DELFT UNIVERSITY OF TECHNOLOGY

MASTER THESIS

Sparse Gaussian Processes in the Longstaff-Schwartz algorithm

Author:
Frederiek WESEL

Supervisor:
Dr. Pasquale CIRILLO

Committee:
Dr. Pasquale CIRILLO
Prof. Dr. Ir. C. W. OOSTERLEE
Dr. Peter DEN ISEGER

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

November 19, 2019

TU Delft

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

EEMCS

Master of Science

Sparse Gaussian Processes in the Longstaff-Schwartz algorithm

by Frederiek WESEL

In financial applications it is often necessary to determine conditional expectations in Monte Carlo type of simulations. The industry standard at the moment relies on linear regression, which is characterised by the inconvenient problem of having to choose the type and number of basis functions used to build the model, task which is made harder by the frequent impossibility to use an alternative numerical method to evaluate the "ground truth". In this thesis Gaussian Process Regression is investigated as potential substitute for linear regression, as it is a flexible Bayesian non-parametric regression model, which requires little tuning to be used. Its downfall is the computational complexity related to its "training" phase, namely $\mathcal{O}\left(N^3\right)$ operations, requiring the use of algorithmic approximations. The most prominent approximations are reviewed and tested in different scenarios requiring the approximation of conditional expectation by regression, among which the Longstaff-Schwartz algorithm for the pricing of Bermudan options. This thesis was carried out in cooperation with ABN-AMRO bank.

Acknowledgements

I would like first of all to thank my supervisor from the TU-Delft side Dr. Pasquale Cirillo for his availability, support, excellent feedback and mentorship, as well as Dr. Peter den Iseger from the ABN-AMRO side, for the many useful discussions, tips and overall practical and theoretical related questions related to the subject of this thesis: I hope this work may somehow result in some interesting applications.

Besides them, I would like to thank Prof. Dr. Ir. Kees Oosterlee for generously offering his time, support, guidance and good will throughout the preparation and review of this document.

Special thanks goes to my former colleague Sebastiaan Jong for the technical support and help when dealing with the ABN-AMRO (Azure) environment.

Gratitude go as well to my colleague Klest Dedja for the very interesting and frequent discussions regarding our theseses.

I would like of course to thank ABN-AMRO for the opportunity offered me by means of this thesis project.

Last but certainly not least, thanks to my family and friends, for the support and the good times we had together in these last months.

Contents

Chapter 1

Introduction

Machine Learning algorithms, which are algorithms that build a mathematical model based on data in order to make predictions or decisions [27], constitute a rapidly growing research field, which is however certainly not new. Most methods originate somewhere in the beginning of the second half of the last century (at the moment of writing [1]), and where probably not conceived to be used on the scale in which they are now. "Old" methods such as Support Vector Machines, Gaussian Process Regression with other kernel methods have become feasible in the last twenty years due to the rise in computational efficiency gifted us by Moore's Law, the larger quantity of data being generated and stored, and due to the poor explainability of Neural Networks.

These statistical learning techniques are seeing more applications in different types of fields, fields which have different needs, requirements and possibly regulations, which have in the past been dominated by simpler parametric models. One of such field is the financial one. It has different players, such as banks, which can be described as rather conservative and regulated entities. The use of Machine Learning algorithms within banks is in fact often heavily restricted by internal and external directives, which advocate the use of explainable, robust models. A good example of external regulation is given by the European Union directives regarding guidelines for trustworthy Artificial Intelligence (superset of Machine Learning) [64], which allow customers to demand knowledge and explanation regarding how (automatic) decisions regarding them are taken.

In this scenario, the field of finance has seen relative little application of Machine Learning techniques, leaving many possibilities for their use open to explainable and robust methods. In this work, the use of one such algorithm is investigated for application in the field of option pricing, but not exclusively, as many encountered problems can be considered rather common. Here below a short introduction regarding the application and the algorithm are given, as well as an outline of the rest of this work.

1.1 The Longstaff-Schwartz algorithm

In financial applications it is often necessary to approximate conditional expectations by means of data, for instance in order to compute relevant risk management indicator variables or price complex options. This can happen both in case when the data is collected, e.g. historical transaction data, or in Monte Carlo type of algorithms. Perhaps the case which exemplifies best the latter class is given by the Longstaff-Schwartz method (LSM), a backwards dynamic programming algorithm which is used as industry standard for the pricing of Bermudan (and American) options.

Its strength, compared to more traditional methods based on finite differences, is its applicability to path-dependent and highly dimensional options: in case of the former it is in fact hard to derive the associated deterministic numerical problem, while in the latter, the amount of memory and operations required by any finite difference methods scales exponentially with the dimensionality of the problem, as one incurs in the so called "curse of dimensionality". The Longstaff-Schwartz algorithm partially sidesteps these issues by resorting to stochastic simulation, i.e. the Monte Carlo approach: the crux of this particular

[1] There is in fact the possibility that this thesis might be in fact discovered by some alien form of life (or worse rediscovered by humans) in the future, so I advise you to keep the physical (and virtual) copy of this work in a very good nuclear and termite proof shelter

method is the determination of the so called continuation value, which is nothing but, given current knowledge, the estimate of the exercise value of the option at the a future time, i.e. the conditional expectation. In current practice, the continuation value is estimated by linear regression, just as the inventors Longstaff and Schwartz did in the seminal paper of the method which bears their names [12].

One of the biggest problems related to this approach is the construction of the linear model, namely the choice of type and number of basis functions, the selection of paths on which to regress, and the choice of covariates, which are all issues related to parametric models. It is in fact particularly hard to engineer meaningful and informative variables to use as regressors since by definition all of the simulated variables are related to the continuation value, however some are more than others.

Furthermore, while for simple options it is possible to determine the option price by solving numerically the associated numerical problem, and to thus tune the linear model in the Longstaff-Schwartz algorithm by minimizing some kind of error metric, it often not possible to do so for complex options, since as we said, the amount of operations and memory required to determine the exact numerical solution increases exponentially as a function of the option underlyings and model parameters.

Another disadvantage regarding the use of parametric models in the Longstaff-Schwartz algorithm is the fact that the regression problem changes, more or less, at every iteration depending on the stochastic model used in the simulation, the type of option, and the option parameters. One would therefore expect better performance when it comes to option pricing by a more flexible method, not limited by some preselected functional form.

All these practical issues demand a different approach to the current parametric one: in order address these problems and greatly simplify this model selection task, this work aims to investigate the effects of approximating the conditional expectation in the Longstaff-Schwartz algorithm by performing the regression by means of a non-parametric regression model, not limited by a predetermined functional form. An excellent candidate is Gaussian Process Regression.

1.2 Gaussian Process Regression

Gaussian Process Regression (GPR) is an explainable Bayesian non-parametric regression technique, which as the name suggests, models the data as noisy realizations of some underlying process which follows a Gaussian distribution. It is considered explainable as the relations between the response variables are clear, as they are assumed to follow a certain distribution, that is in practice characterized (in most cases) only by the prior covariance, which is modelled using a covariance function, known also as kernel function. It can be seen as a Bayesian method, since inference is carried out by using the mean and variance of some posterior distribution. It is finally non-parametric since one is modelling an underlying function everywhere, not just on the available noisy realizations, allowing model complexity to grow as function of the data.

GPR has a long history in geostatistics, therein known as "Kriging", where the regression problem is naturally described on spatial coordinates. Application to more general "Machine Learning" areas are more recent, and made possible due to the steady grow in computational power and general larger availability of data, as well as thanks to the important works of a handful of researchers, in particular by [9], [20], [21] and [31], which have made this approach well known to a broader public. In particular GPR caught the interest of many in the nineties, e.g. because of its astounding connection to neural networks [40], which were at the time catching interest again. Applications to the financial field are way more recent and rare, see for instance [53] and [58].

In general unfortunately, GPR has been always used on relatively small sets of data due to its main drawback, which is the computational complexity associated with "training" the model. The computational complexity related to "training" does in fact amount traditionally to $\mathcal{O}\left(N^3\right)$, N being the number of data points in the training set. This limitation becomes nowadays burdening for $N > 10^5$ [61], also because before the actual "training" can take place, it is common procedure to tune the model hyperparameters, usually by empirical Bayes, requiring again $\mathcal{O}\left(N^3\right)$ operations per optimization step. While this procedure marks

the depart from the pure Bayesian formalism, it allows easy model selection [31], while still retaining the flexibility of modelling with Gaussian Processes.

In order to cope with these computational disadvantages, sparse approximations to the full Gaussian Process Regression models have been developed since the very beginning. In general these approximations are based on a selection of a set of informative support points, which are able to best describe the entire set of data. More specifically, depending on the considered method the support points can be actual points out of the dataset, or as we will see, interpolation points of the kernel function. These approximations coupled with different techniques to solve the predictive equations allow nowadays the use of this powerful regression technique in a plethora of conditions. In this thesis, the various existing (state-of-the-art) approximations are therefore investigated in light of their application to the Longstaff-Schwartz algorithm, and as a consequence to other (financial) scenarios.

1.3 Outline of the thesis

In order to overcome the current limitations of the Longstaff-Schwartz algorithm, Gaussian Process Regression and its approximations need to be tested in plausible scenarios in which an analytical or numerical "ground truth" solution is available.

Before doing this, this work sheds light on the functioning of Gaussian Process Regression in Chapter 2, by presenting different derivations of its predictive equations which carry different underlying ideas. This is done by presenting a brief overview of kernels (Section 2.1), machine learning basics (Section 2.2, Gaussian Processes and finally Gaussian Process Regression (Section 2.3), coupled with an overview of the most used kernels (Section 2.4.

Next, different relevant (state-of-the-art) sparse approximations are reviewed and discussed in Chapter 3, keeping in mind the application within the Longstaff-Schwartz algorithm.

After offering a concise review in Chapter 4 for the not familiar reader of the fundamental concepts behind option pricing (Section 4.1) and the Monte Carlo methods (Section 4.2), the Longstaff-Schwartz algorithm is presented, its disadvantages compared to different pricing approaches are reviewed, and previous work regarding the application of Gaussian Process Regression to the Longstaff-Schwartz algorithm is presented (Section 4.3).

To verify in practice the performance when applied to the Longstaff-Schwartz algorithm, in Chapter 5 the examined methods and approximations are then tested on a variety of sets of data related or not to the problem of the estimation of the continuation value (Sections 5.1, 5.2, 5.3). At last, the various methods are then applied to the Longstaff-Schwartz algorithm for the pricing of actual Bermudan options, and their performance is compared to the one of the traditional, linear regression based, Longstaff-Schwartz method (Section 5.4).

Chapter 2

Introduction to Gaussian processes

The Gaussian Process (GP) is a simple class of probability distribution over functions. In this setting, GPs have been studied for quite a while, but they have been used for predictive purposes only since quite recently.

Making predictions regarding a continuum quantity (regression) using Gaussian Processes is known as Gaussian Process Regression (GPR), sometimes referred to as Kriging or Wiener–Kolmogorov prediction. GPR is a non-parametric Bayesian Machine Learning algorithm [20], which originated in the field of geology based on the work of Krige and Matheron [3], where the process is naturally defined on spatial coordinates. As stated in the Introduction, application to more general "Machine Learning" areas are more recent, and made possible due to the steady grow in computational power and general larger availability of data, as well as thanks to the important works of a handful of researchers, in particular by [9], [20], [21] and [31], which have made this approach well known to a broader public. Applications to the financial field are way more recent, see for instance [53] and [58]. The basic assumption behind regression is that the given realizations of the test variable are an instance of a Gaussian Process. Combined with a noise model, it is possible to determine an analytical predictive distribution.

To better understand what Gaussian Process Regression is, some concepts need to be introduced. At first, we will examine kernels following the approach in [15]: as we will see their choice greatly determines the type of predictive distribution in the GP setting. Some important results such as Mercer's theorem are needed to better explain low-rank approximation of the full GP, which will be examined in the next chapter. Furthermore we will touch on some machine learning basics, in particular the empirical risk minimization framework and regularization. Finally Gaussian Process Regression will be discussed.

2.1 Kernels

A **kernel function** is a real function of two arguments, $k(x, x')$. Typically the function is symmetric (i.e. $k(x, x') = k(x', x)$), and non-negative (i.e. $k(x, x') \leq 0$) and can thus be interpreted as a similarity measure, but that does not have to be the case necessarily.

Positive definite kernels

Definition 2.1.1 (Gram matrix)**.** Given a function $k : \mathcal{X}^2 \to \mathbb{K}$, where $\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{R}$, $\mathcal{X} = \{x_1, ..., x_n\}$ and points $x_1, ..., x_n \in \mathcal{X}$, the matrix defined as:

$$K = \begin{bmatrix} k(x_1, x_N) & \cdots & k(x_1, x_N) \\ & \vdots & \\ k(x_N, x_1) & \cdots & k(x_N, x_N) \end{bmatrix}, \tag{2.1}$$

is called the Gram matrix or kernel matrix of k with respect to $x_1, ...x_n$.

Definition 2.1.2 (Positive (semi)-definite matrix)**.** A complex $n \times n$ matrix K satisfying:

$$\sum_{i,j} c_i \bar{c}_j K_{i,j} \geq 0, \tag{2.2}$$

for all $c_i \in \mathbb{C}$ is called positive semi-definite. Similarly, a real symmetric $n \times n$ matrix is called positive definite if it satisfies Equation 2.2.

Definition 2.1.3 (Positive (semi-)definite kernel)**.** Let $\mathcal{X}$ be a nonempty set. A function k on $\mathcal{X} \times \mathcal{X}$ which for all $n \in \mathbb{N}$ and all $x_1, ..., x_n \in \mathcal{X}$ gives rise to a positive definite Gram matrix is called a positive (semi-)definite kernel.

Definition 2.1.3 and positive semi-definite matrices differ as a positive semi-definite kernel will induce a positive semi-definite matrix for any choice of points. Positive semi-definitness 2.1.2 implies directly non-negativity on the diagonal:

$$k(x, x) \geq 0 \, \forall \, x \in \mathcal{X}, \tag{2.3}$$

and symmetry:

$$k(x, x') = \overline{k(x'x)}. \tag{2.4}$$

Another important property which holds for inner products and kernels is the **Cauchy-Schwarz inequality**:

Proposition 2.1.1 (Cauchy-Schwarz inequality for kernels)**.** If k is a positive definite kernel and $x_1, x_2 \in \mathcal{X}$ then:

$$|k(x_1, x_2)|^2 \leq k(x_1, x_1) \, k(x_2, x_2). \tag{2.5}$$

Remark 2.1.1 (Notation)**.** The brackets in the Definition 2.1.3 indicate that formally the kernel should be called positive semi-definite, however in practice that "semi" is dropped. To avoid confusion we shall simply refer to positive semi-definite kernels as kernels.

Kernels can be regarded as a generalization of dot products, in the sense that they can be defined, as we will see, as the dot product of points transformed into **feature space**. Some properties such as linearity do however not hold. To arrive at this important result, we introduce the concept of **feature map**.

The reproducing kernel map

Definition 2.1.4 (Feature map)**.** Assume that k is a real-valued positive definite kernel, and $\mathcal{X}$ a nonempty set. We define a feature map from $\mathcal{X}$ into the space of function mapping $\mathcal{X}$ into $\mathbb{R}$ denoted as $\mathbb{R}^{\mathcal{X}} := \{f : \mathcal{X} \to \mathbb{R}\}$ via:

$$\Phi : \mathcal{X} \to \mathbf{R}^{\mathcal{X}}; \quad x \to k(\cdot, x). \tag{2.6}$$

Here $\Phi(x)$ denotes the function that assigns the value $k(x', x)$ to x'. Using a feature map we have thus turned each data point into a function on its domain $\mathcal{X}$, which can be interpreted as a similarity function to all the points in $\mathcal{X}$. As we will see it is possible to define a feature space associated with feature map Φ by turning the image of Φ into a inner product space (also known as pre-Hilbert space) and finally showing that $k(x, x') = \langle \Phi(x), \Phi(X') \rangle$.

Definition 2.1.5 (Inner product space)**.** An inner product space is a vector space $\mathcal{H}$ over a field of scalars $\mathbb{K}$ where $\mathbb{K} = \mathbb{C}$ or $\mathbb{K} = \mathbb{R}$ endowed with an inner product $\langle \cdot, \cdot \rangle$ which satisfies the following properties (symmetric bilinear form) for all the vectors x, y, z and scalars $\alpha \in \mathbb{K}$:

1. *Conjugate symmetry*:

$$\langle x, y \rangle = \overline{\langle y, x \rangle}. \tag{2.7}$$

2. *Linearity in the first argument*:

$$\langle \alpha x, y \rangle = \alpha \langle x, y \rangle, \tag{2.8}$$
$$\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle. \tag{2.9}$$

3. *Positive-definiteness*:

$$\langle x, x \rangle > 0 \quad \iff x \in \mathcal{H} \setminus \{0\}. \tag{2.10}$$

We begin by constructing a dot product space containing the images of the input data under Φ. To this end we first need to define a vector space, by taking a linear combination of the form:

$$f(\cdot) = \sum_{i=1}^{n} \alpha_i k(\cdot, x_i), \tag{2.11}$$

where $n \in \mathbb{N}$ and $x_1, ..., x_n \in \mathcal{X}$. Next we define another function

$$g(\cdot) = \sum_{j=1}^{n} \beta_j k\left(\cdot, x'_j\right), \tag{2.12}$$

where $n \in \mathbb{N}$ and $x_1, ..., x_n \in \mathcal{X}$, and a mapping between them as:

$$\langle f, g \rangle := \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \beta_j k\left(x_i, x'_j\right), \tag{2.13}$$

which is well defined as:

$$\langle f, g \rangle = \sum_{j=1}^{n} \beta_j f\left(x'_j\right), \tag{2.14}$$

using $k\left(x'_j, x_i\right) = k\left(x_i, x'_j\right)$. Similarly:

$$\langle f, g \rangle = \sum_{i=1}^{n} \alpha_i g(x_i). \tag{2.15}$$

Hence the mapping is bilinear, and also symmetric, as $\langle f, g \rangle = \langle g, f \rangle$. Moreover by Definition 2.1.2 it is also positive-definite, i.e.:

$$\langle f, f \rangle = \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0 \tag{2.16}$$

Hence $\langle \cdot, \cdot \rangle$ is itself a positive definite kernel. Note in fact that given functions $f_1, ..., f_n$ and coefficients $\gamma_1, ...\gamma_n \in \mathbb{R}$, we have:

$$\sum_{i,j=1}^{n} \gamma_i \gamma_j \langle f_i, f_j \rangle = \left\langle \sum_{i=1}^{n} \gamma_i f_i, \sum_{j=1}^{n} \gamma_j f_j \right\rangle \geq 0, \tag{2.17}$$

where the equality follows from the bilinearity of $\langle \cdot, \cdot \rangle$, and the inequality from equation 2.16. Hence we have that $\langle \cdot, \cdot \rangle$ is itself a symmetric-positive-definite kernel, defined on the function space. We have also shown it is bilinear, hence it were positive-definite it would be an inner product space, or pre-Hilbert space.

Note that by Equations 2.11 and 2.13:

$$\left\langle k(\cdot, x), k\left(\cdot, x'\right) \right\rangle = f(x). \tag{2.18}$$

k is here the **representer of evaluation**. By Equation 2.23 and Proposition 2.1.1 it follows that

$$|f(x)|^2 = |\langle k(\cdot, x), f \rangle|^2 \leq k(x, x) \langle f, f \rangle. \tag{2.19}$$

Hence $\langle f, f \rangle = 0$ directly implies $f = 0$, which shows that $\langle \cdot, \cdot \rangle$ is positive definite and thus defines an inner product space. Furthermore from Equation 2.23 we have:

$$\left\langle k(\cdot, x), k\left(\cdot, x'\right) \right\rangle = k\left(x, x'\right). \tag{2.20}$$

By these properties, positive-definite kernels k are also called **reproducing kernels**. In conclusion, the above has shown that any positive definite kernel can be seen as a dot product in another space, s.t.:

$$\left\langle \Phi(x), \Phi\left(x'\right) \right\rangle = k\left(x, x'\right). \tag{2.21}$$

Hence the inner product space $\mathcal{H}$ constructed is one possible instance of the feature space associated with a kernel.

So far we have seen that a feature map can be constructed from a kernel. Also the opposite can be done: when we have a mapping Φ from $\mathcal{X}$ into an inner product space $\mathcal{H}$, we obtain a positive definite kernel by $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$. This can be seen as we have for all $c_i \in \mathbb{R}, x_i \in \mathcal{X}, i = 1, ..., n$:

$$\sum_{i,j}^{n} k(x_i, x_j) = \left\langle \sum_{i=1}^{n} c_i \Phi(x_i), \sum_{j=1}^{n} c_j \Phi(x_j) \right\rangle = \left\| \sum_{i=1}^{n} c_i \Phi(x_i) \right\|^2 \geq 0. \tag{2.22}$$

This allows us to give an equivalent definition of kernel as a function with the property that there exists a map Φ into a dot product space such that $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ holds. It also allows construction of kernels from feature maps.

Reproducing kernel Hilbert spaces

In the last subsection, a description on how to define a space of functions which is a valid realization of the feature space associated with a given kernel was provided. The space is a vector space, and is endowed with a dot product. Such space is known more generally as pre-Hilbert space, as it can be turned into a Hilbert space by showing that it is complete. In case of our pre-Hilbert space of functions (Equation 2.11) endowed with inner product (Equation2.13), it can be completed by adding the elements to which every Cauchy sequence converges with respect to the norm corresponding to the dot product, i.e. $||f|| = \sqrt{\langle f, f \rangle}$. A thus constructed space is usually called by properties 2.23 and 2.24 a **reproducing kernel Hilbert space**, and is defined as follows:

Definition 2.1.6. Let $\mathcal{X}$ be a non-empty set (also called the index set) and $\mathcal{H}$ be a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with the dot product $\langle \cdot, \cdot \rangle$ (and the norm $||f|| = \sqrt{\langle f, f \rangle}$) if there exists a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties:

1. k has the reproducing property

$$\langle k(\cdot, x), k(\cdot, x') \rangle = f(x), \tag{2.23}$$

in particular:

$$\langle k(\cdot, x), k(\cdot, x') \rangle = k(x, x'). \tag{2.24}$$

2. k spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{\text{span}\{k(x, \cdot) \,|\, x \in \mathcal{X}\}}$ where $\bar{X}$ denotes the completion of set X.

The symmetry of k follows directly from Equation 2.24, as well as its positive-definiteness as $\mathcal{H}$ is a Hilbert space. k is furthermore unique: suppose the existence of two kernels k and k' spanning the same reproducing kernel Hilbert space. Using the symmetry and the representation properties (Equation 2.23) and (Equation 2.24) we have that:

$$\langle k(\cdot, x), k'(\cdot, x') \rangle = k(x, x') = k(x, x') = k'(x', x) \tag{2.25}$$

where the second equality uses the symmetry of the dot product. Symmetry of k yields $k(x, x') = k'(x, x')$.

Mercel kernel map

We have seen how any positive-definite kernel admits a dot product representation in a linear space. This was done by explicitly constructing an appropriate Hilbert space. We will now consider another Hilbert space, constructed using Mercer's theorem. Herein the feature map Φ leads to a different target space, however the distinction is not really important, as we are interested in the existence of some Hilbert space in which the kernel corresponds to the dot product.

Theorem 2.1.1 (Mercer's theorem [1]). Let $(\mathcal{X}, \mu)$ be a finite measure space, and let $k \in L_\infty(\mathcal{X}^\in)$ be a symmetric real-valued function such that the integral operator $T_k : L_2(\mathcal{X}) \to L_2(\mathcal{X})$

$$(T_k f)(x) := \int_{\mathcal{X}} k(x, x') f(x') \, d\mu(x') \tag{2.26}$$

is positive definite, i.e. we have for all $f \in L_2(\mathcal{X})$ that

$$\int_{\mathcal{X}\times\mathcal{X}} k(x, x') f(x) f(x') \, d\mu(x) \, d\mu(x') \geq 0. \tag{2.27}$$

Let $\psi_j \in L_2(\mathcal{X})$ be the normalized orthogonal eigenfunctions of T_k associated with the eigenvalues $\lambda_j > 0$, sorted in non-decreasing order. Then:

1. $(\lambda_j)_j \in l_1$,
2. $k(x, x') = \sum_{j=1}^{n_\mathcal{H}} \lambda_j \psi_j(x) \psi_j(x')$ holds for almost all (x, x'). Either $n_\mathcal{H} \in \mathbb{N}$ or $n_\mathcal{H} = \infty$. In the latter case the series converges for almost all (x, x').

It follows from Equation 2 that $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ with:

$$\Phi : \mathcal{X} \to l_2^{n_\mathcal{H}}, \quad x \to \left(\sqrt{\lambda_j} \phi_j(x) \right)_{j=1,\ldots,n_\mathcal{H}}, \tag{2.28}$$

for almost all $x \in \mathcal{X}$, where Φ is the feature map under consideration, which is a different target space than the one offered by the reproducible kernel Hilbert space map in Equation 2.6:

Proposition 2.1.2 (Mercer kernel map). If k is a kernel satisfying the conditions of Theorem 2.1.1, then we can construct a mapping Φ into a space where k is a dot product, i.e.:

$$\langle \Phi(x), \Phi(x') \rangle = k(x, x'), \tag{2.29}$$

for almost all $x, x' \in \mathcal{X}$. Furthermore, given any $\epsilon > 0$, there exists a map Φ_n into an n-dimensional dot product space, where $n \in \mathbb{N}$. such that:

$$\left| k(x, x') - \langle \Phi_n(x), \Phi_n(x') \rangle \right| < \epsilon, \tag{2.30}$$

for almost all $x, x' \in \mathcal{X}$.

Mercer's kernel map (Proposition 2.1.2) as well as the reproducing kernel Hilbert space map (Equation 2.6) allow us to avoid the explicit mapping Φ that is needed to learn nonlinear functions by means of dot product algorithm by simply substituting the call to $\langle \Phi(\cdot), \Phi(\cdot) \rangle$ with one to $k(\cdot, \cdot)$. This is known as **kernel trick**, the utility of which will be demonstrated in the next subsection by means of an example. In practice the choice of kernel should be approached with care, as in some algorithms some assumptions regarding the input data are made which could be violated by a particular kernel choice. Furthermore depending on the choice of kernel, the data might be more or less separable. The most well known algorithms which make use of kernels are kernel perceptron, support vector machine (SVM), Gaussian Process Regression, principal components analysis (PCA), canonical correlation analysis, ridge regression, spectral clustering, linear adaptive filters and others [15].

2.2 Machine learning basics

The machine learning field can be divided in three different sub-fields: **supervised learning**, **unsupervised learning** and **reinforcement learning** [39]. The most common of all is the supervised learning setting, in which a data set consisting of a series of inputs with their associated responses is available. The goal is usually either **classification** or **regression**. While the former seeks to find the relationship between a given input and a finite number of classes, regression can be seen as finding the relationship between the inputs and a infinite number of classes [39], i.e. a continuum. We now define a framework in which these task, generally to be called supervised learning, can be performed.

The statistical learning framework

The general framework in which statistical learning is performed is composed as follows [48]:

1. **Domain set** $\mathcal{X}$ in which the data is located, which can corresponds to the data we may wish to label.
2. **Label set** $\mathcal{Y}$. In case of simple multi-class classification with c classes $\mathcal{Y} = \{0, 1, ..., c\}$, while in case of regression, as we will see, $\mathcal{Y}$ is a continuum (usually $\mathcal{Y} = \mathbb{R}$).
3. **Training set** composed of n tuples $\mathcal{X} \times \mathcal{Y}$ of elements drawn jointly from $\mathcal{X}$ and $\mathcal{Y}$ form the so called **training data**, or **training set**, which we denote as $\mathcal{S} = \{(x_1, y_1), ..., (x_N, y_N)\}$.
4. **Test set** $\mathcal{T}$ is composed of elements $\mathcal{X} \times \mathcal{Y}$ drawn jointly from $\mathcal{X}$, and is used for validation purposes. In practice one has in general one set of data to work with, having as a result the necessity to either split the data or to use other validation techniques such as k-fold validation.
5. **Learner's output** which is a **prediction rule**, also called **predictor**, **hypothesis** or **classifier** $h : \mathcal{X} \to \mathcal{Y}$. The predictor can be used for predicting the label of other instances not in the training set $\mathcal{S}$.
6. **Data generation model:** it is assumed that the data in the training set and label set are generated instances of an underlying distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$. The learner does not know anything about the underlying distribution.
7. **Measure of success:** in order to measure the quality of a predictor, a metric is needed, which is usually called **loss**, which will be discussed next.

Loss function and empirical risk minimization

Generally speaking one would want to be able to make an estimate as close to the true underlying class, i.e. an estimate which minimizes the loss. Let us begin by giving a definition of loss:

Definition 2.2.1 (Loss function)**.** Denote by $(x, y, f(x)) \in \mathcal{X} \times \mathcal{Y} \times \mathcal{Y}$ the triplet consisting of a data point x, a response y and a prediction $f(x)$. Then the map $\ell : \mathcal{X} \times \mathcal{Y} \times \mathcal{Y} \to [0, \infty)$ with the property $l(x, y, y) = 0$ for all $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ is called a loss function.

Note that the thus defined loss function is always non-negative ensuring that only a correct prediction achieves zero loss. The loss function can take different forms depending on the task which is to be performed, namely classification or regression. In case of 0-1 classification, where $\mathcal{Y} = \{-1, 1\}$ and $f : \mathcal{X} \to \mathcal{Y}$, a very simple loss function can be defined as follows:

$$\ell_{0-1}(x, y, f(x)) := |f(x) - y|. \tag{2.31}$$

which is usually known as 0-1 loss function or simply misclassification error. In case of regression, i.e. when estimating real-valued quantities, it is usually the magnitude of the difference $y - f(x)$ which is indicative of the amount of misprediction. In some contexts l is known, think for instance of mispredicting the value of a financial instrument. In general however, the loss function in case of regression will have the form

$$\ell(x, y, f(x)) := g(f(x) - y). \tag{2.32}$$

The most common choice is to minimize the sum of squares of the residuals $f(x) - y$. This corresponds to the assumption that there is additive normally distributed noise perturbing the observations y:

$$\ell_{\text{sq}}(x, y, f(x)) := (f(x) - y)^2. \tag{2.33}$$

Definition 2.2.2 (Risk function)**.** Given a loss function $\ell(\cdot, \cdot, \cdot)$, the risk function is defined as

$$\mathcal{L}_{\mathcal{D}}(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}}(\ell(x, y, h(x))). \tag{2.34}$$

In practice, the true risk is not known directly to the "learner", as it is defined over $\mathcal{D}$. However the learner can calculate the **training error**, or **empirical risk** on the available data $\mathcal{S}$.

Definition 2.2.3 (Empirical risk)**.** Given a loss function $l(\cdot,\cdot,\cdot)$ the risk empirical risk function is defined as:

$$\mathcal{L}_{\mathcal{S}}(f) := \frac{1}{n}\sum_{i=1}^{n} \ell(x_i, y_i, f(x_i)). \tag{2.35}$$

The **empirical risk minimization** principle argues that under the assumption that the data points in the training set $\mathcal{S}$ are i.i.d., the predictor f should be chosen such that:

$$f^* = \underset{f\in\mathcal{F}}{\operatorname{argmin}}\, \mathcal{L}_{\mathcal{S}}(f), \tag{2.36}$$

where $\mathcal{F}$ is here a (parametric) class of functions, called **hypothesis class**.

One important definition concerning hypothesis classes is the one of **probably approximately correct** (PAC) learnability:

Definition 2.2.4 (Agnostic PAC learnability for general loss functions)**.** An hypothesis class $\mathcal{F}$ is agnostic learnable with respect to a set $\mathcal{Z}$ and a loss function $\ell : \mathcal{F}\times\mathcal{Z} \to [0,\infty)$ if there exists a function $m_{\mathcal{F}} : (0,1)^2 \to \mathbb{N}$ and a learning algorithm with the following property: for every $\epsilon, \delta \in (0,1)$ and for every distribution $\mathcal{D}$ over $\mathcal{Z}$, when running the learning algorithm on $m \geq m_{\mathcal{F}}(\epsilon,\delta)$ i.i.d. examples generated by $\mathcal{D}$, the algorithm returns $f \in \mathcal{F}$, such that, with probability of at least $1-\delta$

$$\mathcal{L}_{\mathcal{D}}(f) \leq \min_{f'\in\mathcal{F}} \mathcal{L}_{\mathcal{D}}(h') + \epsilon, \tag{2.37}$$

where $\mathcal{L}_{\mathcal{D}}(h) := \mathbb{E}_{(x,y)\sim\mathcal{D}}(\ell(x,y,h(x)))$

The PAC framework allows the derivation on bounds of the sample complexity for many function classes $\mathcal{F}$. In particular, bounds regarding Gaussian Process Regression which are very interesting in theory (e.g. Theorem 4 in [42]) exists, In practice however, it is a well known fact that PAC bounds are very loose, hence their practical applicability is severely limited to real life situations, and more interesting in theory. We therefore conclude that estimates of the performance of a model can be obtained, as already seen, by estimating the true risk.

Regularization

Empirical risk minimization can lead to **overfitting**, meaning that the predictor f minimizes the empirical risk more than the true risk. This leads in turn to poor generalization, which is the inability of f to make correct predictions of out-of-sample (unseen) data. A common solution is to restrict the empirical risk minimization rule by performing **regularization**. The key idea in regularization is to restrict the class of possible minimizers $\mathcal{F}$ (with $f \in \mathcal{F}$) of the empirical risk functional $\mathcal{L}_{\mathcal{D}}(\cdot)$ such that $\mathcal{F}$ becomes a compact set [15].

In practice this is often done by adding a **regularizer function** to the risk functional, which is a function that penalizes the risk if the complexity of the predictor is too large. We then have that a regularization function is a mapping $R : \mathbb{R}^d \to \mathbb{R}$ and the regularizes empirical risk minimization rule outputs an hypothesis such that

$$f^* = \underset{f\in\mathcal{F}}{\operatorname{argmin}}\, \mathcal{L}_{\mathcal{S}}(f) + \lambda R(f), \tag{2.38}$$

$\lambda > 0$ being the regularization constant.

It can be shown that a regularizer acts as a stabilizer, meaning intuitively that a small change in input does not change the output much. When dealing with kernels, a very important result regarding regularization is the **representer theorem**, which states that the minimizer f^* of a regularized empirical risk function defined over a reproducing kernel Hilbert space can be represented as a finite linear combination of kernel products evaluated on the input points in the training set data. Formally:

Theorem 2.2.1 (Representer theorem [15]). Denote by $R : [0, \infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by $\mathcal{L} : \left(\mathcal{X} \times \mathbb{R}^2\right)^n \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f^* \in \mathcal{H}$, $\mathcal{H}$ being a reproducing kernel Hilbert space, of the regularized risk

$$\mathcal{L}\left(\left(x_1, y_1, f\left(x_1\right)\right), ..., \left(x_n, y_n, f\left(x_n\right)\right)\right) + R\left(||f||_{\mathcal{H}}\right), \tag{2.39}$$

admits a representation of the form:

$$f^*\left(x\right) = \sum_{i=1}^{n} \alpha_i k\left(x_i, x\right). \tag{2.40}$$

Monotonicity of R is required for the theorem to hold, but it does not guarantee a unique minimizer. A unique minimizer is guaranteed if both R and $\mathcal{L}$ are convex [15]. The importance of the representer theorem is that when dealing with a regularized empirical risk minimization problem in a reproducing kernel Hilbert space $\mathcal{H}$, it states that the solution lies in the span of the kernels centered on the training set $\mathcal{S}$.

2.3 Gaussian Processes and Gaussian Process Regression

Perhaps the most intuitive way to see a Gaussian process is to see it as a random Gaussian distribution over functions [31] [39]. Formally:

Definition 2.3.1 (Gaussian process). A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Another way to think about it is as random variables $f(x)$ indexed with respect to a continuous (vector) variable x. For instance consider the random variables $\mathcal{F} = \{f_i\}_i^N$ with associated index locations $\mathcal{X} = \{x_i\}_i^N$. Then f is a Gaussian process with distribution $p(f|X) = \mathcal{N}\left(m(x), k(x, x')\right)$ The Gaussian process $f(x)$ in question is then defined as:

$$f(x) \sim \mathcal{GP}\left(m(x), k(x, x')\right)\right), \tag{2.41}$$

with:

$$m(x) = \mathbb{E}[f(x)], \tag{2.42}$$

$$k(x, x') = \mathbb{E}[\left(f(x) - m(x)\right)\left(f(x)' - m(x)\right))]. \tag{2.43}$$

Remark 2.3.1 (Notation). It should be noted that a Gaussian process is a conditional probability model, i.e. it models $p(f|X)$ and not $p(f)$. In order to maintain a concise notation we therefore from this points onwards omit the conditioning, unless otherwise noted.

