DEPARTMENT OF INFORMATICS

TUM SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

Master's Thesis

Combining frequency decomposition and spectral mixture kernels with multi-fidelity Gaussian processes

Anastasiya Liatsetskaya

DEPARTMENT OF INFORMATICS

TUM SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

Master's Thesis

Combining frequency decomposition and spectral mixture kernels with multi-fidelity Gaussian processes

Author:	Anastasiya Liatsetskaya
Supervisor:	Prof. Dr. Felix Dietrich
Advisor:	M.Sc. Vladyslav Fediukov
Submission Date:	18.09.2024

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Munich, 18.09.2024   Anastasiya Liatsetskaya

Acknowledgments

I wish to thank my advisor, Vladyslav Fediukov, and my supervisor, Prof. Dr. Felix Dietrich, for supporting me throughout various projects during my Master's studies. It was a great experience to work on these projects.

Abstract

Training a machine learning model might require a large training data set. However, not in every setting is it possible to collect a large number of training samples. In a multi-fidelity setting, samples of a low-fidelity model, a model that might be less accurate in its predictions but is cheaper to evaluate, might be used to aid in learning a more precise model, the high-fidelity model. One considers samples in the signal domain. However, an alternative representation is a discrete spectrum of the samples. In this project, we aim to extract from the low-fidelity spectrum information, that can help to learn the spectrum of the high-fidelity function.
The spectral components of the high-fidelity function are learned with Gaussian processes [Ras03]. There are different kernels one can choose for the Gaussian processes. In the context of training with the spectra, we compared the performance of the spectral mixture kernels [Wil14] with the kernels composed from widely used components such as, for example, SquaredExponential kernels. Additionally, the BNSE initialization algorithm [Tob18] was used to further enhance the performance of the spectral mixture kernels.
The NARGP kernels [Per+17] introduce a structure in a kernel working with both the input information x and the low-fidelity samples. This structure was incorporated in the constructed model for learning in the frequency domain.

Contents

1 Introduction

One central theme of Machine learning is pattern discovery. Given observations from a complex real-world process, one might wish to build a mathematical model of this process. Such a model might help better understand how the process evolves and predict its outcome on the points where the data was not observed before. The data might contain hidden patterns which might be of interest to discover. The signal that generated the observed data might be a mixture of multiple sources and might be affected by noise.
With some signals one can explicitly see the trends and choose the components of a machine learning model, such that they represent the observed patterns in the data. In some settings, it is also possible to separate single sources and to model them separately. An example of such a setting is described in [AAS19]. The authors have trained for each isolated source a separate Gaussian process and then assumed that the observed signal is an additive combination of the sources. However, real-world data is often complicated with no obvious patterns. It might also be difficult to isolate the sources, to determine which sources comprise the signal and how the separate sources are combined. In such a scenario, interpretability of the model might be of significance. The parametric models, for example, Neural Networks [Bis06], trained on the observed data might be difficult to interpret.
Gaussian processes [Ras03] are a family of machine learning models. It contains models that are both powerful and interpretable. They describe a distribution over functions. Gaussian processes can incorporate the observed information about the function with the help of Bayes's theorem. The result will also be a Gaussian process. A defining property of real-valued Gaussian processes is its covariance function. A property, which according to [Mur12] is often encoded in the covariance function, is that for two similar points x_i and x_j one would expect similar function values $f(x_i)$ and $f_(x_j)$ at these points.
There exist many different kernel functions. Examples include the Square exponential, Rational Quadratic, and Matern [Gar23]. Through kernel functions, one can control which properties the sampled functions will have. For example, the Matern kernels with $\nu = 0.5$ can model functions which are continous, but differentiable nowhere. If one knows which types of behavior the separate sources comprising the signal have, then one might choose the appropriate kernels and model the signal.
The difficulty when constructing a kernel function is that a kernel matrix evaluated at a set of points X must be positive definite for every choice of X. This requirement might not be easy to prove. The Bochner's theorem [Boc+59] looks at the spectrum of the covariance function. It derives a requirement how a spectrum of a valid stationary covariance function should look like. Namely, a spectrum resulting from the Fourier

transform of a valid stationary function is proportional to a probability measure. This requirement in the frequency domain might be easier to satisfy compared to positive-definiteness in the signal domain. As a result, one idea could be to construct kernels directly in the frequency domain. The author in [Wil14] models the spectral density of a kernel function as a mixture of Gaussian distributions. This idea gives rise to Spectral mixture kernels. As the authors demonstrate the kernels are very flexible and are able to model many popular kernel families.

A signal can be considered in the signal domain. However, there exists an alternative representation in the frequency domain. The Fourier series allows to represent a signal as a weighted sum of periodic functions with different frequencies. One interesting idea could be to train a Gaussian process in frequency domain. A Gaussian process will extract information from the spectrum of the function's samples.

In some settings, it might be costly to build a large training data set for a model. A possible remedy might be the multi-fidelity setting. The settings have low-fidelity models and high-fidelity models. Low-fidelity models might be cheap to evaluate, but less accurate when predicting a ground-truth function. On the contrary, the high-fidelity functions might provide a very accurate description of the observed process, but are computationally expensive to evaluate. The idea of the multi-fidelity as described in [Rav+24] is to use samples from the low-fidelity as a part of the inputs for the high-fidelity model. The hope is that additional information provided by the low-fidelity samples might simplify the task of learning the ground truth function.

In the project, all three ideas of the spectral mixture kernels, training of Gaussian processes in the frequency domain, and multi-fidelity setting are explored. In the project, we defined a Gaussian process model that takes as inputs the domain points x and frequencies of the low-fidelity function, and then it tries to predict frequencies of the high-fidelity function. The Gaussian process was used with the spectral mixture kernels and custom kernels built from traditional kernels, for example, Squared Exponential. Finally, the NARGP introduces a structure for a kernel in the multi-fidelity setting. This kernel was used together with the derived model.

The section 2 starts with theory on Fourier series, Fourier transform and Discrete Fourier transform. These concepts play a crucial role in the project from being a part of the definition of the spectral mixture kernel to extracting spectrum from a discrete set of points from the signal. Then, the concept of a Gaussian process is explained. Finally, the spectral mixture kernels are described. Additionally, the BNSE initialization method for the parameters of the spectral mixture kernels introduced by [Tob18] is described. The method helps to select better initial parameters during the optimization procedure for the spectral mixture kernels.

The section 3 describes the Gaussian process models trained on the frequencies of the low-fidelity function. In the section, additionally, the model is combined with the

NARGP kernels.
Finally, the section 4 shows the performance of the model on academic examples.

2 State of the art

2.1 Fourier series

This section starts with a question of how one can represent a function via simple building blocks. A close analogy could be to look at vectors in Euclidean space. A vector can be uniquely represented as a weighted sum of basis vectors. In turn, this leads to the next question: whether this idea can be extended to function spaces, what kind of basic functions one should choose, and how many basis functions one needs.
A potential first step is considering a periodic function $f(t)$ with period T. When trying to find building blocks for the periodic functions, one choice could be to consider *sin* and *cos* functions. These functions are periodic and infinitely differentiable. Euler's formula allows shows the relationship between complex exponential function and the *sin* and *cos* functions as:

$$e^{ix} = \cos(x) + i\sin(x).$$

The sequence

$$\sum_{n=-\infty}^{\infty} c_n e^{2\pi i n \frac{t}{T}}$$

is the Fourier series expansion of the function $f(t)$. Its coefficients are defined as

$$c_n = \frac{1}{T}\int_0^T e^{-2\pi i n \frac{t}{T}} f(t)dt.$$

The coefficients result from a complex inner product between functions defined as

$$(f,t) = \int_0^T f(t)\overline{g(t)}dt.$$

According to [Osg] the functions $e_n(t) = \frac{1}{\sqrt{T}} e^{2\pi i n \frac{t}{T}}$ are orthonormal and build a basis of the function space $L^2([0,1])$, a space containing functions with finite square norm. For functions belonging to L^2 the Fourier coefficients c_n exist.
The Fourier series has infinite number of components. In practice, it would be intractable to compute an infinite series. This leads to an approximation of the series by considering only a finite number of components N:

$$S_N(t) = \sum_{n=-N}^{N} c_n e^{2\pi i n \frac{t}{T}}.$$

It should be noted, that not all functions have a Fourier series representation with an infinite number of non-zero coefficients c_n. One important observation is that the Fourier series is defined for periodic signals with period T. In practice, a function $f(t)$ might not be periodic. This raises the question of whether the ideas from the Fourier series can be extended to aperiodic functions. The question will lead to the Fourier transform of functions.

2.2 Fourier transform

Fourier transform allows the mapping of a function in the time domain to a function in the frequency domain. By performing a Fourier transform one can decompose a complicated input signal into separate frequency components. As noted in [Fre11], depending on application, some signal components can be interpreted as effects of physical phenomena that were present when the signal was measured. As an example, in EKG through Fourier transform one can filter out the noise from the signal by cutting off the specific frequencies corresponding to the noise. It requires knowing the typical frequency of the heartbeat and what frequencies might be the noise components. As a result, frequency representation of the signal might allow to incorporate prior knowledge about the signal and how it is generated on one side, on the other side it might help to derive a simpler representation of the signals in terms of its frequencies. One important observation is that some operators in time domain might be transformed in the frequency domain and vice versa. For example, the sum operator is preserved

$$\mathcal{F}(f+g)(s) = \mathcal{F}f(s) + \mathcal{F}g(s).$$

However, pointwise multiplication in frequency domain does not correspond to the pointwise multiplication in the time domain. Instead, based on the Convolution theorem, one obtains convolution in the time domain:

$$\mathcal{F}(f * g)(s) = \mathcal{F}f(s)\mathcal{F}g(s).$$

Since pointwise multiplication in the frequency domain allows the modification of individual frequencies, this approach can also be used for noise filtering by setting specific frequencies to zero.
An idea described in [Fre11] is to treat aperiodic signal $f(t)$ as a periodic one, but with infinite period $T \to \infty$. Then the coefficients of the Fourier series will have the form:

$$\hat{f}(s) = \int_{-\infty}^{\infty} e^{2\pi ist} f(t) dt.$$

Fourier series has a discrete spectrum. It might contain infinitely many elements, but it is still countable. On the contrary, the Fourier transform has a continuous spectrum.

A potential question that may arise is whether there exists an inverse transform. An inverse transform should take the amplitudes $\hat{f}(s)$ as an input and then from this function it should reconstruct the original signal $f(t)$. The idea is to assume for a moment that the function $f(t)$ is periodic with period T and thus has a Fourier series expansion. Then as shown in [Osg] one can derive a relation between the coefficients of the series c_n and the amplitudes of the Fourier transform:

$$c_n = \frac{1}{T}\hat{f}\left(\frac{n}{T}\right).$$

The next step is to look at the limit $T \to \infty$ and to observe that the series forms a Riemann sum which approximates the integral:

$$\sum_{n=-\infty}^{\infty} \frac{1}{T}\hat{f}(\frac{n}{T})e^{2\pi i \frac{n}{T} t} \approx \int_{-\infty}^{\infty} \hat{f}(s)e^{2\pi i s t}ds.$$

This results in the inverse Fourier transform:

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(s)e^{2\pi i s t}ds.$$

The Fourier transform is defined for continuous functions $f(t)$; however, in practice, often one obtains not the whole functions but rather a sequence of samples from the function. A possible question arises on how to extend the idea of the Fourier transform to discrete finite sequences. One observation from [Fre11] is that a sequence of samples does not uniquely define a function from which it was sampled.

2.3 Discrete Fourier Transform

A potential first question is how a Fourier series would look for a discrete-time finite signal. Suppose that the discrete signal contains N elements:

$$\mathbf{f} = (f[0], f[1], ..., f[N-1]).$$

The first important concept is that of a complex root of unity defined as

$$w^n = 1$$

for a complex number w. According to [Cor+22] for a fixed n there are exactly n complex roots of unity having the form $e^{2\pi i \frac{k}{n}}$. The expression

$$w_n = e^{2\pi \frac{i}{n}}$$

is the principal n-th root of unity. The other $n-1$ root s of unity are the powers of the principal root of unity. One important property denoted in [Cor+22] is the Halving lemma. It states that, if $n > 0$ and n is even then the squares of the n complex roots of unity result in $n/2$ complex roots of unity. This property plays an important role in an efficient implementation of the Discrete Fourier Transform.
Let the vector $\mathbf{w}^{-k}$ be defined as

$$\mathbf{w}^{-k} = (1, w^{-k}, w^{-2k}, ..., w^{-(N-1)k}).$$

Then the discrete Fourier transform is defined as

$$\mathcal{F}\mathbf{f} = \sum_{k=0}^{N-1} \mathbf{f}[k]\mathbf{w}^{-k}.$$

According to [Osg] the discrete Fourier transform will define two grids. One is in the signal domain, and one is in the frequency domain. The signal is assumed to be periodic with period N. An important difference to the continuous Fourier transform according to [Fre11] is that in continuous case there is an infinite number of harmonics, however in the discrete case the number is finite. The reason is the wrapping of frequencies, due to them being defined on a circle. This means that for a discrete signal consisting of N samples, its Fourier series will contain N frequencies.
According to [Fre11] one can represent a discrete Fourier transform as a matrix vector product $\mathcal{F}\mathbf{f}$. The matrix $\mathcal{F}$ has NxN entries and is defined as

$$\mathcal{F} = \begin{bmatrix} 1 & 1 & ... & 1 \\ 1 & w^{-1} & ... & w^{-(N-1)} \\ 1 & w^{-2} & ... & w^{-2(N-1)} \\ . & . & ... & . \\ 1 & w^{-(N-1)} & ... & w^{-(N-1)^2} \end{bmatrix}.$$

An inverse of the transform would require to know the inverse of the matrix $\mathcal{F}$. The entries of $\mathcal{F}$ are complex numbers. If a matrix A is complex, then A^* denotes its conjugate transpose. A square complex matrix is unitary if $A^*A = I$. According to [Osg] a complex matrix is unitary if and only if its columns form an orthonormal basis. The columns of $\mathcal{F}$ are the powers of vector $\mathbf{w}$. According to [Osg] for two integers k and l vectors $\mathbf{w}^l$ and $\mathbf{w}^k$ are orthogonal. For orthonormality, they must be normalized with $\frac{1}{\sqrt{N}}$. As a result, the inverse DFT is defined through the matrix $\mathcal{F}^{-1}$:

$$\mathcal{F}^{-1} = \frac{1}{N}\mathcal{F}^*.$$

Fast Fourier transform

A potential problem with the matrix formulation of the DFT is that its cost is $O(n^2)$ which might be intractable for discrete signals with many samples. As the author in [Fre11] describes, one idea could be to use the divide and conquer approach by trying to decompose a large problem into small ones and then to combine the solutions of the smaller problems into the solution of the starting problem.
One way to subdivide the problem is to look at even and the odd entries of the signal separately. For a 4 point signal its DFT is

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} W_4^0 & W_4^0 & W_4^0 & W_4^0 \\ W_4^0 & W_4^1 & W_4^2 & W_4^3 \\ W_4^0 & W_4^2 & W_4^0 & W_4^2 \\ W_4^0 & W_4^3 & W_4^2 & W_4^1 \end{bmatrix} \times \begin{bmatrix} x[0] \\ x[2] \\ x[4] \\ x[6] \end{bmatrix}.$$

This algorithm is known as the Fast Fourier Transform. It allows to compute the DFT in time $O(nlog(n))$.
If the signal is real-valued, then there is a relation between its positive and negative frequency coefficients. Since the signal is periodic the following relations hold for the coefficients a_k:

$$a_{N-k} = a_{-k} = a_k^*$$

with a_k^* denoting the complex conjugate of the a_k. This relation allows to only consider the positive frequency terms. This leads to a reduction of the frequency vector from the length n to the length $n/2 + 1$ for a signal with even number of entries.
If the frequency components of a discrete signal are known a potential task might be to obtain representations in the signal domain. One way to do that is to use real inverse Discrete Fourier Transform.

2.4 Gaussian Processes

The problem of regression is given a set of observed data points $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ to learn a mapping from $f : \mathbf{x} \rightarrow y$. y is assumed to be a continuous variable. This is a supervised learning problem. One might be interested in building a model which given the data $\mathcal{D}$ tries to learn the mapping f. One additional requirement on the model might be a measure of how confident the model is in its prediction $\hat{y}$ for an input point x.
A variety of different models exist for this task. Examples include linear regression, Neural Networks, support vector machines described in detail in [Bis06]. One prominent family of models are Gaussian processes.

One can express the uncertainty in the regression function models by defining a distribution over the potential candidate functions. One way of defining such distribution is by means of a stochastic process. A stochastic process defines a distribution over function values $\rho(y_1, .., y_N)$ for any finite set of points $x_1, ..., x_N$. The authors in [Wil14] provide an interpretation of a stochastic process as a random function. One special case of a stochastic process is when multivariate Gaussian distribution is used as the distribution of the joint function values:

$$(y_1, ..., y_N) \sim \mathcal{N}(\mathbf{y}|\mu, K)$$

with μ and K being mean vector and covariance matrix. This idea leads to the family of Gaussian process models.
A Gaussian process

$$p(f) = \mathcal{GP}(f; \mu, K)$$

is completely specified through its mean and covariance functions. According to [Gar23] the mean function $\mu(x)$ specifies the expected value over the function's value $f(x)$ at the location x:

$$\mu(x) = \mathbb{E}[f(x)|x].$$

The covariance function $K(x, x')$ is defined as:

$$K(x, x') = cov[f(x), f(x')|x, x'].$$

If a Gaussian process is specified, then for any finite set of locations X one can derive a distribution over the function values $\mathbf{y}$:

$$p(\mathbf{y}|X) = \mathcal{N}(\mathbf{y}; \mu(x), \Sigma)$$

with $\Sigma = K(X, X)$. The matrix $G = K(X, X)$ is called the Gram matrix of X. Each entry of the matrix is formed as $G_{ij} = K(x_i, x_j)$ with x_i, x_j being a pair of points from the set X. In G, the covariance is computed for each pair of points in X, with the pair not necessarily consisting of two distinct points.
Gaussian processes are defined though mean and covariance functions. Given the observed data $\mathcal{D}$ one question is how to choose these function in a way that a resulting Gaussian process model explains the observed data well. The prior knowledge about ground-truth function can be encoded through it. As the authors in [Wil14] note, one advantage of Gaussian processes against parametric models with weights $\mathbf{w}$ is that one might have a better prior knowledge about the function rather than about weights $\mathbf{w}$ of the model. This intuition about the function's characteristics can be encoded through a kernel function.
If some features of the functions are not precisely known in advance they can be

encoded as parameters of kernel functions. One example is the length scale parameter by the Squared Exponential kernel. As a result, one obtains a parametric family of kernel functions. If the family is parametric with the parameters θ and a criterion of how good the model approximates the ground-truth is the maximum likelihood of the observed data, then one obtains an optimization problem to be solved:

$$\max_{\theta} log(p(\mathcal{D}|\theta)).$$

The term $p(\mathcal{D}|\theta)$ describes how likely is the observed data under a model defined by the parameter θ. The objective function of the minimization problem might have several optima. Furthermore, there might be several models, which potentially might significantly differ from each other, having the same probability. As a result, the MLE criterion does not always specify the clear preference over the model space.
The authors in [Wil14] suggest using Bayesian model averaging:

$$p(f|\mathcal{D}) = \int p(f|\mathcal{D},\theta)p(\theta|\mathcal{D})d\theta.$$

As the authors in [Gar23] describe the model-marginal distribution is often intractable to compute exactly. One potential reason could be the integral computation. In [Gar23] several approximation strategies are described.
For the mean function the authors in [Bis06] suggest using the constant 0 function:

$$\mu(x) = 0,$$

if one has no prior knowledge about the mean function.
One goal of constructing a model learning the map f is to make a prediction at previously unseen points x. If the model's parameters are specified, one can use this model to obtain a distribution $p(f|x,\mathcal{D})$. This distribution encodes what the model believes will be the value of the map f at the location x, given the training samples the model knows.
An important component is the Bayes's theorem:

$$p(h|\mathcal{D}) = \frac{p(\mathcal{D}|h)p(h)}{\sum_{h' \in \mathcal{H}} p(\mathcal{D},h')}.$$

The theorem shows how a prior belief about a value $p(h)$ can be updated in light of the observed data $\mathcal{D}$. The resulting distribution is the posterior $p(h|\mathcal{D})$. The distribution $p(\mathcal{D}|h)$ is a likelihood of the observed data.
Suppose a set of inputs $\mathbf{x}_* = (x_{*1}, ..., x_{*M})$ is given and one is interested in the predictions $\mathbf{y}_* = (y(x_{*1}), ..., y_(*M))$ given the observed training data $\mathbf{x}$ and $\mathbf{y}$. In this scenario

one assumption could be that the observed function values come from a Gaussian process plus an additive Gaussian noise:

$$y(x) = f(x) + \epsilon(x),$$

$$\epsilon(x) \sim \mathcal{N}(0, \sigma^2).$$

The goal is to derive the distribution

$$p(\mathbf{y}_*|\mathbf{y}) = \int p(\mathbf{y}_*|f(x))p(f(x)|\mathbf{y}d(f(x)). \tag{1}$$

As shown in [Wil14] for this scenario one can compute the expression 1 analytically. The first step is to derive a joint distribution $p(\mathbf{y}, \mathbf{f}_*)$:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*)) \end{bmatrix}),$$

with $\mathbf{f}_*$ being evaluation of the function $f(x_{1*}), ..., f(x_{M*})$. A marginal distribution over the observed function values $\mathbf{y}$ can be obtained from the Gaussian process $p(\mathbf{y}) \sim \mathcal{N}(\mathbf{0}, K(X,X))$. Then one can use the Bayes's formula to derive the prediction $p(\mathbf{f}_*|\mathbf{y})$. As shown in [Wil14] this distribution is Gaussian with the following moments: the mean $\hat{\mu}$

$$\hat{\mu} = K(X_*, X)[K(X,X) + \sigma^2 I]^{-1}\mathbf{y}, \tag{2}$$

and the covariance $\hat{c}$

$$\hat{c} = K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma^2 I]^{-1} K(X, X_*). \tag{3}$$

The distribution over the observations $p(y_*|y)$ has the same mean as in 2. The variance is $\hat{c} + \sigma^2 I$. If the noise in observations is not Gaussian, then the integral in 1 might not be analytically computable. The authors in [Wil14] discuss several possible approximation techniques.

