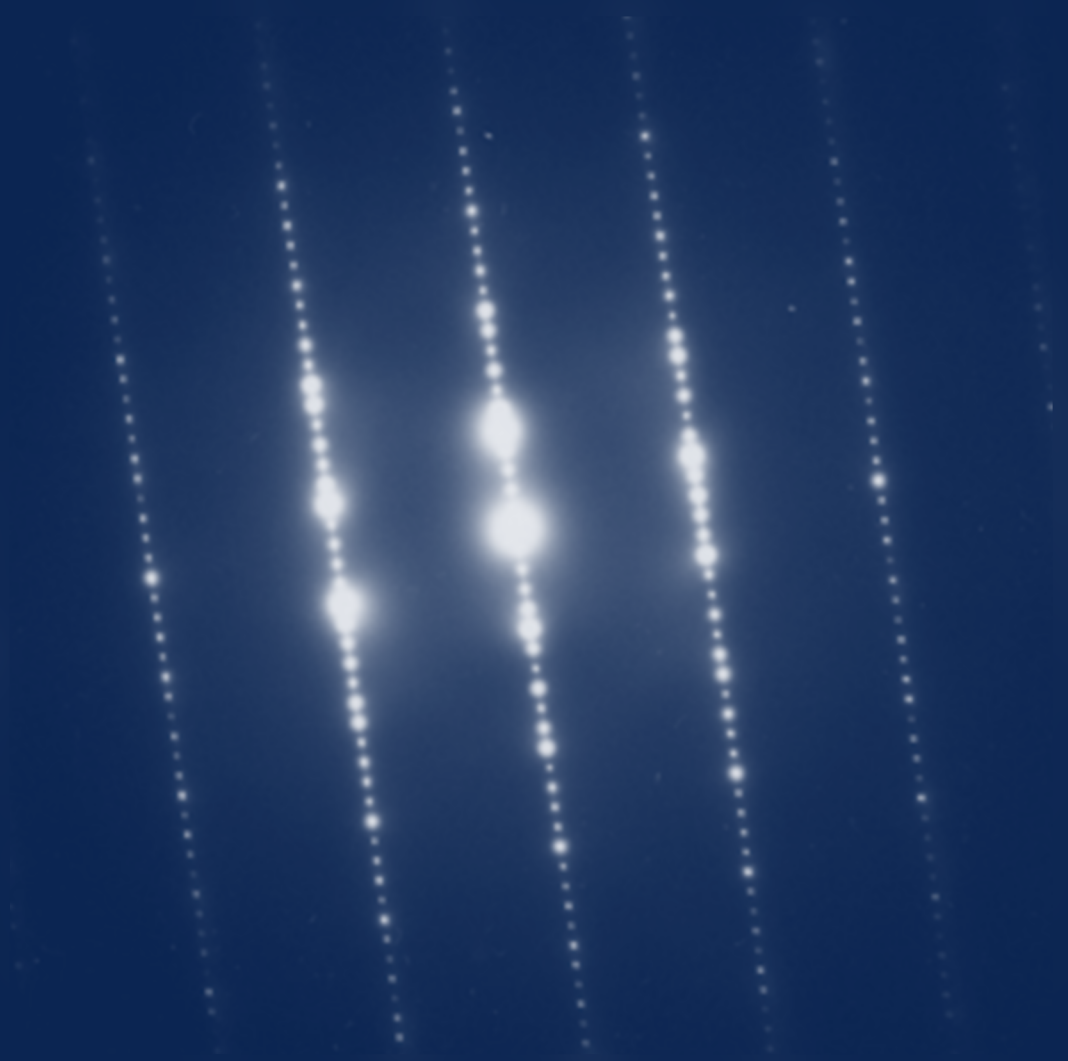
Quantum Gaussian Processes

for data-driven design of metamaterials

Gaweł Kuś



TU Delft

TU Delft

Quantum Gaussian Processes

for data-driven design of metamaterials

by

Gaweł Kuś

to obtain the degree of Master of Science

at the Delft University of Technology,

faculty of Aerospace Engineering,

to be defended publicly on Tuesday December 17, 2019 at 3:00 PM.

Supervisors:		
	Dr. ir. M.A. Bessa	TU Delft, 3ME faculty
	Prof. Dr. ir. S. van der Zwaag,	TU Delft, Aerospace faculty
Committee members:		
	Dr. B. Chen	TU Delft, Aerospace faculty
	Dr. M. Möller	TU Delft, EEMCS faculty

Student number: 4354923

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Cover picture: Britannica, The Editors of Encyclopedia. "Electron Diffraction." Retrieved from: www.britannica.com/science/electron-diffraction.

TU Delft

Summary

The data-driven approach shows great potential for the computational design of materials using machine learning and optimization. A particularly promising is the recently proposed framework relying on Gaussian processes (GP) regression - a method that intrinsically quantifies uncertainty, which allows accounting for material imperfections during the design process, ensuring the robustness of the solution. Despite its superior predictive performance, however, GPs suffer from scaling issues, which prevent this method from application to more complex design problems. This issue is addressed by sparse GPs, which allow reducing the computational cost by approximating the full GPs. The speed-up, however, is not sufficient to overcome the "curse of dimensionality", where adding new design variables results in an exponential growth of the design space.

In the near future, those limitations could be surpassed with quantum computing, by implementing the machine learning step with an exponentially faster quantum algorithm for Gaussian processes (QGP). In this case, scaling of the algorithm could match the exponential growth of the design space, allowing for approximately linear scaling of cost with the number of design variables and thus breaking the "curse of dimensionality". Consequently, this would allow for a radical increase in the number of design variables, which translates onto the complexity of the design problems that could be tackled, opening possibilities for advancements in materials science.

This research explores that concept, with the aim **to demonstrate how quantum computing could enhance computational design of materials by exploring the concept of replacing the expensive machine learning step of a data-driven design framework with an exponentially faster quantum algorithm for Gaussian processes.** This objective was achieved in two steps: 1) by implementation and simulation of the QGP algorithm, which allowed to verify its feasibility and understand its performance; and 2) by building a proof-of-concept demonstrator, where the implemented QGP algorithm was integrated within a data-driven design framework and applied to two example design problems of optimizing a unit cell for a super-compressible metamaterial.

The first step was realized by implementing the QGP algorithm as a Python code within a quantum computing framework (Qiskit). The implementation followed a recently proposed concept of the QGP algorithm, which relies on a quantum algorithm for solving systems of linear equations (HHL) to speed-up the matrix inversion in GP regression. The implemented algorithm was extensively tested using the simulator provided by Qiskit. This allowed to understand how to control the performance of the QGP algorithm and specifically its accuracy, which is an essential element for the QGP applicability.

Those numerical tests also exposed a mechanism for inducing a low-rank approximation, specifically by exploiting the approximate eigendecomposition with Quantum Phase Estimation (QPE, a subroutine of HHL) to leverage quantum principal component analysis (qPCA). This brings to light a new connection of QGP to the classical sparse GPs, which rely on low-rank approximations to reduce the computational cost. In classical computing, however, despite its superior accuracy, PCA is not a viable way for inducing sparsity in GPs, as it does not allow for reducing the computational cost (due to the cost of eigendecomposition which is required for preparing the PCA approximation). On the contrary, inducing qPCA in QGP enables additional speed-up of this quantum algorithm. Therefore, the mechanism of inducing the qPCA approximation in QGP can be considered as a new mode of sparse GPs, exclusive to quantum implementation of Gaussian processes.

Finally, the accuracy of the low-rank approximation induced with QGP was compared with the most common approximation scheme used for sparse GPs, namely the Nystrom method. For this purpose, analytical error bounds were derived and validated numerically. While the bounds alone did not provide a definite proof in favor of either QGP or Nystrom, the numerical tests clearly indicated the superiority of QGP.

The second part of the research provided a proof-of-concept demonstrator of application of QGP in the design of materials. For this purpose, the algorithm was integrated within the work-flow of a data-driven design framework, where the role of QGP was to predict the performance of the design based on provided data (training points). This framework was then applied to two example design cases of optimizing a unit cell for a super-compressible metamaterial. The complexity of the two examples was significantly limited by the computational cost of simulating the QGP algorithm, which allowed for the models with at most eight training points. Nevertheless, the examples were sufficient for verifying the feasibility of the concept.

The first design case involved the optimization of a single design parameter, to maximize the energy absorption of the cell. Prediction of the mean values with QGP was comparable to the results of the classical GP model trained on the same set of eight points. The QGP model, however, struggled with estimating the uncertainty of the predictions, yielding errors in order of 100%. The origin of this issue was identified in the approach that is used to derive the uncertainty values from the QGP results, which suffers from amplification of QGP errors, such that even relatively small discrepancies in QGP translate onto significant errors in the final predicted uncertainty. The problem of reliable prediction of uncertainty, however, is not exclusive to QGP, but is inherent also to the classical sparse GPs, although here the origins of those issues are different. In order to improve the reliability of the QGP bounds, it was proposed to add an additional noise term accounting for the QGP errors, which effects in inflating of the uncertainty intervals. While this makes the prediction more conservative, it eliminates the problem of non-physical negative uncertainty produced by QGP.

The second design case involved optimization of the unit cell for two objectives: energy absorption and critical buckling load. Furthermore, the design space was extended to two dimensions (i.e. two design variables were used). The results led to similar conclusions as in the first example: the QGP model produced a reasonable approximation of the mean values, yet struggled with reliable estimation of uncertainty. The QGP prediction from this example was also compared with results from the literature, where the unit cell design was approached with classical sparse Gaussian processes. While for critical buckling load the QGP results were in good agreement with the model from literature, there was a significant discrepancy in the prediction of the absorbed energy. In order to verify those discrepancies, the results of both QGP and the model from the literature were compared with a high-fidelity classical GP model, which was trained on a larger data-set. The QGP results turned out to be more consistent with the high-fidelity reference model, which indicates on the superiority of QGP over the classical sparse GP model in this particular problem.

Overall, the results demonstrated that the QGP algorithm can be effectively integrated within the data-driven framework and is capable of providing performance comparable to classical methods, proving the feasibility of the proposed concept. Therefore, it exhibits the potential for enhancement of the computational design of materials in the future.

Contents

1 Introduction

This work aims at demonstrating the potential enhancement of the computational design of materials with quantum computing. This is achieved by implementing a quantum algorithm for accelerating Gaussian processes regression – a machine learning method that plays an essential role in building the material response model in a recently proposed computational framework for data-driven design of materials.

Background

Computational materials science is currently undergoing a paradigm shift towards data-driven design. Unlike conventional models based on physical principles, in this novel approach the models rely on machine learning algorithms. Gaussian processes (GPs) have proven to be a particularly relevant machine learning algorithm within the context of materials and metamaterials design. The intrinsic capability for quantifying the uncertainty distinguishes GPs from other machine learning methods, while making them well suited for modeling imperfection-sensitive phenomena and perform optimization under uncertainty – the challenges faced pursuing robust design of materials. Despite its superior predictive performance, Gaussian Processes have limited application due to scaling issues.

This problem is classically addressed by sparse Gaussian processes, which by constructing an approximation to the full GP regression, reduce the computational cost and extend the applicability to larger data-sets by a few orders of magnitude. This significant improvement, however, can be dwarfed by the so called "curse of dimensionality": an exponential growth of the design space with the increase of design parameters. In other words, although sparse Gaussian processes allow for larger design spaces than conventional Gaussian Processes, they still face hard limits in dealing with many input variables.

In the near future, those limitations could be surpassed by quantum computing. This novel model of computation allows for performing calculations by exploiting phenomena of quantum mechanics, which enables carrying out certain operations exponentially faster than classically. This applies also to solving systems of linear equations, which is the origin of scalability issues in GP's. This connection led to a recently proposed concept of quantum algorithm for Gaussian processes (QGP), where the matrix inversion is accelerated by a quantum algorithm for solving systems of linear equations (HHL), to reach unprecedented speed-up over classical GP's. Harnessing this improvement within the context of data-driven design of materials could provide a way of overcoming important classical limitations.

Aim of the research

This research aims at verifying the feasibility of this concept, which is done in two steps. The first part of this research focuses on verifying the feasibility of the QGP algorithm. This is done by implementation and simulation of the algorithm within a quantum computing framework (Qiskit). The numerical tests aim to gain understanding about the algorithm performance, with particular focus on its accuracy such that it could be controlled in a reliable way – essential for practical applications. Furthermore, potential improvements to the QGP algorithm are explored, following the concepts of sparsity and low-rank approximations, used in classical Sparse Gaussian Processes.

In the second part of this work, the QGP algorithm is integrated within the work-flow of a data-driven approach to design a super-compressible metamaterial with a previously investigated geometry. Two example problems of optimizing the unit cell are considered, involving a one-dimensional and a two-dimensional design space, respectively. This approach aims to verify the feasibility of applying QGP as an integral part of the data-driven framework, as well as testing its performance in practical settings. This provides a proof-of-concept demonstration of the quantum-enhanced data-driven framework, despite the infancy of quantum computing hardware and the limitations it imposes on solving practical problems.

Outline

This thesis is structured as follows. First, Chapter 2 provides a background on three fields essential for this research: metamaterials, machine learning and quantum computing. Aside from providing the essential background, the review highlights the relations between those three different domains, and their relevance for this research.

Chapter 3 presents the results of the first part of the research. The chapter guides through the process of implementation and testing of the QGP algorithm. The numerical results of the algorithm performance tests are explained with theoretical insights on the algorithm. The key findings of this part are discussed in Chapter 4. This chapter also reflects on why the obtained results contribute to the advancement of the field of quantum computing and quantum algorithms.

The results of the second part of the research are presented in Chapter 5, which guides the reader through the optimization process of the metamaterial unit cell enabled by the data-driven framework using QGP. The QGP predictions are also compared with classical models, in order to evaluate QGP performance. The findings are discussed further in Chapter 6, which provides a comparison of the QGP predictions with the classical sparse Gaussian processes models from literature, but also evaluates the QGP performance within the framework and proposes several improvements. The thesis is concluded with Chapter 7, which summarizes the reached conclusions, and lists recommendations for the future work.

2

Literature review

The proposed research bridges three different topics, namely: materials science, machine learning and quantum computing. This literature review provides essential background on those three different disciplines. The chapter starts with Section 2.1 introducing metamaterials and the importance of Gaussian processes for their data driven design. Then Section 2.2 reviews this machine learning method and discusses its scalability issue. Section 2.3 provides an overview on quantum computing and Section 2.4 discusses why this computing paradigm offers a promising route to improve machine learning in general, and Gaussian Processes in particular.

2.1. Machine learning in metamaterial design

Materials design usually follows a trial-and-error process that insofar has relied on experimentation as a means to probe the design space. Since most materials design problems involve complex phenomena and a large number of variables, creating new materials by this methodology is too time consuming. Instead, the design process can be accelerated by a computational data-driven approach where machine learning methods play a crucial role. The full advantage of this novel approach can be revealed in the design of metamaterials, which is a particularly demanding task.

This chapter starts with subsection 2.1.1, which gives an overview on the mechanical metamaterials and discusses the key challenges of their design. subsection 2.1.2 presents a state-of-the-art data-driven framework for computational materials design. Finally, subsection 2.1.3 highlights the connection between the metamaterials and the data driven approach by presenting a recent research project where the data-driven framework relying on sparse Gaussian processes was applied to design a metamaterial unit cell that achieves super-compressibility.

2.1.1. Introduction to metamaterials

Metamaterials are a novel class of materials with unusual properties originating from carefully engineered architectures [7, 15, 35, 37] ranging from nanometer to meter scale [35, 42]. Those structures often comprise of repetitive patterns or lattices of unit cells which act in a collective manner. The cells are commonly composed out of slender elements, which enable non-linear behavior at the macroscopic scale [7, 22].

In general, metamaterials can be divided into several classes based on their properties, including electromagnetic [37], acoustic or mechanical metamaterials [15, 35]. This research focuses on the

last class i.e. the mechanical metamaterials, as their properties make them the most attractive for aerospace applications for instance as deployable or shape-morphing space structures, energy absorbers or vibration dampers [7, 15, 60]. This class of metamaterials can be subdivided further into several subclasses, namely: instability-based, linear, topological or mechanism-based metamaterials [7], which are discussed in more detail in the following paragraphs.

Linear Metamaterials

In linear metamaterials, the focus is on altering the elastic tensor, which enables uncommon mechanical properties [7]. One example are the auxetic metamaterials i.e. the metamaterials with negative Poisson's ratio [39, 42, 63]. In this case, the structure is designed in such way that upon loading in tension or compression the material respectively elongates or contracts in all directions. The effect is shown in Figure 2.1, which compares the response of the auxetic metamaterial (the rightmost one) with the zero Poisson's ratio metamaterial (the one in the center) and a common elastic material with positive Poisson's ratio. Linear metamaterials include also examples of negative elastic moduli [15, 50], in which case the material elongates upon compression, and contracts while subjected to tension.

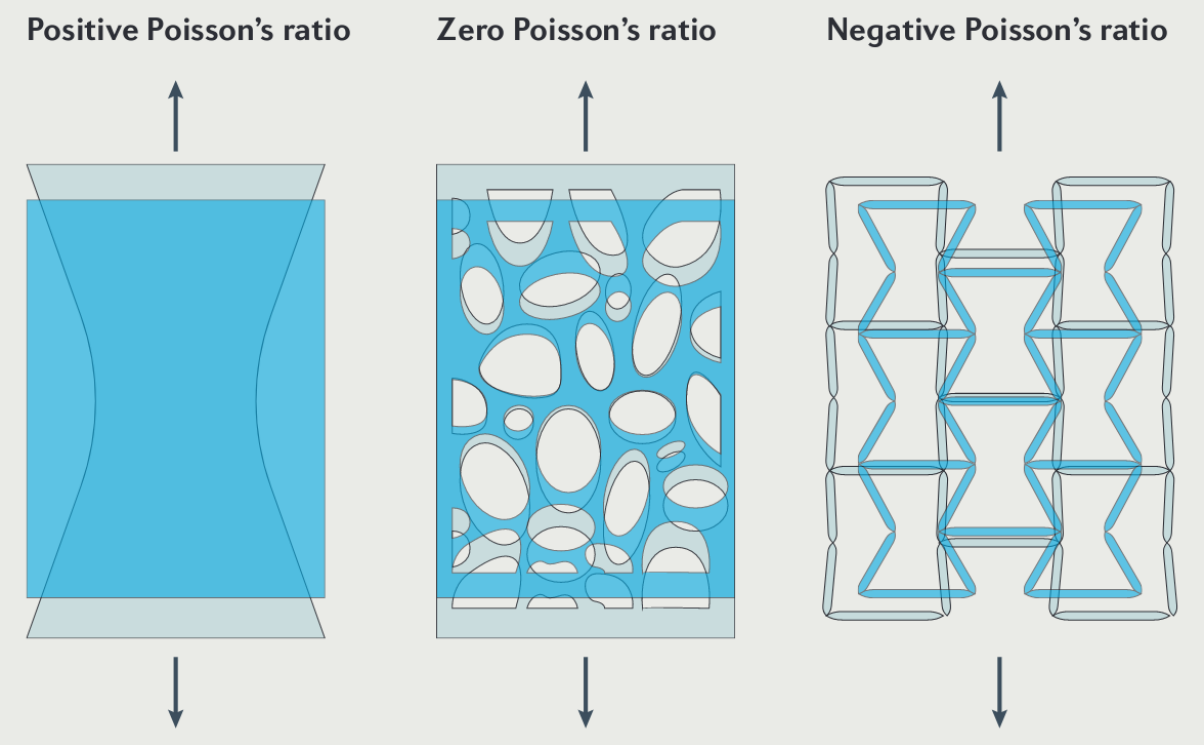


Figure 2.1: Schematic comparison of the response of metamaterials with different Poisson's ratio [7].

The second interesting example are pentamode metamaterials. In this case most of the coefficients in the elastic tensor matrix are close to 0, which results in a diverging ratio of bulk to shear modulus [7, 15]. Consequently, the material resists only against the volumetric deformation, and does not respond to shear [7]. In that sense it behaves like a fluid, which is the reason why those materials are also called meta-fluids [7, 15]. Finally, engineering the metamaterial architecture enables extensive control over its anisotropy, which allows for tuning the vibrational response [15].

Mechanism-based Metamaterials

Mechanism-based metamaterials rely on unit cells linked via elements enabling free-motion, such as hinges [7]. Those features are most often employed to achieve reversible shape-changing morphologies, with example applications for deployable space structures [15]. Those shape-changing mechanisms often rely on bistability (which is discussed at the end of this section), which enables the material to reversibly transform into one of its stable configurations. [15].

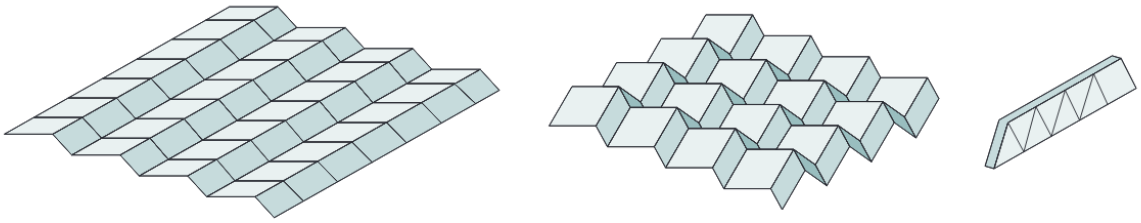


Figure 2.2: Concept of a 2D origami-inspired metamaterial. The square unit cells linked by hinges at the edges enable a foldable structure [7].

Especially common in literature are the concepts inspired by origami and kirigami [15]. In both cases, the material is usually in the form of two-dimensional sheets. The origami-based materials usually consist of unit cells connected with hinges that allow folding and consequently, dramatic changes of shape of the metamaterial. An example of such a foldable metamaterial is shown in Figure 2.2. In kirigami-inspired materials, the unusual properties are enabled by carefully designed arrays of cuts. In such way, the metamaterial can achieve for example larger strains [7].

Topological Metamaterials

The key concept of topological metamaterials is to provide robustness via the design of topology [7, 54]. An example of such concept are topologically protected metamaterials utilizing selective buckling, that were investigated by Paulose et al. [54]. The researchers designed a 3D cellular metamaterial shown in Figure 2.3, where the topology ensures the piling-up of localized states of self-stresses to ensure buckling in certain regions of the structure (marked in the figure with the dashed line). Paulose et al. argue that in terms of node connectivity and materials parameters the buckling-prone regions of the material remain indistinguishable from the other region, thus not affecting for instance thermal or electrical conductivities. The design also provides a certain amount of robustness against structural perturbations.

Stability-based Metamaterials

Stability-based metamaterials are one of the most intensively researched groups of metamaterials. The key role in this concept play slender elements, which allow for large deformations and buckling resulting in tunable non-linear mechanical responses [7, 15, 60]. Specifically, the stability-based metamaterials exploit the phenomena of bistability. In this case the metamaterial cells have multiple stable states and transform from one state to the other given some external stimuli e.g. under load. The transformation between the states usually rely on snap-through mechanisms, which employ instabilities of the structure.

An example of such snapping metamaterial is shown in Figure 2.4, which was investigated by Rafsanjani et al.[60]. As shown in the figure, while subjected to tension the metamaterial undergoes a sequential series of snapping events, during which a row of unit cells transforms from wavy to

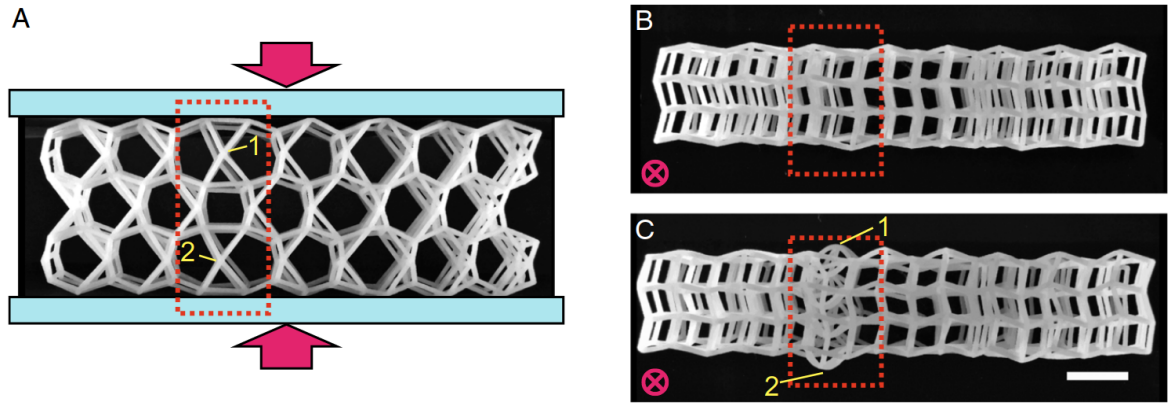


Figure 2.3: A sample of a 3D topological cellular metamaterial designed by Paulose et al. [54]. The buckling region was marked with the red dashed line. Picture A: the top view with the compression direction marked by the arrows. Pictures B and C: view along the compression axis, before and after buckling respectively.

diamond-shaped. The drops in the stress-strain curve correspond to the snapping instabilities experienced during loading. Comparing the three different samples, one can see that the response can be tuned by varying the aspect ratio of the cells (a/l parameter). The potential applications of such metamaterials include shape morphing, energy dissipation or damping vibrations.

Challenges

Metamaterial's performance results from various mechanisms at different length-scales, spanning intrinsic material properties at the nano-scale to structural arrangements at the macro-scale [35, 42]. In that sense metamaterials are similar to composites, where the properties are also a combination of intrinsic characteristics as well as macroscopic features (such as fiber layout). The need for modeling such a combination of different phenomena makes design and optimization of metamaterials particularly complex and challenging. Unlike composites, however, metamaterials exploit mechanisms in much more extreme regimes including buckling and bistability. Furthermore, freedom in the design of structural architectures makes the design space vast, which adds to the overall complexity of the problem, resulting in significantly higher computational costs, and thus motivating the need for more efficient design methods.

2.1.2. Data-driven design & the role of machine learning

Conventional approaches in computational materials design rely mostly on modeling the underlying physics, with the aim to predict properties given the composition, microstructures etc. An example can be molecular dynamics which models the interactions between molecules, or DFT calculations which predict the electronic structure of materials from quantum mechanical calculations. However, those models are often computationally too expensive for exploration of the high-dimensional design spaces prevalent in materials design problems.

Therefore, the research community is shifting to an alternative paradigm, namely the data-driven approach [34], employing methods such as genetic algorithms, topological optimization or machine learning [9] to mention a few. The key feature of this approach is the autonomy of the algorithms in making the decisions on probing the design space, which enables efficient and optimized exploration rather than exploitation of the design space. Secondly, machine learning enables capturing the complex relationships between the design parameters, allowing to gain insight onto underlying

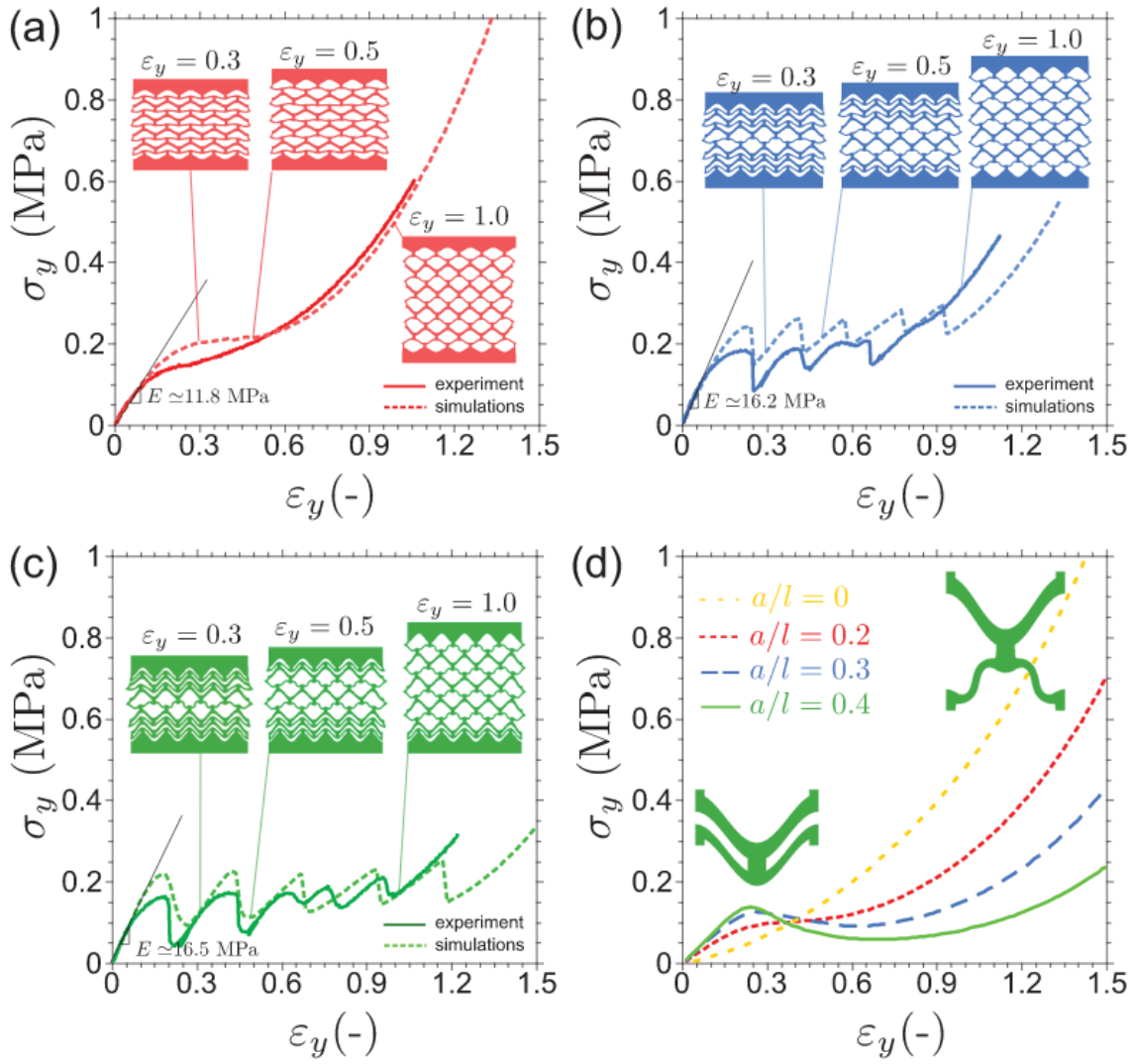


Figure 2.4: Response of snapping metamaterial designed by Rafsanjani et al.[60] under tension. Plots a) b) and c) correspond to samples with a different aspect ratio of the unit cells.

mechanisms, while outperforming the traditional methods.

Framework for data-driven design of materials

A particularly relevant example of such a data-driven framework is the concept developed by Bessa et al. [9]. The proposed framework was designed for accelerating the design of complex materials systems, specifically the design of composites.

The proposed approach integrates three steps, namely: Design of Experiments (DoE), computational analyses and machine learning, as shown schematically in Figure 2.5. In the first step, the design space is defined by a set of descriptors, and the experiments are defined, i.e. the design space is sampled. For instance, in Figure 2.5 three classes of descriptors are defined: the parameters of microstructures, the material properties, and the external conditions. To boost the computational efficiency and minimize the number of samples, the researchers propose to sample the design space with unconventional schemes, namely the Sobol sequence [9]. The second step aims to generate a database for machine learning. For this purpose, a number of computational analyses are performed at the previously defined sample points of the design space. In the last step, a machine learning al-

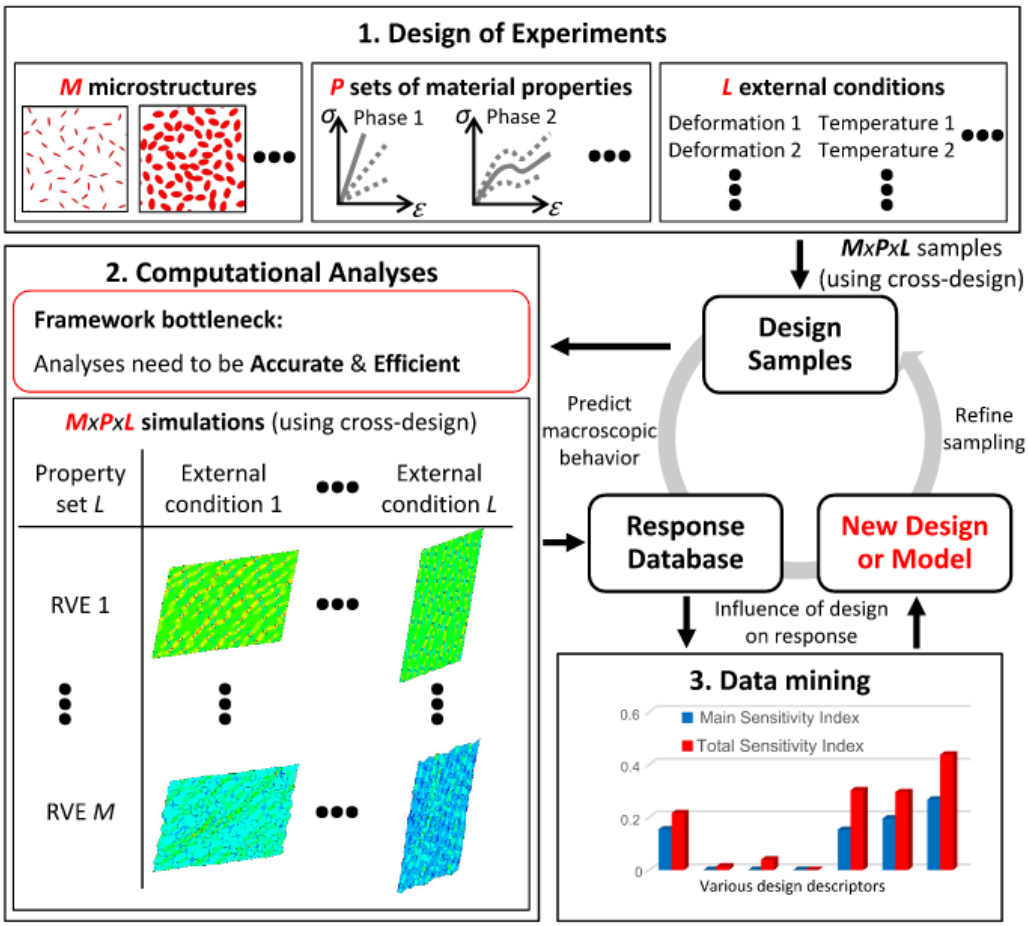


Figure 2.5: Schematic structure of the data-driven design framework developed by Bessa et al. [9].

gorithm is applied to the database to find out the relationships between the descriptors. In Bessa's work, this step relies on either Gaussian processes or neural networks, which are used to define the model.

The performance of the proposed framework was presented by applying it to two example problems of composite design. In the first problem, the researchers aimed *"to find a constitutive law for two-dimensional hyperelastic composites reinforced by multiple elliptical particles"*[9]. The complexity of the problem was additionally heightened by considering the response under large deformations. In the second example the framework was employed to determine the influence of phase parameters on the toughness of a three-dimensional composite [9]. Both examples address challenges such as choice of the sampling scheme, the significance of uncertainty, or choice of different machine learning algorithms, but also prove the advantage of data-driven approach in a practical setting.

In follow-up research, the proposed framework was extended by the fourth step, where the inferred model was subjected to optimization to find the design with the best performance [8]. This extended framework was applied to the design of Triangular Rollable And Collapsible (TRAC) boom for a spacecraft. To facilitate efficient optimization a Surrogate model was used (which will be discussed in the next section).

The role of machine learning

As shown in the aforementioned examples, the primary role of machine learning is to infer the relationships between the parameters, and describe those dependencies by building approximate models. Those models can be then directly used for the design, but also for discovering constitutive laws governing the investigated phenomena [9].

The machine learning algorithm that builds the model is oftentimes coupled with an optimization routine, which usually relies on some machine learning algorithms as well. A common way to combine the model building with the optimization is via so-called Surrogate-Based Analysis and Optimization approach (SBAO) [58]. In this scheme, the surrogate model is created using machine learning that provides a rough approximation based on a small number of data-points. Then based on the developed model the optimization algorithm decides which regions of the design space are the most attractive and require refinement of sampling. The model is then updated, and the loop is repeated. The approach enables efficient exploration of the design space and proved successful for instance in the optimization of the aforementioned TRAC boom [8].

Intrinsic features of the machine learning-based approach provide with three key benefits, namely: computational efficiency, more effective optimization, and quantification of uncertainty.

Computational efficiency The machine learning approach is often successfully used for tasks that are computationally intractable with traditional approaches. Although the algorithms themselves are often computationally very expensive (take for instance Gaussian [8]), their use allows for significant reduction of the overall costs of the design tasks by developing the approximate models and accelerating the optimization. An impressive example of such speed-up is the work by Meredig et al. [47] where the machine learning was applied to explore a database of DFT calculations of over 1.6 million compositions, with the aim to predict stable compositions of ternary compounds, in this case, machine learning allowed to make the predictions six orders of magnitude faster, than if it was done with the conventional calculations.

Effective Optimization The machine learning approaches have proven outstanding effectiveness in optimization. One way the machine learning can contribute to finding the optimum is by capturing the complex relationships between the different parameters by developing a model, which can be then fed into an optimizer to find the best solution. On the contrary, in conventional approaches, those dependencies remain beyond comprehension or are occupied with preventively high costs. The second way is to apply the machine learning algorithms directly as optimizers to enhance the exploration of the design space. In practice, data-driven frameworks usually benefit from the combination of both contributions. However, it should be mentioned that there is a trade-off between the number of function evaluations needed to build the machine learning model that is trying to learn the entire response within the sampled space, and the function evaluations necessary to reach a local or global optima. If one intends to find a single optimum solution, direct optimization without building a surrogate model can be a faster approach.

Most importantly, the data-driven approach has proven to outperform the conventional methods both in terms of speed as well as the optimality of the results. An example is a research by Liu et al. [41], where machine learning was applied to design a microstructure of a magneto-elastic Fe-Ga alloy for the enhanced mechanical and magnetic properties. While the alloy properties could be easily calculated using existing analytical models, provided the microstructure, Liu et al. turned the problem the other way around, with the aim to develop a model that predicts the microstructures able to produce the desired properties. The problem stated in such a way would be traditionally

attempted with search-based optimization, however, in high-dimensional microstructure space, this approach is ineffective. The researchers prove that using machine learning instead results in more optimal results, while also reducing the computing time by 80%.

Another example is the previously discussed design of the TRAC boom [8]. In this case, the model developed with the data-driven approach allowed to improve both buckling moments (which could be considered as the overall performance of the boom) by outstanding 25%. Furthermore, it disclosed different optima, which allowed to optimize for several different objectives with comparably significant potential improvements.

Aside from the remarkable performance, the models developed by machine learning are particularly well suited for facilitating optimization, as briefly discussed by Bessa et al. [8]. For instance, those models enable optimization over noisy data and sensitivity analysis. Furthermore, their generality allows for running different optimization routines aiming for different objectives, without need for additional refinements [8].

Quantification of uncertainty The uncertainty quantification is crucial, though often neglected element of data-driven approach. Its impact affects both predictive and computational performance.

Bessa et al. addressed the importance of uncertainty from the point of view of the design of materials [8]. Specifically, in certain cases of mechanical design, the response of a structure might be particularly sensitive to imperfections and uncertainties. In such cases accounting for uncertainty is essential for predicting the response. While the conventional approaches often struggle with this problem [8], the intrinsic quantification of uncertainty provided by Gaussian Processes makes the method perfectly suited for this purpose.

Lookman et al. [45] present a view from a different perspective. In their research Lookman et al. employed Gaussian processes to discover a new NiTi-based alloy with the smallest thermal dissipation, and used the uncertainty quantification to enhance the exploration of the design space. Specifically, the uncertainty was used to optimize the process of sampling refinement, and the trade-off between the regions with potentially higher reward vs regions with more predictable outcomes. Optimizing this decision process allowed for reducing the number of experiments necessary for reaching the most optimal design, and improve overall efficiency.

2.1.3. Gaussian Processes & metamaterial design

Recently, sparse Gaussian processes were used to facilitate the design and understanding of a new metamaterial that achieves super-compressibility [24]. The work aims to investigate the mechanical response of a metamaterial unit cell and determine its dependency on different design parameters.

The unit cell of the proposed metamaterial consists of two rings connected by a number of longerons (in this research the number of longerons was constrained to three). The longerons are connected with the rings via hinges, which plays a crucial role in buckling. The configuration of the unit cell is parametrized by pitch P (defined as shown in Figure 2.6) and the ratio of the diameters, defined as: $D_R = \frac{D_1 - D_2}{D_1}$. The intrinsic properties of the cell material are specified by means of modulus ratio: $\frac{G}{E}$ (where the values were bound to ensure reasonable ranges).

Furthermore, the design of longerons was parametrized with the following factors:

1. Cross-section area A
2. Second moment of inertia around x axis I_x

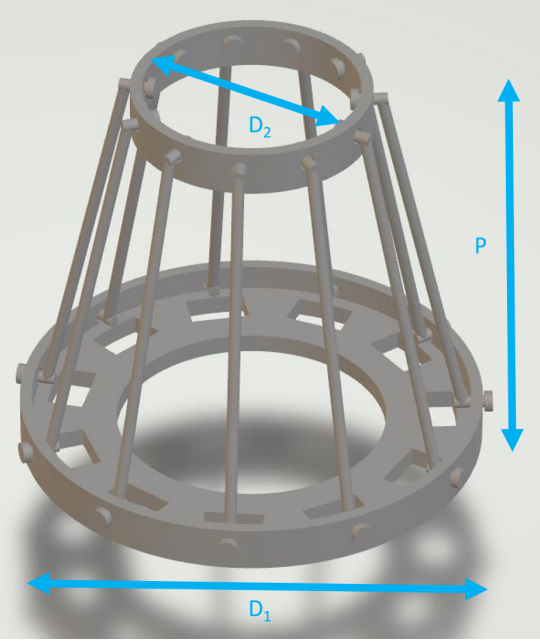


Figure 2.6: Unit cell of the proposed super-compressible metamaterial [24].

3. Second moment of inertia around x axis I_y

4. Torsion constant of the cross-section J_T

The problem was approached following the steps of the data-driven framework described in subsection 2.1.2, where sparse Gaussian processes were the machine learning method of choice. The research demonstrated the applicability of this method in a practical setting. The model proved successful in capturing the relations between the design parameters and the cell response, providing overall good predictive performance. The model together with the sensitivity analysis provided essential insight into the tunability of the cell response and allowed to determine the most significant parameters influencing the performance (i.e. the absorbed energy and the critical buckling load).

The next suggested step extending this research is to carry out the optimization for energy absorption. In a more advanced case, a multi-objective optimization approach is proposed, with the aim to maximize the absorption while minimizing the local strains [24].

2.2. Gaussian Processes

The Gaussian process (GP) method is a non-parametric Bayesian model used for machine learning and reinforcement learning [40]. The most unique feature that distinguishes GPs from several machine learning algorithms is that aside from predicting the mean response, GPs provide the associated uncertainty. This is particularly important to metamaterial design, as seen in subsection 2.1.3. The fundamental concepts behind GPs are introduced in subsection 2.2.1, including their most important limitation: scalability. Current solutions for improved scalability are presented in subsection 2.2.2.

2.2.1. Fundamental concepts

Given some observation data, one is interested in learning (inferring) the underlying function that governed generating this data, such that given an arbitrary input one can predict the value of the function. Gaussian Process (GP) model approaches this problem with Bayesian statistics. In principle, GP assumes a prior distribution over the latent function [1], and then uses conditioning on the observed data to convert the prior to the posterior distribution, which allows predicting the function values. The process is shown in Figure 2.7, and will be explained in more detail in the remaining part of this section.

GP model in principle specifies a distribution over all possible functions that might be a solution. Although defining a distribution over an infinite number of functions might appear as an intractable task, it can be completely specified by merely its mean m and covariance function K, as in Equation 2.1.

$$f(x) \sim GP(\mu(x), K(x, x')) \tag{2.1}$$

Furthermore, it should be emphasized that although the functions are defined over a continuous domain, it is sufficient to define the distribution only over some finite set of points x (functions don't need to be specified with a formula) [49].

Kernel function

One of the key concepts in GPs is the covariance function K, also called the kernel, which provides a measure of similarity between the points. The most commonly chosen kernel is the squared exponential function [75] in a form shown in Equation 2.2.

$$K(x, x') = \sigma_f^2 exp\left(-\frac{1}{2l^2}(x - x')^2\right) \tag{2.2}$$

As shown in Equation 2.2, this kernel function depends also on two hyperparameters: vertical variance σ_f and length-scale parameters l. Those parameters allow for a certain degree of tuning, and have a significant impact on the predictive performance of GPs, for example by controlling the smoothness of the functions with the length-scale parameter l [49].

The kernel is tuned using the observed data, using the empirical Bayesian approach - by maximizing log marginal likelihood [49, 75]:

$$log\, p(y|X) = -\frac{1}{2} y^T (K + \sigma_n^2 I)^{-1} y - \frac{1}{2} log|K + \sigma_n^2 I| - \frac{n}{2} log 2\pi \tag{2.3}$$

The most common approach is to take derivatives with respect to the hyper-parameters and use a gradient descent approach to find the optimal set.

Inference

By definition "*a Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution*". As a consequence GP assumes that the function values f_* that we want to predict at some test locations x_* are jointly distributed with the observations f at points x,

[1] latent function - the inferred function

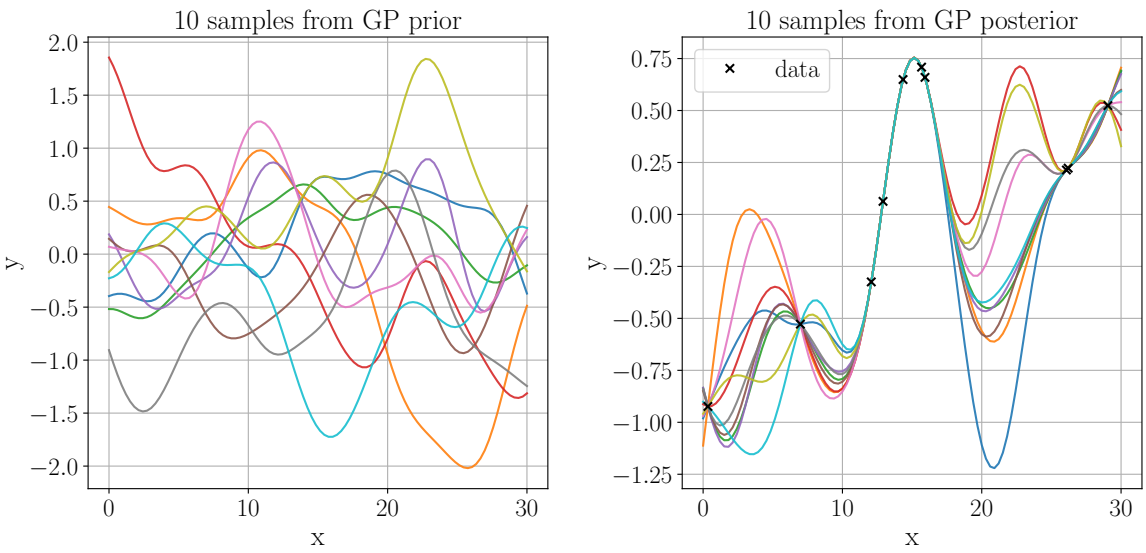


Figure 2.7: Function samples drawn from prior GP distribution (on the left), and from the posterior (on the right) conditioned on the training points (marked as black dots). In this example the data was assumed to be noiseless.

according to some assumed prior distribution with mean $m = 0$ and covariance given by the covariance function K. This joint distribution can be expressed as in Equation 2.4.

$$p(f, f_*) \sim N\left(0, \begin{bmatrix} K(x,x) & K(x,x_*) \\ K(x_*,x) & K(x_*,x_*) \end{bmatrix}\right) \tag{2.4}$$

Then the definition of GPs enables to use the marginalization property, which allows to derive the predictive distribution. In particular, the predictive distribution is obtained by conditioning on the observations (f at x), which can be expressed as:

$$p(f_*|x_*, f, x) \sim N(K(x_*,x)K(x,x)^{-1}f, K(x_*,x_*) - K(x_*,x)K(x,x)^{-1}K(x,x_*)) \tag{2.5}$$

The process can be understood by looking at Figure 2.7. The plot on the left-hand side shows sample functions drawn from the prior distribution, given by Equation 2.4. One can see that the values of the plotted functions oscillate around the mean given by the distribution $m = 0$, while their smoothness is controlled by K. The process of conditioning on the data can be understood as the elimination of all those functions that do not pass through the data-points. The remaining functions, as shown on the right-hand side, follow the posterior distribution given by Equation 2.5. This distribution allows for making predictions. One can notice that the larger the spacing between the data points the larger the spread of function values which is related to the uncertainty.

In real cases, however, the data is affected by some noise, which often can be assumed to follow some normal distribution:

$$y_i = f(x_i) + \epsilon \qquad \epsilon \sim N(0, \sigma_n^2) \tag{2.6}$$

To account for that a so-called jitter term is added to the kernel function in the diagonal elements of the covariance matrix corresponding to the observations: $K(x,x) + \sigma_n^2 I$. In that case, the posterior

predictive distribution is given by following mean and variance [75]:

$$m = K(x_*, x)(K(x, x) + \sigma_n^2 I)^{-1} y \qquad \Sigma = K(x_*, x_*) - K(x_*, x)(K(x, x) + \sigma_n^2 I)^{-1} K(x, x_*)) \tag{2.7}$$

Computational complexity

The problem of Gaussian Processes is its cubic computational complexity $O(n^3)$ [40, 49, 59]. This means that the practical applications are limited to $O(10^4)$ data-points [40], as larger sets become intractable.

2.2.2. Approximation methods: improving scalability

In response to the scalability issues, major research efforts were directed into devising a number of approximation techniques to reduce the computational costs. All the proposed concepts for scalable GPs rely on a single idea i.e. to summarize the relationships from the data with fewest points, such that the size of the kernel matrix can be reduced, and the computational effort minimized.

