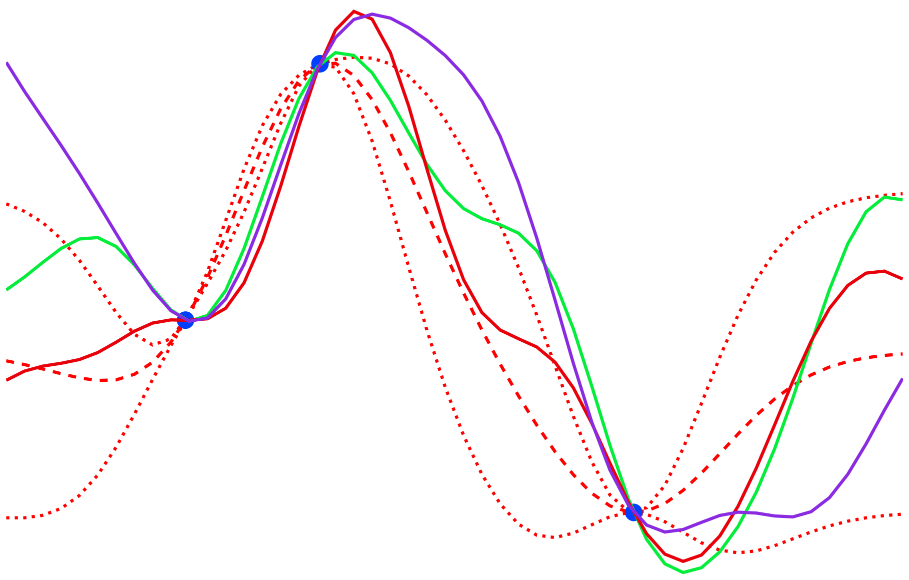
CHALMERS



Gaussian processes for emulating chiral effective field theory describing few-nucleon systems

Bachelor of Science Thesis for the Engineering Physics Program

MARTIN ERIKSSON
RIKARD HELGEGREN
DANIEL KARLSSON
ISAK LARSÉN
ERIK WALLIN

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

BACHELOR OF SCIENCE THESIS FOR THE ENGINEERING PHYSICS PROGRAM

Gaussian processes for emulating chiral effective field theory describing few-nucleon systems

MARTIN ERIKSSON
RIKARD HELGEGREN
DANIEL KARLSSON
ISAK LARSÉN
ERIK WALLIN

Department of Physics
Division of Subatomic and Plasma Physics
CHALMERS UNIVERSITY OF TECHNOLOGY

Gothenburg, Sweden 2017

Gaussian processes for emulating chiral effective field theory describing few-nucleon systems

Martin Eriksson[a], Rikard Helgegren[b], Daniel Karlsson[c], Isak Larsén[d], Erik Wallin[e]

Email:
[a]martieri@student.chalmers.se
[b]rikhel@student.chalmers.se
[c]dakarlss@student.chalmers.se
[d]isakla@student.chalmers.se
[e]walline@student.chalmers.se

Bachelor thesis at the Department of Physics, Chalmers
Bachelor's thesis TIFX04-17-07

Supervisor: Andreas Ekström, Christian Forssén
Examiner: Jan Swenson

Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 (0)31-772 1000

Cover:
Random functions sampled from a posterior predictive distribution with three observed points. Also depicted is the mean and double variance of the Gaussian process, see chapter 3.

Chalmers Reproservice
Gothenburg, Sweden 2017

Abstract

Gaussian processes (GPs) can be used for statistical regression, i.e. to predict new data given a set of observed data. In this context, we construct GPs to emulate the calculation of low energy proton-neutron scattering cross sections and the binding energy of the helium-4 nucleus. The GP regression uses so-called kernel functions to approximate the covariance between observed and unknown data points. The emulation is done in an attempt to reduce the large computational cost associated with exact numerical simulation of the observables. The underlying physical theory of the simulation is χEFT. This theory enables a perturbative description of low-energy nuclear forces and is governed by a set of low-energy constants to define the terms in the effective Lagrangian. We use the research code `nsopt` to simulate selected observables using χEFT.

The GPs used in this thesis are implemented using the Python framework `GPy`. To measure the performance of a GP we define an error measure called *model error* by comparing exact simulations to emulated predictions. We also study the time and memory consumption of GPs. The choice of input training data affects the predictive accuracy of the resulting GP. Therefore, we examined different sampling methods with varying amounts of data.

We found that GPs can serve as an effective and versatile approach for emulating the examined observables. After the initial high computational cost of training, making predictions with GPs is quick. When trained using the right methods, they can also achieve high accuracy. We concluded that the Matérn 5/2 and RBF kernels perform best for the observables studied. When sampling input points in high dimensions, latin hypercube sampling is shown to be a good method. In general, with a multidimensional input space, it is a good choice to use a kernel function with different sensitivities in different directions. When working with data that spans over many orders of magnitude, logarithmizing the data before training also improves the GP performance. GPs do not appear to be a suitable method for making extrapolations from a given training set, but performs well with interpolations.

Keywords: Machine learning, Gaussian processes, Chiral effective field theory, Scattering

Sammandrag

Gaussiska processer (GP:s) kan användas för statistisk regression, d.v.s. att göra prediktioner av ny data givet en mängd av observerad data. I detta sammanhang konstruerar vi GP:s för att emulera beräkningen av spridningstvärsnittet för lågenergiinteraktion mellan protoner och neutroner, samt kärnans bindningsenergi i helium-4. I regression med GP används så kallade kärnfunktioner för att approximera kovariansen mellan observerade och okända datapunkter. Tillämpningen av GP:s är ett försök att reducera den stora beräkningskostnaden förknippad med den exakta numeriska simuleringen av observablerna. Den fysikaliska teori som ligger bakom simuleringen är χEFT, vilken tillåter en perturbativ beskrivning av krafter på nukleär skala vid låga energier. Denna teori styrs av en uppsättning lågenergikonstanter, som definierar termerna i den effektiva Lagrangianen. Vi använder forskningskoden `nsopt`, som implementerar χEFT, för att simulera valda observabler.

De GP:s som används i denna kandidatrapport implementeras med hjälp av Python-ramverket `GPy`. För att mäta hur väl en GP presterar definierar vi ett felmått benämnt *model error*, som jämför exakta simuleringar med emulerade prediktioner. Vi studerar också GP:s tids– och minnesåtgång. Valet av träningsdata påverkar noggrannheten på den resulterande GP:ns prediktioner. Vi undersökte därför olika samplingsmetoder med varierande datamängd.

Vi konstaterade att GP:s kan fungera som en effektiv och mångsidig metod för att emulera de undersökta observablerna. Det går snabbt att göra prediktioner med GP:s efter träningens initialt höga beräkningskostnad. Med väl vald träningsdata och rätt optimering, kan GP:s även ge hög noggrannhet. Vi observerade att Matérn 5/2– och RBF-kärnorna fungerar bäst för de observabler som studeras. Latinsk hyperkubssampling konstateras vara en bra metod när parameterrummet är av hög dimension. Det är generellt ett bra val att använda en kärnfunktion med olika känslighet i olika riktningar när parameterrummet är multidimensionellt. När datan spänner över många storleksordningar förbättrar även logaritmering av datan innan träning GP:ns uppskattningsförmåga. GP:s verkar inte vara lämpade för extrapolering från ett givet träningsset, men de fungerar bra vid interpolering.

Acknowledgements

We would like to thank our supervisors, Christian Forssén and Andreas Ekström, for great guidance and support throughout our project.

The Authors, Gothenburg, June 2017

Contents

1 Introduction

Modern scientific research, with its increasing complexity and large amounts of data, is often limited by the availability of computational power. While simplified models with analytical solutions are useful for gaining a qualitative understanding, advanced numerical computations are often needed for quantitative predictions. Hence, there is a growing need for low-error approximative methods that are also computationally cheap in terms of time complexity and memory consumption. One possible approach to address this need is to introduce machine-learning methods. A simplified view of machine learning is the idea of supplying a computer a limited set of training data, from which a program can make predictions about other similar sets of data. The prospect is that machine-learning methods will be able to utilize the available data to reduce the computational complexity and cost of problems without decreasing the accuracy of solutions [1].

Our goal is to examine the feasibility of applying machine-learning algorithms in the form of statistical regression with *Gaussian processes* (GPs) to the realm of nuclear and particle physics. Computations of low-energy, nuclear-physics observables suffer from the non-perturbative nature of the underlying theory. The most modern solution to this problem utilizes *chiral effective field theory* (χEFT). The χEFT framework allows an expansion in a small parameter, despite the theory being non-perturbative [2]. By introducing a power-counting scheme to the χEFT expansion it is possible to classify the importance of different terms. The first order is known as leading-order (LO), the second order next-to-leading-order (NLO), followed by next-to-next-to-leading-order (N2LO), and so on. The parameters that govern the χEFT are referred to as *low energy constants* (LECs). The number of LECs increase with the order of the expansion. Even though the χEFT approach greatly reduces the computational demand for nuclear physics calculations, it is still a time-consuming task with many parameters in the form of LECs. Due to the strong parameter dependence, even a small change in the input parameters requires a costly new calculation [3].

In this thesis, we construct an emulator using a GP with the aim to reduce the time consumption for the calculation of few-nucleon observables. A GP uses a function known as a *kernel* to calculate the covariance between points in the training data to be able to make predictions about other points in the input space [1]. The training data is supplied to the GP in the form of simulation results from `nsopt`, a nuclear-physics research code using χEFT developed at Chalmers University of Technology [3–5]. The accuracy of the emulator is determined by comparisons between emulated and simulated values. The method is applied to emulate both the total cross section in proton-neutron scattering, as well as the binding energy for the helium-4 nucleus.

1.1 Purpose

The general purpose of this thesis is to examine the viability of using GPs to emulate and predict the simulation of physical observables within nuclear physics, in particular the total proton-neutron scattering cross section and the binding energy of the helium-4 nucleus. This aims to reduce the high computational cost associated with numerical simulations of these observables. We also want to give a general indication of the overall applicability of GPs for research within nuclear physics, and to provide rough recommendations of how to best utilize GPs to achieve the highest predictive accuracy.

More specifically the goals of this thesis can be summarized as follows:

- Construct a GP emulating `nsopt` simulations of proton-neutron scattering and the helium-4 binding energy.
- Examine optimal choices for GP-specific training methods, such as the choice of training data and kernel function.
- Study the computation time and memory usage for GPs to determine whether the use of GP emulations leads to a sufficient reduction in computational cost, compared to the corresponding numerical simulations.
- Test whether GPs also are a useful tool for extrapolated emulations, as opposed to interpolations.

1.2 Limitations

Our goal with this thesis is to create an accurate emulator, not to expand upon the underlying physics of the χEFT. We study the theories to get a basic understanding of the observables and methods we are emulating.

