Technische Universität Braunschweig

INSTITUT FÜR AKUSTIK

# A Statistical Approach for the Fusion of Data and Finite Element Analysis in Vibroacoustics

**Masterarbeit**

**Lucas Hermann**

Matr.-Nr.: 4990987

**Erstprüferin:** Prof. Dr.-Ing. Sabine C. Langer
**Zweitprüfer:** Prof. Dr.-Ing. Ulrich Römer

October 2021

# Declaration

Hiermit versichere ich, Lucas Hermann, durch meine Unterschrift, dass ich die vorliegende Masterarbeit mit dem Titel "<<Titel>>" selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Alle Stellen, die wörtlich oder sinn- gemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen sind, habe ich als solche kenntlich gemacht. Insbesondere sind auch solche Inhalte gekennzeichnet, die von betreuenden wissenschaftlichen Mitarbeiterinnen und Mitarbeitern des Instituts für Akustik eingebracht wurden.

Die Arbeit oder Auszüge daraus haben noch nicht in gleicher oder ähnlicher Form dieser oder einer anderen Prüfungsbehörde vorgelegen.

Mir ist bewusst, dass Verstöße gegen die Grundsätze der Selbstständigkeit als Täuschung betrachtet und entsprechend der Prüfungsordnung geahndet werden.

Braunschweig, September 7, 2021

Lucas Hermann

# Abstract

To increase the accuracy of an FEM model, statFEM is is used in a vibroacoustics scenario. An introduction to vibroacoustics, FEM, Gaussian Processes and statFEM is given before the methods are applied. A 2D vibroacoustics problem is solved via FEM and synthetic measurements are used to condition the FEM solution, which serves as a Bayesian prior, on. It is shown that already comparatively few sensor locations and observations per sensor lead to a much more accurate posterior model.

# Contents

# 1 Introduction

A model is only able to describe reality up to a certain degree. It usually consists of a set of equations and corresponding parameters which can be tweaked in order to make it behave as closely to the desired functionality as possible or to match data from measurements. Uncertainty in the parameters can be propagated to the solution, leading to a probability density for the solution which assigns probabilities to different possible outcomes. However, as this is a valid approach, as soon as the real problem behaves in a more complex manner than the model is able to describe, the orocedure is reaching its limits. The headroom given by the probability densities in the parameters just isn't enough to tweak them in a way that makes the model describe reality accurately. There is now a gap between model and reality which has to be filled.

A common approach to form a model in engineering is the Finite Element Method (FEM). At the base of it stands a partial differential equation (PDE), which is a model of reality itself. The PDE is simplified using so-called ansatz functions which have weights that are tweaked in order to get the most accurate picture of reality. Traditionally, the FEM is purely deterministic but, nevertheless, it is rather simple to form a Bayesian approach to FEM, in which the parameters and the solution are not scalars but random variables which have an underlying probability distribution. Recently, a method, called statFEM, was developed, which also allows for bridging the gap between a Bayesian FEM model and reality. This is in principle done by assuming the discrepancy between model and reality to behave in terms of a Gaussian Process (GP). A GP can be, unlike a probability distribution over a parameter, considered as a probability distribution over a whole function. Since there are no parameters left in the FEM model to tweak, this is the only way to go: The most suitable function to fill the model discrepancy gap has to be found without building another parameterized model. The GP is based on measurement data: By comparing the FEM model with observed data and then updating it in a Bayesian manner, a distribution over functions can be found, which makes the now updated FEM model behave more closely to reality.

In this thesis, statFEM is applied to the field of vibroacoustics. The physical problem under consideration is an acoustical cavity which is excited by a plate at its boundary. The behavior of the fluid inside the cavity is described by the Helmholtz equation and depends strongly on the chosen boundary conditions. Obviously, there is a lot of uncertainty involved in the modeling of the boundary conditions: It is not possible to completely describe the behavior of the vibrating plate or the partially absorbing walls of the cavity. Nevertheless, it is possible to measure the sound pressure inside the cavity: Assuming the sound field to behave in terms of a GP makes it possible to form a more accurate posterior solution using the measured data and the prior in form of the potentially under-complex or simply wrong FEM solution.

In the first chapters, the theory behind GPs, the FEM and statFEM is explained thoroughly. Then, the methods are explained and validated using the textbook-example of the Poisson equation before they are applied to the more complex vibroacoustics problem. The behavior of the new model is explained and discussed in detail and information on possible use cases and limitations are given.

# 2 FEM Solution of the Helmholtz Equation

This thesis is based on two pillars: Solving the governing equation for acoustics using the Finite Element Method (FEM) and implementing a probabilistic approach using GPs and statFEM. For the first one, the equations for the Helmholtz equation and the FEM are going to be derived in this chapter.

An acoustic field is represented in time domain by the wave equation. It returns the acoustical pressure at a point in space $x$ for a time variable $t$. The Helmholtz equation describes the wave equation in a time-independent manner, i.e. in frequency domain. It has a dependency on the frequency, so the goal is to solve the Helmholtz equation for a range of frequencies in order to obtain a frequency response plot which gives insight on the modal behavior of the system.

## 2.1 Wave Equation and Helmholtz Equation

In a typical vibroacoustic problem, for example an interior structural acoustic coupling problem [1], there are usually two coupled domains: At first, there is some kind of solid medium which emits sound. The movement of that solid can be described within the field of elastodynamics using for instance (a) Bernoulli or Timoshenko beam theory for 1D beam-like structures, (b) Kirchhoff or Reissner-Mindlin plate theory for 2D plate-like structures. The focus of this thesis is another: As soon as the sound is emitted from the solid, the wave equation is used to describe the motions of sound waves through an acoustic medium such as air. The following closely follows [2] and [3]. The behavior of a gas in a domain $\Omega$ can be described using the density $\rho$, the pressure $p$ and the velocity $v$. By changing the velocity or pressure locally by e.g. a speaker or a vibrating plate, this change is propagated through the medium. One can think of the gas as a system of infinitesimally small spring-mass systems. Deflecting one spring yields a deflection of the adjacent spring, or gas volume, but time-delayed due to the gas's inertia. That leads to a wave propagating through the medium. Newton's second law states that mass times acceleration equals force. If a gas volume is accelerated in one direction, that leads to a force on the neighboring particle [3].

$$m\dot{v} = S\left[p(x) - p(x + \Delta x)\right] \tag{2.1}$$

with $m$ the mass of the gas volume, $S$ the surface of the contact area of the two volumes and $\Delta x$ the length of a gas volume in the direction of motion. Considering $m = \Delta x S \rho$ there holds

$$\rho\dot{v} = -\frac{p(x) - p(x + \Delta x)}{\Delta x} \,. \tag{2.2}$$

Assuming interaction between infinitessimaly small gas particles and therefore taking the limit of the difference quotient in (2.2) yields the differential quotient and hence

$$\rho\dot{v} = -\nabla p \,. \tag{2.3}$$

$\nabla$ is the nabla operator $\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_n}\right)$ which consists of the partial derivatives of the used coordinates. For cartesian coordinates, applied as a product to a quantity, it describes the gradient of the

quantity, i.e.

$$\nabla p = \mathrm{grad} p = \left( \frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z} \right) . \tag{2.4}$$

If a volume is expanded, the pressure of the contained gas drops and vice versa. Therefore a change of volume is proportional to a negative change of pressure. An expanding volume is also represented by the divergence of the velocity [2]. Imagine quickly pulling a piston out of an airtight cylinder: The air particles will move in the direction of the piston movement while the volume expands and the pressure drops. Hence, there holds

$$\dot{p} = -k \nabla u \tag{2.5}$$

with $k$ a proportionality constant dependent on the medium. Differentiating (2.5) with respect to $t$ yields

$$\ddot{p} = -k \nabla \dot{u} . \tag{2.6}$$

Inserting 2.3 and considering $c^2 = k/\rho$ there holds for the acoustic wave equation

$$\ddot{p} = c^2 \nabla^2 p \tag{2.7}$$

which involves derivatives in both time and space. To obtain a general representation of the behaviour of a pressure field independent of time, the Helmholtz equation can be derived as follows. Considering a separation ansatz [4]

$$p(x,t) = X(x) \cdot T(t) , \tag{2.8}$$

there holds for 2.7

$$X(x) \cdot \frac{\partial^2 T(t)}{\partial t^2} = c^2 \nabla^2 X(x) \cdot T(t) \tag{2.9}$$

$$\frac{1}{T(t)} \frac{\partial^2 T(t)}{\partial t^2} = c^2 \frac{1}{X(x)} \nabla^2 X(x) = const. = -\omega^2 . \tag{2.10}$$

With the wave number $k = \omega / c$

$$\nabla^2 X(x) + k^2 X(x) = 0 \tag{2.11}$$

is the Helmholtz equation which is a representation of the wave equation independent of time [5, p. 1083 ff.]. With $p$[Pa] the pressure field and $k$ the wave number it is in the following going to be referred to by

$$\nabla^2 p + k^2 p = 0 . \tag{2.12}$$

With $\omega$[rad] the angular frequency, $f$[$\mathrm{s}^{-1}$] the frequency and $c_0$[$\mathrm{m\,s}^{-1}$] the speed of sound in the considered medium $k$ is defined as

$$k = \frac{\omega}{c_0} = \frac{2\pi f}{c_0} . \tag{2.13}$$

(2.12) is solved as a boundary value problem which means that the behavior of the system has to be described beforehand for the boundaries of the calculation domain. That is done using boundary conditions.

**Boundary Conditions** The following closely follows [1]. There are three different types of boundary conditions: Dirichlet, Neumann and Robin type. Neumann boundary conditions are also called natural boundary conditions because these automatically arise in the equation while deriving the weak form of the problem. Dirichlet boundary conditions, on the other hand, don't. They are called essential boundary conditions and have to be applied by hand after deriving the weak form and building the linear system of equations.

The Dirichlet boundary condition for the Helmholtz equation is simply a prescribed pressure $\bar{p}(x)$ at the $\Gamma_D$ part of the boundary:

$$p(x) = \bar{p}(x) \quad \forall x \in \Gamma_D \, . \tag{2.14}$$

The Neumann boundary condition at $\Gamma_N$ reads

$$\frac{\mathrm{d}p(x)}{\mathrm{d}n} = \rho\omega^2 U\vec{(x)} \cdot \vec{n} \quad \forall x \in \Gamma_N \tag{2.15}$$

with $\vec{U}$ the displacement and $\vec{n}$ the outer normal direction of the surface. It can be thought of as a piston moving with circular frequency $\omega$ at the boundary which creates a velocity and pressure change. Choosing a homogeneous Neumann BC $\frac{\mathrm{d}p(x)}{\mathrm{d}n} = 0$ resembles a reflecting boundary. The part of the boundary, where a homogeneouos Neumann BC is applied, is going to be called $\Gamma_{N0}$.

The Robin or impedance boundary condition at $\Gamma_R$ reads

$$\frac{\mathrm{d}p(x)}{\mathrm{d}n} + ik\beta p(x) = 0 \quad \forall x \in \Gamma_R \tag{2.16}$$

with $\beta$ the specific normalized acoustic admittance. The Robin boundary condition is also called impedance boundary conditions because an acoustic impedance can be simulated what leads to e.g. partially reflecting walls.

Figure 2.1 shows an example on how boundary conditions could be applied on a 2D domain. $\Gamma_{N0}$ boundary conditions are the mentioned natural boundary conditions which don't have to be applied explicitly. Still, a Neummann boundary condition $\Gamma_N$ can be imposed onto the system which behaves differently. Figure 2.2 shows the setting chosen in this thesis: All walls are reflecting and on the left side of the domain a Neumann boundary conditon is imposed which is used to model the previously mentioned sound source.

FEniCS [6] is a Python/C++ library for solving PDEs with the Finite Element Method. It makes it easy to go directly from the variational formulation of a problem to solving it. Using FEniCS, the boundary conditions can be applied to the system of equations by

```
class BoundaryX_L(SubDomain):
        self.tol = 1E-14
        def inside(self, x, on_boundary):
                return on_boundary and near(x[0], 0, tol)# and (x[1] < 0.3)
boundary_markers.set_all(9999) # all markers default to zero.
bxL = BoundaryX_L()
bxL.mark(boundary_markers,0) # left side is marked as "0"
ds = Measure('ds', domain=self.mesh, subdomain_data=boundary_markers) #define a new operator for the b
```

This construction can then be used when defining the variational problem:

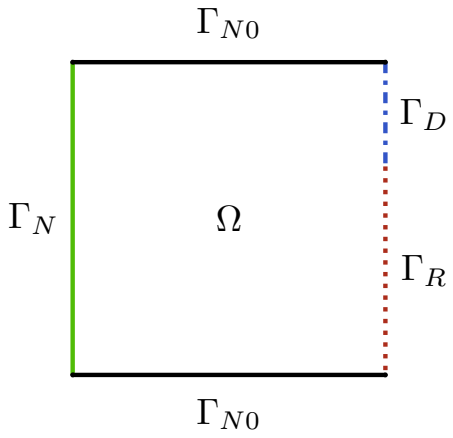



Figure 2.1: Two dimensional representation of various boundary conditions. In this example, the left side is a Neumann boundary, the upper and lower boundaries are totally reflecting and the right side consists of an impedance BC and a Dirichlet BC.

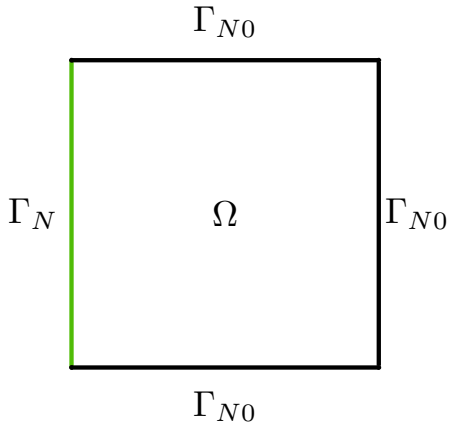



Figure 2.2: Boundary Conditions of the problem for this thesis. A Neumann boundary is applied on the side of the domain. It is going to be modeled as a GP. The rest of the boundary is considered as reflecting, i.e. $\Gamma_N = 0$.

```
U = 0.0001 #Piston displacement
g = self.rho * self.omega**2 * U
a = inner(nabla_grad(self.u), nabla_grad(self.v))*dx - self.k**2 * inner(self.u,self.v)*dx #variationa
L = (self.v*g)*ds(0) # 0 is the chosen boundary marker
```

Having the problem defined mathematically, the equations can now be prepared for being solved by FEM.

## 2.2 The Classical Finite Element Method

The Finite Element Method (FEM) is a procedure which makes it simple to approximately solve Partial Differential Equations (PDEs) which would be very hard or even impossible to solve analytically. As the name suggests, the calculation domain is split into $n_e$ individual elements on which the PDE is approximated using so-called ansatz functions. The process of dividing the domain in $n_e$ elements is called discretization. The polynomial degree of the ansatz functions and the number of elements determine the accuracy of the approximation. In order to set up the FEM, the PDE needs to be in the so-called variational, or "weak", formulation. After discretization of the domain and deriving the weak formulation, the PDE can be approximated on the individual elements. Assembling the element matrices yields a global system of equations which can, after boundary conditions are applied, be solved for the unknown vector.

### 2.2.1 Integral and Weak Formulation

- Explain Vector spaces - Test function: needs to obey the dirichlet conditions - explain what a bilinear and a linear form is!

The Helmholtz equation is the so-called governing equation for the problem considered in this thesis. The governing equation is given in the strong form and is usually of higher order. This property makes it hard to solve the equation, both analytically and numerically. For being able to apply the FEM, the order of the PDE has to be lowered. That is archieved by first bringing the equation in the weak, or also called variational, formulation. One way to derive the weak formulation of the governing equation is the method of weighted residuals.

**The Weigthed Residuals Method** The residual $R$ is the difference between approximation and exact solution. The exact solution is usually not known. Therefore the residual is formed by inserting the approximation of the solution into the PDE. The result is non-zero and a measure for the quality of the approximation [7]. To increase the quality, the residual has to be minimized. An FEM approximation, see 2.2.3 for details, uses FEM basis functions which are weighted with some factors $c_j$. The goal is finding the factors which minimize $R$. The seeked approximated solution vector to the Helmholtz equation $\boldsymbol{p}$ is part of a space $V$ which is spanned by the FEM basis functions $\varphi_i$. Demanding the scalar product of the residual and a test function $v$ to be zero

$$(R, v) = 0, \quad \forall v \in V \,, \tag{2.17}$$

i.e. demanding that they are orthogonal, can be used to minimize $R$. $V$ can be spanned by the FEM basis functions as well. Nevertheless, it doesn't necessarily need to be defined in the same space as $\boldsymbol{p}$. Using the same space would be called the Galerkin method. Applying the Weighted Residuals Method to the Helmholtz equation is executed as follows. At first the strong formulation (PDE) is multiplied with the test function $v$ and integrated over the domain $\Omega$ to get the so-called integral formulation:

$$\int_\Omega v \nabla^2 p \, \mathrm{d}\Omega + \int_\Omega k^2 p v \, \mathrm{d}\Omega = 0 \tag{2.18}$$

The approximated function within an element, here it is $p$, is called the trial function. It is part of a constrained space of trial functions or termed as the trial space. The test function $v$ is also part of a test space. Usually, trial and test spaces are chosen to be the same. Both trial and test spaces have to fulfill

the Dirichlet boundary conditions imposed on the PDE. The Helmholtz equation and therefore also the now formed integral formulation contain derivatives of second order. However, the FEM works by adjoining individual element functions to a global function. Therefore, the global solution is a piecewise polynomial [7] what causes problems when calculating higher-order derivatives because those become discontinuous. In consequence, integration by parts is performed using Green's first identity [1, pp. 53,54] on the higher-order terms to get lower order derivatives. It makes use of the divergence theorem, also called Gauss's theorem:

$$\int_\Omega \nabla \cdot \vec{F} \, \mathrm{d}\Omega = \oint_\Gamma \vec{F} \cdot \vec{n} \, \mathrm{d}\Gamma \, . \tag{2.19}$$

It states that all of the sources and sinks inside a domain (which generate divergence) must be equal to the amount of flux over the boundary of that domain. By defining $F = vu$ with $u = \nabla p$ and making use of the product rule of vector calculus $\nabla \cdot (vu) = v\nabla \cdot u + u \cdot \nabla v$ there holds for Green's first identity

$$\oint_\Gamma v \nabla p \, \mathrm{d}\Gamma = \int_\Omega \nabla p \cdot \nabla v \, \mathrm{d}\Omega + \int_\Omega v \nabla^2 p \, \mathrm{d}\Omega \, . \tag{2.20}$$

Applying it to (2.18) yields

$$- \int_\Omega \nabla p \cdot \nabla v \, \mathrm{d}\Omega + \oint_\Gamma v \nabla p \, \mathrm{d}\Gamma + \int_\Omega k^2 p v \, \mathrm{d}\Omega = 0 \tag{2.21}$$

with the boundary of the domain $\Gamma$. The resulting equation is now called the weak formulation as opposed to the strong formulation since the trial function now only needs to be differentiable once less often. The boundary term can be decomposed into three parts

$$\oint_\Gamma v \nabla p \, \mathrm{d}\Gamma = \oint_{\Gamma_D} v \nabla \bar{p} \, \mathrm{d}\Gamma + \oint_{\Gamma_N} v(\rho \omega^2 \vec{U} \vec{n}) \, \mathrm{d}\Gamma + \oint_{\Gamma_R} v(-ik\beta p) \, \mathrm{d}\Gamma \tag{2.22}$$

with $\oint_{\Gamma_D} v \nabla \bar{p} \, \mathrm{d}\Gamma = 0$ for the Dirichlet part because if the value at the boundary is known, then $\bar{f}$ is a scalar quantitity of which the derivative in outward normal direction of the boundary is zero.

### 2.2.2 Discretization

In this thesis, two different orders of domains are used: In a simple example, a 1D domain is going to serve as an illustration. For the actual vibroacoustics problem, a 2D domain will be examined. In this chapter, the FEM discretization for both orders is discussed. Discretization means that the domain $\Omega$ is split into $n_e$ individual elements $\Omega^{(e)}$

$$\Omega = \bigcup_{e=1}^{n_e} \Omega^{(e)} \quad e = 1, ..., n_e \, . \tag{2.23}$$

This holds not only for 1D meshes but for 2D and 3D meshes as well. For 1D meshes, such as in Figure 2.3, each element has two so-called nodes. This is because between the element boundaries a linear function is spanned which needs two support points to be defined. In a 2D domain, an element would have 3 nodes, see Figure 2.4. Also different shapes of elements, e.g. quaratic elements with 4 nodes, are possible. The collection of nodes is called a mesh. The total number of nodes in a 1D mesh is $n = n_e + 1$, in a 2D mesh it is $n = n_x \times n_y$ with $n_x$ and $n_y$ the number of nodes in each coordinate. The minimum degree of the used function is determined by the weak form: It always needs to be differentiable, therefore a constant approximation function would not work [1].

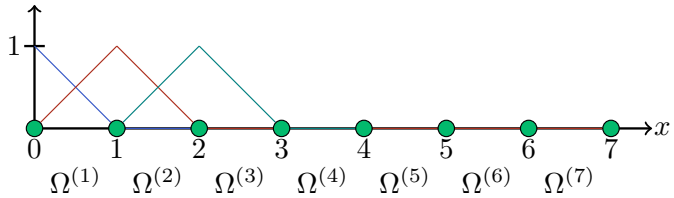



Figure 2.3: A discretized 1D domain with nodes in green and linear ansatz functions shown for the first four nodes. The ansatz functions are 1 on their respective nodes and are 0 on all other nodes.