This led to the emergence of different classes of scalable GPs as discussed in detail by Liu et al. [40]. The division is schematically shown in Figure 2.8. On the highest level, the approximations are divided into local and global. The local approach is based on divide and conquer strategy. The domain is divided, and only the data-points from the vicinity of the queried point are taken, assuming that the influence of further points can be neglected. In the global approach the data is approximated over the whole domain. Here either a subset of data-points is chosen, or new pseudo-data points are introduced in convenient places resulting in Sparse approximations.

Furthermore, a number of alternative concepts were proposed as well, such as for instance stochastic approaches or parallel computing on GPUs – which are particularly suitable for local approximations. The details can be found in [40].

Local vs. global approximations

Local approximations are based on a divide and conquer strategy. This means that the data is divided into smaller subsets, often taking only points from the vicinity of the queried point. The approach is based on the assumption that data-points located far away have negligible impact, which is backed by the design of the radial basis function (RBF) kernel function.

The simplest local approximation class is called naive local experts. One approach is to divide the data in advance and define a local expert at each partition. Then a prediction at a given point depends on the local expert corresponding to the given partition. This approach is called inductive. Alternatively, the partitioning and training of experts can be done dynamically, such that given a query point the closest vicinity is chosen, which is called transductive.

Chunking the data with local experts can provide significant speedups. Furthermore, the setting allows for easy parallelization. However, the predictive performance can be severely limited [40]. The most common issues are discontinuities at boundaries of adjacent partitions, as well as poor prediction of global trends.

There are also plenty of more advanced local approximations, including classes such as mixture of experts and products of experts. Those approaches, however, are of insignificant relevance for this project, therefore for more details the reader is referred to the overview given by Liu et al [40].

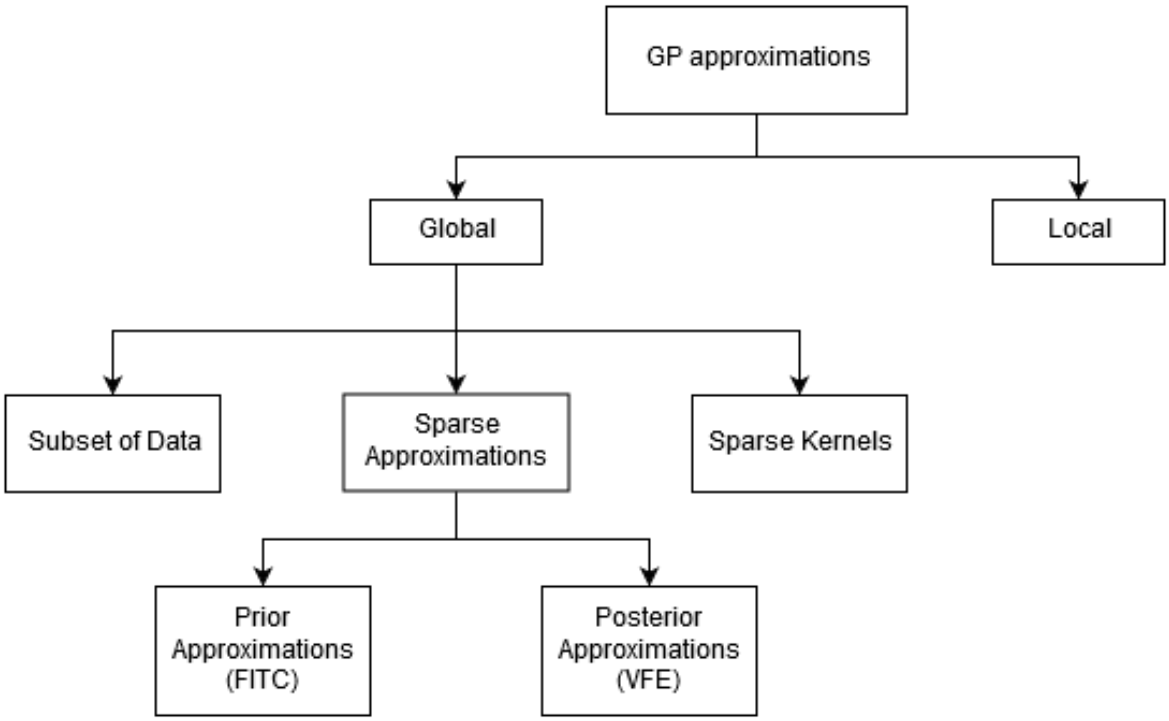


Figure 2.8: Overview of different classes of approximations of Gaussian Processes.

Global approximations aim to approximate the data over the complete domain. This approach is therefore especially useful for applications such as data-driven design, where the global trends are of much higher interest than local variations. Two of the most advanced Sparse Gaussian Processes models are Fully Independent Training Conditional (FITC) and Variational Free Energy (VFE). Therefore the focus of the following sections will be on those two methods. Nevertheless, a number of other scalable methods will be briefly discussed as well.

Subset of Data and Sparse Kernels

The possibly most primitive approach for inducing sparsity of the kernel is by deleting small (thus less important) entries [40]. The method is usually implemented via so-called Compactly Supported kernel (CS) function, which cuts-off the values below a certain threshold. The main challenge of this approach, however, is to ensure that the approximating matrix is positive semi-definite [40]. The method provides only modest speedup, as the time complexity is in order of $O(\alpha n^3)$, where α is a factor between 0 and 1 [40].

Another naive approach is to reduce the kernel size by choosing only a subset of m data-points (Subset of Data – SoD). The kernel matrix is then $K_{m\times m} = k(x_m, x_m)$. This results in the improvement of scalability on the order of $O(m^3)$. For choosing the subset of points there are two common approaches reported in the literature: either randomly, or by clustering techniques [40]. In the second case, methods such as K-means clustering are commonly used [40]. More advanced concepts employing learning criteria are mentioned as well [40], however, their computational cost is already much higher. Performance-wise the SoD is able to produce reasonable predictions of the mean, however, it struggles with variance due to reduced data-set [40].

Alternatively, the covariance matrix can be also approximated using the Nystrom approach, which relies on approximating eigenfunctions [76]. In this case the approximate covariance matrix is de-

fined as: $\tilde{K} = K_{nm}K_{mm}^{-1}K_{mn}$, where K_{mm} is the reduced matrix. The approximate matrix is therefore still of size $n \times n$, however, such form enables use of a Woodbury formula [76], and as a result the size of the matrix that needs to be inverted is reduced to $m \times m$. The Nystrom approximation is also a foundation of the 2 more advanced methods – FITC and VFE [4] – which are discussed in the next sections.

Prior approximation: FITC

In 2006 Snelson and Ghahramani [69] proposed a completely novel approach for inducing the sparsity by introducing pseudo data. In their formulation the standard Gaussian Processes system becomes augmented with an additional artificial set of data Z (and corresponding pseudo-observations u) that does not actually belong to the training set. The prior of the pseudo-data can be defined in a standard way with the assumed mean $\mu = 0$, as shown in Equation 2.8 [69]:

$$p(u|Z) = \mathcal{N}(u|0, K_{uu}) \tag{2.8}$$

The joint prior $p(y, f_*)$ is then approximated by Equation 2.9:

$$p(y, f_*) \approx q(y, f_*) = \int p(y, u)q(f_*, u)p(u)du \tag{2.9}$$

Furthermore, assuming that the data y is generated following i.i.d. assumption (independently, identically distributed) [69], the predictive distribution for a new test input x_* can be derived by integrating the likelihood over the inducing inputs u. The detailed derivation can be found in [69]. The final predictive distribution is given by Equation 2.10:

$$p(y_*|x_*, X, Z) = \mathcal{N}(K_{*u}Q_M^{-1}K_{uf}(\Lambda + \sigma^2 I)^{-1}y, K_{**} - K_{*u}^T(K_{uu}^{-1} - Q_M^{-1})K_{*u} + \sigma^2 I) \tag{2.10}$$

Where the mean and the covariance are given by Equation 2.11 and Equation 2.12 respectively.

$$\mu_* = K_{*u}Q_M^{-1}K_{uf}(\Lambda + \sigma^2 I)^{-1}y \tag{2.11}$$

$$\Sigma_* = K_{**} - K_{*u}^T(K_{uu}^{-1} - Q_M^{-1})K_{*u} + \sigma^2 I \tag{2.12}$$

Where Q_M and Λ are given by Equation 2.13:

$$Q_M = K_{uu} + K_{fu}(\Lambda + \sigma^2 I)^{-1}K_{uf} \qquad \Lambda = diag(K_{ff} - K_{fu}K_{uu}^{-1}K_{fu}^T) \tag{2.13}$$

The predictive mean and covariance in the above formulation require inverting K_{uu} and Q_M which are of size $m \times m$, instead of the full covariance matrix K of size $n \times n$, which clearly shows the advantage of this formulation.

Furthermore, in the FITC model the pseudo-data points u can be treated as hyper-parameters therefore their position can be freely adjusted to optimize the performance e.g. by maximizing log marginal likelihood via gradient descent, just like with any other hyper-parameter optimization. In such a case, the objective function is given by Equation 2.14.

$$F = \frac{N}{2} log(2\pi) + \frac{1}{2} log|K_{fu}K_{uu}^{-1}K_{fu}^{T} + \Lambda + \sigma_n^2 I| + \frac{1}{2} y^T (K_{fu}K_{uu}^{-1}K_{fu}^{T} + \Lambda + \sigma_n^2 I)^{-1} y \tag{2.14}$$

Optimizing the approximate model, however, does not always correspond to better performance of the exact model. Therefore optimizing Z might not necessarily lead to better location, but rather to over-fitting, which is a common problem of FITC [4, 40, 59]. Much deeper insight into the performance of FITC and comparison with alternative approaches was given by Bauer et al. [4].

Posterior approximation: VFE

An opposite approach was proposed in 2009 by Titsias [71]. His Variational Free Energy (VFE) concept takes the exact model and approximates the inference [71]. Similarly as FITC, the VFE model relies on pseudo data Z with corresponding pseudo-observations u. The full derivation can be found in the original paper [71]. The key point, however, is defining the approximate posterior, as in Equation 2.15.

$$p(u|y) \sim q(u) = \int p(u|f_m)p(f|f_m)\phi(f_m)df\,df_m = \int p(u|f_m)\phi(f_m)df_m = \int q(u, f_m)df_m \tag{2.15}$$

In this form $\phi(f_m)$ is chosen to be a free variational distribution. The approximate posterior distribution $q(u)$ is then specified by mean and covariance as shown in Equation 2.16 and Equation 2.17. Here mean and variance depend also on mean vector μ and variance matrix A.

$$m_y^q(x) = K_{xm}K_{mm}^{-1}\mu_A \tag{2.16}$$

$$k_y^q(x, x') = k(x, x') - K_{xm}K_{mm}^{-1}K_{mx'} + K_{xm}K_{mm}^{-1}AK_{mm}^{-1}K_{mx'} \tag{2.17}$$

Then the variational distribution ϕ needs to be found by determining its mean and covariance as well as inducing points X_m. Titsias proposes to solve that issue by defining the approximate distribution $q(f)$ and the exact posterior $p(f|y)$ and then to minimize the distance between the two[71]. This is equivalent to minimizing the distance between augmented posteriors: $p(f, f_m|y)$ and $q(f, f_m) = p(f|f_m)\phi(f_m)$ (which follows from the integral in Equation 2.15). This operation is equivalent to minimizing the KL divergence. Alternatively, it can be expressed as maximization of the variational lower bound of the true log marginal likelihood:

$$F_V(X_m, \phi) = \int p(f|f_m)\phi(f_m) log \frac{p(y|f)p(f_m)}{\phi(f_m)} df\,df_m \tag{2.18}$$

The optimal variational distribution ϕ^* can be found analytically by differentiating the log marginal likelihood expression, which yields:

$$\phi^*(f_m) = \mathcal{N}(f_m|\mu_A, A) \tag{2.19}$$

With the mean given by:

$$\mu = \sigma^{-2}K_{mm}(K_{mm} + \sigma^{-2}K_{mn}K_{nm})^{-1}K_{mn}y \tag{2.20}$$

And covariance given by:

$$A = K_{mm}(K_{mm} + \sigma^{-1}K_{mn}K_{nm})^{-1}K_{mm} \tag{2.21}$$

The found parameters - mean μ_A and the covariance matrix A can be then substituted back to Equations 2.16 and 2.17, to yield the predictive distribution.

Furthermore, with the optimal variational distribution, the log marginal likelihood has the form shown in Equation 2.22.

$$F = \frac{N}{2} log(2\pi) + \frac{1}{2} log|K_{fu}K_{uu}^{-1}K_{fu}^T + \sigma_n^2 I| + \frac{1}{2} y^T (K_{fu}K_{uu}^{-1}K_{fu}^T + \sigma_n^2 I)^{-1} y - \\ - \frac{1}{2\sigma_n^2} tr(K_{ff} - K_{fu}K_{uu}^{-1}K_{fu}^T) \tag{2.22}$$

Comparing with the LML for FITC (Equation 2.14) the only difference is the covariance matrix and the additional trace term which is the regularization term [71]. It can be interpreted as the total variance of the conditional prior $p(f|f_m)$, and maximizing the likelihood it is desired to minimize this term. Similarly as in other methods, the most straightforward way to find the optimal inducing points X_m is to use gradient descent methods (differentiating Equation 2.22 in respect to X_m). Alternatively, Titsias proposes selecting the inducing points from the training set [71]. This, however, is a combinatorial problem with prohibitive complexity making such an approach very uncommon.

In general, comparing VFE with FITC, the most radical difference is the more rigorous formulation of VFE. Posing the problem as minimizing the bound is more valid from the mathematical point of view (there is a true bound), which guarantees that VFE will not over-fit, which is the problem with FITC. Another consequence is that increasing number of inducing points improves the approximation [4]. In case the inducing points cover exactly the training points, VFE will reach the global minimum. In FITC, however, this is not the case, which results in over-fitting and underestimating the noise variance [4]. On the other hand, the VFE is more prone to being stuck in local minima, and thus it is more difficult to optimize than FITC [4]. More insight into the comparison between VFE and FITC can be found in Bauer et al [4]. The most intensive step causing this problem is finding the inverse of the covariance matrix $(K(x,x) + \sigma_n^2 I)^{-1}$ [75]. This essential step is necessary for determining the predictive mean and variance as shown in Equation 2.7, but also for finding the optimal hyper-parameters as in Equation 2.3 (in this case there is also an additional overhead of $O(N^2)$ per parameter for finding the gradient [49]). The operation is often additionally complicated by ill-conditioning of the matrix (which means it is difficult to invert because some eigenvalues are very small). The conditioning can be improved adding the jitter term $\sigma_n^2 I$, which bounds the minimum eigenvalue [4]. Furthermore, finding the inverse often is done with the Cholesky decomposition algorithm which provides better numerical stability over other methods [75].

2.3. Quantum computing: seeking quantum advantage

Quantum computing provides a fundamentally different way of performing calculations when compared to “classical” computing. In order to explore this new computing paradigm and develop Quantum Gaussian Processes, some fundamental concepts are introduced in subsection 2.3.1. Then, subsection 2.3.2 discusses general capabilities and potential application areas, while subsection 2.3.3 presents some of the most relevant quantum algorithms.

2.3.1. Fundamental concepts of quantum computing

Quantum computing is an alternative paradigm of computing, that relies on quantum mechanics. The computation is realized on quantum systems, which enables to utilize quantum phenomena to

perform computations. Quantum computations cannot be efficiently simulated on classical computers, which gives rise to the notion of "quantum advantage".

Superposition and entanglement

In particular, quantum computing takes advantage from two essential concepts of quantum mechanics: superposition and entanglement. Superposition enables the quantum state to be a combination of several different states. For instance, a qubit can be in a superposition of states 0 and 1. Although the superposition state is very specifically defined (for instance as a vector), upon measurement the system still will behave randomly and will collapse to a single state with a certain probability. For example: measuring the qubit in a uniform superposition will yield either 0 or 1 with probability 50 %. The system in superposition behaves as if it was in several states simultaneously, which is a key feature for quantum computing, as it enables quantum parallelism.

The second concept is entanglement. When the elements of a quantum system are entangled, the state of each of the elements is correlated with others [51]. In such case, the state of a system can be expressed only as a whole, and it is impossible to isolate the state of a single element. As a consequence, the measurement of a single element of such a system immediately gives information about the other elements. For example, one can prepare a system of two entangled particles, such that one is spin up, and the other is spin down. Measurement of one of the particles immediately gives information about the state of the other. Most surprisingly, the relation holds even when the particles are separated by a distance, which appears as if the particles were communicating faster than light, which was called "Spooky action at a distance" [5, 28, 51].

Quantum computing paradigms

There are two quantum computing paradigms: adiabatic quantum computing, and universal gate model. The adiabatic model relies on the concept of Hamiltonian evolution [2]. This model is particularly suitable for optimization, in which the problem is stated as an initial Hamiltonian (with easy to prepare ground state) and then the system undergoes quantum annealing process, during which the Hamiltonian is evolved, to reach its final form which gives the answer to the problem. The Canadian company D-Wave [18] currently holds the most advanced systems for quantum annealing.

The second quantum computing paradigm – the universal gate quantum computing has many more applications. In this model, the computation is performed by applying gates which are realized by unitary operators. For this research, the gate model was chosen as a more suitable due to its generality. Therefore, the remaining part of this work concerns only the universal gate model.

Qubits and Gates

The most basic unit of information in quantum computing is the qubit. It is analogous to the classical bit, however, due to its quantum-mechanical nature, it can possess quantum features. Therefore, a qubit can be in superposition, which is a state which is a combination of 0 and 1. Furthermore, systems of qubits can be entangled.

Formally, the state of a qubit is expressed as a stochastic vector, where the basis of the vector consists of its measurable states (0, and 1), while the coefficients correspond to probabilities of collapsing into the respective measurable states.

In general, quantum computation can be easily expressed in terms of linear algebra. The manipulation of qubits is performed by applying gates which are unitary operators. Those gates are simply expressed as unitary matrices, and application of a gate is equivalent to multiplying the qubit state vector by the matrix. More details on notation, basic operations etc. can be found in Appendix A.

2.3.2. Capabilities and potential application areas

The intrinsic features of quantum computing, such as superposition, entanglement, and computation in high-dimensional Hilbert spaces, result in unprecedented speed-ups when solving certain problems. Consequently, quantum algorithms can also tackle problems of sizes beyond what is tractable with classical algorithms, reaching so-called quantum advantage. This section gives a brief overview of those problems and the most attractive domains of quantum computing applications.

Cryptography and communication

One of the first practical examples of quantum computing was found by Peter Shor in 1994, when he invented an algorithm for factoring prime numbers [68]. With classical methods, the task becomes infeasible to solve already for numbers in order of 100s of digits. In fact, this intractability gave foundations of most of the current cryptographic systems [51]. With Shor's quantum algorithm, however, the task can be solved exponentially faster, which endangers those systems.

While quantum computing makes the encryption systems obsolete, it also proposes alternative ways of communication and key distribution [51]. The concept relies on exchange of information via quantum states and utilizing their intrinsic features such as superposition and entanglement. Such a communication channel enables for detection of eavesdropping, as any attempt would require measurement of the quantum system which causes its collapse.

Scientific computation

One of the major advantages of quantum computing is that it can easily tackle operations in high-dimensional Hilbert spaces[51, 57]. As a result, it is able to solve problems of sizes beyond the capabilities of classical computers. Consequently, quantum computing is expected to find its major applications in high-throughput scientific computing. The fact that formulation of quantum mechanics relies on linear algebra, makes quantum computing particularly suitable and intuitive to apply basic linear algebra subroutines. Tasks such as solving systems of linear equations are ubiquitous in any discipline of science and engineering. One particular field where solving linear systems is a performance bottleneck is machine learning. Consequently, quantum computing is expected to have an impact on that field.

Quantum chemistry

As quantum computers rely on quantum mechanics, they are perfectly suited for simulating quantum systems and have considerable potential for application in quantum chemistry[51, 53, 57]. The main problem of quantum simulations is the number of variables, which grows exponentially with the size of the system, making even relatively simple systems unfeasible to simulate. Quantum computing, however, is well suited for performing computation in high dimensional Hilbert spaces (which also grow exponentially with the system size).

The point has been proven by researchers from IBM, who showed that quantum systems can be already simulated on currently available quantum computers (NISQ - Noisy Intermediate Scale Quantum devices)[36, 57]. In their work, the researchers were able to address molecular problems of non-trivial molecules (beyond I period - up to BeH_2), in particular, to determine the ground state of Hamiltonian. The problem was solved using Variational Quantum Eigensolver implemented on a 6-qubit quantum chip.

2.3.3. Quantum algorithms for engineering applications

This section gives an overview of a selection of the most relevant quantum algorithms for scientific and engineering applications.

QPE: finding eigenvalues

Quantum Phase estimation is an algorithm for approximating eigenvalues [51]. Given a unitary operator U with eigenvectors $|u\rangle$ and eigenvalues $e^{2\pi i\phi}$ the QPE allows for estimating the phase ϕ thus allowing for finding the eigenvalues.

The structure of the algorithm is shown in the schematic circuit in Figure 2.9. The algorithm takes two registers: the eigenvalue register of size t and the input vector register (where the input vector of size n is resolved into a state of $k = log_2 n$ qubits). The algorithm essentially consists of three steps: preparation of the eigenvalue register in superposition, applying controlled unitaries, and inverse Fourier transform.

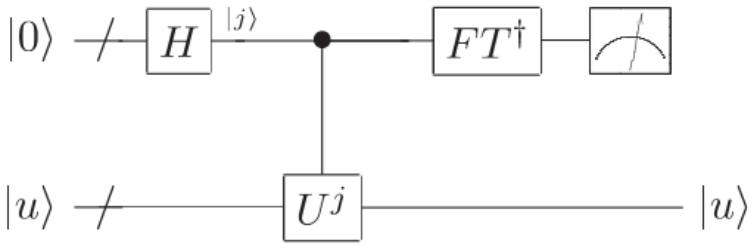


Figure 2.9: Schematic of the complete QPE [51].

For the sake of clarity, the first two steps are shown in more detail in Figure 2.10. The eigenvalue register is put in superposition by applying Hadamard gates. Then a series of controlled U operators is applied, with subsequent operators raised to successive powers of 2. The controlled operations entangle the eigenvalue register, putting the qubits into the state shown on the right side in Figure 2.10.

The state of the eigenvalue register can be written in the tensor product form as shown in Equation 2.23:

$$|\psi\rangle = \frac{1}{2^{t/2}}\left(|0\rangle + e^{2\pi i 0.\phi_t}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.\phi_t\phi_{t-1}}|1\rangle\right)...\left(|0\rangle + e^{2\pi i 0.\phi_t\phi_{t-1}...\phi_0}|1\rangle\right) \tag{2.23}$$

where the notation is clarified herein. The phase ϕ is in a range from 0 to 1, therefore it can be approximated by a t long bit string as: $\phi = 0.\phi_1\phi_2...\phi_t$ (which is the binary representation of fractions, with successive bits $\phi_1, \phi_2...$ corresponding to the successive powers of $\frac{1}{2}$ i.e. $\frac{1}{2}, \frac{1}{4}...\left(\frac{1}{2}\right)^t$). In case the phase cannot be resolved exactly on a register of a given size, the QPE finds only an approximation of the eigenvalues, with the accuracy dependent on the resolution of the eigenvalue register i.e. size t of the register.

Finally, in the third step the Inverse Quantum Fourier Transform is applied, which results in:

$$\frac{1}{2^{t/2}}\sum_{j=0}^{2^t-1} e^{2\pi i\phi j}|j\rangle|u\rangle \rightarrow |\phi\rangle|u\rangle \tag{2.24}$$

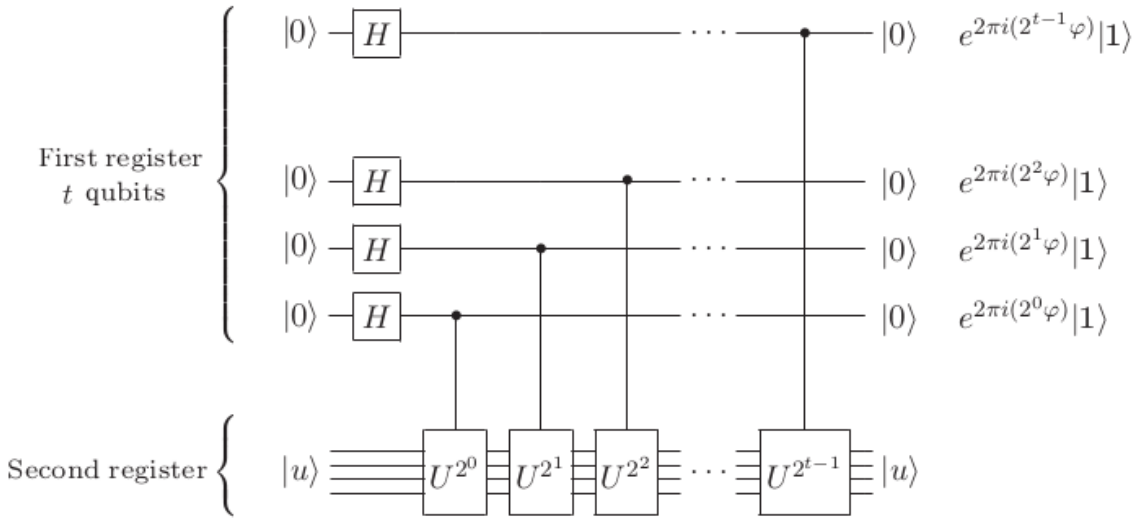


Figure 2.10: Schematic of the first step of QPE [51].

After the QPE is completed, the eigenregister contains ϕ, which approximates the eigenvalue corresponding to the eigenvector $|u\rangle$. In case the input vector is not a single eigenvector, but rather an arbitrary vector that can be resolved into eigenbasis of U, the eigenregister will contain a superposition of eigenvalues corresponding to the eigenvectors supporting the input vector.

HHL: solving systems of linear equations

In 2008 Harrow, Hassidim and Lloyd proposed a quantum algorithm for solving linear equations, named after their names as HHL algorithm [29]. Given linear system: $Ax = b$, to find the solution $x = A^{-1}b$ the algorithm requires only $O(log(n))$ operations, which is exponentially faster than the best classical algorithm with $O(nlog(n))$ [12].

The algorithm essentially consists of three steps: QPE, controlled rotation of ancilla qubit [2] and inverse QPE, as shown on the schematic circuit in Figure 2.11. The circuit consists of three registers: one ancila qubit, eigenvalue register and the register for input vector b. The size of the eigenvalue register depends on the resolution required by QPE, as discussed in subsection 2.3.3. The algorithm assumes that the register containing $|b\rangle$ is prepared beforehand, such that the vector b of size n is already mapped into quantum state of $log_2 n$ qubits. In reality such operation requires significant computational effort, compromising the overall speed-up of the algorithm. The problem could be solved with qRAM [23], however the concept is not developed enough yet. Furthermore, in order to map the vector into quantum state it needs to be normalized. The remaining registers are initialized in state $|0\rangle$. The state of the system at this point can be expressed as in Equation 2.25.

$$|\psi\rangle = |0\rangle_{anc}|0\rangle_{eig}|b\rangle_b \tag{2.25}$$

Once the registers are ready, the QPE is performed according to the procedure discussed in subsection 2.3.3. The input matrix A is assumed Hermitian. If this is not the case (which is very likely in real-life problems), the matrix can be extended to $H = \begin{bmatrix} 0 & A \\ A^\dagger & 0 \end{bmatrix}$.

[2] Ancilla qubit - an auxiliary qubit introduced in order to facilitate the computation, called an *"ancilla"* from the latin for *"servant"*.

With the QPE step, the system is basically diagonalized, with eigenvalues encoded on the eigenvalue register. The state of the system is:

$$|\psi\rangle = \sum_j |0\rangle_{anc} \beta_j |\lambda_j\rangle_{eig} |u_j\rangle_b \tag{2.26}$$

Where, $|u_j\rangle$ are the eigenvectors of the matrix A, and vector $|b\rangle$ becomes decomposed into eigenbasis as: $|b\rangle = \sum_j \beta_j |u_j\rangle$.

In the second step, a rotation is performed on the ancilla qubit, controlled by the inverses of the eigenvalues. In particular, the ancilla qubit is rotated applying $R_y(\theta)$, where the rotation angle is $\theta = \frac{C}{\lambda}$. The normalization constant C is chosen in such way to bound the angle to 2π. The step puts the system into the following state:

$$|\psi\rangle = \sum_j \beta_j \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle_{anc} + \frac{C}{\lambda_j} |1\rangle_{anc} \right) |\lambda_j\rangle_{eig} |u_j\rangle_b \tag{2.27}$$

The final step is the inverse QPE, which is simply a QPE operation applied in reversed order. This step is applied to disentangle the system (uncompute the eigenvalues back to the initial state) and put the system into the final state:

$$|\psi\rangle = \sum \beta_j \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle_a + \frac{C}{\lambda_j} |1\rangle_a \right) |0\rangle_{eig} |u_j\rangle \tag{2.28}$$

In this final state, the ancilla qubit is in a superposition of $|0\rangle_a$ and $|1\rangle_a$. Consequently, upon measurement it will collapse to state $|0\rangle_a$ with the probability of $\left(1 - \frac{C^2}{\lambda_j^2}\right)$ or $|1\rangle_a$ with probability of $\left(\frac{C^2}{\lambda_j^2}\right)$.

Conditioning the solution on the ancilla in state $|1\rangle$ (that is taking only the measurements in which the ancilla qubit is measured to be $|1\rangle_a$), ensures that the system is in a state shown in Equation 2.29.

$$|\psi\rangle = \sum_j \beta_j \frac{C}{\lambda_j} |1\rangle_{anc} |0\rangle_{eig} |u_j\rangle \tag{2.29}$$

This state is proportional (by the normalization constant) to the solution of the linear system:

$$A^{-1} b = x \propto \sum_j \frac{\beta_j}{\lambda_j} |u_j\rangle = |x\rangle \tag{2.30}$$

The solution $|x\rangle$ is in a quantum state, which means that it is encoded as the vector of probability amplitudes. Accessing all the amplitudes requires performing an exponential number of measurements, which compromises the speedup of the complete algorithm. The problem of accessing the solution vector is the second major caveat of the HHL algorithm.

This does not render the whole algorithm useless. Harrow, Hassidim and Lloyd argue that there are classes of problems, where one is interested only in part of the solution rather than the complete vector. In those cases the problem can be reformulated as an expectation value of some measurement operator M, such that $\langle x| M |x\rangle$. This measure can be efficiently evaluated in quantum state and the answer can be retrieved without compromising the speed-up.

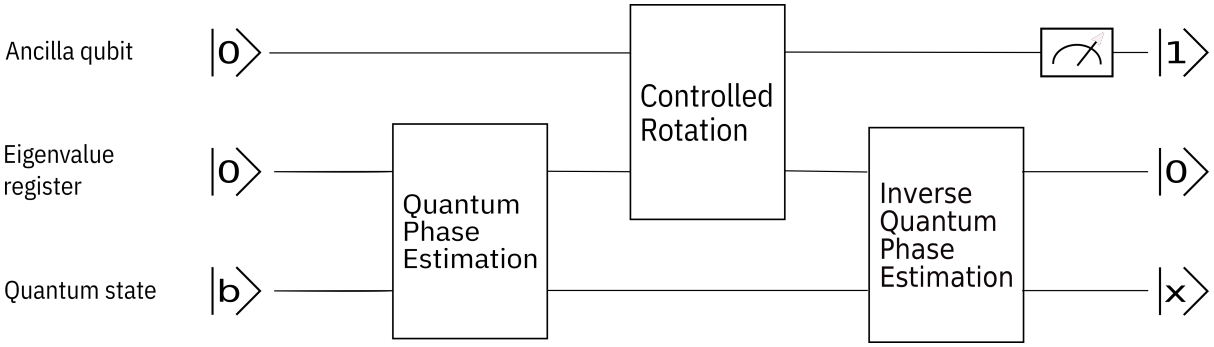


Figure 2.11: Schematic circuit for HHL algorithm.

VQE: quantum optimization

Variational Quantum Eigensolver (VQE) provides an alternative to Quantum Phase Estimation for finding eigenvalues of an operator. It is a hybrid method consisting of quantum and classical routines. VQE is based on the variational principle of quantum mechanics, which guarantees that the expectation of H in some state ψ cannot underestimate the ground energy level (the lowest eigenvalue) [26]:

$$E_g \leq \langle \psi | H | \psi \rangle \tag{2.31}$$

Essentially VQE consists of two parts: Quantum Expectation Estimation routine (QEE) and a classical optimizer. The QEE computes an expectation of Hamiltonian H given some parametrized ansatz input state $|\psi(\lambda)\rangle$, where λ is some variational parameter. Then the parameters λ are optimized with a classical algorithm in a way to minimize the estimate of the energy.

The problem Hamiltonian is first expanded as a sum of Pauli matrices [56]:

$$H = \sum_{i\alpha} h^i_\alpha \sigma^i_\alpha + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \sigma^i_\alpha \sigma^j_\beta + \ldots \tag{2.32}$$

Then using linearity of the operators, the expectation of the Hamiltonian can be computed as a sum of expectations of the component Pauli matrices multiplied with the real numbered factors h.

$$\langle H \rangle = \sum_{i\alpha} h^i_\alpha \langle \sigma^i_\alpha \rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle \sigma^i_\alpha \sigma^j_\beta \rangle + \ldots \tag{2.33}$$

Using this property, the energy can be estimated as shown in Equation 2.34

$$E(\lambda) = \langle \psi(\lambda) | H | \psi(\lambda) \rangle = \sum_{i\alpha} h^i_\alpha \langle \psi(\lambda) | \sigma^i_\alpha | \psi(\lambda) \rangle + \sum_{ij\alpha\beta} h^{ij}_{\alpha\beta} \langle \psi(\lambda) | \sigma^i_\alpha \sigma^j_\beta | \psi(\lambda) \rangle + \ldots \tag{2.34}$$

Estimating the expectations of Pauli operators on a quantum device can be done efficiently [56]. Furthermore, the quantum state can be stored with exponentially fewer resources than in classical paradigm, which prevents from the N-representability problem [56]. Similarly as QPE, VQE provides exponential speedup [56].

Unlike the QPE, where the whole state evolution is executed at once, VQE is executed in iterations with short intervals of evolution of the quantum states, interrupted with measurements of energy

and optimization of the parameters. As a consequence, the VQE requires orders of magnitude less coherence time than QPE, which makes the algorithm well suited to run on NISQ devices [56]. The drawback of the VQE, however, is the need for a high sampling rate, as after each evolution interval the expectation of an operator needs to be measured. Another caveat is a need for classical optimization.

Grover's algorithm: quantum search

In 1996 Lov Grover proposed a quantum algorithm for search of unstructured databases [27]. For a database with N elements, the algorithm finds the answer with only $O(\sqrt{N})$ steps, which compared to $O(N)$ of the classical approach gives a quadratic speedup. Most remarkably, the algorithm is formulated in such way that it takes a superposition of all the inputs, which allows evaluating all the inputs at the same time. Therefore the algorithm provides a very explicit example of how the principles of quantum mechanics are used to the advantage of quantum computing.

More specifically, the algorithm searches over inputs of an oracle function and finds the inputs that satisfy this function. For instance, the oracle can be some boolean function, and the algorithm finds all the inputs for which the function outputs 1. The structure of the algorithm is presented in Figure 2.12. In principle, the algorithm requires a single register for resolving the function input space. In practice, however often a second register is needed for the oracle workings, as shown in the figure.

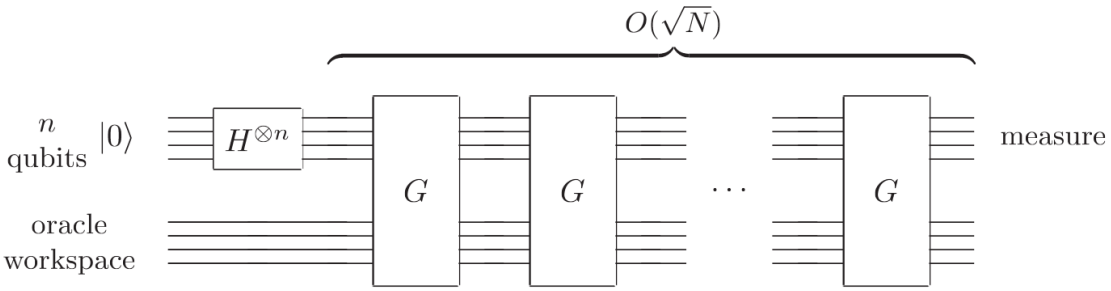


Figure 2.12: Schematic circuit of Grover's algorithm [51].

The algorithm starts with the preparation of superposition of the inputs, by applying the Hadamard operator. Next, Grover's operator (marked as G in the circuit of Figure 2.12) is applied and iterated $\sqrt{N}$ times. The inner circuit of this operator is shown in more detail in Figure 2.13.

First, the oracle function is applied. The role of this operator is to mark the answer by applying the oracle, and then to amplify its probability via amplitude amplification.

The oracle is a unitary operator and is implemented as a quantum circuit. The oracle marks the answer (the input that satisfies the function) by making its amplitude negative. Then an amplitude amplification is carried out, where the marked input is inverted over the average. In this way, the amplitude of the marked entry is stretched, while the amplitudes of other entries shrinks. This step is implemented with the three last blocks shown on the circuit in Figure 2.13, namely Hadamard operator, phase shift, and second Hadamard operator. To maximize the amplitude, the amplification needs to be iterated several times, which explains iteration of the Grover's operator. Furthermore, the procedure has very intuitive geometric interpretation, as a series of reflections and rotations, which is explained in more details in [51] and [32].

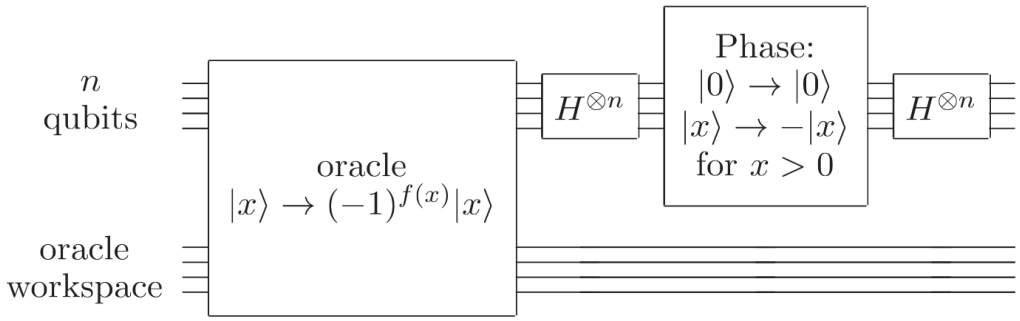


Figure 2.13: Schematic circuit of Grover's operator [51].

2.4. Quantum machine learning

Machine learning algorithms are already revealing limitations with intractably large datasets or complex optimization problems, thus slowly approaching the limits of the classical machines [16]. This barrier could be overcome by quantum computing, which is believed to be intrinsically well suited for machine learning applications. Those speculations are supported by certain fundamental similarities of both fields, such as operating in high-dimensional spaces, or strong reliance on linear algebra [12, 31, 66].

This potential of quantum algorithms for accelerating the linear algebra operations is currently the main driving force in the quantum machine learning research [12]. Already existing algorithms, including quantum Fourier transform [51], HHL for solving systems of linear equations [29] and QPE for finding eigenvalues [51] and eigenvectors [43], gave rise to a number of concepts, such as gradient descent and Newton's method [62], semidefinite programming [14] or topological analysis [44], to mention a few.

Aside from the aforementioned, the majority of concepts reported in the literature are defined on a very high level, and consequently, suffer from the negligence of detail and practical consideration, rendering them unfeasible. For this reason, they are omitted in this review, and the remaining part of this chapter focuses on only three of the most developed quantum machine learning algorithms, namely: qPCA in subsection 2.4.1, Support Vector Machines in subsection 2.4.2, and finally Quantum Gaussian Processes in subsection 2.4.3. The last algorithm is analyzed in more detail, as it provides the essential foundation for the proposed research project.

2.4.1. qPCA

Principal Component Analysis is a common routine in machine learning and data science. It relies on eigendecomposition of the covariance matrix to identify the most dominant trends in the data, which reveal as eigenvectors, with corresponding eigenvalues marking their significance (large eigenvalue means more dominant trend). This technique allows simplifying high-dimensional data by approximating it with its principal components, that is a subset of eigenvectors corresponding to the largest eigenvalues.

The algorithm relies on QPE for diagonalizing the system (thus finding the eigenvalues), and repeated sampling of data. The procedure is summarized by Biamonte et al.[12], and goes as follows: first, a random data vector v_j is chosen and mapped into a quantum state: $v_j \rightarrow |v_j\rangle$. The corresponding

density state is then: $\rho = 1/n \sum_j |v_j\rangle\langle v_j|$ which is equivalent to the covariance matrix. Then using the matrix exponentiation and QPE, one can find eigenvalues and decompose the vector into its principal components: $v \rightarrow \sum_k v_k |e_k\rangle |c_k\rangle$. The algorithm provides exponential speedup over the classical PCA.

The practical implementation of the aforementioned algorithm might not be as straightforward as it seems. The issue with the practical use of this algorithm might be accessing the eigenvectors, which are in the quantum state. More on qPCA can be found in reference[43], which explains the construction of the qPCA algorithm as a combination of QPE with quantum self-tomography.

2.4.2. Support Vector Machines

Support vector machine is one of the most popular methods for classification in machine learning [31]. The main limitation, however, is when the problem feature space is large, which increases the costs of evaluating kernel functions [31]. Therefore there is a lot of interest in enhancing SVMs with quantum algorithms.

Anguita et al. [3] proposed to formulate SVM as an optimization problem, and use Grover algorithm for finding the optimal values. In particular, researchers refer to Durr and Hoyer's [19] concept of minimization algorithm based on Grover search. This concept, however, is developed on a very high level, and does not delve into neither details of possible implementation nor even feasibility of the proposed concept (in particular, the way in which the elements are supposed to be marked is neglected, despite being an essential part of the concept).

Rebentrost et al. [61] proposed to reformulate SVM as a least squares problem, and then solve it with the HHL algorithm. The algorithm is additionally supported by Principal Component decomposition. The step is implicitly implemented via QPE subroutine of HHL which estimates the eigenvalues and eigenvectors of the matrix. Setting parameters of QPE enables the cut-off of eigenvalues below a certain threshold, thus taking only the most significant eigenvalues, which are equivalent to the principal components [29, 61]. The proposed algorithm achieves the overall exponential speedup by accelerating the solution of the linear equations (via HHL) as well as evaluating the dot products in the kernel matrix, which in the quantum feature space can be exponentially faster.

The algorithm assumes that the data is already provided in a quantum state, which could be achieved with qRAM [29, 61]. In case the classical data needs to be loaded, the speedup is compromised, which is also one of the key problems in the HHL algorithm [29].

Additionally, Rebentrost et al.[61] mention a trick for implementing non-linear kernels. The suggested method basically relies on mapping a vector into "$d-times$ tensor product", and manipulate those vectors in the higher dimensional spaces, which can be done efficiently on a quantum computer.

One of the most recent and spectacular examples of quantum enhancement of SVMs was accomplished by researchers from IBM in 2019 [31], who proposed two algorithms for quantum acceleration of SVM, namely variational quantum classifier and quantum kernel estimator. The first concept relies on variational circuits employed to classify the data. The approach is inspired by Mitarai's research [48]. This method is similar to the conventional SVM in the sense that the data is mapped non-linearly to construct a linear decision function in the feature space. In this case, however, the data is mapped in quantum state. The second algorithm is the quantum kernel estimator, which is simply used to optimize the classical SVM.

The most significant feature of both algorithms is that they can take data in classical form, and still provide speed-up, while the concept proposed by Rebentrost [61] assumes data input via QRAM,

which is a significant caveat. Furthermore, both algorithms developed by the IBM researchers aim for application on noisy intermediate-scale quantum computers, which certainly is an advantage over the concepts employing HHL and QPE which require perfect noiseless devices. This approach allowed to apply those algorithms on a real 5-qubit quantum computer.

2.4.3. Quantum Gaussian Processes

The idea of using quantum computing for speeding up the linear algebra tasks was also applied to Gaussian Processes. This resulted in two concepts proposed by Zhao et al., namely for speeding up the inference [79] and for accelerating the training phase [80]. Both concepts are essential for the proposed research, as they provide the foundation for practical implementation. Therefore both concepts are discussed in detail in the following sections.

Speeding up inference

In 2015 Zhao et al. [79] proposed a quantum algorithm for Gaussian Processes that provides exponential speedup. The general idea is to solve the system of equations in form of $A^{-1}v$ (computationally most intensive step in GP) using HHL algorithm (Equation 2.3.3) and then calculate the dot product with some second input vector u, yielding the answer in form of $uA^{-1}v$. The algorithm can be then directly used to predict the mean and the variance of the Gaussian Processes (given by Equation 2.35) by substituting proper vectors and matrices i.e. for finding the mean, vector y is substituted for vector v, $(K+\sigma_n^2 I)$ becomes the input matrix A and k_*^T is substituted as the second input vector u. Variance is obtained in an analogous way.

$$\begin{aligned} \mu &= k_*^T (K+\sigma_n^2 I)^{-1} y \\ \Sigma_* &= K(x_*, x_*) - k_*^T (K+\sigma_n^2 I)^{-1} k_* \end{aligned} \tag{2.35}$$

The whole algorithm consists of three major steps: input preparation, controlled HHL, and measurement of the observable, as shown on the schematic circuit in Figure 2.14. The steps are described in detail in the following subsections.

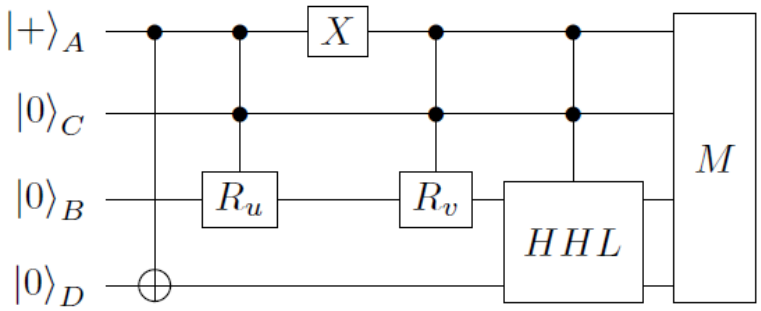


Figure 2.14: Schematic circuit of the algorithm proposed by Zhao et al. [79].

Step I: Input preparation The first step is the state preparation, where the input vectors are mapped into a quantum state. The proposed approach for preparation of a single vector requires 2 registers: register B for storing the vector and ancilla qubit C. For n dimensional vector v, register B needs

to be of size $N = log_2(n)$. The state preparation is performed by a series of rotations on the ancilla qubit, controlled by the i^{th} entries of the B register, such that the system ends up in a state shown in Equation 2.36:

$$|\tilde{v}\rangle = \frac{1}{\sqrt{s_v}} \sum_i^n |i\rangle_B \left(\sqrt{1 - c^2 v_i^2} |1\rangle_C + c v_i |0\rangle_C \right) \tag{2.36}$$

The key for the whole algorithm, however, is preparation of the two input vectors in superposition. This is achieved by entangling the state preparation routine with an additional ancila qubit A, which results in the state shown in Equation 2.37:

$$\begin{aligned} |\tilde{\phi}_{u,v}\rangle = &\frac{1}{\sqrt{2s_u}} \sum_i^n |1\rangle_A |i\rangle_B \left(\sqrt{1 - c^2 u_i^2} |1\rangle_C + c u_i |0\rangle_C \right) + \\ &+ \frac{1}{\sqrt{2s_v}} \sum_i^n |1\rangle_A |i\rangle_B \left(\sqrt{1 - c^2 v_i^2} |1\rangle_C + c v_i |0\rangle_C \right) \end{aligned} \tag{2.37}$$

The overall input preparation procedure can be explained best by following the steps shown on the schematic circuit shown in Figure 2.14.

First, the ancilla qubit A is put in superposition, or equivalently, initialized in state $|+\rangle$. Other registers are initialized in state $|0\rangle$. The A is entangled with HHL ancilla D (the reason will be explained later).

Then the state preparation routine is applied conditioned on qubit A being in state $|0\rangle$, to prepare vector u. In practice, the preparation would be implemented as a series of double controlled rotations, which on the schematic circuit in Figure 2.14 was collectively designated as R_u. In a similar way, preparation of v is conditioned on A being in state $|1\rangle$ (which is achieved by flipping the A qubit before the control with X gate, and flipping back after).

