



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, June 17, 2021

Lucas Hermann

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur 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	Introduction	5
2	FEM Solution of the Helmholtz Equation	6
2.1	Wave Equation and Helmholtz Equation	6
2.2	Boundary Conditions	7
2.3	Weak Formulation and Discretisation	9
2.4	The Classical Finite Element Method	10
2.4.1	Integral and Weak Formulation	10
2.4.2	Discretization	11
2.4.3	Approximation and Building Element Matrices	12
2.4.4	Assembling and Solving the System	13
2.4.5	Setting Constraints	14
2.4.6	Solving and Convergence	14
3	Gaussian Processes and the Statistical Finite Element Method	15
3.1	Basic Probability Theory	15
3.1.1	Gaussian Distribution	15
3.1.2	Bayesian Inference	16
3.2	Gaussian Process Regression	16
3.2.1	Regression	16
3.2.2	Marginalization and Conditioning	17
3.2.3	Gaussian Processes	17
3.2.4	Kernel Functions	17
3.2.5	Prior before observation	20
3.2.6	Posterior after observation	20
3.3	The Statistical Finite Element Method	21
3.3.1	Step 1: Posterior of the FEM Forward Model	22
3.3.2	Inference of the Posterior Density	26
3.3.3	find Posterior mean and variance	26
3.3.4	Modeling Parameters of the PDE	26
3.3.5	Probabilistic FEM Prior	26
3.3.6	Step 2: Estimation of Hyperparameters	27
3.4	Step 3: Estimating the Optimum Mesh Parameters	29
4	Application of statFEM in Vibroacoustics	30
4.1	Simple 1D example	30
4.2	1D Vibroacoustics Example	31
5	Conclusion and Discussion	32
5.1	Section 1	32

1 Introduction

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

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur 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

2 FEM Solution of the Helmholtz Equation

An acoustic field is represented by the wave equation. It returns the acoustical pressure at a point in space X for a certain frequency and time. The Helmholtz equation describes the wave equation in a time-independent manner. 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.

2.1 Wave Equation and Helmholtz Equation

In Vibroacoustics 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 the Bernoulli or Timoshenko beam theory in 1D or, for the 2D case of plates, Kirchhoff or Reissner Mindlin plate theory. 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 a liquid or gas, such as air. The following closely follows [Mats G Larson FEM] and [Möser]. 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 propagated through the medium: One can think of the gas as a system of infinitesimally small spring-mass systems. Deflecting one spring yields in 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 neighbouring particle [Möser]

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

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

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

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

$$\rho \dot{v} = -\nabla p . \quad (2.3)$$

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.[Larson FEM] 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

$$\dot{p} = -k \nabla v \quad (2.4)$$

with k a proportionality constant dependent on the medium. Differentiating 2.4 after t yields

$$\ddot{p} = -k \nabla \dot{v} . \quad (2.5)$$

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

$$\ddot{p} = c^2 \nabla^2 p \quad (2.6)$$

which involves derivations 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 [Westermann]

$$p(x, t) = X(x) \cdot T(t) , \quad (2.7)$$

there holds for 2.6

$$X(x) \cdot \frac{\partial^2 T(t)}{\partial t^2} = c^2 \nabla^2 X(x) \cdot T(t) \quad (2.8)$$

$$\frac{1}{T(t)} \frac{\partial^2 T(t)}{\partial t^2} = c^2 \frac{1}{X(x)} \nabla^2 X(x) = \text{const.} = -\omega^2 . \quad (2.9)$$

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

$$\nabla^2 X(x) + k^2 X(x) = 0 \quad (2.10)$$

is the Helmholtz equation which is a representation of the wave equation independent of time [Mathebuch Helmholtz]. In the following it is going to be referred to by

$$\nabla^2 p + k^2 p = 0 \quad (2.11)$$

with $p[\text{Pa}]$ the pressure field and k the wave number. k is defined as

$$k = \frac{\omega}{c_0} = \frac{2\pi f}{c_0} \quad (2.12)$$

with $\omega[\text{rad}]$ the angular frequency, $f[\text{s}^{-1}]$ the frequency and $c_0[\text{m s}^{-1}]$ the speed of sound in the considered medium. (2.11) is solved as a boundary value problem.

2.2 Boundary Conditions

[ATALLA]

Dirichlet The Dirichlet boundary condition for the Helmholtz equation is simply a prescribed pressure at the Γ_D part of the boundary:

$$p = \bar{p} \quad (2.13)$$

Neumann The Neumann boundary condition at Γ_N reads

$$\frac{dp}{dn} = \rho \omega^2 \mathbf{U} \cdot \vec{n} \quad (2.14)$$

with \mathbf{U} the displacement and \vec{n} the normal direction of the surface. It can be thought of as a piston moving with circular frequency ω at the boundary which created a velocity and pressure change. Choosing a Neumann BC $\Gamma_N = 0$ resembles a reflecting boundary.

