Department of Physics

BAYESIAN OPTIMIZATION OF VARIATIONAL QUANTUM EIGENSOLVERS

GIOVANNI IANNELLI

A dissertation submitted to the University of Cyprus in partial fulfillment of the requirements for the degree of Doctor of Philosophy

May 2023

VALIDATION PAGE

Doctoral candidate: Giovanni Iannelli

Dissertation Title: Bayesian optimization of Variational Quantum Eigensolvers

The present Doctoral Dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Department of Physics and was approved on the first of June 2023 by the members of the Examination Committee:

- Professor Constantia Alexandrou, University of Cyprus – *Research Supervisor*
- Professor Haralambos Panagopoulos, University of Cyprus – *Chairman*
- Assoc. Professor Nicolaos Toumbas, University of Cyprus
- Professor Agostino Patella, Humboldt University of Berlin
- Professor Luca Biferale, University of Rome "Tor Vergata"

DECLARATION OF DOCTORAL CANDIDATE

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned throught references, notes, or any other statements.

Giovanni Iannelli

ΠΕΡΙΛΗΨΗ

Ο μεταβλητός κβαντικός ιδιολύτης (ΜΚΙ) είναι ένας υβριδικός κβαντικός-κλασικός αλγόριθμος που χρησιμοποιείται για την εύρεση της θεμελιώδους κατάστασης μιας Χαμιλτονιανής χρησιμοποιώντας μεθόδους μεταβολής. Έχει ένα ευρύ φάσμα πιθανών εφαρμογών, από την κβαντική χημεία έως τις θεωρίες βαθμίδας στο πλέγμα στο φορμαλισμό της Χαμιλτονιανής. Ο αλγόριθμος ΜΚΙ χρησιμοποιεί κβαντικούς υπολογιστές για να υπολογίσει την ενέργεια του συστήματος με τις παραμέτρους του κυκλώματος και ελαχιστοποιεί αυτήν την παραμετροποιημένη ενέργεια με μια κλασική ρουτίνα βελτιστοποίησης. Αυτή η διατριβή περιγράφει έναν αλγόριθμο βελτιστοποίησης Bayes (ΒΒ) ειδικά σχεδιασμένο για να ελαχιστοποιεί την παραμετροποιημένη ενέργεια που λαμβάνεται με έναν κβαντικό υπολογιστή. Ο αλγόριθμος ΒΒ, που βασίζεται στην παλινδρόμηση διαδικασίας Gauss (ΠΔG), είναι ένας αλγόριθμος για την εύρεση του καθολικού ελάχιστου μιας γενικής συνάρτησης απόκλισης από την στοχευμένη τιμή, π.χ. την ενέργεια, με πολύ μικρό αριθμό επαναλήψεων ακόμη και όταν χρησιμοποιούνται δεδομένα που επηρεάζονται από στατιστικό θόρυβο.

Επιπλέον, η διαδικασία ΠΔG που αναπτύχθηκε στα πλαίσια αυτής της διατριβής, αποδείχθηκε πολύ ευέλικτη καθώς τη χρησιμοποιήσαμε επίσης για τον υπολογισμό διακριτών μετασχηματισμών ολοκλήρωσης δεδομένων με στατιστικά σφάλματα. Συγκεκριμένα, αυτή η διαδικασία χρησιμοποιήθηκε για την ανακατασκευή συναρτήσεων κατανομής parton από δεδομένα της ΚΧΔ πλέγματος.

ABSTRACT

The variational quantum eigensolver (VQE) is a hybrid quantum-classical algorithm used to find the ground state of a Hamiltonian using variational methods. It has a wide range of potential applications, from quantum chemistry to lattice gauge theories in the Hamiltonian formulation. VQE relies on quantum computers to evaluate the energy of the system in terms of circuit parameters, and it minimizes this parametrized energy with a classical optimization routine. This work describes a Bayesian optimization (BO) algorithm specifically designed to minimize the parametrized energy obtained with a quantum computer. BO based on Gaussian process regression (GPR) is an algorithm for finding the global minimum of a black-box cost function, e.g. the energy, with a very low number of iterations even when using data affected by statistical noise.

Furthermore, the GPR procedure developed for this work proved to be very versatile as we also used it to compute discrete integral transforms of noisy data. In particular, this procedure was used to reconstruct parton distribution functions from lattice QCD data.

CONTENTS

LIST OF FIGURES

INTRODUCTION

Quantum computing is an alternative type of computation based on the principles of quantum mechanics. Its fast-paced development in the recent years drew the attention of an interdisciplinary scientific community, since quantum algorithms have the potential to be exponentially quicker than classical alternatives in many noteworthy scientific applications [1]. Some examples are the simulation of quantum systems [2] and quantum machine learning [3].

Unfortunately, many of these algorithms are not yet implementable on current noisy intermediate-scale quantum (NISQ) computers [4] and need to wait until noise sources can be suppressed down to a threshold that enables quantum error correction [5], which can even lead us to the development of fault-tolerant quantum computers [6].

However, many interesting problems can already be studied on NISQ devices, especially the simulation of quantum systems, since their description in terms of quantum bits is more natural compared to that of classical systems. Indeed, some quantum algorithms are currently employed, for example, in quantum chemistry [7] and in lattice gauge theories [8].

In particular, the variational quantum eigensolver (VQE) [9] is a hybrid quantum-classical algorithm widely used on NISQ devices as it is particularly robust to quantum computing noise. It uses a quantum algorithm called quantum expected estimation (QEE) to estimate the expectation value of a given Hamiltonian in a given ansatz quantum state. This expectation value, i.e. the system energy, is then minimized across a family of ansatz states using a classical optimization algorithm, leading to an approximation of the ground state, which is of fundamental importance, for example, to study quantum systems at low temperatures.

Even though the VQE cannot unleash the full power of the pure quantum algorithms that are not yet implementable, it offers an alternative path to classical computation for the simulation of quantum systems. This is particularly valuable when used for systems whose phase space is not fully accessible through standard methods. For instance, the path integral formulation of quantum systems have traditionally been used to map them to equivalent statistical-mechanical systems, which can be analyzed using Markov chain Monte Carlo algorithms. However, in some cases such as, for example, fermionic systems, this mapping leads to negative or, in general, complex weighting factors that are not interpretable as probabilities. This *sign problem* [10] severely affects Monte Carlo simulations, preventing them, in some circumstances, from obtaining any reliable result.

On the other hand, the simulation of quantum systems on quantum computers doesn't require them to be mapped to classical systems as it relies, instead, on their Hamiltonian formulation, which means that their simulation doesn't suffer from the sign problem [11, 12].

The work of this thesis is focused on the VQE algorithm. In particular, on the classical optimization of the energy measurements performed with a quantum computer. This is a rather complicated optimization problem as the target function has many possible local

minima in a potentially high-dimensional space and, on top of this, the energy measurements are subject to statistical noise and quantum hardware noise.

Furthermore, each single energy measurement requires the run of the QEE quantum algorithm, which takes a significant amount of time using the quantum hardware available today. Therefore, it is desirable to perform the energy optimization with the fewest possible measurements in order to extract the maximum value from the allocated quantum computing time.

For this purpose, we decided to minimize energy measurements using Bayesian optimization (BO) [13], which is a framework particularly suitable to optimize expensive black-box functions. Given its flexibility, it can be adapted to perform efficiently in a wide range of problems. Indeed, we will explain in this thesis that BO has possible answers to each of the difficulties present in this optimization problem.

In particular, we implemented an algorithm [14] that allows us to incorporate both the energy measurements and their Gaussian errors into a probabilistic model of the target energy function, which can then be optimized using a procedure called noisy expected improvement [15].

BO is usually based on Gaussian process regression (GPR) [16], which is a machine learning algorithm that interpolates data using Bayesian inference. The very same GPR algorithm that we used for the VQE turned out to be useful to estimate integral transforms of functions using limited and noisy sets of data. In particular, the algorithm that we developed was used to estimate parton distribution functions using lattice QCD data [17], which are formally equivalent to Fourier transforms. This algorithm is further developed in this thesis and extended to compute generic integral transforms and their estimation error.

Structure of the thesis

Here we give a short description of the topics discussed in the following chapters:

CHAPTER 1: After a short introduction to quantum computing, we describe the quantum expected estimation (QEE) and the variational quantum eigensolver (VQE) algorithms. We then discuss the difficulties of the energy optimization required by the VQE, and we describe the optimizers commonly used to tackle this problem. Finally, we explain the motivations that led us to choose Bayesian optimization.

CHAPTER 2: Here we introduce the formalism of Gaussian processes, which will be later used to define a probabilistic model for the target energy function. In particular, we explain in this chapter how to impose to a Gaussian process certain properties and symmetries that can be useful to model the target energy.

CHAPTER 3: We give here an introduction to Bayesian inference, which is an alternative approach to statistics that found wide usage in machine learning techniques. In this chapter, we discuss about Bayesian prediction, Bayesian decision theory and Bayesian model selection, which all find usage in Gaussian process regression and Bayesian optimization.

CHAPTER 4: Here we introduce Gaussian process regression, focusing on how noisy measurements can be used to obtain Bayesian predictions of the target energy function. Finally, we discuss the potential numerical problems that might arise using this algorithm, and possible solutions that may be adopted to solve them.

CHAPTER 5: The predictions obtained with Gaussian process regression depend on the choice of a set of hyperparameters. In this chapter, we discuss the meaning of these hyperparameters and how to efficiently choose them (without overfitting the data) using maximum likelihood estimation or maximum a posteriori estimation.

CHAPTER 6: Here we describe how the Gaussian process regression procedure developed in the previous chapters can be used to estimate integral transforms using a limited amount of noisy data. We test this algorithm for the evaluation of Fourier transforms, comparing the results with those obtained using discrete-time Fourier transforms.

CHAPTER 7: We introduce here the concept of acquisition function maximization, which is the procedure adopted by Bayesian optimizers to choose the domain point of the next measurement. In particular, we describe the expected improvement acquisition function and its roots in Bayesian decision theory. Then, we explain why it doesn't perform well in presence of statistical noise, and how this problem can be solved with the noisy expected improvement, which is its generalization.

CHAPTER 8: After all the steps of Bayesian optimization have been explained, we specify here all the settings that we chose for the VQE optimization. Then, we test this algorithm using the Hamiltonian of the transverse-field Ising model, and we compare the results with other alternatives commonly used for this task.

CHAPTER 9: Finally, we draw here our conclusions and we discuss the possible future directions that this work might have.

1

VARIATIONAL QUANTUM EIGENSOLVER

1.1 INTRODUCTION TO QUANTUM COMPUTING

Quantum computing is an alternative to digital computation that leverages on the principles of quantum mechanics. Instead of digital bits, the most basic unit of information in a quantum computer is a qubit, which is a two-level quantum-mechanical system. Therefore, the state $|q\rangle$ of a qubit is not just either zero $|0\rangle$ or one $|1\rangle$, but rather a linear superposition of them:

$$|q\rangle = c_0 |0\rangle + c_1 |1\rangle$$

where $c_0, c_1 \in \mathbb{C}$ with $|c_0|^2 + |c_1|^2 = 1$ and the definition of a quantum state is equivalent up to a global complex phase $\exp(i\delta)$.

The observation of a qubit state follows the laws of quantum mechanics: a quantum computer can *measure* the state $|q\rangle$, delivering either zero or one with probability $|c_0|^2$ or $|c_1|^2$. After the measurement, the quantum state collapses into $|0\rangle$ or $|1\rangle$, therefore losing information about c_0 and c_1 as further measurements would result in the same value all the times.

Multi-qubit states

Having at its disposal N qubits, a quantum computer can construct multi-qubit states and maintain their coherence for a certain time.

Given N qubits in the states $|q_1\rangle, ..., |q_N\rangle$, a composite state $|q_1 \cdots q_N\rangle \equiv |q_1\rangle \otimes \cdots \otimes |q_N\rangle$ is an N-qubits state in which the i-th qubit has the value q_i, $\forall i$. In general, an N-qubit state $|q^{(N)}\rangle$ is a linear superposition of composite states. If it is possible to express $|q^{(N)}\rangle$ as a composite state $|q_1 \cdots q_N\rangle$, then $|q^{(N)}\rangle$ is called a *separable* state, otherwise it is an *entangled* state.

In a separable state, the qubits are *independent*, which means that measuring the value of a qubit doesn't influence the state of other qubits. The contrary happens for entangled states.

A simultaneous measurement of the N qubits results in one element of the computational basis $|0 \cdots 0\rangle, |0 \cdots 01\rangle, |0 \cdots 10\rangle, ..., |1 \cdots 1\rangle$. For notational simplicity, the elements of the computational basis are usually denoted using the base-2 representation of numbers, which means that the basis vector can be written as $|0\rangle, ..., |2^N - 1\rangle$. Using this notation, a generic N-qubit state has the following form:

$$|q^{(N)}\rangle = c_0 |0\rangle + ... + c_{2^N-1} |2^N - 1\rangle$$

where $c_0, ..., c_{2^N-1} \in \mathbb{C}$ with $|c_0|^2 + \ldots + |c_{2^N-1}|^2 = 1$.

A simultaneous measurement of all the qubits will therefore result in $|n\rangle$ with probability $|c_n|^2$, for $n = 0, ..., 2^N - 1$. On the other hand, the measurement of a subset of the qubits causes a collapse of the state into a lower dimensional space, since the observation removes the superposition of the measured qubit. At the same time, it is also removed the entanglement between the measured qubit and the others. For example, if the measurement of the first qubit yields zero, then the resulting N-qubit state will be expressible as $|q^{(N)}\rangle = |0\rangle \otimes |q^{(N-1)}\rangle$.

It is already possible from here to have an insight of the potential of quantum computing. An N-qubit state is a vector in a 2^N dimensional complex space, and, to represent such a vector, it is necessary to specify $2^{N+1} - 2$ real numbers, where the -2 factor is due to the norm constraint and the overall phase invariance. Considering, for example, a system of 100 qubits, which is approximately the amount present nowadays in the most advanced quantum computers, it would need around 10^{19} terabytes of memory just to store the quantum state in a classical memory using single precision floating point numbers.

Models of quantum computation

There are different possible models of quantum computation and each of them exploits specific hardware properties. For the work presented in this thesis, we only used the *unitary circuit model* of quantum computation, which is the first developed and the most widely known. Other alternative models are the *measurement-based* [18] and the *adiabatic evolution* [19] quantum computations.

The unitary circuit model has analogies with the classical digital computation: quantum algorithms start with a separable basis state, usually the zero state $|0 \cdots 0\rangle$, and use it as an input to a *quantum circuit*, which is a sequence of *quantum logic gates* ending with one or more qubit measurements. The quantum gates are unitary transformations that involve one or more qubits, and they are responsible for constructing the desired entangled states. Quantum computer hardware physically implement, for each qubit, a complete set of quantum gates, which can be used to construct any possible unitary transformation of the N-qubit state.

The measurement-based model has a different approach. The quantum algorithm starts from an entangled *source state*, and then, instead of quantum gates, the computation is performed by executing a sequence of single-qubit measurements. The achievable results are equivalent to those obtained with the unitary circuit model, indeed, it is available a complete set of instructions [20] for translating a quantum algorithm from one computational model to the other.

Even though the development of measurement-based quantum computers is, at the moment, way behind that of circuit-based devices, its implementation might become more scalable in the future, since the only operation needed on single qubits is their measurement, instead of a complete set of quantum gates.

Finally, the adiabatic evolution model is based on the adiabatic theorem of quantum mechanics, according to which, a physical system remains in a correspondent non-degenerate

eigenstate if the Hamiltonian is smoothly modified slowly enough. This means that, if we can construct the ground state of a certain Hamiltonian, we could evolve it to the unknown ground state of a target Hamiltonian.

The physical realization of adiabatic quantum computers is simpler than that of a complete quantum computer, since it doesn't need full control on each single qubit. Adiabatic evolution can, in principle, approximate any quantum algorithm without aggravating the computational complexity by more than polynomial terms [21].

Determining the exact time threshold for satisfying the adiabatic approximation can be very complicated, especially in presence of oscillatory terms in the Hamiltonian [22]. Therefore, the feasibility of a certain adiabatic evolution might need experimental testing. However, it is possible to give estimates of the required time in terms of the energy gap of the interpolating Hamiltonian.

Quantum gates

A generic transformation from a N-qubit state to another always preserves the norm, which means that all quantum gates must be unitary transformations. This is the only restriction and a universal quantum computer could, in principle, construct any unitary transformation.

Similarly to digital computers, a generic unitary transformation can be obtained using a very limited set of universal quantum gates. Considering first the transformations acting on single-qubit states, using $\{|0\rangle, |1\rangle\}$ as the basis, a generic unitary transformation is a matrix of the following form:

$$U_3(\theta, \phi, \lambda) \equiv \begin{pmatrix} \cos(\theta/2) & -e^{i\lambda}\sin(\theta/2) \\ e^{i\phi}\sin(\theta/2) & e^{i(\phi+\lambda)}\cos(\theta/2) \end{pmatrix}$$

where θ, ϕ, λ are any set of angles. This generic transformation is called U_3 gate, and it can be defined up to any global phase shift $\exp(i\delta)$ as it doesn't modify the quantum state.

The U_3 gate can be decomposed using the more elementary $SU(2)$ rotations, which are defined as:

$$R_\alpha(\theta) \equiv \exp\left(-i\frac{\theta}{2}\sigma_\alpha\right)$$

where θ is a real angle and the σ_α are the Pauli matrices:

$$\sigma_x \equiv \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \sigma_y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \quad \sigma_z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Therefore, the explicit form of the qubit rotations around the axes are:

$$\begin{aligned} R_x(\theta) &\equiv \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \\ R_y(\theta) &\equiv \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix} \\ R_z(\theta) &\equiv \begin{pmatrix} \exp(-i\theta/2) & 0 \\ 0 & \exp(i\theta/2) \end{pmatrix} \end{aligned} \tag{1.1}$$

Using these rotations, we can construct the generic U_3 gate:

$$U_3(\theta, \phi, \lambda) = \exp\left(i\frac{\phi+\lambda}{2}\right) R_z(\phi) R_y(\theta) R_z(\lambda)$$

which means that any single-qubit transformation can be generated using only R_y and R_z gates as the global phase $\exp\left(i\frac{\phi+\lambda}{2}\right)$ is irrelevant.

Regarding two-qubit transformations, a gate that is commonly used is the controlled NOT (CNOT), which, using the computational basis, assumes the following form:

$$C_x \equiv \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{1.2}$$

The reason of its name is explained considering the first qubit as a control that activates a NOT operation on the second qubit. Indeed, for example, considering the basis states, the C_x transforms $|10\rangle$ to $|11\rangle$ as the first qubit is one, but leaves unaltered $|00\rangle$ as the control qubit is zero.

The CNOT gate is able to create entanglement if applied on separable states. For example, if the control qubit is in superposition, we can obtain a Bell state:

$$C_x\left(\frac{|0\rangle + |1\rangle}{2} \otimes |0\rangle\right) = \frac{|00\rangle + |11\rangle}{2}$$

which is maximally entangled, since, before any measurement, each qubit has equal chances to be observed in zero or one, while, after one of them is measured, the other is equal to it with certainty.

It can be proven [23] that, using sequences of just $R_y(\theta)$, $R_z(\theta)$ and C_x gates acting on different qubits, it is possible to generate any possible N-qubit unitary transformation. A set of gates with this property is called a *universal* set.

Physical realizations of quantum computers might implement a different universal set of gates depending on the specific hardware. However, it is common to have a set of single-qubit gates and an entangling two-qubit gate. See, for example [24, 25].

Quantum computing noise

Several sources of noise can potentially hinder the execution of quantum algorithms. Nowadays, in fact, only a small subset of quantum algorithm can be successfully run on noisy intermediate-scale quantum (NISQ) computers [4].

The multi-qubit state is a very sensitive form of memory storage and its interaction with the surrounding environment is a possible source of quantum decoherence. This means that the system is no longer describable as a pure multi-qubit state $|\psi\rangle$, but rather as a statistical ensemble of possible states representable by a density matrix:

$$\rho = \sum_s p_s \, |\psi_s\rangle \, \langle\psi_s|$$

where p_s is the probability for the system to be in the state $|\psi_s\rangle$.

In order to keep the magnitude of environmental decoherence noise as small as possible, quantum computers are usually kept at ultra-cold temperatures lower than one Kelvin. Furthermore, any possible element of a quantum circuit is a potentially error-prone. Indeed, the operations of actively running quantum computers are constantly monitored, and the average magnitude of the main error components is available at any time. Typically, quantum computing providers present to the users the magnitude of current average CNOT and qubit-measurement errors, as well as the characteristic time of environmental decoherence. When these values surpass certain guaranteed thresholds, the quantum computers undergo a calibration process.

Clearly, the amount of noise and its complexity significantly increase with the number of qubits. Therefore, in order to make quantum computing scalable, we need to adopt error correction techniques that reduce the effective error to an arbitrary small rate. For classical digital computers, error correction could be simply achieved by keeping multiple copies of the same information. This specific solution, unfortunately, is not possible for quantum states because of the no-cloning theorem [26].

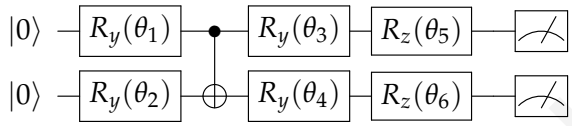
However, the no-cloning theorem could be bypassed by *spreading* the information of one *logical* qubit into an entangled state of several *physical* qubits [5]. This scheme could, in principle, suppress any source of noise up to a desired level and, eventually, lead us to obtain fault-tolerant quantum computation [6].

These error correction techniques require the physical noise to be below certain thresholds, which, unfortunately, are beyond the reach of current technology. Therefore, for now, we have to limit ourselves to use only quantum algorithm that are intrinsically noise-resilient. This doesn't mean that the algorithms implementable now will become obsolete when error correction will be available. Indeed, the transition is likely to be very smooth as noise suppression will be appliable only up to certain levels in order to prevent very high computation times. This means that the noise-resilient algorithms usable nowadays will have a performance advantage when quantum error correction will be available.

Parametrized quantum circuits

As we mentioned earlier, the unitary circuit model of quantum computation consists in applying a sequence of quantum gates to a starting separable state, and then performing qubit measurements at the end.

This scheme can be represented as a quantum circuit and, if we use N parametric gates such as $SU(2)$ rotations, the quantum circuit depend on a set of parameters $\theta_1, ..., \theta_N$. We will make usage of parametrized quantum circuits to construct families of multi-qubit states. For example, let us consider a circuit acting on two qubits:



The meaning of this circuit is the following. The two qubits, which both start from the $|0\rangle$ state, are first subject to a rotation around the y axis, each of them with a different angle: θ_1 and θ_2. Then, a CNOT gate is applied on the two qubits, with the first one used as the control. After this entangling gate, each of the qubits undergoes two rotations around the y and the z axes. Each of these four rotations has a different angle, which brings to six the total number of parameters. At the end, both of the qubits are measured, and their outcomes are stored in a classical memory.

It is possible to prove, for example, using the method described in [27], that this particular parametrized circuit, given an appropriate set of parameters $\theta_1, ..., \theta_6$, is capable to generate any possible two-qubit state, up to an irrelevant global phase.

In general, fully representing an N-qubit state as a parametrized quantum circuit doesn't suppress the exponential growth of the required number of parameters. Indeed, we saw that an N-qubit state requires $2^{N+1} - 2$ real parameters to be fully specified and we cannot expect a better scaling for the number of parameters required to represent it with a parametrized quantum circuit. It is possible, though, to approximate such a space using a lower number of circuit parameters, and the error of the approximation could be estimated, for example, using the procedure described in [28].

However, in many applications, we don't need to cover the full space of the N-qubit states. Indeed, if a certain problem exhibits a symmetry, we might only want to consider unique representative states and exclude all the other states that are symmetric to them. This is the same principle used to exclude states that are equivalent under multiplication by a global phase. Such symmetries could be incorporated into parametric quantum circuit, for example, using the procedure explained in [27].

1.2 QUANTUM EXPECTATION ESTIMATION

The quantum expectation estimation (QEE) [9] is a quantum algorithm for approximating expectation values $\langle\psi|H|\psi\rangle$ of an observable (Hermitian) operator H acting on a N-qubit state ψ.

Any Hermitian operator can be written as a polynomial of sigma matrices:

$$H = h_0 + \sum_{i\gamma} h^i_\gamma \sigma^i_\gamma + \sum_{ij\gamma\delta} h^{ij}_{\gamma\delta} \sigma^i_\gamma \sigma^j_\delta + \dots \tag{1.3}$$

where the h terms are real coefficients, the Latin indices $i, j = 1, ..., N$ indicate the qubit on which the sigma matrices are acting and the Greek indices $\gamma, \delta = x, y, z$ specify the coordinate of the sigma matrices.

The degree of the polynomial cannot, by definition, be higher than N, and terms of degree n correspond to each possible linear n-qubit interactions. In many cases, a potentially complicated Hamiltonian can be truncated to a lower degree polynomial if the system is dominated by local interactions.

Pauli matrices basis

The reason why the formulation (1.3) is always possible could be shown rewriting it in the following form:

$$H = \sum_{t=1}^{T} h_t P_t \tag{1.4}$$

Where the P_t are tensor product elements of $\{\mathbb{1}, \sigma_x, \sigma_y, \sigma_z\}^{\otimes N}$, therefore T cannot be greater than 4^N, but it is usually far lower. Indeed, if only interactions up to n bodies are considered, then T is $\mathcal{O}(N^n)$.

The tensor product elements P_t constitute a basis of the real vector space of the $2^N \times 2^N$ Hermitian matrices, which is the space of all possible observable operators H acting on an N-qubit space. This means that any observable can be written in the form of equations (1.4) and (1.3). Moreover, the P_t are orthonormal under the Hilbert–Schmidt inner product $\langle P_t, P_u \rangle_{\mathrm{HS}} \equiv \mathrm{Tr}(P_t P_u^\top) = N\delta_{tu}$, which means that, given H, the coefficients h_t in equation (1.4) can be obtained in the following way:

$$h_t = \frac{1}{N} \mathrm{Tr}(P_t H) \tag{1.5}$$

Most of the times, however, the real difficulty is to map a continuous physical system into an N-qubit system without using an excessive number of qubits and elements P_t in the Hamiltonian (1.4). Despite the difficulties of this task, systems as complicated as lattice gauge theories have been mapped into qubit spaces [29].

Measurement of Pauli operators

Assuming that we can construct an N-qubit state $|\psi\rangle = U|0\rangle$ by applying a sequence of quantum gates U to the zero state $|0\rangle$, we can use a quantum computer to estimate the expectation value $\langle\psi|H|\psi\rangle$ of an operator H. Indeed, if we rearrange H as in equation (1.4), we can decompose the expectation value of H in a sum of expectation values of sigma matrix tensor products:

$$\langle\psi|H|\psi\rangle = \sum_t h_t \langle\psi|P_t|\psi\rangle$$

The expectation values $\langle\psi|P_t|\psi\rangle$ can be estimated using the qubit states prepared by the quantum computer. To see how, let us first consider the case in which P_t is an operator P_t^z only formed by a tensor product of σ_z and identities. The vectors of the computational basis would be eigenstates simultaneously of P_t and the qubit measurement operators. Indeed, the qubits with value one or zero are respectively eigenvectors of σ_z with eigenvalue one or negative one. Therefore, a *measurement* of the operator P_t^z could be obtained in the following way:

- Prepare the state $|\psi\rangle = U|0\rangle$ by applying a sequence of quantum gates on $|0\rangle$.
- Measure the values of the qubits on which the σ_zs present in P_t^z are acting.
- Replace the zeroes of this sequence with negative ones and multiply all the numbers of the sequence.

Clearly, being a quantum measurement, the obtained result is a stochastic variable and the estimation of $\langle\psi|P_t^z|\psi\rangle$ would require the repeating the measurement several times and averaging over the results.

The measurement of a generic P_t could be transformed into a measurement of P_t^z using the rotation matrices (1.1). Indeed, the operators σ_z could be transformed respectively into σ_x or σ_y with a rotation of $\pi/2$ around $\widehat{y}$ or a rotation of $-\pi/2$ around $\widehat{x}$. This means that expectation values of σ_x and σ_y can be transformed in the following way:

$$\begin{cases} \langle\psi|\sigma_x|\psi\rangle = \langle\psi|R_y\left(\frac{\pi}{2}\right)\sigma_z R_y^{-1}\left(\frac{\pi}{2}\right)|\psi\rangle = \langle\psi'|\sigma_z|\psi'\rangle \\ \langle\psi|\sigma_y|\psi\rangle = \langle\psi|R_x\left(-\frac{\pi}{2}\right)\sigma_z R_x^{-1}\left(-\frac{\pi}{2}\right)|\psi\rangle = \langle\psi''|\sigma_z|\psi''\rangle \end{cases}$$

where $|\psi'\rangle \equiv R_y(-\pi/2)|\psi\rangle$ and $|\psi''\rangle \equiv R_x(\pi/2)|\psi\rangle$ can be constructed by adding one rotation gate to the quantum circuit.

This procedure can be used simultaneously on each qubit in which σ_x and σ_y are measured, since the rotations are single-qubit gates. Therefore, a stochastic measurement of the operator P_t can be performed constructing a single quantum circuit.

Measurement of the target operator

Coming back to equation (1.4), we can measure the target operator summing the measurements of each single P_t, which requires, in general, to execute T quantum circuits. The measurement of H obtained in this way should be repeated S times and averaged in order to estimate the expectation value $\langle\psi|H|\psi\rangle$. In jargon, these S repetitions are also called *shots*.

To summarize, the expectation value of the operator H in (1.3) evaluated in the state $|\psi\rangle = U|0\rangle$ can be estimated in the following way:

- Split H into T tensor products P_t of sigma matrices and consider each of them singularly.
- For each P_t, prepare the quantum state $|\psi\rangle = U|0\rangle$ and add rotations $R_y(-\pi/2)$ and $R_x(\pi/2)$ for each σ_x or σ_y present in P_t in order to transform them in σ_z.

- Measure the qubit values of the T states, replace the zeroes with negative ones and multiply them to obtain observations of each P_t, and finally sum the P_t to obtain a measurement of H.
- Repeat the procedure S times to obtain $H_1, ..., H_S$ measurements of H. An estimation of $\langle\psi|H|\psi\rangle$ and its error are given by:

$$\overline{H} \equiv \frac{1}{S}\sum_i H_i \qquad \Delta H \equiv \sqrt{\frac{\sum_i (H_i - \overline{H})^2}{S(S-1)}} \tag{1.6}$$

Quantum noise vs statistical noise

Since the average $\overline{H}$ is constructed using i.i.d. random variables with finite variance, the central limit theorem guarantees that the distribution of $\overline{H}$ converges to a Gaussian distribution in the limit of $S \to \infty$, and the standard deviation of this Gaussian is estimated by ΔH.

We will therefore refer to ΔH as the *statistical noise* of the $\overline{H}$ measurement. This should not be confused with what we will generically call *quantum noise*, which is the bias of $\overline{H}$ induced by imperfections of the quantum computer.

Indeed, phenomena such as quantum decoherence, measurement and gate errors afflict the estimation in equation (1.6). In particular, if these errors don't average out, they induce a bias into the estimator $\overline{H}$. When quantum error correction will be available, this bias could be reduced up to a desired level, but this option is not yet available with NISQ devices.

However, we could already significantly reduce this bias using instead error mitigation techniques. A review of commonly used methods can be found in section 5.1 of [30].

Operators grouping

The procedure presented here requires the preparation of T independent quantum states for each single shot, where T is the number of Pauli terms P_t in the target operator. In some cases, for performance reasons, we might want to reduce the number of state preparations T.

This reduction could be achieved, for example, grouping together Pauli operators that commute with each others [31], since this allows us to symultaneously measure more operators using their common set of eigenstates.

However, different measurements on the same quantum states have a non-zero covariance, which in turn modifies the variance of each single shot H_i. While this fact shouldn't afflict the Gaussianity of $\overline{H}$, it might, though, have the undesired effect of increasing the statistical noise ΔH. The final effect might therefore be the opposite of what we wanted to achieve, since a higher statistical noise requires an increased number of shots for reaching the same precision, which means a higher number of state preparations.

On the other hand, this potential problem might be turned in our advantage if we group together measurements with a negative mutual correlation, since this actually reduces the final statistical noise.

The difficulty, however, is to define a procedure for selecting the optimal grouping of Pauli operators. A possible strategy is explained in [32].

Comparison with QPE based algorithms

The quantum phase estimation (QPE) [33] is a widely known quantum algorithm for the estimation of the eigenvalue corresponding to an eigenvector of a unitary operator. It requires $\mathcal{O}(1/\varepsilon)$ quantum gates to achieve a precision of ε, and is a building block of many noteworthy algorithms such as Shor's factorization algorithm [34].

The QPE can also be used [35] to find eigenvalues and eigenvectors of an operator H, and, in particular, also to evaluate expectation values of H. It requires, in general, $\mathcal{O}(1/\varepsilon^2)$ quantum gates to reach a precision of ε [31].

While requiring, in principle, only one state preparation and usually a lower number of quantum gates overall, QPE based algorithms are not suitable for NISQ devices as they are quite susceptible to quantum noise. Indeed, since they use a single long chain of quantum gates, all the gate errors and decoherence noise accumulate, destroying the final result.

The QEE, on the other hand, split the computation in a high number of short computations that are achievable on noisy devices. Furthermore, QEE leaves freedom in the way the state $|\psi\rangle$ is prepared. Therefore, we have the possibility to choose the set of gates that are more convenient for the specific hardware at our disposal. This feature further reduces the number of gates required for each short computation, and hence the susceptibility on quantum noise.

Moreover, the QEE remains well-defined even in presence of a small quantum noise. Indeed, if the error sources transform the pure state $|\psi\rangle$ into a mixed state ρ, the QEE correctly estimates the expectation value of H in ρ, which is defined as:

$$\langle H \rangle_\rho \equiv \mathrm{Tr}(\rho H) = \sum_s p_s \langle \psi_s | H | \psi_s \rangle$$

where p_s is the probability for the system to be in the pure state $|\psi_s\rangle$. Since the QEE performs the estimation by averaging single shot measurements in each possible $|\psi_s\rangle$, it correctly estimates the expectation value of H in ρ. In other words, the reason why the QEE is particularly robust to quantum noise is that it emulates a hypothetical physical measurement of the observable H. The only effect on the algorithm is the introduction of a residual bias that is not averaged out in the process caused by the difference between ρ and $|\psi\rangle$.

1.3 VARIATIONAL QUANTUM EIGENSOLVER

The variational quantum eigensolver (VQE) [9] is a hybrid quantum-classical algorithm to compute the lowest eigenvalue (and its eigenvector) of a Hermitian operator H using the variational principle. The computation of the lowest eigenvalue is of crucial interest both in optimization and in physics. Indeed, in optimization problems, the lowest eigenvalue encodes the optimal solution, while, in statistical physics, the ground state of a Hamiltonian

plays a dominant role at modest temperatures. Anyway, using techniques such as variational quantum deflation [36, 37], the VQE can be extended to find the k lowest eigenvalues of H and their eigenspace.

In our work, we focus our attention on the computation of the ground state of Hamiltonians. We will therefore, from now on, refer to H as the target Hamiltonian.

Ansatz states

We saw at the end of section 1.1 that we can construct a family of multi-qubit states using a parametrized quantum circuit. Indeed, calling $U(\theta_\alpha)$ a sequence of quantum gates depending on D parameters θ_α, we can construct the following states:

$$|\psi(\theta_\alpha)\rangle \equiv U(\theta_\alpha)\,|0\rangle \tag{1.7}$$

Among all these states, which are usually called *ansatz states*, we want to find the ground state that minimizes the system energy, which can be measured using the QEE algorithm.

Indeed, after choosing a set of D parameters θ_α, we can prepare the ansatz state $|\psi(\theta_\alpha)\rangle \equiv U(\theta_\alpha)\,|0\rangle$ with a quantum computer and use the QEE to estimate the parametrized energy, which is the expectation value of H in $|\psi(\theta_\alpha)\rangle$:

$$E(\theta_\alpha) \equiv \langle\psi(\theta_\alpha)|H|\psi(\theta_\alpha)\rangle \tag{1.8}$$

In practice, we saw in equation (1.6) that, choosing a number of shots S, the QEE performs a measurement of $E(\theta)$ with a statistical error $\Delta E(\theta)$ that decreases as $\mathcal{O}(1/\sqrt{S})$.

The choice of the ansatz states is of fundamental importance for the success of the algorithm. Indeed, their space should be large enough to include the ground state, or at least a good approximation of it, while, on the other hand, it should be using the lowest possible number of parameters in order to facilitate the search of the ground state.

The choice of the required parameters θ_α should therefore exclude the redundant ones, without which the space of reachable states is not reduced. It is also advisable to further restrict the ansatz space by excluding the states that are equivalent under the symmetries of the Hamiltonian. A general procedure for constructing such ansatz spaces and quantifying the approximation error is explained in [27] and in [28].

Energy optimization

Considering the family of ansatz states (1.7) and the parametrized energy (1.8) measured on them, the ground state $|\psi^0\rangle$ and the ground state energy E^0 of the Hamiltonian H can be approximated using the variational principle:

$$\min_{\theta_\alpha} E(\theta_\alpha) \equiv E(\theta_\alpha^0) \geq E^0 \tag{1.9}$$

If the ground state $|\psi^0\rangle$ is obtainable with the parametrized quantum circuit $|\psi(\theta)\rangle$, then $E(\theta_\alpha^0)$ is exactly the ground state energy E^0 and $|\psi(\theta_\alpha^0)\rangle$ is the ground state $|\psi^0\rangle$. Otherwise,

$E(\theta_\alpha^0)$ is an upper bound estimation of the ground state and $|\psi(\theta_\alpha^0)\rangle$ is an approximation of the ground state.

The VQE is considered a hybrid quantum-classical algorithm because it requires a quantum computer to measure the energy using the QEE, but also a classical computer to perform the optimization (1.9).

Robustness to quantum noise

The main feature of VQE is its robustness to quantum noise as it makes it runnable on NISQ devices. Most of it comes from the usage of QEE, which, as we explained earlier, reduces the effect of noise using only short quantum computations and averages out the uncorrelated sources of noise.