Kernel functions

A kernel function $K(x, x') = cov[y(x), y(x')|x, x']$ together with the mean function μ specify a Gaussian process. As the authors in [Gar23] note through a kernel function one can encode different properties of the functions sampled from the Gaussian process. For example, the sampled function should be continuous and differentiable. Further, a parametric family of kernels can be constructed, which allows one to choose the kernel that best explains the data observed. This leads to a question of how a kernel family can be constructed and which properties of functions can be incorporated in the kernel

functions. On one hand a parametric family should be broad enough, such that a wide range of possible function behaviours can be modeled with kernels from this family. On the other hand, it should not be expensive to evaluate such function and to perform optimization over the function family by looking for the parameter values fitting to the training data. Another question are the requirements to a potential kernel function to be a valid function for a Gaussian process.
According to [Bis06] often kernel functions K are designed to measure similarity. Namely, for two points x_i and x_j which are similar in the input space $\mathcal{X}$, the kernel function will show a high correlation of the output values $y(x_i)$ and $y(x_j)$. The precise definition of similarity between vectors is dependent on the application. An example of such kernel is a member of the exponential covariance kernel, a member of the Matern family, defined as:

$$K_{M_{\frac{1}{2}}}(x, x') = exp(-d)$$

with the Euclidean distance $d = |x - x'|$.
According to [Gar23] two properties are necessary for a kernel function to be admissible. The first property is that a kernel function should be symmetric:

$$K(x, x') = K(x', x).$$

The second requirement is that for any finite set of points X the Gram matrix $K(X, X)$ should be positive semidefinite. One implication of this requirement, that the variance is non negative: $K(x, x) = var[y(x)|x] \geq 0$. Further if the Gram matrix is positive definite, than according to Mercer's theorem [Mur12] one can perform eigenvector decomposition on the Gram matrix G:

$$G = U^T \Lambda U.$$

By defining $\phi(x_i) = \Lambda^{\frac{1}{2}} U_{:i}$ one can observe that the ij-th component of the Gram matrix can be formulated as

$$g_{ij} = \phi(x_i)^T \phi(x_j).$$

The function $\phi : \mathcal{X} \to R^D$ with R^D being potentially infinite dimensional space can be interpreted as extracting features from the input vector x. Then the kernel allows to compute inner product of potentially infinite dimensional feature vectors. This allows to avoid the explicit definition of the feature vectors $\phi(x)$. Kernels which are positive definite are called Mercer kernels. An example of Mercer kernels are kernels from the Matern family and the RBF kernels.
An important result described in [Gar23] is that one can treat valid kernel functions as building blocks. By modifying and combining these blocks new valid kernel functions can be obtained. For example, scaling introduces a valid kernel function:

$$K'(x, x'; \lambda) = \lambda^2 K(x, x').$$

If a function $g : \mathcal{X} \rightarrow \mathcal{Z}$ is used to perform a transformation of the domain $\mathcal{X}$ and $K_{\mathcal{Z}}$ is a valid kernel in the domain $\mathcal{Z}$, then one obtains a valid kernel $K_{\mathcal{X}}$ in the domain $\mathcal{X}$ defined by:

$$K_{\mathcal{X}}(x, x') = K_{\mathcal{Z}}(g(x), g(x')).$$

Finally, as described in [WR06] one can additively combine kernel functions and obtain valid kernel functions. If two random processes $f_1(x)$ and $f_2(x)$ are independent, then a sum $f_{add}(x) = f_1(x) + f_2(x)$ will have a kernel:

$$k_{\text{add}}(x, x') = k_1(x, x') + k_2(x, x').$$

In [WR06] the authors describe further combinations including multiplication and convolution. These results allow to model complex behaviour out of simple blocks.
The question on how to choose a suitable parametric family of kernels provided the observed data is a topic of research. One can rely on prior knowledge to choose a family expressing the believed properties of a random process. The prior knowledge can greatly accelerate the search for a model with the best explanation for the training data. However, in praxis not always a lot is known in advance about the underlying process which generated the observed data. The authors in [Gar23] describe multiple possible strategies on how to select a space of candidate kernel functions. One idea is to define a space of kernels by using a context-free grammar defining what combinations of kernels lead to kernels still belonging to the space. Then one can perform a search over this space. Two ideas described in [Gar23] is to perform a greedy search or alternatively to perform Bayesian optimization over the space of kernel functions. A potential difficulty is how to rapidly perform a search over potentially infinite space of candidates.
One important property of kernel functions is described in [Mur12]. That is a value of kernel function corresponds to a scalar product of two vectors:

$$k(x, x') = \phi(x)\phi(x')^T.$$

The vectors $\phi(x)$ might be infinite dimensional and they correspond to the extraction of features of the input data x. A similar idea could be fined in SVM and Linear regression models of the form

$$y(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}),$$

where $\psi(\mathbf{x})$ represent fixed basis functions. An important difference from is that for kernels one can consider infinite number of basis functions.

Examples of kernel functions

One family of covariance functions is the Matern family. Matern family is a parametric family of kernels having the form [Mur12]:

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

with the distance $r = \|x - x'\|$, parameters $\nu > 0$ and $l > 0$ and a modified Bessel function K_{ν}.This is a family of stationary kernels. Stationarity means that the kernel is invariant to translation since:

$$k(x, x') = k(|x - x'|)$$

with the kernel depending only on the distance between the data points and not on their exact location. According to [Gar23] this family can model Gaussian processes with the sample paths of any desired degree of smoothness. The parameter ν controls the degree. Namely, the sampled paths from a centered Gaussian process with a Matern kernel are $\lceil \nu \rceil - 1$ times continuously differentiable.
The limit case of the parameter $\nu \to \infty$ is the squared exponential kernel:

$$k(x, x') = exp(-\frac{1}{2}(x - x')^T \Sigma^{-1} (x - x')).$$

This is a family of stationary kernels. The squared exponential kernel has the property that the functions drawn from a Gaussian process are infinitely differentiable.
The parameter $\nu = 0.5$ leads to the exponential covariance defined as

$$k_{M^{0.5}}(x, x') = exp(-d)$$

with $d = |x - x'|$. The sampled paths are continuous but not differentiable. Once and twice differentiable functions are modelled by $\nu = 1.5$ and $\nu = 2.5$ with the covariance being defined as

$$k_{M^{1.5}} = (1 + \sqrt{3}d) exp(-\sqrt{3d}),$$
$$k_{M^{2.5}} = (1 + \sqrt{5}d + \frac{5}{3}d^2) exp(-\sqrt{5}d).$$

2.5 Spectral Mixture kernels

One expressive family of kernels is the Spectral Mixture kernels proposed in [Wil14]. The idea of the Spectral Mixture Kernels uses two results as its foundation. The first result is the Bochner's Theorem [Boc+59]:

Theorem (Bochner) 1 *A continuous function $K : \mathbb{R}^n \to \mathbb{R}$ is positive semidefinite (that is, represents a stationary covariance function) if and only it holds*

$$K(\tau) = \int_{R^n} exp^{2\pi i s^T \tau} \psi d$$

where ψ is a finite, positive Borel measure on $\mathbb{R}^n$.

Furthermore, as shown in [Wil14], a density $S(s)$ corresponding to the measure ψ, which is also called the power spectrum of the kernel k, and the covariance function are connected through the Fourier transform:

$$k(\tau) = \int S(s) e^{2\pi i s^T \tau} ds,$$

$$S(s) = \int k(\tau) e^{-2^T \tau} d\tau.$$

As the authors in [PT17] note, constructing a positive definite kernel function is often difficult. Bochner's theorem shows that one can construct a stationary kernel in the frequency domain instead of the signal domain. In the frequency domain the requirements on the function $S(\omega)$ are easier to satisfy according to [PT17]. The reason is that in the spectral domain, one needs to ensure that the function $S(s)$ is a valid probability density function. In the signal domain, the requirement is positive-definiteness of the kernel function. As a result, the idea of the Spectral Mixture kernels is to construct probability distributions in the frequency domain and then to transform them back to the signal domain via Fourier transform, obtaining a valid stationary kernel. This leads to two potential questions. The first question is how to construct a potentially complex probability distribution. The second question is how to obtain a tractable Fourier transform of the probability distribution.
The second key idea of the Spectral Mixture Kernels is to use a mixture of Gaussian distributions to construct a probability distribution for the kernel. The single Gaussians are used as building blocks. Then one constructs a weighted sum of the blocks as:

$$k(\epsilon) = \sum_i w_i \mathcal{N}(\epsilon; \mu_i, \Sigma_i)$$

with w_i being the weights of the components. One can achieve the symmetry by applying

$$\mathbf{k}(\epsilon) = \frac{1}{2}[k(\epsilon) + k(-\epsilon)].$$

According to [Gar23] there exists an analytical representation of the Fourier transform of the Gaussian mixture defined as:

$$K_{SM}(x, x'; \{w_i\}, \{\mu_i\}, \{\Sigma_i\}) = \sum_i w_i exp(-2\pi^2 (x - x')^T \Sigma_i (x - x')) cos(2\pi (x - x')^T \mu_i).$$

The sets $\{w_i\}$, $\{\mu_i\}$ and $\{\Sigma_i\}$ are learnable parameters for the model. One can perform training in the signal domain and optimize these parameters with the help of the observed data from the ground truth process. One requirement to the mixture weights w_i as noted in [Gar23] is that the weights should be positive.
One potential question could be what family of distributions can be approximated with a Gaussian mixture and how flexible this family is. According to [Mur12] any probability density function defined on $\mathbb{R}^D$ can be approximated by a Gaussian mixture provided that the number of mixture components is large enough. According to [Gün22] one might need a very large number of mixture components to represent 2-dimensional densities. The dimensionality poses an additional challenge here since the author notes that the number of needed components might grow exponentially with the dimensionality of the data. A large number of components can not only be computationally expensive but can also lead to the problem of overfitting. Overfitting refers to the problem of having a hypothesis h which perfectly describes the observed data, but which does not generalize to the previously unseen data. Models having a big number of trainable parameters might lead to overfitting, because the model might be too flexible. Since each mixture component has its own parameters, one might obtain an overfitted model if the number of the mixture components is big. There exist models capable of modeling complex distributions and suitable for density evaluation. One example is Normalising flows based on the change of variables formula [Gün22]. However, a significant problem is the tractability of the Fourier transform back to the signal domain. Computing a Fourier transform involves the evaluation of an integral. For a general density function this integral might not be analytically computable. As a result, the use of Normalising flows might be a topic of future work in the context of kernels constructed in the frequency domain.

Initialisation

The kernels often contain parameters that can be optimized. Often, optimization is performed with a gradient descent method. This is an iterative method that, starting from some location in the space, tries to locate a local maximum of a function. The gradient of a function is often used as a way to decide in which direction to move in the next step. One example of such an optimization scheme is Adam. An optimization might require many steps to achieve a low-loss function value. This might be problematic if it is costly to evaluate or to approximate the gradient of a loss function.
The initialization of the hyperparameters defines the starting values of the hyperparameters that are further refined by the following optimization procedure. Often, one uses random initialization of the parameters. However, a choice of parameters closer to

the desired minimum might accelerate the optimization procedure.

BNSE

When initializing parameters of the spectral components for the spectral mixture kernels one idea could be to try to estimate frequencies from the observed data. Then the estimated frequencies can be used as a starting point for the optimization procedure. This leads to the question of how frequency information can be extracted from the data points. One way to do it is to use the Discrete Fourier transform. As the authors in [Tob18] describe, real-world data might be noisy, it might be unevenly sampled and some data points might be missing. As a result, due to the convolution in the Fourier transform this leads to uncertainty over the entire frequency domain. This might lead to a requirement of expressing uncertainty in the obtained frequencies.
An approach to address this requirement is proposed in [Tob18]. The idea is to approximate the unknown signal via a Gaussian process. Then a Fourier transform is performed on the Gaussian process allowing to obtain a Gaussian process over frequencies. If a stationary GP $f(t) \sim GP(m, K)$ is chosen as a prior over the ground-truth signal, then according to [Tob18] the power spectral density of the signal can be derived from the Fourier transform of the kernel:

$$S(\epsilon) = \mathcal{F}\{K(t)\}(\epsilon).$$

One difficulty with the stationary processes as noted by the authors is that the draws from a stationary GP are almost surely not Lebesgue integrable which is required for the Fourier transform to exist. One idea could be to look at the spectrum of the signal in the neighborhood of a center c, the local spectrum. The local spectrum is defined as:

$$F_c(\epsilon) = \mathcal{F}\{f(t-c)e^{-\alpha t^2}\}$$

with the width of the window being defined as $\frac{1}{\sqrt{2\alpha}}$. The authors in [Tob18] provide an analytical form of the covariance function of the local spectrum $F_c(\epsilon)$for the stationary signal $f(t) \sim GP(0, K(t))$:

$$K_F(\epsilon, \epsilon') = \sqrt{\frac{\pi}{2\alpha}} e^{-\frac{\pi^2}{2\alpha}(\epsilon-\epsilon')^2} (\mathcal{K}(\rho) * \sqrt{\frac{2\pi}{\alpha}} e^{-\frac{2\pi^2}{\alpha}\rho^2})$$

with $\mathcal{K}(\epsilon)$ being the Fourier transform of the kernel K and $\rho = \frac{\epsilon+\epsilon'}{2}$. The authors in [Tob18] provide an analytical form for the choice of kernel K as the spectral mixture kernel with Q components $K_{SM}(\tau) = \sum_{q=1}^{Q} \sigma_q^2 exp(-\gamma_q \tau^2) cos(2\pi\theta_q^T \tau)$.
The Gaussian process over frequencies is a complex-valued Gaussian process. To

determine the process one needs the covariance function $K(\epsilon, \epsilon')$ and the pseudo-covariance $P(\epsilon, \epsilon')$. If the signal $f(t)$ is real-valued, then according to [Tob18] there exists a relation

$$P(\epsilon, \epsilon') = K(\epsilon, -\epsilon')$$

allowing to specify pseudo- covariance through the covariance function.
Given the observed data $\mathbf{y}$ one can then refine the prior on frequencies. The authors in [Tob18] derive the following equalities for the mean and covariance of the posterior distribution $p(F_c(\epsilon)|\mathbf{y})$:

$$\mathbb{E}[F_c(\epsilon)||] = K_{F_c}^T(t, \epsilon)K(t,t)^{-1},$$

$$\mathbb{E}[F_c^*(\epsilon)F_c(\epsilon')|\mathbf{y}] = K_F(\epsilon, \epsilon') - K_{\mathbf{y}F_c}^T(t, \epsilon)K(t,t)^{-1}K_{\mathbf{y}F_c}(t, \epsilon)$$

with $K_{\mathbf{y}F_c}(t, \epsilon) = \mathcal{K}(\epsilon)e^{-j2\pi\epsilon t} * \frac{\pi}{\alpha}e^{-\frac{\pi^2\epsilon^2}{\alpha}}$ where $\mathcal{K}(\epsilon)$ is the result of applying the Fourier transform to the kernel K.
From this result, the authors also show that the posterior of the power spectral density of a stationary process $f(t) \sim GP(0, K)$ which considers the observed data, is a χ^2 - distributed stochastic process possessing an analytical representation of its mean.
The following graphs 3 demonstrate the loss function when training on the air passengers data set. The training was performed with the Adam method. 2000 iterations were done in both cases.

On the graphs, one can observe that the initialization with BNSE has led to a much smaller initial loss of 3251 compared to $1 \cdot 10^{16}$ in the random initialization case. As a result, BNSE initialization might result in a smaller loss and a better model compared to the random initialization of parameters.

2.6 Multi-fidelity Models

A traditional approach in machine learning is to first define a family of potential candidate models that might explain the observed data well. Such family is often a parametric family where the models differ in parameter values. An example of this approach are Neural Networks [Bis06]. This brings to a question on how to select out of potentially uncountably many candidate models a model which we think might be the best model approximating the unknown function. A classical way to select a model is to perform optimization over the model space. The optimization requires a criterion to optimize. The criterion often tries to capture how well the model performs. One way to measure performance is to use the observed data for which the correct answer is already known. For example, in an image classification task, one can compare what the model predicted for the observed images against the available labels. This approach

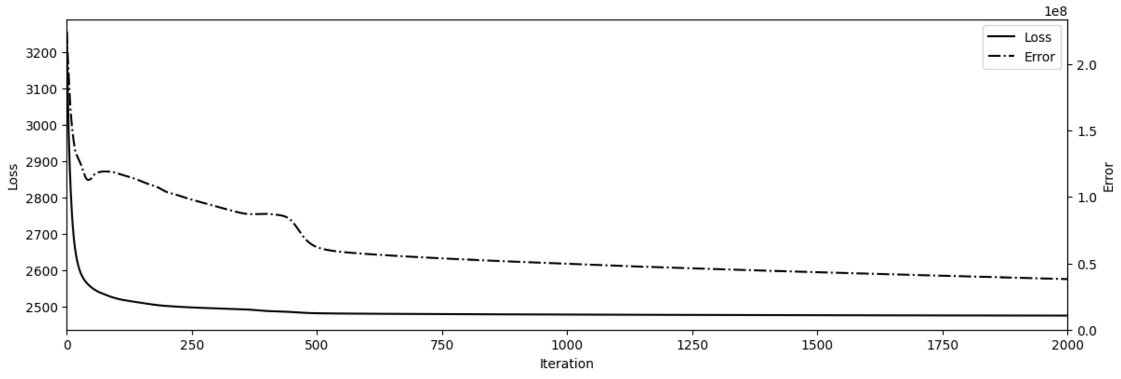


Figure 1: Initialisation with BNSE

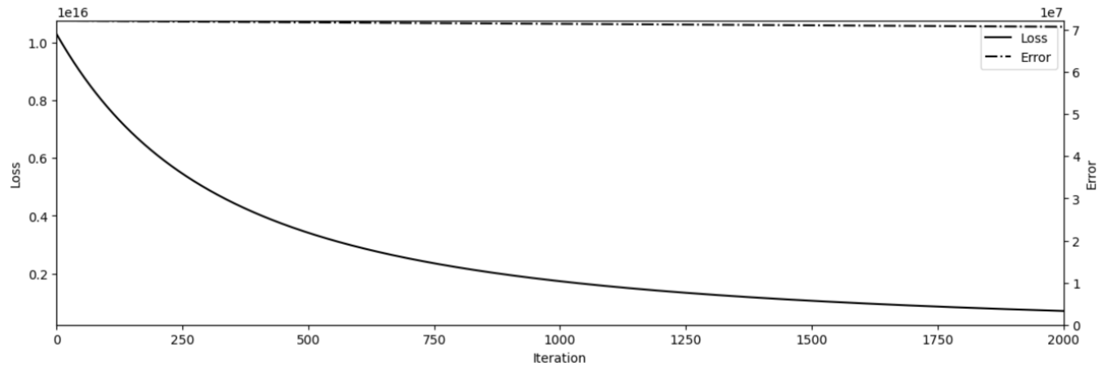


Figure 2: Random initialisation

Figure 3: Loss-function for random initialization and the BNSE initialization. The figure was generated with help of the Mogptk library [Tob].

is known as the empirical risk minimization [SB14]. The empirical risk minimization relies on the training data set.
The size of the data set might pose a tradeoff. On the one hand, the points in the data set provide information about the unknown function and probability distribution. An increased amount of data points might guide towards a better model. On the other hand, in some settings, gathering large data sets might be very expensive. For example, if images are labeled by hand, then labeling of thousands of images requires a lot of work. An additional consideration is the dimensionality of the input space. Intuitively by increasing dimensionality of the input space one might need significantly more points to cover interesting regions of the space. Two potential methods arise from the tradeoff. The first approach is to select the data points which are the most informative.

An example of this approach with the Bayesian optimisation used to select the points was considered in the project [Lia23]. The second approach is to transform the domain. The idea has similarities with the Support Vector Machines approach [Bis06]. Suppose that in the original domain X the function $f(x)$ has a very complicated structure and consequently would require many points from different domain regions to capture its structure. Suppose there is a transformation function $g(x)$ which transforms the input domain X to the feature space X'. Despite the function $f : X \to Y$ being complicated, the function $f' : X' \to Y$ might be significantly simpler. As an example, in Support Vector Machines one can try to fit a linear function f' to the transformed features. If the function $f'(x')$ has a simple form, then the hope is that it will not require many data points to learn its structure.
One difficulty with the described approach is how to select the transformation g. According to [Rav+24] one idea is to look at multi-fidelity modeling. For the same problem different one might obtain levels of approximation accuracy by using different approaches or different amount of available data. For example, when approximating an integral of a function one can construct a mesh on which the data is evaluated. Taking 2 points per dimension or 200 might result in significant differences in the approximation quality depending on the integrated function. Following [Rav+24] in the paper we distinguish low-fidelity models, which might provide a less accurate approximation of the problem but are cheap to evaluate, and high-fidelity models, models providing fine-grained approximations of the target function f, but which are potentially costly to evaluate. Then, a question arises of how multi-fidelity settings can be combined with Gaussian process models.

