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

October 2021

Erstprüferin: Prof. Dr.-Ing. Sabine C. Langer

Zweitprüfer: Prof. Dr.-Ing. Ulrich Römer

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, July 8, 2021

Lucas Hermann

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Contents

1 FEM Solution of the Helmholtz Equation **6**
- 1.1 Wave Equation and Helmholtz Equation . . . 6
- 1.2 Boundary Conditions . . . 7
- 1.3 Weak Formulation and Discretization . . . 9
- 1.4 The Classical Finite Element Method . . . 10
  - 1.4.1 Integral and Weak Formulation . . . 10
  - 1.4.2 Discretization . . . 11
  - 1.4.3 Approximation and Building Element Matrices . . . 13
  - 1.4.4 Setting Constraints . . . 15
  - 1.4.5 Solving and Convergence . . . 15

2 Gaussian Processes and the Statistical Finite Element Method **16**
- 2.1 Basic Probability Theory . . . 16
  - 2.1.1 Gaussian Distribution . . . 16
  - 2.1.2 Bayesian Inference . . . 17
- 2.2 Gaussian Process Regression . . . 18
  - 2.2.1 Regression . . . 18
  - 2.2.2 Gaussian Processes . . . 18
  - 2.2.3 Kernel Functions . . . 19
  - 2.2.4 Prior and Posterior . . . 22
- 2.3 The Statistical Finite Element Method . . . 22
  - 2.3.1 Step 1: Posterior of the FEM Forward Model . . . 24
  - 2.3.2 Inference of the Posterior Density . . . 28
  - 2.3.3 find Posterior mean and variance . . . 30
  - 2.3.4 Modeling Parameters of the PDE . . . 30
  - 2.3.5 Probabilistic FEM Prior . . . 30
  - 2.3.6 Step 2: Estimation of Hyperparameters . . . 30
  - 2.3.7 Step 3: Estimating the Optimum Mesh Parameters . . . 32

3 Application of statFEM in Vibroacoustics **33**
- 3.1 Simple 1D example . . . 33
  - 3.1.1 Posterior . . . 34
- 3.2 1D Vibroacoustics Example . . . 34
  - 3.2.1 FEM prior . . . 34

4 Conclusion and Discussion **39**
- 4.1 Section 1 . . . 39

Introduction

Dies ist ein Text in `tubsSecondary`. Dies ist ein Text in `tubsViolet`. Dies ist ein Text in `tubsGreenDark`.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

- Aufzählungspunkt Eins
- Aufzählungspunkt Zwei
  - Unter-Aufzählungspunkt Eins
  - Unter-Aufzählungspunkt Zwei
- Aufzählungspunkt Drei

1 FEM Solution of the Helmholtz Equation

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 behaviour of the system.

1.1 Wave Equation and Helmholtz Equation

In a typical vibroacoustic problem, for example an interior structural acoustic coupling problem [5], 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 [1] and [2]. The behaviour of a gas in a domain Ω can be described using the density ρ, 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 propageted 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 accelaration equals force. If a gas volume is accelerated in one direction, that leads to a force on the neighbouring particle [2].

$$m\dot{v} = S\left[p(x) - p(x + \Delta x)\right] \tag{1.1}$$

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

$$\rho\dot{v} = -\frac{p(x) - p(x + \Delta x)}{\Delta x} \ . \tag{1.2}$$

Assuming interaction between infinitessimaly small gas particles and therefore taking the limit of the difference quotient in (1.2) yields the differential quotient and hence

$$\rho\dot{v} = -\nabla p \ . \tag{1.3}$$

∇ 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.

$$\nabla p = \mathrm{grad} p = \left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}\right) \ . \tag{1.4}$$

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 [1]. 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

$$\dot{p} = -k\nabla u \tag{1.5}$$

with k a proportionality constant dependent on the medium. Differentiating (1.5) with respect to t yields

$$\ddot{p} = -k\nabla \dot{u} \; . \tag{1.6}$$

Inserting 1.3 and considering $c^2 = k/\rho$ there holds for the acoustic wave equation

$$\ddot{p} = c^2 \nabla^2 p \tag{1.7}$$

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 [3]

$$p(x,t) = X(x) \cdot T(t) \; , \tag{1.8}$$

there holds for 1.7

$$X(x) \cdot \frac{\partial^2 T(t)}{\partial t^2} = c^2 \nabla^2 X(x) \cdot T(t) \tag{1.9}$$

$$\frac{1}{T(t)} \frac{\partial^2 T(t)}{\partial t^2} = c^2 \frac{1}{X(x)} \nabla^2 X(x) = const. = -\omega^2 \; . \tag{1.10}$$

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

$$\nabla^2 X(x) + k^2 X(x) = 0 \tag{1.11}$$

is the Helmholtz equation which is a representation of the wave equation independent of time [4, 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

$$\nabla^2 p + k^2 p = 0 \; . \tag{1.12}$$

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

$$k = \frac{\omega}{c_0} = \frac{2\pi f}{c_0} \; . \tag{1.13}$$

(1.12) 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.

1.2 Boundary Conditions

The following closely follows [5]. 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.

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

$$p(x) = \bar{p}(x) \quad \forall x \in \Gamma_D \; . \tag{1.14}$$

Neumann The Neumann boundary condition at Γ_N reads

$$\frac{\mathrm{d}p(x)}{\mathrm{d}n} = \rho\omega^2 U(\vec{x}) \cdot \vec{n} \quad \forall x \in \Gamma_N \tag{1.15}$$

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 ω 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 Γ_{N0}.

Robin The Robin or impedance boundary condition at Γ_R reads

$$\frac{\mathrm{d}p(x)}{\mathrm{d}n} + ik\beta p(x) = 0 \quad \forall x \in \Gamma_R \tag{1.16}$$

with β 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 1.1 shows an example on how boundary conditions could be applied on a 2D domain. Γ_{N0} boundary conditions are the mentioned natural boundary conditions which don't have to be applied explicitly. Still, a Neummann boundary condition Γ_N can be imposed onto the system which behaves differently. Figure 1.2 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.

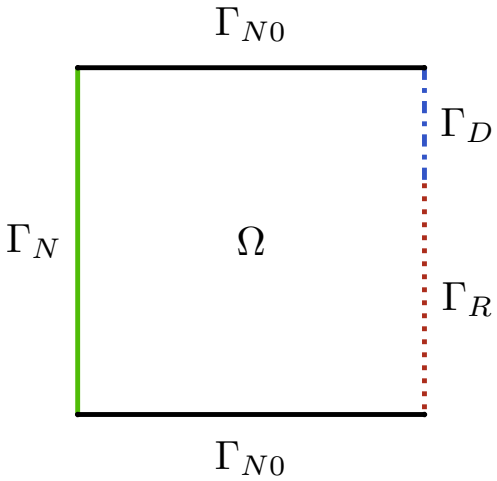


Figure 1.1: Two dimensional representation of various boundary conditions. In this example, the left side is a Neumann boundary, the upper and lower boundaries are totally reflecting and the right side consists of an impedance BC and a Dirichlet BC.