So far the Gaussian process has been defined and modelled with different covariance functions, which allow a great variety of functions to be modelled by varying a (small) number of hyperparameters. In order to perform regression it is sufficient to define a noise model which links the underlying function to the available observation. Predictions can then be made according to Bayes' rule.

There are different ways to interpret Gaussian Process Regression. Arguably, the most intuitive and direct is the so called **function-space view**, in which one can think of a Gaussian process as defining a distribution over functions and inference taking place directly in the related function-space.

Equivalently one can resort to the **weight-space view**, which emphasizes the similarities between Gaussian Process Regression and Bayesian linear regression.

The predictive equation can also be derived by means of the Representer Theorem 2.2.1, and arise as from the minimization of a regularized risk functional. We will call this the **regularized empirical risk minimization view**.

Regularized empirical risk minimization view

By means of Theorem 2.2.1 we know what type of solution we have to consider in case of minimizing the functional in Equation 2.39 in a reproducing kernel Hilbert space. We now consider the following functional, corresponding to the minimization of the regularized square loss of observations corrupted by Gaussian noise with variance σ_n^2:

$$\mathcal{L}(f) = \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \frac{1}{2}||f||_{\mathcal{H}}^2. \tag{2.44}$$

We are interested in finding the minimizer f^* of this functional, i.e. the function which minimizes the regularized empirical risk:

$$f^* = \underset{f\in\mathcal{F}}{\operatorname{argmin}} \frac{1}{2\sigma_n^2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \frac{1}{2}||f||_{\mathcal{H}}^2 \tag{2.45}$$

As both the loss function and the regularizer are convex, we have a unique minimizer. From Equation 2.40 and using the representation property $\langle k(\cdot, x_i), k(\cdot, x_j)\rangle = k(x_i, x_j)$ of the reproducing kernel Hilbert space we get:

$$\begin{aligned}\mathcal{L}(f^*) &= \frac{1}{2}\alpha^T K_{N,N}\alpha + \frac{1}{2\sigma_n^2}|y - K_{N,N}\alpha|^2 \\ &= \frac{1}{2}\alpha^T\left(K_{N,N} + \frac{1}{\sigma_n^2}K_{N,N}^2\right)\alpha - \frac{1}{\sigma_n^2}y^T K_{N,N}\alpha + \frac{1}{2\sigma_n^2}y^t y,\end{aligned} \tag{2.46}$$

where $K_{N,N}$ is the Gram matrix of $\mathcal{S}$. Differentiating with respect to α leads to the minimizer $\alpha^* = \left(K_{N,N} + \sigma_n^2 I\right)^{-1} y$, which gives us the predictive equation

$$f^*(x) = K_{*,N}\left(K_{N,N} + \sigma_n^2 I\right)^{-1} y, \tag{2.47}$$

where $K_{*,N}$ is the Gram matrix constructed with kernel k and the points in test set $\mathcal{T}$ and training set $\mathcal{S}$. We will see that Equation 2.47 does correspond to the predictive posterior mean equation in the context of the function-space view or weight space view, however it was derived without making any assumption regarding the distribution of the data. Unlike in the other formulation it does hence not provide us with an estimate of a credibility interval, which can be a major disadvantage in some situations.

Function-space view

As stated previously, the core assumption behind Gaussian process regression is that the the underlying (latent) process f is a Gaussian Process. Hence we place a joint prior on f and f_*, such that:

$$p(f, f_*) = \mathcal{N}(0, K) = \mathcal{N}\left(0, \begin{bmatrix} K_{N,N} & K_{N,*} \\ K_{*,N} & K_{*,*}\end{bmatrix}\right), \tag{2.48}$$

where again $K_{N,N}$, $K_{*,N}$ and $K_{*,*}$ are the Gram matrices constructed with kernel k and respectively the points in training set $\mathcal{S}$, points in test set $\mathcal{T}$ and training set, and test set. In order to link the underlying function f, which is possibly latent, to the available observation y a noise model is constructed. In case of the full Gaussian process regression framework, f is assumed to be polluted by Gaussian noise ϵ such that

$$y = f + \epsilon, \tag{2.49}$$

with $p(\epsilon) = \mathcal{N}(0, \sigma_n^2)$. This leads to the following likelihood:

$$p(y|f) = \mathcal{N}\left(f, \sigma_n^2 I\right). \tag{2.50}$$

Marginalizing over f using Equation A.2 in the appendix yields the marginal likelihood or evidence:

$$p(y) = \int p(y|f)p(f)df = \mathcal{N}\left(0, K_{N,N} + \sigma_n I\right). \tag{2.51}$$

In order to estimate the value of the underlying function f at locations X^*, the posterior distribution can be obtained by conditioning on the observations y by Bayes' rule:

$$p\left(f, f_*|y\right) = \frac{p\left(f, f_*\right) p\left(y|f\right)}{p\left(y\right)}. \tag{2.52}$$

The predictive distribution can now be obtained by marginalizing out the latent variable f:

$$p\left(f_*|y\right) = \int p\left(f, f_*|y\right) df = \frac{1}{p\left(y\right)} \int p\left(f, f_*\right) p\left(y|f\right) df. \tag{2.53}$$

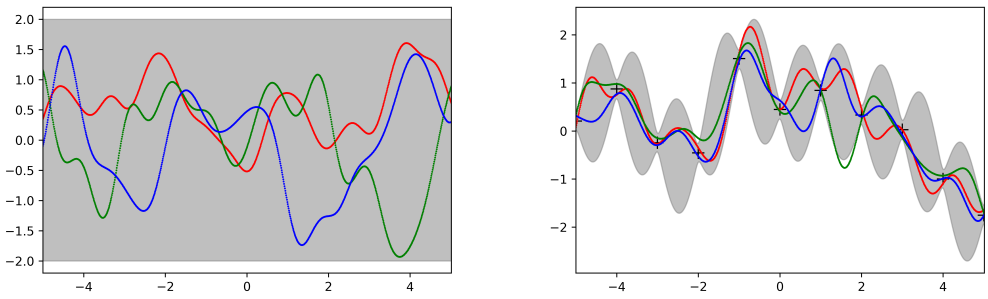
Remark 2.3.2 (Notation)**.** To maintain consistent notation with the following chapter, the value of f at query points X_* is denoted as f_*, while $K_{N,N}$, $K_{N,*}$, $K_{*,N}$ and $K_{*,*}$ are the Gram matrices defined on pairs of X and X_*.

Conditioning on X, Y and X^* using the identities in Equation A.2 and Equation A.3, the posterior $p\left(f_*|y\right) = \mathcal{N}\left(\mathbf{E}\left[f_*\right], \mathrm{cov}\left(f_*\right)\right)$ is obtained which can be used to make predictions:

$$\mathbf{E}[f_*] = K_{*,N}\left(K_{N,N} + \sigma_n^2 I\right)^{-1} y, \tag{2.54}$$

$$\mathrm{cov}(f_*) = K_{*,*} - K_{*,*}\left(K_{N,N} + \sigma_n^2 I\right)^{-1} K_{N,*}. \tag{2.55}$$

Clearly the posterior is again a Gaussian Process, with a given mean and covariance functions, which can be represented as weighted sums, i.e. $\mathbf{E}\left(f_*\right) = K_{*,N}\alpha$, with $\alpha = \left(K_{N,N} + \sigma_n^2 I\right)^{-1} y$. These weights can be precomputed before any prediction is made, guaranteeing a posterior mean estimate for one sample in $\mathcal{O}(N)$ operations, as it amounts to a simple dot product. Interestingly one can see that the covariance matrix of the the posterior does not depend on the observations y, meaning that posterior covariance estimates take $\mathcal{O}(N^2)$ operations per sample. Furthermore, the measurement noise ensures that the matrix is not singular, and thus (at least in theory) always invertible. Compared to other algorithms, Gaussian Process Regression is easy to understand, as the choice of prior mean and covariance functions is made in advance. Furthermore, as it is a probabilistic model, the generated posterior distribution can be used to generate credible confidence bounds, as shown in Figure 2.1.



(A) Prior. (B) Posterior.

FIGURE 2.1: In the figure on the left three draws from a specified GP prior are shown. On the right, three functions are drawn from the same GP and conditioned on a set of points. In both cases the grey shaded area corresponds to the 95% confidence interval.

Since the $K_{N,N} + \sigma_n^2 I$ matrix needs to be inverted in order to compute the weights, the computational complexity during what can be considered "training" is of order $\mathcal{O}(N^3)$, where N is the number of datapoints in the training set. The number of operations necessary during training is the main drawback of this technique. In practice, instead of taking

the inverse it is good practice to work with the Cholesky decomposition, whose complexity amounts to $\mathcal{O}\left(\frac{1}{3}N^3\right)$ operations, is in general more stable than a matrix inversion and provides the determinant of the full Gram matrix for free.

Weight-space view

Suppose that instead of placing a Gaussian prior over our latent function f, we define it to be the weighted sum of a set of M basis functions ϕ_i such that

$$f(x) = \sum_{i=1}^{i=M} w_i \phi_i(x) = w^T \phi(x), \tag{2.56}$$

and place a Gaussian prior on the weights:

$$p(w) = \mathcal{N}(0, \Sigma_w). \tag{2.57}$$

Then we can see that we have defined a Gaussian Process prior on f, since f is now a Gaussian Process (by linear combination) [34]:

$$p(f) = \mathcal{GP}\left(0, \phi(x)^T \Sigma_w \phi(x)\right). \tag{2.58}$$

This Gaussian process is called **degenerate**, as it can can be represented with a finite set of basis functions and converted back into weight-space formulation. Furthermore, the rank of its covariance function and Gram matrix is at maximum M. One such Gaussian process is the constructed using the linear kernel, which was examined previously.

Now by Mercer's Theorem (2.1.1), we have that the kernel function can be decomposed as follows:

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x) \psi_i(x'), \tag{2.59}$$

where λ_i and ψ_i are the i-th eigenvalue and eigenfunction respectively. Mercer's Theorem does thus provide a link between the function-space and weight-space formulations.

If the decomposition yields a finite number of terms we thus speak of a degenerate Gaussian Process. Now if $N \leq M$ the Gram matrix is full rank, hence inversion can (in theory) be performed directly, at cost of $\mathcal{O}(N^3)$ operations.

However if $N > M$, the most sensible way to approach the inversion of the matrix $K + \sigma_n^2 I = \Phi \Sigma_w \Phi^T + \sigma_n^2 I$ is to use the matrix inversion lemma, also known as the Woodbury formula A.6 by expressing the covariance matrix in terms of half-matrices such that $K = VV^T$, where $V = \Phi \Sigma_w^{\frac{1}{2}}$ resulting in:

$$\left(VV^T + \sigma_n^2 I\right)^{-1} = \sigma_n^{-2} I - \sigma_n^{-2} V \left(\sigma_n^2 I + V^T V\right)^{-1} V^T. \tag{2.60}$$

The inversion on the right-hand side requires $\mathcal{O}(M^3)$ operations, instead of $\mathcal{O}(N^3)$. Hence the weight-space formulation, equivalent to linear regression, is advantageous is case of a degenerate Gaussian Process.

Now if the decomposition yields an infinite number of terms, the Gaussian process is known as **non-degenerate**. This is the type of Gaussian process we are interested, as it is **non-parametric** in nature, meaning that the complexity of the posterior increases as more data comes in, as we are modelling an underlying global function.

This is a desirable property, as it provides a more complex solution when more data is presented to it. This can be seen from the posterior mean in Equation 2.54, which is essentially a weighed sum of N kernel function evaluations, which however necessitates always $\mathcal{O}(N^3)$. The power of Gaussian Process Regression can be understood perhaps better by switching to the weight-space formulation which would in this case require the inversion of an infinite-dimensional matrix, as the number of basis functions is in fact infinite.

Model choice

As mentioned in the previous section, at the heart of Gaussian process regression lies the choice of kernel function k which does usually have some free parameters θ. Since (in most cases) the hyperparameters are not known exactly in advance, they somehow need to be estimated. Furthermore, as already mentioned, the choice of kernel structure itself is crucial, but it is assumed to be known and representative of the underlying phenomenon.

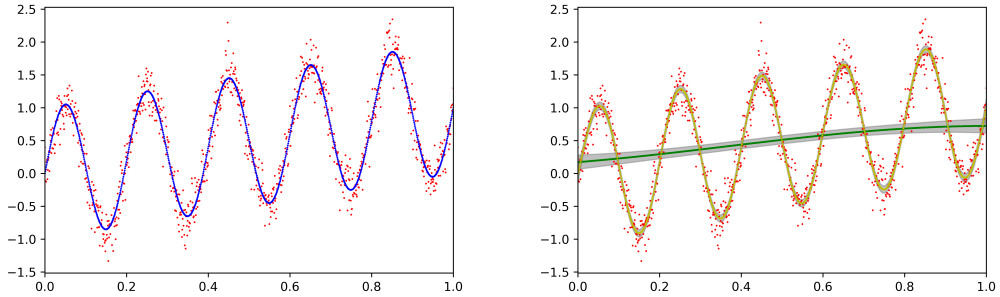
The mean function of the Gaussian Process is often chosen to be zero or modelled in a parametric way. The zero-mean prior assumption, which can be easily guaranteed by normalizing the observations around their mean, does of course not imply a zero-mean posterior. The choice of covariance function is based on the characteristics of the underlying function of interest such as for instance periodicity or monotonicity and determines thus greatly the behaviour of the predictor and can be considered the primary source of bias in the model.

In the context of Gaussian Process Regression, the hyperparameters can be estimated by minimizing the log-marginal-likelihood or evidence, also known as **type II maximum likelihood** or **empirical Bayes**, by essentially adjusting the prior in light of the data. This can be done for instance by cross-validation, as the Gaussian Process Regression framework is a fully probabilistic model. Other options, such as placing hyperpriors over the hyperparameters do not yield an analytical and computationally tractable posterior[20], as one is forced to resort to sampling methods or variational inference. The empirical Bayes approach, which is in practice the standard, does however compromise the fully Bayesian formulation of the method, as the prior belief is in fact shaped according to the data, thereby rendering overfitting a possibility. In general however the behaviour of most kernels is governed by a small number of hyperparameters, meaning that the model is not likely to overfit [20]. From a machine learning theoretical point of view, this corresponds to limiting the complexity of the class one is trying to learn, while from a Bayesian point of view it corresponds to choosing a less informative prior.

Recall that in the examined Gaussian Process model the prior is a Gaussian Process and the marginalized likelihood is Gaussian as well, with the following form:

$$\log p(y) = -\frac{1}{2} y^T (K_{N,N} + \sigma_n^2 I)^{-1} y - \frac{1}{2} \log \det \left(K_{N,N} + \sigma_n^2 I \right) - \frac{N}{2} \log 2\pi. \tag{2.61}$$

It is thus clear that an estimate of the kernel hyperparameters and noise σ_n can be obtained by minimizing this quantity. This is in practice relatively easy as the gradient of the evidence can be determined analytically for many choices of kernel, and thus any optimizer which accepts a gradient can be used. In practice when performing said optimization task, there is always the danger of the optimizer getting stuck in a local minimum. This is especially true when dealing with a little data, as the model is more likely to overfit. Usually, each local minimum reflects a different explanation of the underlying function, as can be seen in Figure 2.2. As the difference between different modes can be big, it is good practice to restart the optimizer multiple times with different initial conditions in order to find the best possible minimum.



(A) Underlying function f and related data points y.

(B) Two predictive posterior distributions with different hyperparameters.

FIGURE 2.2: In the figure the left we see a plot of the underlying function $f(x) = x\sin(10\pi x)$ (in blue) and of its available observations y polluted with noise $\epsilon \sim \mathcal{N}\left(0, 0.2^2\right)$ (in red). On the right, we see the mean and confidence intervals corresponding to twice the standard deviation of the posterior of two different models. Both models use the squared exponential kernel with hyperparameters estimated by minimizing the log-marginalized-likelihood. The less representative solution (in green) was obtained with hyperparameters $l = 0.764$, $,\sigma_s = 0.222$, $,\sigma_n = 0.778$, while the better one (in yellow) used hyperparameters $l = 0.072$, $\sigma_s = 1.567$, $\sigma_n = 0.201$. We can see that the best solution has successfully estimated the noise and recovered the latent function.

Consistency

We have seen previously that the predictive equation in case of Gaussian Process Regression 2.54 can be derived as the minimizer of the regularizer risk functional of Equation 2.44. We would like to know how the solution behave in case of $N \to \infty$, as intuitively we would think that as the data increases the posterior gets overwhelmed and converges to the underlying predictive distribution.

Let $(\mathcal{X}, \mu)$ a finite measure space where μ is the generative distribution of the data points (x, y). Following [31] we note that:

$$\mathbb{E}\left(\sum_{i=1}^{N}\left(y_i - f\left(x_i\right)\right)^2\right) = N\int\left(y - f\left(x\right)\right)^2 d\mu\left(x, y\right). \tag{2.62}$$

Let furthermore $\eta\left(x\right) = \mathbb{E}\left(y|x\right)$ be the **regression function** associated with the probability measure P. The variance of $\eta\left(x\right)$ is denoted as:

$$\sigma^2\left(x\right) = \int\left(y - \eta\left(x\right)\right)^2 d\mu\left(y|x\right). \tag{2.63}$$

By setting

$$y - f = \left(y - \eta\right) + \left(\eta - f\right), \tag{2.64}$$

we get:

$$\int\left(y - f\left(x\right)\right)^2 d\mu\left(x, y\right) = \int\left(\eta\left(x\right) - f\left(x\right)\right)^2 d\mu\left(x\right) + \int\sigma^2\left(x\right) d\mu\left(x\right). \tag{2.65}$$

As the last term is independent of f we get the following regularized risk functional:

$$\mathcal{L}\left(f\right) = \frac{1}{2\sigma_n^2}\int\left(\eta\left(x\right) - f\left(x\right)\right)^2 d\mu\left(x\right) + \frac{1}{2}\left|\left|f\right|\right|_{\mathcal{H}}^2. \tag{2.66}$$

By Mercer's theorem 2.1.1, we can express f by the orthogonal eigenfunctions of the kernel function $k\left(\cdot, x\right)$. Furthermore let $k\left(\cdot, x\right)$ be non-degenerate so that by the reproducing property of Definition 2.1.6 its eigenfunctions span the functions function space of f, forming a complete set, such that

$$f(x) = \sum_{i=1}^{\infty} f_i \phi_i(x), \tag{2.67}$$

and, under the assumption that $\eta(x)$ is well behaved we have

$$\eta(x) = \sum_{i=1}^{\infty} \eta_i \phi_i(x). \tag{2.68}$$

Hence we have that:

$$\mathcal{L}(f) = \frac{N}{2\sigma_n^2} \sum_{i=1}^{\infty} (\eta_i - f_i)^2 + \frac{1}{2} \sum_{i=1}^{\infty} \frac{f_i^2}{\lambda_i}. \tag{2.69}$$

This quantity can be minimized by setting the gradient the with respect to all the f_i to zero, in order to obtain:

$$f_i = \frac{\lambda_i}{\lambda_i + \frac{\sigma_n^2}{N}} \eta_i. \tag{2.70}$$

We can so see that $\frac{\sigma_n^2}{N} \to 0$ as $N \to \infty$. This implies that, given the considered assumptions, namely the non-degeneracy of $k(\cdot, x)$ and the smoothness of $\eta(x)$, as $N \to \infty$ we have that $f_i \to \eta_i$, hence the regression function is recovered. From a Bayesian perspective this corresponds to the prior $f = 0$ being overwhelmed by the data as $N \to \infty$. On the contrary, always by Equation 2.70, we have that if $\sigma_n^2 >> N\lambda_i$ then $f_i \approx 0$. More data is then needed to conduct meaningful inference.

2.4 Commonly used kernels

In context of Gaussian Process regression some kernels are often encountered. They are usually **stationary**, meaning that they are only a function of $x - x'$. Furthermore, the examined ones are all **isotropic** apart from the linear one, meaning that they are invariant to spacial shift and rotations, and depend only on $||x - x'||\,|^2$. Other types of kernels are possible as long as they guarantee a symmetric-positive-definite Gram matrix, however they will not be discussed, as deemed not necessary to the scope of this research.

Polynomial kernel

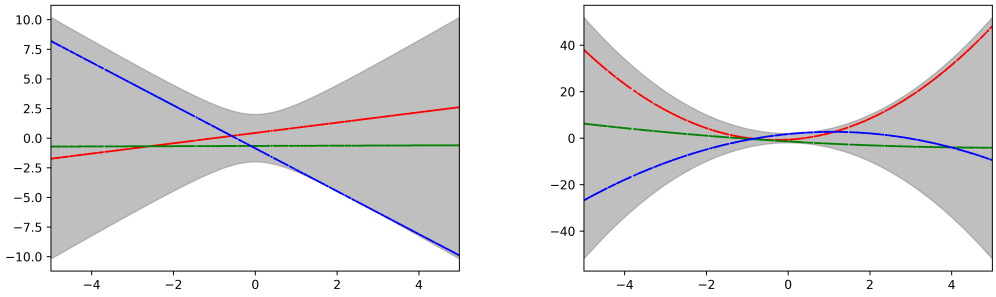
The polynomial kernel is defined as:

$$k(x, x') = \left(x^T x + \sigma_0^2\right)^p, \tag{2.71}$$

which is the common Euclidean distance squared, with order p. The kernel is called inhomogeneous polynomial kernel for $\sigma_0 \neq 0$, and homogenerous polynomial kernel for $\sigma_0 = 0$. It is often used as an example to demonstrate the usefulness of kernel representation. In fact, the kernel is equivalent to a dot product of data points mapped by feature map Φ into vector space consisting of the p-th degree ordered products of its dimensions, which in case of $\sigma_o \neq 0$ encompasses also all its lower degree products [15] [20]. This can be seen, for the homogeneous case, by:

$$\begin{aligned} k(x, x') = \left(x^T x\right)^p &= \left(\sum_{d=1}^{D} x_d x'_d\right)^p = \left(\sum_{d_1=1}^{D} x_{d_1} x'_{d_1}\right) \cdots \left(\sum_{d_p=1}^{D} x_{d_p} x'_{d_p}\right) \\ &= \sum_{d_1=1}^{D} \cdots \sum_{d_p=1}^{D} \left(x_{d_1} \cdots x_d p\right) \left(x'_{d_1} \cdots x'_{d_p}\right) := \left\langle \Phi(x), \Phi(x') \right\rangle, \end{aligned} \tag{2.72}$$

This simple example makes thus clear that the so called kernel trick allows us to avoid the computationally expensive explicit mapping by evaluating the dot product directly in feature space. In the context of Gaussian Process Regression, $p = 1$ is equivalent to performing Bayesian linear regression [20].



(A) $p = 1$ and $\sigma_0 = 1$.

(B) $p = 2$ and $\sigma_0 = 1$.

FIGURE 2.3: In the figure the left three draws from a GP with polynomial kernel with degree $p = 1$ and $\sigma_0 = 1$. On the right, three draws from a GP with polynomial kernel with degree $p = 2$ and $\sigma_0 = 1$. In both cases the grey shaded area corresponds to the bounds which are given by two standard deviations.

Squared exponential

The Gaussian kernel or **squared exponential kernel** also known as **exponentiated quadratic kernel** or **Gaussian kernel** has the general form:

$$k\left(x, x'\right) = \sigma_s^2 e^{-\frac{1}{2}(x-x')^T \Sigma^{-1}(x-x')}. \tag{2.73}$$

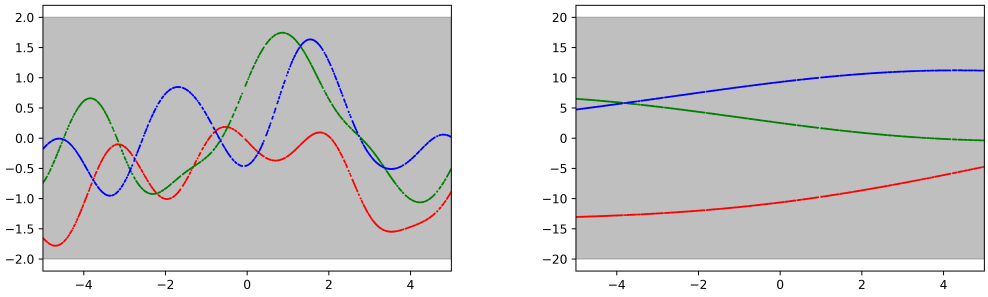
Often Σ is assumed to be diagonal, leading to the following formulation:

$$k\left(x, x'\right) = \sigma_s^2 e^{-\frac{1}{2}\sum_{j=1}^{D}\frac{1}{l_j^2}\left(x_j - x_j'\right)^2}. \tag{2.74}$$

It is clear that l_j can be interpreted as the **characteristic length scale** of the corresponding dimension j, while σ_s governs the amplitude. This kernel is also known as **automatic relevance determination** kernel. The name arises from the fact that the length scales control the relevance of each feature. This is done automatically when maximizing the log-likelihood of Equation 2.61. Sometimes the diagonal terms are assumed to be identical leading to:

$$k\left(x, x'\right) = \sigma_s^2 e^{-\frac{||x-x'||_2^2}{2l^2}}. \tag{2.75}$$

In this case the length scale parameters is sometimes referred to as **bandwidth**. Since the function is differentiable infinite amount of times it is very smooth. Some have argued that this makes the Gaussian kernel an unrealistic choice for modeling physical phenomena [41], and resort to the **Matern kernel**. In practice however, the Gaussian kernel remains the most used kernel for its simplicity [20].



(A) $l = 1$ and $\sigma_s = 1$.

(B) $l = 10$ and $\sigma_s = 10$.

FIGURE 2.4: In the figure the left three draws from a GP with square exponential kernel with length scale $l = 1$ and amplitude $\sigma_s = 1$. On the right, three draws from a GP with length scale $l = 10$ and amplitude $\sigma_s = 10$. In both cases the grey shaded area corresponds to twice the amplitude σ_s.

Matern kernel

The Matern kernel is often used in Gaussian process regression, and has the following form:

$$k(x, x') = \sigma_s^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}||x - x'||_2}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}||x - x'||_2}{l} \right), \tag{2.76}$$

where σ_s is the amplitude, $\nu > 0$, $l > 0$ and K_ν is a modified Bessel function of the second type of order ν. Interestingly, this choice of kernel in the GP setting yields a function which is differentiable n times only if $n > \nu$, a useful characteristic for modeling purposes [20]. Furthermore, if the Gaussian Process is defined on a space with dimension $D = 1$ (x is one dimensional) with the choice of $\nu = \frac{1}{2}$, it is the mean reverting **Ornstein-Uhlenbeck process** [47]. When $\nu \to \infty$ the kernel collapses to the squared exponential kernel. It is thus clear that the Matern kernel offers plenty of possibilities when trying to model a phenomenon having particular known smoothness characteristics.

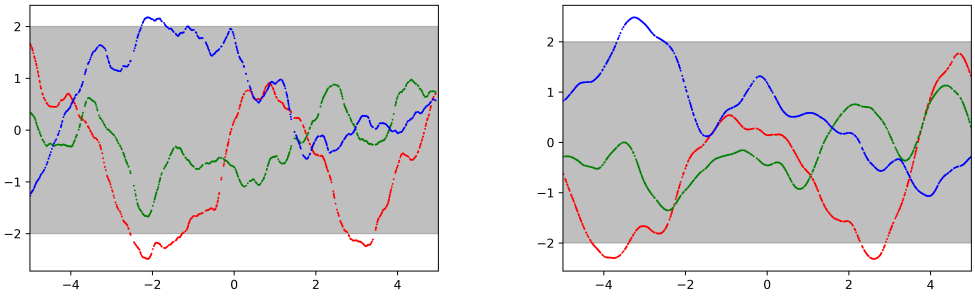


FIGURE 2.5: In the figure the left three draws from a GP with Matern kernel with length scale $l = 1$, amplitude $\sigma_s = 1$ and $\nu = \frac{3}{2}$. On the right, three draws from a GP with length scale $l = 1$, amplitude $\sigma_s = 1$ and $\nu = \frac{5}{2}$. In both cases the grey area corresponds to twice the amplitude.

Neural network kernel A very interesting kernel which shows how Gaussian Process Regression can be seen as an approximation to a neural network with many hidden units is the neural network kernel. It was discovered in the late nineties, when interest in neural networks and kernel methods was once more rising. The rationale behind it is

quite simple. Consider in this regard a neural network with one hidden layer and n hidden units in the regression setting. Its mapping can be described by:

$$f(x) = b + \sum_{i=i}^{n} a_i h(x, w_i), \tag{2.77}$$

where a and b are the weights of the output layer and bias respectively, w_i are the weights of the hidden layer, and $h(\cdot, \cdot)$ is the activation function. Let a an b be independently distributed with zero mean and with variance σ_a^2 and σ_b^2 respectively, and let w_i be i.d.d.. Taking the expectation under all weights, we have that:

$$m(x) = \mathbb{E}(f(x)) = 0, \tag{2.78}$$

$$k(x, x') = \mathbb{E}\left(f(x) f\left(x'\right)\right) = \sigma_b^2 + \sum_{i=1}^{n} \sigma_a^2 \mathbb{E}\left(h(x, w_i) h\left(x', w_j\right)\right) \tag{2.79}$$

$$= \sigma_b^2 + n\sigma_a^2 \mathbb{E}\left(h(x, w_i) h\left(x', w_j\right)\right) \tag{2.80}$$

If we allow σ_a^2 to scale with n i.e. $\sigma_a^2 = \frac{\omega^2}{n}$, we have that, since h is bounded as it is an activation function, for $n \to \infty$, f converges to a Gaussian Process by the Central limit theorem 4.2.2 (in Chapter 4), as they are i.d.d.. Depending on the choice of h it is possible sometimes to obtain an analytical expression the covariance function, for instance in case of $h(x, w) = \operatorname{erf}\left(w_0 + w^T x\right)$ with $w \sim \mathcal{N}(0, \Sigma)$ we have

$$k(x, x') = \frac{2}{\pi} \arcsin\left(\frac{2x^T \Sigma x'}{\sqrt{\left(1 + 2x^T \Sigma x\right)\left(1 + 2x'^T \Sigma x'\right)}}\right), \tag{2.81}$$

where x is here augmented in order for to account for the bias w_0. These kernels are interesting as they enable exact Bayesian inference for infinite width neural networks having one hidden layer [8]. Recently, using an induction argument, these results have been extended as well to deep neural networks [55] with an arbitrary number of hidden layers. These incredible results seem to suggest in this regard that for regression tasks it makes sense to use GPR, however, as the authors point out, the computational burden related to the full GPR makes it hard to apply this (and other types of) kernel to large datasets. Also for this reason, it makes sense to consider approximations to the full method.

Chapter 3

Gaussian Process Regression for large sets of data

As discussed in Chapter 2, the main drawback of employing Gaussian Processes in a regression setting is that their "training" costs $\mathcal{O}\left(N^3\right)$ operations when employing the standard Cholesky decomposition. Apart from computational improvements which regard for instance the inversion (in practice the Cholesky decomposition) of the Gram matrix, such as for instance [59], there have been attempts to make Gaussian Process Regression viable in case of large N, by large meaning usually $N > 10^5$ [61] by means of conducting approximate inference. These attempts have different form and can be divided roughly in two categories, namely **local approximations** and **global approximations**. Local approximations divide the data for subspace learning, wile global approximations try to distillate the entire data [61].

3.1 Local approximations

When considering the $\mathcal{O}\left(N^3\right)$ computational complexity associated with the full Gaussian Process Regression, one might think of partitioning the data in c buckets based one say some distance metric. Applying Gaussian Process Regression to a bucket would then incur in a complexity of $\mathcal{O}\left(\frac{N^3}{c^3}\right)$, hence processing the whole dataset would cost $\mathcal{O}\left(\frac{N^3}{c^2}\right)$ operations. This simple local approximation highlights the problem with this type of approximation, namely the fact that "far" interactions, "far" having here a connotation dictated by the kernel choice and the considered partitioning method, are not considered at all. Prior knowledge of the dataset is thus required, rendering this class of approximations not freely usable in most situations. We thus focus on the more justifieable global approximations.

3.2 Global approximations

As stated previously, global approximations operate on the whole data. They are thus able to capture long-term spatial correlations, but they can fail to see higher-order local patterns. According to [61], they can be divided into three categories, **subset-of-data**, **sparse kernels** and **sparse approximations**. Generally speaking these methods rely on a selection of **support points** also known as **active points** or **inducing points** $\mathcal{A}$, their meaning varying according to the type of approximation that is considered.

Subset-of-data

Subset-of-data is the most simple and intuitive GP approximation. It simply relies on choosing a subset of size $M << N$ of the full data-set, achieving thereby a computational complexity of order $\mathcal{O}\left(M^3\right)$ during training, $\mathcal{O}\left(M\right)$ for posterior mean estimates, and $\mathcal{O}\left(M^2\right)$ for posterior variance. The method can yield extremely good predictions in case of redundant data, but will however fail to deliver a meaningful posterior variance estimates due to the limited data set. Nonetheless, subset-of-data methods have the advantage of being in general fast computational-wise, for M being small enough. The selection of the M points can be done in different ways:

1. **Random selection** is the most simple and can yield good results in case of a redundant data-set and a large number of support points M [61]. While used in practice, in literature it is often used as a baseline for other methods.

2. **Clustering techniques** can be employed to divide the data-set in M clusters, of which for instance the centroid can be taken as subset point. Again, this works well when the data-set is separable and redundant.

3. **Active learning** criteria, such as differential entropy or mutual information can be used to make a selection of points. Since the criteria are usually too computational expensive, they are used in a **greedy** context, meaning that subset points are queried subsequently. They are often encountered in literature.

Many of the criteria found in literature fall into the active learning category, and rely on information-theoretical measures, such as **entropy**, **mutual information** or **KL-divergence**. Information theory is concerned with representing data in a compact fashion as well as with transmitting and storing it in a way that is robust to errors[47]. Since in case of the normal distribution the variance is directly linked to the concept of entropy, and since many active learning information-theoretical criteria make use of the variance estimate, we present some algorithms concerned with two active learning criteria, which can be used to select a subset of points. These criteria can be used as-is, i.e. in principle on any set of data, however their application is not directly practical for large sets of data, as the underlying optimization problem is usually NP-hard. Therefore generally one relies on a greedy approach which can fulfill some performance guarantees when certain conditions are met.

Maximum entropy sampling

One common criterion in literature is **maximum entropy sampling**. In case of continuous random variables, differential entropy is defined as follows.

Definition 3.2.1 (Differential entropy)**.** The differential entropy of a continuous random variable X, with probability density function p is:

$$\mathbb{H}(X) = -\mathbb{E}\left[\log p(X)\right] = -\int_X \log p(x) dP\left(x\right). \tag{3.1}$$

Intuitively entropy can be seen as the average information carried by a random variable. The information measure $I(p) = -p\log(p)$, is a useful measure:

1. $I(p)$ is monotonically decreasing in p, so if an event is more likely it will be considered less informative.

2. $I(p)$ is non-negative. Every event, has the potential to carry some information.

3. $I(1) = 0$. A certain event is not informative, because certain.

4. $I(p_i p_j) = I(p_i) + I(p_j)$. Information gain due to independent events is additive.

One important difference between differential entropy and its continuous counterpart, is that differential entropy can be negative. This has sparked some discussion in the academia, but the two will be referred simply as entropy and thus treated the same way, although according to some incorrectly.