NARGP

One model introduced in [Rav+24] addresses this question. A first model is the linear Autoregressive model [KO00] defined as

$$u_t = \rho \cdot u_{t-1} + u_\sigma.$$

This model takes as input a realization of the previous fidelity level u_{t-1}, multiplies it with a scaling constant ρ and adds a correction term u_σ. The samples u_{t-1} and u_σ are coming from Gaussian processes. According to [Per+17], the scaling constant can be interpreted as a term modeling correlation between the low-fidelity input u_{t-1} and the high-fidelity input u_t. This parameter together with the kernel hyperparameters of u_σ and u_{t-1} are learnable. u_σ is assumed to be independent from u_{t-1}. As a result, one can chain different levels of fidelity in a Neural Network-like structure. The training then involves learning simultaneously hyperparameters of the whole structure. One important observation described in [Per+17] this model has a Markov property:

$$cov(f_t(x), f_{t-1}(x')|f_{t-1}(x)) = 0, \forall x \neq x'.$$

This property means that given a low- fidelity information about the points x we cannot derive any further information useful for learning $f_t(x)$ from the other low-fidelity evaluations at points $x' \neq x$.
The authors in [Per+17] describe a modification of the algorithm allowing more efficient inference. The algorithm assumes the nested structure of the training data sets $\mathcal{D}_1 \subseteq \mathcal{D}_2 \subseteq \mathcal{D}_s$. The prior GP f_{t-1} is replaced by a posterior Gaussian process f^*_{t-1}. This decouples the model resulting in s regression problems.
One way of generalising the Linear Autoregressive model described in [Per+17] is by using a non-linear transformation function z_{t-1}:

$$f_t(x) = z_{t-1}(f_{t-1}(x)) + \sigma_t(x).$$

As the authors note, if a non-parametric GP prior is used to model z_{t-1}, then one obtains a deep Gaussian process. One potential problem with this construction is that the posterior is no longer a Gaussian process. One solution to this problem presented in [DL13] is to use variational inference and find an approximation to the posterior. The approximate posterior is assumed to be Gaussian and is found by maximizing the Evidence Lower Bound.
The authors note that this type of generalization of the AR leads to a significant increase in the computational cost. As a result, an alternative construction is proposed in [DL13]. A prior from the previous fidelity f_{t-1} is again replaced with a posterior f^*_{t-1}. Allowing to train Gaussian processes one by one with an increasing fidelity level. The Gaussian process of level t receives as input as described in [Rav+24] the predictions made by the already trained Gaussian process model of the previous fidelity level. During the predictions, samples are drawn from the lower-fidelity GP and are also used as a part of the input for the higher-fidelity model. The Gaussian process of level t only receives the information from the previous fidelity- level $t-1$.
One additional idea proposed by the authors in [DL13] is to introduce more structure to the kernel of the Gaussian processes. The proposed kernel has the form

$$k_{t_g} = k_{t_p}(x, x'; \theta_{t_p}) \cdot k_{t_f}(f^*_{t-1}(x), f^*_{t-1}(x'); \theta_{t_f}) + k_{t_\sigma}(x, x'; \theta_{t_\sigma}).$$

The first observation is that this kernel separates the inputs x from the low- fidelity observations $f^*_{t-1}(x)$. The authors claim that putting together into the same kernel x and $f^*_{t-1}(x)$ might not be natural since they belong to different spaces.

3 Training in the frequency domain

In the state-of-the-art section, four concepts were discussed. The first is the signal and its alternative representation in terms of frequencies. The second is Gaussian processes. These are flexible models capable of learning distributions over functions. The kernels of Gaussian processes can capture different kinds of correlation between function's outputs. The third concept was a specific kind of the kernel function, the spectral mixture kernels. The power of the spectral mixture kernels lies in its flexibility. Finally, the multi-fidelity setting is described.
The following section is dedicated to bringing all four concepts together. For a multi-fidelity setting where we have the input x and additionally the low-fidelity function samples we aim to learn the high-fidelity function. We extract the spectrum from the low-fidelity samples and use it as an input for Gaussian processes which predict the spectrum of the high-fidelity function. Gaussian processes can be used with different types of kernels. In this project we analyze the spectral mixture kernels compared to a custom kernel. Additionally, we apply the BNSE initialization to the spectral mixture kernel with the hope that it will lead to a better choice of the kernel parameters.

3.1 Motivation

Many Machine learning models are defined and trained in the signal domain. One task might be reconstructing the signal $y = \phi(x)$ from the input signal x. The idea of learning in the frequency domain is to look at the representation of the signal in terms of its frequencies. If ψ is a function that maps a spectrum of one signal in the spectrum of the other signal, then according to [Pol+22], the functions $\psi(x)$ and $\phi(x)$ are related through:

$$\phi(x) = \mathcal{T}^{-1} \circ \psi \circ \mathcal{T}$$

where $\mathcal{T}$ represents the transformation of the input signal into its spectrum. This relation also assumes that the inverse transformation T^{-1} from frequency to the signal domain exists. As noted in [Pol+22] frequency domain models might be able to learn long-range dependencies in signals and they might result in architectures containing a tractable number of layers in the case of Neural Networks.
Some motivational examples in this field include [Pol+22] and [Li+20] where the authors propose Neural Network models that work with the frequencies of the discrete input signal.
In [Pol+22] the discrete signal is transformed into the frequency domain only once, before the first layer of the Neural Network. The Neural Network is defined and trained in the frequency domain. The authors pose the training procedure of the model

parameters θ as a constrained nonlinear optimization problem of the form

$$\begin{array}{ll} \text{minimize}_\theta & \mathbb{E}_{x,y}[||\mathcal{T}(y) - \hat{Y}] \\ \text{subject to} & \hat{Y} = f_\theta \circ \mathcal{T}(x) \\ x \sim p(x) & \\ y = \phi(x) & \end{array}$$

There are multiple transformations $\mathcal{T}$ of the input signal that can be used, for example, DFT, Discrete Cosine Transform, DCT-II. The authors note that a real-valued transformation might be beneficial since it might allow the use of already existing architectures for separate layers of the Neural Network. As a result, the authors propose to use the Discrete Cosine Transform.
A Fourier Neural Operator model presented in [Li+20] uses a slightly different approach. For each layer a transformation of the input data into frequency domain is performed. Then, a potentially complex-valued function is applied to the data. Finally, the obtained data is transformed back to the time domain. According to [Pol+22] the flow can be described as

$$\begin{array}{ll} X = \mathcal{T}(x) & \text{Forward Transform} \\ \hat{X} = f_\theta(X) & \text{Learned Map} \\ \hat{x} = \mathcal{T}^{-1}(\hat{X}) & \text{Inverse Transform} \\ \hat{y} = \hat{x} + g(x) & \text{Residual.} \end{array}$$

The residual can be additionally added to reintroduce frequencies, which may be lost during the transformation f_theta. The transformation $\mathcal{T}$ used in the proposed model is the Discrete Fourier Transform.

3.2 Definition of the model

In this project Gaussian processes were used to learn the function $\psi(x)$. A first idea could be to follow a similar approach to that described in the previous section. Starting from a sequence of samples from the input signal, a low-fidelity samples $(f_l(x_1), ..., f_l(x_N))$, we aim to learn the corresponding sequence of high-fidelity samples $(f_h(x_1), ..., f_h(x_N))$. The figure 4 shows the first architecture of the model. The model takes a sequence of low-fidelity samples as input. The sequence is transformed with the real DFT to its frequency representation. Then the discrete spectrum is used as an input to the Gaussian processes. A discrete spectrum of the high-fidelity function $(\phi_1, ..., \phi_K)$, which Gaussian processes are trying to predict, is a vector. One approach could be to treat each dimension of this vector independently. Each of the processes GP_i tries to reconstruct its respective part of the high-fidelity spectrum ϕ_i. An alternative idea

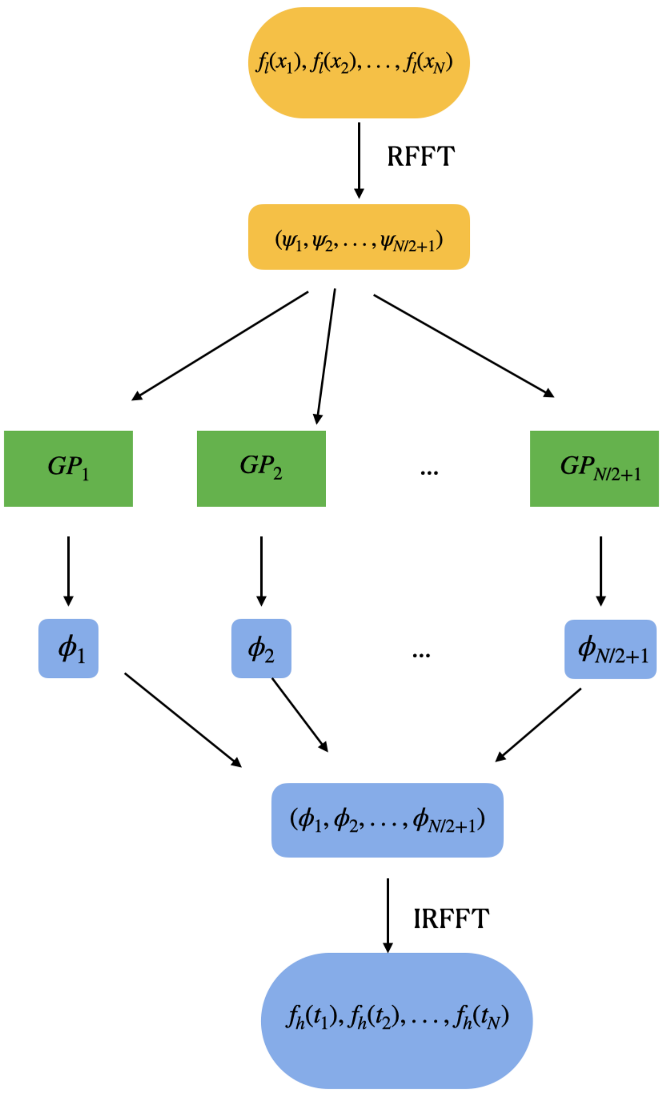

$f_l(x_1), f_l(x_2), \ldots, f_l(x_N)$
RFFT
$(\psi_1, \psi_2, \ldots, \psi_{N/2+1})$
GP_1
GP_2
...
$GP_{N/2+1}$
ϕ_1
ϕ_2
...
$\phi_{N/2+1}$
$(\phi_1, \phi_2, \ldots, \phi_{N/2+1})$
IRFFT
$f_h(t_1), f_h(t_2), \ldots, f_h(t_N)$

Figure 4: First architecture for learning in frequency domain. Starting from the low-fidelity discrete spectrum the model tries to learn the high-fidelity spectrum

could be to model correlations between the parts of the discrete spectrum.
One additional observation is that a real DFT will result in complex-valued frequency components. An approach used in the project is to treat real and imaginary parts separately. That is for a Fourier coefficient ϕ_i two Gaussian processes are used. One predicts the real part $\phi_i.real$ and one predicts the imaginary part $\phi_i.imag$. After both real and imaginary parts of the spectrum are obtained, they are again assembled

into a complex-valued spectrum, which then undergoes the inverse real Fast Fourier transform to obtain the signal samples in the time domain. This approach allows the use of Gaussian processes, which are specified for real-valued input and output vectors. As a result, for a vector of samples $(f_h(x_1), ..., f_h(x_N))$ of the dimensionality N where N is assumed to be even, the model will have $2 \cdot (N/2 + 1)$ Gaussian processes. This might be very computationally expensive to train. However, one observation is that these models can be trained independently and in parallel. This allows to train them on distributed memory systems, allowing to utilize more hardware resources. An alternative approach is to use complex-valued Gaussian processes. One example of such an approach can be found in [Tob18]. As the authors note, the model requires one additional component to be specified together with the usual covariance function. This component is the pseudocovariance function.
The model in 4 has one problem when it is used only with frequencies of the low-fidelity function as input to predict high-fidelity samples as output. The problem can be demonstrated by the low- and high-fidelity functions depicted in 5. The low-fidelity

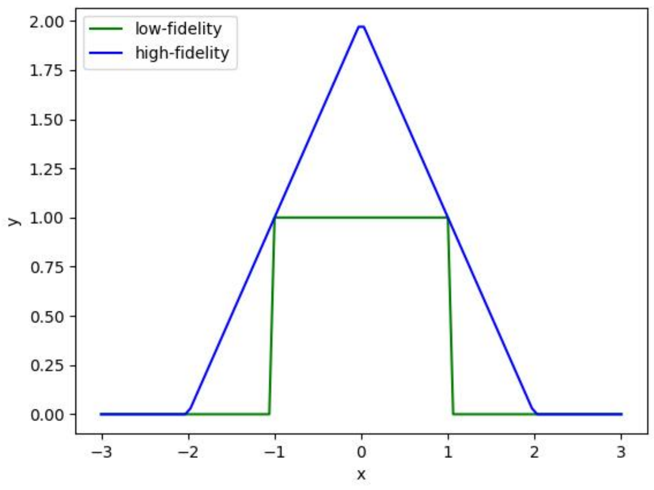


Figure 5: Low- and high-fidelity functions

samples taken from the interval $[-1, 1]$ all have the same value of 1. On the contrary, the high-fidelity function does have different types of behavior in the interval. From the low-fidelity values alone, it is impossible to differentiate which values the high-fidelity function should have. This is no longer a functional relation between the low- and high-fidelity samples. As a result, the model that uses only low-fidelity inputs is restricted to functions that cannot have to two identical input sequences two different output

sequences. An example of such functions could be strictly monotonically increasing functions for low and high-fidelity.
Since in practice this family of functions might be too restrictive, one idea could be to use additional information in the input. One example could be to use together with the frequencies the input points x on which the low-fidelity function was evaluated. Then the model from 4 will have the following structure:
The model in 6 involves transforming the predicted frequencies back into signal domain. The transformation is done via the Inverse real Fourier transform. According to the section 2 this transformation is a multiplication with the matrix. A key question is how the small errors in the frequency vector will behave after undergoing the transformation back to the frequency domain. Is it possible that small errors in the frequency vector might lead to large deviations in the obtained signal vector. This question is of high importance since the frequencies are predicted by a Gaussian process which can produce errors on previously unseen evaluation points. If the result is too sensitive to the errors, then no matter how good the Gaussian process is, the final model is of little use. According to [RSA15] the Parseval's theorem provides an insight into the sensitivity question. Let x and $\hat{x}$ be a discrete signal and its approximation. $\mathcal{F}(x)$ denotes frequencies of the signal x extracted by the Discrete Fourier transform. The theorem states:

$$|x - \hat{x}|_2^2 = |\mathcal{F}(x) - \mathcal{F}(\hat{x})|_2^2.$$

The authors in [RSA15] mention that there exists an equivalent results for the IDFT operator. This means that minimizing the l_2 loss between two frequency vectors will result in signal vectors having minimal l_2 loss in the signal domain.
The model in 6 is suitable for multi-fidelity setting as it involves both the input x_i and the low-fidelity information ψ_i. In the section with the numerical experiments it will be empirically tested whether the frequencies of the low-fidelity signal help to predict the frequencies of the high-fidelity signal. One way to do it is to use a model that does not use the discrete spectrum of the low-fidelity and only relies on the points x_i. Such a model will then have the following structure as illustrated in 7.

Transferring probability distribution

One important aspect of Gaussian processes is that during inference for a vector of arguments $\mathbf{x}$ it gives a multivariate Gaussian distribution back. A distribution expresses uncertainty of the model regarding its predictions. As a result, one potential requirement on the trained model might be to transform a probability distribution over frequencies $\rho(\phi)$, resulting from a Gaussian process, into a probability distribution over the high-fidelity values $\rho(f_h)$.

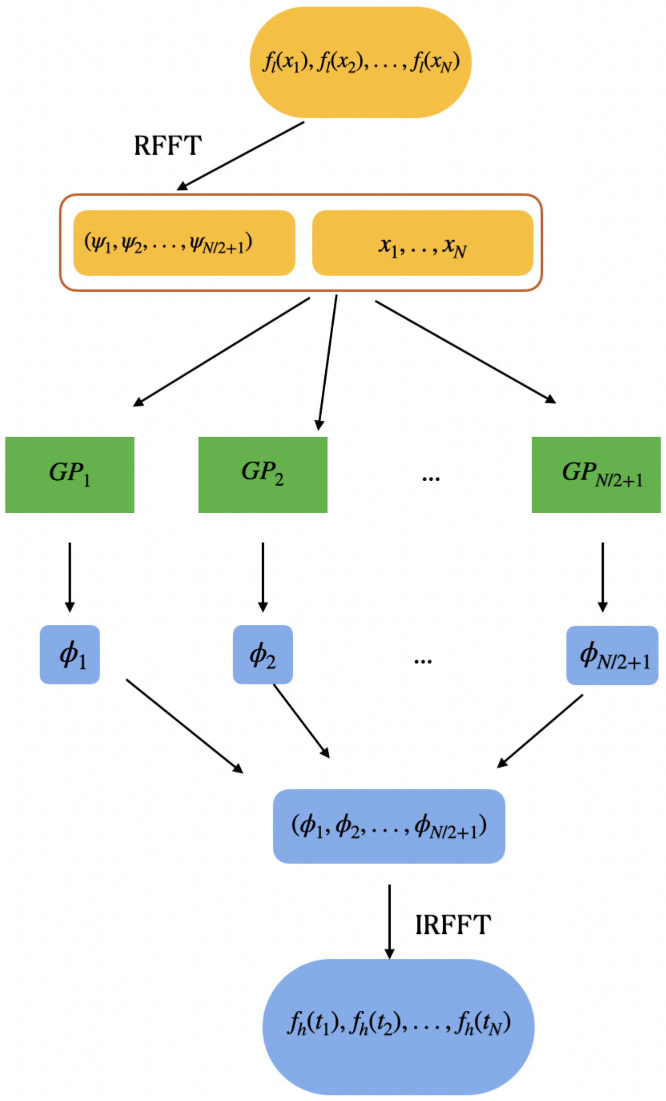

$f_l(x_1), f_l(x_2), \dots, f_l(x_N)$
RFFT
$(\psi_1, \psi_2, \dots, \psi_{N/2+1})$
$x_1, \dots, x_N$
GP_1
GP_2
...
$GP_{N/2+1}$
ϕ_1
ϕ_2
...
$\phi_{N/2+1}$
$(\phi_1, \phi_2, \dots, \phi_{N/2+1})$
IRFFT
$f_h(t_1), f_h(t_2), \dots, f_h(t_N)$

Figure 6: Learning in frequency domain with both x and low-fidelity information. The low-fidelity information is the discrete spectrum of the low-fidelity function. The Gaussian processes predict the discrete spectrum of the high-fidelity function.