Moreover, the VQE minimization remains well-defined even when the quantum noise mildly affects the pure parametrized state $|\psi(\theta_\alpha)\rangle$ transforming it to a mixed state represented by a density matrix $\rho(\theta_\alpha)$. Indeed, the parametrized energy $E(\theta_\alpha) = \mathrm{Tr}(\rho(\theta_\alpha))$ is still approximated by the QEE and the minimization (1.9) becomes:

$$\min_{\theta_\alpha} E(\theta_\alpha) = \min_{\theta_\alpha} \mathrm{Tr}(\rho(\theta_\alpha)) \equiv \mathrm{Tr}(\rho(\theta_\alpha^0)) \geq E^0$$

which means that θ_α^0 is still the optimal choice of circuit parameters to approximate the ground state.

Properties of the parametrized energy

Before discussing possible efficient algorithms to perform the optimization (1.9), it is advisable to study the target function, which is the parametrized energy $E(\theta_\alpha)$.

In order to have a general idea of the target function, Let us focus, for now, on the case of perfect measurements, without either statistical noise or quantum noise.

The parametrized energy depends, of course, on the specific gates chosen to prepare the ansatz states. However, let us restrict the choice, for now, to rotation gates (1.1) and non-parametric gates, such as, for example the CNOT (1.2). Even though this set of universal gates can generate any possible unitary transformation on the multi-qubit space, it is not guaranteed that this is done efficiently.

Indeed, if we want to reduce the number of circuit parameters θ_α as much as possible, it might be necessary to use different types of quantum circuit, especially if we want to impose the symmetries of the Hamiltonian in the ansatz states [27].

However, analyzing this subset of quantum circuits can be very valuable as the analysis leads to an analytical formula that can be used for the design and the benchmark of optimization algorithms. It is possible, in fact, to prove [38] by induction that, if the only parametrized gates are D rotations (1.1) depending on $\theta_1, ..., \theta_D$, then the parametrized energy $E(\theta_\alpha)$ assumes the following form:

$$E(\theta_\alpha) = \sum_{m=1}^{3^D} c_m \left(\bigotimes_{\alpha=1}^{D} \begin{pmatrix} 1 \\ \sin(\theta_\alpha) \\ \cos(\theta_\alpha) \end{pmatrix} \right)_m \tag{1.10}$$

where c_m are real coefficients and each term is a possible product of ones, sines and cosines depending on different θ_α.

On its own, the formula (1.10) is not very helpful as the number of its terms grows exponentially with the number of dimensions D. However, an interesting feature of equation (1.10) is that it holds for in any D'-dimensional subspace.

For example, keeping fixed all the parameters apart from the one with index α', equation (1.10) reduces to:

$$E(\theta_{\alpha'}|\theta_{\alpha\neq\alpha'}) = c_1 + c_2 \sin(\theta_{\alpha'}) + c_3 \cos(\theta_{\alpha'}) \tag{1.11}$$

which is a shifted sinusoidal with a generic phase.

Even though the formula (1.10) is not valid for a generic parametrized quantum circuit, some of its properties remain even if we use a different set of gates. Indeed, most of the commonly used parametrized gates are expressed in terms of one or more angle parameters. Therefore, in most cases, we expect $E(\theta_\alpha)$ to be 2π-periodic in each angle parameter θ_α. Moreover, parametrized gates are usually expressed in terms of smooth C^∞ functions, which means that $E(\theta_\alpha)$ is in C^∞ as well. All these properties could be very useful for the optimization of $E(\theta_\alpha)$.

Difficulties of the optimization

Even in the cases in which the analytical formula (1.10) is valid, the optimization (1.9) of the parametrized energy is a very difficult task. Indeed, several difficulties hinder a successful convergence to the ground state.

Firstly, $E(\theta_\alpha)$ has potentially many local minima, which means that the optimization cannot just be a local gradient descent algorithm as it would remain stuck in suboptimal solutions. Therefore, we need a *global optimizer* that is capable to perform a global search across the domain, or, at least, it should be able to escape from local minima. This difficulty is aggravated by the fact that the ansatz states might have a high number of parameters. Indeed, as long as new quantum computers with better performances are developed, we might want to study problems that require a higher number of qubits and of parametrized gates acting on them.

Secondly, the measurements of $E(\theta_\alpha)$ obtained with (1.6) have a statistical error. This means that we need a so-called *noisy optimizer* that remains well-defined when the target function evaluations are imprecise. Ideally, the optimizer should also use the measurement error in (1.6) to exploit all of the information obtained with the QEE.

Alongside the statistical noise, there is also the quantum noise that affects the measurements obtained with the QEE. As we have already discussed, the VQE is, on its own, already quite robust to quantum noise and techniques such as error mitigation are very helpful for reducing the estimation bias. It might be desirable to include noise models inside the optimization procedure in order to further reduce the effects of quantum noise. However, it is reasonable, for now, to use an optimizer that is robust to small measurement biases, and to rely on error mitigation techniques to make sure that this bias is as small as possible.

On top of these complications, there is, perhaps, the most difficult to overcome, since it makes all the others more problematic. The usage of quantum computers is, at the moment,

very expensive both in time and budget. Therefore, we would like to reach the ground state using as few energy measurements as possible. This means that we cannot find the global minimum with *brute force* approaches, and we cannot suppress the statistical error, which decreases as $\mathcal{O}(1/\sqrt{S})$, simply by using an extremely high number of shots S. Moreover, the resources required for energy measurements are even higher if we need to perform the calibrations required by error mitigation techniques.

The main task of this thesis work is to analyze these problems in order to design and test an optimization algorithm that fares well in all these difficulties.

Optimizers commonly used for VQE

Several optimization algorithms have been proposed to perform the classical optimization (1.9). In the paper that introduced the VQE [9], the authors use the Nelder-Mead optimization method [39], which is a local optimizer that creates a simple heuristic model of the slope direction by evaluating the D-dimensional target function at the vertices of a $D+1$ polytope, which is also called a *simplex*. At each iteration, one vertex of the simplex is moved along the estimated slope direction such that the volume of the simplex is expanded when moving down the slope or shrunk when the target function gets flatter. The algorithm terminates when the target function is approximately constant at each simplex vertex.

Compared to basic gradient-based methods, the Nelder-Mead algorithm is generally less likely to get stuck in local minima and is quite robust to statistical noise. However, in many circumstances, this comes at the price of a slower convergence rate.

While being a viable solution for extremely simple toy-models, the Nelder-Mead algorithm usually doesn't fare well with more realistic high-dimensional VQEs. In particular, its main problem is not the rather slow convergence rate, but its tendency to remain stuck in local minima.

An algorithm that is generally more suitable for optimizing $E(\theta_\alpha)$ is the simultaneous perturbation stochastic approximation (SPSA) [40], which is a gradient-descent method that uses a stochastic estimation of the gradient. At each iteration, the current estimated minimum point is perturbed along a random direction that is not aligned to an axis, and the target function is evaluated at these two points. Given these two (noisy) energy measurements and the perturbation vector, we can construct an unbiased finite-difference estimator of the gradient, which is used to descend to the point that is used in the following iteration of the algorithm. The main feature of SPSA is that this gradient estimation requires only two measurements, regardless of the number of dimensions.

Even though the SPSA is a gradient-based method, it is capable of jumping out of local minima thanks to the stochastic nature of its gradient estimation. Furthermore, the usage of noisy measurements is actually helpful for not getting stuck in local minima.

In order to converge to the global minimum, it is important to carefully choose the hyperparameters of the SPSA. Indeed, both the magnitude of the descent along the gradient and the magnitude of the random perturbations need to be sequences of positive numbers asymptotically tending to zero as the iteration number grows. In particular, the authors of

SPSA give a precise list of sufficient conditions on the algorithm parameters such that the algorithm converges in probability to the global minimum.

The SPSA is applicable with any possible parametrized quantum circuit and it is very accessible as it is part of IBM's Qiskit library [41] with the specifications explained in [25]. However, the downside of SPSA is that it doesn't usually have a great convergence rate, which can be problematic if we have only a limited number of measurements at our disposal.

Finally, the Nakanishi-Fujii-Todo (NFT) [38] algorithm is a very effective optimizer built on top of the analytic formula (1.10), which means that it is only applicable to ansatz states constructed using exclusively rotation gates and non-parametric gates. The NFT avoids the exponential complexity of the formula (1.10) by sequentially optimizing one parameter at a time. Indeed, in a one-dimensional subspace, the analytic formula (1.10) reduces to the sinusoidal (1.11), which is fully determined with just three measurements, and one of them can be taken from the expected minimum of the previous iteration. Therefore, with noiseless measurements, the NFT would require only two energy measurements for optimizing a one-dimensional subspace. However, in presence of statistical noise, after some iterations, it is necessary to perform each of the three measurements to avoid the accumulation of statistical errors that might ruin the optimization accuracy.

Even though the NFT updates only one parameter at a time, empirical tests show a tendency to quickly escape from local minima, since, unlike most local optimizers, even when the gradient is zero, the parameters are allowed to jump at distant locations.

Since it uses an exact formula, the NFT has very fast convergence rates, especially in low-dimensional spaces. However, since it updates only one parameter at a time, its convergence rate in very high-dimensional setups. The algorithm could be easily extended to minimize d-dimensional subspaces, but the number of required measurements increases exponentially with d.

Even though the NFT assumes zero statistical and quantum noise, it performs surprisingly well in their presence. Indeed, since only one parameter is modified at each iteration, the algorithm suppresses the chances of big steps in wrong directions induced by measurement imprecision. Furthermore, independent sources of noise tend to average out across many iterations. This leads to a steady convergence in the long run.

The NFT is way simpler to implement than its main alternatives, and its authors provide the source code at [42]. Its main limitation is the inapplicability when using parametrized gates that are not simple qubit rotations. This is particularly relevant for ansatz states designed to incorporate Hamiltonian symmetries, such as those described in [27].

Why Bayesian optimization?

The optimization procedure that we studied in this thesis is known as Bayesian (global) optimization (BO). Its earlier applications date back in the 60s [43], while its modern implementations are based on a more recent work [13].

The main feature of BO is that it requires a very low number of target function evaluations in order to converge to the global minimum. On the other hand, to achieve this result, BO requires a complicated probabilistic modelling of the target function, which comes at a high

(classical) computational cost. Therefore, BO is mostly used for those tasks in which the evaluation of the target function is the bottleneck of the computation. Indeed, in these cases, it is convenient to put more effort in minimizing the number of required iterations rather than using a simpler algorithm that relies on measuring the target function several times.

For example, BO is widely used for finding the optimal set of neural network hyperparameters [44]. Assessing the effectiveness of a set of hyperparameters requires training the neural network from scratch every time. Therefore, we are willing to pay the cost of a computationally intensive algorithm if this means the avoidance of the training of neural networks more times than strictly necessary.

In our case, the bottleneck are the energy measurements with the QEE, mostly because of the limited availability of quantum hardware compared to classical CPU time. Indeed, running quantum algorithms, at the moment, usually requires spending time in waiting list, while, on the other hand, we can easily reduce the time for classical computation through parallelization.

This situation will slowly change in the future. However, even in this case, BO might still have an important role as it is a very customizable algorithm that can be significantly simplified if shorter CPU times are required.

There are indeed many degrees of freedom in implementing BO and many different variants of it have been published in order to tackle different issues. We have indeed at our disposal possible answers to each of the optimization difficulties explained earlier.

We will see through the next chapters that BO has an elegant end effective way to include Gaussian statistical noise in its analysis. In particular, we [14] implemented a strategy that uses noisy measurements and their errors to predict the values of the parametrized energy $E(\theta_\alpha)$, which is then optimized using the recently published noisy expected improvement acquisition function [15]. The results showed a fast convergence rate in low-dimensional spaces, providing a precise solution in very noisy setups. In this thesis work, we give a detailed explanation of each step of this algorithm, since this information can be very useful to adapt this procedure to a wide range of different problems.

BO is generally used to globally optimize black-box functions, about which we only have information regarding its smoothness level and its symmetries. The parametrized energy, given its complexity, can be effectively treated as a non-convex black-box function. Even in the cases in which its analytical formula (1.10) is available, its exponential number of terms makes it unusable in practical applications, unless we restrict the formula to small subspaces by keeping fixed the other circuit parameters. Furthermore, possible biases caused by quantum noise might cause a discrepancy between the measured energy and its exact formula.

Nevertheless, we recognize that, in most practical applications, $E(\theta_\alpha)$ is a C^∞ function with 2π-periodicity in each parameter. We will see in the next chapters that both of these properties can easily be incorporated into BO.

A big challenge of BO, however, is its usage in high-dimensional spaces. Indeed, using the most common Bayesian optimizers, the CPU and memory requirements might become unbearable if we want to simultaneously optimize $D \gtrsim 40$ parameters. For this reason, many efforts have been put in the recent years in order to solve this problem and possible

strategies such as [45–47] allows us to implement BO in high dimensional spaces while still keeping a performance advantage compared to common alternative solutions.

In particular, it was published very recently [48] an algorithm that combines the scalability of stochastic gradient-based methods with the global search performance of BO in order to efficiently optimize VQE's parametrized energy in high-dimensional setups. This is done by performing an unbiased estimation of the gradient using $2D$ single shot measurements, and using BO in the one-dimensional subspace spanned by the gradient.

This algorithm showed a very good convergence rate when used for the VQE in high-dimensional spaces, and was able to reach the ground state with a higher precision compared to the alternatives. These promising results indicate that BO is definitely a good choice for variational quantum simulations.

2

INTRODUCTION TO GAUSSIAN PROCESSES

Considering a set of N points in a D dimensional space, we will denote with a Greek lowercase letter the coordinate index and with a Latin lowercase letter the index of the element in the set. If the indices are not used in any operation as, for example, in $\sum_i$ or $\prod_\alpha$, then they will indicate sets, vectors or matrices. For example:

$$\theta_{i\alpha} \equiv \begin{cases} (\theta_{11},\dots,\theta_{1D}) \\ \vdots \\ (\theta_{N1},\dots,\theta_{ND}) \end{cases} \qquad K_{ij} \equiv \begin{pmatrix} K_{11} & \cdots & K_{1N} \\ \vdots & \ddots & \vdots \\ K_{N1} & \cdots & K_{NN} \end{pmatrix}$$

Here $\theta_{i\alpha}$ represents a set of N D-dimensional vectors and K_{NN} a $N \times N$ matrix.

A stochastic process f defined on a D-dimensional domain is a map that, for any integer number N, associates an ordered set of N domain points $\theta_{i\alpha}$ to an equally sized ordered set of stochastic variables f_i distributed according to a joint probability distribution $p(f_i)$:

$$\theta_{i\alpha} \xrightarrow{f} f_i \xrightarrow[\text{as}]{\text{distributed}} p(f_i)$$

2.1 GAUSSIAN PROCESS

A Gaussian process (GP) is a stochastic process f in which the image stochastic variables f_i are distributed according to a multivariate Gaussian distribution:

$$p(f_i) = \det(2\pi K)^{-\frac{1}{2}} e^{-\frac{1}{2}\sum_{ij}(f_i-\mu_i)(K^{-1})_{ij}(f_j-\mu_j)} \tag{2.1}$$

where μ_i is the mean vector and K_{ij} is the covariance matrix. Their components are respectively constructed with a mean function $\mu(\theta_\alpha)$ and a covariance function $k(\theta_\alpha, \theta'_\alpha)$ in the following way:

$$\mu_i \equiv \mu(\theta_{i\alpha}) \qquad K_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha}) \tag{2.2}$$

Comparing the definition (2.2) with the multivariate Gaussian (2.1), it is clear the relationship between $\mu(\theta_\alpha)$, $k(\theta_\alpha, \theta'_\alpha)$ and the expected value and the covariance of $f(\theta_\alpha)$ according to (2.1):

$$\mu(\theta_{i\alpha}) = \mathbb{E}[f(\theta_\alpha)] \qquad k(\theta_{i\alpha}, \theta_{j\alpha}) = \mathrm{Cov}[f(\theta_\alpha), f(\theta'_\alpha)] \tag{2.3}$$

Once $\mu(\theta_\alpha)$ and $k(\theta_\alpha, \theta'_\alpha)$ are specified, the GP in unequivocally determined. To be a valid covariance function, $k(\theta_\alpha, \theta'_\alpha)$ needs to be defined such that K is positive definite $\forall \theta_\alpha, \theta'_\alpha$. Functions with this property are called positive-definite kernels.

A stochastic process can be seen as an extension to functions of the concept of random variables. Indeed, a stochastic process can have many possible sample (or outcome) functions in the same way as a random variable has many possible sample values.

In Fig 2.1 is shown a one-dimensional example of GP with three colored sample functions drawn from it. The black dashed line is the mean function $\mu(\theta)$ which is the expectation values of the samples at each value of θ. The grey interval around $\mu(\theta)$ represents the standard deviations $\sqrt{k(\theta,\theta)}$ of the samples at each value of θ. Indeed, at $\theta = \frac{\pi}{2}, \frac{3\pi}{2}$, the standard deviation is zero and all samples touch $\mu(\theta)$ in these points. The bigger is the standard deviation, the wider are the average deviations of the sample from $\mu(\theta)$.

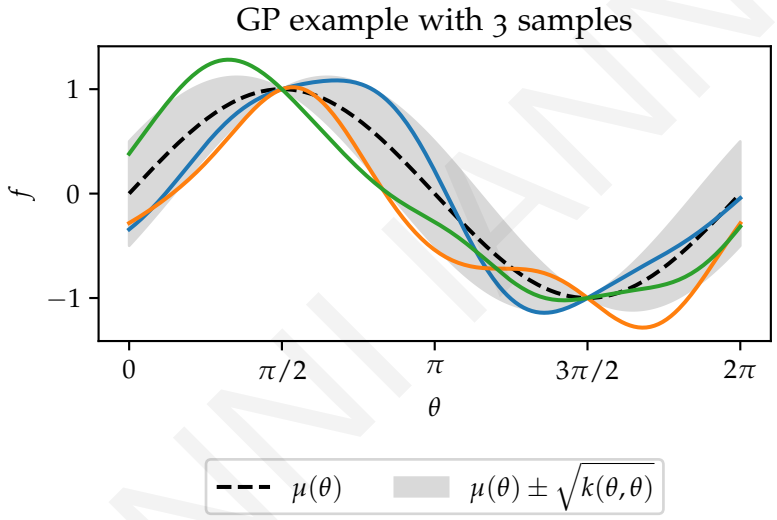


Figure 2.1: Example of GP using $\mu(\theta) = \sin(\theta)$ and $k(\theta,\theta') = \frac{\cos(\theta)\cos(\theta')}{4} \cdot \exp\left(-\frac{(\theta-\theta')^2}{2}\right)$. The colored lines represent 3 random samples of the GP.

In the next section is shown an algorithm to draw discretized sample functions from a given GP.

2.2 SAMPLING FROM A GAUSSIAN PROCESS

Given a Gaussian process with mean function $\mu(\theta_\alpha)$ and covariance function $k(\theta_\alpha, \theta'_\alpha)$, the N values f_i of a sample function evaluated respectively in the D-dimensional points $\theta_{i\alpha}$ can be found sampling their joint Gaussian probability density function (2.1), which is a well-known procedure (see the discussion in section A.2 of the appendix for more details). Here is summarized a method for obtaining such a sample:

1. Generate N independent random real numbers x_i from a Gaussian distribution with zero mean and one standard deviation.

2. Use the mean function $\mu(\theta_\alpha)$ and the covariance function $k(\theta_\alpha, \theta'_\alpha)$ to construct the mean vector $\mu_i = \mu(\theta_{i\alpha})$ and the covariance matrix $K_{ij} = k(\theta_{i\alpha}, \theta_{j\alpha})$.

3. Find a matrix A such that $K = AA^\top$.

4. The wanted sample set is given by $f_i = \sum_j A_{ij} x_j + \mu_i$.

The matrix A in point 3 could be found with Cholesky decomposition, which, given a Hermitian positive definite matrix K, finds a lower triangular matrix L such that $K = LL^\top$.

If f_i have strong correlations, Cholesky decomposition might fail because the matrix K might have zero or negative eigenvalues due to rounding errors [49]. In order to prevent it, it might be helpful to add a small positive number to the diagonal elements of K [50]. If this is not enough, a more stable approach is to use the eigendecomposition $K = Q\Lambda Q^\top$ to find $A \equiv Q\Lambda^{1/2}$, which also implies $K = AA^\top$.

When a sample is evaluated in a large number N of points, this procedure might be slow as the computational cost of Cholesky decomposition and eigendecomposition is $\mathcal{O}(N^3)$. In these cases, it might be helpful to sample the multivariate Gaussian using iterative methods [51] that avoid operations with $\mathcal{O}(N^3)$ complexity.

Fortunately, in many circumstances it is not necessary to sample a high number of variables f_i as many properties and results can be obtained directly from $\mu(\theta_\alpha)$ and $k(\theta_\alpha, \theta'_\alpha)$, which are usually specified in closed form.

2.3 GEOMETRY OF GAUSSIAN PROCESSES

The mean and the covariance functions determine the properties of sample functions drawn from a Gaussian process. The mean function is the function around which the sampled functions oscillate, and the covariance function determines the magnitude and the correlation of typical deviations from the mean function. In particular, the covariance function imposes different geometries on sample functions. Therefore, the choice of $k(\theta_\alpha, \theta'_\alpha)$ is of great importance for modelling a phenomenon with a GP.

We mentioned in section 1.3 that the parametrized energy $E(\theta_\alpha)$ is usually C^∞ and 2π-periodic in each parameter. Since we want to model the parametrized energy with a GP, we want to impose these properties to the GP samples as they constitute the probability space of our guesses about the real form of $E(\theta_\alpha)$.

In the next few paragraphs, we will review some sample properties that can be fixed with an appropriate choice of the covariance function, keeping in mind what we know about the target function.

Smoothness of samples

It is proven [52, 53] that the smoothness level of the samples is determined by the smoothness level of the mean and covariance functions. If $\mu(\theta_\alpha) \in C^k$ and $k(\theta_\alpha, \theta'_\alpha) \in C^{2k}$, then the functions sampled from the Gaussian process are $\in C^k$.

Below are listed covariance functions commonly chosen [16] to impose certain levels of smoothness. The meaning of the positive parameters σ and ℓ is explained later in the next paragraphs.

- The radial basis function (RBF) (or squared exponential) kernel:

$$k^{\mathrm{RBF}}(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 e^{-\sum_\alpha \frac{(\theta_\alpha - \theta'_\alpha)^2}{2\ell^2}} \tag{2.4}$$

  $k^{\mathrm{RBF}}(\theta_\alpha, \theta'_\alpha) \in C^\infty$, therefore the GP samples are in C^∞.

- The exponential kernel:

$$k^{\exp}(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 e^{-\sum_\alpha \frac{|\theta_\alpha - \theta'_\alpha|}{\ell}} \tag{2.5}$$

  $k^{\exp}(\theta_\alpha, \theta'_\alpha) \in C^0$, therefore the GP samples are in C^0.

- The Matérn class [54] of kernels, parametrized by $\nu > 0$:

$$k^{\mathrm{M}}_\nu(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{d(\theta_\alpha, \theta'_\alpha)}{\ell} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{d(\theta_\alpha, \theta'_\alpha)}{\ell} \right) \tag{2.6}$$

  Where K_ν are the modified Bessel functions of second kind [55], Γ is the gamma function and d is the Euclidean distance. $k^{\mathrm{M}}_\nu(\theta_\alpha, \theta'_\alpha) \in C^{2\lceil \nu - 1 \rceil}$, therefore the GP samples are in $C^{\lceil \nu - 1 \rceil}$. The Matérn class is a generalization of RBF and exponential kernels because $k^{\mathrm{M}}_{1/2} = k^{\exp}$ and $k^{\mathrm{M}}_\nu \xrightarrow{\nu \to \infty} k^{\mathrm{RBF}}$.

In figure 2.2 are illustrated three samples obtained from three GPs with different covariance functions: RBF, exponential and Matérn with $\nu = 3/2$. In each case $\sigma = \ell = 1$ is used. The three GPs share the same mean function $\mu(\theta) = \sin(\theta)$ and the same sample variance $k(\theta, \theta) = 1$, but their different covariance functions impose different level of smoothness to the samples: C^∞ with RBF, C^0 with exponential and C^1 with Matérn when ν is set to $3/2$.

Since the parametrized energy in equation (1.8) is a C^∞ function, the RBF kernel in equation (2.4) is an appropriate choice to model it with a GP.

Sample variance and characteristic length-scale

The parameters σ^2 and ℓ of the kernels (2.4), (2.5) and (2.6) are called, respectively, sample variance and characteristic length-scale. Since the sample variance is the diagonal term of the covariance functions, it is straightforward to verify that $k(\theta_\alpha, \theta_\alpha) = \sigma^2$ for the three kernels (2.4), (2.5) and (2.6). An example of the effect on random samples obtained by varying σ^2 is shown in figure 2.3. The deviations of the samples from the mean are, on average, two times wider when the value of σ doubles.

The characteristic length-scale ℓ determines the decay rate of the positive correlation between two points of a sample function. The sample functions are not likely to vary much along lines shorter than ℓ in the θ space as their values have a high positive correlation. Therefore, fixing ℓ to a certain value has the effect of suppressing sample oscillations with wave-lengths smaller than ℓ. An example of this effect is shown in figure 2.4. Three samples are drawn from a GP using different values of ℓ. In the case of $\ell = \pi/8$ are present short wave-length oscillations that are absent in the case of $\ell = \pi/2$.

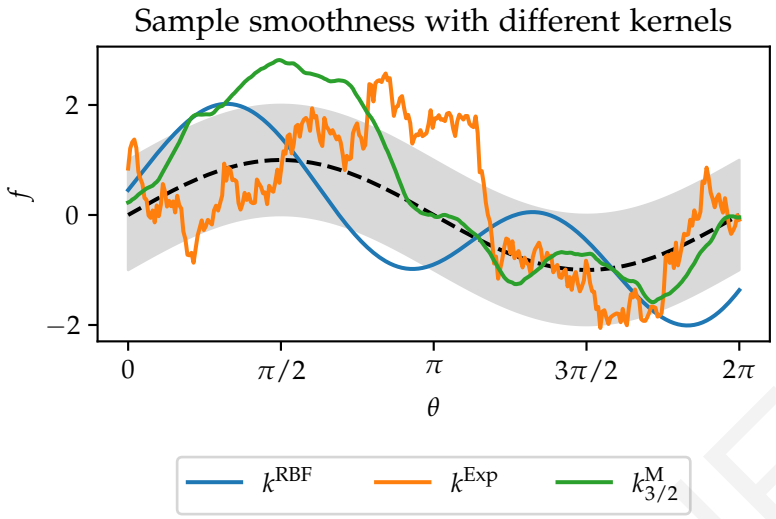


Figure 2.2: Three GP samples drawn using different covariance functions. They share the same mean function represented with a dashed black line and the same standard deviation shown as a grey interval. However, different choices of covariance functions impose different levels of smoothness.

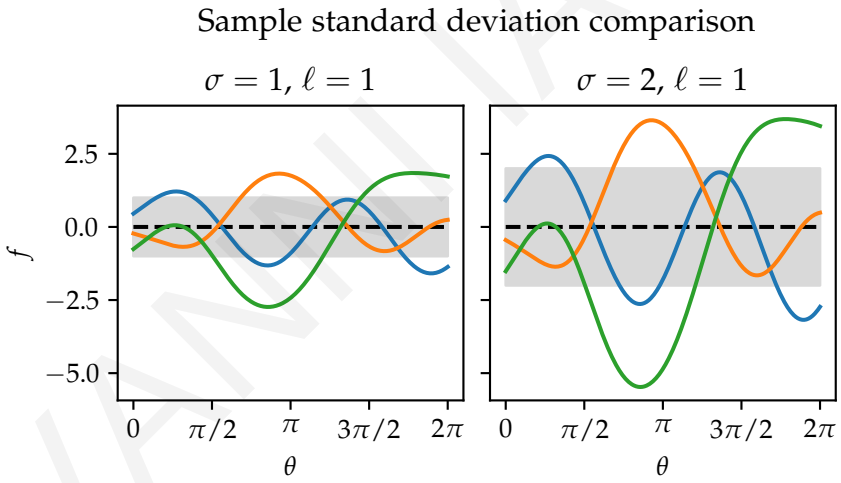


Figure 2.3: Three GP samples drawn using different values of the sample standard deviation. When σ is doubled, also the typical amplitude of oscillations is doubled.

Generalizations of characteristic length-scale

We have considered so far only covariance functions with a unique isotropic characteristic length-scale under which the GP samples vary with the same frequency in all θ space directions. However, in many situations, it is necessary to model phenomena that are more susceptible to some parameters than to others. In these cases, it is possible to use the kernels (2.4), (2.5) and (2.6) using a different ℓ_α for each single coordinate α. Their usage in the context of Bayesian inference is called automatic relevance determination (ARD) [56]. The meaning of this name will be clear in chapter 5, where Bayesian model selection will be described.

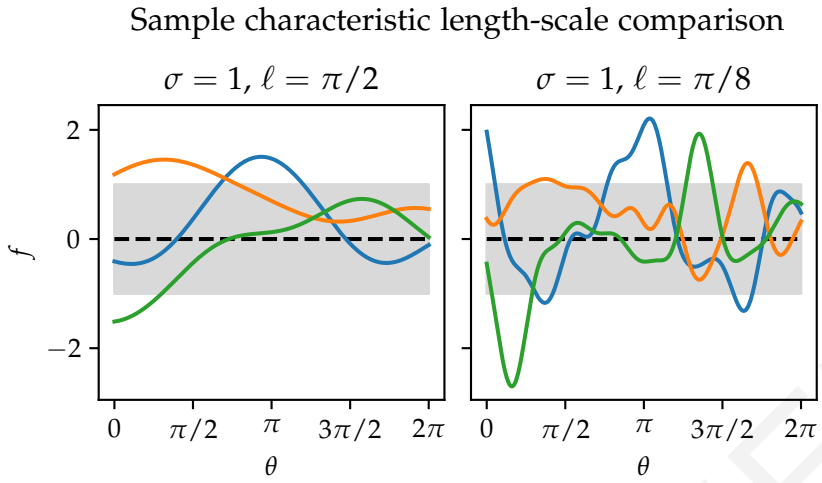


Figure 2.4: Three GP samples drawn using different values of the characteristic length-scale. When ℓ is increased, oscillations with wave-length shorter than ℓ are suppressed.

The ARD version of the RBF kernel (2.4) is the following:

$$k^{\text{RBF}}_{\text{ARD}}(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 \prod_\alpha e^{-\frac{(\theta_\alpha - \theta'_\alpha)^2}{2\ell_\alpha^2}} \tag{2.7}$$

where ℓ_α is the vector containing the different characteristic length-scales corresponding to each parameter in θ_α.

The generalization of ℓ can be further expanded rewriting the RBF kernel (2.4) in the following way:

$$k^{\text{RBF}}(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 \exp\left(-\frac{1}{2}\sum_{\alpha\beta}(\theta_\alpha - \theta'_\alpha)M_{\alpha\beta}(\theta_\beta - \theta'_\beta)\right)$$

where the $d \times d$ matrix M can be parametrized in different ways:

$$M^1_{\alpha\beta} = \ell^{-2}\delta_{\alpha\beta} \quad M^2_{\alpha\beta} = \ell_\alpha^{-2}\delta_{\alpha\beta} \quad M^3_{\alpha\beta} = (\Lambda\Lambda^\top)_{\alpha\beta} + \ell_\alpha^{-2}\delta_{\alpha\beta} \tag{2.8}$$

here Λ is an arbitrary $D \times L$ matrix with $L < D$. The parametrization M^1 corresponds to the isotropic case (2.4), M^2 to the ARD kernel (2.7) and M^3 is a generalization called *factor analysis distance* [16]. In addition to the single parameter directions of M^2, M^3 gives the possibility of tuning the length-scale along k arbitrary directions in the D-dimensional space. In particular, the L columns of Λ define L directions, along which the samples vary with higher frequency.

In figure 2.5 are shown three two-dimensional samples drawn with $\mu(\theta_\alpha) = 0$, $\sigma = 1$ and RBF kernel with the parametrizations (2.8). The M^1 matrix is here constructed with $\ell = \pi/2$. Since it is isotropic, the oscillations of the samples have the same characteristic length-scales in each direction. For M^2 it was used $\ell = (\pi/8, \pi)$, therefore the samples will oscillate far more rapidly in the θ_1 direction than in the θ_2 direction. Indeed, the sample profile

appears stretched in the θ_2 direction. Finally, M^3 was constructed with $\Lambda = (8/\pi, -8/\pi)$, $\ell = (2\pi, 2\pi)$. Thus, the direction along which the samples vary more rapidly is $\hat{\theta}_1 - \hat{\theta}_2$.

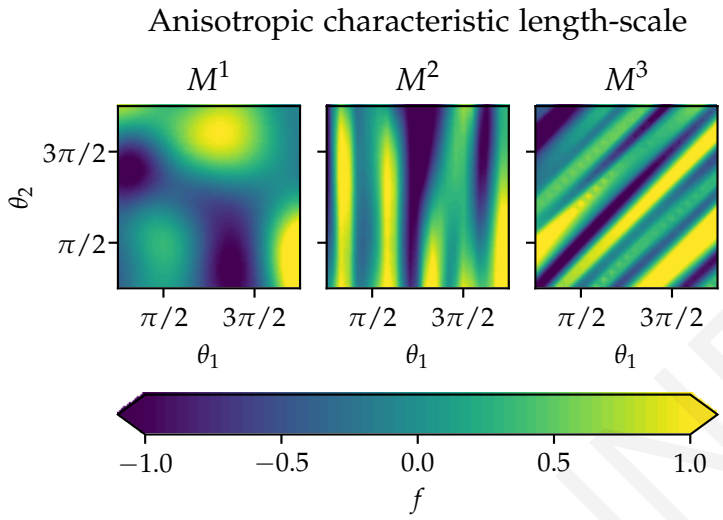


Figure 2.5: A two-dimensional GP sample drawn using different length-scale parametrizations. With M^1 the sample has isotropic oscillations, with M^2 it oscillates more frequently along one axis and with M^3 along an arbitrary direction.

Further generalizations of ℓ have been studied, in particular it is possible to break the stationarity of the covariance function using a characteristic length-scale $\ell(\theta_\alpha)$ that is not constant for each θ_α. For details about constructing kernels with this property, see [57, 58].

Kernel warping and periodic RBF

In many applications it is advised to create new kernels starting from widely known ones. There are several known procedures for modifying or combining kernels, preserving their positive definiteness [16].

A very useful procedure is known as kernel *warping*. It can be used to model complex patterns, such as in [59], or to impose periodic boundary conditions to the RBF kernel [60].

Given a covariance function $k(\theta_\alpha, \theta'_\alpha)$ and an arbitrary nonlinear mapping $u(\theta_\alpha)$, it is possible to define a new covariance function:

$$k_u(\theta_\alpha, \theta'_\alpha) = k(u(\theta_\alpha), u(\theta'_\alpha))$$

The positive definiteness of k_u follows directly from that of k. There is a lot of freedom in the choice of u as it doesn't need to be invertible. Furthermore, domain and codomain don't need to share the same dimensionality if the kernel k is well-defined in the corresponding space.

Periodicity of samples could be achieved mapping each parameter to the points of a circumference whose length corresponds to the period P. Each single parameter is therefore

mapped to a point in a two-dimensional space. The covariance function is then evaluated in this $2D$-dimensional space. Thus, the wanted warping function is:

$$u : \theta_\alpha \equiv \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_d \end{pmatrix} \mapsto u(\theta_\alpha) \equiv \begin{pmatrix} r\sin(\varphi_1) \\ r\cos(\varphi_1) \\ \vdots \\ r\sin(\varphi_d) \\ r\cos(\varphi_d) \end{pmatrix}$$

Where $r \equiv P/2\pi$ is the radius of the circumference and $\varphi_\alpha \equiv 2\pi\theta/P$ are the angular coordinates in the two-dimensional space.

Applying this warping to the RBF kernel (2.4):

$$k_u^{\text{RBF}} = \sigma^2 \exp\left(-\frac{r^2}{2\ell^2}\sum_\alpha (\sin(\varphi_\alpha) - \sin(\varphi'_\alpha))^2 + (\cos(\varphi_\alpha) - \cos(\varphi'_\alpha))^2\right)$$

Using the following trigonometric identities:

$$\begin{cases} \sin(\varphi)\sin(\varphi') + \cos(\varphi)\cos(\varphi') = \cos(\varphi - \varphi') \\ \cos(\varphi - \varphi') = 1 - 2\sin^2\left(\dfrac{\varphi}{2}\right) \end{cases}$$

k_u^{RBF} can be simplified since:

$$\begin{aligned} &(\sin(\varphi) - \sin(\varphi'))^2 + (\cos(\varphi) - \cos(\varphi'))^2 \\ &= 2(1 - \cos(\varphi - \varphi')) \\ &= 4\sin^2\left(\frac{\varphi - \varphi'}{2}\right) \end{aligned}$$

Thus, reintroducing θ_α and P, the periodic RBF kernel with period P is:

$$k^{\text{P}}(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 \exp\left(-\frac{P^2}{2\pi^2\ell^2}\sum_\alpha \sin^2\left(\frac{\pi}{P}(\theta_\alpha - \theta'_\alpha)\right)\right) \tag{2.9}$$

The multiplicative factors inside the exponential could be absorbed redefining ℓ, but we prefer to keep them explicit in order to maintain the correspondence with the RBF in the limit $\theta_\alpha \to \theta'_\alpha$.

In figure 2.6 it is shown a one-dimensional comparison between samples drawn with RBF and periodic kernels. It is clear from the figure that the samples drawn with the periodic kernel have periodic boundary conditions, while those obtained with RBF are not periodic, even though the mean function is periodic.

Finally, the ARD extension of the periodic kernel is straightforward:

$$k^{\text{P}}_{\text{ARD}}(\theta_\alpha, \theta'_\alpha) \equiv \sigma^2 \exp\left(-\frac{P^2}{2\pi^2}\sum_\alpha \frac{\sin^2\left(\frac{\pi}{P}(\theta_\alpha - \theta'_\alpha)\right)}{\ell_\alpha^2}\right) \tag{2.10}$$

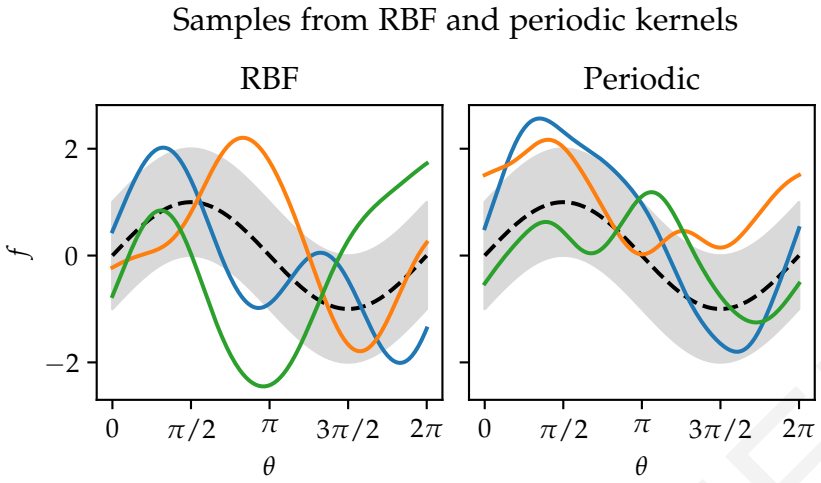


Figure 2.6: Three GP samples drawn using RBF and periodic kernels. If periodic kernel is used, all GP samples are periodic.