Intuitively, one would like to conduct measurements in the locations which are most informative over the whole space. This can be formulated as follows. Denote $\mathcal{A}$ the locations in which measurements are conducted, $\mathcal{V}$ the space in which measurements are possible. Then in order to perform maximum entropy sampling, one would like to minimize the conditional entropy:

$$\mathcal{A}^* = \underset{\mathcal{A}\subset\mathcal{V}:\mathcal{A}=k}{\operatorname{argmin}}\ \mathbb{H}\left(X_{\mathcal{V}\setminus\mathcal{A}}|X_{\mathcal{A}}\right) = \underset{\mathcal{A}\subset\mathcal{V}:\mathcal{A}=k}{\operatorname{argmax}}\ : \mathbb{H}\left(X_{\mathcal{A}}\right), \tag{3.2}$$

where the last equality is due to the entropy chain rule (conditioning). We can thus see that minimizing the information gain on all unobserved locations $\mathcal{A}\setminus\mathcal{V}$ after having observed

location $\mathcal{A}$ is equivalent to select location which are mutually most uncertain. Furthermore note that, again by the chain rule of entropy, we have:

$$\mathbb{H}\left(X_{\mathcal{A}_i}\right) = \mathbb{H}\left(X_{\mathcal{A}_{y_i}} | X_{\mathcal{A}i-1}\right) + ... + \mathbb{H}\left(X_{\mathcal{A}_{y_2}} | X_{\mathcal{A}_1}\right) + \mathbb{H}\left(X_{\mathcal{A}_{y_1}} | X_{\mathcal{A}_0}\right). \tag{3.3}$$

Unfortunately, this optimization problem has been shown to be NP-hard [7]. Therefore, the following greedy algorithm is used instead [5][10]:

Algorithm 1 Greedy maximum entropy sampling

$i = 0$
$\mathcal{A}_i = \{\emptyset\}$
while $i < k$ **do**
  $\mathcal{A}_{i+1} = \arg\max_y \mathbb{H}\left(X_y \middle| X_{\mathcal{A}_i}\right)$
  $i \leftarrow i + 1$
end while

It is shown that such type of greedy algorithm has a bound on its worse case scenario performance when certain conditions are fulfilled [4]. The result is given below:

Theorem 3.2.1 (Nemhauser et el., 1978)**.** Let F be a monotone submodular set function over a finite set $\mathcal{V}$, with $F(\emptyset)$. Let $\mathcal{A}_{g_k}$ be the set of the first k elements chosen by the greedy algorithm. Then:

$$F\left(\mathcal{A}_{g_k}\right) \geq \left(1 - \frac{1}{e}\right) \max_{\mathcal{A}\subset\mathcal{V}:|\mathcal{A}|=k} F(\mathcal{A}) > 0.63 \max_{\mathcal{A}\subset\mathcal{V}:|\mathcal{A}|=k} F(\mathcal{A}). \tag{3.4}$$

Hence if the objective function of the greedy Algorithm 1 is submodular and monotone, the algorithm is guaranteed to find a set of points such that the performance measured by F is at least 63% of the optimal set. Fortunately entropy is a submodular function, however it is generally not monotone. However the following holds:

Theorem 3.2.2 (Sharma et al., 2015 [51])**.** Given any symmetric matrix $\Sigma \in \mathbb{R}^{n\times n}$ with $\lambda_{\min}(A) \geq 1$, the function $F(X) = \log\det(\Sigma(X, X))$ is monotone.

So in order for Theorem 3.2.1 to hold in the case of entropy, it is sufficient to force the eigenvalues to be larger than one. This can be trivially done in a naive way by controlling the scaling factor, also known as signal variance, parameter σ_s^2. Furthermore, if we look into the structure of, for instance, the Gaussian kernel, we can see that if two points x and x' are extremely close with respect to the characteristic length l, the matrix will tend to be badly conditioned, and the eigenvalues will tend to zero. An important result which relates in case of Gaussian kernels the characteristic length, mesh size and minimum eigenvalue is presented below [6][51]:

Theorem 3.2.3 (Narcowich et al., 1992 [6])**.** Let $\Sigma \in \mathbb{R}^{n\times n}$ be a Gaussian kernel such that one has $k(x, x') = \sigma_s^2 - \frac{||x-x'||_2^2}{2l^2}$ for points $x_1, x_2, ..., x_n \in \mathbb{R}^d$ and $l > 0$. If the minimum separation $\epsilon = \min_{i\neq j} ||x_i - x_j||_2$ satisfies $\frac{\epsilon^2}{2l^2} \geq 6d$ then:

$$\frac{\lambda_{\min}(\Sigma)}{\sigma_s^2} = \Omega\left(\exp\left(-\frac{d}{2}\log\left(\frac{d\epsilon}{2l^2}\right)\right)\right). \tag{3.5}$$

Since the minimal eigenvalue can be forced above one by controlling the two parameters one can envision a method for sampling the most informative points by using such greedy rule ensuring that the entropy is monotone. This can be done either by restricting the choice of hyperparameters or by removing points neighbouring a given point if too close. From experimental research it seems however that the bound presented in Theorem 3.2.3 is rather loose [51], however no theoretical guarantees are known, apart from empirical findings which show good results even in case of not too well conditioned covariance matrices [51].

This information-theoretical criterion is in practice easy to implement when it comes to Gaussian Process Regression. In fact, the entropy of a random variable which follows the multivariate normal distribution is the following:

Proposition 3.2.1 (Entropy of a multivariate normal)**.** Let $X \sim \mathcal{N}(\mu, \Sigma)$. Then

$$\mathbb{H}(X) = \frac{1}{2} \log \det (2\pi e \Sigma) . \tag{3.6}$$

Therefore due to the monotonicity entropy of a multivariate normal random variable, maximizing entropy is equivalent to maximizing the posterior variance. The naive implementation of Algorithm 1 in case of Gaussian Process Regression consists of two steps which are to be repeated per loop cycle i, namely training a model on the active points $\mathcal{A}_i$ and querying the point in $\mathcal{V} \setminus \mathcal{A}_i$ which has maximum variance. Training the model costs $\mathcal{O}\left(|\mathcal{A}_i|^3\right)$ and querying the point having maximum variance costs $\mathcal{O}\left(|\mathcal{V} \setminus \mathcal{A}_i| \, |\mathcal{A}_i|^2\right)$ as well, as a posterior variance estimate is necessary for each point not in $\mathcal{A}_i$. Training costs can however be reduced by employing recursively block Cholesky decomposition as presented in appendix A.2. Updating the model in this way brings down training cost to an order of $\mathcal{O}\left(|\mathcal{A}_i|^2\right)$, similarly to matrix-vector multiplication. In literature this criteria is often used in conjunction with a sparse approximation, and serves thus as to select its support points, see for instance the informative vector machine [16][20].

Maximum mutual information sampling

As shown in [33], an improved design criterion is **mutual information**. Mutual information is defined as follows:

Definition 3.2.2 (Mutual information)**.** The mutual information of random variables X and Y is:

$$\mathbb{MI}(X;Y) = \mathbb{H}(X) - \mathbb{H}(X|Y) .$$

Intuitively, mutual information measures the information that X and Y share, as it weights how much knowing one of these variables reduces uncertainty about the other. Sampling by mutual information ensures that the queried points do not accumulate on the boundary (in which the variance is higher), but fill the whole space unlike the case with entropy. We thus wish to maximize the mutual information:

$$\mathcal{A}^* = \underset{\mathcal{A} \subset \mathcal{V} : \mathcal{A} = k}{\operatorname{argmax}} \mathbb{MI}(X_{\mathcal{A}}; X_{\mathcal{A}}) = \underset{\mathcal{A} \subset \mathcal{V} : \mathcal{A} = k}{\operatorname{argmax}} \mathbb{H}\left(X_{\mathcal{V} \setminus \mathcal{A}}\right) - \mathbb{H}\left(X_{\mathcal{V} \setminus \mathcal{A}} | X_{\mathcal{A}}\right) . \tag{3.7}$$

As in case of entropy, this optimization problem has been shown to be NP-hard [7]. Therefore, the following greedy algorithm is used instead [33], which provides the maximum increase in mutual information:

Algorithm 2 Greedy maximum mutual information sampling

$i = 0$
$\mathcal{A}_, = \{\emptyset\}$
while $i < k$ **do**
  $\mathcal{A}_{i+1} = \operatorname{argmax}_y \mathbb{MI}\left(X_{y \cup \mathcal{A}_i}; X_{\mathcal{V} \setminus \mathcal{A}_i}\right) - \mathbb{MI}\left(X_{\mathcal{A}_i}; X_{\mathcal{V} \setminus \mathcal{A}_i}\right)$
  $i \leftarrow i + 1$
end while

By Theorem 3.2.1 and Lemma 5 in [33], we have a result analogue to the one of Theorem 3.2.3. Once more, in order for the technical conditions to be fulfilled, either the hyperparameters must be restricted, or a minimum inter-point distance must be enforced. However since the bound of Lemma 5 in [33] is loose, practice shows that the algorithm performs well even if the covariance matrix is ill-conditioned. Naive implementations runs to similar problems as with Algorithm 1, however note that by Definition 3.2.2 we have that:

$$\mathcal{F}_i = \mathbb{MI}\left(X_{y \cup \mathcal{A}_i}; X_{\mathcal{V} \setminus \mathcal{A}_i}\right) - \mathbb{MI}\left(X_{\mathcal{A}_i}; X_{\mathcal{V} \setminus \mathcal{A}_i}\right) = \mathbb{H}\left(X_y | X_{\mathcal{A}_i}\right) - \mathbb{H}\left(X_y | X_{\mathcal{V} \setminus (\mathcal{A}_i \cup y)}\right) . \tag{3.8}$$

By properties of the log function and by the posterior variance in the Gaussian Process Regression setting of Equation 2.54, we have that:

$$\mathcal{F}_i \propto \frac{K(y,y) - K(y,\mathcal{A}_i) K^{-1}(\mathcal{A}_i,\mathcal{A}_i) K(\mathcal{A}_i,y)}{K(y,y) - K(y,\mathcal{V}\setminus(\mathcal{A}_i\cup y)) K^{-1}(\mathcal{V}\setminus(\mathcal{A}_i\cup y),\mathcal{V}\setminus(\mathcal{A}_i\cup y)) K(\mathcal{V}\setminus(\mathcal{A}_i\cup y),y)}, \tag{3.9}$$

where the numerator is the queried posterior variance of a GP built on $\mathcal{A}_i$ and the denominator is the queried posterior variance of a GP built on $\mathcal{V}\setminus(\mathcal{A}_i\cup y)$. This representation allows again to use the **block Cholesky decomposition** of Equation A.2 in the Appendix, to ensure a cost at training of $\mathcal{O}\left(|\mathcal{A}_i|^2 + |\mathcal{V}\setminus(\mathcal{A}_i\cup y)|^3\right)$ instead of a naive $\mathcal{O}\left(|\mathcal{A}_i|^3 + |\mathcal{V}\setminus(\mathcal{A}_i\cup y)|^3\right)$. Searching the point which maximizes the objective function in Equation 3.9 requires in any case $\mathcal{O}\left(|\mathcal{V}\setminus\mathcal{A}_i|\,|\mathcal{A}_i|^2 + |\mathcal{V}\setminus(\mathcal{A}_i\cup y)|^3\right)$ operations.

Sparse kernels

Sparse kernels try to reduce the computational complexity of the training phase by ensuring that Gram matrix $K_{N,N}$ which takes part in the posterior mean and variance estimation is sparse. This is usually accomplished by **compactly supported kernels**, which impose $k(x,x') = 0$ if $|x-x'|$ exceed a certain threshold. This ensures that only non-zero elements of $K_{N,N}$ take part in the calculation. Computational complexity is reduced to $\mathcal{O}\left(cN^3\right)$, with $0 < c < 1$. One of the biggest difficulties of this approach is ensuring that the thus constructed kernel is symmetric and positive-definite.

Sparse approximations

Sparse approximations rely on constructing a **low-rank approximation** of the full Gram matrix $K_{N,N}$. This low rank approximation is based on the observation that typically the Gram matrix K has many small eigenvalues which can be removed without loosing too much precision[15]. In practice this could be simply accomplished by keeping the first M eigenvalues of the eigenvalue decomposition:

$$K_{N,N} \approx U_{N,M}\Lambda_{M,M}U_{M,N}. \tag{3.10}$$

The inverse of $K_{N,N}$ could then be computed with the Woodbury formula in Equation A.6, and the determinant with its equivalent for determinants, see EquationA.2, requiring $\mathcal{O}\left(NM^2\right)$ operations. This is however not efficient, as the eigenvalue decomposition requires $\mathcal{O}\left(N^3\right)$ operations. Hence another type of approximation, namely the **Nyström approximation** is used, which achieves low-rank by selecting a subset of M points and has form:

$$K_{A,B} \approx Q_{A,B} := K_{A,M}K_{M,M}^{-1}K_{M,B}. \tag{3.11}$$

Its inverse can then be computed with the Woodbury formula in Equation A.6, and the determinant with its equivalent for determinants in Equation A.2 requiring $\mathcal{O}\left(NM^2\right)$ operations.

The Nyström approximation can be derived in different ways, we show the derivation followed in [15] and [20]. Herein the idea behind the Nyström approximation is to select $M << N$ basis functions $k(x_1,\cdot),...,k(x_M,\cdot)$ where $x_1,...,x_M$ are without loss of generality the first M data points, so that every basis function $k(x_i,\cdot)$ can be approximated as linear combination of the others.

$$k(x_i,\cdot) \approx \tilde{k}(x_i,\cdot) := \sum_{j=1}^{N} \alpha_{i,j} k(x_j,\cdot). \tag{3.12}$$

The coefficients are chosen such that they minimize the distance in reproducing kernel Hilbert space:

$$\begin{aligned}\alpha^* &= \underset{\alpha\in\mathbb{R}^{N\times M}}{\operatorname{argmin}} \operatorname{Err}(\alpha) \\ &= \underset{\alpha\in\mathbb{R}^{N\times M}}{\operatorname{argmin}} \sum_{i=1}^{N} \left|\left|k(x_i,\cdot) - \tilde{k}(x_i,\cdot)\right|\right|^2_{\mathcal{H}} \\ &= \operatorname{tr}(K_{N,N}) - 2\operatorname{tr}(\alpha K_{M,N}) + \operatorname{tr}\left(\alpha K_{M,M}\alpha^T\right) \\ &= K_{N,M}K^{-1}_{M,M},\end{aligned} \tag{3.13}$$

which gives finally:

$$K_{N,N} \approx Q_{N,N} := K_{N,M}K^{-1}_{M,M}K_{M,N}. \tag{3.14}$$

It can be shown that [15]:

$$\operatorname{Err}(\alpha^*) = \operatorname{tr}(K_{N,N}) - \operatorname{tr}(Q_{N,N}). \tag{3.15}$$

The quantity in Equation 3.15 is used by [13] as a selection criteria of the M inducing points in a subset-of-data setting.

Substituting the Nyström approximation of Equation 3.11 into the equations of posterior mean and variance 2.54 can however lead to rather unpredictable behaviour, since it does not define a generative probabilistic model, and the low-rank covariance is not necessarily symmetric-positive-definite.

Inspired by the Nyström approximation, sparse approximations build a generative probabilistic model, which achieves sparsity by the choice of M inducing points (also known as active points or support points or pseudo points) to summarize the whole data. Denote these points with (X_M, f_M). In all sparse approximations f_M follows the same GP prior as f. Furthermore, f_M is assumed to be a sufficient statistic for f, i.e. for any variable z it holds that $p(z|f, f_M) = p(z|f_M)$. The joint prior can always be recovered by marginalizing out f_M as

$$p(f, f^*) = \int p(f, f^*|f_M)\, p(f_M)\, df_M. \tag{3.16}$$

Following the classification in [24] and [61] three type of sparse approximations can be defined:

1. **Prior approximations** which approximate the prior but perform exact inference.
2. **Posterior approximation** which retain the exact prior but perform approximate inference.
3. **Structured sparse approximations** which exploit specific structures of the Gram matrix.

In what follows we will go trough the main prior approximations by looking at the various methods as described by their authors and in the unifying comparative framework developed in [24].

Prior approximations

Prior approximations modify the joint prior in Equation 3.16 which is the origin of the cubic complexity [24], using assumptions of independence:

$$p(f, f_*) \simeq q(f, f_*) = \int q(f|f_M)\, q(f_*|f_M)\, p(f_M)\, df_M, \tag{3.17}$$

where $p(f, f_M)$ and $p(f_*, f_M)$ are known as training and test conditionals which are distributed as follows (by Equation A.2):

$$p(f|f_M) = \mathcal{N}\left(K_{NM}K^{-1}_{MM}f_M,\, K_{N,N} - Q_{N,N}\right), \tag{3.18}$$

$$p(f^*|f_M) = \mathcal{N}\left(K_{*,M}K^{-1}_{M,M}f_M,\, K_{*,*} - Q_{*,*}\right). \tag{3.19}$$

f_M is called inducing variable as it induces the dependency between f and f_*, which are conditionally independent. In order to obtain computational gains, the covariances of the train and test conditionals are modified as:

$$p\left(f|f_M\right) = \mathcal{N}\left(K_{NM}K_{MM}^{-1}f_M,\, \tilde{Q}_{N,N}\right), \tag{3.20}$$

$$p\left(f_*|f_M\right) = \mathcal{N}\left(K_{*,M}K_{M,M}^{-1}f_M,\, \tilde{Q}_{*,*}\right). \tag{3.21}$$

The log-marginal-likelihood $p\left(y\right)$ of Equation 2.61 is thus approximated as:

$$\log p\left(y\right) = -\frac{1}{2}y^T\left(\tilde{Q}_{N,N} + Q_{N,N} + \sigma_n^2 I\right)^{-1}y - \frac{1}{2}\log\left|\tilde{Q}_{N,N} + Q_{N,N} + \sigma_n^2 I\right| - \frac{N}{2}\log\left(2\pi\right). \tag{3.22}$$

Specific choices of $\tilde{Q}_{N,N}$ enable computation of $\left(\tilde{Q}_{N,N} + Q_{N,N} + \sigma_n^2 I\right)^{-1}$ and $\left|\tilde{Q}_{N,N} + Q_{N,N} + \sigma_n^2 I\right|$ in $\mathcal{O}\left(NM^2\right)$ operations thanks to Equation A.6 and Equation A.2, and lead to different sparse approximations.

Subset of regressors Subset of regressors (SOR) [13], also called **deterministic inducing conditional** in the unifying framework of [24], is a sparse approximation which uses a linear model in the parameters for any f_*:

$$f_* = K_{*,M}w_M, \tag{3.23}$$

$$p\left(w_M\right) = \mathcal{N}\left(0, K_{M,M}^{-1}\right), \tag{3.24}$$

which assigns one weight to each inducing input. Note that due to the definition of w it is possible to recover the exact prior in f_M:

$$f_M = K_{M,M}w_M \quad \Longrightarrow \quad \left\langle f_M, f_M\right\rangle = K_{M,M}\left\langle w_M, w_M\right\rangle K_{M,M} = K_{M,M}, \tag{3.25}$$

which thus read

$$p\left(f_M\right) = \mathcal{N}\left(0, K_{M,M}\right). \tag{3.26}$$

Using the fact that $f_M = K_{M,M}w_M$ from Equation 3.23 and the exact prior on f_M it is possible to reformulate the subset of regressors model as follows:

$$f_* = K_{*,M}K_{M,M}^{-1}f_M, \tag{3.27}$$

$$p\left(w_M\right) = \mathcal{N}\left(0, K_{M,M}^{-1}\right). \tag{3.28}$$

Conditioning on f_M yields the training and test conditionals:

$$q\left(f|f_M\right) = \mathcal{N}\left(K_{NM}K_{MM}^{-1}f_M, 0\right), \tag{3.29}$$

$$q\left(f^*|f_M\right) = \mathcal{N}\left(K_{*,M}K_{M,M}^{-1}f_M, 0\right), \tag{3.30}$$

which corresponds to choosing $\bar{Q}_{N,N} = 0$ and $\bar{Q}_{*,*} = 0$. The joint prior can then be obtained by Equation 3.17 and reads:

$$q\left(f, f_*\right) = \mathcal{N}\left(0, \begin{bmatrix} Q_{N,N} & Q_{N,*} \\ Q_{*,N} & Q_{*,*}\end{bmatrix}\right). \tag{3.31}$$

The joint prior in Equation 3.16 is factually a Nyström approximation on the set of M support points, with a rank of at most M. This means that when drawing function from the prior we are limited to a maximum of M independent functions, the other being linear combinations of the former [24]. While this restrictive approach might work when considering the predictive posterior distribution, it does considerably underpredict the predictive posterior variances [20] [24] [61]. Hence the predictive equations of the subset of regressors approximation can be obtained simply by replacing the various sub-matrices of the full Gramm

matrix of Equation 2.48 with the ones of Equation 3.31. The resulting predictive equation read:

$$q(f_*|y) = \mathcal{N}\left(Q_{*,N}\left(Q_{N,N}+\sigma_n^2 I\right)^{-1} y,\ Q_{*,*} - Q_{*,N}\left(Q_{N,N}+\sigma_n^2 I\right)^{-1} Q_{N,*}\right) \tag{3.32}$$

$$= \mathcal{N}\left(\sigma_n^{-2} K_{*,N}\Sigma K_{M,N} y,\ K_{*,M}\Sigma K_{M,*}\right), \tag{3.33}$$

where $\Sigma = \left(\sigma_n^{-2} K_{M,N}K_{N,M} + K_{M,M}\right)^{-1}$. The first equation closely resembles the full Gaussian Process Regression predictive equations 2.54, while the second one allows for computational savings thanks to the Woodbury formula in Equation A.6, requiring $\mathcal{O}\left(NM^2\right)$ operations for "training", $\mathcal{O}\left(M\right)$ for posterior mean estimates and $\mathcal{O}\left(M^2\right)$ for posterior variance.

Deterministic training conditional The deterministic training conditional (DTC), also known as **projected process approximation** in [31] was proposed in [18] to improve the posterior variance estimate of the subset of regressors method, while providing the same posterior mean estimates. The original derivation in [18] is a likelihood approximation which makes use of the projection $f = K_{f,f_M}K_{f_M,f_M}f_M$ such that:

$$p(y|f) \simeq q(y|u) = \mathcal{N}\left(K_{f,f_M}K_{f_M,f_M}^{-1} f_M,\ \sigma_n^2 I\right). \tag{3.34}$$

The equivalent derivation in the unifying framework of prior approximations by [24] maintains the likelihood of the full model, but introduces the train and test conditionals which have the following form:

$$q(f|f_M) = \mathcal{N}\left(K_{NM}K_{MM}^{-1} f_M,\ 0\right), \tag{3.35}$$

$$q(f^*|f_M) = p(f_*|f_M). \tag{3.36}$$

The train conditional is the same of the subset of regressors method, and the test conditional is simply the exact test conditional. The associated joint prior is given by:

$$q(f, f_*) = \mathcal{N}\left(0,\ \begin{bmatrix} Q_{N,N} & Q_{N,*} \\ Q_{*,N} & K_{*,*} \end{bmatrix}\right), \tag{3.37}$$

which is the same as the one of the subset of regressors of equation 3.31, except for that $Q_{*,*}$ is replaced by the full $K_{*,*}$. As a result as already said, the predictive mean is the same as in case of the subset of regressors method, while the variance is not:

$$q(f_*|y) = \mathcal{N}\left(Q_{*,N}\left(Q_{N,N}+\sigma_n^2 I\right)^{-1} y,\ K_{*,*} - Q_{*,N}\left(Q_{N,N}+\sigma_n^2 I\right)^{-1} Q_{N,*}\right), \tag{3.38}$$

$$= \mathcal{N}\left(\sigma_n^{-2} K_{*,N}\Sigma K_{M,N} y,\ K_{*,*} - Q_{*,*} + K_{*,M}\Sigma K_{M,*}\right), \tag{3.39}$$

where again $\Sigma = \left(\sigma_n^{-2} K_{M,N}K_{N,M} + K_{M,M}\right)^{-1}$. Now unlike the subset of regressors approach, which yields a Gaussian Process, in the case of the deterministic training conditional, the covariance function is factually not the same for the latent values corresponding to the training set and to the ones corresponding to the test set. Since the process cannot be described as in Definition 2.3.1, it is not properly a Gaussian Process. The computational complexity is the same, which is $\mathcal{O}\left(NM^2\right)$ operations for training, $\mathcal{O}\left(M\right)$ for posterior mean estimates and $\mathcal{O}\left(M^2\right)$ for posterior variance.

Fully independent training conditional The fully independent training conditional (FITC), also known as **sparse pseudo-input Gaussian Process**, was proposed by [30]. As in case of the deterministic training conditional, the original derivation is done by a likelihood approximation which has the form

$$p(y|f) \simeq q(y|f_M) = \mathcal{N}\left(K_{f,f_M}K_{f_M,f_M}^{-1} f_M,\ \mathrm{diag}\left(K_{f_M,f_M} - Q_{f,f}\right) + \sigma_n^2 I\right). \tag{3.40}$$

The corresponding training and test conditionals are [24]:

$$q\left(f|f_M\right) = \prod_{i=1}^{N} = \mathcal{N}\left(K_{NM}K_{MM}^{-1}f_M,\ \text{diag}\left(K_{f,f} - Q_{f,f}\right)\right), \tag{3.41}$$

$$q\left(f^*|f_M\right) = \mathcal{N}\left(f_*|f_M\right). \tag{3.42}$$

As we can see, the approximation does not induce a deterministic relation between f_M and f. The associated effective prior is given by:

$$q\left(f, f_*\right) = \mathcal{N}\left(0,\ \begin{bmatrix} Q_{N,N} - \text{diag}\left(Q_{f,f} - K_{f,f}\right) & Q_{N,*} \\ Q_{*,N} & K_{*,*} \end{bmatrix}\right). \tag{3.43}$$

The only difference with the fully deterministic conditional is that the sub-matrix associated with f is exact on the diagonal. The predictive distribution is:

$$q\left(f_*|y\right) = \mathcal{N}\left(Q_{*,N}\left(Q_{N,N} + \Lambda\right)^{-1} y,\ K_{*,*} - Q_{*,N}\left(Q_{N,N} + \Lambda\right)^{-1} Q_{N,*}\right) \tag{3.44}$$

$$= \mathcal{N}\left(K_{*,N}\Sigma K_{M,N}\Lambda^{-1}y,\ K_{*,*} - Q_{*,*} + K_{*,M}\Sigma K_{M,*}\right), \tag{3.45}$$

where $\Sigma = \left(K_{M,N}\Lambda^{-1}K_{N,M} + K_{M,M}\right)^{-1}$ and $\Lambda = \text{diag}\left(K_{f_M,f_M} - Q_{f,f} + \sigma_n^2 I\right)$. As in case of the deterministic training conditional, the underlying process is not a Gaussian Process. It can however become one if we allow for the following joint prior:

$$q\left(f, f_*\right) = \mathcal{N}\left(0,\ \begin{bmatrix} Q_{N,N} - \text{diag}\left(Q_{f,f} - K_{f,f}\right) & Q_{N,*} \\ Q_{*,N} & Q_{N,N} - \text{diag}\left(Q_{f,f} - K_{f,f}\right) \end{bmatrix}\right), \tag{3.46}$$

as the covariance function would in this case be $k\left(x, x'\right) = k\left(x, x'\right) + \delta_{i,j}\left(k\left(x, x'\right) - k\left(x, x'\right)\right)$. This sparse approximation method is referred to as **fully independent conditional** in the unifying framework of [24], and has the same computational complexity as the other Nyström based methods. One possible improvement that maintains the same computational complexity is to retain the exact covariances on a larger portion than only the diagonal, i.e. considering training conditional having exact block-diagonal matrices. The name assigned by [24] to this approach is the **partially independent training conditional**, and recommends using $\frac{N}{M}$ blocks of size $M \times M$. Once more, the joint prior would not be a Gaussian Process, which can be corrected by allowing the test conditional to have a similar structure.

Posterior approximations

Unlike prior approximations, posterior approximations do not rely on modifying the joint prior $p\left(f, f_*\right)$, and rely instead on approximating the predictive posterior distribution directly. As all sparse approximations, a set of M inducing i.d.d. variables and locations $\left(X_M, f_M\right)$ is considered. By the full Gaussian Process prior assumption of Equation 3.16 and the posterior predictive equations 2.54 we have (using Nyström notation $Q_{N,N} = K_{N,N} - K_{N,M}K_{M,M}^{-1}K_{M,N}$) that:

$$p\left(y|f\right) = \mathcal{N}\left(y|f,\ \sigma_n^2 I\right), \tag{3.47}$$

$$p\left(f|f_M\right) = \mathcal{N}\left(f|K_{N,M}K_{M,M}^{-1}f_M,\ K_{N,N} - Q_{N,N}\right), \tag{3.48}$$

$$p\left(f_M\right) = \mathcal{N}\left(f_M|0,\ K_{M,M}\right). \tag{3.49}$$

Applying Jensen's inequality on $\log p\left(y|f_M\right)$ leads to:

$$\log p\left(y|f_M\right) = \log \mathbb{E}_{p(f|f_M)}\left(p\left(y|f_M\right)\right) \geq \mathbb{E}_{p(f|f_M)}\left(\log p\left(y|f_M\right)\right) := \mathcal{L}_1. \tag{3.50}$$

Note that the difference $\mathcal{L}_1 - \log\left(y|f_M\right)$ is the
Kullback-Leibler divergence $\mathbb{KL}\left(p\left(f, f_M\right);\ p\left(f|f_M, y\right)\right)$ defined as:

Definition 3.2.3 (Kullbach-Leibler (KL) divergence)**.** The Kullbach-Leibler divergence of two random variables X and Y having probability distributions p and q is:

$$\mathbb{KL}\left(p;q\right)=\mathbb{E}_{p(X)}\left(\log\frac{p\left(X\right)}{q\left(X\right)}\right)=\int_{X}\log\frac{p(x)}{q(x)}dP\left(x\right).$$

If the assumption is made that $p\left(y_i|f_i\right)$ are conditionally independent, i.e.

$$p\left(y|f\right)=\prod_{i=1}^{N}p\left(y_i|f_i\right), \tag{3.51}$$

it can be shown that the bound reduces to:

$$e^{\mathcal{L}_1}=\prod_{i=1}^{N}\mathcal{N}\left(y_i|K_{N,M}K_{M,M}^{-1}f_M\right)e^{-\frac{1}{2\sigma_n^2}\mathrm{diag}(K_{N,N}-Q_{N,N})}. \tag{3.52}$$

The $\mathcal{L}_1$ bound in Equation 3.50 is the basis of all posterior approximations, as it provides a measure of distance in the KL sense of of the posterior given the data and inducing variables and the posterior given the inducing variables only. The KL divergence $\mathbb{KL}\left(p\left(f,f_M\right);\,p\left(f|f_M,y\right)\right)$ is minimized when there are M inducing variables f_M such that $f=f_M$ and that $Q_{N,N}=K_{N,N}$, which implies that $e^{\mathcal{L}_1}=p\left(y|f\right)$ achieving equality in the bound of Equation 3.50. The idea behind this type of posterior distribution is to then maximize the bound with respect to variational parameters, minimizing the KL divergence.

Variational free energy The most well known sparse posterior approximation is the variational free energy method (VFE) proposed in [35]. Herein the author derives a bound by marginalizing out the inducing variables f_M such that:

$$\log p\left(y\right)=\log\int p\left(y|f_M\right)p\left(f_M\right)df_M\geq\log\int e^{\mathcal{L}_1}p\left(f_M\right):=\mathcal{L}_2, \tag{3.53}$$

which leads to:

$$\mathcal{L}_2=\log\mathcal{N}\left(y|0,\,K_{N,M}K_{M,M}^{-1}K_{M,M}+\sigma_n^2I\right)-\frac{1}{2\sigma_n^2}\mathrm{tr}\left(K_{N,N}-Q_{N,N}\right). \tag{3.54}$$

Note that:

$$\mathcal{L}_2=\log q_{DTC}-\frac{1}{2\sigma_n^2}\mathrm{tr}\left(K_{N,N}-Q_{N,N}\right). \tag{3.55}$$

Hence the $\mathcal{L}_2$ bound is the same as the DTC likelihood except for the trace term which acts as regularizer [35] and represent the total variance evaluated on the training set, or more simply as we have seen in Equation **??**, the goodness of the Nyström approximation. It has been shown furthermore in [35] that $\mathcal{L}_2$ is increasing as a function of M, i.e. more inducing points give a model that fits the training data better and that is closer (in the KL sense) to the full Gaussian Process Regression model. As one would expect from a posterior approximation, we see that when $f_M=f$ the trace term goes to zero, and the $\mathcal{L}_2$ becomes the likelihood of the full model. The related predictive distribution is the same as the DTC one, which is given by:

$$q\left(f_*|y\right)=\mathcal{N}\left(Q_{*,N}\left(Q_{N,N}+\sigma_n^2I\right)^{-1}y,\,K_{*,*}-Q_{*,N}\left(Q_{N,N}+\sigma_n^2I\right)^{-1}Q_{N,*}\right) \tag{3.56}$$

$$=\mathcal{N}\left(\sigma_n^{-2}K_{*,N}\Sigma K_{M,N}y,\,K_{*,*}-Q_{*,*}+K_{*,M}\Sigma K_{M,*}\right), \tag{3.57}$$

where again $\Sigma=\left(\sigma_n^{-2}K_{M,N}K_{N,M}+K_{M,M}\right)^{-1}$. The only difference with the DTC approach is thus related to the pseudo input determination, which can be performed jointly with the hyperparameters by maximizing $\mathcal{L}_2$. Similarly to the VFE method, the FITC likelihood recovers the full likelihood when $f_M=f$. However by its construction (and in practice due to the absence of the regularization term), the full likelihood is not its the global optimum, leading to predictions which can be potentially better than the ones of the full model. In this sense

the examined prior approximations, and in particular the FITC, can be seen as models which differ from the full Gaussian Process Regression formulation, in contrast with posterior approximations. The approach of [35] presented above requires thus the same computational costs of $\mathcal{O}\left(NM^2\right)$. This was reduced by [45] by applying **stochastic variational inference** (SVI) coupled with with stochastic gradient descent to the bound of Equation 3.53 leading to a complexity of $\mathcal{O}\left(M^3\right)$, being independent of the size of the set of data. The approach was generalized by [50] to DTC, FITC and their variants, although one of the limitations is that the optimization of the inducing points cannot be carried out jointly with the hyperparameters learning. Although the computationally advantageous SVI approach of [45] has no real drawbacks compared to the standard formulation of [35] as it converges to the same predictive distribution, it will not be examined nor implemented, as the size of the sets of data in this thesis do not make its use necessary.

Choice of inducing points and model selection

Until this point the selection of inducing points X_M was considered as given. Historically inducing points have always been selected as a subset of the training set by some kind of criteria in a subset-of-data setting. In fact, every prior approximation presented hereto was presented in its seminal paper as coupled to a subset-of-data method. As explained previously, to address the computational complexity of a full combinatorial search, greedy algorithms have been the standard method of selection. Greedy algorithms such as the ones proposed in the previous section are quite fast, however in order to perform well they need to have access to reasonable model hyperparameters, which practically forces to conduct minimization of the log-likelihood of the full Gaussian process regression model on either the full training set or on a randomly selected subset [20]. In their paper, [30] proposed to jointly optimize the log-likelihood of Equation 3.22 to obtain point estimates of both the hyperparameters and the inducing points simultaneously, which can be done for all sparse method considered until now. The advantages of this approach become less evident for large M [61]. The computational complexity of evaluating the log-likelihood of Equation 3.22 is $\mathcal{O}\left(NM^2\right)$, while the evaluation of its gradient costs $\mathcal{O}\left(NM^2D\right)$ per iteration. In any case, obtaining the global maximum is very unlikely in case of both approaches, as they both carry the risk of finding a poor global optimum [30].

Structured sparse approximations

Until this point the considered methods can be considered general in the sense that they do not require any structure in the data or in the choice of kernel. Considering data that has a particular structure, coupled with a suitable choice of kernel, can give rise to other approximation methods, labelled **structured sparse approximations** by [61].

Toeplitz methods (exact) Toeplitz methods were for Gaussian Process Regression were first introduced in [26]. As one can guess by their denomination, Toeplitz methods are based on the **Toeplitz matrix** structure arising in the Gram matrix of an isotropic kernel combined with the data lying on an evenly spaced one dimensional grid:

Definition 3.2.4 (Toeplitz matrix)**.** Let $A \in \mathbb{R}^{m\times n}$. If $A_{i,i} = A_{i+1,j+1}\, \forall i \in \{1,\dots,m-1\}$ and $\forall\, j \in \{1,\dots,n-1\}$, then A is a Toeplitz matrix.

Such structure is advantageous as it is possible to perform Cholesky decomposition and other normally costly operation in $\mathcal{O}\left(N^2\right)$ provided the matrix is positive-definite [26]. A full representation of the matrix is furthermore never needed, as in case of the square Gram matrix the first row or column contains all the distinct elements. The computational cost of "training" was reduced further to an impressive $\mathcal{O}\left(N\log\left(N\right)\right)$ by [52] using fast matrix vector products. The pitfal of such approximations is that they are suitable for only a very limited number of scenarios.

Kronecker methods (exact) Kronecker methods were introduced in [65] and [43]. As the name suggests they revolve around the **Kronecker product**:

Definition 3.2.5 (Kronecker product). Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$. Then:

$$C = A \otimes B = \begin{bmatrix} A_{1,1}B & \dots & A_{1,n}B \\ & \vdots & \\ A_{m,1}B & \dots & A_{m,n}B \end{bmatrix}, \tag{3.58}$$

where $C \in \mathbb{R}^{mp\times nq}$. More generally, let $A = A_1 \otimes \cdots \otimes A_D = \bigotimes_{d=1}^{D} A_d$, where $A_d \in \mathbb{R}^{G_d \times G_d}$. Then:

$$A_{i,j} = A_{1_{i^{(1)},j^{(1)}}} \dots A_{D_{i^{(D)},j^{(D)}}}, \tag{3.59}$$

where $0 \leq i^{(d)}, j^{(d)} < G_d$ and $0 \leq i, j < N = \prod_{d=1}^{D} G_d$, meaning that $A \in \mathbb{R}^{\prod_{d=1}^{D} G_d \times \prod_{d=1}^{D} G_d}$.

