bayesian Data Analysis, 3rd Edition, A. Gelman et al., CRC Press

Chapter 8

Sensitivity Analysis

8.1 Additional literature

Appendix A

Application of the finite element method in 3D elasticity

A.1 Introduction

Without a doubt the development of accurate mathematical models is essential for the study of physical systems. Under appropriate assumptions concerning the underlying physical mechanisms and the laws that govern them, these models are most often described by a differential equation satisfying specific boundary conditions, also referred to as a *boundary value problem* (BVP). However, in most practical problems, where complicated geometries, loading conditions, material properties and nonlinearities are involved, solving analytically the corresponding BVPs is often impossible. Therefore, the only alternative is to find approximate solutions using numerical methods. In this regard, the *finite element method* is the most powerful computational technique used to obtain numerical solutions of BVPs in engineering.

The aim of this chapter is to revisit the basic principles of the finite element method. Starting from the abstract formulation of the BVP for the case of 3d elasticity, the variational BVP is obtained. Then, the finite element approximation technique is used, in order to convert the differential equations describing the problem into a system of linear equations.

A.2 The abstract problem

Most problems in the theory of elasticity can be written in the following abstract manner:

Let $\mathscr{V}$ be a normed linear space over $\mathbb{R}$. Let $\mathscr{J} : \mathscr{V} \to \mathbb{R}$ be a functional which can be written in the form:

$$\mathscr{J}(\boldsymbol{v}) = \frac{1}{2}a(\boldsymbol{v}, \boldsymbol{v}) - b(\boldsymbol{v}), \quad \forall \boldsymbol{v} \in \mathscr{V} \tag{A.1}$$

where $a(\cdot,\cdot)$ is a continuous, symmetric bilinear form on $\mathscr{V}$ and b is an element of $\mathscr{V}^\star$, which is the dual space of $\mathscr{V}$. Typically, $\mathscr{J}$ represents the energy of some system under consideration. Then the problem consists in finding an element $\boldsymbol{u} \in \mathscr{V}$ such that

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u} \in \mathscr{V}} \mathscr{J}(\boldsymbol{u}) \tag{A.2}$$

Often, instead of minimizing $\mathscr{J}$ over the entire space $\mathscr{V}$, a non-empty convex subset $\mathscr{K}$ of $\mathscr{V}$ is preferred, over which the minimum of $\mathscr{J}$ is sought, that is

$$\boldsymbol{u} = \arg\min_{\boldsymbol{u}\in\mathscr{K}} \mathscr{J}(\boldsymbol{u}) \tag{A.3}$$

If we denote the above problem as (P) then the next step is to examine the existence of a solution to (P) and, if such a solution does exist, its uniqueness.

Let $\mathscr{V}$ be a normed linear space. A bilinear form $a(\cdot,\cdot)$ on $\mathscr{V}$ is said to be $\mathscr{V}$-elliptic if there exists a constant $c > 0$ such that $\forall \boldsymbol{v} \in \mathscr{V}$:

$$a(\boldsymbol{v},\boldsymbol{v}) \geq c\|\boldsymbol{v}\|^2 \tag{A.4}$$

The $\mathscr{V}$-ellipticity of $a(\cdot,\cdot)$ suggests that if $a(0,0) = 0$, then $\boldsymbol{v} = 0$. Also, since $a(\cdot,\cdot)$ is bilinear and symmetric, then it defines an inner-product structure on $\mathscr{V}$. Thus, $\mathscr{V}$ acquires the structure of a Hilbert space. The existence of a unique solution to (P) is given by the following theorem:

Theorem 1. *Let $\mathscr{V}$ be a Banach space and $\mathscr{K}$ a closed convex subset of $\mathscr{V}$. Let $a(\cdot,\cdot)$ be a symmetric, $\mathscr{V}$-elliptic bilinear form. Then, there exists a unique solution for the problem (P).*

The following corollary stems from this theorem: If $\mathscr{K}$ is a subspace of $\mathscr{V}$ then the solution $\boldsymbol{u}$ is characterized by

$$a(\boldsymbol{u},\boldsymbol{v}) = b(\boldsymbol{v}), \quad \forall \boldsymbol{v} \in \mathscr{K} \tag{A.5}$$

In order to prove the existence and uniqueness of the solution in theorem 1, the symmetry of the bilinear form was assumed, which provided the Hilbert space structure. If this condition is relaxed but instead $\mathscr{V}$ is assumed to be a Hilbert space and $\mathscr{V}$ is taken equal to $\mathscr{K}$, then the existence and uniqueness of the solution is given by the following theorem, known as the *Lax-Milgram Lemma*:

Theorem 2. *Let $\mathscr{V}$ be a Hilbert space, $a(\cdot,\cdot)$ be a continuous, $\mathscr{V}$-elliptic bilinear form and $b \in V^\star$. If (P) is the problem: find $\boldsymbol{u} \in \mathscr{V}$ such that $\forall \boldsymbol{v} \in \mathscr{V}$,*

$$a(\boldsymbol{u},\boldsymbol{v}) = b(\boldsymbol{v}) \tag{A.6}$$

then (P) has a unique solution.

A.3 The boundary value problem

Let us denote with $\Omega \subset \mathbb{R}^n$ an open bounded subset and $\partial\Omega$ its boundary. Let us restrict our attention to the case $\Omega \subset \mathbb{R}^3$ and its boundary $\partial\Omega$ can be partitioned into two parts $\partial\Omega_D$ and $\partial\Omega_N$. According to the theory of elasticity, the equilibrium equation of a linear isotropic body occupying Ω which is subjected to body forces $\boldsymbol{f}$ in Ω and surface tractions $\bar{\boldsymbol{t}}$ in the boundary $\partial\Omega_N$ is:

$$\nabla \cdot \boldsymbol{\sigma} + \boldsymbol{f} = 0 \ \text{ in } \Omega \tag{A.7}$$

where $\boldsymbol{\sigma}$ is the stress tensor and $\nabla \cdot \boldsymbol{\sigma}$ its divergence. Next, the strain tensor is defined as:

$$\boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T\right) \tag{A.8}$$

with $\boldsymbol{u} : \Omega \to \mathbb{R}^3$ being the vector field of displacements, that will be developed in the body under the prescribed loading conditions. The following relations hold between $\boldsymbol{\sigma}$, $\boldsymbol{\epsilon}$ and $\boldsymbol{u}$:

$$\begin{cases} \epsilon_{ij}(\boldsymbol{u}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \\ \sigma_{ij}(\boldsymbol{u}) = \lambda\left(\sum_{k=1}^{3} \epsilon_{kk}(\boldsymbol{u})\right)\delta_{ij} + 2\mu\epsilon_{ij}(\boldsymbol{u}) \end{cases} \tag{A.9}$$

for $1 \leq i, j \leq 3$, or, equivalently,

$$\begin{aligned} \boldsymbol{\sigma} &= \lambda Tr(\boldsymbol{\epsilon})\mathbb{I}_{3\times3} + 2\mu\boldsymbol{\epsilon} \\ &= \lambda(\nabla \cdot \boldsymbol{u})\mathbb{I}_{3\times3} + 2\mu\boldsymbol{\epsilon} \end{aligned} \tag{A.10}$$