The first observation is the transformation IRFFT. This is a matrix-vector $f_h(\mathbf{x}) = \mathcal{M} \cdot \psi_h(\mathbf{x})$ product where both the matrix and the vector are complex-valued. However, if the signal f_h is real-valued, then the IRFFT of its frequencies must lead to real-valued

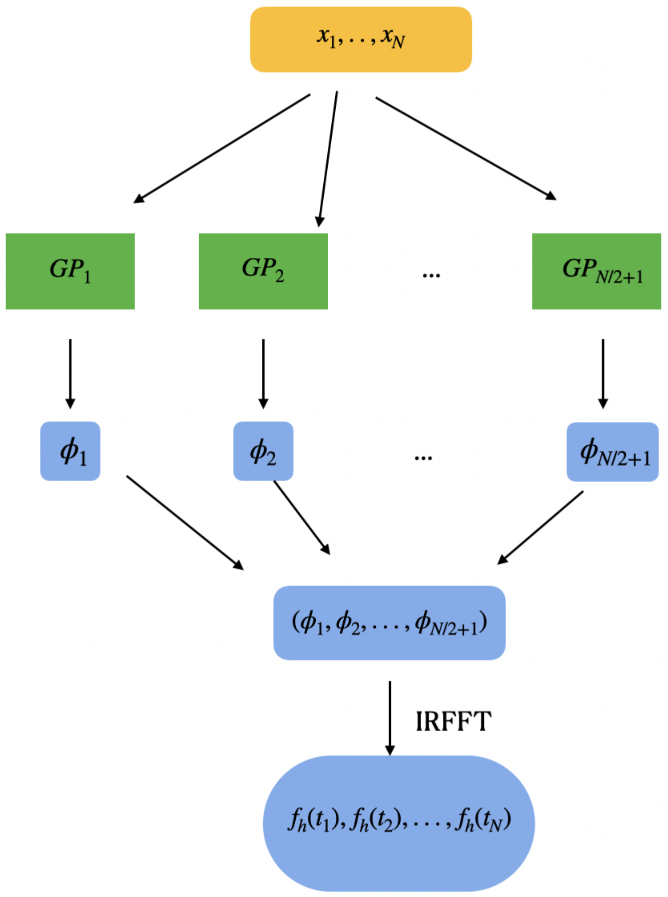


Figure 7: Model without low-fidelity information

outputs. The matrix has a form

$$\mathcal{M} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & w^1 & \dots & w^{(N-1)} \\ 1 & w^2 & \dots & w^{2(N-1)} \\ . & . & \dots & . \\ 1 & w^{(N-1)} & \dots & w^{(N-1)^2} \end{bmatrix},$$

the vector has a form $\psi_h(\mathbf{x}) = \mathbf{re}(\psi_h) + i \cdot \mathbf{im}(\psi_h)$. During the inference, each entry of the vectors $\mathbf{re}(\psi_h)$ and $\mathbf{im}(\psi_h)$ are Normally distributed random variables. This leads to the question of whether from the distributions of the vector entries **re** and **im** one can derive a closed form expression for the distribution of the function values $f_h(\mathbf{x})$. A potential first step is to look at the Fourier coefficients and the complex roots of unity involved in deriving the function values. The following formula shows one row of the matrix $\mathcal{M}$ multiplied with the vector ψ_h resulting in a high-fidelity sample at the point x_n:

$$f_h[x_n] = \frac{1}{N}\sum_{k=0}^{N-1} F[k]w^{kn}$$

where $w^{kn} = e^{\frac{2\pi ik}{N}}$. For a real-valued signal f_h there is one important property according to [Osg]. Suppose that the number of samples is even. Amplitudes which have the same distance i to the center $\frac{N}{2}$ are complex conjugate of each other:

$$F[\frac{N}{2}+i] = \overline{F[\frac{N}{2}-i]}.$$

A similar property holds for the frequencies w^{kn}:

$$w^{k(\frac{N}{2}+d)} = w^{kd}\cdot w^{\frac{kN}{2}} = w^{kd}(-1)^k,$$

$$w^{k(\frac{N}{2}-d)} = w^{-kd}\cdot w^{\frac{kN}{2}} = w^{-kd}(-1)^k,$$

$$w^{-kd} = e^{i(\frac{-2\pi kd}{N})} = \cos(-\frac{2\pi kd}{N}) + i\cdot\sin(-\frac{2\pi kd}{N}) = \cos(\frac{2\pi kd}{N}) - i\cdot\sin(\frac{2\pi kd}{N}) = \overline{w^{kd}}.$$

As a result, for the sum components of $f[n]$ holds:

$$F[\frac{N}{2}+d]\cdot w^{n\frac{N}{2}+d} = \overline{F[\frac{N}{2}-d]}\cdot\overline{w^{n\frac{N}{2}-d}}.$$

As a result, one obtains a sum with $\frac{N}{2}$ real-valued summands:

$$f[x_n] = \frac{1}{N}\sum_{k=0}^{N-1} F[k]w^{kn} = \frac{1}{N}\sum_{k=1}^{\frac{N}{2}-1} 2P(F[k]w^{kn}) + \frac{1}{N}Re(F[0]) + \frac{1}{N}Re(F[\frac{N}{2}])\cdot\cos(\pi n),$$

with $P(F[k]w^{kn})$ being defined as:

$$P(F[k]w^{kn}) = Re(F[k])\cdot\cos(kn\cdot\frac{2\pi}{N}) - \sin(kn\cdot\frac{2\cdot\pi}{N})\cdot Im(F[k]).$$

As a result, the distribution of $f[x_n]$ is an affine combination of Normally distributed random variables $Re(F[k])$ and $Im(F[k])$. Furthermore, if one assumes that random variables $X \sim \mathcal{N}(\mu_x, \sigma_x)$ and $Y \sim \mathcal{N}(\mu_y, \sigma_y)$ are independent, then a linear combination of these distributions results again in a Normal distribution: $X + Y \sim \mathcal{N}(\mu_x + \mu_y, \sigma_x^2 + \sigma_y^2)$. The distributions $Re(F[k])$ and $Im(F[k])$ are modeling predictions of Gaussian processes for the frequency k. More specifically, one distribution models the real part of the frequency, and another models the imaginary part. It is assumed that separate frequencies in the discrete spectrum are uncorrelated with each other. One further assumption is that real and imaginary components are also independent. By using the formula for the sums of independent Normal distributions, one obtains the following formulas for mean and variance:

$$\mathbb{E}[f[x_n]] = \frac{1}{N}\mu_0 + \frac{1}{N}\cos(n\pi)\cdot\mu_{\frac{N}{2}} + \frac{2}{N}\sum_{k=1}^{\frac{N}{2}-1}\cos(kn\frac{2\pi}{N})\mu_k - sin(kn\frac{2\pi}{N})\hat{\mu}_k, \tag{4}$$

where μ_i is the mean of $Re[F_i]$ and $\hat{\mu}_i$ is the mean of $Im[F_i]$. The variance has the form:

$$Var[f[x_n]] = \frac{1}{N^2}\sigma_0^2 + \frac{1}{N^2}cos(n\pi)^2\sigma_{\frac{N}{2}}^2 + \frac{4}{N^2}\sum_{k=1}^{\frac{N}{2}-1} cos(kn\frac{2\pi}{N})^2\sigma_k^2 + sin(kn\frac{2\pi}{N})^2\hat{\sigma_k}^2. \tag{5}$$

Initialisation

F. Tobar in [Tob] mentions one potential problem regarding the spectral mixture kernels. Namely, that it might be difficult to optimize their parameters. One strategy that might help to overcome this challenge to some extent is to make a good first guess about which parameter values might be a good fit.
One possible choice of kernels for Gaussian processes in 6 for predicting frequencies of the high-fidelity model are spectral mixture kernels. The model in 6 uses discrete spectrum as an input for Gaussian processes. Spectral mixture kernels also look at spectra. However, there is an important difference. The model in 6 extracts the spectrum from the input data, whereas spectral mixture kernels optimize spectral components of the kernel itself. In general case these are two different spectra.
If spectral mixture kernels are used for the Gaussian processes in 6, then one can try to combine the Spectral mixture kernels with the BNSE initialization. The spectral mixture kernels use a mixture of Gaussians to model a complex distribution. The parameters to optimize are the means, the variance, and the mixture weights of the separate components in the mixture. The hope is that a better initialization of the parameters might lead to a better model when the number of the iteration steps during the optimization procedure is fixed. The authors in [Tob] suggest to use the BNSE method to approximate the empirical spectral density of the data defined as

$$\hat{S}(s_m) = \frac{|\tilde{y}(s_m)|^2}{N}, m = 0,..,N-1$$

where $\tilde{y}(s_m)$ denotes the m-th element of the Discrete Fourier Transform of the function evaluations y with N being the number of collected samples. If there are Q mixture components, then the initialization method will select the locations of the Q frequency

peaks with the largest magnitude and use them as means for the mixture components. The lengthscale is derived as the width of the peaks multiplied by 2. Finally, the mixture of Gaussians is a weighted sum, where each of the distributions receives its own weight. The authors propose to take the normalized magnitude of the peaks as the weights. The normalization is done so that the sum of weights is equal to the variance of the output.
In this approach, the spectral density of the kernel is constructed with the help of the spectral information from the data. One possible question that might arise is how the two are connected with each other. According to the author in [Wil14] the first step is to look at stationary kernels and regular grid of evaluation points. This setting gives rise to the Toeplitz structure of the kernel matrix. Every diagonal from left to right in the kernel matrix is constant. For example,

$$\mathcal{K} = \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & a_1 & a_2 & a_3 \\ c_1 & b_1 & a_1 & a_2 \\ d_1 & c_1 & b_1 & a_1 \end{bmatrix}.$$

From the covariance matrix one can construct a circulant matrix C. A key observation in [Wil14] is that a circulant matrix in time domain when transformed via a Fourier transform becomes a diagonal matrix:

$$Cz = DFT^{-1}[diag(\tilde{c})]DFT[z].$$

This observation leads to a discrete form of the Bochner's theorem:

$$c_t = \frac{1}{N}\sum_m e^{\frac{2t}{N}}\tilde{c}_m.$$

If a Gaussian process has a circulant covariance matrix, then the log marginal likelihood according to [Wil14] has the form:

$$logp(y|X) = -\frac{1}{2N}\sum_m \frac{|\tilde{y_m}|^2}{\tilde{c_m}} - 0.5\sum_m log(\tilde{c}_m) - 0.5 \cdot N \cdot log(2\pi).$$

According to the authors the circulant covariance kernel in order to maximize the log-likelihood has the following form in the frequency space:

$$\hat{c}(s_m) = \frac{|\tilde{y}(s_m)|^2}{N},$$

which is exactly the empirical spectral density.

Size of the subsequences

In order to extract frequencies, the DFT needs a sequence of the signal samples. From a vector of samples, a vector of amplitudes is derived. The size of the input sequence is a hyperparameter in the model.

One observation is that a sequence of samples can be collected locally, describing a small region of the ground truth function, or globally, going over the whole domain. A local construction might require significantly fewer samples to describe the behavior of the ground truth function in the local region. The idea of approximating locally the ground truth function is similar to splines in the interpolation. However, it only provides the local information. This might also lead to differences in the frequency domain. Local frequencies might be different from the global frequencies. However, this does not necessarily mean that the information about the global patterns will completely vanish from the local spectrum. This information might be contained in the Fourier coefficients. As a result, learning locally over multiple subintervals might still be sufficient to capture the global behavior. The DFT will then fit a whole Fourier series into this local part of the function, assuming that it is one period of the global function. This assumption might be violated depending on the size of the subinterval and the ground-truth function.

On the contrary collecting samples over the whole domain gives information over the global trends. The disadvantage is that for functions with complex behavior, a large number of samples might be required to accurately model different regions of the function in the global sample. Also, the number of samples in the subsequence has an effect on the output of the model in 6. The number of Gaussian processes the model uses depends on the number of frequencies of the high-fidelity function one needs to predict. In turn, the number of frequencies is $\frac{N}{2}+1$ for the number of samples N. As a result, in the project the local sampling was used to minimize the runtime needed to train and evaluate models.

From the frequency point of view, the Nyquist sampling theorem [Jer77] provides an insight of what a sampling rate should be to avoid aliasing. The theorem relates the sampling frequency f_s to the highest frequency of interest in the signal f_N to be preserved. The statement is

$$f_s > 2 \cdot f_N.$$

The theorem provides an additional criterion of how many samples should be in a subsequence.

Another observation comes from the components of the Fourier series itself. The components are defined over the whole interval for which the Fourier series is computed. If a Gaussian process makes a significant error in its prediction of the component's weight, then this error will have an effect on every prediction of the signal's value

$(f_h(x_1), ..., f_h(x_N))$ in the signal domain.
Finally, a set of subsequences is collected to train the model. A limitation of the number of samples comes from the Gaussian process itself. During the inference, the process will build a NxN covariance matrix with N being the number of the training samples. Too many samples will lead to intractable computation time.

Structure of the model's output

The next question is whether the model in 6 specifies a Gaussian or more generally a stochastic process for its predictions of the high-fidelity values $f_h(x)$.
The model starts with an input vector $(\mathbf{x}, \psi)$ where $\mathbf{x}$ is a subsequence of the input locations x_i and ψ is a discrete spectrum of the low-fidelity function over the subsequence. The input vector is then used by Gaussian processes to make a prediction of the frequency components r_i or im_i. Each r_i or im_i is a Gaussian random variable and is assumed to be independent from the predictions of the other Gaussian processes in the model. The next step the model makes is the IRFFT transformation of the predicted spectrum to obtain the signal values of the high-fidelity functions. The formulas 4 and 5 show that a prediction $f[x_n]$ at the location x_n is a normally distributed random variable. During the computation of $f[x_n]$ the predictions of the Gaussian processes r_i and im_i are linearly combined. An example of the coefficients of the linear combination are $\cos(ki \cdot \frac{2\pi}{N})$ and $\sin(ki \cdot \frac{2\pi}{N})$. According to [Gar23] a linear combination of Gaussian processes is also a Gaussian process. As a result, for an input vector $(\mathbf{x}, \psi)$ the prediction $f[x_n]$ for a location x_n will be modelled by a Gaussian process. An important observation is that for two indices m, p with $m \neq p$ the Gaussian processes modelling $f[x_m]$ and $f[x_p]$ are different. This can be observed when looking at the coefficients of the linear combination $\cos(ki \cdot \frac{2\pi}{N})$ and $\sin(ki \cdot \frac{2\pi}{N})$. The coefficients depend on the index i of $f[x_i]$.
Suppose one fixes a point $\hat{x}$ in the signal domain. This point can be part of different subsequences around the point. Suppose the two possible subsequences $s_1 = (x_1, ..., \hat{x})$ and $s_2 = (\hat{x}, ..., x_N)$ are put as inputs in the model 6. Then there might occur a situation when the same point $\hat{x}$ will get two different distributions over the high-fidelity values depending on which subsequence was used as the input. The Gaussian processes predict a discrete spectrum of a subinterval. If the spectra of the two subsequences s_1 and s_2 are predicted without an error then converting the spectra back to the signal domain will result in the same value for the point $\hat{x}$. However, errors in predicted frequencies might lead to a situation when the reconstruction of the point is no longer the same. This is illustrated with the following example: the model from 59 receives as

inputs two subsequences

$$s_1 = (0.100063, 0.100688, 0.101313, 0.101939, 0.102564, 0.103189, 0.103815, 0.104440),$$

$$s_2 = (0.104440, 0.105066, 0.105691, 0.106316, 0.106942, 0.107567, 0.108193, 0.108818).$$

The last element of the s_1 is the same as the first element of the s_2. The model outputs 0.81 and 0.53 as the locations of the mean for that element. Since a Gaussian distribution is defined through mean and variance, different means lead to different Gaussian distributions for the same $x = 0.104440$. Since a point in the signal domain x might not have a unique probability distribution assigned to it, this is not a stochastic process over the domain x.

3.3 NARGP and training in frequency domain

The NARGP extracts information from both the inputs X and the low fidelity model f_l. It constructs a Gaussian process modeling high-fidelity function by considering three components in the kernel:

$$k(x, x') = k_\rho(x, x'; \theta_\rho) k_f(f_l(x), f_l(x'); \theta_f) + k_\sigma(x, x'; \theta_\sigma).$$

This leads to the question of how one can integrate spectral mixture kernels and Gaussian processes trained in the frequency domain into NARGP.
In the NARGP model introduced in [DL13], Squared Exponential kernels were used for all three components k_ρ, k_f and k_σ. If one considers more complex kernels, which are more expressive but at the same time are more costly to train, this brings to a question of how to combine these kernels with the NARGP model. A first potential idea is to use complex kernels only for the kernel working with the low-fidelity function k_f. The hope is to extract from latent features of f_l enough information that a problem of modeling high-fidelity would significantly simplify and it would be enough to consider linear transformations. Also, simpler kernels for k_ρ and k_σ might be considered, which are less costly to train and evaluate. However, as the authors in [DL13] note if a low-fidelity function is strongly correlated with a high-fidelity function, then its samples can provide very valuable information when reconstructing the high-fidelity function. But it can also be that the low-fidelity function is only weakly correlated on specific parts of the domain with the high-fidelity function. Then, the low-fidelity samples might be misleading for the model and might result in overall bad performance. As a result, in the project all three components of the kernel k_ρ, k_f and k_σ have received equally powerful kernels. This increases the computational cost to train the overall model, but it also allows the extraction of complex features from both the input x and

the low-fidelity samples $f_l(x)$.
If spectral mixture kernels are used in the model, this brings a question of how to combine them with the BNSE initialization. One idea could be to initialize each component of the NARGP kernel separately with BNSE as if it would be the only kernel predicting the output of the model. An advantage of the method is the ease of the implementation. A disadvantage is that it does not directly fit parameters for the whole NARGP kernel but only for its part. As a result, the obtained initialization of the parameters might be suboptimal for the whole kernel.
Another idea might be to try to use the BNSE with the whole NARGP kernel. An important observation is that a spectral mixture kernel with Q components defines a certain structure in its spectral density. Namely, it is a mixture of Q Gaussian distributions. The parameters of the kernel correspond to the parameters of the mixture. If a spectral mixture kernel undergoes a transformation and combination with other kernels, then the resulting spectral density might no longer be a mixture of exactly Q Gaussian distributions. The BNSE initialization looks for spectral peaks and assumes that the spectral density corresponds to a mixture of Gaussians with a fixed number of mixture components. As a result, it might be necessary to select a different number than Q of Gaussian distributions in the BNSE. If the resulting spectral density has a more complex form which is difficult to represent as an affine combination of the Gaussians from the component kernels of the NARGP, then one might need to create a mapping between the parameters found by the BNSE and the parameters of the components.
For the model describing training in frequency domain 6 a potential first question would be how to combine it with the NARGP approach. In the first approach demonstrated on the figure 8 a Gaussian process lives completely in the frequency domain. It is trained to predict the frequencies of the high-fidelity signal. During the inference such process would predict separate frequency components which then can be transformed further via IRFFT to the signal components. IRFFT itself is not part of the Gaussian process and the process knows nothing about it. In this scenario, NARGP is formed in the frequency domain. The kernel components k_ρ and k_σ receive the space points x_i as inputs.
A second approach demonstrated in 9 is to train the model in the signal domain. However, this approach would require transforming Gaussian process modeling frequencies into a Gaussian process in the signal domain. The Gaussian process with the NARGP kernel will then predict the high-fidelity samples in the signal domain. IRFFT is a multiplication of the input by a fixed complex-valued matrix $\mathcal{M}$. As a result, this procedure would result in modification of the kernel function of the Gaussian process. When modifying a kernel- function, one needs to ensure that the resulting kernel matrices are always positive-definite.

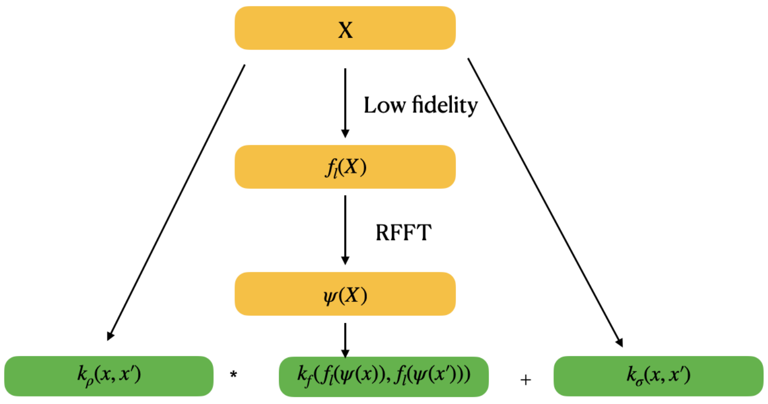


Figure 8: NARGP in frequency domain

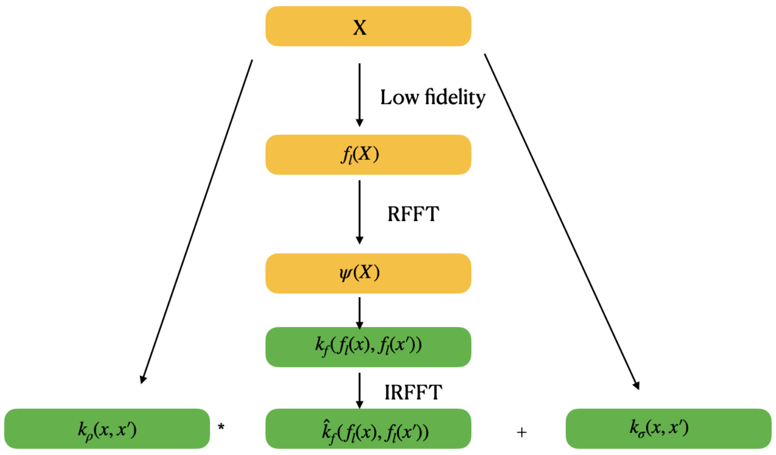


Figure 9: NARGP in signal domain

Figure 10: Two ways to apply the NARGP kernel. In the first, the component k_f of the NARGP working with the low-fidelity inputs will obtain the discrete spectrum of the low-fidelity samples. Overall model will predict the discrete spectrum of the high-fidelity function. In the second way, the NARGP kernel is completely defined in the signal domain. The kernel component $\hat{k}_f$ is a result of transformation, namely multiplication with the IRFFT matrix. This kernel takes frequencies of the low-fidelity spectrum as inputs and directly makes prediction in the signal domain without a need of an explicit IRFFT transformation. This kernel function should be positive-definite

4 Numerical Experiments

The model defined in the previous section was implemented with the help of the Mogptk library [1]. Several tests for multiple low and high-fidelity pairs were performed. The goal was to estimate the model's performance 6 on the test functions, to detect weak and strong sides of the model, and to identify possible improvements. In the experiments different types of kernels are used with the model. That includes the spectral mixture kernel. Also, the BNSE initialization, together with the spectral mixture kernel, is considered. The code of the performed experiments can be found in [2].

4.1 Spectral mixture kernel in signal domain

The first experiment was dedicated to the question of how expressive the spectral Mixture kernels are. In order to perform the experiment a function was chosen

$$f(x) = \sin(\pi x) + 2\sin(2\pi x) + \sin(3\pi x) + \sin(7\pi x) + 2\sin(8\pi x) + \sin(9\pi x).$$

The figure 11 shows the plot of the function in the interval $[0, 3]$. One can observe that the function has both low- and high-frequency components. Also, the function is periodic. The goal of the experiment was to observe whether a Gaussian process with the spectral mixture kernel can learn the function f, capturing both low and high frequencies. Additionally, the runtime required to train the model was measured.
The experiment setting was starting from a value x the Gaussian process must predict the corresponding function value $f(x)$. For the experiment 2400 training pairs (x_i, y_i) were collected over the interval $[0, 3]$. The points x_i build a regular grid over the interval. The kernel has 15 mixture components. Additionally, the BNSE initialization was applied to make an initial choice of the model's parameters prior to the optimization procedure. The optimization algorithm is Adam, which took 500 steps.
During the testing phase the model was evaluated on 3200 samples gathered equidistantly on the interval $[0, 3]$. The figure 12 shows the model's predictions together with the ground-truth values for the collected samples. One can observe that the trained model did capture the structure of the ground- truth function almost everywhere except for a small approximately linear region from $[0.8, 1.3]$. In this region, the model makes an error with the high-frequency components. A possible improvement might be to take more mixture components Q in the spectral mixture kernel. This might allow to capture smaller frequencies of the ground-truth function. The maximum of the error $|f(x) - \hat{y}(x)|$ between the values of the ground-truth sample $f(x)$ and its prediction $\hat{y}(x)$ is 0.23 for the collected test samples.