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

```
1 class BoundaryX_L(SubDomain):
2         self.tol = 1E-14
3         def inside(self, x, on_boundary):
4                 return on_boundary and near(x[0], 0, tol)# and (x[1] < 0.3)
```

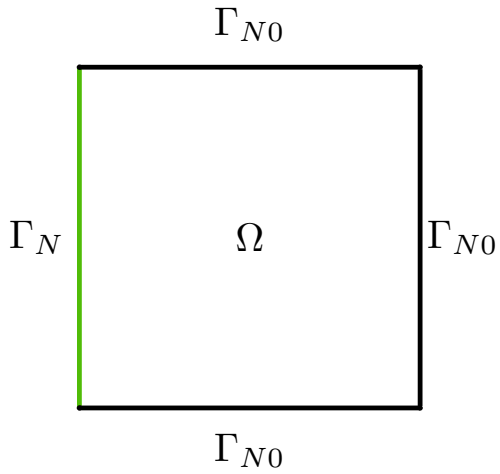


Figure 1.2: Boundary Conditions of the problem for this thesis. A Neumann boundary is applied on the side of the domain. The rest of the boundary is considered as reflecting, i.e. $\Gamma_N = 0$.

```
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 bounda
```

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

```
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 #variational Pro
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.

1.3 Weak Formulation and Discretization

Explain what the weak form is and why we need it. Derive the weak form, completely with integration by parts etc. explain all parts of it and what they represent.

explain what a bilinear and a linear form is!

Eq. (1.12) is considered a *strong* formulation. In order to be able to set up the FEM, the PDE needs to be in the *weak* form. Among other possibilities, the weak form can be derived using the Weighted Residuals Method.

In order to lower the degree of derivation, integration by parts is executed:

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)
```

1.4 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.

1.4.1 Integral and Weak Formulation

- Explain Vector spaces - Test function: needs to obey the dirichlet conditions

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 1.4.3 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 φ_i. Demanding the scalar product of the residual and a test function v to be zero

$$(R, v) = 0, \quad \forall v \in V \,, \tag{1.17}$$

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 Ω to get the so-called integral formulation:

$$\int_\Omega v \nabla^2 p \, \mathrm{d}\Omega + \int_\Omega k^2 p v \, \mathrm{d}\Omega = 0 \tag{1.18}$$

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 [5, 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:

$$\int_\Omega \nabla \cdot \vec{F} \, \mathrm{d}\Omega = \oint_\Gamma \vec{F} \cdot \vec{n} \, \mathrm{d}\Gamma \, . \tag{1.19}$$

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

$$\oint_\Gamma v \nabla p \, \mathrm{d}\Gamma = \int_\Omega \nabla p \cdot \nabla v \, \mathrm{d}\Omega + \int_\Omega v \nabla^2 p \, \mathrm{d}\Omega \, . \tag{1.20}$$

Applying it to (1.18) yields

$$- \int_\Omega \nabla p \cdot \nabla v \, \mathrm{d}\Omega + \oint_\Gamma v \nabla p \, \mathrm{d}\Gamma + \int_\Omega k^2 p v \, \mathrm{d}\Omega = 0 \tag{1.21}$$

with the boundary of the domain Γ. 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

$$\oint_\Gamma v \nabla p \, \mathrm{d}\Gamma = \oint_{\Gamma_D} v \nabla \bar{p} \, \mathrm{d}\Gamma + \oint_{\Gamma_N} v(\rho \omega^2 \vec{U} \vec{n}) \, \mathrm{d}\Gamma + \oint_{\Gamma_R} v(-ik\beta\bar{p}) \, \mathrm{d}\Gamma \tag{1.22}$$

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.

1.4.2 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 Ω is split into n_e individual elements $\Omega^{(e)}$

$$\Omega = \bigcup_{e=1}^{n_e} \Omega^{(e)} \quad e = 1, ..., n_e \, . \tag{1.23}$$

This holds not only for 1D meshes but for 2D and 3D meshes as well. For 1D meshes, such as in Figure 1.3, 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 1.4. 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 [5].

Figures 1.3 and 1.4 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

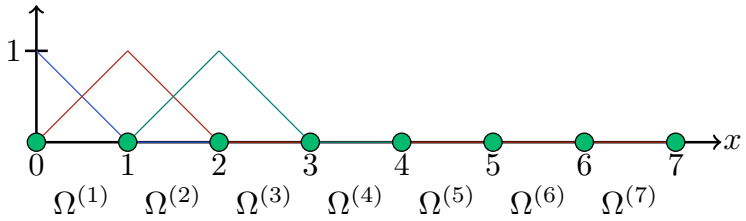


Figure 1.3: A discretized 1D domain with nodes in green and linear ansatz functions shown for the first four nodes

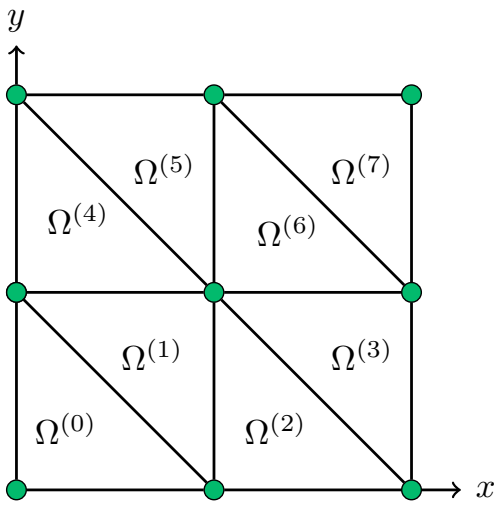


Figure 1.4: A discretized 2D domain with nodes in green

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 1.4. Of course every desired 2D shape can be used as a mesh.

1.4.3 Approximation and Building Element Matrices

In FEM, the values between nodes within an element are interpolated with basis functions φ_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

$$p(x_i) = \sum_{j=1}^{n} c_j \varphi_j(x_i) = f(x_i) \ . \tag{1.24}$$

(1.24) 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 φ_j evaluated at that point. If now orthogonal basis functions are used, as it is commonly the case in FEM, (1.24) 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

$$\varphi_i(x) = \prod_{j=0, j\neq i}^{N} \frac{x - x_j}{x_i - x_j} \ . \tag{1.25}$$

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

$$\varphi_i(x_s) = \delta_{is}, \quad \delta_{is} = \begin{cases} 1 & \text{for} \quad i = s \\ 0 & \text{for} \quad i \neq s \end{cases} \tag{1.26}$$

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 (1.24), if x_i is the location of a node of the mesh:

$$p(x_i) = \hat{p}_i \varphi_i(x_i) = \hat{p}_i \ . \tag{1.27}$$

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 (1.21), 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,

$$\begin{aligned} \tilde{p} &= \boldsymbol{\varphi}^{\boldsymbol{T}} \hat{\boldsymbol{p}} \\ \tilde{v} &= (\boldsymbol{\varphi}^{\boldsymbol{T}} \delta \hat{\boldsymbol{p}})^T \end{aligned} \tag{1.28}$$

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 (1.28) in the weak form, considering no Robin boundary conditions, results in

$$\sum_e \int_{\Omega_e} (\nabla \boldsymbol{\varphi} \nabla \boldsymbol{\varphi}^T) \, \mathrm{d}\Omega_e \hat{\boldsymbol{p}} - k^2 \sum_e \int_{\Omega_e} (\boldsymbol{\varphi}\boldsymbol{\varphi}^T) \, \mathrm{d}\Omega_e \hat{\boldsymbol{p}} = \rho\omega^2 \sum_e \int_{\Gamma_{e,N}} \boldsymbol{\varphi} U_n \, \mathrm{d}\Gamma_{e,N} \ . \tag{1.29}$$