The latter equation is the well-known *Hooke's law* in 3D and λ, μ are positive constants, called the *Lame's coefficients*, given by:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{A.11}$$

$$\mu = \frac{E}{2(1+\nu)} \tag{A.12}$$

where E and ν are the modulus of elasticity and the Poisson coefficient, respectively. Eq. (A.10) can also be recast into the more familiar matrix form:

$$\{\boldsymbol{\sigma}\} = \boldsymbol{D}\{\boldsymbol{\epsilon}\} \tag{A.13}$$

with,

$$\{\boldsymbol{\sigma}\} = \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{31} \end{bmatrix}, \ \{\boldsymbol{\epsilon}\} = \begin{bmatrix} \epsilon_{11} \\ \epsilon_{22} \\ \epsilon_{33} \\ \epsilon_{12} \\ \epsilon_{23} \\ \epsilon_{31} \end{bmatrix} \tag{A.14}$$

and

$$\boldsymbol{D} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \tag{A.15}$$

the 3D elasticity matrix.

Proceeding with the previous formulation, inserting eq. (A.10) to eq. (A.7) gives

$$-\lambda\nabla \cdot ((\nabla \cdot \boldsymbol{u})\,\mathbb{I}_{3\times3}) - 2\mu\nabla \cdot \boldsymbol{\epsilon}(\boldsymbol{u}) = \boldsymbol{f} \tag{A.16}$$

and after some mathematical operations eq. (A.16) becomes

$$-(\lambda+\mu)\nabla(\nabla\cdot\boldsymbol{u})-\mu\Delta\boldsymbol{u}=\boldsymbol{f} \tag{A.17}$$

where $\Delta\boldsymbol{u}=\nabla(\nabla\cdot\boldsymbol{u})-\nabla\times(\nabla\times\boldsymbol{u})$ the Laplacian operator acting on $\boldsymbol{u}$. This equation is known as the *Navier-Cauchy* equation and the boundary value problem of eq. (A.7), that is

$$\begin{cases}-\sum_{j=1}^{3}\frac{\partial}{\partial x_j}\left(\sigma_{ij}(\boldsymbol{u})\right)=f_i, \text{ in } \Omega \text{ for } i=1,2,3\\ \boldsymbol{u}=\bar{\boldsymbol{u}} \text{ on } \partial\Omega_D\\ \boldsymbol{\sigma}\mathbf{v}=\bar{\boldsymbol{t}} \text{ on } \partial\Omega_N\end{cases} \tag{A.18}$$

becomes

$$\begin{cases}-(\lambda+\mu)\nabla(\nabla\cdot\boldsymbol{u})-\mu\Delta\boldsymbol{u}=\boldsymbol{f} \text{ in } \Omega\\ \boldsymbol{u}=\bar{\boldsymbol{u}} \text{ on } \partial\Omega_D\\ \boldsymbol{\sigma}\mathbf{v}=\bar{\boldsymbol{t}} \text{ on } \partial\Omega_N\end{cases} \tag{A.19}$$

with $\mathbf{v}$ denoting the outward pointing unit normal vector. Equations (A.19) constitute the *system of linear elasticity.*

A.4 The variational boundary value problem

The system in eq. (A.19) is referred to as the *strong form* of the BVP and it requires the solution vector field to be at least two times continuously differentiable, that is, $\boldsymbol{u}\in C^2(\Omega)$. It turns out that this condition can be relaxed using the *weak*, or *variational formulation* of the BVP. This approach offers two advantages; the solution space is enlarged and the Neumann boundary conditions can be straightforwardly incorporated in the differential equation. In order to derive the variational BVP of eq. (A.19) some definitions have to be provided first.

The Sobolev space $\mathscr{W}^{m,p}(\Omega)$, with $m\in\mathbb{Z}^{\geq 0}$ and p and integer greater than 1, is given by:

$$\mathscr{W}^{m,p}(\Omega)=\{v\in\mathscr{L}^p(\Omega)|D^\alpha v\in\mathscr{L}^p(\Omega),\ \forall|\alpha|\leq m\} \tag{A.20}$$

where α is a multi-index denoting the n-tuple $(\alpha_1,\alpha_2,...,\alpha_n)$, $\alpha_i\in\mathbb{Z}^{\geq 0}$ and $D^\alpha=\frac{\partial^{|\alpha|}}{\partial x_1^{\alpha_1}\cdots\partial x_n^{\alpha_n}}$ are the partial derivatives of order $|\alpha|$, which are understood in the weak sense. The Sobolev spaces with $p=2$ are particularly important because they also form a Hilbert space $\mathscr{H}^m=\mathscr{W}^{m,2}$.

Consider the vector spaces

$$\begin{aligned}U&=\{\boldsymbol{u}\in\mathscr{W}^{1,2}:\boldsymbol{u}=\bar{\boldsymbol{u}} \text{ on } \partial\Omega_D\}\\ W&=\{\boldsymbol{w}\in\mathscr{W}^{1,2}:\boldsymbol{w}=0 \text{ on } \partial\Omega_D\}\end{aligned} \tag{A.21}$$

with U being the space of admissible functions and W the space of test functions. Multiplying both sides of the equilibrium equation (A.7) with $\boldsymbol{w}\in W$ and integrating over Ω gives:

$$-\int_\Omega(\nabla\cdot\boldsymbol{\sigma})\cdot\boldsymbol{w}d\boldsymbol{x}=\int_\Omega\boldsymbol{f}\cdot\boldsymbol{w}d\boldsymbol{x} \tag{A.22}$$