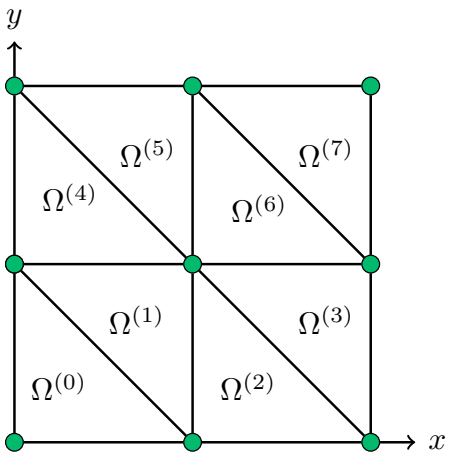



Figure 2.4: A discretized 2D domain with nodes in green. Also here, the ansatz functions are defined so that they are 1 at their respective nodes and zero at all other nodes.

Figures 2.3 and 2.4 illustrate the discretization of a 1D and 2D domain, respectively. For the 1D domain, the first ansatz functions are sketched: It is visible that for each node there is an ansatz function that yields the value 1 only at that node and 0 at all other nodes in the domain. The finer the mesh, i.e. the more elements are used in a domain, the closer the FEM approximation comes to the exact solution. Since computing very fine meshes is computationally expensive, a convergence study is conducted in order to find the element size from which on smaller element sizes don't result in more accurate results.

Using FEniCS, a 1D mesh can be created by calling

```
self.dom_a = 0.0 #domain boundaries
self.dom_b = 1.0
self.ne        = 60 #number of elements
self.mesh = IntervalMesh(self.ne,self.dom_a,self.dom_b) #define mesh
```

The interval of the mesh and the desired number of elements have to be provided.

A 2D mesh can, for example, be obtained by calling

```
self.a,self.b = 2,2
self.mesh = UnitSquareMesh(self.a,self.b) #define mesh
```

In this case a unit square is chosen what means that the domain size will be quadratic and of length $1.0$ in every direction. 2 elements were chosen for every dimension what leads to a mesh looking like 2.4. Of course every desired 2D shape can be used as a mesh.

### 2.2.3 Approximation and Building Element Matrices

In FEM, the values between nodes within an element are interpolated with basis functions $\varphi_i$. For each degree of freedom in the domain an FEM basis function is defined. By weighing those basis functions, the solution can be approximated with

$$p(x_i) = \sum_{j=1}^{n} c_j \varphi_j(x_i) = f(x_i) \; . \tag{2.24}$$

(2.24) means, that the value of one particular point $x_i$ is the sum of all products of all weight coefficients $c_j$ and test functions $\varphi_j$ evaluated at that point. If now orthogonal basis functions are used, as it is commonly the case in FEM, (2.24) simplifies as described below. One popular choice of FEM basis functions is the Lagrange family of polynomials. The right hand side $f(x_i)$ does, in practice, describe the source term of the PDE which does not depend on $p(x)$.

**Lagrange Polynomials** Following [7]. Lagrange polynomials are one possible choice for FEM basis functions and are used in this thesis. They can be calculated for any desired degree $N$ and their main property is that they are $1$ at only a single node on the domain, $0$ at all other nodes and that they are nonzero on only a very limited fraction of the domain, as opposed to e.g. basis functions constructed from $\sin$ and $\cos$ terms. They are used as an interpolation between given points which means that the polynomial always exactly goes through the given points. The behaviour at all other points is determined by the definition of the Lagrange polynomial

$$\varphi_i(x) = \prod_{j=0, j\neq i}^{N} \frac{x - x_j}{x_i - x_j} \; . \tag{2.25}$$

They are frequently used in FEM because there holds, if at $x_s$ there is a node of the mesh

$$\varphi_i(x_s) = \delta_{is}, \quad \delta_{is} = \begin{cases} 1 & \text{for} \quad i = s \\ 0 & \text{for} \quad i \neq s \end{cases} \tag{2.26}$$

which means that the polynomial is only $1$ at one node in the domain. At all other nodes it is $0$. Between nodes the Lagrange polynomial is spanned. This behavior simplifies (2.24), if $x_i$ is the location of a node of the mesh:

$$p(x_i) = \hat{p}_i \varphi_i(x_i) = \hat{p}_i \; . \tag{2.27}$$

The weight factor is therefore the approximated value at the node. Another asset of using polynomials with $\varphi_i(x) = 0$ in most parts of the domain is that they lead to sparse matrices which can be solved much quicker than densely packed ones. Now, to apply it to the weak form of the Helmholtz equation (2.21), an approximation ansatz has to be made for both the trial and the test function. There holds for each element, if the same basis functions are used for both trial and test spaces,

$$\begin{aligned} \tilde{p} &= \boldsymbol{\varphi}^{\boldsymbol{T}} \hat{\boldsymbol{p}} \\ \tilde{v} &= (\boldsymbol{\varphi}^{\boldsymbol{T}} \delta \hat{\boldsymbol{p}})^T \end{aligned} \tag{2.28}$$

with $\delta\hat{\boldsymbol{p}}$ an arbitrary variation [Langer CompAc] of the pressure $\hat{\boldsymbol{p}}$, which gets cancelled out later on because the test function is present in every term. Inserting (2.28) in the weak form, considering no Robin boundary conditions, results in

$$\sum_e \int_{\Omega_e} (\nabla\boldsymbol{\varphi}\nabla\boldsymbol{\varphi}^T)\,\mathrm{d}\Omega_e\hat{\boldsymbol{p}} - k^2\sum_e \int_{\Omega_e} (\boldsymbol{\varphi}\boldsymbol{\varphi}^T)\,\mathrm{d}\Omega_e\hat{\boldsymbol{p}} = \rho\omega^2\sum_e \int_{\Gamma_{e,N}} \boldsymbol{\varphi}U_n\,\mathrm{d}\Gamma_{e,N}\,. \tag{2.29}$$

The sum $\sum_e$ means assembling the global system of matrices out of individual element matrices. In short, (2.29) can be written as

$$\left(\sum_e \boldsymbol{K}_1^e - k^2\sum_e \boldsymbol{K}_2^e\right)\hat{\boldsymbol{p}} = \rho\omega^2\sum_e \boldsymbol{f}^e \tag{2.30}$$

With $\boldsymbol{K}_1^e$ and $\boldsymbol{K}_2^e$ the elemental system matrices and $\boldsymbol{f}^e$ the elemental source vector. After assembly there holds for the global linear system

$$(\boldsymbol{K}_1 - k^2\boldsymbol{K}_2)\hat{\boldsymbol{p}} = \rho\omega^2\boldsymbol{f} \tag{2.31}$$

which is now to be solved for $\hat{\boldsymbol{p}}$. To illustrate the assembly process, the so-called dynamic stiffness matrix $\boldsymbol{A}$ is introduced as

$$\boldsymbol{A} = (\boldsymbol{K}_1 - k^2\boldsymbol{K}_2)\,. \tag{2.32}$$

It is assembled using the individual matrices $\boldsymbol{A}^e$ for each element as visible in (2.36). The goal is to solve the linear system of equations for the weight factors of the basis functions $p$, also called the degrees of freedom. The calculation is usually done for one element at a time. Thereby arises an element matrix $A^{(e)}$ whose individual entries can be computed as

$$A_{r,s}^{(e)} = \int_{\Omega^{(e)}} \nabla\varphi_r\nabla\varphi_s\,\mathrm{d}\Omega^{(e)} - k^2\int_{\Omega^{(e)}} \varphi_r\varphi_s\,\mathrm{d}\Omega^{(e)} \tag{2.33}$$

with $r, s$ the indices of the individual nodes in an element. For 1D linear Lagrange elements, i.e. elements with two nodes each having one degree of freedom per node, a $2\times 2$ matrix results and for 1D quadratic Lagrange elements with three nodes each a $3\times 3$ matrix results. Also the right hand side vector, now called $\boldsymbol{b}$, is computed element-wise using

$$b_r^{(e)} = \rho\omega^2\int_{\Omega^{(e)}}\int_{\Gamma_{e,N}} \varphi_r U_n\,\mathrm{d}\Gamma_{e,N}\,. \tag{2.34}$$

The element matrices are then assembled to a global system matrix. Within an element, the coordinates of the nodes are $[0,\ldots,d]$ with $d$ the dimension of the element. For linear elements, the coordinates are $[0,1]$. For assembly they need to be mapped to the global coordinate system. This is accomplished by using a mapping function, noted as $q$ which maps $[r,s]$ of the elements to $[i,j]$ in the global system: $i = q(e,r)$ and $j = q(e,s)$ with $e$ the number of the element. For the global system matrix there now holds

$$A_{q(e,r),q(e,s)} = A_{r,s}^{(e)} \tag{2.35}$$

for adding the entries of an element matrix to to global matrix. The resulting matrix after assembly is sparse, here for three elements with linear ansatz functions:

$$A_{q(e,r),q(e,s)} = \begin{bmatrix} A_{0,0}^{(0)} & A_{0,1}^{(0)} & 0 & 0 \\ A_{1,0}^{(0)} & A_{1,1}^{(0)} + A_{0,0}^{(1)} & A_{0,1}^{(1)} & 0 \\ 0 & A_{1,0}^{(1)} & A_{1,1}^{(1)} + A_{0,0}^{(2)} & A_{0,1}^{(2)} \\ 0 & 0 & A_{1,0}^{(2)} & A_{1,1}^{(2)} \end{bmatrix} \tag{2.36}$$

For the global right-hand-side vector there holds

$$b_{q(e,r)} = b_r^{(e)} \tag{2.37}$$

what is, again, done for each elemental source vector. With FEniCS, the system is assembled by

```
A = assemble(a)
b = assemble(L)
```

The left-hand side matrix is called the bilinear form since for both trial and test functions the basis functions are used. The right-hand side vector is called the linear form because only basis functions for the test functions are included. Dirichlet boundary conditions can be applied after assembling the system by

```
def boundaryDiri(x):
        return x[0] > (1.0 - DOLFIN_EPS) and x[1] < 0.0 + DOLFIN_EPS
self.bcDir = DirichletBC(self.V, Constant(0.0), boundaryDiri)
A = assemble(a)
b = assemble(L)
self.bcDir.apply(A, b)
```

### 2.2.4 Solving and Convergence

To obtain the solution vector $\hat{p}$, the linear system 2.31 has to be solved. This could be done by inverting the dynamic stiffness matrix $A$ and multiplying it from the left. However, for numeric reasons, directly inverting a matrix is almost never done. [8] and [9] thoroughly discuss that issue: Inverting a matrix is slower and less accurate, especially for rather ill-conditioned matrices, than directly solving the linear system. Solving the system can for example be done using the well-known Gaussian elimination [5, p.509], the Cholesky decomposition, which will be explained in Section 3.3.4, or the LU decompositon. [10] explains the theory behind direct solvers in detail.

The FEM linear system of equations involves sparse matrices, i.e. matrices which are zero in most of their entries. This characteristic makes it possible to use solvers designed for matrices of this kind which are significantly faster than others. [11] gives insight on solvers for sparse systems and iterative solvers. Iterative solvers use a so-called preconditioner, which decreases the condition number a matrix. The lower the condition number, the more stable the system. The condition number of a matrix is defined by the ratio of the largest entry to the smallest entry, according to [12] and [13].

The system of equations can solved for the vector of unknowns by calling

```
self.u = Function(self.V)
U = self.u.vector()
solve(A, U, b)
```

using FEniCS. It automatically chooses a suitable solver for the given problem.

Solving the system for a range of frequencies and averaging the absolute pressure over the domain leads to the frequency response plot, see Figure 2.5.

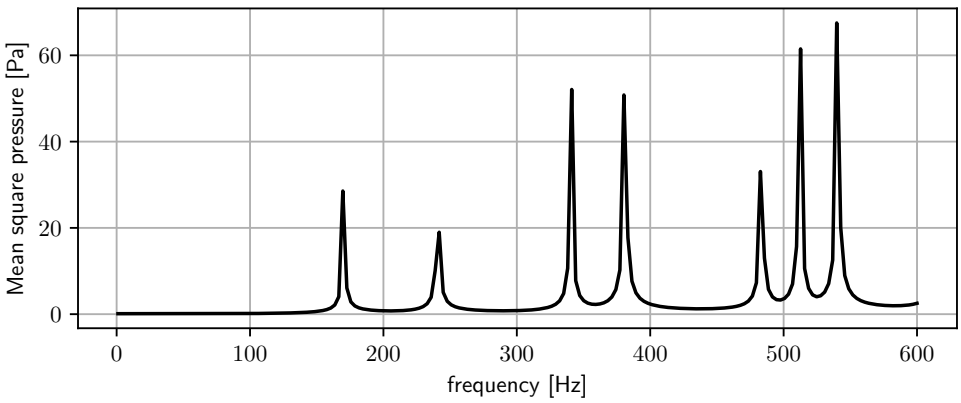



Figure 2.5: The frequency response plot shows the eigenfrequencies of the fluid in the domain. Exciting the fluid with an eigenfrequency leads to a comparatively strong response.

# 3 Gaussian Processes and the Statistical Finite Element Method

In this chapter, an introduction to Gaussian Processes and, building on it, statFEM is given.

## 3.1 Basic Probability Theory

The statistical Finite Element Method (statFEM) is based on Gaussian Processes (GPs) and Bayesian inference. To be able to work with both, some basic understanding of probability theory is necessary.

### 3.1.1 Random Variables and Distributions

speak about linearity somewhere! see Römer lectures

explain PDF/CDF and moments

A random variable $X$ is considered normally, or Gaussian, distributed if $X \sim \mathcal{N}(\mu, \sigma)$ with $\mu$ the mean and $\sigma$ the standard deviation. The shape of the PDF is a Bell curve.

From any distribution, the moments of the random variable can be deduced. Using the PDF there holds for the mean

$$\mu = \mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x)\,\mathrm{d}x \tag{3.1}$$

and for the variance

$$\mathbb{V}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2\,. \tag{3.2}$$

For the covariance between two random variables $X$ and $Y$ there holds (see [5, p. 1434])

$$\mathrm{cov}[X, Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]\,. \tag{3.3}$$

It is clearly visible that $\mathrm{cov}[X, X] = \mathbb{V}[X]$. For the PDF of the Gaussian distribution there holds

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)\,. \tag{3.4}$$

If multiple random variables are observed, which are all normally distributed, they can be aggregated in a so-called multivariate Gaussian disribution. Instead of a random variable it is a random vector which is defined by a mean vector and a covariance matrix. The matrix is named covariance and not variance matrix because not only the variance of the individual random variables but also the covariance among the different variables makes up entries of it. It is similar to auto- and cross correlation, both described in a single matrix. For the PDF of a multivariate Gaussian distribution there holds

$$f_X(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\Sigma|}} \tag{3.5}$$

with $\Sigma$ the covariance matrix, $\boldsymbol{\mu}$ the mean vector and $k$ the dimension, i.e. the number of random variables, of the distribution. An example of a multivariate Gaussian is visible in Figure 3.1. The underlying mean vector and covariance matrix read

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1.5 \end{bmatrix}. \tag{3.6}$$

The smaller the absolute of the non-diagonal entries of the covariance matrix is, the less correlated the individual variables are.

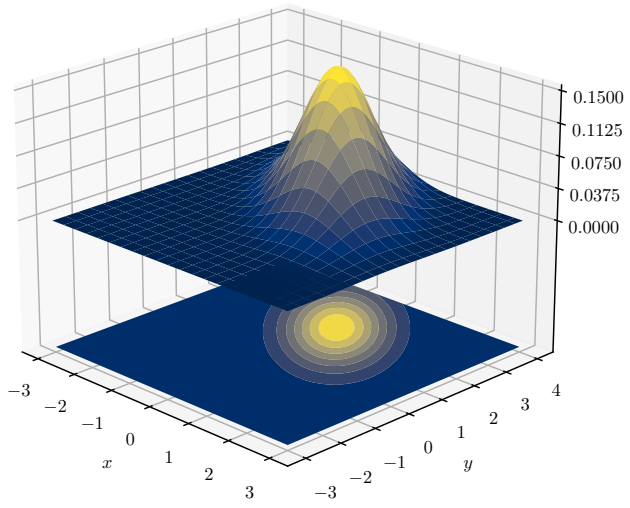



Figure 3.1: A multivariate Gaussian distribution with two dimensions. A strong correlation between the two random variables is visible: No correlation would mean a perfect circle in the lower 2D plot. The more correlated the two dimensions are, the more elliptical the plotted PDF becomes. [14]

### 3.1.2 Bayesian Inference

$posterior = \frac{likelihood x prior}{marginal likelihood}$ [15] explain basics of bayesian inference with bayes rule etc. what is posterior/prior explain what the marginal likelihood is

Bayesian Inference uses Bayes' law to infer a posterior density from a prior distribution and data. The data "updates" the prior. The law states that

$$p(\boldsymbol{u}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})} \tag{3.7}$$

where $p(\boldsymbol{u}|\boldsymbol{y})$ is the posterior, $p(\boldsymbol{y})$ is the likelihood, $p(\boldsymbol{u})$ is the prior and $p(\boldsymbol{y})$ is the marginal likelihood, also called evidence. The posterior to be infered is the new probability density describing the sought quantity after observing data $y$. The data, i.e. measurements, aid understanding the problem what leads to a smaller variance in the posterior as comapared to the prior. The prior density describes

the prior belief of the underlying process. It can be deduced from expert knowledge or from previous measurement. The likelihood describes the probability of the data to be measuered at all given that, in the case of this thesis, the FEM solution holds: "How likely is the measured data to be true, according to the model?" The marginal likelihood (see section 3.2.2 for more detail) serves as a normalization constant and describes how likely it is to measure the observed data.

## 3.2 Gaussian Process Regression

statFEM is based on Gaussian Processes (GPs), a method known from Machine Learning. [16] GPs can be used for regression, i.e. to find a suitable function which describes a given set of points as good as possible, but also for classification and clustering. [17] Gaussian Processes are a class of Bayesian non-parametric models. Non-parametric doesn't mean that there are no parameters involved but rather that there is an infinite number of them. [17] Every realization of a Gaussian process doesn't yield a scalar or vector but a function. One can think of a Gaussian Process as a collection of infinitely many normally distributed random variables, i.e. a generalization of a Gaussian distribution: A vector with infinitely many entries is basically a function. By picking out a finite set of those random variables when discretizing e.g. on an FEM mesh, one obtains a multivariate distribution which is determined by a mean and a covariance matrix. [15, p. 2] Hence, GPs are a method to obtain a distribution over functions: Around a mean function a variance is defined. Possible samples from the GP are functions within these confidence bands. In statFEM, the FEM prior and also the solution are modeled as GPs. This provides an elegant way to incorporate uncertain data in an FEM solution to eventually improve it. To understand how that works, the general idea of regression and the mathematical basis of GPs have to be explained.

### 3.2.1 Linear Regression

Following [16], regression, in most cases linear regression, is a parametric model. This means that there are certain parameters involved in a model which can be changed to match given data as closely as possible regarding some measure of accuracy. A regression is linear if the output of a function depends linearly on its inputs, as in (3.8). [16, p. 19]

$$y(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + \epsilon = \sum_{j=1}^{D} w_j x_j + \epsilon \tag{3.8}$$

$\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ the corresponding vector of input weights and $\epsilon$ is the error between the output of the regression and the true, measured values. The error term is usually modeled as a Gaussian $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$ with zero mean $\mu = 0$ and some defined variance. Because of the addad Gaussian, every solution in the solution space of the function is now not only a scalar but a normally distributed random variable. Therefore, the whole function can be expressed as a probability density:

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x})) \tag{3.9}$$

Note that the mean $\mu(\boldsymbol{x})$ now depends on the input vector. For a linear approximation there holds

$$\mu(\boldsymbol{x}) = w_0 + w_1 x = \boldsymbol{w}^T\boldsymbol{x}\,. \tag{3.10}$$

Unintuitively, using basis function expansion, linear regression can be used to model a non-linear behaviour. If polynomials are used as basis functions, one speaks of polynomial regression. The probability density would read

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\boldsymbol{w}^T \Phi(\boldsymbol{x}), \sigma^2(\boldsymbol{x})) \tag{3.11}$$

with

$$\Phi(\boldsymbol{x}) = [1, x, x^2, x^3, ..., x^d] \tag{3.12}$$

where the complexity of the approximation rises with $d$. [16, p. 20] The model now has to be fit to given data, i.e. the input weight vector has to be defined. This works, for example, by using a maximum likelihood estimation (MLE), also called least squares.

"Regression is used to find a function that rerpesents a set of data points as closely as possible" [17] [17] "A Gaussian process is a probabilistic method that gives confidence for the predicted function"

"GPs allow us to make predictions about our data incorporating prior knowledge"

The following closely follows [15].

A GP is described by its mean function and the covariance function, also called kernel.