After the state preparation, the state of the complete system is given with Equation 2.38.

$$\begin{aligned} |\psi_1\rangle = &\frac{1}{\sqrt{2s_u}} \sum_i^n |0\rangle_D |0\rangle_A |i\rangle_B \left(\sqrt{1 - c^2 u_i^2} |0\rangle_C + c u_i |1\rangle_C \right) + \\ &+ \frac{1}{\sqrt{2s_v}} \sum_i^n |1\rangle_A |i\rangle_B \left(\sqrt{1 - c^2 v_i^2} |0\rangle_C + c v_i |1\rangle_C \right) \end{aligned} \tag{2.38}$$

Step II: Conditioned HHL In the second step the prepared input is fed into the HHL algorithm. Register B carries the input vector ($|b\rangle$ in HHL notation), while register D is used as the HHL ancilla. The whole HHL algorithm is conditioned on A and C registers. In practice, it could be achieved by converting all the gates in HHL circuit to their controlled versions, where the ancillae A and C are added as controls. Furthermore HHL also requires a register for storing the eigenvalues in QPE, which was not shown on the schematic in Figure 2.14. After HHL is executed the state of the system is given by Equation 2.39:

$$
\begin{aligned}
|\psi\rangle = &\frac{1}{2s_u}|0\rangle_A \sum |i\rangle_B \left(\sqrt{1-c_u^2 u_i^2}\,|0\rangle_C + c_u u_i\,|1\rangle_C\right)|1\rangle_D \\
&+ \frac{1}{2s_v}|1\rangle_A \sum c_v \beta_i \left|\mu_i\right\rangle_B |1\rangle_C \left(\sqrt{1-\frac{c^2}{\lambda_i^2}}\,|0\rangle_D + \frac{c}{\lambda_i}|1\rangle_D\right) + \\
&+ \frac{1}{2s_v}|1\rangle_A \sum \sqrt{1-c_v^2 v_i^2}\,|i\rangle_B |0\rangle_C |0\rangle_D
\end{aligned}
\tag{2.39}
$$

Step III: Measuring the Observable Finally, a measurement operator $M = X_A I_B |1\rangle\langle 1|_C |1\rangle\langle 1|_D$ is applied, as designated by the block M on the schematic circuit in Figure 2.14. The expectation value of the operator is proportional to the desired dot product form, as shown in Equation 2.40:

$$
\langle M \rangle = \frac{c_u c_v c}{\sqrt{s_u s_v}} u^T A^{-1} v \tag{2.40}
$$

Constant c is the HHL constant that bounds the angle of rotation, while $\frac{c_u}{\sqrt{s_u}}$ and $\frac{c_v}{\sqrt{s_v}}$ are the values used to normalize the input vectors u and v. In this way, taking $u = k_*$, $A = K - \sigma_n^2 I$ and $v = y$ the predicted mean can be calculated, while if $v = k_*$ is taken instead, the predictive variance can be retrieved, as shown in Equation 2.41:

$$
\mu_* = \frac{\sqrt{s_u s_v}}{c c_{k_*} c_y}\langle M \rangle \qquad\qquad \Sigma_* = K_{**} - \frac{s_{k_*}}{c c_{k_*}^2}\langle M \rangle \tag{2.41}
$$

In principle measurement of the operator yields a Bernoulli random variable (either 0 or 1) with a certain probability. Therefore with a sufficient number of samples the expectation of the observable M can be retrieved. Most importantly, with such an operator the number of measurement samples is independent from the size of the solved system. This is not the case when the complete solution needs to be retrieved from HHL, as in such situation the number of measurements grows exponentially with the size of the system, killing the speedup. Therefore, the operator ensures that the exponential speed-up can be achieved in practice.

Remark 1 *The performance of the algorithm is mostly driven by the performance of HHL, where one of the most significant factors is conditioning of the input matrix, in this case* $(K+\sigma^2 I)$*. While sparsity can be improved by using one of the approximation methods (sparse GP), condition number can be altered by increasing the noise variance [79], which increases the lower bound on the minimum eigenvalue.*

Remark 2 *In Zhao's concept the HHL should be conditioned on both registers A and C. In practice, this can be achieved by adding two controls to each gate of the HHL circuit. Since in practical implementations usually only basic gates can be applied, the multi-controlled gates need to be expanded as a series of the basic gates. This consequently makes the circuit few orders of magnitude longer and results in an additional overhead to the computational cost.*

Speeding up the training

In a follow-up research Zhao et al. presented a concept of using quantum computing for speeding up the training [80], in particular for evaluation of the Log marginal likelihood. The research is motivated by the fact that the training phase is the origin of a considerable overhead, and might hinder the application of Gaussian Processes even in case QGP is employed.

The log marginal likelihood (LML) that is to be calculated can be expressed as follows:

$$LML = -\frac{1}{2}\log\left(det[K+\sigma_n^2 I]\right) - \frac{1}{2}y^T(K+\sigma_n^2 I)^{-1}y - \frac{n}{2}log\,(2\pi) \tag{2.42}$$

The origin of the high computational cost are the first and the second term of LML, which require evaluation of determinant, and solving a linear system, respectively. Therefore, Zhao proposes to evaluate those terms using quantum algorithms.

For evaluation of the first term, Zhao proposes to use the identity shown in Equation 2.43, where the eigenvalues can be found efficiently using QPE.

$$\langle log\lambda_i \rangle = \frac{1}{n}\sum_{i=1}^{n}\lambda_i = \frac{1}{n}log[det[A]] \tag{2.43}$$

The second term can be evaluated using the same algorithm as used for inference as discussed in subsection 2.4.3, i.e. the modified HHL algorithm with measurement operator and state preparation routine. The proposed algorithm scales logarithmically with the size of the systems, resulting in exponential speed-up against the classical methods which are $O(n^3)$.

3

Implementation of Quantum Gaussian Processes

The main objective of this research project is to explore the potential application of quantum computing to enhance the computational framework for data-driven design of metamaterials. The general concept is to replace the computationally expensive machine learning part of the framework with an exponentially faster quantum algorithm for Gaussian Processes. To achieve this objective, a proof-of-concept demonstrator is built, consisting of two major elements: 1) implementation of the quantum algorithm for Gaussian Processes, and 2) demonstrating its application in a sample metamaterial design problem.

This chapter provides the details of the first of the two steps, namely the implementation. Analysis of the algorithm performance in practical setting is the crucial step in verifying the proposed concept, specifically, whether applying the QGP algorithm in the computational design framework yields any advantage, and if so, for what conditions.

For this purpose, the algorithm was implemented as a Python code within Qiskit (a quantum computing framework), as discussed in Section 3.1. Next, Section 3.2 provides with an analysis of the approximation schemes employed in the algorithm, which have crucial influence on the QGP performance. The effect of the different approximations are studied also experimentally, by numerical testing of the implemented algorithm, presented in Section 3.3. The tests give invaluable insight into two crucial performance metrics of the algorithm: accuracy and computational cost. The accuracy is quantified in terms of error, which was measured for different parameter settings and presented in the form of error contour plots. The computational cost is measured in terms of circuit depth, which together with complexity analysis was used to build an empirical cost model of the algorithm, presented in Section 3.4. Finally, Section 3.5 explores sparsity inducing mechanism in QGP and compares its performance with the classical Nystrom method by means of analytically derived error bounds and numerical tests.

3.1. Implementation details

This section provides details of the implementation of the algorithm. First, a choice of platform for the implementation is discussed in subsection 3.1.1. This step is important as it constrains the implementation, leading to some modifications as discussed in detail in subsection 3.1.2.

3.1.1. Quantum computing framework

Implementing quantum software requires a specially designated environment, providing means to implement the algorithm as a code, compiling quantum circuits, as well as execution either on simulators or on physical quantum device. There are already many quantum computing software platforms providing those utilities at a different level. An overview of those can be found in [21].

In this research, IBM's Qiskit was selected as the framework for implementing QGP algorithm. Qiskit is full-stack quantum computing software framework, integrating the following elements[1]:

- **Software development kit** – Qiskit provides a framework for developing quantum algorithms in a high-level programming language (Python). It includes a number of Python libraries focusing on different aspects of quantum computing, such as: development of algorithms, simulation, and error mitigation.
- **Compilers** – Qiskit includes a compiler to translate quantum algorithm written in high-level code (Python) into quantum circuit expressed in terms of basic machine instructions (quantum assembly language - QASM). Qiskit additionally supports the compilation with transpilers, which allow for optimizing circuits for different objectives, such as: minimizing resources (circuit width and depth), minimizing errors, or mapping the circuit onto real devices by including connectivity of the chip architecture.
- **Backends** – Qiskit integrates different backends for executing quantum algorithms. It provides a number of simulators that allow for simulating algorithms locally on classical computers. It also provides interface to IBM's quantum devices available via IBM's cloud service [64].

Qiskit is currently the largest open-source quantum computing framework, which translates to broader range of functionalities, as well as better community support compared with other platforms. The choice was additionally motivated by the availability of the algorithms that were already implemented in Qiskit libraries, and particularly the HHL algorithm, which is the core of the QGP concept.

3.1.2. Detailed description of the algorithm

The implementation relies on Zhao's concept, discussed in subsection 2.4.3. The implemented algorithm consists of three major steps, as shown on the top-level circuit in Figure 3.1.

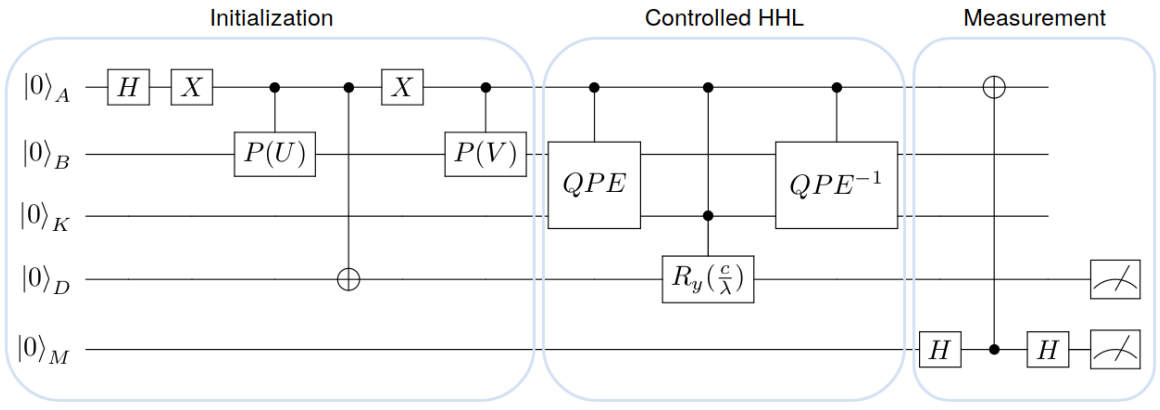


Figure 3.1: Schematic of the circuit for implementation of the HHL-GP algorithm, based on Zhao's [79] concept. *A* is the ancilla register, *B* is the input state register, *K* is the register storing the eigenvalues, *D* is the HHL ancilla, and *M* is the register for measuring the observable.

I. Initialization

The first step of QGP is the initialization, which prepares the two input vectors in superposition. A crucial part of the initialization is the vector preparation, which maps the input vectors into the quantum state. In Qiskit implementation of QGP the state preparation is carried out with an already existing Qiskit initialization procedure, which is fundamentally different than the original Zhao's concept which relies on repeated rotations assisted by an ancilla qubit. In Qiskit, on the other hand, the initialization is based on an iterative method proposed by Shende et al. [67]. In this alternative approach, an algorithm takes the input vector and searches for a combination of gates that bring this state-vector back to state $|0\rangle$, by unentangling successive qubits one-by-one. The found combination of quantum gates can be then applied in reversed order to bring the register from state $|0\rangle$ to the state given by the input vector, thus effectively initializing the desired state vector. More details on the method can be found in the Qiskit documentation [1] and paper by Shende et al. [67].

Most importantly, using the Qiskit method allowed to eliminate one of the ancilla qubits (register C on the schematic in Figure 2.14), which has tremendous impact on the overall performance of the whole algorithm. In the original Zhao's concept, this eliminated ancilla played a role of a second control qubit for HHL, which means that every basic operation (quantum gate) of HHL was controlled by two qubits. In practice, controlled gates are relatively expensive as they can be implemented only as an expansion consisting of a series of basic gates (2-qubit CNOT and basic 1-qubit gates). In such case, limiting entanglement of HHL from 2 to 1 control ancillae, dramatically improves scaling of the algorithm in terms of basic gates and consequently reduces depth (the number of gates) of the complete QGP circuit by a few orders of magnitude, making the implementation significantly more efficient compared to the original concept.

Having the state preparation procedure implemented for a single vector, a complete QGP initialization circuit is assembled to initialize the two vectors in superposition, as shown in Figure 3.1. This is achieved by entangling the vector preparation circuits (marked in the circuit as $P(U)$ and $P(V)$) with the ancilla qubit A. In Qiskit implementation of the entanglement is realized by replacing all the gates with their controlled versions, with qubit A acting as the control bit. To enable superposition of the two vectors, the qubit A is initialized in uniform superposition by applying gate H, and then the entangled circuits $P(U)$ and $P(V)$, conditioned on A being in state $|0\rangle_A$ and $|1\rangle_A$ respectively, which is achieved by flipping the A qubit with X gates as shown in the figure. The qubit D is also conditioned on A in state $|0\rangle$, which is implemented by applying the CNOT gate.

The state of the system at the end of this procedure is given by Equation 3.1.

$$|\psi\rangle = \frac{1}{\sqrt{2}}|1\rangle_D|0\rangle_A|u\rangle_B + \frac{1}{\sqrt{2}}|0\rangle_D|1\rangle_A|v\rangle_B \tag{3.1}$$

II. Controlled HHL

In the second step of QGP, the conditioned HHL algorithm is applied. For the purpose of QGP implementation, the already existing HHL module from Qiskit Aqua library was used. The HHL circuit generated by Aqua was modified to enable conditioninng of the HHL on qubit A, which was done simply by converting all the gates from the HHL circuit to their controlled version, with qubit A acting as the control qubit. In this step the system evolves to state shown in Equation 3.2

$$|\psi\rangle = \frac{1}{\sqrt{2}}|1\rangle_D|0\rangle_A|u\rangle_B + \sum_i \frac{1}{\sqrt{2}}\left(\sqrt{1-\frac{c^2}{\lambda_i^2}}|0\rangle_D + \frac{c}{\lambda_i}|1\rangle_D\right)|1\rangle_A\beta_i|\mu_i\rangle_B \tag{3.2}$$

Here, c is the constant of rotation used in the HHL, while the expression $\beta_i|\mu_i\rangle_B$ is the vector v written in eigenbasis of the HHL input matrix A. Considering the state written in such form, it becomes apparent that conditioning the HHL on qubit A effectively results in applying the HHL only to the vector v.

III. Measurement

The last step is the measurement, which aims to determine the dot product $u \cdot (A^{-1}v)$. Originaly in Zhao's concept this step is performed by applying a measurement operator $M = X_A \bigotimes I_B |1\rangle\langle 1|_D$. In Qiskit the measurement can be done only along the standard basis, which means that the allowed measurement operators are in the form $M = |i\rangle\langle i|$. Therefore, the operator M can be effectively implemented by applying a circuit that brings the state from basis M to a standard basis in which the measurement can be done. In such way, the measurement along the standard basis allowed in Qiskit yields the results equivalent to applying M.

The circuit enabling the measurement was constructed as follows: first a measurement qubit M is introduced in superposition (applied H gate). The state of the system can be written as shown in Equation 3.3.

$$\begin{aligned}\psi &= \frac{1}{\sqrt{2}}(|0\rangle_M + |1\rangle_M)\frac{1}{\sqrt{2}}|1\rangle_D|0\rangle_A|u\rangle_B + \sum_i \frac{1}{\sqrt{2}}(|0\rangle_M + |1\rangle_M)\frac{1}{\sqrt{2}}\left(\sqrt{1-\frac{c^2}{\lambda_i^2}}|0\rangle_D + \frac{c}{\lambda_i}|1\rangle_D\right)|1\rangle_A\beta_i\left|\mu_i\right\rangle_B = \\ &= \frac{1}{2}|0\rangle_M|1\rangle_D|0\rangle_A|u\rangle_B + +\frac{1}{2}|1\rangle_M|1\rangle_D|0\rangle_A|u\rangle_B \\ &+ \sum_i \frac{1}{2}|0\rangle_M\left(\sqrt{1-\frac{c^2}{\lambda_i^2}}|0\rangle_D + \frac{c}{\lambda_i}|1\rangle_D\right)|1\rangle_A\beta_i|\mu_i\rangle_B + \sum_i \frac{1}{2}|1\rangle_M\left(\sqrt{1-\frac{c^2}{\lambda_i^2}}|0\rangle_D + \frac{c}{\lambda_i}|1\rangle_D\right)|1\rangle_A\beta_i|\mu_i\rangle_B\end{aligned} \tag{3.3}$$

Next, a CNOT gate is applied with qubit M used as control and qubit A as the target. Its effect can be seen by comparing expressions 3.3 and 3.4. Note that in all the components where $M = |1\rangle$ in Equation 3.3 the state of A is flipped (from 0 to 1 and from 1 to 0), which results in state shown in Equation 3.4:

$$\begin{aligned}\psi = &\frac{1}{2}|0\rangle_M|1\rangle_D|0\rangle_A|u\rangle_B + \\ &+\frac{1}{2}|1\rangle_M|1\rangle_D|1\rangle_A|u\rangle_B + \\ &+\frac{1}{2}|0\rangle_M\left(\sqrt{1-\frac{c^2}{\lambda^2}}|0\rangle_D + \frac{c}{\lambda}|1\rangle_D\right)|1\rangle_A\sum_i\frac{c}{\lambda_i}\beta_i|\mu_i\rangle_B + \\ &+\frac{1}{2}|1\rangle_M\left(\sqrt{1-\frac{c^2}{\lambda^2}}|0\rangle_D + \frac{c}{\lambda}|1\rangle_D\right)|0\rangle_A\sum_i\frac{c}{\lambda_i}\beta_i|\mu_i\rangle_B\end{aligned} \tag{3.4}$$

Finally, a Hadamard gate is re-applied on the M register, bringing the state $|0\rangle_M \rightarrow \frac{1}{\sqrt{2}}(|0\rangle_M + |1\rangle_M)$ and $|1\rangle_M \rightarrow \frac{1}{\sqrt{2}}(|0\rangle_M - |1\rangle_M)$. The state of the complete system can be then written as in Equation 3.5.

$$\begin{aligned}\psi = &\frac{1}{2\sqrt{2}}(|0\rangle_M + |1\rangle_M) \otimes |1\rangle_D |0\rangle_A |u\rangle_B + \\ &+ \frac{1}{2\sqrt{2}}(|0\rangle_M - |1\rangle_M) \otimes |1\rangle_D |1\rangle_A |u\rangle_B + \\ &+ \frac{1}{2\sqrt{2}}(|0\rangle_M + |1\rangle_M) \otimes \left(\sqrt{1-\frac{c^2}{\lambda^2}}|0\rangle_D + \frac{c}{\lambda}|1\rangle_D\right)|1\rangle_A \sum_i \beta_i |\mu_i\rangle_B + \\ &+ \frac{1}{2\sqrt{2}}(|0\rangle_M - |1\rangle_M) \otimes \left(\sqrt{1-\frac{c^2}{\lambda^2}}|0\rangle_D + \frac{c}{\lambda}|1\rangle_D\right)|0\rangle_A \sum_i \beta_i |\mu_i\rangle_B\end{aligned} \tag{3.5}$$

The resulting state vector encodes the probabilities which encode the solution. Those probabilities can be accessed by repeatedly measuring qubits A and D (thus repeatedly executing the QGP algorithm and measuring A and D after each execution) and finding how many times each of them was 0 or 1.

Then, the results can be derived as follows: first, the solution is conditioned on $D = |1\rangle_D$, which means that the qubit D will be measured, and the results for which D is measured to be in state $|0\rangle_D$ are discarded, as for those the HHL was not successful (see explanation in Equation 2.3.3). The part of the statevector corresponding to $D = |1\rangle_D$ is shown in Equation 3.6. Note that the vector u was expressed in the eigenbasis of the input matrix A as $u = \sum_i \gamma_i |\mu_i\rangle_B$.

$$\begin{aligned}\psi = &\frac{1}{2\sqrt{2}}|0\rangle_M |0\rangle_A \sum_i \left(\frac{c}{\lambda_i}\beta_i + \gamma_i\right)|\mu_i\rangle_B + \frac{1}{2\sqrt{2}}|1\rangle_M |0\rangle_A \sum_i \left(\gamma_i - \frac{c}{\lambda_i}\beta_i\right)|\mu_i\rangle_B + \\ &+ \frac{1}{2\sqrt{2}}|0\rangle_M |1\rangle_A \sum_i \left(\frac{c}{\lambda_i}\beta_i + \gamma_i\right)|\mu_i\rangle_B + \frac{1}{2\sqrt{2}}|1\rangle_M |1\rangle_A \sum_i \left(\gamma_i - \frac{c}{\lambda_i}\beta_i\right)|\mu_i\rangle_B\end{aligned} \tag{3.6}$$

As the qubit A is not measured, the states corresponding to $|0\rangle_A$ or $|1\rangle_A$ are indistinguishable, so the statevector can be written in a form depending merely on M, as shown in Equation 3.7.

$$\psi = \frac{1}{\sqrt{2}}|0\rangle_M \sum_i \left(\frac{c}{\lambda_i}\beta_i + \gamma_i\right)|\mu_i\rangle_B + \frac{1}{\sqrt{2}}|1\rangle_M \sum_i \left(\gamma_i - \frac{c}{\lambda_i}\beta_i\right)|\mu_i\rangle_B \tag{3.7}$$

Therefore, the probability of measuring state $|0\rangle_M$ is $\sum \frac{1}{2}\left(\frac{c}{\lambda_i}\beta_i + \gamma_i\right)^2$, while the probability of measuring state $|1\rangle_M$ is $\sum \frac{1}{2}\left(\gamma_i - \frac{c}{\lambda_i}\beta_i\right)^2$ and the difference of those two probabilities is equivalent to the dot product of vector u and $x = A^{-1}v$:

$$u \cdot (A^{-1}v) = P(M=0) - P(M=1) = \sum_i c\gamma_i \frac{\beta_i}{\lambda_i} \tag{3.8}$$

This derivation proves that applying the procedure brings the system to the state equivalent to the Zhao's M operator. It also shows that measuring only two qubits: M and D yields the solution to the complete problem regardless of the size of the input system.

3.2. Approximate eigendecomposition with QPE

In its essence, the Quantum Gaussian Processes algorithm is an approximation of classical Gaussian processes because of the QPE subroutine used for system diagonalization in the underlying HHL algorithm. As reviewed in the previous chapter, two approximation steps are taken in QPE: matrix exponentiation, and the discretization of the eigenspectrum.

3.2.1. Matrix exponentiation

In principle, QPE can find eigenvalues only for unitary operators, which are complex numbers with absolute value of 1. Oftentimes those eigenvalues are conveniently written as: $\lambda = e^{2\pi\varphi i}$, where $\varphi \in (0,1)$. In order to apply this routine for finding eigenvalues of a Hermitian matrix A, HHL algorithm exploits an exponentiation artifact. This artifact relies on two facts:

1. It can be shown that hermitian matrix A in exponential form is unitary, i.e. $U = e^{iAt}$
2. It can be shown that if matrix A has eigenvalues: $[\lambda_0, \lambda_1, ...\lambda_n]$, then the exponential form e^A has eigenvalues: $[e^{\lambda_0}, e^{\lambda_1}, ...e^{\lambda_n}]$.

Using those two facts, HHL applies exponentiation of the input matrix A to obtain a unitary $U = e^{iAt}$ with eigenvalues $[e^{i\lambda_0 t}, e^{i\lambda_1 t}, ...e^{i\lambda_n t}]$. By setting the evolution time to $t \sim \frac{2\pi}{\lambda_{max}}$, the eigenvalues, and specifically ratios of eigenvalues $\frac{\lambda_j}{\lambda_{max}}$ can be encoded in φ. In such way, finding ϕ with QPE allows to retrieve the eigenvalues of A. The technical details of encoding the eigenvalues are discussed further in subsection 3.2.2.

In the context of quantum mechanics, the exponentiation is interpreted as a wave function according to the Schrodinger equation (Equation 3.9), therefore it is also called Hamiltonian simulation.

$$|\psi\rangle = e^{iAt}|\psi_0\rangle \tag{3.9}$$

Formally, matrix exponentiation can be defined only by a Taylor series, as shown in Equation 3.10, therefore it cannot be carried out exactly.

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!} \tag{3.10}$$

For this reason, an approximation scheme is used. In Qiskit this operation can be performed with Suzuki-Trotter expansion[30]. As a default QGP setting, the 2^{nd} order Suzuki-Trotter exponentiation scheme was chosen, as it provides a balance between the computational expense and the quality of approximation. The scheme is given by Equation 3.11.

$$e^{(A+B)x+O(x^3)} = e^{\frac{x}{2}A} e^{xB} e^{\frac{x}{2}A} \tag{3.11}$$

In QPE this scheme is used to approximate the evolution of Hamiltonian over some time t by splitting it on a sequence of smaller (and easier to implement) Hamiltonians, and simulating the evolution over some smaller time steps t/r (where r is the number of time slices - also called the Trotter number[30]). In this case, expanding over r time slices can be formally written as in Equation 3.12

$$e^{(A+B)\frac{t}{r}+O\left(\frac{t^3}{r^3}\right)r} = \left[e^{\frac{t}{2r}A} e^{\frac{t}{r}B} e^{\frac{t}{2r}A}\right]^r \tag{3.12}$$

Note that with such implementation, the unitary U constructed with Suzuki-Trotter scheme is only an approximation to the exponentiated matrix , i.e. $U \approx e^{iAt}$, therefore the eigenvalues of U will be only approximation to $e^{i\lambda_j t}$.

3.2.2. Discretization of eigenspectrum

Given a unitary operator U with eigenvalues $e^{2\pi i\varphi}$, QPE allows to approximate the phase φ with a k-bit long binary expansion $\tilde{\varphi}$ [51], resulting in approximation error ϵ_φ, specified in Equation 3.13.

$$\epsilon_\varphi = \varphi - \tilde{\varphi} \qquad 0 \leqslant \epsilon_\varphi \leqslant \frac{1}{2^k} \tag{3.13}$$

The approximation can take only discrete values: $\tilde{\varphi} \in \left(0, \frac{1}{2^k}, \frac{2}{2^k}, ..., \frac{2^k-1}{2^k}\right)$, therefore this step can be considered as discretization of the eigenspectrum. To encode the eigenvalues λ_j of the input matrix A in phase φ, the evolution time is set to:

$$t_{evo} = \left(\frac{2\pi}{\tilde{\lambda}_{max}}\right)\frac{2^k-1}{2^k} \tag{3.14}$$

The factor $\left(\frac{2^k-1}{2^k}\right)$ was added to minimize the error in resolving the maximum eigenvalue[1] and to bound φ strictly below 1 which prevents over-flowing[2]. With evolution time set as in Equation 3.14, the eigenvalues become encoded in φ as shown in Equation 3.15.

$$\varphi = \left(\frac{\lambda_j}{\lambda_{max}}\right)\frac{2^k-1}{2^k} \qquad \varphi \in \left[0, \frac{2^k-1}{2^k}\right] \tag{3.15}$$

In this case, the discrete approximation of the eigenvalues takes form of Equation 3.16.

$$\tilde{\varphi} = \left(\frac{\tilde{\lambda}_j}{\lambda_{max}}\right)\frac{2^k-1}{2^k} \qquad \frac{\tilde{\lambda}_j}{\lambda_{max}} \in \left(\frac{0}{2^k-1}, \frac{1}{2^k-1}, \frac{2}{2^k-1}, ..., \frac{2^k-1}{2^k-1}\right) \tag{3.16}$$

Finally, the error approximating phase: ϵ_φ from Equation 3.13 can be transformed into error of eigenvalue approximation ϵ_λ, according to Equation 3.17.

$$\lambda - \widetilde{\lambda_j} = \epsilon_\lambda \qquad 0 \leqslant \epsilon_{\widetilde{\lambda_j}} \leqslant \frac{\lambda_{max}}{2^k-1} \tag{3.17}$$

Note that according to this scheme, eigenvalues $\lambda_j < \frac{\lambda_{max}}{2^k-1}$

[1] Note that without the correcting factor, $\phi = \frac{\lambda_{max}}{\lambda_m ax}$ would be approximated with $\tilde{\varphi} = \frac{2^k-1}{2^k}$, which would result in a maximum error (i.e. $\epsilon_\varphi = \frac{2^k-1}{2^k}$)

[2] Note that in case that $\varphi = 1$, the unitary eigenvalue is $e^{2\pi i}$, which equivalent to e^{0i}, which corresponds to $\varphi = 0$.

Eigenvalue reciprocal

Approximating small eigenvalues as 0 becomes problematic in the next subroutine of HHL following the QPE, i.e. in the controlled rotation of an ancilla qubit. The rotation is applied to evolve the ancilla qubit D from state $|0\rangle_d$ to $|\psi\rangle = \left(\sqrt{1-\frac{c^2}{\widetilde{\lambda}^2}}|0\rangle_D + \frac{c}{\widetilde{\lambda}}|1\rangle_D\right)$, as required by HHL. The operation can be carried out by applying the R_y operator shown Equation 3.18.

$$R_y(\theta) = \begin{bmatrix} cos(\theta/2) & -sin(\theta/2) \\ sin(\theta/2 & cos(\theta/2) \end{bmatrix} \tag{3.18}$$

The angle $\frac{\theta}{2}$ is set according to Equation 3.19:

$$cos(\theta/2) = \left(\sqrt{1-\frac{c^2}{\widetilde{\lambda}^2}}\right) \qquad sin(\theta/2) = \left(\frac{c}{\widetilde{\lambda}}\right) \tag{3.19}$$

In case $\lambda_j < \frac{\lambda_{max}}{2^k-1}$ the approximation $\widetilde{\lambda_j} = 0$, which results in singularity in Equation 3.19. To circumvent this problem, the rotation is implemented, with a modified R'_y, given by Equation 3.20.

$$R'_y(\theta) = \begin{cases} \widetilde{\lambda}_j = 0: & \theta = 0, & sin(\theta/2) = 0 \\ \widetilde{\lambda}_j > 0: & \theta = 2arcsin\left(\frac{c}{\widetilde{\lambda}_j}\right), & sin(\theta/2) = \frac{c}{\widetilde{\lambda}_j} \end{cases} \tag{3.20}$$

The constant c needs to be set in such way, that it bounds the $\frac{c}{\widetilde{\lambda}}$ below 1. On the other hand to maximize the probability of success ($|1\rangle_D$) this term should be as close to 1 as possible. Satisfying those two constraints is achieved by setting c to the minimum resolvable eigenvalue, i.e. $c = \frac{\lambda_{max}}{2^k-1}$.

3.2.3. Inducing low-rank approximation

Implementation of the controlled rotation according to Equation 3.20 ensures that for small eigenvalues, i.e. $\lambda_j < \frac{\lambda_{max}}{2^k-1}$ for which $\lambda_j = 0$, the state of ancilla qubit D remains unchanged, i.e. $|\psi\rangle = |0\rangle_D$. In HHL this state flags the part of the output for which the matrix inversion was not successful. Therefore implementing the controlled rotation according to Equation 3.20 suppresses inverting the matrix on a subspace corresponding to the smallest eigenmodes. Consequently, the HHL effectively implements an approximate matrix inversion on a subspace, which formally can be written as shown in Equation 3.21.

$$A^{-1} \approx \sum_{j=0}^{m} \mu_j \widetilde{\lambda}_j^{-1} \mu_j^T \tag{3.21}$$

Where m is the number of eigenmodes contributing to the approximation. In QPE parameter m can be defined as shown in Equation 3.22.

$$m = max(j) \ni \lambda_j \geqslant \frac{\lambda_{max}}{2^k-1} \tag{3.22}$$

In this form the approximate inverse is similar to inversion of a low-rank approximation of A constructed with Principal component analysis (PCA), shown in Equation 3.23. In this case, the PCA

approximation is constructed by eigendecomposition of matrix A, and selecting first m eigenmodes, which results in best possible rank-m approximation (can be shown by the means of Frobenius norm [49]).

$$\widetilde{A} = \sum_{j=0}^{m} \mu_j \lambda_j \mu_j^T \tag{3.23}$$

Comparing the two expressions: the quantum-induced low-rank approximation in Equation 3.21 and the classical Equation 3.23, it can be concluded that the HHL allows to induce a quantum version of PCA approximation, equivalent to that describe in [43]. That can be done by choice of parameter k such that a part of the eigenspectrum is approximated as 0, i.e. $m < n$, where $m = max(j) \ni \lambda_j \geqslant \frac{\lambda_{max}}{2^k - 1}$.

Nevertheless, PCA (3.21) and qPCA (3.21) are not strictly equivalent, due to two differences:

1. In qPCA the eigenvalues are approximated up to the accuracy given by the eigenregister resolution: $\widetilde{\lambda} = \lambda - \epsilon_\lambda$, while in PCA the eigenvalues are exact.

2. In classical PCA the user has explicit control over the parameter m, while in qPCA this parameter can be controlled only implicitly via parameter k, as shown in Equation 3.22.

The general concept of using low-rank approximation in HHL was proposed already in the original publication [29]. In this case, the researchers proposed a mechanism for inducing approximate matrix inversion on a subspace corresponding to the well-conditioned part of the matrix, as an approach for dealing with ill-conditioned systems. In that sense, HHL is similar to the classical methods of low-rank approximations and subspace inversion, which rely on the same concept of filtering out the ill-conditioned part of the matrix (i.e. the higher order eigenmodes) by projecting the matrix on a low-rank subspace that makes it easier to compute the inverse [11, 52].

In Gaussian Processes, however, the capability of inducing low-rank approximation could be exploited in a new context, namely for inducing sparsity of the system, making the QGP algorithm analogous to the classical Sparse Gaussian Processes. This concept is explored further in more details in Section 3.5.

3.3. Performance and verification

Each of the two approximation schemes in QPE is controlled by a parameter - in case of the matrix exponentiation it is the number of time slices r and for the eigenvalue discretization it is the size of the eigenvalue register k. Both parameters play crucial role in balancing the accuracy and the computational costs. Finding the combination that minimizes the computational efforts, while keeping the errors sufficiently small is paramount for optimal use of the algorithm.

3.3.1. Testing approach

The algorithm performance was measured by means of error and size of the corresponding circuit, which quantify respectively, the accuracy and the computational cost. Those metrics were measured for different settings of the approximation parameters k and r, as well as for different conditions specified by the condition number κ and size of the system n. The summary of the testing parameters is shown in Table 3.1.

Table 3.1: Summary of algorithm parameters considered in testing.

Test Parameters		
Input parameters	k	Size of eigenvalue register
	r	Number of time slices
Input conditions	n	Size of the system
	κ	Condition number of the system
Output parameters	ϵ	Error defined as $\epsilon = \frac{\lvert\mu_{GP}-\mu_{QGP}\rvert}{\lvert\mu_{GP}\rvert} \cdot 100\%$
	w	Circuit width - number of qubits
	d	Circuit depth - number of gates

Sample GP systems

For each n, a number of sample GP systems was prepared, defined by the input matrix K and the input vectors k_* and y. Out of those, three samples were selected to span a range of condition numbers κ. Then, each of the sample systems was solved by the QGP algorithm with a range of settings for k and r to generate the error contour plots. Comparing the error contours for systems with different settings (k and r) and conditions (n and κ) allowed to distinguish different numerical effects: size-dependent (n), system-dependent (κ) as well as originating from the approximations (k, r).

Testing systems of different sizes (n) allowed to distinguish any potential size-related numerical effects. The system sizes selected for testing were $n = 2$, 4 and 8, where the maximum size was limited by the computational cost of simulating the algorithm on classical computers.

Testing platform

Testing was done on a quantum simulator, specifically the QASM simulator that is a built-in Qiskit[1]. The main reason for this choice is insufficient performance of contemporary quantum computers. Nevertheless, by testing the algorithm on a simulator the influence of all the errors originating from quantum hardware could be eliminated, allowing to distill the errors that come merely from the algorithmic side, i.e. the approximations. The third reason is that the simulator gives access to a complete state-vector, that is expressing the state of the system once the algorithm is executed. When run on a quantum computer, this state-vector remains inaccessible, as it is in a quantum state, and it can only be reconstructed by a series of measurements. Those, however are also affected by errors, leading to limited accuracy.

3.3.2. Approximation parameters: error contours

Eigenregister resolution k

The discretization of the eigenspectrum is controlled by the parameter k, which corresponds to the size of the register for storing the eigenvalues. The register spans a space of 2^k states allowing to store 2^k values. In principle, the number k can be freely adjusted, but in practice it is limited by the available resources, specifically, the number of qubits.

The expectation is that increasing the resolution k allows for resolving the eigenspectrum more accurately, leading to smaller errors. The effect of the QPE-induced low-rank approximation, however,

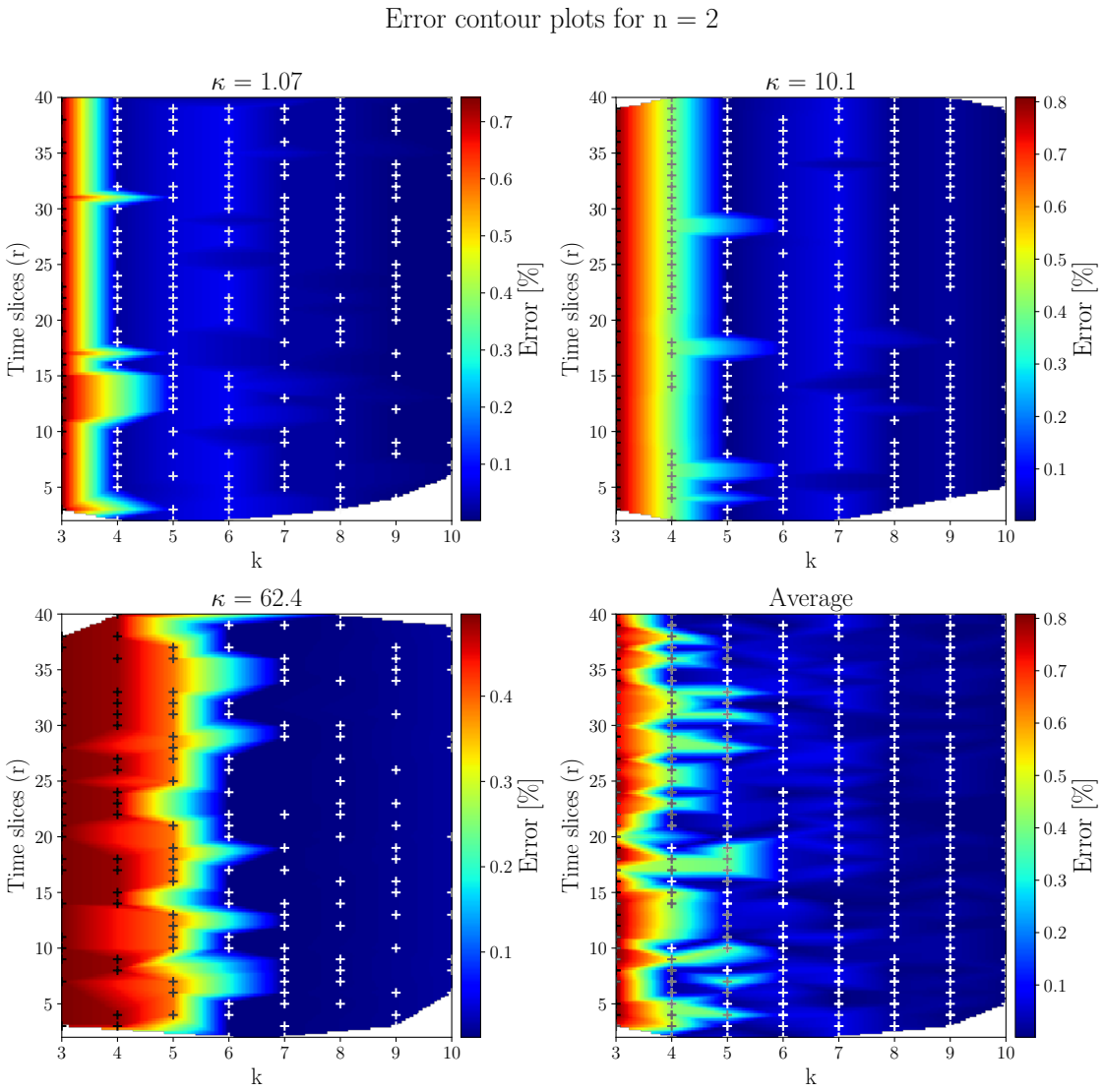


Figure 3.2: Numerical results of the algorithm performance tests: interpolated relative error obtained for different $k-r$ settings, for system size $n=2$. The results were obtained for three different GP systems with different condition number κ.

plays a major role as discussed in subsection 3.2.3. It is expected that truncating the eigenmodes affects the quality of the solution more than the inaccuracies in approximating the resolved eigenvalues. Therefore the error is expected to be an order of magnitude higher for settings which induce a low rank approximation.

The results of simulations verify this hypothesis. As shown in the error-contour plotted in Figure 3.2, the error converges with increasing k. In this case the error values are in order of 1% for low k and drop below 0.1% for higher k. Comparing the three plots corresponding to the systems with three different condition numbers κ, one can notice how the error drop frontier shifts to higher k values for systems with higher κ. This effect can be directly related to the low-rank approximation, where the k at which the error drop occurs, is roughly $k \sim log_2(\kappa)$. This means that once k is sufficiently high to resolve the complete eigenspectrum, i.e. $k > log_2\left(\frac{\lambda_{max}}{\lambda_{min}}\right)+1$, where $\kappa = \frac{\lambda_{max}}{\lambda_{min}}$, which yields: $k > log_2(\kappa)+1$. In this case the QGP includes all the eigenmodes and is no longer the low-rank approximation, converging to the full GP solution.

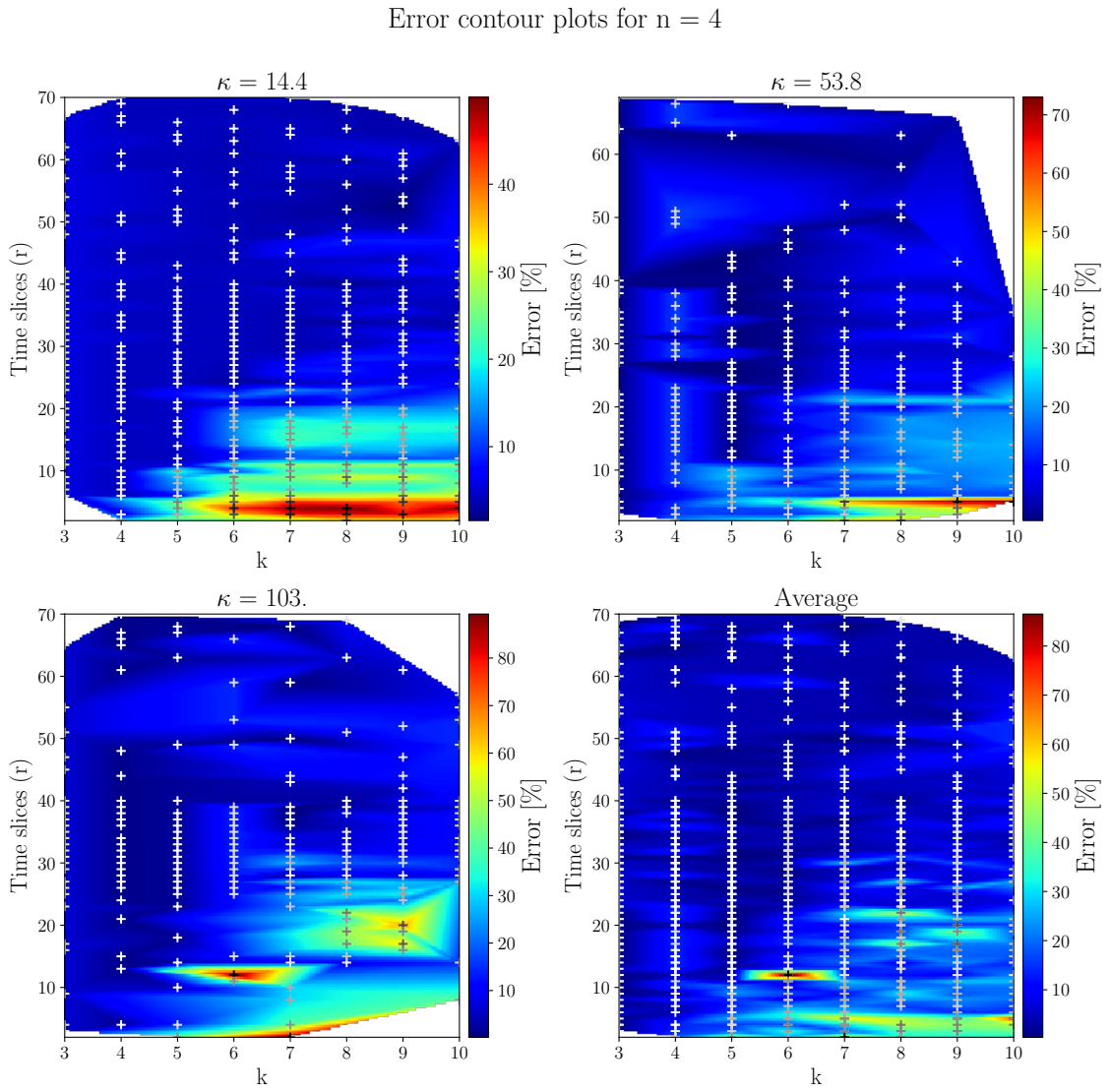


Figure 3.3: Numerical results of the algorithm performance tests: interpolated relative error obtained for different $k-r$ settings, for system size $n = 4$. The results were obtained for three different GP systems with different condition number κ.

A similar behavior is found for systems $n = 4$, shown in Figure 3.3. In this case, however, a peculiar effect can be seen, as for a small number of time slices the error is growing with k, reaching values as high as 50%. This unexpected behavior of growing error can be explained as an effect of interaction between the approximation of the matrix exponentiation and the discretization of the eigenspectrum. For low number of time slices, the matrix exponentiation (discussed in the next section) is less accurate. As a result, the eigenvalues of the unitary resulting from this approximate exponentiation ($U \approx e^{-iAt}$) might differ from the eigenvalues of the input matrix A. The lower the r, the larger differences might occur.

The growth of errors with the eigenvalue resolution can be explained even further by looking directly into the output of QPE, i.e. the approximated eigenspectrum. Figure 3.5 shows such an example of eigenspectrum approximated with QPE for two different settings. In an ideal case there should be a single peak for each eigenvalue (in the shown example at $\lambda = 200$). This appears to be the case in the upper histogram, the one corresponding to $k = 5$. In the lower histogram, the peaks are affected by

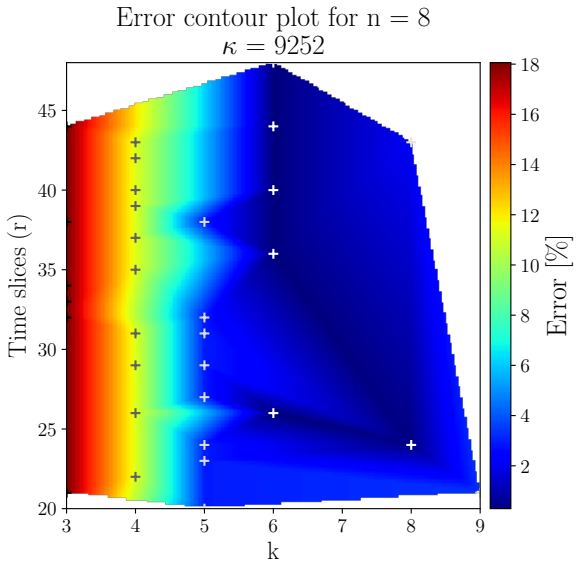


Figure 3.4: Interpolated error for different r and k settings, for system size $n = 8$

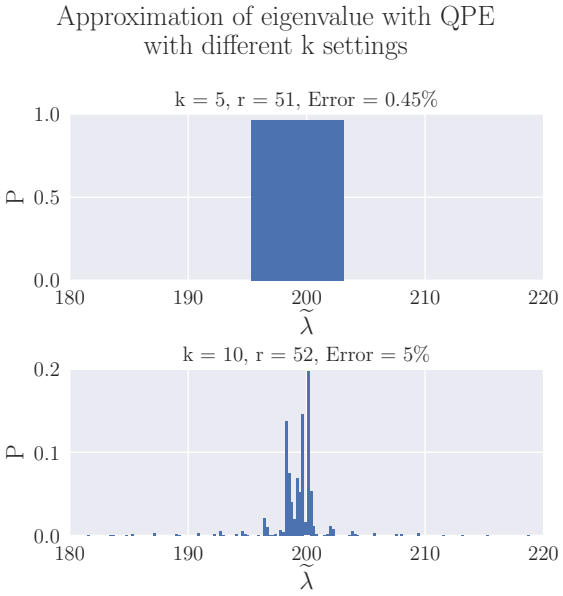


Figure 3.5: Comparison of eigenspectrum approximation obtained with QPE for different eigenvalue resolution k

some noise, which results in multiple peaks, spread around the eigenvalue. The origin of this effect is most likely the approximate exponentiation, as discussed in the previous paragraph. While with higher resolution those inaccuracies can be distinguished, with lower resolution the accuracy is not sufficient to resolve this noise, which results in an effect similar to filtering. This example shows that lower k has an effect of providing with a certain level of robustness, or filtering that counteracts the inaccuracies due to the approximate matrix exponentiation. This effect eventually results in lower errors, which explains the unexpected error growth in Figure 3.3.

The k dependence was also tested for system with size $n = 8$. The results are shown in Figure 3.4. In this case the error decreases with k to levels below 2%, without any anomalies. The data-set in this example, however, is relatively small comparing with the previously discussed cases. In general it is expected that the previously discussed effects can occur for larger system sizes as well, and lack of those anomalies in Figure 3.4 is only a matter of limited data. The computational cost of simulating $n = 8$ is already quite high (single simulation takes several hours on a cluster), and the data here is shown only to illustrate the general trend, and convergence.

Time slices r

The error of this 2^{nd} order scheme is expected to be $O\left(\frac{t^3}{r^2}\right)$. This theoretical error scaling was fitted with data as shown in Figure 3.6, and plotted with the green dashed line. In general the fitted curve provides a good representation of the global trend in the data, showing a reasonable agreement with the theoretical $O\left(\frac{t^3}{r^2}\right)$ scaling.

Figure 3.7 shows another important feature: the error fluctuations, particularly well visible on plots corresponding to $k = 8$ and $k = 9$. Those fluctuations are the direct effect of exponentiation with the Suzuki-Trotter scheme. Similar effects were mentioned by Suzuki and Hatano in [30], where they demonstrated the problem on two practical examples: simulation of spin precession and chaotic dynamics. In both problems, the exponentiation scheme resulted in considerable oscillations in energy, as shown in Figure 3.7, which are conjectured here to have the same nature of the behavior observed for QGP.

