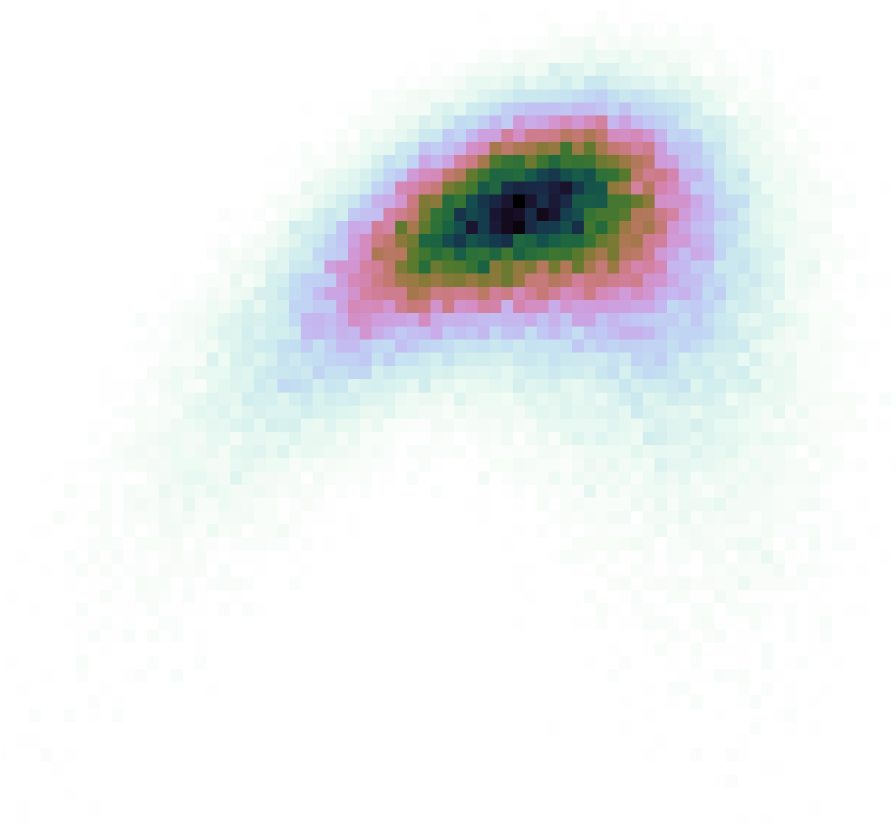
CHALMERS
UNIVERSITY OF TECHNOLOGY



Chiral effective field theory with machine learning

Bachelor of Science Thesis for the Engineering Physics Program

JOHANNES ASPMAN, EMIL EJBYFELDT,
ANTON KOLLMATS, MAXIMILIAN LEYMAN

Department of Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

BACHELOR OF SCIENCE THESIS FOR THE ENGINEERING PHYSICS PROGRAM

Chiral effective field theory with machine learning

Johannes Aspman
Emil Ejbyfeldt
Anton Kollmats
Maximilian Leyman

Department of Physics
Division of Subatomic and Plasma Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Chiral effective field theory with machine learning
Johannes Aspman[a], Emil Ejbyfeldt[b], Anton Kollmats[c], Maximilian Leyman[d]

Email:
[a]aspmanj@student.chalmers.se
[b]emilej@student.chalmers.se
[c]kollmats@student.chalmers.se
[d]gusleymma@student.gu.se

Bachelor thesis at the Department of Physics, Chalmers
Bachelor's thesis TIFX04-16-04

Supervisor: Andreas Ekström, Christian Forssén
Examiner: Daniel Persson

Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone: +46 31-772 10 00

Cover:
Joint probability distribution of the binding energy of the alpha particle and the radius of the deuteron, see Chapter 5.

Chalmers Reproservice
Gothenburg, Sweden 2016

Abstract

Machine learning is a method to develop computational algorithms for making predictions based on a limited set of observations or data. By training on a well selected set of data points it is in principle possible to emulate the underlying processes and make reliable predictions.

In this thesis we explore the possibility of replacing computationally expensive solutions of the Schrödinger equation for atomic nuclei with a so-called Gaussian process (GP) that we train on a selected set of exact solutions. A GP represents a continuous distribution of functions defined by a mean and a covariance function. These processes are often used in machine learning since they can be made to emulate a wide range of data by choosing a suitable covariance function.

This thesis aims to present a pilot study on how to use GPs to emulate the calculation of nuclear observables at low energies. The governing theory of the strong interaction, quantum chromodynamics, becomes non-perturbative at such energy-scales. Therefore an effective field theory, called chiral effective field theory (χEFT), is used to describe the nucleon-nucleon interactions.

The training points are selected using different sampling methods and the exact solutions for these points are calculated using the research code `nsopt`. After training at these points, GPs are used to mimic the behavior of `nsopt` for a new set of points called prediction points. In this way, results are generated for various cross sections for two-nucleon scattering and bound-state observables for light nuclei.

We find that it is possible to reach a small relative error (sub-percent) between the simulator, i.e. `nsopt`, and the emulator, i.e. the GP, using relatively few training points.

Although there seems to be no obvious problem for taking this method further, e.g. emulating heavier nuclei, we discuss some areas that need more critical attention. For example some observables were difficult to emulate with the current choice of covariance function. Therefore a more thorough study of different covariance functions is needed.

Sammandrag

Maskininlärning är en metod för att utveckla beräkningsalgoritmer som gör prediktioner från en begränsad mängd observationer eller data. Genom att träna på en mängd väl valda datapunkter är det möjligt att emulera de underliggande processerna och göra pålitliga prediktioner.

I denna rapport undersöker vi möjligheterna att ersätta beräkningstunga lösningar till Schrödingerekvationen för atomkärnor med en så kallad gaussisk process (GP) som vi tränar på en vald mängd exakta lösningar. En GP representerar en kontinuerlig distribution av funktioner som definieras av ett medelvärde och en kovariansfunktion. Dessa processer används ofta inom maskininlärning då de kan emulera en mängd olika typer av data genom att välja en passande kovariansfunktion.

Denna rapport syftar till att vara en introduktionsstudie i användandet av GP:s för att emulera beräkningar av observabler inom kärnfysiken. Den ledande teorin för den starka kraften, kvantkromodynamik, blir icke-perturbativ vid de låga energier som är relevanta för kärnfysiken, och därför används istället en effektiv fältteori, kallad kiral effektiv fältteori (χEFT), för att beskriva nukleon-nukleon växelverkan.

Träningspunkterna väljs med hjälp av olika metoder och de exakta lösningarna i dessa punkter beräknas med forskningskoden `nsopt`. Efter att ha tränat på dessa punkter användes GP:s till att emulera beteendet hos `nsopt` för en ny uppsättning punkter som kallas för prediktionspunkter. Resultat tas sedan fram för olika tvärsnitt för tvånukleonspridning samt för bundna tillstånd för lätta atomkärnor.

Vi finner att det är möjligt att nå små relativa fel (sub-procent) mellan simulatorn, dvs `nsopt`, och emulatorn med relativt få träningspunkter.

Även om det inte finns några uppenbara hinder för att utnyttja denna metod för mer komplexa system, som tyngre atomkärnor, diskuterar vi vissa områden som behöver mer uppmärksamhet. Exempelvis var vissa observabler svåra att emulera med den använda kovariansfunktionen. En mer genomgående studie av olika kovariansfunktioner skulle därför behövas.

Acknowledgements

We would like to thank our supervisors Christian Forssén and Andreas Ekström for their support and guidance during this project. We would also like to express our appreciation to Boris Carlsson for his support in questions regarding `nsopt`.

The Authors, Gothenburg, May 2016

Contents

Chapter 1

Introduction

Modern day physics very often require large amounts of computational power. This is due to the fact that you often want to solve equations depending on many parameters, resulting in large systems of equations. Such calculations often become time-consuming despite the computational capacity of modern computers.

One way to circumvent this problem is by using a method called machine learning. In the machine learning process a computer is allowed to train on a limited amount of data points and then use this training to predict the outcomes of similar processes. The goal is to replace the original simulation of the process with an emulator with reduced computational complexity and thereby drastically reduce the time of computation.

Machine learning is a vast scientific field with a wide range of applications. In this thesis, we will focus on an aspect of machine learning called regression analysis, and more specifically on the Gaussian process model of regression analysis. This model uses the mean and covariance between the training points to estimate the targeted function, called the simulator.

Earlier studies have shown that machine learning can be used to increase the effectiveness of calculations within fields such as cosmology, many-body physics as well as nuclear physics [1, 2, 3]. In this project, we will examine whether it is possible to utilize machine learning to predict cross sections in nucleon-nucleon scattering, as well as propagating the statistical uncertainties in the description of the nuclear interaction to bound-state energies of few-nucleon systems.

We will use an effective description of the strong force for nuclear physics, called chiral effective field theory (χEFT), to describe the nucleon-nucleon interaction. χEFT uses an expansion of terms proportional to $(Q/\Lambda)^\nu$ to describe the force between the nucleons, where Q and Λ are parameters representing the soft and the hard energy scales of the effective theory. The number $\nu = 0, 1, 2\ldots$ determines the size of the term and is called the chiral order. Since the expansion contains an infinite number of terms a truncation is made whereby the higher order terms are omitted. The most important terms in χEFT have $\nu = 0$ and are labeled the leading order (LO). The next-to-leading order (NLO) have $\nu = 2$, since $\nu = 1$ is forbidden for symmetry reasons, and so on [4].