### 3.2.2 Gaussian Processes

Gaussian Process regression is another type of regression but ultimately has the same goal as, for instance, a polynomial regression. The difference is that a GP isn't constrained to a parametric model, i.e. it is more flexible. (And therefore prone to overfitting! Really? Check that.) Instead of using a polynomial or some other equation as a basis function, GPs utilize a kernel function which is able to output an infinite amount of different functions. More on kernels later. A GP is more flexible than a standard linear or polynomial regression: A GP regressor finds a suitable function out of infinitely many possible functions. For a linear regression, the function type and therefore basis functions need to be known beforehand in order to achieve accurate results. For a problem of which the function shape cannot be determined beforehand, GPs yield the better results. See an example in Figure 3.2. For both plots the same polynomial regressor and GP were used. GPs are non-parametric: This means that, as opposed

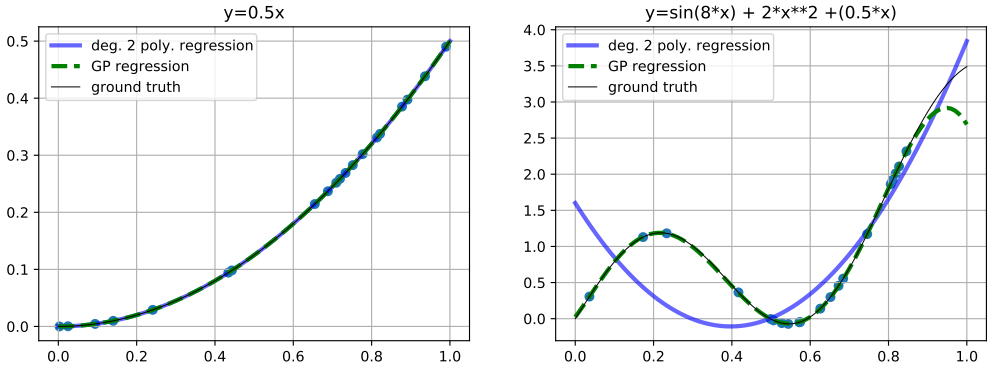



Figure 3.2: On the left: Regression for a 2nd degree polynomial. Both linear regression and GP approximate it fine. On the right: The same GP as on the left is able to model the ground truth accurately, the polynomial regression does not. The GP behaves more flexible in this case.

to standard linear regression where parameters are optimised, a GP does not have a certain number of parameters. Rather, the number of parameters depends on the number of test and training points and could in theory be ininitely large. [16] GPs are constructed using multivariate Gaussians which were explained in Section 3.1.1. Also, as mentioned before, they represent a distribution over functions: The term "process" refers to the generalization from a Gaussian random variable to a random function. [15, p.13] The relationship between multivariate Gaussians and functions will be explained following [18].

The simplest GP is a bivariate Gaussian. For each of the two random variables, i.e. dimensions, a mean and a variance are defined. Also, the covariance between both random variables is given. Now, samples can be drawn from each random variable. As an example see Figure 3.3. In the lower half of the plot the samples are visible in the already described 2D plane. The upper half shows the individual samples compared to each other. Drawing samples from a bivariate Gaussian leads to different results depending on the correlation between the two random variables. The more correlated they are, the smaller the difference between the sampled values in each dimension. Of course, this also holds true

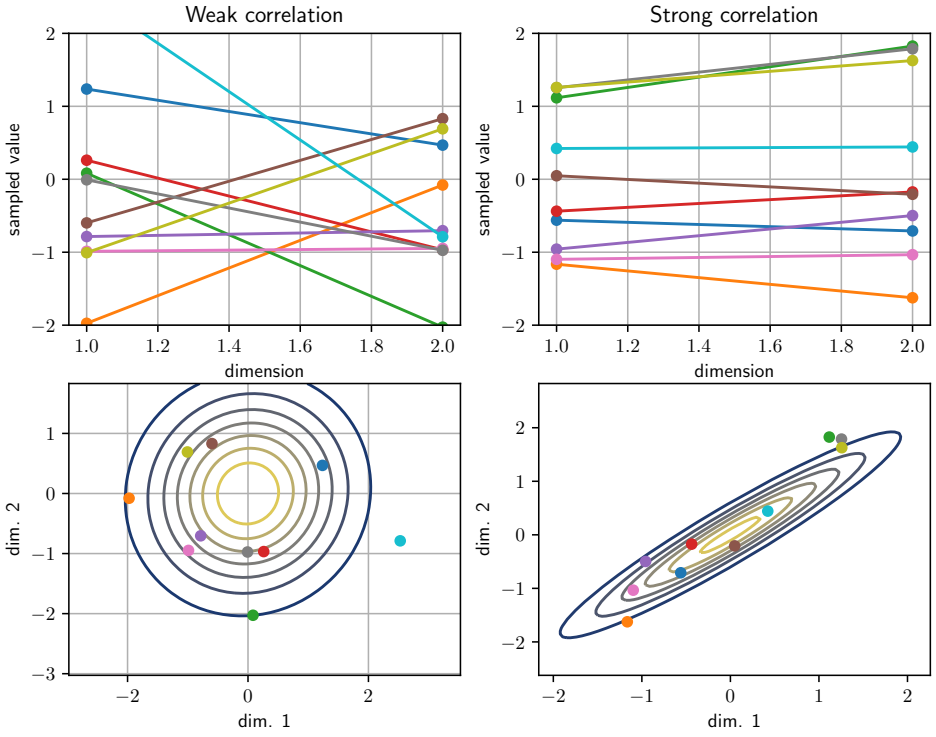



Figure 3.3: A bivariate Gaussian can be interpreted as a GP. The more corellated the random variables are, the closer together the sampled values are. The plots on the top show the samples drawn from the well-known 2D PDF in another way: Sampling from a multivariate Gaussian yields a vector of correlated samples which can be interpreted as a subset of an infinitely dimensional vector, i.e. a function.

for multivariate Gaussians with even more random variables, as visible in Figure 3.4. In theory, a GP can consist of infinitely many correlated random variables and a vector with infinitely many entries is

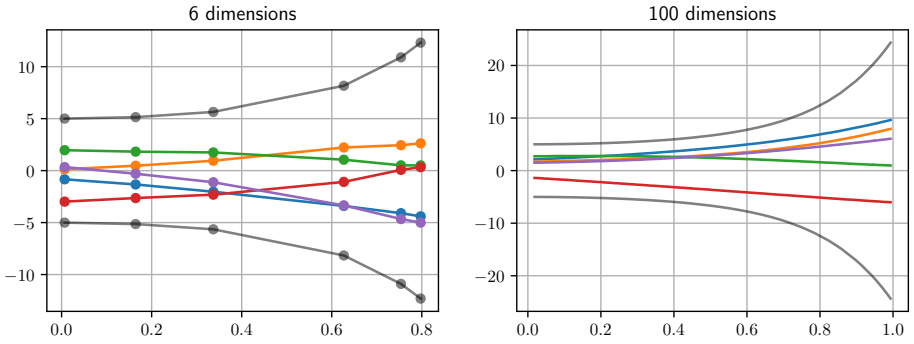



Figure 3.4: The more random variables are used to construct the multivariate Gaussian, the more the result looks like a smooth function. Since each variable is a random variable, there is also a variance available for every dimension, visible in grey.

basically a function. Therefore every sample of a GP is a function and also the variance is given as a function. Drawing samples from a GP can be imagined as drawing correlated samples from a large set of random variables. However, a GP is, at first, defined continuously: To construct a GP, a mean function and a covariance function have to be given. The mean function can be any function, but the covariance function, also called kernel, has to obey certain rules. More on that later. Nevertheless, to use Bayesian inference, i.e. to form a posterior distribution over functions, the continuous GP has to be represented as a multivariate Gaussian, i.e. discretely, again. To understand how and why this is done, the concepts of marginalization and conditioning have to be explained at first.

**Marginalization and Conditioning** Marginalisation means that out of a multivariate Gaussian the single variables can directly be taken out of the mean vector and the covariance matrix. E.g. for a point on the axis described by the GP, a single value can be picked out. A univariate Gaussian comes out by simply using the mean vectors value at that point and the i,i th index of the covariance matrix. This means that the underlying distributions for single variables do not change if a GP is used to describe an arbitrary set of variables [p13]. [Visual Exploration] From a multivariate, in this case bivariate, Gaussian

$$P(X,Y) = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left( \begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right) \tag{3.13}$$

one single random variable, say $X$, can be marginalised out by simply only looking at the entries corresponding solely to $X$:

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX}) \,. \tag{3.14}$$

If the covariance matrix and mean vector are not known, a random variable can be marginalised out of an arbitrary distribution by

$$p_X(x) = \int_y p_{X,Y}(x,y)\,\mathrm{d}y = \int_y p_{X|Y}(x|y)p_Y(y)\,\mathrm{d}y \tag{3.15}$$

which intuitively means that summing the probability for $X$ at every possible $Y$ gives the total probability for $X$ independent of $Y$.

(3.15) already uses the principle of conditioning. The condition operator | yields the probability of one variable under the condition that another variable is true. As an illustration in words one could ask: "What is the probability of event $X$ to happen under the condition, that the event $Y = y$ happens simultaneously?" For the multivariate Gaussian there holds for the conditioning

$$X|Y \sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}) \,. \tag{3.16}$$

Conditioning yields a so-called modified Gaussian distribution which can be imagined as a "cut" through a multivariate Gaussian $P(X, Y)$ at a certain point $y$. A new Gaussian with the dimension reduced by 1 arises as illustrated in Figure 3.5.

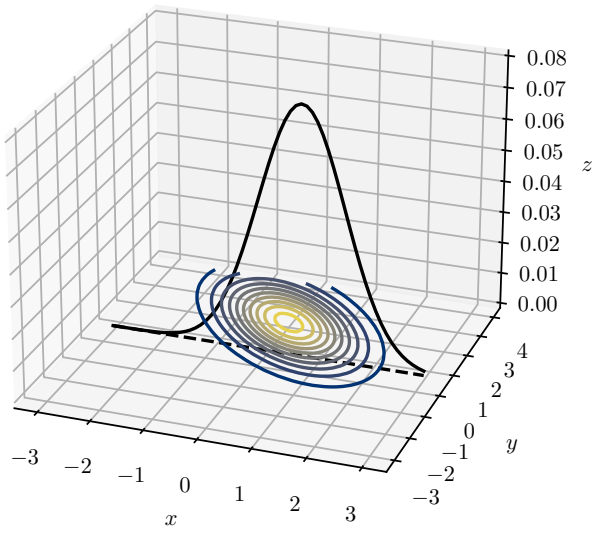



Figure 3.5: Cutting, i.e. conditioning, a bivariate Gaussian leads to a new Gaussian with only one dimension. This example shows the result for a cut at the dashed line. The new Gaussian has a new mean and variance.

### 3.2.3 Kernel Functions

Kernel functions have already been mentioned before. They are what makes a GP regression different from a standard linear regression approach: A kernel function allows to not form a distribution over parameters of some predefined function but rather over the output functions themselves. This means that no assumptions for the shape of the approximated function have to be made. For example, a polynomial could be resembled by a kernel function which is not defined using any polynomial. The same kernel function can be used to resemble a sine wave or anything else. If linear regression is used, there needs to be much more prior knowledge about the shape of the underlying function of the observed data to make accurate predictions: Approximating a sine wave needs periodical basis functions and approximating polynomials of a certain degree needs polynomial basis functions of at least that degree.

A kernel $k$ is a function which takes two points $x \in \mathbb{R}^n$ and $x' \in \mathbb{R}^n$ and maps them into $\mathbb{R}$ [15, p.80], i.e. it returns a scalar which serves as a similarity measure between the two points [17]:

$$k : \mathbb{R}^n \times \mathbb{R}^n \mapsto \mathbb{R}, \quad \Sigma = \mathrm{Cov}(X, X') = k(x, x') \; . \tag{3.17}$$

If a kernel function is evaluated at a finite number of points, say all the test points, a covariance matrix $\Sigma$ is created. It can be used as the covariance of a multivariate Gaussian from which samples can be drawn. The values of each entry are a function over the input points of the kernel function and describe, how much the two points are correlated. [15, p.14]

In a covariance function all the assumptions of the function to be modeled are contained. The kernel defines or rather assesses how near or similar a value of one point is to the one of another. A training point which is close to a test point therefore gives information on how the value at the test point should be. [15, p.79] A kernel function is usually symmetric ($k(x, x') = k(x', x)$) and greater than zero ($k(x, x') \geq 0$). [16, p.481] Free parameters inside a kernel function are called hyperparameters. The term 'hyper' states that they are not parameters of the model itself but only of the kernel which defines properties of the model.

**Stationarity** Kernels which are a function of $x - x'$ are called *stationary* kernels. They are stationary because it doesn't matter what value $x$ and $x'$ have as a starting value, the difference is always the same. Example: $110\mathrm{Pa} - 100\mathrm{Pa} = 10\mathrm{Pa}, 1010\mathrm{Pa} - 1000\mathrm{Pa} = 10\mathrm{Pa}$. A radial basis function is a function which is only a function of $r = x - x'$ and which is therefore isotropic, i.e. there are no rigid motions. r is like a radius so these functions are called radial basis functions. [15, p.80] Covariance matrices always have to be symmetric, i.e. $k(x - x') = k(x' - x)$ because $x - x'$ is a distance measure. The so-called Gram matrix is the function containing $x - x'$ evaluated at a set of points $x_i$. If the function is a covariance funtion, the gram matrix is called covariance matrix.

**Positive Definiteness** A covariance function has to be positive semidefinite, i.e. all eigenvalues are greater or equal zero. A covariance matrix K is positive semidefinite when $Q(v) = v^T K v \geq 0$. Q is the quadratic form. A kernel function is positive semidefinite if it only outputs matrices which are positive semidefinite.

**The Characteristic Length Scale** The number of "upcrossings" in the interval, i.e. the number of crossings of the x axis can be thought of as the characteristic length scale of the kernel function. It can be calculated with [15, Eq. 4.3]. The more upcrossings, the shorter the length scale and the more jagged the function looks. Long length scales lead to rather smooth looking functions.

A comprehensive overview of different kernel functions and their behavior is given in [19] and [17]. Nevertheless, some important kernels will be introduced below.

**Linear Kernel Functions** A general linear kernel is defined as

$$\sigma_b^2 + \sigma^2 (x - c)(x' - c) \; . \tag{3.18}$$

As the name suggests, samples drawn from a GP with a linear kernel are linear functions. The paramter $c$ controls the point which all samples pass. Depending on the added noise, the variance around that point is much lower as in the other intervals. Figure 3.6 shows exemplaric samples. Different to the other kernels introduced here, the linear kernel is non-stationary, i.e. the absolute position of the two inputs matters.

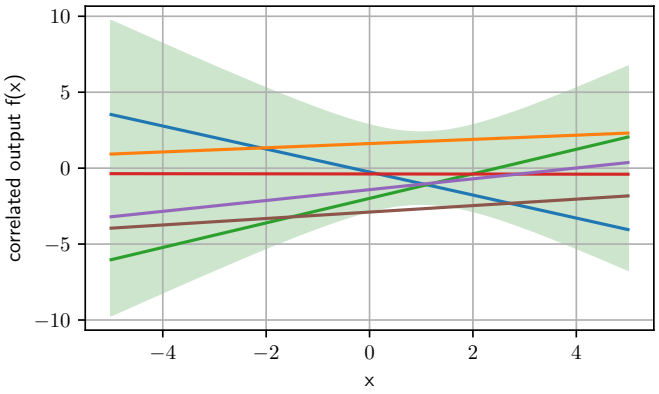



Figure 3.6: Samples drawn from a GP with a linear covariance function. There is a region with relatively low variance through which all samples pass. The poosition can be mived by varying a hyperparameter of the kernel.

**Squared Exponential Kernel Functions** The squared exponential kernel function is the most commonly used kernel function for GPs. It is stationary and infinitely differentiable, hence very smooth. A GP with this kernel function is the same as the above discussed linear regression model with infinitely many basis functions. [15] This makes clear, why GPs are considered non-parametric models. Not the parameters are optimized but the function itself. It wouldn't be possible to learn an infinite amount of parameters. It is defined as

$$k_{SE} = \sigma \exp\left(-\frac{r^2}{2l^2}\right) \tag{3.19}$$

and can be defined as a Python function with

```
def squared_exponential(xa, xb, l, sig):
    sq_norm = -0.5 * scipy.spatial.distance.cdist(xa, xb, 'sqeuclidean') \
      * (1/l**2)
    return sig**2 * np.exp(sq_norm) .
```

[16]

**Periodic Kernel Functions** The periodic kernel is defined as

$$\sigma^2 \exp\left(-\frac{2\sin^2(\pi r/p)}{l^2}\right). \tag{3.20}$$

The parameter $p$ defines the periodicity, i.e. the higher $p$, the larger the period. $\sigma$ and $l$ have the same function as in the squared exponential kernel. [19] It can be used if a periodic behavior in the modeled process is expected. Figure 3.12 shows an example of the generated covariance matrix. The periodicity is visible as a periodic high correlation between the dimensions. Figure 3.8 shows three exemplaric samples from a GP with zero mean and a periodic kernel.

**Dirichlet Kernel Functions**

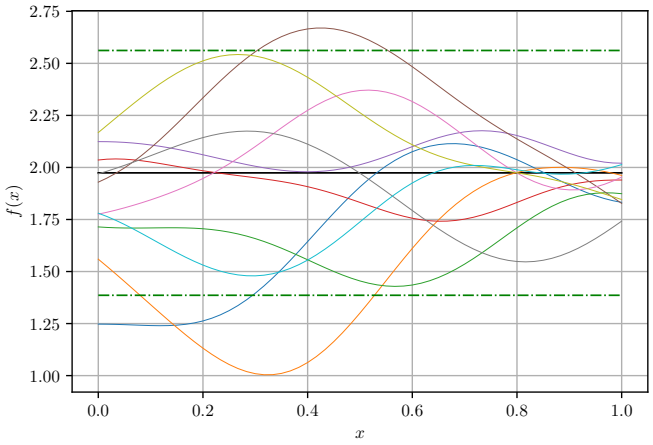



Figure 3.7: The source term GP of $f(x)$ sampled for a Squared Exponential kernel with $l = 0.25$ and $\sigma = 0.3$. $\bar{f}(x)$ in solid black, $2\sigma$ confidence bands in green. The samples are very smooth, which is a charcteristic of this particular kernel.

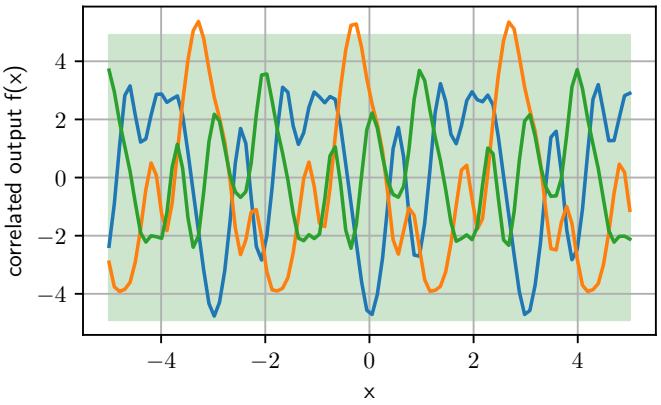



Figure 3.8: Three samples from a periodic kernel. The periodicity is clearly discernible. The variance is constant.

**Matern Kernel Functions** [15] The squared exponential kernel is infinitely differentiable, which means that it behaves very smoothly. This is considered as unphysical by [20]. Therefore, the Matern class of covariance functions [21] is recommended by [20] which is finitely differentiable and therefore less smooth. It is called a "class" because by varying the parameters, very different behaviours can be obtained. The squared exponential kernel is actually a special case of the Matern class.

The general Matern kernel is defined as

$$k_M(r) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu} r}{l} \right)^{\nu} K_\nu \left( \frac{\sqrt{2\nu} r}{l} \right) \tag{3.21}$$

with $\Gamma(\nu) = (\nu - 1)!$ the gamma function and $K_\nu$ a modified Bessel function [22, p.84 ff.], [15]. $\nu$ and $l$ are free positive parameters. For $\nu \to \infty$ (3.21) becomes the squared exponential kernel. The covariance function simplifies for half integer values of $\nu$ and can be expressed as a product of an exponential and

a polynomial. According to [15] $\nu = 3/2$ and $\nu = 5/2$ are commonly used for GPs; the simplest Matern kernel is obtained with $\nu = 1/2$:

$$k_{M;\nu=1/2}(r) = \sigma^2 \exp\left(-\frac{r}{l}\right), \tag{3.22}$$

$$k_{M;\nu=3/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right), \tag{3.23}$$

$$k_{M;\nu=5/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right). \tag{3.24}$$

(3.23) - (3.24) are used in this thesis. Figures 3.9 - 3.11 show the differences between the different choices for $\nu$: The higher it is chosen, the smoother the GP becomes and the closer it gets to the squared exponential kernel.

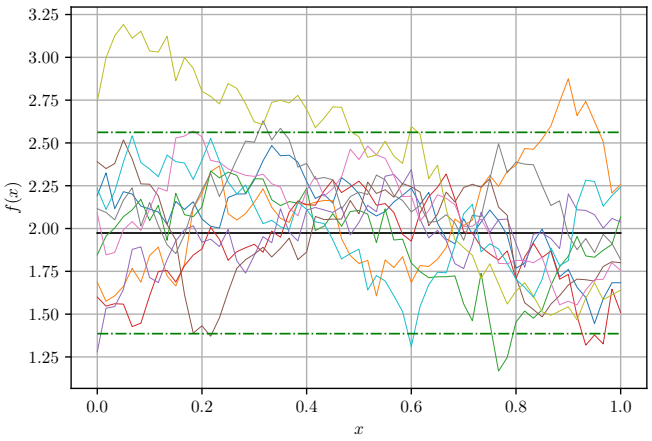



Figure 3.9: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 1/2$. $\bar{f}(x)$ in solid black, $2\sigma$ confidence bands in green. For this $\nu$ the samples behave very rough.