[1]https://github.com/GAMES-UChile/mogptk

[2]https://github.com/NastassiaLiatsetskaya/Learning_in_frequency_domain

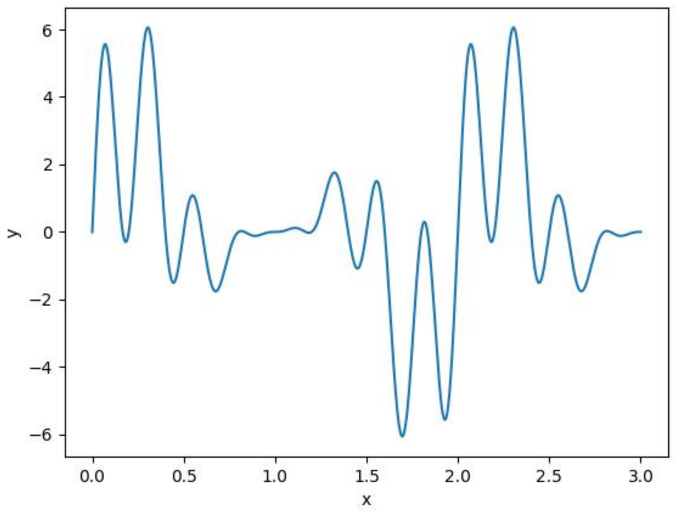


Figure 11: Ground-truth function

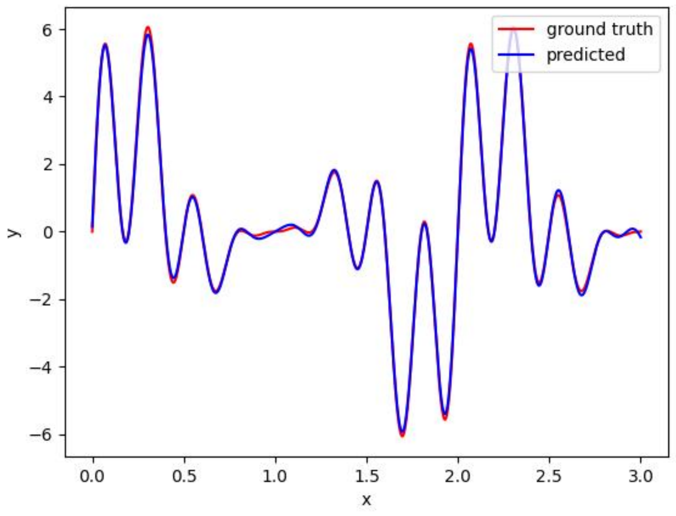


Figure 12: Ground-truth function and the model's prediction

Figure 13: Experiment: learning with spectral mixture kernels a function with both low and high frequencies.

As a result, for a function having both low- and high-frequency components a spectral mixture kernel was able to learn the structure of the function except for a small region. This leads to a hope, that spectral mixture kernels are good candidates to be combined with the model 6 allowing learn complex relations between low-fidelity and high-fidelity functions.
One additional observation is that it required 2715 s seconds to train the model on 2400 samples. In the project, multi-fidelity setting is considered. It is assumed that it is computationally intractable to gather many samples from the high-fidelity function. In the following experiments for the case of complex relation between high-fidelity and low-fidelity functions, each Gaussian process model will receive a smaller set of 800 high-fidelity samples during the training procedure. The idea is to extract information from the low-fidelity samples, hopefully simplifying the learning problem. The performance of the spectral mixture kernels will be compared to the performance of the kernels combined from the widely used component kernels, for example, SquaredExponetial and RationalQuadratic.
In practice, there is often a tradeoff between the accuracy of the model and the time needed to train it. In the following experiments, together with the accuracy of the model, its training time will be measured. Additionally, the performance and runtime of the spectral mixture kernel with and without the BNSE initialization will be compared. The goal is to observe whether a spectral mixture without BNSE can produce results that do not significantly deviate from the version with the BNSE, but at the same time, the runtime improves significantly.

4.2 Learning in frequency domain. Simple relation between low and high-fidelity

The second experiment was dedicated to see whether a Gaussian process model trained in the frequency domain can learn a simple relation between frequencies of low-fidelity and high-fidelity functions. A linear transformation between functions $f_h(t) = a \cdot f_l(t) + b$ is a simple relation chosen for the experiments. The advantage of the linear transformation is that a linear transformation of the low-fidelity function in signal space corresponds to the linear transformation of the frequencies of the low-fidelity function in the frequency space.
The low-fidelity function used in the experiment has the form

$$y_l(t) = \sin(t) + \cos(t^2) + max(30, t).$$

The high-fidelity function

$$y_h(t) = 10 \cdot y_l(t) + 5.$$

The figure 14 shows samples of the high-fidelity function. One can observe three

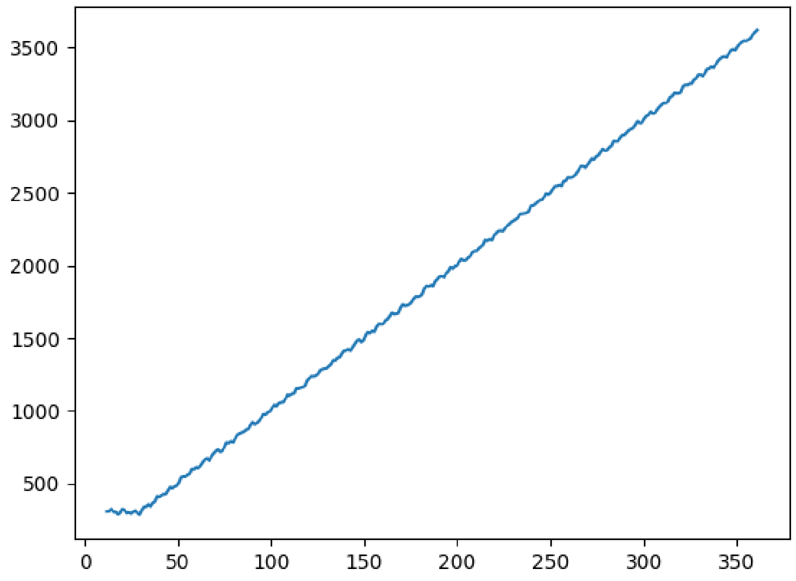


Figure 14: High-fidelity function in signal domain

components in the behavior of the function. The first component is the linear rising trend starting at 30. The second component is the constant behavior until the input value 30. Both of these components are caused by the RELU function. The third component is the oscillations resulting from sin and cos components of the function.

Figures 15 and 16 show the relation between frequency components of low-fidelity and high-fidelity functions. One can observe that for both real and imaginary components the relations are linear. As a result, the model trained in the frequency domain should learn a linear relation between the part of the input, the low-fidelity frequencies, and the output.
In the experiments performed, two kinds of models were tested. The first was using the Spectral mixture kernels in every Gaussian process. The second was using a linear combination of the Exponential, Linear and Matern kernels. The Matern kernel had the parameter $\nu = 0.5$. The goal of using two models was to compare the performance of the model 6 using the Spectral mixture kernel against the model with the combination of the Exponential, Linear and Matern kernels.

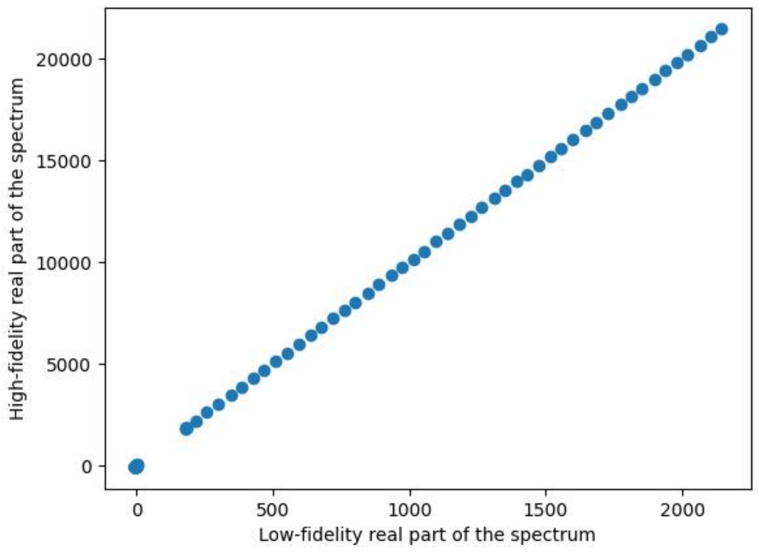


Figure 15: Relation between real components of frequencies

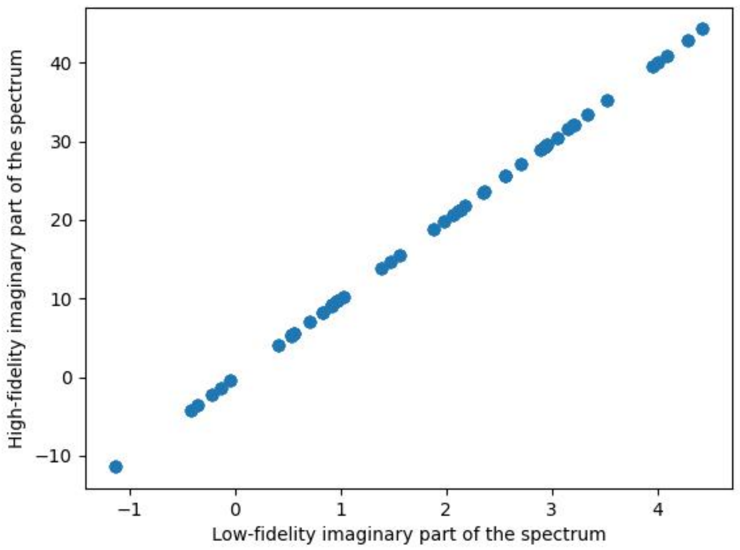


Figure 16: Relation between imaginary components of frequencies

The input to the Gaussian processes were sequences of 12 elements. The sequence corresponds to frequencies of 6 low-fidelity samples containing both real and imaginary parts of the spectrum. Since the signal is real, one obtains for 6 samples 4 complex frequency components. Additionally, the locations x_i were added to the input. The goal was given the input information $\mathbf{x}$ and the low-fidelity discrete spectrum to predict the high-fidelity spectrum on the subinterval $\mathbf{x}$. In the experiments, 200 sample sequences were used as a training data set. All of the models trained in the experiment were trained with 500 optimization steps each.

The figure 17 shows the predictions of the model with the linear combination of the

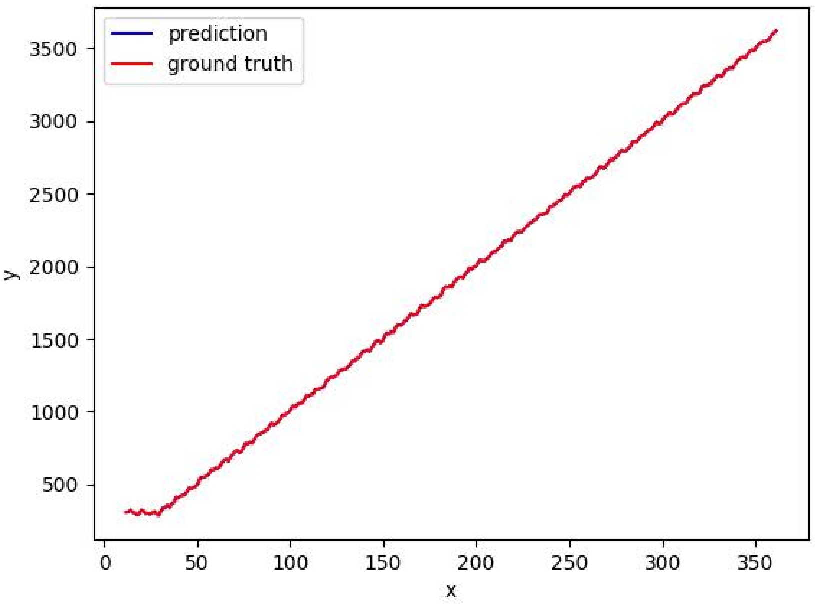


Figure 17: High-fidelity function samples predicted by the trained model

Linear, Matern and Exponential kernels transformed to the signal domain. On the plot, the mean values of the multivariate Gaussian distribution are shown. One can observe that means follow the structure of the high-fidelity function. They capture both the rising trend, the constant behavior caused by the RELU function, and oscillations. The time needed to train the model is 88.98s.

The figure 18 shows the results obtained for the models with the pure spectral mixture kernels. One can observe that the trained models were not able to learn the high-fidelity function. The learned result resembles more random oscillations without a clear struc-

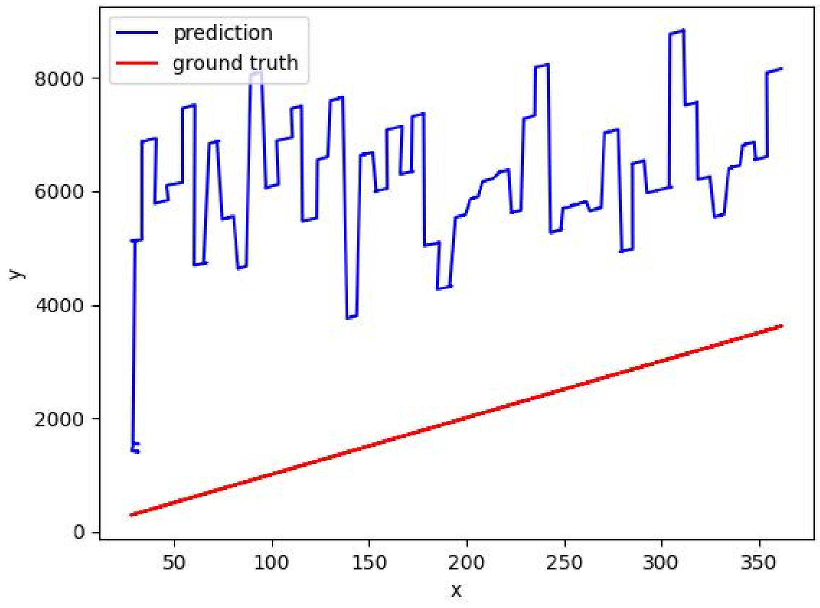


Figure 18: High-fidelity function samples predicted by the SM model

ture. The time needed to train the model is 2352.10 s.
To investigate a possible reason behind the poor performance of the spectral mixture kernels additional experiments with the model were performed. The hypothesis was that different values of the hyperparameters of the model might improve the performance. There exist multiple different parameters one can vary in an experiment. The first is the size of the training data set. The more data the model has, the more information about the underlying ground truth function it obtains. This might increase the probability of learning the underlying function. The second parameter is the number of the mixture components. The model, which has few parameters, might not be flexible enough to capture the behavior of the function. The third parameter is the number of optimization steps. The first potential idea might be to make as many optimization steps as possible. However, the authors in [HRS16] suggest restricting the number of steps in the context of Stochastic gradient descent in order to achieve better stability of the optimization algorithm and to potentially generalize better.
In the experiment, the number of training data points was varied. A linear function $f(x) = x$ was taken as the ground truth function that needs to be learned. As a model a Gaussian process with a spectral mixture kernel was taken. The kernel has 15 mixture

components. The model received samples of the form $(x_i, f(x_i))$. The hypothesis was that more training data might lead to a better quality of the model. For the experiments the model was trained with 500 steps. On figures 19, 20 and 21 one can observe the

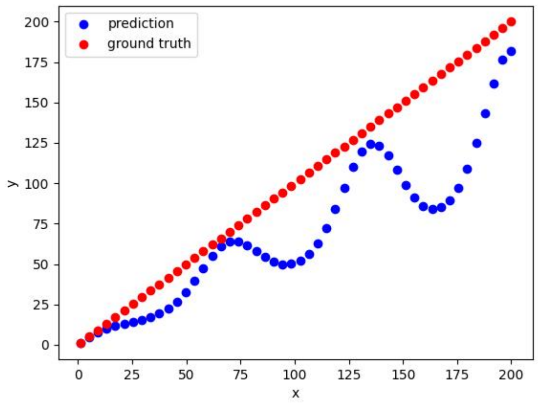


Figure 19: 200 samples in the training set

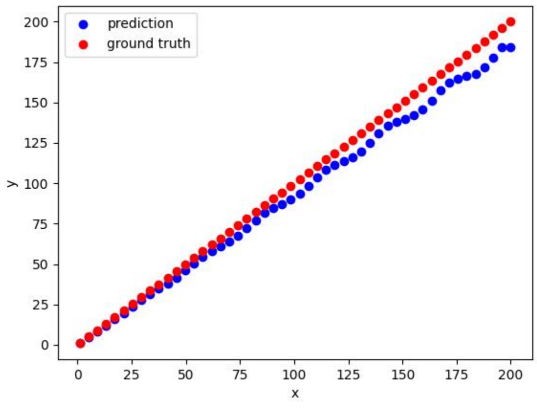


Figure 20: 400 samples in the training set

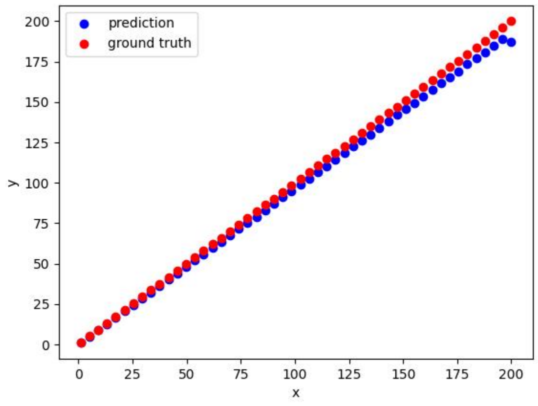


Figure 21: 600 samples in the training set

Figure 22: Predicted values by a model with the spectral mixture kernel. The ground truth function is linear. In the experiments the size of the training data set was varied.

obtained results. With the increasing number of samples, the model's predictions capture more accurately the linear structure of the ground truth function. A drawback of this approach is that in a multi-fidelity scenario it might not always be possible to

collect a larger data set to improve performance of spectral mixture kernels.
The performed experiments have demonstrated that in case of a linear relation between frequencies of the low and high- fidelity models Gaussian process models using traditional kernels have significantly outperformed models using pure spectral mixture kernels.

4.3 Learning in frequency domain. Complex relation

More complex low- and high-fidelity functions were considered in the third group of experiments. Adding frequency as inputs provides additional information for the model, but at the same time, it also increases the runtime. This raises the question of whether this information is effective and whether it significantly helps to learn the high-fidelity function or whether the effects of the additional inputs are rather negligible, considering the increased computational effort.
As a result, two types of Gaussian process models were evaluated. The first type uses only the vector x as an input, and no low-fidelity information is provided to the model. The used type of architecture is illustrated in 7. The second type adds frequencies of the low-fidelity samples to the input vector according to the schema in 6. Both groups of models were aiming at predicting the frequencies of the high-fidelity samples. The goal of the experiments was to compare their performance against each other. Additionally, the runtime needed to train the models was measured.
A subgoal was also to observe whether the results from the second experiment for the pure spectral mixture kernels having weaker performance than custom kernels can be reproduced for new pairs of low and high-fidelity functions. The BNSE initialization procedure allows selecting of parameter values for the spectral mixture kernel based on the data observed. Then the parameters are further optimized via the Adam procedure. In the experiments, a combination of the spectral mixture kernel together with the BNSE initialization was tested for both models with low-fidelity and without low-fidelity inputs. This allowed us to compare the performance of the spectral mixture kernel with BNSE against pure spectral mixture on the test functions and observe how does the BNSE improve the final predictions.
A widespread type of training in regression task is first to collect a set of points $\{(x_1, f(x_1)), ..., (x_K, f(x_K))\}$ and then from x_i to directly predict the function's value $f(x_i)$. This type of training does not involve building subsequences of samples and working with the spectrum of the samples. As a result, on the final stage of the experiments the performance of the previously constructed models working with the subsequences and predicting the discrete spectrum is compared against models which starting from exactly one point x predict its corresponding function value $f(x)$.

Both models 6 and 7 allow to use different kernels in the Gaussian process models. In the experiments performed, three different groups of Gaussian process models were considered as illustrated in 26. The models differ in the kernel used for Gaussian process models and how the parameters of the kernels are initialised.

The first group was Gaussian processes with the custom kernel being a sum of Exponential, Matern with $\mu = 0.5$, Polynomial kernel of degree 2, Rational Quadratic kernel, and Squared exponential kernel. The second group was Gaussian processes, with the spectral mixture kernel having 15 mixture components. The third group was Gaussian processes with 15 mixture components, and additionally, the BNSE initialization was applied. Each group consists of two Gaussian processes. The first process constructed according to 7 was trained by using only the x- values as input. The second process following 6 used additional frequencies of the low-fidelity function. The goal for all Gaussian processes was given a subinterval $\mathbf{x} = (x_1, ..., x_N)$ to predict discrete spectrum $(\psi_1(\mathbf{x}), ..., \psi_N(\mathbf{x}))$ of the high-fidelity models. In all of the experiments performed, the data points for both fidelity levels were nested. The parameters of kernels were optimized with Adam. The number of iterations was set to 500.