When working with the χEFT expansion, we are limited to expansion orders up to N2LO and also ignore three-body interactions. Including more calculations would not be beneficial for the use of GPs, and would mostly increase the computational complexity.

When dealing with GPs, we limit ourselves to a subset of kernels which are designed for purposes similar to ours. The training sets used are limited to 3000 points, as larger sets would be more demanding than practical for the computational power available to us.

1.3 Thesis structure

A general overview of χEFT and the concept of LEC parameters can be found in chapter 2. There are also explanations of the physical observables cross section and binding energy, as well as their connection to χEFT. These are the observables we later use to compare `nsopt` to their GP counterparts. We introduce some of the mathematics behind GP machine learning in chapter 3, to explain more of the behaviours of the GP emulators. We focus on the introduction of covariance functions (i.e. kernels) after the general introduction. We motivate our decisions of how to apply GPs towards χEFT in chapter 4, in terms of, amongst other things, sampling sizes, sampling methods, kernels that are used, and the defined error measure. Presentations and discussion of the results of the GPs performance alternate in chapter 5 in order to convey the logic behind some simplifications in terms of dropping certain GP parameters, such as the amount of validation data. We conclude the thesis with a brief summary of the general findings of the thesis and recommendations for future work in chapter 6.

2 Theoretical description of few-nucleon systems

In this chapter we present a general overview of the underlying theory used by the `nsopt` simulator. The simulation is based on an χEFT description of low-energy nuclear physics, which is governed by LECs. The simulated observables we study in this thesis are the total cross section for proton-neutron scattering (σ^{t}_{np}) and the nuclear binding energies of the helium-4 nucleus ($E_b(^4\mathrm{He})$).

2.1 χEFT and LECs

An effective theory is an approach used to obtain a description of physics in a limited energy region that is specified by a separation of scales, inherently manifested by the system under study [6]. The concept of separation of scales is used when, for example, we go from relativistic mechanics to the special case of classical Newtonian mechanics. By assuming that the velocity of an object v is small compared to the speed of light c, we can expand the expression for relativistic kinetic energy using a Taylor series in the small parameter v/c. The expansion rapidly converges, and by keeping only the first order term we end up with the well known Newtonian expression for kinetic energy [7]. A similar approach based on separation of scales is used in χEFT to study the low-energy nucleon interaction. In this thesis we will only briefly summarize the χEFT approach, for an introduction see Machleidt [2].

The strong force between quarks and gluons is very well explained by the equations of *quantum chromodynamics* (QCD). Particles that are made up of quarks and hence influenced by the strong nuclear force, such as protons and neutrons forming atomic nuclei, are referred to as hadrons. In the low-energy region characteristic for nuclear physics, QCD is non-perturbative. However, by using χEFT a small parameter expansion can still be found, and the strong force interaction between nucleons can be seen as an exchange of pions, the lightest of the hadrons. The separation of scales used in χEFT is provided by the experimentally observed gap in the hadron energy spectrum between the pions and the heavier mesons. The upper energy limit of the scale is denoted by Λ and is approximately the mass of a rho meson $\Lambda \approx m_\rho \approx 800\,\mathrm{MeV}$. Physics above this limit is integrated out, leaving only the lower-energy physics in the form of pions [2].

χEFT facilitates an expansion in the parameter defined as Q/Λ. Here Λ is the previously introduced upper energy limit defined by the rho meson and Q is the pion momenta in the low-energy processes we want to study. Typically, the scale of Q is given by the mass of a pion $Q \approx m_\pi \approx 140\,\mathrm{MeV}$. By counting the chiral order ν of the resulting expansion terms $(Q/\Lambda)^\nu$ it is possible to organize the terms in the expansion. The terms themselves can be represented by a set of Feynman diagrams describing the nucleon interaction. A finite set of terms belong to each order, while the whole expansion is infinite. The chiral order can take on values $\nu \geq 0$, with terms of order $\nu = 1$ disappearing due to symmetry reasons. By including the weaker higher order terms more information is included. The terms with $\nu = 0$ are referred to as leading order (LO), terms with $\nu = 2$ as next-to leading order (NLO), third order terms as next-to-next-to leading order (N2LO), and so on [2, 6].

The coupling coefficients in the effective Lagrangian are called low energy constants (LECs). The LECs govern the strength of the terms in the χEFT. Consequently, when increasing the number of terms kept in the expansion more LECs are introduced. The values of the LECs are not given by the χEFT, but have to be determined from fits of experimental data and are therefore subject to statistical errors. The LEC values we use to simulate the total scattering cross section and binding energy of the helium-4 nucleus with `nsopt` are values in error ranges previously calculated by Carlsson et al. [3]. We include terms up to N2LO, where 12 LECs are needed to describe the χEFT. Further information on how the values of the LECs are chosen is presented in section 4.2.

2.2 Scattering cross section

The scattering cross section σ is used as a measure of the absolute probability for a reaction to happen. Consider an incoming current of I_a particles per unit time hitting target particles spaced with N particles per unit area. A detector defining a solid angle $\mathrm{d}\Omega$ from the reaction point will measure a signal of R_b outgoing particles per unit time. A schematic overview of this process is presented in figure 1. These parameters give us the scattering cross section, proportional to the probability of the outgoing particle hitting the detector at a specific direction described by angles θ and ϕ.

$$\sigma = \frac{R_b}{I_a N}. \tag{2.1}$$

This is often referred to as the total cross section $\sigma_\mathrm{t} = \sigma$. We must note that while the cross section has the units of area, it tells us nothing of the actual geometrical area of the target particles. Since the detector in the above example only occupies a small solid angle it will only record some of the particles, $\mathrm{d}R_b$ and thus only describe a fraction of the cross section, $\mathrm{d}\sigma$. The distribution of particles over all outgoing angles can be described by a function $r(\theta, \phi)$, which gives us $\mathrm{d}R_b = r(\theta, \phi)\,\mathrm{d}\Omega\,/4\pi$. This allows us to formulate the differential cross section

$$\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega} = \frac{r(\theta, \phi)}{4\pi I_a N}. \tag{2.2}$$

This differential tells us the angular distribution of the reaction products. We note that we may calculate the total cross section, telling us the probability of *any* reaction, by integrating over all angles [8].

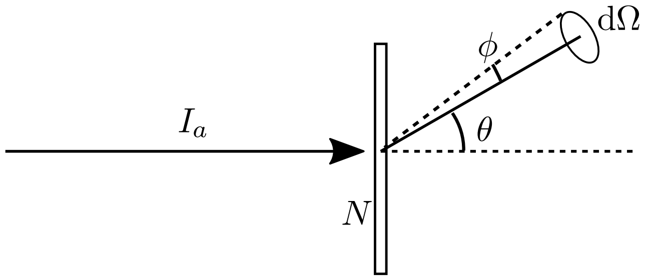


Figure 1: Schematic overview of an incoming current of particles I_a scattered against an area of target particles showing N particles per area. The scattered particle is detected in the solid angle $\mathrm{d}\Omega$.

The $r(\theta, \phi)$ distribution is ultimately decided by how the two particles interact, for example Coulomb scattering is decided by the electromagnetic interaction. In our case, χEFT is used to model the potential of the strong force between the particles. We study the total cross section of proton-neutron scattering, where the neutrons are the targets for the accelerated protons. Given a set of LECs and the kinetic energy of the proton in the lab frame of reference, we can numerically solve the Schroedinger equation and compute this cross section. A simulated total cross section curve in the range 1 MeV to 150 MeV is shown in figure 2.

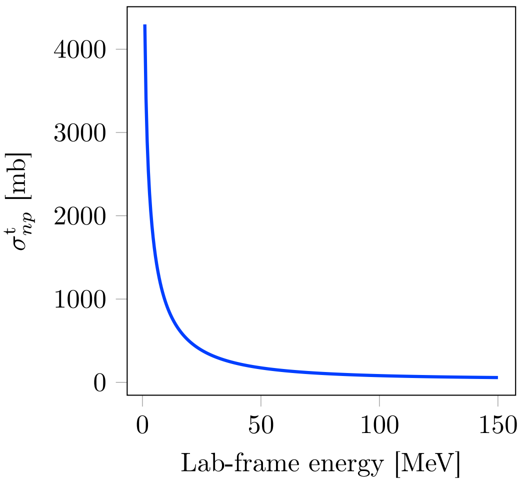


Figure 2: Simulated total cross section for proton-neutron scattering. The cross section is a decreasing function of energy that spans over many orders of magnitude.

2.3 Binding energy of few-nucleon systems

Another observable we study is the binding energy of the helium-4 nucleus. This is done by solving the Schroedinger equation for a system of A nucleons

$$H\,|\Psi_A\rangle = E\,|\Psi_A\rangle\,. \tag{2.3}$$

The Hamiltonian includes the kinetic energy, the strong force, and Coulomb interaction between particles:

$$H = T + V = \sum_{i=1}^{A} \frac{p_i^2}{2m_i} + \sum_{i<j=1}^{A} V_{ij} + \sum_{i<j<k=1}^{A} V_{ijk}. \tag{2.4}$$

Here, the eigenstate $|\Psi_A\rangle$ is an anti-symmetrical product state of single-nucleon states. We can expand the many-body eigenstate into a complete set of orthonormal basis states

$$|\Psi_A\rangle = \sum_{i}^{\infty} c_i |\phi_i\rangle . \tag{2.5}$$

More specifically, we choose $|\phi_i\rangle$ to be the product state of four non-interacting nucleons in a harmonic oscillator potential. The energy levels of each oscillator state is governed by the oscillator energy $\hbar\omega$. The number of included basis states is decided by the cutoff parameter N_{max}. Four-nucleon states with a total energy above $N_{\text{max}}\hbar\omega$ will not be included. Inserting this sum into the Schroedinger equation and multiplying from the left by $\langle\phi_j|$ gives us

$$\sum_{i}^{N_{\text{max}}} c_i \langle\phi_j|H|\phi_i\rangle = E \sum_{i}^{N_{\text{max}}} c_i \langle\phi_j|\phi_i\rangle , \tag{2.6}$$

and since $\langle\phi_i|\phi_i\rangle$ is unity for an orthonormal basis, we find that

$$H_{ij} c_i = E c_j. \tag{2.7}$$

Here, $H_{ij} = \langle\phi_j|H|\phi_i\rangle$ is a matrix element of the Hamiltonian. The lowest energy solution to this eigenvalue problem will be the closest to the ground state energy of the system according to the variational principle. We note that the value of a specific matrix element depends on the choice of the oscillator energy $\hbar\omega$. However, we have a complete basis for each energy. As N_{max} tends to infinity the energy eigenvalue will converge towards the actual binding energy, and the final result should be independent of $\hbar\omega$. Every N_{max} and $\hbar\omega$ give a new matrix of H_{ij}-values and this matrix increases in size with N_{max}. The lowest eigenvalue is found through diagonalization of the matrix of H_{ij}-elements, which can be computationally demanding for large N_{max}. This means that the `nsopt` binding energy often will be a function of $\hbar\omega$ and N_{max}. In general, it is not possible to find a closed-form analytical expression for this function [9].