1.1 Purpose and aims

This thesis is an introductory study on the effects of using Gaussian processes to emulate observables of nuclear physics. The aim is to give an overview of both the benefits and the costs of this model, as well as to present some possible directions for further studies.

The present study is guided by the following three research questions:

- Is it possible to implement an emulator capable of learning from the exact solutions of the nuclear Schrödinger equation and from this make reliable predictions for the nuclear many-body problem?
- For which levels of complexity does this emulator maintain its effectiveness?
- What are the limitations of using machine learning to predict physical systems?

1.2 Method and limitations

Our initial studies were mainly directed towards reading literature about practical examples of the use of machine learning in physics as well as on the underlying theory of Gaussian processes. The theory behind χEFT was only given minor attention. Most of the work was carried out in Python, using Scikit-learn 0.14.1, where the code for our emulator was written [5]. To gather training data for our emulator we used the simulator `nsopt`. This is a research code for obtaining numerically exact solutions of the few-nucleon Schrödinger equation.

We focus on calculating observables at NLO in the expansion of χEFT. The machine learning implemented in the emulator is limited to regression by Gaussian processes, with only the squared exponential used as covariance function. As for the physics, we will restrain ourselves to training our emulator to extrapolate observables associated with bound-states for less than four nucleons and nucleon-nucleon scattering cross sections, with the ability to propagate errors.

1.3 Structure of the thesis

Chapters 2 to 4 present the theoretical basis of the thesis. The Gaussian processes which make up the foundation for this thesis are given a basic description in Chapter 2. Chapter 3 presents an introduction to effective theories in general and outlines the χEFT and the physical equations relevant to our computational models. In Chapter 4 we present the sampling methods used to generate relevant data points for the Gaussian processes.

Our results for the emulation of scattering and bound-state observables are presented in Chapter 5, and these are given a more general discussion in Chapter 6. The thesis is concluded in Chapter 7 with a summary and an outlook for further research.

Chapter 2

Gaussian processes for machine learning

In this chapter a short introduction to the theory of Gaussian processes is presented, and a discussion is given on how these are implemented in machine learning. The statistical formulation of regression analysis is presented, along with a discussion regarding the role of the covariance function. The chapter is concluded with an example where a Gaussian process is used in regression analysis of a simple mathematical function.

2.1 Gaussian process

A collection of normally distributed random variables is known as a Gaussian process (GP) [6]. There exist several equivalent interpretations of the GP, of which we shall adapt the so called function-space-view.

In the function-space-view, the GP represents a distribution over functions, which is denoted as

$$f(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')), \tag{2.1}$$

where the function f is a random variable specified by an argument $\boldsymbol{x}$. In general, a GP is defined by its mean- and covariance functions, which are denoted as $m(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}')$, respectively. Thus, the above distribution is effectively a distribution over random functions compliant with the specified correlations.

In machine learning, the GP is used as a tool for regression analysis by eliminating functions not coherent with observed data, which for a GP is known as the training data. The method by which non-interesting distributions are removed is probabilistic conditioning [6].

2.2 Regression analysis with Gaussian processes

Suppose F is a function of $\boldsymbol{x}$ whose explicit evaluation is computationally cumbersome, making direct evaluation on its entire domain highly inefficient. By evaluating the function on a small, but carefully selected, subset of its domain one can try to reduce the computational complexity by emulating F with a GP thus hopefully enabling more efficient, but less accurate, predictions to be made on its entire domain.

These predictions are made by extrapolating the values assumed by F in the vicinity of a few sampled points, referred to as training points. There is no general method for choosing the best training points, it is an optimization problem of its own [6]. The points for which F is to be predicted are henceforth referred to as the prediction points, which represent another subset of the domain of F. The set of $\boldsymbol{x}$-values specifying the prediction- and training points shall be denoted D, for reference in this chapter.

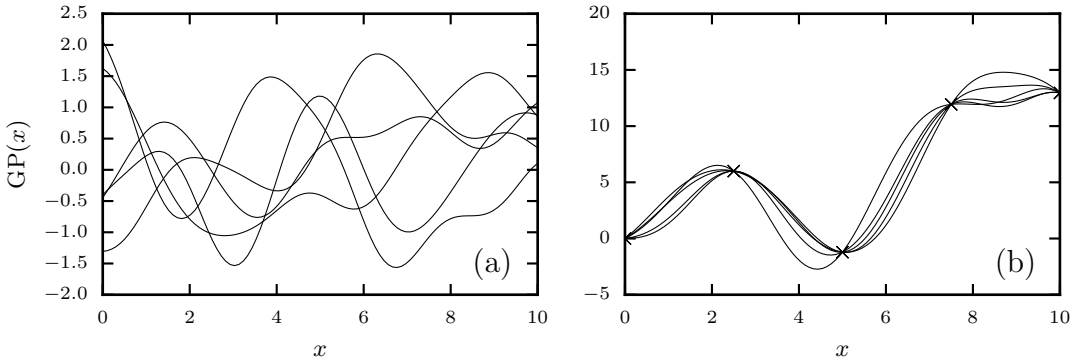


Figure 2.1: Black lines represent samples drawn from the prior distribution (a) and from the posterior distribution with 5 training points, marked by × (b) of a Gaussian process with a squared exponential covariance function.

As mentioned in the beginning of this chapter, a GP is specified by its mean- and covariance functions, $m(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}')$. One general property that these functions share is that they must be defined for all values of D. While it is often the case that the mean can be set identically to zero, the choice of covariance function is not as straight forward. As the purpose of the sought GP is to replicate the behavior of the real function, F, it is necessary to select a covariance function that has characteristics similar to those of the original function. For this reason, as was the case with the selection of training points, the procedure of finding a suitable covariance function is heavily dependent on the nature of F (see Section 2.3 for a more elaborate description).

The specified mean- and covariance functions span a distribution of functions (called the prior distribution), as in Equation (2.1) and it is the aim of GP modeling to use the training points to extract the set of functions that are most similar to F (called the posterior distribution). A schematic picture of such a process is shown in Figure 2.1. Every function specified by the GP represents a particular joint event of a random variable that corresponds to a point $\boldsymbol{x}$ in D. It is then a matter of probabilistic conditioning to single out the functions coherent with the training points. This results in a set of Gaussian distributions for to every $\boldsymbol{x}$ in D [6].

It is convenient to arrange the arguments and the function values of the training points in matrices, such that

$$X_t = \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{n_t} \end{bmatrix}, \qquad Y_t = \begin{bmatrix} F(\boldsymbol{x}_1) \\ F(\boldsymbol{x}_2) \\ \vdots \\ F(\boldsymbol{x}_{n_t}) \end{bmatrix}, \tag{2.2}$$

where n_t is the number of training points.

Similarly, we arrange the points $\boldsymbol{x}_{n_t+1}, \boldsymbol{x}_{n_t+2}, \ldots \boldsymbol{x}_{n_t+n_p}$ for which the function will be predicted in a matrix X_p, where n_p denotes the number of prediction points. Finally, we introduce the correlation matrix, K, defined such that its matrix element $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ where $i, j \in \{1, 2, \ldots, n_t+n_p\}$. The correlation matrix becomes

$$K = \begin{bmatrix} K_{X_t,X_t} & K_{X_t,X_p} \\ K_{X_p,X_t} & K_{X_p,X_p} \end{bmatrix}, \tag{2.3}$$

where, for instance, $K_{X_pX_t}$ defines a submatrix of K whose entries corresponds to the points defined by the elements of X_p and X_t. Defining two column vectors, M and Σ^2, to contain the mean and variances for each $\boldsymbol{x}$ in the posterior distribution in D, it can be shown that [6]

$$M = K_{X_p,X_t} K^{-1}_{X_t,X_t} Y_t \tag{2.4}$$

$$\Sigma^2 = K_{X_p,X_p} - K_{X_p,X_t} K^{-1}_{X_t,X_t} K_{X_t,X_p}. \tag{2.5}$$

The value of $F(\boldsymbol{x})$, where $\boldsymbol{x}$ is in X_p, is then estimated by the predicted mean value with its corresponding variances, extracted from the matrices as

$$\mu(\boldsymbol{x}) = M_{\boldsymbol{x}} \tag{2.6}$$

$$\sigma^2_{\text{std}}(\boldsymbol{x}) = \left(\Sigma^2\right)_{\boldsymbol{x}}. \tag{2.7}$$

2.3 Covariance functions

When modeling a function, F, that can be quite complicated, the objective is to capture its features as well as possible, using a minimal amount of training data extracted from the function itself. The manner in which a GP model accomplishes this is by assuming that adjacent training points exhibit similar behavior, and then extrapolate accordingly. In modeling arbitrary functions, this opens up the freedom to define adjacency by means of a covariance function. As previously established in this chapter, the adjacency measure is a function of the form $k(\boldsymbol{x}, \boldsymbol{x'})$.

In theory, any function that is positive semidefinite can be used as a covariance function. In practice, the covariance function is best chosen ad hoc to fit a particular problem at hand. Often, this choice is partially done by means of optimizing a set of so-called hyperparameters included in the function. For some problems, one may wish to combine behaviors of several previously known covariance functions. It can be shown that the addition or multiplication of two covariance functions yield another valid covariance function. Similarly, one can produce a new function by taking the convolution of two covariance functions [6].

One of the most frequently used covariance functions, and also the covariance function used throughout this thesis, is the squared exponential, defined as

$$k(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\frac{|\boldsymbol{x}-\boldsymbol{x'}|^2}{2l^2}}. \tag{2.8}$$

This function has the hyperparameter l, which can be interpreted as a characteristic length scale of the function F. When modeling smooth functions the squared exponential covariance function is suitable, since functions drawn from a GP with it are infinitely differentiable.