If we consider inputs (X, y), with $x \in \mathbb{R}^D$ lying on a Cartesian grid such that $x \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_D$, and a product kernel $k(\cdot,\cdot)$ having form:

$$k(x, x') = \prod_{i=1}^{D} k_i(x_i, x_i'), \tag{3.60}$$

where $k_i(\cdot,\cdot)$ is the factor of $k(\cdot,\cdot)$ associated with dimension i, then the i,j-th entry of the Gram matrix associated with $k(\cdot,\cdot)$ can be expressed in the form of Equation 3.59, and thus by Definition 3.2.5 as:

$$K_{N,N} = \bigotimes_{d=1}^{D} K_{N,N_d}, \tag{3.61}$$

where K_{N,N_i} is the Gram matrix constructed on basis of the i-th dimension of the whole X, N being in this case $N = \prod_{i=1}^{D} N_i$. By the mixed-product property of the Kronecker product ($(A \otimes B)(C \otimes D) = AC \otimes BD$), the Gram matrix $K_{N,N}$ can be efficiently decomposed by Cholesky decomposition:

$$K_{N,N} = LL^T, \tag{3.62}$$

with $L = \bigotimes_{d=1}^{D} L_d$, L_d being the lower triangular matrix resulting from the Cholesky decomposition of the Gram matrix belonging to the d-th dimension. More importantly, in the work of [65] and [43], the eigenvalue decomposition of the full Gram matrix plays an important role, and similarly the Cholesky decomposition, can be decomposed as:

$$K_{N,N} = U\Lambda U^T, \tag{3.63}$$

where again the matrices can be decomposed as Kronecker products along its dimensions. Both decompositions are comparable in complexity, namely $\mathcal{O}(N^3)$, and are obtainable basically for free for $D << N$. The reason why eigenvalue decomposition is substituting the in this context more familiar Cholesky decomposition, is that it is used by the authors to perform the matrix vector multiplication in the predictive mean posterior distribution of Equation 2.54 in $\mathcal{O}(N)$ instead of the usual $\mathcal{O}(N^2)$ by expressing it as a tensor product [65]. The overall computational complexity regarding "training", i.e. what in the normal Gaussian Process Regression framework encompasses Gram matrix inversion and the aforementioned product, requires only $\mathcal{O}\left(DN^{\frac{D+1}{D}}\right)$ operations as thanks to eigenvalue decomposition as we can efficiently use fast matrix vector products:

$$\left(K + \sigma_n^2 I\right)^{-1} y = \left(U\Lambda U^T + \sigma_n^2\right)^{-1} y = U\left(\Lambda + \sigma_n^2\right)^{-1} U^T y, \tag{3.64}$$

Similarly to the Toeplitz methods, which complement Kronecker methods, the incredible performance does however require the data to be structured, which is a major drawback, as in most real-life scenarios it is either simply not structured or might be affected by corruption. The latter case was analyzed by [43] by introducing dummy variables and responses, and non-stationary noise. Furthermore, Kronecker methods suffer from the "curse of dimensionality", as the grid blows up in size exponentially, limiting further their applicability.

Structured kernel interpolation As explained in the previous paragraphs, the Toeplitz and Kronecker methods reduce tremendously the amount of operations needed for "training", the main requirement being that the data is structured on a lattice. In order to overcome this limitation, one could think of considering a sparse approximation e.g. SOR, which as has been seen can be regarded as a standard Gaussian Process Regression model with a kernel embedding the Nyström approximation of the type $k(x, x') = Q_{x,x'} = K_{x,M}K_{M,M}^{-1}K_{M,x'}$, and then place the M inducing points on a grid, in order to allow computational gains to be made by Kronecker or Toeplitz methods when it comes to the handling of $K_{M,M}$ in the predictive posterior mean of Equation 3.32 (in case of SOR). In this way the $\mathcal{O}\left(M^3\right)$ operations embedded in the $\mathcal{O}\left(NM^2\right)$ total operations could be reduced to superlinear.

In order to further reduce the total $\mathcal{O}\left(NM^2\right)$ operations of the considered (unstructured) sparse approximations, the idea behind **structured kernel interpolation** (SKI) presented in [52] is to approximate the $K_{N,M}$ Gram matrix by interpolating with the smaller Gram matrix of the inducing points $K_{M,M}$, e.g. if one considers linear interpolation in a one dimensional scenario in which the Toeplitz approximations is applicable, the i, j-th entry of $K_{N,M}$ given by $k\left(x_i, x_{M_j}\right)$ it can be expressed approximated as $k\left(x_i, x_{M_j}\right) \approx w_i k\left(x_{M_a}, x_{M_j}\right) + (1-w)\, k\left(x_{M_b}, x_{M_j}\right)$, where x_{M_a} and x_{M_b} are the inducing points bounding x_i, and w_i and $(1-w_i)$ are the interpolation weights. The general form of SKI interpolation can thus be written as:

$$K_{N,M} \approx W K_{M,M}, \tag{3.65}$$

where W is the sparse matrix containing the interpolation weights. When substituting the approximation of Equation 3.65 in the training set Gram sub-matrix of the joint prior of SOR of Equation 3.31, we have the following:

$$K_{N,N} \approx K_{N,M}K_{M,M}^{-1}K_{M,N} \approx W K_{M,M}K_{M,M}^{-1}K_{M,M}W^T = W K_{M,M}W^T. \tag{3.66}$$

We can see that the SKI approximation of Equation 3.66 allows matrix-vector multiplication with $\mathcal{O}\left(N + M^2\right)$ if the inducing points are unstructured and W is sparse. If Kronecker structure is exploited, we have again $\mathcal{O}\left(DM^{\frac{D+1}{D}}\right)$ operations, $\mathcal{O}\left(N + M\log\left(M\right)\right)$ in case of Toeplitz structure, allowing fast "training" to take place by solving Equation 2.54 with conjugate gradient. To evaluate the log-marginal likelihood, it is also necessary to compute $\log\left|K_{N,N} + \sigma_n^2 I\right|$, which is in SKI approximated by using the first N eigenvalues of $K_{M,M}$:

$$\log\left|K_{N,N} + \sigma_n^2 I\right| = \sum_{i=1}^{N} \log\left(\lambda_{N,Ni} + \sigma_n^2\right) \approx \sum_{i=1}^{N} \log\left(\frac{N}{M}\lambda_{M,Mi} + \sigma_n^2\right), \tag{3.67}$$

where here $\lambda_{N,N}$ and $\lambda_{M,M}$ are the eigenvalues of the full and grid Gram matrices respectively. The approximation is shown to be asymptotically consistent for large N.

Generally speaking, SKI works well, i.e. requires less inducing points, if the kernel is smooth. More involved kernels require better interpolations, leading to a fuller W. Furthermore, as SKI can make use of Toepliz or Kronecker structure, it works well for small dimensional datasets, in particular with $D \leq 4$ as it otherwise incurs in the "curse of dimensionality" [61], as the grid size blows up exponentially in D.

The authors in [52] also point out the interesting connection of SKI with the unifying framework of [24]. In particular, by performing interpolation using the Gaussian Process Regression predictive posterior mean of Equation 2.54 (by setting $\sigma_n = 0$) on the kernel itself, such that the training set is $\mathcal{S} = \left\{\left(x_{M_i}, k\left(x_{M_i}, x_N\right)\right)_{i=1}^{M}\right\}$, we get back the SOR kernel. Likewise all other sparse approximations examined by [24] can be derived in this way, highlighting that the inducing points are actually performing the duty of interpolation points. The authors name SKI configured to use Kronecker or Toeplitz structure in combination with sparse interpolation **KISS-GP**, as the interpolated kernel is used in Gaussian Process Regression.

The SKI approach which makes use of structure in the data and in the kernel (such in the "full" Kronecker or Toeplitz methods) is able to partially lift the "curse of dimensionality", since inference operations are carried out dimension-wise. In particular, such approach can benefit as well from early stopped Conjugate Gradient solves, see [60].

Chapter 4

Longstaff-Schwartz algorithm and Option evaluation

In this section the basics behind the Monte Carlo methods and options are introduced, as well as rudiments of option pricing, in order to explain the Longstaff-Schwartz algorithm.

4.1 Options

Options are financial instruments which are members of the more general family of **derivatives**, which as the name implies, derive their value on some underlying asset. For the sake of completeness and for the unfamiliar reader, we give below the definition of option in its most general form and a summary of the most important results in the field of option pricing:

Definition 4.1.1 (Option [29])**.** An option is the right (but not the obligation) to buy or to sell one unit of one (or more) risky asset(s) at a prespecified fixed price within a specified period.

As one can imagine, different constructions are possible, i.e. different types of options arise depending on the type and number of underlying assets, the definition of time period, and so on. In practice however, two very simple options are often encountered, namely the **call** and **put** options, whose definitions are given below:

Definition 4.1.2 (Call option [29])**.** An call option is the right (but not the obligation) to buy one unit of a risky asset S at a prespecified fixed price K within a specified period. The claim of a call option is given by:

$$\psi(S) = \max(S - K, 0). \tag{4.1}$$

Definition 4.1.3 (Put option [29])**.** An put option is the right (but not the obligation) to sell one unit of a risky asset S at a prespecified fixed price K within a specified period. The claim of a put option is given by:

$$\psi(S) = \max(K - S, 0). \tag{4.2}$$

The "prespecified fixed price" is usually called **strike** price. Assigning a value to options is called **pricing**. Option pricing is in practice done by using the **Black-Scholes** model [2], or on improved models which build on it. In the Black-Scholes model we assume the existence of a risk-free asset B and a stock S whose dynamics are described as follows:

$$dB_t = rB_t dt, \tag{4.3}$$
$$dS = \mu S_t dt + \sigma S_t dW_t. \tag{4.4}$$

Here the dynamics of the risk-free asset are described by a simple differential equation, with B driven by r, known as the **risk-free** rate, while the one of the stock are described by the stochastic differential equation of the Geometric Brownian Motion (GBM), described by $\mu \in \mathbb{R}$, the drift, and $\sigma \in \mathbb{R}^+$, the volatility. W_t here indicates the Brownian or Wiener increment at time t, s.t. $W_t - W_s \sim \mathcal{N}(0, t-s) \ : \ \forall s, t \ : \ s < t$. The model is assumed to be valid in $t \in [0, T]$, where T is the **maturity**, and the information about the state of the market is contained in the natural filtration $\mathcal{F}_t$ generated by the GBM.

The most important result arising from said model is given by the **Black-Scholes-Merton theorem** which we report below:

Theorem 4.1.1 (Black-Scholes-Merton theorem)**.** In the Black-Scholes-Merton model, any option ψ, which is a non-negative $\mathcal{F}_t$-measurable random variable, square integrable under the risk-neutral probability measure $\mathbb{Q}$, is replicable by means of an admissible strategy, and the value at time t of any replicating portfolio is equal to:

$$V_t = \mathbb{E}_{\mathbb{Q}}\left(e^{-r(T-t)}\psi|\mathcal{F}_t\right). \tag{4.5}$$

This gives us a practical tool to price options, which is in fact easily extendable to other types of models. Here $\mathbb{Q}$ is the risk-neutral measure, a probability measure such that the $\frac{S_t}{B_t}$ is a Martingale, i.e.

$$\mathbb{E}_{\mathbb{Q}}\left(\frac{S_t}{B_t}|\mathcal{F}_s\right) = \frac{S_s}{B_s}, \quad \forall s : s \leq t, \tag{4.6}$$

$$\mathbb{E}_{\mathbf{Q}}\left(|S_t|\right) < \infty, \quad \forall t. \tag{4.7}$$

The existence and the uniqueness of $\mathbb{Q}$ are a consequence of the **fundamental theorems of asset pricing**, see for instance [29].

Depending on the allowed exercise times, different options types arise. Commonly encountered options are:

1. **European options** allow only one exercise time $t = T$, the option maturity.
2. **American options** allow the holder to exercise $\forall t \in [0, T]$.
3. **Bermudan options** allow exercise at a set of specified exercise times $\{t_0, ..., t_n\}$ such that $t_0 < t_1 < ... < t_n = T$.

At first glance, the Bermudan option is similar to both the European and American options. As we will see later in fact, this similarity allows the pricing of American options by means of a Bermudan option with many exercise opportunities, highlighting the importance of this thesis work.

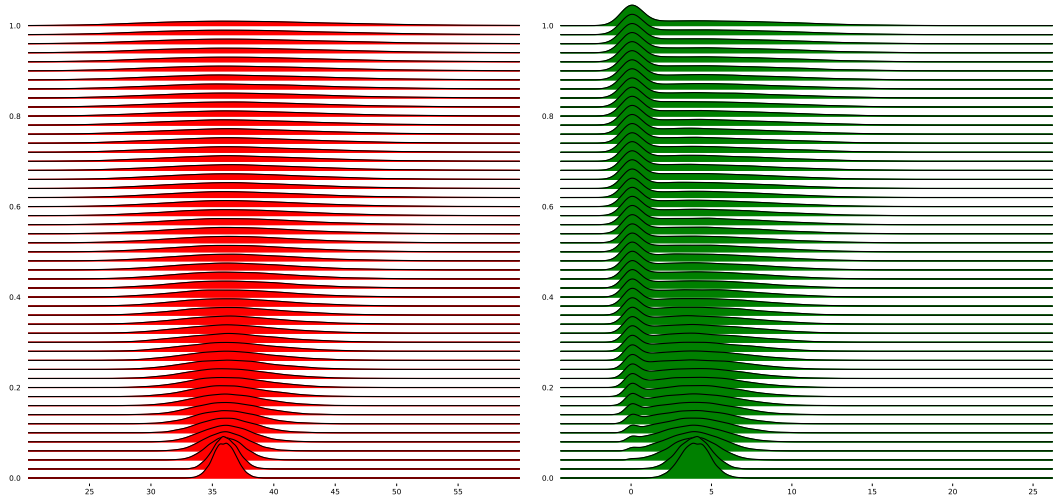
European options

European options are arguably the simplest type of options: the holder is allowed to exercise at one time, in which $t = T$. Let us formally define an European option:

Definition 4.1.4 (European option)**.** A European option consists of a $\mathcal{F}_t$ measurable payout process terminal payment $\psi_T \geq 0$ at time $t = T$ such that:

$$\mathbb{E}_{\mathbb{Q}}\left((\psi_T)^{\mu}\right) < \infty \text{ for some } \mu > 1. \tag{4.8}$$

Pricing such option is done using Theorem 4.1.1. In case of European options, it is often not possible to have analytical or closed for pricing. A notable exceptions is given by the pricing of European calls and puts options valued under the Black-Scholes model, see Subsections A.3 and A.3 in Appendix A. In general however European options are priced either by solving the Partial Differential Equation associated with the pricing problem (see Feynmann-Kac theorem, for instance in [36]), or by simulating the underlying stock and determining the option price by approximating the conditional expectation under risk neutral measure in Equation 4.5 either by trees (e.g. bynomial trees) or Monte Carlo simulations. An example of a Monte Carlo simulation used to price a European Put option is given below in Figures 4.1a and 4.1b for the simulation of the stock value and the values of European puts respectively at different times.



(A) Distribution over time of stock following GBM with $S_{t_0} = 36$, $r = 0.06$ and $\sigma = 0.2$.

(B) Distribution of the value of a European put option with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $K = 40$ and maturities $\mathcal{T} = \{1, ..., 10\}$.

American options

Unlike European options, American options do not have closed form solution for pricing as American options allow to exercise at every moment in time before expiration. We first of all give a definition of American option:

Definition 4.1.5 (American option)**.** An American option consists of a $\mathcal{F}_t$ measurable stochastic process $(\psi_t)_{t\in[0,T]}$ with $\psi_t \geq 0$ and a final payment ψ_τ at the exercise time $\tau \in [0, T]$ chosen by the holder of the option. Furthermore τ is assumed to be a stopping time, ψ_t is assumed to posses continuous paths and that

$$\mathbb{E}_{\mathbb{Q}}\left(\sup_{\tau\in[0,T]}(\psi_\tau)^\mu\right) < \infty \text{ for some } \mu > 1. \tag{4.9}$$

Since the buyer of a American (or Bermudan) option is allowed to choose when to exercise, the exact time of the payment is not known in advance to the seller. However for each fixed exercise strategy, the payment ψ_τ, where τ is a stopping time in $[0, T]$, is uniquely determined. Intuitively we can therefore imagine that the fair price is given by following the exercise strategy which maximises the fair price in Equation 4.5, see [36] for a more rigorous argumentation, which is beyond the scope if this work. We hence state the result regarding the fair price of the American option.

Theorem 4.1.2 (Fair price of an American option)**.** The fair price of an American option ψ is given by:

$$V_{t_0} = \sup_{\tau\in[0,T]} \mathbb{E}_{\mathbb{Q}}\left(e^{-r\tau}\psi_\tau|\mathcal{F}_{t_0}\right), \tag{4.10}$$

and there exists a stopping time $\tau^* \in [0, T]$ such that the supremum will be attained for the hedging strategy corresponding to τ^*.

Note that showing the existence of the optimal stopping time τ and the form of the valuation process is involved. However once can see quite easily that if the price of the American option differs from its fair price, an arbitrage opportunity arises. Furthermore by Theorem 4.1.2 we can see that the optimal strategy is to exercise the American option at the first time τ^* when B_{τ^*} coincides with the option price. However there is no explicit formulation for τ unless one consider the Perpetual American Put as approximation, meaning that computational methods are needed for their pricing.

Bermudan options

Bermudan options can be seen as a middle ground between European and American options, as apart from allowing the holder to exercise at maturity, they allow other finite exercise opportunities.

Definition 4.1.6 (Bermudan option)**.** Consider the time instants such that $t_0 < t_1 < ... < t_n = T$, and where $\mathcal{T} = \{t_0, ..., t_n\}$ denotes all possible exercise moments. A Bermudan option consists of a set of $\mathcal{F}_t$ measurable random variables $\psi_t \geq 0$ and a final payment ψ_τ at the exercise time $\tau \in \mathcal{T}$. τ is assumed to be a stopping time and furthermore

$$\mathbb{E}_{\mathbb{Q}}\left(\sup_{\tau \in \mathcal{T}} (B_\tau)^\mu\right) < \infty \text{ for some } \mu > 1. \tag{4.11}$$

Similarly to the American, we state the corresponding fair price theorem and the existence of an optimal strategy.

Theorem 4.1.3 (Fair price of a Bermudan option)**.** The fair price V_{t_0} of a Bermudan option ψ is given by:

$$V_{t_0} = \sup_{\tau \in \mathcal{T}} \mathbb{E}_{\mathbb{Q}}\left(e^{-r\tau}\psi_\tau | \mathcal{F}_{t_0}\right), \tag{4.12}$$

where $\mathcal{T} = \{t_0, ..., t_n = T\}$ is the set of possible exercise moments and, and there exists a stopping time τ^* such that the supremum will be attained for the hedging strategy π^* corresponding to τ^*.

So in order to price American or Bermudan options with Monte Carlo methods it is not only necessary to generate a number of paths, but also to determine when it is optimal for the holder to exercise. Furthermore in practice the pricing of American options is done by approximating them with their Bermudan counterpart, under the condition that $\max_i (t_{i+1} - t_i) \to 0$. The goodness of this approximation is analyzed in [22] and [23]. Different approaches to do so are viable, one of them is the Longstaff-Schwartz algorithm, which is based on the Monte Carlo method.

4.2 Monte Carlo methods

Monte Carlo methods are a class of algorithms which rely on repeated sampling of random variables in order to obtain numerical results. They are most useful when it is difficult or impossible to rely on other approaches, which is often the case when pricing exotic options.

The main idea here is to approximate the expected value by an arithmetic average of the results of a number of independent experiments which have the same distribution. The basis of the method is the strong law of large numbers [36][44].

Theorem 4.2.1 (Strong law of large numbers)**.** Let $(X^m)_{m \in \mathbb{N}}$ be a sequence of integrable, real-valued random variables that are independent, identically distributed and defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Then, we have almost-surely:

$$\frac{1}{m}\sum_{j=1}^{m} X^j(\omega) \to \mathbb{E}\left(X^j\right) \quad \text{for } m \to \infty. \tag{4.13}$$

This bring us to the (crude) Monte Carlo method:

Algorithm 3 (Crude) Monte Carlo method

Generate a sequence $(X^m)_{m \in \mathbb{N}}$ of integrable, real-valued random variables that are independent, identically distributed and defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$.
Approximate $\mathbb{E}(X) \approx \frac{1}{m}\sum_{j=1}^{m} X^j(\omega)$.

This method is referred as crude since there are many improvements that can be made when dealing with the error. Given the assumption of finite variance, one can get idea about the error one by measuring the sampled standard deviation. This is possible as for the central limit theorem:

Theorem 4.2.2 (Central limit theorem)**.** Let $(X^m)_{m\in\mathbb{N}}$ be a sequence of independent real-valued random variables that are identically distributed with mean μ and finite variance σ^2 and are defined on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$. Then, the normalized and centralized sum of these random variables converges in distribution towards the standard normal distribution, i.e. we have:

$$\frac{\sum_{j=1}^{m} X^j - m\mu}{\sqrt{m}\sigma} \to \mathcal{N}(0,\, 1) \quad \text{as } m \to \infty. \tag{4.14}$$

So for large m we see that the standard deviation of the error is of order $\mathcal{O}\left(\frac{1}{\sqrt{m}}\right)$. The simple idea behind the Monte Carlo method is what lies behind the Longstaff-Schwartz method for the pricing of Bermudan (and American) options.

4.3 Longstaff-Schwartz algorithm

The **Longstaff-Schwartz** algorithm [12], also known as **least-squares method** (LSM) is the most popular algorithm used in real-life applications when it comes to pricing Bermudan and American options with more than one underlying [36]. When dealing with one underlying, binomial methods or finite difference methods perform better, but become impractical from a computational point of view as the number of underlying increases [36]. The algorithm makes use of **backward dynamic programming** by which the optimal stopping time τ^* is determined for each path by starting at $t = T$ and progressing backwards.

To elucidate the algorithm we introduce again the set of exercise times $\mathcal{T} = \{t_0, ..., t_n\}$ and associated state variable $\mathcal{X} = \{X_0, ..., X_n\}$, such that $X_i \in \mathbb{R}^m$, where m is the number of generated paths.

Now consider one realized path j. Since the optimal exercise strategy is not known a priori, we start from the time of expiration $t = T$ and progress backwards in time. At each step we evaluate whether it is optimal to exercise or to wait.

At $t = t_n = T$ if the holder has still the opportunity to exercise, he should do so, since the intrinsic value $\psi_{T,j}$ of the option is non-negative. The value of the option is thus equal to its intrinsic value, i.e. $V^j_{t_n} = e^{-rt_n}\psi^j_{t_n}$. Going back one step at $t = t_{n-1}$, the holder needs to evaluate whether it is optimal to exercise and receive $\psi^j_{t_{n-1}}$ or wait. The value of keeping the option is in such case $\mathbb{E}_{\mathbb{Q}}\left(e^{-rt_{n-1}}B^j_{t_n}|\mathcal{F}_{t_n}\right)$, hence the holder needs to exercise if $e^{-rt_{n-1}}\psi^j_{tn-1} \geq \mathbb{E}_{\mathbb{Q}}\left(e^{-rt_n}\psi^j_{t_n}|\mathcal{F}_{t_{n-1}}\right)$, in other words if $V^j_{t_{n-1}} \geq \mathbb{E}_{\mathbb{Q}}\left(V^j_{t_n}|\mathcal{F}_{t_n-1}\right)$. Depending on this, the optimal exercise time τ^* conditioned on not having exercised before t_{n-1} and on X^j_{tn-1} equals t_n or t_{n-1}. The same decision making process is applied to all paths j, and then recursively to the previous time steps i. This decision making process can be described as follows in Algorithm 4, where $\tau^j_i = 1$ indicates that the optional exercise at time t_i for paths j is to exercise:

Algorithm 4 Backward dynamic programming for determining the exercise strategy of Bermudan options

Set $\mathcal{T} = \{t_0, ..., t_n\}$
Generate $\mathcal{X} = \{X_0, ..., X_n\}$, with $X_i \in \mathbb{R}^m$
Initialize exercise time vector $\mathrm{T} = \{\tau_1, ..., \tau_n\}$, with $\tau_i \in \mathbb{R}^m$ and $\tau_n = 1$ for all paths j
Set $V_{t_n}^j = e^{-rt_n}\psi\left(X_t^j\right)$ for all paths j
for $i \in \{n, n-1, ..., 1\}$ **do**
  for $j \in \{1, ..., m\}$ **do**
    Determine the optimal conditional exercise strategy τ_i^j by:

$$\tau_i^j = \begin{cases} \tau_i^j, & \text{if } V_{t_i}^j \geq \mathbb{E}\left(V_{\tau_{i+1}}^j | \mathcal{F}_{t_i}\right) \\ \tau_{i+1}^j, & \text{else} \end{cases} . \tag{4.15}$$

  end for
end for

In order to use this algorithm in practice, a reliable way of computing the expectation in Algorithm 4 is needed, also known as **continuation value**. The idea behind Longstaff-Schwartz is to approximate the continuation value as a linear combination of L basis functions:

$$\mathcal{C}_{\tau_i} := \mathbb{E}\left(V_{\tau_{i+1}} | \mathcal{F}_{t_i}\right) \approx \sum_{k=0}^{L} w_{i,k}\phi_k(X_{t_i}), \tag{4.16}$$

where X_i is the state process at the i-th time-step, ϕ_k is the k-th basis function and $w_{i,k}$ is its associated weight corresponding to time i. The justification behind the approach is related to the definition of conditional expectation as projection.

Theorem 4.3.1 (Conditional expectation as projection)**.** Let $(\Sigma, \mathcal{F}, \mathbb{P})$ be a probability space, let $\tilde{\mathcal{F}} \subset \mathcal{F}$ be a sub σ-algebra of $\mathcal{F}$ and let X be a L^2 random variable. Then $\mathbb{E}\left(X|\tilde{\mathcal{F}}\right)$ is the projection of $X \in L^2\left(\mathcal{F}\right)$ onto $L^2\left(\tilde{\mathcal{F}}\right)$.

By Theorem 4.3.1, an estimate of $w_{i,k}$ can be obtained by minimizing the L^2 distance between our variates and regressor function by solving the following optimization problem:

$$w_i^* = \underset{w_i}{\operatorname{argmin}} \left(\mathbb{E}_{\mathbb{Q}}\left(V_{\tau_{i+1}} | X_{t_i}\right) - \sum_{j=0}^{w} {}_{i,k}\phi_k(X_{t_i}) \right)^2 . \tag{4.17}$$

The solution of the linear regression problem is the optimal coefficient vector $w_0^* = \left(w_{i,1}^*, ..., w_{i,N}^*\right)$ such that:

$$w_i^* = \left(H_i^T H_i\right)^{-1} H_i^T y_i, \tag{4.18}$$

where H_i is the design matrix with $H_i(j,k) = \phi_k\left(X_{t_i,j}\right)$ and y_i is the vector $y_{i,j} = \mathbb{E}\left(V_{\tau_{i+1}^j} | X_{t_i}\right)$.

The above regression-based method can be improved by considering the stopping time of each path, enabling to set up an interleaving mechanism over the time levels for comparing the cash flows. This gives rise to the Longstaff-Schwartz Algorithm 5, where only points in-the-money enter the regression:

Algorithm 5 Longstaff-Schwartz for calculating the price in Bermudan options

Set $\mathcal{T} = \{t_0, ..., t_n\}$
Generate m independent paths $\mathcal{X} = \{X_0, ..., X_n\}$, with $X_i \in \mathbb{R}^m$
Set $V_{t_n}^j = e^{-rt_n}\psi\left(X_t^j\right)$ for all paths j
for $i \in \{N-1, ..., 1\}$ **do**
Solve the regression problem in Equation 4.17 i.e. compute optimal weights by:

$$w_i^* = \left(H_i^T H_i\right)^{-1} H_i^T y_i. \tag{4.19}$$

Compute the estimates of the continuation values for all paths j by:

$$\mathcal{C}_{t_i} := \mathbb{E}_{\mathbb{Q}}\left(V_{t_{i+1}}|\mathcal{F}_{t_i}\right) \approx \mathbb{E}_{\mathbb{Q}}\left(V_{t_{i+1}}|\left\{X_{t_i}^j\right\}_{j=1}^{j=m} : \forall j : \psi\left(X_{t_i}^j\right) > 0\right) \approx \sum_{k=0}^{L} w_{i,k}^* \phi_j(X_{t_i}). \tag{4.20}$$

Determine the optimal conditional exercise strategy for all paths j by:

$$V_{t_i}^j = \begin{cases} \psi\left(X_{t_i}^j\right), & \text{if} \quad \psi\left(X_{t_i}^j\right) \geq \mathcal{C}_{t_i}^j \\ V_{t_{i+1}}^j, & \text{else} \end{cases}. \tag{4.21}$$

end for
Set $V_{t_0} = \max\left(\frac{1}{m}\sum_{j=1}^m V_{t_1}^j, \psi\left(S_{t_0}\right)\right)$.

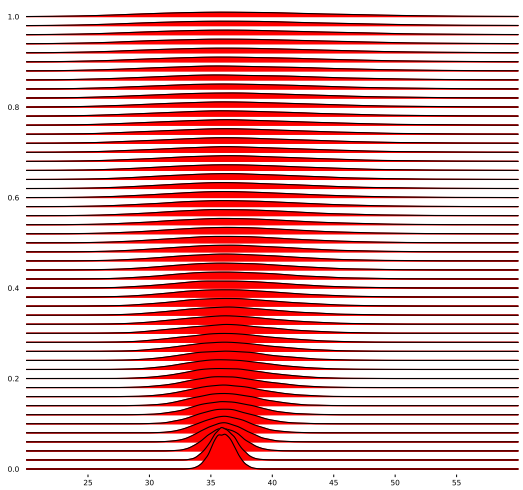
The last step is necessary as there cannot be an exercise opportunity at the first time-step t_0. The alternative is to perform regression over the all paths, essentially substituting Equation 4.20 with:

$$\mathcal{C}_{t_i} := \mathbb{E}_{\mathbb{Q}}\left(V_{t_{i+1}}|\mathcal{F}_{t_i}\right) \approx \mathbb{E}_{\mathbb{Q}}\left(V_{t_{i+1}}|X_{t_i}\right), \tag{4.22}$$

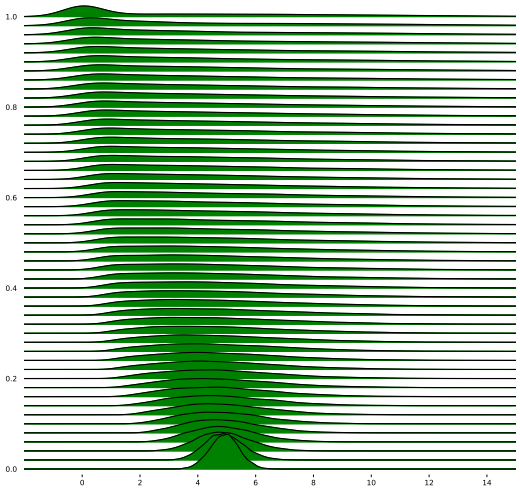
and Equation 4.21 with:

$$V_{t_i}^j = \max\left(\psi\left(X_{t_i}^j\right), \mathcal{C}_{t_i}^j\right). \tag{4.23}$$

An example of pricing a simple Bermudan put with a the LSM algorithm is given below. In Figure 4.2a we can see the distribution over time of the considered stock, while on the right in Figure 4.2b we can see the distribution of the value of the option evolve (backwards) in time: at the last exercise opportunity at $t = 1$ the distribution is quite flat accounting for the uncertainty in the underlying stock price, while progressing to earlier times the unbiased option price is recovered by means of the stock model.



(A) Distribution over time of stock following GBM with $S_{t_0} = 36$, $r = 0.06$ and $\sigma = 0.2$.



(B) Distribution of the value of a Bermudan put option with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $K = 40$ and exercise possibilities $\mathcal{T} = \{0, ..., 10\}$.

There are several things that need to be addressed when it comes to Algorithm 5.

Convergence

There are two sources of error in the LSM algorithm, namely the error in the Monte Carlo simulation consisting in a time discretization error and a truncation error, and the discretization error in the approximation of the continuation value. The convergence of the LSM estimate was first analyzed by Longstaff and Schwartz themselves in [12] in case of geometric Brownian motion paths. More general rigorous results regarding convergence can be found in [14]. Therein the authors have firstly introduced:

$$V_{t_0}^{L} = \sup_{\tau \text{ in } \mathcal{S}\{\phi_1,...\phi_L\}} \mathbb{E}_{\mathbb{Q}} \left(e^{-r\tau} \psi_\tau \right), \tag{4.24}$$

where the set $\mathcal{S}\{\phi_1, ...\phi_L\}$ contains only exercise strategies based on the solution of the regression problem with basis functions $\phi_1, ..., \phi_L$. Secondly the authors introduced $V_{t_0}^{L,m}$ which equals the Longstaff-Schwartz estimate as computed above in the LSM algorithm 5, i.e. with m simulated paths and L basis functions. Under some technical conditions, among which completeness of the basis function set, it is proven that with a growing number L of basis functions the approximating option price $V_{t_0}^{L}$ converges to the real option price with a convergence rate of $\mathcal{O}\left(\frac{1}{\sqrt{m}}\right)$ [14], i.e.

$$V_{t_0}^{L} \to V_{t_0} \quad \text{as} \quad L \to \infty. \tag{4.25}$$

The authors also prove that with a growing number of simulated paths, the Longstaff-Schwartz estimate $V_{t_0}^{L,m}$ converges almost surely towards the approximating option price $V_{t_0}^{L}$, i.e.

$$V_{0}^{L,m} \to V_{t_0}^{L} \quad \text{almost surely as} \quad L \to \infty. \tag{4.26}$$

Hence for a fixed number L of basis functions the LSM algorithm converges to the solution of the optimal stopping problem of Equation 4.25, and not to the option price. It has also been shown in [19] that the number of paths required for convergence grows exponentially with the order of polynomials used. These results are important, however the practical question regarding how many, and more importantly which, basis function to choose remains open. In fact, the complexity of the model can be addressed efficiently when approaching the problem from an **empirical risk minimization** point and choosing a model which minimizes the **true risk** (generalization error) by means of cross-validation [39]. However in financial literature and practice the choice of basis function and their number seems often to be based on experience or trial-and-error.

Basis functions

The parametric estimate which is the core of the Longstaff-Schwartz algorithm, is greatly affected by the choice of basis functions. In their seminal paper [12], Longstaff and Schwartz made use of weighted Laguerre polynomials, however nothing prohibits another choice. Some have argued in favour of simple polynomials, power series, Legendre polynomials, Hermite polynomials, etc.

The choice of basis functions, explanatory variables and their number has serious repercussion on the results due to the danger of overfitting and are in general strongly dependent on the underlying process and payoff function. This is why some have rather arbitrarily suggested to use only polynomial terms up to second degree of meaningful financial variables that drive future exercise [17]. Other have argued instead of using monomial basis functions up to the third order, possibly including the payoff function [44].

As stated previously, pricing of American options can be done using the LSM algorithm by approximating them with their Bermudan counterpart, under the condition that $\max_i (t_{i+1} - t_i) \to 0$. The goodness of this approximation is analyzed in [22] and [23], having as result that the L^p error bounds are of order $\mathcal{O}\left(\frac{1}{\sqrt{n}}\right)$, where n is the number of time-steps.

Improvements

The basic LSM algorithm performs the regression only on the paths that are in-the-money, since early exercise is only relevant when the option is in the-money. This was firstly suggested by Longstaff and Schwartz themselves [12]. While this does improve the estimate, it also introduces the problem of having possibly fewer paths to regress. When so, it is common practice to not early exercise.

To lower the bias it has been suggested in [19] to perform first the optimization of the exercise strategy and then simulate another set of independent, identically distributed paths to price the option. In practice however, the bias of the LSM algorithm is already low [44].

If the value of the corresponding European option is available at exercise, it can be used as lower bound for the continuation value [44]. One should therefore early exercise only if the intrinsic value is above both the computed continuation value and European value. The European value, if available, can also be used as control variate as done in [25] and [63].

Lastly, the parametric regression step can be substituted by a **non-parametric** regression step, rendering thus model selection way easier. For Gaussian Process Regression, this was done recently in [62] and [63].