First function

For the tests two pairs of low and high-fidelity functions were chosen. The first pair is

$$f_l(x) = \sin(8 \cdot \pi \cdot x),$$

$$f_h(x) = \sin(8\pi x + \pi/10)^2 + \cos(4\pi x).$$

The figure 27 demonstrates the graphs of the low and high-fidelity functions. One can observe that both high- and low-fidelity functions are periodic. The high-fidelity function has both low and high frequencies. On the other hand, the low-fidelity function only has a low-frequency component. Also the period of the high-fidelity function is larger compared to the period of the low-fidelity function.
For the training of Gaussian process models, a training data set is needed. The following steps were performed:

- a sequence of 800 equidistant x_i values in range $[0, 1]$ was formed
- 800 samples $f_l(x_i)$ and $f_h(x_i)$ of both low- and high-fidelity functions were collected
- subsequences are formed with each subsequence being defined as $(x_i, ..., x_{i+8})$ and containing 8 samples. The same procedure is applied to the samples of f_l and f_h leading to the vectors of the form $(f_l(x_i), ..., f_l(x_{i+8}))$ and $(f_h(x_i), ..., f_h(x_{i+8}))$

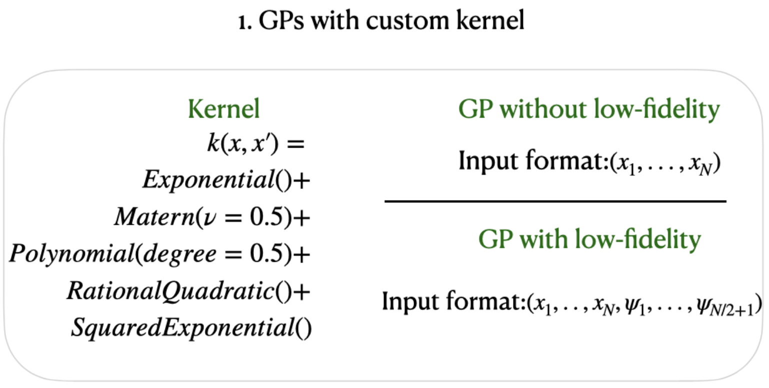

1. GPs with custom kernel
Kernel
$k(x, x') =$
$Exponential()+$
$Matern(\nu = 0.5)+$
$Polynomial(degree = 0.5)+$
$RationalQuadratic()+$
$SquaredExponential()$
GP without low-fidelity
Input format:$(x_1, \ldots, x_N)$
GP with low-fidelity
Input format:$(x_1, \ldots, x_N, \psi_1, \ldots, \psi_{N/2+1})$

Figure 23: Gaussian processes with custom kernel

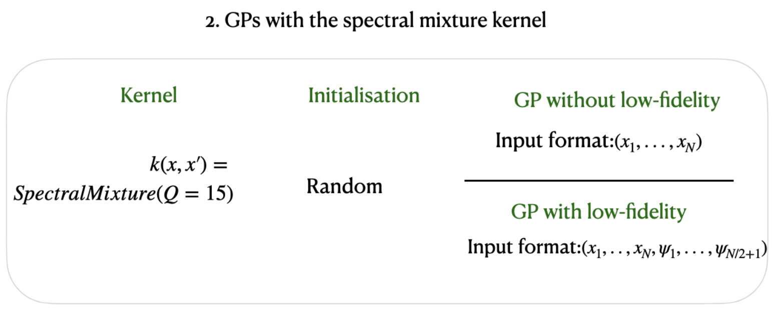

2. GPs with the spectral mixture kernel
Kernel
Initialisation
GP without low-fidelity
Input format:$(x_1, \ldots, x_N)$
$k(x, x') =$
$SpectralMixture(Q = 15)$
Random
GP with low-fidelity
Input format:$(x_1, \ldots, x_N, \psi_1, \ldots, \psi_{N/2+1})$

Figure 24: Gaussian processes with the spectral mixture kernel

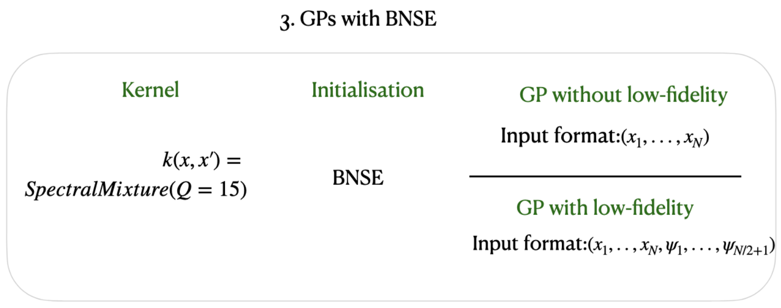

3. GPs with BNSE
Kernel
Initialisation
GP without low-fidelity
Input format:$(x_1, \ldots, x_N)$
$k(x, x') =$
$SpectralMixture(Q = 15)$
BNSE
GP with low-fidelity
Input format:$(x_1, \ldots, x_N, \psi_1, \ldots, \psi_{N/2+1})$

Figure 25: Gaussian processes with the BNSE initialisation

Figure 26: Three groups of Gaussian processes considered in the experiments. The groups differ in the choice of the kernel function for Gaussian processes and initialisation of the kernel's parameters. Each group consists of two Gaussian processes. One works purely with the input points x, another also takes discrete spectrum of the high-fidelity function as a part of the input.

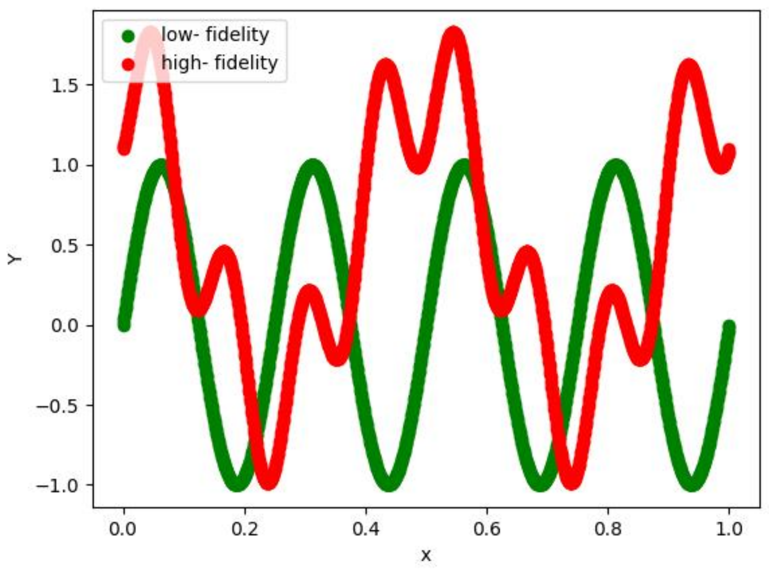


Figure 27: Low and High-fidelity functions in the signal domain

- the RFFT is applied to the subsequences $(f_l(x_i), ..., f_l(x_{i+8}))$ and $(f_h(x_i), ..., f_h(x_{i+8}))$

The Real Discrete Fourier transform of a subsequence returns a 5-dimensional vector of amplitudes for a subsequence of 8 samples. Since real and imaginary components are predicted separately by two separate Gaussian processes, this results in 10 Gaussian process models which are trained independently. The choice of the 8 samples per sequence was made after empirical observation that this number of points does not lead to a very long training time of the models. At the same time models resulting from the sunsequences are able to capture some of the behavioral patterns of the high-fidelity function. For testing the procedure 4.3 was applied to generate a data set with 200 subsequences for each x, f_l and f_h.
The first experiment was to train Gaussian processes from the first group defined in 26 with custom kernel. The group contains two Gaussian processes. One uses only the subsequences $(x_i, ..., x_{i+8})$ to form its predictions. Another also takes into consideration the values of the discrete spectrum of the low-fidelity function. Figure 28 demonstrates the model trained on inputs with frequencies of the low-fidelity function. One can observe that the model has learned the location of peaks and pits. However, in most cases, the amplitudes of the peaks are predicted wrong. This might indicate that for a subinterval $(x_i, ..., x_{i+8})$ in the predicted high-fidelity spectrum by the model the low-frequency

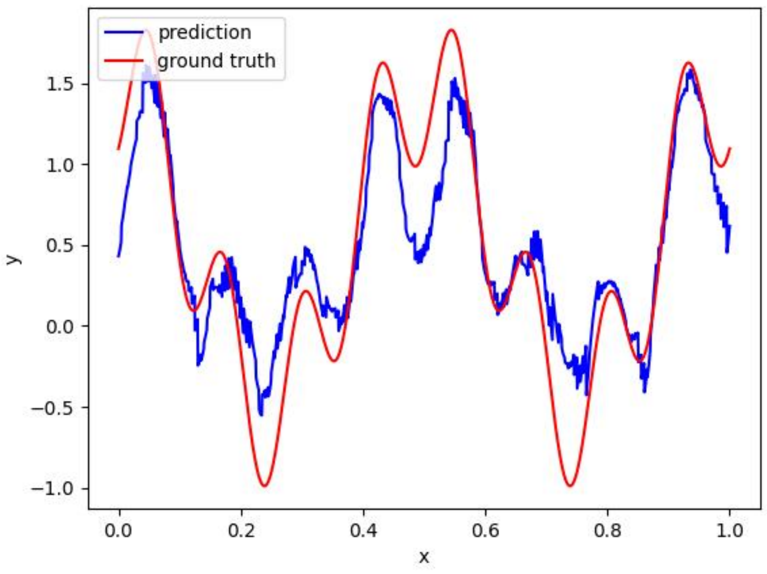


Figure 28: Inputs with frequencies

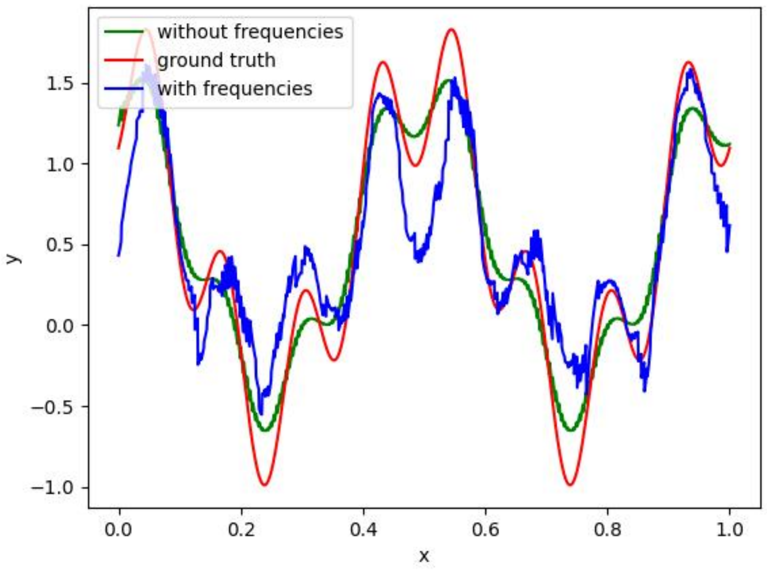


Figure 29: Inputs without frequencies

Figure 30: Complex relation, first pair of low- and high-fidelity functions: model with the kernel from 23 trained on the inputs with and without low-fidelity discrete spectrum. The model was trained to predict the first pair of low- and high-fidelity functions.

components might contain an error which causes the effect observed. The values of the model's prediction are contained in the interval of the approximate width of 2. This is the same value as the amplitude of the low-fidelity function. For the high-fidelity function the values are in the interval between $[-1, 2]$ of width 3. In this scenario it might be that the model has failed to learn the structure of the high-fidelity function making errors in the amplitudes of both the low and high-frequency components of the ground-truth function. Instead, the model might inherit the interval width of 2 directly from the low-fidelity function. One additional observation is that the ground truth function is symmetric. However, there is no strict symmetry in the model's prediction. Figure 28 shows an example of this around the peaks at $x = 0.3$ and $x = 0.8$. The period of the ground truth function is 0.5, so the function around $x = 0.3$ and $x = 0.8$ should have exactly the same behavior. But the prediction shows a shift upwards of the values around $x = 0.3$. This might indicate that the model did not completely capture the periodicity of the ground truth function. This, in turn, might be an indication that the low-frequency components on the subintervals were not captured correctly. Also, the predicted function shows a lot of high-frequency oscillations. These oscillations might come from the errors when predicting high frequencies on the subintervals. A possible source of the errors might be a potentially complex relation between the frequencies of the low-fidelity function and the frequencies of the high-fidelity function. The low-fidelity spectrum might not be informative when learning the high-fidelity spectrum. Also, the frequencies of the high-fidelity might define a complex function to fit for a Gaussian process. The chosen kernels might not be flexible enough to fit it. An alternative source of errors might be the length of the subintervals used as an input. If the interval contains too few samples, then it might be insufficient to capture and model high-frequency terms.

The figure 29 also shows the predictions of the model trained without frequency input as the green curve. One can observe that the model, similarly to the model trained with frequency inputs, has the problem of overshooting or undershooting peaks and pits of the ground truth function. Since both models operate on subintervals of the same length, in both cases this might lead to errors when capturing the high frequencies. One can observe that in different regions either the model trained on frequencies gives a better prediction or the model trained without frequencies. Overall, the model without frequencies shows fewer jumps and is more smooth. This might be an indication that the model is more certain on how the ground truth function looks locally compared to the model with frequencies. As a result, one conclusion might be that for this setting adding additional frequency input did not lead to a significant increase in the prediction quality.

An interesting insight might be derived from the functions that Gaussian processes are trying to learn. The final goal is given a subinterval of inputs $(x_1, ..., x_N)$ to predict a discrete spectrum of the high-fidelity function $(\psi_1, ..., \psi_N)$ on that subinterval. Each component of the discrete spectrum ψ_i containing a real and an imaginary part is learned by two Gaussian processes. The first process learns the real part, and the second the imaginary part. This forms a function with the function domain being a vector $(x_1, ..., x_N)$, which depending on the setting can contain additional dimensions with low-fidelity spectrum, and the function value being the real or imaginary part of the $i-$th component of the spectrum, $\psi_i.real$ or $\psi_i.imag$. Figure 31 shows the first Fourier ψ_1 coefficient projected on the first dimension of the input vector x_1. For the first coefficient of the Fourier series holds that its imaginary part is zero. Which is why the coefficient is a real number. The graph also shows the predictions of the Gaussian process with custom kernel and frequency inputs. One can observe that the model makes a rather significant error when predicting some of the sharp peaks and pits. The graphs 32 and 33 show the real and the imaginary part of the second coefficient. One can observe that the real part was modeled relatively accurately. The imaginary part shows slightly more errors when it comes to the amplitudes of the oscillations. On one hand, the graphs suggest improving the prediction of the first coefficient. One potential way to do it is to select a more flexible kernel. On the other hand, the plots for the second component $\psi_2.real$ and $\psi_2.imag$ show a function that significantly differs from the first coefficient $\psi_1.real$. As a result, if such a situation occurs, one might consider having different kernels optimized for a particular behavior of the ground truth function $\psi_i.real$ or $\psi_i.imag$.

The second experiment was to look at the spectral mixture kernels without the BNSE initialization. The figure 35 shows the results for the Gaussian process model with the spectral mixture kernel using inputs without frequency components. One can observe that the model did capture the low-frequency component corresponding to $cos(4\pi x)$. The high-frequency peaks and pits resulting from the addition of the squared sinus component were not captured. This might be connected to the results obtained in section 5.2 for the spectral mixture kernels, which show that spectral mixture kernels might need more training data to capture the function accurately. Figure 36 shows models trained with spectral mixture kernels and using inputs with frequency components. One can observe that the model also recognizes the global oscillation. However, unlike the model without frequency components, figure 36 shows small peaks and pits on locations where the ground truth function also has them. This might be an indication that adding information in the form of frequencies of the low-fidelity function did help to guide the model toward more accurate predictions. One might also observe that the amplitudes of the peaks are too small compared to the ground truth. This might be caused by errors in low-frequency components on the subintervals when forming a prediction by the model. Also, the model oscillates heavily. This might be an effect of predicting the high frequencies wrong with their amplitudes being too high.

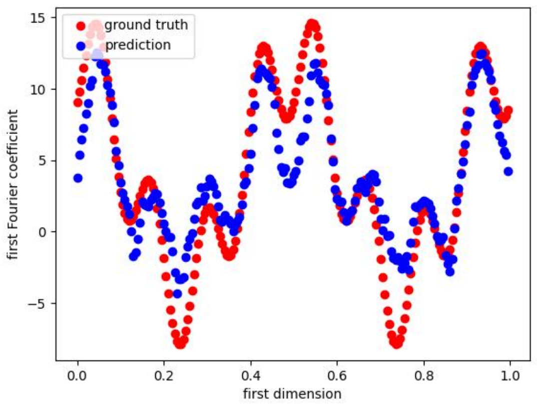


Figure 31: First coefficient

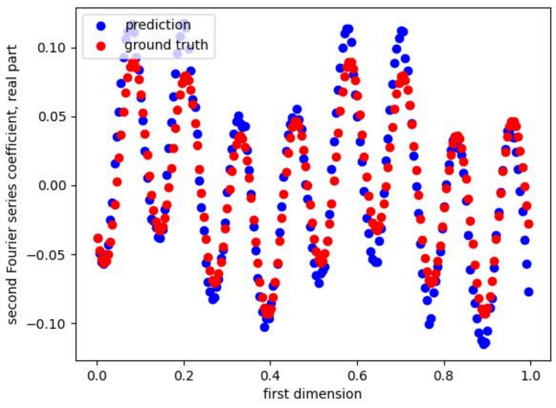


Figure 32: Second coefficient, real part

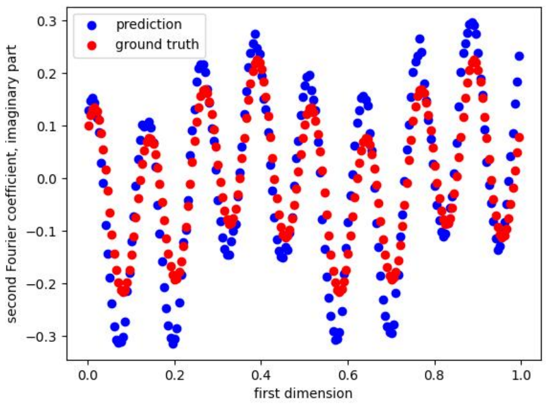


Figure 33: Second coefficient, imaginary part

Figure 34: Complex relation: Target functions corresponding to the first real, second real and second imaginary Fourier coefficients. Also, the model's predictions

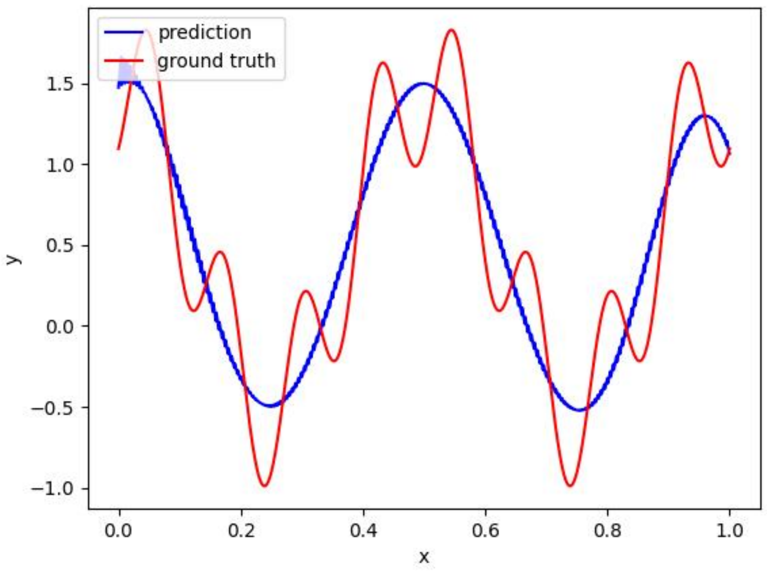


Figure 35: Inputs without frequencies

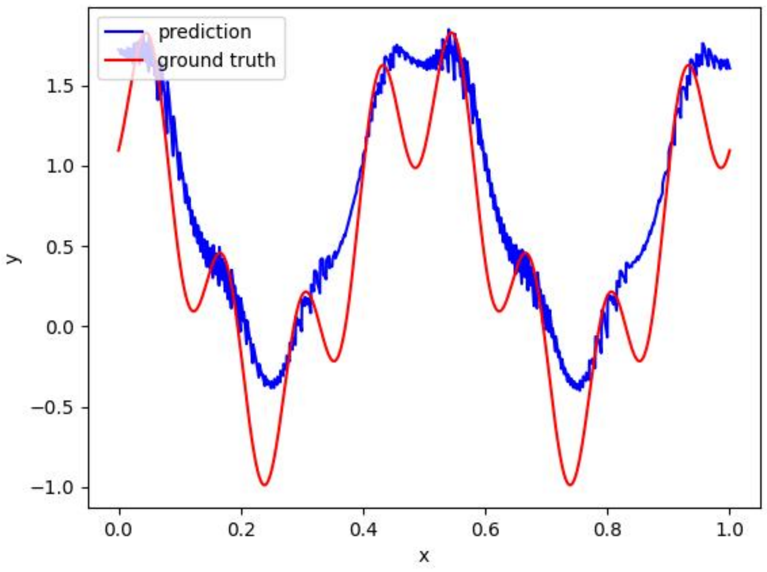


Figure 36: Inputs with frequencies

Figure 37: Complex relation: the model with the spectral mixture kernel trained on inputs with and without low-fidelity discrete spectrum. The model predicts the high-fidelity function from the first pair.

Table 1: Time needed to train models with low-fidelity spectrum and wihtout the spectrum

Model	Runtime in s
Spectral mixture	334.81
Custom kernel	40.74
Spectral mixture with frequencies	653.88
Custom kernel with frequencies	107.34
Spectral mixture with BNSE	430.931
Spectral mixture with BNSE and frequencies	853.0.37

The third experiment was to add the BNSE initialization to the spectral mixture model. On the one hand, this allows to see whether adding frequencies will significantly improve the prediction. On the other hand, this allows to compare the performance of the spectral mixture kernel with BNSE initialization against pure spectral mixture and against custom kernel from the first experiment. Figure 38 shows the obtained results for the model trained without frequency inputs. One can observe that the model shows very accurate predictions and significantly outperforms all the models trained in the chapter. A drawback of using the BNSE initialization is the increased runtime.
One important consideration when selecting a Machine learning model is the computational resources needed to train and evaluate the model. In the experiments, multiple models were trained. The training was done sequentially. The following table 1 shows computational time needed to train the models from 28, 29, 36, 35, 38, 39: One can observe that the models using the custom kernel were much faster to train. The increase of the input dimensionality through the low-fidelity input leads to an approximately linear increase in the computational time. The BNSE initialization did increase the time by approximately 25 percent.