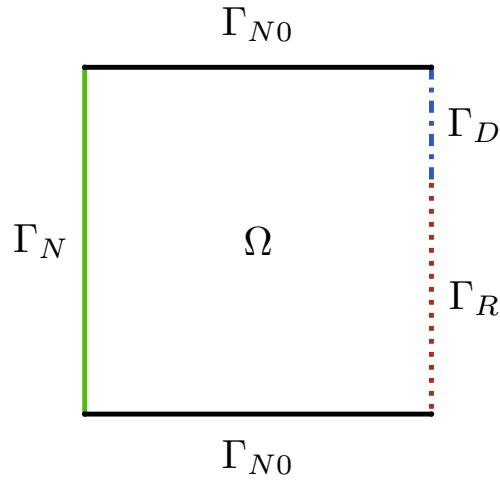


Figure 2.1: Boundary Conditions of different type can be arranged on the boundary. In this example, the left side is a Neumann boundary, the upper and lower boundaries are totally reflecting and the right side consists of an impedance BC and a Dirichlet BC.

Robin The Robin or impedance boundary condition at Γ_R reads

$$\frac{dp}{dn} + ik\beta p = 0 \quad (2.15)$$

with β the specific normalized acoustic admittance. Figure 2.1 shows an example on where boundary conditions could be applied on a 2D domain. Figure 2.2 shows the approach chosen in this thesis.

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)
5
6 class BoundaryX_R(SubDomain):
7     def inside(self, x, on_boundary):
8         return on_boundary and near(x[0], 1, tol) # and (x[1] < 0.3)
9
10 boundary_markers.set_all(9999) # all markers default to zero.
11 bxL = BoundaryX_L()
12 bxR = BoundaryX_R()
13 bxL.mark(boundary_markers, 0) # left side is marked as "0"
14 bxR.mark(boundary_markers, 1) # right side is marked as "1"
15
16 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:

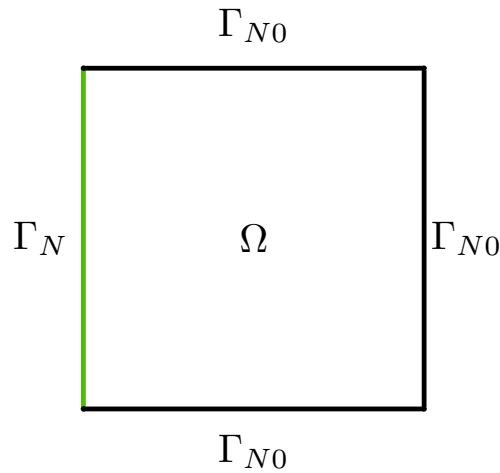


Figure 2.2: Boundary Conditions of the problem for this thesis. A Neumann boundary is applied on the side of the domain. The rest of the boundary is considered as reflecting, i.e. $\Gamma_N = 0$.

```

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

```

Dirichlet boundary conditions can be applied after assembling the system by

```

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

```

2.3 Weak Formulation and Discretisation

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. (2.11) 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:

2.4 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, it 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.

2.4.1 Integral and Weak Formulation

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 achieved by first bringing the equation in the weak, or also variational, formulation. Bringing the governing equation in the weak form can be accomplished using the Method of Weighted Residuals.

The Weigthed Residuals Method At first the strong formulation (PDE) is multiplied with a so-called test function v and integrated over the domain Ω to get the so-called integral formulation:

$$\int_{\Omega} \nabla^2 p v \, d\Omega + \int_{\Omega} k^2 p v \, d\Omega = 0 \quad (2.16)$$

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, 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 boundary conditions imposed on the PDE. The Helmholtz Equation and therefore also the now formed integral formulation contain derivations of second order. However, the FEM works by adjoining individual element functions to a global function. Therefore the global solution is a piecewise polynomial (FEM Lang/Mard) what causes problems when calculating higher-order derivatives because those become discontinuous. In consequence, integration by parts is performed on the higher-order terms to get lower order differentiations:

$$- \int_{\Omega} \nabla p \cdot \nabla v \, d\Omega + \int_{\Omega} \nabla^2 p \cdot v \, d\Omega + \int_{\Omega} k^2 p v \, d\Omega = 0 \quad (2.17)$$

The now still present second derivative can be eliminated by using the divergence theorem.

Divergence Theorem explain what the divergence theorem is and apply it to the weak form. explain the analogy with the container which is filled over the domain boundary. For the divergence theorem, also called Gauss's theorem, there holds in general:

$$\int_V \nabla F \, dv = \oint_A F \cdot \vec{n} \, da \quad (2.18)$$

It describes 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. For (2.17) it yields

$$- \int_{\Omega} \nabla p \cdot \nabla v \, d\Omega + \oint_{\Gamma} \nabla p \cdot v \, d\Gamma + \int_{\Omega} k^2 p v \, d\Omega = 0 \quad (2.19)$$

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 instead of twice. It is usual to make a weak formulation have only first order derivatives.