The sum $\sum_e$ means assembling the global system of matrices out of individual element matrices. In short, (1.29) can be written as

$$\left(\sum_e \boldsymbol{K}_1^e - k^2 \sum_e \boldsymbol{K}_2^e \right) \hat{\boldsymbol{p}} = \rho\omega^2 \sum_e \boldsymbol{f}^e \tag{1.30}$$

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

$$(\boldsymbol{K}_1 - k^2\boldsymbol{K}_2)\hat{\boldsymbol{p}} = \rho\omega^2\boldsymbol{f} \tag{1.31}$$

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

$$\boldsymbol{A} = (\boldsymbol{K}_1 - k^2\boldsymbol{K}_2)\ . \tag{1.32}$$

It is assembled using the individual matrices $\boldsymbol{A}^e$ for each element as visible in (1.36). 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

$$A_{r,s}^{(e)} = \int_{\Omega^{(e)}} \nabla\varphi_r \nabla\varphi_s \,\mathrm{d}\Omega^{(e)} - k^2 \int_{\Omega^{(e)}} \varphi_r\varphi_s \,\mathrm{d}\Omega^{(e)} \tag{1.33}$$

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 2×2 matrix results and for 1D quadratic Lagrange elements with three nodes each a 3×3 matrix results. Also the right hand side vector, now called $\boldsymbol{b}$, is computed element-wise using

$$b_r^{(e)} = \rho\omega^2 \int_{\Omega^{(e)}} \int_{\Gamma_{e,N}} \varphi_r U_n \,\mathrm{d}\Gamma_{e,N}\ . \tag{1.34}$$

The element matrices are then assembled to a global system matrix. Within an element, the coordinates of the nodes are $[0, ..., d]$ 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

$$A_{q(e,r),q(e,s)} = A_{r,s}^{(e)} \tag{1.35}$$

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:

$$A_{q(e,r),q(e,s)} = \begin{bmatrix} A_{0,0}^{(0)} & A_{0,1}^{(0)} & 0 & 0 \\ A_{1,0}^{(0)} & A_{1,1}^{(0)} + A_{0,0}^{(1)} & A_{0,1}^{(1)} & 0 \\ 0 & A_{1,0}^{(1)} & A_{1,1}^{(1)} + A_{0,0}^{(2)} & A_{0,1}^{(2)} \\ 0 & 0 & A_{1,0}^{(2)} & A_{1,1}^{(2)} \end{bmatrix} \tag{1.36}$$

For the global right-hand-side vector there holds

$$b_{q(e,r)} = b_r^{(e)} \tag{1.37}$$

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

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

1.4.4 Setting Constraints

1.4.5 Solving and Convergence

explain how the linear system is solved (LU decomp or Cholesky)

2 Gaussian Processes and the Statistical Finite Element Method

2.1 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.

2.1.1 Gaussian Distribution

speak about linearity somewhere! see Römer lectures

explain PDF/CDF and moments

explain the multivariate gaussian

A random variable X is considered normally, or Gaussian, distributed if $X \sim \mathcal{N}(\mu, \sigma)$ with μ the mean and σ 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

$$\mu = \mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x)\,\mathrm{d}x \tag{2.1}$$

and for the variance

$$\mathbb{V}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2\,. \tag{2.2}$$

For the covariance between two random variables X and Y there holds (see [4, p. 1434])

$$\mathrm{cov}[X, Y] = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]\,. \tag{2.3}$$

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

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)\,. \tag{2.4}$$

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

$$f_X(\boldsymbol{x}) = \frac{\exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\Sigma|}} \tag{2.5}$$

with Σ 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 2.1. The underlying mean vector and covariance matrix read

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \Sigma = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1.5 \end{bmatrix}\,. \tag{2.6}$$

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

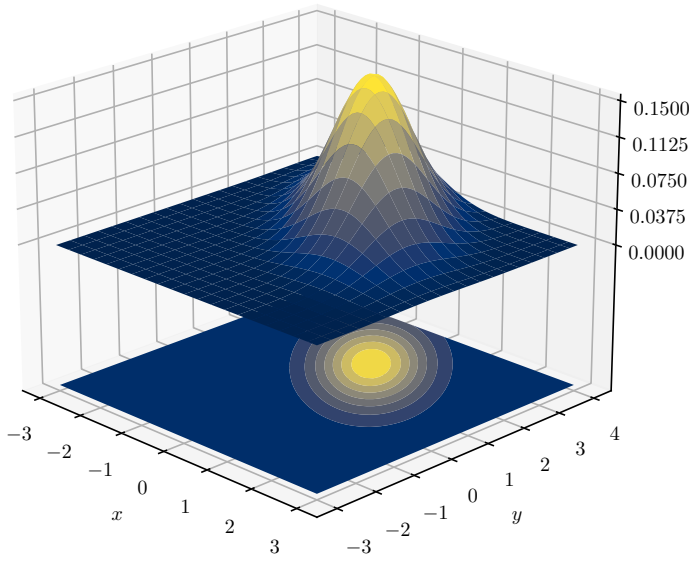


Figure 2.1: A multivariate Gaussian distribution with two dimensions. A strong correlation between the two random variables is visible. [scipy tutorial]

2.1.2 Bayesian Inference

$posterior = \frac{likelihood x prior}{marginal likelihood}$ (Rasmussen p.9) 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

$$p(\boldsymbol{u}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})} \tag{2.7}$$

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 2.2.2 for more detail) serves as a normalization constant and describes how likely it is to measure the observed data.

2.2 Gaussian Process Regression

2.2.1 Regression

"Regression is used to find a function that rerpesents a set of data points as closely as possible" [8] [8] "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 [9].

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. 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. [9, p. 2] Hence, the GP assigns a confidence band to a function where a usual polynomial regression wouldn't.

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

2.2.2 Gaussian Processes

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 describved 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

$$P(X,Y) = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right) \tag{2.8}$$

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

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX}) \, . \tag{2.9}$$

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

$$p_X(x) = \int_y p_{X,Y}(x,y) \, \mathrm{d}y = \int_y p_{X|Y}(x|y) p_Y(y) \, \mathrm{d}y \tag{2.10}$$

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

(2.10) 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 mutlivariate Gaussian there holds for the conditioning

$$X|Y \sim \mathcal{N}(\mu_X + \Sigma_{XY}\Sigma_{YY}^{-1}(Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY}\Sigma_{YY}^{-1}\Sigma_{YX}) \, . \tag{2.11}$$

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 2.2.

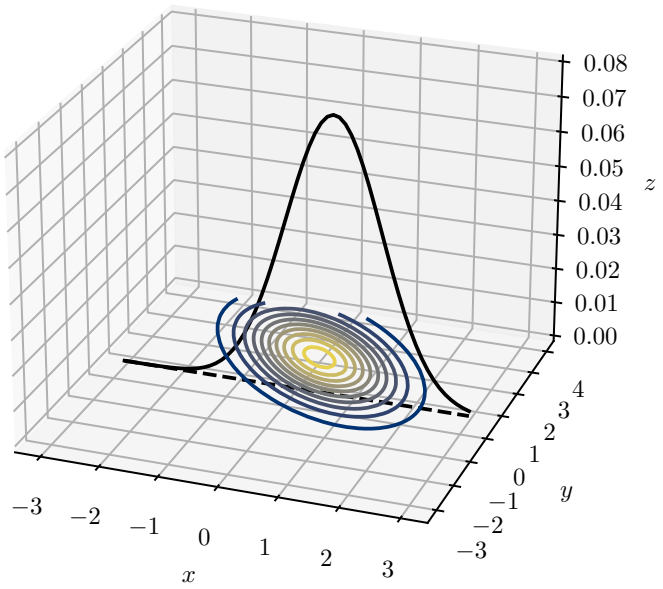


Figure 2.2: Cutting, i.e. conditioning, a bivariate Gaussian at the dashed line leads to a new Gaussian with only one dimension.