Figure 39 shows the obtained results for the BNSE models using frequencies as a part of the input. One can observe that the frequency of input did not lead to a significant improvement in the model's predictions. One small observation is that the BNSE on figure 39 has made slightly more errors compared to the performance on 38. One possible explanation is that BNSE is optimal for kernel matrices having the Toeplitz structure discussed in 3.2. According to [Wil14], this structure arises when the grid is regular and the kernel is stationary. The input subsequences of x were collected at regular intervals. As a result, in the setting where only the x is used as input, the BNSE initialization leads to an accurate fit of the model. However, if the frequencies

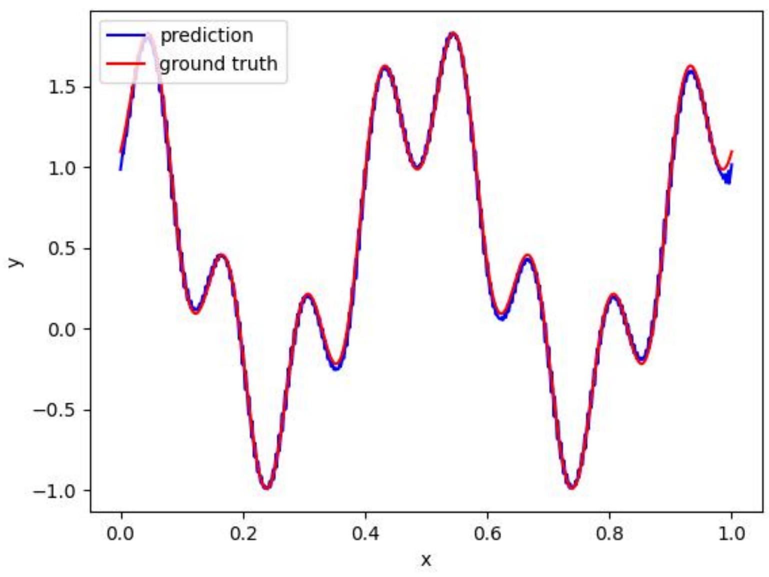


Figure 38: Inputs without frequencies

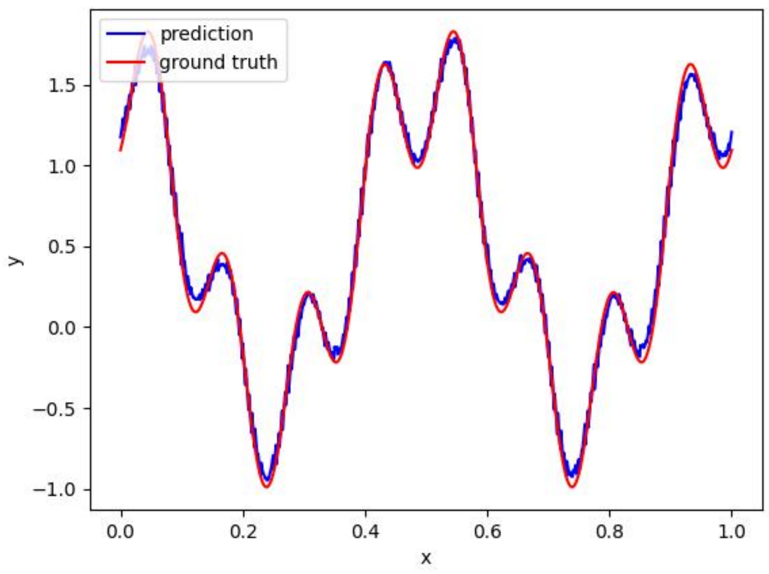


Figure 39: Inputs with frequencies

Figure 40: Complex relation: The model with the spectral mixture kernel and the BNSE initialisation trained on inputs with and without low-fidelity inputs. The model was trained on the first pair of the low- and high-fidelity functions.

Table 2: RMSE of the models with low-fidelity spectrum and without the spectum

Model	RMSE
Spectral mixture	0.3452
Custom kernel	0.193
Spectral mixture with frequencies	0.3933
Custom kernel with frequencies	0.3419
Spectral mixture with BNSE	0.037
Spectral mixture with BNSE and frequencies	0.057

of the low-fidelity function are added to the input, the resulting vectors might not be equidistanced anymore. Then, the BNSE initialisation might be suboptimal, and further optimisation might be necessary to improve the predictions.
The spectral mixture kernel for all models has 15 mixture components. This is also a hyperparameter which can be optimized. The Gaussian process with a pure spectral mixture kernel has shown significantly worse performance compared to the model with custom kernel. One could consider the insufficient number of the mixture components as a potential reason for the performance problems. The kernel is not flexible enough to capture the ground truth function. However, the results for the BNSE kernel show that a SM model with 15 components can perfectly capture the underlying function. As a result, a source of error for the pure spectral mixture kernel might lie not in the number of components but in the found parameters for these components. This is a problem of optimization. One hypothesis might also be that Spectral mixture kernels introduce a more complex objective function to optimize compared to the custom kernels.
One statistics which can be used to measure performance of the models is the RMSE. For all 6 models considered in this chapter the following table 2 presents the RMSE values: Interestingly, one can observe that in terms of the RMSE error the models with pure spectral mixture kernel and the spectral mixture with frequencies with frequencies have a similar performance to the custom kernel with frequencies. Both models with the spectral mixture kernel mistly capture the low-frequency component of the ground-truth function. Whereas the custom kernel also tries to model the high-frequencies. However, this modelling is no perfect and in some regions, for example around $x = 0.5$ leading to large deviations from the ground truth function. This deviation might lead to an effect that on average imprecise modelling of high-frequencies gives no advantage compared to models which largely focus on low-frequencies. The second observation is that for the custom kernel with frequencies despite better on some regions of the ground-truth function compared to the custom kernel without frequencies the average

RMSE is worse.

Second function

The second pair of the low- and high-fidelity functions considered is defined as

$$f_l(x) = \sin(8 \cdot \pi \cdot x),$$

$$f_h(x) = \sin(8 \cdot \pi \cdot x)^2.$$

Figure 41 shows plots of 800 samples from the low and high-fidelity functions. The high-

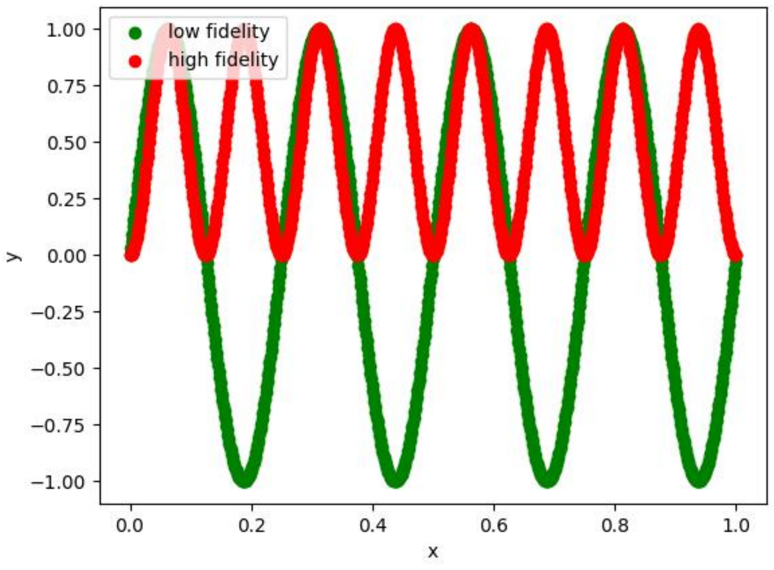


Figure 41: Low and high-fidelity functions

fidelity function shows a simpler behavior compared to the high-fidelity function in the previous section. In the performed experiments, the number of training subsequences was reduced to 60, each consisting of 8 samples. The same models as in the previous section were considered. The models were trained with 500 steps using the Adam algorithm and evaluated on 100 subsequences.

The figure 42 shows the obtained results for the Gaussian process model using the kernel from 23 and no frequencies in input. One can observe that the model makes an error in the amplitudes of the oscillations. Figure 43 depicts the results for the

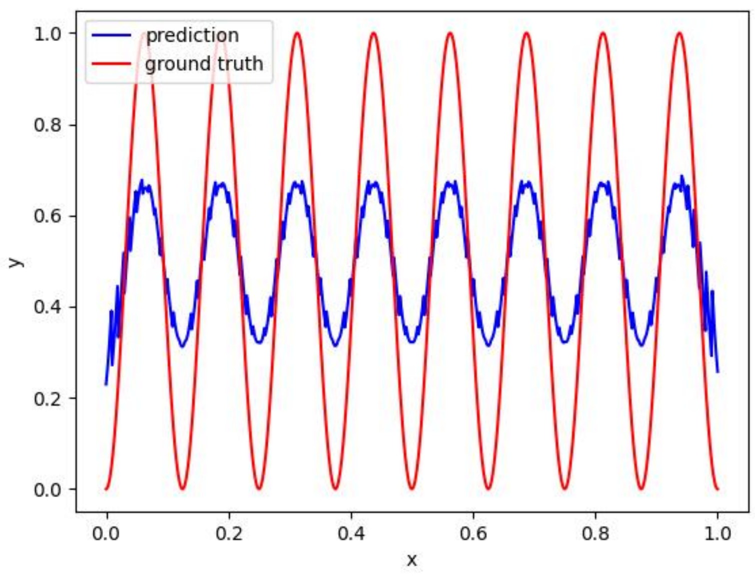


Figure 42: Inputs without frequencies

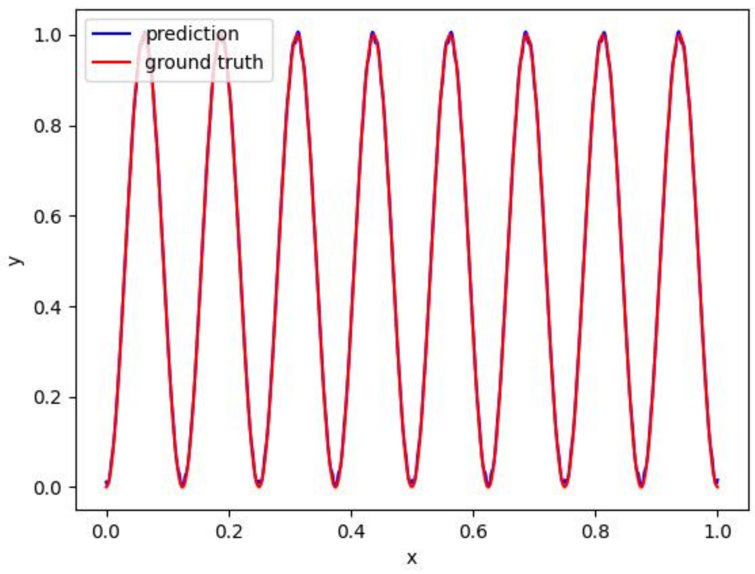


Figure 43: Inputs with frequencies

Figure 44: Complex relation: Models with the kernel from 23 trained with and without low-fidelity frequencies as part of the input. The models predict the second pair of the low- and high-fidelity functions.

model using frequency inputs. Its predictions are more accurate compared to the model without frequencies. This might be an indication that the additional information provided by the low-fidelity frequencies might have helped to improve the prediction of the model.
The figure 45 shows the observed results for the model with the spectral mixture kernel and without frequencies in the input. One can observe that the model did not catch the structure of the ground truth function. The figure 46 shows the same model but with the frequencies in the input. The model does see the oscillations. The amplitude of the oscillations is not predicted correctly. Also, the prediction is shifted upwards. This might result from the wrong prediction of the low-frequency components on the subintervals. The model itself shows sharp jumps. This might be an effect of errors in the high-frequency components on the subintervals.
Finally, the spectral mixture model with the BNSE initialization without frequency inputs in shown on the figure 48. The model did capture the behavior of the ground truth function. A similar result can be seen on predictions 49 generated by the model with spectral mixture kernel, the BNSE initialization, and using frequencies as input. The observed results might be an indication that for the second pair of the low- and high-fidelity functions the frequencies of the low-fidelity function have led to an improvement in prediction. One possible reason might be a simpler correlation between low and high-fidelity functions, which allows for the model to extract easier the information needed to accurately predict the high-fidelity function.
The experiments that were performed showed that using frequencies of the low-fidelity function as additional input to the model might improve the model's prediction. However, this is not necessarily the case and for some low- and high-fidelity pairs adding frequencies might worsen the quality of the model. One additional observation was that the length of the subintervals used as inputs might influence the quality of the model's predictions.

Training in signal domain

In the experiments performed in this chapter, the question was whether the information of low-fidelity frequencies can help when trying to predict high-fidelity frequencies. Another approach as described in [Rav+24] is to directly take samples from low-fidelity without forming subsequences and try to predict high-fidelity points. In the experiments, the number of training data points is kept constant compared to the experiments with frequencies. The chosen Gaussian process kernels are also the same. A low-fidelity sample is also used as a part of the input to a Gaussian process. A goal is to predict a high-fidelity sample on the same location x. The data points are assumed to

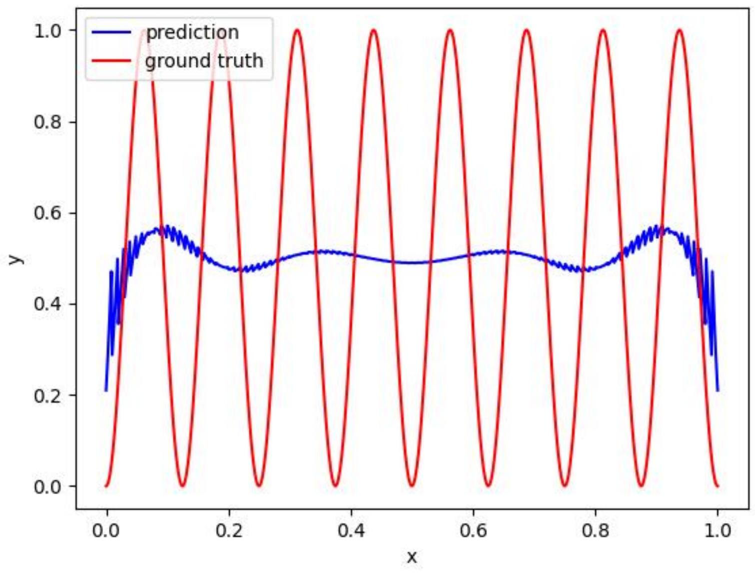


Figure 45: Inputs without frequencies

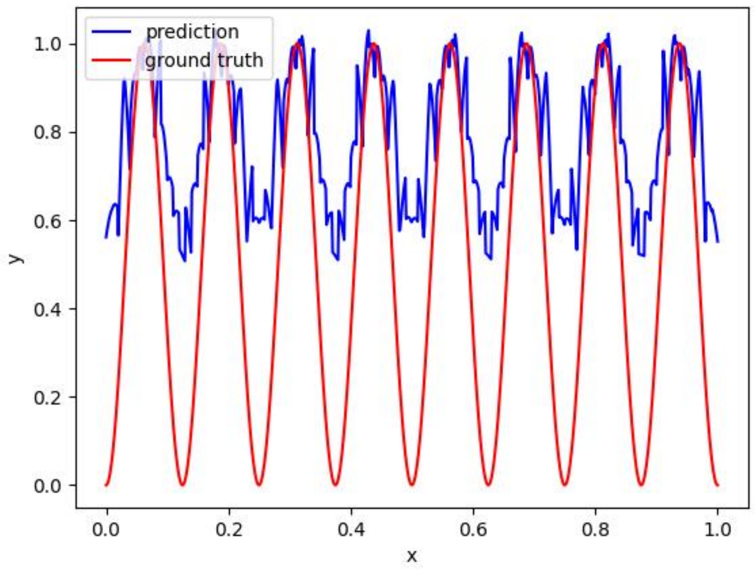


Figure 46: Inputs with frequencies

Figure 47: Complex relation: Models with the spectral mixture kernel trained with and without low-fidelity frequencies as part of the input. The models were trained on the second pair of the low- and high-fidelity functions

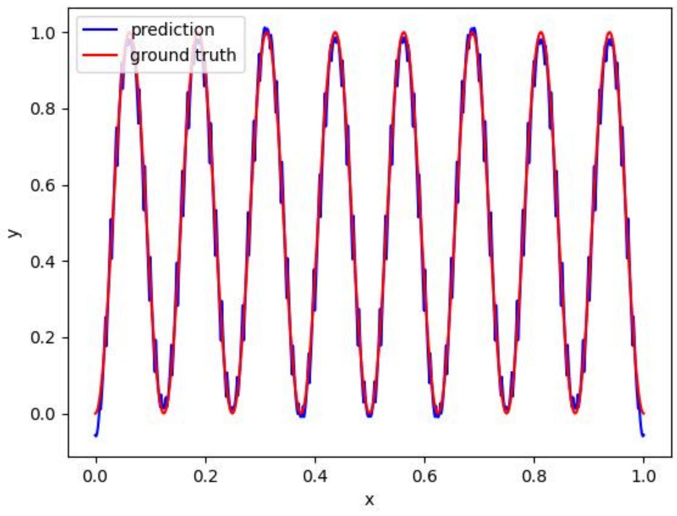


Figure 48: Inputs without frequencies

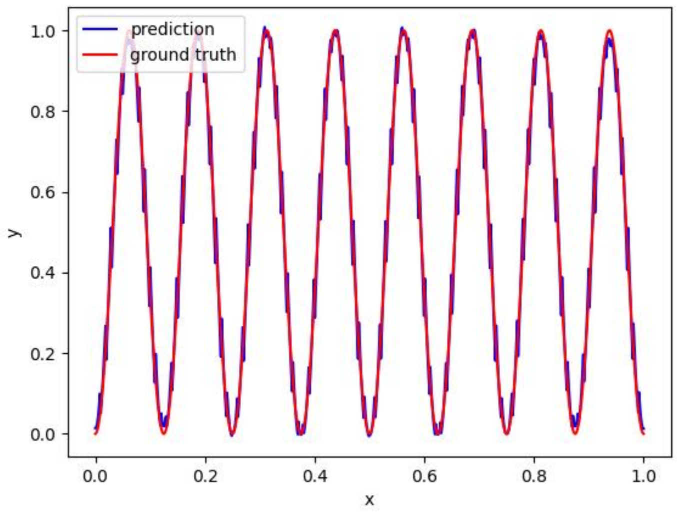


Figure 49: Inputs with frequencies

Figure 50: Complex relation: Models trained with the spectral mixture kernels and the BNSE initialisation trained with and without low-fidelity frequencies as part of the input. The models were trained on the second pair of the low- and high-fidelity functions.

be nested. Since no frequencies were involved, in each case only one Gaussian process model was trained, taking as input 2 dimensional input $(x, f_l(x))$ and producing the output $f_h(x)$.

The figure 51 shows the obtained results for the custom kernel. The model also shows not purely periodic prediction. This might support the hypothesis that for the predictions with frequencies, one possible reason for the lack of symmetry in 28 was the chosen kernel. The prediction's quality is increased compared to the models with the custom kernel trained on subsequences.
The figure 52 shows the predictions made by the model with the pure spectral mixture kernel. The quality of predictions is significantly improved compared to the training on the subsequences for the spectral mixture kernel. One source of the observed improvement might be the amount of available information. In the models trained on subsequences each Gaussian process predicts its own entry in the frequency vector. It does see the full subsequence of the low-fidelity functions. But from the high-fidelity, it only sees its vector entry, which it has to predict. For a subsequence with 5 samples. It would be $1/5$ of the overall high-fidelity training data. The results of the experiments in the previous chapters might indicate that the performance of the spectral mixture kernels might be improved with the increased amount of available training data. This might have been a reason for the improved performance in the 52 having more training data on high-fidelity function available to the model. Another source of improvement might be the low-fidelity samples in the signal domain. They might have helped to simplify the function to be optimized. Which, in turn, has led to a better choice of the parameters for the spectral mixture kernels.
Finally, the figure 53 plots the predictions of the BNSE initialization together with the spectral mixture kernel. It has the best performance among all three models. Its performance is similar to the models with BNSE trained on the subsequences. The results of the models with BNSE have shown that the initialization method might help a lot in selecting better parameters for a spectral mixture GP.
The figures 55, 56, 57 plot the results obtained for the second pair of low- and high-fidelity functions. All three models have accurately predicted the ground-truth function.

4.4 Incorporating frequency training in NARGP

In the previous experiments, the multi-fidelity structure was very simple. Gaussian processes using the low-fidelity information had one kernel

$$k_h(x, x') = k((x, f_l(x)), (x', f_l(x'))), \tag{6}$$

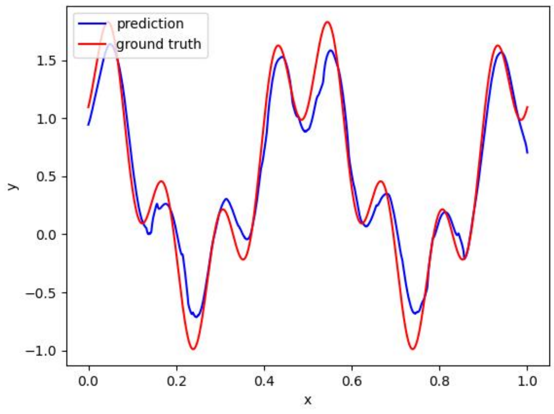


Figure 51: Custom kernel

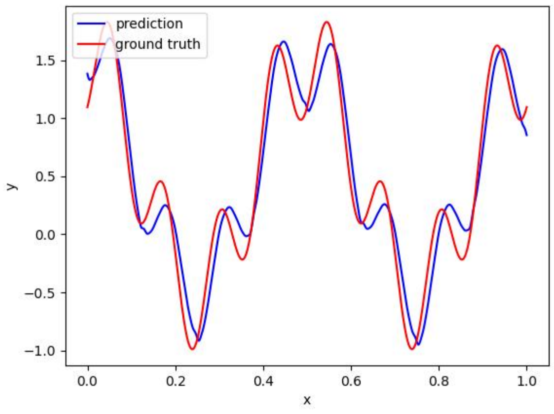


Figure 52: Spectral mixture kernel

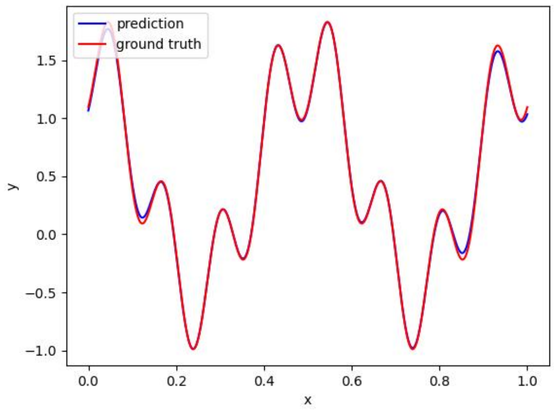


Figure 53: Spectral mixture with BNSE

Figure 54: Multi-fidelity models with the same kernels, but trained purely in the signal domain. Each model uses as input $(x, f_l(x))$.