Put shortly, the physical observables studied in this thesis are the total cross section for proton-neutron scattering (σ_{np}^{t}) and the nuclear binding energies of the helium-4 nucleus ($E_b(^4\text{He})$). These observables will be emulated by the GPs presented in the next chapter. The LECs and the lab-frame kinetic energy, as well as $\hbar\omega$ and N_{max} for the bound-state model (presented in this chapter) will serve as input parameters for our emulation problem.

3 Machine learning

Solving the Schroedinger equation for nucleon scattering observables or nuclear binding energies can be done on a modern computer using χEFT. However, the time costs quickly add up, and for large numerical studies that require many evaluations, it may be too costly to use simulations. Using machine-learning techniques we may predict values based on an already known set of data. This can be done in a fraction of the time used to simulate the same values. The specific branch of machine learning used in this thesis is GPs.

3.1 Predictions using Gaussian processes

We want to find a mathematical way to predict values of an unknown function, based on observed function values. Formally, a GP is defined as a collection of random variables with a joint Gaussian distribution. The statistical model is defined by its mean and covariance functions:

$$\begin{aligned} m(x) &= \mathbf{E}[g(x)], \\ k(x,x') &= \mathbf{E}[(g(x)-m(x))(g(x')-m(x'))], \end{aligned} \tag{3.1}$$

where $g(x)$ is an arbitrary function. The mean is often chosen to be 0, this may be done since we do not have any information about the function we want to predict. The covariance function k describes the covariance between two points in input space. These parameters define the prior distribution, which is our distribution before observing any function values. By choosing a set of points X in input space, we build a covariance matrix $K(X,X)$ by evaluating the covariance function element-wise. We want to find values of a function $\mathbf{f}$ at every point in X. The result is a multivariate distribution

$$\mathbf{f} \sim \mathcal{N}(0, K(X,X)), \tag{3.2}$$

from which we can draw sample functions. An example of this is shown in the left panel of figure 3.

To make useful predictions we need observations $\mathbf{f}'$, located at points X' in input space. The observations are commonly referred to as training data. Generally, these values may have an associated noise, or measurement error. However, the data used in this thesis is simulated using numerical methods and can be assumed to be noise-free. Again, we construct a covariance matrix, and find the following multivariate distribution:

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}' \end{pmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) & K(X,X') \\ K(X',X) & K(X',X') \end{bmatrix}\right). \tag{3.3}$$

From this distribution we may find another, conditional distribution of $\mathbf{f}$ given $\mathbf{f}'$, X, and X'. This is known as the posterior predictive distribution, which uses all known data to give an estimate of unknown data. The distribution is defined by [1]

$$\begin{aligned} \mathbf{f}|\mathbf{f}', X, X' \sim \mathcal{N}\Big(& K(X,X')K(X',X')^{-1}\mathbf{f}', \\ & K(X,X) - K(X,X')K(X',X')^{-1}K(X',X)\Big). \end{aligned} \tag{3.4}$$

Finding the mean and variance at X is done by evaluating the matrices, and sampling a random function may be done in the same way as before. An example with arbitrarily chosen training data is shown in the right panel of figure 3. When using GPs to make predictions, it is in general the mean of these sampled functions that is of interest. We also note that all of these definitions may be extended to multiple input dimensions by extending the covariance function to allow vector inputs [1].

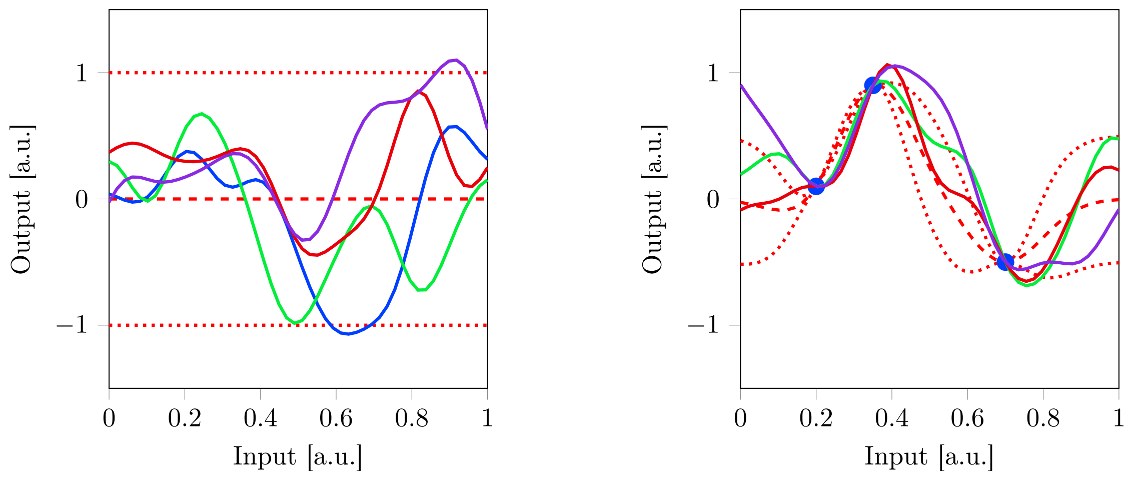


Figure 3: Randomly sampling functions from a Gaussian process corresponds to sampling values from a multivariate distribution. The left panel shows samples from an unconditioned prior distribution. The right panel demonstrates sampling from a conditioned posterior distribution with the 3 arbitrarily chosen points marked as blue dots. Both panels show the mean values of the GP as dashed lines and 2σ limits as dotted lines. The sample functions are shown as solid lines.

3.2 Kernel functions and optimization

The covariance function k is often referred to as a *kernel* function because of its importance in defining the GP. In machine learning, the purpose of a kernel function is to describe the similarity in output between arbitrary points in input space. Here we will use *stationary* kernels, which weigh the covariance between two points by their euclidean distance in input space. This class of kernels is invariant to translations in input space (hence stationary). A commonly used function is the *radial basis function* (RBF), sometimes referred to as the *squared exponential* function

$$k_{\mathrm{RBF}}(x, x') = \sigma^2 \exp\left(-\frac{|x - x'|^2}{2l^2}\right). \tag{3.5}$$

Here, l denotes the characteristic length scale of the kernel, and σ its variance. These are referred to as hyperparameters. We note that this function tends to zero quickly as the distance between the input points increase. Varying l is demonstrated in figure 4. Changing this hyperparameter can be seen as changing the *sensitivity* of the kernel. A large l means training points at a larger distance will influence the predictions more, and vice versa. The scale factor σ is used to the adjust the variance of the distribution to fit the studied data.

We note that the quality of the predictions depend strongly on the hyperparameters. We have to find a set of hyperparameters that are most likely to fit our data. This is done by evaluating the log likelihood function of observations $\mathbf{f}'$ given X':

$$\log \mathrm{P}(\mathbf{f}'|X', l, \sigma) = -\frac{1}{2}\mathbf{f}'^{\intercal} K^{-1}\mathbf{f}' - \frac{1}{2}\log|K| - \frac{n}{2}\log 2\pi. \tag{3.6}$$

K is the covariance matrix as earlier, which implicitly depends on both hyperparameters. To determine the optimal hyperparameters, we must find the max of the log likelihood function. This maximum can be found using a number of different optimization algorithms. The implementation of GPs used for this thesis uses the *L-BFGS-B*-algorithm, which is a common method for solving non-linear optimization problems [10, 11].

In many applications with multidimensional input spaces, each dimension might exhibit different sensitivities. The RBF kernel described in equation (3.5) uses a single length scale for all inputs, which assumes the desired function must behave similarly in all directions. It is straightforward to adjust the kernel function to accommodate a different length scale for every input dimension, and the result is given by

$$\begin{aligned} k_{\mathrm{RBF}}(\mathbf{x}, \mathbf{x}') =& \sigma^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^{\intercal} L (\mathbf{x} - \mathbf{x}')\right), \\ L =& \operatorname{diag}(\mathbf{l})^{-2}. \end{aligned} \tag{3.7}$$

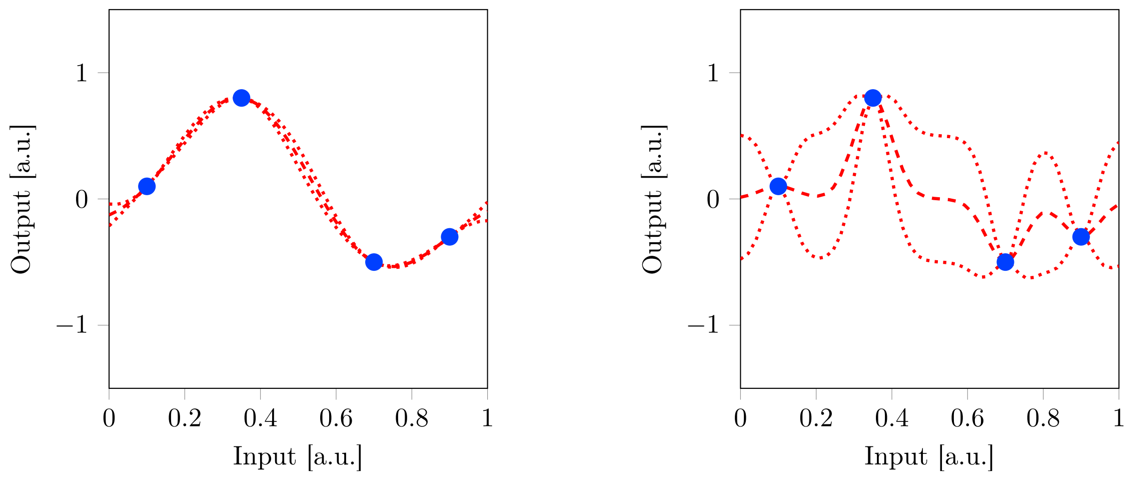


Figure 4: Both plots use the RBF kernel and the same arbitrarily chosen training points to find mean (dashed) and variance (dotted). The left one has a long length scale of 0.5, this conditions the function to change more slowly. The right panel shows a short length scale of 0.05, which allows for quick changes in the function, and a higher variance.

Clearly, this will increase the complexity of the optimization of the hyperparameters.

In summary, GPs can be used as a statistical approach to regression. The GP is defined by its kernel function, hyper parameters and observations. The next step in constructing an accurate emulator is the choice of appropriate training data (observations) and a suitable kernel. The methods for making these choices will be discussed in the next chapter.