In [62], the authors introduce Gaussian Process Regression as a substitute for the linear model giving rise to the least-squares problem of Equation 4.18. Furthermore they introduce **batched** Gaussian Process Regression in order to account for the heteroscedasticity of the noise present in the realizations of the process characterizing the continuation value. As will be shown, the associated noise violates the Gaussian assumption due to the non-negativity of the continuation value. In this context, for every realization of a simulated path, a set of n' i.d.d. replicating paths is generated. For each realization x, the corresponding sample mean and sample standard deviation of the continuation value are computed. Considering a total of m generated paths, the regression step takes place using the $\frac{m}{m'}$ couples consisting of the realizations of the underlying and the batch-average of its corresponding continuation values and the obtained estimate of the noise. Apart from accounting for heteroscedasticity, this approach offers various benefits such as a reduced computation complexity amounting to $\mathcal{O}\left(\left(\frac{m}{m'}\right)^3\right)$, a better signal-to-noise ratio $\mathbb{E}\left(\epsilon^2(x)\right) = \frac{\sigma^2(x)}{m'}$ resulting in a smoother marginalized log-likelihood and with less local maxima. The author also analyzes some sparse approximations to Gaussian Process Regression in order to alleviate the computational burden even further by limiting the number of paths to be included in the regression step. The numerical results show that the batching can speed up the pricing and improve the accuracy of the results.

In [63], the authors improve the work of [62] by using the value of the European counterpart of the option in consideration as control variate, based on the works of [25]. They consider homoscedastic Gaussian noise in the regression step, which is estimated jointly

together with the other hyperparametrs by marginal log-likelihood maximization on a randomly chosen control set. The reduced variance estimated continuation value is then given by:

$$\mathcal{C}_{t_{i-1}} = \mathcal{P}_{t_{i-1}}\left(V_{t_i}\right) - \bar{\eta}_{t_{i-1}}\left(\mathcal{P}_{t_{i-1}}\left(Y_{t_i}\right) - \mathbb{E}\left(\psi\left(X_{t_n}\right)|X_{t_{i-1}}\right)\right), \tag{4.27}$$

where $\mathcal{P}_{t_{i-1}}(\cdot)$ is the predictive mean of the Gaussian Process Regression model at $\mathcal{F}_{t_{i-1}}$ and $\bar{\eta}_{t_{i-1}}$ is chosen such to minimize the variance of $\mathcal{C}_{t_{i-1}}$, resulting in:

$$\bar{\eta}_{t_{i-1}} = \frac{\mathcal{P}_{t_{i-1}}\left(V_{t_i}Y_{t_i}\right) - \mathcal{P}_{t_{i-1}}\left(V_{t_i}\right)\mathcal{P}_{t_{i-1}}\left(Y_{t_i}\right)}{\mathcal{P}_{t_{i-1}}\left(Y_{t_i}^2\right) - \mathcal{P}_{t_{i-1}}\left(Y_{t_i}\right)^2}. \tag{4.28}$$

The control variates are adjusted as follows:

$$\begin{cases} Y_{t_{i-1}} = \mathbb{E}_Q\left(\psi\left(X_{t_n}\right)|X_{t_{i-1}}\right) & \text{if, } \psi\left(X_{t_{i-1}}\right) > \mathcal{C}_{t_{i-1}} \\ Y_{t_{i-1}} = Y_{t_i}, \quad \text{else} \end{cases}, \tag{4.29}$$

and initialized as $Y_{t_i} = \psi\left(X_{t_i}\right)$.

The numerical results show that Gaussian Process Regression can be successfully employed in the estimation of the continuation value in the setting of Equation 4.18, consistently outperforming the traditional Longstaff-Schwartz method. Adding control variates further improves the results at a non-negligible computational overhead keeping in mind that posterior mean estimates scale with $\mathcal{O}\left(m^2\right)$, m being the number of paths under consideration.

Based on these results, it seems a sensible idea to better investigate the effects of using sparse Gaussian Process Regression models instead of the full GPR in order to reduce overall computational cost, possibly combining it with batching, in order for even greater computational savings to be made. To do this the examined sparse approximations to Gaussian Process Regressions are compared in different scenarios having the form of numerical experiments. The methods which are best fit to the application to the LSM algorithms are then compared in the LSM setting.

Chapter 5

Numerical results

In this chapter the various approximations presented so far are tested. The examined approximations, except for SKI, as well as the full method are implemented in Python using the NumPy [28] scientific computing library as well as TensorFlow [49], both free to use and open source. The TensorFlow implementation allows, besides performance gains obtainable by GPU computing, to compute the gradient of the marginalized log-likelihood function by automatic differentiation at little computational cost, reducing further the "training" time. SKI is adopted from the GPyTorch [59] package, based on the deep learning package PyTorch [59], similar to TensorFlow, while the implementation of linear regression used is taken from scikit-learn [38]. The code and results shown below can be found in the ABN-AMRO Azure DevOps environment at: `https://dev.azure.com/cbsp-abnamro/FRR-RiskModels/_git/GaussianProcessActiveLearning`.

The choice of hyperparameters and inducing points is carried out by maximizing the log-marginal-likelihood, said optimization being carried out by means of the default optimizer of the SciPy library [11], the quasi-Newton method of Broyden, Fletcher, Goldfarb, and Shanno (BFGS). In case of SKI, the method of gradient ascent is utilized, with learning rate determined by empirical risk minimization on a new validation set. Optimization of the hyperparameters is carried out in $\mathbb{R}^+$ by optimizing with respect to their exponentiated value, and are initialized by sampling from the uniform distribution such that $\theta_i \sim \mathcal{U}\left(10^{-5}, 10\right)$ to in principle minimize relative error. As shown in [57] this choice has however little impact on the final results of the optimization problem. When present, support points are optimized jointly with the hyperparameters, and are initialized uniformly from the training set as recommended by [30]. Five optimization runs are performed, the best is used for inference.

When dealing with linear regression we consider polynomial basis functions, their number being chosen by empirical risk minimization on an i.d.d. generated set of data.

The choice of error metric befalls on the mean-absolute-error and maximum-absolute-error which are easier to interpret than say the means-square-error. Apart from these two error metric the R^2 score is presented for each experiment, which is nonetheless related to the sample variance and thus to the mean-square-error.

A general overview of the various methods is given in the first set of data, the normalized sinc function. Subsequently, the continuation value estimation problem is analyzed by means of examples which allow the determination of the exact continuation value, used as reference. Finally pricing of Bermudan options is performed and compared against the traditional LSM algorithm.

5.1 Sinc function

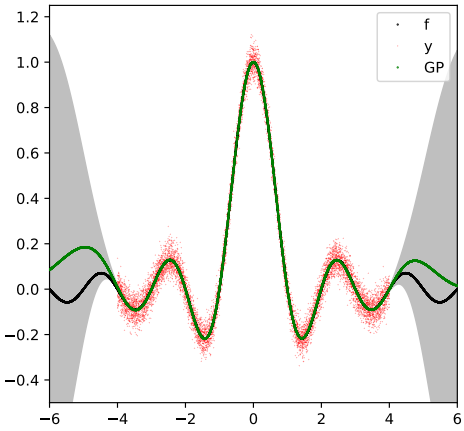
In order to compare the methods which have been brought forth in the previous chapter, a simple and easily observable one dimensional toy problem is generated following the example in [61]. A relatively large (when considering full Gaussian Process Regression) amount of points, 10000, are sampled uniformly across the $[-4, 4]$ interval in order to accentuate the differences in performance of the examined algorithms. The latent function f is chosen to be the normalized sinc function:

$$f(x) = \frac{\sin(\pi x)}{\pi x}. \tag{5.1}$$

Observations are produced by adding homoscedastic noise such that $y = f + \epsilon$, where $\epsilon \sim \mathcal{N}\left(0, 0.04^2\right)$. A plot of the underlying function and its relative observations is given in Figure 5.1a. Subsequently a test set (X, f) of 100000 i.d.d. samples is generated similarly to the training set, which is used to make out-of-sample predictions, i.e. compute the resulting mean-absolute-error, maximum-absolute-error and R^2 score. This is done for a different number of inducing points, which are in this setting selected in both scenarios using a subset-of data method and by jointly optimizing the model log-likelihood together with the hyperparameters, when dealing with sparse approximations. The considered kernel is in all cases the squared exponential kernel of Equation 2.75, namely:

$$k\left(x, x'\right) = \sigma_s^2 e^{-\frac{||x-x'||_2^2}{2l^2}}. \tag{5.2}$$

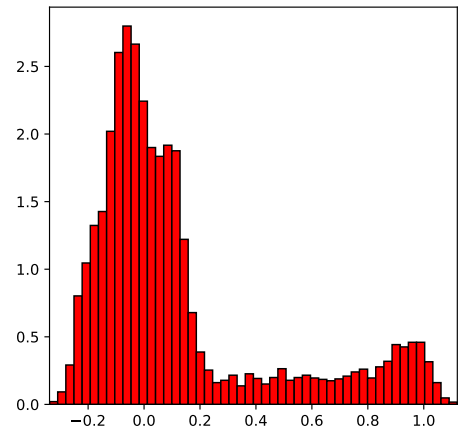
An overview of the set of data is given in Figure 5.1.



(A) Set of data: the red crosses are the data, the black line is the underlying function. The green line is the GPR estimate, with its confidence interval in grey.

	y	X
# observations	10000	10000
Minimum	-0.33	-3.99
Maximum	1.11	3.99
Mean	0.12	-0.02
Variance	0.11	5.36
Skewness	1.47	0.01
Kurtosis	1.09	-1.20

(B) Descriptive statistics.

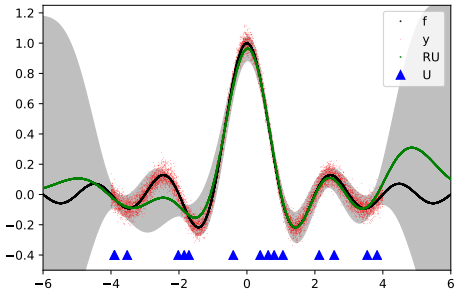


(C) Histogram of the y values.

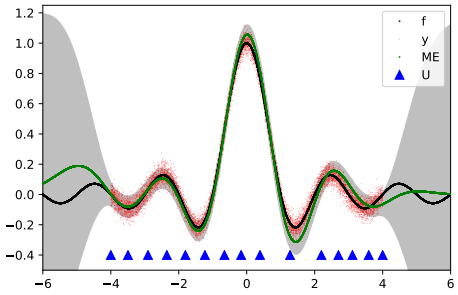
(D) Histogram of the X values.

FIGURE 5.1: Overview of the set of data.

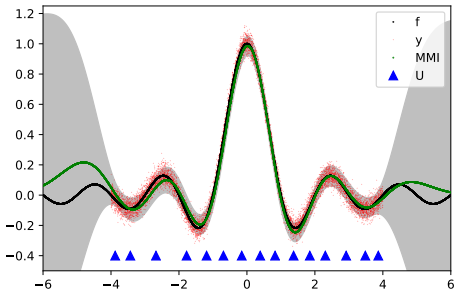
An graphic overview of the various estimates of the examined methods is given below, in Figure 5.2, where in-sample prediction are showed in case of 15 inducing points.



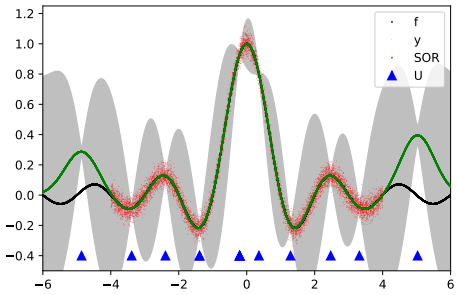
(A) Random sampling.



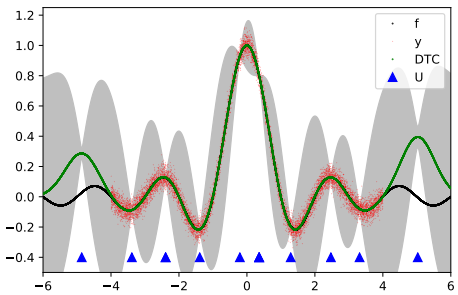
(B) ME sampling.



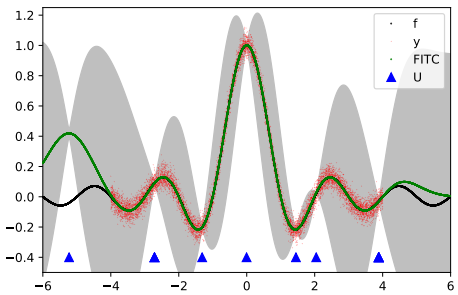
(C) MMI sampling.



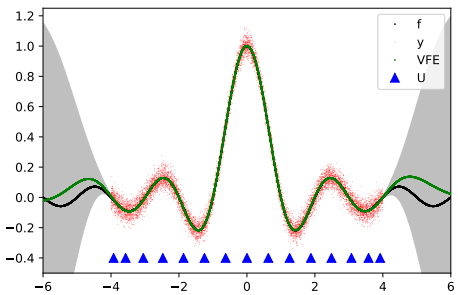
(D) SOR.



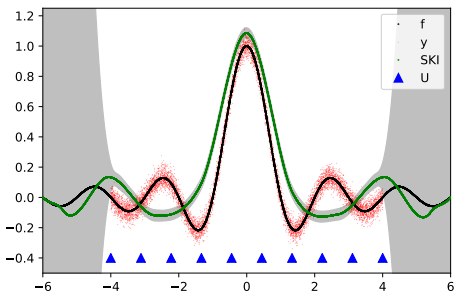
(E) DTC.



(F) FITC.



(G) VFE.



(H) SKI.

FIGURE 5.2: Plots of the various approximations. The black line is the underlying function, the green line is the prediction of the considered approximation, the blue triangles are the locations of the inducing points. The grey shaded area represents the 95% confidence interval.

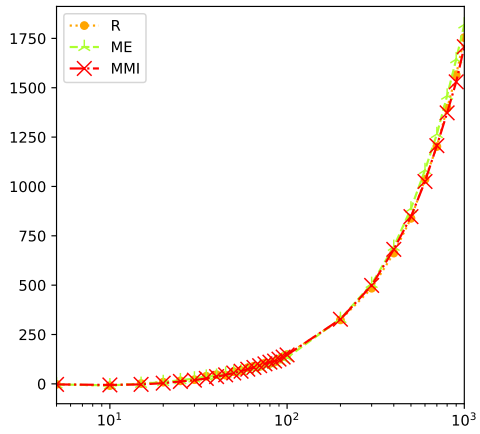
A plot of the in-sample results (of Tables B.1, B.3, B.5, B.7, B.9, B.15, B.11 and B.13 in the Appendix B) is given in the next couple of subsections for all examined methods. In-sample results are displayed as they are related to the continuation value estimation problem of

the Longstaff-Schwartz algorithm. Comparing in-sample results to out-of-sample results (of Tables B.2, B.4, B.6, B.8, B.10, B.16, B.12 and B.14 in the Appendix B), we can however see that in-sample results are worse than their out-of-sample counterpart for most approximations, meaning that the model is able to generalize well.

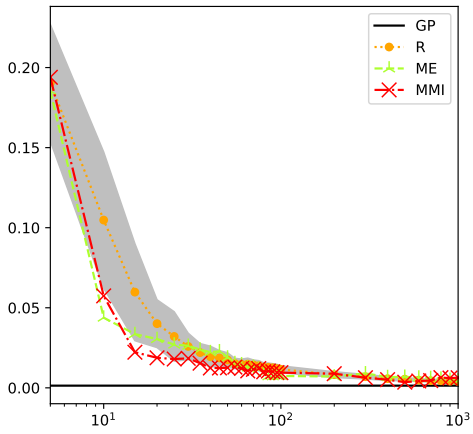
Subset-of-data

The examined subset-of-data approximations are compared among each other and with the full GP. In this setting the hyperparameters are determined on the full set of data for all methods.

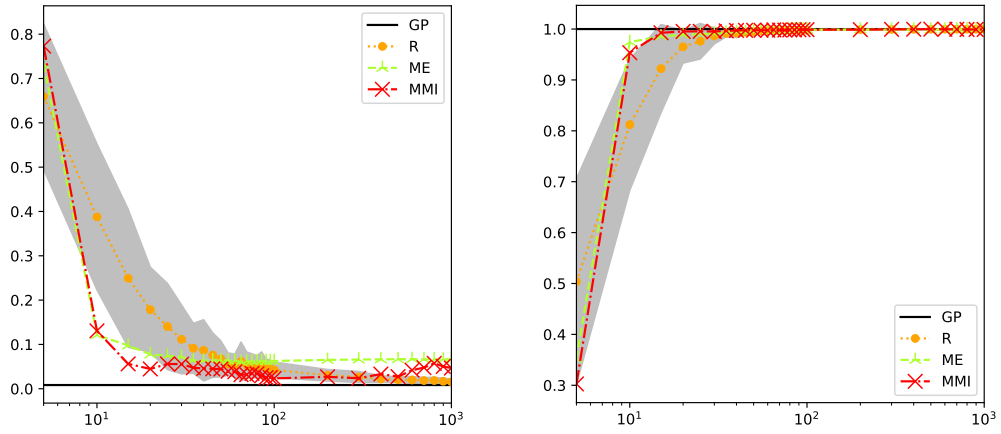
When comparing visually the results of the different SOD methods in Figure 5.2a, Figure 5.2b and Figure 5.2c, we can see that the inducing points chosen with maximum entropy in Figure 5.2b are located (in this case slightly) more towards boundaries of the data set as observed in [33] due to the fact that the posterior covariance is largest further away from sampled locations. These points are likely to waste information, while the ones sampled with the mutual information sampling scheme in Figure 5.2c does not suffer from this, as the rule tries to find the set of points which reduce the entropy over the unseen locations. The 95% confidence region, as well as the predictive posterior mean of all SOD methods appears to be quite faithful with respect to the original model, as the data is quite redundant.



(A) Marginalized log-likelihood.



(B) Mean-absolute-error.



(C) Max-absolute-error. (D) R^2.

FIGURE 5.3: Marginalized log-likelihood and error metric in case of SOD approximations. The grey shaded area represents the 95% confidence interval.

We can see in Figure 5.3a that the marginalized log-likelihood of all methods increases as a function of the number of points in a similar fashion. This is because in this experiment the hyperparameters are chosen to be the ones of the full Gaussian Process Regression model. From the error metrics in Figure 5.3 we can see that maximum entropy and maximum mutual information sampling outperform a full Gaussian Process Regression model built on random sampling for small $M < 100$ for all error metrics. As argued in [61], all methods are thus equivalent for a large number of inducing points, which can also be noted by the shrinking 95% confidence intervals in Figure 5.3. ME and MMI are however significantly better under the $M = 100$ threshold, in particular MMI as point selection is performed based not only on previously selected points. ME is shows similar performance, as it essentially tends to place points equidistantly. ME and MMI however do require an a-priori estimate of the hyperparameters, either on the full set [30] (which is rarely possible) or on a random subset as for instance in [62]. Random sampling does not require this knowledge in advance, and is thus much more easily applicable in practice, see for instance [63].

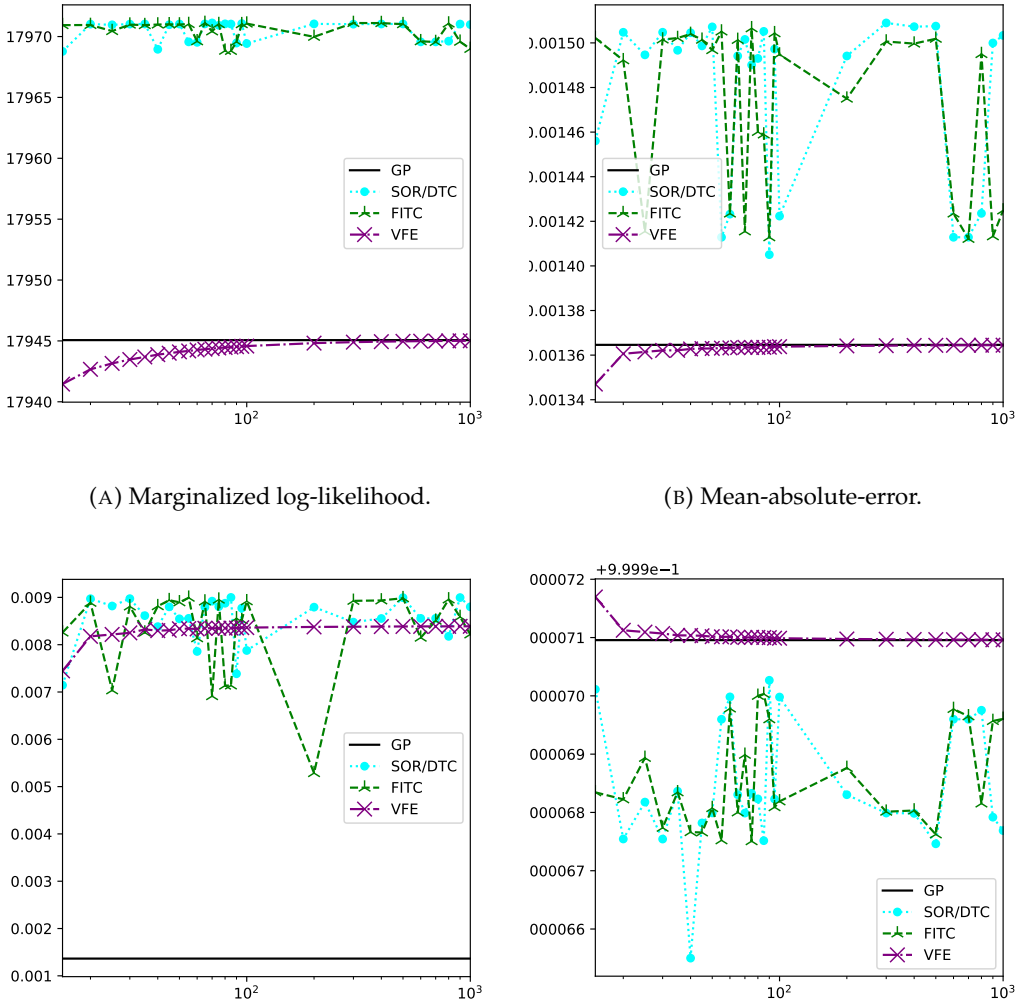
Alternating hyperparameters learning (model exploration) with sampling (model exploitation) is treated for instance in [32] and in [46], who both perform these task in a fully Bayesian framework with a discretized hyperparameter space, requiring an extensive amount of computations, making its implementation far from ideal in practice, especially in a possibly large scenario. In particular the approach of [32] is based on bounds on the disadvantage of greedy sampling which are very loose [51], rendering the results hardly useful.

Sparse approximations

The various sparse approximations are compared among each other for different number of inducing points. Note that in this setting the inducing points are chosen by the maximum log-likelihood optimization and are jointly optimized with the hyperparameters.

The choice of inducing points made by SOR, DTC and FITC appears visually similar in Figure 5.2d, Figure 5.2e and Figure 5.2f (note again that the difference between SOR and DTC lies only in the predictive covariance), as well as their posterior mean and confidence intervals, which do not resemble the ones of the full model in Figure 5.1a. What all these approximations do have in common, is that they tend to place the inducing points in the neighbourhood of inflection points and local minima. It is in these points in fact that the marginalized log-likelihood is more sensitive to change, as a better representation of the underlying kernel functions in these "critical" points leads to a higher likelihood compared

to a more evenly spaced configuration. We do expect such approximations to be prone to overfitting, in particular when dealing with datasets containing little variation.



(A) Marginalized log-likelihood.

(B) Mean-absolute-error.

(C) Max-absolute-error.

(D) R^2.

FIGURE 5.4: Marginalized log-likelihood and error metric in case of sparse approximations.

When examining visually the results presented in Figure 5.5 and considering the resulting locations of the optimized inducing points the variance-reducing effect of the regularizing trace term present in the VFE likelihood appears evident. We can in fact see in Figure 5.4a that for all choices of points, the prior approximations (SOR, DTC and FITC) reach a higher marginalized log-likelihood that both the full model as well as VFE. VFE is here the only model which strives to approximate the posterior of the full model as good as possible, its likelihood converging almost monotonically (due to the varying hyperparameters) to the one of the full model. Apart from having a higher value than the one of the VFE, the marginalized log-likelihoods of the prior approximations do not increase monotonically due to the absence of a regularizing term, rendering these approximations quite unreliable to use in practice, as they render more difficult choosing the number of inducing points M. The treated prior approximations, do hence by construction not try to approximate the posterior but to find a set of inducing points and hyperparameters that maximize their likelihood, which unlike posterior approximations, does not recover the one of the full Gaussian Process Regression of Equation 2.61 when $M \to N$.

When comparing the error metrics (Figure 5.4b, Figure 5.4c and Figure 5.4d), we can see that VFE scores in all cases way better than its counterparts, surpassing the performance in the mean-absolute and max-absolute sense of the full model for small M. In this simple example, its errors do in fact increase as a function of M. This can be again explained by looking at the trace term in Equation 3.55, which tends to shift the inducing points where the difference between the approximated Gram matrices is minimized. When looking at the marginalized log-likelihood and the various error metrics in relation to the other error metrics it can be concluded that all prior approximations are overfitting, as pointed out by [35].

In light of these results if one were to make a choice between the existing sparse approximations to be employed on a large set of data, it would be advisable to use VFE if the aim is to recover the full Gaussian Process Regression model and if the confidence intervals are deemed important. Prior approximations have been shown to not work well and to underperform with respect to the VFE model because in practice when optimizing jointly over the inducing points and hyperparameters, the absence of the regularizing trace term allows placing the inducing points outside of the domain of the training set. Although this behaviour can be corrected by performing constrained optimization (at a higher computational cost), it is problematic since it is a waste of computational resources both at "training" (optimization over inducing points and hyperparameters, and predictive mean precomputations) and at "test", rendering these prior approximation ill-suited for practical use.

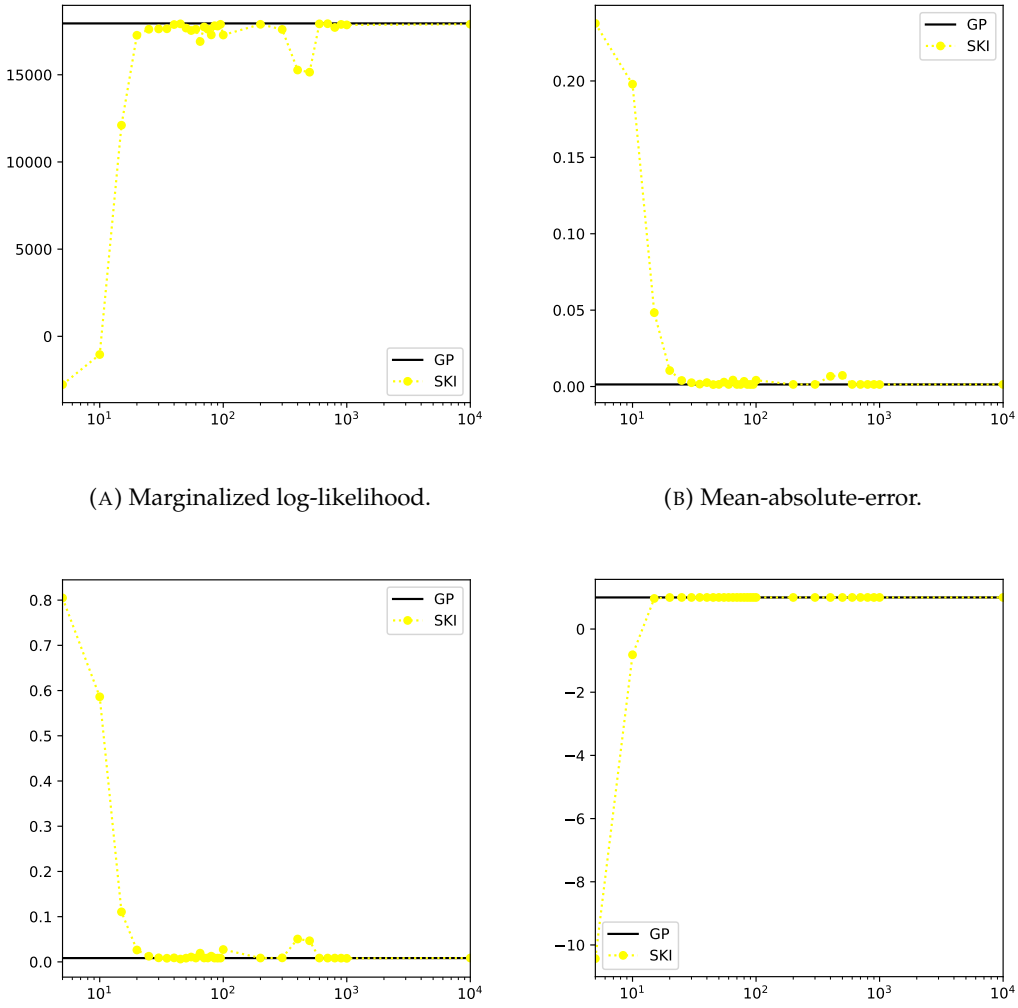
Structured sparse approximations

In this example, as in most sets of data, the Toeplitz and Kronecker methods are not applicable, as the data does not lie on a grid. Therefore, the only structured sparse approximation which is applicable is SKI, which as discussed in the dedicated chapter, places the support points on a grid and exploits the thus generated structure.

The SKI method in this example was "trained" using gradient ascent with learning rate $\alpha = 0.1$ and a maximum of 60 iterations. The number of support points was chosen to reach at maximum $M = N = 10000$ as the number of points in the training set, as suggested by [52]. Note that since the type of interpolation employed in SKI is cubic, the effects of adding a support point are quite different than when considering Nyström-based methods.

From Figure 5.2h we can see the SKI estimate of the underlying function. Due to the fact that the inducing points are evenly spaced, we have worse performance near the inflection points compared to the examined sparse approximation, but we do not expect this approximation to overfit compared to the full GPR. Overall performance seems also worse, most probably due to the fact that cubic interpolation is used instead of the Nyström low-rank form, hence the number of interpolation points is not related to the rank of the approximated Gram matrix.

To confirm this intuition, we show below the results of this particular method and its performance when compared to the full model.



(A) Marginalized log-likelihood.

(B) Mean-absolute-error.

(C) Max-absolute-error.

(D) R^2.

FIGURE 5.5: Marginalized log-likelihood and error metric in case of SKI.

We see in Figure 5.5a that the marginalized log-likelihood of SKI approaches the one of the full method at $M = 30$, since the inducing points are evenly spread over the whole domain. We can observe that due the fact that "training" is performed using gradient descent, the obtained local maxima are in some cases a little off due to the choice of learning rate.

Similar observations hold for the error metrics, as they all approach the baseline of the full GPR, rendering this approximation method interesting for use considering its extremely low computational complexity.

5.2 Continuation value - Bermudan Put

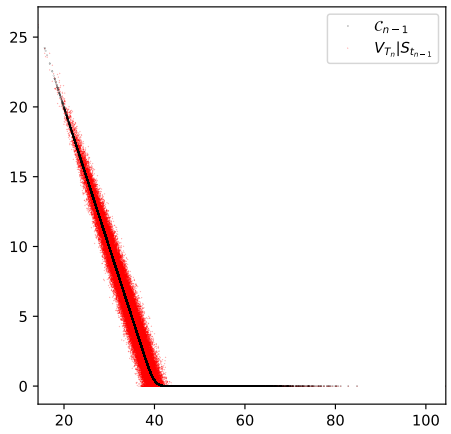
In order to successfully apply Gaussian Process Regression to the Longstaff-Schwartz algorithm for the pricing of Bermudan and American options, it is necessary to ensure that the continuation value is estimated correctly. In the traditional variant of the LSM algorithm (see Algorithm 5), this step is performed by resorting to the use of a linear model, which seen its downsides, namely the choice of basis functions, we wish to replace by GPR. To verify the performance gain obtained by making use of the full Gaussian Process Regression model and its sparse approximation, another test problem is set up in which the continuation value is estimated from simulated data. More specifically, we note that the first step in the Longstaff-Schwartz Algorithm 5 in which the continuation value $\mathcal{C}_{T_{n-1}} = \mathbb{E}_{\mathbb{Q}}\left(V_{t_n} | X_{t_{n-1}}\right)$

is determined coincides with the value at $t = t_{n-1}$ of the related European option having maturity T, as the holder exercises in any case at $t = t_n = T$. Under Geometric Brownian Motion dynamics this is given by the Black-Scholes formula (see Equation A.10 for a European Call option and Equation A.13 for a European Put option). This provides us an exact benchmark to test the working in the Longstaff-Schwartz setting of the various examined methods, which can be considered representative of the whole problem also because the first step has influence on the goodness of the Longstaff-Schwartz estimator, as the continuation value estimation error accumulates as more regressions are performed.

We consider the estimation of the continuation value in the following different contexts, namely the pricing of Bermudan Puts options under Geometric Brownian Motion dynamics with initial stock values $S_{t_0} \in \{36, 38, 40, 42, 44\}$, risk-free rate $r = 0.06$, volatility $\sigma \in \{0.2, 0.4\}$, strike $K = 40$, maturities $T \in \{1, 2\}$ and 50 exercise opportunities per year, so to account for a different level of moneyness.

We benchmark the various considered methods against the estimation of $\mathcal{C}_{n-1} = \mathbb{E}_{\mathbb{Q}}\left(V_{t_n} | X_{t_{n-1}}\right)$ determined path-wise with the Black-Scholes put formula of Equation A.13, i.e. the first step of the LSM algorithm. We consider $N = 100000$ paths in order to provide a realistic benchmark, as done in the LSM paper [12]. To exclude any source of bias relative to the discretization, the paths are sampled from their underlying log-normal distribution, i.e. no discretization is employed, as in [12]. The performance of the selected sparse approximations is compared against the one of linear regression.

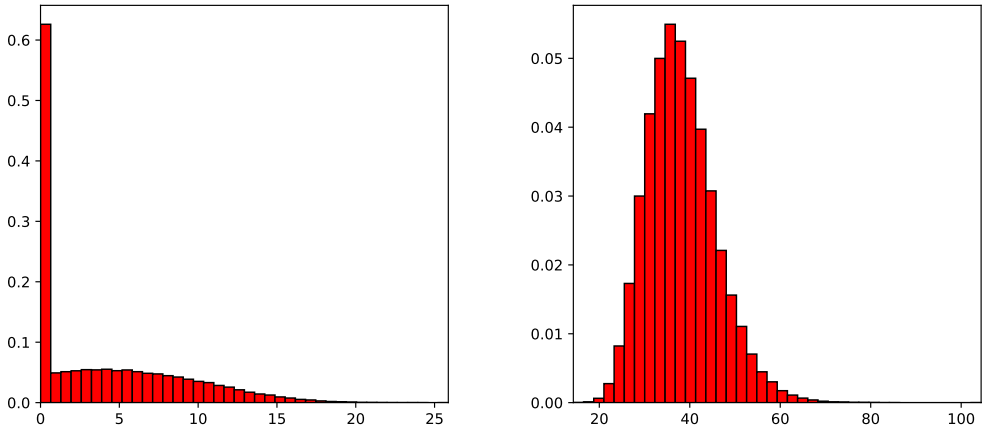
An overview of the set of data is given in Figure 5.6.



(A) Set of data: the red crosses are the data, the black line is the underlying function, given by the Black-Scholes formula.

	V_{t_n}	$S_{t_{n-1}}$
# observations	100000	100000
Minimum	0.00	14.25
Maximum	25.87	104.40
Mean	4.06	38.18
Variance	20.98	58.14
Skewness	0.94	0.61
Kurtosis	-0.04	0.72

(B) Descriptive statistics.



(C) Histogram of the V_{t_n} values.

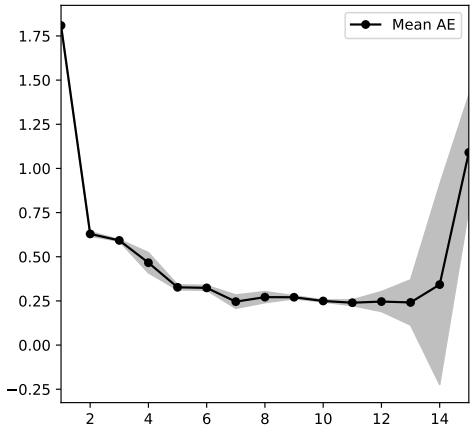
(D) Histogram of the $S_{t_{n-1}}$ values.

FIGURE 5.6: Overview of the set of data.