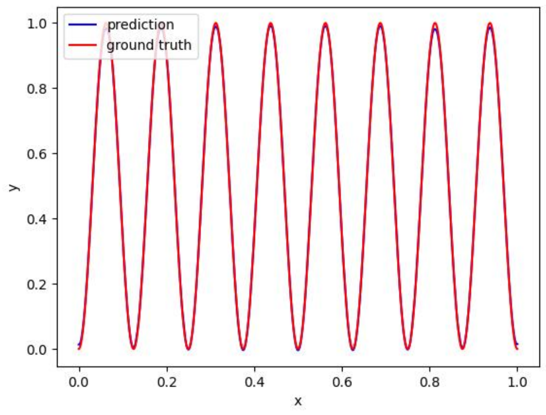


Figure 55: Custom kernel

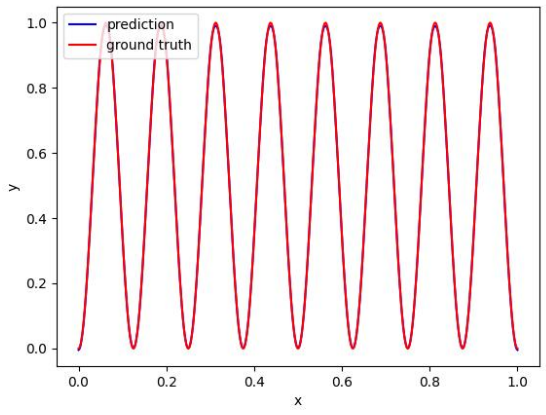


Figure 56: Spectral mixture kernel

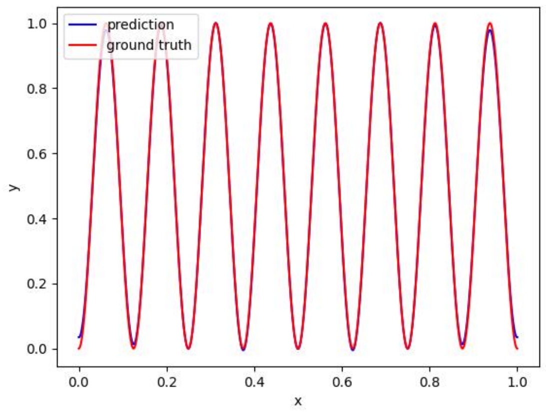


Figure 57: Spectral mixture with BNSE

Figure 58: Multi-fidelity models trained on the signal's samples plotted for the second pair of the low- and high-fidelity functions

which received both the input x and the low- fidelity frequencies. The idea of NARPG is to use a more complex kernel the components of which work on separate parts of the input vector

$$k_p(x, x') \cdot k_f(f_l(x), f_l(x')) + k_\sigma(x, x'). \tag{7}$$

In this chapter the NARGP kernel was combined with the model 6. Each Gaussian process predicting a component of the discrete spectrum has received a NARGP kernel. Each of the kernel's three components k_p, k_f and k_ρ received the same kernel $\hat{k}$. Different choices for the kernel $\hat{k}$ are possible. In the performed experiments two kernels same as in the experiments for the first and the second groups from 26 were chosen. This allows to compare the results for the NARGP kernel with results computed in the previous sections.
The experiments in this chapter were aiming at answering two questions. The first question is whether the use of the NARGP kernel 7 in the model 6 leads to much better results, compared to the use of the kernel 6 in the same model. Can a better structure of kernel lead to a more efficient extraction of the information from the low-fidelity frequencies? The second question is to compare performance of the model with the NARGP kernel in two scenarios, using the samples $f_l(x_i)$ and the discrete spectrum of the samples ψ_l to form the prediction. The idea is to empirically observe which type of the input, a discrete spectrum of the low-fidelity or the values in the signal domain, was the most helpful for the model to learn the high-fidelity function.
The number of optimization steps and training points was kept constant compared to the previous experiments. Throughout this chapter the custom kernel refers to the kernel used in the first group of 26. The figure 59 shows the obtained results for the custom kernel combined with NARGP. The results show a significant improvement compared to the model using the same kernel with frequency input from 28. This might indicate that a structure of the chosen kernel might play a significant role when learning a Gaussian process with frequency inputs in the multi-fidelity setting. Separation of the spatial x inputs and the low-fidelity $f_l(x)$ inputs did lead to better results in the performed experiments. An additional observation is that the Gaussian process with custom kernel evaluated only on the spatial inputs x in 29 makes an error with the global amplitude. Its predictions cannot capture the magnitude of the sharp peaks and deep pits of the function. The performed experiments demonstrate that a model with the NARGP kernel and frequency input was able to more accurately model high-frequency components of the ground-truth function. The RMSE of the model is 0.1288. The figure 60 shows the results for the pure spectral mixture kernel. One can observe that the NARGP model did not capture the ground truth function. The predictions from the model closely resemble that of the experiment 36. This might be a piece of additional evidence that in this experiment, it is not that the kernel is not flexible

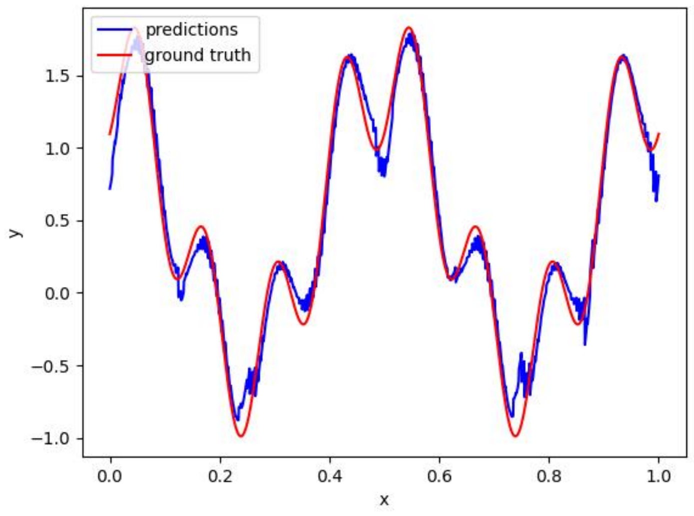


Figure 59: Custom kernel with frequencies

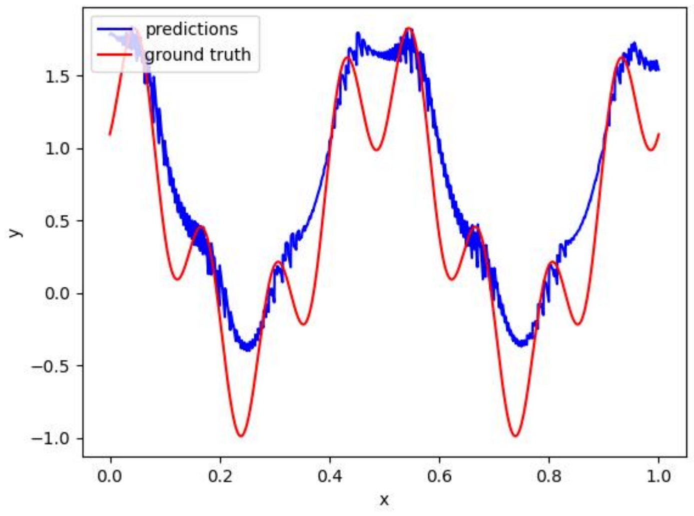


Figure 60: SM with frequencies

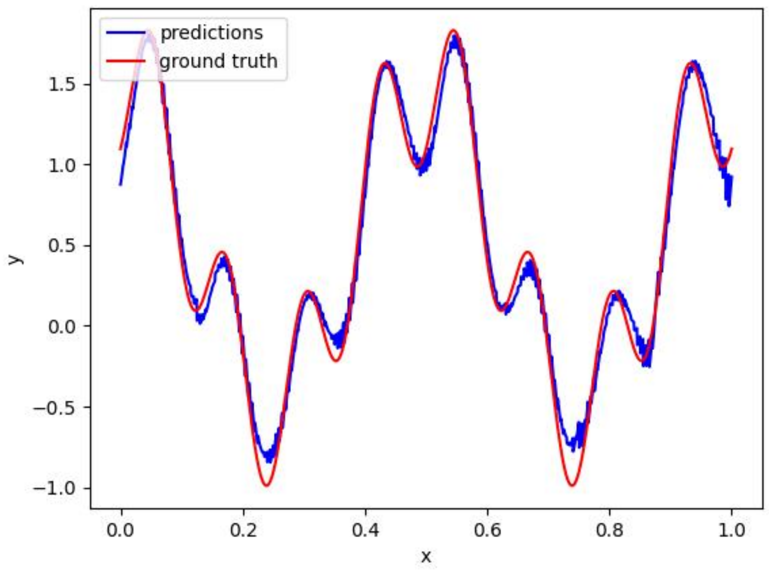
Figure 61: Models with the NARGP kernel and frequencies

enough to capture the ground-truth function or that the low-fidelity frequencies are

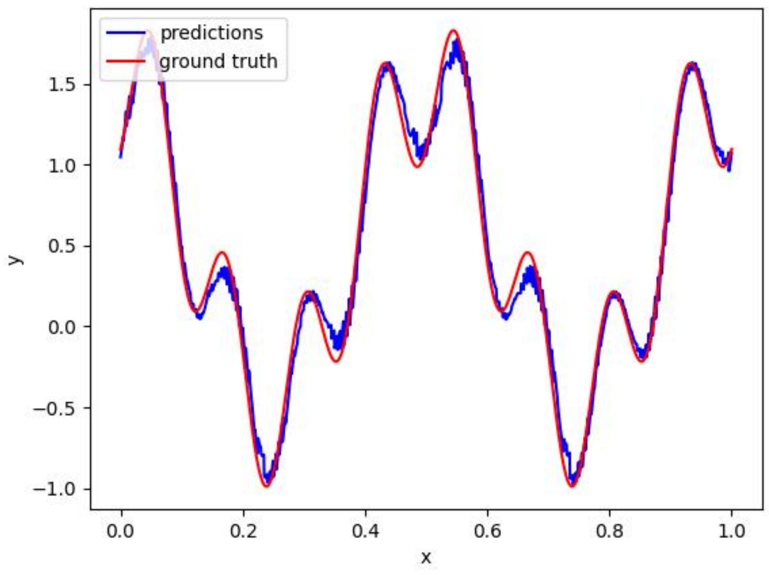
Table 3: RMSE error of the models trained with NARGP kernel

Model	RMSE
NARGP with spectral mixture kernel and low-fidelity samples	0.097
NARGP with custom kernel and low-fidelity samples	0.1066
NARGP with spectral mixture kernel and low-fidelity spectre	0.387
NARGP with custom kernel and low-fidelity spectre	0.1288

not informative, but rather the optimization procedure returns parameters that lead to poor performance of the model. The RMSE of the model is 0.387.
Similarly to the previous chapters the experiments were repeated on the same models, but without the frequency input. Instead, the values of the low-fidelity function $f_l(x)$ were used as input to the NARGP kernel. The goal was to observe which form, in signal or in frequency domain, of the low-fidelity samples will lead to better predictions. The figure 62a shows the obtained results for the custom kernel. The results are slightly more accurate compared to the 59. Interestingly, in both settings with low-fidelity information in signal and in frequency domain the models struggle with the two pits near 0.2 and 0.7. It might be that it is a problem with the structure of the custom kernel itself, that it is not flexible enough to capture high- frequencies of the function.
The figure 62b illustrates predictions for the spectral mixture kernel. Compared to the 60 and 35 the predictions have improved significantly. The model accurately captures the amplitudes of the peaks. Contrasting to the results for the frequency inputs in 59 the prediction has become much more exact. Since from 39, it is known that a spectral mixture kernel can learn the ground truth function. So, the reason for the poor performance is not the inflexibility of the kernel. But it is the choice of the parameters that leads to good or bad performance of this model. It might be that NARGP and inputs without frequencies did simplify the optimization problem, so that nearly optimal parameter values were found for the kernel. A similar effect for the frequency inputs was not observed in the experiments.
The following table 3 demonstates the RMSE errors measured for all four models considered in this chapter. In the table one can observe that the NARGP models using the low-fidelity spectrum have shown bigger RMSE errors compared to the models using the low-fidelity samples. The pure spectral mixture with the low-fidelity samples has outperformed the model with custom kernel and the low-fidelity samples. An opposite situation can be observed for the models having low-fidelity spectrum. The model with custom kernel has a three times smaller RMSE than the model with the spectral mixture kernel.



(a) Custom kernel without frequencies



(b) SM without frequencies

Figure 62: Models with NARGP kernel and signal inputs

As a result of the performed experiments, the NARGP kernels might significantly improve the results of the models. The additional structure in kernel might lead to more effective extraction of the low-fidelity information. Additionally, NARGP might help to simplify the optimization problem for the spectral mixture kernels. Leading even without the BNSE initialization to a good quality of the predictions.

5 Conclusion

In the project, the idea of training Gaussian processes in the frequency domain was explored. A model 6 uses both, the input x and the discrete spectrum of the low-fidelity function to learn the spectrum of a high-fidelity function. Gaussian processes in the model are trained independently. The model allows to use different kernels for the Gaussian processes. An important kernel is the spectral mixture kernel. Its performance was compared against the performance of the custom kernel defined in 23. In a group of experiments the spectral mixture kernel was enhanced by applying the BNSE initialization of the kernel's parameters. Finally, the NARGP kernels were tested in combination with the model 6.
The experiments that were performed led to multiple interesting results. The first outcome of the project is that training the Gaussian process in the frequency domain might improve predictions in the multi-fidelity setting. However, for the constructed model, it is not guaranteed that the results will automatically improve when the low-fidelity frequencies are added. Rather, it might be function dependent. In the experiments also the cases were observed when adding the frequency information either did not play a decisive role when forming the prediction or has even slightly made the prediction's quality worse.
The second observation is that optimizing the spectral mixture kernels might be difficult. The BNSE initialization procedure might lead to significant improvement of the parameters. However, when applying BNSE to a model with frequency inputs, the resulting model might be suboptimal. A potential reason might be that when adding frequencies to the inputs, the kernel matrix does not have the Toeplitz structure anymore, because the inputs no longer form a regular grid. This leads to the BNSE algorithm providing parameter values that are suboptimal. Another observation is that spectral mixture kernels might have larger runtime needed to train them compared to kernels composed from the Matern, Polynomial, Exponential, SquaredExpoenntial and RationalQuadratic. The BNSE initialization additionally increases the runtime.
The structured kernel of the NARGP model has significantly improved performance on the tested models. An interesting result is that a spectral mixture kernel, together with the NARGP kernel, outperformed all cases with the spectral mixture kernel except

for the cases with the BNSE initialization. A potential reason might be that NARGP has effectively extracted the low-fidelity information. That might have simplified the optimization procedure for the spectral mixture kernel.

5.1 Future work

The model in 6 assumes that the frequency components of the spectrum are independent. As a result, an independent multi-output Gaussian process is constructed. An extension of this setting is to also consider multioutput Gaussian processes where the outputs are correlated with each other. This would mean that for a signal it is assumed that separate frequency components are dependent on each other. Various kernels for modeling channel dependencies exist. For example, the Linear Model of Corregoization defined in [Goo97]. This kernel can be applicable to different combinations of the latent Gaussian process kernels. In the spectral mixture setting, there are also kernels that consider dependencies. One example is the multi-output spectral mixture kernel defined in [PT17].

Another potential direction is to see whether training in signal domain can improve the predictions. In the model 6 Gaussian processes are trained independently of each other. Each process predicts its own frequency component. The model doesn't know that later these frequency components will be combined together in the IRFFT to predict samples in the signal domain. One could consider defining the loss function in the signal domain. This loss function will compare the predicted high-fidelity samples against the ground truth. This loss will couple together different Gaussian processes used in the model. Also, it will incorporate the IRFFT transformation. The hope is that if the loss function knows the final goal, predicting the signal samples, it can select better parameters for all the kernels. Potential drawbacks of the scheme are that the loss function might become more complicated to implement. Additionally, such a scheme would optimize all Gaussian processes at the same time, which might be more difficult in terms of computational resources.

In the NARGP setting, one can consider incorporating the BNSE initialization together with the SM kernels into the kernels with the NARGP structure. The current implementation of the BNSE initialization in the [Tob] library allows the direct use of the BNSE only with objects of the type Spectral Mixture model, a pure spectral mixture kernel. NARGP will involve combining multiple spectral mixture kernels together. The hope is that introducing the BNSE initialization might improve predictions of the NARGP kernel with the spectral mixture components.

Also, in the project, the test functions did not have any noise. On the contrary, real-world processes often are corrupted by noise signals. One problem with the Fourier transform mentioned in [Tob18] is that noisy or missing data will introduce uncertainty over the entire frequency domain. The authors suggest using the BNSE to obtain a probabilistic model of signal's frequencies.

List of Figures

List of Tables

Bibliography

[AAS19] P. A. Alvarado, M. A. Alvarez, and D. Stowell. "Sparse Gaussian process audio source separation using spectrum priors in the time-domain." In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 995–999.

[Bis06] C. M. Bishop. "Pattern recognition and machine learning." In: *Springer google schola* 2 (2006), pp. 1122–1128.

[Boc+59] S. Bochner et al. *Lectures on Fourier integrals*. Vol. 42. Princeton University Press, 1959.

[Cor+22] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*. MIT press, 2022.

[DL13] A. Damianou and N. D. Lawrence. "Deep gaussian processes." In: *Artificial intelligence and statistics*. PMLR. 2013, pp. 207–215.

[Fre11] P. D. Freeman. *Lecture notes in Signals and Systems*. `https://ocw.mit.edu/courses/6-003-signals-and-systems-fall-2011/resources/lecture-21-sampling/`. 2011.

[Gar23] R. Garnett. *Bayesian optimization*. Cambridge University Press, 2023.

[Goo97] P. Goovaerts. *Geostatistics for natural resources evaluation*. Vol. 483. Oxford University Press, 1997.

[Gün22] P. D. S. Günneman. *Lecture notes in Machine Learning for Graphs and Sequential Data*. 2022.

[HRS16] M. Hardt, B. Recht, and Y. Singer. "Train faster, generalize better: Stability of stochastic gradient descent." In: *International conference on machine learning*. PMLR. 2016, pp. 1225–1234.

[Jer77] A. J. Jerri. "The Shannon sampling theorem—Its various extensions and applications: A tutorial review." In: *Proceedings of the IEEE* 65.11 (1977), pp. 1565–1596.

[KO00] M. C. Kennedy and A. O'Hagan. "Predicting the output from a complex computer code when fast approximations are available." In: *Biometrika* 87.1 (2000), pp. 1–13.

[Li+20] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar. "Fourier neural operator for parametric partial differential equations." In: *arXiv preprint arXiv:2010.08895* (2020).

[Lia23] A. Liatsetskaya. "Bayesian Optimisation for the design of experiments." In: (2023).

[Mur12] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[Osg] P. B. Osgood. *Lecture notes for the Fourier Transform and its applications*. `https://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf`.

[Per+17] P. Perdikaris, M. Raissi, A. Damianou, N. D. Lawrence, and G. E. Karniadakis. "Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling." In: *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* 473.2198 (2017), p. 20160751.

[Pol+22] M. Poli, S. Massaroli, F. Berto, J. Park, T. Dao, C. Ré, and S. Ermon. "Transform once: Efficient operator learning in frequency domain." In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 7947–7959.

[PT17] G. Parra and F. Tobar. "Spectral mixture kernels for multi-output Gaussian processes." In: *Advances in Neural Information Processing Systems* 30 (2017).

[Ras03] C. E. Rasmussen. "Gaussian processes in machine learning." In: *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[Rav+24] K. Ravi, V. Fediukov, F. Dietrich, T. Neckel, F. Buse, M. G. Bergmann, and H.-J. Bungartz. "Multi-fidelity Gaussian process surrogate modeling for regression problems in physics." In: *Machine Learning: Science and Technology* (2024).

[RSA15] O. Rippel, J. Snoek, and R. P. Adams. "Spectral representations for convolutional neural networks." In: *Advances in neural information processing systems* 28 (2015).

[SB14] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[Tob] F. Tobar. *Mogptk library*. `https://games-uchile.github.io/mogptk/examples.html?q=03_Parameter_Initialization`.

[Tob18] F. Tobar. "Bayesian nonparametric spectral estimation." In: *Advances in Neural Information Processing Systems* 31 (2018).

[Wil14] A. G. Wilson. "Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes." PhD thesis. University of Cambridge Cambridge, UK, 2014.

[WR06] C. K. Williams and C. E. Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT press Cambridge, MA, 2006.