-Regression in general -differences of a GP regression to e.g. a polynomial regression: every point is assgined an uncertainty

-a distribution over functions

-Herleitung der Kovarianzmatrix, evtl. auch für das standard linear regression modell

-parameter variations

2.2.3 Kernel Functions

If a kernel function is evaluated at a finite number of points, one gets a covariance matrix. This can be used as the covariance of a Gaussian vector from which samples can be drawn. The values of each entry are a function over the input points of the kernel function. [9, 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. [9, p.79]

Stationary Kernels are a function of $x - x'$. They are stationary because it doesn't matter what value x and x_prime have as a starting value, the difference is always the same. Example: $1100mm - 1000mm = 100mm, 100100mm - 100000mm = 100mm$

p80: A radial basis function is a function which is !only! a function of $r = x - x'$ and whioch is therefore isotropic, i.e. there are no rigid motions. r is like a radius so these functions are called radial basis functions.

Covariance matrices always have to be symmetric, i.e. $k(x - x') = k(x' - x)$, since everything else wouldnt make sense

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.

The Squared Exponential Kernel Function is the most commonly used kernel function for GPs. It is stationary and infinitely differentiable, hence very smooth. It is defined as

$$k_{SE} = \sigma \exp\left(-\frac{r^2}{2l^2}\right) \tag{2.12}$$

and can be defined as a Python function with [Code von Murphy]

```
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) .
```

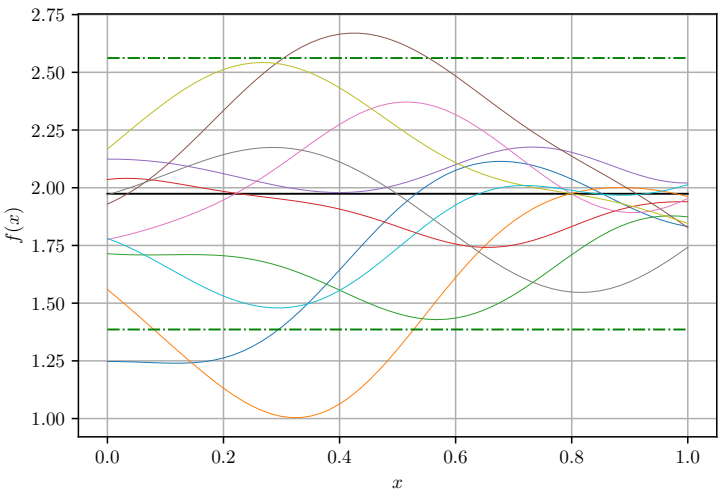


Figure 2.3: The source term GP of $f(x)$ sampled for a Squared Exponential kernel with $l = 0.25$ and $\sigma = 0.3$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

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 can be thought of as the number of "upcrossings" in the interval, i.e. the number of crossings of the x axis. It can be calculated with eq 4.3. rasmussen. The more upcrossings, the shorter the length scale.

Matern Kernel Functions [9] The squared exponential kernel is infinitely differentiable, which means that it behaves very smoothly. This is considered as unphysical by [10]. Therefore, the Matern class of covariance functions [11] is recommended by [10] which is finitely differentiable and therefore less smooth. It is called a "class" of kernels 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

$$k_M(r) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{l} \right) \tag{2.13}$$

with $\Gamma(\nu) = (\nu - 1)!$ the gamma function and K_ν a modified Bessel function [12, p.84 ff.], [9]. ν and l are free positive parameters. For $\nu \to \infty$ (2.13) becomes the squared exponential kernel. The covariance function simplifies for half integer values of ν and can be expressed as a product of an exponential and a polynomial. According to [9] $\nu = 3/2$ and $\nu = 5/2$ are commonly used for GPs; the simplest Matern kernel is obtained with $\nu = 1/2$:

$$k_{M;\nu=1/2}(r) = \sigma^2 \exp\left(-\frac{r}{l}\right), \tag{2.14}$$

$$k_{M;\nu=3/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right), \tag{2.15}$$

$$k_{M;\nu=5/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right). \tag{2.16}$$

(2.15) - (2.16) are used in this thesis. Figures 2.4 - 2.6 show the differences between the different choices for ν: The higher it is chosen, the smoother the GP becomes and the closer it gets to the squared exponential kernel.

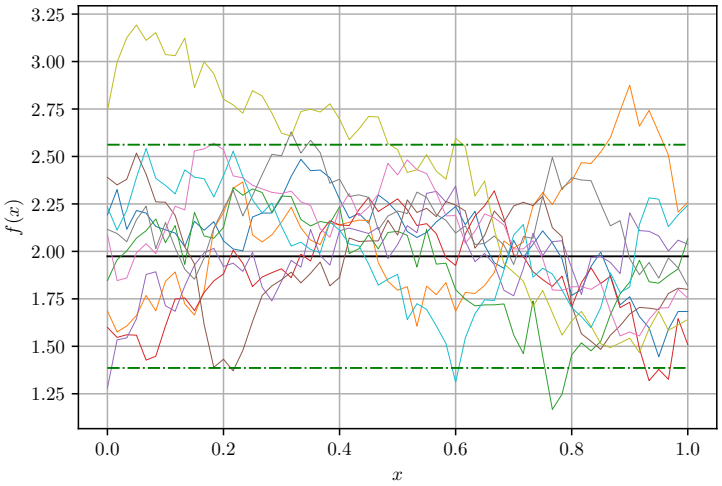


Figure 2.4: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 1/2$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

Periodic Kernel Functions

Combination of Different Kernel Functions

hyperparameters

hyperparameter learning

marginal likelihood: integral of the likelihood times the prior marginal means marginalization over the function values so these dont appear in the equation anymore. in this case it's marginalizing over the test points X so those dont appear anymore.

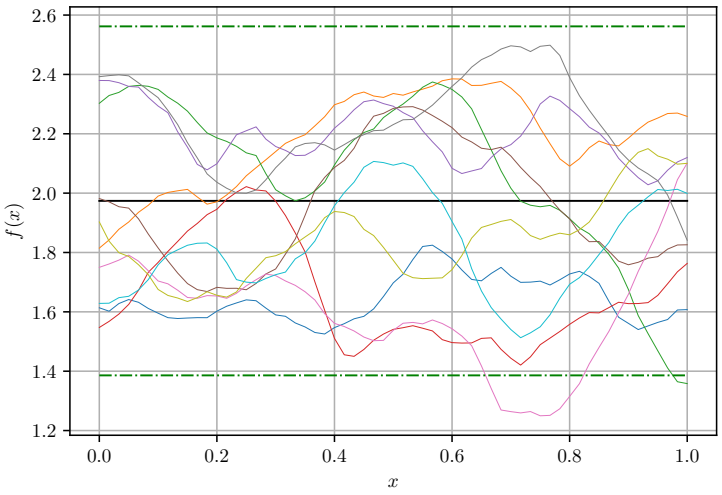


Figure 2.5: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 3/2$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