2.4 Optimization of the hyperparameters

The likelihood is the probability density of the observations given the parameters [6]. The marginal likelihood $p(Y_t|X_t)$ is defined as the integral of the likelihood times the prior, where the prior is a probability distribution answering to our prior knowledge of the parameters. If this knowledge is weak the prior distribution is broad and vice versa.

Using that Y_t is normally distributed, with a zero-mean and a covariance $K_{X_tX_t}$, and performing the integration it can be shown that the log marginal likelihood can be calculated as [6]

$$\log p(Y_t|X_t) = -\frac{1}{2}Y_t^\top K_{X_tX_t}^{-1} Y_t - \frac{1}{2}\log|K_{X_tX_t}| - \frac{n_t}{2}\log 2\pi. \tag{2.9}$$

The log marginal likelihood is used to give a measure of how good the model fits the training points. Since the log marginal gives a value of how good a covariance function is it can be used to compare different models. But given a covariance function that is dependant on one or more hyperparameters you can formulate a maximization problem of the log marginal likelihood. Solving this problem should give you the best hyperparameters for the given training points.

2.5 A basic example

For demonstration purposes we will emulate a known one-dimensional target function, which has been chosen arbitrarily as

$$F(x) = x + x(\sin x - \cos x). \tag{2.10}$$

The training data consists of five points, $n_t = 5$, whose x-values are uniformly sampled from the interval $x \in [0, 10]$. Let, as in (2.2), X_t and Y_t denote column vectors whose coordinates are specified by the values of x and their corresponding samples F and we get

$$X_t = \begin{bmatrix} 1 \\ 3 \\ 5 \\ 7 \\ 9 \end{bmatrix}, \qquad Y_t = \begin{bmatrix} 1.30 \\ 6.39 \\ -1.21 \\ 6.32 \\ 20.91 \end{bmatrix}. \tag{2.11}$$

Inferring from these training points, we will predict the values of F corresponding to 100 uniformly spaced x-values in the same interval as above, similarly arranged in a column vector X_p.

The x-values are used for calculation of the correlation matrix, specified by the squared exponential covariance function of Equation (2.8). For the one-dimensional case, with $l = 0.5$ to help illustrate how the hyperparameters affects the prediction, the covariance function is

$$k(x, x') = e^{-2(x-x')^2}. \tag{2.12}$$

Using the covariance function the correlation matrix of the training points is calculated

$$K_{X_t, X_t} = \begin{bmatrix} 1 & 1.06\text{e}-1 & 1.26\text{e}{-4} & 1.68\text{e}{-9} & 2.51\text{e}{-16} \\ 1.06\text{e}{-1} & 1 & 1.06\text{e}{-1} & 1.26\text{e}{-4} & 1.68\text{e}{-9} \\ 1.26\text{e}{-4} & 1.06\text{e}{-1} & 1 & 1.06\text{e}{-1} & 1.26\text{e}{-4} \\ 1.68\text{e}{-9} & 1.26\text{e}{-4} & 1.06\text{e}{-1} & 1 & 1.06\text{e}{-1} \\ 2.51\text{e}{-16} & 1.67\text{e}{-9} & 1.26\text{e}{-4} & 1.06\text{e}{-1} & 1 \end{bmatrix}. \tag{2.13}$$

The hyperparameter l is now optimized by maximizing the log marginal likelihood in Equation (2.9). A maximum is found for $l = 1.75$, which will be considered the optimal value of l.

Now the full correlation matrix is calculated using the optimal l. The matrix is then decomposed into submatrices according to (2.3), and the mean and variances for the posterior distribution are derived from (2.4) and (2.5), finally $\mu(x)$ and $\sigma^2_{\text{std}}(x)$ are calculated according to (2.6) and (2.7). The standard deviation function $\sigma_{\text{std}}(x)$ has been obtained by taking the positive square root of the variance function. Figure 2.2 illustrates the prediction for both the initial value of $l = 0.5$ and the optimal $l = 1.75$. The dashed green curve is $\mu(x)$, the blue curve is the true target function, $F(x)$, and the shaded area indicates the $2\sigma_{\text{std}}$ (95%) confidence interval as deduced from the standard deviation.

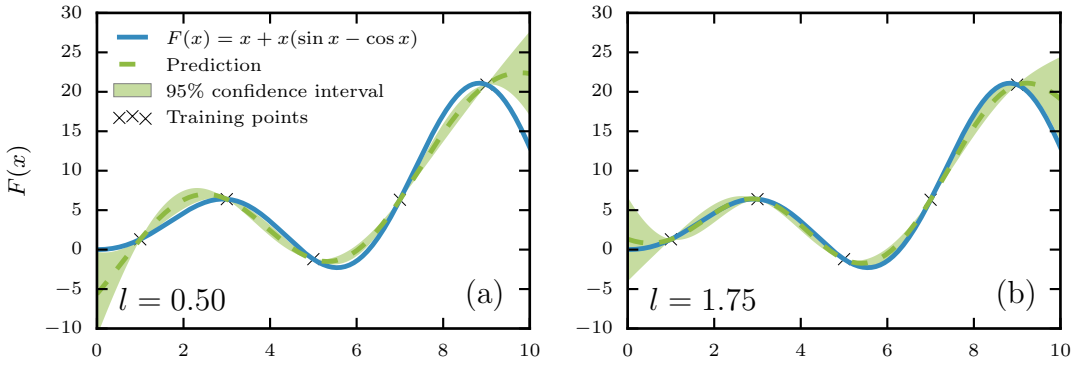


Figure 2.2: Example of a Gaussian process used in predicting the function (2.10). Panel (a) shows the prediction with a guessed value of the hyperparameter $l = 0.5$ and panel (b) shows the prediction for the optimal value of $l = 1.75$.

Chapter 3

Underlying theory of the simulator

This chapter serves to give an introduction to the physical theories that make up the backbone of this thesis. A brief introduction to effective theories is given and the usefulness of χEFT is established. Next, its connection to scattering theory is made. Finally, an introduction to bound states is provided.

3.1 Effective theories

It is widely known that the physical phenomena present at the various scales of the universe appear quite distinct from each other. While, in principle, a theory of everything would be sufficient to make predictions on any scale, in practice it is often necessary to isolate a set of phenomena and study them without considering the effects of irrelevant physics. By diverting attention only to the relevant degrees of freedom of a physical system, calculations can often be made significantly simpler, increasing the ability to make predictions. The practice of isolating certain parts of physics, relevant only at certain scales, has resulted in so-called effective theories, as shall be further illustrated below.

The principle of an effective theory can be illustrated by the familiar correspondence of relativistic- and Newtonian mechanics. In inertial frames moving with constant velocities relative to one another the relationship between momentum and velocity of an object is a non-linear one. If one assumes that the object is moving slowly in comparison to the speed of light, one may expand its momentum in terms of a power series with rapidly vanishing terms. For many practical applications, this assumption is plausible and one may simply truncate the series, for reasons of computational simplicity, and yet retain a valid description of the physics observed at this particular velocity scale. By discarding all but the leading-order term (LO), one obtains the classic expression for momentum, for which only the first power of velocity is a relevant degree of freedom. Naturally, this introduces a systematic error of the model. However, while a more accurate description of the physics is obtained by the inclusion of higher order terms (referred to as expansions of next-to-leading-order (NLO), next-to-next-to-leading-order (NNLO), and so on) they are generally not needed when dealing with classical mechanics [7].

Similarly, one may obtain a relatively simple description of low-energy nuclear forces by means of a series expansion.

3.1.1 Chiral effective field theory

Strong nuclear forces are described by the theory of quantum chromodynamics (QCD). By means of a series expansion, low-energy nuclear forces can be given a relatively simple description in terms of an expansion parameter, Q/Λ. The quantities Q and Λ are energies whose magnitudes are of the same order as the rest energies of the pion and rho meson, respectively. Thus, Λ defines the energy scale for which the expansion is valid. This limiting case of QCD is called chiral effective field theory, χEFT, and is the physical theory underlying this thesis [4, 8].

In the same manner as with the case of classical- and relativistic mechanics, one obtains an infinite sum of terms proportional to $(Q/\Lambda)^\nu$, where ν is referred to as the chiral order of the term.

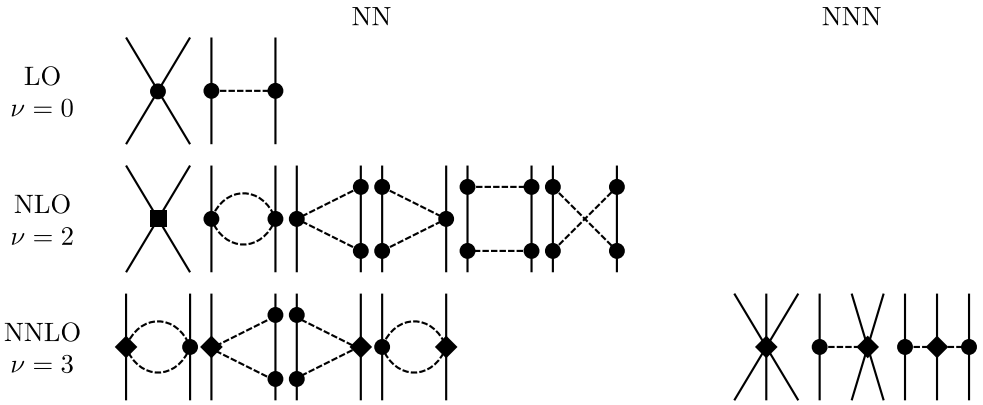


Figure 3.1: Schematic figure showing Feynman diagrams representing nucleon-nucleon interactions. The horizontal orientation classify the diagrams by the number of nucleons involved, while the vertical direction specify the order of the term. Solid lines represent nucleons and dashed lines pions. Circles, diamonds and squares represent the chiral order of the interaction. The order $\nu = 1$ does not contain any new diagrams because of parity- and time-symmetries. At the order $\nu = 3$ (NNLO), diagrams for three-nucleon interactions (NNN) are also present.