The following integral formula holds for the left hand side of eq. (A.22)

$$\begin{aligned}\int_\Omega (\nabla \cdot \boldsymbol{\sigma}) \cdot \boldsymbol{w} d\boldsymbol{x} &= -\int_\Omega \boldsymbol{\sigma} : \nabla \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \cdot (\boldsymbol{\sigma}\mathbf{v}) d\boldsymbol{s} + \int_{\partial\Omega_D} \boldsymbol{w} \cdot (\boldsymbol{\sigma}\mathbf{v}) d\boldsymbol{s} \\ &= -\int_\Omega \boldsymbol{\sigma} : \nabla \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \cdot \bar{\boldsymbol{t}} d\boldsymbol{s} + \cancelto{0}{\int_{\partial\Omega_D} \boldsymbol{w} \cdot (\boldsymbol{\sigma}\mathbf{v}) d\boldsymbol{s}} \end{aligned} \tag{A.23}$$

where : denotes the tensor inner product and $d\boldsymbol{s}$ is the differential volume element on $\partial\Omega$. Also, the last term in eq. (A.23) is zero due to the compact support of $\boldsymbol{w}$. Therefore, eq. (A.22) can be rewritten as

$$\int_\Omega \boldsymbol{\sigma} : \nabla \boldsymbol{w} d\boldsymbol{x} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \cdot \bar{\boldsymbol{t}} d\boldsymbol{s} \tag{A.24}$$

or, due to the symmetry of $\boldsymbol{\epsilon}$

$$\int_\Omega \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\boldsymbol{w}) d\boldsymbol{x} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \cdot \bar{\boldsymbol{t}} d\boldsymbol{s} \tag{A.25}$$

Applying eq. (A.10) to (A.25) we have:

$$\int_\Omega \Big(\lambda \left(\nabla \cdot \boldsymbol{u}\right) \mathbb{I}_{3\times 3} + 2\mu \boldsymbol{\epsilon}\left(\boldsymbol{u}\right)\Big) : \boldsymbol{\epsilon}(\boldsymbol{w}) d\boldsymbol{x} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \cdot \bar{\boldsymbol{t}} d\boldsymbol{s} \tag{A.26}$$

which is equivalent to

$$\int_\Omega \lambda \left(\nabla \cdot \boldsymbol{u}\right) \cdot \left(\nabla \cdot \boldsymbol{w}\right) + 2\mu \boldsymbol{\epsilon}\left(\boldsymbol{u}\right) : \boldsymbol{\epsilon}(\boldsymbol{w}) d\boldsymbol{x} = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \cdot \bar{\boldsymbol{t}} d\boldsymbol{s} \tag{A.27}$$

Next, we define the bilinear form $a(\cdot, \cdot)$ by,

$$\begin{aligned} a(\boldsymbol{u}, \boldsymbol{w}) &= \int_\Omega \Big(\lambda \left(\nabla \cdot \boldsymbol{u}\right) \cdot \left(\nabla \cdot \boldsymbol{w}\right) + 2\mu \boldsymbol{\epsilon}\left(\boldsymbol{u}\right) : \boldsymbol{\epsilon}(\boldsymbol{w})\Big) d\boldsymbol{x} \\ &= \int_\Omega \left(\lambda \left(\nabla \cdot \boldsymbol{u}\right) \cdot \left(\nabla \cdot \boldsymbol{w}\right) + 2\mu \sum_{i,j=1}^{3} \epsilon_{ij}(\boldsymbol{u}) \epsilon_{ij}(\boldsymbol{w})\right) d\boldsymbol{x} \\ &= \int_\Omega \sum_{i,j=1}^{3} \sigma_{ij}(\boldsymbol{u}) \epsilon_{ij}(\boldsymbol{w}) d\boldsymbol{x} \end{aligned} \tag{A.28}$$

and the linear functional $b(\cdot)$ by:

$$b(\boldsymbol{w}) = \int_\Omega \boldsymbol{f} \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N} \boldsymbol{w} \bar{\boldsymbol{t}} d\boldsymbol{s} \tag{A.29}$$

Using the notation of operation theory, eq. (A.27) can be expressed as:

$$a(\boldsymbol{u}, \boldsymbol{w}) = b(\boldsymbol{w}) \tag{A.30}$$

Equations (A.27) and (A.30) are known as the *variational boundary value problem* or the weak formulation. It can be proven that a and b satisfy the conditions for Theorem 2 to hold, and, thus, the problem (P), i.e. $a(\boldsymbol{u}, \boldsymbol{w}) = b(\boldsymbol{w})$ admits a unique solution.

In this context, the request for the relation $a(\boldsymbol{u}, \boldsymbol{w}) = b(\boldsymbol{w})$ to be satisfied for all $\boldsymbol{w} \in \mathscr{W}$ can be interpreted via the *principle of virtual work* with $\frac{1}{2}a(\boldsymbol{u}, \boldsymbol{u})$ being the *strain energy* and $b(\boldsymbol{u})$ the *potential energy of exterior forces.* Generally, these equations cannot be solved analytically and the finite element method constitutes a generic numerical approach to approximate the solutions of (A.17) and its generalizations (nonlinear elasticity, dynamic, etc.).

A.5 The finite element method

In practice, obtaining an exact solution $\boldsymbol{u}$ of eq. (A.25) is not feasible in most applications of interest. Instead, an approximate solution is sought and the Galerkin approximation technique can provide it. The main idea behind this approach is to consider a finite-dimensional subspace $U_h \subseteq U$, spanned by a finite number of basis vectors $\{\boldsymbol{N}_i\}_{i=1}^{\bar{N}}$, and to find the best approximation $\boldsymbol{u}_h \in U_h$ of $\boldsymbol{u} \in U$. Since, $U_h \subseteq U$ and $span\{\boldsymbol{N}_1, ..., \boldsymbol{N}_{\bar{N}}\} = U_h$, then $\boldsymbol{u}_h$ will be a linear combination of the basis vectors, that is $\boldsymbol{u}_h = v_1\boldsymbol{N}_1(\boldsymbol{x}) + v_2\boldsymbol{N}_2(\boldsymbol{x}) + ... + v_{\bar{N}}\boldsymbol{N}_{\bar{N}}(\boldsymbol{x})$. Then, then problem consists in finding these coefficients $\boldsymbol{v} = (v_1, v_2, ..., v_{\bar{N}})^T$ of the linear combination that minimize the error $\|\boldsymbol{u} - \boldsymbol{u}_h\|$.