2.2.4 Prior and Posterior

Prior explain, how a sample from the multivariate gaussian canbe drawn (A2 Rasmussen). To obtain a sample from the mutlivariate 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

$$\boldsymbol{x} = \boldsymbol{m} + L\boldsymbol{u} \tag{2.17}$$

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, abut computationally demanding, to sample many many functions out of the GP and only take those which go through the training points.

explain conditioning

2.3 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

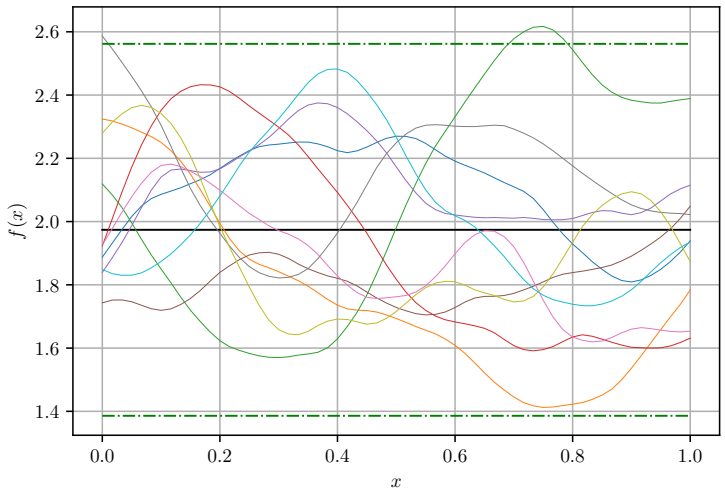


Figure 2.6: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 5/2$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

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 [13], [14] for a vector of n_y measurements $\boldsymbol{y} \in \mathbb{R}^{n_y}$,

$$\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{e} = \rho \boldsymbol{P}\boldsymbol{u} + \boldsymbol{d} + \boldsymbol{e}\,, \tag{2.18}$$

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

$$\boldsymbol{e} \sim p(\boldsymbol{e}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{e}}) \tag{2.19}$$

with zero mean and the covariance matrix $\boldsymbol{C}_{\boldsymbol{e}} = \sigma_e^2 \boldsymbol{I}$ 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

$$\boldsymbol{d} \sim p(\boldsymbol{d}|\sigma_d, l_d) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_{\boldsymbol{d}}) \tag{2.20}$$

which σ_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 $\boldsymbol{C}_{\boldsymbol{d}} \in \mathbb{R}^{n_y \times n_y}$.

Basic statFEM procedure statFEM can basically be broken down into three major steps [13]. 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.