2.4.2 Discretization

Discretization means that the domain Ω is split into n_e individual elements $\Omega^{(e)}$

$$\Omega = \bigcup_{e=1}^{n_e} \Omega^{(e)} \quad e = 1, \dots, n_e. \quad (2.20)$$

For linear elements which are used here, 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. A quadratic element would have 3 nodes. The collection of nodes is called a mesh. The total number of nodes in a 1 dimensional mesh is $n_e + 1$. 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. [ATALLA]

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 is 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

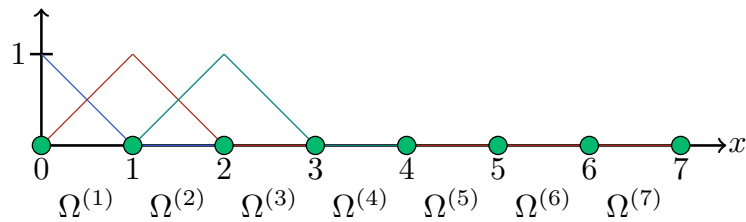


Figure 2.3: A discretized 1D domain with nodes in green and linear ansatz functions shown for the first four nodes

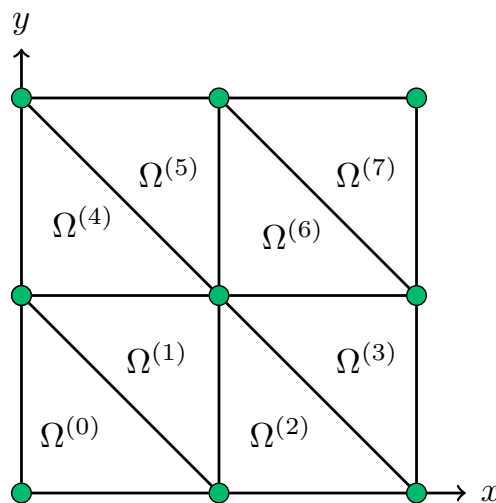


Figure 2.4: A discretized 2D domain with nodes in green

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

```
1 self.dom_a = 0.0 #domain boundaries
2 self.dom_b = 1.0
3 self.ne     = 60 #number of elements
4 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

```
1 self.a, self.b = 2, 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.

2.4.3 Approximation and Building Element Matrices

In the FEM, the values between nodes within an element are interpolated with basis functions φ_i . For each node in the domain an FEM basis function is defined. By weighing those basis functions, the solution can be approximated with

$$p(x_i) = \sum_{j \in \Omega} c_j \varphi_j(x_i) = f(x_i) . \quad (2.21)$$

(2.21) 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, (2.21) simplifies as described below. One popular choice of orthogonal 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 [Langtangen FEM] 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 orthogonal to one another. They are used as an interpolation between given points which means that they polynomial always exactly goes through the given points. The behaviour at all other points is determined by the definition of the Lagrange polynomial

$$\varphi_i(x) = \prod_{j=0, j \neq i}^N \frac{x - x_j}{x_i - x_j} . \quad (2.22)$$

They are frequently used in FEM because there holds

$$\varphi_i(x_s) = \delta_{is}, \quad \delta_{is} = \begin{cases} 1 & \text{for } i = s \\ 0 & \text{for } i \neq s \end{cases} \quad (2.23)$$

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 (2.21):

$$p(x_i) = c_i \varphi_i(x_i) = c_i . \quad (2.24)$$

The weight factor is therefore the true value at the node. Another asset of using orthogonal polynomials is that $\varphi_i(x)$ is 0 in most parts of the domain what leads to diagonal matrices which can be solved much quicker than densely packed ones. The goal is to solve the linear system of equations for the weight factors c_i and therefore $p(x)$. It can be notated as

$$\sum_j A_{i,j} c_j = b_i \quad (2.25)$$

with

$$A_{i,j} = \varphi_j(x_i), \quad (2.26)$$

$$b_i = f(x_i). \quad (2.27)$$

The calculation is usually done for one element at a time. Therby arises an element matrix $A^{(e)}$ whose individual entries can be computed as

$$A_{r,s}^{(e)} = \int_{\Omega^{(e)}} \varphi_r \varphi_s \, dx \quad (2.28)$$

with i, j the numbers of the individual nodes in an element. For linear elements, i.e. elements with two nodes each, a 2×2 matrix results and for quadratic elements with three nodes each a 3×3 matrix results.

Also the right hand side vector b is computed element-wise:

$$b_r^{(e)} = \int_{\Omega^{(e)}} f(x) \varphi_r \, dx. \quad (2.29)$$

2.4.4 Assembling and Solving the System

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

$$A_{q(e,r),q(e,s)} := A_{q(e,r),q(e,s)} + A_{r,s}^{(e)} \quad (2.30)$$

through which can be iterated using a loop in Python. The resulting matrix is sparse, here for three elements with linear ansatz functions:

$$A_{q(e,r),q(e,s)} = \begin{bmatrix} A_{0,0}^{(0)} & A_{0,1}^{(0)} & 0 & 0 \\ A_{1,0}^{(0)} & A_{1,1}^{(0)} + A_{0,0}^{(1)} & A_{0,1}^{(1)} & 0 \\ 0 & A_{1,0}^{(1)} & A_{1,1}^{(1)} + A_{0,0}^{(2)} & A_{0,1}^{(2)} \\ 0 & 0 & A_{1,0}^{(2)} & A_{1,1}^{(2)} \end{bmatrix} \quad (2.31)$$