The Galerkin approximation considers $\boldsymbol{w} = \boldsymbol{N}_i$, $i = 1, ..., \bar{N}$ as test functions and by substituting $\boldsymbol{w}$ in eq. (A.30) one obtains:

$$a(\boldsymbol{N}_i, \sum_{j=1}^{\bar{N}} v_j \boldsymbol{N}_j) = b(\boldsymbol{N}_i), \quad \text{for } i = 1, ..., \bar{N} \tag{A.31}$$

and due to the linearity of a,

$$\sum_{j=1}^{\bar{N}} a(\boldsymbol{N}_i, \boldsymbol{N}_j) v_j = b(\boldsymbol{N}_i), \quad \text{for } i = 1, ..., \bar{N} \tag{A.32}$$

Equation (A.32) describes a linear system of equations of the form:

$$[\boldsymbol{K}][\boldsymbol{v}] = [\boldsymbol{P}] \tag{A.33}$$

where $[\boldsymbol{K}]$ is an $\bar{N} \times \bar{N}$ matrix with element $K_{ij} = a(\boldsymbol{N}_i, \boldsymbol{N}_j)$ and $[\boldsymbol{P}]$ is an $\bar{N} \times 1$ vector with elements $P_i = b(\boldsymbol{N}_i)$. The solution of this system will give the vector of coefficients $\boldsymbol{v}$ and thus the approximate solution $\boldsymbol{u}_h$ to the boundary value problem.

In a similar fashion, the finite element method, which is a variation of the Galerkin method, considers piecewise polynomials as the basis functions $\{N_i\}_{i=1}^{n}$. These polynomials are supported on small patches of Ω, called *finite elements* and are obtained by discretizing $\Omega \in \mathbb{R}^n$ into polygonal domains. The displacement vector field $\boldsymbol{u}_h \in U^h$ and test functions $\boldsymbol{w} \in W^h$ can be approximated within each element e via the basis functions as:

$$\boldsymbol{u}_h^e = \sum_{i}^{n} N_i^e \boldsymbol{v}_i^e = [\boldsymbol{N}^e][\boldsymbol{v}^e] \tag{A.34}$$

$$\boldsymbol{w}^e = \sum_{i}^{n} N_i^e \boldsymbol{w}_i^e = [\boldsymbol{N}^e][\boldsymbol{w}^e] \tag{A.35}$$

where $\boldsymbol{v}_i^e, \boldsymbol{w}_i^e$ are 3×1 vectors (case of 3D elasticity), n is the number of nodes in the element e and $[\boldsymbol{N}^e]$ is $3 \times 3n$ matrix with entries

$$[\boldsymbol{N}^e] = [N_1^e \mathbb{I}_{3\times 3} \dots N_n^e \mathbb{I}_{3\times 3}] \tag{A.36}$$

The strains and displacements within each element are related via the following relation:

$$\{\boldsymbol{\epsilon}(\boldsymbol{u}_h)\} = \sum_{i=1}^{n} [\boldsymbol{B}_i^e] \boldsymbol{v}_i^e = [\boldsymbol{B}^e][\boldsymbol{v}^e] \tag{A.37}$$

$$\{\boldsymbol{\epsilon}(\boldsymbol{w})\} = \sum_{i=1}^{n} [\boldsymbol{B}_i^e] \boldsymbol{w}_i^e = [\boldsymbol{B}^e][\boldsymbol{w}^e] \tag{A.38}$$

where the 6×3 matrix $[\boldsymbol{B}_i^e]$ is introduced, such that

$$[\boldsymbol{B}_i^e] = \begin{bmatrix} \frac{\partial N_i^e}{\partial x_1} & 0 & 0 \\ 0 & \frac{\partial N_i^e}{\partial x_2} & 0 \\ 0 & 0 & \frac{\partial N_i^e}{\partial x_3} \\ \frac{\partial N_i^e}{\partial x_2} & \frac{\partial N_i^e}{\partial x_1} & 0 \\ 0 & \frac{\partial N_i^e}{\partial x_3} & \frac{\partial N_i^e}{\partial x_2} \\ \frac{\partial N_i^e}{\partial x_3} & 0 & \frac{\partial N_i^e}{\partial x_1} \end{bmatrix} \tag{A.39}$$

and also

$$[\boldsymbol{B}^e] = [\boldsymbol{B}_1^e \dots \boldsymbol{B}_n^e] \tag{A.40}$$

Eq. (A.25) then must be satisfied for every element Ω^e, or

$$\int_{\Omega^e} \boldsymbol{\sigma} : \boldsymbol{\epsilon}(\boldsymbol{w}) d\boldsymbol{x} = \int_{\Omega^e} \boldsymbol{f} \cdot \boldsymbol{w} d\boldsymbol{x} + \int_{\partial\Omega_N \cap \partial\Omega^e} \boldsymbol{w} \cdot \bar{\boldsymbol{t}} d\boldsymbol{s} + \int_{\partial\Omega^e \setminus \partial\Omega_N} \boldsymbol{w} \cdot \boldsymbol{t}_b d\boldsymbol{s} \tag{A.41}$$

Using matrix notation the above equation can be written as

$$\int_{\Omega^e} \{\boldsymbol{\epsilon}(\boldsymbol{w})\}^T \boldsymbol{D} \{\boldsymbol{\epsilon}(\boldsymbol{u}_h)\} d\boldsymbol{x} = \int_{\Omega^e} \{\boldsymbol{w}\}^T \{\boldsymbol{f}\} d\boldsymbol{x} + \int_{\partial\Omega_N \cap \partial\Omega^e} \{\boldsymbol{w}\}^T \{\bar{\boldsymbol{t}}\} d\boldsymbol{s} + \int_{\partial\Omega^e \setminus \partial\Omega_N} \{\boldsymbol{w}\}^T \{\boldsymbol{t}_b\} d\boldsymbol{s} \tag{A.42}$$

The last term in eq. (A.42) accounts for the tractions developed between the element and its adjacent elements. Inserting eqs. (A.37) and (A.38) into eq. (A.42) gives:

$$\begin{aligned} \int_{\Omega^e} ([\boldsymbol{B}^e][\boldsymbol{w}^e])^T \boldsymbol{D} ([\boldsymbol{B}^e][\boldsymbol{v}^e]) d\boldsymbol{x} = & \int_{\Omega^e} ([\boldsymbol{N}^e][\boldsymbol{w}^e])^T \{\boldsymbol{f}\} d\boldsymbol{x} + \int_{\partial\Omega_N \cap \partial\Omega^e} ([\boldsymbol{N}^e][\boldsymbol{w}^e])^T \{\bar{\boldsymbol{t}}\} d\boldsymbol{s} \\ & + \int_{\partial\Omega^e \setminus \partial\Omega_N} ([\boldsymbol{N}^e][\boldsymbol{w}^e])^T \{\boldsymbol{t}_b\} d\boldsymbol{s} \end{aligned} \tag{A.43}$$