2.3.1 Step 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)

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

```
def sample_f(self):
        c_f = self.matern_log(self.coordinates,
                self.coordinates, l=np.log(lf), sig=np.log(sigf))
        f_mean = (np.pi)**2*(1/5)*np.ones(self.ne+1)
        fGP = np.random.multivariate_normal(
                mean = f_mean, cov=c_f,
                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 κ is assumed to be a random variable and dependent on x, it can be modeled as a GP as well and there holds

$$\kappa \sim \mathcal{GP}(\bar{\kappa}, C_\kappa) \tag{2.21}$$

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,

$$u \sim \mathcal{GP}(\bar{u}, C_u) \tag{2.22}$$

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

$$\boldsymbol{u} = \boldsymbol{A}^{-1}\boldsymbol{f} \,. \tag{2.23}$$

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

$$\mathbb{E}\left[\sum_{i=1}^{n} X_i\right] = \sum_{i=1}^{n} \mathbb{E}[X_i] \,. \tag{2.24}$$

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

$$\bar{u} = \mathbb{E}[\boldsymbol{A}^{-1}\boldsymbol{f}] = \boldsymbol{A}^{-1}\mathbb{E}[\boldsymbol{f}] = \boldsymbol{A}^{-1}\bar{f} \tag{2.25}$$

and, acc. to [15], [16], making use of the definition of covariance (2.3),

$$\begin{aligned} C_u &= \mathbb{E}[(\boldsymbol{u} - \bar{u})(\boldsymbol{u} - \bar{u})^T] \\ &= \mathbb{E}\left[\left((\boldsymbol{A}^{-1}\boldsymbol{f}) - (\boldsymbol{A}^{-1}\bar{f})\right)\left((\boldsymbol{A}^{-1}\boldsymbol{f}) - (\boldsymbol{A}^{-1}\bar{f})\right)^T\right] \\ &= \mathbb{E}\left[\left(\boldsymbol{A}^{-1}(\boldsymbol{f} - \bar{f})\right)\left(\boldsymbol{A}^{-1}(\boldsymbol{f} - \bar{f})\right)^T\right] \\ &= \mathbb{E}\left[\boldsymbol{A}^{-1}(\boldsymbol{f} - \bar{f})(\boldsymbol{f} - \bar{f})^T\boldsymbol{A}^{-T}\right] \\ &= \boldsymbol{A}^{-1}\mathbb{E}\left[(\boldsymbol{f} - \bar{f})(\boldsymbol{f} - \bar{f})^T\right]\boldsymbol{A}^{-T} \\ &= \boldsymbol{A}^{-1}C_f\boldsymbol{A}^{-T} \,. \end{aligned} \tag{2.26}$$

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

$$\boldsymbol{u} \sim p(u) = \mathcal{N}\left(A^{-1}\bar{f}, A^{-1}C_f A^{-T}\right) \,. \tag{2.27}$$

As visible in (2.27), there holds for $\bar{u}$

$$\bar{u} = A^{-1}\bar{f} \,, \tag{2.28}$$

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

$$A^{-1}C_f A^{-T} \,. \tag{2.29}$$

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

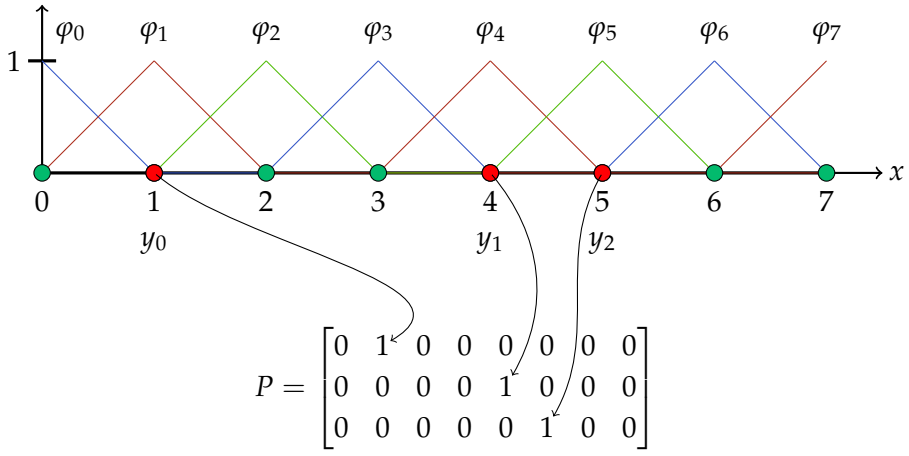
$$\boldsymbol{\sigma_u} = \mathrm{diag}(C_u). \tag{2.30}$$

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 (2.30).

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 2.7 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



$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.31}$$

Figure 2.7: 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.

can easily be implemented in 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.

2.3.2 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

$$\boldsymbol{y} = \boldsymbol{z} + \boldsymbol{e} \tag{2.32}$$

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 (2.18), 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 [13]:

$$p(\boldsymbol{u}|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})} \propto p(\boldsymbol{y}|\boldsymbol{u})p(\boldsymbol{u}) \; . \tag{2.33}$$

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 (2.18) and (2.5):

$$p(\boldsymbol{y}|\boldsymbol{u}) \sim \mathcal{N}(\rho P\boldsymbol{u}, C_d + C_e) \tag{2.34}$$

$$p(\boldsymbol{u}) \sim \mathcal{N}(\bar{\boldsymbol{u}}, C_u) \tag{2.35}$$

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

$$p(\boldsymbol{u}|\boldsymbol{y}) \propto \exp\Big((\rho P\boldsymbol{u} - \boldsymbol{y})^T (C_d + C_e)^{-1} (\rho P\boldsymbol{u} - \boldsymbol{y})\Big) \exp\Big((\bar{\boldsymbol{u}} - \boldsymbol{u})^T C_u^{-1}\Big) \tag{2.36}$$

Following [13], with

$$\begin{aligned} B &= \rho^2 P^T (C_d + C_e)^{-1} P + C_u^{-1} \\ \boldsymbol{a} &= \rho P^T (C_d + C_e)^{-1} \boldsymbol{y} + C_u^{-1} \bar{\boldsymbol{u}} \end{aligned} \tag{2.37}$$

there holds

$$\begin{aligned} p(\boldsymbol{u}|\boldsymbol{y}) &\propto \exp\Big(\boldsymbol{u}^T B \boldsymbol{u} - 2\boldsymbol{a}^T \boldsymbol{u} + ...\Big) \\ &= \exp\Big((B^{-1}\boldsymbol{a} - \boldsymbol{u})^T B (B^{-1}\boldsymbol{a} - \boldsymbol{u})\Big) \; . \end{aligned} \tag{2.38}$$