For the global right-hand-side vector there holds

$$b_{q(e,r)} := b_{q(e,r)} + b_r^{(e)}. \quad (2.32)$$

With FEniCS, the system is assembled by

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

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

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

2.4.5 Setting Constraints

2.4.6 Solving and Convergence

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

3 Gaussian Processes and the Statistical Finite Element Method

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 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

$$\mu = \mathbb{E}[X] = \int_{\mathbb{R}} x f_X(x) dx \quad (3.1)$$

and for the variance

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

For the PDF of the Gaussian distribution there holds

$$f_X(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \quad (3.3)$$

If multiple random variables are observed, which are all normally distributed, they can be aggregated in a so-called multivariate Gaussian distribution. 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

$$f_X(\mathbf{x}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \quad (3.4)$$

with $\boldsymbol{\Sigma}$ 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

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & -0.7 \\ -0.7 & 1.5 \end{bmatrix}. \quad (3.5)$$

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

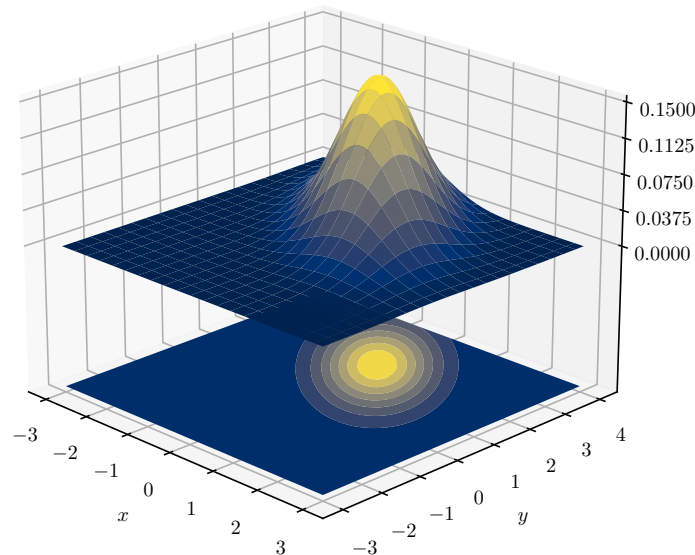


Figure 3.1: A multivariate Gaussian distribution with two dimensions. A strong correlation between the two random variables is visible. [scipy tutorial]

3.1.2 Bayesian Inference

$posterior = \frac{likelihood \times prior}{marginal likelihood}$ (Rasmussen p.9) explain basics of bayesian inference with bayes rule etc. what is posterior/prior explain what the marginal likelihood is

Marginal Likelihood

3.2 Gaussian Process Regression

3.2.1 Regression

"Regression is used to find a function that represents a set of data points as closely as possible" [Visual Exploration]

[Visual Exploration] "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 [Rasmussen].

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

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

3.2.2 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

$$P(X, Y) = \begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix} \right) \quad (3.6)$$

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

$$X \sim \mathcal{N}(\mu_X, \Sigma_{XX}) . \quad (3.7)$$

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

$$p_X(x) = \int_y p_{X,Y}(x, y) dy = \int_y p_{X|Y}(x|y) p_Y(y) dy \quad (3.8)$$

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

(3.8) 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

$$X|Y \sim \mathcal{N}(\mu_X + \Sigma_{XY} \Sigma_{YY}^{-1} (Y - \mu_Y), \Sigma_{XX} - \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX}) . \quad (3.9)$$

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 3.2.

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

-a distribution over functions

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

-parameter variations

3.2.3 Gaussian Processes

3.2.4 Kernel Functions

p14:

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.

p79: 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.

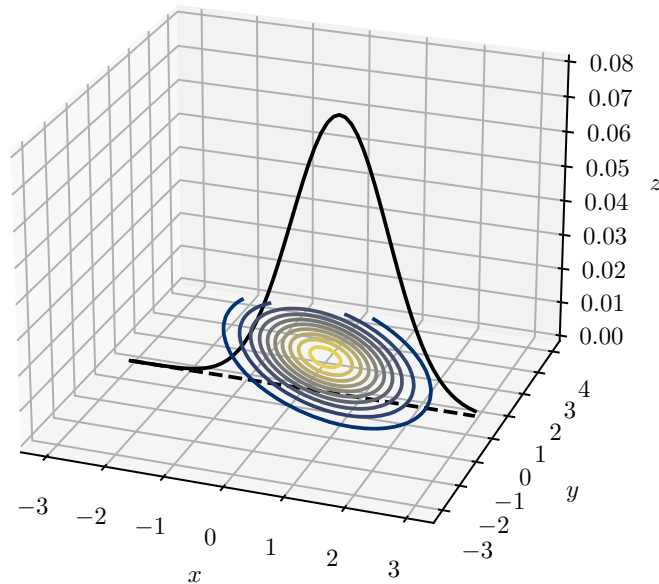


Figure 3.2: Cutting, i.e. conditioning, a bivariate Gaussian at the dashed line leads to a new Gaussian with only one dimension.

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: $1100mm - 1000mm = 100mm$, $100100mm - 100000mm = 100mm$

p8o: 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.