From Figure 5.6 we get an overview of the highly non-standard regression problem. As expected, since we are simulating the Geometric Brownian Motion exactly, the distribution of $S_{t_{n-1}}$ in Figure 5.6d is in fact log-normal. Although the values observed in Figure 5.6a might appear to follow a Gaussian distribution, we see that there is a relevant mass at zero, corresponding to the paths being out of the money, and also some skew. Furthermore, we can infer from Figure 5.6a that the noise is heteroscedastic and highly non-standard, becoming suddenly zero when most paths are in the money. We thus get insight on why in their paper [12] Longstaff and Schwartz consider only in-the-money paths to regress upon, namely to reduce the mass at zero and to facilitate their (polynomial) fitting, and why the choice of basis functions and covariates is so important.

When it comes to Gaussian Process Regression, the heteroscedastic noise does not directly violate the noise assumption of Equation 2.49. In practice however, the assumption of homoscedastic Gaussian noise is common practice, as it limits to σ_n the number of noise-related hyperparameters to be estimated by maximizing the marginalized log-likelihood of Equation 2.61 or otherwise. As discussed in the previous section, when using GPR it is possible to circumvent the problem by performing batched Monte Carlo simulation as done in [62], essentially constructing a state-dependant estimate $\epsilon(x)$ by sampling the standard deviation from the batched replicates. This approach is however not applicable to the examined sparse approximations which are suitable to this particular problem, as it is impossible to sample the underlying function at the locations of the varying support points. The batched approach does therefore only work when no interpolation of the kernel function takes place, i.e. either the full model is considered, or a subset-of-data approach, or a structured sparse approximation which does not rely on support points, such the Kroenecker or Toeplitz methods.

We proceed to the determination of the continuation value using all examined techniques. In case of the traditional Longstaff-Schwartz algorithm, polynomials basis functions are considered, with degree chosen by empirical risk minimization of the continuation value estimation problem on an i.d.d. generated set of data, see for instance below Figure 5.7a, in case of a Bermudan put. An overview of the thus selected degrees is given below in Table 5.7b, where we can observe great variability in the optimal polynomial degree needed for regression depending on whether the moneyness of the option at the related timestep.



(A) Mean absolute error of the continuation value estimation problem as a function of the grade of the polynomial basis functions. The black line shaded grey area represents the 95% confidence interval. Bermudan Put with 50 exercise opportunities per year, $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

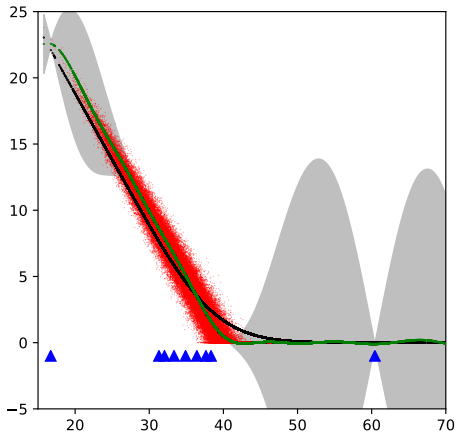
S_{t_0}	T	1		2	
	σ	0.2	0.4	0.2	0.4
36		11	7	8	6
38		10	7	7	6
40		8	7	7	6
42		7	7	7	6
44		7	7	7	6

(B) Optimal grade of the set of polynomials used in the linear regression case. All the values are parameters of the corresponding Bermudan puts.

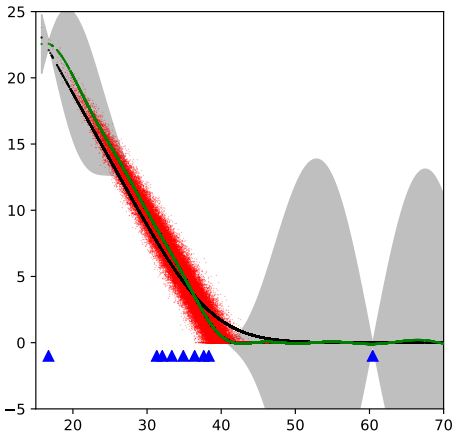
Unlike in the other examples, when considering the full Gaussian Process Regression as well as the subset-of-data methods, the hyperparameters cannot be determined on the whole set, because of the great computational expense. GPR is thus not considered in this scenario. In case of all the other sparse approximations, the hyperparameters are optimized jointly together with the support points, at reduced costs as we have seen. The used kernel is the squared exponential kernel of Equation 2.75 for all methods, namely:

$$k\left(x, x'\right) = \sigma_s^2 e^{-\frac{||x-x'||_2^2}{2l^2}}. \tag{5.3}$$

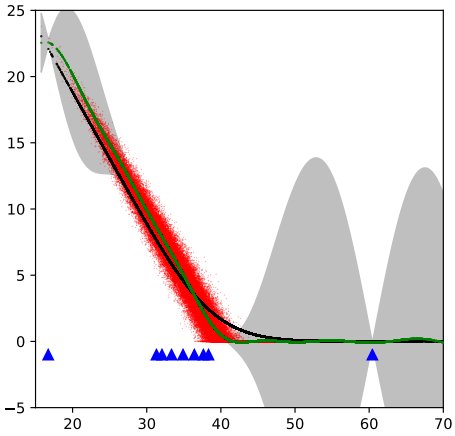
A plot of the regression problem and the related in-sample solution of the examined methods is shown below in Figure 5.8.



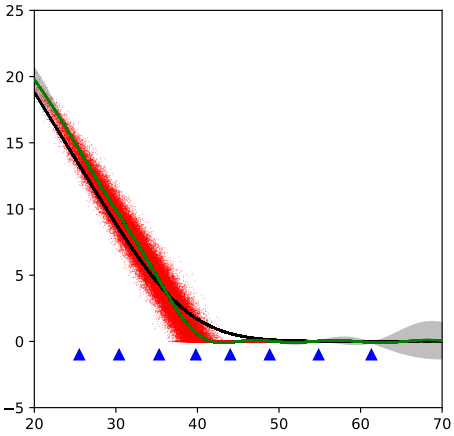
(A) SOR regression.



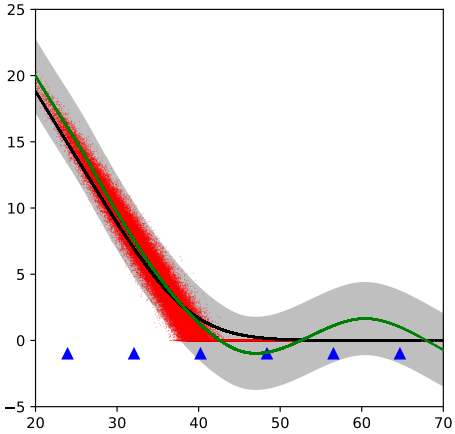
(B) DTC regression.



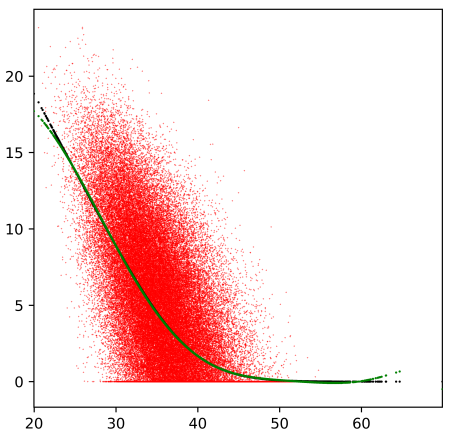
(C) FITC regression.



(D) VFE regression.



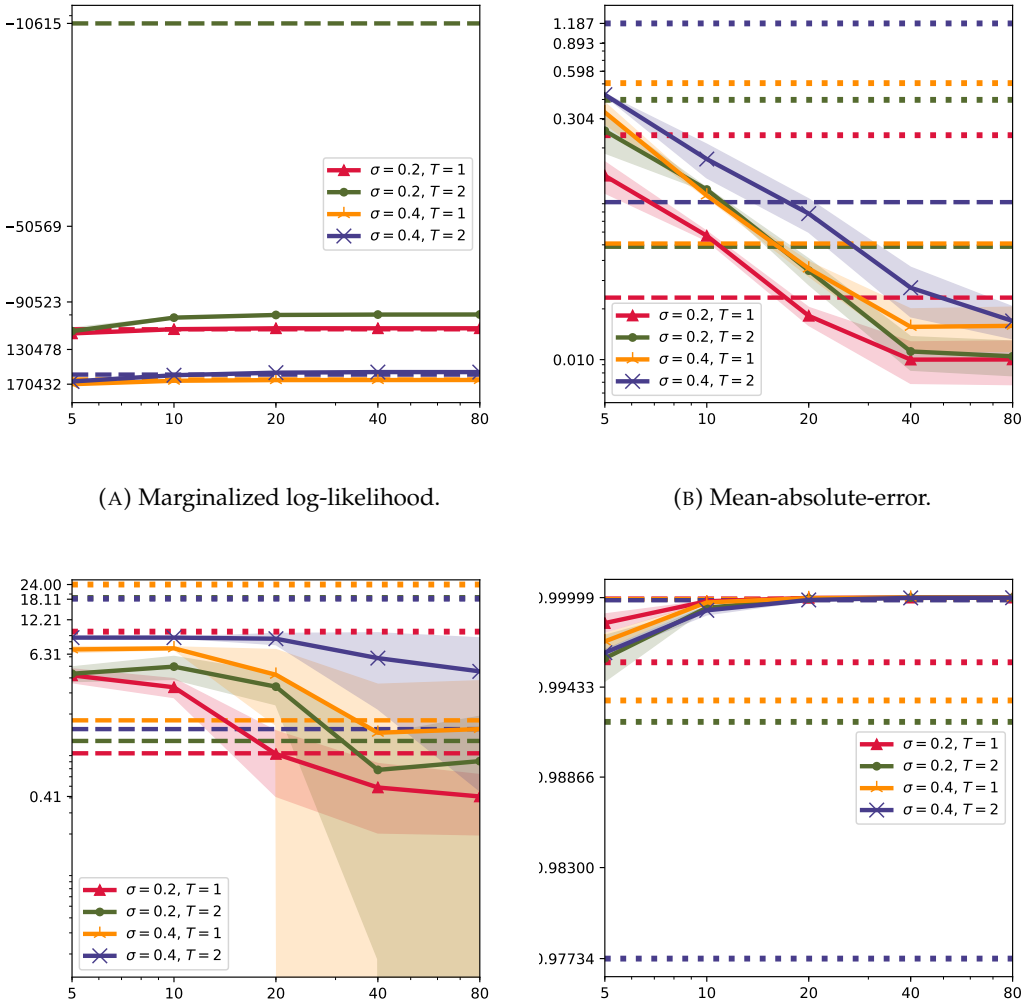
(E) SKI regression.



(F) Linear regression.

FIGURE 5.8: Plots and histograms related to the examined models in case of a the continuation value estimation problem related to a Bermudan Put with 50 exercise opportunities per year, $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

Due to the fact that all prior approximations overfit similarly to what experienced in the previous section and the overall bad performance of subset-of-data methods, we proceed to test the best performing methods which remain, i.e. VFE, SKI and linear regression. Below one can find a plots of the various error metrics in case of one four scenarios, namely $S_{t_0} = 36$, $r = 0.06$, $\sigma \in \{0.2, 0.4\}$, $T \in \{1, 2\}$, $K = 40$ comparing the different methods. VFE is tested for a different number of support points $M \in \{5, 10, 20, 40, 80\}$, while SKI uses the same number of support points as observations as recommended by [52] i.e. $M = N = 100000$. As in the previous case, linear regression makes use of polynomial basis functions of degree given in Table 5.7b.



(A) Marginalized log-likelihood.

(B) Mean-absolute-error.

(C) Max-absolute-error.

(D) R^2.

FIGURE 5.9: Marginalized log-likelihood and error metrics in case of the continuation value estimation problem for a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma \in \{0.2, 0.4\}$, $T \in \{1, 2\}$, $K = 40$ and 50 exercise possibilities per year. Full lines represent VFE, dotted lines linear regression, and dashed lines SKI.

In the Figure 5.9 the results in case of the VFE method of Tables B.17, B.18, B.19, B.20, the SKI method of Tables B.37, B.38, B.39, B.40 and the linear regression of Tables B.57, B.58, B.59, B.60 are presented in the form of a plot, the results related to the other options can be find in tabular form in the Appendix B.1. One can see from Figure 5.9 that in most cases all the methods perform better when the parameters relate to higher moneyness since when more paths are in-the-money or close to being in-the-money, less points are zero in the regression problem. Linear regression scores overall quite bad: although the number of basis function was chosen by empirical risk minimization, it seems that polynomial basis functions are unable to describe well enough the underlying function. VFE and SKI perform in fact strictly better than linear regression, the former achieving good results already with 20 supports points. Furthermore, note that when pricing a Bermudan or American option having maturity $T = 2$, an estimation problem similar to the one presented here for an option having maturity $T = 1$ occurs halfway i.e. at $t = 25\Delta t$ after the first continuation value estimation. This implies that when examining these results we should keep in mind that the degree of the polynomial basis functions was chosen as it being the minimizer of the empirical risk of the examined continuation value estimation problem, not of the final option price i.e. of the

LSM estimator. This in turns implies that when pricing actual Bermudan or American options the performance of linear regression on the continuation value estimation problem will be strictly worse than the ones reported here, as the degree of the polynomial basis functions which minimizes the overall empirical risk will differ from the ones reported in the $T = 1$ and $T = 2$ scenarios (see Table 5.7b), as the continuation value estimation problem changes at each time step. Gaussian Process based methods will by default not have this limitation as they are non-parametric methods, and by the Bayesian formalism in this regards not prone to overfit given enough data and a simple enough kernel function. One should however be careful when selecting hyperparameters by empirical Bayes (type II maximum likelihood estimation) as done in this thesis, as the prior distribution becomes in fact modified (in the form of kernel function) by the data. As mentioned previously, a generic choice of kernel function renders overfitting unlikely, as it limits the informativeness of the prior.

5.3 Continuation value - Geometric Bermudan Basket Put

In order to depart from the one-dimensional case examined in the previous section and still have a closed form solution to the estimation value problem we consider another type of option, namely the **Bermudan Geometric Basket Put**, having payoff:

$$\psi\left(S_t^1, \dots, S_t^N\right) = \max\left(K - \left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}}, 0\right). \tag{5.4}$$

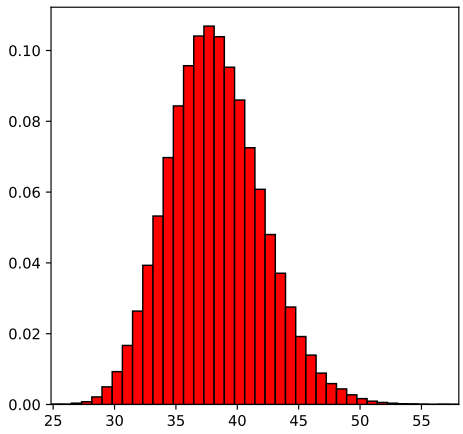
The payoff of such option is similar to the one of an European Put, having as argument however the geometric averages of D stocks, which we model again as Geometric Brownian Motions, each having different volatility σ_i. Under this condition it is possible to derive an analytical pricing formula similar to the one used for pricing a standard European Put under Geometric Brownian Motion dynamics (see Equation A.13 in the Appendix A), given in Equation A.20 in Appendix A. Interestingly, as we are dealing with a geometric average and log-normally distributed underlying, the price is a function of the arithmetic averages of the volatilities and the geometric averages of the starting values of the underlyings, and not on a function of these variables, i.e. when pricing such option in a Monte Carlo simulation it does not matter which underlying is assigned which volatility as long as they are all the same.

Similarly to the previous section we then proceed to the evaluation of the performance of VFE, SKI and linear regression in case of the continuation value estimation problem related to last step of the Monte Carlo simulation, i.e. $\mathcal{C}_{n-1} = \mathbb{E}_{\mathbb{Q}}\left(V_T | S_{T_{n-1}}\right)$, this case however, in a multi-dimensional setting.

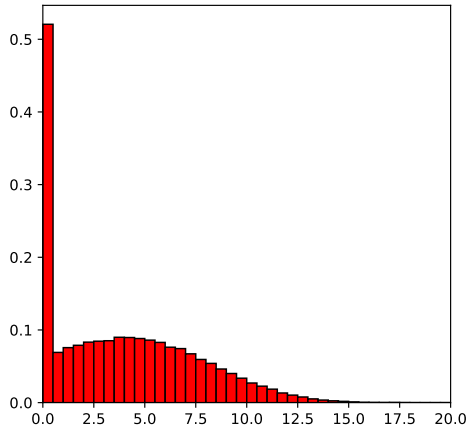
We consider the estimation of the continuation value in the following different contexts, namely the pricing of Geometric Bermudan Basket Puts options under Geometric Brownian Motion dynamics with initial stock values
$S_{t_0} \in \left\{S_{t_0}^1, S_{t_0}^2\right\} = \{\{36, 36, 144, 9, 16, 81\}, \{44, 44, 121, 16, 16, 121\}\}$, risk-free rate $r = 0.06$, volatility $\sigma \in \{\{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}\}$, strike $K = 40$, maturities $T \in \{1, 2\}$ and again 50 exercise opportunities per year, so to account for a different level of moneyness.

We benchmark the various considered methods against the estimation of $\mathcal{C}_{n-1} = \mathbb{E}_{\mathbb{Q}}\left(V_T | S_{T_{n-1}}\right)$ determined path-wise with the Geometric European Basket Put pricing formula of Equation A.20, i.e. the first step of the LSM algorithm. We consider $N = 100000$ paths in order to provide a realistic benchmark, as done in the LSM paper [12]. To exclude any source of bias relative to the discretization, the paths are sampled from their underlying log-normal distribution, i.e. no discretization is employed, as in [12]. The performance of the selected sparse approximations is compared against the one of linear regression with polynomial basis function, with degree determined again by empirical risk minimization.

An overview of the set of data is given below in Figure 5.10:



(A) Histogram of the V_{t_n} values.



(B) Histogram of the $S^1_{t_{n-1}}$ values.

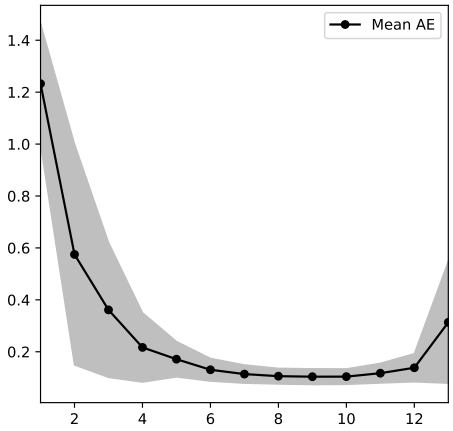
	V_{t_n}	$S^1_{t_{n-1}}$	$S^2_{t_{n-1}}$	$S^3_{t_{n-1}}$	$S^4_{t_{n-1}}$	$S^5_{t_{n-1}}$	$S^6_{t_{n-1}}$
# observations	100000	100000	100000	100000	100000	100000	100000
Minimum	0.00	24.82	15.97	38.90	1.41	1.92	69.00
Maximum	19.99	58.04	85.06	544.57	52.86	145.78	108.52
Mean	3.93	38.20	38.15	152.43	9.54	16.97	85.90
Variance	12.12	14.36	57.86	2154.26	15.51	80.09	18.19
Skewness	0.60	0.29	0.59	0.93	1.32	1.73	0.14
Kurtosis	-0.46	0.15	0.61	1.56	3.31	5.87	0.04

(C) Descriptive statistics.

FIGURE 5.10: Overview of the set of data.

As in the one dimensional case there is considerable mass at zero, yielding a similar yet multidimensional regression problem.

We proceed to the determination of the continuation value using all examined techniques. In case of the traditional Longstaff-Schwartz algorithm, polynomials basis functions are considered, with degree chosen by empirical risk minimization of the continuation value estimation problem on an i.d.d. generated set of data, see for instance below Figure 5.11a, in case of a Bermudan put. An overview of the thus selected degrees is given below in Table 5.11b, where we can observe variability in the optimal polynomial degree needed for regression depending on whether the moneyness of the option at the related timestep.



(A) Mean absolute error of the continuation value estimation problem as a function of the grade of the polynomial basis functions. The black line shaded grey area represents the 95% confidence interval. Bermudan Put with 50 exercise opportunities per year, $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

		T	
		1	2
S_{t_0}	$S_{t_0}1$	9	10
	$S_{t_0}2$	10	11

(B) Optimal grade of the set of polynomials used in the linear regression case. All the values are parameters of the corresponding Bermudan puts.

The kernel used for the sparse Gaussian Process approximations is the squared exponential of Equation 2.75, i.e. with one common length-scale (bandwidth). The reduced number of hyperparameters is also beneficial in terms of time spend in the "training" phase. To keep the computational time reasonable we consider $M - 40$ support points for VFE, and $M = N = 100000$ for SKI.

The results are presented below in Table 5.1:

S_{t_0}	T	LSM (Polynomial)		VFE		SKI	
		Mean AE	Std	Mean AE	Std	Mean AE	Std
$\{36, 36, 144, 9, 16, 81\}$	1	0.104	3.030	0.293	8.761	2.844	15.756
$\{36, 36, 144, 9, 16, 81\}$	2	0.138	5.4117	0.576	11.514	3.444	20.051
$\{44, 44, 121, 16, 16, 121\}$	1	0.313	23.443	0.246	6.742	0.864	12.962
$\{44, 44, 121, 16, 16, 121\}$	2	0.435	23.245	0.549	10.750	1.397	17.051
Mean error:		0.247	0.134	0.416	0.147	2.137	1.045

TABLE 5.1: Geometric Bermudan Basket Put, mean absolute error, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$.

From the results we can see that in this scenario linear regression outperforms both VFE with $M = 40$ support points, and SKI, with $M = 100000$. In case of SKI, there were some technical problems related to determining the hyperparameters, in fact the gradient ascent optimizer of the GPyTorch package [59], which is used to examine SKI(P), would often crash, even with extremely low values of the learning rate, forcing early stopping of the ascent procedure and thus far from ideal hyperparameters. In case of VFE, the results, which are twofolds worse that the linear regression benchmark, suffer from the low number of inducing points. In fact, even though the considered Gaussian kernel has one common lengthscale for all dimensions, it is clear that when considering a dataset with more dimensions the conditioning of the resulting Gram matrix lowers as the interactions across the dimensions are smoothed out, necessitating more inducing points (i.e. a higher rank approximation) than otherwise. When looking at the computational times, presented in below in Table 5.2, it is clear that when dealing with only a six dimensional regression problem, linear regression becomes unmanageable for the considered number of basis functions, since the interaction terms are computed. When considering computational time, VFE performs more or less

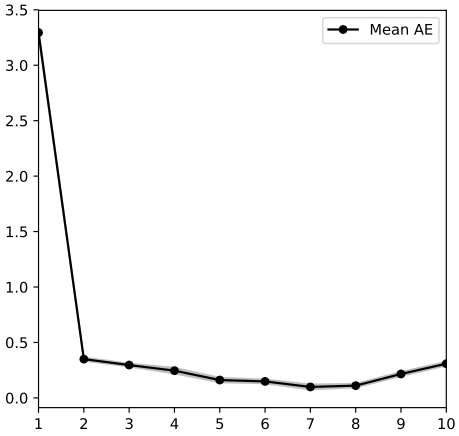
as linear regression, as the support points are optimized jointly with the hyperparameters, incurring in higher computational costs, which as we have seen, scale linearly with the number of dimensions. SKI(P) on the other hand, when considering the computational time, is clearly the winner, as the support poitns structure is fixed, as is able to decompose the solution of the predictive equations over each dimension.

S_{t_0}	T	LSM (Polynomial)		VFE		SKI	
		Mean AE	Std	Mean AE	Std	Mean AE	Std
$\{36, 36, 144, 9, 16, 81\}$	1	111.19	5.02	470.51	19.40	44.90	0.52
$\{36, 36, 144, 9, 16, 81\}$	2	773.93	0.75	593.61	34.34	45.25	0.41
$\{44, 44, 121, 16, 16, 121\}$	1	798.15	0.81	449.77	17.83	47.11	0.58
$\{44, 44, 121, 16, 16, 121\}$	2	2076.16	210.63	448.91	60.04	47.64	0.27
Mean time:		939.84	771.57	490.70	0.147	46.22	1.17

TABLE 5.2: Geometric Bermudan Basket Put, mean computational time, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$.

5.4 Bermudan Put pricing

After having analyzed the crucial step of the LSM algorithm, namely the estimation of the continuation value, we resort to test the goodness of the examined approximation and linear regression in the full algorithm. We hence setup a series of experiments, having the same parameters as the ones in the one-dimensional continuation value estimation problem, namely the pricing of Bermudan Puts options under Geometric Brownian Motion dynamics with initial stock values $S_{t_0} \in \{36, 38, 40, 42, 44\}$, risk-free rate $r = 0.06$, volatility $\sigma \in \{0.2, 0.4\}$, strike $K = 40$, maturities $T \in \{1, 2\}$ and 50 exercise opportunities per year, so to account for a different level of moneyness, i.e. we try to replicate and improve Table 1 in the LSM paper [12]. As in [12] we consider $N = 100000$ paths in order to provide a realistic benchmark, and simulate the process of the underlying exactly. The results from the traditional LSM algorithm as well as the one PSOR finite difference solution are taken from [12] who use for pricing all the options (scaled) Laguerre polynomials of third degree $\phi = \left\{e^{-\frac{X}{2}}, e^{-\frac{X}{2}}(1 - X), e^{-\frac{X}{2}}(1 - 2X)\right\}$. In addition we attempt pricing with polynomial basis functions whose degree is again determined by empirical risk minimization, see below Figure 5.12a and Table 5.12b, using the LSM Algorithm 5.



(A) Mean absolute error of the continuation value estimation problem as a function of the grade of the polynomial basis functions. The black line shaded grey area (not visible) represents the 95% confidence interval. Bermudan Put with 50 exercise opportunities per year, $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

S_{t_0}	T	1		2	
	σ	0.2	0.4	0.2	0.4
36		7	7	10	6
38		9	7	7	6
40		9	7	7	6
42		8	7	7	6
44		7	7	7	6

(B) Optimal grade of the set of polynomials used in the linear regression case. All the values are parameters of the corresponding Bermudan puts.

We consider both VFE and SKI approximation, the first with $M = 40$ support points in light of the results of 5.9b, the latter with $M = 100000$ as advised in [52] and learning rate $\alpha = 0.5$. In case of VFE, the hyperparamters and support points are estimated jointly every 25 iteration, while the support points are optimized at every other iteration, keeping the hyperparameters constant, similarly to what is done by [62]. The same holds for SKI except of course for the support points, which are automatically placed on a grid, resulting in a Toeplitz structure. The kernel used for both sparse Gaussian Process approximations is the squared exponential of Equation 2.75. For both approximations we use the estimator of Equation 4.21 and perform regression on all the paths, since when considering the formulation of Algorithm 5 we often incur in numerical instability when trying to find suitable hyperparameters when the option gets out-of-the-money.

The resulting errors are shown below in Table 5.3, while the results themselves are in Table B.89 in the Appendix A.

S_{t_0}	σ	T	LSM (Laguerre)		LSM (Polynomial)		VFE		SKI	
			Mean	Std	Mean	Std	Mean	Std	Mean	Std
36	0.2	1	0.006	0.010	0.099	0.012	0.000	0.004	0.000	0.007
36	0.2	2	0.019	0.012	0.196	0.058	0.006	0.010	0.009	0.014
36	0.4	1	0.010	0.020	0.311	0.030	0.011	0.015	0.008	0.018
36	0.4	2	0.020	0.024	0.708	0.018	0.014	0.025	0.011	0.029
38	0.2	1	0.006	0.009	0.156	0.008	0.004	0.008	0.007	0.008
38	0.2	2	0.010	0.011	0.253	0.011	0.007	0.006	0.013	0.011
38	0.4	1	0.009	0.019	0.358	0.043	0.004	0.019	0.007	0.024
38	0.4	2	0.001	0.022	0.715	0.027	0.024	0.025	0.022	0.028
40	0.2	1	0.001	0.009	0.175	0.014	0.007	0.007	0.010	0.010
40	0.2	2	0.006	0.010	0.279	0.020	0.006	0.005	0.008	0.014
40	0.4	1	0.004	0.018	0.413	0.036	0.005	0.019	0.008	0.020
40	0.4	2	0.001	0.022	0.719	0.042	0.010	0.029	0.008	0.029
42	0.2	1	0.000	0.007	0.179	0.012	0.002	0.006	0.002	0.008
42	0.2	2	0.006	0.010	0.311	0.020	0.009	0.009	0.007	0.011
42	0.4	1	0.006	0.017	0.467	0.039	0.010	0.013	0.012	0.015
42	0.4	2	0.005	0.021	0.751	0.044	0.029	0.015	0.029	0.018
44	0.2	1	0.008	0.007	0.172	0.013	0.002	0.007	0.004	0.007
44	0.2	2	0.015	0.009	0.338	0.019	0.004	0.007	0.003	0.010
44	0.4	1	0.009	0.017	0.511	0.044	0.007	0.015	0.005	0.018
44	0.4	2	0.025	0.021	0.796	0.069	0.015	0.020	0.016	0.023
Mean error:			0.008	0.007	0.395	0.223	0.009	0.007	0.009	0.007

TABLE 5.3: Bermudan Put numerical results errors. The Laguerre LSM and PSOR results used as benchmark are from [12].

We can see that the results of the VFE and SKI methods when applied to the full Longstaff-Schwartz algorithm are in most scenarios better than the ones obtained by using linear regression. This holds especially true when one regards the results obtained using polynomial basis functions, which are clearly not suitable for the pricing of these types of options, while the LSM results regarding Laguerre polynomials shown in the Table B.89 are overall very good, as they are carefully selected for this particular task [12]. In particular, we can observe that options priced with linear regression using Laguerre basis functions obtain on average an error which is two orders of magnitude lower than when using polynomial basis functions, highlighting the large difference induced by different model choices.

It is therefore clear that, when not estimating the hyperparameters at every iteration, the advantage of using Gaussian Process Regression or its approximations in the LSM algorithm is not desirable in order to attain superior performance (which would be in principle obtainable also with linear regression, provided good knowledge regarding the basis functions etc.), but rather because of the fact that it allows to sidestep the previously discussed issues which plague parametric models, in particular linear regression, and obtain good results.

As in the previous sections, options which are less often in-the-money give worse and uncertain results compared to the PSOR benchmark, independent of the method. Furthermore as one can expect, SKI performs slightly worse (higher mean and standard deviation of the option price) than VFE due to the choice of learning rate, which was kept constant for all iterations of the algorithm, leaving surely some margin for improvement in that sense.

Furthermore, when examining the estimated option prices of Table B.89 in the Appendix, we see that both VFE and SKI methods are always overestimating the option price, i.e. are affected by positive bias. This might be related to the fact that the hyperparameters are estimated every 25 iterations, and do therefore induce a temporal dependence among the different iterations, resulting in positive bias. In order to verify this claim it would be interesting to replicate the experiment while estimating the hyperparameters every iteration, which is in our experience however particularly costly since the closer the option gets in the money, the more mass is located at zero, invalidating further the assumption of (homoscedastic) Gaussian noise, resulting in a non-smooth marginalized log-likelihood function, and requiring more restarts of the optimizer to yield sensible results.

Chapter 6

Conclusion and Outlook

In this thesis, Gaussian Process Regression and a variety of related approximation based on a selection of support points were reviewed in relation to their application regarding the determination of conditional expectations, in particular in the Monte Carlo based Longstaff-Schwartz algorithm for the pricing of Bermudan and American options. The full method, as well as the selected approximation were implemented and tested on different sets of data in order to evaluate their application to the Longstaff-Schwartz algorithm.

6.1 Summary

Gaussian Process Regression has proven itself to be a flexible regression technique, however hard to use on large, possibly high-dimensional datasets, since the computational complexity related to "training" is of order $\mathcal{O}\left(N^3\right)$ if one uses Cholesky factorisation. Solving the (possibly preconditioned) linear system by a modified version of the Conjugate Gradient algorithm with early stopping, as done in [59], can lower down the complexity to $\mathcal{O}\left(N^2\right)$, which is a very big improvement, but similar to the case of SVMs with Gaussian kernel, still severely limits the applicability to larger datasets. Furthermore, the examined exact structured methods, namely Toeplitz or Kronecker based methods, are very promising as they greatly reduce the computational complexity of the "training" and "test" phases, bringing it down to superlinear. Unfortunately the cases in which the training data is located on a Cartesian grid or on evenly spaced one-dimensional intervals are very limited, rendering these methods not applicable in most situations. Therefore sparse approximations are at this moment still a necessity.

Judging from the empirical comparison which was carried out, i.e. Section 5, it is clear that the examined subset-of-data methods related to information theory are hard to apply in a real-life scenario, since they require some form of prior knowledge of the hyperparameters and base their support points selection on the posterior variance estimate, which does not depend on the observations. As pointed out before, a fully Bayesian approach to make use of such information-theoretical measures is highly impractical, as hierarchical modelling with Gaussian Process Regression requires resorting to the use of sampling methods or variational inference to obtain an approximation of the posterior, as no hyperprior on the hyperparameters is known which can yield an analytical posterior. Using then such posterior approximation to select informative points from the training set with computational complexity of $\mathcal{O}\left(N^2\right)$ per sample does not seem an appealing course of action. Lastly, discarding available data is clearly a poor choice.

On the other hand the examined sparse approximations based on the Nyström low-rank approximation seem to be promising. Unfortunately, as pointed out previously, prior approximations have been observed to overfit on the examined datasets and to place their support points on local minima or inflection points, or even outside of the training set. Posterior approximations, such as the examined VFE have been shown to correct this behaviour by effectively adding a penalty term to the marginalized log-likelihood of the Discrete Training Conditional, forcing it to recover the one of the full Gaussian Process Regression as $M \to N$. Further related observed benefits of the penalty term are faster hyperparameter estimation and more importantly an even distribution of the support points inside the training set. One drawback of all these methods is the selection of support points by itself, which if done by marginal log-likelihood maximization is relatively costly. A priori selection of the support

points is of course possible, but not always applicable and not convenient, as one might then just as well resort to the use of a structured sparse approximation such as SKI, which places the support points on a grid.

By placing the support points on a grid such that the $K_{M,M}$ Gran matrix is either Toeplitz or Kronecker, and by approximating the $K_{N,M}$ Gram matrix associated with Nyström-based methods by (cubic) interpolation, Structured Kernel Interpolation makes Toeplitz and Kronecker methods applicable to every set of data. Unfortunately all these methods suffer from the "curse of dimensionality", as the number of support points per grid dimension decreases exponentially as a function of the number of dimensions of the data. A notable exception is the SKIP approach of [60], which is able to partially lift the "curse of dimensionality" when considering structured kernels (and data).

When it comes to numerical results, the estimation of the continuation value, i.e. the approximation of the conditional expectation in the LSM algorithm, was carried out in case of all examined approximations. The most promising results are obtained from the VFE and SKI methods, which represent the state-of-the-art of the current approximations of Gaussian Process Regression. When comparing the two methods it can be concluded that SKI is overall cheaper computational-wise when dealing both with low dimensional problems $D < 5$ [52] and high dimensional problems (SKIP) [60]: its superlinear computational complexity obtained thanks to its structured support points (and kernel) is hard to beat. The superiority over other methods in this regard is clear when considering the estimation of the continuation value in a multidimensional setting as done in Section 5.3, where the computation time of VFE (and linear regression) becomes quickly unmanageable as the number of dimensions increases.

The practicality of these methods is clear when comparing them with the industry standard, namely linear regression, Although the problem of specifying the prior, i.e. mean and kernel function, seems similar to the tedious one regarding the choice of basis functions in case of linear regression, this is not the case, as theory and practice have shown that in most cases a simple but flexible kernel such as, the Gaussian kernel in one of its isotropic variants, is most often a good choice, as the number of hyperparameters is very limited, limiting thus to some extent, as we have discussed, the informativeness of the prior and thus the possibility of overfitting, which is introduced by the empirical Bayes procedure. Empirical Bayes ensures the departure from the full Bayesian formalism, and renders the problem of overfitting or underfitting, i.e. the goodness of the model fit only reliably examinable by looking at the empirical and true risks (training and test errors), as from a frequentist point of view. In return, however, model flexibility is made possible.

The application of the considered sparse approximation to the Longstaff-Schwartz algorithm was considered in case of a variety of Bermudan put options with different moneyness. As stated in the related section, the empirical Bayes method was not applied at every iteration in order to reduce computational time, but instead the kernel was "frozen" for a number of iterations. Although the results are good and comparable if not better than the one obtained by linear regression in [12] with "optimal" choice of basis functions, we have that both methods (Longstaff-Schwartz using VFE and SKI) overestimate the option price and seem thus to have slight positive bias, which is especially noticeable in case of options with long maturity. This might be due to the kernel "freezing" procedure that was just mentioned, which effectively induces a temporal dependency in the continuation value estimation problem since θ_t are not themselves $\mathcal{F}_t$ measurable at every timestep t, $\mathcal{F}_{t+k}$ for some k depending on the iteration and "freezing". This issue can be solved by updating the hyperparameters at each timestep, and possibly by determining the option exercise strategy on a different set of id.d. paths than the ones used for the actual pricing, as done in [19] to further reduce the bias.