4 Data collection and GP regression

In this chapter we present the methods used to obtain data and the GP training process. First, an introduction to the simulation method is given, specifying the requirements and capabilities of the `nsopt` code. This is followed by a description of the sampling schemes used to choose values for the input parameters given to `nsopt` for the calculation of cross sections and binding energies. In the next section we treat the GP training and the different kernels used to construct an accurate emulator. We test the accuracy by introducing an error measurement. The computational cost is evaluated by measurements of time complexity and memory consumption.

4.1 Simulation of observables with nsopt

Adequate training data has to be provided to be able to train a GP with high predictive accuracy. To generate this data we use a research code called `nsopt`, which is developed at Chalmers University of Technology [3–5]. The code implements χEFT to model the strong interaction and solve the Schroedinger equation for few-nucleon systems. More specifically, we study the proton-neutron cross section σ_{np}^{t} and the binding energy of a helium-4 nucleus $E_b(^4\mathrm{He})$.

When studying scattering cross sections, `nsopt` requires a set of 12 LECs and the lab-frame kinetic energy of the incoming particle. The simulated value represents the total proton-neutron scattering cross section of the interaction in millibarn (mb).

The simulations of nuclei binding energies are done with regard to another parameter space. Here, a single set of LECs is supplied together with the model parameters presented in section 2.3. `nsopt` then calculates the binding energy for the specific $\hbar\omega$ value, and every even integer cutoff up to $N_{\max}$. A typical value of $N_{\max}$ is 20, and $\hbar\omega$ is typically in the range 10 MeV to 50 MeV.

4.2 Sampling of input parameters

To train a GP we use the input parameters to nsopt together with the simulated value as target output. To be able to make predictions in a large volume of input space these data points need to span the space, and be somewhat evenly distributed.

When simulating cross sections, we explore a 12-dimensional parameter space that is composed of the relevant LECs from χEFT up to N2LO. We consider a restricted sub-volume that is spanned by the LEC ranges specified in table 1, obtained from an earlier study [3]. The typical range of data points used for training is between 100 to 3000 points.

The sampling methods used to choose LEC values are: (i) *Latin hypercube sampling* (LHS), and (ii) sampling from a multivariate Gaussian distribution.

LHS is a sampling method for generating a space-filling, evenly distributed, and close to random set of data points in a multidimensional space [12]. LHS is performed by dividing every axis into N equally spaced intervals and randomly selecting a value in each interval. Thereafter, one value from each axis is randomly selected to create data points , which is repeated N times and each value is used once and only once [13]. A two-dimensional example of LHS sampling is shown in the left panel of figure 5. The LHS implementation used in the thesis is part of `pyDOE`, a Python package for designing experiments [14].

The second sampling method that we use is sampling from a Gaussian distribution. The samples are distributed according to the standard normal distribution. Hence, the selected data points mainly appear in the center of the multidimensional space as is shown in a two dimensional example in the right panel of figure 5.

For the work with bound states, the LECs are not varied, which leaves the only relevant input parameters $\hbar\omega$ and $N_{\max}$. The values of $N_{\max}$ used by `nsopt` are equally spaced (2, 4, 6, ..., 20) so it is also logical to use equally spaced $\hbar\omega$-values. This kind of equally-spaced grid of points is achievable in a lower-dimensional parameter space, but in higher-dimensional spaces (e.g. 12-dimensional as for the cross sections), the number of required data points would quickly increase above manageable numbers. Instead of generating a separate data set for testing the GP predictions of bound states, a subset of the full data set was used for training while the rest of the data was used for validation.

Table 1: The 12-dimensional LEC volume from which parameter values are sampled for calculating the cross section. The LECs are specified by an experimentally determined range with lower and upper bounds. The fit to experimental data has been done by Carlsson et al. [3].

LEC	Range	
	Lower	Upper
$\widetilde{C}^{(np)}_{^1S_0}$	-0.1519	-0.1464
$C_{^1S_0}$	2.4188	2.5476
$\widetilde{C}_{^3S_1}$	-0.1807	-0.1348
$C_{^3S_1}$	0.5037	0.7396
C_{E_1}	0.2792	0.6574
$C_{^1P_0}$	0.9924	1.6343
$C_{^1P_1}$	0.0618	0.6635
$C_{^3P_1}$	-0.9666	-0.4724
$C_{^3P_2}$	-0.7941	-0.6324
c_1	-0.8329	0.2784
c_3	-4.3601	-3.4473
c_4	1.8999	4.2353

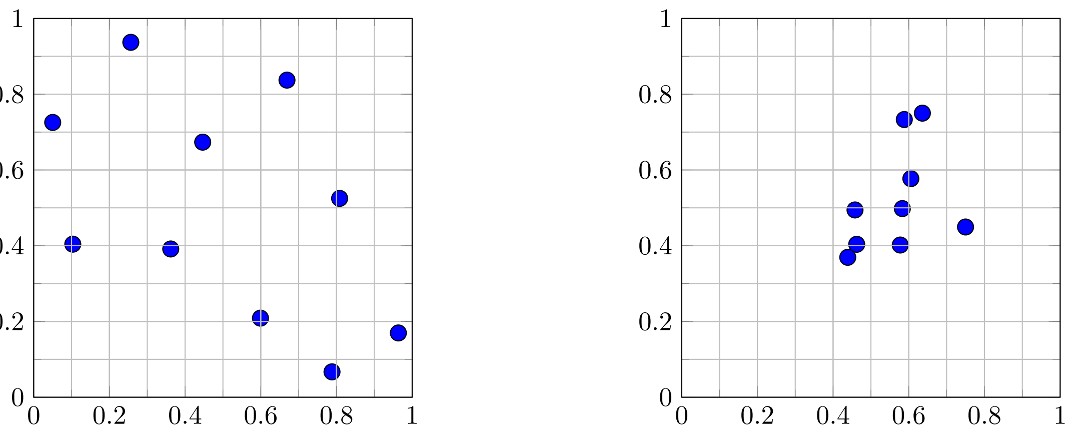


Figure 5: Comparing the two sampling methods, LHS and Gaussian sampling, in two dimensions with 10 samples. LHS presented in the left panel distributes points more evenly, while Gaussian sampling is more likely to place points near the center as seen in the right panel.

4.3 Kernels and hyperparameters

To implement the GP we use a Python framework called `GPy` [10]. This framework contains the kernels, algorithms and optimization methods needed to use GP regression for predictions. `GPy` implements four of the most common stationary kernels, which are all tested and compared. These kernels are RBF, exponential, Matérn 3/2 and Matérn 5/2, all of them listed below:

$$\begin{aligned}
k_{\mathrm{RBF}}(x,x') =& \sigma^2 \exp\left(-\frac{|x-x'|^2}{2l^2}\right),\\
k_{\mathrm{Exp}}(x,x') =& \sigma^2 \exp\left(-\frac{|x-x'|}{l}\right),\\
k_{\mathrm{Mat32}}(r) =& \sigma^2(1+\sqrt{3}r)\exp\left(-\sqrt{3}r\right),\\
k_{\mathrm{Mat52}}(r) =& \sigma^2(1+\sqrt{5}r+\frac{5}{3}r^2)\exp\left(-\sqrt{5}r\right),\\
\text{where } r(x,x') =& \sqrt{\sum_{i=1}^{\mathrm{dim}}\frac{(x_i-x_i')^2}{l_i^2}}.
\end{aligned} \tag{4.1}$$

If we look at the multidimensional length scale defined in section 3.2 we note that RBF and exponential may be rewritten to include it. We also see that the Matérn kernels already include this feature in the definition of r. The kernel and the dimension of the length scale are two important variables when finding an optimal GP for predictions.

The LECs are defined on different intervals, and affect the simulated value to different degrees. This results in different sensitivities in different input directions, and suggests that multidimensional length scales are a good idea to use. `GPy` has the option to use both single and multiple dimensions for all 4 kernels used in this thesis.

4.4 Error and complexity Analysis

An analysis of how an emulator compares to `nsopt` simulations is required to determine the accuracy of the GP predictions in the physical context of χEFT. Another factor to consider is how time and memory usage differ when using different GP training methods; the errors that are inherent with using GP predictions as opposed to the simulated values can not be justified if the emulation is as computationally expensive as `nsopt` itself.

To test the accuracy of the predictions acquired with the GP models, another data set of input parameters and simulated observables is needed. The set of validation points is compared to the predictions to give an error measure of the model and can therefore not be the same as the training points. When considering how close the predictions of the GP come to the true values, a measure of the errors should be defined in a consistent way. We decided to use a root-mean-square measure of the relative errors of predicted values $Y_{p,i}$ compared to the simulated validation values $Y_{v,i}$. We will refer to this measure as the *model error*

$$e = \frac{1}{\sqrt{N}}\sqrt{\sum_{i=1}^{N}\left(\frac{Y_{p,i}-Y_{v,i}}{Y_{v,i}}\right)^2}. \tag{4.2}$$

We use this measurement because the errors cannot cancel each other and important outliers affect the measure in a significant way. Figure 6 presents an example of how the model error is calculated for two different sets of predicted data. The left panel represents an accurate prediction, with a corresponding low model error while the right panel results in a high model error due to the larger deviations from the simulated values.

In summary, a lot of our work consists of choosing useful input parameters for the GP training process. The simulations used for training the GPs are generated with the research code `nsopt`. In addition to choosing suitable input parameters, we try to find the most accurate kernel. How well the GP performs is tested with the model-error parameter, that can be calculated by comparing GP predictions to a set of validation data. The results of our work is presented in the next chapter.

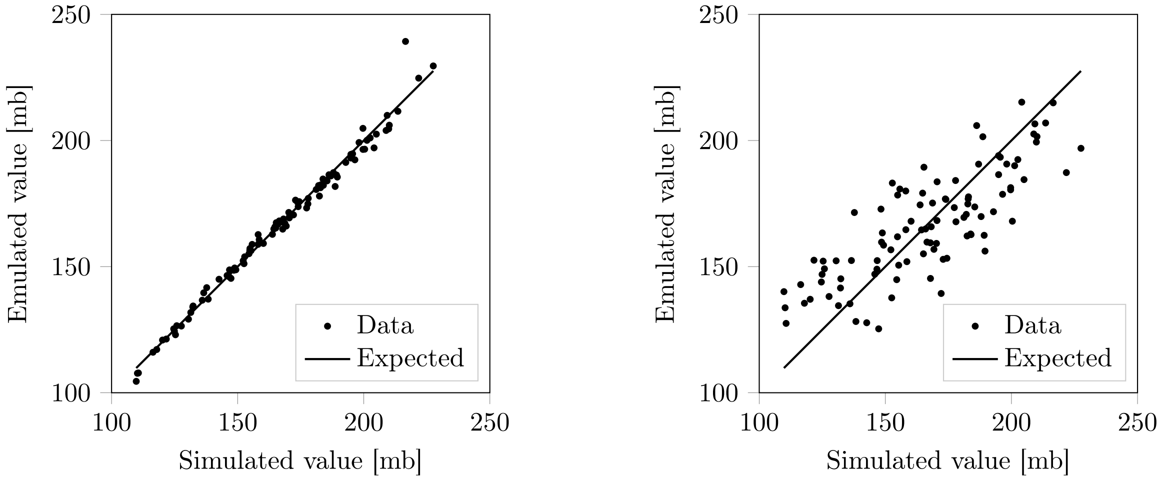


Figure 6: Two examples of how the emulated values (i.e. predictions) compare to the simulated values. The data points in the right panel are for the most part further from the expected line. This is also reflected in the model errors: 0.02 for the left panel and 0.10 for the right panel.