Comparing 2.38 to 2.5 it can be argued that 2.38 can be normalized to

$$p(\boldsymbol{u}|\boldsymbol{y}) = \frac{1}{\sqrt{(2\pi)^{n_u}|B|}} \exp\Big((B^{-1}\boldsymbol{a} - \boldsymbol{u})^T B (B^{-1}\boldsymbol{a} - \boldsymbol{u})\Big) \tag{2.39}$$

The result of this operation is

$$p(u|y) = \mathcal{N}(\bar{u}_{|y}, C_{u|y}) \tag{2.40}$$

with

$$\bar{\boldsymbol{u}}_{|y} = C_{\boldsymbol{u}|\boldsymbol{y}} \left(\rho P^T (C_d + C_e)^{-1} y + C_u^{-1} \bar{u}\right) \tag{2.41}$$

and

$$C_{\boldsymbol{u}|\boldsymbol{y}} = \left(\rho^2 P^T (C_d + C_e)^{-1} P + C_u^{-1}\right)^{-1} \; . \tag{2.42}$$

To avoid having to invert matrices with the size of $n_u \times n_u$, the Sherman-Morrison-Woodbury identity [17] is applied [13] in order to only invert matrices with the size of $n_y \times n_y$ which yields

$$C_{\boldsymbol{u}|\boldsymbol{y}} = C_u - C_u P^T \left(\frac{1}{\rho^2}(C_d + C_e) + P C_u P^T\right)^{-1} P \, C_u \; . \tag{2.43}$$

2.3.3 find Posterior mean and variance

2.3.4 Modeling Parameters of the PDE

2.3.5 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 2.3.5, 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

2.3.6 Step 2: 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 ρ is not known, as well. The measurement data can be used to estimate these parameters, collected in the hyperparameter vector $\boldsymbol{w}$, with Bayes' rule

$$p(\boldsymbol{w}|\boldsymbol{y}) = \frac{P(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})}{\int P(\boldsymbol{y}|\boldsymbol{w})p(\boldsymbol{w})\,\mathrm{d}\boldsymbol{w}} \tag{2.44}$$

where the denominator, which only serves as a normalization constant, can be dropped if only a point estimate of the hyperparameters is needed. This approach is 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 [Rasmussen p.9],

$$P(\boldsymbol{Y}|\boldsymbol{w}) = \prod_{i=1}^{n_o} p(\boldsymbol{y_i}|\boldsymbol{w}) \tag{2.45}$$

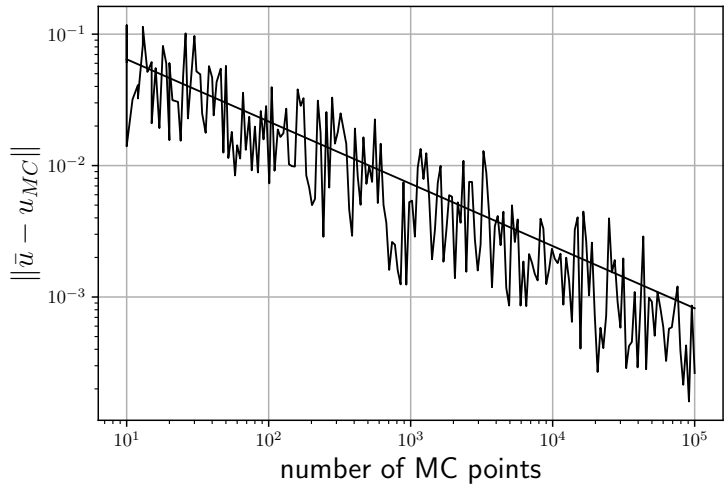


Figure 2.8: Convergence of the error norm for the FEM prior mean

which replaces $P(\boldsymbol{y}|\boldsymbol{w})$. If no prior knowledge of the parameters is available, it is possible to use an uninformative prior $p(\boldsymbol{w}) = 1$. 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 2.2.2, can be expressed as

$$p(\boldsymbol{y}|\boldsymbol{w}) = \mathcal{N}(\rho \boldsymbol{P}\bar{u}, \boldsymbol{C_d} + \boldsymbol{C_e} + \rho^2 \boldsymbol{P}\boldsymbol{C_u}\boldsymbol{P}^T) \tag{2.46}$$

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

$$p(\boldsymbol{y}|\boldsymbol{w}) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{K}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{y} - \rho\boldsymbol{P}\bar{u})^T \boldsymbol{K}^{-1}(y - \rho\boldsymbol{P}\bar{u})\right) \tag{2.47}$$

with

$$\boldsymbol{K} = \boldsymbol{C_d} + \boldsymbol{C_e} + \rho^2 \boldsymbol{P}\boldsymbol{C_u}\boldsymbol{P}^T \, . \tag{2.48}$$

To make further calculations easier the exponential function can be removed by taking the negative $\log$ of the function (Rasmussen):

$$-\log p(\boldsymbol{y}|\boldsymbol{w}) = \frac{n}{2}\log(2\pi) + \frac{1}{2}\log|\boldsymbol{K}| + \frac{1}{2}(\rho\boldsymbol{P}\bar{u} - \boldsymbol{y})^T \boldsymbol{K}^{-1}(\rho\boldsymbol{P}\bar{u} - \boldsymbol{y}) \tag{2.49}$$

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 $\log$, (2.49) now needs to be minimized instead of maximized.

Cholesky Decompositon [18, 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

$$\boldsymbol{K} = \boldsymbol{L}\boldsymbol{L}^T \tag{2.50}$$

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 (2.49) 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

$$\boldsymbol{K}\boldsymbol{K}^{-1}y = y \tag{2.51}$$

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 [19]. Therefore the lower triangular matrix $\boldsymbol{L}$ can be used to efficiently calculate the determinant of $\boldsymbol{K}$. There holds

$$\det\boldsymbol{K} = \det\boldsymbol{L}\det\boldsymbol{L}^T = \det\boldsymbol{L}^2 \tag{2.52}$$

and therefore

$$\det\boldsymbol{K} = \left(\prod_{i=1}^{n_y} \boldsymbol{L}_{i,i}\right)^2 \tag{2.53}$$