Covariance matrices always have to be symmetric, i.e. $k(x - x') = k(x' - x)$, since everything else wouldn't 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 function, 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

$$k_{SE} = \sigma \exp\left(-\frac{r^2}{2l^2}\right) \quad (3.10)$$

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

```
1 def squared_exponential(xa, xb, l, sig):
2     sq_norm = -0.5 * scipy.spatial.distance.cdist(xa, xb, 'sqeuclidean') \
3         * (1/l**2)
4     return sig**2 * np.exp(sq_norm) .
```

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

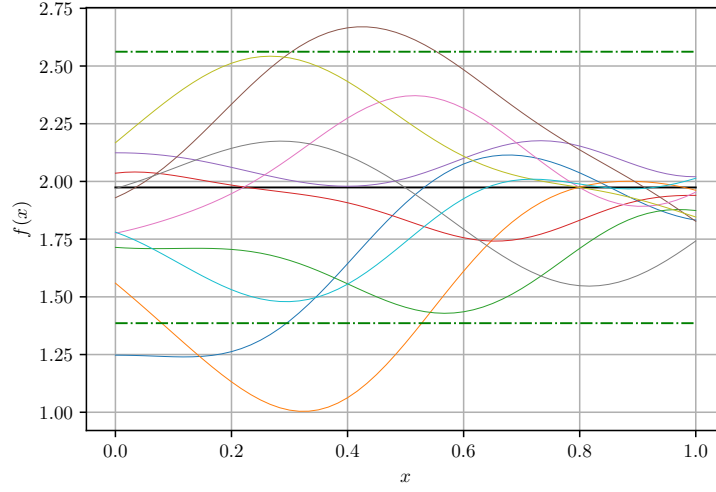


Figure 3.3: The source term GP of $f(x)$ sampled for a Squared Exponential kernel with $l = 0.25$ and $\sigma = 0.3$. $\tilde{f}(x)$ in solid black, 2σ confidence bands in green.

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 [Rasmussen] The squared exponential kernel is infinitely differentiable, which means that it behaves very smoothly. This is considered as unphysical by [Stein]. Therefore, the Matern class of covariance functions is recommended 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

$$k_M(r) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}r}{l} \right)^\nu K_\nu \left(\frac{\sqrt{2\nu}r}{l} \right) \quad (3.11)$$

with $\Gamma(\nu) = (\nu - 1)!$ the gamma function and K_ν a modified Bessel function [Abramowitz Stegun] [Rasmussen]. ν and l are free positive parameters. For $\nu \rightarrow \infty$ (3.10) 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 [Rasmussen] $\nu = 3/2$ and $\nu = 5/2$ are commonly used for GPs; the simplest Matern kernel is obtained with $\nu = 1/2$:

$$k_{M;\nu=1/2}(r) = \sigma^2 \exp\left(-\frac{r}{l}\right), \quad (3.12)$$

$$k_{M;\nu=3/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right), \quad (3.13)$$

$$k_{M;\nu=5/2}(r) = \sigma^2 \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right). \quad (3.14)$$

(3.12) - (3.13) are used in this thesis. Figures 3.4 - 3.6 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.

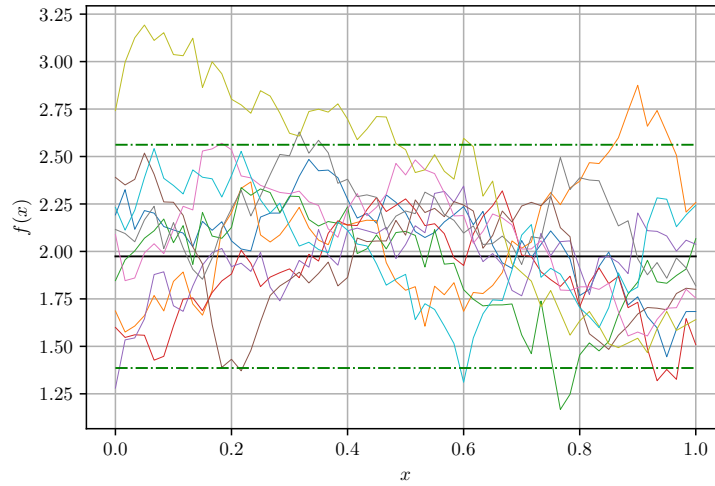


Figure 3.4: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 1/2$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

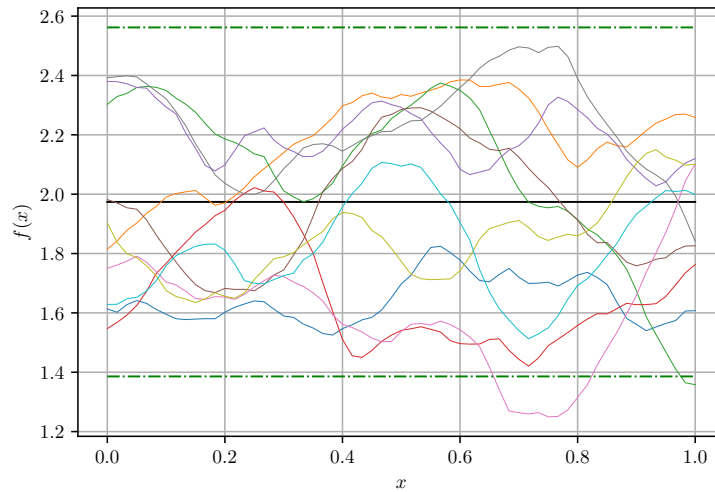


Figure 3.5: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 3/2$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

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.