5 Results

In this chapter we present the results from our work with GPs for constructing emulators for the calculations of total scattering cross sections and the binding energy of the helium-4 nucleus. The previously introduced model error is used to compare the performance of different emulators. First, we discuss our findings about the sampling methods used to choose input parameters for the GPs, both for training and validation of the acquired predictions. The next part compares results obtained with the four different kernels, both in terms of model error as well as computational time and memory complexity.

5.1 Emulation of scattering total cross sections

The simulation of a single proton-neutron total cross section at a specific energy requires 12 LECs as parameters. In the calculations we fixed the energy of the incoming proton to 50 MeV in the lab frame of reference. To construct an emulator capable of predicting total cross section values at different energies, the energy interval has to be taken into account as a separate 13th parameter. In this work we have preformed such a study for an interval between 1 MeV to 150 MeV.

5.1.1 Usage of validation data

Our main measure of model performance is the model error defined in equation (4.2). To compare model errors from different kernels or training sets we must make sure that it does not depend on the number of validations points. As seen in figure 7, this is generally true. The model error still varies in the range 100 to 2500 but only by about 5×10^{-3}. The variations can be accredited to random elements from sampling. Using multiple validation sets and averaging the model error could reduce this variation.

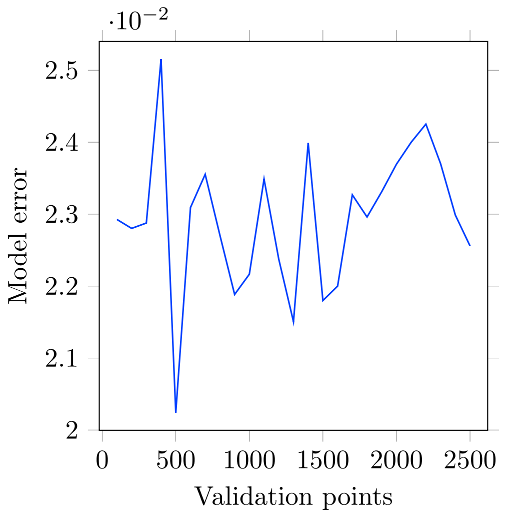


Figure 7: Model error as a function of increasing validation points calculated at 50 MeV using GP emulators for σ^{t}_{np}. All datasets are sampled using LHS, and the training is done with 1600 data points. The model error is relatively constant except for variations due to random elements from sampling.

5.1.2 Comparison of different sampling methods

In constructing an accurate emulator it is crucial to find methods for sampling input data in a way that minimizes the prediction errors over the entire input space. This make it necessary to compare combinations of sampling methods for training and validation data.

From table 2 we see that LHS and Gaussian sampling of training data perform about as well in terms of model error, but Gaussian sampling generally has a much larger variance. A model trained with Gaussian-sampled data has a large point density in the center of the volume and thus gives very good predictions in this area. Predictions towards the edges of the volume however, have a very large variance.

Based on this finding we will continue to use LHS for both training and validation. The idea is that we want the data points to span the input volume both when training and when validating data.

Table 2: Comparison of combinations of sampling methods. The comparative model error uses the method described in equation (4.2) normalized so that the lowest model error is unity. The variance, as presented in section 3.2, of the predicted values is compared in the same way. The last column shows the maximum relative error. All data sets have 1000 points and uses the RBF kernel.

Combinations of sampling methods		Comp. model error	Comp. variance	Max error
Training sampling	Validation sampling			
LHS	LHS	1.30	1.43	0.15
	Gaussian	1.00	1.00	0.17
Gaussian	LHS	2.04	6.44	0.23
	Gaussian	1.33	2.69	0.18

5.1.3 Minimizing model error of emulations

An emulator should preferably both be time efficient and make accurate predictions. The number of training points that the GP is constructed from is therefore a highly important parameter since it affects both the time consumption and the model error. The model error decreases as a function of the number of training points, as shown in figure 8. Another key ingredient is the GP kernel. The four kernels that we compare are RBF, Exponential, Matérn 3/2, and Matérn 5/2. The two kernels that result in the lowest model errors in the studied interval, 100 to 3000 training points, are Matérn 5/2 and RBF. The comparison between the four kernels is also shown in figure 8.

Beyond increasing the number of training points and choosing a good kernel, we also utilized the option of using a multidimensional length scale as described in section 3.2. This allowed for different sensitivities in different directions in the LEC volume, making a good fit more probable. The difference in model error when using the multidimensional length scale is seen in figure 8. It is clear that the use of a multidimensional length scale improves the accuracy of the predictions.

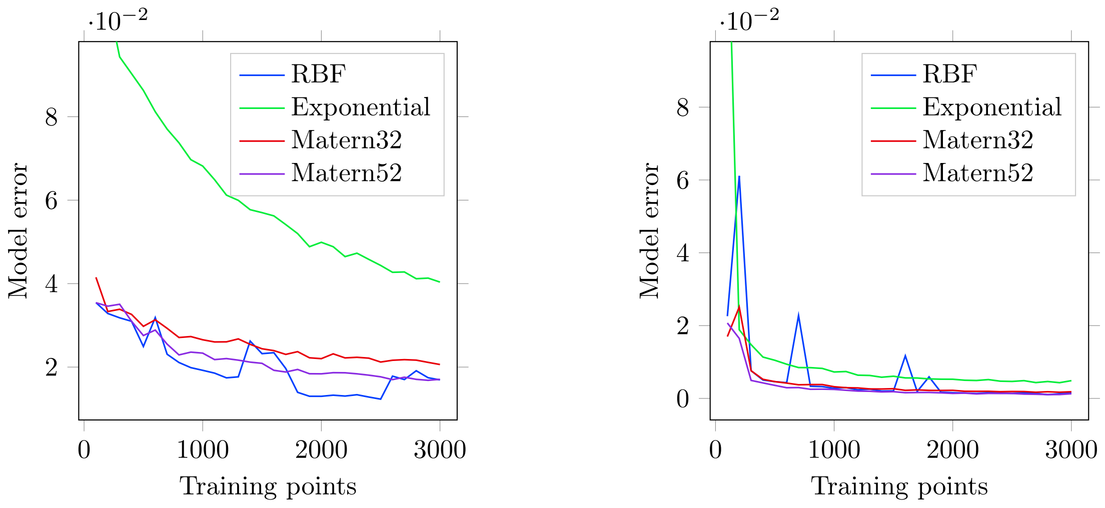


Figure 8: Measured model error for σ_{np}^{t} as a function of the number of points used for training GPs with four different kernels. The model error decreases as a function of the number of training points. Matérn 5/2 and RBF are the kernels with lowest model errors. In the right panel a multidimensional length scale was used when training the GPs.

5.1.4 Time and memory complexity

To quantify the possible computational gains from replacing `nsopt` simulations with the GP emulator we measure the time consumption and memory use of the GP training and prediction process. The procedure is done for training input in the form of LHS generated LEC samples ranging from 100 to 3000 points in the LEC volume space, in steps of 100 points. The prediction is performed for a separate set of 1000 LEC sample points. The same LEC data is used to measure time and memory both for GP training with the multidimensional length scale and without. This is to compare the increased computational cost of the multidimensional length scale to the gain in predictive accuracy when using the setting.

The training time measurements are presented in figure 9. The right panel shows GPs trained with the multidimensional length scale, while GPs in the left panel do not use it. Notice that the multidimensional setting increases the training time with about a factor ten, compared to training without this setting. No comparable increase can be seen for the corresponding prediction times, as illustrated in figure 10. This difference is explained by the fact that when already optimized, the lookup time complexity for a one-dimensional length scale is the same as for a multidimensional length scale. A GP prediction is therefore more or less independent of whether the training process uses multidimensional length scale or not. The training process does not exhibit this independence. This is due to the fact that a N-dimensional length scale increases the dimension of the space the optimization has to be performed for [10]. Disregarding the initial cost of the training process, the cost of predicting new points is low. This means that it is possible to use adequately trained GPs to quickly make predictions about unknown subsets in the LEC volume that the GPs is trained for [10].

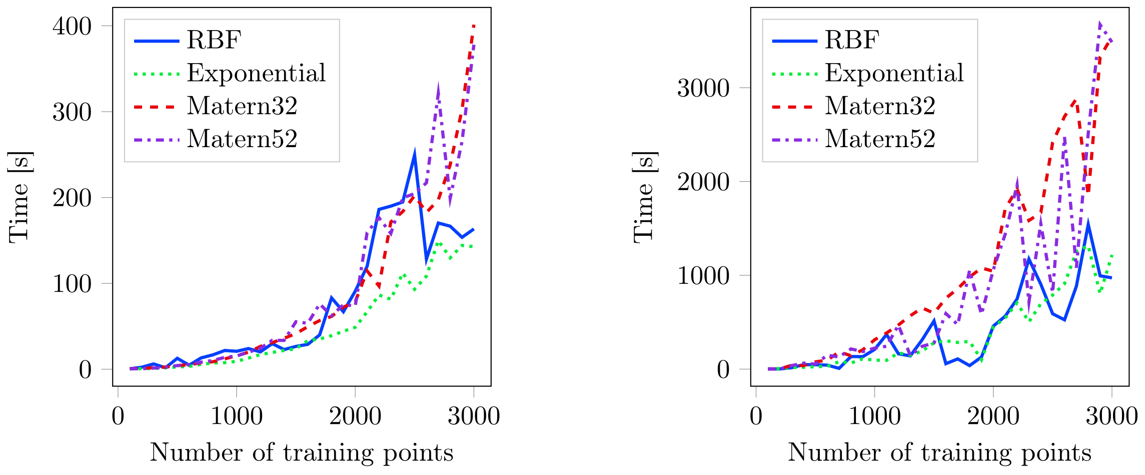


Figure 9: Measured time consumption when training a GP to emulate σ_{np}^{t} . The number of data points used for training ranges from 100 to 3000 points in steps of 100 points between each training set. In the right panel a multidimensional length scale was used when training the GPs. This is seen to increase the training time with about a factor ten.