**Combination of Different Kernel Functions** Different kernels can now easily be combined by adding or multiplying individual functions. For example, a periodic kernel can be superimposed on a linear kernel by addition which would lead to a periodic kernel with a mean that changes linearly. Multiplying two linear kernels leads to sampled functions which are quadratic. Multiplying across dimensions, i.e. for example

$$k_{new}(x, y, x', y') = k_x(x, x')k_y(y, y'), \tag{3.25}$$

yields functions in which the function values are correlated on two axes. [19]

### 3.2.4 Bayesian Inference

Now that the mathematical prerequisites have been discussed, the actual Bayesian Inference with GPs can be explained. The goal is condition a prior GP on observed data. This means that the variance over the mean function in a GP should become lower at positions where measurements have been taken,

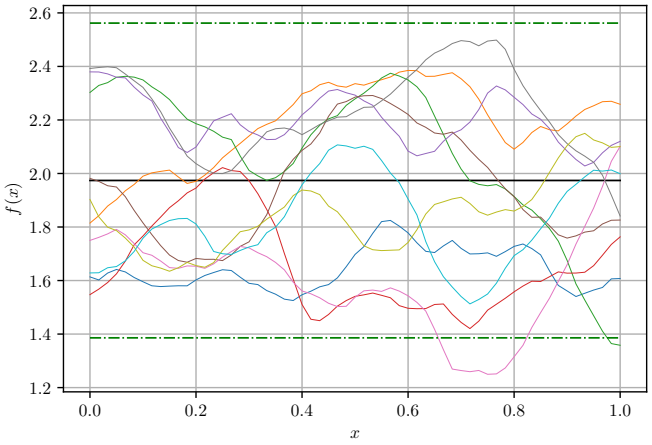



Figure 3.10: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 3/2$. $\bar{f}(x)$ in solid black, $2\sigma$ confidence bands in green. For this $\nu$ the samples behave less rough.

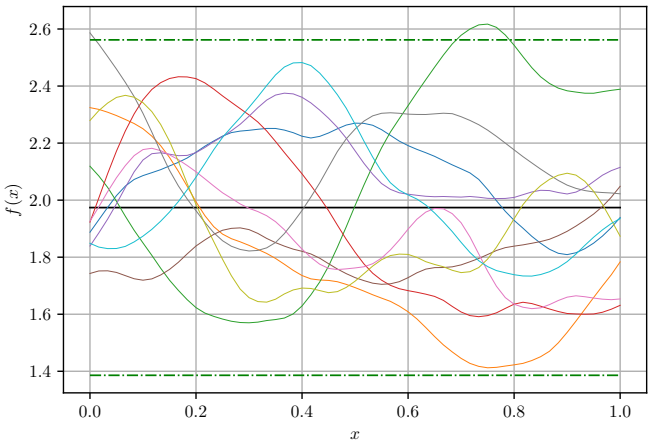



Figure 3.11: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 5/2$. $\bar{f}(x)$ in solid black, $2\sigma$ confidence bands in green. For this $\nu$ the samples behave almost as smooth as in the Squared Exponential kernel.

since at these locations the value of the measured quantity is known accurately. Also, the mean function will change to that now well known value. At first, knowledge of the measured quantity is collected as information within the Bayesian prior.

**Prior** In a differential equation, such as the ones used in this thesis, one or more spatial parameters or the source term can be considered as random. Assuming a probability distribution for these parameters according to already given knowledge about and solving the PDE leads to the so called prior. The prior, because it is modeled as random, already involves a mean and a covariance for the parameters. Constructing the prior as a GP already at this point is a useful approach to make the further steps easier, since a covariance matrix and a mean function are already known, which are later used for inference.

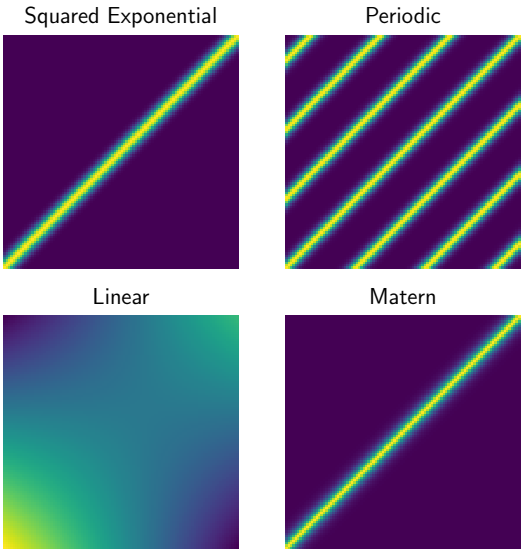



Figure 3.12: The covariance matrices for the different kernel functions. 100 linearly spaced test points between $-5$ and $5$ were used as inputs. The Squared Exponential and Matern kernels yield similar results. For the periodic kernel, the periodicity is also visible in the covariance matrix. For the covariance matrix derived from the linear kernel, the region of lower variance is visible.

Therefore, the mean and covariance of the random parameters are constructed as mean and covariance *functions* within the framework of a GP, which in turn yields a prior distribution over functions. To obtain the covariance matrix, a kernel function suitable to the problem has to be chosen. At this point, the hyperparameters are chosen manually according to prior belief and knowledge: They can for example be adjusted to make the function behave more smoothly or to change the confidence bands in which possible functions may appear. Then, evaluating the kernel function at a set of test points, which could for instance be the nodes in an FEM mesh, leads to the covariance matrix. The mean vector is chosen manually, as well. To visualise the prior, samples from it can be drawn. The following closely follows [15, A2]. To obtain a sample from the multivariate Gaussian which represents the GP, at first the Cholesky decomposition of the covariance matrix $\Sigma = LL^T$ has to be computed. Next, a sample from a standard multivariate normal distribution $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ is drawn. The final sample $\boldsymbol{x}$ can now be obtained using

$$\boldsymbol{x} = \boldsymbol{m} + L\boldsymbol{u} \tag{3.26}$$

with the prescribed mean vector $\boldsymbol{m}$. With Python and numpy, the sample is obtained with

```
x = np.random.multivariate_normal(
                        mean = mean, cov=Sig,
                        size=1)
```

The numpy function does the Cholesky decomposition internally.

### Occam's Razor

**Posterior after observation** p16: The prior is conditioned on the observations. It would also be possible, but computationally demanding, to sample many many functions out of the GP and only take those which go through the training points.

As observed data, also called training data, is available, the prior GP can be conditioned on those in order to compute a posterior distribution. If the observations are assumed to be noise-free, a joint distribution of the prior GP (here $X$) (with the kernel function evaluated at the chosen test points to form the covariance matrix) and the training data GP (here $Y$). The training data, even though noise-free, is a GP and therefore a multivariate Gaussian, because also here a covariance is formed by evaluating the kernel function at the sensor locations. The measured values form the mean vector. The joint density can be expressed as

$$P(X,Y) \sim \mathcal{N}(\boldsymbol{\mu},\Sigma) = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{\mu}, \begin{bmatrix} K_{XX} & K_{XY} \\ K_{YX} & K_{YY} \end{bmatrix}\right) . \tag{3.27}$$

The non-diagonal entries in 3.27 are formed by evaluating the kernel function with both the test and training points as arguments: For $r = x - x'$ there holds $r = \boldsymbol{x} - \boldsymbol{y}$ with $\boldsymbol{x}$ and $\boldsymbol{y}$ the vectors of the spatial coordinates for test and training points. This density now has to be conditioned on the training data. This means that all samples from the new, conditioned, density will pass through the training points. This process can be thought of as sampling one function at a time from 3.27 and collecting all samples which pass through the training points, discarding all other ones. Of course, mathematically, this can be done using the already shown equations for conditioning densities, see Section 3.2.2.

For noisy observations, i.e. there is an uncertainty in the measurement method, a diagonal noise term is added to the observation GP. For the joint distribution there now holds

$$P(X,Y) \sim \mathcal{N}(\boldsymbol{\mu},\Sigma) = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{\mu}, \begin{bmatrix} K_{XX} & K_{XY} \\ K_{YX} & K_{YY} + \sigma_n^2 I \end{bmatrix}\right) \tag{3.28}$$

with $\sigma_n^2$ the variance of an independent and identically distributed Gaussian noise and $I$ the identity matrix. Conditioning on $Y$ now yields

$$\begin{aligned} X|Y \sim \mathcal{N}(&\mu_X + K_{XY}[K_{YY} + \sigma_n^2 I]^{-1}(Y - \mu_Y), \\ &K_{XX} - K_{XY}[K_{YY} + \sigma_n^2 I]^{-1} K_{YX}) . \end{aligned} \tag{3.29}$$

The conditioned distribution is a GP again: Conditioning therefore basically means interpolating between and extrapolating from the sensor locations. [18] Figure 3.13 shows both the GP prior and posterior. It is visible that the prior on the left has a zero mean. Arbitrarily many samples can be drawn from it which themselves do not necessarily have a zero mean, as well. If that prior GP is now conditioned on training points, two in this case, the mean of the newly formed GP will follow the training points and therefore deviate from the prior mean. All samples drawn from the posterior GP will go through the training points. It is also visible, that the variance decreases in the surrounding of a measurement: Exactly on the (here noise-free) training points it is zero and in between them it is still much lower than the prior variance. Finding a new mean and variance between training points is called interpolation and, as visible, yields in solutions with relatively low variance. However, extrapolation yields results with

higher variance: If only on one side of the domain a training point is available,the variance and the mean quickly become those of the prior again.

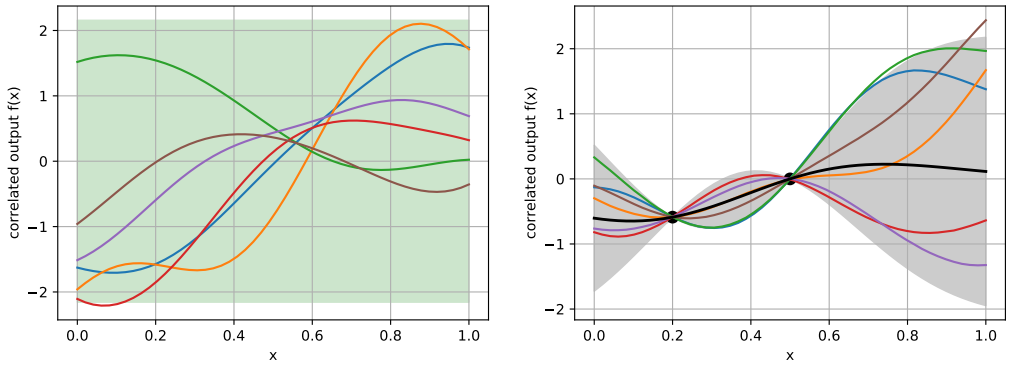



Figure 3.13: The prior GP on the left and the posterior GP after conditioning on two training points on the right. The variance approaches zero at the noise-less training points. The further away from the training points, the closer the variance is back at the prior variance.

Figure 3.14 shows the new covariance matrix after conditioning. It is visible that the variance is highest in the top right corner, which corresponds to values around $1.0$ and is in agreement with Figure 3.13: The entries of the covariance matrix become lower in regions around a sensor location.

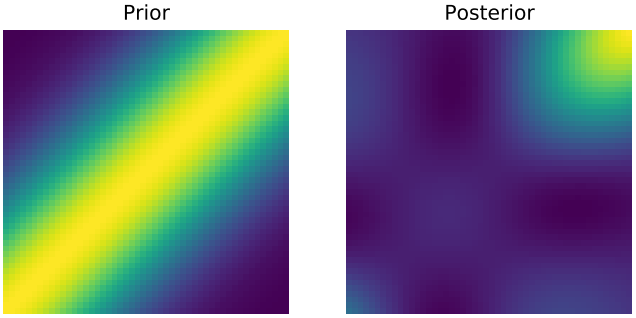



Figure 3.14: The covariance matrix before and after conditioning. The lighter areas show higher (co)variance. The part of the domain on the right, where no observations were made, is visible as entries with high variance in the top right.

As introduced in Section 3.2.2, the marginal likelihood can also be computed. It is used in the procedure for optimizing the hyperparameters of the chosen kernel function. Marginalizing over

## 3.3 The Statistical Finite Element Method

The FEM is based on a model in form of a differential equation which is used to describe some kind of phenomenon, in the case of this thesis vibroacoustics, as accurately as possible. However, every model is only an approximation of reality which gives rise to a model inadequacy error: There is some unknown

physics hidden between the model and reality. By collecting data, basically measuring reality with some kind of measurement error, that discrepancy $d$ between model and reality can be infered. That is what statFEM aims to do: It finds a new model, based on data, which solves the model inadequacy of the FEM.

**The Statistical Generating Model** Measuring data always involves a measurement error: One can never measure any physical quantity without some measurement noise. Therefore, the so-called statistical generating model [23], [24] for a vector of $n_y$ measurements $\boldsymbol{y} \in \mathbb{R}^{n_y}$,

$$\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{e} = \rho \boldsymbol{P}\boldsymbol{u} + \boldsymbol{d} + \boldsymbol{e}\ , \tag{3.30}$$

consists of response of the true underlying process $\boldsymbol{z} \in \mathbb{R}^{n_y}$ at the measurement points and the measurement noise $\boldsymbol{e} \in \mathbb{R}^{n_y}$. The true process $\boldsymbol{z}$ can further be expressed as a combination of the used model $\boldsymbol{u} \in \mathbb{R}^{n_u}$, which tries to describe the true process, and a model discrepancy error $d$ which accounts for the model not being a completely accurate representation of the true physics. The model is scaled using the scaling parameter $\rho \in \mathbb{R}$. It can be evaluated at $n_u$ points which are determined by the FEM mesh but only its evaluations at the measurement points are taken into account by using the projection matrix $\boldsymbol{P} \in \mathbb{R}^{n_y \times n_u}$. $n_u$ is the number of the so-called degrees of freedom of the domain, i.e. the number of nodes on which a solution is approximated. $e$ is modeled as a multivariate Gaussian

$$\boldsymbol{e} \sim p(\boldsymbol{e}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{C_e}) \tag{3.31}$$

with zero mean and the covariance matrix $\boldsymbol{C_e} = \sigma_e^2 \boldsymbol{I}$ which adds the measurement noise to each entry of $\boldsymbol{z}$. The model discrepancy error is, in order to account for a possibly complex behavior, modeled as a GP

$$\boldsymbol{d} \sim p(\boldsymbol{d}|\sigma_d, l_d) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{C_d}) \tag{3.32}$$

which $\sigma_d, l_d$ the unknown parameters of a squared exponential kernel function. Evaluating it at the measurement positions, the GP is represented by a multivariate Gaussian with a covariance matrix $\boldsymbol{C_d} \in \mathbb{R}^{n_y \times n_y}$.

**Basic statFEM procedure** statFEM can basically be broken down into three major steps [23]. The first one is applying Bayesian inversion to the FEM part of the solution. The prior GP is constructed and evaluated before data is introduced. Equations are derived which incorporate the data in order to compute a posterior density for the FEM solution GP. The equations for that solution are dependent on certain hyperparameters. These are estimated in the second step: For each of the hyperparameters a posterior density is calculated using a prior and the marginal likelihood. The third and last step is in principle finding a proper mesh size which is able to describe data the best. This last step may appear rather unimportant since usually in FEM the consensus is that a finer mesh leads to a more accurate result. However, it will be demonstrated that the optimum element size strongly depends on the value of certain hyperparameters estimated in step 2 and that there is a correlation between relatively inaccurate statFEM results and a too small element size.

### 3.3.1 Posterior of the FEM Forward Model

**Assembling the FEM System Matrix** The FEM assembly and solving is handled by Fenics. To compute the system matrix and to later solve for the vector of unknowns, the PDE has to be defined in a

variational formulation as a boundary value problem. At first, the FEM solver is initiated by defining the domain, the wished number of elements and the kind of elements to be used. In this example, a 1 dimensional mesh with Lagrange basis functions is used.

```
def __init__(self, nMC = 200):
  self.dom_a = 0.0 #domain boundaries
  self.dom_b = 1.0
  self.ne = 100 #number of elements
  self.mesh = IntervalMesh(self.ne,self.dom_a,self.dom_b) #define mesh
  self.bbt = self.mesh.bounding_box_tree()
  self.coordinates = self.mesh.coordinates() # vertices vector
  self.V = FunctionSpace(self.mesh, "Lagrange", 1) # Function space
```

The variational formulation for the chosen PDE consists of a linear form $L$ and a bilinear form $a$. These need to be specified individually for Fenics. The boundary condition is set to $u_0 = 0.0$ for both boundaries in the 1D mesh. The system matrix $A$ can be returned by calling `A = assemble(a)`. The boundary conditions are applied to $A$ after assambly.

```
def doFEM(self):
        # Define Dirichlet boundary (x = 0 or x = 1)
        def boundary(x):
                return x[0] < (self.dom_a + DOLFIN_EPS) or x[0] > self.dom_b - DOLFIN_EPS
        # Define boundary condition
        u0 = Constant(0.0)
        self.bc = DirichletBC(self.V, u0, boundary)
        # Define variational problem
        self.u = TrialFunction(self.V)
        self.v = TestFunction(self.V)
        # find index of cell which contains point.
        class DoGP(UserExpression):
                def eval(self, value, x):
                        collisions1st = bbt.compute_first_entity_collision(Point(x[0]))
                        value[0] = f_vals[collisions1st] # f is constant in a cell.
                def value_shape(self):
                        return ()
        f_vals = self.draw_FEM_samples(coordinates)
        f_vals = f_vals[0]
        f = DoGP()

        a = dot(grad(self.u), grad(self.v))*dx #variational Problem
        L = f*self.v*dx

        A = assemble(a)
        b = assemble(L)
        self.bc.apply(A, b)
```

```
28
29          U = self.u.vector()
30          solve(A, U, b)
31          return U, A
```

**Generate and Sample From Source Term GP** The differential equation is treated from a Bayesian viewpoint: all parameters are random variables. For dependent parameters, such as a source term $f(x)$ dependent on the spatial coordinate, not a single distribution but a Gaussian Process is applied. Therefore prior to solving the FEM linear system, the parameter $\mathcal{GP}(\bar{f}, C_f)$ is sampled. That sample is evaluated for each cell in the FEM mesh which makes it necessary to assemble the system matrix with that in mind. $\bar{f}$ is set to a prescribed value. $C_f$ is calculated using the prescribed hyperparameters of the chosen kernel function. As an example, a sample from $\mathcal{GP}(\bar{f}, C_f)$ can be drawn for a Matern kernel by calling

```
1 def sample_f(self):
2         c_f = self.matern_log(self.coordinates,
3                 self.coordinates, l=np.log(lf), sig=np.log(sigf))
4         f_mean = (np.pi)**2*(1/5)*np.ones(self.ne+1)
5         fGP = np.random.multivariate_normal(
6                 mean = f_mean, cov=c_f,
7                 size=10)
```

Here, the Matern kernel accepts only the $\log$ of the parameters $lf$ and $sigf$.

**Generate and sample from the diffusion coefficient GP** If the diffusion coefficient $\kappa$ is assumed to be a random variable and dependent on $x$, it can be modeled as a GP as well and there holds

$$\kappa \sim \mathcal{GP}(\bar{\kappa}, C_\kappa) \tag{3.33}$$

**Computing the FEM Prior** Having the GPs for the input parameters defined, the FEM system of equations can be solved. With the input parameters defined as a GP, also the FEM solution is going to be a GP. Therefore,

$$u \sim \mathcal{GP}(\bar{u}, C_u) \tag{3.34}$$

is to be calculated. Multiplying the FEM system matrix by $\boldsymbol{A}^{-1}$ from the left there holds

$$\boldsymbol{u} = \boldsymbol{A}^{-1}\boldsymbol{f}\,. \tag{3.35}$$

[5, p. 1428] states that the expectation operator $\mathbb{E}$ is linear for independent random variables $X_1, X_2, ..., X_n$ what means that

$$\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i]\,. \tag{3.36}$$

A GP can be described by a set of independent normally distributed random variables. Because of the linearity there holds

$$\bar{u} = \mathbb{E}[\boldsymbol{A}^{-1}\boldsymbol{f}] = \boldsymbol{A}^{-1}\mathbb{E}[\boldsymbol{f}] = \boldsymbol{A}^{-1}\bar{f} \tag{3.37}$$

and, acc. to [25], [26], making use of the definition of covariance (3.3),

$$\begin{aligned} C_u &= \mathbb{E}[(\boldsymbol{u} - \bar{u})(\boldsymbol{u} - \bar{u})^T] \\ &= \mathbb{E}\left[\left((\boldsymbol{A}^{-1}\boldsymbol{f}) - (\boldsymbol{A}^{-1}\bar{f})\right)\left((\boldsymbol{A}^{-1}\boldsymbol{f}) - (\boldsymbol{A}^{-1}\bar{f})\right)^T\right] \\ &= \mathbb{E}\left[\left(\boldsymbol{A}^{-1}(\boldsymbol{f} - \bar{f})\right)\left(\boldsymbol{A}^{-1}(\boldsymbol{f} - \bar{f})\right)^T\right] \\ &= \mathbb{E}\left[\boldsymbol{A}^{-1}(\boldsymbol{f} - \bar{f})(\boldsymbol{f} - \bar{f})^T\boldsymbol{A}^{-T}\right] \\ &= \boldsymbol{A}^{-1}\mathbb{E}\left[(\boldsymbol{f} - \bar{f})(\boldsymbol{f} - \bar{f})^T\right]\boldsymbol{A}^{-T} \\ &= \boldsymbol{A}^{-1}C_f\boldsymbol{A}^{-T}\,. \end{aligned} \tag{3.38}$$

There fore there holds for the multivariate Gaussian, which describes the new $\boldsymbol{u} \sim \mathcal{GP}(\bar{u}, C_u)$

$$\boldsymbol{u} \sim p(u) = \mathcal{N}\left(A^{-1}\bar{f}, A^{-1}C_f A^{-T}\right)\,. \tag{3.39}$$

As visible in (3.39), there holds for $\bar{u}$

$$\bar{u} = A^{-1}\bar{f}\,, \tag{3.40}$$

which can be computed using Python by calling

```
def get_U_mean(self):
  u_mean = Function(self.V)
  U_mean = u_mean.vector()
  b_mean = assemble(f_mean*self.v*dx)
  self.bc.apply(b_mean)
  solstd,A = self.doFEM()  # solstd is unused here.
  solve(A,U_mean,b_mean)
  return U_mean .
```

$C_u$ is obtained by calculating

$$A^{-1}C_f A^{-T}\,. \tag{3.41}$$

In Python code there holds

```
def get_C_u(self):
  C_f = self.get_C_f()
  solstd,A = self.doFEM()
  ident = np.identity(len(self.coordinates))
  self.bc.apply(A)
  A = A.array()
  A_inv = np.linalg.solve(A,ident)
  thresh = 1e-16
  C_u = np.dot( np.dot(A_inv,C_f), np.transpose(A_inv))
  C_u = C_u + 1e-16*(self.sigf**2)*ident
  c_u = np.transpose(self.integratedTestF) * C_u * self.integratedTestF
  return C_u .
```

With $\bar{u}$ and $C_u$ the GP for the prior is completely defined.