2.4 CONDITIONAL GAUSSIAN PROCESSES

In many applications, it is useful to consider only the GP samples that have certain values at specific domain points. This is the case, for example, when it is necessary to model a function that has few known values after previous measurements. Restricting the samples of a stochastic process to only those which pass through certain points might be in general computationally intensive, however this is not the case for GPs as the conditional probability of a multivariate Gaussian is available in closed form.

Given a GP f, we want to express $p(f_m^*|f_i)$, which is the conditional distribution of its sample values f_m^* evaluated at $\theta_{m\alpha}^*$ given the fixed sample values f_i at $\theta_{i\alpha}$. It can be proven [61] that the conditional distribution of a multivariate Gaussian is another multivariate Gaussian whose mean and covariance are:

$$\begin{cases} \mathbb{E}[f_m^*|f_i] = \mu(\theta_{m\alpha}^*) + \sum_{ij} k(\theta_{m\alpha}^*, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} (f_j - \mu(\theta_{j\alpha})) \\ \mathrm{Cov}[f_m^*, f_n^*|f_i] = k(\theta_{m\alpha}^*, \theta_{n\alpha}^*) - \sum_{ij} k(\theta_{m\alpha}^*, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} k(\theta_{j\alpha}, \theta_{n\alpha}^*) \end{cases} \tag{2.11}$$

where $\mu(\theta_\alpha)$ and $k(\theta_\alpha, \theta'_\alpha)$ are the mean and covariance function of the GP f, $K_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha})$ and $K_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha})$. Comparing this result with equations (2.1), (2.2) and (2.3), it is clear that the conditional GP is a new GP whose mean and covariance functions are:

$$\begin{cases} \mu(\theta_\alpha|f_i) = \mu(\theta_\alpha) + \sum_{ij} k(\theta_\alpha, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} (f_j - \mu(\theta_{j\alpha})) \\ k(\theta_\alpha, \theta'_\alpha|f_i) = k(\theta_\alpha, \theta'_\alpha) - \sum_{ij} k(\theta_\alpha, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} k(\theta_{j\alpha}, \theta'_\alpha) \end{cases}$$

In Fig. 2.7 is shown an example of conditional GP. In the left panel are shown three samples from a GP with constant zero mean and RBF covariance function. In the right panel are shown three samples from the conditional GP obtained using the fixed values f_i marked

with black circles. We can notice that the sample standard deviation converges to zero at the fixed values. Indeed, all the samples of the conditional GP are required to pass through f_i. Therefore, the samples of the conditional GP can be seen as possible interpolations of the fixed values f_i.

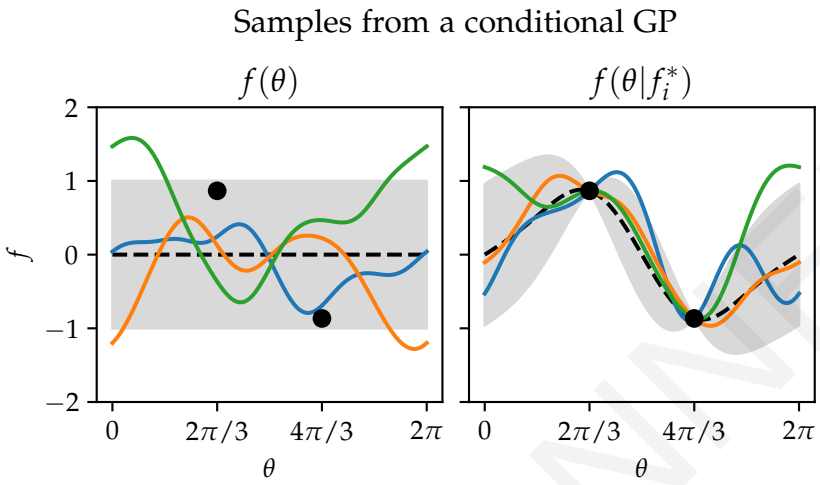


Figure 2.7: The three samples drawn from the conditional GP interpolate the points marked with black circles.

We will explain in Chapter 4 how this result is of crucial importance for GP regression. In particular, it will be shown that this result coincides with the GP regression of noiseless measurements.

3

INTRODUCTION TO BAYESIAN INFERENCE

Bayesian probability is an interpretation of the concept of probability that is alternative to the widely known frequentist probability. Whereas frequentist probability is defined as the relative frequency of an event, Bayesian probability is rather interpreted as reasonable expectation [62] or personal belief [63]. Its name was given in the 1950s for the pioneering work of Thomas Bayes, a Presbyterian minister and mathematician of the eighteenth century [64]. These ideas were known before under the name of inverse probability.

Even though frequentist statistics is the most followed approach in experimental sciences, Bayesian methods have found many applications, especially in the context of machine learning [65], since they offer many strategies to deal with model uncertainty and to include prior beliefs in the analysis.

3.1 BAYES THEOREM

The starting point of any Bayesian analysis is a hypothesis about the generative probabilistic process giving rise to observable data x. This generative model is the *sampling distribution* $p(x|\phi, \xi)$, which is expressed in terms of a set of parameters ϕ and hyperparameters ξ. The subject of statistical inference is to determine the parameters ϕ. In some cases, the parameters ϕ themselves are considered subject to another generative process $p(\phi|\xi)$ expressed in terms of hyperparameters ξ.

The sampling distribution is also called *likelihood* function, since, after gathering the data x from measurements, the value of $p(x|\phi, \xi)$ indicates the *likelihood* of the parameters ϕ given the measured value x. Indeed, the values of the parameters ϕ that are compatible with the data x are considered to be more probable than those that are not compatible with x.

In order to perform Bayesian inference, it is necessary to specify a *prior* distribution $p(\phi|\xi)$ that describes the beliefs and uncertainties about the true values of the parameters ϕ before any measurement is taken. Here is where prior knowledge about the problem should be used. The prior distribution could be determined using physical properties, symmetries, or even subjective beliefs. It is important that $p(\phi|\xi) > 0$ for any plausible value of ϕ, since a zero prior probability excludes a value with certainty. This precaution is also known as Cromwell's rule [66] to refer to a famous quotation:

> *I beseech you, in the bowels of Christ, think it possible that you may be mistaken.*
>
> — General Oliver Cromwell, letter to the General Assembly of the Kirk of Scotland, 1650

The main step of Bayesian inference is to find the *posterior* distribution $p(\phi|x,\xi)$ of the parameters ϕ after having observed x. This is done *updating* the prior using Bayes' theorem:

$$p(\phi|x,\xi) = \frac{p(x|\phi,\xi)p(\phi|\xi)}{p(x|\xi)} \tag{3.1}$$

where $p(x|\xi) = \int p(x|\phi,\xi)p(\phi|\xi)\mathrm{d}\phi$ is the *marginal likelihood* (or *evidence*). Bayes' theorem follows directly from the definition of conditional probability. It can be interpreted as a reweighting of the prior in which the prior beliefs about ϕ are rescaled according to their compatibility with the measured data x.

To summarize, in order to perform Bayesian inference of some parameters ϕ, two elements are required: the prior distribution of ϕ and the likelihood function of the parameters ϕ given the measured data x. These two choices might be immediate or tricky depending on the specific problem. While the choice of the likelihood should always have objective motivations dictated by the considered problem, the prior can be set to be very generic as long as Cromwell's rule is satisfied. In many practical situations, the prior is chosen to be a *conjugate prior* for a specific likelihood, which means that the posterior $p(\phi|x,\xi)$ will be in the same distribution family as the prior $p(\phi|\xi)$ after applying Bayes' theorem (3.1). This choice is an algebraic convenience in order to have a closed-form expression for the posterior.

In some cases, it is not possible to use conjugate priors and the integration of the likelihood might be problematic. Fortunately, it is possible to sample the posterior distribution $p(\phi|x,\xi)$ without having to evaluate the marginal likelihood at all. Indeed, since $p(\phi|x,\xi) \propto p(x|\phi,\xi)p(\phi|\xi)$, it is possible to use Markov Chain Monte Carlo (MCMC) techniques (e.g. Gibbs sampling [67] or slice sampling [68]) to sample the posterior as they usually don't require evaluating the normalizing constant.

The choice of the prior distribution is the most peculiar and controversial aspect of Bayesian analysis because there isn't a general rule to determine it and the final result depends on this choice. However, as it will be explained in the rest of the chapter, Bayesian inference offers several advantages over frequentist approaches in prediction, decision-making and model selection. The arbitrariness of the prior is not a problem when enough data is collected such that the final result becomes almost independent of the prior choice. In some cases this freedom of choice might even be an advantage, for example when the availability of reliable data is very limited and the analysis has to rely on subjective information that could be included into the prior. This is the situation, for example, in the search and recovery of lost objects at sea. The famous recovery of a nuclear bomb lost by the United States Air Force off the coast of Spain in 1966 was achieved using the information given by local fishermen and Bayesian search theory [69].

3.2 BAYESIAN PREDICTIONS

In the context of machine learning, the interest of the analysis is not concerning ϕ itself, but rather the prediction of another unobserved random variable y dependent on ϕ and ξ under a probabilistic model $p(y|\phi,\xi)$ using the information gathered measuring x.

Once the posterior distribution $p(\phi|x,\xi)$ is obtained with Bayes' theorem (3.1), the *posterior predictive distribution* is found by marginalizing the parameters ϕ:

$$p(y|x,\xi) = \int p(y|\phi,\xi)p(\phi|x,\xi)\mathrm{d}\phi \tag{3.2}$$

Often it is useful to produce analogous predictions using the prior distribution instead of the posterior in order to observe the impact of measurements on the predictions. The *prior predictive distribution* is then:

$$p(y|\xi) = \int p(y|\phi,\xi)p(\phi|\xi)\mathrm{d}\phi \tag{3.3}$$

Here is possible to appreciate an advantage of Bayesian statistics. While (3.2) estimates the whole distribution of y, predictions with frequentist approaches are usually performed after a point or interval estimate of ϕ, which means that only limited information about the distribution of ϕ would be used for the prediction of y. In some cases, the distribution of y could still be obtained using specific distribution properties, for example using the central limit theorem and Student's t-distribution. In other cases, a frequentist analysis might have to rely on resampling methods [70] to get more information about the distribution of y.

On the other hand, the Bayesian posterior predictive distribution (3.2) is available in closed form if conjugate priors are used, and, in the general case, it could be approximated with arbitrary precision using MCMC. This could be very useful if further analysis needs to be performed using the predictive values of y, for example in decision theory, as will be explained in the next section.

3.3 BAYESIAN DECISION THEORY

Bayesian analysis usually requires deducing further statements after finding the posterior distribution $p(\phi|x,\xi)$ or the posterior predictive distribution $p(y|x,\xi)$. These statements might regard ϕ or y themselves (e.g. their expected value) or other variables dependent on them. A wide class of such problems may be tackled by Bayesian decision theory. The point estimate of ϕ, for example, falls into this category.

A decision problem involves defining a loss function $L(z,y)$ or $L(z,\phi)$ in terms of a decision variable z, the prediction variable y or the parameters ϕ. The loss function is then integrated over the posterior predictive distribution (or the posterior distribution) in order to obtain the expected loss. The optimal Bayesian decision is then defined as the minimum of the expected loss among the possible values of the decision variable z. In some cases it is more natural to define a utility function rather than a loss function, and the optimal

Bayesian decision will then be the maximum point of the expected utility. Instances of utility function maximization are the acquisition functions covered in Chapter 7.

Let us illustrate the most common example of Bayesian decision theory, which is the point estimation of ϕ. Calling ϕ' a candidate to be the point estimate of the true value of ϕ, the loss function $L(\phi, \phi')$ measures the discrepancy between ϕ and ϕ'. Here ϕ' has the role of the decision variable z of the definition above, therefore the function to minimize is the expected loss defined as:

$$EL(\phi'|x, \xi) \equiv \int L(\hat{\phi}, \phi) p(\phi|x, \xi) \mathrm{d}\phi \tag{3.4}$$

where $p(\phi|x, \xi)$ is the posterior distribution (3.1). The optimal Bayesian decision about the estimation of ϕ is then:

$$\hat{\phi} \equiv \arg\min_{\phi'} EL(\phi'|x, \xi)$$

The final result depends on the definition of the loss function: if $L(\phi, \phi') = (\phi - \phi')^2$, then $\hat{\phi}$ is the mean value of the posterior; if $L(\phi, \phi') = |\phi - \phi'|$, then $\hat{\phi}$ is the median of the posterior; if $L(\phi, \phi') = \delta(\phi - \phi')$, then $\hat{\phi}$ is the mode of the posterior.

This last example is also called maximum a posteriori (MAP) and it is of particular interest as it can be computed without evaluating the marginal likelihood and without using MCMC techniques. Indeed, the maximum point of the posterior (3.1) is the same regardless of the multiplicative constant, so the MAP estimate can simply be found with the following optimization:

$$\hat{\phi}^{\mathrm{MAP}} \equiv \max_{\phi} p(\phi|x, \xi) = \max_{\phi} p(x|\phi, \xi) p(\phi|\xi)$$

since $p(\phi|x, \xi) \propto p(x|\phi, \xi) p(\phi|\xi)$ from (3.1).

3.4 BAYESIAN MODEL SELECTION

The missing piece of our analysis is the choice of the hyperparameters ξ that identify the generative process of the parameters ϕ. We will describe three different approaches. Each of them is a generalization of the previous one.

Maximum likelihood estimation

The likelihood of the hyperparameters ξ given the measured data x is $p(x|\xi)$. Keeping the measured data x fixed, its maximum point over the possible hyperparameters ξ represents the values of ξ that are most compatible with the data x. Recalling the steps of the Bayesian inference described so far, we can notice that $p(x|\xi)$ already appeared in equation (3.1). Indeed, it is the marginal likelihood of the hyperparameters, in which the dependence on the parameters ϕ has been marginalized away:

$$p(x|\xi) = \int p(x|\phi, \xi) p(\phi|\xi) \mathrm{d}\phi \tag{3.5}$$

Therefore, the maximum likelihood estimation (MLE) of ξ is performed as:

$$\xi^{\text{MLE}} \equiv \arg\max_{\xi} p(x|\xi) \tag{3.6}$$

In many cases (3.5) can be evaluated in closed form, so that the optimization (3.6) can be performed efficiently.

At first sight, it may seem that (3.6) overfits the data as it just selects the model with the highest data compatibility. However, at closer look, it is clear that this is not the case [71]. Overfitting means fitting the data with a model that is too complex and generic, so that it adapts very well to a wide variety of different possible measurements. This usually means having a weak predictive power on points that are distant to those used for the inference. For example, this happens when a high degree polynomial is used to fit a noisy dataset generated with a power law with lower degree. The fitted function will track very well the training dataset, but it will totally miss predictions of new data taken further from the rest as the higher degree components of the polynomial dominate asymptotically.

The reason why overfitting shouldn't happen for the MLE (3.6) is that $p(x|\xi)$ is always normalized to one in the x space, for each value of ξ. Complex models that could fit a high variety of datasets x correspond to wide distributions $p(x|\xi)$ as they are compatible with wide intervals of x. However, the distribution $p(x|\xi)$ of complex models is low valued because it has to be normalized to one. On the other hand, very simple models tend to have a distribution peaked in small intervals of x and close to zero in the rest of the domain. This means that their probability density function is far higher close to the peaks than any values found in wider distributions.

An example of this behaviour is illustrated in Fig. 3.1. The three curves represent three Gaussians centered at the same point but with different sigmas. The one with the highest variance represents a *complex* model, the one with the lowest variance a *simple* model and the one in between an *intermediate* model. The complex model is compatible with the highest amount of possible data because its wider probability density function keeps high values for a larger interval of possible measurements x. However, considering the measurement x depicted as a dashed line, even though its value is at half sigma from the mean of the complex model, the likelihood is higher for the intermediate model, even though it is at one sigma from its mean. On the other hand, the likelihood of the simple model is already too low as the marked point is at two sigmas from the mean.

Model selection through MLE, therefore, implements a trade-off between model fit and model complexity, or, rephrasing this in other terms, it selects the simplest model among those that are compatible with the data. This is an instance of Occam's razor, named after William of Ockham, a Franciscan friar and philosopher that frequently applied this principle in his works:

> *Frustra fit per plura quod potest fieri per pauciora.*
>
> — William of Ockham, Summa Logicae, 1323

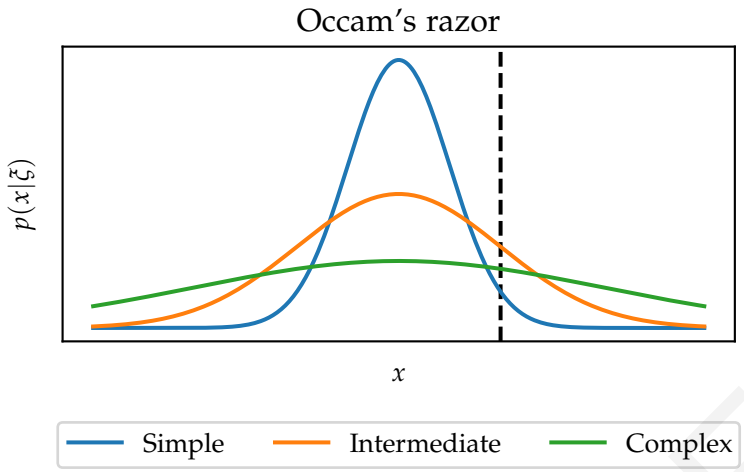


Figure 3.1: Three possible models for explaining the data x marked with a dashed line. The model with the highest marginal likelihood is the one of intermediate complexity.

The procedure of fixing the hyperparameters with MLE is usually referred as maximum likelihood estimation of type II (MLE-II) because MLE is used to fix the hyperparameters, which are a *second level* of parameters.

Maximum a posteriori

In some cases, especially when there is not much available data or when there are many hyperparameters to fix, it might happen that MLE estimates unreasonable values for ξ. For example, it might select extremely simple or extremely complex models. In these cases, it is advised to rely on regularization techniques, which, in the context of Bayesian analysis, usually involve introducing a new prior distribution on the hyperparameters [71, 72].

We will call *hyperprior* $p(\xi)$ this prior distribution of the hyperparameters. Its main purpose it is to avoid results clearly outside the range of plausibility, but it can also be used to incorporate some prior knowledge about ξ or to strengthen the Occam's razor.

The *hyperposterior* is then found applying Bayes' theorem:

$$p(\xi|x) = \frac{p(x|\xi)p(\xi)}{p(x)} \tag{3.7}$$

where $p(x|\xi)$ is the marginal likelihood (3.5), in which only ϕ has been marginalized out, while in $p(x) = \int p(x|\xi)p(xi)\mathrm{d}\xi$ also the hyperparameters are marginalized out using the hyperprior. The MAP estimation of ξ is then:

$$\xi^{\text{MAP}} \equiv \arg\max_{\xi} p(\xi|x)$$

From its definition, it is clear that MAP is a generalization of MLE as they yield the same result if a constant hyperprior is used.

Usually $p(x)$ is not available in closed form and its evaluation might be not trivial. Fortunately, as explained in Sec. 3.3, evaluating $p(x)$ is not necessary for computing the MAP as the multiplicative constant is irrelevant for the optimization.

At first sight, fixing the hyperparameters ξ with MLE or MAP might seem to violate the rules of Bayesian inference as the empirical data x is used to fix ξ, which defines the prior $p(\phi|\xi)$ that should be specified before any measurement. The prior is then not only expressing prior beliefs, but it is also the result of a data fit. However, it is the hyperprior $p(\xi)$ that is chosen before any data is taken. The hyperprior could therefore be seen as the *real* prior that defines the ensemble of all possible priors $p(\phi|\xi)$.

Fully Bayesian

The introduction of the hyperprior $p(\xi)$ opens different possibilities for the estimation of ξ. For example, all the point estimation methods introduced in section 3.3 could be used. However, in order to exploit all the potential of Bayesian inference, we could use all the information of the hyperposterior $p(\xi|x)$ considering all the possible values of the hyperparameters ξ weighted according to the hyperposterior. This might sound unrealistic, but can in fact be done efficiently with MCMC, or, in some cases, even in closed form [56].

Let us consider the posterior prediction distribution $p(y|x,\xi)$ in equation (3.2). The predictions can take into account the contribution of all possible ξ if we integrate $p(y|x,\xi)$ over ξ:

$$p(y|x) = \int p(y|\phi,\xi)p(\phi|x,\xi)p(\xi|x)\mathrm{d}\phi\mathrm{d}\xi$$

Which means that the hyperparameters are marginalized out of the predictions, adding a second layer to the Bayesian inference. The predictions obtained in this way are also known as *fully Bayesian* predictions. The same integration could be done to get the fully Bayesian posterior $p(\phi|x)$ or fully Bayesian expected loss $EL(z|x)$.

In same cases, there are further levels of hyperparameters $\xi_1, \xi_2, ...$, where the prior $p(\xi_n|x_{n+1}, ...)$ of each set of hyperparameters ξ_n is determined by those present in all deeper levels. The fully Bayesian procedure can then be applied up to any arbitrary level, marginalizing out all the hyperparameters ξ_n. If it is impossible to obtain closed-form fully Bayesian formulae, it might become very resource demanding to approximate them with MCMC if many layers of hyperparameters are present. Indeed, the complexity of the approximation increases exponentially with the number of layers.

4

GAUSSIAN PROCESS REGRESSION

Gaussian process regression (GPR) is a procedure that interpolates spacial data with a GP using Bayesian inference. It was first introduced in geostatistics in the 1960s with the name of Kriging [73], but nowadays is used in many fields, most notably in machine learning [16] and Bayesian optimization [13].

GPR falls into the category of non-parametric regression procedures as the resulting function is not expressed in a parametric form, but rather in terms of the training data. Indeed, GPR could in principle interpolate any set of data without changing its hyperparameters, since it is an interpolation method. However, as will be explained later, GPR offers methods to infer functions from noisy data, which is an uncommon feature among interpolation methods.

Our interest lies in using GPR to interpolate a set of noisy energy measumeremts performed with a quantum computer. In this chapter, we will show how GPR accomplishes this task using the concepts introduced in Chapters 2 and 3. It uses a GP as a prior distribution, and uses the empirical data to model the unknown parametrized energy with Bayesian prediction theory. We will discuss both GPR in case of noiseless data and in case of data afflicted by Gaussian noise. Finally, we will discuss how the operations required by GPR can be efficiently implemented.

4.1 BAYESIAN INFERENCE USING QUANTUM MEASUREMENTS

Let us consider a set of N energy measurements $E_i \pm \Delta E_i$, each of them obtained respectively using a D-dimensional circuit parameter vector $\theta_{i\alpha}$.

In order to perform Bayesian inference, we need first to specify a probabilistic generative model of the data in terms of a set of parameters f_i and hyperparameters ξ. In our model, the parameters f_i are the estimators of the unknown target energy function $E(\theta_{i\alpha})$ evaluated in a set $\theta_{i\alpha}$ of circuit parameters. The parameters f_i have here the role of the parameters ϕ in chapter 3. Since the N energy measurements are taken independently, the probability $p(E_i|f_i)$ of obtaining the measurements E_i given certain values of the parameters f_i is a product of Gaussians centered in f_i with sigmas equal to ΔE_i.

Once it is specified how to evaluate the likelihood $p(E_i|f_i)$, the next step is placing a prior distribution on the parameters f_i. The exact unknown energy function can be seen as a possible sample of a GP f, so we can treat f_i as the values assumed by this specific GP sample when evaluated at $\theta_{i\alpha}$. Therefore, the prior distribution of the parameters f_i

is determined by the GP f. Indeed, it is the multivariate Gaussian (2.1). Since the GP f generates the prior distribution of f_i, we will call it prior GP.

As was explained in Chapter 2, in order to fully specify the GP f, it is sufficient to define a mean and a covariance function. The mean function of the prior GP is often chosen to be a constant function as this choice expresses prior neutrality about the possible measurements. In cases where prior knowledge about the values of the target function is available, it is possible to define a mean function that reflects this knowledge. In our tests we used a constant prior mean function:

$$\mu(\theta_\alpha) = \mu$$

For the covariance function, it is possible to use any of the kernels introduced in chapter 2. In any case, they will depend on the sample variance σ^2 and on the characteristic length-scale ℓ, which could be just a scalar or a more complex form as explained in Sec. 2.3. The values of μ, σ^2 and ℓ can be treated as hyperparameters of the prior distribution. As such, we will call them ξ in analogy with the discussion of chapter 3, and their value will be fixed with Bayesian model selection in chapter 5.

Finally, the posterior distribution of the parameters f_i is evaluated using Bayes' theorem (3.1):

$$p(f_i|E_i, \xi) = \frac{p(E_i|f_i)p(f_i|\xi)}{p(E_i|\xi)} \tag{4.1}$$

where the marginal likelihood is given by $p(E_i|\xi) = \int p(E_i|f_i, \xi)p(f_i|\xi)\mathrm{d}f_i$.

An advantage of using a GP prior is that the posterior (4.1) is available in closed form if the likelihood is a Gaussian. Indeed, a multivariate Gaussian distribution is a conjugate prior for a multivariate Gaussian likelihood. This is a great computational advantage, especially when the posterior is used to perform predictions and take decisions.

In the next section, we will explain how to perform predictions of the target energy function, which will be used in chapter 7 to estimate the global minimum of the energy using Bayesian decision theory.

4.2 BAYESIAN PREDICTION OF QUANTUM MEASUREMENTS

In our formalism, the unknown energy function $E(\theta_\alpha)$ is a possible sample of the prior GP f. In section 2.4 we saw how to restrict the space of the samples of a GP to those that pass through certain fixed values, and the remaining samples constitute another conditional GP. Since the parameters f_i are our estimators of $E(\theta_{i\alpha})$, we are interested in conditioning the prior GP f on the values f_i at $\theta_{i\alpha}$. The resulting conditional GP allows us to assign a predictive probability density function $p(f^*_m|f_i, \xi)$ to any set of energy estimators f^*_m evaluated at $\theta^*_{m\alpha}$. We also know that the mean and the covariance of their multivariate Gaussian distribution $p(f^*_m|f_i, \xi)$ are given by equation (2.11).

As was explained in section 3.2, rather than fixing particular values for the parameters f_i, Bayesian predictions are usually made considering the whole possible outcomes of f_i, according to their posterior distribution. This is done marginalizing out the parameters f_i

(see equation (3.2)). Thus, the posterior predictive distribution of the energy estimators f_m^* is:

$$p(f_m^*|E_i, \xi) = \int p(f_m^*|f_i, \xi) p(f_i|E_i, \xi) \mathrm{d}f_i \tag{4.2}$$

while their prior predictive distribution $p(f_m^*|\xi)$ is simply given by the prior GP f.

Another advantage of using GPs is that also the posterior predictive distribution $p(f_m^*|E_i, \xi)$ is a multivariate Gaussian. Therefore, equation (4.2) defines a posterior predictive GP, whose mean and covariance functions define our energy estimator for any value of the circuit parameters θ_α. Indeed, denoting them as $\mu(\theta_\alpha|E_i)$ and $k(\theta_\alpha, \theta'_\alpha|E_i)$, the parametrized energy $E(\theta_\alpha)$ is estimated as:

$$\mu(\theta_\alpha|E_i) \pm \sqrt{k(\theta_\alpha, \theta_\alpha|E_i)} \xrightarrow{\text{estimates}} E(\theta_\alpha) \tag{4.3}$$

The one-sigma prediction intervals defined by (4.3) are called *credible intervals* in Bayesian inference. They are analogous to *confidence intervals* found in frequentist statistics.

Apart from giving energy predictions with errors, the posterior predictive GP can generate samples of possible energy functions, which means not only generating independent energy estimations for single value of θ_α, but also generating independent noiseless sample energy functions defined for all θ_α. These samples can be useful in decision theory. Indeed, they will be used in section 7.2 for evaluating the noisy expected improvement acquisition function.

In the next two sections, we will evaluate the posterior predictions (4.2) both in the case of noiseless measurements and in the case of measurements affected by Gaussian noise.

4.3 INFERENCE AND PREDICTION: NOISELESS CASE

In absence of statistical noise, the measurements E_i will always correspond to the exact values of the energy function, which in our formalism are estimated by the parameters f_i. Therefore, the sampling distribution of E_i is simply a Dirac delta function centered in f_i:

$$p(E_i|f_i) = \delta(E_i - f_i)$$

which means that the marginal likelihood is:

$$p(E_i|\xi) = \int p(E_i|f_i) p(f_i|\xi) \mathrm{d}f_i = p(f_i = E_i|\xi)$$

and, using Bayes' theorem (4.1), the posterior is:

$$p(f_i|E_i, \xi) = \frac{\delta(f_i - E_i) p(f_i|\xi)}{p(f_i = E_i|\xi)}$$

This result can be plugged in equation (4.2) to perform posterior energy predictions f_m^*:

$$p(f_m^*|E_i, \xi) = p(f_m^*|f_i = E_i, \xi) \tag{4.4}$$

which means that the posterior predictive distribution of the energy estimators f_m^* is the multivariate Gaussian obtained conditioning the prior GP f on having $f_i = E_i$ at $\theta_{i\alpha}$. Thus, the mean and covariance of (4.4) are found substituting f_i with E_i in equation (2.11):

$$\begin{cases} \mathbb{E}[f_m^*|E_i] = \mu(\theta_{m\alpha}^*) + \sum_{ij} k(\theta_{m\alpha}^*, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} (E_j - \mu(\theta_{j\alpha})) \\ \mathrm{Cov}[f_m^*, f_n^*|E_i] = k(\theta_{m\alpha}^*, \theta_{n\alpha}^*) - \sum_{ij} k(\theta_{m\alpha}^*, \theta_{i\alpha}) \left(K^{-1}\right)_{kl} k(\theta_{j\alpha}, \theta_{n\alpha}^*) \end{cases} \tag{4.5}$$

They define a posterior predictive GP whose mean and covariance functions are:

$$\begin{cases} \mu(\theta_\alpha|E_i) = \mu(\theta_\alpha) + \sum_{ij} k(\theta_\alpha, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} (E_j - \mu(\theta_{j\alpha})) \\ k(\theta_\alpha, \theta'_\alpha|E_i) = k(\theta_\alpha, \theta'_\alpha) - \sum_{ij} k(\theta_\alpha, \theta_{i\alpha}) \left(K^{-1}\right)_{ij} k(\theta_{j\alpha}, \theta'_\alpha) \end{cases} \tag{4.6}$$

This posterior GP allows us to predict the energy $E(\theta_\alpha)$ for any value of the circuit parameters θ_α as shown in equation (4.3).

An example of this inference procedure is depicted in Fig. 4.1. The data marked in orange is *measured* in the points $\theta = 2\pi/3, 4\pi/3$ from the sine function drawn as a black dashed line. The prior GP has a zero valued mean function and its covariance function is the periodic kernel (2.9) with $\sigma = \ell = 1$. The prior and posterior predictions are depicted as blue lines while their one-sigma credible intervals are drawn in light blue. The prior predictions and their errors are given by the prior mean and the sample variance, which in this case are uniformly zero and one. The posterior predictions are given by equations (4.6) and (4.3).

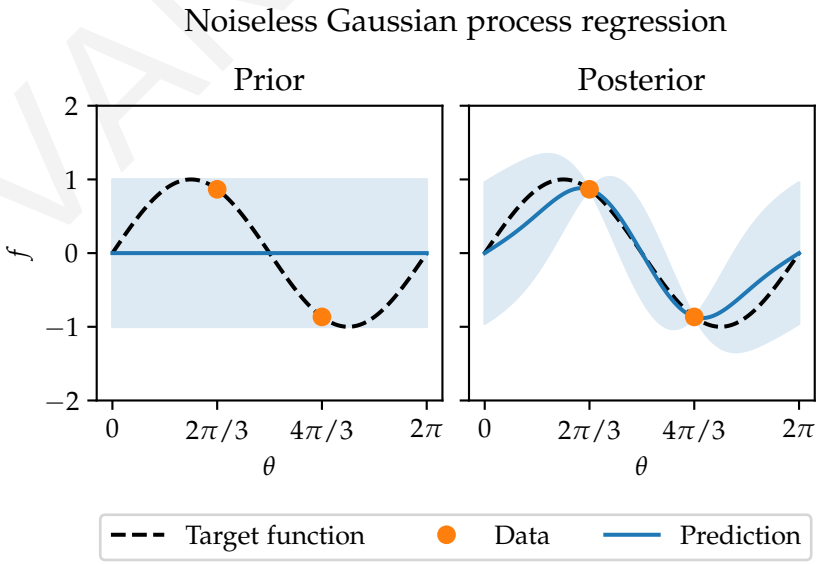


Figure 4.1: GPR of noiseless data using the periodic kernel, $\mu = 0$, $\sigma^2 = 1$ and $\ell = 1$.

4.4 INFERENCE AND PREDICTION: HETEROSCEDASTIC GAUSSIAN NOISE

In this case, the measurements E_i have independent Gaussian noise with standard deviations ΔE_i. The sigmas of the Gaussian noise are different for each measurement. In statistics, this property is called *heteroscedasticity*.

Assuming that the parameters f_i represent the correct values of the energy function, the sampling distribution $p(E_i|f_i, \xi)$ of the measurements E_i is a multivariate Gaussian centered in f_i with a diagonal covariance matrix whose elements are ΔE_i^2.

We will use the notation $\mathcal{N}(x|a, B)$ to indicate the multivariate Gaussian distribution of a variable x with mean a and covariance matrix B. Thus, the sampling distribution of the energy measurements can be written as:

$$p(E_i|f_i) = \mathcal{N}(E_i|f_i, S_{ij}) \tag{4.7}$$

where $S_{ij} \equiv \Delta E_i^2 \delta_{ij}$. Similarly, the prior distribution of the parameters f_i is:

$$p(f_i|\xi) = \mathcal{N}(f_i|\mu_i, K_{ij}) \tag{4.8}$$

where $\mu_i \equiv \mu(\theta_{i\alpha})$ and $K_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha})$ are constructed using the mean and the covariance functions of the prior GP f.

Using the equation (A.2) of the appendix, it is possible to rearrange the product between (4.7) and (4.8):

$$\mathcal{N}(E_i|f_i, S_{ij})\mathcal{N}(f_i|\mu_i, K_{ij}) = \mathcal{N}(E_i|\mu_i, (K+S)_{ij})\mathcal{N}(f_i|m_i, C_{ij})$$

where $m \equiv C(K^{-1}\mu + S^{-1}f)$ and $C \equiv (K^{-1} + S^{-1})^{-1}$.

This result can be used to compute the marginal likelihood and the posterior (4.1). Indeed, the marginal likelihood is:

$$p(E_i|\xi) = \int p(E_i|f_i)p(f_i|\xi)\mathrm{d}f_i = \mathcal{N}(E_i|\mu_i, (K+S)_{ij}) \tag{4.9}$$

and the posterior:

$$p(f_i|E_i, \xi) = \frac{p(E_i|f_i)p(f_i|\xi)}{p(E_i|\xi)} = \mathcal{N}(f_i|m_i, C_{ij}) \tag{4.10}$$

At this point, there are a few observations worth mentioning. The posterior distribution is a multivariate Gaussian along with the likelihood and the prior. Indeed, the multivariate Gaussian distribution is a conjugate prior for a multivariate Gaussian likelihood. Furthermore, the measurement errors ΔE_i have been used for the inference of the posterior distribution as they constitute the covariance matrix S. The only necessary assumption concerning the measurement errors is that they are Gaussian, which is approximately true in many applications thanks to the central limit theorem. For example, in our case, single measurements from the quantum computer are not necessarily Gaussian [25], but their average across many shots is approximately Gaussian for the central limit theorem.

The posterior (4.10) can be plugged in the integral (4.2) in order to find the posterior predictive distribution $p(f_m^*|E_i, \xi)$ of the energy estimators f_m^* at $\theta_{m\alpha}^*$:

$$p(f_m^*|E_i, \xi) = \int \mathcal{N}(f_m^*|\mu_i^*, K_{ij}^*)\mathcal{N}(f_i|m_i, C_{ij})\mathrm{d}f_i$$

where $\mu_i^* \equiv \mathbb{E}[f_m^*|f_i]$ and $K_{ij}^* \equiv \mathrm{Cov}[f_m^*, f_n^*|f_i]$ are defined using equation (2.11). The product $\mathcal{N}(f_m^*|\mu_i^*, K_{ij}^*)\mathcal{N}(f_i|m_i, C_{ij})$ can be rearranged using using the equation (A.3) of the appendix in order to isolate the integrating variable f_i.

After performing the integration, the posterior prediction distribution $p(f_m^*|E_i, \xi)$ is again a multivariate Gaussian whose mean and covariance are:

$$\begin{cases} \mathbb{E}[f_m^*|E_i] = \mu(\theta_{m\alpha}^*) + \sum_{ij} k(\theta_{m\alpha}^*, \theta_{i\alpha}) \left(\widetilde{K}^{-1}\right)_{ij} (E_j - \mu(\theta_{j\alpha})) \\ \mathrm{Cov}[f_m^*, f_n^*|E_i] = k(\theta_{m\alpha}^*, \theta_{n\alpha}^*) - \sum_{ij} k(\theta_{m\alpha}^*, \theta_{i\alpha}) \left(\widetilde{K}^{-1}\right)_{ij} k(\theta_{j\alpha}, \theta_{n\alpha}^*) \end{cases} \tag{4.11}$$

where μ and k are the mean and covariance functions of the prior GP f, and $\widetilde{K}_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha}) + \Delta E_i^2 \delta_{ij}$. We can notice that the only difference between the noisy posterior predictions (4.11) and the noiseless ones (4.4) is the addition of the noise variances to the diagonal of K.