The time measurements steadily increase with the number of sample points used. The discrepancies between close points, as seen for example in figure 10 with sharp peaks up and down, are due to the load from other processes on the computer at the time when the measurements were made. These differences are even more noticeable for the training processes in figure 9 since the GP employs an iterative optimization algorithm, a different number of iterations are needed before convergence for different sets of training data. The general trend shows a faster growth for the two Matérn kernels than for the RBF and exponential kernels. The time complexity of the training process with n_{t} training points is at most $\mathcal{O}(n_{\mathrm{t}}^3)$, due to matrix inversion operations on the covariance matrix. The optimization process itself is of order $\mathcal{O}(n_{\mathrm{hp}}^2)$ from the optimization of n_{hp} hyperparameters [10].

The memory measured during training is presented in figure 11. It is shown that the memory consumption is the same no matter which kernel is used and whether the multidimensional length scale is applied or not. This is due to the number of training points n_{t} being the only determining factor. The memory used by the GP increases as $\mathcal{O}(n_{\mathrm{t}}^2)$. Since the GP training is run in a Python environment there is also some computational overhead of about 200 MiB. By taking the overhead into account the memory curves in figure 11 follow a second degree polynomial fit quite closely, in line with the theoretical limit [10].

Similar results were seen for the memory consumption when we used the trained GP to generate predictions.

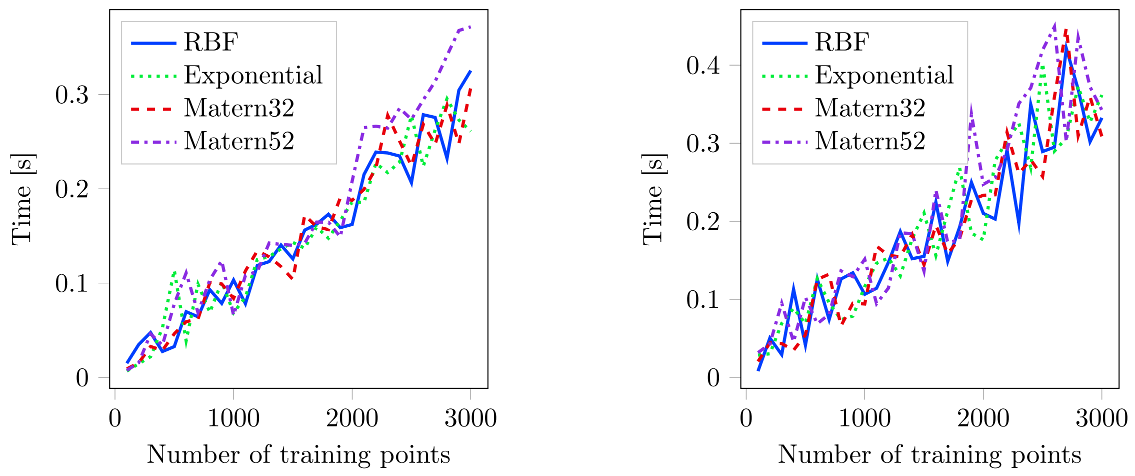


Figure 10: Measured time consumption when using 1000 data points to get predictions from a GP trained to emulate σ_{np}^{t} . The number of data points used for training ranges from 100 to 3000 points in steps of 100 points between each training set. In the right panel a multidimensional length scale was used when training the GPs. Using this setting does not greatly impact the time needed for calculating predictions.

The measurements are presented in figure 12. Noticeable differences are in the beginning and end of the sample interval. For low number of training points the memory graph exhibits strong fluctuations. These are a side effect of the quick execution times of GP predictions. The measurement software is not reliable for times below 0.1 s, corresponding to number of samples below 1000, since the memory changes may happen so quickly that the measurement misses it. In the end of the measurement interval, the graph shows a drop in measured memory around 2000 training points followed by continued growth. The reason for this drop is not known, but may be an effect from the Python garbage collection removing memory not in use.

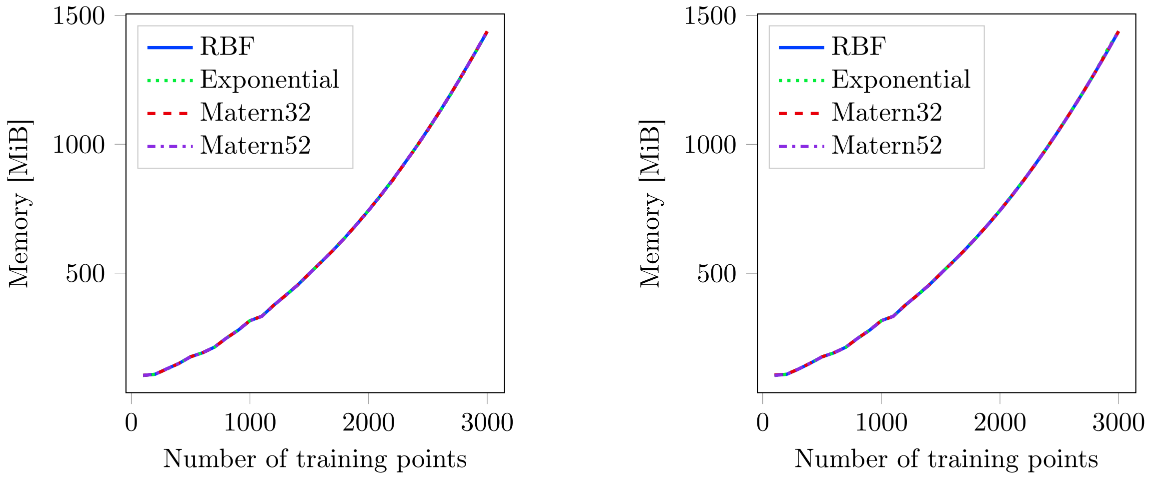


Figure 11: Measured memory consumption when training a GP. The number of data points used for training ranges from 100 to 3000 in steps of 100 points between each training set. In the right panel a multidimensional length scale was used when training the GPs. Clearly, the memory consumption is the same whether this setting is used or not. About 200 MiB of Python overhead is included in the measured memory.

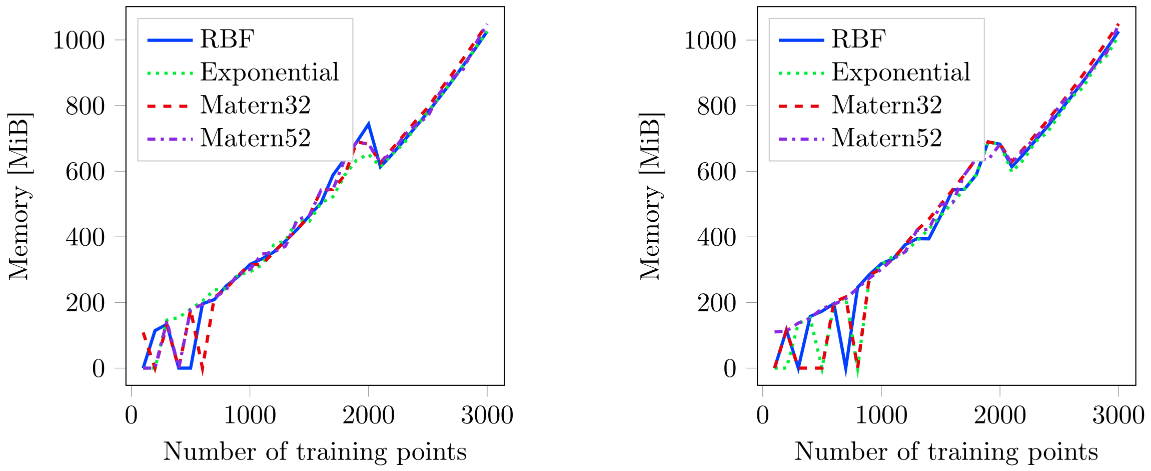


Figure 12: Measured memory consumption when using 1000 validation points to get predictions from a trained GP. The number of data points used for training ranges from 100 to 3000 in steps of 100 points between each training set. In the right panel a multidimensional length scale was used when training the GPs. The memory consumption is about the same whether this setting is used or not. Included in the measured memory is about 200 MiB of Python overhead. The high variance at the beginning of the interval is an artifact of the measurement software.

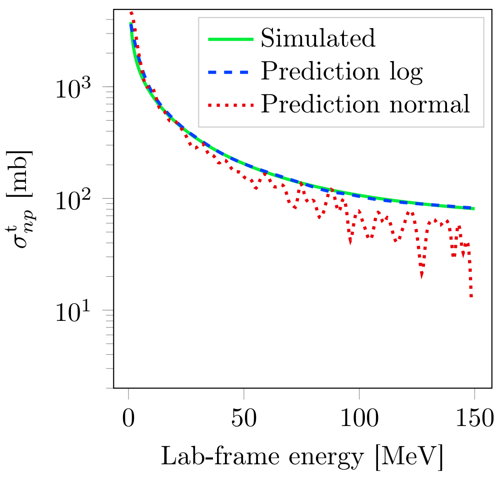
5.2 Energy dependent total cross section

The previously presented measurements are all done for simulations and emulations of the total cross section at a single lab-frame kinetic energy (50 MeV). Since we want our emulator to be able to predict the cross section for arbitrary energies within a specified interval, we introduce an extra energy parameter in addition to the previous 12 LECs. This means that the LHS and GP training has to be performed in a 13 dimensional parameter space, further increasing the difficulty of the emulation problem.

Another difficulty of introducing the kinetic energy as a parameter is the large variation in cross section values as a function of energy. The cross section spans three orders of magnitude across an energy interval from 1 MeV to 150 MeV and it rapidly decreases from its peak value at low energies to a very low, almost constant value at high energies. This makes GP training difficult, since a large number of training points are needed in the low-energy region to correctly emulate this transition. To solve this problem we train GPs using logarithmized cross-section data.

A comparison between GPs trained with and without logarithmized cross-section training-data is shown in figure 13. As seen in the figure, training with logarithmized data improves the accuracy of the prediction for both a sparsely trained GP with 500 training points and a GP trained using 3000 training points. The y-axis is a logarithmic scale to better show the difference in predictive accuracy between the two GPs. We see that normal training data decreases the accuracy of the prediction towards the end of the interval, where the cross section is relatively constant. This effect is more noticeable for the sparsely trained GP. The reason for the shift being downwards and not up, is due to the fact that the GP tends to zero outside of training points, as described in chapter 3. With logarithmized training data the performance gap between the two GPs is reduced. Both predicted curves follow the simulated curves without any major deviations, with a slightly better accuracy for the GP trained using 3000 data points.