Each term can be interpreted to represent a finite set of interaction diagrams, known as Feynman diagrams. Shown in Figure 3.1 are Feynman diagrams representing nucleon-nucleon interactions. The diagrams are arranged schematically with the vertical orientation representing the order of the term associated with the diagrams, while the horizontal orientation classify the diagrams by the number of nucleons involved. Two-nucleon interactions are denoted as NN and three-nucleon interactions by NNN. As for the diagrams themselves, the lines represent particles propagating through space-time, with solid lines for nucleons and dashed lines for pions. Circles, diamonds and squares represent the chiral order of the interactions and their corresponding coupling constants, a measure of the relative strengths among the interactions [9, 10].

As seen in Figure 3.1, several diagrams correspond to a single term in the expansion, where each term itself is given by the sum of the contributions from short-range interactions as well as the exchange of a number of pions. For reasons of parity- and time-symmetry terms with powers $\nu = 1$ vanish, resulting in the inclusion of terms with $\nu \leq 2$ in the NLO-expansion. As also indicated in Figure 3.1, diagrams corresponding to three-nucleon interactions are not present until an expansion up until NNLO, which is to be expected as three-nucleon interactions are significantly weaker than that of two-nucleons. The effective field theory is obtained by discarding the weak, higher order terms, completely analogous to the truncation of the relativistic momentum expansion in the example above.

3.1.2 Low-energy parameters and their statistical errors

Effective field theories are governed by certain parameters, referred to as low-energy constants (LECs) in the case of χEFT for nuclear physics. The dimension of the parameter space is dictated by the chiral order of the expansion. For instance at LO and NLO there are 2 and 9 LECs respectively. The parameters are determined by means of fitting model predictions to experimental data. Effectively, this amounts to determination of the coupling coefficients mentioned in Section 3.1.1. In our case, the data stem from experiments in nucleon-nucleon scattering, where the measured observable is scattering cross sections (see Section 3.2 for details) [9].

Since the LECs are determined from experimental data, which always exhibit a certain uncertainty, the LECs themselves inherit some flexibility in their determined values. This uncertainty is referred to as the statistical error of the model.

The LECs also exhibit a certain dependence on each other due to the fact that they describe the same physics. So if the value of one is changed the others will also be changed. This results in the need to use their covariance matrix when propagating the statistical errors (for further details

see Section 4.2).

3.2 Scattering theory

Most of this thesis revolves around the computation of one specific type of observable: nuclear cross sections, denoted σ. It is a measure of how many of the incident particles of a beam are scattered from a target particle, and experimental measurements of the cross section have been used to determine the size and shape of the atomic nucleus [11]. For convenience we will neglect the spin-dependence of the nuclear force in this section.

If you know the flux of incoming particles and measure the amount of outgoing scattered particles you can calculate the probability scattering. That probability corresponds to the cross section, σ. The rate at which a reaction takes place, W_r, is proportional to the flux of incoming particles, J, and the number of particles in the target, N, hit by the beam.

$$W_r = JN\sigma \tag{3.1}$$

Usually what is measured is just a differential reaction rate, $\mathrm{d}W_r$, since the detectors commonly used do not measure all the scattered particles,

$$\mathrm{d}W_r = JN\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}\,\mathrm{d}\Omega \tag{3.2}$$

where $\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}$ is the differential cross section and $\mathrm{d}\Omega = \sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi$ is the differential solid angle subtended by the detector as shown in 3.2. In order to obtain the total cross section you have to integrate the differential cross section over all angles [11],

$$\sigma_{\mathrm{tot}} = \int_0^{2\pi}\mathrm{d}\phi\int_0^{\pi}\sin\theta\,\mathrm{d}\theta\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}. \tag{3.3}$$

If one would use a beam of polarized particles and/or a polarized target particle the situation becomes more complex. The polarization is given as the expectation value of the spins of all particles in the target or beam and can be changed by rotating the spins of the particles using a magnet.

The cross section of this scattering will show azimuthal asymmetry from which the transverse component of the polarization can be determined. By rotating the spin it is also possible to determine the longitudinal part of the polarization [12]. Due to the combinations of polarized, unpolarized beams and targets there are many different types of cross sections, for some examples of this see Figure 3.3. In this thesis we will focus on calculating only three different cross sections, selected because of their varying degrees of complexity. These are the total cross section, σ_{tot}, a differential cross section where both target and beam are unpolarized, $\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}$, and a cross section where the beam and target are polarized in the x-z plane, A_{zx}, as shown in Figure 3.3 [12].

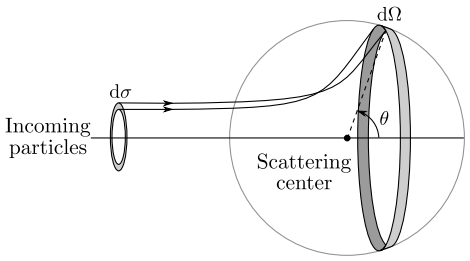


Figure 3.2: An illustration of a differential cross section that possess rotational symmetry about the incoming axis. The differential cross section is often represented in spherical coordinates with the origin in the scattering center.

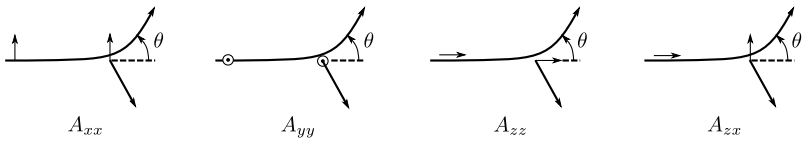


Figure 3.3: Scattering experiments with different combinations of polarizations for beam and target. Polarization is represented with an arrow and ⊙ is an arrow out of the page.

The total cross section depends on the energy of the incoming beam, while being independent of the angle, θ, between the beam and target, while the differential cross sections, $\frac{\mathrm{d}\sigma}{\mathrm{d}\Omega}$ and A_{zx}, are dependent on both energy and θ. Figure 3.4 shows these cross sections calculated with `nsopt` in NLO. A_{zx} assumes negative values in Figure 3.4 for some values of the energy and θ. This is because it is normalized against the unpolarized differential cross section.

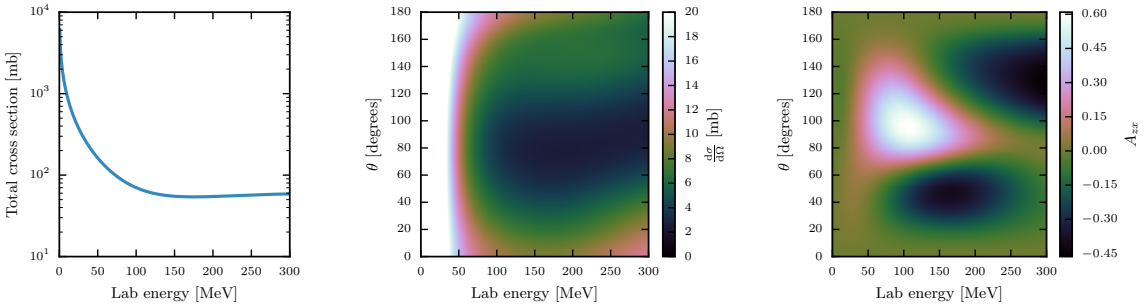


Figure 3.4: The three different cross sections studied in this thesis, calculated with `nsopt` for the LEC values given in Table 4.1 for NLO. The reasons for the chosen intervals of θ and energy are that for angles greater than 180° you only receive a mirroring of the cross section, and the 300 MeV limit is set because of the limits of χEFT.

3.3 Bound-states and the many-body problem

One important and often occurring task in nuclear physics is the calculation of properties for bound-states in the nucleus. For nuclei consisting of more than one nucleon this requires solving the many-body Schrödinger equation. One model often used for computing few-nucleon states is known as NCSM, or the no-core-shell-model. Here the problem is to find the eigenvalues given by the many-body Hamiltonian [13]

$$H = \sum_{j=1}^{A} t_j + \sum_{j<k=1}^{A} v_{2,jk} + \sum_{j<k<l=1}^{A} v_{3,jkl} + \dots \tag{3.4}$$

where t_j represents the kinetic energies of the different nucleons and $v_{2,jk}$ and $v_{3,jkl}$ represents the two- and three-nucleon interactions respectively.

If $|\psi\rangle$ is an eigenstate of this Hamiltonian

$$H\,|\psi\rangle = E\,|\psi\rangle \tag{3.5}$$

you can express $|\psi\rangle$ in an orthonormal complete harmonic oscillator basis $|\phi_i\rangle$ as

$$|\psi\rangle = \sum_i c_i\,|\phi_i\rangle\,. \tag{3.6}$$

which turns the Schrödinger equation into

$$H\sum_i^N c_i\,|\phi_i\rangle = E\sum_i^N c_i\,|\phi_i\rangle\,. \tag{3.7}$$

Since the sum is infinite you need to truncate it at some finite N.