Since the posterior prediction distribution is a multivariate Gaussian for any set of domain point $\theta_{m\alpha}^*$, it defines a posterior GP whose mean and covariance functions are:

$$\begin{cases} \mu(\theta_\alpha|E_i) = \mu(\theta_\alpha) + \sum_{ij} k(\theta_\alpha, \theta_{i\alpha}) \left(\widetilde{K}^{-1}\right)_{ij} (E_j - \mu(\theta_{j\alpha})) \\ k(\theta_\alpha, \theta'_\alpha|E_i) = k(\theta_\alpha, \theta'_\alpha) - \sum_{ij} k(\theta_\alpha, \theta_{i\alpha}) \left(\widetilde{K}^{-1}\right)_{ij} k(\theta_{j\alpha}, \theta'_\alpha) \end{cases} \tag{4.12}$$

An example of noisy GPR is shown in figure (4.2). The result is the noisy equivalent of figure (4.1). In this case, the data is smeared with heteroscedastic Gaussian noise. We can see that the posterior predictions in the right panel fit the data taking into account the error intervals instead of just interpolating their mean values.

4.5 EFFICIENT IMPLEMENTATION OF GPR

The evaluation of posterior predictions (4.6) and (4.12) requires the inversion of the covariance matrix K, or $\widetilde{K}$ when noise is present. In the following discussion, we will refer generically to the matrix K, but the same statements hold for $\widetilde{K}$.

The size of K is $N \times N$, where N is the number of measurements. When N is large, this operation might become expensive both in terms of CPU time and memory, especially because model selection (chapter 5) and acquisition function optimization (chapter 7) require evaluating K^{-1} several times. In this section, we will discuss some common strategy to efficiently implement this matrix inversion.

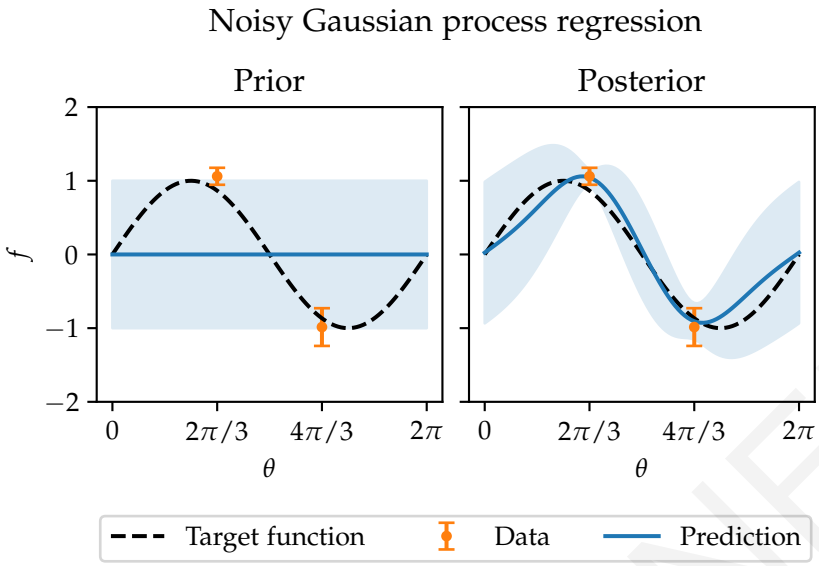


Figure 4.2: GPR of noisy data using the periodic kernel, $\mu = 0$, $\sigma^2 = 1$ and $\ell = 1$.

Direct methods

The matrix K is Hermitian positive-definite. The most straightforward approach is to directly invert K with any exact inversion algorithm such as Gauss-Jordan elimination, Cholesky decomposition or eigendecomposition. These methods have $\mathcal{O}(N^3)$ complexity, but an advantage is that once K^{-1} is known, it takes only $\mathcal{O}(N)$ flops to perform predictions evaluating $\mu(\theta|E_i, \xi)$. Evaluating the covariance function $k(\theta, \theta')$, however, needs $\mathcal{O}(N^2)$ flops.

Apart from the long evaluation time, the inversion of K might run into numerical problems as K is prone to be an *ill-conditioned* matrix [57]. A high condition number might imply that the evaluation of equations (4.6) and (4.12) requires summing numbers of very different orders of magnitude, which is likely to cause numerical inaccuracies.

A possible regularization method to avoid numerical instabilities is to use the truncated eigendecomposition inversion method [74], which is slower than other methods, but more stable in critical situations.

After decomposing $K = Q\Lambda Q^\top$, the inverse matrix is computed as $K^{-1} = Q\Lambda^{-1}Q^\top$. Some eigenvalues in the diagonal matrix Λ might be very small, if not slightly negative due to rounding errors. Therefore, K^{-1} might have very high eigenvalues, which are the main source of instability. The truncated eigendecomposition procedure removes this instability setting these problematic eigenvalues to zero, effectively removing the problematic eigenspaces. This is done by selecting a small threshold ε: if any element of Λ is smaller than ε, then the corresponding element of Λ^{-1} is set to zero. It might seem odd setting to zero values that otherwise would be very big. However, due to rounding errors, these very big values would actually badly reduce the numerical accuracy of the computation, so just excluding them produces a better result.

Indirect methods

Indirect methods avoid the explicit computation of K^{-1} by reformulating equation (4.12). Indeed, explicit inversion can be replaced by finding solution of linear systems, which still have $\mathcal{O}(N^3)$ computational cost, but they are usually numerically more stable.

Since K is Hermitian and positive definite, Cholesky decomposition is usually the recommended method to solve the system. However, in some cases it might fail because rounding errors might make K losing its positive definiteness.

A common simple solution to alleviate this problem is to add a small positive *jitter* to the diagonal of K before inversion [50]. This, not only might prevent the failure of Cholesky decomposition, but it also reduces the condition number, improving the overall numerical stability accuracy or the result, at the cost of slightly modifying the problem. In the case of noisy measurements, this might not be necessary as the Gaussian noise variances are already added to the diagonal, while in the case of noiseless measurements it is strongly recommended. If adding the jitter is not enough and Cholesky decomposition still fails, it is possible to use the more general LU decomposition.

Defining $y_i \equiv E_i - \mu$, $\kappa_i \equiv k(\theta_{i\alpha}, \theta_\alpha)$ and $\kappa'_i \equiv k(\theta_{i\alpha})$, equation (4.12) can be rewritten as:

$$\begin{cases} \mu(\theta_\alpha|E_i) = \mu(\theta_\alpha) + \kappa^\top K^{-1} y \\ k(\theta_\alpha, \theta'_\alpha|E_i) = k(\theta_\alpha, \theta'_\alpha) - \kappa^\top K^{-1} \kappa' \end{cases} \tag{4.13}$$

The indirect method requires the Cholesky decomposition $K = LL^\top$ or the LU decomposition $K = LU$, where L indicate a lower triangular matrix and U a upper triangular matrix.

After performing LU decomposition, equation (4.13) can be rewritten in terms of L and U as follows:

$$\begin{cases} \mu(\theta_\alpha|E_i) = \mu(\theta_\alpha) + \kappa^\top U^{-1} L^{-1} y \\ k(\theta_\alpha, \theta'_\alpha|E_i) = k(\theta_\alpha, \theta'_\alpha) - \kappa^\top U^{-1} L^{-1} y \kappa' \end{cases} \tag{4.14}$$

Analogous equations are valid if Cholesky decomposition used. The only difference is the replacement of U with $L^\top$.

Denoting as $A\backslash b$ the solution x of a linear system $Ax = b$, matrix-vector multiplications involving A^{-1} could be written as:

$$\begin{cases} A^{-1} b = A\backslash b \\ b^\top A^{-1} = \left(\left(A^\top \right)^{-1} b \right)^\top = \left(A^\top \backslash b \right)^\top \end{cases} \tag{4.15}$$

Using identities (4.15), the scalar products in equations (4.14) could be evaluated with two different ordering. If the multiplication is always going from right to left:

$$\begin{cases} \mu(\theta_\alpha|E_i) = \mu(\theta_\alpha) + \kappa^\top (U\backslash(L\backslash y)) \\ k(\theta_\alpha, \theta'_\alpha|E_i) = k(\theta_\alpha, \theta'_\alpha) - \kappa^\top (U\backslash(L\backslash \kappa')) \end{cases} \tag{4.16}$$

Otherwise, it could be performed both from left and right:

$$\begin{cases} \mu(\theta_\alpha|E_i) = \mu(\theta_\alpha) + \left(U^\top \backslash \kappa\right)^\top (L \backslash y) \\ k(\theta_\alpha, \theta'_\alpha|E_i) = k(\theta_\alpha, \theta'_\alpha) - \left(U^\top \backslash \kappa\right)^\top (L \backslash \kappa') \end{cases} \tag{4.17}$$

The advantage of the ordering 4.16 is that $K^{-1}y = U\backslash(L\backslash y)$ could be reused for any other set of predictions. On the other hand, it was shown in [57] that the ordering 4.17 is more stable.

Approximate methods

When the number of measurements N becomes very large, an inversion method, direct or indirect, whose computational cost scales as $\mathcal{O}(N^3)$ might become too expensive. In these cases, it is possible to use approximate methods to evaluate functions of the inverse matrix $\widetilde{K}^{-1}$ with a computational cost that scales better than $\mathcal{O}(N^3)$.

Many of these approximate methods are based on conjugate gradient [75] and Lanczos algorithm [76]. These methods avoid $\mathcal{O}(N^3)$ operations using instead T iterations of $\mathcal{O}(N^2)$ operations. Thus, the computational cost of conjugate gradient iterative methods scales as $\mathcal{O}(TN^2)$ Excluding rounding errors, there is the guarantee of finding the exact solution in $T = N$ iterations. However, when N is big enough, an approximate solution can be found with $T \ll N$.

Gibbs and MacKay [57] developed a conjugate gradient iterative method specifically for GPR based on the ideas of Skilling [77]. Their study shows that this approximate method starts outperforming direct and indirect methods when N is $\mathcal{O}(100)$. Thus, iterative methods should be considered for the regression of hundreds of data or for Bayesian optimization with hundreds of iterations.

5

BAYESIAN MODEL SELECTION IN GPR

The only missing piece of GPR is the choice of the hyperparameters ξ, which, in our case, are the prior mean μ, the sample variance σ^2 and the characteristic length-scale ℓ. This choice is of crucial importance as it determines the quality of the predictions obtained with GPR. In section 2.3, we saw the impact of the hyperparameters on the prior GP, and we can now discuss their impact on the predictive posterior GP.

5.1 CHANGING THE HYPERPARAMETERS

Prior mean

The prior mean is the value where the predictions converge in absence of other information, i.e. at points whose distance from the measured data is far more than ℓ in the θ_α space. This happens because, at these distances, the values of the energy are considered uncorrelated, therefore the prediction cannot benefit from data information.

In figure 5.1, it is shown an example of noisy GPR using different values of the prior mean μ keeping fixed $\sigma = \ell = 1$. We can see that, with extreme values of μ, the predictions tend to shift from the data to get closer to μ.

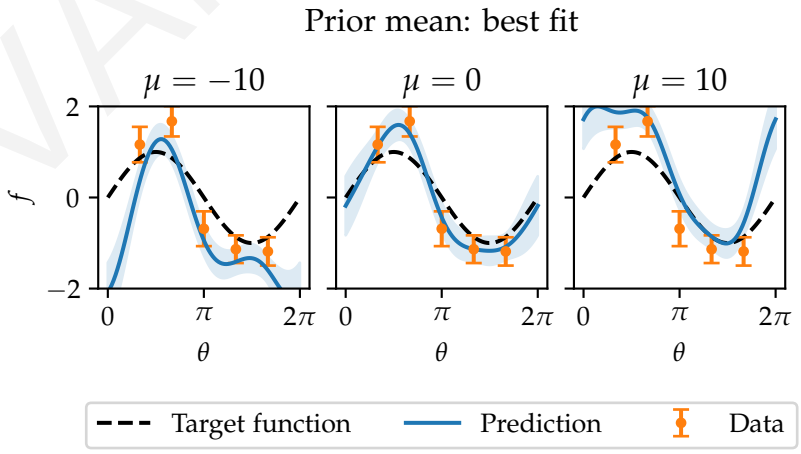


Figure 5.1: Noisy GPR using different values of the prior mean. The periodic kernel was used with $\sigma = \ell = 1$.

Sample variance

The sample variance σ^2 determines the wideness of the expected energy oscillations. A value of σ that is far lower than the distance between μ and the data assigns a very low prior probability to samples that move away from the prior mean to fit the data. Therefore, GPR would underfit the data producing predictions flattened around the prior mean μ. On the other hand, high values of σ are more permissive and don't penalize samples that overfit the data. The predictions would then perfectly interpolate the data mean values without filtering out the noise or possible outliers.

An example of this behaviour is shown in figure 5.2. The hyperparameters $\mu = 0$ and $\ell = 1$ are kept fixed and noisy GPR is performed with three different values of σ. With $\sigma = 0.1$ the data is too far from the prior mean compared to the credible interval radius σ, therefore, the narrow prior distribution flattens the predictions around itself, underfitting the data. The opposite happens with $\sigma = 10$: the wide prior distribution allows the samples to overfit the data without filtering out the noise. The case of $\sigma = 1$ seems instead to produce predictions that are balanced between data fitting and noise filtering.

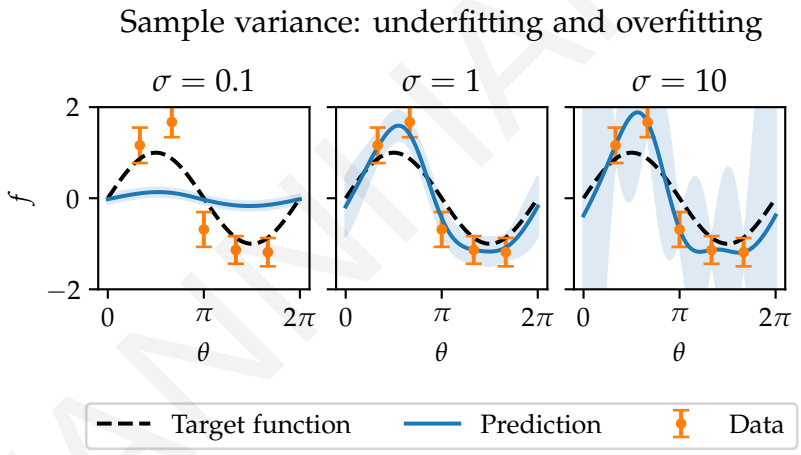


Figure 5.2: Noisy GPR using different values of the sample variance. The periodic kernel was used with $\mu = 0$ and $\ell = 1$.

Characteristic length-scale

A very similar situation arises varying the characteristic length-scale ℓ. Setting a certain value of ℓ penalizes samples whose oscillations have a length-scale lower than ℓ. If ℓ is set correctly, the samples that change more quickly than the target function are then filtered out. Samples that pass exactly through all the data are unlikely according to the prior as they would require to have quick variations in order to pass through all the mean values.

However, if ℓ is too large, the samples that vary with the correct rate are filtered out and GPR underfits the data, selecting only the samples that are equidistant to the data points, without explaining them. The opposite happens when ℓ is too small. The samples that vary quickly are not penalized, so GPR selects the samples that overfit the data points passing through all of them, without regressing the actual shape of the underlying function.

An example of these three circumstances is shown in figure 5.3. The hyperparameters $\mu = 0$ and $\sigma = 1$ are kept fixed and noisy GPR is performed with three different values of ℓ. With $\ell = 10$ the effect is similar to what we obtained with $\sigma = 0.1$: the predictions are flattened around the prior because samples that pass through the data are strongly penalized. With $\ell = 0.1$ the situation is really pathological: not only the predictions overfit the data passing though all the data, but also quickly converge to the prior mean in the space separating the data points. This happens because the predictions became uncorrelated at a very short range because ℓ is small and the information acquired from the data is already lost in a small $\mathcal{O}(\ell)$ radius. The case of $\ell = 1$ is instead producing predictions close to the target function.

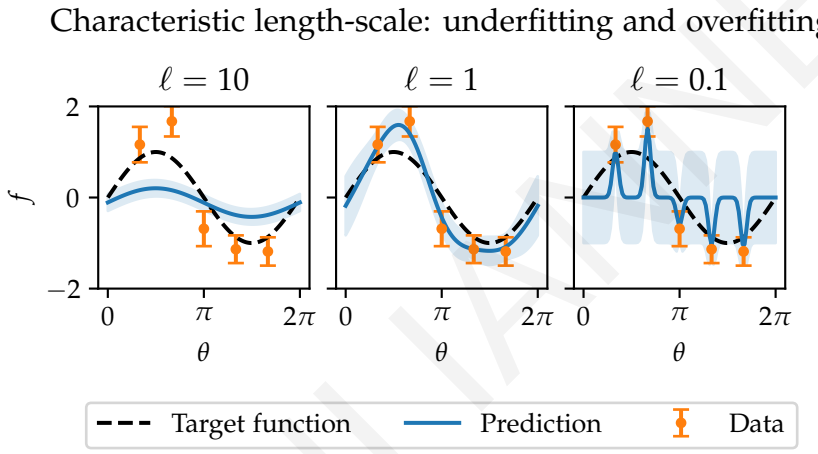


Figure 5.3: Noisy GPR using different values of the characteristic length-scale. The periodic kernel was used with $\mu = 0$ and $\sigma = 1$.

5.2 MAXIMUM LIKELIHOOD ESTIMATION OF TYPE II FOR GPR

The objective of model selection procedures is to automate the hyperparameters selection that we manually performed in the previous section. As we saw in section 3.4, a simple, but not ineffective, model selection procedure is to select the values of ξ that maximize the marginal likelihood $p(E_i|\xi)$. For computational simplicity, rather than the marginal likelihood, it is usually maximized the marginal log-likelihood:

$$\xi^{\text{MLE}} \equiv \arg\max_{\xi} \mathcal{L}(\xi) \equiv \arg\max_{\xi} \log p(E_i|\xi) \tag{5.1}$$

In presence of Gaussian noise, we have shown that the marginal likelihood is the multivariate Gaussian (4.9). Thus:

$$\mathcal{L}(\xi) = -\frac{1}{2}\left(y^\top \widetilde{K}^{-1} y + \log\det \widetilde{K} + d\log 2\pi\right) \tag{5.2}$$

where $y_i \equiv E_i - \mu(\theta_{i\alpha}|\xi)$ and $\widetilde{K}_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha}|\xi) + \Delta E_i^2 \delta_{ij}$. It is important to notice that the constant depending on $\widetilde{K}$ should be included in the optimization as it depends on σ and ℓ.

The marginal likelihood may have more than one local maximum. However, practical experience [16] with simple covariance functions seems to indicate that local maxima are not very problematic as they usually correspond to reasonable sets of hyperparameters. Thus, local optimizers could also be used to evaluate equation (5.1) if global optimization requires too much computing time.

Some optimizers may improve their performance if they are provided with the analytical formula of the gradient. Using the equations (A.4) and (A.5) of the appendix, the gradient of the marginal log-likelihood is:

$$\partial_{\xi}\mathcal{L} = \frac{1}{2}\left(y^{\top}\widetilde{K}^{-1}(\partial_{\xi}\widetilde{K})\widetilde{K}^{-1}y - \mathrm{tr}(\widetilde{K}^{-1}\partial_{\xi}\widetilde{K})\right) \tag{5.3}$$

where $\partial_{\xi}K$ is the matrix formed by the partial derivatives of the elements of K.

Let us analyze the examples illustrated in figures 5.1, 5.2 and 5.3. The prior mean shown in figure 5.1 is the easiest hyperparameter to fit as it shouldn't change model complexity. Indeed, it only determines a translation of the GP prior, without changing the normalization constant of sample distributions. MLE usually fixes μ without problems as there is not a tradeoff between good data fit and low model complexity.

In terms of MLE, the cases of figures 5.2 and 5.3 are quite similar. The marginal likelihood $p(E_i|\xi)$ is obtained integrating the product between the likelihood $p(E_i|f_i)$ and the prior $p(f_i|\xi)$. When underfitting hyperparameters like $\sigma = 0.1$ or $\ell = 10$ are used, if the likelihood $p(E_i|f_i)$ is high valued, then the prior $p(f_i|\xi)$ is low valued, and vice-versa. This happens because values of f_i close to E_i have a small prior probability and those close to μ have a small likelihood. Since the marginal likelihood is found integrating over f_i the product between the marginal likelihood and the prior distribution, MLE excludes these hyperparameters as the marginal likelihood is suboptimal.

On the other hand, when overfitting hyperparameters like $\sigma = 10$ or $\ell = 0.1$ are used, the prior has a very large normalization constant because it is compatible with a great variety of possible f_i values. Therefore the marginal likelihood is suboptimal even if the likelihood is very high, and MLE excludes these overfitting hyperparameters.

In figure 5.4 is shown the GPR performed with the hyperparameters obtained with MLE using the same data of figures 5.1, 5.2 and 5.3. The values found by MLE are $\mu \approx -0.059$, $\sigma \approx 1.2$, $\ell \approx 0.9$, which are similar to what we found heuristically in figures 5.1, 5.2 and 5.3. Using these hyperparameters, overfitting and underfitting were avoided and the resulting predictions are similar to the target sine function.

We would like now to come back to a topic left unresolved in section 2.3. The kernels introduced in chapter 2 could have a different characteristic length-scale ℓ_α for each coordinate direction. In many circumstances, target functions have some parameters that are more relevant than others, which means that changing some parameters makes the target function vary more then when changing less relevant ones.

A model selection technique such as MLE could then adapt each single ℓ_α to the target function. Therefore, the resulting ℓ_α identify the coordinates that are more or less relevant. In particular, the values of α corresponding to the lower valued ℓ_α represent the more

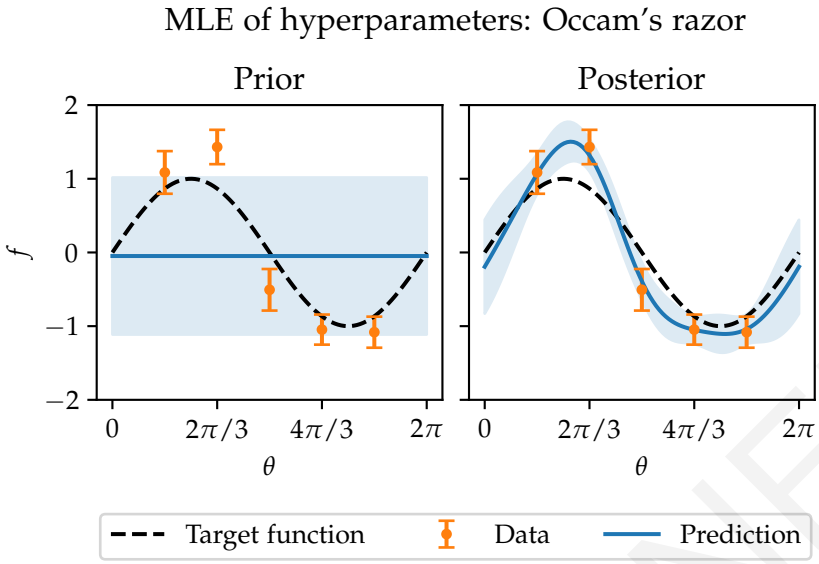


Figure 5.4: Noisy GPR performed with the hyperparameters $\mu \approx -0.059$, $\sigma \approx 1.2$, $\ell \approx 0.9$ that were found using MLE.

relevant coordinates. This is the reason why it is said that such kernels implement automatic relevance determination (ARD), or, for simplicity, these kernels are called ARD kernels.

Efficient implementation of MLE-II

In order to implement MLE-II, it is sufficient to maximize the marginal log-likelihood (5.2), possibly using its gradient (5.3). It is not necessary to perform predictions using equations (4.12). However, computing the marginal log-likelihood has difficulties similar to those found in computing predictions. Indeed, it also involves inverting the $N \times N$ matrix $\widetilde{K}$, where N is the number of measurements.

We reviewed in section 4.5 some direct, indirect and approximate methods to evaluate the product between $\widetilde{K}^{-1}$ and a vector. The same considerations hold for the terms involving the product $\widetilde{K}^{-1}y$ in equations (5.2) and (5.3).

The term $\log\det\widetilde{K}$ present in the marginal likelihood could be easily computed if $\widetilde{K}$ has already been decomposed for evaluating $\widetilde{K}^{-1}y$. Indeed, once it is known the Cholesky decomposition $\widetilde{K} = LL^\top$ or the eigendecomposition $\widetilde{K} = Q\Lambda Q^\top$, the determinant is given by:

$$\log\det\widetilde{K} = 2\sum_i \log L_{ii} = \sum_i \log \Lambda_{ii}$$

The trace term $\operatorname{tr}(\widetilde{K}^{-1}\partial_\xi\widetilde{K})$ in the log-likelihood gradient (5.3), requires the evaluation of the explicit inverse matrix $\widetilde{K}^{-1}$. However, it is not necessary to compute the full matrix-matrix product between $\widetilde{K}^{-1}$ and $\partial_\xi\widetilde{K}$ because:

$$\operatorname{tr}(\widetilde{K}^{-1}\partial_\xi\widetilde{K}) = \sum_{ij}\left(\widetilde{K}^{-1}\right)_{ij}\partial_\xi\widetilde{K}_{ij} \equiv \sum_{ij}\left(\widetilde{K}^{-1}\circ\partial_\xi\widetilde{K}\right)_{ij}$$

Where the symbol $\circ$ indicates the Hadamard product, which is simply the element-wise product of the two operand matrices.

If the number of measurements N is $\mathcal{O}(100)$ or greater, it might be more convenient to use approximate methods to compute the marginal log-likelihood and its gradient. Indeed, with iterative procedures it is possible to avoid operations whose computational cost scales as $\mathcal{O}(N^3)$.

We already mentioned in section 4.5 that conjugate gradient iterative methods can be used to solve linear systems with a computational cost that scales as $\mathcal{O}(TN^2)$, where $T < N$ is the number of iterations. These methods can then be used to evaluate $\widetilde{K}^{-1}y$, which appears both in the marginal log-likelihood and in its gradient.

There are still two terms with $\mathcal{O}(N^3)$ complexity: the trace in (5.3) and the determinant in (5.2). In [77] and [57] are discussed Monte Carlo methods to evaluate these determinant and trace without using operations with $\mathcal{O}(N^3)$ complexity.

5.3 MAXIMUM A POSTERIORI AND REGULARIZATION

Unfortunately, we noticed that in some cases MLE selects overfitting values of ℓ. This happens when the data is spread apart at distances higher than the correct ℓ, which is a common circumstance with a low number of data points or in a high number of dimensions.

An example of this behaviour is shown in figure 5.5. With only two distant data points, MLE selects the hyperparameters $\mu \approx 0.063$, $\sigma \approx 1$, $\ell \approx 0.3$. Such a small value of ℓ leads to overfitting predictions similar to those shown in figure 5.3.

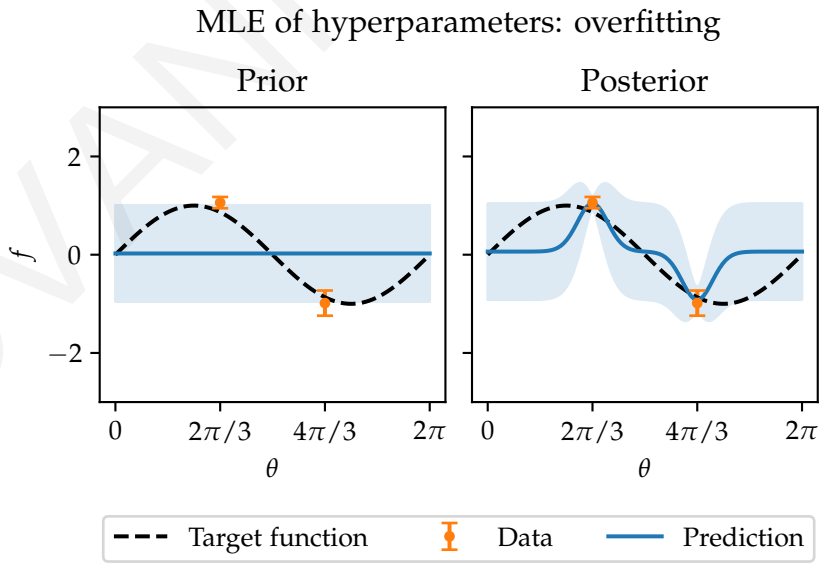


Figure 5.5: GPR of noisy data that is spread apart using the hyperparameters $\mu \approx 0.063$, $\sigma \approx 1$, $\ell \approx 0.3$. MLE selected an overfitting ℓ.

The reason why it happens is quite subtle. If ℓ is smaller than the point distances in the θ_α space, then the parameters f_i are effectively uncorrelated according to their prior distribution. Indeed, using the covariance functions introduced in chapter 2, the diagonal

terms of the covariance matrix decay to zero with exponential law or faster. The prior distribution of f_i is then approximately the same for all values of ℓ significantly smaller than the points distance. This implies that the normalizing constant of the prior is not increasing enough to flatten the prior associated with more complex models. Thus, the Occam's razor effect illustrated in figure 3.1 is not observed in this circumstance.

Selecting values of ℓ that are too small has very problematic implications. Not only the predictions might become meaningless as in the right panel of figure 5.3, but also the whole computation of GPR becomes numerically unstable as it implies sums between numbers of very different order of magnitude. This in particular might be problematic for evaluating the inverse matrices present in (4.12) and (5.2).

In machine learning is called *regularization* the practice of adding information in order to avoid ill-posed problems and overfitting. Avoiding numerical instability caused by extreme values of the hyperparameters could be easily solved imposing boundaries on them during the optimization procedure (5.1). This however doesn't prevent overfitting.

Fortunately, Bayesian inference offers an elegant way to regularize hyperparameters, which is fixing them with a maximum a posteriori (MAP) procedure after the definition of a hyperprior distribution $p(\xi)$. This has been shown to be a very effective and flexible way to facilitate the hyperparameters to have reasonable properties based on prior beliefs [72].

The MAP estimation of the hyperparameters ξ given the hyperprior $p(\xi)$ is obtained with the following optimization:

$$\xi^{\text{MAP}} \equiv \arg\max p(E_i|\xi)p(\xi) \tag{5.4}$$

where $p(E_i|\xi)$ is the marginal likelihood (4.9).

We used the regularization implemented in the BoTorch library [78], which proven to be reliable during Bayesian optimization tests performed by BoTorch developers. It is composed of two stages: the data is first rescaled and normalized, and then some generic, empirically determined hyperpriors are placed on ξ. It is desirable to have a hyperprior that is the most generic possible, so that it could be applied to a wide range of different circumstances. In order to achieve this, a common practice is to define a procedure to map the data points into a standardized space, so that the hyperparameters have similar order of magnitudes in many different problems.

The regularization procedure we used rescales the $\theta_{i\alpha}$ parameters from $[0, 2\pi)^d$ to $[0, 1)$ and normalizes the measured energies E_i using their mean and standard deviation:

$$\begin{cases} \theta_{i\alpha} \mapsto \theta_{i\alpha}/2\pi \\ E_i \mapsto (E_i - \text{mean}(E_i))/\text{std}(E_i) \\ \Delta E_i \mapsto \Delta E_i/\text{std}(E_i) \end{cases}$$

The hyperpriors used in BoTorch are expressed in terms of the gamma distribution, which is a continuous probability distribution $\Gamma(x|k, \theta)$ identified by its shape α and its rate β:

$$\Gamma(x|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x}$$

where $\Gamma(\alpha) \equiv \int_0^\infty t^{\alpha-1}e^{-t}dt$ is the gamma function. As an alternative, it is possible to define the hyperpriors in terms of the inverse gamma distribution [50].

Some examples of gamma distribution probability density functions are shown in figure 5.6. It is a common choice for priors of positive variables as it is easily customizable to many phenomena tuning α and β. Furthermore, gamma distributions satisfy Cromwell's rule as they are non-zero for all positive domain points and they are conjugate priors to many likelihoods as, for example, the Poisson and exponential distributions. The gamma distribution also generalizes the $\chi^2(\nu)$ distribution with ν degrees of freedom. Indeed $\chi^2(\nu) = \Gamma(\nu/2, 2)$. The expected value and the variance of the gamma distribution are easily expressed in terms of its shape and rate:

$$\begin{cases} \mathbb{E}[x] = \dfrac{\alpha}{\beta} \\ \mathrm{Var}[x] = \dfrac{\alpha}{\beta^2} \end{cases} \tag{5.5}$$

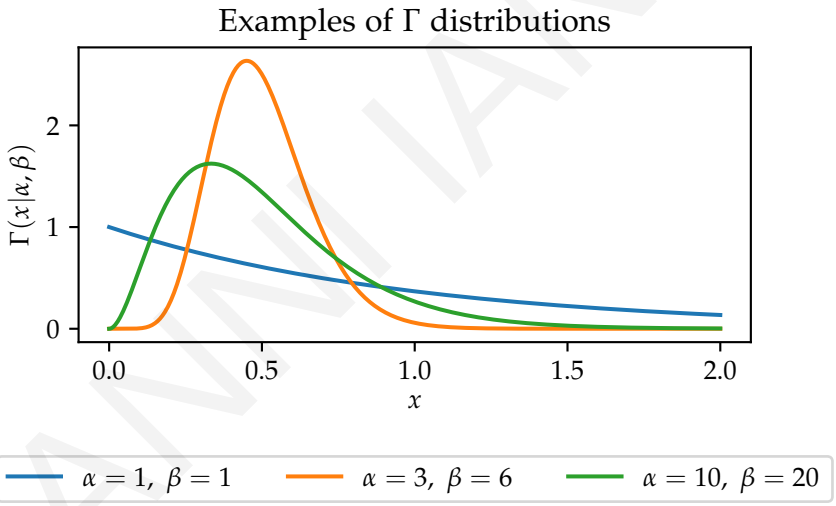


Figure 5.6: Some examples of gamma distributions obtained varying the shape α and the rate β.

The prior mean μ is usually not problematic to fix, therefore placing a hyperprior on it is not necessary if the MLE optimization starts from a reasonable value of μ. The characteristic length-scale ℓ is the most delicate hyperparameter as it cause severe overfitting and instability when is too low, but also underfitting when is too high. Since each circuit parameter θ_α has been constrained in the interval $[0, 1)$, we want to avoid very small values $\ell \lesssim 10^{-3}$ as well as high values $\ell \gtrsim 2$. Thus, a possible choice for the rescaled characteristic length-scale hyperprior is:

$$p(\ell) = \Gamma(\ell|3, 6) \tag{5.6}$$

Indeed, its mean value is $3/6 = 1/2$, its standard deviation is $\sqrt{3}/6 \approx 0.29$ and it quickly converges at 0 for $\ell \to 0$ as shown in the left panel of figure 5.7.

The value of the sample variance σ^2 is not as problematic as ℓ, but it is still useful to regularize it in order to make the optimization more stable. Furthermore, in the context

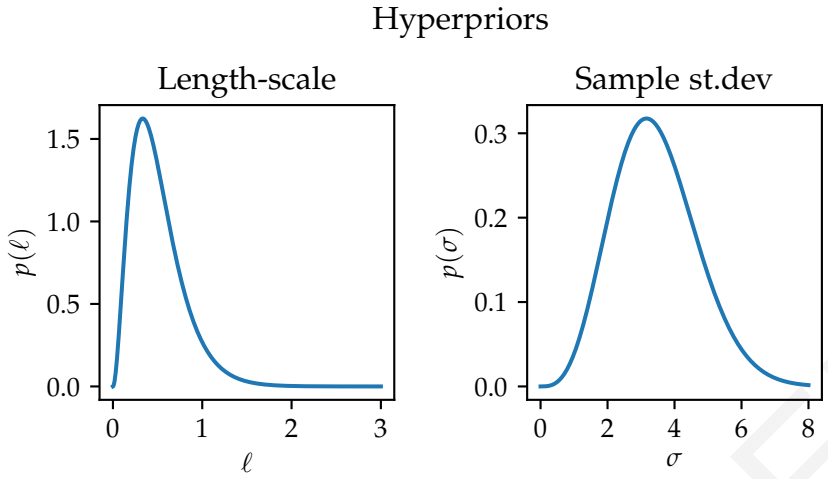


Figure 5.7: In the left panel is shown the gamma distribution used as the hyperprior on the characteristic length-scale, while in the right panel is plotted the Nakagami distribution used as the hyperprior on the sample standard deviation.

of Bayesian optimization, we will see in chapter 7 that σ^2 has a significant impact on the degree of exploration of the global optimization procedure.

The chosen hyperprior for the normalized sample variance is:

$$p(\sigma^2) = \Gamma(\sigma^2|2, 0.15) \tag{5.7}$$

Placing a prior on σ^2 rather than of σ has a small computational advantage. However, in order to get an intuition about the properties of this hyperprior, it is useful to observe the probability density function of the σs obtained evaluating the square root of the variances σ^2 sampled from the hyperprior. The sample standard deviation σ obtained in this way follows the Nakagami distribution shown in the right panel of figure 5.7. Its mean is ≈ 3.4 and its standard deviation is ≈ 1.25. Since it is applied on the normalized data, a value of $\sigma = 3.4$ is quite high. The purpose of this choice is to have a large credible interval in areas of the domain that are far from the measured points. We will see in chapter 7 that this larger credible interval translates to a more explorative optimization algorithm.

Finally, we can test this MAP procedure on the data in figure 5.5 that where overfitted by MLE. In figure 5.8 are shown the results obtained with the MAP hyperparameters $\mu \approx 0.05$, $\sigma \approx 2.6$, $\ell \approx 2$ and the hyperpriors (5.6), (5.7) used with the rescaled and normalized data. In this case, the predictions are very close to the underlying function, the data is not overfit and the predictive model assigns cautious credible interval in unknown regions of the domain.