In figure 14 the large effects of logarithmized data on the model error is presented for GPs with 250 to 3000 training points in steps of 250. Here, the advantages of logarithmized data can be clearly seen. The GP model errors are calculated with 50 sets of validation LECs data with 300 points each at different energies. In the right panel the cross section data is logarithmized before GP training, while the GPs in the left panel are trained using normal data. The two methods were applied to GPs with both the RBF kernel and the Matérn 5/2 kernel. The use of logarithmized data is seen to decrease the model error for both kernels. Generally, Matérn 5/2 performs better than its RBF counterpart. The irregularities, such as for 1500 samples when a drop in model error is present for the RBF kernel, are likely due to the randomized sampling of training data and not a general feature.



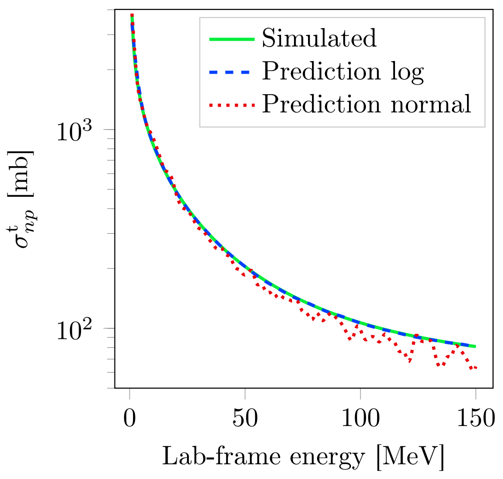
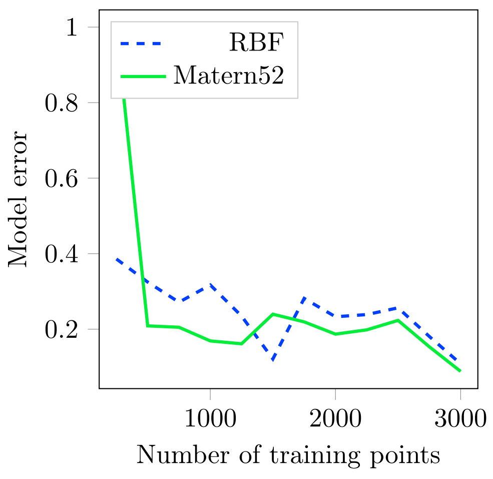


Figure 13: Simulated and emulated cross section curves with a logarithmic scale. The left panel is for GPs trained with 500 LHS sampling points representing a sparse set of training points in the 12-dimensional LEC-volume. The GPs in the right panel are trained using 3000 LHS sample points. The prediction is completed using `nsopt` simulations with 300 data points at different energies. The same prediction set is used in both panels and all GPs are trained using the Matérn 5/2 kernel. The solid line represents the `nsopt` simulated cross sections while the dashed and dotted lines are predictions using logarithmized and normal training data, respectively. The y-axis is logarithmized to better show the difference in predictive accuracy between the two GPs. The sparsely trained GP give worse results for both training data types. The use of logarithmized data improves the results of both GPs greatly reducing the performance gap between the GPs.



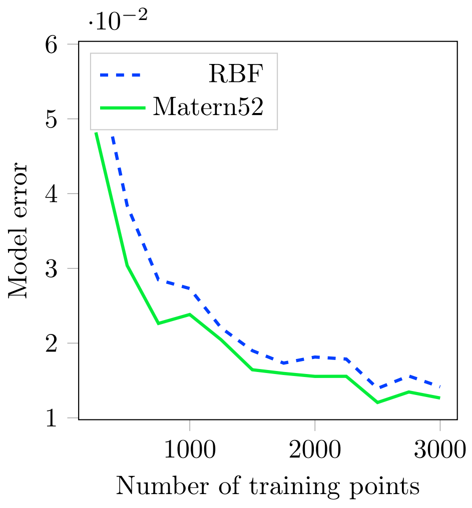


Figure 14: Calculated model error as a function of number of training samples for cross section GPs trained using an energy interval as an extra parameter. The kernels used are RBF and Matérn 5/2. The GPs in the right panel is trained with logarithmic data. As seen, Matérn 5/2 is the better performing kernel. The only exception being two outliers in the left panel, one at 250 training samples and one at 1500.

5.3 Emulating the ^{4}He binding energy

For the emulation of the helium-4 binding energy, $E_b(^4\text{He})$, we consider a fixed χEFT interaction, instead varying the many-body model-space. The data of input parameters considered for this problem is an equally spaced grid of even integers from 2 to 20 for N_{max}, and integers from 5 MeV to 130 MeV for $\hbar\omega$. To calculate the model-error parameter, some of this data is used for training while the rest is used for validation. However, the model error is only calculated with interpolated predictions. The subset of training data is taken by using only a range of N_{max}-values (e.g. $N_{\text{max}} = 16$, 18 and 20 instead of all values form 2 to 20) and using only $\hbar\omega$-values at some specific interval. We specify these subsets of training data by introducing a *training-interval* number: a training interval of four means that every fourth integer value in MeV for $\hbar\omega$ is used for training, starting at $\hbar\omega = 5$ MeV going up to but not above $\hbar\omega = 130$ MeV (i.e. $\hbar\omega = 5\,\text{MeV}, 9\,\text{MeV}, 13\,\text{MeV}, \ \ldots, \ 129\,\text{MeV}$).

Since one of the input parameters for the bound-state problem is discretely valued (N_{max}), we have the option of training several discrete one-dimensional GPs for different values of N_{max} with only $\hbar\omega$ as input parameter. The accuracy of one– and two-dimensional models with different kernels and training interval are compared using the model error. The multi-dimensional kernel option is used for all bound-state models, since the input dimensions operate on different length scales.

The value of the parameter N_{max} determines the number of oscillator states in the expansion we use to approximate the state of the helium-4 nucleus. Increasing the the value of N_{max} makes the approximation more accurate but also makes the simulations more computationally demanding. An accurate prediction of high N_{max}-values based simulated low N_{max}-values would therefore be of great interest.

Attempts to extrapolate N_{max} are shown in figure 15. Here all available training data is used to train a two-dimensional model using the RBF kernel and predictions are made for $\hbar\omega = 10\,\text{MeV}, 30\,\text{MeV}$, and $50\,\text{MeV}$. The variational principle dictates that the binding energy should be decreasing for increasing N_{max}. However, the predictions shown in figure 15 do not behave this way. The predictions of binding energies for $N_{\text{max}} > 20$ are quickly increasing or decreasing before approaching zero. It seems that GPs are not a good tool to make extrapolations in this parameter. The reason for the poor results from the extrapolations might be that GPs in general are not the best approach to make extrapolations of non-periodic functions. As predictions are made further away from the training data, the covariance between training points and predicted points decrease, since we are using stationary kernels. It might be worth mentioning that extrapolations generally are difficult to make, regardless of method and situation. GPs seem to be no exception.

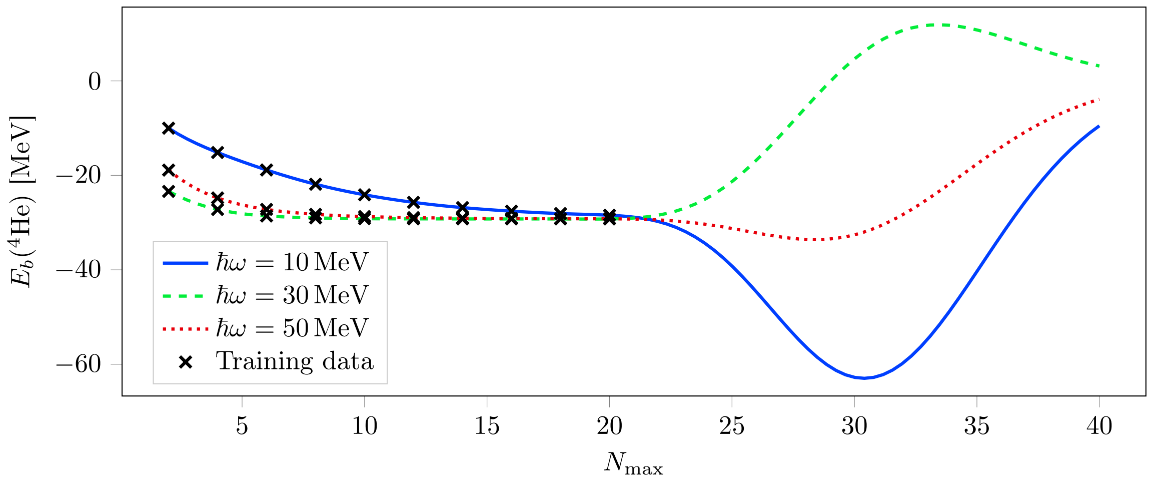


Figure 15: Extrapolation of $E_b(^4\text{He})$ as a function of the many-body model-space parameter N_{max}. Crosses show training data, which ends at $N_{\text{max}} = 20$. All training data from $\hbar\omega = 5$ MeV to 130 MeV with N_{max} from 2 to 20 were used to train this model. The kernel used is RBF. It is known that the binding energy should decrease for increasing N_{max}; it is however apparent that this is not that case for this extrapolation.

The performance of the different kernels is tested by interpolation in $\hbar\omega$ and comparing model-error values. An example showing training data, non-used data and predictions for a two-dimensional model can be seen in figure 16, where $N_{\text{max}} = 16, 18, 20$-results are used for training. For the model in this figure the training interval is four, which means that the model is trained with every fourth value of $\hbar\omega$ while the values in between are used for calculating the model error.

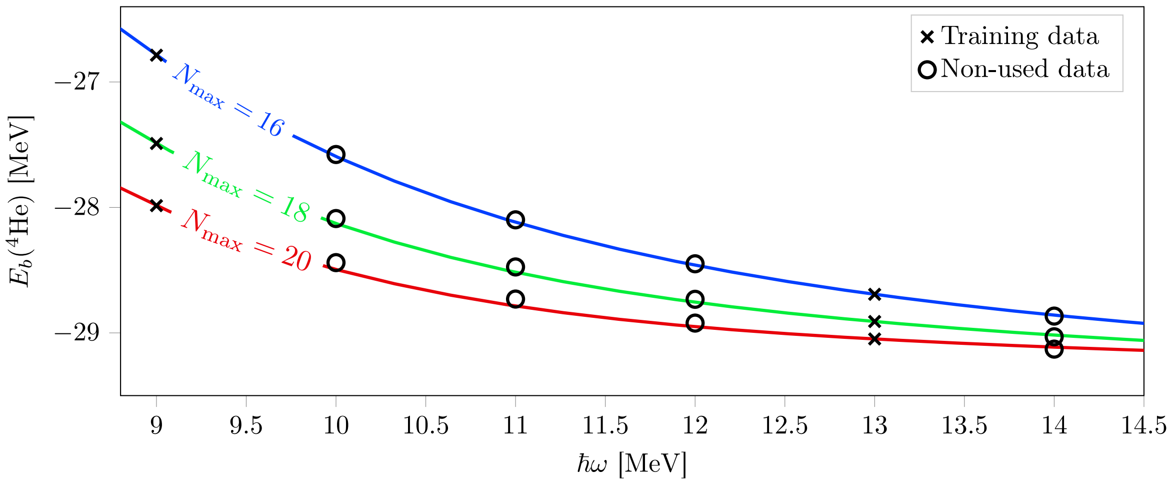


Figure 16: Training data and non-used data together with predictions for a two-dimensional model for emulating $E_b(^4\text{He})$. Although the figure only shows $\hbar\omega$ from 9 MeV to 14 MeV the training data spans from 5 MeV to 130 MeV. Three $N_{\max}$-values are used for training in this case: 16, 18, and 20 as shown in the figure. The training interval is 4, which means the model uses every fourth $\hbar\omega$-value for training. The used kernel is RBF. For this particular model, the model error is 9.58e-04.