**Find Prior Mean and Variance** Once the prior is computed, it can be sampled. The mean is already given but the parameters of the underlying GP are unknown. It is possible to evaluate $C_u$ for the standard deviation with

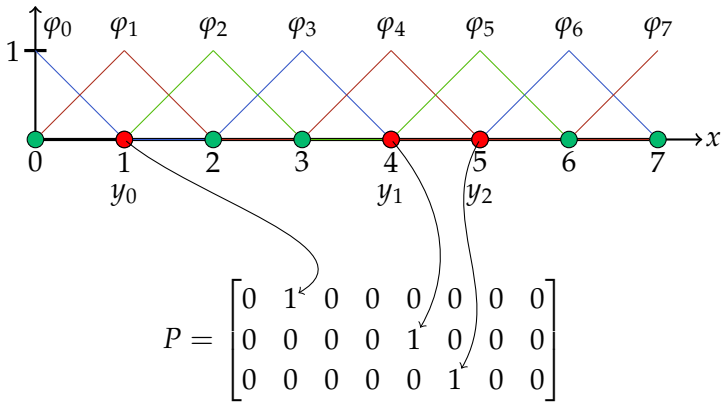
$$\sigma_u = \mathrm{diag}(C_u). \tag{3.42}$$

This yields the diagonal entries of the covariance matrix in a vector which now represents the variance at all test points. In Python

```
C_u_sig = np.sqrt(np.diagonal(C_u))
```

solves (3.42).

**The Projection Matrix $P$** The projection matrix $P \in \mathbb{R}^{n_y \times n_u}$ is constructed by evaluating all $n_u$ FEM ansatz functions of the mesh at the $n_y$ measurement positions $y_i$. For each node there is a corresponding ansatz function which can be evaluated all over the domain. The result is a sparse matrix with most entries zero and only those entries non-zero which correspond to the ansatz functions $\Phi_i(\boldsymbol{x})$ associated with the finite element in which the measurement positions lie. The matrix is used below to project the matrices resulting from the FEM prior onto the coordinates of the observations. Figure 3.15 shows how the individual entries of $P$ are created for a 1D problem. Observation points, i.e. the training points, are displayed in red whereas all other points, i.e. test points, are displayed in green. The hat functions are 1 only at one single node. This computation for linear ansatz functions can easily be implemented in




$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.43}$$

Figure 3.15: The matrix $P$ contains the values of the FEM ansatz functions belonging to the nodes on which observations are available. Since linear Lagrange elements yield 1 at their respective node number and 0 on all other nodes, $P$ consists of only 1s and 0s.

Python using Fenics:

```
def getP(self,y_points):
  ny = len(y_points)
  P = np.zeros((ny, self.ne+1), dtype = float) #with ne the number of
    elements
  for j,point in enumerate(y_points):
```

```
    x = np.array([point])
    x_point = Point(x)
    bbt = self.mesh.bounding_box_tree()
    cell_id = bbt.compute_first_entity_collision(x_point)
    cell = Cell(self.mesh, cell_id)
    coordinate_dofs = cell.get_vertex_coordinates()
    values = np.ones(1, dtype=float)
    for i in range(self.el.space_dimension()):
      phi_x = self.el.evaluate_basis(np.array(i),x
        ,np.array(coordinate_dofs),cell.orientation())
      P[j,cell_id+i] = phi_x
  self.Pu = np.dot(P,self.U_mean)
  return P
```

**Calculate $C_e$** $C_e$ is the measurement error term. It is defined in Python by

```
def get_C_e(self,size):
        sige_square = 2.5e-5
        C_e = sige_square * np.identity(size)
        return C_e
```

**Calculate $C_d$** The model discrepancy term $C_d$ is modeled as a GP with mean $\bar{d} = 0$. Therefore it can be obtained by calling

```
def get_C_d(self,y_points,ld, sigd):
        C_d = self.matern_log(y_points, y_points, l=ld, sig=sigd)
        return C_d
```

The points in the domain and the hyperparameters have to be provided in order to calculate the covariance function. The latter ones are not yet known and need to be estimated, see below for a thorough explanation on that.

### 3.3.2 Inference of the Posterior Density

To infer the posterior density, observed data is necessary. These can be obtained by either actually measuring a physical process or by generating synthetic data which allows developing and improving the statFEM model already before having to set up an experiment and take real measurements beforehand. Sampling from a synthetic source or from the real physical response yields the observation vector

$$\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{e} \tag{3.44}$$

with $\boldsymbol{z}$ the real response and $\boldsymbol{e}$ the measurement error which is also a part of $\boldsymbol{y}$ for the synthetic observations. It can be seen that the model mismatch error $\boldsymbol{d}$ is, as opposed to (3.30), not part of the equation because it is only part of the FEM modeling error. The data is observed on multiple measurement points $y_i$ throughout the domain. It can be chosen if the data is assumed to be deterministic or to be random.

For the deterministic case only one value is measured for each measurement point. In the random case many measurements are taken per point what leads to a probability distribution for each point. Now, Bayes' rule can be applied to update the prior with the observations [23]:

$$p(\boldsymbol{u}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u}) \; . \tag{3.45}$$

The marginal likelihood is, in form of a normalization term, going to be reintroduced in the end. Both densities $p(\boldsymbol{y}|\boldsymbol{u})$ and $p(\boldsymbol{u})$ are considered multivariate Gaussian, see (3.30) and (3.5):

$$p(\boldsymbol{y}|\boldsymbol{u}) \sim \mathcal{N}(\rho P\boldsymbol{u}, C_d + C_e) \tag{3.46}$$

$$p(\boldsymbol{u}) \sim \mathcal{N}(\bar{\boldsymbol{u}}, C_u) \tag{3.47}$$

That property makes the multiplication simple, again excluding the marginal likelihood:

$$p(\boldsymbol{u}|\boldsymbol{y}) \propto \exp\Big((\rho P\boldsymbol{u} - \boldsymbol{y})^T (C_d + C_e)^{-1} (\rho P\boldsymbol{u} - \boldsymbol{y})\Big) \exp\Big((\bar{\boldsymbol{u}} - \boldsymbol{u})^T C_u^{-1}\Big) \tag{3.48}$$

Following [23], with

$$\begin{aligned} B &= \rho^2 P^T (C_d + C_e)^{-1} P + C_u^{-1} \\ \boldsymbol{a} &= \rho P^T (C_d + C_e)^{-1} \boldsymbol{y} + C_u^{-1} \bar{\boldsymbol{u}} \end{aligned} \tag{3.49}$$

there holds

$$\begin{aligned} p(\boldsymbol{u}|\boldsymbol{y}) &\propto \exp\Big(\boldsymbol{u}^T B\boldsymbol{u} - 2\boldsymbol{a}^T\boldsymbol{u} + ...\Big) \\ &= \exp\Big((B^{-1}\boldsymbol{a} - \boldsymbol{u})^T B (B^{-1}\boldsymbol{a} - \boldsymbol{u})\Big) \; . \end{aligned} \tag{3.50}$$

Comparing 3.50 to 3.5 it can be argued that 3.50 can be normalized to

$$p(\boldsymbol{u}|\boldsymbol{y}) = \frac{1}{\sqrt{(2\pi)^{n_u}|B|}} \exp\Big((B^{-1}\boldsymbol{a} - \boldsymbol{u})^T B (B^{-1}\boldsymbol{a} - \boldsymbol{u})\Big) \tag{3.51}$$

The result of this operation is

$$p(u|y) = \mathcal{N}(\bar{u}_{|y}, C_{u|y}) \tag{3.52}$$

with

$$\bar{\boldsymbol{u}}_{|y} = C_{\boldsymbol{u}|\boldsymbol{y}} \left(\rho P^T (C_d + C_e)^{-1} y + C_u^{-1} \bar{u}\right) \tag{3.53}$$

and

$$C_{\boldsymbol{u}|\boldsymbol{y}} = \left(\rho^2 P^T (C_d + C_e)^{-1} P + C_u^{-1}\right)^{-1} \; . \tag{3.54}$$

To avoid having to invert matrices with the size of $n_u \times n_u$, the Sherman-Morrison-Woodbury identity [27] is applied [23] in order to only invert matrices with the size of $n_y \times n_y$ which yields

$$C_{\boldsymbol{u}|\boldsymbol{y}} = C_u - C_u P^T \left(\frac{1}{\rho^2}(C_d + C_e) + P C_u P^T\right)^{-1} P \, C_u \; . \tag{3.55}$$

The same result can be achieved using the approach discussed in 3.2.4. The joint probability for the FEM prior and the observed data states

$$p(\boldsymbol{u}, \boldsymbol{y}) = \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \bar{\boldsymbol{u}} \\ \bar{\boldsymbol{y}} \end{bmatrix}, \begin{bmatrix} C_u & \rho C_u P^T \\ \rho P C_u & \rho^2 P C_u P^T + C_d + C_e \end{bmatrix}\right) \tag{3.56}$$

which can now be conditioned according to 3.29, which ultimately leads to 3.52.

### 3.3.3 Probabilistic FEM Prior

The prior before observation is obtained by computing the FEM response $u_h$ of the PDE given all parameters. The random parameters are modeled as GPs. Thereby result a mean $\bar{u}_h$ and a covariance matrix $C_u$. For this being the prior, the hyperparameters for the source term GP are chosen deterministically. The covariance matrix is obtained by calculating $C_u = A^{-1}C_{par}A^{-T}$ with $A$ the FEM system matrix and $C_{par}$ the covariance matrix of the respective random parameter's GP. The mean can easily be determined by calculating the deterministic mean of the FEM by using only the GP's mean. The resulting response $u_h$ is, since it is discretized and not continuous, a multivariate Gaussian distribution. To check convergence of the FEM model, it can be compared to the exact analytically determined system response $u$. It is assumed that there is a difference between the exact and the calculated FEM response to the true system response. That difference will be reduced by updating the prior with observations. With the basic FEM code set up for a random source term and a deterministic diffusion coefficient, $\bar{u}$ can be obtained by calling

$C_u$ can be obtained by calling

For numerical reasons a small nugget is added to the diagonal of $C_u$. This does not significantly change the result but allows using the Cholesky decomposition. With $\bar{u}$ and $C_u$ at hand the prior GP is defined as $u \sim \mathcal{GP}(\bar{u}, C_u)$. To check if the covariance matrix is correct, a MC approximation $\bar{u}_{MC}$ of the mean can be computed by evaluating the GP $n_{MC}$ times and taking the average. As visible in Figure 3.16, the error of $\bar{u}_{MC}$ converges to zero with a slope of 2 on a logarithmic scale. Therefore, $C_u$ indeed has a mean of $\bar{u}$. The MC samples can be drawn via

```
def samplePrior(self):
  U_mean = self.get_U_mean()
  C_u = self.get_C_u()
  priorGP = np.random.multivariate_normal(
    mean = U_mean, cov=C_u,
    size=self.nMC)
  return priorGP
```

Code for calculating u direct calculation MC approximation from GP samples -error bands

Code for calulating Cu direct calucaltion MC approximation error bands

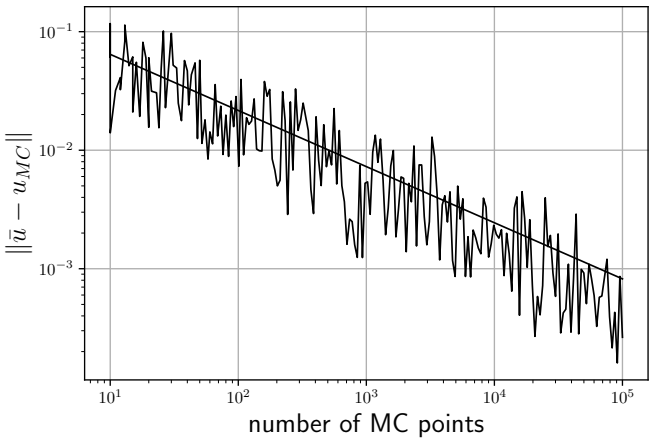



Figure 3.16: Convergence of the error norm for the FEM prior mean with a slope of 2 on a log-log scale.

### 3.3.4 Estimation of Hyperparameters

Below it was already stated that the hyperparameters for computing the model discrepancy covariance matrix $\boldsymbol{C_d}$ are not known. The scaling factor $\rho$ is not known, as well. The measurement data can be used to estimate these parameters, collected in the hyperparameter vector $\boldsymbol{w}$. This is done using Bayes' rule again: There holds

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})}{\int P(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})\,\mathrm{d}\boldsymbol{w}} \tag{3.57}$$

for the posterior. The posterior $p(\boldsymbol{w}|\boldsymbol{y})$ states that the vector of hyperparameters $\boldsymbol{w}$ depends on the measurements data $\boldsymbol{y}$, which makes sense because the data is meant to be used to estimate and optimal $\boldsymbol{w}$. $p(\boldsymbol{w})$ is the prior for the hyperparameters. Obviously, maximizing the likelihood $P(\boldsymbol{y}|\boldsymbol{w})$ maximizes the posterior which basically means finding the set of hyperparameters which are most probable to explain $\boldsymbol{y}$. Comparing the likelihood $P(\boldsymbol{y}|\boldsymbol{w})$ with Bayes' rule as it was applied for the FEM solution prior in 3.45 shows that it has already been introduced before, namely as the marginal likelihood $p(\boldsymbol{y})$ in that case. It makes sense stating that $p(\boldsymbol{y}) = p(\boldsymbol{y}|\boldsymbol{w})$ because, following the statistical generating model 3.30, $\boldsymbol{y}$ is defined using the hyperparameters and therefore its probability function is conditional on $\boldsymbol{w}$. [23, p.12] The denominator in 3.57, which only serves as a normalization constant, can be dropped if only a point estimate of the hyperparameters is needed. This approach is then indeed non-bayesian because no probability density of the hyperparameters is computed. For multiple measurements $y_i$ with a total number of $n_o$ there holds, because it is assumed that individual measurements are independent [15, p.9],

$$P(\boldsymbol{Y}|\boldsymbol{w}) = \prod_{i=1}^{n_o} p(\boldsymbol{y_i}|\boldsymbol{w}) \tag{3.58}$$

which replaces $P(\boldsymbol{y}|\boldsymbol{w})$. If no prior knowledge of the parameters is available, it is possible and even common to use an uninformative prior $p(\boldsymbol{w}) = 1$. [24, p.433] The equation then basically states that $p(\boldsymbol{w}|Y) \propto P(Y|\boldsymbol{w})$ which means that maximizing $P(\boldsymbol{Y}|\boldsymbol{w})$, i.e. the likelihood to measure $Y$ given a set of hyperparameters, yields the optimal vector of hyperparameters for the given data. The marginal likelihood $P(\boldsymbol{y}|\boldsymbol{w})$, marginal because it has been marginalized over the function values $\boldsymbol{u}$ as in 3.2.2, can be expressed as a Gaussian normal distribution

$$p(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\rho\boldsymbol{P}\bar{u}, \boldsymbol{C_d} + \boldsymbol{C_e} + \rho^2\boldsymbol{P}\boldsymbol{C_u}\boldsymbol{P}^T) \tag{3.59}$$

because, as stated above, all components of the statistical generating model are Gaussian. The equation for the corresponding PDF reads, in agreement with Section 3.1.1,

$$p(\boldsymbol{y}|\boldsymbol{w}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{K}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - \rho\boldsymbol{P}\bar{u})^T\boldsymbol{K}^{-1}(y - \rho\boldsymbol{P}\bar{u})\right) \tag{3.60}$$

with

$$\boldsymbol{K} = \boldsymbol{C_d} + \boldsymbol{C_e} + \rho^2\boldsymbol{P}\boldsymbol{C_u}\boldsymbol{P}^T\,. \tag{3.61}$$

To make further calculations easier the exponential function can be removed by taking the negative $\log$ of the function [15, p.113]:

$$-\log p(\boldsymbol{y}|\boldsymbol{w}) = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\boldsymbol{K}| + \frac{1}{2}(\rho\boldsymbol{P}\bar{u} - \boldsymbol{y})^T\boldsymbol{K}^{-1}(\rho\boldsymbol{P}\bar{u} - \boldsymbol{y}) \tag{3.62}$$

This also improves numerical stability because products of possibly very small factors can become smaller than machine accuracy. The log replaces products with sums. Because of taking the negative $\log$, (3.62) now needs to be minimized instead of maximized. The individual terms fulfill different roles:

**Non-identifiability** priors need to be informative, p12 statFEM **Eberly2000** MCMC can lead to wrong sets of parameters or doesn't converge. **Bayarri** [24]

**Cholesky Decompositon** This follows [28, p.93-95]. For numerical stability reasons, $\boldsymbol{K}^{-1}$ should not be calculated directly. Instead, the linear system is solved. Additionally, if $\boldsymbol{K}$ is symmetric and positive-semidefinite, computing the lower triangular matrix using the Cholesky decomposition

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^T \tag{3.63}$$

makes the calculation quicker and yields the also needed determinant of $\boldsymbol{K}$ as a by-product. A generic linear system is defined as $Ax = b$. It is solved for $x$, so the inverse of $A$ is needed. In this case for (3.62) there holds $A = \boldsymbol{K}$ and $b = y$, since the inverse of $\boldsymbol{K}$ is meant to be found. Defining $x = \boldsymbol{K}^{-1}y$ there holds

$$\boldsymbol{K}\boldsymbol{K}^{-1}y = y \tag{3.64}$$

With $\boldsymbol{L}$ the lower triangular matrix of $\boldsymbol{K}$ obtained by

```
L = scipy.linalg.cho_factor(K)
```

the linear system can now be solved for $x = \boldsymbol{L}y$ with

```
K_inv_y = scipy.linalg.cho_solve(L, y) .
```

The determinant of any triangular matrix is defined as the product of its diagonal entries [29]. Therefore the lower triangular matrix $\boldsymbol{L}$ can be used to efficiently calculate the determinant of $\boldsymbol{K}$. There holds

$$\det(\boldsymbol{K}) = \det(\boldsymbol{L})\det(\boldsymbol{L}^T) = \det(\boldsymbol{L}^2) \tag{3.65}$$

and therefore

$$\det(\boldsymbol{K}) = \left(\prod_{i=1}^{n_y} \boldsymbol{L}_{i,i}\right)^2 \tag{3.66}$$

Since in (3.62) the $\log$ likelihood is used, (3.66) can be simplified to a sum:

$$\begin{aligned}\log\det(K) &= \log\det(L^2)\\ &= 2\sum_{i=0}^{n_y} \log L_{i,i}\end{aligned}$$

### Minimization of the Negative log-Likelihood

The best possible set of hyperparameters $w$ for (3.62) to be as small as possible has to be found, i.e. the negative log-likelihood has to be minimized in order to get a model which represents given data the best. That can be achieved by using a gradient based optimizer or by sampling the parameter space with, e.g., a MCMC approach. Both find a point estimate for the set of hyperparameters which lead to the minimal negative log-likelihood. Caution has to be taken for problems in which the negative log-likelihood yields multiple local minima. [30] These and another, more simple, approach to find the minimum are given:

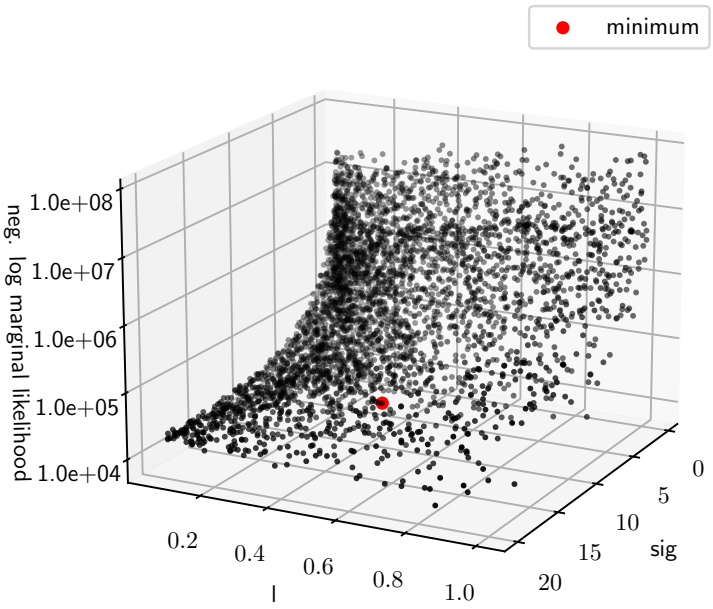



Figure 3.17: See the Animation to get a better understanding of where the minimum lies. Sampling throughout a carefully predefined parameter space and solving for the negative log-likelihood yields a field of solutions. The set of hyperparameters is chosen which yields the smallest negative log-likelihood. The scaling factor $\rho$ is also part of the random sampling but not shown here. The found minimum is shown as a red dot.