The very practical "freezing" approach is, as explained, formally unjustified, as well as the practice of fitting the hyperparameters to a subset of the "training" set, which is very uncommon in literature (but is applied to the LSM algorithm by [63]), since one assumes that the sampled data is synthesising the whole dataset. As we have seen in the subset-of-data example, this assumption does not lead to good results depending on the choice of kernel, and the characteristics of the dataset.

In general, the use of Gaussian Process Regression and of its sparse approximation in the LSM algorithm is still limited by the "curse of dimensionality", which plagues more or

less all examined methods, rendering them unpractical or unusable when pricing basket options with many underlying or with complicated highly-dimensional underlying models. Of course as stated previously, SKI is especially plagued by this problem, limiting its applicability to cases in which $D < 5$.

6.2 Conclusion

Gaussian Process Regression is an extremely interesting explainable machine learning algorithm, with many connections to different branches of mathematics and computer science, most notably its resemblance to SVMs, splines, and most notably neural Neural Networks. Applications to large datasets are becoming easier thanks to the advances made in the last years when it comes to GPR approximations and computational advances, however high dimensionality plagues many such approximations.

In this thesis the applicability of GPR and its approximations have been studied in light of a possible application to the LSM algorithm. In light of the results presented in Chapter 5, it is clear that the use of GPR, in particular of the VFE and SKI methods, delivers good results in the context of option pricing, nearly identical to the good linear regression benchmark of [12]. In particular, although there remain some practical issues to be solved such as deciding how often to estimate the hyperparameters or the choice of kernel, it is clear that the main advantage of such algorithms when it comes to the LSM algorithm is the fact that they do not require to know in advance what, how many, and which basis functions to use as opposed to linear regression.

The computational time involved in pricing options is still however one order of magnitude higher than when using linear regression with few basis functions. We have seen however that in higher dimensional problems, all methods fail to deliver results in an acceptable time, except for SKIP, which partially bypasses the "curse of dimensionality". SKI(P) is therefore surely the most versatile approximation, as it can be applied to datasets of (in theory) arbitrary dimension, without requiring to optimize the inducing points. It is by all means not perfect, as relying on a grid can be a severe drawback if most data is concentrated locally, but its fast training time combined with the possibility to exploit the structure of the Gram matrix, makes it the best approximation so far examined.

In conclusion, applications (in finance) which involve regression are a good scenario for GPR, and might prove it to be the regression algorithm of choice, given its excellent explainability, flexibility, and its connection to other algorithms such as SVMs, splines, and Neural Networks.

6.3 Further research

The main drawback which limits the use of Gaussian Process Regression and its approximations in applications such as the LSM algorithm, is at this point in time is not only the computational burden arising from a large dataset, but also the "curse of dimensionality" associated with high dimensionality when considering the reviewed approximations.

One promising idea to attempt to lift said curse is given by the Tensor Train Decomposition [37], a type of tensor decomposition which generalizes the concept of low-rank form and renders many linear algebra operations cheap if one considers its decomposed form. Its computation is based on the low-rank approximation of auxiliary unfolding matrices, similarly to what is done in case of kernel matrices having Kronecker structure in case of Kronecker structured methods. Applications of Tensor Train Decomposition to Gaussian Process Regression are still lacking, one notable exception being [54]. The authors herein use a stochastic variational inference approach similar to the stochastic VFE method of [45] (in this case inducing points located on a Cartesian grid are considered), using however Tensor Train decomposition in order to decompose the mean vector of the variational distribution, while decomposing the covariance matrix using the resulting Kronecker structure. The predictive posterior distribution is then be approximated by the variational distribution, obtained as in [45] by stochastic optimization, and having linear complexity with respect to dimensionality of the dataset. The drawback of this approach is however that once more inducing points

are used which are located on a Cartesian grid, therefore limiting the expressiveness of the approximated kernel function on the small manifold on which most of the data lies, requiring again (exponentially) more inducing points than necessary. One would therefore ideally wish to utilize Tensor Train decomposition or another type of decomposition in a way that does not require the use of (structured) inducing points, similarly to what is done with exact Kronecker and Toeplitz methods, which seem intuitively to be some kind of particular case in a more general (at the moment non-existing) framework.

In order to further improve the applicability of GPR to the LSM algorithm, one may want to first of all to investigate why regression on only in-the-money paths as done in [12] performs worse with Gaussian Process Regression. Also, the non-standard continuation value estimation problem can perhaps benefit from modelling the noise as heteroscedastic following [56], which however introduces more hyperparameters to be fitted with empirical Bayes, rendering the model more complex and potentially prone to overfit. As mentioned previously in Chapter 5, one might also want to investigate the source of the positive bias affecting the option pricing results by verifying that the costly performance gain obtainable by estimating the hyperparameters at every iteration is bias-free. Furthermore, it would be a good idea to experiment also with different kernels departing from the commonly used Gaussian kernel, for example the Neural Network kernel, or perhaps some other kernel derived keeping in mind the fact that moments of the various stochastic processes used to model the underlying stocks and assets are very often known. Lastly, option pricing in more realistic condition should be explored, involving higher dimensional models for the assets.

Appendix A

Mathematical appendix

A.1 Gaussian identities

Conditional multivariate normal density

Let x and y be jointly normally distributed, such that:

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_x \\ m_y \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right) = \mathcal{N}\left(\begin{bmatrix} m_x \\ m_y \end{bmatrix}, \begin{bmatrix} \bar{A} & \bar{C} \\ \bar{C}^T & \bar{B} \end{bmatrix}^{-1} \right) \tag{A.1}$$

Then the marginal distribution of x reads [34]:

$$p(x) = \mathcal{N}(m_x, A) \tag{A.2}$$

and the distribution of x given y reads [34]:

$$p(x|y) = \mathcal{N}\left(m_x + CB^{-1}(y - m_y), A - CB^{-1}C^T \right) \tag{A.3}$$

$$= \mathcal{N}\left(m_x - \bar{A}^{-1}\bar{C}(y - m_y), \bar{A}^{-1} \right) \tag{A.4}$$

A.2 Matrix identities

Woodbury matrix identity

Let A be an $N \times N$ invertible matrix. Then [34]:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U\left(C^{-1} + VA^{-1}U\right)^{-1}VA^{-1} \tag{A.5}$$

where A, U, V, C have conformable size.

Sylvester determinant theorem

Let A be an $N \times N$ invertible matrix. Then [34]:

$$|A + UCV| = |A|\,|C|\left|C^{-1} + VA^{-1}U\right| \tag{A.6}$$

where A, U, V, C have conformable size.

Block Cholesky decomposition

Consider a symmetric-positive-definite block matrix:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{A.7}$$

Then the matrix can be decomposed as [34]:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = LL^T \tag{A.8}$$

with L:

$$L = \begin{bmatrix} L_A & 0 \\ CL_A^{-1} & \left(D - CA^{-1}B\right)^{\frac{1}{2}} \end{bmatrix} \tag{A.9}$$

A.3 Black-Scholes

Black-Scholes formula for the pricing of an European Call option

The value V_t at time t of an European Call option having payoff $\psi(S_t) = \max(S_t - K, 0)$, maturity T and strike K related to a stock S with GBM dynamics such that $dS_t = \mu S_t dt + \sigma S_t dW_t$, where μ is the rate of return, σ is the volatility, and under a risk-free rate r ensures that:

$$\begin{aligned} V_t &= e^{-r(T-r)}\mathbb{E}_{\mathbb{Q}}\left(\psi(S_T)\,|\mathcal{F}_t\right) \\ &= S_t\Phi(d_1) - Ke^{-r(T-r)}\Phi(d_2), \end{aligned} \tag{A.10}$$

where $\Phi(\cdot)$ is the cumulative density function of a normally distributed random variable, and d_1 and d_2 are given by:

$$d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}, \tag{A.11}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}. \tag{A.12}$$

Black-Scholes formula for the pricing of an European Put option

The value V_t at time t of an European Put option having payoff $\psi(S_t) = \max(K - S_t, 0)$, maturity T and strike K related to a stock S with GBM dynamics such that $dS_t = \mu S_t dt + \sigma S_t dW_t$, where μ is the rate of return, σ is the volatility, and under a risk-free rate r ensures that:

$$\begin{aligned} V_t &= e^{-r(T-r)}\mathbb{E}_{\mathbb{Q}}\left(\psi(S_T)\,|\mathcal{F}_t\right) \\ &= Ke^{-r(t-T)}\Phi(-d_2) - S_t\Phi(-d_1) \end{aligned} \tag{A.13}$$

where $\Phi(\cdot)$ is the cumulative density function of a normally distributed random variable, and d_1 and d_2 are given by:

$$d_1 = \frac{\ln\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma\sqrt{T-t}}, \tag{A.14}$$

$$d_2 = d_1 - \sigma\sqrt{T-t}. \tag{A.15}$$

Black-Scholes formula for the pricing of an European Geometric Basket Call option

The value V_t at time t of an European Geometric Call option having payoff $\psi\left(S_t^1, \ldots, S_t^N\right) = \max\left(\left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}} - K, 0\right)$, maturity T and strike K related to a stock S^i with GBM dynamics such that $dS_t^i = \mu S_t^i dt + \sigma^i S_t^i dW_t$, where μ is the rate of return, σ^i is the volatility, and under a risk-free rate r ensures that:

$$\begin{aligned} V_t &= e^{-r(T-r)}\mathbb{E}_{\mathbb{Q}}\left(\psi\left(S_t^1,\ldots,S_t^N\right)|\mathcal{F}_t\right) \\ &= \left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}} e^{-\frac{1}{2}(D-1)\bar{\sigma}^2}\Phi\left(d_1\right) - Ke^{-r(T-r)}\Phi\left(d_2\right), \end{aligned} \tag{A.16}$$

where $\Phi(\cdot)$ is the cumulative density function of a normally distributed random variable, and d_1, d_2 and $\bar{\sigma}$ are given by:

$$d_2 = \frac{\ln\left(\frac{\left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}}}{K}\right) + r\left(T-t\right) - \frac{1}{2}D\bar{\sigma}^2}{\bar{\sigma}}, \tag{A.17}$$

$$d_1 = \bar{\sigma} + d_2 \tag{A.18}$$

$$\bar{\sigma}^2 = \frac{\sum_{i=1}^{D}{\sigma^i}^2}{D^2}\left(T-t\right) \tag{A.19}$$

Black-Scholes formula for the pricing of an European Geometric Basket Put option

The value V_t at time t of an European Geometric Call option having payoff $\psi\left(S_t^1,\ldots,S_t^N\right) = \max\left(K - \left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}}, 0\right)$, maturity T and strike K related to a stock S^i with GBM dynamics such that $dS_t^i = \mu S_t^i dt + \sigma^i S_t^i dW_t$, where μ is the rate of return, σ^i is the volatility, and under a risk-free rate r ensures that:

$$\begin{aligned} V_t &= e^{-r(T-r)}\mathbb{E}_{\mathbb{Q}}\left(\psi\left(S_t^1,\ldots,S_t^N\right)|\mathcal{F}_t\right) \\ &= Ke^{-r(T-r)}\Phi\left(d_2\right) - \left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}} e^{-\frac{1}{2}(D-1)\bar{\sigma}^2}\left(1-\Phi\left(d_1\right)\right), \end{aligned} \tag{A.20}$$

where $\Phi(\cdot)$ is the cumulative density function of a normally distributed random variable, and d_1 and d_2 are given by:

$$d_2 = \frac{\ln\left(\frac{\left(\prod_{i=1}^{D} S_t^i\right)^{\frac{1}{D}}}{K}\right) + r\left(T-t\right) - \frac{1}{2}D\bar{\sigma}^2}{\bar{\sigma}}, \tag{A.21}$$

$$d_1 = \bar{\sigma} - d_2 \tag{A.22}$$

$$\bar{\sigma}^2 = \frac{\sum_{i=1}^{D}{\sigma^i}^2}{D^2}\left(T-t\right) \tag{A.23}$$

Appendix B

Numerical appendix

B.1 Numerical results

In this Appendix, the results present in the thesis in the form of plots are presented in tabular form.

Sinc function

Results related to Section 5.1.

GP

log-likelihood	Mean AE	Max AE	R^2	time
17945	0.0013646	0.00839399	0.99997096	51.75

TABLE B.1: In-sample error metrics in case of the full GP method on the sinc dataset.

log-likelihood	Mean AE	Max AE	R^2	time
17945	0.00135226	0.00838319	0.99997209	51.75

TABLE B.2: Out-of-sample error metrics in case of the full GP method on the sinc dataset.

SOR/DTC

M	log-likelihood	Mean AE	Max AE	R^2	time
5	11249	0.03126938	0.21358278	0.95678191	0.27
10	17970	0.00149866	0.0088006	0.99996782	0.34
15	17968	0.00145605	0.00714424	0.99997011	0.89
20	17971	0.00150475	0.00897212	0.99996754	1.13
25	17970	0.00149464	0.00881789	0.99996818	1.54
30	17971	0.00150477	0.00897328	0.99996754	1.62
35	17971	0.00149669	0.00861424	0.99996836	1.42
40	17968	0.00150456	0.00838172	0.9999655	1.88
45	17970	0.00149866	0.0088006	0.99996782	1.63
50	17971	0.00150718	0.00854673	0.99996799	2.46
55	17969	0.00141285	0.00855699	0.9999696	3.38
60	17969	0.00142321	0.00785756	0.99996998	2.77
65	17971	0.00149413	0.00879333	0.9999683	1.17
70	17971	0.00150155	0.00892016	0.99996799	3.36
75	17970	0.00149012	0.00880565	0.99996833	3.15
80	17971	0.00149303	0.00887613	0.99996823	3.41
85	17971	0.00150516	0.00899921	0.99996752	5.76
90	17969	0.00140505	0.00738545	0.99997027	4.25
95	17971	0.00149724	0.00877288	0.99996823	3.85
100	17969	0.00142234	0.00787802	0.99996998	5.96
200	17971	0.00149414	0.0087934	0.9999683	12.19
300	17971	0.00150895	0.00847688	0.99996799	14.20
400	17971	0.00150727	0.00855382	0.99996798	19.51
500	17971	0.00150752	0.00899396	0.99996746	24.36
600	17969	0.00141284	0.00855648	0.9999696	34.28
700	17969	0.00141287	0.00855714	0.9999696	38.43
800	17969	0.00142356	0.0081767	0.99996975	40.85
900	17971	0.00149996	0.00899616	0.99996792	46.78
1000	17970	0.00150336	0.00880083	0.99996769	48.21

TABLE B.3: In-sample error metrics in case of the SOR/DTC method on the sinc dataset.

M	log-likelihood	Mean AE	Max AE	R^2	time
5	11249	0.0305816	0.21358277	0.95902923	0.21
10	17970	0.00148919	0.00878611	0.99996899	0.36
15	17968	0.00145105	0.00713663	0.99997090	0.76
20	17971	0.00149514	0.00895696	0.99996874	0.99
25	17970	0.00148865	0.00880338	0.99996923	1.15
30	17971	0.00149516	0.00895812	0.99996874	1.60
35	17971	0.00149128	0.00860017	0.9999694	1.11
40	17968	0.00148765	0.00836812	0.99996683	2.42
45	17970	0.00148919	0.00878612	0.99996899	2.27
50	17971	0.00150009	0.00853254	0.99996908	2.17
55	17969	0.00140087	0.00854288	0.99997081	2.65
60	17969	0.00141437	0.0078453	0.99997105	4.52
65	17971	0.00148846	0.0087787	0.99996936	3.03
70	17971	0.00149557	0.00890507	0.99996906	2.96
75	17970	0.00148421	0.00879118	0.99996938	4.07
80	17971	0.00148684	0.00886126	0.9999693	4.44
85	17971	0.00149552	0.00898395	0.99996871	4.71
90	17969	0.00141438	0.00737417	0.99997077	5.85
95	17971	0.00149155	0.00875829	0.99996928	3.90
100	17969	0.00141336	0.00786574	0.99997105	5.44
200	17971	0.00148846	0.00877877	0.99996936	10.31
300	17971	0.00150209	0.00846293	0.99996907	9.61
400	17971	0.00150016	0.00853961	0.99996908	26.32
500	17971	0.00149813	0.00897873	0.99996865	18.70
600	17969	0.00140087	0.00854237	0.99997081	21.53
700	17969	0.00140090	0.00854303	0.99997080	31.08
800	17969	0.00141403	0.0081634	0.99997087	40.03
900	17971	0.00149359	0.00898093	0.99996901	42.89
1000	17970	0.0014941	0.00878634	0.99996886	39.17

TABLE B.4: Out-of-sample error metrics in case of the full GP method on the sinc dataset.

FITC

M	log-likelihood	Mean AE	Max AE	R^2	time
5	16167	0.02140347	0.05279918	0.99375242	0.32
10	17971	0.00149348	0.00863559	0.99996844	0.58
15	17970	0.00150217	0.00825691	0.99996835	1.03
20	17970	0.00149221	0.00888536	0.99996822	0.79
25	17970	0.00141544	0.0070347	0.99996894	0.85
30	17970	0.00150217	0.00825688	0.99996835	1.51
40	17970	0.00150401	0.00881403	0.99996766	2.32
45	17971	0.0015014	0.00894103	0.99996766	2.04
50	17970	0.00149676	0.00889583	0.99996807	2.76
55	17971	0.0015051	0.00899251	0.99996752	3.20
60	17969	0.00142348	0.00813646	0.99996978	3.18
65	17971	0.00150134	0.00891295	0.999968	3.08
70	17970	0.00141547	0.00691219	0.99996899	2.86
75	17971	0.00150656	0.00895251	0.99996751	4.10
80	17968	0.00146008	0.00713451	0.99997	4.22
85	17968	0.00145893	0.0071379	0.99997003	3.84
90	17969	0.00141284	0.00855656	0.9999696	5.46
95	17971	0.00150433	0.00851773	0.99996809	4.81
100	17971	0.0014948	0.00891777	0.99996819	3.94
200	17969	0.001475	0.00528559	0.99996876	10.72
300	17971	0.00150061	0.0089268	0.99996801	10.906
400	17971	0.00149968	0.00893564	0.99996803	11.20
500	17971	0.00150185	0.00898295	0.99996762	18.13
600	17969	0.00142348	0.00814805	0.99996977	23.42
700	17969	0.00141211	0.0085062	0.99996965	32.88
800	17971	0.00149499	0.00895716	0.99996815	39.66
900	17969	0.00141327	0.00858578	0.99996957	62.13
1000	17968	0.00142486	0.00821734	0.99996961	42.51

TABLE B.5: In-sample error metrics in case of the FITC method on the sinc dataset.

M	log-likelihood	Mean AE	Max AE	R^2	time
5	16167	0.021723	0.05279917	0.99368879	0.32
10	17971	0.00148809	0.00862147	0.99996947	0.52
15	17970	0.00149568	0.00824368	0.99996939	0.54
20	17970	0.00148612	0.00887056	0.99996929	0.70
25	17970	0.00141323	0.00702452	0.99996969	1.20
30	17970	0.0014922	0.00879338	0.99996891	1.18
35	17970	0.00149568	0.00824365	0.99996939	1.50
40	17970	0.00149482	0.00879952	0.99996883	1.26
45	17971	0.00149168	0.00892598	0.99996885	2.59
50	17970	0.0014906	0.008881	0.99996914	3.17
55	17971	0.00149547	0.00897727	0.99996872	1.91
60	17969	0.00141403	0.0081233	0.99997089	2.54
65	17971	0.00149537	0.00889789	0.99996907	4.68
70	17970	0.00141335	0.00690242	0.99996973	5.20
75	17971	0.0014971	0.00893743	0.99996869	3.45
80	17968	0.00145506	0.00712693	0.99997079	3.56
85	17968	0.00145391	0.00713031	0.99997082	4.01
90	17969	0.00140087	0.00854245	0.99997081	4.43
95	17971	0.00149713	0.00850362	0.99996918	6.29
100	17971	0.0014889	0.00890274	0.99996926	6.85
200	17969	0.00147694	0.0052855	0.9999691	8.69
300	17971	0.00149463	0.0089117	0.99996909	16.96
400	17971	0.0014937	0.00892051	0.99996911	24.64
500	17971	0.00149213	0.00896774	0.99996882	25.78
600	17969	0.00141401	0.00813485	0.99997088	25.56
700	17969	0.00140022	0.00849228	0.99997085	40.70
800	17971	0.00148903	0.008942	0.99996923	39.75
900	17969	0.00140125	0.00857156	0.99997078	29.04
1000	17968	0.00141364	0.00820403	0.99997077	54.83

TABLE B.6: Out-of-sample error metrics in case of the FITC method on the sinc dataset.

VFE

M	log-likelihood	Mean AE	Max AE	R^2	time
5	14687	0.02157777	0.05442483	0.99366071	0.26
10	17797	0.00148314	0.01527733	0.99995763	0.50
15	17941	0.00134701	0.00744574	0.9999717	1.09
20	17942	0.0013606	0.00817657	0.99997112	0.96
25	17943	0.00136144	0.00821661	0.99997109	1.32
30	17943	0.00136212	0.00824998	0.99997107	1.93
35	17943	0.00136215	0.0083228	0.99997104	1.81
40	17943	0.00136277	0.00829698	0.99997104	1.48
45	17943	0.00136295	0.00830496	0.99997103	2.12
50	17944	0.00136299	0.00833651	0.99997102	2.20
55	17944	0.00136314	0.00833905	0.99997101	3.13
60	17944	0.00136324	0.0083327	0.99997101	3.24
65	17944	0.00136339	0.00834705	0.999971	4.64
70	17944	0.00136349	0.00834855	0.999971	2.01
75	17944	0.0013634	0.00834649	0.999971	2.85
80	17944	0.00136344	0.00834911	0.999971	5.91
85	17944	0.00136346	0.00835076	0.999971	4.43
90	17944	0.00136354	0.00835291	0.999971	5.85
95	17944	0.0013638	0.00835996	0.99997099	4.70
100	17944	0.0013638	0.00836171	0.99997099	4.55
200	17944	0.00136413	0.00837541	0.99997098	9.19
300	17944	0.00136423	0.00838002	0.99997097	15.96
400	17944	0.00136433	0.00838355	0.99997097	17.80
500	17944	0.00136438	0.00838536	0.99997096	15.48
600	17944	0.0013644	0.00838648	0.99997096	39.35
700	17944	0.00136443	0.00838734	0.99997096	31.98
800	17945	0.00136443	0.00838776	0.99997096	55.44
900	17945	0.00136445	0.0083886	0.99997096	40.76
1000	17945	0.00136445	0.00838896	0.99997096	42.95

TABLE B.7: In-sample error metrics in case of the VFE method on the sinc dataset.

M	log-likelihood	Mean AE	Max AE	R^2	time
5	14687	0.02191061	0.05442511	0.99358794	0.00
10	17797	0.00146294	0.0152562	0.99996114	0.00
15	17941	0.00133761	0.00743802	0.9999727	0.01
20	17942	0.00134844	0.00816655	0.99997222	0.02
25	17943	0.00134923	0.00820645	0.9999722	0.03
30	17943	0.00134984	0.0082397	0.99997218	0.04
35	17943	0.00134986	0.00831224	0.99997216	0.05
40	17943	0.00135043	0.00828652	0.99997215	0.06
45	17943	0.0013506	0.00829448	0.99997215	0.11
50	17944	0.00135063	0.00832591	0.99997214	0.12
55	17944	0.00135079	0.00832843	0.99997213	0.15
60	17944	0.00135089	0.00832211	0.99997214	0.25
65	17944	0.00135101	0.00833641	0.99997213	0.24
70	17944	0.00135112	0.0083379	0.99997212	0.36
75	17944	0.00135104	0.00833585	0.99997213	0.38
80	17944	0.00135108	0.00833845	0.99997213	0.43
85	17944	0.00135111	0.0083401	0.99997213	0.49
90	17944	0.00135117	0.00834224	0.99997212	0.61
95	17944	0.0013514	0.00834928	0.99997212	0.57
100	17944	0.00135141	0.00835102	0.99997211	0.54
200	17944	0.00135175	0.00836467	0.9999721	1.99
300	17944	0.00135186	0.00836927	0.9999721	5.56
400	17944	0.00135195	0.00837279	0.9999721	11.46
500	17944	0.00135201	0.00837459	0.99997209	18.79
600	17944	0.00135203	0.00837571	0.99997209	16.63
700	17944	0.00135206	0.00837656	0.99997209	18.03
800	17945	0.00135207	0.00837699	0.99997209	42.87
900	17945	0.00135208	0.00837782	0.99997209	44.85
1000	17945	0.00135209	0.00837817	0.99997209	73.42

TABLE B.8: Out-of-sample error metrics in case of the VFE method on the sinc dataset.

SKI

method			SKI		
M	log-likelihood	Mean AE	Max AE	R^2	time
5	-2766	0.23766244	0.80488472	0.63210363	0.00
10	-1044	0.19788665	0.58621741	0.81504959	0.00
15	12106	0.04833559	0.11058767	0.96799256	0.01
20	17265	0.01047249	0.02632327	0.99846063	0.02
25	17610	0.00395782	0.01247529	0.99978282	0.04
30	17630	0.00250444	0.008977	0.99990794	0.06
35	17638	0.00158953	0.00804558	0.99995978	0.07
40	17880	0.00257322	0.00888747	0.99990615	0.11
45	17919	0.00128148	0.00622999	0.99997432	0.13
50	17678	0.00138329	0.00823215	0.99997013	0.17
55	17537	0.00288868	0.01074471	0.99987609	0.18
60	17610	0.00137575	0.00853562	0.99997049	0.16
65	16906	0.00422475	0.01930995	0.99973114	0.20
70	17742	0.00137622	0.00864696	0.99997045	0.26
75	17612	0.00137642	0.00865705	0.99997047	0.25
80	17287	0.00339482	0.01259642	0.99983649	0.29
85	17812	0.00136599	0.00843102	0.99997088	0.40
90	17793	0.00135747	0.00830784	0.99997114	0.43
95	17893	0.00136668	0.00841673	0.99997089	0.72
100	17279	0.0041205	0.02734016	0.99969353	0.63
200	17893	0.00137191	0.00855777	0.99997071	2.42
300	17600	0.00137982	0.00891471	0.99997026	5.13
400	15272	0.00670047	0.05057857	0.99928856	10.52
500	15148	0.0072856	0.0467018	0.9991955	17.83
600	17919	0.00136955	0.00848417	0.99997081	19.67
700	17928	0.00136662	0.0084208	0.9999709	26.70
800	17712	0.00136448	0.00837771	0.99997098	40.22
900	17887	0.00135853	0.00828972	0.99997114	34.09
1000	17859	0.00132323	0.00787663	0.99997225	43.79
10000	17897	0.00137278	0.00855628	0.9999707	155.80

TABLE B.9: In-sample error metrics in case of the SKI method on the sinc dataset.

M	log-likelihood	Mean AE	Max AE	R^2	time
5	-2766	0.24007628	0.80488467	10.70863677	0.00
10	-1044	0.19922318	0.58621834	0.83583501	0.00
15	12106	0.04912239	0.11058724	0.96747716	0.01
20	17265	0.01056446	0.02632299	0.99845042	0.02
25	17610	0.00395296	0.01247663	0.99978605	0.03
30	17630	0.00246586	0.00896219	0.99991163	0.05
35	17638	0.00157614	0.00803465	0.99996106	0.08
40	17880	0.00252801	0.00887327	0.99990972	0.08
45	17919	0.00129001	0.00622677	0.99997472	0.10
50	17678	0.00137107	0.00822205	0.99997126	0.17
55	17537	0.00292754	0.01074365	0.99987522	0.19
60	17610	0.00135803	0.00852455	0.99997176	0.15
65	16906	0.00425469	0.01930901	0.99973325	0.22
70	17742	0.00136543	0.00863556	0.99997159	0.19
75	17612	0.00136293	0.00864558	0.99997165	0.32
80	17287	0.00339782	0.01259635	0.99983889	0.36
85	17812	0.00135457	0.00842032	0.99997199	0.55
90	17793	0.00134564	0.00829758	0.99997225	0.64
95	17893	0.00135412	0.0084061	0.99997203	0.40
100	17279	0.00411176	0.02731309	0.99970785	0.78
200	17893	0.00135929	0.00854645	0.99997186	2.25
300	17600	0.00136786	0.00890206	0.99997146	6.84
400	15272	0.00686001	0.05072173	0.99926607	10.88
500	15148	0.00748222	0.04661886	0.9991759	14.29
600	17919	0.00135704	0.00847303	0.99997195	23.95
700	17928	0.00135424	0.00840989	0.99997204	25.24
800	17712	0.00135209	0.00836697	0.99997211	52.94
900	17887	0.00134653	0.00827931	0.99997225	64.43
1000	17859	0.00131364	0.00786776	0.99997328	64.66
10000	17897	0.00136033	0.00854486	0.99997185	132.03

TABLE B.10: Out-of-sample error metrics in case of the VFE method on the sinc dataset.

ME

M	log-likelihood	Mean AE	Max AE	R^2	time
5	-2	0.18543448	0.73152732	0.35215214	0.04
10	-7	0.04382552	0.1210511	0.97434358	0.07
15	-0	0.03356221	0.09776485	0.98572373	0.10
20	9	0.03048575	0.07751189	0.98879067	0.19
25	18	0.02702565	0.07438436	0.99012817	0.21
30	27	0.02626958	0.07197084	0.99118725	0.21
35	36	0.02390141	0.06382236	0.99318009	0.34
40	40	0.02227904	0.06339869	0.99385649	0.34
45	51	0.02282144	0.0640439	0.99352089	0.48
50	58	0.01892062	0.06374239	0.99528839	0.46
55	63	0.0154454	0.06354367	0.99676362	0.41
60	69	0.01470911	0.06360358	0.99694112	0.62
65	73	0.01267962	0.06077794	0.99776532	0.65
70	82	0.0131888	0.06195334	0.99773212	0.77
75	90	0.01181846	0.06206089	0.99796115	0.77
80	97	0.00933381	0.06196227	0.99840502	0.77
85	105	0.00721587	0.06275165	0.9988615	0.96
90	114	0.00720123	0.06231883	0.99898555	1.04
95	125	0.00738555	0.0622856	0.99898332	1.04
100	133	0.00740302	0.06265939	0.99895965	1.21
200	328	0.0070762	0.06511289	0.99905121	3.07
300	506	0.00905463	0.06600798	0.99863832	5.71
400	693	0.00671628	0.06642679	0.99907589	9.15
500	888	0.00688539	0.06673003	0.99904276	13.31
600	1082	0.00639854	0.06701568	0.99901974	18.46
700	1265	0.00613421	0.06642545	0.99914797	25.10
800	1459	0.00570428	0.06626452	0.99925747	33.39
900	1644	0.00497743	0.06646336	0.99929725	44.01
1000	1820	0.00453953	0.06648251	0.99931018	55.48

TABLE B.11: In-sample error metrics in case of the ME method on the sinc dataset.

M	log-likelihood	Mean AE	Max AE	R^2	time
5	-2	0.18932218	0.73152721	0.35134151	0.04
10	-7	0.04503656	0.1210511	0.97373733	0.07
15	-0	0.0343897	0.09776479	0.98539747	0.10
20	9	0.03114042	0.07751179	0.98860427	0.19
25	18	0.02755516	0.0743844	0.98997681	0.21
30	27	0.02676955	0.07197085	0.99107481	0.21
35	36	0.02437404	0.06367622	0.99306741	0.34
40	40	0.0228515	0.0632454	0.9937535	0.34
45	51	0.02332425	0.06388785	0.9934478	0.48
50	58	0.01923048	0.06356669	0.99527836	0.46
55	63	0.01565477	0.0633546	0.99676957	0.41
60	69	0.01478178	0.06341089	0.99696779	0.62
65	73	0.01270502	0.06052108	0.9977942	0.65
70	82	0.01323638	0.06169674	0.99775961	0.77
75	90	0.01179763	0.06180332	0.9980013	0.77
80	97	0.00924969	0.06171032	0.99846181	0.77
85	105	0.00712846	0.06249946	0.998897	0.96
90	114	0.00714501	0.06205999	0.99901226	1.04
95	125	0.00730057	0.06202618	0.99901419	1.04
100	133	0.00737248	0.06239859	0.9989835	1.21
200	328	0.00712201	0.06489222	0.99905656	3.07
300	506	0.00909528	0.0658026	0.99864928	5.79
400	693	0.00682375	0.06623666	0.99906582	9.15
500	888	0.00698314	0.06655713	0.99902996	13.31
600	1082	0.00645891	0.06686809	0.99900708	18.46
700	1265	0.00612947	0.06624724	0.99914104	25.10
800	1459	0.00573587	0.06607131	0.99924839	33.39
900	1644	0.00500479	0.06627498	0.99928819	44.01
1000	1820	0.00456646	0.0663063	0.99929717	55.48

TABLE B.12: Out-of-sample error metrics in case of the ME method on the sinc dataset.

MMI

M	log-likelihood	Mean AE	Max AE	R^2	time
5	-2	0.19393654	0.77307944	0.30266861	22.01
10	-5	0.0574218	0.13066505	0.95355062	49.15
15	-2	0.02224221	0.0561663	0.99270404	75.79
20	3	0.01876017	0.04467476	0.99527085	101.85
25	12	0.01798621	0.05598774	0.99543425	130.25
30	18	0.01841691	0.05554927	0.99518968	156.96
35	27	0.01538395	0.04744228	0.99642995	186.47
40	36	0.01238852	0.04560722	0.99725045	214.67
45	45	0.01238428	0.04494226	0.99748595	238.94
50	55	0.01253622	0.0446003	0.99766725	264.06
55	64	0.01295751	0.03944784	0.99753271	292.69
60	74	0.01215219	0.03840304	0.99771029	319.76
65	83	0.010578	0.03185543	0.99838976	346.10
70	92	0.01155432	0.03190874	0.99814853	376.90
75	101	0.01204016	0.03461431	0.99798372	401.91
80	109	0.0100182	0.03389753	0.99831343	426.78
85	115	0.01048806	0.0293536	0.99834336	455.87
90	125	0.01003838	0.02506804	0.99859778	484.28
95	135	0.00950035	0.02524885	0.99871198	509.64
100	146	0.00948327	0.02337092	0.99876294	536.41
200	327	0.00884771	0.02688212	0.99878911	1069.88
300	498	0.00631997	0.02370055	0.99937137	1587.26
400	681	0.00510943	0.03316376	0.9995673	2099.09
500	847	0.00353343	0.02738125	0.99976229	2599.03
600	1025	0.00428554	0.04190526	0.99958656	3110.30
700	1206	0.00453174	0.0477211	0.99950215	3578.65
800	1372	0.00638223	0.05735533	0.99917266	3964.75
900	1529	0.00611358	0.05035897	0.99922174	4378.77
1000	1707	0.00631532	0.04668259	0.99922001	4799.27

TABLE B.13: In-sample error metrics in case of the MMI method on the sinc dataset.

M	log-likelihood	Mean AE	Max AE	R^2	time
5	-2	0.19707343	0.77307842	0.30073987	22.01
10	-5	0.05742635	0.13066509	0.95446365	49.15
15	-2	0.0217686	0.05616623	0.99307711	75.79
20	3	0.01864092	0.04467478	0.99538668	101.85
25	12	0.01775461	0.05598772	0.99562351	130.25
30	18	0.01829773	0.05554917	0.99536362	156.96
35	27	0.01517947	0.04744227	0.99658414	186.47
40	36	0.01212132	0.04560721	0.99739879	214.67
45	45	0.01216022	0.04494219	0.99761185	238.94
50	55	0.01229039	0.04444235	0.99778554	264.06
55	64	0.01265953	0.03944784	0.99765498	292.69
60	74	0.01187212	0.03840284	0.99782202	319.76
65	83	0.01033177	0.03185543	0.99847014	346.10
70	92	0.01135879	0.03190874	0.99822503	376.90
75	101	0.01184544	0.03461431	0.9980711	401.91
80	109	0.00985201	0.03389751	0.99838812	426.78
85	115	0.0103172	0.02935361	0.99841397	455.87
90	125	0.00989711	0.02506804	0.99865098	484.28
95	135	0.00934418	0.02524886	0.99876615	509.64
100	146	0.00933751	0.02337089	0.99881199	536.41
200	327	0.00864642	0.02688212	0.99884433	1069.86
300	498	0.00627131	0.02363621	0.99938488	1587.26
400	681	0.00512678	0.03308194	0.99956928	2099.09
500	847	0.00355089	0.02730718	0.99976352	2599.03
600	1025	0.00424306	0.04187528	0.99961676	3110.30
700	1206	0.00447505	0.04768839	0.99954317	3578.63
800	1372	0.00631297	0.05731693	0.99922996	3964.75
900	1529	0.00599254	0.05033048	0.99927563	4378.77
1000	1707	0.00618387	0.04665776	0.99926957	4799.27

TABLE B.14: Out-of-sample error metrics in case of the MMI method on the sinc dataset.