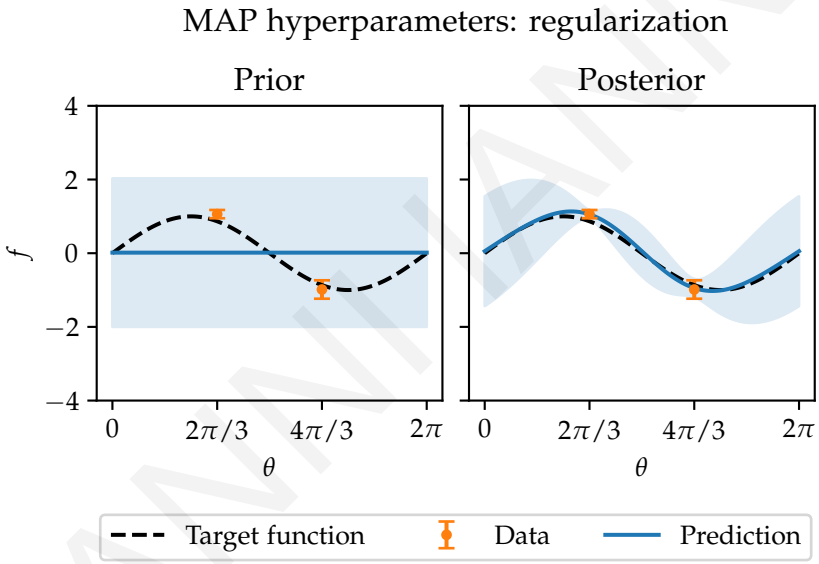


Figure 5.8: GPR performed on noisy data that is spread apart with the hyperparameters $\mu \approx 0.05$, $\sigma \approx 2.6$, $\ell \approx 2$ that were found using MAP. The regression is succesful in this problematic case.

6

BAYES-GAUSS INTEGRAL TRANSFORMS

There is a wide range of possible applications for the modelling of an unknown function with GPR. We have shown that a key property of GPR is its capability of using noisy data to infer unknown functions with a required level of smoothness.

Once the underlying function generating the observable data is inferred, most applications involve computing other quantities using the inferred function. For example, in Bayesian optimization, the inferred surrogate model is used to find the position of the next measurement using Bayesian decision theory, as will be explained in chapter 7.

An interesting algorithm based on GPR is the Bayes-Hermite quadrature [79], which defines a procedure to evaluate definite integrals of unknown functions inferred with GPR. The properties of GPs are used to define quadrature rules in analogy with the Gauss-Hermite quadrature rules [55].

Another similar usage of GPR involves the evaluation of integral transforms of the underlying function. We call this procedure Bayes-Gauss integral transform (BGIT). This family of problems is not relevant for VQE, but is recurrent in scientific and engineering applications [80]. The most common usage of integral transforms is for the solution of differential equations, which are often easier to solve in the image space of the integral transform rather than in the original space of the observable function.

As in the case of Bayesian optimization, methods that are based on GPR become convenient over other approaches when the shape of the function is unknown and the available data is limited and affected by statistical errors. Indeed, in these circumstances, it is problematic to fit the target function in terms of known functions, or to rely on a high number of measurements to reconstruct the target function profile.

Furthermore, due to the fact that GPR produces results in a simple closed form, many further manipulations of the inference results can be performed analytically. This is very helpful to improve the speed and stability of the algorithm.

A common example of integral transforms is the Fourier transform, and the estimation of them using BGIT has been called Bayes-Gauss-Fourier transforms (BGFT) [17, 81]. In particular, in [17], this method was used to evaluate quasi parton distribution functions using lattice quantum chromodynamics data, obtaining results that are more stable and precise compared to those achieved using methods based on discrete-time Fourier transforms.

In this chapter, after deriving the generic formulae for evaluating BGITs, we will describe an example problem, in which the Fourier transform of an even function needs to be reconstructed from noisy data. In this context, we will introduce the method that is commonly used for these cases, which makes usage of discrete-time Fourier transform

(DTFT). The same problem is then tackled using BGFT and the results are compared with those obtained with DTFT.

6.1 ESTIMATING A GENERIC INTEGRAL TRANSFORM

A generic integral transform $\mathcal{T}$ can be written as:

$$\mathcal{T}[x(t)](u) = \int_{t_a}^{t_b} x(t)\mathcal{A}(t,u)\mathrm{d}t$$

where $x(t)$ is the function that needs to be transformed and $\mathcal{A}(t,u)$ is the *kernel* (or *nucleus*). An integral transform is identified by the kernel and the integration interval (t_a, t_b). For example, the Fourier transform of L^2 functions has $\mathcal{A} = e^{-2\pi iut}$, $t_a = -\infty$ and $t_b = +\infty$.

Inference of $x(t)$

If we can access the function $x(t)$ only through noisy measurements, we first select a set of domain points $t_1, ..., t_N$ in which we can measure the values $x(t_i)$ obtaining $x_i \pm \Delta x_i$, where the errors Δx_i are assumed to be Gaussian.

In section 4.4 we have shown that applying GPR to noisy data yields a posterior GP, whose mean and covariance are given by (4.12). In this case, they are given by:

$$\begin{cases} \mu(t|x_i) = \mu(t) + \sum_{ij} k(t,t_i)\left(\widetilde{K}^{-1}\right)_{ij}(x_j - \mu(t_j)) \\ k(t,t'|x_i) = k(t,t') - \sum_{ij} k(t,t_i)\left(\widetilde{K}^{-1}\right)_{ij} k(t_j,t') \end{cases} \tag{6.1}$$

where $\mu(t)$ is the prior mean, $k(t,t')$ the prior covariance and $\widetilde{K}_{ij} \equiv k(t_i,t_j) + \Delta x_i^2 \delta_{ij}$.

Integral transform estimator

As we explained in chapter 2, we can sample functions from a GP. Since the posterior GP is our surrogate target function, we can interpret the sample functions as possible profiles of the target function. Our objective is to evaluate an integral transform of the target function. Therefore, our estimation can be built generating samples from the posterior GP, applying the integral transform on them, and then taking the mean value of the transformed samples as our estimator.

In section 2.2 we saw a procedure for evaluating samples in a finite set of domain points. Calling t_i^s a set of S values of the variable t that constitute a grid of the domain, samples $x^s(t)$ of the posterior GP can be evaluated in these S points in the following way:

$$x^s(t_i^s) = \mu(t_i^s|x_i) + \sum_j L_{ij}^s n_j \tag{6.2}$$

where L^s is the Cholesky decomposition of $K_{ij}^s \equiv k(t_i^s, t_j^s|x_i)$ and n_j is a set of S samples of the unit Gaussian distribution.

The integral transform of the samples could be approximated using discretized version of the integral transform (e.g. discrete Fourier transform) on the finite set (6.2). However, if possible, it is advisable to evaluate sample integral transforms analytically in order to have a more stable and precise algorithm. This requires a closed form expression of the samples $x^s(t_i^s)$ defined in all the integration interval $t \in (t_a, t_b)$. Unfortunately, the extension of equation (6.2) for a continuous real variable t is not trivial as the second term involves a discrete matrix vector multiplication.

Approximation: samples interpolation

A possible solution of this problem is achieved using the very same idea of this whole procedure, which is to use GPR to interpolate a set of data and using the outcome to evaluate the integral transform. Indeed, the discrete samples x_i^s obtained from the GP posterior can be interpolated with another GPR, and the resulting new posterior mean can be used to evaluate the integral transform.

The key difference between the original and this second GPR is that here the input data x_i^s is noiseless and available at any desired density just increasing the number of points in the grid t_i^s. Furthermore, as will be shown later, the GPRs of the samples don't actually need to be performed in practice as the distribution of their outcome will be encapsulated in the final formula. Therefore, this passage should be just considered as an approximation of the final integral transform on the grid t_i^s: in the limit of an infinite grid that covers the whole domain, the posterior mean of the second GPR would be identical to the continuous samples, and it could be used to evaluate their exact integral transform.

Let us show, in practice, how this approximation works. Before proceeding with the GPR of the discrete sample x_i^s, we need to specify the prior mean and covariance. It is reasonable to use the posterior mean and covariance of the first GPR (6.1) as the prior mean and covariance of this second GPR. This means that, in areas far from the grid points t_i^s, the interpolated samples will converge to the posterior mean $\mu(t|x_i)$, which is reasonable as the samples themselves are coming from the posterior distribution. Another advantage of the choice of this prior is that it doesn't require fixing the hyperparameters a second time as they were fixed in the first GPR.

Calling $x_{\text{int}}^s(t)$ the sample interpolations that approximate $x(t)$, their values are given by the posterior mean of the second GPRs, which is, using (4.6):

$$x^s(t) \simeq x_{\text{int}}^s(t) = \mu(t|x_i) + \sum_{jk} k(t, t_j^s|x_i) \left((K^s)^{-1}\right)_{jk} (x^s(t_k^s) - \mu(t_k^s|x_i))$$

where $K_{ij}^s \equiv k(t_i^s, t_j^s|x_i)$. This equation can be simplified using (6.2):

$$x_{\text{int}}^s(t) = \mu(t|x_i) + \sum_{jkl} k(t, t_j^s|x_i) \left((K^s)^{-1}\right)_{jk} L_{kl}^s n_l$$

and finally, using $K^s = L^s(L^s)^\top$, we have:

$$x^s_{\text{int}}(t) = \mu(t|x_i) + \sum_{jk} k(t, t^s_j|x_i) \left((L^s)^{\top -1} \right)_{ij} n_j \tag{6.3}$$

Integral transform of the interpolated samples

Observing (6.3), we notice that the interpolated samples are a linear combination of the functions $\mu(t|x_i)$ and $k(t, t^s_i|x_i)$. Thus, the linearity of the integral transforms leads us to a simple formula for the transform of the interpolated samples:

$$\mathcal{T}[x^s_{\text{int}}(t)](u) = \mathcal{T}[\mu(t|x_i)](u) + \sum_{jk} \mathcal{T}[k(t, t^s_j|x_i)](u) \left((L^s)^{\top -1} \right)_{jk} n_k \tag{6.4}$$

In order to better understand the distribution of the transformed interpolated samples (6.4), let us consider a set u^*_i of M values of the transformed variable u. Then, using equation (6.4), we can evaluate the transformed interpolated samples in those points:

$$\mathcal{T}[x^s_{\text{int}}(t)](u^*_i) = \mathcal{T}[\mu(t|x_i)](u^*_i) + \left(k^{\mathcal{T}} \right)^\top (L^s)^{\top -1} n \tag{6.5}$$

where $k^{\mathcal{T}}_{ij} \equiv \mathcal{T}[k(t, t^s_i|x_k)](u^*_j)$ is a $K \times M$ matrix. We can see that equation (6.5) is just an affine transformation of the unit Gaussian vector n_i. Therefore, using the equation (A.1) of the appendix, the values of the interpolated samples (6.5) are distributed as the following multivariate Gaussian:

$$\begin{aligned} \mathcal{T}[x^s_{\text{int}}(t)](u^*_i) &\xrightarrow[\text{as}]{\text{distributed}} \mathcal{N}\left(\mathcal{T}[\mu(t|x_i)](u^*_i), {k^{\mathcal{T}}}^\top (L^s)^{-1\top} (L^s)^{-1} k^{\mathcal{T}} \right) \\ &= \mathcal{N}\left(\mathcal{T}[\mu(t|x_i)](u^*_i), {k^{\mathcal{T}}}^\top (K^s)^{-1} k^{\mathcal{T}} \right) \end{aligned} \tag{6.6}$$

In most cases, we are just interested in the prediction errors rather than in the full covariance matrix. The errors are the square roots of the covariance matrix diagonal elements, and they can be obtained without having to compute the covariance matrix ${k^{\mathcal{T}}}^\top (K^s)^{-1} k^{\mathcal{T}}$. For this purpose, we can use the following: given a matrix A, the matrix product diagonal $(A^\top A)_{ii}$ can be simplified to $\sum_j A^2_{ji}$, which is constructed with the columns of the matrix A. Applying this property to the first line of equation (6.6), we find that the prediction variance is given by:

$$\text{Var}[\mathcal{T}[x^s_{\text{int}}(t)](u^*_i)] = \sum_{jk} \left(\left((L^s)^{-1} \right)_{jk} k^{\mathcal{T}}_{ki} \right)^2$$

which is easy to evaluate after having performed the Cholesky decomposition $K^s = L^s(L^s)^\top$. Indeed, $(L^s)^{-1} k^{\mathcal{T}}$ can be found solving the lower triangular linear system $L^s x = k^{\mathcal{T}}$.

Integral transform posterior predictive GP

Since equation (6.6) holds for any possible set u_i^*, the transformed interpolated samples constitute a new GP whose mean and covariance functions are:

$$\begin{cases} \mu_s^{\mathcal{T}}(u|x_i) = \mathcal{T}[\mu(t|x_i)](u) \\ k_s^{\mathcal{T}}(u,u'|x_i) = \sum_{jk} \mathcal{T}[k(t,t_j^s|x_i)](u) \left((K^s)^{-1}\right)_{jk} \mathcal{T}[k(t,t_k^s|x_i)](u') \end{cases} \tag{6.7}$$

The GP defined by equation (6.7) represents our posterior predictions about the target integral transform. In particular, its mean function $\mu_s^{\mathcal{T}}(u|x_i)$ estimates $\mathcal{T}[x(t)](u)$ and $\sqrt{k_s^{\mathcal{T}}(u,u|x_i)}$ the prediction errors. Equation (6.7) is therefore the final formula for the Bayes-Gauss integral transform.

In order to evaluate equation (6.7), we need to specify how to compute $\mathcal{T}[\mu(t|x_i)](u)$ and $\mathcal{T}[k(t,t'|x_i)](u)$, which are the transforms of the posterior mean and covariance[1]. Using the linearity in t of equation (6.1), the transformed posterior can be written in terms of the transforms of the prior mean and covariance:

$$\begin{cases} \mathcal{T}[\mu(t|x_i)](u) = \mathcal{T}[\mu(t)](u) + \sum_{ij} \mathcal{T}[k(t,t_i)](u) \left(\widetilde{K}^{-1}\right)_{ij} (x_j - \mu(t_j)) \\ \mathcal{T}[k(t,t'|x_i)](u) = \mathcal{T}[k(t,t')](u) - \sum_{ij} \mathcal{T}[k(t,t_i)](u) \left(\widetilde{K}^{-1}\right)_{ij} k(t_j,t') \end{cases} \tag{6.8}$$

If the prior mean and covariance are chosen such that their integral transform is available in closed form, then also the transformed posterior (6.7) will be.

The result (6.7) becomes an exact transformation of the posterior GP (6.1) when the discretization t_i^s is infinitely dense and covers the whole integration interval. In section 6.3 we will discuss how to choose the approximating grid t_i^s in the case of Fourier transforms.

Limit of infinite grid and approximate GPR

In some circumstances, the covariance $k_s^{\mathcal{T}}(u,u'|x_i)$ in equation (6.7) can be simplified, and the limit of x_i^s covering the whole domain can be evaluated in closed form. For example, in the case of the identity transform $\mathcal{T} = \mathrm{id}$, which means that no transformation is performed at all, we should be able to recover the exact starting formula (6.1) for the posterior predictions.

In order to show this, we first write down the transformed covariance function in equation (6.7) for the case $\mathcal{T} = \mathrm{id}$:

$$k_s^{\mathrm{id}}(t,t'|x_i) = \sum_{jk} k(t,t_j^s|x_i) \left((K^s)^{-1}\right)_{jk} k(t_k^s,t'|x_i) \tag{6.9}$$

In case t is in the set t_j^s, then the vector $k(t,t_j^s|x_i)$ is a row of K^s. Therefore, if j' is a number such that $t = t_{j'}^s$, the product $\sum_j k(t,t_j^s|x_i) \left((K^s)^{-1}\right)_{jk}$ becomes the Kronecker delta $\delta_{j'k}$. If t_j^s is an infinite grid, then such a j' must exist, and the above formula reduces to $k_s^{\mathrm{id}}(t,t'|x_i) = k(t,t'|x_i)$, which is the expected exact result.

1 In our notation, only the variable t, and not t', is transformed by $\mathcal{T}$.

In most cases, it is not possible to obtain the limit of infinite grid in closed form. However, in many applications, it might even be advantageous to transform the grid interpolation of the samples rather than the exact samples. For example, the exact samples might not be integrable, while their grid interpolation will be.

This is the case for the example shown in section 6.3, where using the approximate posterior covariance (6.9) instead of the exact one ensures that the samples are L^2 integrable, which is required to obtain their Fourier transform.

The integrability of the samples could also be ensured by choosing a different prior covariance function. However, if an approximation is required anyway, it could also be used to make the samples integrable without the need to modify the prior covariance.

6.2 FOURIER TRANSFORMS OF DISCRETE DATA

After having discussed the general case of an integral transform $\mathcal{T}$, we want to test our algorithm evaluating the Fourier Transform $\mathcal{F}[x(t)]$ of a function $x(t)$ accessible only through a set of measurements $x(t_1), ..., x(t_N)$.

We will use the following definition of Fourier transforms:

$$\begin{aligned} \mathcal{F}[x(t)](\omega) &= \int_{-\infty}^{+\infty} x(t) e^{-i\omega t} \mathrm{d}t \\ \mathcal{F}^{-1}[X(\omega)](t) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} X(\omega) e^{i\omega t} \mathrm{d}\omega \end{aligned} \tag{6.10}$$

The choice of testing the GPR based algorithm to compute Fourier transform is justified by its simplicity, but also by the possibility of comparing its results with well-known procedures such as the discrete-time Fourier transform (DTFT) algorithm.

In this section, we are going to describe the DTFT and the sinc interpolation techniques and apply them to a test problem. Then, in section 6.3, we are going to apply the GPR based algorithm on the same problem, so that the results can be compared.

Discrete-time Fourier transform (DTFT)

Letting $t_1, ..., t_N$ to be a set of consecutive integer multiples of an interval Δt and assuming the target function to be zero outside the sampling interval, the Fourier transform (6.10) can be trivially approximated with the midpoint rule:

$$\mathcal{F}[x(t)](\omega) \approx \Delta t \sum_i x_i e^{-i\omega t_i} \tag{6.11}$$

Observing this result, we notice that (6.11) is always $2\pi/\Delta t$-periodic in the frequency domain. Indeed $e^{-it_i(\omega + 2\pi/\Delta t)} = e^{-it_i\omega} e^{-2\pi i t_i/\Delta t} = e^{-it_i\omega}$, since $t_i/\Delta t$ is an integer number.

This result is not surprising as, according to Nyquist-Shannon theorem [82], only frequencies between $-\pi/\Delta t$ and $+\pi/\Delta t$ can be fully determined with a sampling rate of Δt. Since the results outside this interval don't provide any additional information, the approximate transform (6.11) can be set to zero outside $[-\pi/\Delta t, \pi/\Delta t]$. The resulting truncated approximate transform is also known as discrete-time Fourier transform (DTFT).

The restriction is usually achieved with the rectangular function, which is defined as:

$$\text{rect}(\omega) \equiv \begin{cases} 1, \text{ if } |\omega| < 1/2 \\ 0, \text{ otherwise} \end{cases}$$

Therefore, we define the DTFT as:

$$\text{DTFT}[x_i](\omega) = \Delta t \ \text{rect}\left(\frac{\omega \Delta t}{2\pi}\right) \sum_i x_i e^{-i\omega t_i} \tag{6.12}$$

Sinc interpolation

To better understand the DTFT approximation, it is useful to perform an inverse Fourier transform of equation (6.12) so that we can reconstruct the signal whose exact transform is equation (6.12). Since the DTFT restricts the ω spectrum within $\pm\pi/\Delta t$, this reconstruction can be seen as a rectangular low-pass filter of x_i.

Applying the inverse transform in (6.10) to the DTFT in (6.12):

$$\begin{aligned} \mathcal{F}^{-1}[\text{DTFT}[x_i](w)](t) &= \frac{\Delta t}{2\pi} \int_{-\pi/\Delta t}^{\pi/\Delta t} \sum_i x_i e^{i\omega(t-t_i)} \mathrm{d}\omega \\ &= \sum_i \frac{x_i \Delta t}{2\pi i (t-t_i)} \left[e^{\pi i (t-t_i)/\Delta t} - e^{-\pi i (t-t_i)/\Delta t} \right] \\ &= \sum_i x_i \frac{\sin(\pi(t-t_i)/\Delta t)}{\pi(t-t_i)/\Delta t} \\ &= \sum_i x_i \ \text{sinc}\left(\frac{t-t_i}{\Delta t}\right) \equiv \text{SINC}[x_i](t) \end{aligned} \tag{6.13}$$

where $\text{sinc}(t) \equiv \sin(\pi t)/\pi t$ is the normalized sinc function. $\text{SINC}[x_i](t)$ is called *sinc interpolation* or *Whittaker-Shannon interpolation* [82] of the data x_i. It is indeed an interploation procedure as $\text{SINC}[x_i](t)$ passes through all x_i. This property holds because $\text{sinc}((t_i - t_j)/\Delta t) = \delta_{ij}$.

Sinc interpolation is particularly useful if the bandwidth of sampled signal is limited by the Nyquist frequency $\pi/\Delta t$ as it leads to perfect reconstruction. If this condition is not met, as we mentioned before, sinc interpolation is equivalent to a brick-wall low-pass filter.

The case of even functions

We are going to test our algorithm in the case of real even functions $x(t)$ as the Fourier transform of such functions is guaranteed to be real valued and even. This property makes it easier to visualize and compare the results. The procedure can anyway be extended to the case of generic complex functions, for example by decomposing the complex function into two real functions as explained in [17].

Sampling an even $x(t)$ requires fewer measurements since the symmetry can be used to mirror them on the other side of the $t = 0$ axis. We will then consider only measurements at $t \geq 0$. In particular, the values of $t_0, t_1, ..., t_N$ will correspond to $t_n = n\Delta t$.

Using these definitions and properties, the formulae for sinc interpolation (6.13) and DTFT (6.12) can be rewritten and restricted to the case of even functions. For instance, the sinc interpolation becomes:

$$\mathrm{SINC}[x_i](t) = x_0 \,\mathrm{sinc}\left(\frac{t}{\Delta t}\right) + \sum_{i=1}^{N} x_i \left(\mathrm{sinc}\left(\frac{t-t_i}{\Delta t}\right) + \mathrm{sinc}\left(\frac{t+t_i}{\Delta t}\right)\right) \tag{6.14}$$

and the DTFT:

$$\mathrm{DTFT}[x_i](\omega) = \Delta t \,\mathrm{rect}\left(\frac{\omega \Delta t}{2\pi}\right)\left(x_0 + 2\sum_{i=1}^{N} x_i \cos(\omega t_i)\right) \tag{6.15}$$

Testing example

Let us define the following testing target function:

$$x(t) = e^{-|t|}\cos(t) \tag{6.16}$$

Which is an even function whose Fourier transform is available in closed form:

$$\mathcal{F}[x(t)](\omega) = 2(w^2+2)/(w^4+4)$$

The example target function (6.16) is sampled from $t = 0$ to $t = 4$ at a time interval of $\Delta t = 0.5$. The frequency cutoff induced by the DTFT is therefore $|\omega| < \pi/\Delta t = 2\pi$. The samples x_i obtained at times t_i can then be used to evaluate the sinc interpolation (6.14) and the DTFT (6.15). The results are shown in figure 6.1.

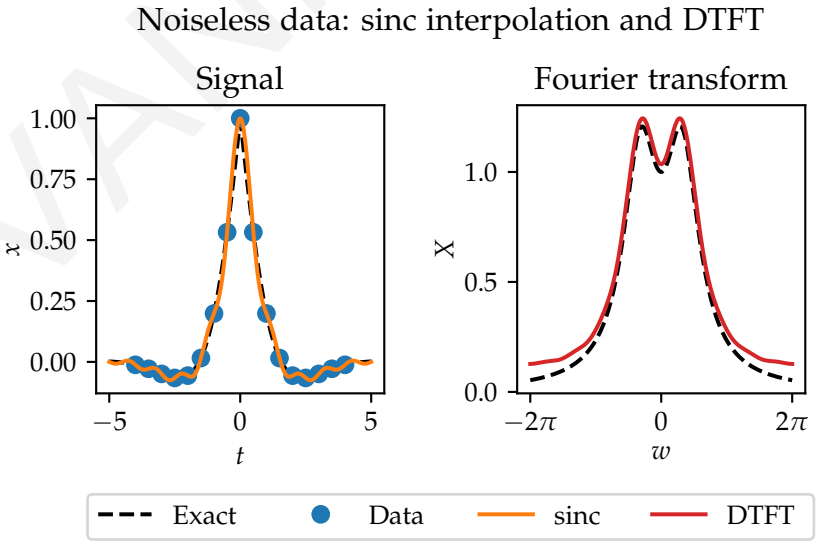


Figure 6.1: Sinc interpolation and DTFT of the noiseless testing data.

The reconstructions of the target function and of its Fourier transform look fairly accurate. The only noticeable discrepancy resides in the small oscillations at $|t| > 1$ in the sinc interpolation, which induce a deviation of the DTFT from the exact transform for values of $|\omega|$ approaching to the limit of 2π. This problem is arising because sinc interpolation

doesn't have a control on the level of smoothness of the interpolation, which could be tuned with GPR using a characteristic length-scale ℓ higher than the length of the unwanted oscillations.

Noisy sinc interpolation and DTFT

In many real world applications, measurements come with a statistical error, and, therefore, this procedure should be extended to take account of it. Let us introduce independent Gaussian errors with standard deviations Δx_i that afflict the measurements x_i. The sinc interpolation (6.14) and the DTFT (6.15) can be generalized considering the space of possible outcomes x_i^s drawn from $\mathcal{N}(x_i|\Delta x_i^2\delta_{ij})$. Each possible x_i^s can be plugged into (6.14) or (6.15), and the estimators of the noisy sinc interpolation and DTFT can then be defined as the mean value of $\mathrm{SINC}[x_i^s](t)$ and $\mathrm{DTFT}[x_i^s](\omega)$ among each possible x_i^s.

Due to the linearity in x_i of equations (6.14) and (6.15), we can use the equation (A.1) of the appendix to find the exact multivariate Gaussian of these estimators in terms of x_i and Δx_i. In particular, we are interested in their expectation values and variances. For instance, for the sinc interpolation we find:

$$\begin{cases} \mathbb{E}[\mathrm{SINC}[x_i^s](t)] = x_0\ \mathrm{sinc}\left(\frac{t}{\Delta t}\right) + \sum_{i=1}^N x_i \left(\mathrm{sinc}\left(\frac{t-t_i}{\Delta t}\right) + \mathrm{sinc}\left(\frac{t+t_i}{\Delta t}\right)\right) \\ \mathrm{Var}[\mathrm{SINC}[x_i^s](t)] = \Delta x_0^2\ \mathrm{sinc}^2\left(\frac{t}{\Delta t}\right) + \sum_{i=1}^N \Delta x_i^2 \left(\mathrm{sinc}\left(\frac{t-t_i}{\Delta t}\right) + \mathrm{sinc}\left(\frac{t+t_i}{\Delta t}\right)\right) \end{cases} \tag{6.17}$$

Here we notice that, due to the linearity, the expected value of the possible sinc interpolation is equal to the sinc interpolation of the mean values x_i. We can similarly find equivalent formulae for the DTFT:

$$\begin{cases} \mathbb{E}[\mathrm{DTFT}[x_i^s](\omega)] = \Delta t\ \mathrm{rect}\left(\frac{\omega\Delta t}{2\pi}\right) \left(x_0 + 2\sum_{i=1}^N x_i \cos(\omega t_i)\right) \\ \mathrm{Var}[\mathrm{DTFT}[x_i^s](\omega)] = \Delta t^2\ \mathrm{rect}\left(\frac{\omega\Delta t}{2\pi}\right) \left(\Delta x_0^2 + 4\sum_{i=1}^N \Delta x_i^2 \cos^2(\omega t_i)\right) \end{cases} \tag{6.18}$$

We tested formulae (6.17) and (6.18) on the same example function introduced in the previous section. In addition, a Gaussian smearing with standard deviation of $\Delta x_i = 0.04$ is applied to the measurements x_i. The results are shown in figure 6.2. We can see that here, when noise is present, the main weakness of the noiseless case shown in 6.1 is magnified. The noisy sinc interpolation oscillates even more at $|t| > 1$ as it interpolates all the noisy mean values x_i. This fact leads to a greater discrepancy between the noisy DTFT and the exact transform when $|w|$ approaches 2π.

What this procedure is most in need of is a way to infer the real values of the underlying function, rather than overfitting the data interpolating the noisy measurements. This problem can also be mitigated with GPR, since it finds a compromise between data fit and the geometrical properties of the prior.

Another defect of the sinc interpolation implemented here is the absence of an estimation of the modeling error. The error shown in figure 6.2 corresponds to the statistical error induced by the noisy measurements. Indeed, the noiseless case shown in 6.1 shows zero error even though a modeling error is clearly present.

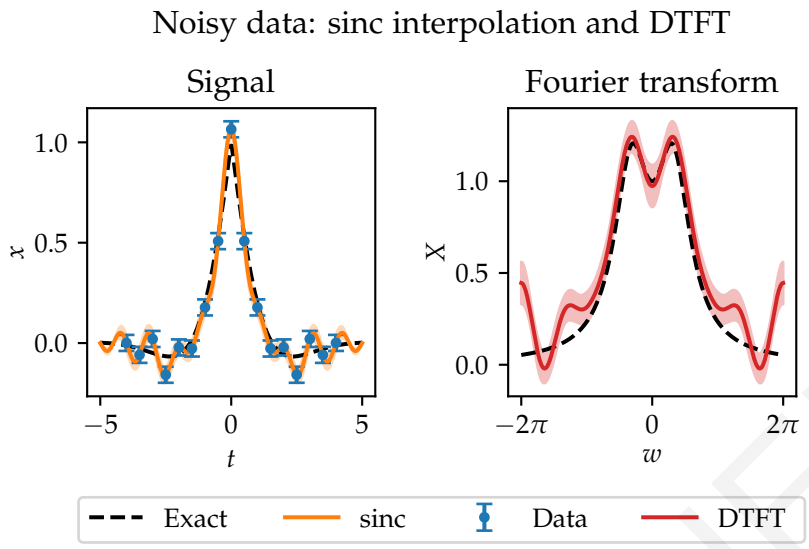


Figure 6.2: Sinc interpolation and DTFT of the noisy testing data.

There are however possible ways to estimate the modeling error of discrete Fourier transforms, see for example [83]. Regarding the BGIT, the modeling error of the interpolation is given by the credible interval of the GPR. Therefore, there is no need of an additional analysis to give such an estimate as it already comes with the interpolation itself.

6.3 BAYES-GAUSS-FOURIER TRANSFORMS

The Bayes-Gauss-Fourier transform (BGFT) is the BGIT algorithm introduced in section 6.1 applied to estimate Fourier transforms. The BGFT will be tested here on the same example introduced in the previous section for testing the noisy DTFT.

A first evident advantage of BGFT over DTFT is that it doesn't require to perform measurements at a constant time rate Δt, since GPR works with any set of domain points. However, in order to perform a fair comparison, we have tested the BGFT algorithm on the very same setup used before. Thus, the measurements $x_i \pm \Delta x_i$ of the even function (6.16) are taken at $t_i = 0, 0.5, ..., 4$.

Outline of BGFT algorithm

The most important step of the whole procedure is the first GPR of the target function (6.1), since the choice of the prior mean $\mu(t)$, the prior covariance $k(t, t')$ and their hyperparameters determines the quality of the probabilistic model of the target function.

In analogy with the procedure introduced in the previous section, the posterior GP obtained with GPR is the replacement for the sinc interpolation, and the transformed

posterior GP, obtained with (6.7), replaces the DTFT. In this case, (6.7) can be rewritten in the following way:

$$\begin{cases} \mu_s^{\mathcal{F}}(\omega|x_i) = \mathcal{F}[\mu(t|x_i)](\omega) \\ k_s^{\mathcal{F}}(\omega,\omega'|x_i) = \sum_{jk} \mathcal{F}[k(t,t_j^s|x_i)](\omega) \left((K^s)^{-1}\right)_{jk} \mathcal{F}[k(t,t_k^s|x_i)](\omega') \end{cases} \tag{6.19}$$

where $\mu_s^{\mathcal{F}}(\omega|x_i)$ and $k_s^{\mathcal{F}}(\omega,\omega'|x_i)$ are the mean and covariance functions of the posterior transformed GP that gives us the Fourier transform estimation and its error. As we explained in section 6.1, evaluating this covariance function requires defining a time grid t_i^s, which is needed to compute the matrix $K^s \equiv k(t_i^s, t_j^s|x_i)$. The choice of t_i^s, as will be explained later, should take in consideration the results of the GPR.

The Fourier transforms of the posterior mean and covariance used in equation (6.19) can be obtained with (6.8), which, in this case, becomes:

$$\begin{cases} \mathcal{F}[\mu(t|x_i)](\omega) = \mathcal{F}[\mu(t)](\omega) + \sum_{ij} \mathcal{F}[k(t,t_i)](\omega) \left(\widetilde{K}^{-1}\right)_{ij} (x_j - \mu(t_j)) \\ \mathcal{F}[k(t,t'|x_i)](\omega) = \mathcal{F}[k(t,t')](\omega) - \sum_{ij} \mathcal{F}[k(t,t_i)](\omega) \left(\widetilde{K}^{-1}\right)_{ij} k(t_j,t') \end{cases} \tag{6.20}$$

where the inverse of $\widetilde{K}_{ij} \equiv k(t_i,t_j) + \Delta x_i^2 \delta_{ij}$ is evaluated in the GPR (6.1).

Tuning the GPR

We saw in chapters 4 and 5 that the choice of the prior (and the hyperprior) has a decisive role in selecting the geometrical properties of the GPR posterior.

In order to identify these properties, let us have a look at the available data shown in the left panel of figure 6.2. The first evident property of the data is the reflection symmetry across the $t = 0$ axis. Indeed, the data is actually only measured at $t \geq 0$. The measurements at $t < 0$ are just a copy of the corresponding ones at $t > 0$.

A possible way to implement this symmetry is by performing a GPR in the $t \geq 0$ half plane, and then copying the same result to the $t < 0$ half plane. An equivalent and more elegant way of achieving the same result is to incorporate this symmetry into the covariance function through the warping method introduced in chapter 2. Considering, for example, the RBF covariance function:

$$k(t,t') = \sigma^2 \exp\left(-\frac{(t-t')^2}{2\ell^2}\right) \tag{6.21}$$

The even symmetry can be enforced using the warping function $\phi(t) = |t|$:

$$k^{\text{even}}(t,t') = \sigma^2 \exp\left(-\frac{(|t|-|t'|)^2}{2\ell^2}\right) \tag{6.22}$$

Observing the GPR in equation (6.1), we can see that the usage of the even kernel (6.22) is equivalent to mirroring the result obtained at $t \geq 0$ with the RBF kernel (6.21) to the $t < 0$ half plane. Indeed, for $t \geq 0$, the posterior mean and covariance in equation (6.1) are the

same regardless of whether kernel (6.21) or kernel (6.22) is used. Furthermore, assuming that the prior mean $\mu(t)$ is an even function, the only other variable dependent on t in the posterior is $k(t, t_i)$, which is an even function if the kernel (6.22) is used. Therefore, using an even $\mu(t)$, we have the guarantee of obtaining the wanted even result.

In general, all the samples drawn using the even kernel show this symmetry. To see why it happens, let us consider the values that a sample might have at a positive $t = a$ and at its negative $t = -a$. Using the even kernel (6.22), the covariance between these two sample points is σ^2, which means that their correlation is 1. This full correlation produces equal values at $t = \pm a$. In other words, the even kernel uses a modified distance function that considers symmetric points to be at zero distance. Since the kernel imposes continuity of the samples, zero distance implies equal sample values.

In case of a non-even, generic prior mean $\mu(t)$, the samples are not necessarily even functions, but their oscillations around the mean are symmetric across the $t = 0$ axis. Because of this property, the even kernel introduced here might be useful to model certain phenomena that are a superposition between a generic function and an even function.

Coming back to our example, after choosing the even covariance function, we need to specify an even prior mean $\mu(t)$. Looking at the data in the left panel of figure 6.2, we notice that the measurements have a value around $x \approx 1$ at $t \to 0$, and decays at a rate similar to that of an exponential decay.

At $|t| > 1$, the data oscillates with a magnitude compatible with their statistical noise. Thus, we can assume that the underlying function is way smoother, and the oscillations are mostly due to the noise.

A possible choice for the prior mean is the constant function $\mu(t) = \mu$ used in chapter 4. However, since we want to apply the Fourier transform to the signal $x(t)$, the target function needs to converge to zero as it required to be L^2 integrable. Using the even covariance (6.22), the behaviour of the posterior mean at $t = \pm\infty$ is fully determined by the prior mean as the measured data is too far to influence it. This means that the prior mean must be square integrable and the only acceptable constant prior mean is $\mu(t) = 0$.

This choice is viable but not optimal as it doesn't incorporate the insights that we have observed about the decay of $x(t)$. When the information given by the data is limited and noisy as in our example, it is advisable to put in the prior all the available knowledge about the underlying function.

Another possibility is to use the exponential prior:

$$\mu(t) = \exp\left(-\frac{|t|}{\mu}\right) \tag{6.23}$$

where μ is a hyperparameter that needs fixing. This prior mean satisfies the properties discussed above as it is an even function that decays exponentially and is L^2 integrable. Furthermore, its Fourier transform is available in closed form:

$$\mathcal{F}[\mu(t)](\omega) = \frac{2\mu}{\mu^2\omega^2 + 1} \tag{6.24}$$

This last property comes useful in evaluating the BGFT transform as (6.24) can be used in equation (6.20).

The only missing step needed to perform the GPR is the determination of the hyperparameters μ, σ and ℓ in the prior mean (6.23) and covariance (6.22). We saw in section 5.3 that the maximum a posteriori (MAP) model selection is useful for selecting the hyperparameters avoiding extreme unwanted values (regularization) and enforcing the Occam's razor principle explained in section 3.4.

In section 5.3 we saw that the hyperparameter for which regularization is most important is the characteristic length-scale ℓ. Indeed, a small value of ℓ can lead to severe overfitting. In this case, ℓ represents the time length-scale of typical oscillations that the target function realizes around the exponential prior mean. Looking at the data in the left panel of figure 6.2, we expect that fictitious oscillations induced by the statistical noise have wavelength around $\ell \sim \Delta t = 0.5$, which is the sampling rate. On the other hand, we expect the target function to be fairly smooth across the sampling window $t \in [-4, 4]$. Therefore, we expect oscillations around the exponential prior mean to be around $\ell \sim 4$.

MAP model selection can be tuned such that values of $\ell \sim 4$ have more priority than $\ell \sim 0.5$. By doing so, the danger of overfitting is less likely, and, at the same time, we incorporate in the regression procedure our requirement of having a function that doesn't oscillate much in the sampling window $t \in [-4, 4]$.