If we now multiply the equation by $\langle\phi_j|$ we get

$$\sum_i^N c_i \langle\phi_j| H |\phi_i\rangle = E \sum_i^N c_i \langle\phi_j|\phi_i\rangle, \tag{3.8}$$

but since the basis is orthonormal the right-hand side becomes Ec_j and on the left-hand side you have $\langle\phi_j| H |\phi_i\rangle$ which corresponds to the matrix element H_{ji} of the Hamiltonian, and the many-body Schrödinger equation results in an eigenvalue problem

$$H_{ji}c_i = Ec_j \tag{3.9}$$

which is solved by diagonalizing the matrix of the Hamiltonian. The lowest eigenvalue E_i gives the ground state energy of the system represented by the Hamiltonian. The eigenvector $\boldsymbol{c}$ provides the amplitudes of the corresponding wave function $|\psi\rangle$ in the basis $|\phi_i\rangle$. For a more thorough description of this see [13].

One of the quantities that we will be working with is the theoretically calculated point-proton radius which is related to electric charge radius measured in experiments. This can be calculated using the eigenvectors obtained in the diagonalization of the Hamiltonian [10].

Chapter 4

Statistical framework for sampling

An important aspect to consider when creating an efficient machine learning emulator is how to select your training data. This chapter serves to present the different sampling techniques used in this thesis. The latin hypercube sampling is introduced as an effective way of sampling, and a discussion on how to sample from a correlated set of parameters concludes the chapter.

4.1 Latin hypercube sampling

The latin hypercube sampling (LHS) was introduced by Mackay et al. in 1979 [14], and has been used extensively in fields where an efficient sampling method is needed [15, 16]. Earlier studies have shown that LHS is preferable in Gaussian process modeling [1, 16].

LHS is generated by dividing every axis of the sample space into a number of intervals, N, with equal marginal probability, $1/N$. For every axis and every interval, one, and only one, coordinate is picked randomly, and together these coordinates define the N sample points [14].

Figure 4.1 demonstrates LHS used on a two dimensional sample space and five sample points, where the dots are the different sample points and the grid outlines the five sampling intervals for every axis.

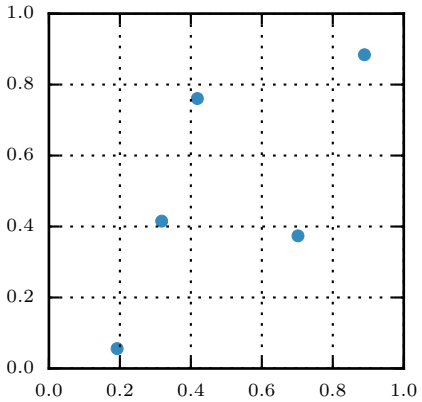


Figure 4.1: Latin hypercube sampling (LHS) performed for two dimensions with five sample points. In two-dimensional LHS, the points are distributed in a manner that ensures the presence of only one point per row and column.

There are several different categories of LHS, stemming from different constraints put on how the samples are selected inside the intervals. For instance they can be chosen randomly inside the given intervals, or with the criteria that they should be picked so to maximize the minimum distance between sampling points, LHS_{mm} [15].

We will use a package for Python called `pyDOE` that generates a latin hypercube sample for any number of dimensions and sample points [17]. This package also allows you to put different criteria on how to sample the points inside the given intervals, where we will examine both the criteria of maximizing the minimum distance between points as well as to select them randomly.

4.2 Sampling the statistical error space of correlated parameters

The parameters, i.e. the so-called LECs, we use have been obtained from separate fits to experimental data, and as such they exhibit a certain statistical error. The parameters with their statistical errors, taken from [9], are shown in Table 4.1.

Table 4.1: The LECs from NLO with their 1σ uncertainty (given in the parenthesis) taken from [9].

LECs from NLO	
$\tilde{C}_{^1S_0}^{(np)}$	$-0.150623(79)$
$\tilde{C}_{^1S_0}^{(pp)}$	$-0.14891(11)$
$\tilde{C}_{^1S_0}^{(nn)}$	$-0.14991(27)$
$C_{^1S_0}$	$+1.6935(83)$
$\tilde{C}_{^3S_1}$	$-0.1843(16)$
$C_{^3S_1}$	$-0.218(14)$
C_{E_1}	$+0.263(16)$
$C_{^3P_0}$	$+1.2998(85)$
$C_{^1P_1}$	$+1.025(59)$
$C_{^3P_1}$	$-0.336(10)$
$C_{^3P_2}$	$-0.2029(15)$

4.2.1 The covariance of the low-energy constants in NLO

The LECs of χEFT hold a dependency on each other which has to be accounted for when sampling over their error space [9]. The correlation matrix of the LECs in NLO is presented in Figure 4.2. This is a symmetric matrix with element

$$C_{ij} = \mathrm{corr}(c_i, c_j) = \frac{\mathrm{cov}(c_i, c_j)}{\sigma_{c_i}\sigma_{c_j}} \tag{4.1}$$

where c_i are the different LECs and $\mathrm{cov}(c_i, c_j)$ is the covariance between LECs c_i and c_j, a result of the statistical fit of the LECs in [9]. A value of $C_{ij} = -1$ means that parameters c_i, c_j are fully anti-correlated, and a value of $+1$ means that they are fully correlated.

4.2.2 Sampling from covariance matrix

Since the parameters have a certain dependence on each other, a correlation matrix must be used to form orthogonal vectors with corresponding error estimates and a normally distributed sample is taken over the variance of these vectors. To capture the properties of this distribution, sampling will be done from a multivariate Gaussian distribution [18].

More explicitly, this is done via a linear transformation of the parameters with the eigenvectors of the correlation matrix as the coefficients of the transformation matrix. If $\tilde{\alpha}_i$ is our transformed parameter with index i and α_i the original, we get

$$\tilde{\alpha}_i = \boldsymbol{v}_{i,1}\alpha_1 + \boldsymbol{v}_{i,2}\alpha_2 + \cdots + \boldsymbol{v}_{i,11}\alpha_{11} \tag{4.2}$$

with the different $\boldsymbol{v}$'s being the different eigenvectors of the correlation matrix. For the whole vector of parameters we get

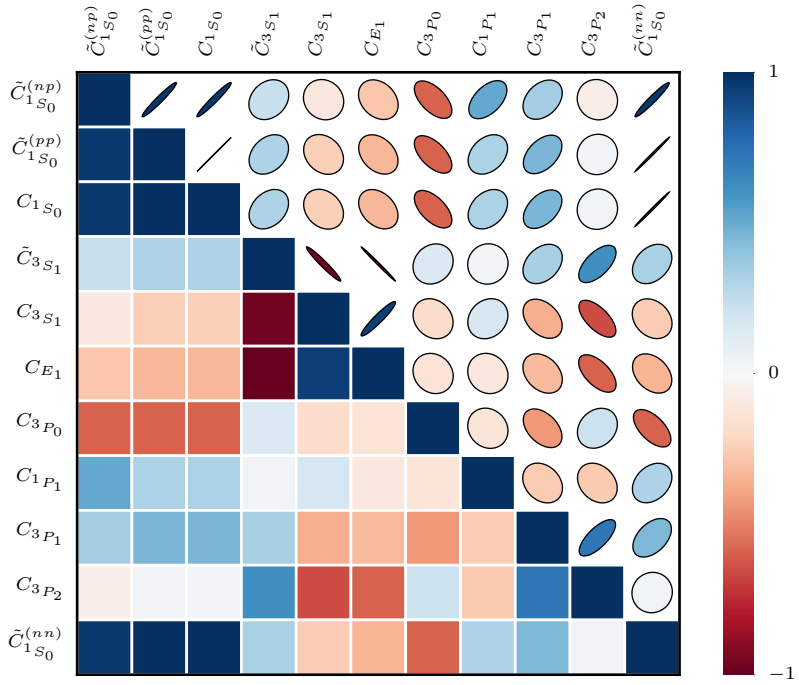


Figure 4.2: Correlation matrix of LECs in NLO constructed using data from [9]. The eccentricity of the ellipses correspond to the value of the correlation while the slope of the major axis determines whether the LECs are correlated or anti-correlated. The squares describes the correlation according to the colorbar. -1 means fully anti-correlated and $+1$ means fully correlated.

$$\tilde{\alpha} = V\alpha \implies \alpha = V^{-1}\tilde{\alpha} \tag{4.3}$$

where V is the matrix of eigenvectors received from the diagonalization of the covariance matrix.

The eigenvector of the correlation matrix corresponding to the largest eigenvalue represents the direction in which the sample varies the most, and so on. The corresponding eigenvalues represents the magnitudes of these variances [19]. By multiplying the eigenvectors with the LECs, the sample space is rotated and it is then possible to sample independent vectors along the rotated interval. Finally, the matrix of eigenvectors is inverted and multiplied with the rotated parameters as in Equation (4.3).

It is still important to recognize that the uncertainties given in Table 4.1 are the 1σ deviation when assuming that the parameters are normally distributed. In this thesis, in order to account for this, we do the LHS over the previously mentioned multivariate Gaussian distribution.

Chapter 5

Results

This chapter presents the results of using Gaussian processes to emulate certain observables in nuclear physics. We first present the results from predicting different cross sections. After that we present some introductory results from predicting bound-states of nuclear physics. The chapter ends with results concerning the time efficiency of the Gaussian processes.

5.1 Calculating cross sections using Gaussian processes

The cross sections that we study in this thesis are from neutron-proton scattering. The results in Sections 5.1.1 and 5.1.2 are based on a set of LECs according to the mean values given in Table 4.1. In Section 5.1.3 however, we perform and present the results from an error propagation of the statistical uncertainties given in the same table.