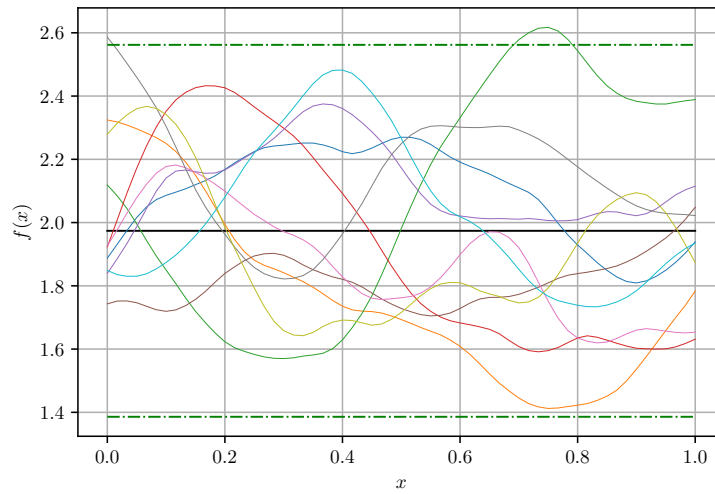


Figure 3.6: The source term GP of $f(x)$ sampled for a Matern kernel with $\nu = 5/2$. $\bar{f}(x)$ in solid black, 2σ confidence bands in green.

3.2.5 Prior before observation

explain, how a sample from the multivariate gaussian can be drawn (A2 Rasmussen). 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 $u \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is drawn. The final sample x can now be obtained using

$$x = m + Lu \quad (3.15)$$

with the prescribed mean vector m . With Python and numpy, the sample is obtained with

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

The numpy function does the Cholesky decomposition internally.

Occam's Razor

3.2.6 Posterior after observation

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

explain conditioning

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 inferred. 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 (Cirak) for a vector of n_y measurements $\mathbf{y} \in \mathbb{R}^{n_y}$,

$$\mathbf{y} = \mathbf{z} + \mathbf{e} = \rho \mathbf{P} \mathbf{u} + \mathbf{d} + \mathbf{e}, \quad (3.16)$$

consists of response of the true underlying process $\mathbf{z} \in \mathbb{R}^{n_y}$ at the measurement points and the measurement noise $\mathbf{e} \in \mathbb{R}^{n_y}$. The true process \mathbf{z} can further be expressed as a combination of the used model $\mathbf{u} \in \mathbb{R}^{n_u}$, which tries to describe the true process, and a model discrepancy error \mathbf{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 $\mathbf{P} \in \mathbb{R}^{n_y \times n_u}$. \mathbf{e} is modeled as a multivariate Gaussian

$$\mathbf{e} \sim p(\mathbf{e}) = \mathcal{N}(\mathbf{0}, \mathbf{C}_e) \quad (3.17)$$

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

$$\mathbf{d} \sim p(\mathbf{d} | \sigma_d, l_d) = \mathcal{N}(\mathbf{0}, \mathbf{C}_d) \quad (3.18)$$

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 $\mathbf{C}_d \in \mathbb{R}^{n_y \times n_y}$.

Basic statFEM procedure statFEM can basically be broken down into three major steps[Cirak, abstract]. 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 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.

```
1 def __init__(self, nMC = 200):
2     self.dom_a = 0.0 #domain boundaries
```

```

3 self.dom_b = 1.0
4 self.ne = 100 #number of elements
5 self.mesh = IntervalMesh(self.ne, self.dom_a, self.dom_b) #define mesh
6 self.bbt = self.mesh.bounding_box_tree()
7 self.coordinates = self.mesh.coordinates() # vertices vector
8 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 $A = \text{assemble}(a)$. The boundary conditions are applied to A after assembly.

```

1 def doFEM(self):
2     # Define Dirichlet boundary (x = 0 or x = 1)
3     def boundary(x):
4         return x[0] < (self.dom_a + DOLFIN_EPS) or x[0] > self.dom_b - DOLFIN_EPS
5     # Define boundary condition
6     u0 = Constant(0.0)
7     self.bc = DirichletBC(self.V, u0, boundary)
8     # Define variational problem
9     self.u = TrialFunction(self.V)
10    self.v = TestFunction(self.V)
11    # find index of cell which contains point.
12    class DoGP(UserExpression):
13        def eval(self, value, x):
14            collisions1st = bbt.compute_first_entity_collision(Point(x[0]))
15            value[0] = f_vals[collisions1st] # f is constant in a cell.
16        def value_shape(self):
17            return ()
18    f_vals = self.draw_FEM_samples(coordinates)
19    f_vals = f_vals[0]
20    f = DoGP()
21
22    a = dot(grad(self.u), grad(self.v))*dx #variational Problem
23    L = f*self.v*dx
24
25    A = assemble(a)
26    b = assemble(L)
27    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 κ is assumed to be a random variable and dependent on x , it can be modeled as a GP as well and there holds

$$\kappa \sim \mathcal{GP}(\bar{\kappa}, C_\kappa) \quad (3.19)$$

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,

$$u \sim \mathcal{GP}(\bar{u}, C_u) \quad (3.20)$$

is to be calculated. Expressed in terms of the FEM system matrix, there holds

$$u = A^{-1}f \quad (3.21)$$

Because of discretizing, the multivariate Gaussian

$$u \sim p(u) = \mathcal{N}\left(A^{-1}\bar{f}, A^{-1}C_f A^{-T}\right) \quad (3.22)$$

arises. As visible in (3.21), there holds for \bar{u}

$$\bar{u} = A^{-1}\bar{f}, \quad (3.23)$$

which can be computed using Python by calling

```

1 def get_U_mean(self):
2     u_mean = Function(self.V)
3     U_mean = u_mean.vector()
4     b_mean = assemble(f_mean*self.v*dx)
5     self.bc.apply(b_mean)
6     solstd, A = self.doFEM() # solstd is unused here.
7     solve(A, U_mean, b_mean)
8     return U_mean .