Since in (2.49) the $\log$ likelihood is used, (2.53) 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 (2.49) to be as small as possible has to be found. That can be achieved by using a gradient based optimizer or by sampling the parameter space with, e.g., a MCMC approach.

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. Metropolis algorithm as code.

L-BGFS 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 (2.49) is needed. It can be computed as follows.

2.3.7 Step 3: Estimating the Optimum Mesh Parameters

3 Application of statFEM in Vibroacoustics

3.1 Simple 1D example

Choice of PDE The Poisson equation

$$-\nabla \cdot \mu(x)\nabla u(x) = f(x) \tag{3.1}$$

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 [Cirak]

$$f(\boldsymbol{x}) \sim \mathcal{GP}\left(\bar{f}(\boldsymbol{x}), c_f(\boldsymbol{x}, \boldsymbol{x}')\right) . \tag{3.2}$$

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

$$c_f(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2l_f^2}\right) \tag{3.3}$$

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 [Murphy]:

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 (3.3) 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 [Rasmussen] 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 σ_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.

3.1.1 Posterior

Figure 3.1.1 shows that, if the observations are made slightly out of the FEM confidence band, a posterior can be inferred which fits to the data.

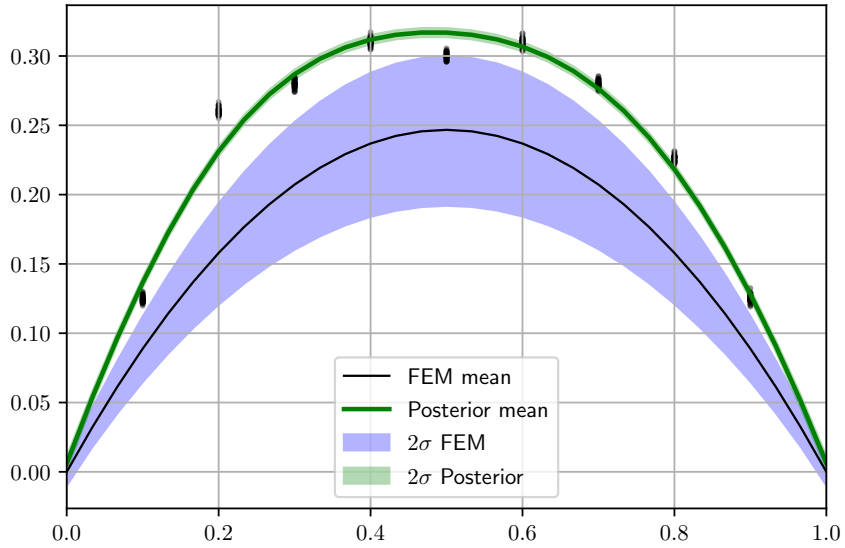


Figure 3.1: Per observation point 50 individual observations were used with a standard deviation of $2.5e-3$. 30 Elements.

Figure 3.1.1 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 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.

3.2 1D Vibroacoustics Example

Helmholtz Equation. Looks similar to the Poisson equation but the right side differs. It's the eigenvalue problem for the laplace operator.

3.2.1 FEM prior

Figure 3.2.1 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.

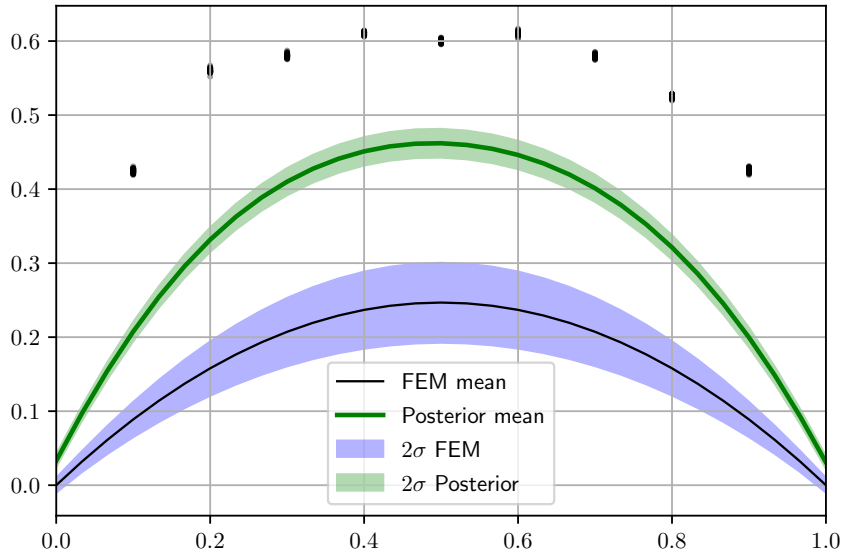


Figure 3.2: Per observation point 50 individual observations were used with a standard deviation of $2.5e-3$. 30 Elements.

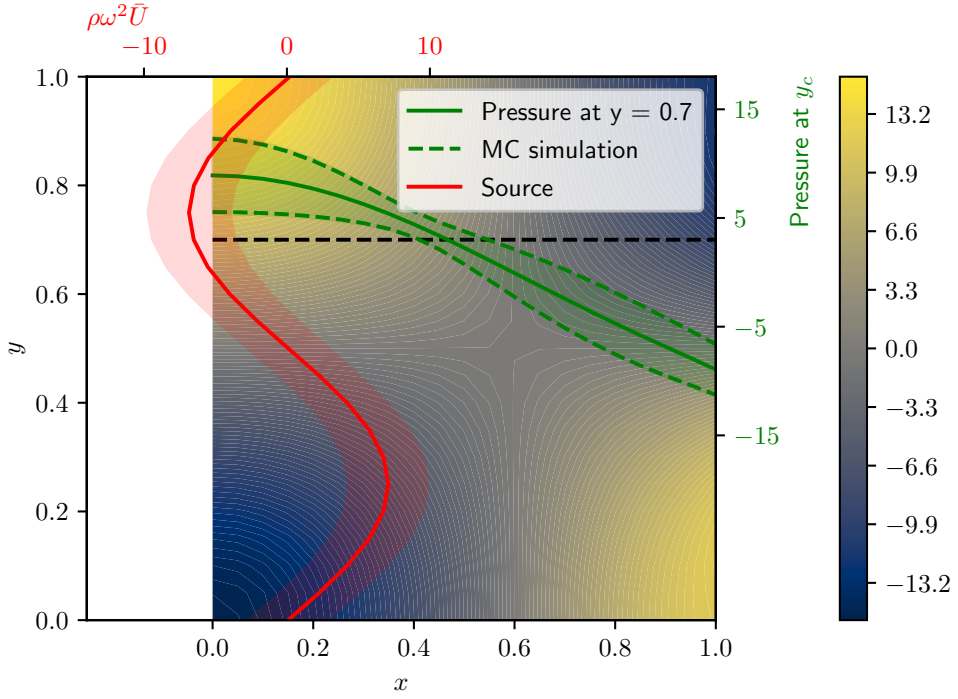


Figure 3.3: The source GP, the mean prior and, at the cut $y = y_c$, the prior FEM GP for 222Hz.

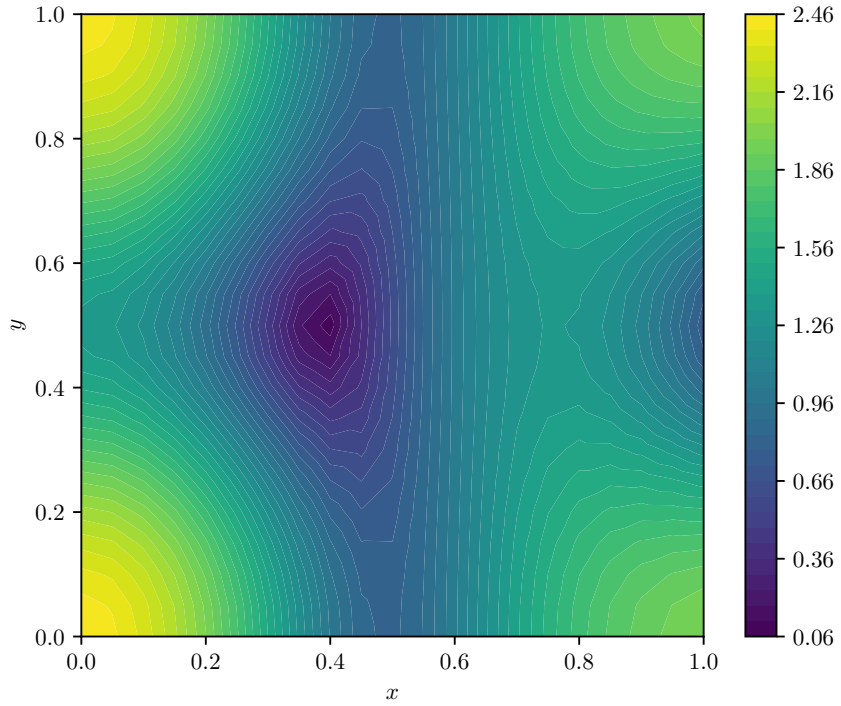


Figure 3.4: The variance of the FEM prior for $222Hz$ at all points in the 2D space. Regions with high and low variance are clearly visible.

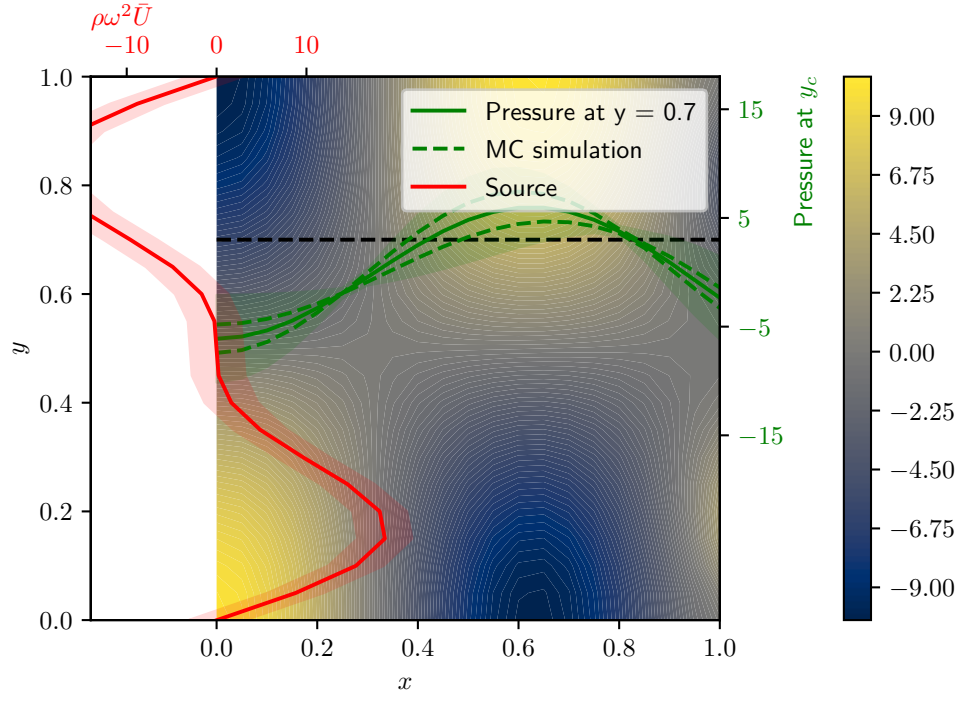


Figure 3.5: The source GP, the mean prior and, at the cut $y = y_c$, the prior FEM GP for 320Hz.

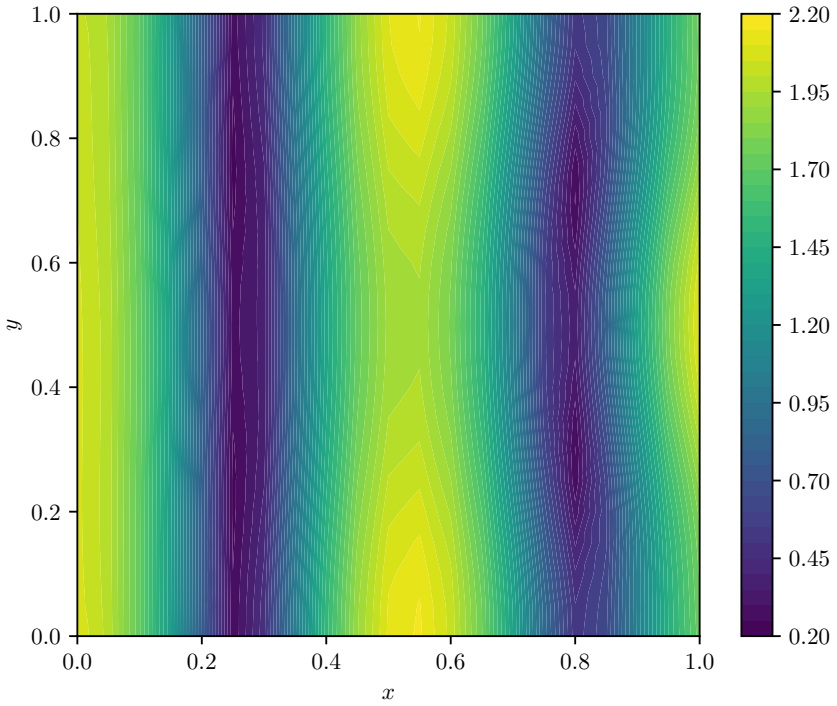


Figure 3.6: The variance of the FEM prior for $320Hz$ at all points in the 2D space. Regions with high and low variance are clearly visible.

4 Conclusion and Discussion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

4.1 Section 1

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non

odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Bibliography

[1] 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., ISBN: 978-3-642-33286-9 978-3-642-33287-6 (cit. on pp. 6, 7).

[2] 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).

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

[4] 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, 16, 26).

[5] 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, 7, 11).

[6] H. P. Langtangen and A. Logg, *Solving PDEs in Python*. Cham: Springer International Publishing, 2016, ISBN: 978-3-319-52461-0 978-3-319-52462-7. 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).

[8] 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 p. 18).

[9] 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).

[10] 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. 20).

[11] 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. 20).

[12] 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. 21).

[13] 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. 23, 24, 29).

[14] 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 p. 23).

[15] 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. 26).

[16] 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. 26).

[17] 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. 29).

[18] 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. 31).

[19] 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. 32).