RU

M	log-likelihood		Mean AE		Max AE		R^2	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-2	0	0.0364	0.65909625	0.164	0.50361807	0.20224	
10	-9	0	0.0425	0.38778933	0.164	0.81204787	0.12915	
15	1	0	0.0303	0.24944018	0.155	0.92220334	0.08684	
20	8	0	0.0149	0.17853251	0.095	0.96516769	0.03215	
25	15	0	0.0154	0.14038716	0.097	0.97607407	0.03498	
30	21	0	0.0083	0.11156898	0.077	0.98647077	0.01597	
35	38	0	0.0056	0.09131751	0.056	0.99149431	0.00679	
40	46	0	0.0066	0.08669047	0.069	0.99250153	0.00934	
45	44	0	0.0046	0.07595616	0.050	0.99394228	0.00495	
50	61	0	0.0044	0.06748399	0.040	0.99485231	0.00442	
55	67	0	0.0032	0.05552961	0.025	0.99639353	0.00170	
60	81	0	0.0032	0.05304931	0.025	0.99667145	0.00162	
65	89	0	0.003	0.06182251	0.043	0.9962052	0.00308	
70	88	0	0.0030	0.05274497	0.030	0.99676429	0.00274	
75	105	0	0.0031	0.0514035	0.027	0.99697603	0.00191	
80	103	0	0.0028	0.05013763	0.024	0.99741819	0.00114	
85	110	0	0.0027	0.05315087	0.030	0.99736081	0.00128	
90	124	0	0.0025	0.04633058	0.020	0.99760139	0.00099	
95	137	0	0.0026	0.04585622	0.019	0.99776408	0.00096	
100	148	0	0.0021	0.0424709	0.0178	0.99799665	0.00081	
200	323	0	0.0017	0.03042285	0.0130	0.99898448	0.00043	
300	484	0	0.0014	0.02661855	0.0124	0.99927854	0.00029	
400	662	0	0.0012	0.02196403	0.007	0.99950208	0.00018	
500	838	0	0.0009	0.02041042	0.006	0.99957865	0.00013	
600	1035	0	0.0009	0.01975732	0.008	0.99965248	0.00012	
700	1202	0	0.0008	0.01856492	0.007	0.99969388	0.00010	
800	1396	0	0.0009	0.0185336	0.007	0.99970275	0.00012	
900	1565	0	0.0009	0.01690154	0.007	0.99974042	0.00010	
1000	1753	0	0.0007	0.01568948	0.006	0.99976462	0.00008	

TABLE B.15: In-sample error metrics in case of the RU method on the sinc dataset.

M	log-likelihood		Mean AE		Max AE		R^2
	Mean	Std	Mean	Std	Mean	Std	Mean
5	-2	0	0.0359	0.65909275	0.164	0.50937223	0.20235
10	-9	0	0.0422	0.38776683	0.164	0.81479966	0.12718
15	1	0	0.0300	0.24939902	0.155	0.92319926	0.08608
20	8	0	0.0148	0.17853251	0.095	0.96516769	0.03150
25	15	0	0.0154	0.14035325	0.097	0.97654333	0.03462
30	21	0	0.0081	0.11151835	0.077	0.98681545	0.01537
35	38	0	0.0055	0.09127016	0.056	0.99166854	0.00661
40	46	0	0.0065	0.08663841	0.069	0.992705	0.00899
45	44	0	0.0045	0.0759	0.050	0.9940857	0.00470
50	61	0	0.0044	0.0674258	0.040	0.99494954	0.00426
55	67	0	0.0032	0.0554833	0.025	0.9964412	0.00168
60	81	0	0.0032	0.05300257	0.025	0.99672396	0.00160
65	89	0	0.0036	0.06178169	0.043	0.99629927	0.00293
70	88	0	0.0030	0.05270263	0.030	0.99682593	0.00257
75	105	0	0.00314	0.05136519	0.027	0.99704324	0.00180
80	103	0	0.0028	0.05010034	0.024	0.99745855	0.00112
85	110	0	0.0027	0.05308592	0.030	0.99741643	0.00122
90	124	0	0.0025	0.04628951	0.020	0.99763501	0.00099
95	137	0	0.0026	0.04581677	0.019	0.99780906	0.00095
100	148	0	0.0021	0.04242491	0.017	0.99803275	0.00080
200	323	0	0.0017	0.03038505	0.013	0.99900321	0.00042
300	484	0	0.0014	0.02658694	0.012	0.99929103	0.00028
400	662	0	0.0012	0.02194189	0.007	0.99951003	0.00018
500	838	0	0.0009	0.02038533	0.006	0.99958614	0.00013
600	1035	0	0.0009	0.01972826	0.007	0.99966034	0.00012
700	1202	0	0.0008	0.01853344	0.006	0.99970022	0.00010
800	1396	0	0.0009	0.01850535	0.007	0.99970942	0.00011
900	1565	0	0.0009	0.01687427	0.006	0.99974551	0.00010
1000	1753	0	0.0007	0.01567081	0.006	0.99976983	0.00008

TABLE B.16: Out-of-sample error metrics in case of the RU method on the sinc dataset.

Continuation Value - Bermudan Put

Results related to Section 5.2.

VFE

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-115323	306	0.13432294	0.0146	4.20194969	0.5492	0.9983793	0.0002	8.63	3.21
10	-111629	300	0.05690284	0.0018	3.34085516	0.5955	0.99976112	0.0000	8.66	2.57
20	-110862	274	0.01810467	0.0011	0.92883407	0.5164	0.99997615	0.0000	9.81	2.75
40	-110880	274	0.00967454	0.0013	0.48829547	0.2839	0.99999262	0.0000	29.44	6.29
80	-110849	259	0.00969784	0.0014	0.41092232	0.2140	0.99999252	0.0000	74.4	29.69

TABLE B.17: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-113323	1364	0.25545325	0.0347	4.33293242	0.6007	0.99614644	0.0007	7.30	2.51
10	-102132	341	0.11023685	0.0013	4.96605704	1.0700	0.99934845	0.0000	9.67	2.55
20	-100047	348	0.03458839	0.0031	3.38080061	0.9899	0.99992982	0.0000	9.70	3.15
40	-99752	343	0.01087566	0.0012	0.68433346	0.6661	0.99999289	0.0000	24.91	7.90
80	-99692	329	0.01014562	0.0012	0.80984769	1.1027	0.99999314	0.0000	73.44	25.77

TABLE B.18: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-170432	254	0.33256577	0.9972	6.90656069	0.3896	0.99721842	0.0002	8.37	2.29
10	-165993	278	0.10128177	0.0015	7.05699897	0.1872	0.99968755	0.0000	9.86	2.61
20	-165139	310	0.03562613	0.0018	4.24846637	2.6328	0.99996165	0.0000	10.19	2.47
40	-165092	325	0.01546232	0.0022	1.39563334	2.1403	0.99999216	0.0000	32.40	6.00
80	-165014	304	0.01568638	0.0023	1.49930209	2.2818	0.99999162	0.0000	71.82	27.90

TABLE B.19: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-166914	339	0.42828118	0.0029	8.66662364	0.0298	0.99650616	0.0000	8.08	2.67
10	-159081	339	0.17055218	0.0200	8.66860392	0.0296	0.99918635	0.0001	9.22	2.98
20	-155973	326	0.07814909	0.0090	8.48171917	0.8944	0.99984555	0.0000	9.01	2.93
40	-155271	318	0.02705529	0.0045	5.84741024	3.6421	0.99997607	0.0000	27.04	5.72
80	-155288	340	0.01682747	0.0018	4.53338919	4.07623	0.99998654	0.0000	81.74	21.32

TABLE B.20: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-108702	313	0.15336173	0.0153	3.21739705	0.9016	0.9975007	0.0003	6.75	2.40
10	-104133	329	0.07445897	0.0028	4.56983885	1.2332	0.99947845	0.0000	9.35	2.51
20	-103204	365	0.01830223	0.0012	1.24575451	0.8864	0.99996834	0.0000	9.38	2.54
40	-103145	351	0.00909887	0.0014	0.50860658	0.2684	0.99999109	0.0000	27.25	5.35
80	-103064	388	0.00919951	0.0014	0.44681682	0.2414	0.99999107	0.0000	78.78	34.63

TABLE B.21: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-107540	140	0.25601862	0.0452	7.50678895	1.7280	0.99558842	0.0010	7.23	2.57
10	-96299	376	0.10521484	0.0025	4.75352275	1.2129	0.99926438	0.0000	8.88	2.44
20	-93961	379	0.03975772	0.0019	3.1509920	0.4938	0.99989345	0.0000	7.98	2.03
40	-93492	359	0.01095227	0.0011	0.87967942	0.9040	0.99999081	0.0000	20.33	4.59
80	-93468	350	0.00975132	0.0013	0.72067636	0.8079	0.99999235	0.0000	73.83	27.43

TABLE B.22: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-167823	298	0.32762285	0.0017	6.46512568	0.4594	0.99702071	0.0000	8.55	2.68
10	-163308	330	0.10244986	0.0021	6.17392665	0.0434	0.99964104	0.0000	8.73	2.30
20	-162362	347	0.03675706	0.0020	4.83346118	1.8410	0.99995326	0.0000	10.18	2.68
40	-162287	376	0.01530334	0.0025	1.21563428	1.7594	0.99999174	0.0000	30.40	6.10
80	-162263	353	0.01518682	0.0023	1.20357501	1.7658	0.99999160	0.0000	73.61	26.19

TABLE B.23: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-165946	283	0.43049406	0.0022	7.85738056	0.0283	0.99618511	0.0000	7.69	2.23
10	-157824	364	0.17618113	0.0208	7.85384497	0.0306	0.99908562	0.0001	9.09	2.66
20	-154322	381	0.08875623	0.0104	7.79298404	0.4129	0.99981204	0.0000	9.27	3.20
40	-153647	351	0.02871037	0.0035	6.03959254	2.9331	0.99997228	0.0000	25.72	5.33
80	-153548	341	0.01600179	0.0019	5.18618317	3.5369	0.99998453	0.0000	80.43	20.81

TABLE B.24: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-100401	409	0.16487204	0.0144	2.46903422	1.0974	0.99632702	0.0004	7.39	2.72
10	-94410	421	0.17618113	0.0016	7.85384497	1.4310	0.99908562	0.0000	9.31	2.82
20	-93465	382	0.08875623	0.0019	7.79298404	0.5876	0.99981204	0.0000	8.37	2.17
40	-93345	421	0.02871037	0.0013	6.03959254	0.3140	0.99997228	0.0000	24.90	4.53
80	-93368	343	0.01600179	0.0013	5.18618317	0.3064	0.99998453	0.0000	77.17	32.66

TABLE B.25: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-100917	770	0.24038043	0.0264	7.96214975	1.7470	0.99505573	0.0007	7.78	2.41
10	-89455	397	0.09745350	0.0035	4.73262027	1.2256	0.99918876	0.0000	9.48	2.80
20	-86739	402	0.03882846	0.0011	2.69527555	0.4216	0.99987102	0.0000	8.23	2.37
40	-86243	409	0.01117587	0.0013	0.87255885	0.8290	0.99998778	0.0000	20.71	5.78
80	-86223	390	0.00915326	0.0012	0.60686592	0.5253	0.99999102	0.0000	65.88	20.59

TABLE B.26: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-164718	328	0.3157927	0.0019	6.50988571	0.7919	0.99685567	0.0000	7.90	2.58
10	-160195	375	0.10288734	0.0023	5.36096559	0.0217	0.99957913	0.0000	9.01	2.79
20	-159055	340	0.03844238	0.0024	4.85639108	1.095	0.99994202	0.0000	10.48	3.11
40	-158948	344	0.01485763	0.0025	1.7121116	1.9542	0.99999039	0.0000	28.97	4.54
80	-158905	356	0.01456976	0.0022	1.47017781	1.7776	0.99999112	0.0000	78.13	24.19

TABLE B.27: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-164696	359	0.42544263	0.0125	7.12486994	0.0249	0.99598901	0.0003	8.09	2.82
10	-156336	381	0.18035653	0.0204	7.12848521	0.0291	0.99898859	0.0001	9.35	2.43
20	-152518	340	0.09316215	0.0078	7.11004979	0.0932	0.99977930	0.0000	9.06	2.75
40	-151638	410	0.03067318	0.0027	6.02343506	2.2056	0.99996667	0.0000	25.08	5.45
80	-151499	357	0.01605621	0.0018	5.32300968	2.9512	0.99998351	0.0000	79.53	25.13

TABLE B.28: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-83091	70000	0.16952838	0.0332	6.74823717	2.0863	0.99422872	0.0033	7.11	2.62
10	-83089	448	0.06131753	0.0015	3.87174594	1.3860	0.99929906	0.0000	9.00	2.46
20	-81905	412	0.02150919	0.0010	1.65025943	0.5150	0.99991212	0.0000	8.02	2.06
40	-81842	453	0.00752196	0.0012	0.48179458	0.3079	0.99998589	0.0000	22.79	4.50
80	-81744	452	0.00754794	0.0013	0.51682675	0.3180	0.99998559	0.0000	74.67	25.53

TABLE B.29: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-93351	510	0.21423839	0.0215	8.41718881	2.0255	0.99472242	0.0007	7.60	2.29
10	-81828	412	0.08893690	0.0029	5.0074352	1.4172	0.99909837	0.0000	9.31	2.33
20	-78766	431	0.03670870	0.0014	2.48689725	0.5525	0.99984829	0.0000	7.88	2.12
40	-78121	451	0.01119737	0.0012	0.73828574	0.5895	0.99998413	0.0000	19.04	5.52
80	-78178	444	0.00811433	0.0010	0.67403662	0.5247	0.99999019	0.0000	68.42	18.06

TABLE B.30: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-161178	333	0.30301681	0.0018	6.67485638	0.6872	0.99666001	0.0000	7.61	2.54
10	-156609	347	0.10354263	0.0019	4.65764461	0.1168	0.99950746	0.0000	9.13	2.47
20	-155342	378	0.04111474	0.0027	4.43093525	0.6318	0.99992734	0.0000	9.46	2.47
40	-155212	352	0.01542604	0.0028	1.58190803	1.6378	0.99998830	0.0000	28.07	5.42
80	-155131	383	0.01496856	0.0026	1.16442713	1.2570	0.99998966	0.0000	79.30	24.55

TABLE B.31: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-163166	411	0.42102008	0.0170	6.46215674	0.0295	0.99571983	0.0004	7.60	2.68
10	-154632	365	0.1744398	0.0172	6.46339731	0.0267	0.99897660	0.0001	9.12	2.90
20	-150377	370	0.09726191	0.0042	6.45936304	0.0270	0.99975332	0.0000	8.71	2.66
40	-149521	378	0.03191160	0.0022	5.83726464	1.5655	0.99996091	0.0000	25.07	7.34
80	-149411	418	0.01543913	0.0021	5.03155084	2.5184	0.99998379	0.0000	77.79	19.63

TABLE B.32: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-77365	526	0.13341328	0.0110	7.82946417	1.5186	0.99419185	0.0007	7.25	2.46
10	-70154	430	0.0547805	0.0010	3.28376837	1.0402	0.99915965	0.0000	8.59	2.39
20	-68810	496	0.01916242	0.0013	1.4344297	0.4918	0.9998938	0.0000	8.65	2.28
40	-68614	479	0.00640639	0.0010	0.44306026	0.3004	0.99998218	0.0000	23.49	5.91
80	-68574	582	0.00656072	0.0011	0.47906048	0.2533	0.99998079	0.0000	81.97	32.81

TABLE B.33: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-85211	496	0.19563202	0.0067	8.82933974	1.5122	0.99405421	0.0003	8.08	2.68
10	-73674	464	0.07997007	0.0036	4.82424893	1.3675	0.99900303	0.0000	9.20	3.30
20	-70025	470	0.03419116	0.0011	2.07078534	0.5507	0.99982283	0.0000	8.08	2.15
40	-69406	494	0.01098718	0.0010	0.86452235	0.5135	0.99997893	0.0000	17.75	4.55
80	-69412	417	0.00721437	0.0010	0.56876061	0.4791	0.99998856	0.0000	65.68	16.89

TABLE B.34: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-157363	352	0.28999211	0.0017	6.92137471	0.9592	0.99643805	0.0000	7.53	2.72
10	-152820	382	0.10402011	0.0022	4.03721148	0.1334	0.99942263	0.0000	9.29	2.66
20	-151322	395	0.04379968	0.0028	3.93627156	0.3072	0.99990806	0.0000	9.06	2.55
40	-151058	364	0.01527153	0.0026	1.77394868	1.4974	0.99998623	0.0000	26.34	5.60
80	-151043	355	0.01438716	0.0028	1.57725522	1.4205	0.99998789	0.0000	82.58	32.65

TABLE B.35: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
5	-161511	455	0.4157561	0.0210	5.90771584	0.1805	0.99543051	0.0004	7.89	2.96
10	-152634	411	0.17645334	0.0175	5.8556139	0.0256	0.99888728	0.0001	9.44	2.96
20	-148039	388	0.09682039	0.0025	5.85832130	0.0246	0.99973259	0.0000	8.58	2.55
40	-147123	394	0.03206126	0.0017	5.72052989	0.6646	0.99995768	0.0000	22.73	5.12
80	-147014	430	0.01528469	0.0020	4.78156175	2.0910	0.99998329	0.0000	79.62	18.30

TABLE B.36: In-sample error metrics for the VFE method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

SKI

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-11171	236	0.02348796	0.9410	0.99993234	0.0125	0.40573911	0.0000	86.30	26.53

TABLE B.37: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-10614	685	0.04865295	0.0083	1.19274856	0.3077	0.99985114	0.0000	137.34	15.15

TABLE B.38: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-16437	254	0.05091184	0.0224	1.77168807	0.7168	0.99989607	0.0000	89.56	20.59

TABLE B.39: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15829	440	0.09207003	0.0076	1.49750811	0.5225	0.99982994	0.0000	113.02	11.20

TABLE B.40: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-10567	414	0.03015784	0.0132	1.13519556	0.3605	0.99986609	0.0001	101.73	30.71

TABLE B.41: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-9904	693	0.04342493	0.0054	1.08943762	0.3105	0.99985761	0.0000	139.66	14.86

TABLE B.42: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-16166	243	0.05684489	0.0192	1.57689492	0.5609	0.99987172	0.0000	95.12	20.54

TABLE B.43: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15610	193	0.08796388	0.0062	1.41244121	0.5585	0.99983583	0.0000	113.78	10.39

TABLE B.44: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-10049	812	0.03238061	0.0129	1.08319521	0.3424	0.99978521	0.0002	117.17	28.31

TABLE B.45: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-9406	470	0.04242062	0.0062	1.05288815	0.2798	0.99982116	0.0000	140.54	15.49

TABLE B.46: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15892	302	0.05989392	0.0143	1.48555804	0.5782	0.99985495	0.0000	104.67	17.25

TABLE B.47: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15379	0	0.08393862	0.6167	1.49958331	0.5782	0.99984156	0.0000	114.27	10.59

TABLE B.48: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-8963	0	0.03313712	0.3603	1.1781852	0.5782	0.99970361	0.0001	130.98	17.64

TABLE B.49: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-8785	443	0.04097914	0.0079	1.11718795	0.2968	0.99976418	0.0001	139.92	16.29

TABLE B.50: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15620	1216	0.06166912	0.0121	1.43739914	0.6271	0.99982948	0.0001	108.88	14.24

TABLE B.51: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15214	238	0.07953481	0.0050	1.31256018	0.4506	0.99984483	0.0000	115.59	9.68

TABLE B.52: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-8229	313	0.0417056	0.0087	1.59138907	0.3678	0.99901338	0.0005	129.38	12.31

TABLE B.53: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-8322	342	0.04320576	0.0069	1.29373739	0.3487	0.99960108	0.0001	141.70	12.61

TABLE B.54: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15139	297	0.05757976	0.0070	1.33032458	0.4996	0.99983756	0.0000	110.81	10.58

TABLE B.55: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-15011	255	0.07494083	0.0072	1.34082644	0.4897	0.99985261	0.0000	115.72	11.20

TABLE B.56: In-sample error metrics for the SKI method in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Linear regression

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.23989264	0.0082	9.71971165	3.6564	0.99589716	0.0004	0.04	0.01

TABLE B.57: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.39779119	0.0193	18.56740754	6.2392	0.9921359	0.0009	0.02	0.01

TABLE B.58: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.50531203	0.0202	24.00460813	9.5757	0.99348577	0.0008	0.02	0.00

TABLE B.59: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
1.18718252	0.0101	18.28645778	8.2092	0.97733513	0.0003	0.02	0.00

TABLE B.60: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 36$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.25814969	0.0131	12.08053463	4.1988	0.99405516	0.0007	0.03	0.00

TABLE B.61: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.40520975	0.0085	15.97785982	4.4914	0.99000505	0.0003	0.02	0.00

TABLE B.62: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.53552586	0.0388	30.82606504	10.8669	0.9917618	0.0014	0.02	0.00

TABLE B.63: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
1.168016	0.0069	14.87485881	8.2457	0.97648133	0.0003	0.02	0.00

TABLE B.64: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 38$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.26276347	0.0055	12.31728577	2.8407	0.991456	0.0003	0.02	0.00

TABLE B.65: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.39956049	0.0196	18.36903964	5.9830	0.98789848	0.0011	0.02	0.00

TABLE B.66: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.59157594	0.0553	34.48178117	10.9219	0.98888348	0.0021	0.02	0.00

TABLE B.67: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
1.14175044	0.0099	17.39859893	9.9450	0.9756607	0.0004	0.02	0.00

TABLE B.68: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 40$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.25133502	0.0054	15.58881883	4.6799	0.98809457	0.0005	0.02	0.00

TABLE B.69: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.41827952	0.0305	23.41086516	5.9140	0.98314268	0.0025	0.02	0.00

TABLE B.70: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.63547327	0.0560	35.93860343	9.3157	0.98582736	0.0025	0.02	0.00

TABLE B.71: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
1.12680611	0.0254	30.52090191	16.5161	0.97429009	0.0012	0.02	0.00

TABLE B.72: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 42$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.22719862	0.0097	13.4191642	4.1141	0.98577521	0.0009	0.02	0.00

TABLE B.73: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.2$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.43582748	0.0277	23.24357222	6.0478	0.97587461	0.0034	0.02	0.00

TABLE B.74: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.2$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.69226273	0.0653	39.72504615	13.3534	0.98087177	0.0046	0.02	0.00

TABLE B.75: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.4$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
1.12093733	0.0409	42.40980774	23.8037	0.97218987	0.0026	0.02	0.00

TABLE B.76: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Bermudan Put with $S_{t_0} = 44$, $r = 0.06$, $\sigma = 0.4$, $T = 2$, $K = 40$.

Continuation Value - Geometric Bermudan Basket Put

Results related to Section 5.3.

VFE

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
40	-112967	361	0.2937975	8.76	1.22138941	0.00	0.98442853	0.00	470.51	19.40

TABLE B.77: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
40	-151499	606	0.57626496	11.51	1.01465011	0.00	0.96264223	0.00	593.61	34.34

TABLE B.78: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
40	-82227	930	0.24600697	6.74	1.11752664	0.00	0.98228196	0.00	449.77	17.83

TABLE B.79: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{44, 44, 121, 16, 16, 121\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
40	-139903	264	0.54948562	10.75	3.15817292	0.28	0.96437910	0.00	488.91	40.08

TABLE B.80: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{44, 44, 121, 16, 16, 121\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 2$, $K = 40$.

SKI

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-171213	697	2.84498958	15.75	4.99640825	0.02	0.74334325	0.25	44.90	0.52

TABLE B.81: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-259945	318	3.44433084	20.05	7.77677043	0.01	0.84087422	0.16	45.25	0.41

TABLE B.82: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-464926	386	0.86457555	12.96	67.25570763	0.00	0.84998194	0.05	47.11	0.58

TABLE B.83: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{44, 44, 121, 16, 16, 121\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 2$, $K = 40$.

M	log-likelihood		Mean AE		Max AE		R^2		time	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
10^5	-661371	546	1.39763649	17.05	73.10885677	0.00	0.58646864	0.06	47.64	0.27

TABLE B.84: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{44, 44, 121, 16, 16, 121\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 2$, $K = 40$.

Linear regression

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.10403400	3.03	0.99795668	0.01	0.65854412	0.00	111.19	5.02

TABLE B.85: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.13845224	5.41	0.99581288	0.01	0.7541375	0.00	773.93	0.75

TABLE B.86: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{36, 36, 144, 9, 16, 81\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 1$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.31341389	23.44	0.95771014	0.24	0.68288676	0.08	798.15	0.81

TABLE B.87: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{44, 44, 121, 16, 16, 121\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 2$, $K = 40$.

Mean AE		Max AE		R^2		time	
Mean	Std	Mean	Std	Mean	Std	Mean	Std
0.43512548	23.24	1.3770118	0.04	0.57587461	0.03	2076.16	210.63

TABLE B.88: In-sample error metrics for linear regression in case of the continuation value estimation problem of a Geometric Bermudan Put with $S_{t_0} = \{44, 44, 121, 16, 16, 121\}$, $r = 0.06$, $\sigma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.05\}$, $T = 2$, $K = 40$.

Option Pricing - Bermudan Put

Results related to Section 5.4.

S_{t_0}	σ	T	PSOR	LSM (Laguerre)		LSM (Polynomial)		VFE		SKI	
				Mean	Std	Mean	Std	Mean	Std	Mean	Std
36	0.2	1	4.478	4.472	0.010	4.577	0.012	4.478	0.004	4.478	0.007
36	0.2	2	4.840	4.821	0.012	5.036	0.058	4.846	0.010	4.849	0.014
36	0.4	1	7.101	7.091	0.020	7.412	0.030	7.112	0.015	7.109	0.018
36	0.4	2	8.508	8.488	0.024	9.216	0.018	8.522	0.025	8.519	0.029
38	0.2	1	3.250	3.244	0.009	3.406	0.008	3.254	0.008	3.257	0.008
38	0.2	2	3.745	3.735	0.011	3.998	0.011	3.752	0.006	3.758	0.011
38	0.4	1	6.148	6.139	0.019	6.506	0.043	6.152	0.019	6.155	0.024
38	0.4	2	7.670	7.669	0.022	8.385	0.027	7.694	0.025	7.692	0.028
40	0.2	1	2.314	2.313	0.009	2.489	0.014	2.321	0.007	2.324	0.010
40	0.2	2	2.885	2.879	0.010	3.164	0.020	2.891	0.005	2.893	0.014
40	0.4	1	5.312	5.308	0.018	5.725	0.036	5.317	0.019	5.320	0.020
40	0.4	2	6.920	6.921	0.022	7.639	0.042	6.930	0.029	6.928	0.029
42	0.2	1	1.617	1.617	0.007	1.796	0.012	1.619	0.006	1.619	0.008
42	0.2	2	2.212	2.206	0.010	2.523	0.020	2.221	0.009	2.219	0.011
42	0.4	1	4.582	4.588	0.017	5.049	0.039	4.592	0.013	4.594	0.015
42	0.4	2	6.248	6.243	0.021	6.999	0.044	6.277	0.015	6.277	0.018
44	0.2	1	1.110	1.118	0.007	1.282	0.013	1.112	0.007	1.114	0.007
44	0.2	2	1.690	1.675	0.009	2.028	0.019	1.694	0.007	1.693	0.010
44	0.4	1	3.948	3.957	0.017	4.459	0.044	3.955	0.015	3.953	0.018
44	0.4	2	5.647	5.622	0.021	6.443	0.069	5.662	0.020	5.663	0.023

TABLE B.89: Bermudan Put numerical results. PSOR benchmark and LSM results are from [12].

Bibliography

[1] J. Mercer, "Xvi. functions of positive and negative type, and their connection the theory of integral equations", *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, vol. 209, no. 441-458, pp. 415–446, 1909.

[2] F. Black and M. Scholes, "The pricing of options and corporate liabilities", *Journal of political economy*, vol. 81, no. 3, pp. 637–654, 1973.

[3] G. Matheron, "The intrinsic random functions and their applications", *Advances in applied probability*, vol. 5, no. 3, pp. 439–468, 1973.

[4] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i", *Mathematical programming*, vol. 14, no. 1, pp. 265–294, 1978.

[5] M. D. McKay, R. J. Beckman, and W. J. Conover, "Comparison of three methods for selecting values of input variables in the analysis of output from a computer code", *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[6] F. J. Narcowich and J. D. Ward, "Norm estimates for the inverses of a general class of scattered-data radial-function interpolation matrices", *Journal of Approximation Theory*, vol. 69, no. 1, pp. 84–109, 1992.

[7] C.-W. Ko, J. Lee, and M. Queyranne, "An exact algorithm for maximum entropy sampling", *Operations Research*, vol. 43, no. 4, pp. 684–691, 1995.

[8] R. M. Neal, "Priors for infinite networks", in *Bayesian Learning for Neural Networks*, Springer, 1996, pp. 29–53.

[9] C. K. Williams and C. E. Rasmussen, "Gaussian processes for regression", in *Advances in neural information processing systems*, 1996, pp. 514–520.

[10] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: Active data selection and test point rejection", in *Mustererkennung 2000*, Springer, 2000, pp. 27–34.

[11] E. Jones, T. Oliphant, P. Peterson, *et al.*, *SciPy: Open source scientific tools for Python*, [Online; accessed <today>], 2001–. [Online]. Available: http://www.scipy.org/.

[12] F. A. Longstaff and E. S. Schwartz, "Valuing american options by simulation: A simple least-squares approach", *The review of financial studies*, vol. 14, no. 1, pp. 113–147, 2001.

[13] A. J. Smola and P. L. Bartlett, "Sparse greedy gaussian process regression", in *Advances in neural information processing systems*, 2001, pp. 619–625.

[14] E. Clément, D. Lamberton, and P. Protter, "An analysis of a least squares regression method for american option pricing", *Finance and Stochastics*, vol. 6, no. 4, pp. 449–471, 2002.

[15] B. Schölkopf, A. J. Smola, F. Bach, *et al.*, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002.

[16] R. Herbrich, N. D. Lawrence, and M. Seeger, "Fast sparse gaussian process methods: The informative vector machine", in *Advances in neural information processing systems*, 2003, pp. 625–632.

[17] V. V. Piterbarg, "A practitioner's guide to pricing and hedging callable libor exotics in forward libor models", 2003.

[18] M. Seeger, C. Williams, and N. Lawrence, "Fast forward selection to speed up sparse gaussian process regression", Tech. Rep., 2003.

[19] P. Glasserman, B. Yu, *et al.*, "Number of paths versus number of basis functions in american option pricing", *The Annals of Applied Probability*, vol. 14, no. 4, pp. 2090–2119, 2004.

[20] C. E. Rasmussen, "Gaussian processes in machine learning", in *Advanced lectures on machine learning*, Springer, 2004, pp. 63–71.

[21] M. Seeger, "Gaussian processes for machine learning", *International journal of neural systems*, vol. 14, no. 02, pp. 69–106, 2004.

[22] V. Bally, G. Pagès, and J. Printems, "A quantization tree method for pricing and hedging multidimensional american options", *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, vol. 15, no. 1, pp. 119–168, 2005.

[23] P. Dupuis, H. Wang, *et al.*, "On the convergence from discrete to continuous time in an optimal stopping problem", *The Annals of Applied Probability*, vol. 15, no. 2, pp. 1339–1366, 2005.

[24] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression", *Journal of Machine Learning Research*, vol. 6, no. Dec, pp. 1939–1959, 2005.

[25] N. S. Rasmussen, "Control variates for monte carlo valuation of american options", *Journal of Computational Finance*, vol. 9, no. 1, 2005.

[26] Y. Zhang, W. E. Leithead, and D. J. Leith, "Time-series gaussian process regression based on toeplitz computation of o (n 2) operations and o (n)-level storage", in *Proceedings of the 44th IEEE Conference on Decision and Control*, IEEE, 2005, pp. 3711–3716.

[27] C. M. Bishop, *Pattern recognition and machine learning*. Springer Science+ Business Media, 2006.

[28] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[29] R. Seydel and R. Seydel, *Tools for computational finance*. Springer, 2006, vol. 3.

[30] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs", in *Advances in neural information processing systems*, 2006, pp. 1257–1264.

[31] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, 3. MIT Press Cambridge, MA, 2006, vol. 2.

[32] A. Krause and C. Guestrin, "Nonmyopic active learning of gaussian processes: An exploration-exploitation approach", in *Proceedings of the 24th international conference on Machine learning*, ACM, 2007, pp. 449–456.

[33] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies", *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[34] K. B. Petersen, M. S. Pedersen, *et al.*, "The matrix cookbook", *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008.

[35] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes", in *Artificial Intelligence and Statistics*, 2009, pp. 567–574.

[36] R. Korn, E. Korn, and G. Kroisandt, *Monte Carlo methods and models in finance and insurance*. CRC press, 2010.

[37] I. V. Oseledets, "Tensor-train decomposition", *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[38] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

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

[40] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[41] M. L. Stein, *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[42] T. Suzuki, "Pac-bayesian bound for gaussian process regression and multiple kernel additive model", in *Conference on Learning Theory*, 2012, pp. 8–1.

[43] E. Gilboa, Y. Saatçi, and J. P. Cunningham, "Scaling multidimensional inference for structured gaussian processes", *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 424–436, 2013.

[44] P. Glasserman, *Monte Carlo methods in financial engineering*. Springer Science & Business Media, 2013, vol. 53.

[45] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data", *arXiv preprint arXiv:1309.6835*, 2013.

[46] T. N. Hoang, B. K. H. Low, P. Jaillet, and M. Kankanhalli, "Nonmyopic -bayes-optimal active learning of gaussian processes", 2014.

[47] C. Robert, *Machine learning, a probabilistic perspective*, 2014.

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

[49] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[50] T. N. Hoang, Q. M. Hoang, and B. K. H. Low, "A unifying framework of anytime sparse gaussian process regression models with stochastic variational inference for big data.", in *ICML*, 2015, pp. 569–578.

[51] D. Sharma, A. Kapoor, and A. Deshpande, "On greedy maximization of entropy", in *International Conference on Machine Learning*, 2015, pp. 1330–1338.

[52] A. Wilson and H. Nickisch, "Kernel interpolation for scalable structured gaussian processes (kiss-gp)", in *International Conference on Machine Learning*, 2015, pp. 1775–1784.

[53] J. Han, X.-P. Zhang, and F. Wang, "Gaussian process regression stochastic volatility model for financial time series", *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 6, pp. 1015–1028, 2016.

[54] P. Izmailov, A. Novikov, and D. Kropotov, "Scalable gaussian processes with billions of inducing inputs via tensor train decomposition", *arXiv preprint arXiv:1710.07324*, 2017.

[55] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes", *arXiv preprint arXiv:1711.00165*, 2017.

[56] M. Binois, R. B. Gramacy, and M. Ludkovski, "Practical heteroscedastic gaussian process modeling for large simulation experiments", *Journal of Computational and Graphical Statistics*, vol. 27, no. 4, pp. 808–821, 2018.

[57] Z. Chen and B. Wang, "How priors of initial hyperparameters affect gaussian process regression models", *Neurocomputing*, vol. 275, pp. 1702–1710, 2018.

[58] J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, "Machine learning for quantitative finance: Fast derivative pricing, hedging and fitting", *Quantitative Finance*, vol. 18, no. 10, pp. 1635–1643, 2018.

[59] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration", in *Advances in Neural Information Processing Systems*, 2018, pp. 7576–7586.

[60] J. R. Gardner, G. Pleiss, R. Wu, K. Q. Weinberger, and A. G. Wilson, "Product kernel interpolation for scalable gaussian processes", *arXiv preprint arXiv:1802.08903*, 2018.

[61] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, "When gaussian process meets big data: A review of scalable gps", *arXiv preprint arXiv:1807.01065*, 2018.

[62] M. Ludkovski, "Kriging metamodels and experimental design for bermudan option pricing", 2018.

[63] G. Mu, T. Godina, A. Maffia, and Y. C. Sun, "Supervised machine learning with control variates for american option pricing", *Foundations of Computing and Decision Sciences*, vol. 43, no. 3, pp. 207–217, 2018.

[64] E. Commission, *Ethics guidelines for trustworthy ai*, E. Commission, Ed. [Online]. Available: ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[65] Y. Saatçi, "Scalable inference for structured gaussian process models", PhD thesis, Citeseer.