```


C_u is obtained by calculating

$$A^{-1}C_fA^{-T}. \quad (3.24)$$

In Python code there holds

```

1 def get_C_u(self):
2     C_f = self.get_C_f()
3     solstd,A = self.doFEM()
4     ident = np.identity(len(self.coordinates))
5     self.bc.apply(A)
6     A = A.array()
7     A_inv = np.linalg.solve(A,ident)
8     thresh = 1e-16
9     C_u = np.dot( np.dot(A_inv,C_f), np.transpose(A_inv))
10    C_u = C_u + 1e-16*(self.sigf**2)*ident
11    c_u = np.transpose(self.integratedTestF) * C_u * self.integratedTestF
12    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

$$\sigma_u = \text{diag}(C_u) \quad (3.25)$$

In Python

```

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

```

solves (3.24).

The Projection Matrix P The projection matrix P is constructed by evaluating all FEM ansatz functions of the mesh at the n_y measurement positions. The result is a sparse matrix with most entries zero and only those entries non zero that correspond to the ansatz functions $\Phi_i(x)$ associated with the finite element in which the measurement positions lie. This computation for linear ansatz functions can easily be implemented in Python using Fenics:

```

1 def getP(self,y_points):
2     ny = len(y_points)
3     P = np.zeros((ny, self.ne+1), dtype = float) #with ne the number of elements
4     for j,point in enumerate(y_points):
5         x = np.array([point])
6         x_point = Point(x)
7         bbt = self.mesh.bounding_box_tree()
8         cell_id = bbt.compute_first_entity_collision(x_point)
9         cell = Cell(self.mesh, cell_id)
10        coordinate_dofs = cell.get_vertex_coordinates()
11        values = np.ones(1, dtype=float)
12        for i in range(self.el.space_dimension()):
13            phi_x = self.el.evaluate_basis(np.array(i),x
14                ,np.array(coordinate_dofs),cell.orientation())
15            P[j,cell_id+i] = phi_x
16        self.Pu = np.dot(P,self.U_mean)
17        return P

```

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

```

1 def get_C_e(self,size):
2     sige_square = 2.5e-5
3     C_e = sige_square * np.identity(size)
4     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

```

1 def get_C_d(self,y_points,ld, sigd):
2     C_d = self.matern_log(y_points, y_points, l=ld, sig=sigd)
3     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

$$\mathbf{y} = \mathbf{z} + \mathbf{e} \quad (3.26)$$

with \mathbf{z} the real response and \mathbf{e} the measurement error which is also a part of \mathbf{y} for the synthetic observations. It can be seen that the model mismatch error \mathbf{d} is, as opposed to eq. (), not part of the equation because it is only part of the FEM modeling error. The data is observed on multiple measurement points n_y 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. The result of this operation is

$$p(\mathbf{u}|\mathbf{y}) = \mathcal{N}(\bar{\mathbf{u}}_{|\mathbf{y}}, \mathbf{C}_{u|\mathbf{y}}) \quad (3.27)$$

with

$$\bar{\mathbf{u}}_{|\mathbf{y}} = \mathbf{C}_{u|\mathbf{y}} \left(\rho \mathbf{P}^T (\mathbf{C}_d + \mathbf{C}_e)^{-1} \mathbf{y} + \mathbf{C}_u^{-1} \bar{\mathbf{u}} \right) \quad (3.28)$$

and

$$\mathbf{C}_{u|\mathbf{y}} = \left(\rho^2 \mathbf{P}^T (\mathbf{C}_d + \mathbf{C}_e)^{-1} \mathbf{P} + \mathbf{C}_u^{-1} \right)^{-1} \quad (3.29)$$

3.3.3 find Posterior mean and variance

3.3.4 Modeling Parameters of the PDE

3.3.5 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 GP hyperparameters 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 3.3.5, 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