Since $\boldsymbol{w}^e$ is arbitrary, it can be omitted from the above equation, that is

$$\cancel{[\boldsymbol{w}^e]^T} \int_{\Omega^e} [\boldsymbol{B}^e]^T \boldsymbol{D} [\boldsymbol{B}^e] d\boldsymbol{x}\ [\boldsymbol{v}^e] = \cancel{[\boldsymbol{w}^e]^T} \int_{\Omega^e} [\boldsymbol{N}^e]^T \{\boldsymbol{f}\} d\boldsymbol{x} + \cancel{[\boldsymbol{w}^e]^T} \int_{\partial\Omega_N \cap \partial\Omega^e} [\boldsymbol{N}^e]^T \{\bar{\boldsymbol{t}}\} d\boldsymbol{s}$$
$$+ \cancel{[\boldsymbol{w}^e]^T} \int_{\partial\Omega^e \setminus \partial\Omega_N} [\boldsymbol{N}^e]^T \{\boldsymbol{t}_b\} d\boldsymbol{s} \tag{A.44}$$

In essence, eq. (A.44) represents the $n \times n$ system of linear equations

$$[\boldsymbol{K}^e][\boldsymbol{v}^e] = [\boldsymbol{P}^e] + [\boldsymbol{P}^e_b] \tag{A.45}$$

where

$$[\boldsymbol{K}^e] = \int_{\Omega^e} [\boldsymbol{B}^e]^T \boldsymbol{D} [\boldsymbol{B}^e] d\boldsymbol{x} \tag{A.46}$$

$$[\boldsymbol{P}^e] = \int_{\Omega^e} [\boldsymbol{N}^e]^T \{\boldsymbol{f}\} d\boldsymbol{x} + \int_{\partial\Omega_N \cap \partial\Omega^e} [\boldsymbol{N}^e]^T \{\bar{\boldsymbol{t}}\} d\boldsymbol{s} \tag{A.47}$$

$$[\boldsymbol{P}^e_b] = \int_{\partial\Omega^e \setminus \partial\Omega_N} [\boldsymbol{N}^e]^T \{\boldsymbol{t}_b\} d\boldsymbol{s} \tag{A.48}$$

The vector $[\boldsymbol{P}^e_b]$ represents the force vector between the element e and its adjacent elements and does not need to be calculated since it cancels out when adding all element contributions to form the global system of equations

$$[\boldsymbol{K}][\boldsymbol{v}] = [\boldsymbol{P}] \tag{A.49}$$

The solution of the above linear system will eventually yield the coefficients for the approximate displacement field $\boldsymbol{u}_h$.

A.6 Special case: Plane Stress Problems - implementation

In this section, we will discuss about plane stress problems, which are special cases of 3D-elasticity, where the dimension of the structure along one direction (say the z-axis) is considerable smaller than the other two. Also, the loads are all in the $x - y$ plane. The displacement field has components $u(x, y)$ in the x-direction (horizontal) and $v(x, y)$ in y-direction (vertical).

In this case, the strain and stress vectors are:

$$\{\boldsymbol{\epsilon}\} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix}, \quad \{\boldsymbol{\sigma}\} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} \tag{A.50}$$

where, $\epsilon_x = \frac{\partial u}{\partial x}$, $\epsilon_y = \frac{\partial v}{\partial y}$, $\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$.

Strains and stresses are related using the following equation:

$$\begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \gamma_{xy} \end{bmatrix} = \frac{1}{E} \begin{bmatrix} 1 & -\nu & 0 \\ -\nu & 1 & 0 \\ 0 & 0 & 2(1+\nu) \end{bmatrix} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \tau_{xy} \end{bmatrix} \tag{A.51}$$

or

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \tau_{xy}\end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix}1 & \nu & 0\\ \nu & 1 & 0\\ 0 & 0 & \frac{1-\nu}{2}\end{bmatrix} = \begin{bmatrix}\epsilon_x\\ \epsilon_y\\ \gamma_{xy}\end{bmatrix} \tag{A.52}$$

that is, $\{\boldsymbol{\sigma}\} = \boldsymbol{E}\{\boldsymbol{\epsilon}\}$, with $\boldsymbol{E}$ being the elasticity matrix.

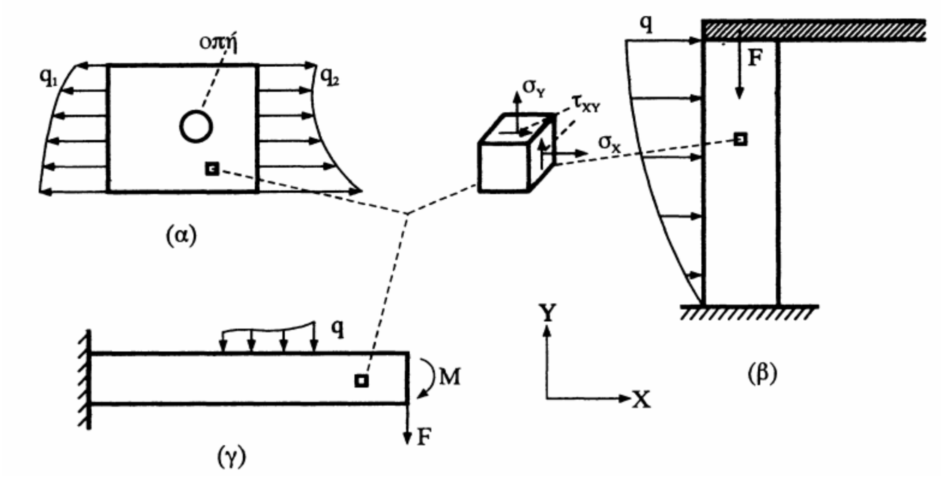


Figure A.1: Some examples of plane stress problems (from the book 'Analysis of Structures with the Finite Element Method' by M. Papadrakakis)

A.6.1 Quadrilateral plane stress finite elements

In this subsection, we discuss the technology of plane stress quadrilateral elements (quads).

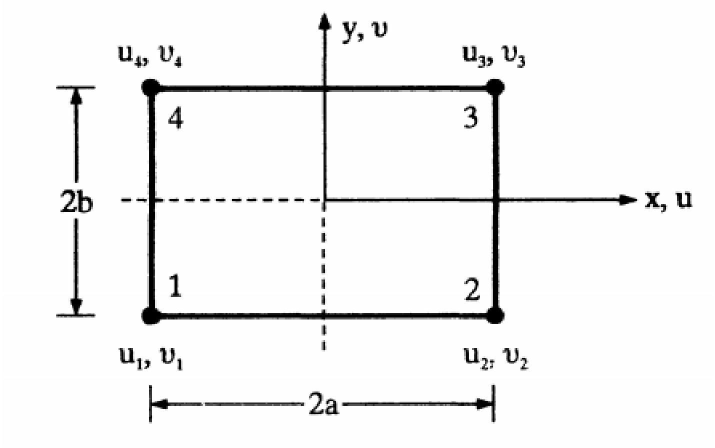


Figure A.2: Quadrilateral plane stress element (from the book 'Analysis of Structures with the Finite Element Method' by M. Papadrakakis)