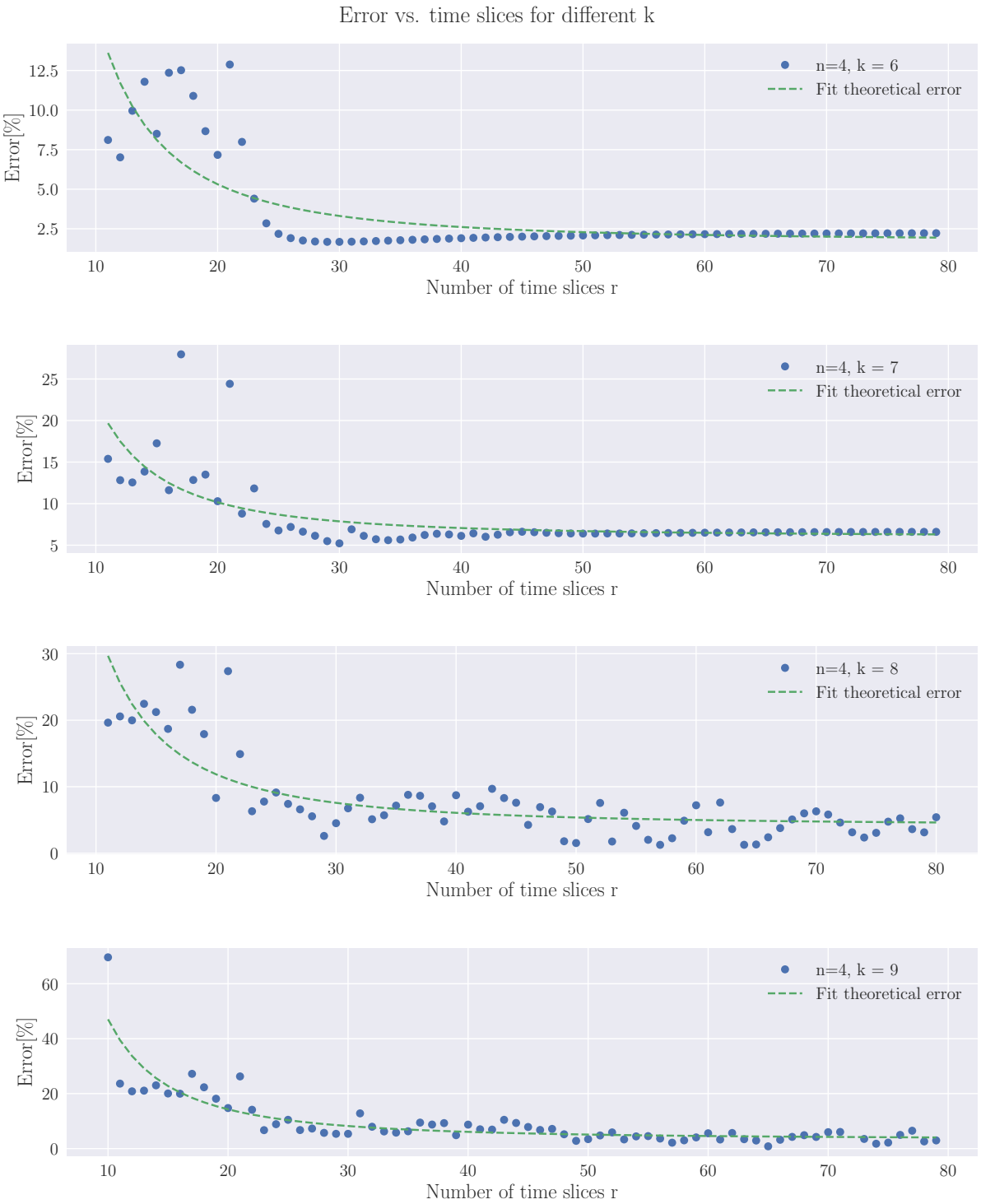


Figure 3.6: Oscillation of QGP error with changing number of time slices (r). The data observed for a 4×4 system with the condition number $\kappa = 55$. For different k the oscillations assume different wave-lengths. The oscillation indicates that the error originates from the Suzuki-Trotter approximation.

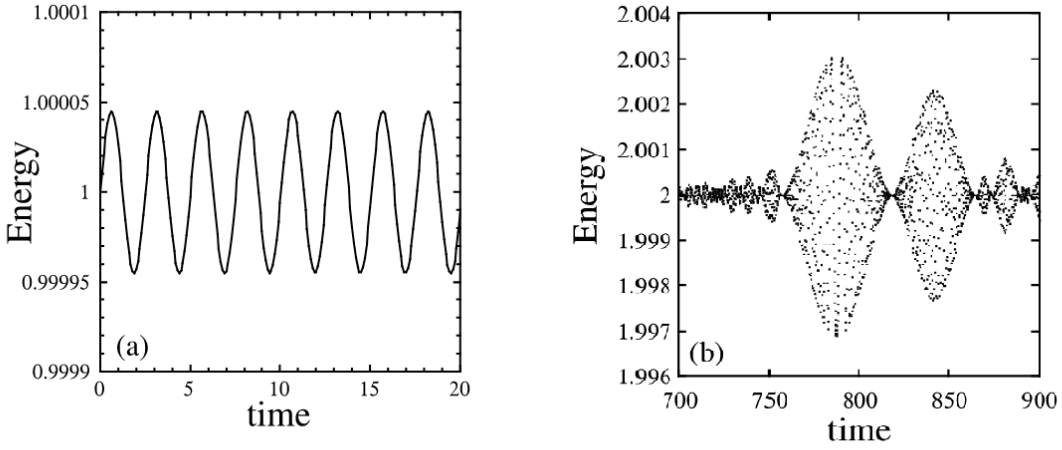


Figure 3.7: Energy oscillation due to Trotter-Suzuki approximation observed in simulations of spin precession and chaotic dynamics [30]

3.3.3. Tests on Quantum Computers

Besides all the tests conducted on a simulator, the algorithm was also tested on real quantum computers, for demonstration purposes. Quantum computers used for this tests were the IBM's devices which can be accessed via IBM's *Quantum Experience* website[64]. At the time of testing, the best performance in terms of error rates was provided by the *IBM-qx2 Yorktown* device, with specifications shown in Figure 3.8. The performance of this quantum computer is very limited. It provides only five qubits, limiting the width of the circuit, and it has relatively high error rates that restrict the circuit depth.

In order to cope with those constrains, while also minimizing the influence of errors (originating from the device), a special test case was prepared that implemented the QGP algorithm with minimal use of resources. For this purpose, a 2×2 GP system was prepared, with $\kappa \approx 2$. The input vectors, were adjusted such that solving the system with the lowest possible settings ($r = 1$ and $k = 2$) on a noiseless simulator the error was only 2% (which is purely due to the QGP approximation).

The quantum circuit corresponding to this system was thoroughly optimized, through several transpilation passes, using Qiskit transpiler[1]. This resulted in a circuit consisting of five qubits and approximately 200 basic gates (CNOT and U3). Despite the heavy optimization, circuit length turned out to be still impractically long, given the error rates of the gates in the device, which resulted in errors in order of 100%.

3.4. Computational cost

Next to accuracy, the computational cost is the second crucial metric of an algorithm performance. For the QPG algorithm the computational cost is studied following two approaches: 1) analytical complexity analysis and 2) numerical simulation. The results from the two approaches are then combined to build a cost model of the algorithm by fitting the numerical data with analytically derived scaling functions.

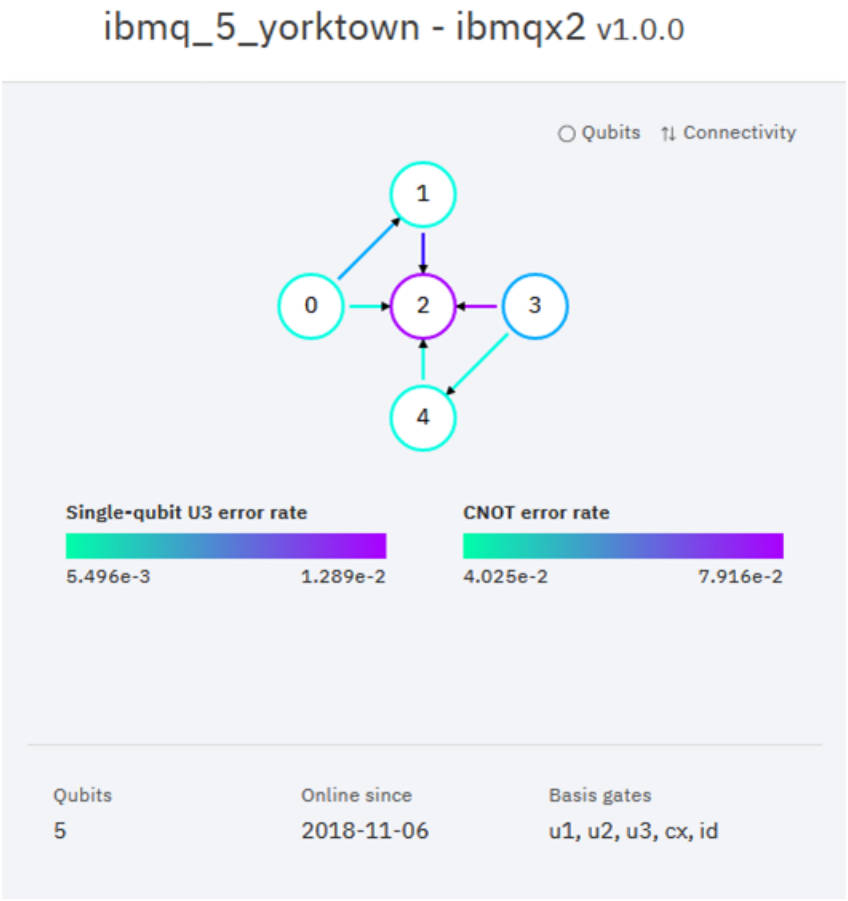


Figure 3.8: Overview of basic specifications of *IBM-qx2 Yorktown* quantum computer: schematic of qubit connectivity and statistics for the gate error rates [64].

3.4.1. Complexity analysis

Complexity of QGP is naturally dominated by the complexity of HHL, which is the major part of QGP. Qiskit implementation of HHL differs in some details from the HHL algorithm originally proposed by Harrow, Hassidim and Lloyd in reference [29]. Most importantly the Qiskit version lacks the implementation of the efficient Hamiltonian simulation that provides the exponential $O(log(n)s^2t)$ scaling. Instead, Qiskit uses an alternative method which decomposes an s-sparse Hamiltonian (that is one having s entries per row) size $n \times n$ onto $O(snlog(n))$ local Hamiltonians by decomposing the matrix with all possible tensor products (Pauli terms).

Secondly, Qiskit implementation of HHL does not use amplitude amplification, which in the original HHL complexity analysis increases the scaling from linear to quadratic with κ. In case of QGP application, however, it is expected that the amplitude amplification could be applied more efficiently after the measurement operator is applied, to boost the probability corresponding to state of success ancilla qubit being in state $|1\rangle$.

Accounting for those two differences, the scaling of the implemented algorithm is determined by analyzing each subroutine, which results in the following terms:

- **State preparation** - the state preparation is neglected, as the HHL algorithm assumes that the step is carried out with qRAM [29].

- **Quantum Phase Estimation** - this step consists of the following substeps:
  - **Hamiltonian simulation** - in current Qiskit implementation the simulated Hamiltoninan is decomposed on $O(ns\log n)$ local Hamiltonians (via Pauli matrices), where s is the sparsity of the matrix (number of entries per row), therefore this step scales as $O(ns\log n)$. Nevertheless, it has been shown that this step can be implemented with more efficient Hamiltonian simulation techniques that scale as $O(s^2\log(n))$. Furthermore, the evolution is carried out in r time steps, which yields overall complexity of $O(ns\log nr)$ for the current implementation and $O(s^2\log(n)r)$ in case the Hamiltonian decomposition is improved.
  - **Mapping the eigenvalues with controlled unitaries** takes exactly k steps. Up until this point the complexity of QPE is $O(krns\log n)$ and $O(s^2\log nkr)$ for the current and the potentially improved version respectively.
  - **Quantum Fourier Transform** that follows the controlled unitaries takes k^2 steps, resulting in additional k^2 overhead.
- **Eigenvalue inversion** - the step that finds inverses of the eigenvalues is implemented with Qiskit partial table lookup technique, which implements the step with 2^k gates. Although the number of gates here scales exponentially with the eigenregister resolution, it will not be critical to the algorithms complexity, For larger systems (that is large n) the overall complexity of the algorithm will be dominated by the terms scaling with n.

Putting all the terms together, the algorithm can be executed with $O(krns\log n + k^2 + 2^k)$ steps, and in case the efficient Hamiltonian simulation is implemented, the scaling reduces to $O(krs^2\log n + k^2 + 2^k)$.

Both complexity terms show unfavorable scaling with parameter k. In practice, however, the terms depending on k add only finite overhead to the cost, as only finite accuracy is required. For most practical applications $k \sim O(10-50)$ [3] suffices to provide the necessary eigenvalue resolution. Therefore, for larger systems the cost will be dominated by the terms scaling with n, and other terms will be suppressed, resulting in $O(krns\log n)$ and $O(krs^2\log(n))$.

3.4.2. Empirical model of circuit scaling

Based on complexity analysis and experimental data, a cost model of the algorithm was developed. The model estimates the circuit depth in terms of number of basic single and 2-qubit gates, which corresponds directly to the computational cost of the algorithm. The numerical data was generated by compiling QGP circuits for four different system sizes n with different settings of parameters k and r, with ranges specified in Table 3.2. This allowed to obtain the depth of the circuit for each combination of parameters. Furthermore, to eliminate the influence of variations in circuit length due to specifics of the input system (e.g. some input matrices might be easier to decompose in the time-slicing routine), for each of the four considered system sizes the experiments were repeated at three different randomly generated input systems. This resulted in three datasets for each n, which yields in total 12 datasets.

This data was then fitted to a model function, shown in Equation 3.24, which was obtained by assuming that the true cost function can be expressed as a linear combination of the theoretical complexity. Fitting the data allowed to estimate the model parameters $a_1 - a_4$.

[3] Floating point numbers typically used in classical scientific computing have precision $O(1e-15)$ (e.g. Numpy [72]). Matching this accuracy with QPE requires setting $k = 50$.

Table 3.2: Parameters and ranges of values used for numerical

Parameter	Range of values
k	2-10
r	10-80
n	2, 4, 8, 16

$$d = \alpha_1 k r log(n) n^2 + \alpha_2 \cdot k + \alpha_3 \cdot k^2 + \alpha_4 \cdot 2^k \tag{3.24}$$

Components of the cost function

The cost function can be split into two components: the first term, which corresponds to the Hamiltonian simulation, and the overhead terms (α_2-α_4), which correspond to the Fourier transform used in QPE and to the inversion of the eigenvalues. While the first part dominates the overall cost of the algorithm, the overhead cost can be significant too, but they are commonly neglected by the quantum computing community [12].

The currently implemented Hamiltonian simulation relies on sub-optimal decomposition on tensor products of all possible permutations of Pauli terms, which in scales as $log(n)n^2$ (verified with numerical tests), prohibiting the exponential speed-up, therefore ultimately, it needs to be replaced with a more efficient method. While interesting for this work, the cost of this sub-optimal routine is therefore of little relevance for quantum community. The overhead terms, on the other hand, are expected to remain unchanged for the final exponentially faster version of QGP. Therefore, the results of this work provide insight for the quantum computing community on the actual cost of those neglected terms, thus contributing to the discussion on actual cost of the quantum algorithms and reaching quantum advantage.

Inference of the model parameters

Fitting the model with all 12 datasets at once, however, resulted in poor estimate of the coefficients α (e.g. some where found to be negative, which is unreasonable in this context). Therefore, to obtain a more reliable estimate of the model coefficients, each data-set was fitted separately, and considered a distinct sample. The parameter was obtained by taking average of parameters obtained with the 12 different fits. Similarly, the uncertainty of the estimated parameters was estimated by taking standard deviation of values from the 12 fits.

The final values are shown in Table 3.3. This approach indeed produced more consistent estimates of the parameters, as revealed by the relatively low values of standard deviation. The only exception is the parameter α_1.

Table 3.3: Model parameters with their corresponding uncertainty

$\overline{\alpha_1} \pm \sigma_{\alpha_1}$	$\overline{\alpha_2} \pm \sigma_{\alpha_2}$	$\overline{\alpha_3} \pm \sigma_{\alpha_3}$	$\overline{\alpha_4} \pm \sigma_{\alpha_4}$
5.93± 0.874	51.714 ± 0.31	19.88 ± 0.02	8.723 ± 0.002

It was observed, that for the three subsets corresponding to $n = 16$, the coefficient α_1 was significantly larger than for the remaining subsets (i.e. $\alpha_1 \approx 7.2$ compared to 5.3 for the remaining n). This

difference is most likely originating from the decomposition of the input matrix on Pauli terms, which for larger matrices appears to be sub-optimal. For small system-sizes ($n = 2$ to 8) the Pauli decomposition routine (a preparatory step of the the Hamiltonian simulation in QPE) leverages symmetry of the input matrix, which results in optimal matrix decomposition onto exactly $n^2 + \frac{n}{2}$ terms. For larger systems (n = 16, 32) however, it was found that this relationship tends to underestimate the number of terms resulting from the decomposition routine (which was verified with numerical tests). As the currently implemented Hamiltonian evolution, however, ultimately needs replacement with a more efficient method to ensure the exponential speed-up of the overall algorithm (as the current sub-optimal implementation compromises the performance), it was decided that further investigation of those effects is beyond the scope of this work.

Model uncertainty

The uncertainty of the general model was estimated using uncertainties of the obtained parameters. This was done using Equation 3.25. In this case, uncertainties of the parameters $\Delta\alpha$ correspond to the standard deviation, as listed in Table 3.3.

$$\Delta f = \sqrt{\sum_{i=1}^{4} \left(\frac{\partial f}{\partial \alpha_i} \Delta \alpha_i \right)^2} \tag{3.25}$$

Substituting the cost function, and differentiating with respect to α leads to Equation 3.26, where Δd designates uncertainty in circuit depth.

$$\Delta d = \sqrt{\left(krlog(n)n^2\sigma_{\alpha_1}\right)^2 + \left(k\sigma_{\alpha_2}\right)^2 + \left(k^2\sigma_{\alpha_3}\right)^2 + \left(2^k\sigma_{\alpha_i}\right)^2} \tag{3.26}$$

Model verification

The final cost model is compared with one of the data-sets, specifically corresponding to $n = 8$. The model reasonably approximates the cost of the circuit - the discrepancies between the model and the experimental data remain within 15% for most of the cases, with only few local spikes reaching 18%.

The prediction of the uncertainty, however, is consistently over-predicted (approximately by 100%), making it a conservative estimate. Nevertheless, having more conservative uncertainty intervals makes the model more reliable in extrapolation for larger system sizes.

Cost predictions

Finally, the model was used to predict the cost of the current implementation, given the size of the input system, as shown on the left plot in Figure 3.10. The cost was predicted for two configurations of the QGP parameters: 'low settings' which correspond to $k = 8$, $r = 40$, and 'high settings' with $k = 30$ and $r = 40$. The model uncertainty is also shown (as the shaded interval around the line).

The plot on the right presents an estimate of the QGP overhead costs, which depends only on parameter k (according to the model function Equation 3.24). This parameter can be also directly related to the condition number of the solved system, which was marked on the top horizontal axis of the plot. The uncertainty of the overhead cost is not visible on the plot.

Comparing the overhead costs from the right plot to the complete QGP cost on the right, it can be concluded that for small systems (n<100,000) those additional costs can be of the same order of magnitude, thus becoming non-negligible. This is illustrated with the example of the algorithm with

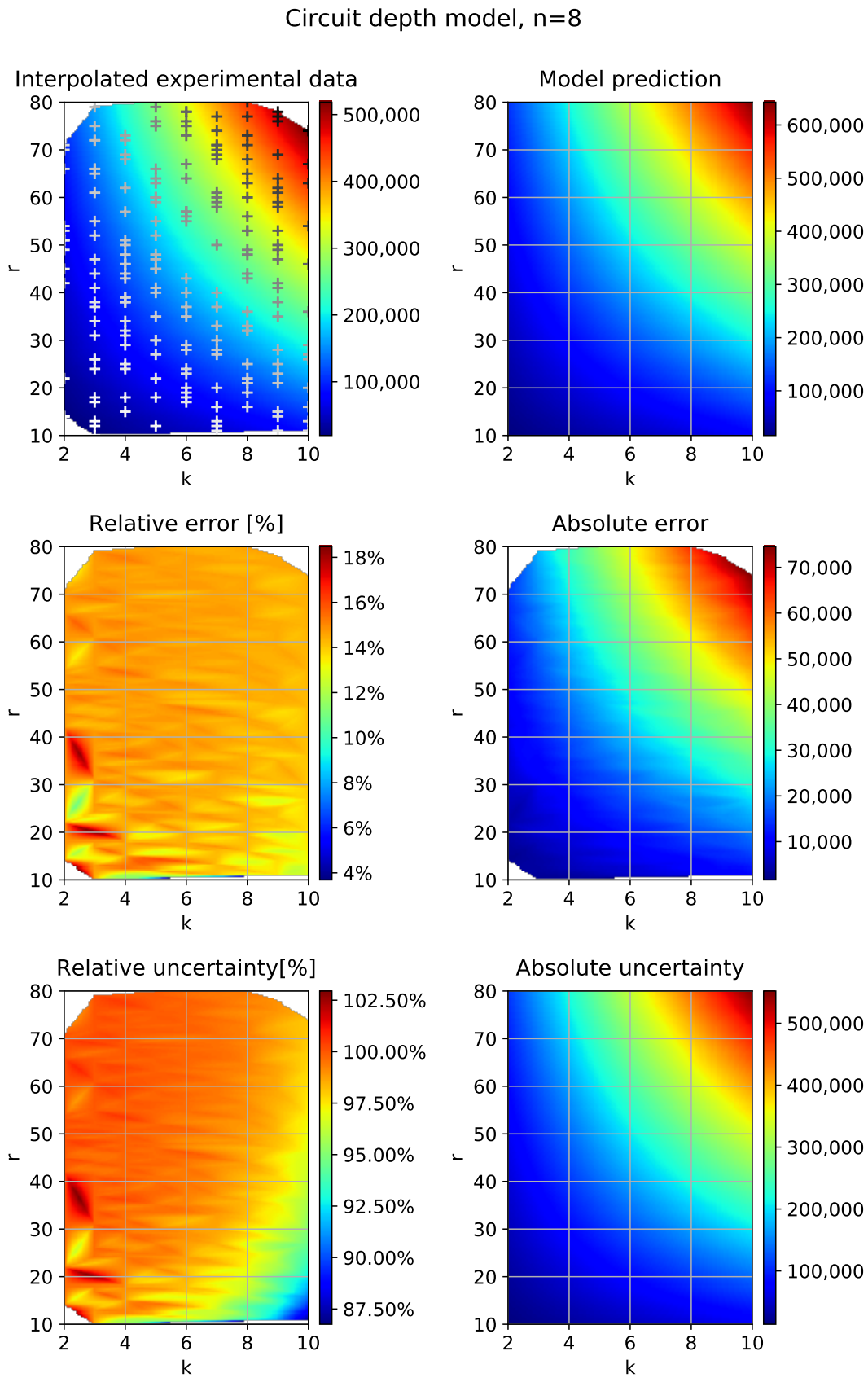


Figure 3.9: Depth of the circuit: comparison of the experimental data and the inferred cost model.

'high-settings', were large k results in overhead costs that dominate the overall cost of the algorithm, which results in plateau for $n < 300$. For larger systems, however, the overhead indeed becomes orders of magnitude smaller.

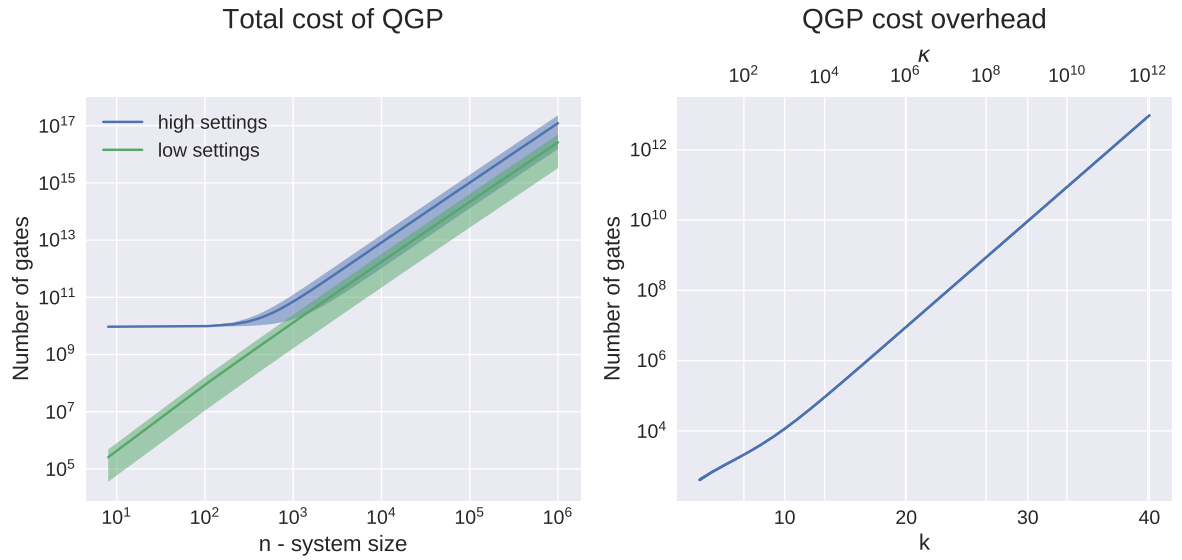


Figure 3.10: Prediction of the depth of QGP circuit obtained with numerical cost model.

3.5. Inducing Sparsity: qPCA vs classical Nystrom method

As discussed in subsection 3.2.2, the discretization of the eigenspctrum with QPE can be exploited to induce a low-rank approximation and matrix inversion on a subspace. This is realized by setting the value of parameter k such that the part of the eigenspectrum is approximated as 0, which induces a qPCA approximation in HHL. In the context of Gaussian processes this property becomes particularly interesting, as it relates QGP algorithm to the classical sparse Gaussian processes.

3.5.1. Low-rank approximation in context of GPs

The concept of low-rank approximation is central to sparse Gaussian processes, where approximating the system provides the means to reduce the computational cost of inference. The two most commonly proposed methods for constructing such approximations are either via eigendecomposition of the original system (equivalent to PCA), or by the Nystrom method [40, 75, 76].

I. Approximation via eigendecomposition: PCA

Classically, the most straightforward approach to construct a low-rank approximation of a matrix is by eigendecomposition and selecting a subset of first m eigenmodes. In case of covariance matrix K_{nn} the approximation can be formally written as in Equation 3.27.

$$\tilde{K}_{nn} = U_{nm}\Lambda_{mm}U_{mn} \tag{3.27}$$

This method is in fact equivalent to PCA (Equation 3.23). While it can be shown that this approach produces the best possible rank-m approximation [49], the method relies on eigendecomposition

of the original K_{nn} matrix, which scales as $O(n^3)$. This means that merely assembling the approximate covariance matrix is as costly as solving the full GP system. With no speed-up, this approach is therefore not used in practice.

II. Nystrom approximation

The alternative Nystrom approximation is widely used because the approximate matrix is constructed by the form shown in Equation 3.28, which can be inverted in only $O(m^2 n)$ steps [76], providing significant speed-up over the original $O(n^3)$.

$$\tilde{K}_{nn} = K_{nm} K_{mm}^{-1} K_{mn} \tag{3.28}$$

Formally, approximating the matrix with Nystrom scheme results in approximating the first m eigenfunctions of the full covariance matrix K_{nn}. The inner matrix K_{mm} can be constructed in various ways, which is paramount to the quality of the approximation. The naive approach is to select a random subset of m columns and rows of the nxn matrix , yet this method often results in sub-optimal approximations. On the other hand, the Nystrom concept is used also by sparse Gaussian processes – the most advanced class of GP approximations. In this case the inner matrix K_{mm} is built around the artificially introduced set of inducing points. The position of those points can be optimized, which in general results in better approximations.

Remark 3 *Although Nystrom approximation provides a significant speed-up, i.e. reducing the cost from $O(n^3)$ to $O(m^2 n)$, it relies on approximate eigenfunctions. Compared with the low-rank approximation built with PCA, which relies on exact eigenfunctions, the second approach always results in better accuracy.*

QGP: enabling PCA

By inducing qPCA, QPE allows to realize the first of the discussed methods - the PCA approximation for GP - within the QPG algorithm. While classically approximating a GP system via PCA results in additional computational expenses, in QGP the PCA is intrinsically embedded in the algorithm and can be triggered during the diagonalization of the system by exploiting the QPE mechanism, without adding additional cost.

In fact, inducing qPCA approximation allows to reduce the cost of the algorithm. The approximation is triggered by raising the eigenvalue resolution threshold $\lambda_{min} = \frac{\tilde{\lambda}_{max}}{2^k - 1}$, which can be done by lowering the value of k. As shown in the complexity analysis in Section 3.4, the algorithm scaling depends linearly on parameter k, which suggests that reducing the eigenvalue resolution (thus lowering k) can be leveraged to obtain additional speed-up, linear with k. Considering the total cost of algorithm, shown in Equation 3.24, the gain is even higher - as lowering k reduces the overhead terms (neglected in complexity analysis).

3.5.2. Comparison of the cut-off strategy

Compared to the classical PCA and Nystrom methods, qPCA relies on a fundamentally different strategy for cutting-off the eigenspectrum. While Nystrom method allows for direct control over the number of eigenmodes contributing to the approximation by choice of m, in QGP, the contributing eigenmodes are selected based on threshold value $\lambda_{min} = \frac{\tilde{\lambda}_{max}}{2^k - 1}$ controlled by parameter k.

Such an approach makes it impossible to determine what portion of the eigenspectrum contributes to the approximation. In general, the number of contributing eigenmodes is expected to be directly

related to the quality of the approximation, therefore, the lack of direct control over this parameter (i.e. the number of eigenmodes) limits the control over approximation quality.

Remark 4 *The shape of the eigenspectrum, or the rate of decay of the consecutive eigenvalues of the kernel matrix (K) is related to the smoothness of the kernel function [75]. For example, for more rough functions, the higher order eigenmodes are more significant, and the eigenspectrum decays at lower rates.*

To clarify the discussion, the differences between the two strategies are discussed by analyzing a numerical example. For this purpose two sample eigenspectra were prepared and approximated with naive Nystrom method and with QGP, as shown in Figure 3.11. In the first example, in Figure 3.11a the threshold value of QGP was set in such way that the same number of eigenmodes contribute to the approximation as in the Nystrom example.

Then, keeping the same settings, both methods were applied to a slightly different system, shown in Figure 3.11b. In this case the decay of the subsequent eigenvalues is milder than in the first example (Figure 3.11a). As a consequence, six more eigenvalues are above the cut-off threshold in QGP, which means that the resulting QGP approximation includes those six additional eigenmodes. In this case QGP turns out to be more beneficial as without increasing the computational cost (i.e. keeping the same k) it resolves more eigenmodes than the Nystrom method, which results in better quality.

For some systems the situation might be the other way around, e.g. for a system with a particularly steep eigenspectrum, the QGP would require significantly improved eigenvalue resolution (by increasing parameter k), in which case selecting eigenmodes by number might turn out more beneficial. The main point of the example above is to illustrate that both methods exhibit fundamentally different behavior, and their relative performance depends on the solved system and specifically, on the rate of decay of the eigenspectrum.

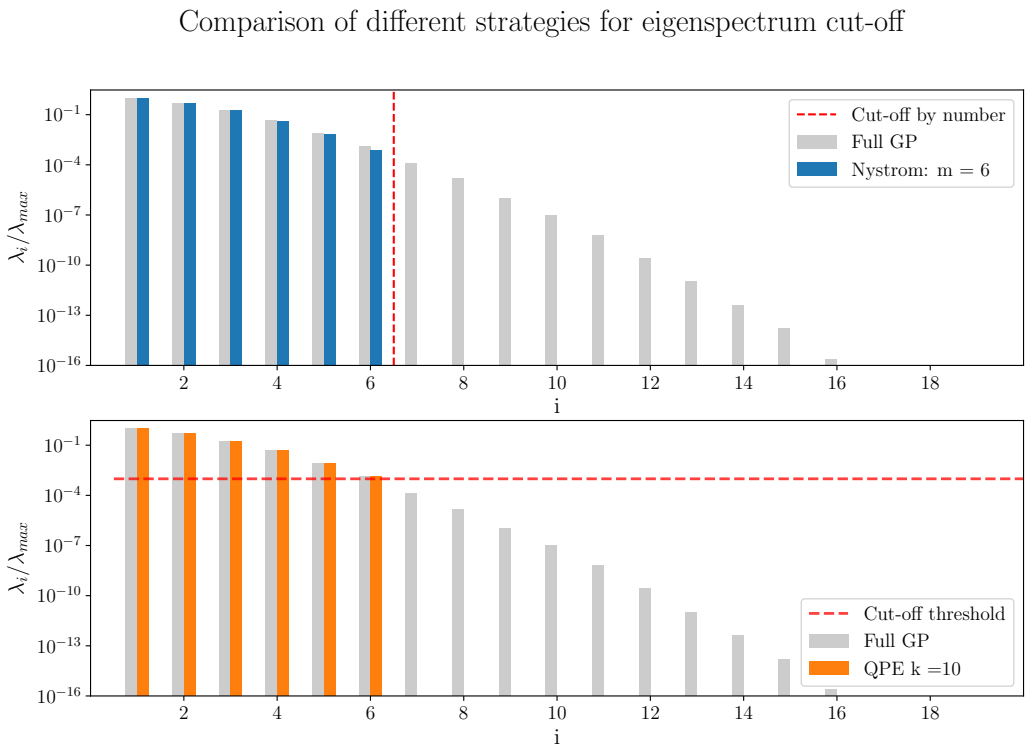
3.5.3. Accuracy: formal error bounds & numerical tests

The accuracy of the two algorithms can be compared more formally by means of their error bounds shown in Table 3.4 which were derived analytically in Appendix B). Note that the bounds of Nystrom method presented in the table are dependent on parameter k, which does not naturally occur in that scheme. Including k was imposed artificially in order to make the bounds for the two methods directly comparable. This was done by assuming that $\epsilon_{PCA} = \frac{\lambda_{max}}{2^k-1}$, where ϵ_{PCA} is originally used in the Nystrom bounds; this stands as the error of the best possible low-rank approximation (i.e. PCA).

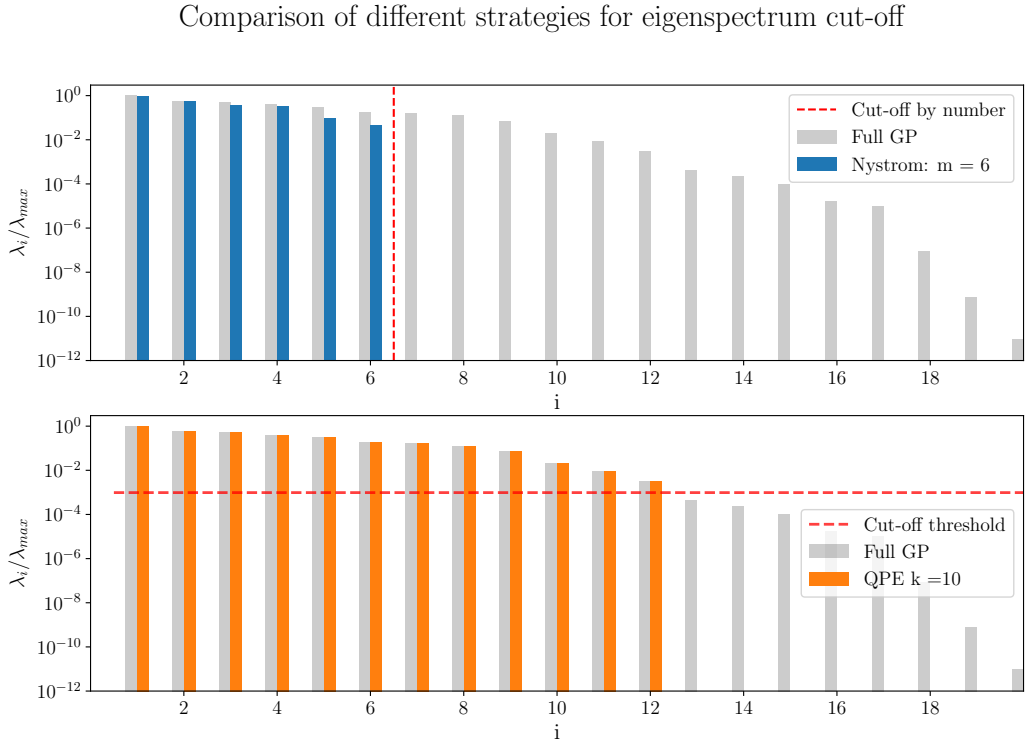
Table 3.4: Summary of error bounds for the two different low-rank approximation methods

Approximation method	Upper bound	Lower bound
Nystrom	$\lambda_{max}\left(\frac{1}{2^k-1}+2\sqrt{2}n\sqrt{1+\sqrt{log\left(\frac{2}{\delta}\right)\left(1-\frac{m}{n}\right)}}\right)$	$\frac{\lambda_{max}}{2^k-1}\left(\sqrt{1+\frac{n+m}{m}}\right)$
QPE	$\lambda_{max}\left(\frac{1}{2^k-1}+\frac{\sqrt{m}}{2^{k-log(2+\frac{1}{2\delta})}-2}\right)$	$\frac{\lambda_{max}}{2^k-1}\approx\epsilon_{PCA}$

By comparing the lower bounds of the two methods in Table 3.4, it can be concluded that the low rank approximation induced with qPCA can always be more accurate compared to the Nystrom scheme. It would be interesting to verify how does the higher bound of QPE relate to the lower bound of Nystrom, as this would address whether the qPCA approximation is always better or not. As the



(a) Example 1: eigenspectrum with relatively steep decay of subsequent eigenvalues.



(b) Example 2: eigenspectrum with relatively milder decay of subsequent eigenvalues.

Figure 3.11: Comparison of eigenspectrum approximation strategies: blue bars - an approximation constructed with Nystrom method, that relies on eigenmode cut-off by number; orange bars: approximation composed with QGP where the eigenmodes are selected with cut-off threshold on eigenvalues.

two expressions depend on different terms (which cannot be rigorously related to each other), this problem cannot be tackled analytically. Therefore, to analyze the relative performance of the two methods and verify the error bounds, the two approximation methods were compared via numerical tests.

Testing Approach

The two methods were employed to approximate a number of test matrices with different settings (m and k for Nystrom and qPCA respectively). For each sample an error was evaluated numerically, using the Frobenius norm of the difference between the original and approximate matrices (A and $\widetilde{A}$ respectively), as specified in Equation 3.29.

$$\epsilon = \left\| A - \widetilde{A} \right\|_F \tag{3.29}$$

The testing set consisted of 35 test matrices, each of size 1000×1000. The matrices were generated by RBF kernel function with seven different Gaussian noise variance settings (2, 5, 10, 20, 50, 100, 200), applied to five different datasets (the datasets can be found in Section B.2). The remaining hyper-parameters of the kernel function (length-scale and variance) were obtained by optimizing with the $optimize$ method from the GPy package[25]. The goal of such approach was to generate a test-set with variety of eigenspectra (which have crucial impact on the performance of the approximation schemes).

Nevertheless, to verify the approximation methods under a diverse set of learning problems, this testing ensemble was additionally complemented by tests on standard benchmarking functions. Those additional results serve as verification to the results from the primary test-set (35 matrices). For more details see Section B.3.

For each test matrix, two types of approximations were generated: qPCA and Nystrom. The Nystrom approximations were obtained for numbers m ranging from 1 to 1000, by deleting randomly selected rows and columns of the full test matrix to obtain K_{mm}. Similarly, the qPCA approximations were generated for k numbers ranging from 1 to 40. To limit computational cost and eliminate errors from other sources, the qPCA approximation was emulated classically (i.e. without using simulator) by diagonalizing the matrix and approximating the eigenvalues to a finite precision defined by k. This also allowed to count the number of eigenmodes (m) resolved for each k which facilitated direct comparison with the Nystrom method.

Experimental results

Out of the 35 test-cases, the two most extreme ones are presented here, in Figure 3.12 and Figure 3.13. The figure corresponds to a well conditioned matrix with $\kappa \approx 10^6$ and mild decay of the eigenvalues, as shown in the upper right plot. The overwhelming majority of the samples, however, resemble the results shown in Figure 3.13. In this case the eigenspectrum follows a rapid decay, until the eigenvalues reach approximately $\frac{\lambda_{max}}{10^{18}}$, where the eigenspectrum plateaus.

This effect corresponds to the unexpected growth of error in Nystrom approximation, as shown in the upper left plot in Figure 3.12. First, as m increases the approximation is more accurate and the error decreases as expected. At $m \approx 100$, the error starts to increase rapidly. This is caused by the influence of the eigenmodes corresponding to the ill-conditioned part of the matrix (marked in green). With increasing m more of those small eigenvalues are included in the inner covariance matrix K_{mm}, which becomes more difficult to invert, as required to construct the Nystrom approximation $\widetilde{K} \approx K_{nm} K_{mm}^{-1} K_{mn}$. At this point the round-off errors of the floating point arithmetic start to play

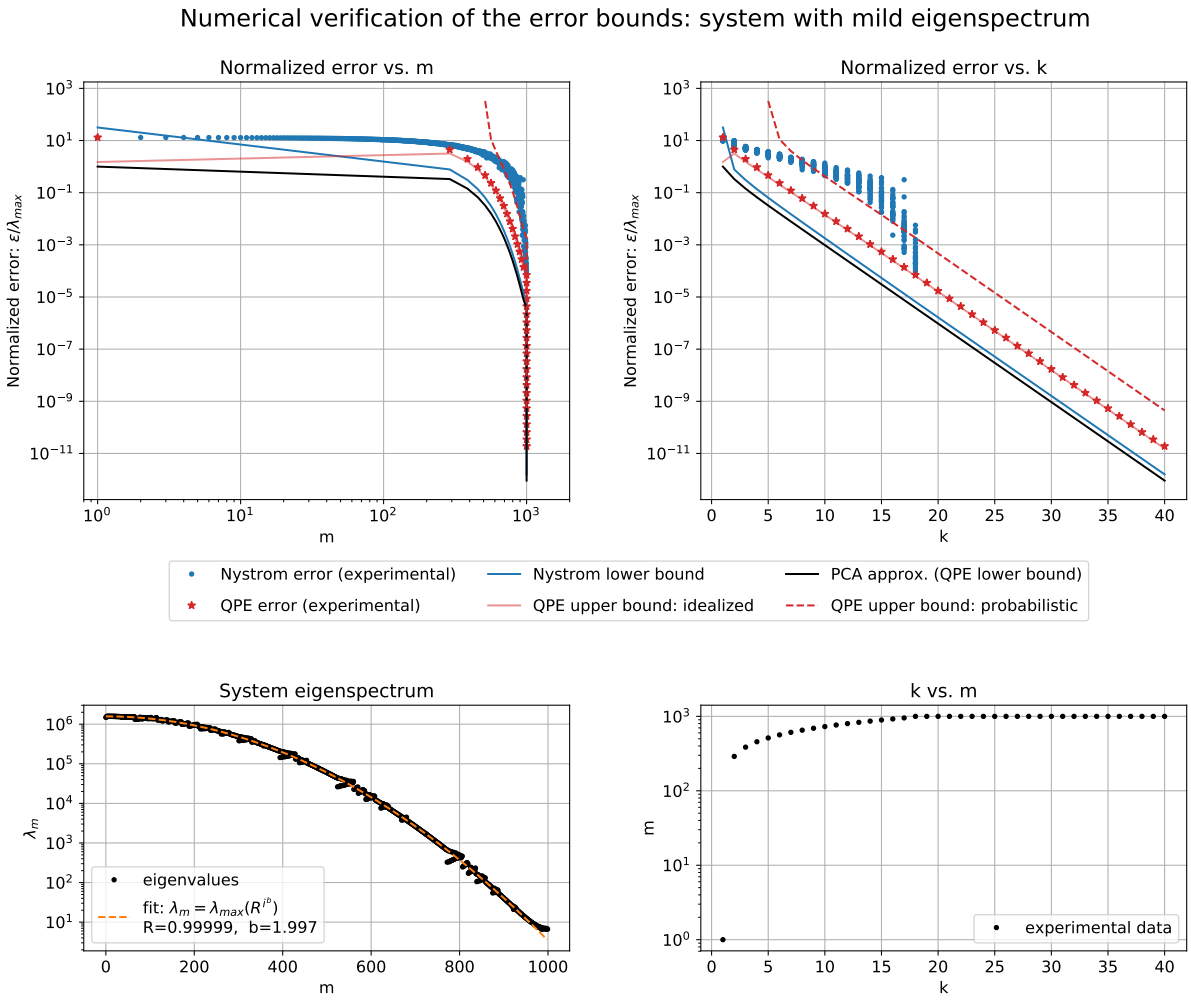


Figure 3.12: Comparison of experimentally obtained error against the theoretical bounds for two different low-rank approximation methods: Nystrom and QPE. Here, parameter m corresponds to the number of eigenmodes contributing to an approximation and k is the eigenvalue resolution parameter.

a role and result in large errors of the inversion K_{mm}^{-1} and consequently the overall Nystrom approximation as exposed in the upper left plot in Figure 3.12. This hypothesis was verified numerically by testing the deviation of $K_{mm}^{-1}K_{mm}$ from the identity matrix. It was observed that indeed for ill-conditioned matrices (exhibiting plateau in eigenspectrum) the deviation was considerable.

The unexpected behavior of error growth with m was also sought for in the experimental data reported in literature (e.g. [73]). It was concluded, however, that literature focuses only on cases where $m \ll n$, as only in such conditions the Nystrom approach yields significant speed-up. The growth of error with increasing m has not been reported.

Interestingly, the error of QPE did not exhibit such behavior, showing consistent convergence with increasing k regardless the conditioning of the matrix and its eigenspectrum. This is the result of the approximation scheme of QPE, which by capping the minimum eigenvalues (via parameter k) filters out the smallest eigenvalues corresponding to the ill-conditioned part of the matrix, which eliminates their severe influence on the quality of the approximation.

The experimentally obtained errors of QPE follow well within the theoretically derived bounds. The more rigorous probabilistic QPE bound, here plotted for $P = 1 - \delta = 0.99$, bounds the errors in all

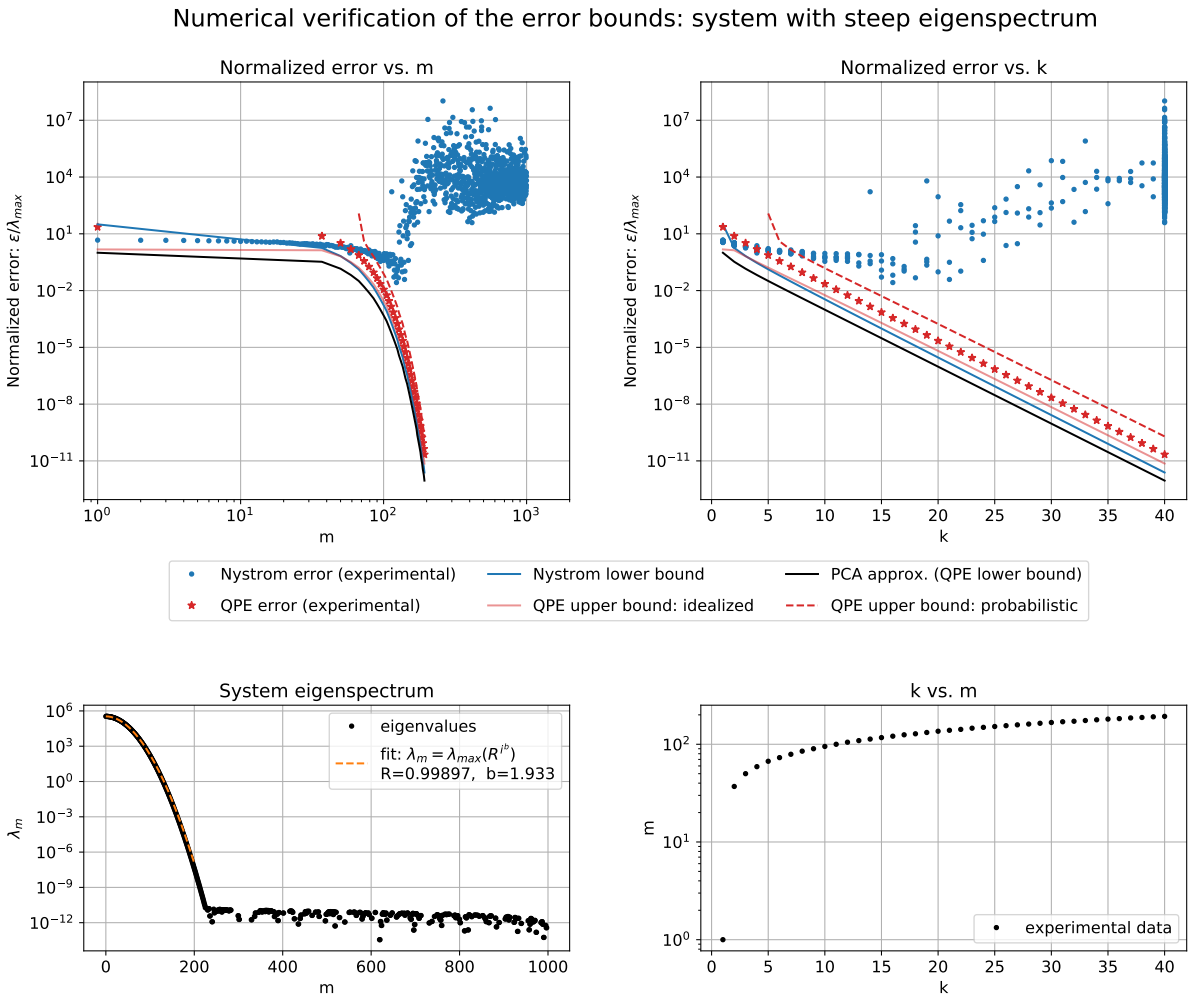


Figure 3.13: Comparison of experimentally obtained error against the theoretical bounds for two different low-rank approximation methods: Nystrom and QPE. Here, parameter m corresponds to the number of eigenmodes contributing to an approximation and k is the eigenvalue resolution parameter.