The emulator was trained using differently sized training data sets and the predicted values were compared to exact data computed with `nsopt`.

Figure 5.1 shows the relative error of the emulator, compared to the data from `nsopt`, from the prediction of A_{zx} with 10, 100 and 1000 training points. It shows that the relative errors are symmetrically distributed around zero, and we use the standard deviation as a measure of the quality of the emulator. Similar results were found for the distribution of the errors of all emulations done in this thesis.

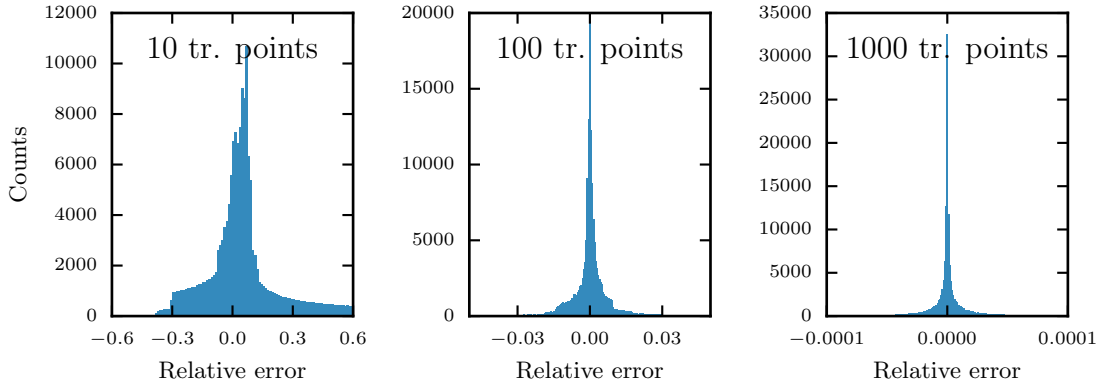


Figure 5.1: The distribution of the relative error of the emulator against data from the simulator, `nsopt`, for A_{zx} predictions using 10, 100 and 1000 randomly sampled training points.

5.1.1 The total cross section

For a given set of fixed LECs, the total cross section depends only on the energy of the incoming beam.

The blue line in Figure 5.2(a) shows the standard deviation of the relative error between our emulator and `nsopt` for different sizes of the training data set. We used 10, 100, 500 and 1000 training points selected with equal spacing on the energy interval, and the predictions were cross-validated against a set of 2000 calculations with `nsopt`.

One problem with predicting the total cross section was its great variance between high and low energies. In order to flatten out this variance, and then see if the precision of the predictions could be increased, we also trained the emulator using the logarithm of the total cross section. This corresponds to the green dashed line in Figure 5.2(a).

To further illustrate the difference between training with the logarithm of the total cross section, Figure 5.2(b) shows the different results for predicting the total cross section. The blue line corresponds to using the regular data when training and the green dashed line corresponds to training on the logarithm of the training points.

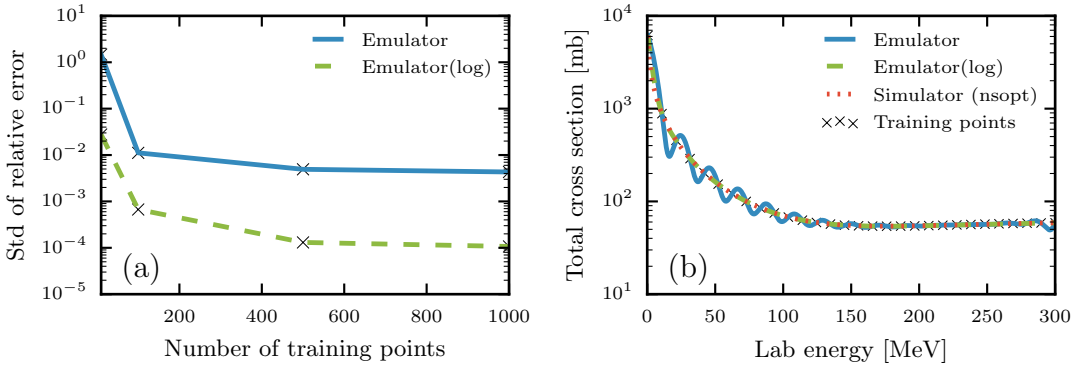


Figure 5.2: (a) The standard deviation, std, of the relative error when the total cross section was predicted for various amounts of training points, both with the logarithm of the total cross section and the regular values of the total cross section. (b) the total cross section trained with regular data and logarithm taken before training for 30 training points.

5.1.2 Differential cross sections and A_{zx}

With fixed LECs, both the unpolarized differential cross section and the polarized A_{zx} depend only on the energy of the incoming beam and the scattering angle, θ. Because of the inefficiency of the equal-spacing sampling method in sampling functions of several variables, the training points for the differential cross section and A_{zx} were sampled using both random sampling and the LHS_{mm} method. Both methods were used to obtain training sets with 10, 100, 500 and 1000 points.

The sampling was performed 20 times for every size of the training set. In Figure 5.3 we show the mean standard deviation of these sets for different number of training points. The width of the bands represents the 1σ interval of the standard deviations of the twenty different groups of sampling points.

Before training the emulator to predict the differential cross section, the logarithm of the training data was taken to even out the steepness of the curve for low-energies, as in the above case of the total cross section.

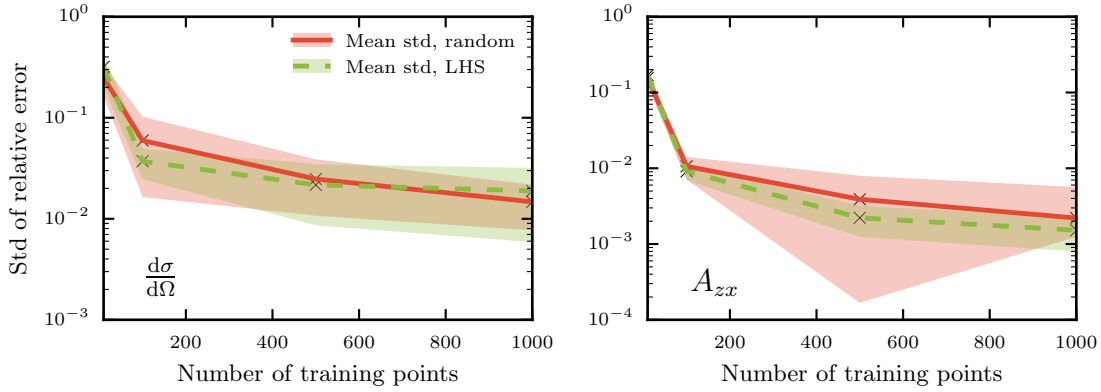


Figure 5.3: The mean of the standard deviation of the relative error when predicting for differently sized training sets from random sampling and LHS_{mm} for the differential cross section and A_{zx}.

5.1.3 Propagating statistical errors in LECs

At NLO we have a set of 11 LECs. Two of them are only relevant for proton-proton and neutron-neutron interactions. Since we only study the neutron-proton interaction in this thesis, we will only have to consider a set of 9 LECs, selected from the distribution according to Section 4.2.2.

As a primary implementation, we explored the error propagation abilities of the emulator by varying the LECs for the total cross section at a single energy, namely $T_{\text{lab}} = 19.665\,\text{MeV}$. Figure 5.4 shows the total cross section values predicted by `nsopt` (the emulator) on the left (right) hand side. The `nsopt` values have been generated for 100,000 parameter samples. Predictions were then made with the emulator for the same sets of parameters based on 50 training points. In both graphs, a normal distribution specified by the mean value of the total cross section and its standard deviation is shown in red. The standard deviation of the values predicted by the emulator differs on the third decimal from that of `nsopt`, i.e. a relative uncertainty of order 10^{-5}.

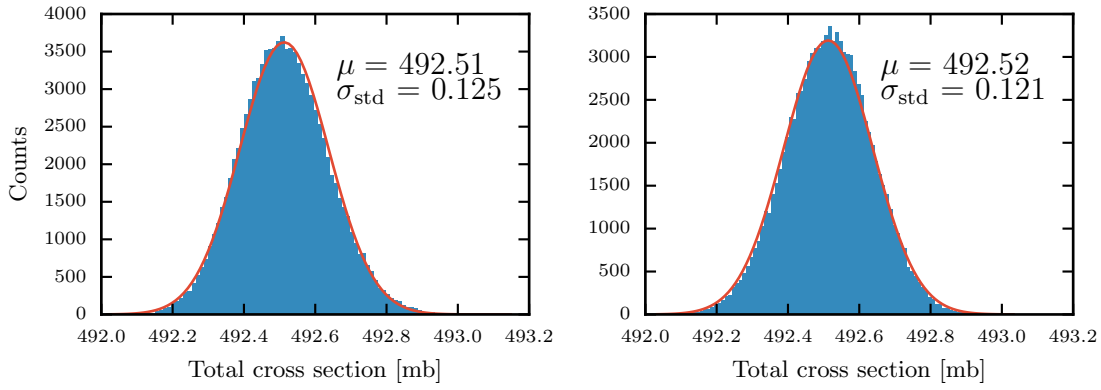


Figure 5.4: The left panel shows the distribution of values calculated for 100,000 parameter samples by `nsopt`. The right panel instead shows the predictions of the emulator for the same sets of parameters after training on 50 sets. The red curve is the same in both panels and it shows a normal distribution specified by the mean and standard deviation of the `nsopt` distribution.