**Random Sampling in the Parameter Space** This method is used in this thesis because of its simplicity in implementation. For real-world applications it is not recommended since it is the slowest of all three proposed methods. It works by sampling randomly in the hyperparameter space, solving 3.62 and keeping the hyperparameter set which yields the lowest value. The better the hyperparameter-space is chosen, i.e. the better the prior describes the best combination of parameters already, the better the approximatin considering a fixed number of possible samples. For example, it makes sense to sample all parameters only for positive values, the distance measure $l$ only for values smaller than the domain and the variance $\sigma$ only in the same order of magnitude as the FEM prior. Figure 3.17 gives an impression of how the samples are distributed.

**MCMC** Markov Chain Monte Carlo (MCMC) is a very efficient sampling method. Just as standard MC, it can be used to draw samples from a given distribution. The advantage of MCMC is now that an algorithm finds a relatively small set of samples which is able to describe the distribution very accurately. The result is a distribution for the hyperparameters of which e.g. the mean can be deduced. One common algorithm is the metropolis algorithm. The metropolis algorithm is a special case of MCMC

for a symmetric proposal distribution. According to [23, p.31] the MCMC metropolis algorithm can be implemented with the following steps:

1. Sample a proposal density, e.g. a normal distribution, $q(w|v) = \mathcal{N}(v, \sigma_q^2 I)$

**L-BGFS** C. Zhu, R. H. Byrd, P. Lu, J. Nocedal, Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization, ACM 32Transactions on Mathematical Software (TOMS) 23 (1997) 550–560.

L-BGFS is a gradient based optimizer. It doesn't deliver a distribution for the hyperparameters but only a point estimate. For the optimization with L-BGFS the derivative of (3.62) is needed. It can be computed as follows.

# 4 A statFEM Approach for Vibroacoustics

In the previous chapters, the theoretical basis for an application of statFEM has been built. So now at first, in order to verify the methods and the programmed software, as a simple 1D problem the Poisson equation is going to be solved. Although having no direct connection to acoustics, it is a very common sample problem for numeric methods and thus there are many resources [23] [7] with which the results can be compared.

Having verified the methods, the complexity can be increased to a 2D problem: The Helmholtz equation is solved. A suitable kernel function is chosen and the boundary conditions are applied. The problem is considered to part of the vibroacoustics field because one boundary condition resembles a vibrating plate under uncertainty and the acoustical behavior of the adjoining domain is studied.

## 4.1 Simple 1D example

**Choice of PDE** The Poisson equation

$$-\nabla \cdot \mu(x)\nabla u(x) = f(x) \tag{4.1}$$

is chosen as the governing equation. It is an elliptic partial differential equation (PDE). In this work it is used as a simple 1D example to illustrate how the statistical FEM and especially the Gaussian Process Regression works. The most standard form of it does not, contrary to this example, include $\mu(x) \in \mathbb{R}^+$ which is the diffusion coefficient dependent on the spatial variable $x$. The right-hand side consists of the source term $f(x) \in \mathbb{R}$. Both $\mu(x)$ and $f(x)$ are the free parameters in this case. The equation is solved for the unknown $u(x)$ in the domain $\Omega = (0, 1)$ with the boundary condition $u(x) = 0$ on $x = 0$ and $x = 1$. For a first example there holds $\mu(x) = 1$. $f(x)$ is modeled as a Gaussian Process: [23]

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(\bar{f}(\boldsymbol{x}), c_f(\boldsymbol{x}, \boldsymbol{x}')\right) . \tag{4.2}$$

**Construction of the GP** The mean function of the GP is set to $\bar{f}(x) = 1.0$. For the covariance function at first a squared exponential kernel

$$c_f(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2l_f^2}\right) \tag{4.3}$$

is used with the standard deviation $\sigma_f = 0.1$ and the length scale $l_f = 0.4$. In Python the kernel is directly implemented as a function which takes two lists and the parameters as input variables [16]

Here, the parameters are fixed but in a later example a method on how to infer the optimum position for these will be studied.

Having prepared the mean function and the kernel, which can be considered the prior in a GP regression setting, a sample of the GP can be drawn. For this points have to be chosen on which the kernel is evaluated. These are the test points and, according to [Cirak], correspond to the center of the FEM cells.

This implies that there are as many test points as FEM cells and therefore the coordinates of the FEM mesh can directly be used to compute the covariance matrix. For that (4.3) is evaluated at $\boldsymbol{x} = \boldsymbol{x}'$ with $\boldsymbol{x}$ the vector of test points.

A GP has the marginalization property: if you sample it at a finite number of points it yields a multivariate Gaussian distribution $\mathcal{N}(\bar{\boldsymbol{f}}, \boldsymbol{C}_f)$ with $\bar{\boldsymbol{f}}$ the mean vector and $\boldsymbol{C}_f$ the covariance matrix while still describing the underlying continuous sample. According to [15] sampling from a multivariate Gaussian distribution works as follows: To obtain $n$ samples from the prior, at first $n$ samples of a standard normal distribution, also called Gaussian white noise, $e \sim \mathcal{N}(0, 1)$ have to be drawn. Computing the Cholesky decomposition of the covariance matrix $\boldsymbol{C}_f = LL^T$, which can also be thought of as taking the square root of a matrix, yields the lower triangular matrix $L$. Samples can now easily be drawn from $\boldsymbol{f} = \bar{\boldsymbol{f}} + Le$ which is a multivariate Gaussian distribution with a mean $\boldsymbol{f}$ and a covariance $\boldsymbol{C}_f$.

Sample: Normal distribution times std dev. GP with n points is basically a multivariate gaussian with n dimensions. therefore we need the std dev. for a univariate gaussian thats .

For the observations made in a later step it is assumed that there is some measurement noise. This is modeled as an added variance $\sigma_n^2$ on the diagonal of the prior covariance matrix. An additional effect of this added noise is the improved numerical stability of the covariance matrix which is important for computing the Cholesky decomposition.

### 4.1.1 FEM Prior

The mean, $2\sigma$ confidence bands and realisations of the FEM prior are visible in Figure 4.1. As expected, the general shape of the solutions follows the smooth behaviour of the Laplacian. The variance becomes larger the further away from the Dirichlet zero-boundary conditions.

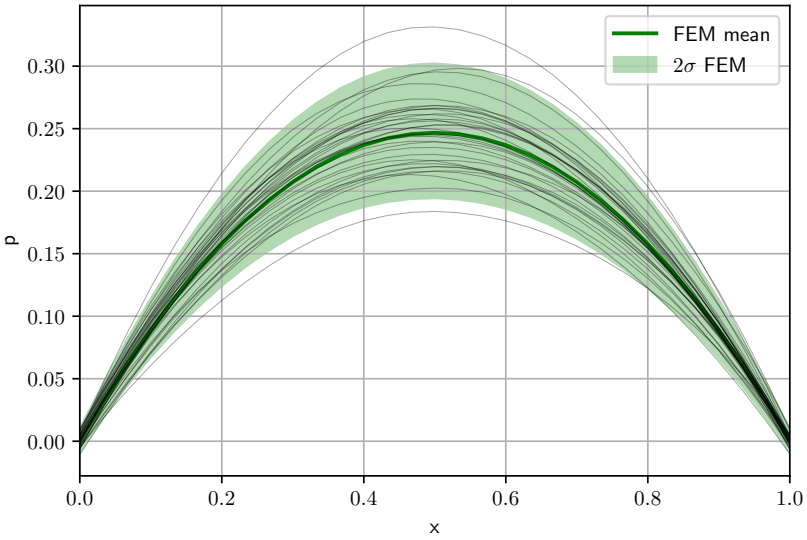
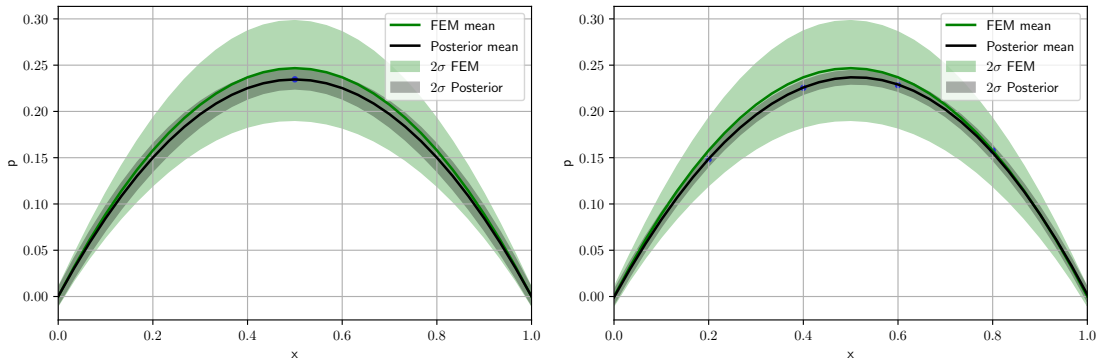



Figure 4.1: FEM Prior for the 1D example. The mean and confidence band as well as samples drawn from the GP are shown. As the boundary conditions imply, both mean and variance are zero on the boundary of the domain.

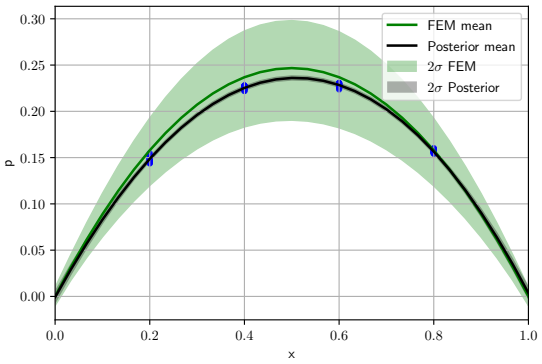
### 4.1.2 Posterior

For the posterior, at first one single sensor location was introduced at whcih one single measurement has been taken. The point has been picked arbitrarily and a measurement noise has been introduced. Already it is visible that the general shape of the FEM prior remains but itis scaled to the data. Also, the variance becomes smaller as there now is evidence on the true solution. Figure 4.2b shows that, if the




(a) One observation at one sensor location. Shape of FEM Prior remains but scaling and variance differ.

(b) Per observation point 1 individual observation was taken with a standard deviation of $2.5e-3$.



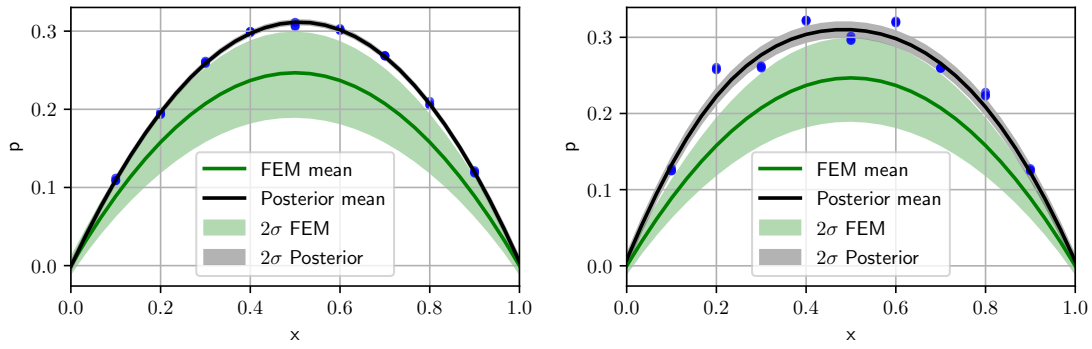

(c) Per observation point 20 individual observations were taken with a standard deviation of $2.5e-3$.

Figure 4.2: The posterior for different measurement scenarios. The more sensor locations are introduced and the more observations are taken, the smaller the variance gets.

observations are made slightly out of the FEM confidence band, a posterior can be inferred which fits to the data. In this example multiple measurement locations were chosen and one single observation has been taken per point. The variance drops significantly and the solution is scaled to fit the data. As Figure 4.2c shows, increasing the number of obervations per sensor leads to a smaller variance.

If now the observations are taken from a sample which does not resemble the overall shape of the FEM prior as closely, model error is introduced. Figure 4.3b shows significantly more variance in the posterior as Figure 4.3a.

Figure 4.4b shows the inferred solution in the case of the data being measured as far out of the FEM prior confidence bands. A prior/data-conflict is visible: The solution isn't scaled correctly according to

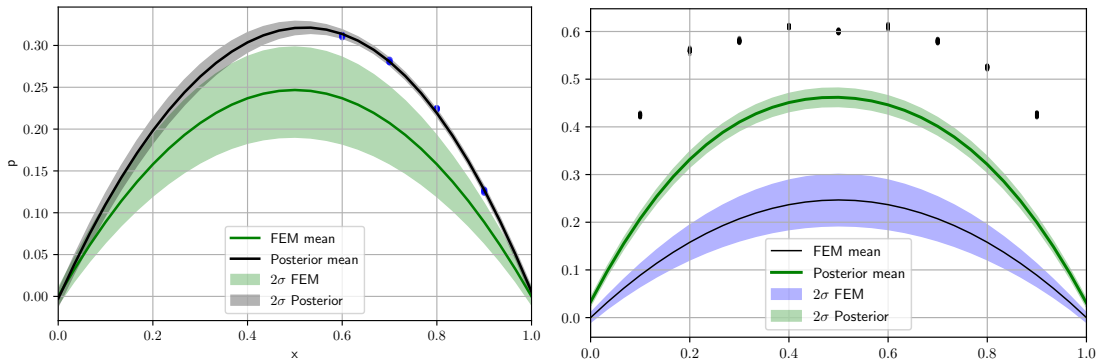



(a) Observing data from a scaled sample of the FEM prior leads to a very small variance.

(b) Introducing significant model error to the observations leads to a higher variance. It could be reduced by using more observations per sensor location.

Figure 4.3: Comparison of the influence of different amounts of model error on the posterior variance.

either data or FEM prior. The data lies way outside the possible FEM solutions. Observing more data would eventually move the curve further in direction of the data because the FEM prior looses importance as the data gains more weight. The posterior confidence band grows.




(a) If data is measured only on one side of the domain the overall shape of the FEM prior still remains but the variance changes according to the presence of data in a region of the domain.

(b) Per observation point 50 individual observations were used with a standard deviation of $2.5e-3$. 30 Elements.

Figure 4.4: Taking observations slightly out of the prior confidence bands works, but the further away the measurements are taken, the more weight is on thr prior and the data is not matched.

## 4.2 2D Vibroacoustics Example

Based on the knowledge of the methods gained in the simple 1D example, the methods are now applied on a 2D vibroacoustics problem. At first, the FEM prior has to be constructed. This is done by introducing

the Neumann boundary on the left side of the domain as a GP. Afterwards, synthetic observation data can be generated and the prior is conditioned in order to form the posterior.

**Generating Fake Data** To generate a posterior, the prior needs to be conditioned on measured data. In this thesis no real measurements have been taken. Instead, fake measurement data is created. One advantage of using fake measurements at first is that the ground truth is very well known: Any number of measurements can be taken at any location in the domain, i.e. there is a whole GP available for the ground truth with which the posterior can be compared. That ground truth can be created in different ways. Here, it is based on the FEM prior: The prior is modified using some predifined parameters which are then to be estimated by statFEM. Scaling the prior GP by a certain factor $\rho$ leads to a new GP without any model error. Superimposing the prior with a function or another GP leads to a new GP with some model error.

From the generated ground truth, observations have to be taken. Here, two methods have been applied to do so: The first one is to choose the sensor locations randomly in the domain. Another method is an "active learning" approach: The first sensor location is chosen randomly. After calculating the posterior GP, the next sensor location is placed at the point of the highest variance. This should lead to a quicker convergence to a low level of variance.

**Experimental Design** The following numerical experiments are conducted and the results are discussed in the next chapters.

- Formation of a FEM prior GP, approximation of variance and mean function
- Conditioning of the FEM prior on 1 sensor location with 1 observation taken from an FEM prior sample. Expected outcome: Variance drop in the region around the sensor, mean function remains unchanged, model inadequacy GP insignificant
- Increase of the number of sensor locations and observations: A further variance drop is expected.
- Same as before but measurements taken on a 75 % scaling of the FEM prior sample. Expected outcome: Scaling factor of $\rho = 0.75$ should be found, model inadequacy GP stays insignificant
- Alteration of the FEM prior sample in a way that a significant model inadequacy is expected. Expected outcome: Increased values for $\sigma_d$ and $l_d$ will be observed.
- Taking observations far beyond the FEM prior variance. An increased variance and a wrong scaling are expected.
- Observation only on a fraction of the domain. The overall shape should still follow the FEM prior. Variance drop in the region with measurements is expected but not in the other parts of the domain.
- A constant Neumann source term is chosen: The problem should reduce to an 1D equivalent.

### 4.2.1 FEM prior

Figure 4.6 shows how the FEM prior is formed. On the left face of the domain, the source term GP is applied. The mean and confidence bands are visible in red. Exciting the domain with the source leads to an uncertain FEM solution in the 2D domain. At $y_c = 0.7$, an arbitrary value, the domain is cut and the

mean and error bands for that position are displayed in green. Clearly, uncertainty in the source leads to uncertainty in the FEM prior. A MC simulation for the variance converges to the calculated variance. Figure 4.5 shows the GP which is used to model the $\bar{U}$ variable in the Neumann source BC. The mean follows

$$\bar{U}(y) = 0.00001 \sin(\pi y) + 0.000005 \sin(2\pi y) \tag{4.4}$$

and the Matern kernel is evaluated with $l = \ln(0.8)$ and $\sigma = \ln(9)$ to form $C_U$, so there holds

$$U \sim \mathcal{GP}(\bar{U}, C_U) \, . \tag{4.5}$$

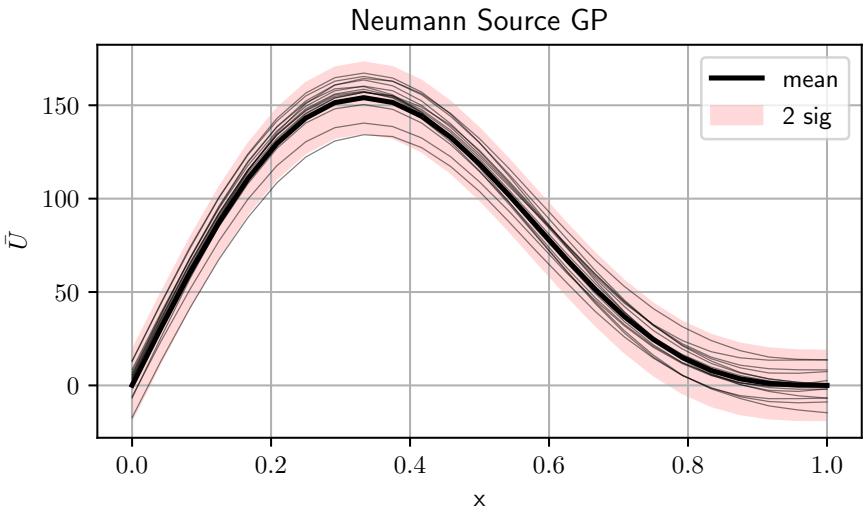



Figure 4.5: The Neumann BC is modeled as a GP with a harmonic mean and a Matern kernel. The mean, the $2\sigma$ confidence band and some drawn samples are shown.

Figure 4.7 shows the FEM prior variance for every point in the 2D solution space for $222Hz$. It is visible that the variance in some regions is higher than in others. The regions of low variance correlate with the troughs of the mode shape.

### 4.2.2 Posterior, observations on prior sample

For the posterior, data has to be generated or measured on which the FEM prior can be conditioned. For that, a ground truth has to be created: In the simplest case it is a sample from the FEM prior. The next step is scaling the sample. Then, to make the problem more complex, the prior sample is changed in a way that it does not resemble the prior GP anymore: Model error is introduced.

**Hypothesis** As stated and described in [15], the variance is supposed to become lower with increasing the number of sensor locations and/or observations. It is expected that this behavior also holds for the Helmholtz problem. Also, it is expected that the mean of the posterior GP gets closer to the observations the more data is introduced. For both observations with and without artificially introduced model error, statFEM is expected to form a suitable posterior which modifies the FEM prior to account for the data. The generated hyperparameters for the posterior GP are expected to reflect the scaling and/or model error.