35 tested cases. The idealized QPE bound is generally less robust and tends to underestimate the errors. In the well-conditioned cases, however, it becomes a good estimator of the error as shown in Figure 3.12. The lower bound of QPE is equivalent to the PCA error.

Although the lower bound on Nystrom approximation falls below the upper bounds of QPE, theoretically allowing the Nystrom method to outperform QPE, experimental data proves otherwise. The results show that values of k as low as 5 are already sufficient for the QPE to outperform the classical method. The reason for that can be explained with the inferred $k-m$ relation shown in the bottom right plots in Figure 3.12 and Figure 3.13. In both cases, a small number k allows for resolving large number of eigenmodes m.

3.5.4. Implications on computational cost

The numerical results indicate that qPCA outperforms the classical Nystrom method in terms of accuracy of approximation. A naturally following question is: what is the cost of this advantage? Addressing this point requires comparing the computational cost required to reach a certain level of accuracy. The exact computational cost of the algorithms, however, is strongly dependent on im-

plementation, and thus subjected to large uncertainty. For this reason the algorithms are compared based on relative cost increase (in respect to some base value) rather than the exact cost. This relative increase are derived from the theoretical complexity of the two algorithms, where Nystrom method scales quadratically with m (i.e. m^2), while qPCA scales linearly with k.

The relative cost increase was plotted against the experimentally obtained accuracy of the two methods in Figure 3.14.

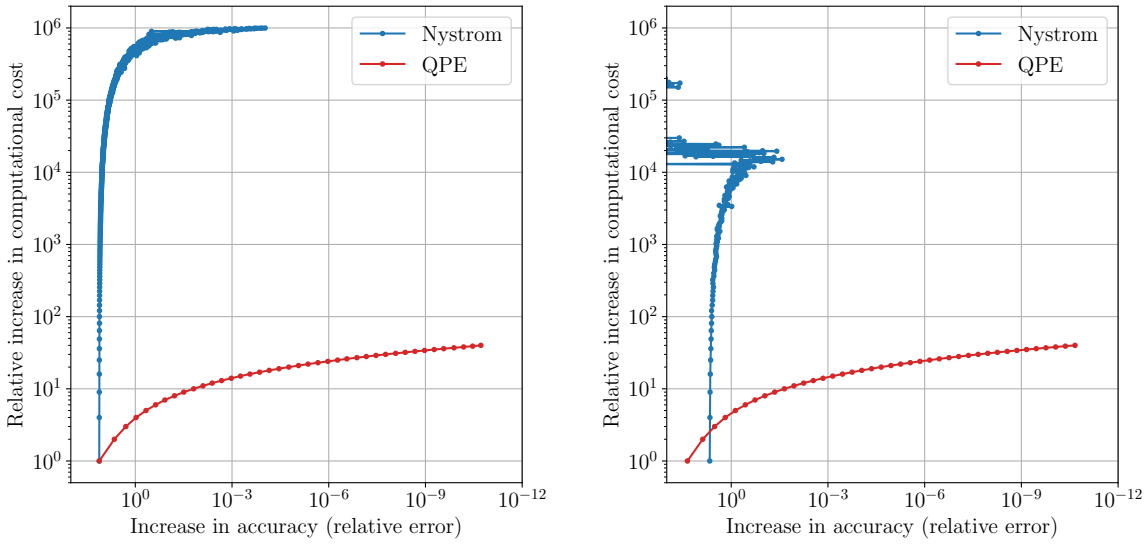


Figure 3.14: Increase in computational cost against the accuracy compared between Nystrom and QPE approximation methods. The error was obtained from two example numerical tests (shown in Figure 3.12 and Figure 3.13)

The results again indicate the advantage of quantum algorithm. As shown in both plots, qPCA exhibits much more favorable scaling than the classical alternative. Its exponential improvement of accuracy with k (as observed in Figure 3.13 and Figure 3.12) combined with linear dependency of computational cost on k, results in logarithmic scaling of the relative computational cost when compared to the improvement in accuracy.

On the contrary, scaling of the Nystrom method is much less favorable. In particular, for small values of m, the increase in computational cost is very large while the effect on precision is negligible. In this regime, lowering error by just 1 order of magnitude requires 100,000-fold increase of the computational expense. Although for larger values of m this dependency becomes much more favorable, actually outperforming qPCA, this regime is not interesting from the practical point of view, as large m compromises the Nystrom speed-up.

Therefore, the numerical tests indicate that the quantum alternative for inducing low-rank approximation to GPs outperform the classical counterpart, namely the Nystrom method, both in terms of accuracy as well as computational cost.

3.5.5. Numerical example of sparse QGP

This section provides a numerical example that illustrates the effect of inducing sparsity in QGP and compares its behavior with the equivalent classical approximation method i.e. the rank reduction with eigendecomposition. The example problem was set-up using eight semi-randomly selected data-points, following the approach discussed in subsection 5.2.2. Those points together with kernel

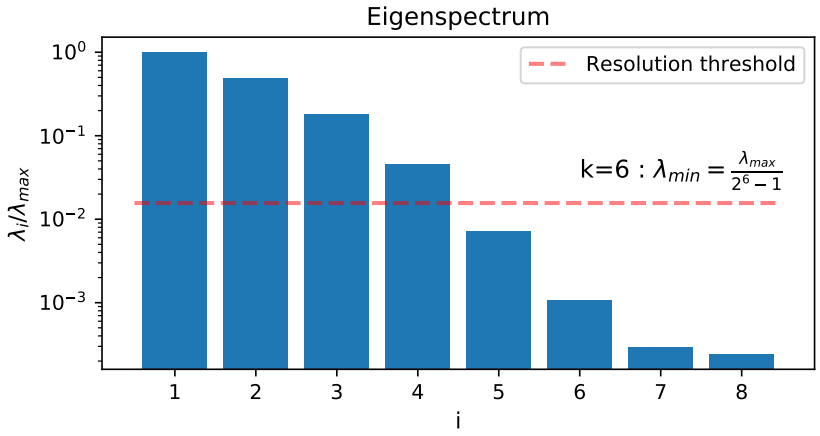


Figure 3.15: Eigenspectrum of the system used for the example compared with the eigenvalue cut-off threshold obtained with the QGP settings used in the trial (specifically: $k = 6$).

parameters listed in Table 3.5 (which were obtained in advance) allowed to define the 8×8 input matrix.

Table 3.5: List of the example GP model parameters

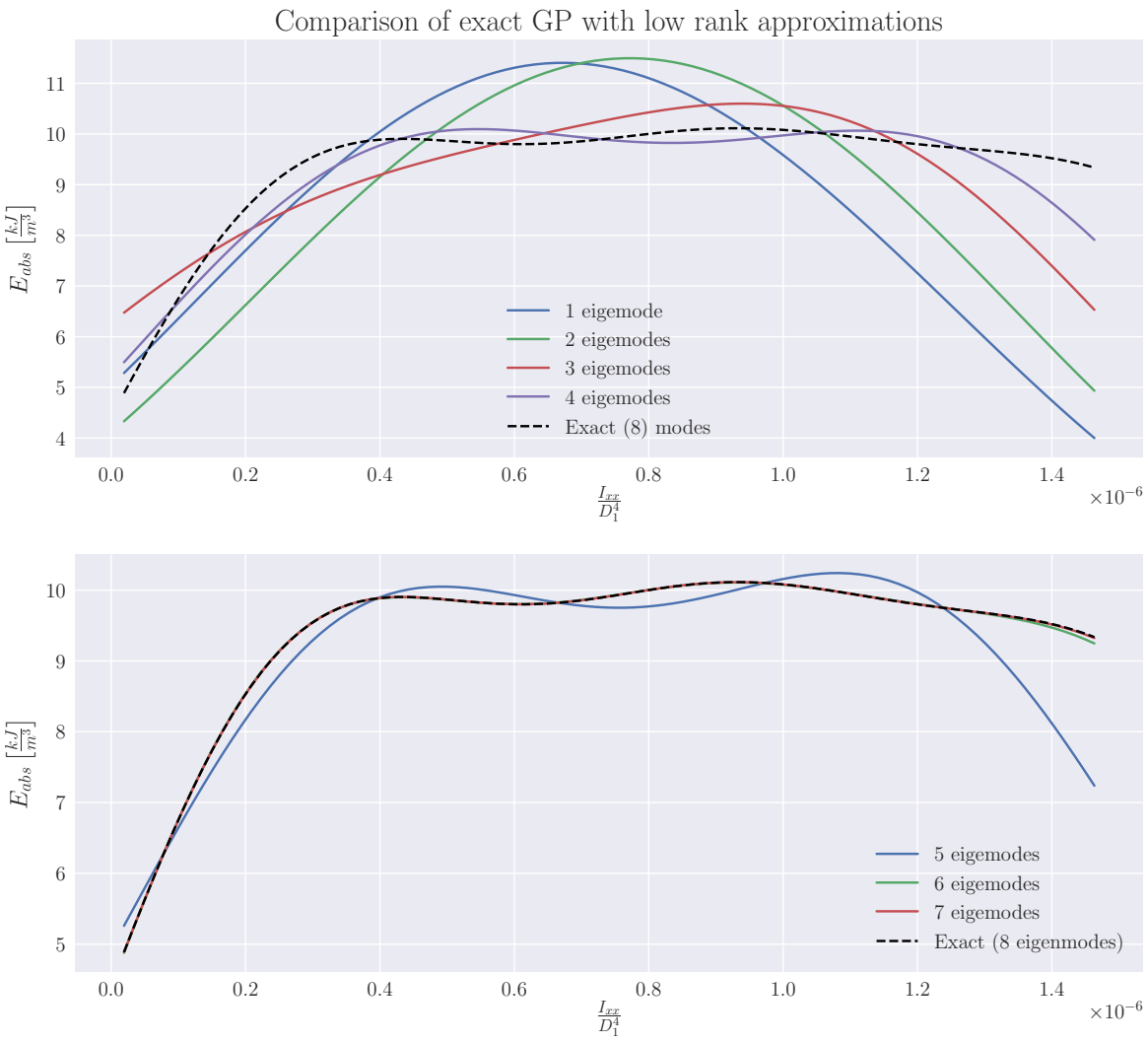
Parameter		Value
σ	RBF kernel variance	86.30
l	RBF kernel length-scale	3.85×10^{-7}
σ_n	Input noise variance	0.09

Such a system was then subjected to the classical method of rank reduction via eigendecomposition to produce an approximation in general form: $\tilde{K}_{nn} = U_{nm}\Lambda_{mm}U_{mn}$. In this way a number of approximations were composed using different number of eigenmodes m. The resulting low-rank solutions are plotted in Figure 3.16a.

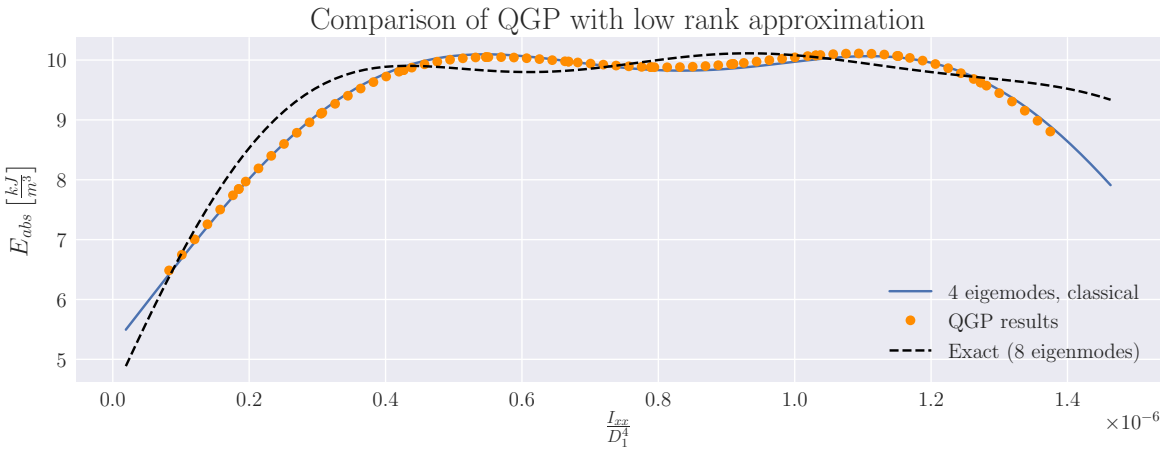
Comparing those different solutions shows how adding subsequent eigenmodes improves the approximation quality. The functions composed of more eigenmodes are better following the trend of the full GP solution, which was to be expected.

Next, the sample 8×8 system was solved with the implemented QGP algorithm. For this trial, the QGP was run with $k = 6$ and $r = 25$. Knowing the parameter k, the eigenvalue cut-off threshold could be calculated. This value was then compared with the eigenspectrum of the example system (obtained classically), as shown in Figure 3.15. The 4 smallest eigenvalues fall below the threshold, which suggests that the corresponding eigenmodes will be cut off by QPE. Therefore according to the predictions, QGP run with such settings shall produce a low-rank approximation composed of the four lowest eigenmodes.

The QGP results are plotted in Figure 3.16b. Compared against the classical approximation, the results from the quantum algorithm follow very closely the 4-eigenmode approximation, which is in complete agreement with the analytical predictions. Consequently, this example not only illustrates the effect of low-rank approximations, but also provides an empirical evidence on the QPE-induced sparsity mechanism occurring in the QGP algorithm.



(a) Comparison of different low-rank GP approximations. The upper subfigure shows models built out of 1-4 eigenmodes compared with the exact model. The lower subfigure includes models built out of 5-8 eigenmodes.



(b) Comparison of 8×8 QGP example with a classical low-rank approximation that includes 4 out of 8 eigenmodes

Figure 3.16: Numerical example of QGP with induced sparsity.

4

Discussion about QGP algorithm

The aim of the first part of this research was to verify the feasibility of the QGP algorithm. This objective was realized by implementing the algorithm and conducting a numerical analysis. The results allowed to understand and control the algorithm performance, but also identify its deficiencies.

The first part of this chapter addresses the feasibility of the algorithm. The discussion focuses on the technical improvements required in the QGP algorithm in order to enhance the data-driven framework. In particular, it focuses on improving the Hamiltonian simulation step, which was identified as the most critical step of the algorithm. This topic is addressed in Section 4.1. Next, Section 4.2 analyzes the performance of the algorithm when using a real quantum computer. The discussion is supported with quantitative analysis which confronts the performance provided by currently available quantum hardware, with the performance required by the algorithm. The discussion concludes with Section 4.3, which embeds the results in a broader context, addressing how the findings fit within the current developments of quantum computing and quantum algorithms.

4.1. Hamiltonian simulation

Hamiltonian simulation step of HHL is the most critical element in the QGP, as it is the most computationally expensive routine of the algorithm (consequently also the dominant term in the algorithm complexity) and therefore determining the speed-up. In this research the QGP implementation employed a sub-optimal Hamiltonian simulation, which relies on decomposing the matrix onto a sum of tensor products of all possible permutations of Pauli operators, which does not allow to reach the advantage over the classical algorithms.

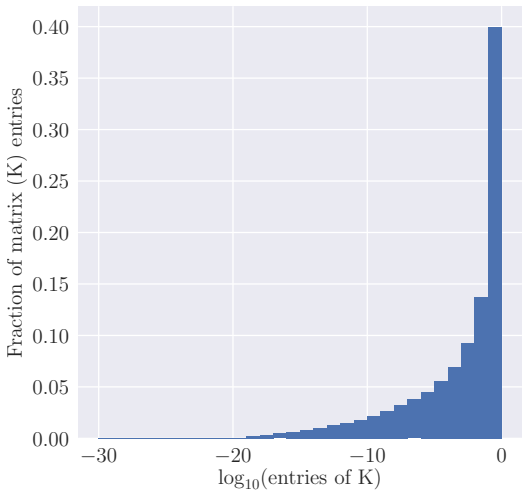
Remark 5 *Considering that the focus of this research is on application of the algorithm, rather than its improvement, it was decided to use the sub-optimal Hamiltonian evolution, despite its deficiency, as this routine was already implemented in Qiskit.*

In order for QGP to reach the promised speed-up, the Hamiltoniann simulation needs to be implemented with a subroutine originally proposed for HHL, which relies on graph coloring method [6, 29]. While this alternative allows to reach the exponential speed-up, it is restricted to rigorously sparse systems, i.e. systems with at most $s \sim O(log(n))$ non-zero entries per row. The importance of the sparsity becomes apparent by examining the complexity of the HHL algorithm: $O(log(n)\kappa^2 s^2/\epsilon^3)$. The scaling with s^2 means that the speed-up becomes achievable only when $s \ll n$. Otherwise, in case $s \sim n$, the speed-up is compromised and the overall algorithm scales as $O(log(n)\kappa^2 n^2/\epsilon^3)$.

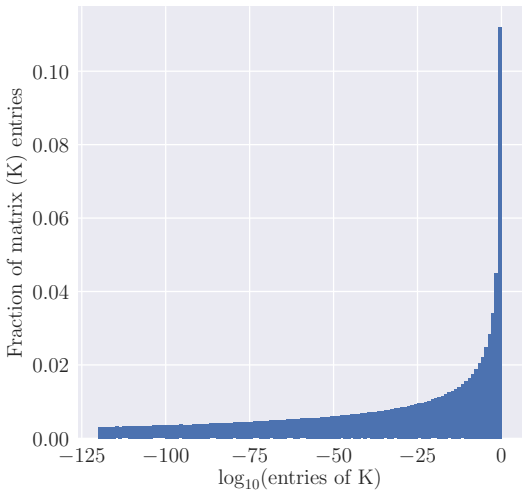
Sparsity of the kernel matrices in GP systems

In principle, the covariance matrices used in GPs are usually dense and it can be expected that all entries are non-zero. This depends of course on the kernel function, which implies that there might be some exceptions, however, the statement holds in most of the commonly used kernels, such as e.g. RBF. On the other hand, it is common that the majority of those non-zero terms are small in magnitude, to the extent that they become negligible. This is expected especially for large systems. In such case, by neglecting the small terms it would be possible to achieve sparsity - required by the algorithm.

In order to verify to what extent sparsity could be achieved with such an approach, a number of matrices were analyzed by generating histograms on number of terms within given range of magnitude. The matrices were generated with RBF kernel function on several different data-sets. An example of such analysis is shown in Figure 4.1, which compares the distribution of entry values for two matrices defined for two different GP problems. The idea behind this analysis is to determine the order of magnitude of sparsity factor s that occurs in GP systems. This factor is estimated by generating a histogram of matrix entries, from which it can be determined how many entries of the matrix are small enough to be assumed as negligible (here a threshold of 1×10^{-6} was assumed). The proportion of entries that remain non-zero after canceling the terms below the chosen threshold provides an indication of the magnitude of s factor.



(a) In this example 45% of the entries are below 1e-6. The matrix was generated for 5000 noisy data-points, generated with $f(x) = 0.1xsin(x) + \epsilon$.



(b) In this example 78% of the entries are below 1e-6. The matrix was generated for 5000 noisy data-points, drawn from a multivariate normal distribution given by mean=0 and covariance specified by the exponential kernel.

Figure 4.1: Distribution of entry values of a sample kernel matrix. The values were normalized to the variance (the diagonal entry).

The two examples shown in Figure 4.1 indicate a sparsity factor of $s = 0.55$ and $s = 0.22$, respectively. The tests did not show any quantitative dependence of sparsity on the size of the system. Only qualitative insight was provided, with the conclusion that sparsity is very dependent on the considered data-set and it may vary from one system to another. For instance, when the data was generated with a simple linear function, all entries are close to 1, while for other functions the spread in values of the matrix entries up to 10 orders of magnitude was observed. Nevertheless, the scope of the tests was

too narrow to support more general conclusions.

An alternative for satisfying the sparsity condition could be by designing a kernel function, that intrinsically induces sparsity in the covariance matrix. In this case, however, obtaining predictive performance comparable to the commonly used kernel functions could be challenging.

Algorithmic solution

The problem of dense matrices could be also approached with an algorithmic improvement. Recently, Wosning et al.[77] proposed a quantum algorithm based on HHL concept, with modifications to solve systems of equations with non-sparse matrices. The algorithm relies on an assumption that the rows and columns of the matrix could be efficiently mapped into quantum state (e.g. via qRAM), which could be then processed with a quantum singular value estimation subroutine – a step similar to diagonalization with QPE.

4.2. QGP on quantum hardware

As discussed briefly in subsection 3.3.3, execution of QGP algorithm was attempted on a quantum computer provided by IBM to the public. Despite severe optimization of the quantum circuit, no meaningful results were obtained. This section explains the outcomes by quantitative comparison between the performance of the quantum hardware, and the requirements of the QGP algorithm, and in particular the mismatch between the two.

4.2.1. Performance of quantum computers

The performance of a quantum computer depends on several factors, such as: the number of qubits, the gate error rates, as well as the chip architecture (specifically the physical connectivity between the qubits). The number of qubits translates directly onto the width of quantum circuit, while the gate error rate determines the practical length of the circuit in terms of number of gates. Finally, the physical connectivity of qubits is crucial for implementing 2-qubit gates, which enable quantum entanglement – a key resource in quantum computing.

Depending on the intended use and underlying technology, different physical realizations of quantum hardware provide different balance between those performance factors. Therefore, to facilitate an easy comparison of performance of different quantum hardware, researchers from IBM introduced a single number metric called quantum volume [17]. This measure accounts for all the important performance factors, and facilitates comparing devices with different architectures and specifications.

By definition, the quantum volume benchmarking is done by testing the quantum device on an ensemble of random square quantum circuits (where the depth is equal to the width). The quantum volume is then defined as the largest depth (or width) for which the probability of passing the test is above a certain threshold [17]. While the benchmark is defined on square circuits, quantum devices usually can also perform well for rectangular circuits, which are more representative for the real-life applications as discussed by Bume-Kohout et al. [13]. This situation is illustrated in Figure 4.2, which presents a hypothetical performance profile of a quantum device.

4.2.2. Quantum volume estimation for QGP circuits

Based on this discussion, it can be assumed that the quantum volume of a device necessary to execute a given circuit can be roughly estimated as a square root of the product of circuit width and

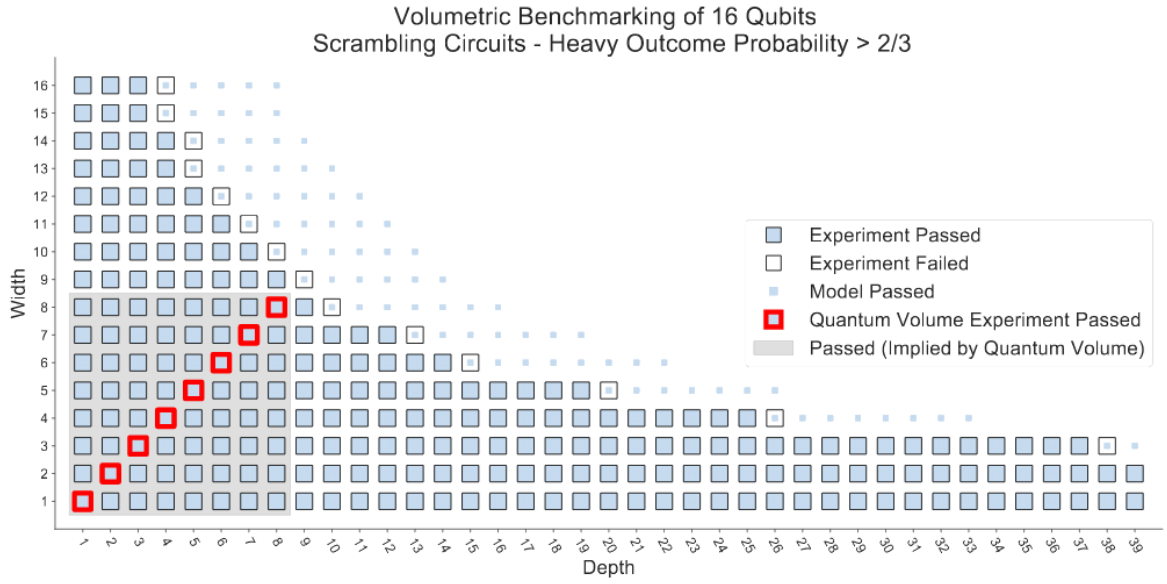


Figure 4.2: Example profile of quantum device performance. (Illustrative purposes only)[13]. Here, $V_Q = 8$, which corresponds to the largest square circuit for which the device passed the tests(marked with red).

depth (w, d):

$$V_Q \approx \sqrt{w \cdot d} \tag{4.1}$$

In such way, it is possible to estimate the required quantum volume, given the circuit width and depth, which for QGP can be determined respectively from the size of registers and the circuit scaling model, for any system size n. With this assumption, the minimum quantum volume needed to execute the QGP algorithm for different system sizes (n=2, 4, and 8) was estimated, as presented in Table 4.1. For this example, circuit parameters were set such that the expected errors are less than 5% (determined from the contour plots).

Such an estimation can be also done for the QGP circuit that was optimized for execution on a quantum computer (consisting of 200 gates and 5 qubits), as discussed in subsection 3.3.3, for which the required quantum volume would be roughly 32.

Those values can be compared with the performance reached by contemporary devices, marked in Figure 4.3. The best currently available device reaches quantum volume of 16 which is half of what is needed for the optimized demonstration circuit, and several orders of magnitude below the V_Q required by the small general QGP systems, listed in Table 4.1.

The performance of quantum devices, however, kept growing exponentially over the past few years, with quantum volume doubling every year as shown in Figure 4.3. The trend resembles the Moore's law, which defined the advancements of integrated circuits since 1960s [33]. The number of datapoints for quantum computing, however, is too limited to provide grounds for assessing whether a similar relationship might hold.

As both the progress of quantum computing as well as the scaling of quantum volume with system size are subjected to large uncertainties, this analysis does not allow to predict when quantum advantage might be reached for Gaussian Processes. However, it provides a good indication of the order of magnitude – if the progress in quantum hardware continues as projected, it would take around one decade until quantum devices deliver sufficient performance to run a modest-size demonstration

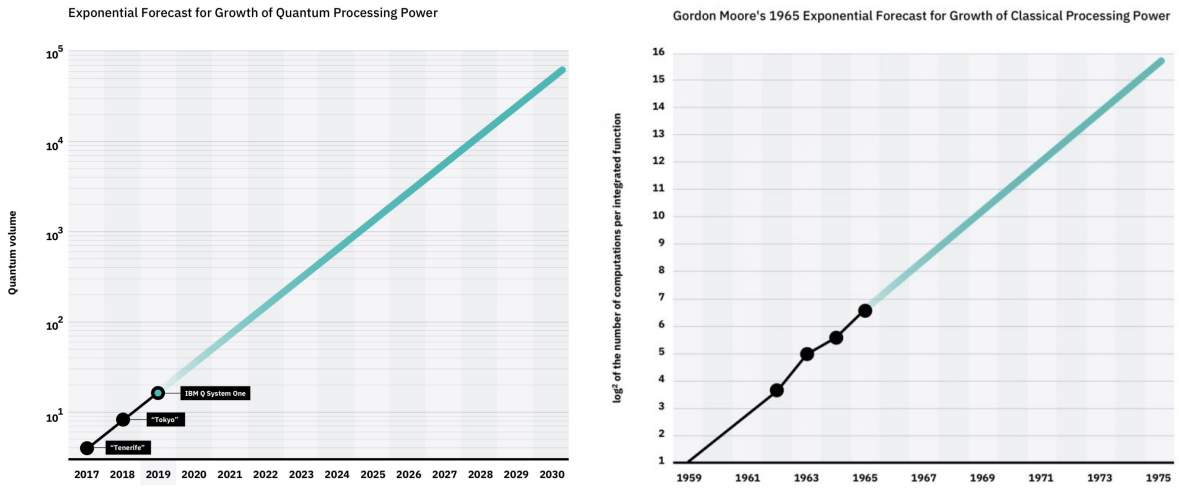


Figure 4.3: Comparison of the advancements in quantum computing measured by the quantum volume (left figure) with the advancements in integrated circuits in 1960s following the so-called Moors' law catching the exponential growth in performance[33]

Table 4.1: Estimation of quantum volume needed for QGP execution with given system size

System size (n)	Circuit width	Circuit depth	Required quantum volume V_Q
8	13	10^6	11400
4	10	10^5	3100
2	9	10^4	950

circuit listed in Table 4.1. It could be expected that it would take then a few more decades (one to two) to provide the performance necessary for solving larger systems ($n \sim O(10,000)$).

4.3. Novel approaches to quantum computing

In a broader perspective, this research provides insight onto more general aspects regarding the applicability of quantum computing. The method used in the research, specifically, studying the algorithm via numerical analyses of this implementation, is a first step towards addressing one of the key problems of contemporary quantum algorithms – computational cost. The results also provide new perspective on quantum machine learning, and application of quantum computing in materials science.

4.3.1. The computational cost problem of quantum algorithms

The results obtained by testing of the implemented quantum Gaussian Processes, and specifically the quantitative analysis of the circuit scaling, provide a first step toward solving one of the key problems of the contemporary quantum computing, namely the computational cost problem of quantum algorithms.

Currently, research on quantum algorithms is still practiced mostly at a theoretical level. This means that the algorithms are developed as abstract mathematical concepts. Considering how limited the current quantum hardware is, this is a reasonable approach. However, it has a major drawback: the

real cost of the algorithm remains unknown [12].

The primary tool used for analyzing the cost of algorithms is the complexity analysis, which in principle allows to determine scaling of the number of fundamental operations with the size of the input. This approach is used also for proving quantum advantage. In this case, the complexity bounds are determined by considering an asymptotic case, which in practice means that the complexity bound is defined by the term with the worst scaling, assuming that for sufficiently large input other terms become negligibly small. This practice led to another tendency in quantum computing community: while proposing new algorithms, certain routines such as e.g. arithmetic units or oracles (e.g. finding the eigenvalue reciprocal in HHL) are treated as black boxes, without getting into details of realizing such subroutines[65]. The argumentation behind this practice is that for 'sufficiently large' inputs, the cost of those skipped routines becomes negligible.

This becomes a problem, when one is interested in proving quantum advantage in a practical setting. In such case the objective is to find out what is the 'sufficiently large' system, or equivalently, the smallest quantum circuit (e.g. in terms of the true number of gates) that would allow to outperform the classical algorithms. As discussed by Biamonte et al.[12] this issue is one of the most urging questions in quantum computing, yet for now remains unanswered mainly due to the computational cost issue.

This research contributes toward solving this key problem for the case of Gaussian Processes. The cost model discussed in Figure 3.4.2 allowed for quantitative determination of the circuit scaling in terms of the basic gates implementable on quantum devices. Furthermore, the scaling was analyzed for the dependence on several approximation parameters i.e. r, k which can be related directly to the algorithm accuracy via error contour plots. As the current QGP implementation relies on the suboptimal Hamiltonian evolution routine, the cost model does not directly apply to the exponentially faster version of the algorithm, therefore it does not allow to determine what would be the cost of reaching quantum advantage (in quantum gates). However, it still gives a valuable insight into true cost of the overhead terms, i.e. the additional steps including eigendecomposition and inversion of eigenvalues, which are required to execute the algorithm.

4.3.2. Quantum machine learning

Most of the currently proposed quantum machine learning algorithms rely on replacing the expensive linear algebra operations with their quantum counterparts that provide computational speed-up. This is also the case in QGP, where the matrix inversion is carried out with exponentially faster HHL algorithm. The results of this research prove that in case of QGP, however, shifting to quantum paradigm brings additional benefits.

The QGP algorithm can take advantage from exploiting the approximations intrinsic to HHL, specifically the eigenspectrum approximation implemented with QPE, to induce sparsity. This results in a low rank approximation, in a similar effect to that in classical sparse Gaussian processes. Furthermore, the research shows that this strategy of QPE induced approximation enables a speed-up of QGP that is linear with parameter k.

4.3.3. Quantum computing in materials science

So far, quantum computing is promising for several fields of science [51, 57]. Materials science is often mentioned as one of those fields [51, 57]. It is believed that quantum computing, with formulation based on quantum mechanics, is well suited for facilitating calculations addressing problems of quantum mechanics such as e.g. electronic structure of atoms, from which in principle most of

the intrinsic properties of material can be derived[51, 53, 57]. In this way, new compounds could be found with desired properties.

In practice, however, material properties are the resultant of different phenomena happening at many different length-scales. In design of materials it is essential to address all those mechanisms, as some might dominate over others, determining the overall material performance. For instance, in metals the mechanical properties such as yield strength are dominated by microstructure and defects at the micro-scale, while the calculations at the atomic level overestimate the strength, sometimes even several orders of magnitude [46].

Therefore, enhancing the design with quantum computing only at the atomic level, as it was proposed so far, only tackles part of the complex design problem. Instead, the proposed data-driven framework enhanced with Quantum Gaussian Processes enables a comprehensive approach to designing materials, allowing to address different phenomena at different length-scales simultaneously. Consequently, this research shows a different path for application of quantum computing in materials science.

5

QGP application in design of metamaterials

The overall objective of this research is to explore the potential enhancement of data-driven design of materials with quantum computing, by replacing the machine learning step with an exponentially faster quantum algorithm for Gaussian Processes. This goal is realized in two steps: first, by implementing the QGP algorithm and verifying its feasibility, which was discussed in Chapter 3 and Chapter 4; and secondly by demonstrating the applicability of the implemented QGP algorithm in a proof-of-concept case of metamaterial design – the subject of this chapter.

The application is demonstrated on two design cases of optimization of the unit cell for the super-compressible metamaterial (discussed in subsection 2.1.3). The first case considers optimization of a single design variable (I_{xx}) for maximizing the cell's capacity for energy absorbtion. The second design case considers optimization of two design variables (I_{xx} and P), for two objectives - to simultaneously maximize the energy absorption and the critical buckling load.

The chapter is structured as follows: first Section 5.1 gives a brief introduction into the set-up of the design problem, and explains how the problem is approached with the data-driven framework. Next the analysis of the two design cases is presented in Section 5.2 and 5.3.

5.1. Approach

The design of the unit cell, shown in Figure 5.1 is parameterized with seven design variables, listed in Table 5.1. Note that the parameters are normalized with respect to the lower ring diameter D_1. The purpose of normalization is to lead to scale-independent predictions. The convention follows from the *F3DAS* code [10], which implements the data-driven approach for design and analysis of this particular unit cell.

The two considered design cases were set-up with the purpose of demonstrating application of the QGP algorithm and its integration within the data-driven design process. The design problems were therefore chosen primarily to provide a non-trivial case for demonstration purposes, rather than any other practical reason. The problems were set-up based on results from Glowacki's work [24], and in particular the design contour plots, which were used as a reference for setting the values and ranges of the design parameters, to produce the desired (non-trivial) E_{abs} and P_{crit} landscapes. Nevertheless, the complexity of the problems was severely limited by the costs of the classical simulation of QGP algorithm.

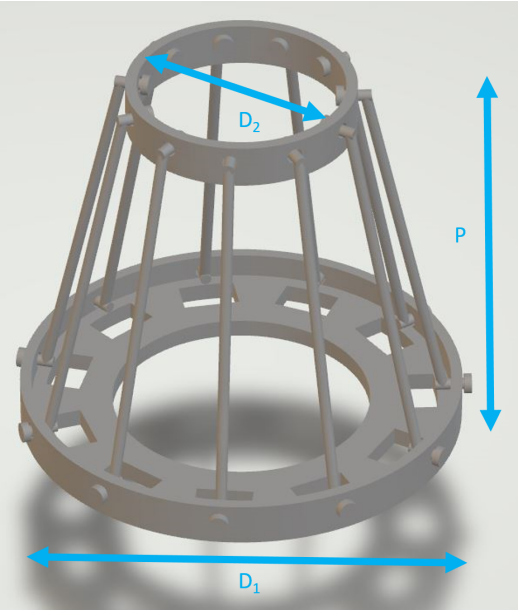


Figure 5.1: Unit cell of the super-compressible metamaterial

Design parameters	
$\frac{P}{D_1}$	Pitch
$\frac{D_1-D_2}{D_1}$	Slope
$\frac{A}{D_1^2}$	Area of the longeron cross-section
$\frac{G}{E}$	Shear modulus to Young modulus ratio
$\frac{I_{xx}}{D_1^4}$	Second moment of area of the longeron cross-section around x axis
$\frac{I_{yy}}{D_1^4}$	Second moment of area of the longeron cross-section around y axis
$\frac{J}{D_1^4}$	Polar moment of area of the longeron cross-section

Table 5.1: A list of design parameters defining the design of the unit cell, normalized by the diameter of the lower ring.

Both design problems were approached following the steps of the data-driven design framework mentioned in subsection 2.1.2, except for the machine learning which here was carried out with the QGP algorithm. In this way, not only the practical application of the QGP algorithm is demonstrated, but also its integration within the complete process of computational design of materials. The general procedure consists of the following steps:

1. **Design of experiments:** first the design space spanned by the considered design variables is sampled following Sobol sequence[1]. Secondly, the design imperfections are generated for each sample – in this case, the initial twist of the unit cell (upper ring with respect to the lower), which is sampled from a lognormal distributions with a mean of 1.2 deg and standard deviation of 4 deg (the values found by Glowacki [24]). The whole design of experiments procedure is carried out using the *F3DAS* code[10].

2. **Computational analyses:** the sample designs are then analyzed using finite elements models to predict their response. The numerical model is used for two types of analysis: 1) linear buckling analysis, which predicts the critical buckling load P_{crit} and 2) RIKS analysis, which allows to analyze the post-buckling deformation – essential for deriving the energy E_{abs} absorbed during coiling of the cell.

   In order to reduce the computational costs of the analysis, however, the numerical model assumes several simplifications:

   - The analysis considers only the first buckling mode.
   - The connection between the longerons and the rings are modeled as frictionless hinges.
   - The interaction between the longerons (e.g. contact) is neglected.

   The procedure is facilitated by *F3DAS* code [10], which provides an interface to Abaqus, where the numerical experiments are carried out.

[1] Sobol sequence is usually used for sensitivity analysis of the design parameters [24][70]. It is the default sampling scheme in *F3DAS* code [10].

3. **Classification:** the results of the post-buckling RIKS analyses determine whether given sample design promotes coiling or not. Those results are therefore used as an input to a classification model, which partitions the design space on two domains, according to the coilability of the designs. This step effectively filters out the data corresponding to the non-coilable designs, which are not of interest for the design of the metamaterial. The classification step was carried out using a standard model from *GPy* Python package[25], which implements classification based on Gaussian processes.

4. **Machine learning - GP regression:** the design points classified as coilable are used to build a GP regression model, which aims to predict the performance over the complete design space (i.e. interpolate in between the training points). For demonstration purposes this step is carried out with the QGP algorithm, however two additional classical GP models are created as well, to provide a reference for validation and verification of the QGP results. In total three models are created:

   (a) **High-fidelity classical GP model:** built using the whole available data-set (corresponding to the coilable samples), this model achieves highest accuracy and is intended to provide a reference for validation of the QGP model prediction.

   (b) **Reduced GP model:** this classical model is built using a subset of training points (here eight points were used in all examples). The purpose of this model is to provide a kernel function optimized on the chosen training points, which can be used to generate inputs for the QGP model (8×8 input matrix K, cross-covariance input vectors k_*). Furthermore, this model solved classically can be directly compared with the QGP results, playing an essential role in the verification process of the QGP results.

   (c) **QGP model:** realizes the main objective of this part of the work, i.e. demonstrates the application of the QGP algorithm within the data-driven framework. The model solves the system created with the classical reduced model, using inputs generated with the classically optimized kernel function. The algorithm is executed using simulator of quantum computer provided with Qiskit[1]. The computational cost of the simulation was the main factor limiting the size of the systems that could be tackled. For this reason, the design examples are limited to eight training points.

5.2. Single parameter optimization: Energy absorption vs I_{xx}

The first design case considers optimizing a single design variable, namely the moment of inertia of the longeron cross-section around x axis (I_{xx}) to maximize the metamaterial's capacity for energy absorption. The problem was selected based on Glowacki's results [24], which showed that the relationship between those two parameters exhibits considerable amounts of non-linearity (comparing to other parameters), making it an interesting case for illustration of the QGP application in 1D. The range of I_{xx} values, as well as the constant design parameters were set as show in Table 5.2.

The design process was approached with the data-driven framework. In the first step, design of experiments, 245 sample design points were generated, with various I_{xx} values, including also imperfections (i.e. the pre-twist). Those samples were then analyzed with geometric imperfections (causing uncertainty to the quantity of interest) to generate a data-base of responses. As the chosen values for the design parameters guarantee coilability of the design over the complete range of values of I_{xx}, the classification step could be skipped. Therefore, the database could be directly used for assembling a machine learning model

Table 5.2: Design parameters used in the 1D design case

Parameter	Value
$\frac{P}{D_1}$	0.66
$\frac{D_1-D_2}{D_1}$	0.0
$\frac{A}{D_1^2}$	1×10^{-3}
$\frac{G}{E}$	0.36
$\frac{I_{xx}}{D_1^4}$	$[1.1275 \times 10^{-11} - 1.3918 \times 10^{-6}]$
$\frac{I_{yy}}{D_1^4}$	7.5×10^{-7}
$\frac{J}{D_1^4}$	1.0×10^{-6}

Table 5.3: Kernel function hyper-parameters inferred for the classical high fidelity GP model.

Parameter	Value
RBF kernel variance	86.30
RBF kernel length-scale	3.85×10^{-7}
Input noise variance	0.09

5.2.1. Classical GP model

The 245 sample design points were then used to build a classical high-fidelity GP regression model, shown in Figure 5.2. The model used the most common covariance function – the RBF kernel. The hyper-parameters were obtained with a standard optimization routine from *GPy* Python package, which relies on maximizing the log of marginal likelihood. The values are listed in Table 5.3.

Overall, the model provides a good representation of the data - the predicted mean represents well the general trend, while the uncertainty bounds accurately account for variation in the metamaterial performance. The relationship, however, turned out to be more steady than anticipated from Glowacki's results – the E_{abs} increases steadily for $\frac{I_{xx}}{D_1^4}$ up to 0.3×10^{-6}, after which it reaches a plateau and remains approximately constant. The model presented in Figure 5.2, however, uses relatively large data-set, which results in smoothing out any potential variation (which could be slightly more prominent in Glowacki's results due to relatively fewer data-points).

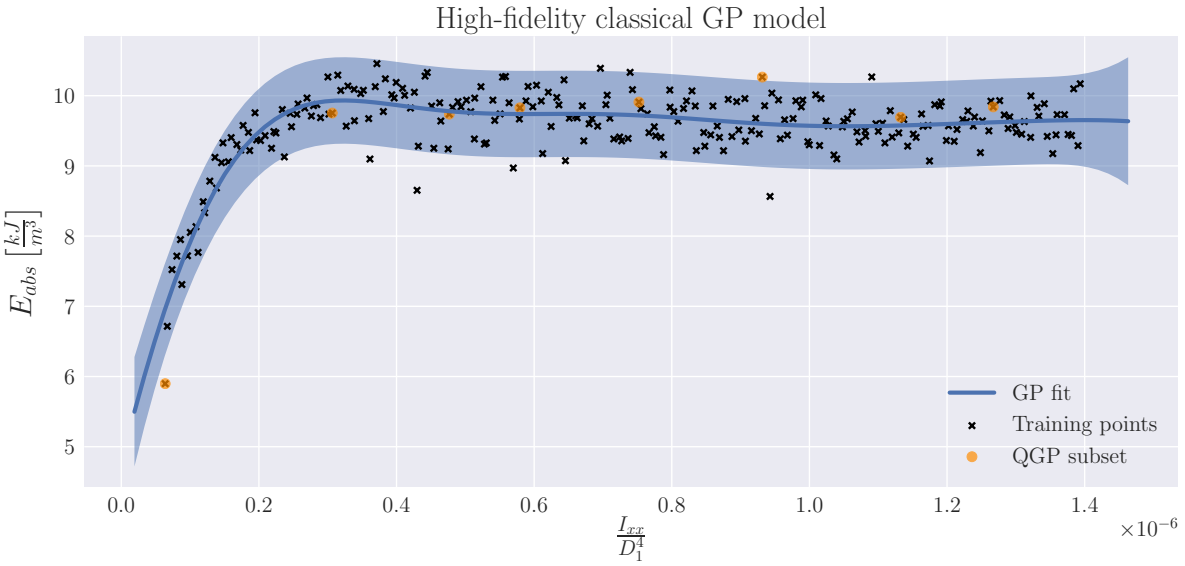


Figure 5.2: High-fidelity GP model trained on 245 training points, used for validation of the QGP results. The uncertainty interval shown as the shaded area, corresponds to 95%, which explains why few out of the 245 data-points remain outside.

Remark 6 *Figure 5.2 illustrates well the effect of imperfections on the design performance, resulting in scatter of the data-points. As a consequence, two samples with exactly the same design (i.e. with the same values of the design parameters) can have very different performance. For instance at* $\frac{I_{xx}}{D_1} = 0.95 \times 10^{-6}$ *the lower sample point indicates on* $E_{abs} = 8.5 \left[\frac{kJ}{m^3}\right]$ *, while the upper point achieves* $E_{abs} = 10.2 \left[\frac{kJ}{m^3}\right]$*, which results in over* 15% *difference. While challenging for inference, those variations represent well the reality, where geometric imperfections are inherent to any physical realizations of materials and structures. This illustration motivates the importance of quantifying uncertainty throughout the design process, but also presents how well GP can deal with this issue, allowing for robust design.*

5.2.2. Reduced GP model

To build the reduced GP model, a subset of eight training points was selected out of the 245 data-points, as shown in Figure 5.2, marked with the orange dots. To ensure that the subset spans the entire range of $\frac{I_{xx}}{D_1^4}$ values, the points were selected semi-randomly, by splitting the domain on eight intervals and randomly drawing one point out of each interval.

This reduced dataset, however, did not allow for obtaining reasonable kernel hyperparameters using standard optimization method. The reason for that is the peculiar shape of the inferred function, specifically the kink, which in the reduced data-set is probed with only a single point (the first point in the reduced subset, with the smallest $\frac{I_{xx}}{D_1^4}$ value). This point is treated by the optimizer as an outlier and neglected, which results in kernel hyperparameters that make the GP model fit the remaining seven points with a straight line.

To counteract this effect and allow for a reasonable replication of the true E_{abs} vs. $\frac{I_{xx}}{D_1^4}$ relation (given by the high-fidelity GP model), while using only eight training points, the kernel parameters of the high-fidelity model were used instead. In this way, the resulting model provides a reasonable approximation of the true relationship, as shown in Figure 5.3.

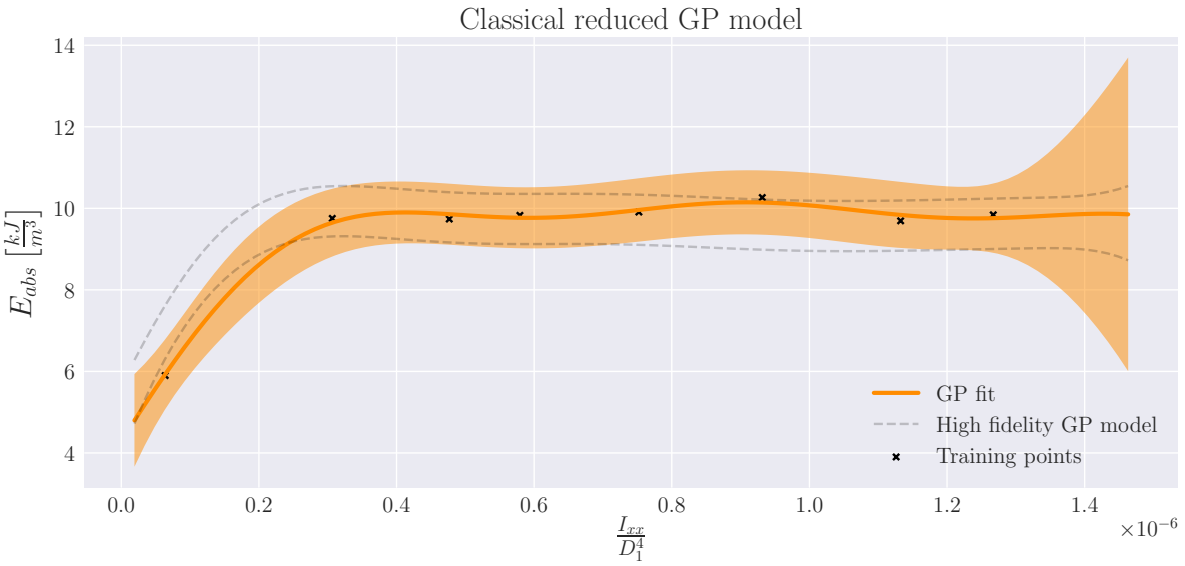


Figure 5.3: Reduced classical GP model built on eight training points.

5.2.3. Inference of E_{abs} with QGP

The energy absorption was predicted using the QGP algorithm fed with inputs prepared for the reduced GP system (i.e. the covariance matrix K and the cross-covariance vector k_*). The QGP parameters used for running this example were set based on the analysis of the algorithm performance (discussed in Chapter 3). The parameter values are listed in Table 5.4. The results are shown in the upper plot in Figure 5.4, which compares the QGP predictions to that of the classical reduced GP model.