In section 5.3, we saw how the gamma distribution can be very useful to define hyperpriors of positive hyperparameters that need to be fixed with MAP model selection. Indeed, we can define a gamma distribution hyperprior $p(\ell)$ that incorporates our heuristic considerations about ℓ, which are $\ell \sim 4$, with a low probability of having $\ell \lesssim 0.5$. In our example, we have therefore used the following hyperprior for ℓ:

$$p(\ell) = \Gamma(\ell|3, 3/4) \tag{6.25}$$

which means that, using equation (5.5), the mean value and the standard deviation of the ℓ hyperprior are:

$$\begin{cases} \mathbb{E}[\ell] = 4 \\ \sqrt{\text{Var}[\ell]} = 4/\sqrt{3} \approx 2.3 \end{cases}$$

The plot of $p(\ell)$ is shown in the left panel of figure 6.3.

The hyperparameter μ of the prior mean (6.23) is not very problematic to fix for the reasons explained in 5.3. Therefore, it is not necessary to place a hyperprior on μ, which can be fixed with maximum likelihood estimation. On the other hand, it is advisable to define a hyperprior on the sample standard deviation σ to avoid underfitting and overfitting. Since we expect the exponential prior mean to fit the data quite well, the typical sample deviation from the prior mean are expected to be much greater than the statistical noise $\Delta x_i = 0.04$. Thus, we expect the sample standard deviation to be in the order of magnitude $\sigma \sim 0.1$.

A gamma distribution can be used again as a hyperprior for the positive value of σ. The distribution of our choice is the following:

$$p(\sigma) = \Gamma(\sigma|3, 30) \tag{6.26}$$

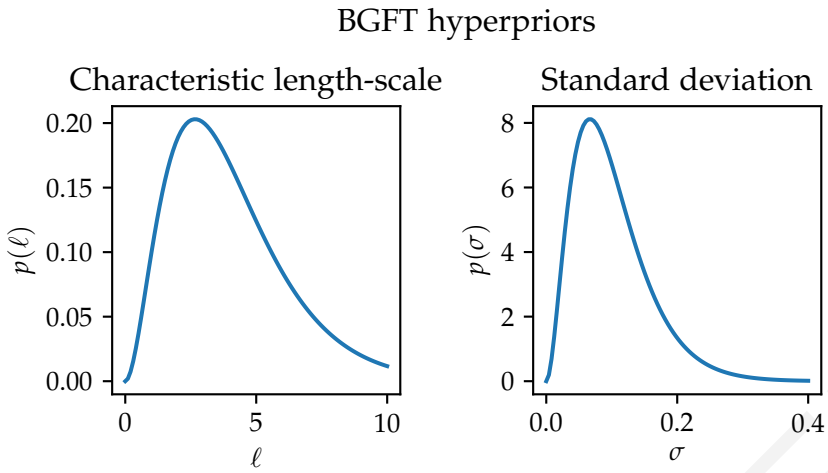


Figure 6.3: Hyperpriors used for the GPR of the testing data.

whose mean and variance are:

$$\begin{cases} \mathbb{E}[\sigma] = 0.1 \\ \sqrt{\text{Var}[\sigma]} = 1/10\sqrt{3} \approx 0.058 \end{cases}$$

The plot of $p(\sigma)$ is shown in the right panel of figure 6.3.

After using the hyperpriors (6.25) and (6.26) to fix the hyperparameters with the MAP estimation (5.4), we can use the prior mean (6.23) and the prior covariance (6.22) to perform the GPR in equation (6.1).

The results are shown in the left panel of figure 6.4. The MAP model selection found $\mu \approx 0.64$, $\sigma \approx 0.073$, $\ell \approx 1.2$ and the GPR posterior mean is compatible with the exact target function. When $|t| < 1$, the posterior is compatible with the exponential prior (6.23), while, for $|t| > 1$, the posterior deviates from the prior mean and infers the noisy data.

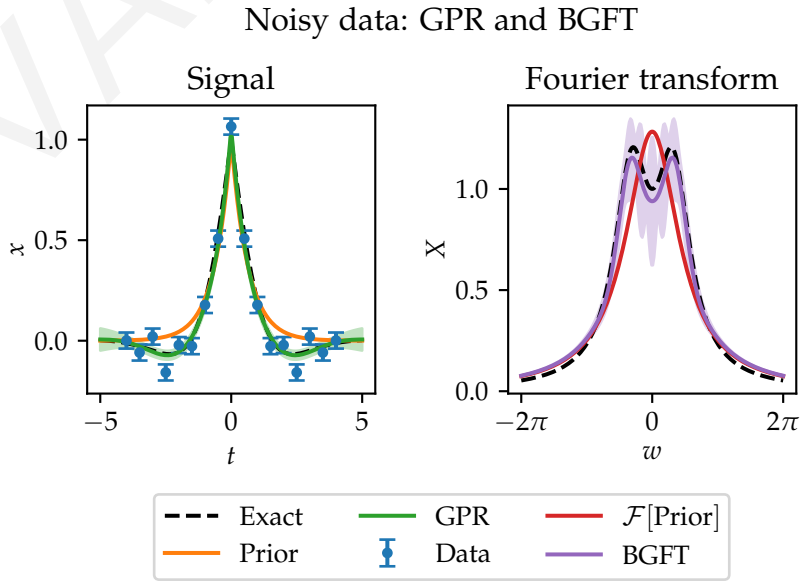


Figure 6.4: GPR and BGFT of the testing data. The hyperparameters found with MAP are $\mu \approx 0.64$, $\sigma \approx 0.073$, $\ell \approx 1.2$. The approximating grid used for the BGFT is $t_i^s = -4, -3.5, ..., 3.5, 4$.

This GPR is then used to evaluate the approximate BGFT with equation (6.7). The results are shown in the right panel of figure 6.4. The transform posterior mean is quite close to the exact target Fourier transform as it has the same shape. Comparing the results to those shown in figure 6.2, both reconstructions are accurate in the regions $|t| < 2$ and $|\omega| < \pi$, while the predictions of the GPR and the BGFT are far better in the rest of the domains. This improvement is due to the fact that the GPR does a better job in inferring the real value of noisy data at $|t| > 2$, which, in the case of DTFT, induce fictitious high frequency deviations.

The small differences of the BGFT from the exact transform at $|\omega| < \pi/2$ are well within the credible interval, while the tails at $|\omega| \to 2\pi$ have a small offset as they are almost equal to the transform of the prior mean. Anyway, this small discrepancy is even smaller than the one obtained in the noiseless DTFT shown in figure 6.1. The error of the BGFT tails seems slightly underestimated because of the few data in the interval $|t| < 2$. Indeed, the model selection procedure manages to find an exponential that overfit the data in this interval. This translates to a Fourier transform that strictly follows the tails of the prior mean transform. This small underestimation of the error could be solved by choosing a covariance function that doesn't have a homogeneous sample variance σ^2. Such a covariance function could be used to increase the modeling error in the region $|t| < 2$ so that the modeling error of the BGFT could be more realistic.

The evaluation of the approximate BGFT (6.7) needs the choice of the S time points of the approximating grid t_i^s. The results shown in figure 6.4 were obtained using $t_i^s = -4, -3.5, ..., 3.5, 4$. The choice of the grid t_i^s has an impact on the posterior credible intervals and we will discuss it in the next subsection.

Choosing the approximating grid t_i^s

In order to fully understand the implications of this choice of t_i^s, it is helpful to first visualize the effects of this approximation on the GPR using equation (6.9). Indeed, studying the approximate GPR will give us some insights about the BGFT that is its Fourier transform.

The discretization t_i^s of the time space could be done in any possible way, however, to keep it simple, we impose t_i^s to be equally spaced at Δt^s distance with a cutoff $t_{\max}^s \geq |t_i^s| \; \forall i$.

In the left panel of figure 6.5 is shown a comparison of two approximate GPRs obtained with equation (6.9) using a cutoff $t_{\max}^s = 4$ and the grid distances $\Delta t^s = 4, 0.5$. A grid with such a high spacing was chosen to better visualize the impact of approximation errors. The credible intervals of the two approximate GPRs are represented by two colored intervals, while the credible intervals of the exact GPR are delimited by two dashed lines. Since the differences of the various approximations are quite small compared to the variations of the posterior mean function, in the right panel of figure 6.5 are shown the error intervals without the mean values in order to make the results more distinguishable.

Observing these results, we can see that, with a grid distance of $\Delta t^s = 0.5$, the approximate credible intervals are overlapping with those of the exact GPR. On the other hand, with a grid distance of $\Delta t^s = 4$, the credible intervals are equal to the exact ones only in the surrounding of the grid points, which are $t = 0$ and $t = \pm 4$. However, when t gets further from the grid points, the approximate GPR underestimate the errors, and this discrepancy

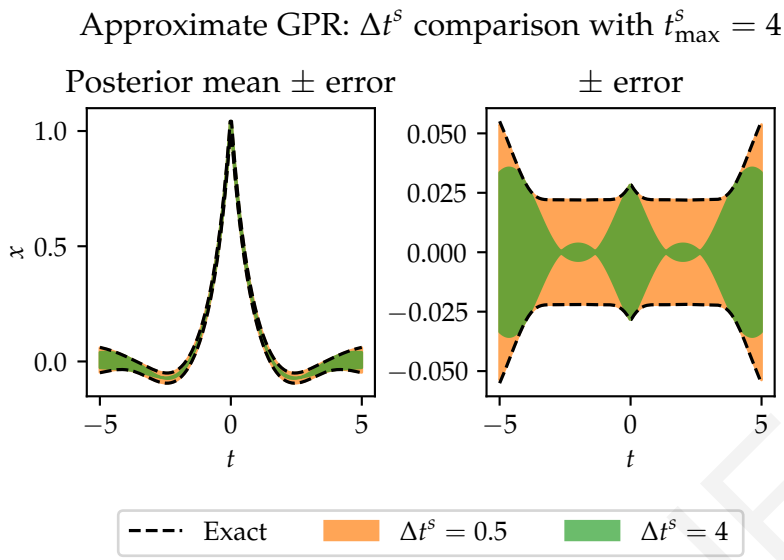


Figure 6.5: Credible intervals of the exact GPR (dashed) and of two approximate GPRs (colored) obtained with grid distances $\Delta t^s = 0.5, 4$. In the right panel, the posterior mean is subtracted from the results for a better visualization.

gets enhanced at a rate that is in the order of magnitude of the characteristic wave-length $\ell \approx 1.2$.

This behaviour becomes clear if we review the reasoning that brought us to the approximate formula (6.9). Indeed, the posterior GP (6.1) is sampled at the grid points t^s_i, and these sample values are interpolated with a second GPR that uses the posterior mean (6.1) as its prior mean. This means that, in the proximity of t^s_i, the interpolated samples behave as the real continuous samples, and their distribution is therefore the exact one, which is the posterior mean in (6.1). This is why the credible intervals of the approximate GPRs in figure 6.5 are equal to those of the exact GPR in proximity of t^s_i.

On the other hand, at time values with a distance from t^s_i that is higher than $\ell \approx 1.2$, all the interpolated samples tend to converge to the posterior mean $\mu(t|x_i)$, since, in these regions, the x values are considered uncorrelated to the values sampled at t^s_i. This is why, in figure 6.5, the standard deviation of the interpolated samples obtained with the sparse grid $t^s_i = 0, 4$ is lower than the exact one.

From these observations, we can find a sufficient condition to suppress the discrepancies induced by a finite grid distance. Indeed, if the distance between the grid points and a value t is far smaller than the characteristic length-scale ℓ, then the discrepancy between the approximate credible intervals and the exact ones is negligible. Using a grid distance Δt^s, the maximum distance between values of $t \in [-t^s_{\max}, t^s_{\max}]$ and grid points is $\Delta t^s/2$. In the example shown in figure 6.5, $\Delta t^s/2 = 0.25$, which is approximately five times lower than $\ell \approx 1.2$ and this is sufficient to suppress approximation errors. Indeed, the correlation at such distance, using the covariance (6.22), is suppressed by a factor $\exp(-5^2/2) \sim 10^{-6}$.

Now that we have seen how to set the grid distance Δt^s, we should discuss the choice of the cutoff $t^s_{\max}$, which involves some additional complications. The even covariance function (6.22) sets σ^2 to be the sample variance at all points that are far from the input data x_i. This means that the samples of the exact posterior GP are not L^2 integrable as

they keep oscillating at $t \to \pm\infty$. However, with a grid cutoff $t^s_{\max}$, the samples converge to the posterior mean at $t \to \pm\infty$. This means that the credible interval starts shrinking at $|t| > t^s_{\max}$ and becomes zero at $t \to \pm\infty$. The samples are therefore L^2 integrable if the posterior mean is L^2 integrable, which is guaranteed using the exponential prior mean (6.23).

The cutoff $t^s_{\max}$ has then the purpose of suppressing sample oscillations for $|t| > t^s_{\max}$. This behaviour can be observed in the approximate GPRs shown in figure 6.6. The grid distance used is $\Delta t^s = 0.5$ and the cutoffs are $t^s_{\max} = 2, 4$. After the cutoffs, as expected, the credible intervals start shrinking to the posterior mean. Hence, the samples have then less freedom to oscillate after the cutoff.

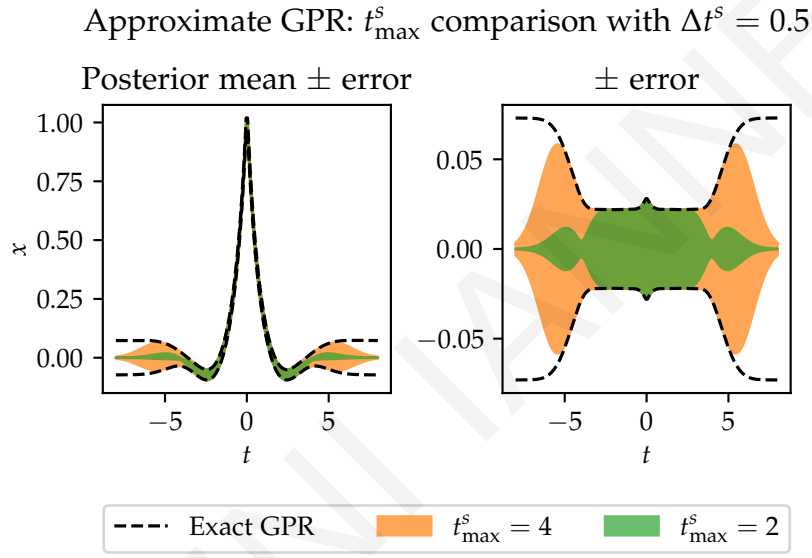


Figure 6.6: Credible intervals of the exact GPR (dashed) and of two approximate GPRs (colored) obtained with grid cutoffs $t^s_{\max} = 2, 4$, and grid distance $\Delta t^s = 0.5$. In the right panel, the posterior mean is subtracted from the results for a better visualization.

This is particularly relevant when evaluating their Fourier transforms. Indeed, oscillations with long wave-length have an impact on the transform at small frequencies. This effect can be seen in the approximate BGFT shown in figure 6.7, which are obtained using the cutoffs $t^s_{\max} = 2, 4, 8$. As expected, if the credible intervals of the GPRs have wide tails, samples have more freedom to have long wave-length oscillations that are not suppressed by setting the hyperparameter ℓ. Thus, having a larger cutoff $t^s_{\max}$ means having a wider credible interval of the transform, especially at small $|\omega|$.

The decision on where to place the cutoff is problem dependent, and it should take into consideration the prior knowledge about the underlying function. In our testing example, we are confident that the main features of the target function are shown in the sampling interval $t \in [-4, 4]$, and that, outside of it, the target function just converges to zero. We can then conclude that placing the approximating grid is not just a numerical necessity, but it can also be useful to build a better probabilistic model of the target function, without having to suffer from finite grid effects, as shown in figure 6.5. In general, similar cutoffs could be achieved using different, non homogenous, covariance functions. However, since the grid

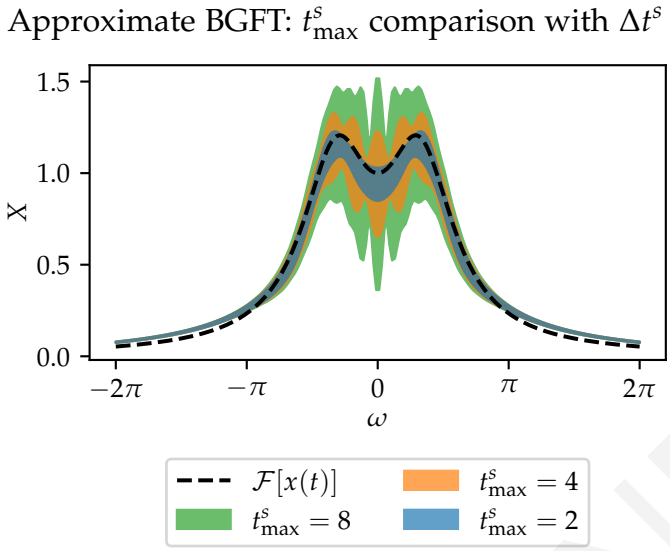


Figure 6.7: Credible intervals of three approximate BGFTs obtained with grid cutoffs $t^s_{\max} = 2, 4, 8$, and grid distance $\Delta t^s = 0.5$.

approximation is necessary for evaluating the BGFT, it could be exploited to improve the probabilistic model, without having to rely on more complicated covariance functions.

7

ACQUISITION FUNCTIONS

As explained in chapter 3, it is possible to use Bayesian inference for making decisions using Bayesian decision theory. This is done by maximizing a utility function or minimizing a loss function defined using the inferred predicting model. In particular, this is the main component of Bayesian optimization (BO).

After having performed N (noisy) measurements of the target function in N different domain points, we have to decide the next domain point at which the new measurement is performed. In BO, our aim is to find the global minimum of the target function. For this, we have to decide which is the domain point that is most likely to be the global minimum.

This choice should be a compromise between exploitation and exploration: we want to observe the target function at the point with a low predicted value (exploitation), while we also want to explore the areas of the domain about which our information is limited as we don't have enough knowledge about the target function to exclude that the global minimum might be there. Exploring new areas of the domain is a crucial property of a global optimizer as it prevents it from getting stuck into local minima.

The compromise between exploration and exploration is set by choosing a specific acquisition function, which uses the predictive model to assign a score to each domain point in terms of how much they could contribute to the exploitation and the exploration of the target function. The next measurement of the target function is then performed at the maximum point of the acquisition function.

Bayesian optimizers could, in principle, be built on top of any probabilistic model of the target function. However, GPR is the most common choice as many acquisition functions can be efficiently evaluated in closed form.

In this chapter, we will describe the expected improvement (EI) [13], which is a very commonly used acquisition function for noiseless BO, and the noisy expected improvement (NEI) [15] which is a generalization of the EI to the case of noisy measurements.

Finally, the functioning of BO will be illustrated finding the global minimum of an example one-dimensional function using EI and NEI.

7.1 EXPECTED IMPROVEMENT

Global optimization using the EI follows the typical steps of Bayesian decision theory. Previous target function measurements are used to construct a predictive model that associates a posterior probability to the possible values that the target function might assume.

These hypothetical possible values are then used to define a utility function that quantifies how beneficial would it be to actually measure such a hypothetical value. The utility function is then integrated over all possible hypothetical values according to our predictive model. By doing so, we obtain the EI, which assigns an expected utility value to each domain point.

The final step of the procedure is looking for the global maximum of the EI across all domain points. This maximum point represents the *most promising* domain point to be the global minimum of the target function, at least according to our predictive model and the utility function that we have chosen. The domain point identified by EI optimization can then be used for the next target function measurement.

GPR predictions

Coming back to the case of our interest, the target function is the parametrized energy measured using a quantum computer and the domain points are quantum circuit parameters.

Let us assume that the target energy function is being observed at N D-dimensional parameter values $\theta_{1\alpha}, ..., \theta_{N\alpha}$ delivering the energy measurements $E_1 \pm \Delta E_1, ..., E_N \pm \Delta E_N$. Then, using equation (4.12), we can perform a GPR to obtain the posterior mean function $\mu(\theta_\alpha|E_i)$ and the posterior covariance function $k(\theta_\alpha, \theta'_\alpha|E_i)$.

We know from equation (4.11), that we can use the posterior GP to perform Bayesian predictions f^* of the values that the target energy would assume with the circuit parameters θ^*_α. Indeed, the posterior predictions f^* are distributed according to a Gaussian distribution:

$$p(f^*|E_i) = \mathcal{N}(f^*|\mu^*, \sigma^{*2}) \tag{7.1}$$

where the mean μ^* is given by the posterior mean function and the variance σ^{*2} is the diagonal term of the posterior covariance function:

$$\begin{cases} \mu^* \equiv \mu(\theta^*_\alpha|E_i) = \mu(\theta^*_\alpha) + \sum_{jk} k(\theta^*_\alpha, \theta_{j\alpha}) \left(\widetilde{K}^{-1}\right)_{jk} (E_k - \mu(\theta_{k\alpha})) \\ \sigma^{*2} \equiv k(\theta^*_\alpha, \theta^*_\alpha) - \sum_{jk} k(\theta^*_\alpha, \theta_{j\alpha}) \left(\widetilde{K}^{-1}\right)_{jk} k(\theta_{k\alpha}, \theta^*_\alpha) \end{cases} \tag{7.2}$$

Here $\mu(\theta_\alpha)$ and $k(\theta_\alpha, \theta'_\alpha)$ are the prior mean and covariance functions, and $\widetilde{K}_{ij} \equiv k(\theta_{i\alpha}, \theta_{j\alpha}) + \Delta E_i^2 \delta_{ij}$.

Improvement utility function

The energy predictions f^* can now be used to define the following utility function:

$$u_{\mathrm{EI}}(\theta^*_\alpha|f^*) \equiv \begin{cases} 0 & \text{if } f^* \geq E^{\min} \\ E^{\min} - f^* & \text{if } f^* < E^{\min} \end{cases} \tag{7.3}$$

where $E^{\min}$ is the minimum of all the previous target energy measurements. This utility function is called *improvement*, since, given a hypothetical energy outcome f^* obtained at θ^*_α, it assigns to θ^*_α a score that corresponds to the improvement over the current best

solution if the energy measurement $E_{N+1} = f^*$ is actually obtained. Indeed, before having measured $E_{N+1} = f^*$, the best solution is $\min\{E_i\}$ and, if $f^* < \min\{E_i\}$, the best solution becomes f^*, which is $\min E_i - f^*$ lower than the previous minimum. On the other hand, if $f^* > \min\{E_i\}$, the best solution remains the same after having measured $E_{N+1} = f^*$, hence the improvement is zero.

At first sight, this utility function seems to be purely exploitative as it doesn't explicitly give a higher score to points with high uncertainty. However, when the uncertainty is high, it is also more likely to observe values far lower than the expected mean. Since the utility function is going to be integrated among all possible values of f^*, points with high uncertainty will benefit from the contribution of small outliers.

In the considered case, the energy measurements E_i are subject to statistical noise, therefore $E^{\min}$ is not a good estimator of the current best solution as its value is susceptible to low outliers. A simple fix for this problem is to replace $E^{\min}$ with $\min_i\{\mu(\theta_{i\alpha}|E_i)\}$, which are the expected values of the previous measurements according to our predictive model. We will discuss in section 7.2 that this solution is not satisfactory in most cases, and that EI needs more substantial modification to optimize noisy target functions.

Expected improvement

In order to remove the dependency of the utility function $u_{\mathrm{EI}}(\theta_\alpha|f^*)$ on specific values of f^*, we can marginalize out this variable integrating the utility function over all the possible f^* according to their posterior distrubution $p(f^*|E_i)$. By doing so, we define the EI acquisition function:

$$a_{\mathrm{EI}}(\theta^*_\alpha) \equiv \int_{-\infty}^{+\infty} u_{\mathrm{EI}}(\theta^*_\alpha|f^*)p(f^*|E_i)\mathrm{d}f^* \tag{7.4}$$

We saw in (7.1) that the distribution of the posterior predictions is a Gaussian, whose mean and variance are given by (7.2). Using the properties of the Gaussian distribution, we can evaluate this integral in closed form. To keep the calculations compact, we denote with $\varphi(t)$ the unit Gaussian and with $\Phi(t)$ the cumulative unit Gaussian:

$$\begin{cases} \varphi(t) \equiv \dfrac{1}{\sqrt{2\pi}} \exp\left(-\dfrac{t^2}{2}\right) \\ \Phi(t) \equiv \dfrac{1}{\sqrt{2\pi}} \displaystyle\int_{-\infty}^{t} \exp\left(-\frac{t'^2}{2}\right) \mathrm{d}t' \end{cases}$$

Plugging the improvement utility function (7.3) and the posterior prediction distribution (7.1) into the EI definition (7.4), we obtain:

$$\begin{aligned} a_{\mathrm{EI}}(\theta_\alpha) &= \int_{-\infty}^{E^{\min}} (E^{\min} - f^*)\mathcal{N}(f^*|\mu^*, \sigma^{*2})\mathrm{d}f^* \\ &= \int_{-\infty}^{t^{\min}} \sigma^*(t^{\min} - t)\frac{\varphi(t)}{\sigma^*}\sigma^*\mathrm{d}t \\ &= \sigma^* \left(t^{\min}\Phi(t^{\min}) - \int_{-\infty}^{t^{\min}} t\varphi(t)\mathrm{d}t \right) \end{aligned}$$

where we have defined $t \equiv (f^* - \mu^*)/\sigma^*$, $t^{\min} \equiv (E^{\min} - \mu^*)/\sigma^*$. Finally, using $\varphi'(t) = -t\varphi(t)$, we find the analytical formula for the EI:

$$\begin{aligned} a_{\mathrm{EI}}(\theta^*_\alpha) &= \sigma^*(t^{\min}\Phi(t^{\min}) + \varphi(t^{\min})) \\ &= (E^{\min} - \mu^*)\Phi\left(\frac{E^{\min} - \mu^*}{\sigma^*}\right) + \sigma^*\varphi\left(\frac{E^{\min} - \mu^*}{\sigma^*}\right) \end{aligned} \tag{7.5}$$

The properties of this formula are not immediate to see. For this reason, it is helpful to compare it to other widely used acquisition functions.

Alternatives to the expected improvement

The most commonly used acquisition functions, together with the EI, are the *probability of improvement* (PI) [43] and the *lower confidence bound* (LCB) [84].

The PI, as the name suggests, assigns to the domain points a score equal to the probability of improving the current best measurement. It can be defined using the following utility function:

$$u_{\mathrm{PI}}(\theta^*_\alpha | f^*) \equiv \begin{cases} 0 & \text{if } f^* \geq E^{\min} \\ 1 & \text{if } f^* < E^{\min} \end{cases}$$

comparing it to the utility function in (7.3), we notice that the positive score here is the same regardless of the amount of improvement. As a consequence, the PI is less exploitative and more explorative than EI.

By integrating (7.1), we obtain the PI acquisition function:

$$a_{\mathrm{PI}}(\theta^*_\alpha | f^*) \equiv \int u_{\mathrm{PI}}(\theta^*_\alpha | f^*)\mathrm{d}f^* = \Phi\left(\frac{E^{\min} - \mu^*}{\sigma^*}\right)$$

As we can see, the PI is simply the cumulative of the Gaussian predictions evaluated at $E^{\min}$, that is the probability of observing a f^* lower than the current best $E^{\min}$. We can clearly see how the PI sets the compromise between exploitation and exploration: domain points with lower values of the expected energy μ^* and higher values of its uncertainty σ^* correspond to higher values of the PI. Seeking for low μ^* is exploitative as it means finding the minimum according to the current model, while seeking for high σ^* is explorative as unexplored regions have a higher prediction variance.

The formula of the EI (7.5) contains the PI in its first term, but it is multiplied by $E^{\min} - \mu^*$, which means prioritizing points where the expected energy is low and penalizing those in which the expected energy is high. This term is therefore way more exploitative than the PI. On the other hand, the second term of the EI (7.5) is mostly explorative as it is proportional to the prediction uncertainty σ^*.

Overall, by definition, the EI is meant to be more exploitative than PI. Indeed, it jumps out of a local minimum only when the uncertainty around it is quite low. Generally, this leads to a faster convergence as the algorithm will reach local minima earlier and immediately start looking for a lower minimum in other areas.

A negative aspect of the EI is that it not very clear, from its formula, how the compromise between exploration and exploitation is set. In case we wanted to change it to favor either one or the other, the EI, in its original form, doesn't provide a way to do it.

However, in our setup, we can prioritize/penalize exploration by increasing/decreasing the sample variance hyperparameter of GPR. This could be done by choosing an appropriate hyperprior on σ^2 as we explained in section 5.3. Indeed, as we saw in chapter 5, this hyperparameter mostly affects the uncertainty of predictions that are far from the already measured data.

In cases where hyperpriors are not used, or, in general, if we want a direct and explicit control over the level of exploration and exploitation, we could instead use the LCB acquisition function. Its definition is straightforward and is not obtained integrating a utility function:

$$a_{\mathrm{LCB}}(\theta_\alpha^*|\lambda) \equiv \lambda\sigma^* - \mu^*$$

With LCB, the points are ranked according to how low is the energy value at λ sigmas lower than the mean. The locus of these points is also called lower confidence bound. The higher is λ, the more the exploration is prioritized, since the lower confidence bound gets lower with a higher multiplicative factor where the uncertainty σ^* is higher.

The LCB shows us that it is not necessary for an acquisition function to be obtained through the integration of a utility function. Indeed, many times, procedures based on heuristic considerations could be helpful to avoid certain problems.

A defect of LCB, however, is that it needs the choice of a value for the parameter λ, which should be fixed accordingly, depending on the specific problem. Since we can already change the balance between exploration and exploitation by modifying the sample variance hyperprior, in our case it would be redundant to introduce another hyperparameter that needs fixing.

Gradient of the expected improvement

Another advantage of having the EI in closed form is that we can easily compute its gradient. This comes useful as we need to optimize the EI to find the set of parameters θ_α^* of the next measurement. Having the possibility to evaluate the gradient in closed form can reduce the number of iterations required to find the maximum of the EI as well as improving the stability and the precision of the procedure. A possible strategy for maximizing the EI is the multistart optimization described in section A.3 of the appendix.

The computation of EI gradient becomes easier if we express it in terms of the partial derivatives in μ^* and σ^*:

$$\frac{\partial a_{\mathrm{EI}}}{\partial \theta_\alpha} = \frac{\partial a_{\mathrm{EI}}}{\partial \mu^*}\frac{\partial \mu^*}{\partial \theta_\alpha} + \frac{\partial a_{\mathrm{EI}}}{\partial \sigma^*}\frac{\partial \sigma^*}{\partial \theta_\alpha}$$

These partial derivatives can be easily obtained from equation (7.5):

$$\begin{cases} \dfrac{\partial a_{\mathrm{EI}}}{\partial \mu^*} = -\dfrac{E^{\min}\varphi}{\sigma^*} - \Phi + \dfrac{\mu^*\varphi}{\sigma^*} + \dfrac{E^{\min} - \mu^*}{\sigma^*}\phi = \Phi \\ \dfrac{\partial a_{\mathrm{EI}}}{\partial \sigma^*} = -\left(\dfrac{E^{\min} - \mu^*}{\sigma^*}\right)^2 \varphi + \varphi - \left(\dfrac{E^{\min} - \mu^*}{\sigma^{*2}}\right)^2 \varphi = \varphi \end{cases} \tag{7.6}$$

The derivatives of μ^* and σ^* can be obtained from equation (7.2) in terms of the derivatives of the chosen prior mean and covariance functions.

Computing explicitly this long chain of partial derivatives and implementing it in a computer program can be lengthy and error-prone. Furthermore, if we want to test different prior covariance or acquisition functions, we would need to also specify the gradient for each of them.

For this purpose, it can be helpful to rely on *automatic differentiation* techniques [85], which evaluate all the partial derivatives together with the function, and then the gradient is obtained using the chain rule. Automatic differentiation software as, for example, PyTorch [86] have implemented the analytic formulae for the derivatives of common functions. Hence, there is no need to explicitly write the derivatives of the acquisition function or of the prior mean and covariance, since all the formulae that we used are expressed in terms of standard functions whose exact derivatives are available in automatic differentiation libraries.

Testing the expected improvement with noiseless measurements

Let us now test the EI on the following one-dimensional example target energy:

$$E(\theta) = \cos(2\theta) + \frac{1}{4}\sin\left(\theta - \frac{\pi}{4}\right) \tag{7.7}$$

This function is shown as a dashed black line in figure 7.1.

Three starting parameter values $\theta_1, \theta_2, \theta_3$ are selected randomly, and the energy (7.7) is measured at these points. Assuming, for now, that the measurements are noiseless, the resulting energies are the three orange points in the top-left panel of figure 7.1.

These three measurements are used to perform the noiseless GPR (4.6), and the resulting posterior mean and covariance functions, which are shown in blue, are used to evaluate the EI (7.5), which is plotted in green. For visual convenience, the EI is shown in arbitrary units and with a vertical offset. The real values of the EI are always greater than or equal to zero.

The green acquisition function is then optimized, and the resulting maximum point (indicated with a vertical green line) is used as the parameter for the fourth measurement, which is shown in the top-right panel. From here, the algorithm iterates the same procedure: the four measurements are used to define a new acquisition function and to select the following value of θ.

Observing the four iterations shown in figure 7.1, we notice that the EI first selects a point close to a local minimum. However, in the next iteration, instead of remaining in the surrounding of the local minimum, the EI selects a point in a region with high uncertainty, since it cannot exclude that the global minimum might be located there. This hypothesis resulted to be correct, and that the global minimum was actually located in the unexplored region. The sixth and the seventh measurements were then chosen to be in the surrounding of this new minimum as there weren't new regions with an uncertainty high enough to make the EI decide to check them. Indeed, we can see that, after six measurements, the predictive model looks very similar to the target function and that the uncertainty is low across the whole domain.

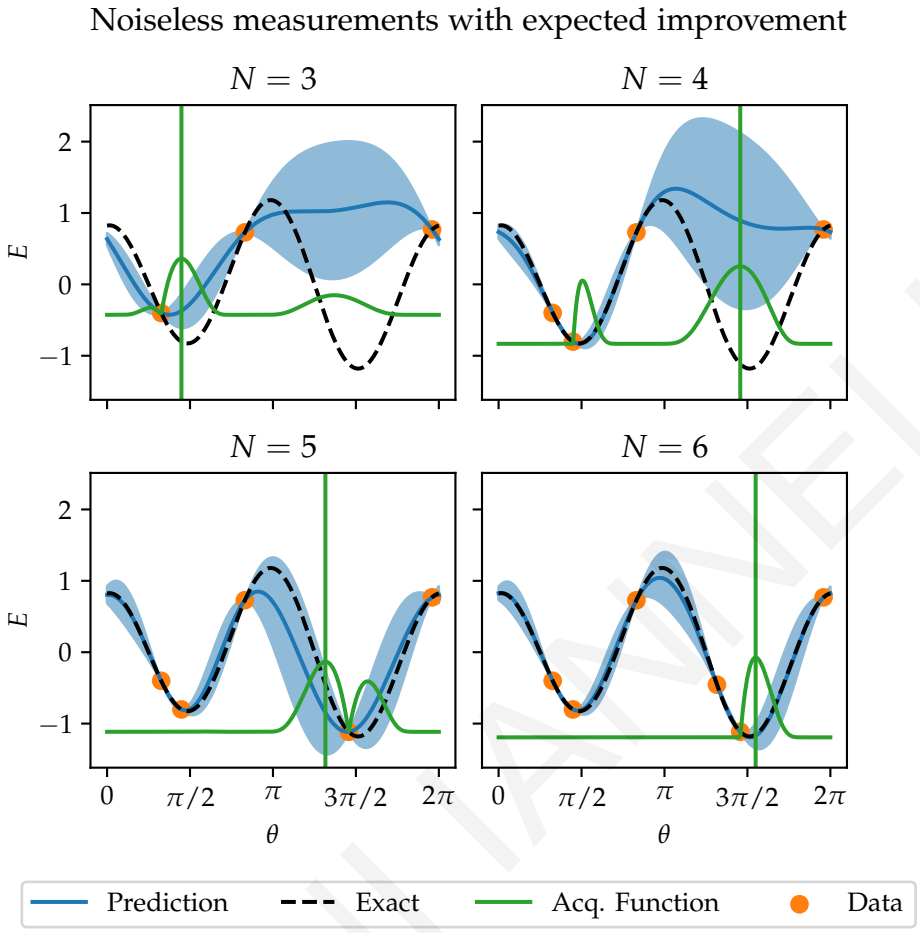


Figure 7.1: Four iterations of a Bayesian optimization using the EI. With noiseless measurements, the global minimum is found quickly.

Testing the expected improvement with noisy measurements

As we anticipated earlier in this chapter, the EI doesn't perform very well if the measurements are afflicted by statistical noise. Let us assume that the energy measurements are subject to a Gaussian noise with $\sigma = 1/2$. Similarly to what happens with a quantum computer, each energy measurement is repeated for a number of $S = 20$ shots, so that our energy estimations $E_i \pm \Delta E_i$ are obtained evaluating the average and the standard error of the shots.

Starting from the same three parameter values $\theta_1, \theta_2, \theta_3$ of the previous example, the new measurements and their errors are shown in orange in the top-left panel of figure 7.2. The result of the first iteration is quite similar to what was obtained with noiseless measurements, since the selected parameter is close to the local minimum. However, in the following iterations, the profile of the EI differs from the noiseless case. Here, instead of exploring the unknown region, the EI keeps selecting measurements around the local minimum without actually improving the predictive model.