Additionally, the emulator was used to perform error propagation throughout the entire energy range. Treating the energy as another parameter, 10 parameters had to be varied simultaneously (9 LECs, 1 energy). The energies were uniformly sampled from the interval 0.5 to 290 MeV. The LECs were randomly sampled from a normal distribution specified by the mean and standard deviation of Table 4.1 and covariances as explained in Section 4.2.2.

Figure 5.5 show energy on the horizontal axis and the total cross section on the vertical axis for two different configurations of 100 and 1000 training points respectively. In both cases, the emulator was trained on the logarithm of the total cross section values from `nsopt`, in the same manner as in Section 5.1.1. In total, 20 curves were generated from non-identical, but equally sized, training sets for each of the two size configurations. The total cross section values were then predicted by the emulator for 1000 points with the LECs held constant, as opposed to the training, throughout the entire energy range.

The precision of the emulator was tested against `nsopt`. Figure 5.6 shows the mean standard deviation of the error of the emulator relative to `nsopt` for various sizes of the training set. The predictions were made by the emulator for energies in the 0-290 MeV interval. For every energy, a distinct set of randomly sampled LECs were used. The predictions were made using constant LECs, i.e one set of LECs were used throughout the entire spectrum of energy. Predictions were made 20 times for training sets of varying sizes (10, 100 and 1000 points).

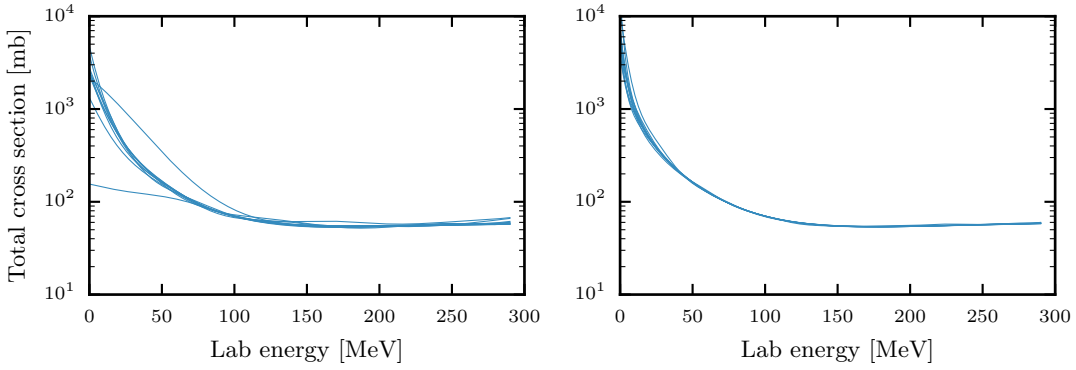


Figure 5.5: 20 total cross section curves as predicted by the emulator for energies in the 0-290 MeV interval. For every energy, a distinct set of randomly sampled LECs were used. On the left (right) a training set with a total number of 100 (1000) points were used. The subsequent predictions were made using non-varying LECs, i.e one set of LECs were used throughout the entire energy range. As can be seen in the left panel, two significant outliers indicate the low precision of the predictions.

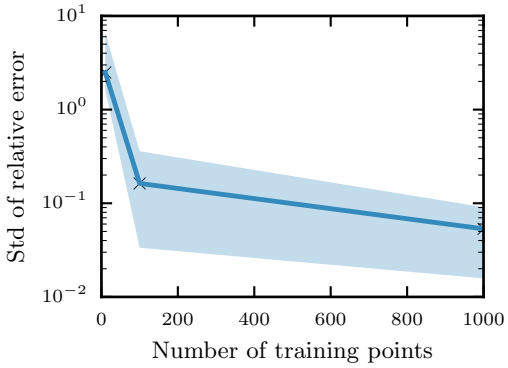


Figure 5.6: The solid line shows the standard deviation of the relative error when the total cross section was predicted for various amounts of training points. The relative error has been computed as the mean value of 20 different sets of parameters for three different set sizes (10, 100 and 1000). Each set of parameters consists of a number of energies evenly distributed in the interval 0-290 MeV, with each energy associated with a set of randomly sampled LECs. The predictions that were compared to `nsopt` were made using constant LECs throughout the entire energy range.

5.2 Bound-states

So far we have only looked at scattering observables. However the emulator was also used to predict bound-state observables for systems with up to four nucleons.

To measure the emulator's efficiency we performed an error propagation for different LEC values similar to the one performed for the cross sections. Training points were sampled from a random multivariate Gaussian distribution of LECs using the previously mentioned covariance matrix.

The simulator, `nsopt`, calculates a set of 9 bound-state observables for each set of LECs and correlation between these observables has been documented [9].

5.2.1 Correlation between binding energy and point-proton radius

At N2LO the correlation between the binding energy of the alpha particle E(^{4}He) and the point-proton radius of the deuteron R_{pt-p}(^{2}H) had previously been plotted as a joint probability distribution [9]. That, however, was never done for bound-state observables at the NLO-level which is the focus of this thesis. Plotting the joint probability distribution of the above named observables for this level was therefore the first task in this area. Also the contour lines corresponding to the 1σ and 2σ intervals were to be generated in order to better visualize the limits of the distribution.

12,000 sets of LECs were sampled and observables were calculated for each set with `nsopt`. This is the distribution shown in Figure 5.7. The contour lines corresponding to 1σ and 2σ confidence intervals were then generated and are used as a reference (solid black line) in both Figure 5.7 and Figure 5.8 to show the accuracy of the emulator.

Earlier predictions made for respective observable at NLO had set the limits at 1.970 and 1.972 fm for the point-proton radius and -27.3 and -27.6 MeV for the binding energy using statistica l analysis[9]. As seen in Figure 5.7 most of the points calculated by `nsopt` stay within these intervals.

To test the accuracy of the emulator, two training sets consisting of 50 and 100 training points were used to predict the observables of all 12,000 points. The contour lines of these predicted distributions were also generated, also shown in Figure 5.7.

The dotted blue (dashed red) lines correspond to the 1σ and 2σ intervals of the distribution predicted by utilizing 50 (100) training points. At 100 training points, the relative error between the predicted data and `nsopt` are on the order of 10^{-6} for both observables. A remarkably small number with so few training points.

Figure 5.8 was made in order to get a picture of what the joint probability distribution would look like for a larger number of points. The emulator was provided a set of 100 training points and was then set to predict the observables for 100,000 new sets of LECs. Contour lines corresponding to 1σ and 2σ were generated to show the confidence intervals for the predicted points, and these are shown together with the confidence intervals of the 12,000 points from `nsopt`.

These contours coincide to a great degree and the distribution in Figure 5.8 is very similar to the one in Figure 5.7. This shows that a valid distribution is obtained even when predicting for new LEC values. Since so many points are plotted one can assume that the distribution shown in Figure 5.8 is representative of what the real distribution would look like.

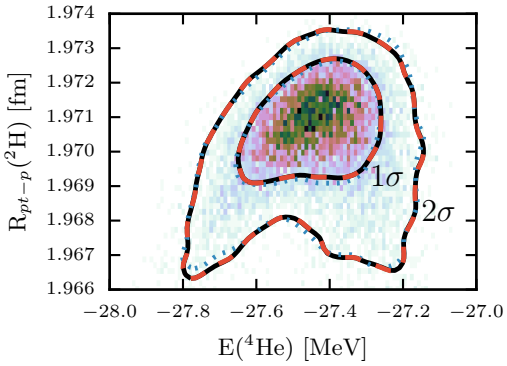


Figure 5.7: Joint probability distribution of the radius of the deuteron, $R_{pt-p}(^2H)$, and the binding energy of the alpha particle, E(^{4}He) at NLO. The black line is from 12,000 calculations with `nsopt`, dotted blue lines are from a prediction of 12,000 points using 50 training points and the dashed red line corresponds to a prediction of 12,000 points using 100 training points.

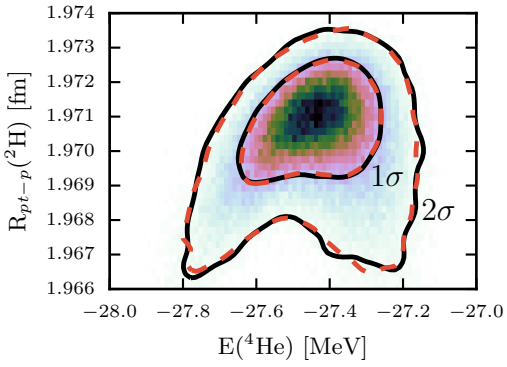


Figure 5.8: Joint probability distribution of the radius of the deuteron, $R_{pt-p}(^2H)$, and the binding energy of the alpha particle, E(^{4}He) for NLO. The black line is from 12,000 calculations with `nsopt`, dashed red lines are from 100,000 points predicted using 100 training points.

5.2.2 Relative errors and time-scales

In this section we will give a more rigorous account of how the errors generated by the emulator were reduced and show how the processor time scales with the number of training points used.

For a given amount of training points, the procedure of training and predicting was repeated 50 times with different training sets. The relative error was taken between the simulator, `nsopt`, and the emulator for every iteration. The processor time for initialization and predicting were also measured respectively.

The mean of the values generated by the iterations was taken for each number of training points in order to eliminate statistical fluctuations.

In Table 5.1 the relative errors as well as the time it took to train and predict for a set number of training points for the point-proton radius are given. Since the errors and the times were very similar for both observables we show only the results for the radius.