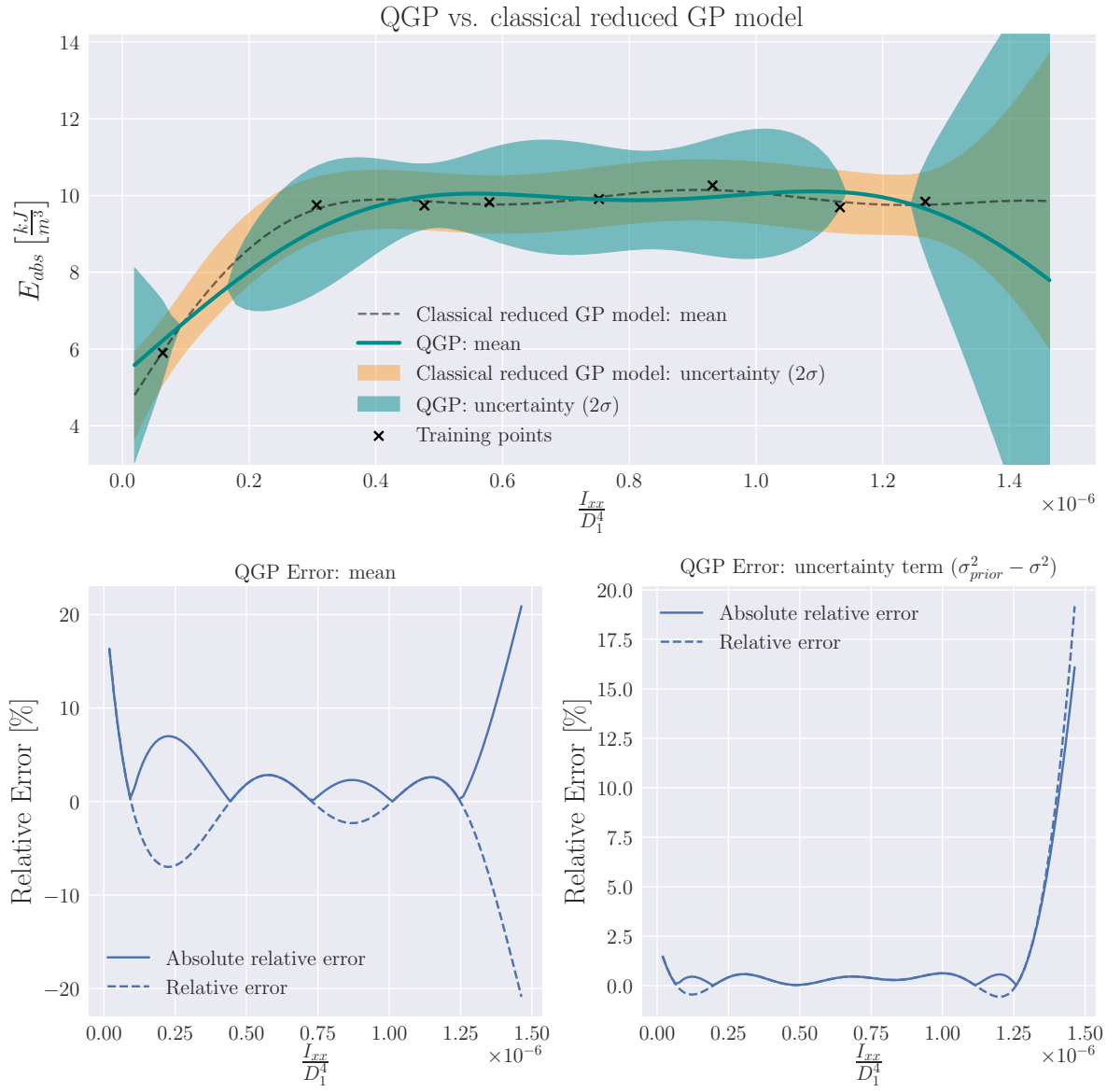


Figure 5.4: Comparison of the quantum GP with classical counterpart, trained on a data subset of 8 training points.

Mean

The mean predicted with the QGP algorithm provides a reasonable approximation of the trend obtained with the classical reduced GP model. Nevertheless significant local deviations occur. Those deviations were quantified in terms of errors in the bottom left plot in Figure 5.4. While in most parts

of the design domain, the errors are in order of $1-5\%$, yielding a reasonable approximation, large errors are observed as well, particularly at the domain boundaries, where the discrepancies reach as much as 15%.

The main source of errors is the low rank approximation induced in QGP, as discussed in subsection 3.2.3. The hypothesis is proved with the numerical example presented in subsection 3.5.5, which studies the same GP system from the perspective of numerical performance. This effect also explains the wave-like shape of the error function, which in fact corresponds to a superposition of the eigenmodes cut-off in the approximation.

Table 5.4: Parameters used for running the QGP example.

Parameter	Value
n	8
r	25
k	6
Expansion	Suzuki 2^{nd} order

Uncertainty

The confidence bounds obtained with QGP shown in Figure 5.4 indicate a significant struggle of QGP with predicting the uncertainty. While the classical reduced GP model produces a reasonable confidence interval, with values in similar order of magnitude as that of the high-fidelity reference model, the QGP results exhibit considerable variations - from overestimating the uncertainty by as much as factor 3 in some locations, to producing negative values in some others (on the plot the interval width is 0).

Surprisingly, those issues are not so apparent from the QGP errors, shown in the lower right plot in Figure 5.4, which indicate on exceptionally low values - in order of 1%. The reason for this is that the QGP algorithm does not calculate the variance directly, but only allows to calculate the term: $k_*^T(K+\sigma_n^2 I)^{-1}k_*$, which is the difference between the prior and posterior variances, as shown in page 77.

$$\sigma^2 = \sigma^2_{prior} - k_*^T(K+\sigma_n^2 I)^{-1}k_* \tag{5.1}$$

In this example, however, the term $k_*^T(K+\sigma_n^2 I)^{-1}k_*$ is very close in magnitude to σ^2_{prior}. As a result, the posterior variance σ^2 is few orders of magnitude smaller than the two other terms of Equation 5.1. Consequently, even the exceptionally small errors in $k_*^T(K+I\sigma_n^2)^{-1}k_*$ are in the same order of magnitude as σ^2, which results in errors in the uncertainty (σ) in order of 100%.

This behavior also explains why QGP predicts local intervals with 0 uncertainty in Figure 5.4. Note that those locations correspond to the intervals for which the QGP error in estimating $(\sigma^2_{prior}-\sigma^2)$ was negative (see the lower right plot). Negative error indicates that the QGP overestimated the term $k_*^T(K+\sigma_n^2 I)^{-1}k_*$, which even with a deviation in order of 1% becomes larger than σ^2_{prior} yielding non-physical negative σ^2 term. This problem is discussed further in Chapter 6.

5.3. Optimizing design with two parameters

In this second design case, the unit cell is optimized with two variables: moment of inertia of the longeron (I_{xx}) and pitch of the unit cell (P). Furthermore, the cell is simultaneously optimized for two objectives: maximizing the absorbed energy and maximizing the critical buckling load P_{crit}. This makes the problem more representative for the challenges faced while designing a super-compressible metamaterial, where increasing the energy absorption is key for its functionality (as an energy absorber), yet ensuring high critical buckling load prevents the material from collapsing under its own weight (once the unit cells would be assembled in a 3-D lattice).

Similarly as in the first design case, the problem was set-up based on Glowacki's work [24]. The choice of design variables, as well as the values of the remaining design parameters were dictated by providing a non-trivial demonstration case for the quantum-enhanced method[2], rather than practical reasons. By analyzing the design contour plots provided in Glowacki's work, the parameters values were set as shown in Table 5.5. The ranges of the design variables were chosen in the same way.

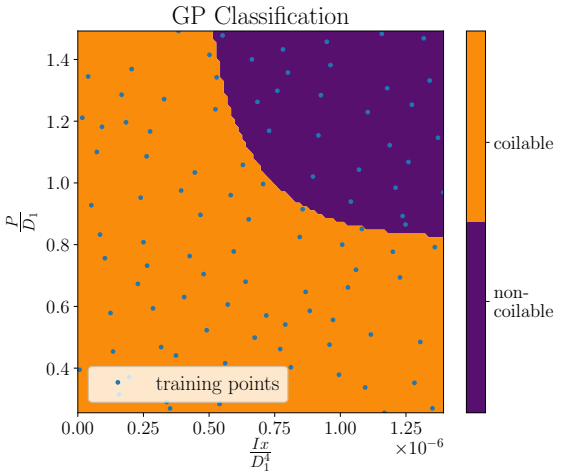


Figure 5.5: Classification of the design space.

Parameter	Value
$\frac{P}{D_1}$	$[0.25 - 1.5]$
$\frac{D_1 - D_2}{D_1}$	0.5
$\frac{A}{D_1^2}$	1×10^{-3}
$\frac{G}{E}$	0.36
$\frac{I_{xx}}{D_1^4}$	$[2.5 \times 10^{-11} - 1.3918 \times 10^{-6}]$
$\frac{I_{yy}}{D_1^4}$	7.5×10^{-7}
$\frac{J}{D_1^4}$	2.5×10^{-6}

Table 5.5: Design parameters used in the 2D design case

Similarly as in the previous example, the design problem was approached following the procedure of the data-driven design framework. First, 99 sample designs were generated with the design of experiments step. Next, the response of the sample designs was obtained with numerical models, followed by classification and partitioning of the design space on coilable and non-coilable domains, as show in Figure 5.5. Out of the initial 99 design points, 69 were qualified as coilable and used as an input to finally build a machine learning model with GPs.[3].

GP models set-up

The GP models used in this example were set up with *Matern52* covariance function, instead of the more common RBF. The choice was motivated by findings of Glowacki [24], which show that *Matern52* function provides a better fit in this data-set. Additionaly, this makes the QGP results more comparable with Glowacki's work.

[2] By analyzing the design contour plots in Glowacki's work, it can be concluded that the values of interest obey rather simple relationships with respect to most of the design variables. For this proof-of concept design case, a combination of parameters was sought to provide non-linearity making the case more interesting for the demonstration purposes.

[3] As this design problem deals with optimizing two design objectives, two models were built - one for the absorbed energy, and one for the critical buckling load.

To facilitate better optimization of the hyperparameters, (and consequently better fit) the GP models were set up on scaled data. This step is a standard procedure used especially in multidimensional cases, where the data along different dimensions differs by few orders of magnitude (in this case, six orders of magnitude). The scaling was carried out using standard *scale* routine from *scikit learn* Python package[55].

The reduced GP models were built using subset of eight training points, which were chosen by arbitrary selection out of several random sets (consisting of eight points each), such that the resulting reduced model produced a reasonable imitation of the high-fidelity reference (note that the subset of eight training points for E_{abs} model and P_{crit} model is not the same). Unlike the first design example, here the hyperparameters of the reduced model were obtained by optimization directly on the training subset. The resulting kernel function was then used to generate inputs for the classical reduced GP and QGP. Finally, the quantum algorithm was executed using quantum simulator. The algorithm parameters settings are shown Table 5.6.

Table 5.6: Overview of the QGP parameters used in the simulations for the 2D design case.

Parameter	Value
n	8
r	40
k	8
Expansion	Suzuki 2^{nd} order

5.3.1. Critical buckling load P_{crit}

The classical GP models predicting the critical buckling load are compared in Figure 5.6. The reduced model approximates reasonably well the mean from the reference model, both in terms of the shape of the function landscape, as well as the values. The uncertainty landscape however, is much different: while in the vicinity of the eight training points the confidence of the reduced model is similar to that of the high-fidelity reference, in regions further away, the values increase up to two orders of magnitude. This could be expected, as using less data increases uncertainty.

QGP prediction

The prediction of the P_{crit} landscape obtained with the QGP model is shown in the bottom plot in Figure 5.6. Comparing the predicted mean values with the classical reduced model, it can be concluded that the quantum version replicates relatively well the general trends, and the values are in similar order of magnitude. Most importantly, QGP results overestimate the maximum of the critical buckling load, providing a value of $54\frac{kN}{m^2}$, while both classical models indicate values closer to $48\frac{kN}{m^2}$.

Those errors in the predicted mean values are quantified and shown in Figure 5.7. The left plot, which shows the relative QGP error (with respect to the classical reduced model), indicates that indeed for majority of the design space the errors are in order of $10-15\%$. For small values of $\frac{I_{xx}}{D_1^4}$, however, the errors are particularly elevated, reaching values in order of 100%. This is caused by combination of two effects. First, the absolute error values are indeed elevated in this domain, as shown in the right plot in Figure 5.7, which has its origin in low rank approximation (proved in Appendix D). Secondly,

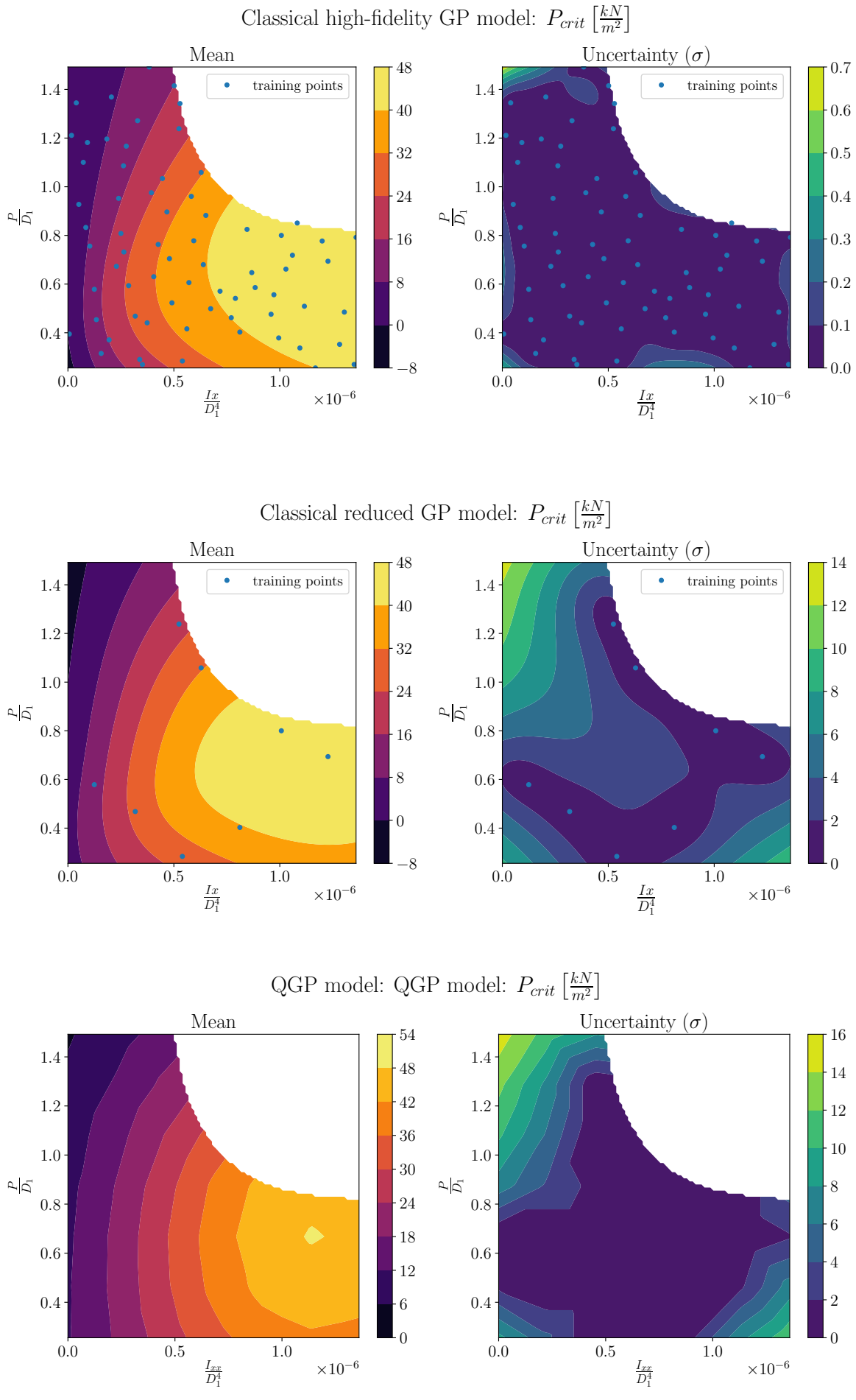


Figure 5.6: Three GP models for prediction of the critical buckling load.

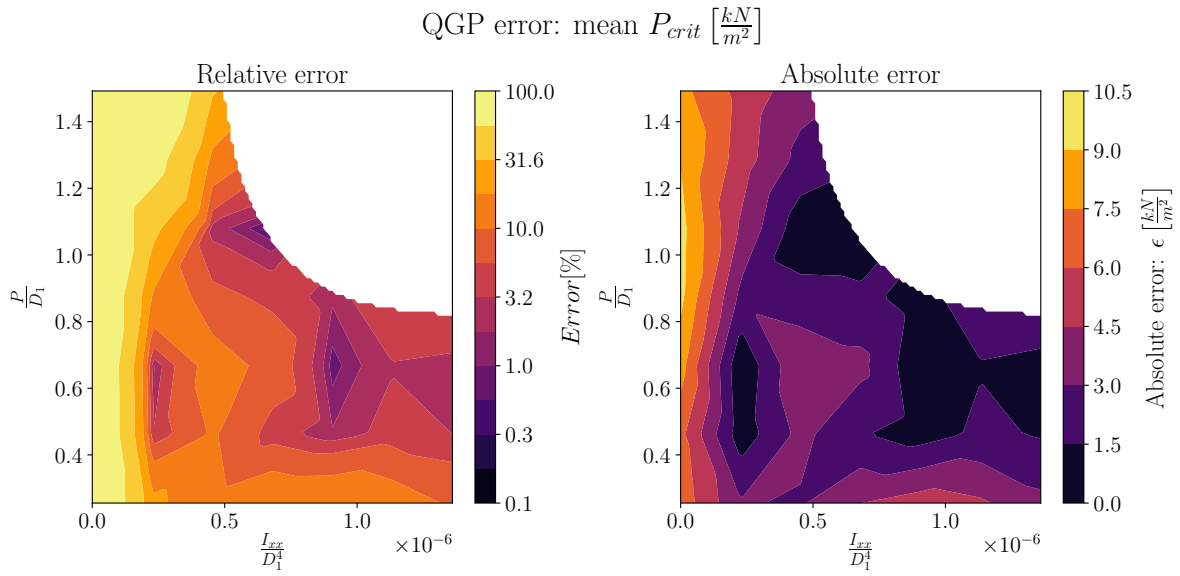


Figure 5.7: Critical buckling load (P_{crit}) predicted with QGP algorithm

the reference model in Figure 5.6 indicates that the P_{crit} values are particularly small for small $\frac{I_{xx}}{D_1^4}$ which additionally amplifies the relative error.

Considering the prediction of uncertainty shown in the right contour plot in Figure 5.6, the QGP results indicate a significant struggle, similarly as in the first design case considered before. In most of the design space the predicted uncertainty shown in the right contour plot in Figure 5.6 is 0 (while in fact some of the results of numerical experiments were found to be negative). On the other hand, in domains where the QGP yields positive values, those are mostly overestimated (compared to the classical GP). The errors in uncertainty are analyzed further in more detail in Appendix D, focusing on the numerical aspects.

5.3.2. Absorbed energy E_{abs}

Examining the high fidelity model for the absorbed energy shown in the top plot in Figure 5.8, the function landscape is less regular compared to that observed for P_{crit}, as can be seen by more curved contours. This poses a challenge for the reduced model, which aims to approximate this landscape based on only few training points, shown in the middle plot in Figure 5.8. While the overall trend, as well as magnitude of the maximum is replicated rather well, some details of the contours curvature are missing.

Regarding the uncertainty predicted by the reduced model, the situation is very similar to that observed in the P_{crit} - the uncertainty approaches the values of the reference model only in close vicinity of the training points, while for the remaining parts of the design space, the uncertainty is by up to two orders of magnitude higher.

QGP prediction

The prediction obtained with QGP algorithm is presented in the bottom plot in Figure 5.8. The predicted mean represents very well the one obtained with the classical reduced GP model, both in shape of the function as well as magnitude. The discrepancies quantified in Figure 5.9, show that the errors are in order of 5% in the majority of the design space, which is similar to the accuracy reached

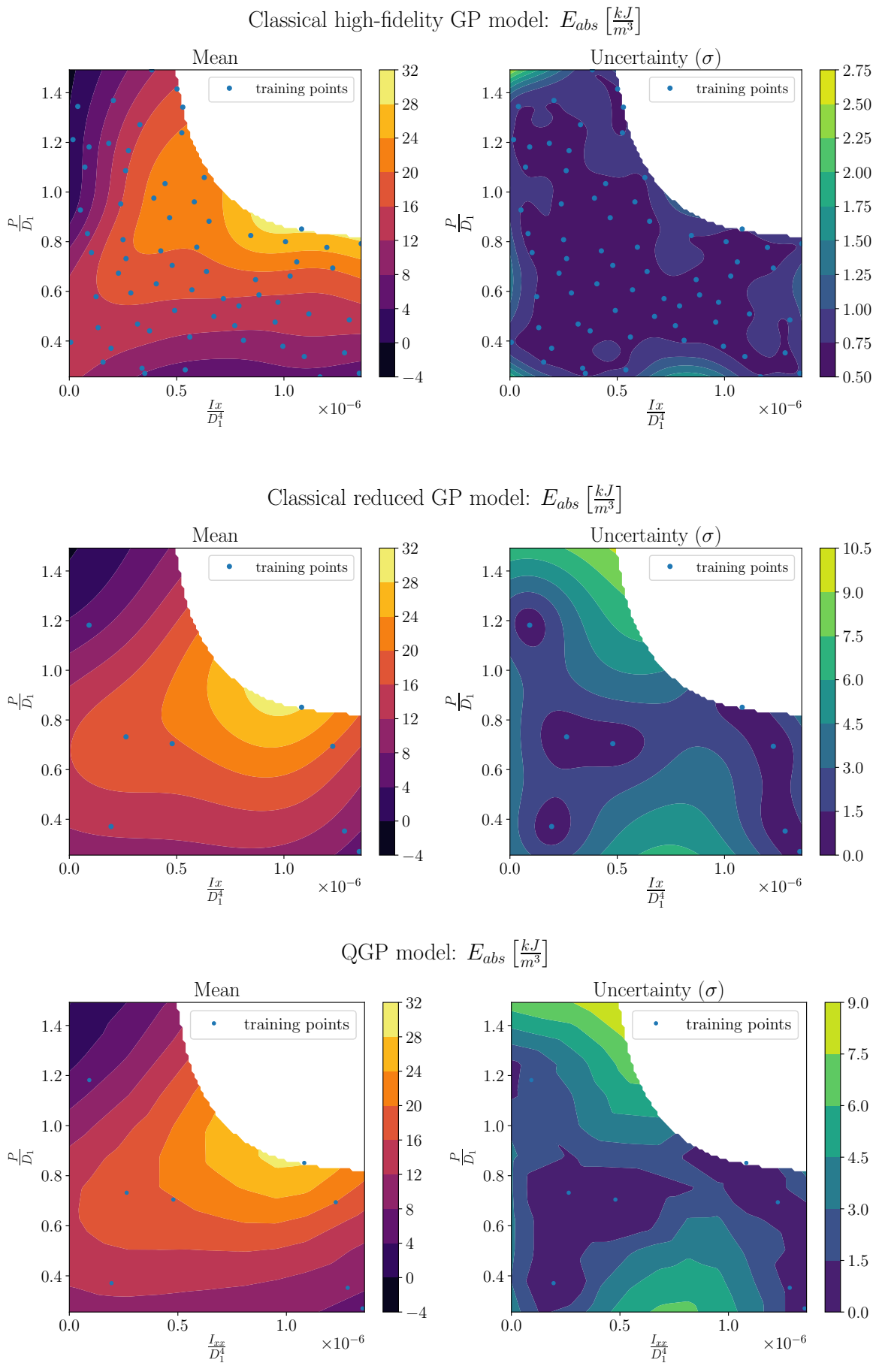


Figure 5.8: Classical GP models for inference of the absorbed energy depending on two design parameters: $\frac{P}{D_1}$ and $\frac{I_{xx}}{D_1^4}$

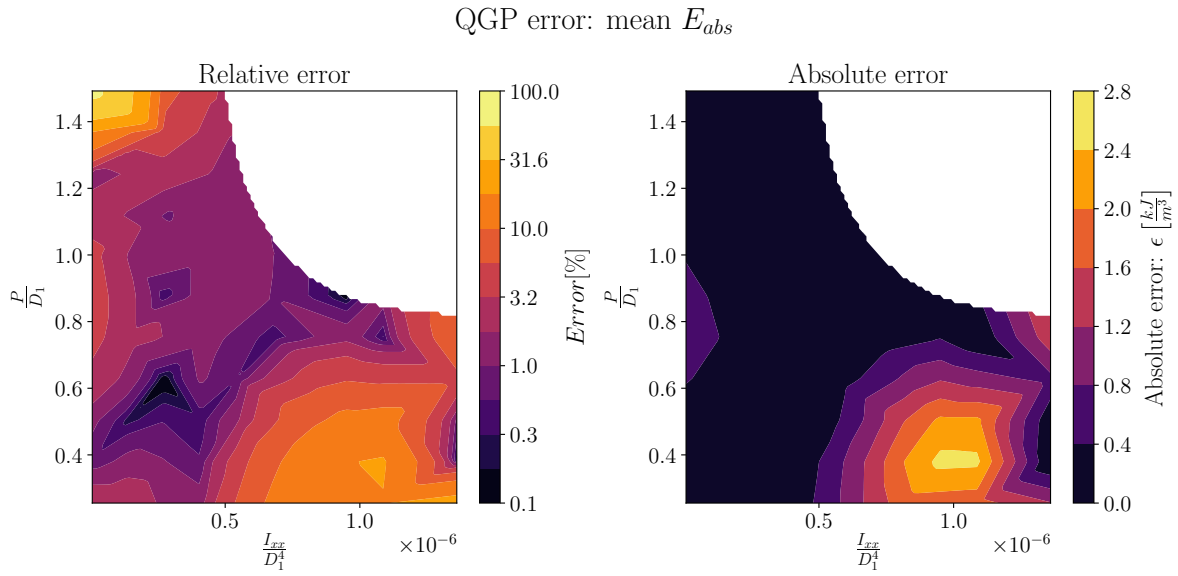


Figure 5.9: Error of the QGP prediction for E_{abs} with respect to the classically solved system

in inference of P_{crit}. However, local spikes up to 100% still occur at the boundaries of the design domain. In the vicinity of the global maximum (which is the region of interest) the errors fall below 1%.

Unlike the for P_{crit}, for E_{abs} the QGP model provides a reasonable approximation of the prediction uncertainty compared to that obtained with the classical reduced model. This observation is supported by the quantitative error analysis provided in Appendix D.

5.3.3. Optimal design solution

The main objective of the problem is to find the values of the design variables $\frac{P}{D_1}$ and $\frac{I_{xx}}{D_1^4}$, such that the resulting design maximizes both E_{abs} and P_{crit}. The problem is therefore a simple case of multi-objective optimization. The design point can be found by maximizing an objective function, which for this problem can be specified as follows:

$$f\left(\frac{I_{xx}}{D_1^4}, \frac{P}{D_1}\right) = E_{abs}\left(\frac{I_{xx}}{D_1^4}, \frac{P}{D_1}\right) \cdot P_{crit}\left(\frac{I_{xx}}{D_1^4}, \frac{P}{D_1}\right) \tag{5.2}$$

This function is used to find the optimum design point predicted by QGP and the classical high-fidelity GP model. For this purpose, the objective function was evaluated using both models, as shown in Figure 5.10. The maxima of the function, which correspond to the optimal designs predicted by the two model, were marked with a red 'x'. The exact locations of those points expressed in terms of design variables are listed in Table 5.7.

The optimal design predicted by QGP is relatively close to the point from the reference model. The $\frac{P}{D_1}$ coordinates differ by less than 5%. The difference in $\frac{I_{xx}}{D_1^4}$ is considerably larger, however, as in the vicinity of the maximum the function is elongated along $\frac{I_{xx}}{D_1^4}$, the performance at the two optimal points is very similar. The critical buckling load differs by 0.2 $\frac{kN}{m^2}$, while the discrepancy in the energy absorption is below 6%.

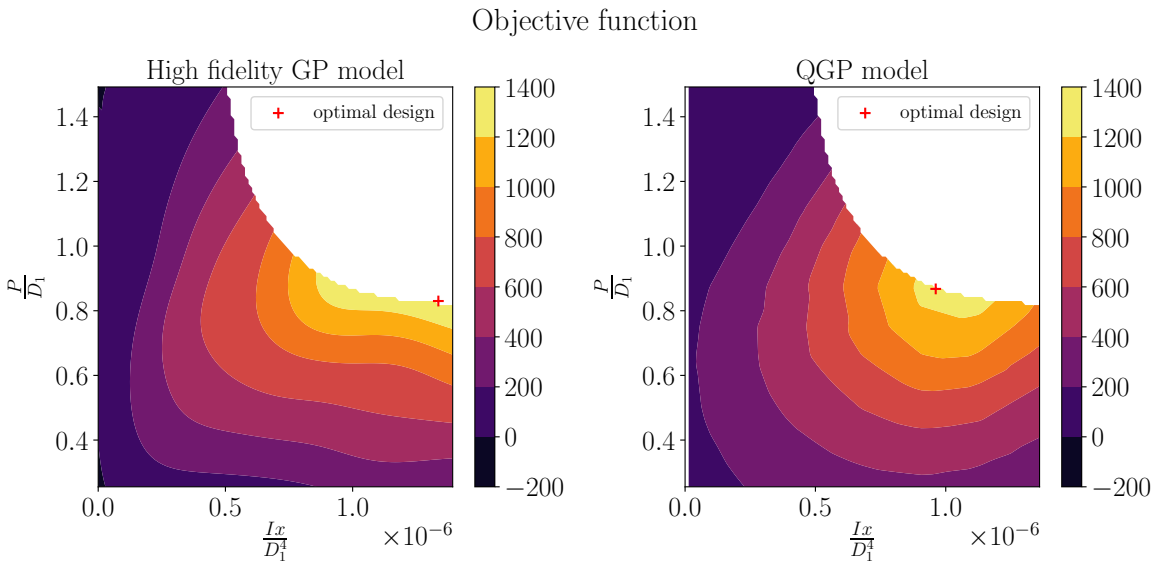


Figure 5.10: Objective function obtained with QGP model, validated with high-fidelity classical GP model.

Table 5.7: Optimal designs obtained with QGP and accurate classical reference model.

Model	$\frac{I_{xx}}{D_1^4}$	$\frac{P}{D_1}$	E_{abs}	P_{crit}
Classical reference GP	1.34×10^{-6}	0.83	47.3	29.2
QGP	9.61×10^{-7}	0.87	44.5	29.0

6

Discussion on QGP application

This chapter provides a discussion on the applicability of the QGP algorithm as part of the data-driven design framework, by analyzing the outcomes of the two proof-of-concept design cases, presented in the previous chapter. This discussion addresses three key subjects: first it verifies the applicability of QGP as an integral part ot the data-driven design framework, by evaluating its performance in a practical setting, which is done by comparing the QGP results with Glowacki's work [24], who tackled the same design task with classical sparse Gaussian processes. This comparison is presented in Section 6.1. Secondly, the discussion points out the performance deficiencies of the method, and proposes solutions to improve applicability of the algorithm in practical problems. This part is addressed in Section 6.2. Finally, the limitation of the examples are addressed, specifically, regarding the limited datasets. Section 6.3 provides with a quantitative analysis of influence of the dataset selection on the model performance, while Section 6.4 comments on QGP's performance when facing limitations in number of training points and discusses different strategies for improving effectiveness of inference in such conditions.

6.1. Comparing results with literature

The cell performance predictions obtained with QGP for the 2D example are compared with the results obtained by Glowacki, to verify how well does the QGP perform in respect to the classical models. For this purpose, Glowacki's model and data-sets were used to generate the design contour plot for E_{abs} and P_{crit} for the design parameters set to the values used in the QGP application example (see Table 5.5). The resulting contour plots are shown in Figure 6.1.

The differences between the models

Before comparing the results, it needs to be emphasized that there are several fundamental differences between the QGP and Glowacki's model. First of all, Glowacki's model uses not two, but seven design variables. Therefore, the contour plot shown in Figure 6.1 is only a two-dimensional slice of the seven-dimensional prediction model covering the complete design space. Note that this case also required sampling in all seven dimensions. Consequently, the prediction shown in Figure 6.1 could be also slightly affected by the variations in the remaining five dimensions.

This also results in a difference in classification of the design space. Specifically, in the QGP application example, the non-coilable domain is slightly larger (more conservative). This effect is most likely due to scarcity of the data in case of Glowacki's model, while in the QGP application the domain was

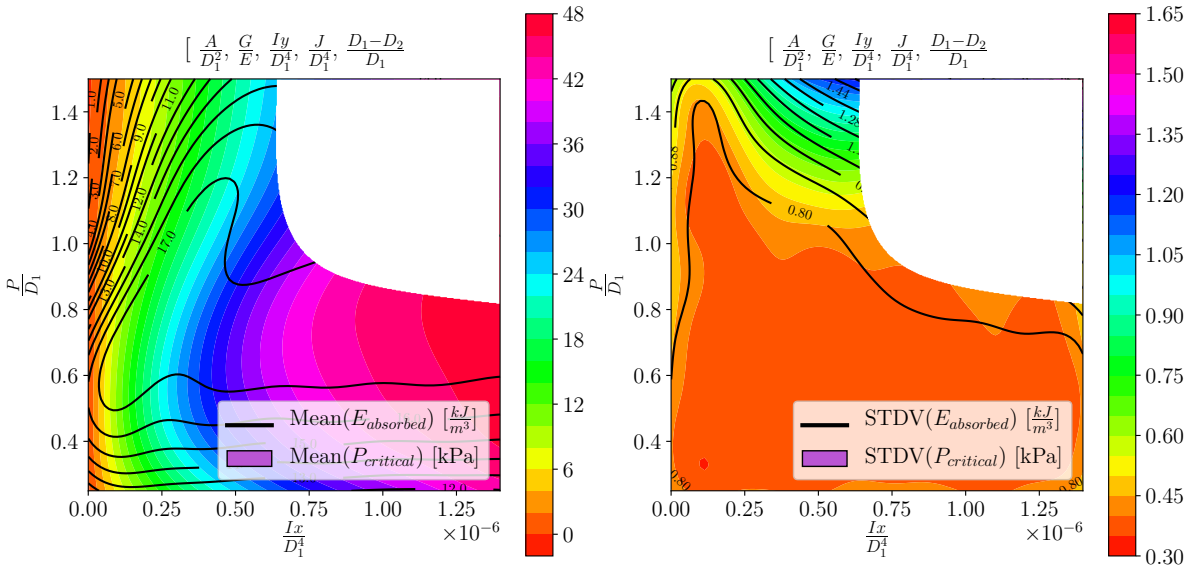


Figure 6.1: Results generated for the 2d example with Glowacki's model

partitioned using a classical classification model based on the 99 training points, which allowed for better accuracy.

Furthermore, Glowacki's model relies on sparse Gaussian processes, which is only an approximation. Consequently, it is expected that this model is less accurate than the full GP solution (including the high fidelity reference models, Figure 5.6 and 5.8). On the other hand, it makes an adequate case for comparison with QGP algorithm, which is an approximation as well.

Mean

Examining the prediction of the critical buckling load in Figure 5.6 and 6.1, both QGP and Glowacki's models are in good agreement, regarding replicating the trends as well as estimating the values. Those results also show consistency with the predictions of the high-fidelity classical GP model. The two models, however, differ significantly in predicting the energy absorption. Glowacki's result predict the maximum absorbed energy to roughly $17\frac{kJ}{m^3}$, while QGP predicts almost double of that, estimating the maximum absorbed energy of $32\frac{kJ}{m^3}$, which is more consistent with the high-fidelity reference model.

Furthermore, Glowacki's results are quite ambiguous in providing the location of the maximum.This is because of relatively poor function reconstruction in that region (can be seen by the shape of the contours). As a result, the contour in Figure 6.1 indicates that that the maximum is likely to be anywhere along the line corresponding to $\frac{P}{D_1} = 0.8, \frac{I_{xx}}{D_1^4} \in [0.3, 1.4]$.

Uncertainty

On the other hand, the classical sparse Gaussian processes appears to well replicate uncertainty of the prediction. As show in Figure 6.1, the values turn out to be in a very good agreement with the classical high-fidelity model - the uncertainty on the predicted energy is in order of 0.8 in the majority of the domain - the same as in the reference model. The uncertainty of $\sigma \approx 0.4$ for the critical buckling load is only slightly larger, yet still in the same order of magnitude.

Considering the differences in the predicted mean E_{abs}, however, this only shows how unreliable this predicted uncertainty really is. For instance, consider an extreme case when Glowacki's model predicts that the design with $\left(\frac{P}{D_1} = 0.8, \frac{I_{xx}}{D_1^4} = 1.0 \times 10^{-6}\right)$ yields $E_{abs} = 17 \pm 1.6 \frac{kJ}{m^3}$ with 95% confidence, while the high-fidelity model predicts for this point a value of $E_{abs} = 30 \pm 2 \frac{kJ}{m^3}$ (with the same 95% confidence). This indicates that similarly as in QGP, Glowacki's model based on classical sparse Gaussian processes also might be struggling with reliable prediction of the confidence bounds.

6.2. Struggle with estimating uncertainty

Both application examples presented in Chapter 5 exposed the QGP struggle with estimating the variance. While at certain sampled points the QGP overestimates the confidence bounds by a factor of 2, at others it estimates the variance as negative. These can be explained by how the variance is being calculated as: $\sigma^2 = \sigma^2_{prior} - k_*^T (K + \sigma_n^2 I)^{-1} k_*$, where the QGP algorithm is used to evaluate the second term, i.e: $k_*^T (K + \sigma_n^2 I)^{-1} k_*$. The difference of the two terms, equivalent to the posterior variance σ^2, is much smaller than any of the terms (from numerical tests: $O(0.1)\%$). In this case, the error in the QGP result becomes amplified by $O(1000)$, which leads to large overall errors in σ.

On the other hand, the problems with variance estimations are inherent to all sparse GPs, as discussed by Bauer et al. [4]. For instance, one of the most common sparse GP methods - FITC is know for its tendency to underestimate the variance - in some cases severely, as shown in Figure 6.2. The alternative method - VFE prevents that by using its more rigorous formulation. This, however, often leads to overestimation.

Although the underlying mechanisms responsible for the issues in variance estimation for QGP are completely different than in FITC or VFE, it can be considered that they are also related to the sparsity of QGP, as the major contribution to the error is an effect of the rank-reduction.

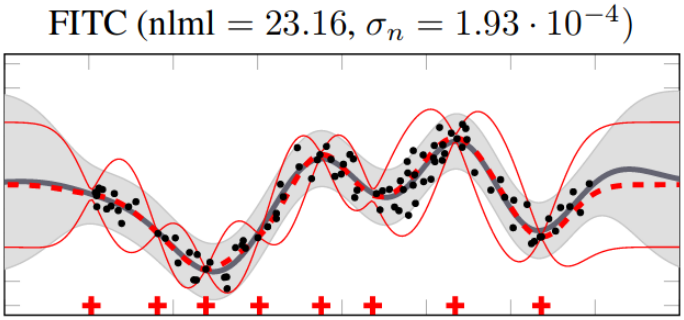


Figure 6.2: Example of variance underestimation in Sparse GP approximation [4]. The predicted mean and confidence bounds are plotted with gray for the exact GP, and with red for FITC

Nevertheless, the ability of estimating uncertainty of the prediction is an essential feature of GP, and oftentimes the key factor that favors choice of this method for solving problems. A reliable variance estimation is therefore paramount for the applicability of this method.

6.2.1. Correction of the uncertainty prediction

As the QGP intrinsically gives an approximation to GP, the QGP results are in general always affected by some errors which might lead to numerical artifacts such as negative variance. Consequently the QGP predictions can be thought of as subjected to a larger uncertainty.

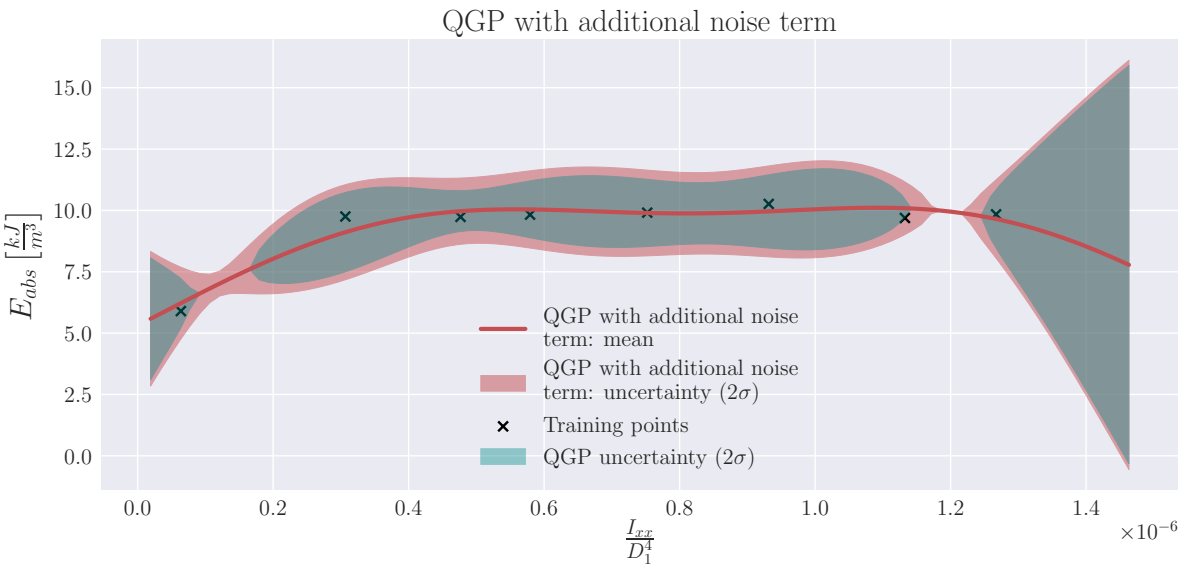


Figure 6.3: Comparison of the quantum GP with classical counterpart, trained on a data subset of 8 training points. The GP model (used in both QGP and classical GP) has an additional noise term $\sigma^2_{QGP}I$ to account for the uncertainty from the QGP approximation.

A one way to account for this additional QGP uncertainty is to add another noise term to the covariance matrix: $\sigma^2_{QGP}I$ accounting only for the component of uncertainty due to the QGP approximations. This value of σ^2_{QGP} could be estimated from the error contour plots given the parameter settings (k and r).

In order to verify this concept, it was implemented and tested for the QGP model used in the 1D example problem. In this case, the QGP error variance (σ^2_{QGP}) was estimated assuming QGP error of 2.5% of the mean value, based on the error contour plots. The results are shown in Figure 6.3, which illustrates the new corrected error bounds imposed on the standard QGP model. By examining the plot it can be seen that the additional error term adds robustness to the uncertainty interval, resulting in a more conservative prediction. Furthermore, it can be observed that the strongest effect of this additional error term occurs particularly at the locations where the uncertainty is severely underestimated, which helps to prevent from reaching non-physical negative variance values.

While this solution allows to circumvent the problem, it does not completely solve it, as the resulting uncertainty prediction of this approximation is still far from the full GP solution. Ideally this problem should be addressed with an algorithmic solution, such that the amplification of the QGP error could be eliminated, which remains a subject for the future work.

6.3. Influence of training point selection

Selection of the training points is a key factor in performance of the predictive model. Placing points in crucial function locations (close to maxima, minima, in regions of large gradients) improves chances of more accurate reconstruction.

In the 2D design example of QGP application, the training points were selected semi-randomly by testing few different sets of randomly chosen training points, and selecting one that provides a reasonable replication of the high-fidelity model. This selection certainly has a tremendous influence on quality of the QGP results.

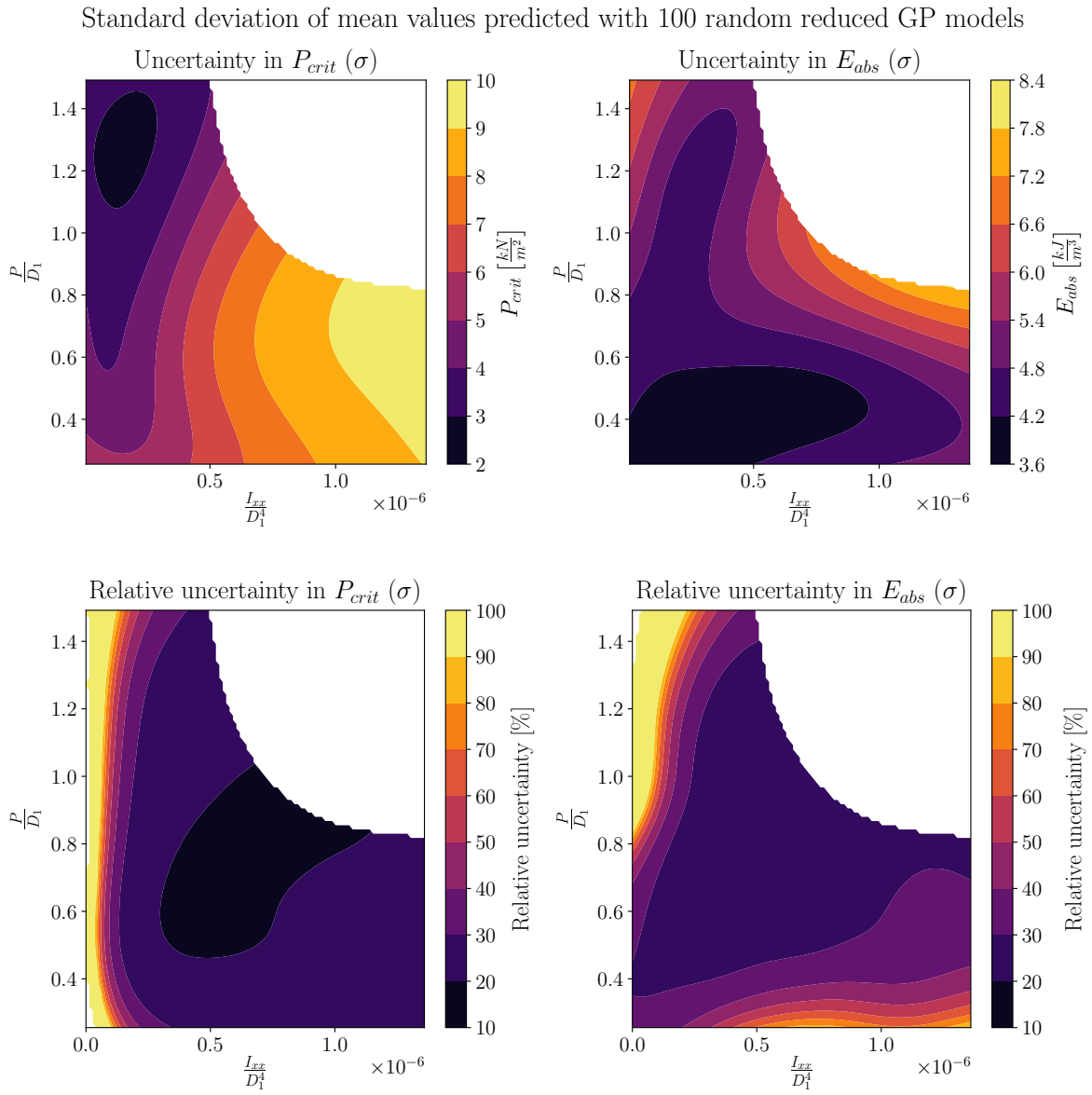


Figure 6.4: Sensitivity of the predicted mean values on selection of training dataset. The relative uncertainty was calculated with respect to the mean value of the function obtained with the high fidelity GP models.

Therefore, in order to verify the impact of the choice of training points on the model performance, 100 sample reduced GP systems were composed, each using a set of randomly selected training points. This ensemble of test systems was then solved classically to predict mean value. The difference in predictions obtained with different systems was quantified by calculating standard deviation in the prediction values. The results are shown in Figure 6.4.

The obtained deviation values are relatively large. The largest absolute variation (two upper contour plots in Figure 6.4) in variables is observed in vicinity of maxima in both cases. The relative variation, however, is mostly uniform across the whole design domain, as shown the bottom contour plots in Figure 6.4, with values at a constant level of around 20% with respect to the mean values for both P_{crit} and E_{abs}.

The variation in the predicted values is also slightly elevated close to the domain boundaries, which

is particularly well illustrated with the relative uncertainty in Figure 6.4. The approach of completely random selection of training points, however, is rather conservative, and not completely representative for the practical approach. In practice, the training points are usually sampled according to some scheme, which ensures quasi-random and space-filling distributions over the domain and increases chances of inferring all the essential features of the underlying function. In contrary, completely random selection does not exclude unrealistic cases of particularly inefficient sampling, where for instance all the points are gathered in only one part of the considered domain.

6.4. Scarcity of data and the problem of dimensionality

In most practical cases, the amount of data used to build a model translates directly onto quality of the model and its predictive capability. In case of QGP examples, discussed in previous chapter, the training sets were limited by the computational cost (of simulating the QGP) to systems of merely eight points. However, the challenge of reconstructing function from limited data is not exclusive to QGP examples, and relates closely to the issues faced by the classical high-dimensional models. In general case, it can be expected that the amount of data points needed to sample a design space scales exponentially with the number of design parameters (i.e. the space dimensions). To prevent exponentially high computational costs, those models can afford only scarce data-sets.

6.4.1. Comparison of sampling density

For instance, for modeling seven-dimensional design space Glowacki uses 59,991 points. Assuming that the number of points is given roughly by $n = \rho^d$, where d is number of dimensions and ρ is some sampling density parameter which relates to the number of points per dimension [1], for Glowacki's dataset the sampling density parameter is $\rho \approx 5$ (which means that in grid sampling scheme there would be roughly 5 points per dimension). This dataset, however is further approximated with 1,200 inducing points (which if treated as refined dataset, would yield the sampling density factor of $\rho \approx 2.75$ data-points per dimension). For comparison, the QGP example of 2D optimization is sampled with density parameter of $\rho \approx 2.81$ points per dimension.

Furthermore, the two models rely on different sampling schemes: Glowacki uses an efficient Sobol sequence[70] (which however in his case is disrupted due to classification process that truncates part of the domain)[24], while the QGP examples rely on a semi-random selection of the training points. Therefore, the derived density parameters do not provide grounds for rigorous comparison, but only an indication. Nevertheless, the same order of magnitude of the parameter estimated for both cases indicates that the QGP examples are sampled with a similar density as the Glowacki's model.