The reason of this problem resides in the fact that noisy measurements taken at a certain θ do not nullify the uncertainty of the predictive model at θ. Repeating M times the same measurements can just suppress the uncertainty at the slow rate of $\mathcal{O}(1/\sqrt{M})$. Therefore,

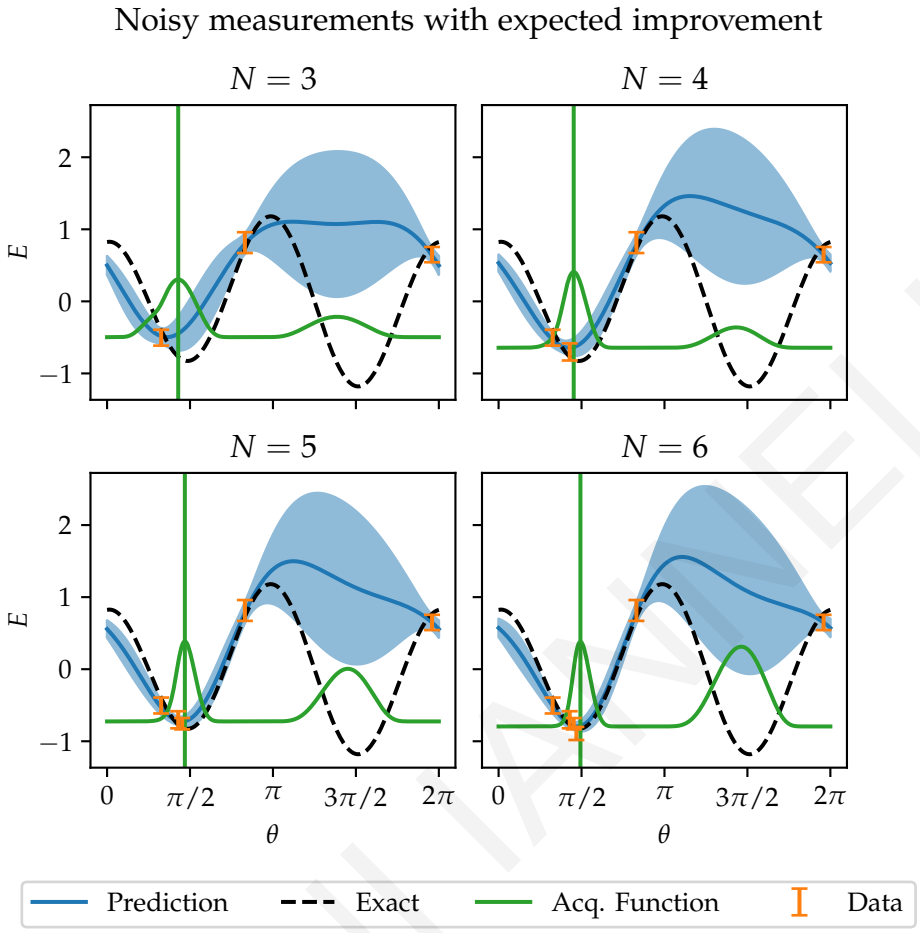


Figure 7.2: Four iterations of a Bayesian optimization using the EI. With noisy measurements, the algorithm is stuck around a local minimum.

if the credible interval at θ doesn't significantly reduce after a measurement at θ, the EI is likely to reselect a very similar value of θ as the setup is similar to what we had in the previous iteration. This doesn't happen in the noiseless setting as the credible interval is zero by definition around noiseless measurements.

We can see this behaviour in the iterations $N = 4, 5, 6$ shown in figure 7.2: the uncertainty around the local minimum is shrinking very slowly, hence the peak of the EI around the local minimum remains high valued. This prevents the EI from jumping out of the local minimum, selecting the second peak located in the unexplored region, thus wasting many iterations without significantly improving the predictive model.

In the next section, we will review different possible ways of extending the EI in order to solve this problem.

7.2 NOISY EXPECTED IMPROVEMENT

Several extensions of the EI have been proposed to make it perform well in the presence of noise. A comparison of different approaches [87] showed that, among the tested acquisition

functions, the best performing were the *augmented expected improvement* (AEI) [88] and the *knowledge gradient* (KG) [89].

The AEI implements heuristic modifications to the EI function in order to mitigate its problems in the noisy setup. In particular, the acquisition function is rescaled with a multiplier that gives priority to points with a higher predictive variance.

However, even if measurement repetitions are discouraged, AEI doesn't exclude them with certainty. Furthermore, the AEI assumes the noise of all the measurements to be identically distributed. Even though this last limitation could be overtaken introducing some further assumptions based on heuristic considerations, this kind of solutions are problem dependent and they are not justified by a rigorous theoretical analysis.

On the other hand, the advantage of AEI with respect to other extensions of EI is its simplicity of calculation. It is available in closed form and is therefore suitable for optimization problems that need a high number of iterations.

The opposite considerations are valid for the KG, which has solid theoretical foundations and it doesn't suffer from the same problems as the EI in the noisy setup. Furthermore, the design of the KG seeks to improve the minimum of the whole predictive model rather than just trying to find an improvement over the previous measurements. The disadvantage of this approach is its computational cost, especially for a high number of parameters, even though this cost can be mitigated using stochastic optimization [90].

Other possible acquisition functions similar to KG are the *entropy search* (ES) [91] and the *predictive entropy search* (PES) [92]. They also have great performance and solid theoretical foundations, but their computational cost is even higher than KG.

Since VQE could potentially have many parameters to optimize, the computational cost of optimizing the acquisition function should be taken into consideration. For this reason, instead of using KG, ES or PES, we decided to use the recently published *noisy expected improvement* (NEI) [15], which solves the problems of EI and AEI without being as computationally expensive as the aforementioned methods.

Hypothetical noiseless energy measurements

As we observed earlier in this chapter, there is a conceptual fault in the EI procedure in presence of noisy measurements. Indeed, the utility function $u_{\mathrm{EI}}(\theta_\alpha^*|f^*)$ defined in (7.3) assumes $E^{\min}$ to be a given constant, while, in the noisy setup, it is a stochastic variable.

Under the hypothesis that the posterior GP obtained with noisy GPR models the target energy, the sample functions of this GP are possible profiles of the target energy. In particular, from the posterior GP, not only we can sample single energy predictions f^* at a single θ_α^*, but we can also, at the same time, sample possible energies f_i at $\theta_{i\alpha}$. Such values of f_i are hypothetical noiseless energy measurements that, under the assumption that our surrogate model is correct, correspond to the real target energy function evaluated at $\theta_{i\alpha}$.

The energy prediction f^* and the hypothetical noiseless measurements f_i are sampled from a $N+1$-dimensional multivariate Gaussian $p(f^*, f_i|E_i)$, whose mean and covariance are given by (4.11) if we redefine there the predictions vector f_m^* as (f^*, f_i).

Noisy improvement utility function

After having specified the joint probability distribution of the predictions f^* and the hypothetical noiseless measurements f_i, we can now define the utility function in terms of them:

$$u_{\text{NEI}}(\theta_\alpha^*|f^*, f_i) \equiv \begin{cases} 0 & \text{if } f^* \geq f^{\min} \\ f^{\min} - f^* & \text{if } f^* < f^{\min} \end{cases} \tag{7.8}$$

where $f^{\min} \equiv \min\{f_i\}$ is the minimum of the hypothetical noiseless measurements.

We can see that this is the same improvement utility function defined in (7.3), but, instead of being the improvement of the prediction f^* over the minimum of the previous noisy measurements $E^{\min}$, it is the improvement over the minimum of the hypothetical noiseless energy measurements. This means that the utility function is no longer evaluated using a constant estimator of the current best solution, but it is rather expressed in terms of N additional stochastic variables f_i.

Noisy expected improvement

Once the utility function (7.8) is defined, the NEI acquisition function is obtained by integrating it among all the possible values of its random variables, according to their probability distributions.

The difference from the EI is that the random variables that need to be integrated out are f^* and f_i, not just f^*. Therefore, the NEI is defined as:

$$a_{\text{NEI}}(\theta_\alpha^*) \equiv \iint u_{\text{NEI}}(\theta_\alpha^*|f^*, f_i) p(f^*, f_i|E_i) \mathrm{d}f^* \mathrm{d}f_i \tag{7.9}$$

Unfortunately, this integral is not available in closed form. However, we can rearrange it so that we can integrate out the posterior predictions f^*. For doing so, we split the integration isolating the conditional probability of f^* using $p(f^*, f_i|E_i) = p(f^*|f_i, E_i)p(f_i|E_i)$:

$$a_{\text{NEI}}(\theta_\alpha^*) = \int \left(\int u_{\text{NEI}}(\theta_\alpha^*|f^*, f_i) p(f^*|f_i, E_i) \mathrm{d}f^* \right) p(f_i|E_i) \mathrm{d}f_i \tag{7.10}$$

From the discussion in section 2.4, we know how to compute the conditional probability of the GP sample value f^* given the fixed sample values f_i. Indeed, fixing a subset of the GP sample values creates a new conditional GP. Therefore, the remaining random variable f^* is distributed according to a Gaussian, whose mean and variance are given by (2.11).

Furthermore, in section 4.3, we saw that a conditional GP is equivalent to a GP obtained performing a GPR on noiseless measurements. This means that $p(f^*|f_i, E_i)$ could be obtained performing a GPR on the hypothetical noiseless measurements f_i.

Having in mind this observation and the EI definition (7.4), we notice that the inner integral in equation (7.10) is an EI $a_{\text{EI}}(\theta_\alpha^*|f_i)$ constructed with the hypothetical noiseless energy measurements f_i:

$$a_{\text{NEI}}(\theta_\alpha^*) = \int a_{\text{EI}}(\theta_\alpha^*|f_i) p(f_i|E_i) \mathrm{d}f_i \tag{7.11}$$

We have therefore integrated out the random variable f^* using the closed form formula (7.5).

With equation (7.11), we have a better understanding of the NEI. Indeed, it can be seen as the expected value of the EI across all the possible hypothetical noiseless measurements generated according to our predictive model. From this formula, it is also clear to see how the NEI is a generalization of the EI: if the measurements E_i are noiseless, then we have $p(f_i|E_i) = \delta(f_i = E_i)$, which reduces equation (7.11) to the EI $a_{\text{EI}}(\theta^*_\alpha|E_i)$.

Being a superposition of noiseless EIs, the NEI is well-defined in the noisy setup: its value is always zero at domain points that have already been measured, since this is true, by definition, for the noiseless EIs. This makes it impossible for the NEI to select the same parameters multiple times.

(Quasi-)Monte Carlo approximation of the noisy expected improvement

As we said earlier, the NEI is not available in closed form. However, it is possible to approximate it using the Monte Carlo method[1]. The simplest way to do it comes directly from the definition (7.9). Indeed, we can generate K independent samples of the variables f^* and f_i from the posterior GP (4.12) using the procedure described in section 2.2. Calling f^*_k and f_{ik} such samples, the NEI is approximated by:

$$a_{\text{NEI}}(\theta^*_\alpha) \approx \frac{1}{K}\sum_k u_{\text{NEI}}(\theta^*_\alpha|f^*_k, f_{ik}) \tag{7.12}$$

According to the central limit theorem, the error of this approximation is $\mathcal{O}(1/\sqrt{K})$.

This method is very straightforward, but it could become very computational intensive when $a_{\text{NEI}}(\theta^*_\alpha)$ needs to be evaluated for a high number of different θ^*. Indeed, we know from section 2.2 that sampling f^*_k and f_{ik} has complexity $\mathcal{O}(K(N+1)^3)$, where N is the number of previous measurements. For each new value of θ^*_α, we need to sample a new set of f^*_k and f_{ik}, and this might be problematic as $a_{\text{NEI}}(\theta^*_\alpha)$ needs to be globally optimized, which usually requires several evaluations of a_{NEI}.

The computational effort could be reduced if we approximate the NEI using a procedure suggested by the alternative formula (7.11). Indeed, we saw that the NEI is the expected value of the EIs obtained using all the possible f_i. Therefore, as we did before, from the posterior GP we generate f_{ik}, which are K samples of the N hypothetical noiseless measurements f_i. In this case, there is no need to generate also f^* because it was integrated away. The f_{ik} are then used to perform the K noiseless GPRs required to compute K different EIs $a_{EI}(\theta^*_\alpha|f_{ik})$ using the closed form formula (7.5). The NEI is finally approximated by the average of these EIs:

$$a_{\text{NEI}}(\theta^*_\alpha) \approx \frac{1}{K}\sum_k a_{\text{EI}}(\theta^*_\alpha|f_{ik}) \tag{7.13}$$

At first sight, the approximation (7.13) doesn't seem a great improvement over the approximation (7.12). Indeed, in order to evaluate $a_{\text{NEI}}(\theta^*_\alpha)$, not only we need to generate

1 More details about the Monte Carlo and the quasi-Monte Carlo methods are covered in section A.2 of the appendix.

all the f_{ik}, which has complexity of $\mathcal{O}(KN^3)$, but also to perform K GPRs, which also has a total complexity of $\mathcal{O}(KN^3)$.

However, the advantage of the approximation (7.13) is that it doesn't need to resample new f_{ik} if we change θ^*_α, since the only random variable dependent on θ^*_α is f^*, but it was integrated out using the analytical formula of the EI. Therefore, the same K EIs $a_{EI}(\theta^*_\alpha|f_{ik})$ can be used through the whole optimization of $a_{\text{NEI}}(\theta^*_\alpha)$. Equation (7.13) is, in fact, an average of K independent EIs, whose values are available in the whole θ^*_α domain thanks to the closed form formula (7.5).

Another advantage of using the approximation (7.11) is that we can easily compute its gradient using the analytical formula (7.6). Indeed, since the NEI is approximated as an average of EIs, its gradient is approximated as an average of EI gradients, and, also in this case, we don't need to resample f_{ik} when we evaluate the gradient at a different value of θ^*_α.

Being just a superposition of EIs, the NEI could be maximized in the same way of the EI, for example, using the multistart optimizazion procedure described in section A.3.

Finally, in order to reduce the number of required samples K, instead of generating stochastic samples from the posterior GP, it is possible to compute them using quasi-random numbers as, for example, the Sobol sequence [93]. Once the f_{ki} are obtained in this way, the rest of the procedure is unchanged.

The integration error of the quasi-Monte Carlo method has, in general, the improved scaling of $\mathcal{O}(K^{-1}\left(\log K\right)^N)$ [94], and it performs better than this in many applications [95]. This error is not Gaussian as in the case of classic Monte Carlo integration, but this is not a concern in our case as we are not interested in performing further statistical analysis with it. For further discussions about the quasi-Monte Carlo integration, we refer to the section A.2 of the appendix.

Testing the noisy expected improvement

A good way of testing the NEI is to try it in the same setup in which the EI performed poorly. Let us then consider again the problem shown in figure 7.2. The same starting parameter values $\theta_1, \theta_2, \theta_3$ are chosen and the same energy measurements with the same errors are obtained in these points as we used the same seed in the random number generator.

Having used the same data, also the noisy GPR, which is plotted in blue in the top-left panel of figure 7.3, resulted to be the same of figure 7.2. The NEI acquisition function obtained in the first iteration ($N = 3$) looks also very similar to the EI, and the chosen fourth parameter value is therefore similar in both cases. The NEI was approximated using $K = 512$ quasi-random samples.

After the fourth measurement is performed, delivering almost the same value as before, and after a similar posterior GP is obtained at the iteration $N = 4$, here is where the two algorithms start behaving very differently. The EI remained stuck around the local minimum as the acquisition function remained high valued even at parameters θ that have already been measured. The situation remains almost unchanged through the next few iterations.

On the other hand, the NEI is, by definition, zero at values of θ that have already been measured. This property breaks the peak of the NEI around the local minimum, and the

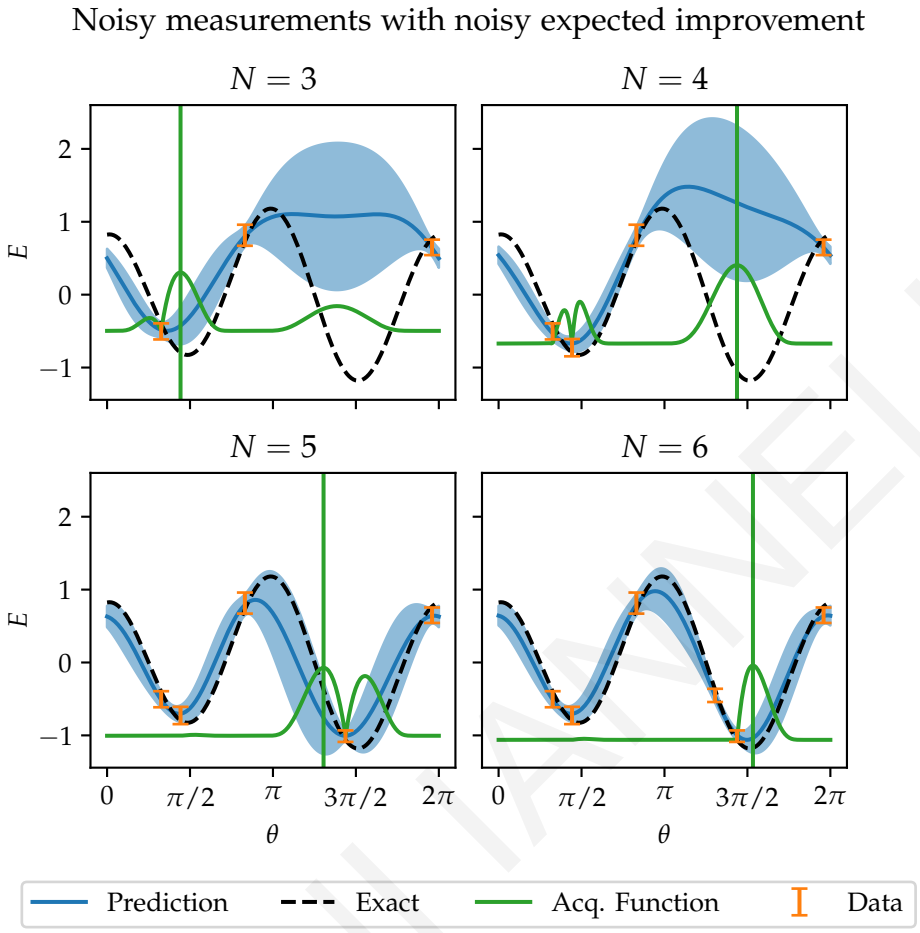


Figure 7.3: Four iterations of a Bayesian optimization using the NEI. With noisy measurements, the algorithm quickly finds the global minimum, without suffering from the same problems of the EI.

algorithm immediately jumps out of it. If, instead, we compare the NEI results in figure (7.3) with those of figure (7.1), which were obtained using EI and noiseless measurements, we notice that the NEI and the EI look very similar in each iteration. This is a good sign as the NEI is supposed to be an extension of the EI to the case of noisy measurements. Being able to obtain results almost identical with those of the noiseless case is exactly the desired behaviour.

We have now explained all the steps required to perform Bayesian optimization. In the next chapter, we will apply it to the variational quantum eigensolver.

8

TESTING THE BAYESIAN VQE

8.1 SUMMARY OF THE BAYESIAN VQE

Now that all the components of Bayesian optimization have been explained in the previous chapters, we summarize here all the steps of the algorithm, explaining also the specific settings that we chose in our tests. We try to add all the details that are necessary to reproduce the results shown in this chapter.

The Hamiltonian H in the form of equation (1.3) and the ansatz states (1.7) are problem dependent. Therefore, for now, we just assume that we have defined a routine that estimates a target parametrized energy $E(\theta_\alpha)$ dependent on D angle parameters θ_α. In particular, using the quantum expected estimation (QEE) algorithm described in section 1.3, we have only access, using equation (1.6), to stochastic measurement of $E(\theta_\alpha)$ and to an estimator of their Gaussian error:

$$E(\theta_\alpha) \xrightarrow{\text{measurement}} E_{\theta_\alpha} \pm \Delta E_{\theta_\alpha} \tag{8.1}$$

Once we have defined the measurement routine (8.1) that makes usage of a quantum computer, the rest of the algorithm runs entirely on a classical computer. Our implementation of Bayesian optimization is built on top of the libraries Ax [96], BoTorch [78] and GPyTorch [97]. The algorithm is then formed by the following steps:

1. Generate N (quasi-)random D-dimensional domain points $\theta_{1\alpha}, ..., \theta_{N\alpha}$ and use (8.1) to measure their corresponding energies $E_1, ..., E_N$ and their errors $\Delta E_1, ..., \Delta E_N$.

   These N points constitute the starting points required for constructing the first surrogate model of the target function $E(\theta_\alpha)$. All the subsequent measurement points will be determined by Bayesian optimization. In order to quickly understand the shape of the target function, the first steps of the optimization algorithm should be mostly exploratory. Therefore, we would like to start from $\theta_{i\alpha}$ equally distributed across the domain. These points could be sampled randomly with the risk of obtaining points close to each other. A more effective way of choosing them is by using a quasi-random number generator (see section A.2 in the appendix), whose output points are guaranteed to be distant from each other. In our tests, we sampled $N = 3$ points using the Sobol sequence [93]. Three points is the minimum amount required for a meaningful Gaussian process regression (GPR) and we didn't notice significant performance difference with higher values of N, since Bayesian optimization is defined anyway to be exploratory when the information about the target energy is low.

2. Given a prior mean function $\mu(\theta_\alpha)$ and a prior covariance function $k(\theta_\alpha, \theta'_\alpha)$, use the data $E_i \pm \Delta E_i$ at $\theta_{i\alpha}$ to fix their hyperparameters with the maximum a posterior (MAP) estimation (5.4).

   In our tests, we followed the procedure described in section 5.3, where we used a constant prior mean $\mu(\theta_\alpha|\mu) = \mu$ dependent on μ and a prior covariance function $k(\theta_\alpha, \theta'_\alpha|\sigma, \ell)$ dependent on σ and ℓ. We wanted to compare the results obtained with the RBF covariance function (2.4) and with its periodic version (2.9) in order to check whether the periodic boundaries significantly improved the optimization convergence rate. Furthermore, we compared these covariance functions with their ARD versions (2.7) and (2.10), which depend on D length-scales ℓ_α rather than on just a scalar ℓ.

   We fixed the hyperparameters μ, σ, ℓ_α using the hyperpriors defined in 5.3 and the hyperposterior optimization (5.4) was performed using the multistart algorithm explained in section A.3 of the appendix, where the number of scans S was set to $40P$ and the number of local search L to $2P$, where P is the number of hyperparameters that need fixing. The gradient of the hyperposterior used for this optimization was evaluated using the automatic differentiation tools available in PyTorch.

3. Compute the posterior mean $\mu(\theta_\alpha|E_i)$ and the posterior covariance $k(\theta_\alpha, \theta'_\alpha|E_i)$ using equation (4.12). For doing this, we need the energy measurements E_i, their Gaussian errors ΔE_i and the hyperparameters fixed at point 2. Then, use this posterior to estimate the current minimum point $\theta_\alpha^{(N)}$ after N iterations across the $\theta_{i\alpha}$ already measured:

$$\theta_\alpha^{(N)} \equiv \text{argmin}_{\theta_{i\alpha}} \mu(\theta_{i\alpha}|E_i) \tag{8.2}$$

   Accordingly, the current estimator of the ground state energy is:

$$E^{(N)} \equiv \mu(\theta_\alpha^{(N)}|E_i) \tag{8.3}$$

   with a credible interval of:

$$\Delta E^{(N)} \equiv \sqrt{k(\theta_\alpha^{(N)}, \theta_\alpha^{(N)}|E_i)} \tag{8.4}$$

   The algorithm is interrupted here if $E^{(N)}$ remains constant within a tolerance ε for k iterations, or simply if we run out of quantum computing time necessary to evaluate again the parametrized energy $E(\theta)$ in a new $\theta_{N+1,\alpha}$.

   In principle, the current minimum point (8.2) could be searched across each possible θ_α, and not just across the already measured $\theta_{i\alpha}$. However, this would require an additional D-dimensional global optimization of $\mu(\theta_\alpha|E_i)$ and the result is likely to have a higher uncertainty (8.4). Furthermore, this estimation would suffer from a larger modelling error as it might select a point far from where we have actually measured the target energy.

4. Use the procedure described in section 2.2 and the posterior found at point 3 to sample K sets of N hypothetical noiseless energy measurements $f_{1i}, ..., f_{Ki}$, where each set

$f_{k1}, ..., f_{kN}$ is sampled at the N measured points $\theta_{1\alpha}, ...\theta_{N\alpha}$. Then, use these K sets and the hyperparameters found at point 2 to perform K noiseless GPRs with equation (4.6). The resulting K posterior GPs can now be used to construct K noiseless EI acquisition functions (7.5), which can finally be used to approximate the NEI $a_{\text{NEI}}(\theta_\alpha)$ using equation (7.12).

In order to obtain a more accurate approximation of the NEI (7.12), we sampled the hypothetical noiseless energy measurements f_{ki} using the Sobol quasi-random generator as explained in section A.2 of the appendix. In our tests, we used $K = 512$, which allowed us to have a fairly consistent profile of the approximated NEI when changing the seed of the quasi-random sequence. However, the algorithm would still work with lower K, the only risk is a higher chance of selecting a suboptimal point for the next iteration.

5. Perform a D-dimensional global optimization of the NEI $a_{\text{NEI}}(\theta_\alpha)$ to find its maximum point $\theta_\alpha^{\text{NEI}}$ and measure the target parametrized energy in this point obtaining $E_{\theta_\alpha^{\text{NEI}}} \pm \Delta E_{\theta_\alpha^{\text{NEI}}}$. Then, add the NEI maximum point to the list of other N measured points $\theta_{N+1,\alpha} \leftarrow\!\shortmid\, \theta_\alpha^{\text{NEI}}$ and the measurement performed in it to the list of energy measurements $E_{N+1} \pm \Delta E_{N+1} \leftarrow\!\shortmid\, E_{\theta_\alpha^{\text{NEI}}} \pm \Delta E_{\theta_\alpha^{\text{NEI}}}$. Finally, iterate the algorithm from point 2 using $N \leftarrow\!\shortmid\, N + 1$.

   We performed the D-dimensional global optimization of $a_{\text{NEI}}(\theta_\alpha)$ using the multistart algorithm defined in section A.3 of the appendix, where the number of scans S was set to $200D$ and the number of local search L to $5D$. Letting the number of local search to scale linearly with the number of dimensions allows the algorithm to be scalable. This doesn't guarantee a successful location of the global maximum, but even a suboptimal point can still be useful. The gradient of the hyperposterior used for this optimization was evaluated using the automatic differentiation tools available in PyTorch.

8.2 TESTING ON TWO QUBITS

Definition of the problem

We tested the algorithm described in section 8.1 on the following two-qubit transverse-field Ising Hamiltonian:

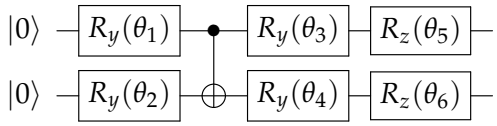
$$H = -\sigma_x^1 \otimes \sigma_x^2 - \sigma_z^1 - \sigma_z^2 \tag{8.5}$$

The exact ground state $|\psi^{\text{min}}\rangle$ and ground state energy E^{min} of the Hamiltonian (8.5) are:

$$E^{\text{min}} = -\sqrt{5} \qquad |\psi^{\text{min}}\rangle = \frac{(2+\sqrt{5})\,|00\rangle + |11\rangle}{\sqrt{10+4\sqrt{5}}} \tag{8.6}$$

which is a highly entangled state that we want to find using the VQE.

For this problem, we constructed the two-qubit ansatz states using the following quantum circuit dependent on six parameters:



(8.7)

it is possible to prove, for example, using the procedure described in [27], that this parametrized quantum circuit can generate any possible two-qubit quantum state up to an irrelevant global phase.

The parametrized energy $E(\theta_\alpha)$ was evaluated with the quantum simulator of the Qiskit library [41] using the quantum noise model of the IBMQ Santiago device. Each single energy measurement was obtained using a fixed number S of shots and its outcome $E \pm \Delta E$ was evaluated using equation (1.6).

In our tests, each measurement was obtained using $S = 32$ shots, which is high enough to justify the usage of the central limit theorem. We could have clearly used a higher S to obtain more precise results, but this is not the point of this benchmark. We want to simulate a realistic case in which we have a limited amount of allocated quantum computing time and we try to find the minimum using the least possible number of cumulative shots, which are $S \times N$, where N is the total number of energy measurements. Furthermore, the amount of statistical noise is higher when using more complex Hamiltonians, therefore, lowering the number of shots when testing on toy models is a good way to simulate the level of noise expected in more complex models.

Anyway, if we want a precise final result, we could run few iterations of a local optimizer starting from the solution of the Bayesian optimizer. For these refinement iterations, we could use a high number of shots to obtain the desired precision.

Therefore, the idea is to find an approximate location of the global minima using the least possible amount of cumulative shots and then increase the number of shots per measurement to refine this solution with a local gradient-based optimizer as, for example, the L-BFGS-B [98].

This is, in general, a recommended procedure in global optimization as global optimizers are designed to quickly locate the global minima, while gradient-based local optimizer are designed to efficiently compute the closest local minimum with an arbitrary precision.

Assessing the performance

The performance of the Bayesian VQE was compared against two other algorithms commonly used for this specific problem: the SPSA [25, 40] and the NFT [38] optimizers. We gave a short description of these algorithms at the end of chapter 1. We used the SPSA implementation included in Qiskit with its default settings and the NFT implementation released by its authors [42] setting the variable `reset_interval=4`, which means that an additional energy measurement is performed every four iterations in order to mitigate the errors induced by the statistical noise.

In order to assess the performance of the optimizers, at each iteration, we stored the current energy minimum as predicted by the optimization algorithm. For the Bayesian VQE, the current minimum estimator $E^{(N)}$ is given by equation (8.3). While the NFT algorithm can estimate the current minimum using the one-dimensional exact formula (1.11), the SPSA is designed to seek the minimum at each iteration, therefore, for the SPSA, we consider the last measured energy as the current minimum estimator.

However, since the energy measurements are affected by statistical noise, we haven't a direct correspondence between the parameters θ_α and the energy $E(\theta_\alpha)$ measured in them. Therefore, it would be desirable to assess the performances using the minimum point estimator θ_α^N rather than the noisy minimum energy estimator $E^{(N)}$.

In order to do this, we have to define an appropriate distance to quantify how close is θ_α^N to the correct solution, which is the exact ground state $|\psi^{\min}\rangle$ in equation (8.6). Since the exact solution is a quantum state vector and our reference point is a real vector θ_α^N, we first need to map these two entities into the same space.

This can be done using the quantum circuit (8.7). Indeed, by definition, we have $U(\theta_\alpha^{\min})\,|0\rangle$, where $\theta_\alpha^{\min}$ is the parameter vector that minimizes the parametrized energy $E(\theta_\alpha)$. Therefore, given a certain θ_α^N, we can use a state-vector simulator to construct $|\psi^{(N)}\rangle \equiv U(\theta_\alpha^{(N)})\,|0\rangle$ by applying the exact parametrized quantum gates to $|0\rangle$.

Then, the compatibility between the current minimum state $|\psi^{(N)}\rangle$ and the exact ground state $|\psi^{\min}\rangle$ can be measured with the *state fidelity*, which, in case of pure states, is their squared overlap:

$$F^{(N)} \equiv |\,\langle\psi^{(N)}|\psi^{\min}\rangle\,|^2 \tag{8.8}$$

which is one if we reach the ground state and lower than one otherwise.

Clearly, this exact fidelity can be computed only in toy models that we can simulate on a classical computer. In more complex cases, we can just rely on the estimated energy. However, the fidelity is still useful to perform accurate benchmarks on toy models in order to select the most promising algorithm that can eventually be used in more complicated systems.

Numerical results

For each different setup, we performed 20 independent optimizations starting with different initial conditions, generated with different seeds. For each configuration, the estimated sequences of estimated minumum energies $E^{(N)}$ and fidelities $F^{(N)}$ were obtained averaging over these 20 results, while their uncertainty was estimated using standard error of their mean. All the optimizers ran for $N = 100$ iterations, which means that the parametrized energy was measured 100 times using $S = 32$ shots per measurement.

In figure 8.1 we show the results obtained with the RBF (2.4) and the periodic (2.9) covariance functions in their simplest form, in which we use a unique characteristic length-scale ℓ for each direction. These results are compared with those obtained with the NFT and the SPSA algorithm.

The two BOs have a similar performance. This means that, in this problem, there isn't a significant advantage in using the periodic boundaries. Anyway, their implementation

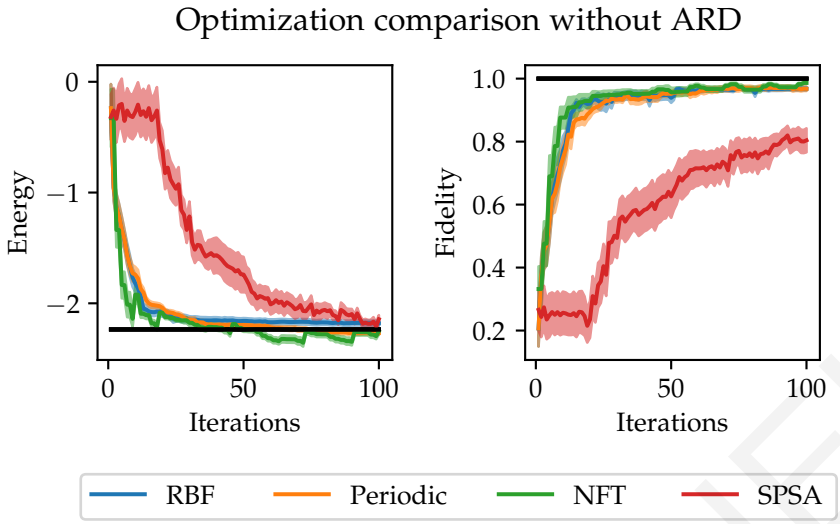


Figure 8.1: Optimization comparison between two BOs using different kernels, the NFT and the SPSA optimizers. Here the BOs are not using the automatic relevance determination.

doesn't increase neither the algorithm complexity, nor the required CPU time. Therefore, we still recommend their usage because in certain cases they might be advantageous, for example, when the solution is close to a boundary. A possible explanation of the lack of improvement with periodic boundaries is that, in six dimensions, the target function is quite complicated and, therefore, we can make accurate prediction only at points close to our measurements, and, at short distances, the two kernels have similar properties as they both impose C^∞ smoothness.

Comparing these results with the SPSA, we notice that the BOs have a significant performance advantage over the SPSA. Indeed, after 100 energy measurements the BOs are very close to the exact solution, while the SPSA has a state fidelity that is still quite distant from one.

On the other hand, the NFT has a small advantage over the BOs, even though their performance is very similar. However, we want to emphasize that the NFT uses an exact formula that is not appliable for a generic parametrized quantum circuit. Therefore, if we want to optimize a non-standard quantum circuit that used parametrized gates that are not qubit rotations, then the NFT wouldn't be appliable and BO generally offers a much more measure-efficient solution than SPSA.

In figure 8.2, we produced a similar plot but using the automatic relevance determination (ARD) versions of the RBF (2.7) and the periodic (2.10) covariance functions. This means that the GPR is allowed to use a different characteristic length-scale ℓ_α in each direction α. The more *relevant* directions α should obtain smaller ℓ_α in the model selection procedure.

However, these result don't show any significant advantage in using ARD compared to what was obtained in figure 8.1. The ARD kernels, however, come with a significantly higher computational cost as all the ℓ_α need to be simultaneously fixed with a hyperparameter global optimization, whose complexity increases with the number of hyperparameters that need fixing.

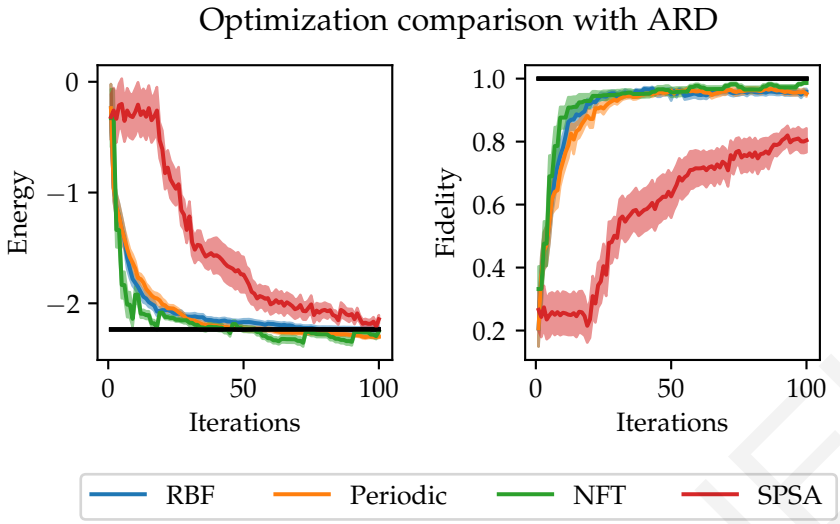


Figure 8.2: Optimization comparison between two BOs using different kernels, the NFT and the SPSA optimizers. Here the BOs are using the automatic relevance determination.

Even though the ARD gives us a much more flexible modelling tool, this flexibility might have some drawbacks other than the higher computational cost. Indeed, a more flexible model is more susceptible to overfitting, especially if the number of hyperparameters is not far lower than the number of data points. The hyperposteriors defined in section 5.3 are very helpful to avoid overfitting, but we cannot exclude it with certainty. Furthermore, since the global optimization of many hyperparameters can be very complex, we are likely to end up finding only suboptimal hyperparameters that don't allow us to fit the data in the best possible way.

Therefore, the usage of ARD is only recommended when we have a good reason to believe that the target function varies with very different length-scales in different axes directions. This might be the case, for example, in optimization problems where groups of hyperparameters have a very different meaning, and some of them might be almost irrelevant. In our case, we know that our circuit is dimensionally efficient, therefore, we don't have strong reasons to believe that the ARD could identify axes that are far more relevant than others. There are symmetries of the Hamiltonian that haven't been removed from the quantum circuit, but these symmetries don't trivially correspond to single parameters identifiable with ARD.

8.3 POSSIBLE EXTENSIONS TO HIGH-DIMENSIONAL SPACES

We saw in section 8.2 and in chapter 7 that the BO algorithm summarized in section 8.1 is very efficient at optimizing low-dimensional black-box functions, even in presence of high statistical noise. Indeed, it showed performances that are only slightly inferior to the NFT algorithm, which uses a closed-form formula not appliable everywhere, and outperforms the SPSA, which is another black-box noisy optimizer frequently used for VQE.

However, the BO implementation summarized in 8.1 becomes very CPU-intensive in high-dimensional spaces. In particular, the point number 5 of this algorithm requires, at each iteration, a global optimization of a D-dimensional acquisition function. This might become very difficult for $D \gtrsim 20$ since the acquisition function usually has many local minima. Even though selecting a suboptimal point could still be beneficial for the optimization, this is likely to significantly decrease the convergence rate.

Possible solutions of this particular problem are the *dropout* [46] and the *lineBO* [47] algorithms. They are based on the same principle, which is to restrict the domain of the acquisition function optimization. In particular, the dropout algorithm consists in randomly selecting a subset of $D' < D$ parameters and optimize the acquisition function in the subspace spanned by them. The global convergence is still obtained as the D' parameters are randomly sampled at each iteration. Therefore, the full domain is covered in the long-run. On the other hand, the lineBO selects a random straight line across the domain and optimizes the acquisition function along this one-dimensional line. These two algorithms were tested in $D \sim 40$ dimensions outperforming common alternative algorithms [46, 47].