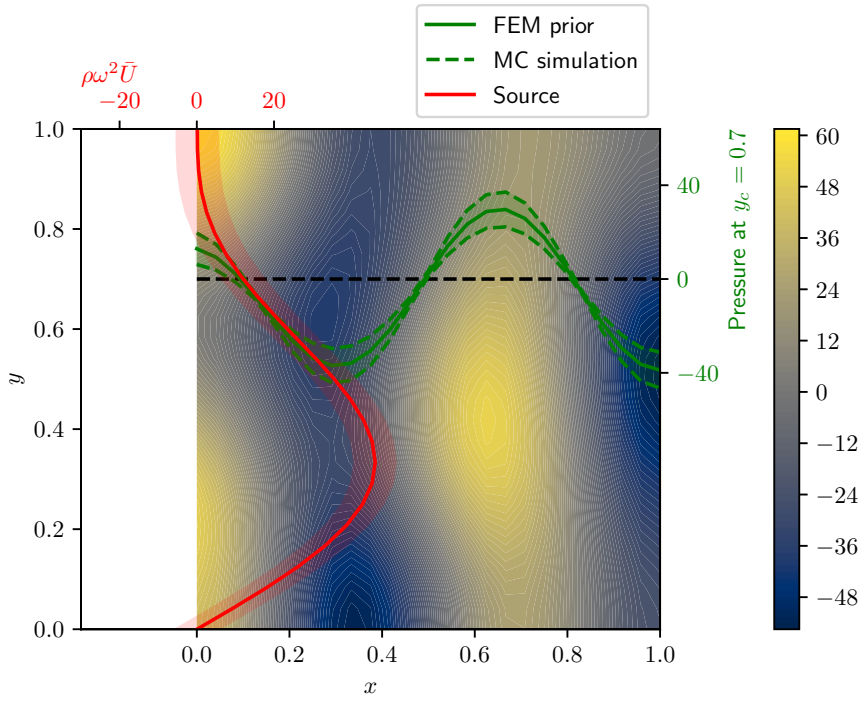



Figure 4.6: The source GP, the mean prior and, at the cut $y = y_c$, the prior FEM GP for 222Hz. The variance around the source leads to a variance in the pressure field which is the lowest where the mean absolute pressure is the lowest.

**Without Model Error** As a first example, a sample of the FEM prior is used as a ground truth from which observations are made. It is basically the "true solution" which is measured. A random measurement error of $e \sim \mathcal{U}(-2.5\text{e}{-3}, 2.5\text{e}{-3})$ is added to the artificial measurement values. At first only one sensor location is chosen and one measurement is taken. Then it is shown that introducing more sensor locations and measurements lead to an overall smaller variance. The sensor locations are chosen randomly. It is expected that the scaling hyperparameter $\rho$ is approximated as close to 1 whereas $\sigma_d$ becomes very small because measuring from the FEM prior means measuring from a possible solution of the prior and therefore no model error. Figure 4.8 shows that this holds approximately.

In Figure 4.8 the variance field in the calculation domain is shown for different combinations of the number of sensor locations $n_p$ and the number of observations $n_o$. It is visible that the variance gets lower the more sensor locations are introduced and the more observations are made. For easier comparison, the $l_2$ norm for the whole domain is given in each plot. Also, it is discernible that where a sensor location is introduced, the variance does not only get lower in the immediate surrounding of the sensor but rather in patterns which have their origin at the sensor location. The more sensors are introduced the more complicated the pattern becomes until no pattern is visible anymore. Even with large $n_p$ and $n_o$, there is still the variance of the Neumann boundary visible in the left side of the domain. Table 4.1 shows the approximated values for $\rho$ for each case. It is always close to 1. The small difference is probably due to the introduced measurement noise. The values for $\sigma_d$ are always very small which corresponds with the fact that no model error was introduced.

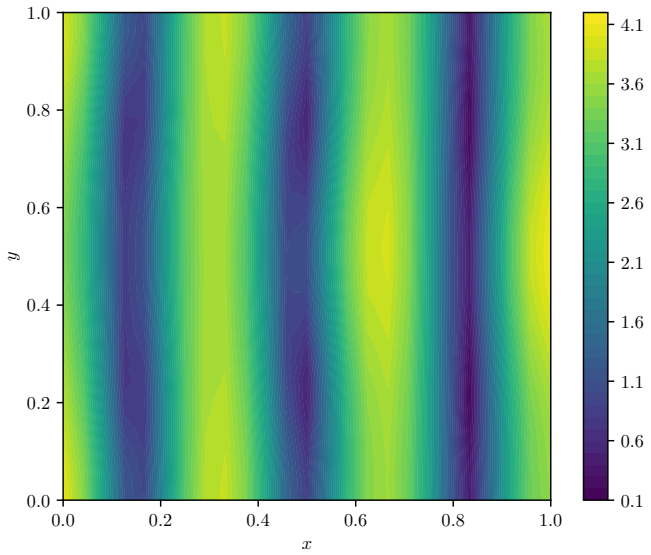



Figure 4.7: The variance of the FEM prior for $500Hz$ at all points in the 2D space. Regions with high and low variance are clearly visible: Regions with high absolute pressure peaks also have a high variance. This implies that the positions of the nodal points in the mode shape are approximated very accurately.

Table 4.1: For a $100\%$ scaling and no introduced model error, $\rho$ is close to $1$ for every case, which is expected for a direct FEM prior sample as ground truth. The model error parameters are close to zero which is expected, because no model error was introduced.

| | $n_o = 10$ | $n_o = 100$ | $n_o = 1000$ |
|---|---|---|---|
| $n_p = 2$ | $\rho = 1.013, \sigma_d = 0.016$ | $\rho = 1.022, \sigma_d = 0.011$ | $\rho = 1.001, \sigma_d = 0.034$ |
| $n_p = 4$ | $\rho = 0.989, \sigma_d = 0.019$ | $\rho = 0.977, \sigma_d = 0.022$ | $\rho = 0.974, \sigma_d = 0.007$ |
| $n_p = 6$ | $\rho = 0.963, \sigma_d = 0.008$ | $\rho = 0.974, \sigma_d = 0.008$ | $\rho = 0.960, \sigma_d = 0.008$ |

### 4.2.3 Posterior, observations on scaled prior sample

For a $75\%$ scaling of an FEM prior realization as ground truth the scaling parameter $\rho$ is correctly approximated to be close to $0.75$ in every case, see Table 4.2. Again, the model error is very low due to the low $\sigma_d$, which is expected. Scaling the FEM prior sample by $75\%$ should lead to a scaling factor of $\rho = 0.75$. Figure 4.2.3 shows that this is approximately the case.

The 2D field shows the mean posterior whereas the green curve describes a cut through the FEM prior field which gives insight on the variance of the prior. In black, a cut through the posterior mean is shown. The $75\%$ scaling is visible. Figure 4.10 shows the variance fields for the posterior when taking measurements from the $75\%$ scaling. Still, the variance gets lower with higher $n_p$ and $n_o$.

Figure 4.11 shows ten sampled points in space and the FEM prior on the left as well as the corresponding posterior on the right. It is visible that the posterior matches the data exactly.

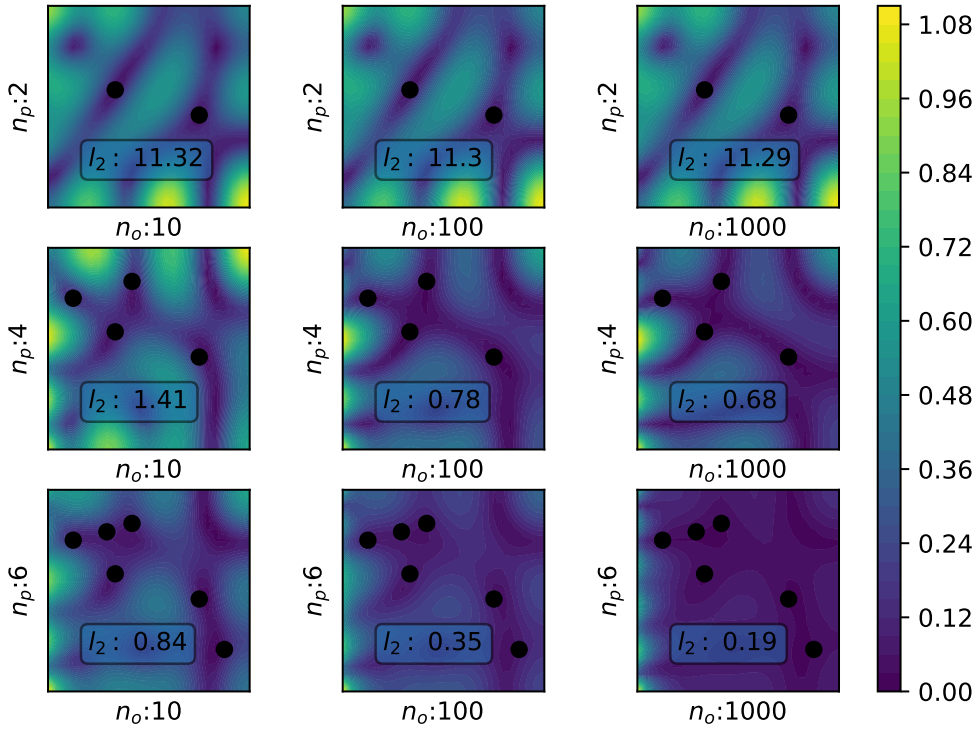



Figure 4.8: The variance of the posterior for $500Hz$ at all points in the 2D space for different combinations of sensor and observation numbers. Regions with high and low variance are clearly visible. Where sensor locations are introduced, the variance gets smaller in patterns around the location all over the domain. Adding more sensor locations induces a thinner mesh of patterns. The more observations are taken per sensor, the smaller the overall variance becomes, which reflects in the specified l2 norm for it.

### 4.2.4 Posterior, observations on altered prior sample

In a second example, the ground truth to measure from was again the FEM prior but with certain changes. In a certain region of the domain the values of the FEM prior sample were changed according to a 2D Gaussian, resulting in a clearly visible deviation. That deviation should lead to a larger model inadequacy term whereas the scaling factor should stay roughly the same. Figure 4.12 shows, how observations with model error are created. At first, a rescaling pattern is created. It comprises values of around one and is then multiplied with a sample from the FEM prior. This new, altered, sample is now used to make observations on. These observations differ from the standard prior sample by the amount specified in the rescaling pattern. Note that if the prior would only have been rescaled by a constant factor throughout the domain, no model error would have been introduced: statFEM would yield a fitting scaling parameter $\rho$ and model inadequacy hyperparameters of close to zero. Here, it is forced to work with all three hyperparameters since a simple constant rescaling would not be enough. The non-constant rescaling follows

$$K_{res} \odot K_{samp} = K_{obs} \tag{4.6}$$

where $K_{res}$ is the matrix which contains the values of the rescaling pattern and $K_{samp}$ is a sample drawn from the FEM prior GP. An element-wise multiplication yields $K_{obs}$, the resulting matrix to draw samples

Table 4.2: For a 75% scaling and no introduced model error, $\rho$ is close to 0.75 for every case, which is expected for a 75% scales FEM prior sample as ground truth. The model error is close to zero.

| | $n_o = 10$ | $n_o = 100$ | $n_o = 1000$ |
|---|---|---|---|
| $n_p = 2$ | $\rho = 0.741, \sigma_d = 0.011$ | $\rho = 0.746, \sigma_d = 0.010$ | $\rho = 0.735, \sigma_d = 0.007$ |
| $n_p = 4$ | $\rho = 0.750, \sigma_d = 0.011$ | $\rho = 0.753, \sigma_d = 0.010$ | $\rho = 0.739, \sigma_d = 0.025$ |
| $n_p = 6$ | $\rho = 0.764, \sigma_d = 0.012$ | $\rho = 0.762, \sigma_d = 0.010$ | $\rho = 0.764, \sigma_d = 0.007$ |

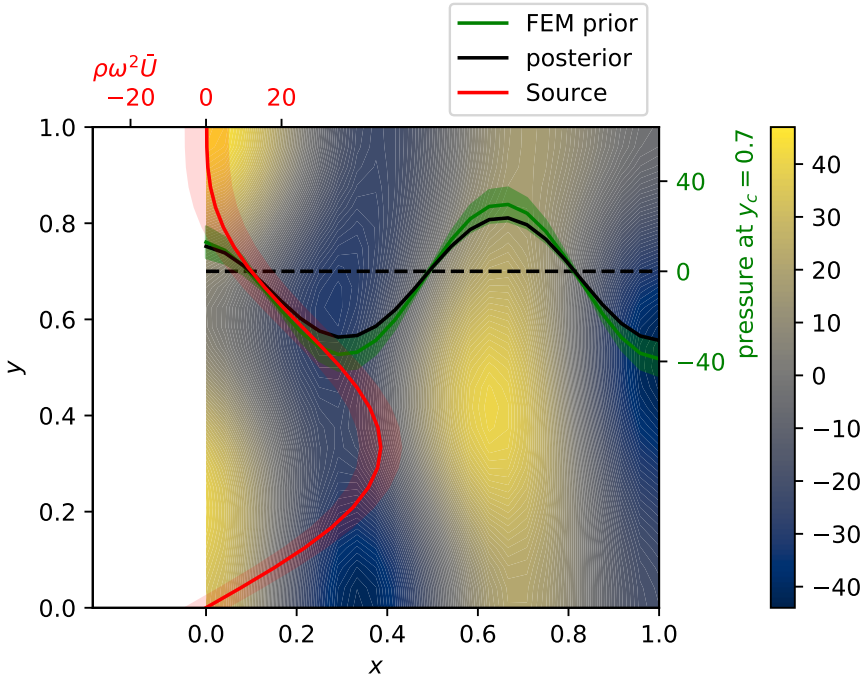



Figure 4.9: The variance of the posterior for $500Hz$ at all points in the 2D space for different combinations of sensor and observation numbers. Sampled from a 75% scaling. Regions with high and low variance are clearly visible.

from.

The pattern is described by

$$z = 1 + \frac{0.5}{\exp(5((x-0.5)^2 + (y-0.5)^2))} \,. \tag{4.7}$$

As visible in Figure 4.13, the observations follow the general shape of the prior but deviate slightly in terms of magnitude. It is now expected that statFEM is able to form a posterior which resembles the observed data more closely while maintaining the general shape of the FEM prior. As it is hard to see by eye in the plots, the difference of the magnitude of the observations compared to the prior and the posterior are cumulated in the mean squared error (MSE). As expected, the MSE for the FEM prior is comparatively high. statFEM manages to find the best hyperparameters to find a posterior which matches the data as close as possible. This is also reflected in the generated hyperparameters: Since the rescaling pattern only comprises values greater than one, the overall prior is rescaled with $\rho > 1$. But, to

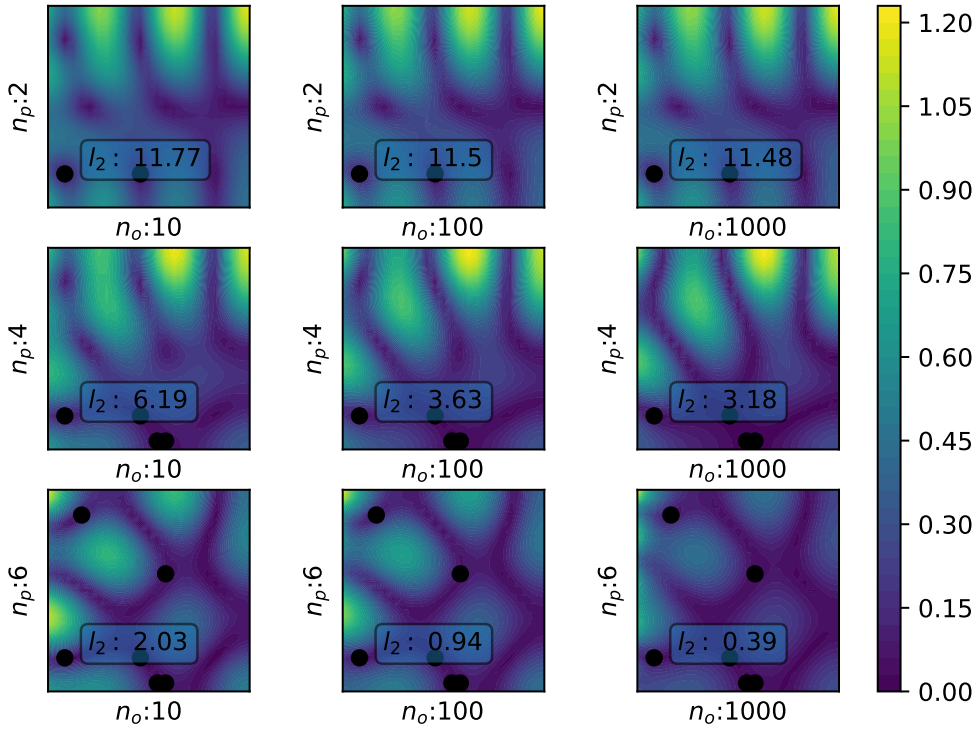



Figure 4.10: The variance of the posterior for $500Hz$ at all points in the 2D space for different combinations of sensor and observation numbers. Sampled from a 75% scaling. Regions with high and low variance are clearly visible. Although the observations were taken from the scaled sample, the variance quickly gets smaller with more sensors and observations.

account for the non-constant rescaling, also the model inadequacy hyperparameters $\sigma_d$ and $_d$ are now much larger than in the example without model error, as it is visible in Table 4.3. It is visible in Figure

Table 4.3: Hyperparameters for the case of observed data with a model error. The solution is scaled by around 20% and the hyperparameters describing the model inadequacy are not close to zero anymore.

| | $n_o = 10$ | $n_o = 100$ | $n_o = 500$ |
|---|---|---|---|
| $n_p = 15$ | $\rho = 1.206, \sigma_d = 3.230,$ $l_d = 0.471$ | $\rho = 1.206, \sigma_d = 2.213,$ $l_d = 0.471$ | $\rho = 1.181, \sigma_d = 3.648,$ $l_d = 0.556$ |
| $n_p = 30$ | $\rho = 1.125, \sigma_d = 3.333$ $l_d = 0.147$ | $\rho = 1.220, \sigma_d = 3.019$ $l_d = 0.154$ | $\rho = 1.239, \sigma_d = 2.848$ $l_d = 0.105$ |
| $n_p = 45$ | $\rho = 1.284, \sigma_d = 7.071$ $l_d = 0.343$ | $\rho = 1.195, \sigma_d = 7.633$ $l_d = 0.429$ | $\rho = 1.181, \sigma_d = 4.157$ $l_d = 0.301$ |

4.14 that the variance doesn't become smaller for 10 observations when the number if sensor locations is changed from 4 to 6. Because of the small number of sensors and observations, the model error couldn't be resolved adequately which lead in turn to the higher variance. This has already been shown for the

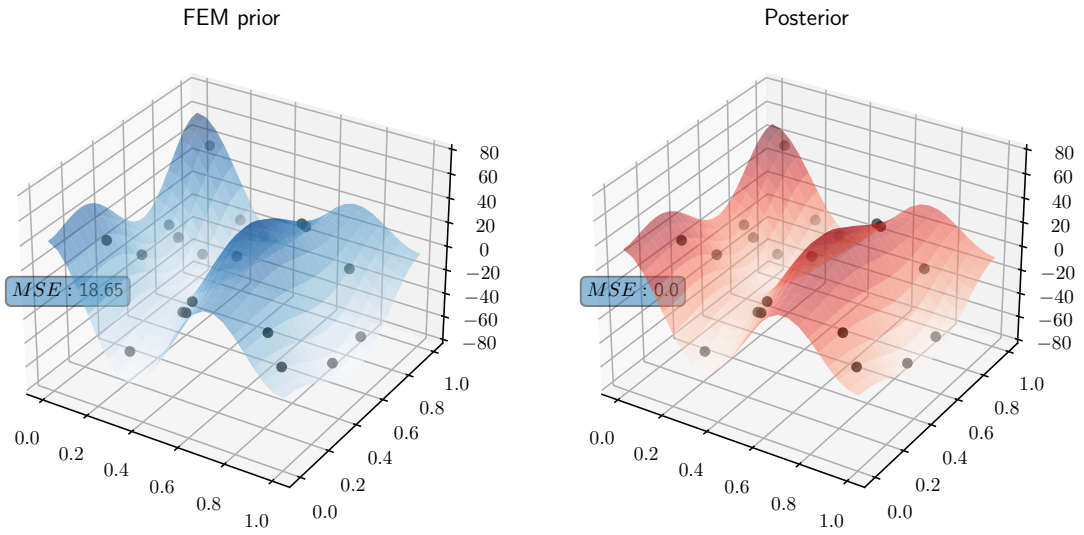



Figure 4.11: See the Animation to see the differences better. With observations on a 75% scaling of the FEM prior and no other introduced model error, the posterior is able to match the data with an MSE of 0.0.

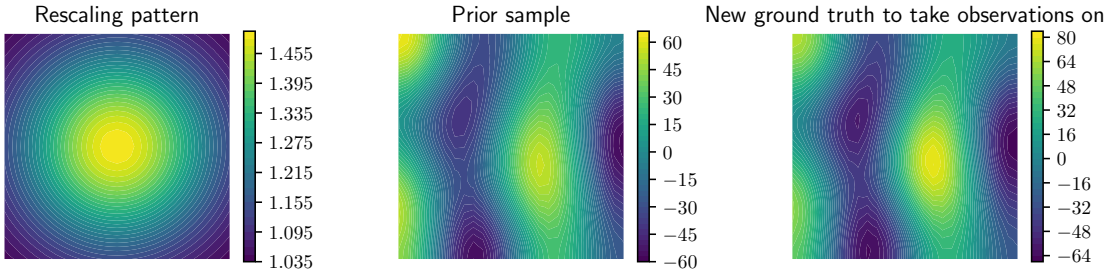



Figure 4.12: Multiplying a prior sample with a rescaling pattern leads to a new ground truth with model error. The pattern strongly scales the sample in the middle of the domain and less strongly on the corners.