Consequently it could be expected that despite so few training points available, yet with similar sampling density, the QGP model provides a similar predictive performance as the classical sparse GP model built by Glowacki - a hypothesis which turns out consistent with the numerical results analyzed in Section 6.1.

6.4.2. Performance in limited data-sets

The amount of data-points required to reconstruct a function is related to the function complexity (i.e. the amount of variations, local minima etc.). Even in small data-sets, however, reasonable models can be created by taking advantage of the prior knowledge on the inferred function.

[1] Note that this sampling density model holds well in case of simple grid sampling, but in more advanced sampling schemes (which is the case here) it can not be always related to the properties of the data-structure, therefore in those cases it is used only as an indication.

Complexity of the inferred function

In many practical cases, the functions are often relatively smooth (i.e. without abrupt variations). In the considered QGP examples, for instance, the smoothness condition could be assumed on grounds of the modeled physics - as in general it could be expected that the design performance changes gradually with the change of parameters. The only exception is the transition from coilable to non-coilable regime, where the change of the response mechanism might result in abrupt changes and even discontinuities in the response landscape. To prevent that, however, the classification process was carried out which partitioned the design domain.

The smoothness assumption made for the reduced classical systems, is validated with the high-fidelity models (which show that indeed there are no abrupt local variations in the inferred functions). Consequently, in such conditions even the reduced models operating on very limited datasets were able to produce approximations of reasonable quality.

Accounting for the expected data-structure

Even in cases where the underlying function is expected to be more complex (i.e. with more variations, local minima etc.), there are still a number of techniques which increase the chances of inferring a reasonably accurate model. In case of GP models, a particularly effective way of improving the model performance is by leveraging the prior knowledge of the expected function to tailor the covariance function.

Many covariance functions are well suited for data with particular type of trends (e.g. linear, periodic, bias). For instance, in case where the inferred relationship is expected to show periodic variations, it could be reconstructed using only few training points if approached with a periodic kernel function (that accounts for the periodicity). Furthermore, those kernel functions can be combined together (as a mathematical expression, by addition and multiplication of different components) to produce a composite kernel function, which allows to account for multiple trends simultaneously. An example of inferring a model from limited data using composite kernel functions is shown in Appendix C. For the QGP application examples, however, this was not necessary.

7

Conclusions and recommendations

The objective of this dissertation was to demonstrate how quantum computing could enhance computational design of materials, specifically by exploring the concept of replacing the expensive machine learning step of a data-driven design framework with an exponentially faster quantum counterpart – a quantum algorithm for Gaussian processes (QGP). This research objective was realized in two steps: first by implementing the quantum algorithm and analyzing its performance, and secondly by integrating the algorithm within the work-flow of the computational framework for data-drive design of materials, and demonstrating its applicability on two example problems of unit cell design for a super-compressible metamaterial.

7.1. QGP algorithm

Conclusion

The aim of implementing the QGP algorithm was to verify whether the algorithm can provide sufficient performance in terms of accuracy and efficiency (scalability). The following conclusions have not been encountered in the literature and represent a novel contribution of this work:

- QGPs are intrinsically connected to classical sparse Gaussian processes (not just to full Gaussian processes);
- In general, QGP is more accurate and scalable than classical sparse Gaussian processes. This results from a double gain (accuracy and scalability) when converting Gaussian processes to quantum computing because it uses qPCA to create a low-rank approximation instead of the Nystrom approximation used in classical sparse Gaussian processes. Here, qPCA was shown to have clear advantages over Nystrom approximation;
- Reasonable error bounds for QGP have been proposed and numerically validated.

One can elaborate on the above mentioned points. This work started by implementing the QGP algorithm proposed by Zhao [79] in 2015, adapting it accordingly to Qiskit as a Python module. The algorithm was extensively tested using a simulator of a quantum computer provided with Qiskit, and numerical tests found that by controlling the approximation parameters of the eigendecomposition (carried out with the QPE subroutine) it is possible to induce a low-rank approximation equivalent to quantum principal component analysis. This exposed a new connection of QGP algorithm to the classical sparse Gaussian processes, which also rely on low-rank approximations to reduce the computational cost. In classical Gaussian processes, however, despite providing superior accuracy, the

PCA approximation is not feasible as it requires carrying out a computationally expensive eigendecomposition. On the contrary, in the quantum implementation of Gaussian processes, inducing PCA can reduce the computational cost, allowing for additional speed-up (complexity reduction linear with the QPE resolution parameter $k \sim log_2(\kappa')$, where κ' is the condition number of the approximate system; and additional reduction in overall costs due to overhead terms).

The accuracy of qPCA approximation was compared to the classical Nystrom scheme which is commonly used in sparse Gaussian processes. This was done with two approaches: analytically, by deriving error bounds, and empirically by numerical experiments. The derived error bounds indicated that qPCA could provide better accuracy than the classical Nystrom, yet it did not exclude the opposite case, as the lower Nystrom bound was found below the upper bound of qPCA. The numerical tests, however, provided evidence in favor of qPCA, proving its superior accuracy in all cases, with only a few exceptions.

The algorithm was extensively tested for different settings of approximation parameters, which allowed to obtain the algorithm approximation errors, and size of the corresponding circuit. The error data was used to generate error contour plots, which allow to estimate the expected errors for given settings. Similarly, the circuit scaling data was used together with analytically derived complexity to derive a cost model for the algorithm, which allows to predict circuit size for given settings of the approximation parameters. Although the tested QGP implementation employed a sub-optimal Hamiltonian simulation routine which does not provide the exponential speed-up, the precise estimates of the algorithm cost still provide a valuable insight about the true cost of the QGP subroutines, which are required to determine the needed resources (provided by a quantum computer) and estimate the point of reaching quantum advantage.

Finally, a special test case of QGP algorithm was prepared for execution on a real quantum computer provided by IBM for public access. For this purpose a special GP example was prepared, which allowed to optimize the corresponding quantum circuit to merely 5 qubits and 200 gates. Nevertheless, no meaningful results could be obtained as the gate error rates were still too large for a circuit of such depth.

Recommendations

The Hamiltonian simulation step, which carries out the matrix exponentiation is the most critical part of the QGP algorithm, as it is the most computationally expensive subroutine which determines the algorithm speed-up. The QGP implementation studied in this research relied on a sub-optimal Hamiltonian simulation, however, to achieve speed-up this step needs to be replaced by a more efficient one.

This challenge could be tackled in two ways. The first alternative is to explore implementation of the Hamiltonian simulation relying on a graph-coloring method, proposed originally for the HHL algorithm. The method provides speed-up, but is limited only to strictly sparse matrices, which is in general not the case for the covariance matrices. However this limitation could be circumvented by designing a kernel function that produces strictly sparse covariance matrices. Alternatively the problem of dealing with non-sparse matrices could be tackled by replacing the HHL algorithm in QGP with the one proposed by Wossnig et. al in 2018 [77]. This concept, however, relies on loading the matrix entries using qRAM, thus making it less feasible in the near future.

7.2. QGP application

Conclusion

The applicability of the QGP algorithm was demonstrated on two example design cases of optimizing a unit cell of super-compressible metamaterial. The first problem involved optimization of cells energy absorption with a single design parameter, while in the second problem the design space was extended to two parameters, and the optimization to two objectives. Both design cases were approached following the steps of the data-driven design framework, with QGP algorithm replacing the classical machine learning step, which proved the feasibility of the proposed concept.

The complexity of the demonstrated problems was severely limited by the computational cost of simulating the QGP. Therefore both examples were limited to merely eight training points. The results from this example turned-out crucial for revealing a significant deficiency of QGP in estimating the uncertainty. The problem, however, was found to be common also for classical sparse Gaussian Process approximations. The second design case (the 2D problem) was closer to real-life design challenges and provided an adequate case for comparison with the results of the classical sparse Gaussian processes from literature.

The overall predictive performance of QGP was comparable to that achieved by the classical sparse Gaussian processes, which proves applicability of this quantum algorithm in computational design of materials, and realizes the research objective.

Recommendations

In order to improve the applicability of QGP algorithm, the key issue to address is the reliability of the uncertainty prediction. The current approach relies on inefficient formulation that translates even small errors of QGP onto large errors in the posterior variance. Although the robustness of the uncertainty prediction could be improved by adding an additional error term accounting for the QGP discrepancies. Ideally this problem should be solved with an alternative algorithmic formulation, by eliminating the error amplification effect.

Besides improvement in the performance of the algorithm, the applicability of QGP could be enhanced by extending it to Bayesian optimization, which allows to find minima (or maxima) of functions with limited number of queries, promoting exploration over exploitation of the design space. The method relies on mean and uncertainty prediction obtained from GPs, which are used to construct a so-called acquisition function, that predicts probability of location of the function minimum (or maximum) and guides the exploration process. Bayesian optimization could be therefore integrated with the QGP as a single algorithm, which could potentially open-up new functionalities enabled with quantum-specific features (e.g. leveraging the principle of superposition to query multiple locations simultaneously). Such an extension could be also integrated within the data-driven framework, enhancing exploration of the design space, while reducing the size of the required data.

Acknowledgements

First of all, I would like to thank Miguel Bessa for his contribution to this project as the main supervisor. I am extremely grateful for his incredible, inspiring involvement, and his support on all aspects of this project. I would also like to thank my second supervisor, Sybrand van der Zwaag for his great support and guidance. The discussions we had were crucial for pointing the project in the right direction. His critical comments provided an important contribution to the quality of this research. Finally, I would like to thank Matthias Möller for his support with the mathematical aspects of this work.

Bibliography

[1] Héctor Abraham, Ismail Yunus Akhalwaya, Gadi Aleksandrowicz, Thomas Alexander, Eli Arbel, Abraham Asfaw, Carlos Azaustre, Panagiotis Barkoutsos, George Barron, and Luciano Bello et al. Qiskit: An open-source framework for quantum computing.

[2] Tameem Albash and Daniel A Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[3] Davide Anguita, Sandro Ridella, Fabio Rivieccio, and Rodolfo Zunino. Quantum optimization for training support vector machines. *Neural Networks*, 16(5-6):763–770, 2003.

[4] Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In *Advances in neural information processing systems*, pages 1533–1541, 2016.

[5] John S Bell and John Stewart Bell. *Speakable and unspeakable in quantum mechanics: Collected papers on quantum philosophy*. Cambridge university press, 2004.

[6] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2): 359–371, 2007.

[7] Katia Bertoldi, Vincenzo Vitelli, Johan Christensen, and Martin van Hecke. Flexible mechanical metamaterials. *Nature Reviews Materials*, 2(11):17066, 2017.

[8] MA Bessa and S Pellegrino. Design of ultra-thin shell structures in the stochastic post-buckling range using bayesian machine learning and optimization. *International Journal of Solids and Structures*, 139:174–188, 2018.

[9] MA Bessa, R Bostanabad, Z Liu, A Hu, Daniel W Apley, C Brinson, Wei Chen, and Wing Kam Liu. A framework for data-driven analysis of materials under uncertainty: Countering the curse of dimensionality. *Computer Methods in Applied Mechanics and Engineering*, 320:633–667, 2017.

[10] Miguel A Bessa, Piotr Glowacki, and Michael Houlder. Bayesian machine learning in metamaterial design: Fragile becomes supercompressible. *Advanced Materials*, page 1904845, 2019.

[11] Bart Besselink, Umut Tabak, Agnieszka Lutowska, Nathan Van de Wouw, H Nijmeijer, Daniel J Rixen, ME Hochstenbach, and WHA Schilders. A comparison of model reduction techniques from structural dynamics, numerical mathematics and systems and control. *Journal of Sound and Vibration*, 332(19):4403–4422, 2013.

[12] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[13] Robin Blume-Kohout and Kevin C Young. A volumetric framework for quantum computer benchmarks. *arXiv preprint arXiv:1904.05546*, 2019.

[14] Fernando GSL Brandao and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–426. IEEE, 2017.

[15] Johan Christensen, Muamer Kadic, Oliver Kraft, and Martin Wegener. Vibrant times for mechanical metamaterials. *Mrs Communications*, 5(3):453–462, 2015.

[16] Carlo Ciliberto, Mark Herbster, Alessandro Davide Ialongo, Massimiliano Pontil, Andrea Rocchetto, Simone Severini, and Leonard Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018.

[17] Andrew W Cross, Lev S Bishop, Sarah Sheldon, Paul D Nation, and Jay M Gambetta. Validating quantum computers using randomized model circuits. *arXiv preprint arXiv:1811.12926*, 2018.

[18] D-Wave systems inc. D-Wave 2000Q Technology overview, 2019. URL https://www.dwavesys.com/sites/default/files/D-Wave%202000Q%20Tech%20Collateral_1029F.pdf. Retrieved on 20.04.2019.

[19] Christoph Durr and Peter Hoyer. A quantum algorithm for finding the minimum. *arXiv preprint quant-ph/9607014*, 1996.

[20] David Duvenaud, James Robert Lloyd, Roger Grosse, Joshua B Tenenbaum, and Zoubin Ghahramani. Structure discovery in nonparametric regression through compositional kernel search. *arXiv preprint arXiv:1302.4922*, 2013.

[21] Mark Fingerhuth, Tomáš Babej, and Peter Wittek. Open source software in quantum computing. *PloS one*, 13(12):e0208561, 2018.

[22] NA Fleck, VS Deshpande, and MF Ashby. Micro-architectured materials: past, present and future. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 466 (2121):2495–2516, 2010.

[23] Vittorio Giovannetti, Seth Lloyd, and Lorenzo Maccone. Quantum random access memory. *Physical review letters*, 100(16):160501, 2008.

[24] Piotr Glowacki. Data-driven design of metamaterial unit cell using sparse Gaussian Processes. Master's thesis, TU Delft, 2019.

[25] GPy. GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy, since 2012.

[26] D.J. Griffiths. *Introduction to Quantum Mechanics*. Prentice Hall, 1995. ISBN 9780131244054.

[27] Lov K Grover. A fast quantum mechanical algorithm for database search. *arXiv preprint quant-ph/9605043*, 1996.

[28] Johannes Handsteiner, Andrew S Friedman, Dominik Rauch, Jason Gallicchio, Bo Liu, Hannes Hosp, Johannes Kofler, David Bricher, Matthias Fink, Calvin Leung, et al. Cosmic Bell test: measurement settings from Milky Way stars. *Physical review letters*, 118(6):060401, 2017.

[29] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum Algorithm for Linear Systems of Equations. *Phys. Rev. Lett.*, 103:150502, Oct 2009. doi: 10.1103/PhysRevLett.103.150502. URL https://link.aps.org/doi/10.1103/PhysRevLett.103.150502.

[30] Naomichi Hatano and Masuo Suzuki. Finding exponential product formulas of higher orders. In *Quantum annealing and other optimization methods*, pages 37–68. Springer, 2005.

[31] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209, 2019.

[32] IBM Research and the IBM QX team. IBM Q Experience Documentation, 2017. URL https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/introduction.html.

[33] Jay Gambetta, Sarah Sheldon. Cramming More Power Into a Quantum Device, 2019. URL https://www.ibm.com/blogs/research/2019/03/power-quantum-device/.

[34] Rajan Jose and Seeram Ramakrishna. Materials 4.0: Materials big data enabled materials discovery. *Applied Materials Today*, 10:127–132, 2018.

[35] Muamer Kadic, Tobias Frenzel, and Martin Wegener. Mechanical metamaterials: When size matters. *Nature Physics*, 14(1):8, 2018.

[36] 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. *Nature*, 549(7671):242, 2017.

[37] Rakhesh S Kshetrimayum. A brief intro to metamaterials. *IEEE Potentials*, 23(5):44–46, 2004.

[38] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling techniques for the nystrom method. In *Artificial Intelligence and Statistics*, pages 304–311, 2009.

[39] Roderic Lakes. Foam structures with a negative poisson's ratio. *Science*, 235:1038–1041, 1987.

[40] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When Gaussian process meets big data: A review of scalable GPs. *arXiv preprint arXiv:1807.01065*, 2018.

[41] Ruoqian Liu, Abhishek Kumar, Zhengzhang Chen, Ankit Agrawal, Veera Sundararaghavan, and Alok Choudhary. A predictive machine learning approach for microstructure optimization and materials design. *Scientific reports*, 5:11551, 2015.

[42] Yanping Liu and Hong Hu. A review on auxetic structures and polymeric materials. *Scientific Research and Essays*, 5(10):1052–1063, 2010.

[43] Seth Lloyd, Masoud Mohseni, and Patrick Rebentrost. Quantum principal component analysis. *Nature Physics*, 10(9):631, 2014.

[44] Seth Lloyd, Silvano Garnerone, and Paolo Zanardi. Quantum algorithms for topological and geometric analysis of data. *Nature communications*, 7:10138, 2016.

[45] Turab Lookman, Prasanna V Balachandran, Dezhen Xue, John Hogden, and James Theiler. Statistical inference and adaptive design for materials discovery. *Current Opinion in Solid State and Materials Science*, 21(3):121–128, 2017.

[46] J.P. Mercier, G. Zambelli, and W. Kurz. *Introduction to Materials Science*. Series in applied chemistry and materials sciences. Elsevier Science, 2012. ISBN 9780080950716. URL https://books.google.nl/books?id=J32GxYdM0aMC.

[47] Bryce Meredig, Ankit Agrawal, Scott Kirklin, James E Saal, JW Doak, Alan Thompson, Kunpeng Zhang, Alok Choudhary, and Christopher Wolverton. Combinatorial screening for new materials in unconstrained composition space with machine learning. *Physical Review B*, 89(9):094104, 2014.

[48] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[49] K.P. Murphy and F. Bach. *Machine Learning: A Probabilistic Perspective*. Adaptive Computation and Machine Learning series. MIT Press, 2012. ISBN 9780262018029. URL https://books.google.nl/books?id=NZP6AQAAQBAJ.

[50] Zachary G Nicolaou and Adilson E Motter. Mechanical metamaterials with negative compressibility transitions. *Nature materials*, 11(7):608, 2012.

[51] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information, 2000.

[52] Silvia Noschese and Lothar Reichel. Inverse subspace problems with applications. *Numerical Linear Algebra with Applications*, 21(5):589–603, 2014. doi: 10.1002/nla.1914. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/nla.1914.

[53] Peter JJ O'Malley, Ryan Babbush, Ian D Kivlichan, Jonathan Romero, Jarrod R McClean, Rami Barends, Julian Kelly, Pedram Roushan, Andrew Tranter, Nan Ding, et al. Scalable quantum simulation of molecular energies. *Physical Review X*, 6(3):031007, 2016.

[54] Jayson Paulose, Anne S Meeussen, and Vincenzo Vitelli. Selective buckling via states of self-stress in topological metamaterials. *Proceedings of the National Academy of Sciences*, 112(25): 7639–7644, 2015.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[56] 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. *Nature communications*, 5:4213, 2014.

[57] John Preskill. Quantum Computing in the NISQ era and beyond. *Quantum*, 2:79, 2018.

[58] Nestor V Queipo, Raphael T Haftka, Wei Shyy, Tushar Goel, Rajkumar Vaidyanathan, and P Kevin Tucker. Surrogate-based analysis and optimization. *Progress in aerospace sciences*, 41(1):1–28, 2005.

[59] Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6(Dec):1939–1959, 2005.

[60] Ahmad Rafsanjani, Abdolhamid Akbarzadeh, and Damiano Pasini. Snapping mechanical metamaterials under tension. *Advanced Materials*, 27(39):5931–5935, 2015.

[61] Patrick Rebentrost, Masoud Mohseni, and Seth Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[62] Patrick Rebentrost, Maria Schuld, Leonard Wossnig, Francesco Petruccione, and Seth Lloyd. Quantum gradient descent and Newton's method for constrained polynomial optimization. *arXiv preprint arXiv:1612.01789*, 2016.

[63] Xin Ren, Raj Das, Phuong Tran, Tuan Duc Ngo, and Yi Min Xie. Auxetic metamaterials and structures: A review. *Smart materials and structures*, 27(2):023001, 2018.

[64] IBM Research and the IBM QX team. IBM Quantum Experience. https://quantumexperience.ng.bluemix.net, 2019. Accessed: 2019-08-10.

[65] Artur Scherer, Benoît Valiron, Siun-Chuon Mau, Scott Alexander, Eric Van den Berg, and Thomas E Chapuran. Concrete resource analysis of the quantum linear-system algorithm used to compute the electromagnetic scattering cross section of a 2d target. *Quantum Information Processing*, 16(3):60, 2017.

[66] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

[67] Vivek V Shende, Stephen S Bullock, and Igor L Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[68] Peter W Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[69] Edward Snelson and Zoubin Ghahramani. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems*, pages 1257–1264, 2006.

[70] Ilya M Sobol. Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates. *Mathematics and computers in simulation*, 55(1-3):271–280, 2001.

[71] Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574, 2009.

[72] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[73] Shusen Wang and Zhihua Zhang. Efficient algorithms and error analysis for the modified nystrom method. In *Artificial Intelligence and Statistics*, pages 996–1004, 2014.

[74] Andrew J Wathen and Shengxin Zhu. On spectral distribution of kernel matrices related to radial basis functions. *Numerical Algorithms*, 70(4):709–726, 2015.

[75] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT Press Cambridge, MA, 2006.

[76] Christopher KI Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[77] Leonard Wossnig, Zhikuan Zhao, and Anupam Prakash. Quantum linear system algorithm for dense matrices. *Physical review letters*, 120(5):050502, 2018.

[78] Kai Zhang, Ivor W Tsang, and James T Kwok. Improved nyström low-rank approximation and error analysis. In *Proceedings of the 25th international conference on Machine learning*, pages 1232–1239. ACM, 2008.

[79] Zhikuan Zhao, Jack K Fitzsimons, and Joseph F Fitzsimons. Quantum assisted Gaussian process regression. *arXiv preprint arXiv:1512.03929*, 2015.

[80] Zhikuan Zhao, Jack K Fitzsimons, Michael A Osborne, Stephen J Roberts, and Joseph F Fitzsimons. Quantum algorithms for training Gaussian Processes. *arXiv preprint arXiv:1803.10520*, 2018.

A

Fundamental concepts of Quantum computing

This appendix introduces basic concepts of quantum computing. For more details reader is referenced to sources [51] and [64].

A.1. Qubit

Qubit is the most fundamental unit of information in quantum computing. From the physics point of view qubit is a two state quantum system, with two levels $|0\rangle$ and $|1\rangle$. The peculiar brackets mean that Dirac notation is used. The two states can be also represented by vectors (computational basis vectors), as follows:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{A.1}$$

On the other hand qubit can be also considered as a mathematical object, represented by a two-dimensional complex vector space.

The complete state of a qubit can be described with just two complex coefficients α and β, as shown in Equation A.2

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{A.2}$$

The state of Qubit can be also represented as a vector pointing on the Bloch sphere, as show in Figure A.1.

A.1.1. Superposition

Superposition principle is one of the most fundamental concepts of Quantum mechanics. Superposition means that the quantum state can be a combination of multiple measurable states. For instance the state of the qubit as shown in Equation A.2 is a combination of two measurable states: $|0\rangle$ and $|1\rangle$. Upon measurement, however, the quantum state is disturbed and always collapses to a

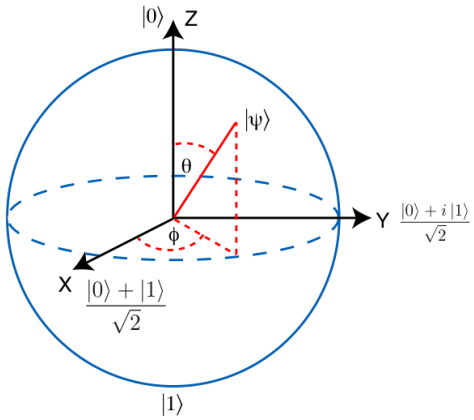


Figure A.1: Representing qubit on a Bloch sphere [64]

single measurable state with a probability given by the square of the corresponding complex coefficient. For example, probability of measuring the qubit in state $|0\rangle$ is $|\alpha^2|$.

On the side note, considering that the sum of probabilities must be 1, the sum of the squares of the complex coefficients must be 1 as well - e.g. for single qubit: $|\alpha^2| + |\beta^2| = 1$

The most counter-intuitive implication of the concept of superposition is that the state of the system can be deterministic - defined with unique state vector, but subjected to measurement it still behaves randomly - it always collapses to a measurable state with some probability.

A.2. Operators

In Quantum mechanics operators are represented with matrices. Then the action of the operator on a quantum state can be represented by multiplication of the matrix by the state vector, which results in a new state vector representing evolved system. In order to preserve the probabilities (keep it equal to 1) only unitary operators are allowed.

The most basic set of operators are Pauli operators, as listed in Equation A.3

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{A.3}$$

Another important operator is the Hadamard gate as shown in Equation A.4, which can put state in a superposition.

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{A.4}$$

For example, applying Hadamard gate on state $|0\rangle$ results in unifor superposition of $|0\rangle$ and $|1\rangle$. Therefore, before applying the Hadamard gate, the probability of measuring the qubit in state $|0\rangle$ was 100% while after putting it in superposition the probability changes to 50-50.

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 1\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\ 1\end{pmatrix} + \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ 0\end{pmatrix} \tag{A.5}$$

In quantum computing the operators are equivalent to gates, which realize the most basic operations on qubits.

A.3. Multiple qubits

With n qubits the state of the system can be expressed in 2^n dimensional space. Multiple qubits are also represented with Dirac notation e.g. $|00\rangle$, which is equivalent to tensor product of $|0\rangle \otimes |0\rangle$. Consequently, the basis vectors for the multiple qubits states can be obtained with tensor product of the single qubits basis vectors, as shown in Equation A.6.

$$|0\rangle \otimes |0\rangle = \begin{pmatrix}1\\ 0\end{pmatrix} \otimes \begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}1\\ 0\\ 0\\ 0\end{pmatrix} \tag{A.6}$$

A.3.1. Entanglement

Entanglement is one of the most strange concept of Quantum mechanics. Entangled particles interact in such way, that their state can not be described independently.

To understand it better consider famous example of Bell state shown in Equation A.7, which is a superposition of states $|00\rangle$ and $|11\rangle$.

$$|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \tag{A.7}$$

Now, if the first qubit is measured to be in state $|0\rangle$, the observer knows that the system must have collapsed to the state $|00\rangle$, which implies that the second qubit must be in state $|0\rangle$ as well. Therefore one knows the state of the second qubit without even measuring it.

A.4. Quantum circuits

Quantum algorithms can be represented with a quantum circuit model, as for example shown in Figure 2.10. The circuit consists of registers (represented by the horizontal lines) which are assembled out of qubits. The algorithm is then executed via applying series of gates (which, as discussed before are in fact operators and can be represented with matrices). In the end usually a measurement is done on some of the registers.

B

Detailed error analysis: qPCA vs Nystrom approximation

This appendix provides details on error analysis of the QGP algorithm, to support the discussion in subsection 3.5.3. First, a detailed derivation of the error bounds is provided in Section B.1. Next, Section B.2 presents the data-sets used in the primary numerical tests verifying the error bounds, as discussed in subsection 3.5.3. Finally, Section B.3 supplements the discussion by providing results of the numerical error verification using GP systems built using data-sets from four different benchmarking functions.

B.1. Derivation of the error bounds due to the eigenspectrum cut-off

The error of approximating matrix A with $\widetilde{A}$ is quantified in terms of Frobenius norm, following the approach from literature [78], as shown in Equation B.1:

$$\epsilon = \left\| A - \widetilde{A} \right\|_F \tag{B.1}$$

The Frobenius norm is defined as follows:

$$\|A\|_F = \sqrt{Tr(AA^H)} \tag{B.2}$$

Considering that in QGP we only deal with Hermitian matrices (which is a requirement on the QGP input) the trace can be also calculated as a sum of eigenvalues of the matrix A and the Frobenius norm can be rewritten as[1]:

$$\|A\|_F = \sqrt{Tr(A^2)} = \sqrt{\sum_{i=1}^{n} \lambda_i^2} \tag{B.3}$$

[1] The Frobenius norm of a matrix is often defined in terms of singular values. It is sometimes confusing. For Hermitian and symmetric matrices the singular values are equal to the absolute value of the eigenvalues.

To obtain the error of approximating matrix with QPE i.e. $\epsilon = \left\| A - \widetilde{A} \right\|_F$, following steps were taken:

- Both matrices share the same eigenbasis U, therefore they can be rewritten in a diagonal form as:

$$A = U\Lambda U^{-1} \qquad \widetilde{A} = U\widetilde{\Lambda}U^{-1} \tag{B.4}$$

  where, $\widetilde{\Lambda}$ is a diagonal matrix with entries approximated by QPE

- Both A and $\widetilde{A}$ are hermitian, and a sum of two Hermitian matrices is also a Hermitian, therefore:

$$\left(A - \widetilde{A}\right) = \left(A - \widetilde{A}\right)^H \tag{B.5}$$

- The trace term in numerator can be rewritten as follows:

$$Tr\left(\left(A - \widetilde{A}\right)\left(A - \widetilde{A}\right)^H\right) = Tr\left(\left(A - \widetilde{A}\right)^2\right) = Tr\left(\left(U\Lambda U^{-1} - U\widetilde{\Lambda}U^{-1}\right)^2\right) = \tag{B.6}$$

$$= Tr\left(U\left(\Lambda - \widetilde{\Lambda}\right)^2 U^{-1}\right) = \sum diag\left(\Lambda - \widetilde{\Lambda}\right)^2 \tag{B.7}$$

  The above derivation proves that comparing the exact and approximate matrices in terms of their entries (via Frobenius norm) is equivalent to comparing their eigenspectra.

Considering the entries of the two diagonal matrices shown in Equation B.8, it becomes apparent that there will be two sources of errors: due to approximating the eigenvalues with finite precision (a round-off error), and truncation of eigenspectrum due to the QPE resolution threshold, which constitute the entries of the resultant matrix $\Lambda - \widetilde{\Lambda}$ shown in Equation B.9.

$$\Lambda = \begin{bmatrix} \lambda_1 & & & & & \\ & \lambda_2 & & & & \\ & & .. & & & \\ & & & .. & & \\ & & & & .. & \\ & & & & & \lambda_n \end{bmatrix} \qquad \widetilde{\Lambda} = \begin{bmatrix} \widetilde{\lambda_1} & & & & & \\ & \widetilde{\lambda_2} & & & & \\ & & .. & & & \\ & & & \widetilde{\lambda_m} & & \\ & & & & 0 & \\ & & & & & 0 \end{bmatrix} \tag{B.8}$$

$$\Lambda - \widetilde{\Lambda} = \begin{bmatrix} \epsilon_1^q & & & & & & \\ & \epsilon_2^q & & & & & \\ & & .. & & & & \\ & & & \epsilon_m^q & & & \\ & & & & \lambda_{m+1} & & \\ & & & & & .. & \\ & & & & & & \lambda_n \end{bmatrix} \tag{B.9}$$

In order to gain insight into behavior of those two errors, this resultant matrix is rewritten as a sum of two different diagonal matrices D_q and D_t that correspond to the two sources of errors, as shown in Equation B.10, B.11 and B.12.

$$\Lambda - \widetilde{\Lambda} = D_q + D_t \tag{B.10}$$

$$D_t = \begin{bmatrix} 0 & & & & & & \\ & 0 & & & & & \\ & & .. & & & & \\ & & & 0 & & & \\ & & & & \lambda_{m+1} & & \\ & & & & & .. & \\ & & & & & & \lambda_n \end{bmatrix} \tag{B.11}$$

$$D_q = \begin{bmatrix} \epsilon_1^q & & & & & & \\ & \epsilon_2^q & & & & & \\ & & .. & & & & \\ & & & \epsilon_m^q & & & \\ & & & & 0 & & \\ & & & & & .. & \\ & & & & & & 0 \end{bmatrix} \tag{B.12}$$

To decouple the two errors, the triangular inequality can be used:

$$\|A+B\|_F \leqslant \|A\|_F + \|B\|_F \tag{B.13}$$

Then the overall error can be split on two components:

$$\left\|A - \widetilde{A}\right\|_F = \left\|D_q + D_t\right\|_F \leqslant \left\|D_q\right\|_F + \|D_t\|_F \tag{B.14}$$

In this way, the two sources of error can be separated, and analyzed individually.

Error due to QPE discretization

The error of approximating the resolved eigenvalues with QPE (the entries of D_q) corresponds to the QPE resolution[51]. It can be shown, that given a k-qubit eigenregister, the eigenvalues can be approximated within accuracy given by Equation B.15 with probability $1-\delta$.

$$\epsilon < \left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})} - 1}\right) \tag{B.15}$$

Note that this error is always positive, which is an effect of the assumed discretization scheme (for more details see [51]). Considering that m eigenvalues were resolved (i.e. approximated as non-zero), the error due to the QPE resolution can be expressed as:

$$\|D_q\|_F = \sqrt{Tr(D_q)} = \sqrt{\sum_{i=1}^{m} \epsilon_i^2} \tag{B.16}$$

To evaluate the sum in Equation B.16, a mean error could be used for simplicity::

$$\bar{\epsilon} \approx \frac{1}{2}\left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})}-1}\right) \tag{B.17}$$

Using Equation B.17, the expression B.16 for the complete error due to QPE can be approximated as shown in Equation B.18:

$$\|D_q\|_F = \sqrt{\sum_{i=1}^{m} \epsilon_i^2} \approx \sqrt{m\bar{\epsilon}^2} = \sqrt{m\left[\frac{1}{2}\left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})}-1}\right)\right]^2} = \frac{\sqrt{m}}{2}\left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})}-1}\right) \tag{B.18}$$

Equation B.18 provides only an estimate of the error, but not a strictly rigorous bound, which is a consequence of assuming mean error in Equation B.16. In order to bound the error $\|D_q\|_F$, the worst case scenario needs to be considered, which is that all the errors ϵ_i in the sum in Equation B.16 take maximum allowed value, which can be determined from Equation B.15. This results in following bound on error $\|D_q\|_F$:

$$\|D_q\|_F = \sqrt{\sum_{i=1}^{m} \epsilon_i^2} \leq \sqrt{m\epsilon_{max}^2} = \sqrt{m\left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})}-1}\right)^2} = \sqrt{m}\left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})}-1}\right) \tag{B.19}$$

Error due to eigenspectrum cut-off

The second error component – the one due to the eigenspectrum cut-off (D_t) can be estimated in the same way as in case of low rank approximation obtained with PCA [49], which relies on the same principle of constructing rank-m approximation using first m eigenmodes. The error of truncation can be then estimated as:

$$\|A - \widetilde{A}_{rank-m}\|_F \approx \sqrt{\lambda_{m+1}} \tag{B.20}$$

Knowing that the minimum resolved non-zero eigenvalue is : $\lambda_{min} = \left(\frac{\lambda_{max}}{2^k-1}\right)$, therefore the largest eigenvalue that could not be resolved is bound by: $\lambda_{m+1} < \frac{\lambda_{max}}{2^k-1}$. Substituting this assumption into Equation B.20, the error due to eigenspectrum cut-off is bounded by:

$$\|D_t\|_F \approx \lambda_{m+1} \leqslant \left(\frac{\lambda_{max}}{2^k-1}\right) \tag{B.21}$$

Notice that the probability of success $(1-\delta)$ is not taken into account in this case.

B.1.1. Overall error

Custom bound

The overall error can is obtained by summing the partial errors originating from the two sources: QPE round-off and truncating the eigenspectrum, according to Equation B.14. Substituting terms from Equation B.19 and Equation B.21.

$$\left\|A-\widetilde{A}\right\|_{custom} \leq \left\|D_q\right\|_F + \|D_t\|_F \leq \lambda_{max}\left(\frac{1}{2^k-1}+\frac{\sqrt{m}}{2^{k-log(2+\frac{1}{2\delta})}-2}\right) \tag{B.22}$$

The default QPE bound

Alternatively, the error bound could be derived more rigorously by treating all the errors as inaccuracy of the QPE. In this case, following the same approach as in Equation B.19 results in:

$$\left\|A-\widetilde{A}\right\|_{QPE} \leq \frac{\sqrt{n}}{2}\left(\frac{\lambda_{max}}{2^{k-log(2+\frac{1}{2\delta})}-1}\right) \tag{B.23}$$

This approach, however greatly overestimates the errors on the truncated side of the eigenspectrum, as the errors estimated in such way are few orders of magnitude larger than the truncated eigenvalues.

Relating m to k

The parameter m in the derived bounds Equation B.22 is unknown, and because the diagonalization happens in a quantum state, the user does not have direct access or control over this parameter. On the other hand, this parameter can be related the resolution parameter k.

For the covariance matrices, the distribution of eigenvalues is bound by an exponential decay [74] , such that $\lambda_{m+1} \approx O(\lambda_m R) = O(\lambda_m R^m)$, where the parameter $R < 1$. Following this assumption, the minimum resolvable eigenvalue can be related to the parameter m as follows:

$$\frac{\lambda_{max}}{2^k-1} \approx O(\lambda_{max} R^m) \tag{B.24}$$

Therefore: $\left(\frac{1}{R}\right)^{-m} \approx O\left(2^{-k}\right)$, which after rearranging results in:

$$m \approx O\left(\frac{-k}{log_2 R}\right) \tag{B.25}$$

This relationship was validated with numerical tests, which indicated that in practice the eigenvalues decay follows rather a super-exponential decay, as can be seen in Figure B.1.

Therefore in practical case, m is more likely to be related to k via:

$$m \approx O\left(\frac{-k}{log_2 R}\right)^{\frac{1}{b}} \tag{B.26}$$

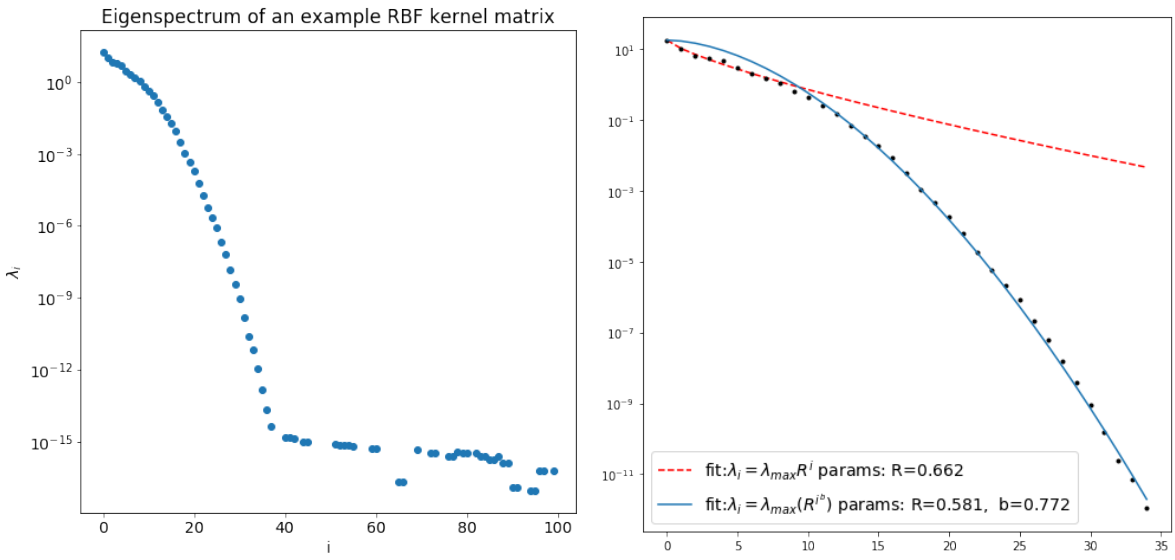


Figure B.1: A numerical example illustrating an eigenspectrum of an example covariance matrix (left plot), and fit against the assumed theoretical exponential decay according to $\lambda_{m+1} \approx O(\lambda_m R) = O(\lambda_m R^m)$ in the right plot. Note that the fit uses only part of eigenspectrum (as for eigenvalues above $i = 40$ the eigenspectrum decay reaches a plateau due to round off errors).

B.1.2. Comparison with the Nystrom approximation and classical PCA

For the standard Nystrom approximation constructed by random deleting rows and columns according to a uniform distribution, and furthermore, assuming that the number of sampled columns is the same as the rank of the approximation matrix m, a following upper bound was found by Kumar et al. [38]:

$$\|A - \tilde{A}_m\|_F \leqslant \|A - A_m\|_F + 2\sqrt{2}\left[\left(\frac{n}{m}\sum_{i\in D(m)} A_{ii}\right)\left(\sqrt{n\sum_{i=1}^{n} A_{ii}^2} + \sqrt{log\left(\frac{2}{\delta}\right)\left(1-\frac{m}{n}\right)}(max(nA_{ii}))\right)\right]^{\frac{1}{2}} \tag{B.27}$$

Then applying several inequalities (for details see [38]), the bound can be simplified to following form:

$$\|A - \tilde{A}_m\|_F \leqslant \|A - A_m\|_F + \epsilon \cdot max(nA_{ii}) \tag{B.28}$$

Where the ϵ is defined in following way:

$$\epsilon = \left[64\frac{m}{l}\left(1 + \sqrt{log\left(\frac{2}{\delta}\right)\left(1-\frac{l}{n}\right)}\right)^2\right]^{\frac{1}{4}} \tag{B.29}$$

Parameter l is the size of the approximation matrix (number of sampled columns and rows). As this analysis considers the most basic case, it is assumed that $l = m$. Here, the parameter δ corresponds the probability of success, specified as: $1 - \delta$.

Term $max(G_{ii})$ is bounded by the maximum eigenvalue: $max(G_{ii}) \leqslant \lambda_{max}$. With all those assumptions, the bound can be rewritten into following form:

$$\|A-\tilde{A}_m\|_F \leqslant \|A-A_m\|_F + 2\sqrt{2}n\lambda_{max}\sqrt{1+\sqrt{log\left(\frac{2}{\delta}\right)\left(1-\frac{m}{n}\right)}} \tag{B.30}$$

Term $\|A-A_m\|_F$ stands for the best achievable rank m approximation, which happens to be the one obtained via PCA[49]. Therefore this term can be approximated as: $\|A-A_k\|_F \approx \lambda_{m+1}$, which allows to rewrite the upper bound on Nystrom approximation as follows:

$$\|A-\tilde{A}_m\|_F \leqslant \lambda_{max}\left(\frac{1}{2^k-1} + 2\sqrt{2}n\sqrt{1+\sqrt{log\left(\frac{2}{\delta}\right)\left(1-\frac{m}{n}\right)}}\right) \tag{B.31}$$

The bound however greatly overestimates the error due to a number of inequalities used throughout the derivation.

Whang and Zhang [73] provide a lower error bound, which can be written in following form (after assuming that the number of column is equal to the rank of the approximating matrix, which is the case considered in the context of Sparse GPs):

$$\|A-\tilde{A}_m\|_F \leqslant \|A-A_m\|_F\left(\sqrt{1+\frac{n+m}{m}}\right) = \frac{\lambda_{max}}{2^k-1}\left(\sqrt{1+\frac{n+m}{m}}\right) \tag{B.32}$$

All the bounds derived for QGP and Nystrom approximation are compared in Table B.1.

Table B.1: Summary of error bounds for the two different low-rank approximation methods

Approximation method	Upper bound	Lower bound
Nystrom	$\lambda_{max}\left(\frac{1}{2^k-1} + 2\sqrt{2}n\sqrt{1+\sqrt{log\left(\frac{2}{\delta}\right)\left(1-\frac{m}{n}\right)}}\right)$	$\frac{\lambda_{max}}{2^k-1}\left(\sqrt{1+\frac{n+m}{m}}\right)$
QPE	$\lambda_{max}\left(\frac{1}{2^k-1} + \frac{\sqrt{m}}{2^{k-log(2+\frac{1}{2\delta})}-2}\right)$	$\frac{\lambda_{max}}{2^k-1} \approx \epsilon_{PCA}$

B.2. Test matrices data

This section presents the five datasets that were used for numerical verification of error bounds of Nystrom and qPCA low-rank approximations, discussed in subsection 3.5.3. The datasets, shown in Figure B.2 were generated from stocks prices, which provide with an interesting test case as they exhibit a highly non-linear relationships, with a number of concurrent trends (including long-term nearly-linear trends, intermediate term oscillations, short-term movements and random noise).

B.3. Numerical error verification on benchmarking functions

The error bounds discussed in subsection 3.5.3 were also tested on an ensemble of functions typically used for benchmarking of optimization algorithms. For this purpose, four different functions were selected, to provide a range of typical problems faced in optimization tasks. The first two examples - the Ackley and Rastrigin functions, shown in Figure B.3a and B.4a respectively, challenge the optimization algorithms with a periodic landscape, resulting in multiple local minima which often

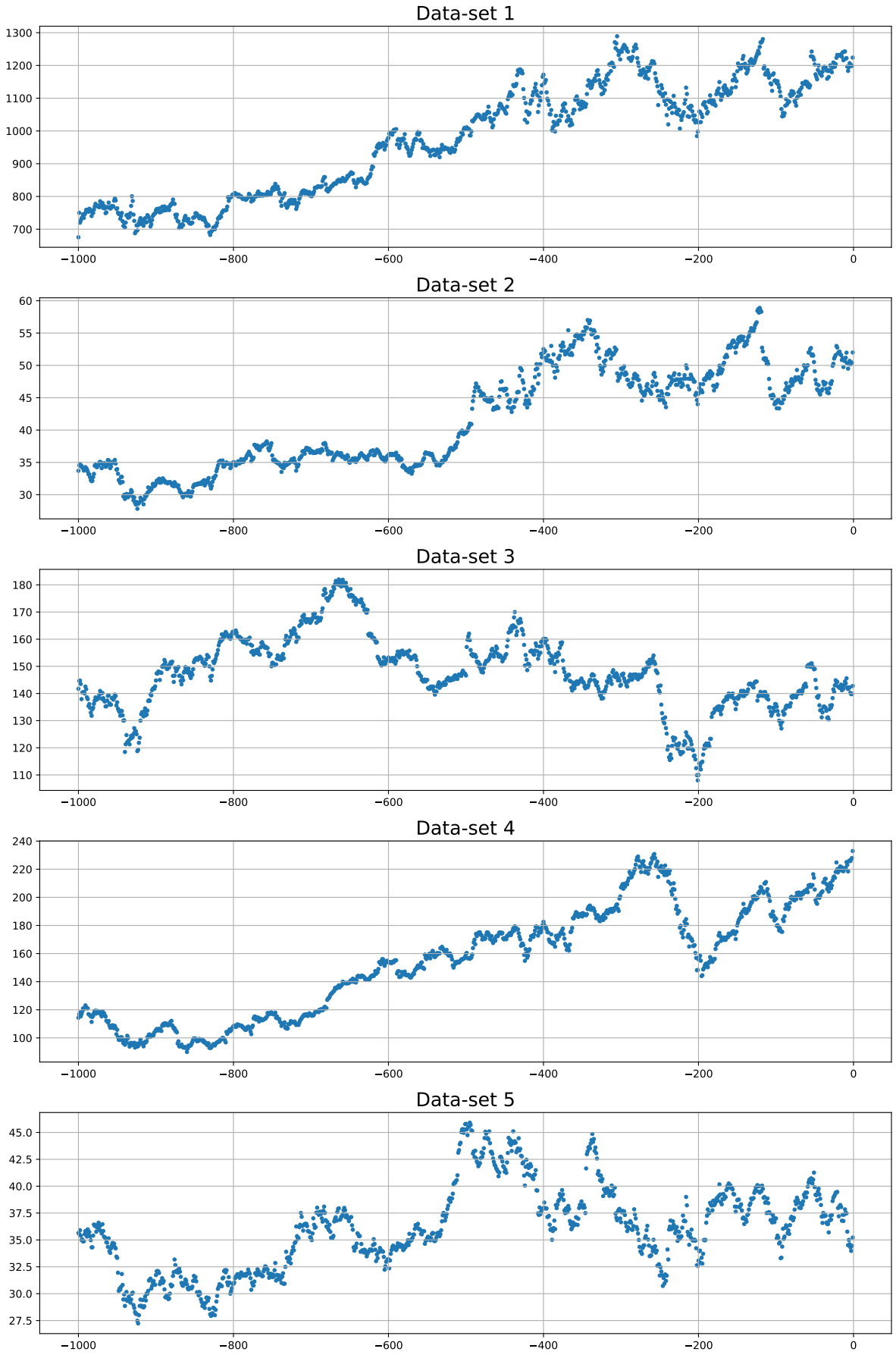


Figure B.2: Five datasets used in numerical verification o the error bounds.

tend to hinder the optimization algorithms from finding the global minimum. The other two cases, namely - Rosenbrock and Goldstein functions shown inFigure B.5a and B.6a, are non-periodic examples, with wide range of magnitudes and large variations in gradient, which make finding the global minimum non-trivial.

B.3.1. Approach

The numerical errors and error bounds were obtained following similar approach to that discussed in subsection 3.5.3. First, each benchmarking function was sampled with 1000 random points, which allowed to construct a GP model (with the aim of inferring the underlying function) and the corresponding covariance matrix, which provided the test input. The test matrices were then approximated with both QPE and Nystrom method for range of parameters k and m, and the resulting error was quantified by means of Frobenius norm between the approximate and exact matrix.

While building the GP models, however, it turned out that despite the relatively large training sets (1000 points), the GP predictions obtained with standard techniques were rather mediocre. Therefore to improve the performance, more advanced techniques had to be employed - specifically by using non-default kernel functions, which were selected using the algorithm for evolutionary construction of kernel functions, discussed in Appendix C. The found optimal kernel functions are listed in Table B.2.

For Ackley and Rastrigin functions composite kernels were used (consisting of more than one component, construction process for this case is illustrated in Appendix C). For the remaining two functions, a reasonable fit could be obtained with a single component kernel, which was selected by the algorithm out of ensemble of base kernel functions (RBF, periodic, exponential, linear, bias) to provide the best fit.