Unfortunately, another potential problem might arise in high dimensional spaces. Indeed, when the number of dimensions is $D \gtrsim 40$, it is very likely that the optimization would require a total number of iterations N that is $\mathcal{O}(10^3)$ or greater. As we saw in section 4.5, GPR requires the inversion of a $N \times N$ matrix, which is generally a $\mathcal{O}(N^3)$ operation. Therefore, this might become both CPU and memory intensive, especially if we use the NEI that requires performing GPR several times in each iteration of the algorithm. We explained in section 4.5 that the complexity of the inversion can be reduced to $\mathcal{O}(TN^2)$ using T iterations of an approximate method. This can be very useful when N is $\mathcal{O}(10^3)$, but it might not be enough for N in $\mathcal{O}(10^4)$, which might occur when $D \gtrsim 50$.

A possible naive solution to this problem could be to *forget* old measurements, which means defining a threshold $N^{\max}$ and performing the GPR using only the last $N^{\max}$ measurements. However, this solution showed poor performances in our tests as the algorithm ended up re-exploring the areas that were forgotten. In order to properly work, this solution would also require to progressively reduce the domain around the current best minimum, at the cost of increasing the chances of getting stuck in a local minimum.

Another possibility is to reduce the dimensionality of the target function domain. Since we are capable of optimizing well a black-box function in a lower number of dimensions, a possible solution is to embed the target function into a lower dimensional space and optimize it in there.

A popular algorithm that implements this idea in BO is the REMBO [45] algorithm, which uses the properties of random matrices to find an approprate embedding. However, this algorithm relies on the assumption that the effective dimensionality of the function is lower than D. In our case, this means assuming that our parametrized quantum circuit is not dimensionally efficient, which is not true if we constructed it, for example, using the procedure described in [27, 28]. Indeed, we have already reduced the number of parameters to the minimum and the REMBO, in our case, would just find a suboptimal solution.

Another possible approach would be to follow the same idea used in the NFT algorithm, which is to sequentially optimize the target function in different subspaces. Instead of fixing

just one parameter at a time, we could use the BO to sequentially optimize random subsets of parameters. This is similar to the idea used in the dropout algorithm, but it is different here, since we start a new optimization from scratch in each new subspace, therefore without having to invert $N \times N$ matrices, but only $N' \times N'$, where N' is the number of iterations carried out in the current subspace.

However, this algorithm showed poor performance in our tests when using subspaces of $D \sim 10$ parameters. Indeed, it had problems refining the solution since each subspace optimization takes many iterations while giving a very small improvement to the current best solution. On the other hand, by optimizing only one parameter at a time, as in the NFT algorithm, we can give small refinement with very few iterations, and, overall, obtaining better performances.

Using BO to optimize sequentially one parameter at a time would be a viable solution in high-dimensional spaces in which the formula used in the NFT algorithm is not appliable. However, a better alternative called stochastic gradient line Bayesian optimization (SGLBO) [48] was recently published specifically for the VQE optimization. The SGLBO uses BO to minimize the target function along a generic one-dimensional straight line, not necessary aligned with an axis.

This line is chosen in the direction of the current gradient, which is estimated using a sequence of single-shot measurements in each axis direction. This gradient estimator is clearly stochastic, but this property could also be helpful to jump out of local minima as it happens in the SPSA. Furthermore, being able to modify simultaneously many parameters in the gradient direction, the SGLBO is more efficient in refining the solution compared to the NFT. Indeed, the tests performed by its authors converged to a solution that was more precise than the one obtained with the NFT algorithm, without having to rely on an additional final local optimization.

The idea of the SGLBO could also be applied to the algorithm developed in this thesis. Indeed, both the noisy GPR that we defined in section 4.4 and the NEI that we explained in 7.2 can be applied for the gradient line optimization.

The advancements in quantum hardware are giving us the opportunity to use quantum circuits with increasingly more parameters that need to be defined efficiently exploiting all the possible symmetries. For doing this, we will need to use a wider variety of quantum gates or even to define mutable circuits that have different gates for different with different parameter values. All these complex circuits cannot be optimized with the closed formula used by the NFT algorithm, and, in light of these promising results, we think that BO will be a fundamental tool to optimize them in the near future. Therefore, we hope that this thesis could be helpful for the multidisciplinary community of scientists that will try to use the VQE in their field.

9

CONCLUSIONS AND OUTLOOKS

CONCLUSIONS

In this thesis, we developed two algorithms for two different tasks: the approximation of integral transforms and the minimization of parametrized energies measured using quantum computers. Both of these algorithms are based on the same Gaussian process regression (GPR) procedure defined in chapter 4, which uses a set of measurements and their statistical errors to build predictive models of black-box functions.

In particular, in chapter 6, we developed a novel method called Bayes-Gauss integral transform (BGIT) for estimating integral transforms of black-box functions using limited sets of noisy data. The estimation obtained with this method is available in closed form and we deduced a formula for approximating the uncertainty of the integral transform at, in principle, arbitrary precision.

In section 6.3, we tested the BGIT for the estimation of a Fourier transform obtaining better results and more realistic error estimations compared to what we achieved using an algorithm based on discrete-time Fourier transforms. Furthermore, we used this method to estimate parton distribution functions using lattice QCD data [17].

Nevertheless, the main focus of the thesis is the variational quantum eigensolver (VQE) algorithm introduced in section 1.3. In particular, we developed a strategy to globally optimize the parametrized energy, which is evaluated using a quantum computer. This procedure can then be used to find the ground state of a given Hamiltonian.

In this work, we proposed the use of Bayesian optimization (BO) that is a method particularly suitable for optimization problems in which we need to minimize a black-box function that is expensive to evaluate. The classical optimization step of VQE falls into this category as the measurements of the parametrized energy function are the computational *bottleneck* of the procedure. Indeed, they require the execution of a high number of quantum circuits using quantum computers, the availability of which, at the moment, is much more limited than that of high-performance classical computers. Therefore, we are willing to perform the CPU-intensive operations required for BO if this allows us to reduce the number of energy measurements that are required to minimize the energy function.

Our implementation of BO relies on GPR to construct a surrogate model of the target parametrized energy and uses the noisy expected improvement (chapter 7) to choose the point for the next iteration. A distinctive feature of our algorithm is that it uses both the energy measurements and their errors, incorporating them into a single probabilistic model. Most of the optimizers commonly used for this task makes don't make usage of the

information provided by the measurement errors. This feature allows our algorithm to be particularly robust to statistical noise, obtaining stable and precise results even when using very few repetitions (or *shots*) per measurement.

In principle, the measurement errors could be suppressed by using a high number of shots. However, this requires performing a higher number of expensive energy measurements that are not strictly necessary to find an approximate location of the global minimum. Indeed, by using an optimization algorithm that is robust to statistical noise, we can reduce the cumulative number of shots necessary to get close to the global minimum, and then, only at this point, increase the number of shots per measurement in order to refine the final result.

The performance of our BO algorithm, which is summarized in section 8.1, is compared to those obtained with two other optimizers, the SPSA and the NFT, that have been used, with success, for the VQE optimization in many scientific works. While the BO and the SPSA are black-box function optimizers applicable to any parametrized quantum circuit, the NFT uses an analytical formula that is valid if the only used parametrized quantum gates are qubit rotations. In this work, we chose a testing circuit that falls into this category in order to compare the convergence rates of the three algorithms on the same problem, keeping in mind that the NFT is not applicable to the general case. In the results shown in section 8.2, the BO clearly outperformed the other black-box function optimizer, the SPSA, and performed almost as well as the NFT.

In this test, we also compared different possible variants of the BO that seek to incorporate different properties into the surrogate model using the kernels defined in chapter 2. We tested the RBF and the periodic kernels, which both assume the target function to be C^∞, but the periodic kernel also assumes it to be 2π-periodic in each parameter.

However, even though we know that the target function is both C^∞ and periodic, we didn't observe significant differences in using these kernels. Our interpretation of this result is that, in high-dimensional spaces, the target function becomes very complicated, and the construction of the surrogate model becomes more *data-driven*: generic assumptions about points of the domain that are far from our current location become irrelevant, since, with such a complicated target function, we can get accurate predictions only in the surrounding of the measured points. Then, their behaviour is similar because each of the tested kernels has a similar behaviour at short distances, since they all assume that the target function has C^∞ smoothness.

We also tested the automatic relevance determination (ARD) versions of these kernels, which can adaptively identify whether some parameters are more relevant than others during their optimization. However, also in this case, we didn't notice any significant advantage in using ARD in our testing example.

In conclusion, without significant performance differences, we recommend choosing the kernel with the lowest computational cost. Since the RBF and the periodic kernels have a similar cost, we recommend the periodic kernel as it might have an advantage if the solution is close to a boundary. On the other hand, the ARD kernels have a much higher computational cost as they require fixing a higher number of hyperparameters. Therefore, we advise against using them unless there is a good reason to do so, for example, if the quantum circuit is not dimensional-efficient and some parameters might be irrelevant.

We also discussed, in section 8.3, possible strategies to extend our algorithm into a high-dimensional space, since, in its standard implementation, BO struggles to work in more than $D \sim 30$ dimensions because of the high CPU and memory requirement. Among the discussed methods, the most promising is the recently published gradient line Bayesian optimization (SGLBO) [48], which sequentially optimizes the target function along the direction of a stochastic estimator of its gradient. This technique is applicable to the algorithm developed in this thesis and offers a natural extension to it.

OUTLOOKS

These final considerations outline a clear path for the continuation of this work. We would like to implement SGLBO to our algorithm or, maybe, to design a different criterion to identify the subspace to optimize. This would allow us to optimize high-dimensional ansatz circuits, which is definitely what we plan to do, given the current fast-paced development of quantum devices.

Indeed, our goal is to use the VQE to simulate lattice gauge theories and we are particularly interested in situations where the simulation with Markov-chain Monte algorithm is very problematic, as, for example, in presence of a chemical potential or of a topological θ term.

This task requires the development of each of the components of the VQE: a better quantum hardware with more qubits and with lower levels of noise, better noise mitigation techniques, a better quantum circuit design and, finally, better optimization algorithms. In this regard, we hope that this thesis gave a useful contribution.

BIBLIOGRAPHY

[1] Ashley Montanaro. "Quantum algorithms: an overview." In: *npj Quantum Information* 2 (2015), p. 15023.

[2] Iulia M Georgescu, Sahel Ashhab, and Franco Nori. "Quantum simulation." In: *Reviews of Modern Physics* 86.1 (2014), p. 153.

[3] Jacob D. Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. "Quantum machine learning." In: *Nature* 549 (2017), pp. 195–202.

[4] John Preskill. "Quantum Computing in the NISQ era and beyond." In: *Quantum* (2018).

[5] Shor. "Scheme for reducing decoherence in quantum computer memory." In: *Physical review. A, Atomic, molecular, and optical physics* 52 4 (1995), R2493–R2496.

[6] Peter W. Shor. "Fault-tolerant quantum computation." In: *Proceedings of 37th Conference on Foundations of Computer Science* (1996), pp. 56–65.

[7] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C. Benjamin, and Xiao Yuan. "Quantum computational chemistry." In: *Reviews of Modern Physics* (2020).

[8] Marí Carmen Bañuls et al. "Simulating lattice gauge theories within quantum technologies." In: *The European Physical Journal D* (2020).

[9] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O'brien. "A variational eigenvalue solver on a photonic quantum processor." In: *Nature communications* 5 (2014), p. 4213.

[10] Matthias Troyer and U. Wiese. "Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations." In: *Physical review letters* 94 17 (2005), p. 170201.

[11] Giuseppe Magnifico, Timo Felser, Pietro Silvi, and Simone Montangero. "Lattice quantum electrodynamics in (3+1)-dimensions at finite density with tensor networks." In: *Nature Communications* 12 (2021).

[12] Angus Kan, Lena Funcke, Stefan Kühn, Luca Dellantonio, Jinglei Zhang, Jan F. Haase, Christine A. Muschik, and Karl Jansen. "Investigating a (3+1)D topological θ-term in the Hamiltonian formulation of lattice gauge theories for quantum and classical simulations." In: *Physical Review D* (2021).

[13] Donald R. Jones, Matthias Schonlau, and William J. Welch. "Efficient Global Optimization of Expensive Black-Box Functions." In: *Journal of Global Optimization* 13 (1998), pp. 455–492.

[14] Giovanni Iannelli and Karl Jansen. "Noisy Bayesian optimization for variational quantum eigensolvers." In: *Proceedings of The 38th International Symposium on Lattice Field Theory — PoS(LATTICE2021)* (2022).

[15] Benjamin Letham, Brian Karrer, Guilherme Ottoni, Eytan Bakshy, et al. "Constrained Bayesian optimization with noisy experiments." In: *Bayesian Analysis* 14.2 (2019), pp. 495–519.

[16] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. MIT press Cambridge, MA, 2006.

[17] Constantia Alexandrou, Giovanni Iannelli, Karl Jansen, and Floriano Manigrasso. "Parton distribution functions from lattice QCD using Bayes-Gauss-Fourier transforms." In: *Physical Review D* 102.9 (2020), p. 094508.

[18] Robert Raussendorf and Hans J Briegel. "A one-way quantum computer." In: *Physical Review Letters* 86.22 (2001), p. 5188.

[19] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. "Quantum computation by adiabatic evolution." In: *arXiv preprint quant-ph/0001106* (2000).

[20] Vincent Danos, Elham Kashefi, and P. Panangaden. "The measurement calculus." In: *J. ACM* 54 (2007), p. 8.

[21] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. "Adiabatic quantum computation is equivalent to standard quantum computation." In: *45th Annual IEEE Symposium on Foundations of Computer Science* (2004), pp. 42–51.

[22] Daniel Comparat. "General conditions for quantum adiabatic evolution." In: *Physical Review A* 80 (2006), p. 012106.

[23] Barenco, Bennett, Cleve, DiVincenzo, Margolus, Shor, Sleator, Smolin, and Weinfurter. "Elementary gates for quantum computation." In: *Physical review. A, Atomic, molecular, and optical physics* 52 5 (1995), pp. 3457–3467.

[24] Rami Barends et al. "Logic gates at the surface code threshold: Superconducting qubits poised for fault-tolerant quantum computing." In: *Nature* (2014).

[25] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets." In: *Nature* 549.7671 (2017), pp. 242–246.

[26] Michael A Nielsen and Isaac L Chuang. "Quantum Computation and Quantum Information." In: (2010).

[27] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, and Paolo Stornati. "Dimensional Expressivity Analysis of Parametric Quantum Circuits." In: *Quantum* 5 (2021), p. 422.

[28] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, Manuel Schneider, and Paolo Stornati. "Best-approximation error for parametric quantum circuits." In: *2021 IEEE International Conference on Web Services (ICWS)* (2021), pp. 693–702.

[29] Jan F. Haase, Luca Dellantonio, Alessio Celi, Danny Paulson, Angus Kan, Karl Jansen, and Christine A. Muschik. "A resource efficient approach for quantum and classical simulations of gauge theories in particle physics." In: *Quantum* 5 (2021), p. 393.

[30] Lena Funcke, Tobias Hartung, Karl Jansen, Stefan Kühn, Paolo Stornati, and Xiaoyang Wang. "Measurement error mitigation in quantum computers through classical bit-flip correction." In: *Physical Review A* (2022).

[31] Jarrod R. McClean, Jonathan Romero, Ryan Babbush, and Alán Aspuru-Guzik. "The theory of variational hybrid quantum-classical algorithms." In: *New Journal of Physics* 18 (2015).

[32] O Crawford, Barnaby van Straaten, Daochen Wang, Thomas Parks, Earl T. Campbell, and Stephen Brierley. "Efficient quantum measurement of Pauli operators in the presence of finite sampling error." In: *Quantum* 5 (2021), p. 385.

[33] Alexei Y. Kitaev. "Quantum measurements and the Abelian Stabilizer Problem." In: *Electron. Colloquium Comput. Complex.* 3 (1996).

[34] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." In: *SIAM J. Comput.* 26 (1999), pp. 1484–1509.

[35] Daniel S. Abrams and Seth Lloyd. "Quantum Algorithm Providing Exponential Speed Increase for Finding Eigenvalues and Eigenvectors." In: *Physical Review Letters* 83 (1999), pp. 5162–5165.

[36] Suguru Endo, Tyson Jones, Sam McArdle, Xiao Yuan, and Simon C. Benjamin. "Variational quantum algorithms for discovering Hamiltonian spectra." In: *Physical Review A* (2019).

[37] Oscar Higgott, Daochen Wang, and Stephen Brierley. "Variational Quantum Computation of Excited States." In: *Quantum* (2019).

[38] Ken M Nakanishi, Keisuke Fujii, and Synge Todo. "Sequential minimal optimization for quantum-classical hybrid algorithms." In: *Physical Review Research* 2.4 (2020), p. 043158.

[39] John A. Nelder and Roger Mead. "A Simplex Method for Function Minimization." In: *Comput. J.* 7 (1965), pp. 308–313.

[40] James C Spall et al. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation." In: *IEEE transactions on automatic control* 37.3 (1992), pp. 332–341.

[41] Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, D Bucher, FJ Cabrera-Hernández, J Carballo-Franquis, A Chen, CF Chen, et al. "Qiskit: An open-source framework for quantum computing." In: *Accessed on: Mar* 16 (2019).

[42] Ken M Nakanishi, Keisuke Fujii, and Synge Todo. *Nakanishi-Fujii-Todo method for scipy.optimize*. 2019. URL: `https://github.com/ken-nakanishi/nftopt`.

[43] Harold J. Kushner. "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise." In: *Journal of Basic Engineering* 86 (1963), pp. 97–106.

[44] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. "Algorithms for Hyper-Parameter Optimization." In: *NIPS*. 2011.

[45] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. "Bayesian optimization in a billion dimensions via random embeddings." In: *Journal of Artificial Intelligence Research* 55 (2016), pp. 361–387.

[46] Cheng Li, Sunil Gupta, Santu Rana, Vu Nguyen, Svetha Venkatesh, and Alistair Shilton. "High dimensional Bayesian optimization using dropout." In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017, pp. 2096–2102.

[47] Johannes Kirschner, Mojmir Mutny, Nicole Hiller, Rasmus Ischebeck, and Andreas Krause. "Adaptive and safe Bayesian optimization in high dimensions via one-dimensional subspaces." In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3429–3438.

[48] Shiro Tamiya and Hayata Yamasaki. "Stochastic gradient line Bayesian optimization for efficient noise-robust optimization of parameterized quantum circuits." In: *npj Quantum Information* 8.1 (2022), pp. 1–13.

[49] Ondřej Straka, Jindřich Duník, Miroslav Šimandl, and Jindřich Havlík. "Aspects and comparison of matrix decompositions in unscented Kalman filter." In: *2013 American Control Conference*. IEEE. 2013, pp. 3075–3080.

[50] Radford M Neal. "Monte Carlo implementation of Gaussian process models for Bayesian regression and classification." In: *arXiv preprint physics/9701026* (1997).

[51] Erlend Aune, Jo Eidsvik, and Yvo Pokern. "Iterative numerical methods for sampling from high dimensional Gaussian distributions." In: *Statistics and Computing* 23.4 (2013), pp. 501–521.

[52] RJ Adler. "The Geometry of Random Fields." In: *The Geometry of Random Fields, Chichester: Wiley, 1981* (1981).

[53] Petter Abrahamsen. *A review of Gaussian random fields and correlation functions*. 1997.

[54] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[55] *NIST Digital Library of Mathematical Functions*. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller, B. V. Saunders, H. S. Cohl, and M. A. McClain, eds. URL: `http://dlmf.nist.gov/`.

[56] Radford M Neal. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media, 2012.

[57] Mark N Gibbs. "Bayesian Gaussian processes for regression and classification." PhD thesis. University of Cambridge, 1998.

[58] Christopher J Paciorek and Mark J Schervish. "Nonstationary covariance functions for Gaussian process regression." In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. 2003, pp. 273–280.

[59] Paul D Sampson and Peter Guttorp. "Nonparametric estimation of nonstationary spatial covariance structure." In: *Journal of the American Statistical Association* 87.417 (1992), pp. 108–119.

[60] David JC MacKay. "Introduction to Gaussian process." In: *Neural Networks and Machine Learning* (1998).

[61] Morris L Eaton. *Multivariate statistics: a vector space approach.* John Wiley & Sons, Inc., 1983.

[62] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

[63] Bruno De Finetti. *Theory of probability: A critical introductory treatment*. Vol. 6. John Wiley & Sons, 2017.

[64] Stephen E Fienberg. "When did Bayesian inference become" Bayesian"?" In: *Bayesian analysis* 1.1 (2006), pp. 1–40.

[65] David Barber. *Bayesian reasoning and machine learning*. Cambridge University Press, 2012.

[66] Dennis V Lindley. *Understanding uncertainty*. John Wiley & Sons, 2013.

[67] Stuart Geman and Donald Geman. "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images." In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), pp. 721–741.

[68] Radford M Neal. "Slice sampling." In: *The annals of statistics* 31.3 (2003), pp. 705–767.

[69] John Pina Craven. *The Silent War: The Cold War Battle Beneath the Sea*. Simon and Schuster, 2002.

[70] Bradley Efron. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

[71] David JC MacKay. "Bayesian interpolation." In: *Neural Computation* 4.3 (1992), pp. 415–447.

[72] Stephen F Gull. "Developments in maximum entropy data analysis." In: *Maximum entropy and Bayesian methods*. Springer, 1989, pp. 53–71.

[73] Georges Matheron. *Traité de géostatistique appliquée*. 14. Editions Technip, 1962.

[74] Per Christian Hansen. "The truncatedsvd as a method for regularization." In: *BIT Numerical Mathematics* 27.4 (1987), pp. 534–553.

[75] Magnus Rudolph Hestenes, Eduard Stiefel, et al. *Methods of conjugate gradients for solving linear systems*. Vol. 49. 1. NBS Washington, DC, 1952.

[76] Cornelius Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators." In: (1950).

[77] John Skilling. "Bayesian numerical analysis." In: *Physics and Probability* 1 (1993), pp. 203–221.

[78] Maximilian Balandat, Brian Karrer, Daniel R Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. "BoTorch: Programmable Bayesian Optimization in PyTorch." In: *arXiv preprint arXiv:1910.06403* (2019).

[79] Anthony O'Hagan. "Bayes–hermite quadrature." In: *Journal of statistical planning and inference* 29.3 (1991), pp. 245–260.

[80] Lokenath Debnath and Dambaru Bhatta. *Integral transforms and their applications*. Chapman and Hall/CRC, 2016.

[81] Luca Ambrogioni and Eric Maris. "Integral Transforms from Finite Data: An Application of Gaussian Process Regression to Fourier Analysis." In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 217–225.

[82] Claude Elwood Shannon. "Communication in the presence of noise." In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21.

[83] Ronald I Becker and Norman Morrison. "The errors in FFT estimation of the Fourier transform." In: *IEEE transactions on signal processing* 44.8 (1996), pp. 2073–2077.

[84] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias W. Seeger. "Gaussian Process Optimization in the Bandit Setting: No Regret and Experimental Design." In: *ICML*. 2010.

[85] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. "Automatic differentiation in machine learning: a survey." In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[86] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. "PyTorch: An imperative style, high-performance deep learning library." In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035.

[87] Victor Picheny, Tobias Wagner, and David Ginsbourger. "A benchmark of kriging-based infill criteria for noisy optimization." In: *Structural and multidisciplinary optimization* 48.3 (2013), pp. 607–626.

[88] Deng Huang, Theodore T Allen, William I Notz, and Ning Zeng. "Global optimization of stochastic black-box systems via sequential kriging meta-models." In: *Journal of global optimization* 34.3 (2006), pp. 441–466.

[89] Warren Scott, Peter Frazier, and Warren Powell. "The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression." In: *SIAM Journal on Optimization* 21.3 (2011), pp. 996–1026.

[90] Jian Wu and Peter Frazier. "The parallel knowledge gradient method for batch bayesian optimization." In: *Advances in Neural Information Processing Systems*. 2016, pp. 3126–3134.

[91] Philipp Hennig and Christian J Schuler. "Entropy Search for Information-Efficient Global Optimization." In: *Journal of Machine Learning Research* 13.6 (2012).

[92] José Miguel Hernández-Lobato, Matthew W Hoffman, and Zoubin Ghahramani. "Predictive entropy search for efficient global optimization of black-box functions." In: *Advances in neural information processing systems* 27 (2014).

[93] IM Sobol. "On the distribution of points in a cube and the approximate evaluation of integrals." In: *USSR Computational Mathematics and Mathematical Physics (English translation)* 7.4 (1967), pp. 86–112.

[94] Russel E Caflisch. "Monte carlo and quasi-monte carlo methods." In: *Acta numerica* 7 (1998), pp. 1–49.

[95] Josef Dick, Frances Y Kuo, and Ian H Sloan. "High-dimensional integration: the quasi-Monte Carlo way." In: *Acta Numerica* 22 (2013), pp. 133–288.

[96] Eytan Bakshy, Lili Dworkin, Brian Karrer, Konstantin Kashin, Benjamin Letham, Ashwin Murthy, and Shaun Singh. *AE: A domain-agnostic platform for adaptive experimentation*.

[97] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration." In: *Advances in Neural Information Processing Systems*. 2018, pp. 7576–7586.

[98] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. "A limited memory algorithm for bound constrained optimization." In: *SIAM Journal on scientific computing* 16.5 (1995), pp. 1190–1208.

[99] Malte Kuss. "Gaussian process models for robust regression, classification, and reinforcement learning." PhD thesis. echnische Universität Darmstadt Darmstadt, Germany, 2006.

[100] Makoto Matsumoto and Takuji Nishimura. "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator." In: *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 8.1 (1998), pp. 3–30.

[101] Melissa E O'Neill. "PCG: A family of simple fast space-efficient statistically good algorithms for random number generation." In: *ACM Transactions on Mathematical Software* (2014).

[102] Michael J Wichura. "Algorithm AS 241: The percentage points of the normal distribution." In: *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 37.3 (1988), pp. 477–484.

[103] William J Morokoff and Russel E Caflisch. "Quasi-monte carlo integration." In: *Journal of computational physics* 122.2 (1995), pp. 218–230.

[104] John H Halton. "On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals." In: *Numerische Mathematik* 2.1 (1960), pp. 84–90.

[105] Henri Faure. "Discrépance de suites associées à un système de numération (en dimension s)." In: *Acta arithmetica* 41.4 (1982), pp. 337–351.

[106] George Marsaglia and Thomas A Bray. "A convenient method for generating normal variables." In: *SIAM review* 6.3 (1964), pp. 260–264.

[107] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. "Optimization by simulated annealing." In: *science* 220.4598 (1983), pp. 671–680.

A

APPENDIX

A.1 MULTIVARIATE GAUSSIAN DISTRIBUTION

We give here a list of properties of the multivariate Gaussian distribution that are useful for the calculations carried out in this thesis.

We will indicate as $\mathcal{N}(\mu, K)$ a multivariate Gaussian with mean vector μ and covariance matrix K. Its probability density function is:

$$\mathcal{N}(x|\mu, K) = \det(2\pi K)^{-\frac{1}{2}} e^{-\frac{1}{2}(x-\mu)^\top K^{-1}(x-\mu)}$$

Linear transformation of a Gaussian random variable

Let a be an N-dimensional vector, A an $N \times N$ (semi-)positive definite matrix and x a random variable distributed as the multivariate Gaussian $\mathcal{N}(a, A)$. Let us also define the affine transformation $y = b + Bx$ determined by the M dimensional vector b and by the $M \times N$ matrix B. Then, we have:

$$y \equiv b + Bx \xrightarrow{\text{distributed as}} \mathcal{N}(y|b + Ba, BAB^\top) \tag{A.1}$$

A proof of this proposition can be found, for example, in [61].

Product of Gaussians

Let a, x, y be N-dimensional vector, and A, B be $N \times N$ (semi-)positive definite matrices. Then, the products between the two multivariate Gaussians $N(x|a, A)$ and $N(y|x, B)$ can be rearranged as follows:

$$\mathcal{N}(x|a, A)\mathcal{N}(y|x, B) = \mathcal{N}(x|c, C)\mathcal{N}(y|a, A + B) \tag{A.2}$$

where $c \equiv C(A^{-1}a + B^{-1}y)$ and $C = (A^{-1} + B^{-1})^{-1}$. A procedure for deriving this identity can be found, for example, in [16].

Introducing another $N \times N$ matrix D, we have a similar identity [99]:

$$\mathcal{N}(x|a, A)\mathcal{N}(y|Dx, B) \propto \mathcal{N}(x|c, C) \tag{A.3}$$

where $c \equiv C(A^{-1}a + D^\intercal B^{-1}y)$ and $C = (A^{-1} + D^\intercal B^{-1}D)^{-1}$.

Matrix derivatives

Given an invertible matrix K, the elementwise partial derivatives of its inverse can be rearranged in the following way [16]:

$$\partial_i K^{-1} = -K^{-1}(\partial_i K)K^{-1} \tag{A.4}$$

While the partial derivatives of log determinant of K are equal to [16]:

$$\partial_i \log|K| = \mathrm{tr}(K^{-1}\partial_i K) \tag{A.5}$$

A.2 (QUASI-)MONTE CARLO INTEGRATION

Traditionally, Monte Carlo integration relies on pseudo-random number generators (PRNG) such as the Mersenne Twister [100] or the PCG [101], which are deterministic algorithms that generate sequences of integer numbers whose properties approximate those of random numbers.

Rescaling the output of a PRNG, it is possible to restrain the sequence to rational numbers between zero and one. For most applications, these rescaled sequences could be assumed to be uniformly distributed in $[0, 1)$. This comes very useful to sample other non-uniform distributions.

Inverse transformation method

For instance, given a sequence $u_1, ..., u_K$ of samples from a uniform distribution between zero and one, the *inverse transformation method* is a commonly used procedure to generate a sequence $x_1, ..., x_K$ of samples distributed according to a a generic one-dimensional $p(x)$.

Calling $F(x)$ the cumulative distribution function of $p(x)$:

$$F(x) \equiv p(x' < x) = \int_{-\infty}^{x} p(x')\mathrm{d}x'$$

samples x of the target distribution $p(x)$ can be obtained by applying the inverse function $F^{-1}(u)$ to the uniform samples u:

$$x \equiv F^{-1}(u) \xrightarrow{\text{distributed as}} p(x) \tag{A.6}$$

To check the correctness of this procedure, we need to verify that the random variable $x = F^{-1}(u)$ is actually distributed according to $p(x)$. This is equivalent to verify that $p(u < u_0) = p(x' < F^{-1}(u_0))$, which can be obtained with the following change of variables:

$$\begin{aligned} p(u < u_0) &= \int_0^{u_0} \mathrm{d}u = \int_{-\infty}^{F^{-1}(u_0)} \frac{\mathrm{d}u}{\mathrm{d}x}\mathrm{d}x \\ &= \int_{-\infty}^{F^{-1}(u_0)} p(x)\mathrm{d}x = p(x < F^{-1}(u_0)) \end{aligned}$$

Sampling a unit Gaussian

Considering, for example, the case of the unit Gaussian $\varphi(t) \equiv \mathcal{N}(t|0,1)$, we can use the inverse transformation method to obtain samples of $\varphi(u)$ using the inverse of its cumulative distribution function $\Phi(t)$. Indeed, if u is a random variable uniformly distributed between zero and one, then:

$$t \equiv \Phi^{-1}(u) \xrightarrow{\text{distributed as}} \varphi(t) \tag{A.7}$$

The inverse cumulative of the unit Gaussian $\Phi^{-1}(u)$, also called *probit* function, can be efficiently approximated in many ways, for example using [102].

Sampling a multivariate Gaussian

The procedure for sampling $\varphi(t)$ can also be useful to sample a generic multivariate Gaussian $\mathcal{N}(x|\mu, K)$. Indeed, calling t_i a vector of independent samples of the unit Gaussian $\varphi(t)$ and A a matrix such that $K = AA^\top$, we know from equation (A.1) that:

$$x_i \equiv \mu_i + \sum_{ij} A_{ij} t_j \xrightarrow{\text{distributed as}} \mathcal{N}(x|\mu, K) \tag{A.8}$$

Monte Carlo integration

Let us consider a multidimensional integral of the following form:

$$\mathcal{I} \equiv \int f(x_i) p(x_i) \mathrm{d}x_i \tag{A.9}$$

where $f(x_i)$ is a $\mathbb{R}^N \to \mathbb{R}$ function, and $p(x_i)$ is the multivariate probability density function of the N-dimensional x_i.

Pseudo-random numbers are commonly used to approximate integrals in the form of equation (A.9). Indeed, if we can generate K samples $x_{1i}, ..., x_{Ki}$ of the N-dimensional x_i from its distribution $p(x_i)$, then the integral (A.9) can be approximated as:

$$\mathcal{I} \approx \mathcal{A} \equiv \frac{1}{K} \sum_k f(x_{ki})$$

This directly follows from the central limit theorem, which states that, if the random variable $f(x_i)$ has a finite variance, then, as $K \to \infty$, $\mathcal{A}$ converges to a Gaussian variable whose mean and variance are:

$$\begin{cases} \mathbb{E}[\mathcal{A}] = \mathbb{E}[f(x_i)] \equiv \mathcal{I} \\ \mathrm{Var}[\mathcal{A}] = \frac{1}{K}\mathrm{Var}[f(x_i)] \equiv \frac{1}{K}\left(\int f^2(x_i) p(x_i) \mathrm{d}_i - \mathcal{I}^2\right) \end{cases}$$

The error of Monte Carlo integration is therefore $\mathcal{O}(K^{-1/2})$, and its advantage over standard numerical quadrature techniques arises in high dimensional spaces. Indeed, quadrature rules have a convergence rate that becomes slower as the number of dimension

increases. For example, the error of integrals approximated with the midpoint rule is $\mathcal{O}(K^{-1/N})$, where K is the number of grid points and N the number of dimensions.

Quasi-Monte Carlo integration

An even better convergence of high-dimensional integrals could be achieved using quasi-random numbers, which are deterministic sequences that, instead of imitating the behaviour of random numbers, they are designed to uniformly occupy the space.

Sequences of random numbers inevitably tend to form clumps of higher density in some regions of the space, and this is suboptimal if we want to converge to the correct solution as quick as possible. Indeed, a point that is very close to another provides little additional information about a smooth integrand function.

On the other hand, an quasi-random sequence $x_{1i}, ..., x_{Ki}$ of N-dimensional points is defined such that, for any value of K, any region of the space should have roughly the same ratio between number of points and volume. This property is also called *low discrepancy* and a formal definition of it can be found in [94].

Quasi-random sequences are therefore meant to cover the integration domain more quickly so that the integration could converge with a faster rate. Indeed, while the error of a standard Monte Carlo integration is $\mathcal{O}(K^{-1/2})$, using quasi-random numbers the error is generally $\mathcal{O}(K^{-1}(\log K)^N)$, or much lower in many applications [95].

Many different quasi-random sequences are available. A comparison [103] between the Halton [104], Sobol [93] and Faure [105] sequences showed that the Halton sequence achieved the best integral approximations in $N \lesssim 6$ dimensions, while the Sobol sequence outperformed the others when the number of dimensions was higher.

These quasi-random generators produce sequences of points u in the unit hypercube $[0, 1]^N$, which, in most cases, can be simply used as a replacement for pseudo-random numbers sampled from a uniform distribution. Indeed, the inverse transformation method (A.6) can be used to generate quasi-random samples x that distribute themselves as $p(x)$ more quickly than pseudo-random samples would do.

There are some drawbacks of the quasi-Monte Carlo method that should be taken into consideration. For instance, the integration error, which can be estimated using the Koksma-Hlawka inequality [94], is not Gaussian as it is for the classical Monte Carlo integration. This can be disadvantageous if we intend to use the results for further statistical analysis.

Another disadvantage is that some sampling method based on accept/reject procedures cannot be used with quasi-random numbers, since removing a subset of the sequence makes it lose its property. For example, the Marsaglia polar method [106], which is commonly used to generate Gaussian samples, cannot be used with quasi-random numbers.

A.3 MULTISTART OPTIMIZATION

Bayesian global optimization requires itself to use a global optimizer in order to work. Indeed, we need to find the global maximum of the hyperposterior (5.4) and of the NEI acquisition function (7.13) at each iteration of the algorithm.

The difference between these optimizations and the original problem of optimizing a parametrized quantum circuit is that they don't require the usage of a quantum computer. Indeed, they are way easier to optimize as the target functions that need optimizing can be evaluated cheaply many times, and without measurement noise.

Both the hyperposterior (5.4) and the NEI acquisition function (7.13) usually have several local minima separated by large and almost flat areas. See, for example, the NEI acquisition functions in figure 7.3. This is problematic for many global optimizers such as, for example, those based on simulated annealing [107], since tunneling between local maxima might take a very high number of iterations.

On the other hand, the availability of the gradient of the target functions makes it easy, given a starting point, to ascend to a local maximum using a gradient-based local optimizer. The L-BFGS-B algorithm [98] is a good choice for this task given its good performance in high dimensions with the presence of boundaries. Furthermore, it requires the target function and its gradient to be always be evaluated together, which is convenient if we use automatic differentiation techniques [85], since they evaluate the target function and its gradient at the same time.

The multistart procedure that we used is composed by two steps. At first, the target function is evaluated on a uniform grid generated using the Sobol sequence (see section A.2). Then, among all the scanned points, we select a subset of those with the highest function values, and we run the L-BFGS-B starting from them. By doing so, we obtain a list of local maxima, and we select the highest.

The reason for the first scan is to concentrate the local search only around the highest local maxima, avoiding then the lower local maxima and the flat areas of the target function that wouldn't be worth it to explore.

The algorithm can be summarized as follows:

- Given a target real function $f(x)$ defined in an N-dimensional bounded domain, we specify a number of local searches L and a number of scans $S > L$.
- An N-dimensional grid of S domain points x_i is generated using the Sobol sequence.
- The target function is evaluated on the grid points and, among them, are selected the L points $x_{i_1}, ..., x_{i_L}$ in which $f(x_i)$ had the highest values.
- The points $x_{i_1}, ..., x_{i_L}$ are used to initiate L instances of the L-BFGS-B local optimizer, which end at L local maxima $x_1^{\max}, ..., x_L^{\max}$.
- The algorithm returns the maximum point among the local maximum points: $\max_{x_i^{\max}}\{f(x_i^{\max})\}$.

The values of L and S should be selected depending on the problem using some heuristic considerations. Indeed, a high value of S is recommended if the target function is mostly flat in a significant part of the domain, while a high value of L should be used if we believe that there are many local maxima.

The convergence to the global maximum is not guaranteed using this procedure. However, this is not a big problem for the Bayesian optimizer as suboptimal values of the hyperposterior and the acquisition function are still useful for the algorithm as they correspond to plausible solutions.