Tables showing model errors for different kernels and training intervals can be seen in table 3. Both one– and two-dimensional models are tested. It seems to be clear that the RBF kernel gives the highest accuracy, followed in decreasing order by Matérn 5/2, Matérn 3/2 and exponential for both one– and two-dimensional models. The table also shows that the model error grows with increasing training interval as expected.

Table 3: Model errors for one– and two-dimensional models with different training intervals and kernels. The lowest model error for all training intervals and both one– and two-dimensional models is obtained with the RBF kernel, then in ascending order: Matérn 5/2, Matérn 3/2 and exponential. The models are trained with $N_{\max} = 16, 18,$ and 20.

2D-models

	Training interval		
Kernel	2	4	6
RBF	2.81e−5	9.58e−4	4.77e−3
Exponential	3.44e−3	1.07e−2	1.94e−2
Matern 3/2	1.07e−3	5.69e−3	1.30e−2
Matern 5/2	3.37e−4	3.18e−3	8.50e−3

1D-models

	Training interval		
Kernel	2	4	6
RBF	3.76e−5	8.80e−4	5.26e−3
Exponential	2.98e−3	9.35e−3	1.77e−2
Matern 3/2	1.27e−3	6.23e−3	1.36e−2
Matern 5/2	2.73e−4	3.89e−3	1.05e−2

In table 4, model errors for one– and two-dimensional models using training data with different ranges of $N_{\max}$-values are compared. An $N_{\max}$-range of 12 to 20 means that $N_{\max} = 12, 14, 16, 18, 20$-results are used for training. For these models only the RBF kernel is used. While there are no big differences between the model errors for one– and two-dimensional models, the two-dimensional models seem to perform slightly better. There is no obvious correlation between the $N_{\max}$-ranges and the model errors.

Table 4: Model errors with different $N_{\max}$-ranges for training. The RBF kernel is used for all models. There are no significant differences between one– and two-dimensional models in terms of performance. The two-dimensional models perform slightly better across the board, except for the case with training interval = 6 and $N_{\max}$-range: 8 to 20. For this particular case the two-dimensional model fails to optimize the hyperparameters for the given training set, making the the predictions go to zero between all training points.

2D-models

$N_{\max}$-range		Training interval		
Lower	Upper	2	4	6
16	20	2.81e−5	9.58e−4	4.77e−3
12	20	2.00e−5	9.99e−4	5.86e−3
8	20	2.26e−5	6.51e−4	7.80e−1

1D-models

$N_{\max}$-range		Training interval		
Lower	Upper	2	4	6
16	20	3.76e−5	8.80e−4	5.26e−3
12	20	2.92e−5	9.08e−4	4.38e−3
8	20	2.67e−5	7.78e−4	4.17e−3

Since the input space is two-dimensional, the GP problem is not very computationally demanding compared to the GPs for cross sections. Therefore the time and memory costs are not studied in detail. The simulations with `nsopt` are, however, rather costly. It is only a matter of seconds to optimize the models presented in this section, while it took a couple of hours to simulate the full data set with `nsopt`. For nuclei larger than helium-4, the simulation times could get much longer. This would make repeated calculations of binding energies for specific $\hbar\omega$ and $N_{\max}$-values very time consuming. By training a GP-model it is possible to make the time consuming calculations once, for a general set of training data, and then emulate results for specific $\hbar\omega$-values very quickly.

The data on $E_b(^4\text{He})$ for different $\hbar\omega$ and $N_{\max}$ is relatively well-behaved, without many minima, maxima and quick fluctuations. This is probably a reason that regression with GPs seems to work really well for this problem, giving relatively accurate predictions with many different sets of training data. Even though the functions are well behaved, an analytic expression for the binding energy as a function of $\hbar\omega$ and $N_{\max}$ is not available. This makes the types of curve fitting that tries to fit the function to an analytic expression unsuitable for this problem. GPs do not rely on any analytic function expression, which is a reason why they seem to be a good tool for emulating bound states.

6 Conclusion

The goal of this thesis has been to study the capabilities of GPs for emulating physical observables in χEFT, and to provide guidelines of how GPs can be used to decrease computational costs while still maintaining low-error results. A general observation is that GPs are a powerful tool for regression, and that the GP approach to regression is likely to work well not only in nuclear physics, but also in other fields of research. While the good results from GPs can be accredited interpolated emulations, the same can not be said for extrapolations where the predictions are very inaccurate.

It is clear that there is no perfect set of GP options that works for every kind of problem, as we have shown in our two applications. However, increasing the number of training points will almost exclusively give better predictions. Balancing the gains of more training points to the time and memory that is required to perform the training is the key to a good model.

For sampling of input parameters in many dimensions (as with the cross-section emulation), LHS provides an evenly filled space with little to no bias towards specific areas. This allows the GP to give consistent results over the entire parameter volume. With a few-dimensional input space it might be feasible to use a uniformly spaced training set with good results.

The choice of kernel also affects the performance of the GP significantly. We found that Matérn 5/2 and RBF are the best performing kernels for our many-dimensional problem with cross sections, while RBF alone gave the best results for our few-dimensional problem with binding energies. Matérn 5/2, however, needs more training time than RBF. The time difference between the two kernels increases when we use higher numbers of training data, especially with the multidimensional length scale setting.

We found that the usage of multiple length scales improved performance. Different input dimensions influence the observable to different degrees, and the prediction will perform better with length scales optimized for their specific direction. The cost of this improvement is increased complexity of the optimization problem, which increases the time spent training the model. We conclude with the general statement that this option is often worth the added training time.

6.1 Outlook

Further work on the subject could include studies of more complex observables. The data on $\sigma_{\rm np}^{\rm t}$ and $E_b(^4\rm He)$ are generally well-behaved functions. Other observables, such as binding energies of heavier nuclei or differential cross sections may have more irregular behavior, which can be harder to emulate. The methods to apply GPs to more difficult functions could also be developed, similarly to how we used the logarithm to improve predictions.

We focus on four different stationary kernels. There are a huge number of possible kernels, and `GPy` provides a framework to combine them arithmetically. This is a possible route for further development on the subject.

References

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive computation and machine learning. The MIT Press, 2006, ISBN: 0-262-18253-X.

[2] R. Machleidt and D. R. Entem, "Chiral effective field theory and nuclear forces", *Physics Reports*, vol. 503, no. 1, pp. 1–75, Jun. 2011, ISSN: 0370-1573. DOI: 10.1016/j.physrep.2011.02.001.

[3] B. D. Carlsson, A. Ekström, C. Forssén, D. Fahlin Strömberg, G. R. Jansen, O. Lilja, M. Lindby, B. A. Mattsson, and K. A. Wendt, "Uncertainty analysis and order-by-order optimization of chiral nuclear interactions", *Physical Review X*, vol. 6, no. 1, p. 011 019, Feb. 2016. DOI: 10.1103/physrevx.6.011019.

[4] A. Ekström, G. Baardsen, C. Forssén, G. Hagen, M. Hjorth-Jensen, G. R. Jansen, R. Machleidt, W. Nazarewicz, T. Papenbrock, J. Sarich, and S. M. Wild, "Optimized chiral nucleon-nucleon interaction at next-to-next-to-leading order", *Physical Review Letters*, vol. 110, p. 192 502, 19 May 2013. DOI: 10.1103/PhysRevLett.110.192502.

[5] B. D. Carlsson, "The data perspective on chiral effective field theory", PHD thesis, Department of Physics, Subatomic and Plasma Physics, Chalmers University of Technology, Gothenburg, 2017, ISBN: 978-91-7597-530-6. [Online]. Available: http://publications.lib.chalmers.se/records/fulltext/248612/248612.pdf (visited on 05/10/2017).

[6] A. Pich, "Effective field theory", in *Probing the standard model of particle interactions. Proceedings, Summer School in Theoretical Physics, NATO Advanced Study Institute, 68th session, Les Houches, France, July 28-September 5, 1997. Pt. 1, 2*, 1998, pp. 949–1049. arXiv: hep-ph/9806303 [hep-ph].

[7] W. Rindler, *Relativity, Special, General and Cosmological.* Oxford University Press, 2006, ISBN: 978-3-540-07970-5.

[8] K. S. Krane, *Introductory nuclear physics.* Wiley, 1988, pp. 392–401.

[9] P. Navrátil, G. P. Kamuntavičius, and B. R. Barrett, "Few-nucleon systems in a translationally invariant harmonic oscillator basis", *Physical Review C*, vol. 61, p. 044 001, 4 Mar. 2000. DOI: 10.1103/PhysRevC.61.044001.

[10] GPy, *GPy: a gaussian process framework in python*, since 2012. [Online]. Available: http://github.com/SheffieldML/GPy (visited on 02/08/2017).

[11] Y. Fei, G. Rong, B. Wang, and W. Wang, "Parallel L-BFGS-B algorithm on GPU", *Computers & Graphics*, vol. 40, pp. 1–9, May 2014, ISSN: 0097-8493. DOI: 10.1016/j.cag.2014.01.002.

[12] M. D. Shields and J. Zhang, "The generalization of latin hypercube sampling", *Reliability Engineering & System Safety*, vol. 148, pp. 96–108, 2016, ISSN: 0951-8320. DOI: 10.1016/j.ress.2015.12.002.

[13] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code", *Technometrics*, vol. 21, pp. 239–245, 2 May 1979. DOI: 10.2307/1268522.

[14] pyDOE, *pyDOE: design of experiments for python*, since 2013. [Online]. Available: https://pythonhosted.org/pyDOE/index.html (visited on 02/15/2017).

[15] B. R. Barrett, P. Navrátil, and J. P. Vary, "Ab initio no core shell model", *Progress in Particle and Nuclear Physics*, vol. 69, pp. 131–181, Mar. 2013, ISSN: 0146-6410. DOI: 10.1016/j.ppnp.2012.10.003.