We assume that the displacement field inside the finite element has horizontal u and vertical v components given by:

$$
\begin{aligned}
u &= \alpha_1 + \alpha_2 x + \alpha_3 y + \alpha_4 xy \\
v &= \alpha_5 + \alpha_6 x + \alpha_7 y + \alpha_8 xy
\end{aligned}
\tag{A.53}
$$

The above equation can also be written as:

$$
\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} 1 & x & y & xy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x & y & xy \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix} \tag{A.54}
$$

In fig. A.2 we see that at each node i of the 4 nodes of the element we have two displacements, namely, u_i, v_i. These are called **degrees of freedom**, or dofs for short, and we can associate the unknown parameters $\{\alpha_1, ..., \alpha_8\}$ with the dofs $\{u_1, v_1, ..., u_4, v_4\}$, as

$$
\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix} = \begin{bmatrix}
1 & -a & -b & ab & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -a & -b & ab \\
1 & a & -b & -ab & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & a & -b & -ab \\
1 & a & b & ab & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & a & b & ab \\
1 & -a & b & -ab & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -a & b & -ab
\end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \\ \alpha_8 \end{bmatrix} \tag{A.55}
$$

or,

$$
\{\boldsymbol{d}\} = [\boldsymbol{A}]\{\boldsymbol{\alpha}\} \tag{A.56}
$$

which gives

$$
\{\boldsymbol{\alpha}\} = [\boldsymbol{A}]^{-1}\{\boldsymbol{d}\} \tag{A.57}
$$

Inserting the above relation in eq. (A.54) we get

$$
\{\boldsymbol{u}\} = \begin{bmatrix} 1 & x & y & xy & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & x & y & xy \end{bmatrix} [\boldsymbol{A}]^{-1}\{\boldsymbol{d}\} \tag{A.58}
$$

which, after some algebra becomes:

$$
\{\boldsymbol{u}\} = \begin{bmatrix} N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 & 0 \\ 0 & N_1 & 0 & N_2 & 0 & N_3 & 0 & N_4 \end{bmatrix} \{\boldsymbol{d}\} \tag{A.59}
$$

with N_i, $i = 1, ..., 4$ being the **shape functions** given by:

$$
\begin{aligned}
N_1 &= \frac{1}{4}(1-\frac{x}{a})(1-\frac{y}{b}) \\
N_2 &= \frac{1}{4}(1+\frac{x}{a})(1-\frac{y}{b}) \\
N_3 &= \frac{1}{4}(1+\frac{x}{a})(1+\frac{y}{b}) \\
N_4 &= \frac{1}{4}(1-\frac{x}{a})(1+\frac{y}{b})
\end{aligned}
$$

We are in a position now, where we can evaluate the derivatives of the displacements and obtain a formula for the strain vector.

$$
\begin{aligned}
&\frac{\partial u}{\partial x} = N_{1,x}u_1 + N_{2,x}u_2 + N_{3,x}u_3 + N_{4,x}u_4 \\
&\frac{\partial v}{\partial y} = N_{1,y}v_1 + N_{2,y}v_2 + N_{3,y}v_3 + N_{4,y}v_4 \\
&\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} = N_{1,y}u_1 + N_{1,x}v_1 + N_{2,y}u_2 + N_{2,x}v_2 + N_{3,y}u_3 + N_{3,x}v_3 + N_{4,y}u_4 + N_{4,x}v_4 \quad \text{(A.60)}
\end{aligned}
$$

or, using matrix notation:

$$
\{\boldsymbol{\epsilon}\} = \begin{bmatrix} u_{,x} \\ v_{,y} \\ u_{,y}+v_{,x} \end{bmatrix} = \begin{bmatrix} N_{1,x} & 0 & N_{2,x} & 0 & N_{3,x} & 0 & N_{4,x} & 0 \\ 0 & N_{1,y} & 0 & N_{2,y} & 0 & N_{3,y} & 0 & N_{4,y} \\ N_{1,y} & N_{1,x} & N_{2,y} & N_{2,x} & N_{3,y} & N_{3,x} & N_{4,y} & N_{4,x} \end{bmatrix} \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix} \quad \text{(A.61)}
$$

or,

$$
\{\boldsymbol{\epsilon}\} = \boldsymbol{B}\{\boldsymbol{d}\} \quad \text{(A.62)}
$$

with $\boldsymbol{B}$ called the deformation matrix and its entries are:

$$
\boldsymbol{B} = \frac{1}{4ab} \begin{bmatrix} y-b & 0 & -y+b & 0 & y+b & 0 & -y-b & 0 \\ 0 & x-a & 0 & -x-a & 0 & x+a & 0 & -x+a \\ x-a & y-b & -x-a & -y+b & x+a & y+b & -x+a & -y-b \end{bmatrix} \quad \text{(A.63)}
$$

Finally, we can now compute the elements stiffness matrix that relates the external forces at the nodes of the element with the displacements expressed by the element's dofs. This is given by the following equation:

$$
\begin{aligned}
\boldsymbol{k} &= \int_{V_e} \boldsymbol{B}^T \boldsymbol{E} \boldsymbol{B} dV_e \\
&= t \int_{y=-b}^{y=b} \int_{x=-a}^{x=a} \boldsymbol{B}^T \boldsymbol{E} \boldsymbol{B} dx dy \quad \text{(A.64)}
\end{aligned}
$$

with t being the thickness of the element in the z-direction.

The result of integration will give the (symmetric) matrix:

$$\boldsymbol{k} = \frac{Et}{12(1-\nu^2)} \begin{bmatrix} 4r^{-1}+4\rho r & & & & & & & \\ \mu & 4r+4\rho r^{-1} & & & & & & \\ -4r^{-1}+2\rho r & \lambda & 4r^{-1}+4\rho r & & & & & \\ -\lambda & 2r-4\rho r^{-1} & -\mu & 4r+4\rho r^{-1} & & & & \\ -2r^{-1}-2\rho r & -\mu & 2r^{-1}-4\rho r & \lambda & 4r^{-1}+4\rho r & & & \\ -\mu & -2r-2\rho r^{-1} & -\lambda & -4r+2\rho r^{-1} & \mu & 4r+4\rho r^{-1} & & \\ 2r^{-1}-4\rho r & -\lambda & -2r^{-1}-2\rho r & \mu & -4r^{-1}+2\rho r & \lambda & 4r^{-1}+4\rho r & \\ \lambda & -4r+2\rho r^{-1} & \mu & -2r-2\rho r^{-1} & -\lambda & 2r-4\rho r^{-1} & -\mu & 4r+4\rho r^{-1} \end{bmatrix} \tag{A.65}$$

where, $r = \frac{a}{b}$, $\rho = \frac{1-\nu}{2}$, $\mu = \frac{3(1+\nu)}{2}$ and $\lambda = \frac{3(1-3\nu)}{2}$.