Table B.2: Optimized kernel functions used for improving fit of GP models on the benchmarking functions

Function	Kernel function
Ackley	periodic × exponential
Rastrigin	$\text{periodic}_1 + \text{periodic}_2$
Rosenbrock	periodic
Goldstein	RBF

Remark 7 *Using other than RBF kernel functions extends the scope of the test from subsection 3.5.3, and allows for generalizing the results to less-trivial examples, representative for the real-life problems.*

B.3.2. Results

The Ackley function was found to be the most challenging case, which reflects on the numerical results, shown in Figure B.3b. The composite kernel used for the GP model in this dataset resulted in an peculiar eigenspectrum, with a remarkably dominant first eigenmode with an eigenvalue that is three orders of magnitude larger compared to the next largest eigenvalue. This means that the first eigenmode alone explains significant amount of dependencies in the data, and consequently using merely the first eigenmode already gives an exceptionally good first order approximation. Those predictions confirm in the numerical results shown in Figure B.3b, which indicate on exceptionally low error observed for $m = 1$ and the corresponding $k = 1$.

Another consequence is great overestimation in the approximate PCA error, which serves as a lower bound for the QPE. As shown in the upper right plot in Figure B.3b, this lower bound overestimates the actual error by up to two orders of magnitude. This lower bound however, is only an estimate of the PCA error, which relies on two assumptions:

1. The PCA error is in order of magnitude of the largest eigenvalue that does not contribute to the approximation (i.e. for the PCA approximation consisting of m largest eigenvalues, the estimated error is $\tilde{O}(\lambda_{m+1})$)
2. The subsequent eigenvalue is comparable to the resolution limit of QPE with given k, bounded by the smallest resolvable eigenvalue $\frac{\lambda_{max}}{2^k-1}$.

This yields overall estimated PCA error of $\widetilde{O}\left(\frac{\lambda_{max}}{2^k-1}\right)$. Due to the exceptionally large difference between the first eigenvalues, the second eigenmode is not $\widetilde{O}\left(\frac{\lambda_{max}}{2^1-1}\right)$, but rather $\widetilde{O}\left(\frac{\lambda_{max}}{2^9-1}\right)$, which can be seen from the lower right plot (m vs. k), which shows that number of resolved eigenmodes m starts increasing above 1 only for $k = 9$.

Remark 8 *As the QPE is dependent on k and Nystrom method on m, the corresponding QPE and Nystrom errors can be directly related only to k and m respectively. However, to enable a direct comparison between the two methods, i.e. conversion of QPE errors in respect to m and Nystrom to k, a relationship between k and m needs to be found, to relate those two parameters. This is achieved by interpolating the numerical k-m data (shown in each plot) that was obtained during calculation of the QPE errors by counting the number of the resolved eigenmodes m for each tested k. Those approximated k-m relationships, however, often show a number of jumps and discontinuities, which lead to abnormal artifacts in the error plots (e.g. vertical lines, jumps).*

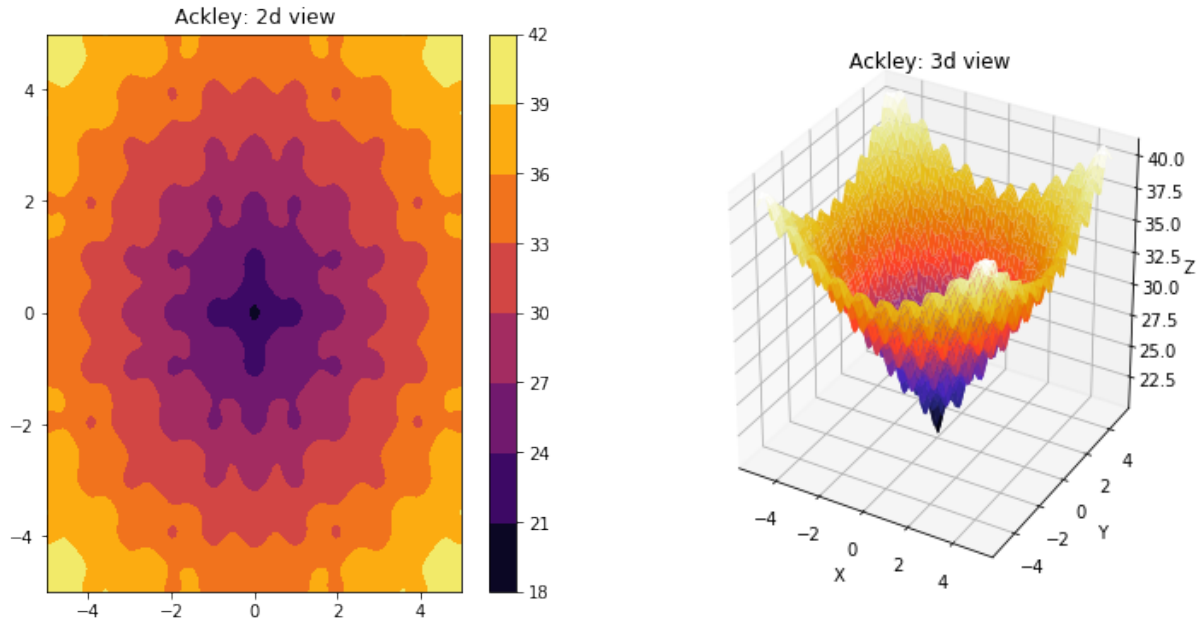
For $k > 10$, and corresponding $m > 10$, the higher order eigenmodes become included in the approximation. As those eigenmodes follow a more consistent decay (i.e. without such dramatic changes in magnitude between subsequent eigenvalues), the estimated bounds provide a more reliable prediction. Note that despite the anomalies, for all values of k and m the numerical errors of the classical the Nystrom method are larger then the QPE approximation.

B.3.3. Conclusion

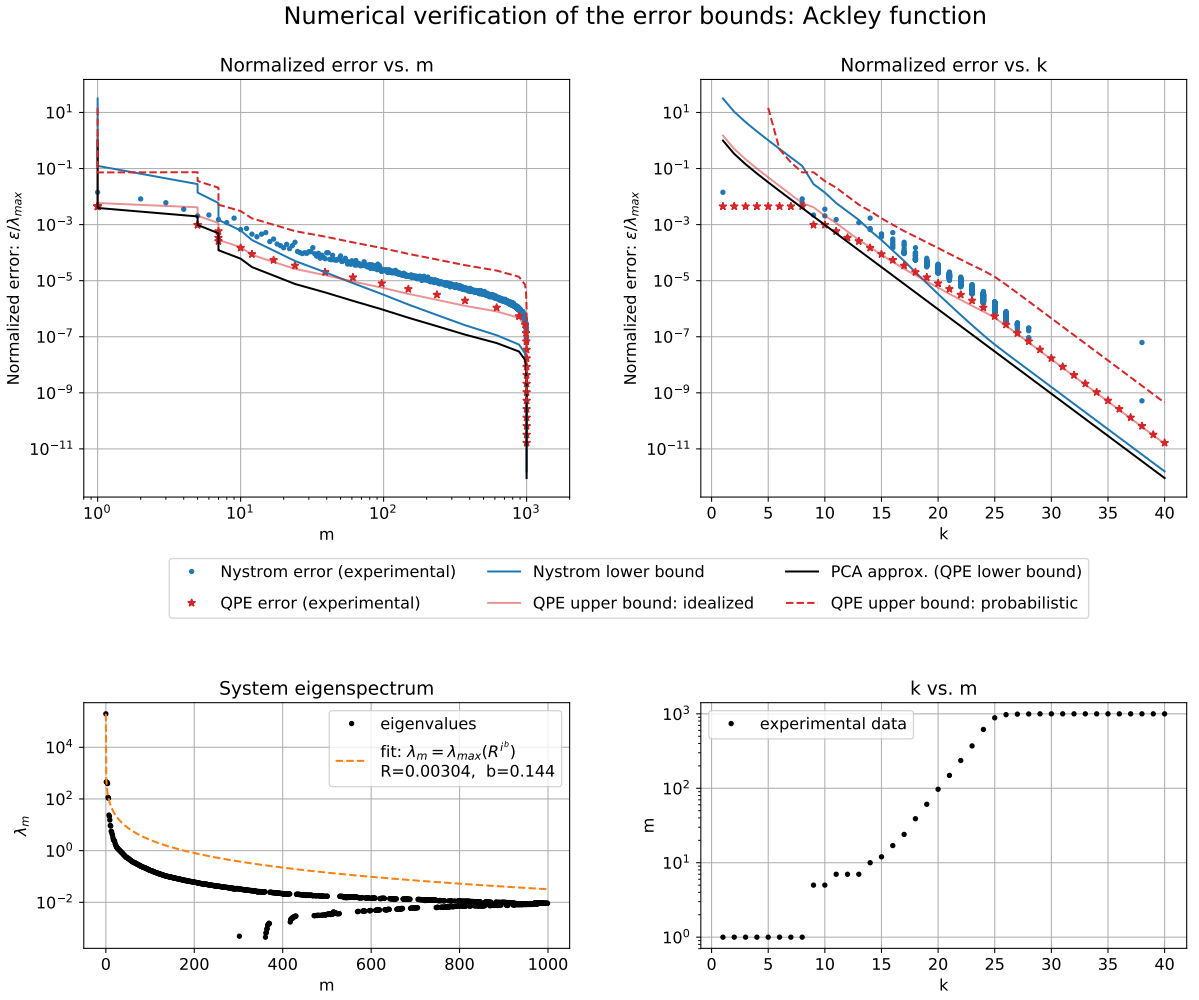
For the three remaining functions no particular anomalies were found, and the results show consistency with the conclusions drawn in subsection 3.5.3 i.e.:

- Numerical results indicate that the Nystrom method provides with lower accuracy compared to QPE, despite the theoretical lower bound of Nystrom method laying below the upper bounds of QPE (thus theoretically allowing for outperforming the QPE). The only exception is for QPE with low k settings (roughly $k < 5$).
- The probabilistic upper bound for QPE error was found to hold in all cases.
- The idealized QPE bound was found to provide a good estimate of the actual QGP error.

In general the error bounds were found to hold well, except for a one particular case, where an particularly large variation in magnitudes of the first few eigenvalues violated the assumptions underlying behind the expression for the estimated lower bound, in which case the bound overestimated the actual error by few orders of magnitude. Most importantly, the tests extended the scope of primary tests by providing variety of different non-trivial conditions (various composite kernel functions, different eigenspectra), supporting the generality of the conclusions drawn in subsection 3.5.3.

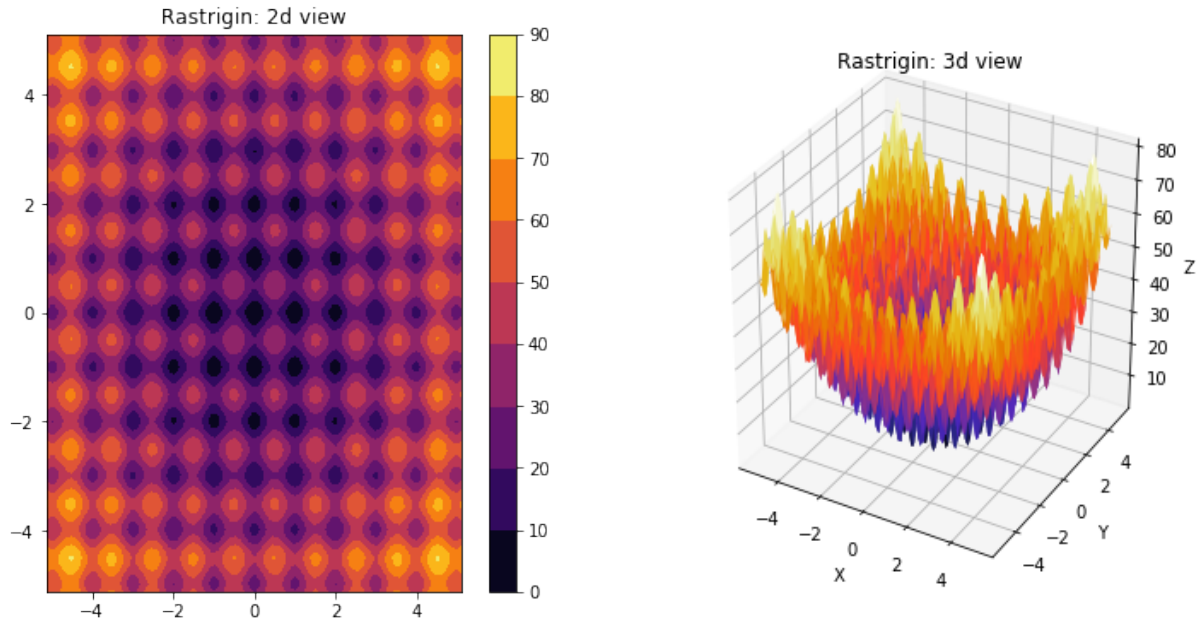


(a) Ackley function

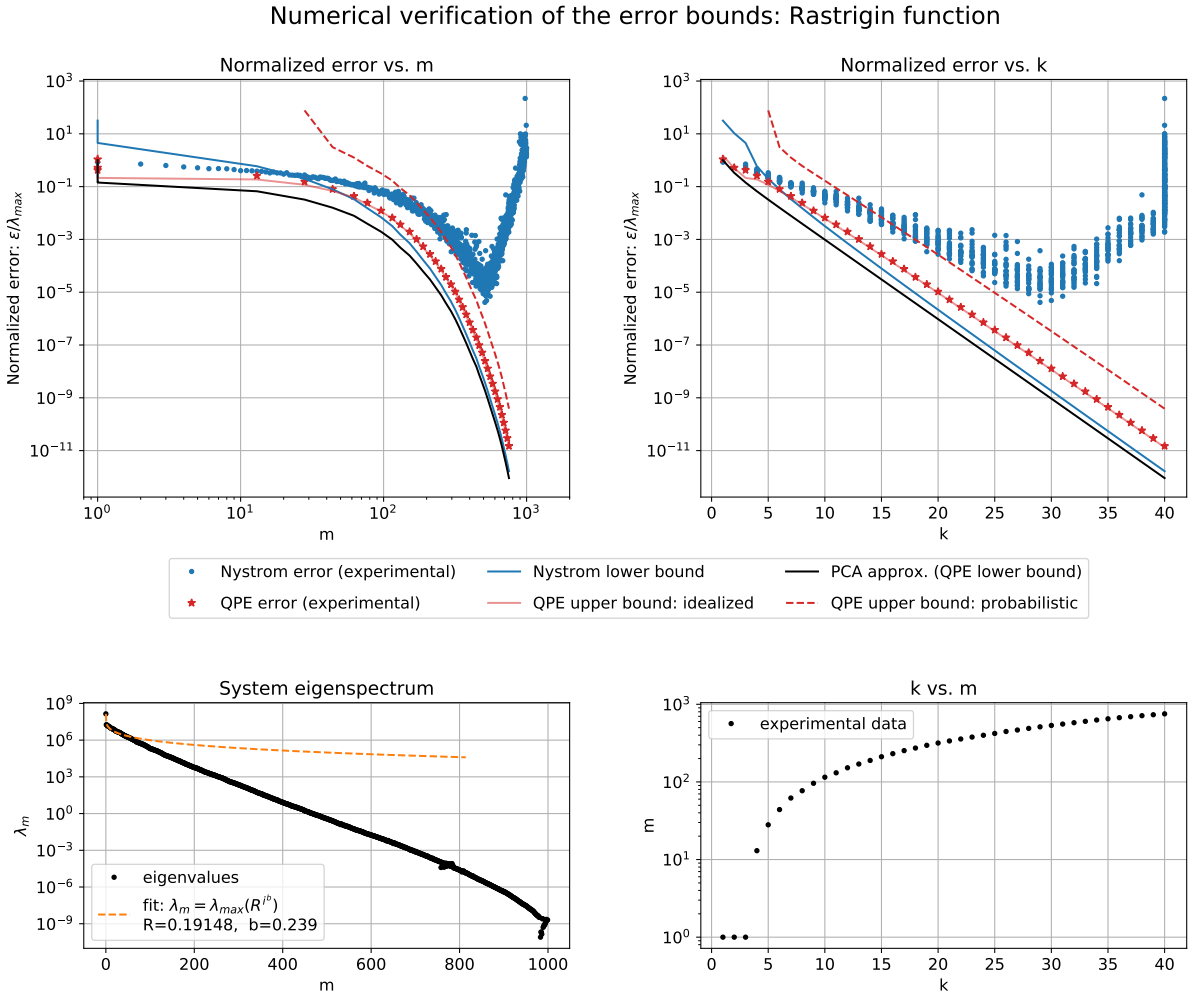


(b) Results of the numerical verification of the errors bounds, tested on GP model inferring the Ackley function using 1000 training points.

Figure B.3: Error verification on ensemble of benchmarking functions: Ackley function

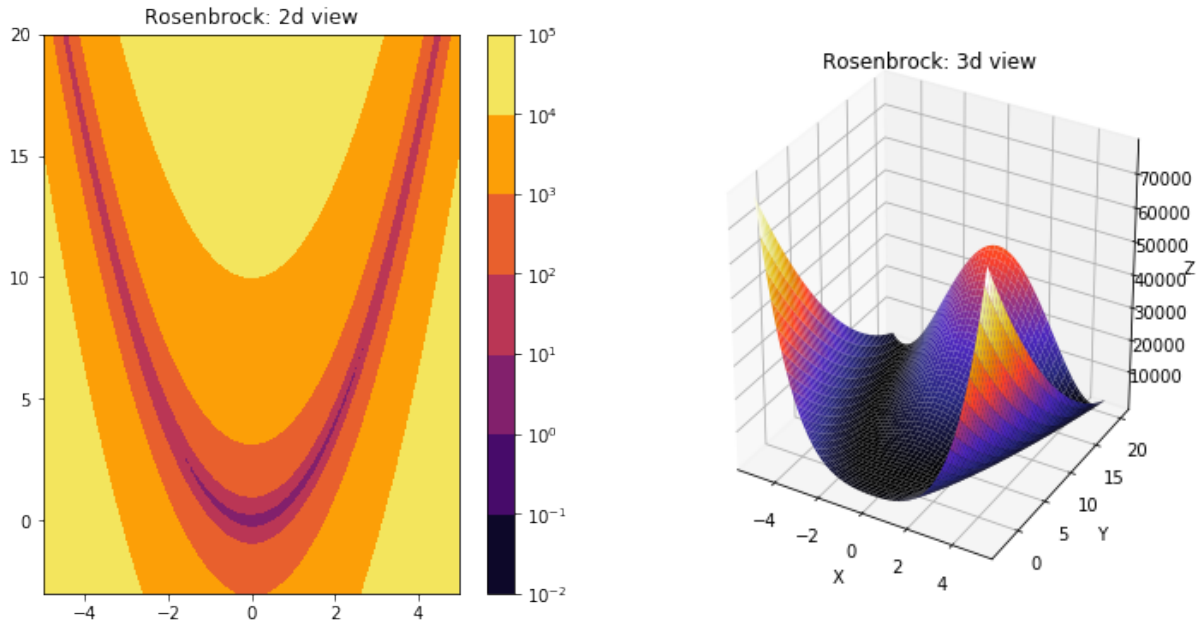


(a) Rastrigin function

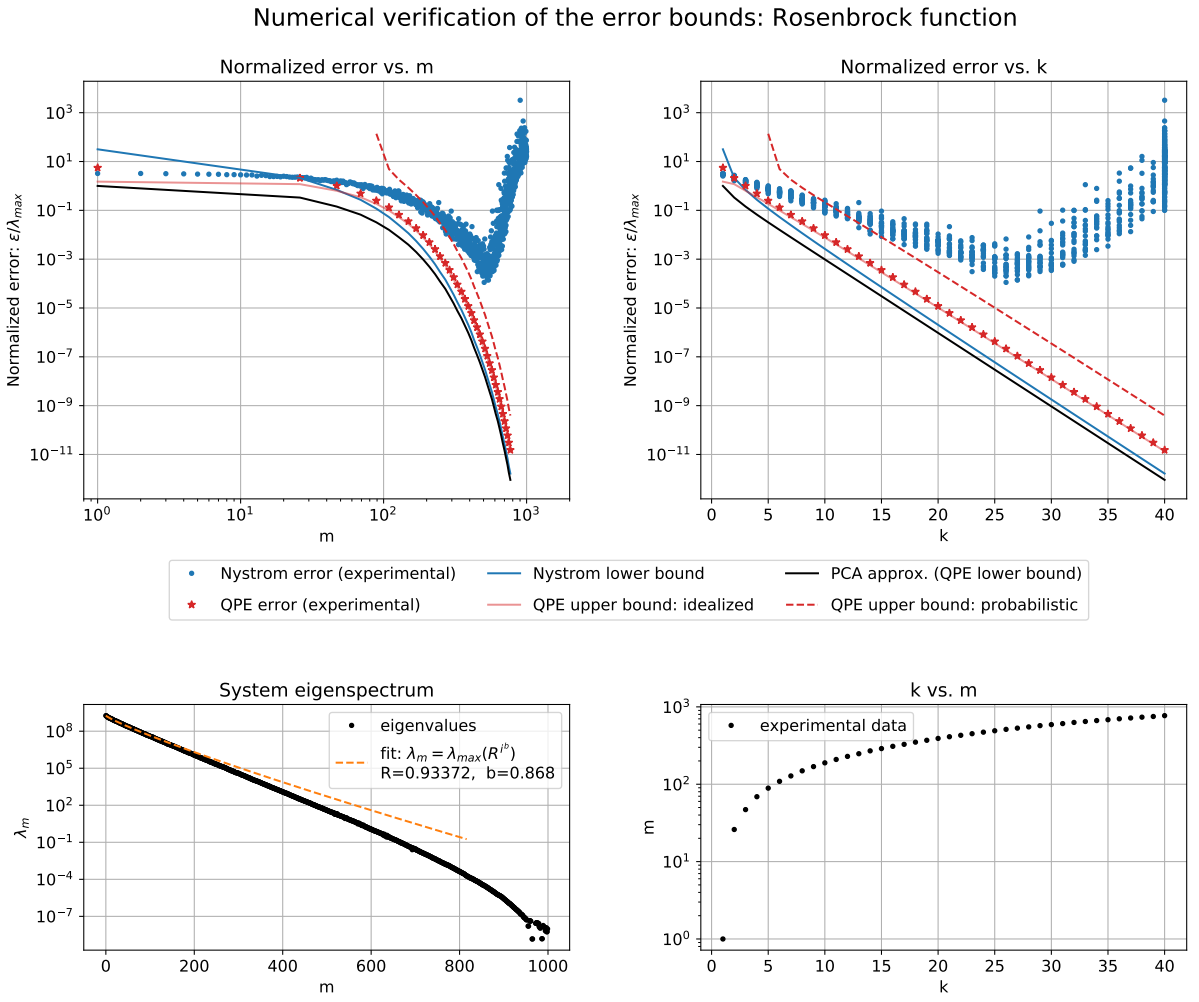


(b) Results of the numerical verification of the errors bounds, tested on GP model inferring the Rastrigin function using 1000 training points.

Figure B.4: Error verification on ensemble of benchmarking functions: Rastrigin function

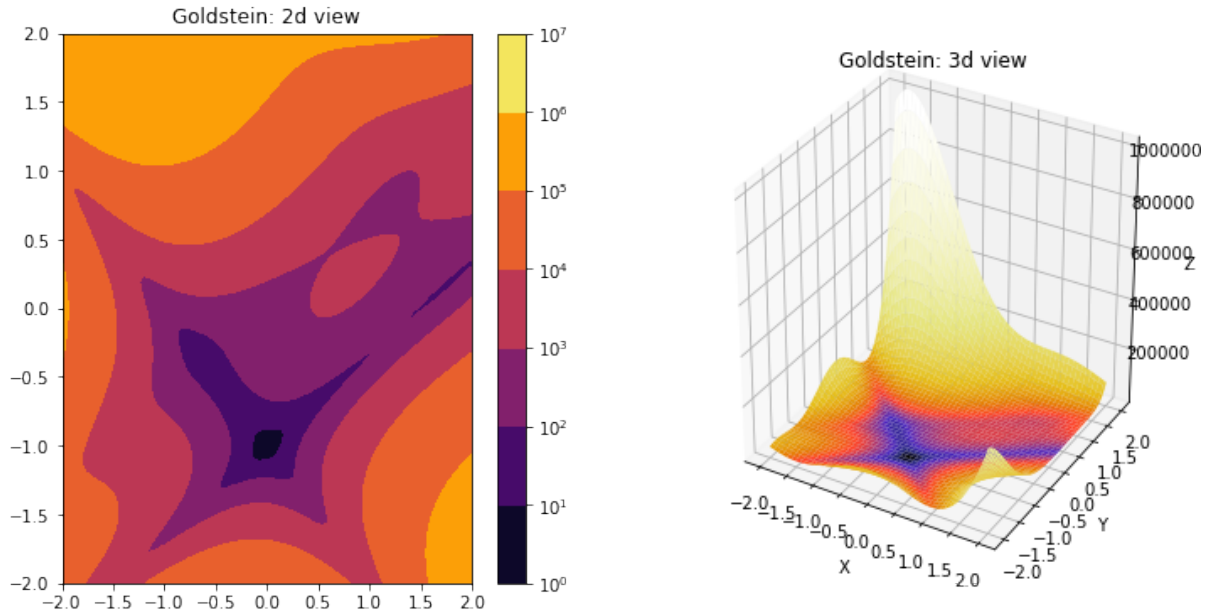


(a) Rosenbrock function

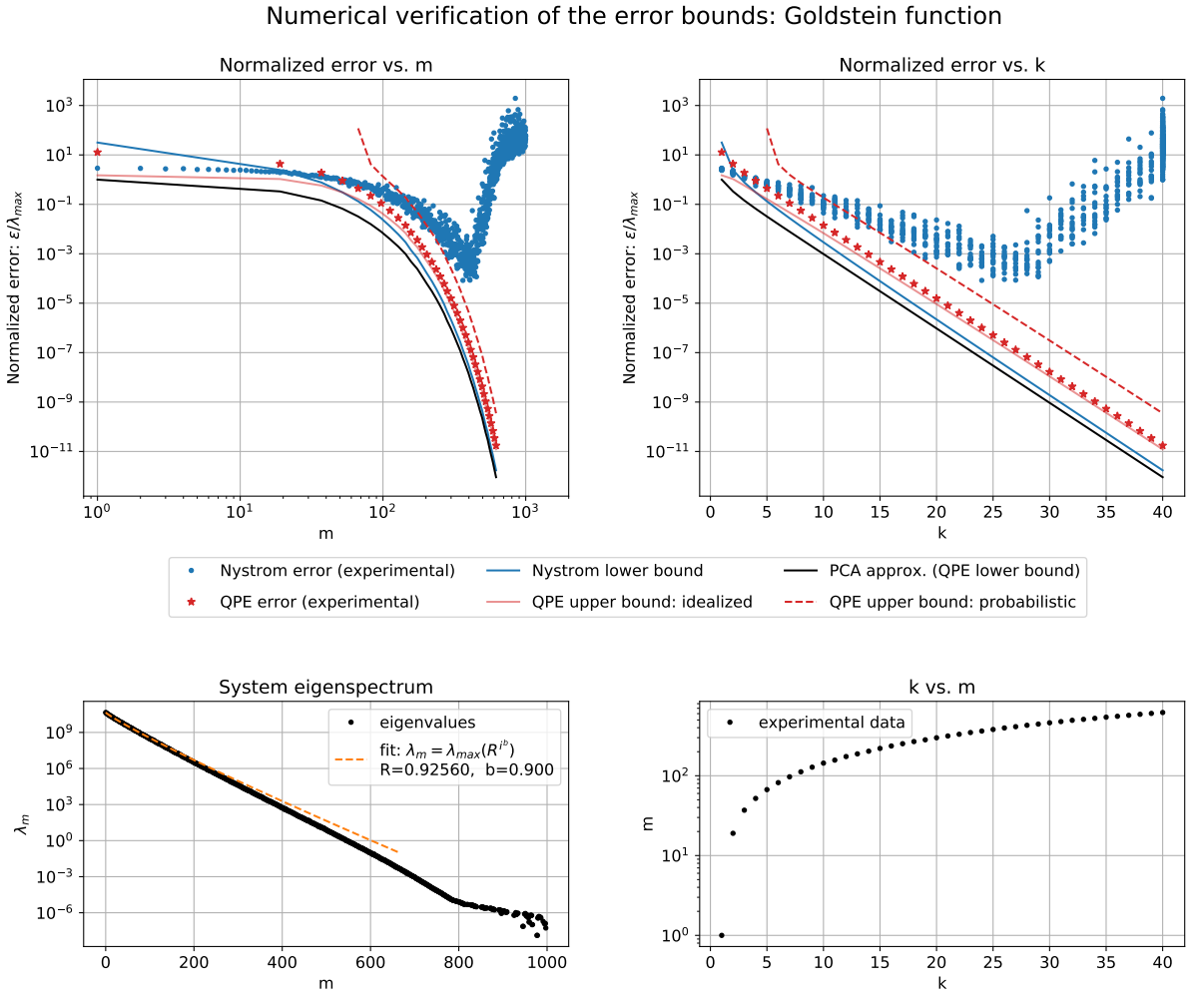


(b) Results of the numerical verification of the errors bounds, tested on GP model inferring the Rosenbrock function using 1000 training points

Figure B.5: Error verification on ensemble of benchmarking functions: Rosenbrock function



(a) Goldstein function



(b) Results of the numerical verification of the errors bounds, tested on GP model inferring the Goldstein function using 1000 training points

Figure B.6: Error verification on ensemble of benchmarking functions: Goldstein function

C

Evolutionary construction of composite kernels

In practical problems, data oftentimes exhibits elements of certain structures, follows simultaneously several different trends. Accounting for this structure, however, allows to create better models. In case of GPs, the data-structure can be accounted for by selecting the covariance function. In cases when the data follows a superposition of multiple trends, a particularly effective are the composite kernel functions, which essentially are built out of several basic kernel function, where each component corresponds to one trend in the data.

In practice oftentimes it is difficult to accurately identify the functions governing the data, which results in difficulties in constructing a suitable kernel function. While in 1- and 2-dimensional cases, this issue might be tackled with simple trial and error, in high-dimensional cases it is not a feasible option. For this reason, Douvenaud et al.[20] proposed a method that automates the process of search for a suitable composite kernel function. The method relies on building an expression S out of several base kernel functions K_B (such as linear, periodic, square exponential, rational quadratic), by adding and multiplying them (e.g: $S = K_{B1} + K_{B2} \times K_{B2}$...). The expression is then evolved by either: 1) replacing components of the expression ($K_B i$); or 2) extending the expression ($S + K_{Bi}$ or $S \times K_{Bi}$), where the choice is governed by simple greedy optimization that minimizes the log marginal likelihood of fit.

This concept was implemented during this research project, with the primary intention to improve fit of the GP models built on the benchmarking functions, which was done in context of providing a more realistic case for numerical verification of the error bounds discussed in subsection 3.5.3.

The approach also appears to tackle well the problem of optimization of the hyper-parameters. Search for minimum in the high-dimensional space of hyper-parameters of composite kernel is prone to being stuck in local minima. In practice, it was found that fitting all the parameters at the same time often results in domination of just single component of the composite kernel function, which effectively defies its purpose [1]. Instead, by fitting parameters for only one component at a time can avoid those problems, as the fitted component effectively fits to the residual not explained by the remaining (fixed) components of the kernel function.

[1] Fitting the hyperparameters most often relies on a log of marginal likelihood as an objective function, which intrinsically penalizes for model complexity. Therefore, in case of composite kernels this objective function tends to promote the simplest model (thus a single dominating kernel component)

C.1. Expression optimization in small datasets

Use of composite kernel functions is particularly powerful in limited datasets. The capabilities of this method are illustrated with an example of inferring the Rastrigin function from just 25 randomly selected data-points, shown in Figure C.1. In this case the expression was limited to two terms, assembled using only expansion step (i.e. without replacement). As shown in the first iteration, the one-component kernel is fitted such that the global trend is approximated. In second iteration, however, the algorithm chooses to add the periodic term, which results in composite kernel that already well replicates the original function.

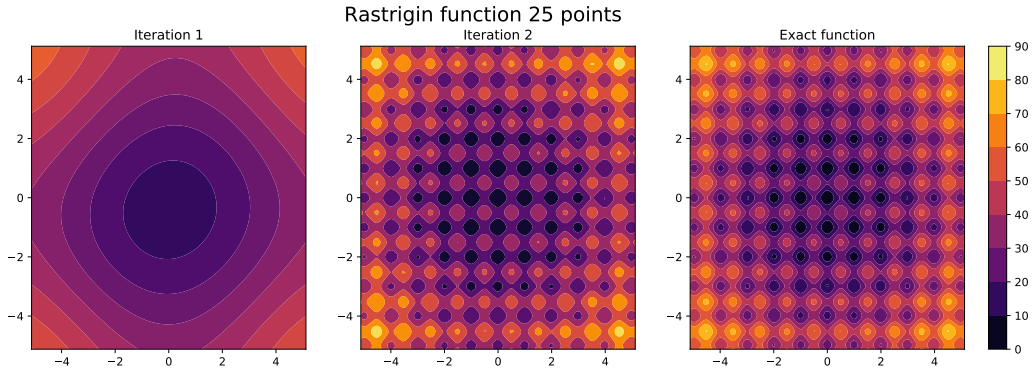


Figure C.1: Inference of the Rastrigin function using 25 random sample points.

C.2. Expression optimization in large datasets

Use of composite kernel can also improve model quality even in relatively large data-sets. Such a case is illustrated in Figure C.2, which presented reconstruction of the Ackley function sampled with 1000 data points. While the first iteration already provides a reasonable approximation, the second iteration still yields a significant improvement by inferring the periodic structure of the function (as the algorithm selects to add the periodic component to the kernel function).

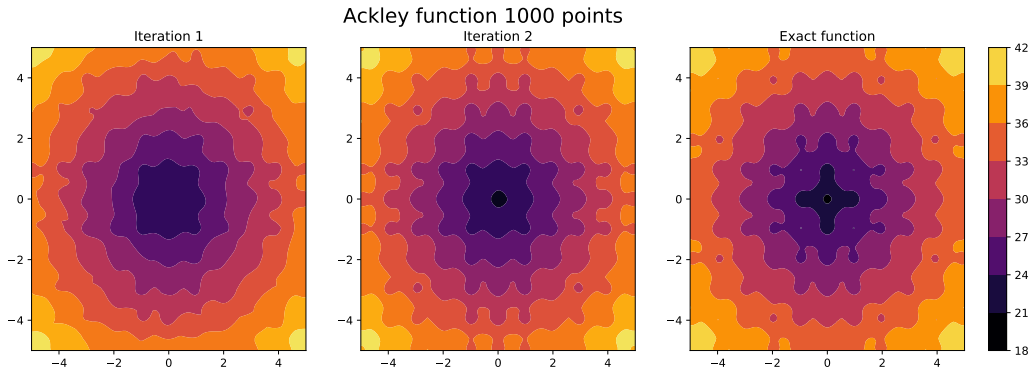


Figure C.2: Reconstruction of Ackley function using evolutionary Kernel approach and 1000 data-points.

C.3. Conclusion and future work

The algorithm certainly shows great potential, especially for use in high-dimensional inference problems. Nevertheless the method requires considerable refinement and optimization. In current im-

plementation the algorithm relies on Greedy optimization and brute-force exploitation of all possible options in every iteration, which results in significant computational time, making the current approach completely unfeasible for larger systems. A one particularly promising concept is that of switching to the genetic optimization which could be easily facilitated within the current implementation.

Secondly, the optimization could be improved by modifying the objective function. In the current implementation, the objective function relies only on log marginal likelihood (which is commonly used as an objective for optimizing hyperparameters). Log-marginal likelihood, however, inherently penalizes model complexity to prevent from over-fitting. This is a problem in case of inference of a more non-trivial function from limited datasets.

To solve this issue, it could be beneficial to extend the objective function by adding elements of cross-validation (that would check the solution on a small validation subset of data, and add penalty proportional to the residual between the prediction of the proposed solution and the validation dataset).

D

Verification of the QGP model for the metamaterial design

This appendix supplements the results from Chapter 5 with additional error data. Comparison with classical low-rank approximation provides additional evidence for the error analysis in Chapter 5. Finally, the analysis of the uncertainty error supports the discussion in Chapter 6 on origins of the QGP issue with estimating the prediction uncertainty.

D.1. Uncertainty errors

Figure D.1 and D.2 present error in uncertainty (one standard deviation σ) predicted with QGP for the 2D. The results support the qualitative discussion in Chapter 5 - for P_{crit} the uncertainty prediction is rather unreliable, which reveals by exceptionally large errors in order of 100% in major part of he domain. For E_{abs} the errors are large as well, however the peaks of in order of 100% occur rather locally than globally.

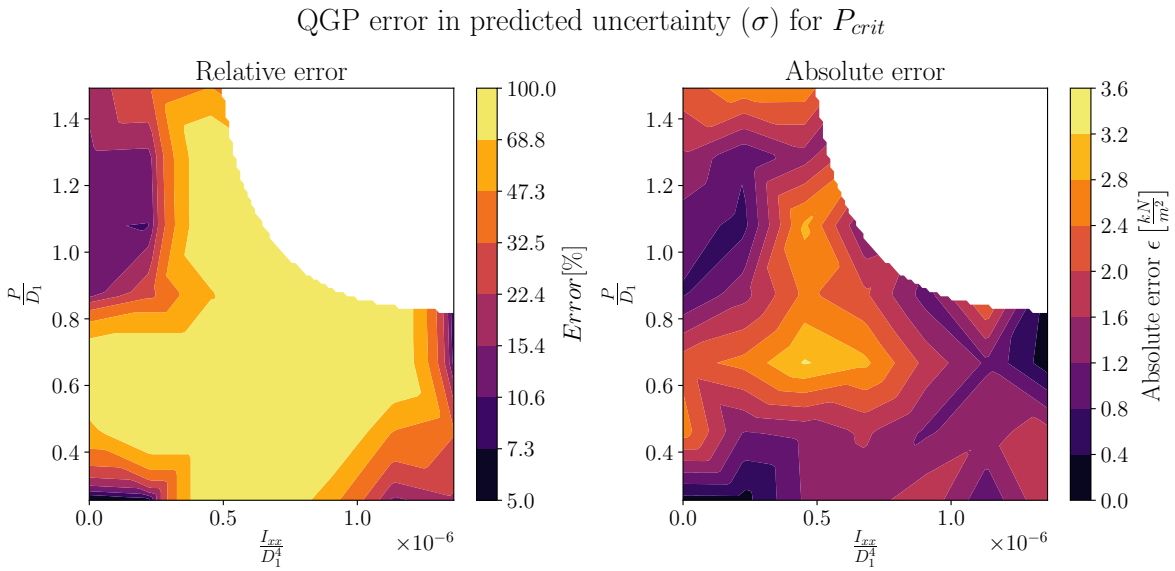


Figure D.1: Error in predicted uncertainty of the critical buckling load (P_{crit}) obtained with QGP algorithm

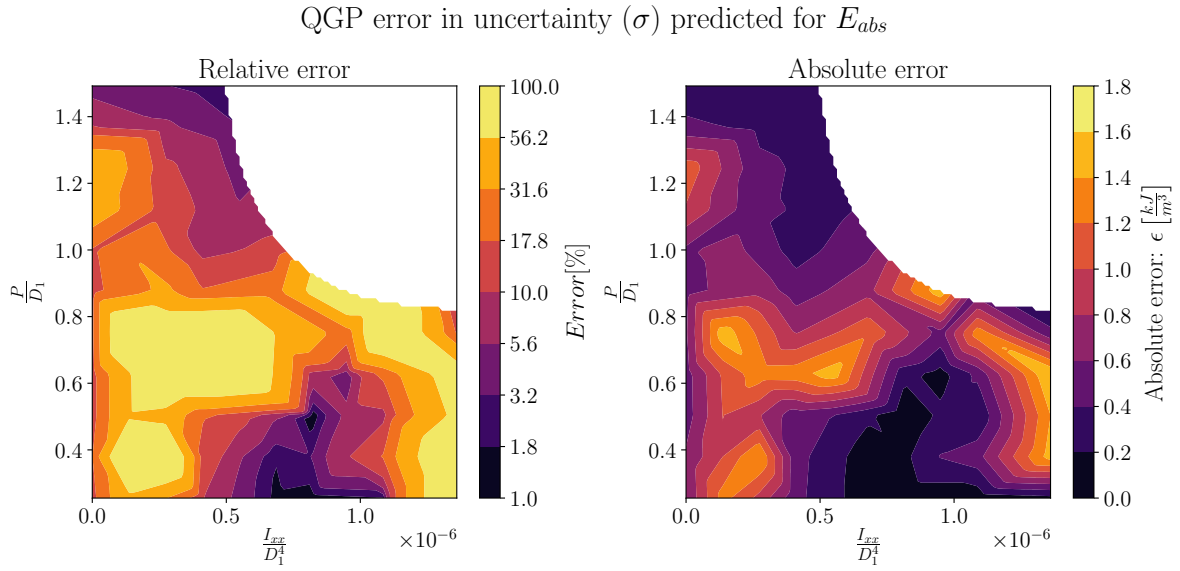


Figure D.2: Error of the QGP prediction uncertainty E_{abs} with respect to the classically solved system

D.2. Comparison with the PCA approximation

As discussed in analysis of the QGP algorithm in Chapter 3, the errors due to low-rank approximation are usually orders of magnitude larger than the other errors (e.g. due to time slicing). It is expected to be the case also in the considered examples, i.e. it gives rise to a hypothesis that the major part of the error originates from the low-rank approximation effects.

This hypothesis is verified by comparing the QGP errors with errors obtained with classical PCA approximation. The results are shown in Figure D.3 and D.4. Note that aside from the mean values, the plots compare uncertainty term $(\sigma^2_{prior} - \sigma^2)$, rather than just uncertainty σ. The reason for this is that QGP is in fact used to evaluate the difference between the prior and posterior variance, according to Equation 2.41. Therefore indication of errors due to a low rank approximation are found by comparing this term.

For the predicted mean values of P_{crit} and E_{abs} the correlation between the QGP and PCA is apparent, which proves that low-rank approximation is indeed the main source of error in this case.

For the variance term, the correlation, although present, is not as strong. In this case, the errors originating from the approximate matrix exponentiation are in the same order of magnitude. Note, however that compared with the prediction of the mean values, in this case the errors are at leas one order of magnitude lower, where the major part of the QGP approximates the solution with accuracy in order of 5%. Nevertheless, this translates onto errors in uncertainty in order of 100% as shown in Figure D.1 and D.2, which proves that the QGP problem of unreliable uncertainty prediction indeed originates from the error amplification (while deriving σ from $(\sigma^2_{prior} - \sigma^2)$ rather than from the QGP accuracy.

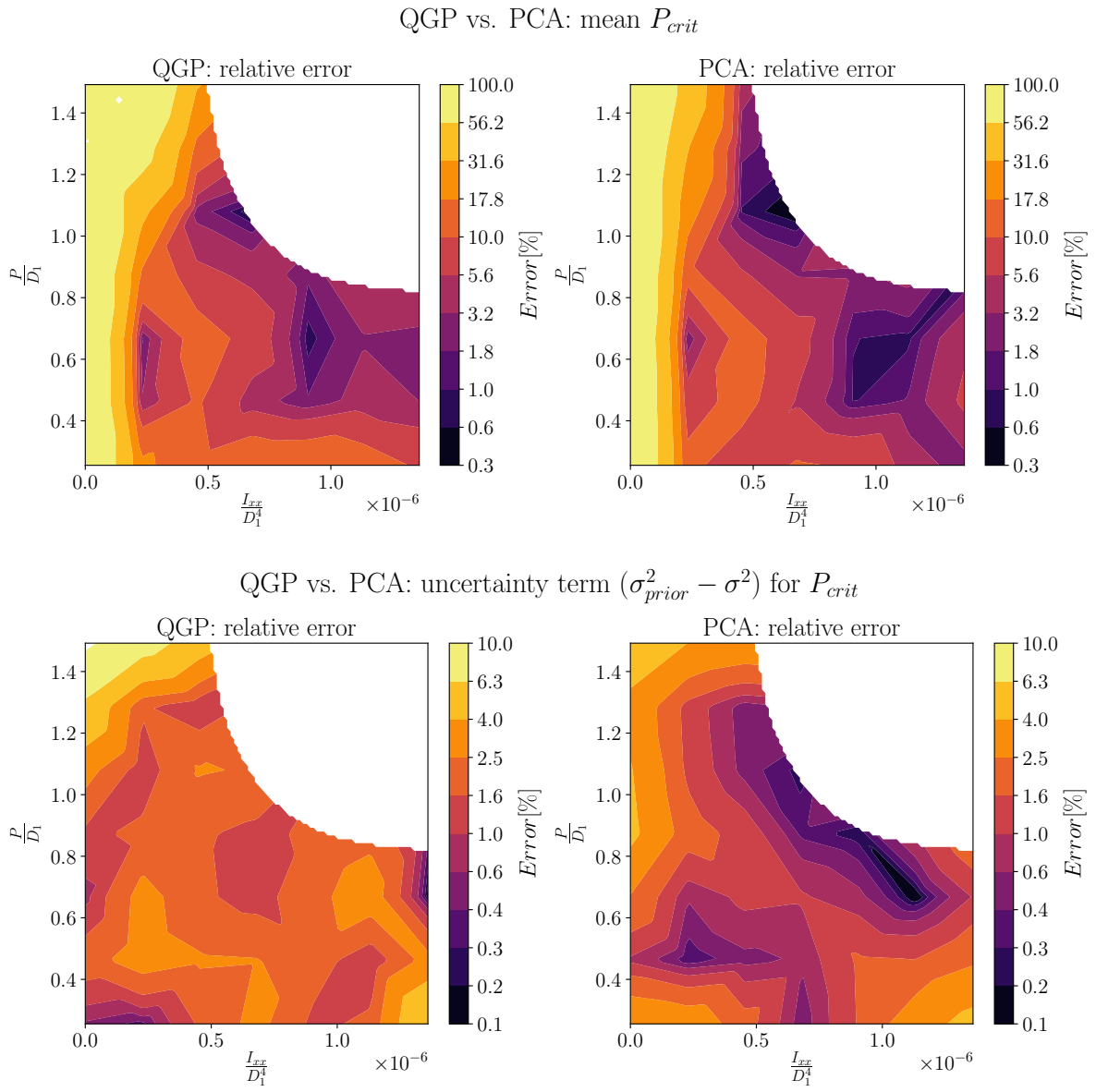


Figure D.3: Comparison of the QGP errors with the errors of the classical PCA approximation for the predicted mean and uncertainty of the critical buckling load.

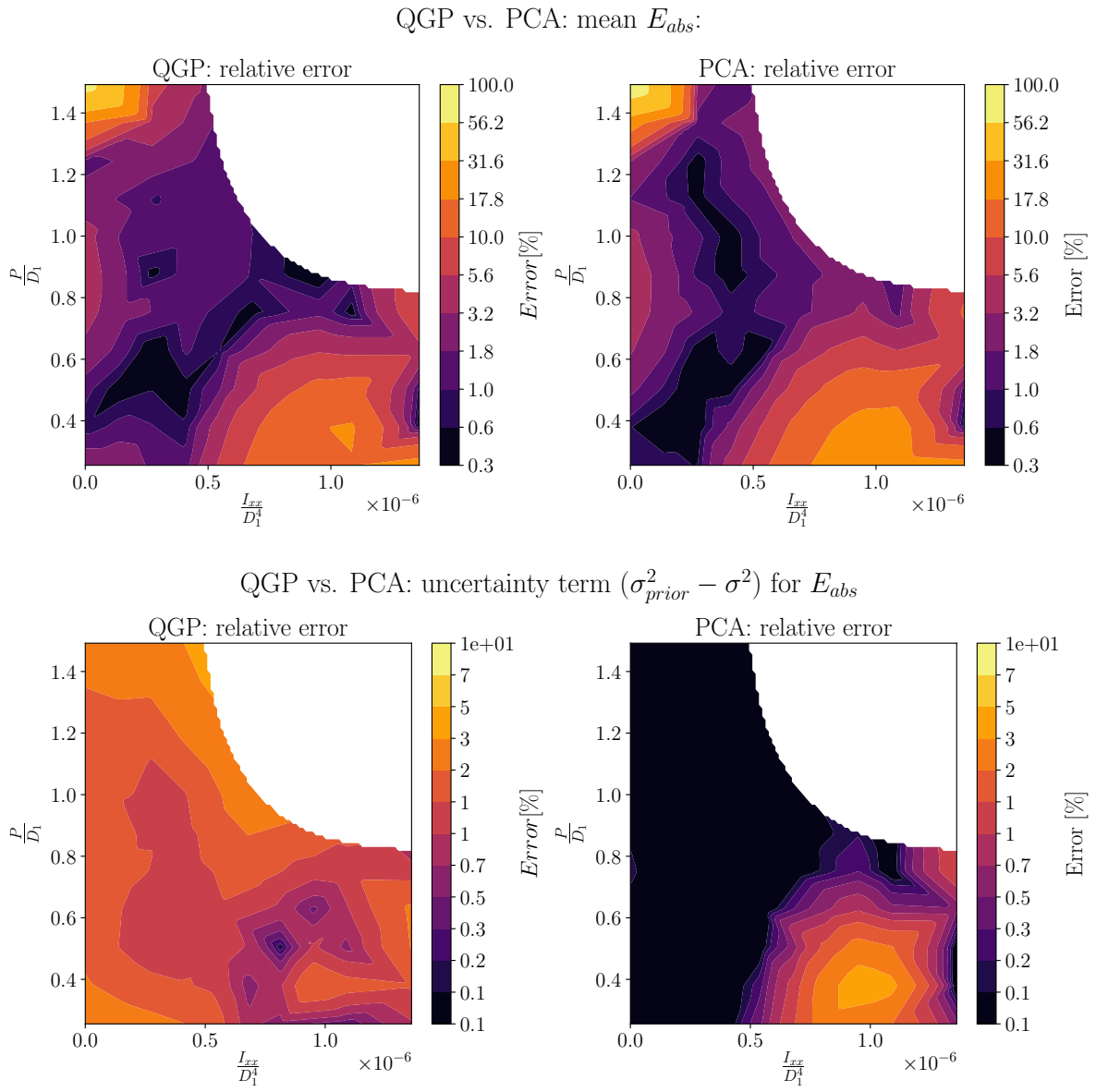


Figure D.4: Comparison of the QGP errors with the errors of the classical PCA approximation for the predicted mean and uncertainty of the energy absorption