1D example in Figure 4.3 and is therefore expected. The effect can also be explained in another way: As already discussed, Bayesian regression always favors a less complex model over a more complex one. If now only very few sensors are used, it would be possible to find a posterior which adequately describes the data by only scaling the prior using the scaling hyperparameter $\rho$. The other hyperparameters could stay unchanged, i.e. zero, resulting in a simple model. If then more sensors are introduced, it is not possible to describe the data by only scaling the prior, so the model inadequacy part $d$ comes into play. The posterior variance quickly converges in this case by observing more data per sensor. Figure 4.15 shows that this is the case: Introducing more sensor locations leads to the typical behavior of decreasing variance with more sensors and observations.

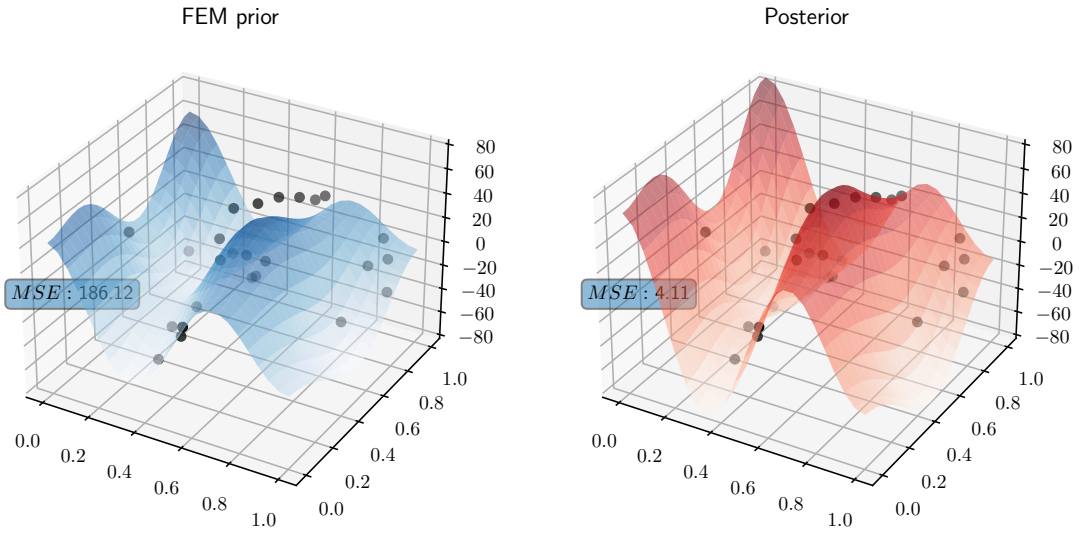



Figure 4.13: On the left: A sample from the FEM prior and the observations. It is visible that the points, first if al those in the middle, are not coincident with the surface formed by the prior. The mean squared error (MSE) is given and rather high. On the right: The posterior now fits to the observation data which is also reflected in a lower MSE.

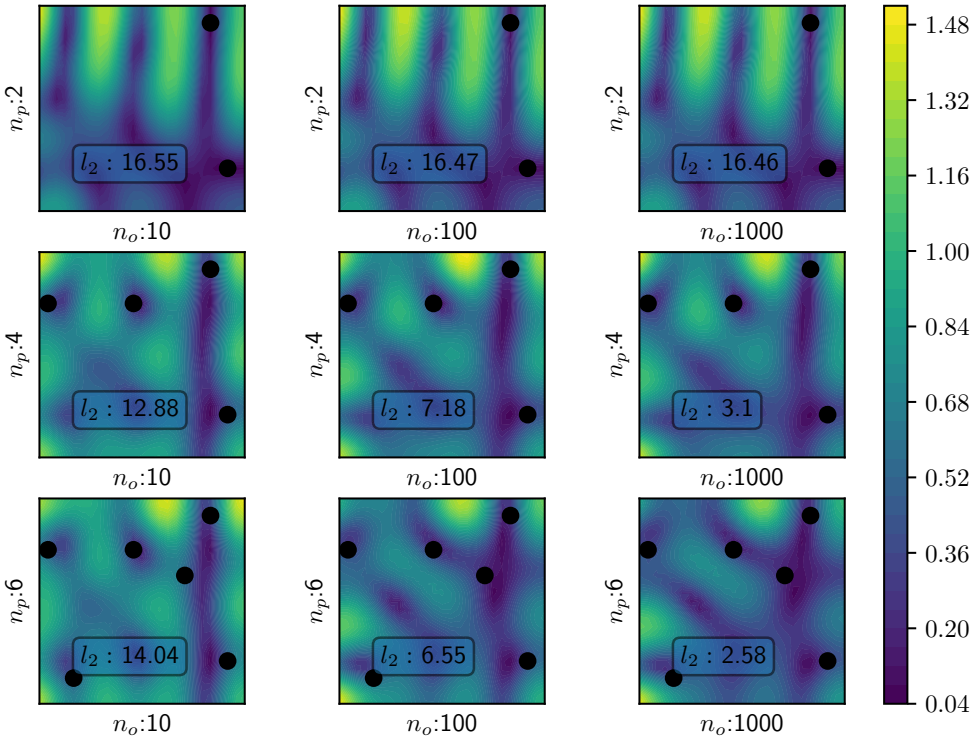



Figure 4.14: Introducing model error leads to a higher variance for a small number of sensor locations.

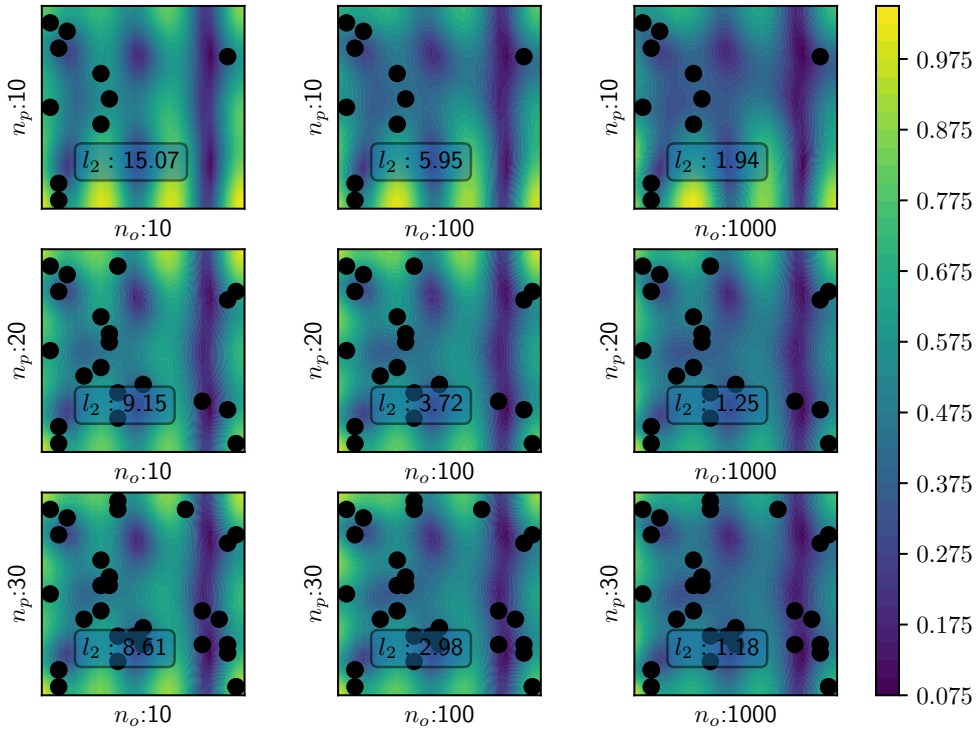



Figure 4.15: Introducing more sensor locations, it is now visible that the variance goes down with mmore observations and more sensors, just as expected.

### 4.2.5 Prior-Data Conflict

As already shown for the 1D example, as soon as data is observed far outside the confidence region of the prior, i.e. there is a prior-data conflict, the inference doesn't yield accurate results anymore. Figure **??** shows that especially the observations in the middle of the domain are not matched and that the MSE remains very high.

### 4.2.6 Variations in the Neumann BC

If instead of a sinusoidal source a flat one is chosen, the solution changes significantly, see Figure 4.16. Certain mode shapes can be excited more strongly by using a corresponding boundary function. The variance field, on the other hand, does not change significantly, see Figure 4.17. With the FEM prior, data can now be introduced to form the posterior.

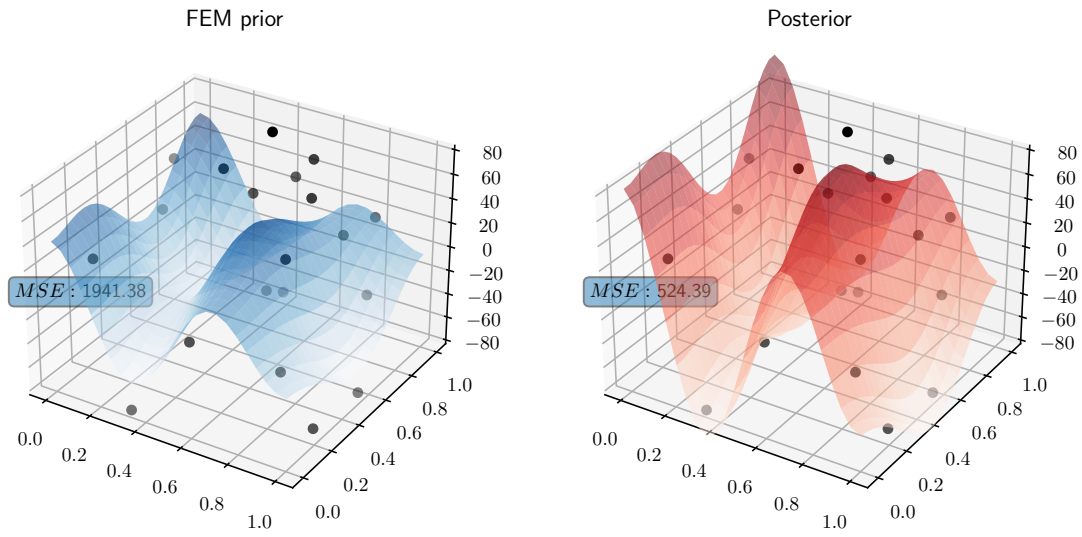



Figure 4.16: See the Animation to see the differences better. With observations on a 75% scaling of the FEM prior and no other introduced model error, the posterior is able to match the data with an MSE of 0.0.

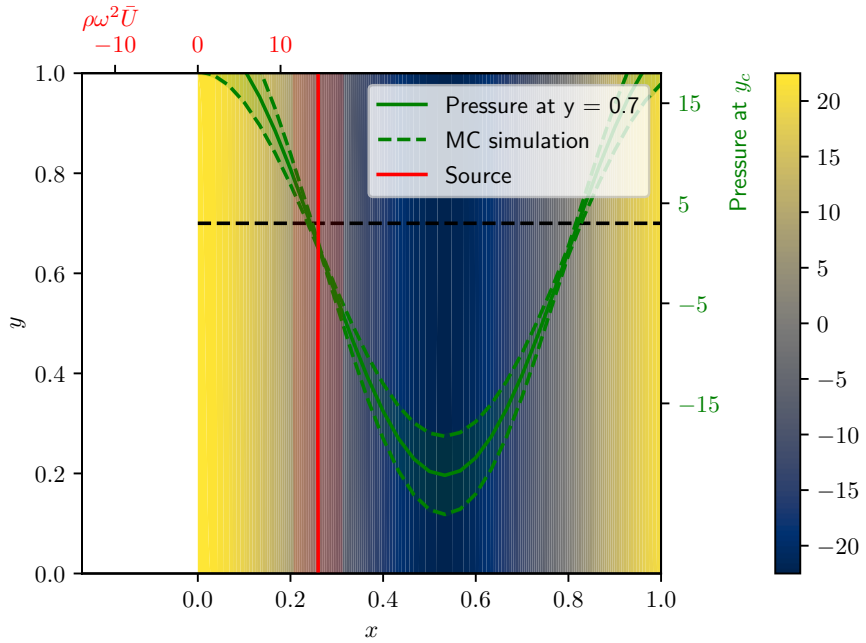



Figure 4.17: The source GP, the mean prior and, at the cut $y = y_c$, the prior FEM GP for 320Hz.

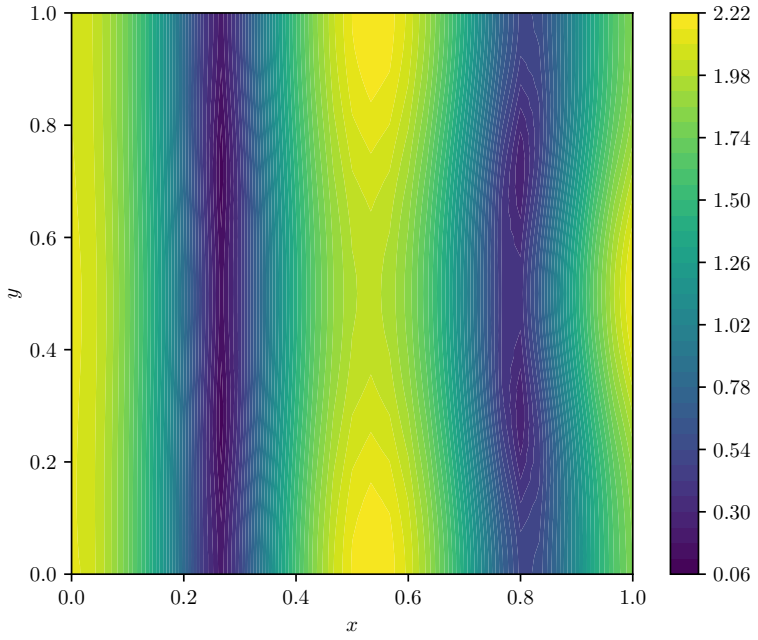



Figure 4.18: The variance of the FEM prior for $320Hz$ at all points in the 2D space. Regions with high and low variance are clearly visible.

# 5 Conclusion and Discussion

# Bibliography


[1] N. Atalla and F. Sgard, *Finite Element and Boundary Methods in Structural Acoustics and Vibration*. 2015, ISBN: 978-1-4665-9287-2 (cit. on pp. 6, 8, 11).

[2] M. G. Larson and F. Bengzon, *The Finite Element Method: Theory, Implementation, and Applications*, ser. Texts in Computational Science and Engineering 10. Berlin Heidelberg: Springer, 2013, 385 pp. (cit. on pp. 6, 7).

[3] M. Möser, *Technische Akustik*, 6., erw. und aktualisierte Aufl, ser. VDI-[Buch]. Berlin: Springer, 2005, 356 pp., ISBN: 978-3-540-22510-2 (cit. on p. 6).

[4] T. Westermann, *Mathematik für Ingenieure*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, ISBN: 978-3-642-54290-9 (cit. on p. 7).

[5] T. Arens, F. Hettlich, C. Karpfinger, U. Kockelkorn, K. Lichtenegger, and H. Stachel, *Mathematik*, 3. Auflage. Berlin Heidelberg: Springer Spektrum, 2015, 1630 pp., ISBN: 978-3-642-44918-5 (cit. on pp. 7, 15, 17, 34).

[6] H. P. Langtangen and A. Logg, *Solving PDEs in Python*. Cham: Springer International Publishing, 2016. DOI: 10.1007/978-3-319-52462-7. [Online]. Available: http://link.springer.com/10.1007/978-3-319-52462-7 (visited on 06/24/2021) (cit. on p. 8).

[7] H. P. Langtangen and K.-A. Mardal, *Introduction to Numerical Methods for Variational Problems*. 2019, ISBN: 978-3-030-23788-2 (cit. on pp. 10, 11, 13, 44).

[8] A. Druinsky and S. Toledo, "How accurate is inv(a)*b?", Jan. 29, 2012. arXiv: 1201.6035 [cs.NA] (cit. on p. 15).

[9] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*. CAMBRIDGE, Aug. 1, 2002, 680 pp., ISBN: 0898715210. [Online]. Available: https://www.ebook.de/de/product/6906578/nicholas_j_higham_accuracy_and_stability_of_numerical_algorithms.html (cit. on p. 15).

[10] T. Davis, *Direct methods for sparse linear systems*. Philadelphia: Society for Industrial and Applied Mathematics, 2006, ISBN: 9780898716139 (cit. on p. 15).

[11] H. Elman, D. Silvester, and A. Wathen, *Finite Elements and Fast Iterative Solvers*. Oxford University Press, Jun. 2014. DOI: 10.1093/acprof:oso/9780199678792.001.0001 (cit. on p. 15).

[12] MathWorks. (). Condition number of a matrix, [Online]. Available: https://de.mathworks.com/help/symbolic/cond.html (cit. on p. 15).

[13] D. Belsley, *Regression diagnostics : identifying influential data and sources of collinearity*. New York: Wiley, 1980, pp. 100–104, ISBN: 0471058564 (cit. on p. 15).

[14] P. Virtanen, R. Gommers, T. E. Oliphant, and M. Haberland, "SciPy 1.0: Fundamental algorithms for scientific computing in python", *Nature Methods*, vol. 17, no. 3, pp. 261–272, Feb. 2020. DOI: 10.1038/s41592-019-0686-2 (cit. on p. 18).

[15] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006, 248 pp., ISBN: 978-0-262-18253-9 (cit. on pp. 18–21, 24–27, 29, 40, 45, 49).

[16] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, ser. Adaptive Computation and Machine Learning Series. Cambridge, MA: MIT Press, 2012, 1067 pp., ISBN: 978-0-262-01802-9 (cit. on pp. 19–21, 24, 25, 44).

[17] J. Görtler, R. Kehlbeck, and O. Deussen, "A Visual Exploration of Gaussian Processes", *Distill*, vol. 4, no. 4, 10.23915/distill.00017, Apr. 2, 2019, ISSN: 2476-0757. DOI: 10.23915/distill.00017. [Online]. Available: https://distill.pub/2019/visual-exploration-gaussian-processes (visited on 06/24/2021) (cit. on pp. 19, 20, 24).

[18] A. Damianou, *Gaussian process lecture*, University of Sheffield, Jun. 21, 2021. [Online]. Available: https://github.com/adamian/adamian.github.io/blob/1074c865911e2f83f29c52(
talks/Brown2016.ipynb (visited on 08/11/2021) (cit. on pp. 21, 30).

[19] D. K. Duvenaud, "Automatic model construction with gaussian processes", PhD thesis, University of Cambridge, (cit. on pp. 24, 25, 27).

[20] M. L. Stein, *Interpolation of Spatial Data Some Theory for Kriging*. 1999, ISBN: 978-1-4612-1494-6. [Online]. Available: https://doi.org/10.1007/978-1-4612-1494-6 (visited on 06/24/2021) (cit. on p. 26).

[21] B. Matern, *Spatial Variation*. New York, NY: Springer, 2013, ISBN: 978-1-4615-7892-5. [Online]. Available: https://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=5590166 (visited on 06/24/2021) (cit. on p. 26).

[22] M. Abramowitz and I. A. Stegun, Eds., *Handbook of Mathematical Functions: With Formulas, Graphs, and Mathematical Tables*, 9. Dover print.; [Nachdr. der Ausg. von 1972], ser. Dover Books on Mathematics. New York, NY: Dover Publ, 2013, 1046 pp., ISBN: 978-0-486-61272-0 (cit. on p. 26).

[23] M. Girolami, E. Febrianto, G. Yin, and F. Cirak, "The statistical finite element method (statFEM) for coherent synthesis of observation data and model predictions", *Computer Methods in Applied Mechanics and Engineering*, vol. 375, p. 113 533, Mar. 2021, ISSN: 00457825. DOI: 10.1016/j.cma.2020.113533. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0045782520307180 (visited on 06/24/2021) (cit. on pp. 32, 38, 40, 43, 44).

[24] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models", *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001, ISSN: 13697412. DOI: 10.1111/1467-9868.00294. [Online]. Available: http://doi.wiley.com/10.1111/1467-9868.00294 (visited on 06/24/2021) (cit. on pp. 32, 40, 41).

[25] T. B. Schon and F. Lindsten, "Manipulating the Multivariate Gaussian Density", p. 10, 2011. [Online]. Available: http://user.it.uu.se/~thosc112/pubpdf/schonl2011.pdf (visited on 07/01/2021) (cit. on p. 35).

[26] J. Bardsley, *Computational Uncertainty Quantification for Inverse Problems*, ser. Computational Science and Engineering. Philadelphia: Society for Industrial and Applied Mathematics, 2018, 1 p., ISBN: 978-1-61197-538-3 (cit. on p. 35).

[27] K. S. Riedel, "A Sherman–Morrison–Woodbury Identity for Rank Augmenting Matrices with Application to Centering", *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 2, pp. 659–662, Apr. 1992, ISSN: 0895-4798, 1095-7162. DOI: 10.1137/0613040. [Online]. Available: http://epubs.siam.org/doi/10.1137/0613040 (visited on 06/24/2021) (cit. on p. 38).

[28] J. Chambers, W. Eddy, and W. Härdle, *Numerical Linear Algebra for Applications in Statistics*. New York: Springer New York, 1998, ISBN: 978-1-4612-0623-1 (cit. on p. 41).

[29] Tomaskovic-Moore, "Math 240 — Calculus III", [Online]. Available: https://www2.math.upenn.edu/~moose/240S2013/slides7-16.pdf (visited on 07/01/2021) (cit. on p. 41).

[30] A. Svensson, J. Dahlin, and T. B. Schön, "Marginalizing gaussian process hyperparameters using sequential monte carlo", Feb. 6, 2015. DOI: 10.1109/CAMSAP.2015.7383840. arXiv: 1502.01908 [stat.ML] (cit. on p. 41).