A.6.2 Structure discretization and global system of equations

In this section, we will consider the plane stress problem shown in figure A.3 so as to demonstrate the process of obtaining the global system of equations $\boldsymbol{KU} = \boldsymbol{F}$, applying the boundary conditions and eventually solving it.

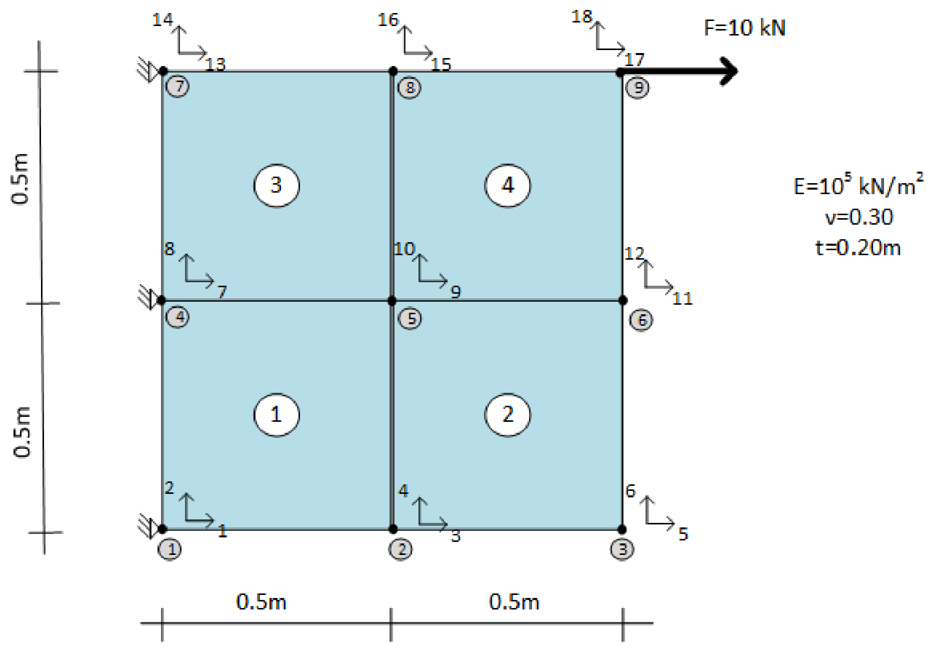


Figure A.3: Plane stress problem

The above problem is discretized into 4 identical quadrilateral elements, leading to 9 nodes $(1-9)$. Now, since each node is associated to 2 degrees of freedom, the problem has 18 dofs (1-18), which means that we have to solve a linear system of 18 equations for the unknown vector of displacements $\boldsymbol{U} \in \mathbb{R}^{18}$,

$$\boldsymbol{U} = [u_1, u_2, ..., u_{17}, u_{18}]^T \tag{A.66}$$

Also, the dimensions of $\boldsymbol{K}$ will be 18×18 which is built by 'adding' the 8×8 elements' stiffness matrices $\boldsymbol{k}^e$, $e = 1, 2, 3, 4$. Since these elements have the same geometry and E, ν, t parameters, then $\boldsymbol{k}^1 = \boldsymbol{k}^2 = \boldsymbol{k}^3 = \boldsymbol{k}^4$ and they are obtained by eq. (A.65).

So, each $\boldsymbol{k}^e$ will be

$$\boldsymbol{k}^e = \begin{array}{c} u'_1 \\ u'_2 \\ \\ u'_8 \end{array} \begin{bmatrix} k^e_{11} & k^e_{12} & \cdots & k^e_{18} \\ k^e_{21} & k^e_{22} & \cdots & k^e_{28} \\ \vdots & \vdots & \cdots & \vdots \\ k^e_{81} & k^e_{82} & \cdots & k^e_{88} \end{bmatrix} \tag{A.67}$$

where in the above notation $u'_1, u'_2, ..., u'_8$ refer to the element's **local** degrees of freedom. We have to associate them to the global $u_1, u_2, ..., u_{18}$ dofs of the system. This is done by writing each $\boldsymbol{k}^e$ (8×8) as $\boldsymbol{k}^e_{glob}$ (18×18) by associating the local dofs $u'_1 - u'_8$ to the global $u_1 - u_{18}$. To illustrate the process, let's first show the dofs of element 1 in the local and in the global system:

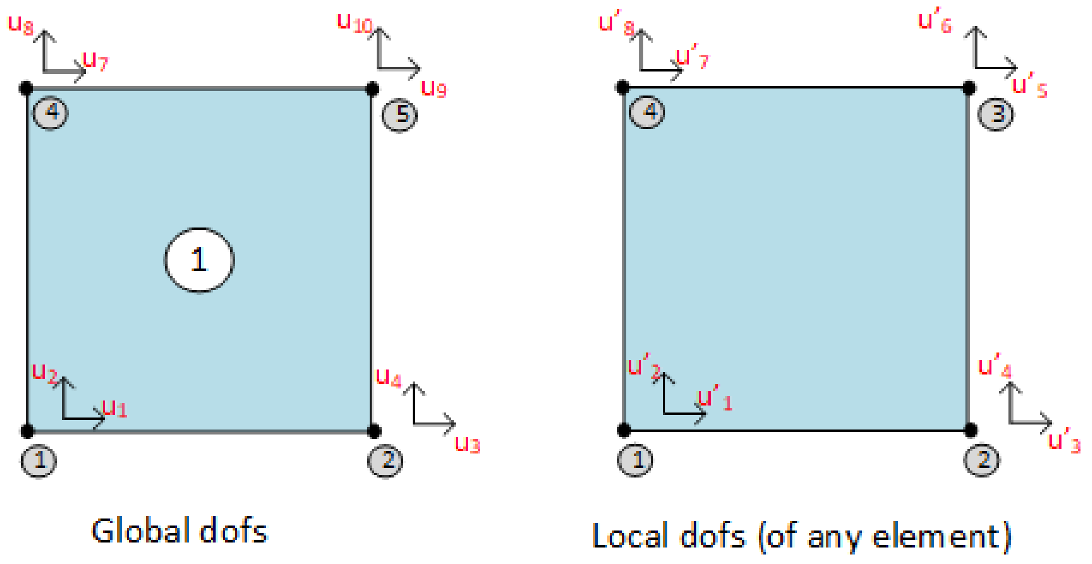


Figure A.4: Numbering of element 1 nodes and dofs in the global and local system

Then, $\boldsymbol{k}^1_{glob}$ (18×18) becomes

$$\boldsymbol{k}^1_{glob} = \begin{array}{c|cccccccccc} & 1 & 2 & 3 & 4 & 5-6 & 7 & 8 & 9 & 10 & 11-18 \\ \hline u_1 & k^1_{11} & k^1_{12} & k^1_{13} & k^1_{14} & \mathbf{0} & k^1_{17} & k^1_{18} & k^1_{15} & k^1_{16} & \mathbf{0} \\ u_2 & k^1_{21} & k^1_{22} & k^1_{23} & k^1_{24} & \mathbf{0} & k^1_{27} & k^1_{28} & k^1_{25} & k^1_{26} & \mathbf{0} \\ u_3 & k^1_{31} & k^1_{32} & k^1_{33} & k^1_{34} & \mathbf{0} & k^1_{37} & k^1_{38} & k^1_{35} & k^1_{36} & \mathbf{0} \\ u_4 & k^1_{41} & k^1_{42} & k^1_{43} & k^1_{44} & \mathbf{0} & k^1_{47} & k^1_{48} & k^1_{45} & k^1_{46} & \mathbf{0} \\ u_5-u_6 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ u_7 & k^1_{71} & k^1_{72} & k^1_{73} & k^1_{74} & \mathbf{0} & k^1_{77} & k^1_{78} & k^1_{75} & k^1_{76} & \mathbf{0} \\ u_8 & k^1_{81} & k^1_{82} & k^1_{83} & k^1_{84} & \mathbf{0} & k^1_{87} & k^1_{88} & k^1_{85} & k^1_{86} & \mathbf{0} \\ u_9 & k^1_{51} & k^1_{52} & k^1_{53} & k^1_{54} & \mathbf{0} & k^1_{57} & k^1_{58} & k^1_{55} & k^1_{56} & \mathbf{0} \\ u_{10} & k^1_{61} & k^1_{62} & k^1_{63} & k^1_{64} & \mathbf{0} & k^1_{67} & k^1_{68} & k^1_{65} & k^1_{66} & \mathbf{0} \\ u_{11}-u_{18} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array} \tag{A.68}$$

Similarly, we get

$$\boldsymbol{k}^2_{glob} = \begin{array}{c|ccccccccccc|} & 1{-}2 & 3 & 4 & 5 & 6 & 7{-}8 & 9 & 10 & 11 & 12 & 13{-}18 \\ u_1{-}u_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ u_3 & \mathbf{0} & k^2_{11} & k^2_{12} & k^2_{13} & k^2_{14} & \mathbf{0} & k^2_{17} & k^2_{18} & k^2_{15} & k^2_{16} & \mathbf{0} \\ u_4 & \mathbf{0} & k^2_{21} & k^2_{22} & k^2_{23} & k^2_{24} & \mathbf{0} & k^2_{27} & k^2_{28} & k^2_{25} & k^2_{26} & \mathbf{0} \\ u_5 & \mathbf{0} & k^2_{31} & k^2_{32} & k^2_{33} & k^2_{34} & \mathbf{0} & k^2_{37} & k^2_{38} & k^2_{35} & k^2_{36} & \mathbf{0} \\ u_6 & \mathbf{0} & k^2_{41} & k^2_{42} & k^2_{43} & k^2_{44} & \mathbf{0} & k^2_{47} & k^2_{48} & k^2_{45} & k^2_{46} & \mathbf{0} \\ u_7{-}u_8 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ u_9 & \mathbf{0} & k^2_{71} & k^2_{72} & k^2_{73} & k^2_{74} & \mathbf{0} & k^2_{77} & k^2_{78} & k^2_{75} & k^2_{76} & \mathbf{0} \\ u_{10} & \mathbf{0} & k^2_{81} & k^2_{82} & k^2_{83} & k^2_{84} & \mathbf{0} & k^2_{87} & k^2_{88} & k^2_{85} & k^2_{86} & \mathbf{0} \\ u_{11} & \mathbf{0} & k^2_{51} & k^2_{52} & k^2_{53} & k^2_{54} & \mathbf{0} & k^2_{57} & k^2_{58} & k^2_{55} & k^2_{56} & \mathbf{0} \\ u_{12} & \mathbf{0} & k^2_{61} & k^2_{62} & k^2_{63} & k^2_{64} & \mathbf{0} & k^2_{67} & k^2_{68} & k^2_{65} & k^2_{66} & \mathbf{0} \\ u_{13}{-}u_{18} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{array} \tag{A.69}$$

and we repeat the process to get $\boldsymbol{k}^3_{glob}, \boldsymbol{k}^4_{glob}$. Then, we can compute the global stiffness matrix $\boldsymbol{K}$ by simply adding the $\boldsymbol{k}^e_{glob}$ matrices:

$$\boldsymbol{K} = \boldsymbol{k}^1_{glob} + \boldsymbol{k}^2_{glob} + \boldsymbol{k}^3_{glob} + \boldsymbol{k}^4_{glob} \tag{A.70}$$

The next step is apply the problem's boundary conditions. From figure A.3 we see that we have $u_1 = u_2 = u_7 = u_8 = u_{13} = u_{14} = 0$ as our boundary conditions. One easy way to insert these boundary conditions to the linear system of equations is the following. For each constrained dof u_i, we set the elements of the i-th row and i-th column of the global stiffness matrix $\boldsymbol{K}$ to zero, except for the diagonal element $\boldsymbol{K}_{ii}$, to which we assign a very large positive number (e.g. 10^{10}).

The last thing we need is to define the elements of the external force vector $\boldsymbol{F} \in \mathbb{R}^{18}$. From figure A.3 we see that the only external load we have is at node 9, in the direction of dof u_{17} with magnitude $10kN$. So,

$$\boldsymbol{F} = [0, 0, ..., 0, 10, 0] \tag{A.71}$$

Now, we can solve the system of equations $\boldsymbol{KU} = \boldsymbol{F}$ and get:

$$\boldsymbol{U} = \boldsymbol{K}^{-1}\boldsymbol{F} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \\ u_{16} \\ u_{17} \\ u_{18} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -3.77 \\ -3.84 \\ -4.90 \\ -11 \\ 0 \\ 0 \\ 1.80 \\ -2.67 \\ 1.49 \\ -13 \\ 0 \\ 0 \\ 9.28 \\ -3.88 \\ 22 \\ -16 \end{bmatrix} \times 10^{-4} \tag{A.72}$$

A.7 Additional literature