```

1 def samplePrior(self):
2     U_mean = self.get_U_mean()
3     C_u = self.get_C_u()
4     priorGP = np.random.multivariate_normal(
5         mean = U_mean, cov=C_u,
6         size=self.nMC)
7     return priorGP

```

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

Code for calculating Cu direct calculation MC approximation error bands

3.3.6 Step 2: Estimation of Hyperparameters

Below it was already stated that the hyperparameters for computing the model discrepancy covariance matrix 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 w , with Bayes' rule

$$p(w|y) = \frac{P(y|w)p(w)}{\int P(y|w)p(w)dw} \quad (3.30)$$

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],

$$P(Y|w) = \prod_{i=1}^{n_o} p(y_i) \quad (3.31)$$

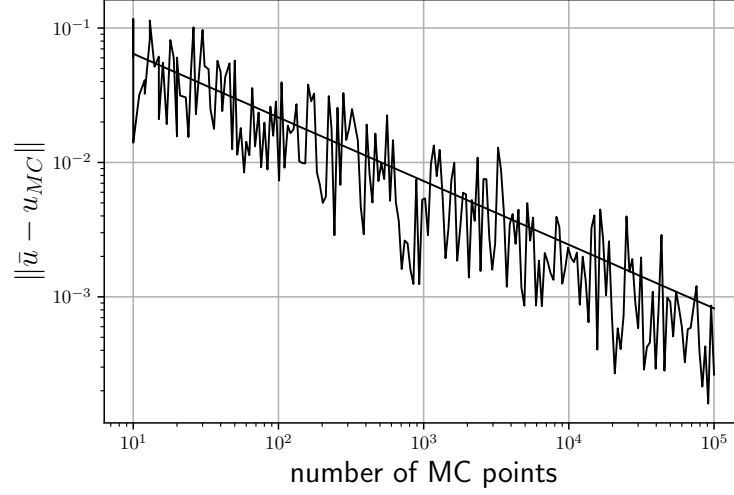


Figure 3.7: Convergence of the error norm for the FEM prior mean

which replaces $P(y|w)$. If no prior knowledge of the parameters is available, it is possible to use an uninformative prior $p(w) = 1$. The equation then basically states that $p(w|Y) \propto P(Y|w)$ which means that maximizing $P(Y|w)$, 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 $P(y|w)$, marginal because Y , can be expressed as

$$p(y|w) = \mathcal{N}(\rho P\bar{u}, C_d + C_e + \rho^2 P C_u P^T) \quad (3.32)$$

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

$$p(y|w) = \frac{1}{(2\pi)^{n/2} |K|^{1/2}} \exp \left(-\frac{1}{2} (y - \rho P\bar{u})^T K^{-1} (y - \rho P\bar{u}) \right) \quad (3.33)$$

with

$$K = C_d + C_e + \rho^2 P C_u P^T. \quad (3.34)$$

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

$$-\log p(y|w) = \frac{n}{2} \log(2\pi) + \frac{1}{2} \log |K| + \frac{1}{2} (\rho P\bar{u} - y)^T K^{-1} (\rho P\bar{u} - y) \quad (3.35)$$

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, (3.34) now needs to be minimized instead of maximized.

Cholesky Decomposition [Gentle 1998 p.93-95] For numerical stability reasons K^{-1} should not be calculated directly. Instead, the linear system is solved. Additionally, if K is symmetric and positive-semidefinite, computing the lower triangular matrix using the Cholesky decomposition

$$K = LL^T \quad (3.36)$$

makes the calculation quicker and yields the also needed determinant of 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 (3.34) there holds $A = K$ and

$b = y$, since the inverse of K is meant to be found. Defining $x = K^{-1}y$ there holds

$$K \times K^{-1} \times y = y \quad (3.37)$$

With L the lower triangular matrix of K obtained by

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

the linear system can now be solved for $x = K^{-1}y$ 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 [<http://www.math.lsa.umich.edu/hon>]. Therefore the lower triangular matrix L can be used to efficiently calculate the determinant of K . There holds

$$\det K = \det L \det L^T = \det L^2 \quad (3.38)$$

and therefore

$$\det K = \left(\prod_{i=1}^{n_y} L_{i,i} \right)^2 \quad (3.39)$$

Since in (3.34) the log likelihood is used, (3.38) can be simplified to a sum:

$$\begin{aligned} \log \det K &= \log \det L^2 \\ &= 2 \sum_{i=1}^{n_y} \log L_{i,i} \end{aligned}$$

Minimization of the Negative log-Likelihood

The best possible set of hyperparameters w for (3.34) 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 (3.34) is needed. It can be computed as follows.

3.4 Step 3: Estimating the Optimum Mesh Parameters

4 Application of statFEM in Vibroacoustics

4.1 Simple 1D example

Choice of PDE The Poisson equation

$$-\nabla \cdot \mu(x) \nabla u(x) = f(x) \quad (4.1)$$

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 [Cirak]

$$f(x) \sim \mathcal{GP}(\bar{f}(x), c_f(x, x')) \quad (4.2)$$

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

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

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 [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 (4.3) is evaluated at $x = x'$ with 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 $\mathcal{N}(\bar{f}, C_f)$ with \bar{f} the mean vector and 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 $C_f = LL^T$, 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 $f = \bar{f} + Le$ which is a multivariate Gaussian distribution with a mean \bar{f} and a covariance 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.

4.2 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.

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.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur 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 consectetur.

5 Conclusion and Discussion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur 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.

5.1 Section 1

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur 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 consectetur.

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] Campolina, B.: Vibroacoustic modelling of aircraft double-walls with structural links using Statistical Energy Analysis (SEA). Acoustics [physics.class-ph]. Université de Sherbrooke; Université Pierre et Marie Curie - Paris VI (2012)
- [2] Langer, S.C., Blech, C.: Cabin noise prediction using wave-resolving aircraft models. Proc. Appl. Math. Mech., Vol. 12 (1) (2019): e201900388. doi:10.1002/pamm.201900388
- [3] Yaghoubi, V., Marelli, S., Sudret, B., Abrahamsson, T.: Sparse polynomial chaos expansions of frequency response functions using stochastic frequency transformation, Probabilistic Engineering Mechanics, volume 48, pages 39-58 (2017)