It takes approximately 46 seconds for `nsopt` to calculate one set of observables from a set of LECs on a computer with a quad core CPU clocked at 3.4 GHz. Thus, it took 153 hours to calculate the 12 000 points used as a reference in Figure 5.7 and Figure 5.8. In order to reduce time the training sets were calculated on separate computers in groups of 1000. If one were to use the emulator instead, time could be drastically reduced as seen in Table 5.1. 40 or 80 minutes would

be needed to calculate the number of training points necessary to get an accurate distribution. After that, only seconds would be needed to generate a distribution of 12,000 points.

The relative speedup, defined as the quotient between execution times for `nsopt` and for the emulator with the same number of prediction points, is also given in Table 5.1. For 50 (100) training points and 12,000 prediction points a remarkable speedup of 240 (120) is obtained while retaining a relative error on the order of 10^{-5} (10^{-6}). When we go from 100 to 500 training points the relative speedup is reduced by 80% while the relative error decreases by 89%. However for most tasks a relative error of 10^{-6} would probably be acceptable.

Table 5.1: Overview of the calculation time and relative error for different number of training points. Training data generation is the approximate time it takes to calculate the training points with `nsopt`, initialization is the time it takes to prepare the emulator, prediction is the time it takes to calculate 12,000 points with the emulator. The relative speedup, compared with the time it would take to calculate 12,000 simulations, is shown in the last column.

Number of training points	Relative error	Training data generation time [s]	Initialization time [s]	Prediction time [s]	Relative speedup
50	4.82e-5	2300	0.15	0.52	240
100	3.86e-6	4600	0.28	0.68	120
500	4.32e-7	23,000	4.27	1.14	24
1000	2.32e-7	46,000	15.20	1.68	12

5.3 Calculation time for the algorithm

The time analysis of the algorithm was divided into three different steps, (a) optimization of the hyperparameters, (b) decomposition of the covariance matrix and (c) predicting new data points. Figure 5.9 shows the calculation time as a function of both the number of training points and dimension of the training space. It can be seen that the prediction step depends linearly on the number of training points, while optimization and decomposition exhibit at least a polynomial dependency. It was also shown that the calculation time for the prediction was linearly dependant on the number of prediction points.

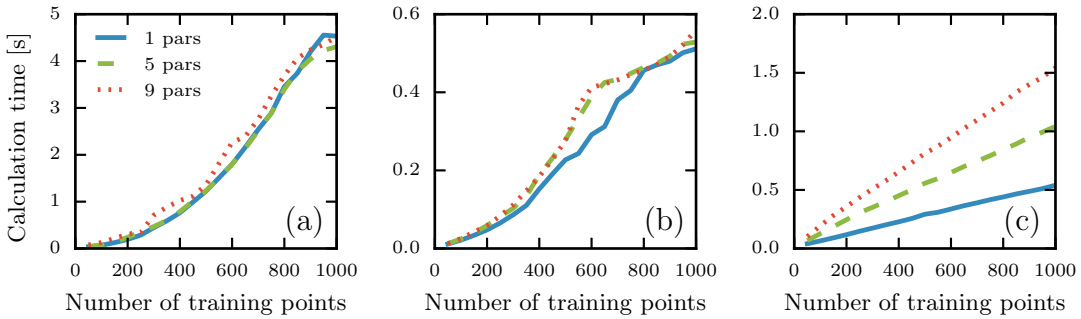


Figure 5.9: The calculation time for differently dimensioned training sets and for three different steps of the algorithm, (a) optimization, (b) decomposition of the covariance matrix and (c) prediction. (a) the calculation time for one run of the optimization with a random starting position. (b) the calculation time for decomposition of the matrices needed in the prediction. (c) calculation time for predicting the value for 10,000 points.

Chapter 6

Discussion

This chapter includes a discussion and an interpretation of the results from the previous chapter in a broader sense, including the difficulties and uncertainties found in the method. First, a discussion is presented on the prediction of different cross sections and the difficulties therein. The subject of emulating bound-states is then examined. The chapter ends with a more general discussion on the benefits and drawbacks of the method.

6.1 Predicting cross sections

In predicting the different cross sections we found that the emulator can be quite successful even with relatively small amounts of training points. For the total cross section and A_{zx} we get to the sub-percent level of errors for just above 100 training points, see Figures 5.2 and 5.3, while the differential cross section reaches a few-percent level around the same amount of training points, Figure 5.3.

6.1.1 Difficulties in predicting some behaviours of the cross sections

A problem we found with the chosen method was in predicting functions that have a large variance in their values, like the total- and differential cross sections for low versus high energies. This is probably due to the fact that the covariance function wants to correlate the low-energy points with the high-energy points where the function takes an almost constant value, much lower than for the low-energies. Because of this it overfits the data in the transition between slopes, see blue curve in Figure 5.2(b).

One way to circumvent this overfitting is by doing a transformation of the training data to decrease its variance [20]. The green dashed curve in Figure 5.2 shows the improvement in the predictions when the logarithm of the function used for training. This is a fairly easy transformation, and the prediction could probably be improved even more if a more advanced method of transformation were to be used. This is left as an outlook for further research.

In the case of error propagation for the energy-dependence of the total cross section, additional difficulties appeared. As shown in Figure 5.4, there were no problems in replicating the errors predicted by `nsopt` with the emulator for a single energy (with a relative uncertainty of order 10^{-5}). However, when using the energy as a 10th parameter alongside the LECs in training the emulator, the errors became significantly larger, as can be seen in Figure 5.6. This is further demonstrated in Figure 5.5 where several outlier curves are observed. These curves unexpectedly deviate from the others with significant magnitudes, indicating large variations in the predictions. We believe that this problem is similar to the problem of capturing the behavior of functions with large variations in their values, i.e., observables with significant dependence on individual parameters, as discussed in the beginning of this subsection. With this behavior in mind, one may expect additional difficulties when training on the full space of the LECs and energy. Since LECs vary by similar orders of magnitude among themselves, causing only slight variations of the total cross section as compared to the variations related to the energy, the emulator may encounter difficulties in simultaneously capturing the distinct behaviors of both energy and the LEC variations. Despite

using the previously discussed logarithm transformations prior to training, these errors remained significant.

6.2 Bound-state observables

As seen in Section 5.2 the joint probability distribution of E(^{4}He) and R_{pt-p}(^{2}H) could be emulated with great accuracy while only using relatively few training points. Figure 5.7 shows that the distribution becomes very similar to the one from `nsopt` with only 100 training points. As seen in Table 5.1 the relative error is also remarkably low for as few as 50 training points. As expected it is reduced somewhat when the number of training points is increased, however, the reduction slows down for larger numbers of training points.

Since a calculation of 100,000 points of the joint probability distribution does not exist at the NLO-level a full evaluation of the predicted results was not possible. We could, however, compare the distribution with one generated from 12,000 points calculated with `nsopt` and see that they still show a similar behaviour, see Figure 5.8.

Regarding the computation time, Table 5.1 shows that it increases with a rising number of training points. This is particularly evident with the time spent on training. For larger numbers of training points, initializing the emulator takes more time than predicting new points but in relation to the time it takes to generate the training points with `nsopt` it is still negligible. The time it takes to generate a certain amount of points for the distribution is foremost dependent on the calculation time of `nsopt`. This means that a great deal of time can be saved by using the emulator in this case. Instead of calculating 100,000 points with `nsopt` we could get a fairly good distribution by only calculating 100 points to use for training, as shown in Figure 5.8. If the calculation of one point takes roughly one minute it means that we save around 1665hours of calculation time.

6.3 Systematic emulator uncertainties

When we set up an emulator and choose a covariance function we also make an assumption on the given data. With this model it is possible to calculate the uncertainty according to Equation 2.5. Given that the model was chosen correctly this would describe the uncertainty of the emulator.

In this project, however, the squared exponential covariance functions was used for all calculations. Even though this covariance function works well for making predictions, we have no statistical reason for assuming the data points are correlated in this way. Therefore it is possible and even likely that this introduces a systematic error to the predictions.

6.4 Hyperparameters

An important aspect of the GP model lies in the optimization of the hyperparameters of the covariance function (see Section 2.4 for a formulation of this problem). Using starting points selected from a large interval, we found that, for some cases, the accuracy of the emulator decreased when the number of optimizations performed was increased. One possible explanation could be that the function maximized by the likelihood estimation has some maxima far away from the optimum value for the target function. However, this is just one possible explanation and more research would be needed to understand this.

6.5 Time and memory complexity

When calculating the mean according to Equation (2.4), and the log marginal likelihood (2.9) that is used for the objective function, a solution to a linear system of equations is needed. In order to solve the linear system a Cholesky decomposition is used on the matrix, an algorithm that scales with the number of training points as $\mathcal{O}(n_t^3)$ in time complexity. After the matrix is decomposed, we obtain a triangular system that can be solved with an algorithm that is $\mathcal{O}(n_t^2)$.

This is confirmed in Figure 5.9 where it can be seen that the calculation time scales polynomially with the number of training points.

For making predictions the only step left in Equation (2.4) is the multiplication of a matrix with size $n_p \times n_t$ multiplied with the solution of the linear system, which is a vector of length n_t. The multiplication has time complexity $\mathcal{O}(n_t n_p)$, which means that making the predictions is linear in both number of training points n_t and prediction points n_p. The predictive variances in Equation (2.5) can also be calculated if needed but since it requires additional matrix multiplication it has the higher time complexity $\mathcal{O}(n_t^2 n_p)$.

Storage of the Cholesky decomposition is needed for calculation of the predictive variances, which is $\mathcal{O}(n_t^2)$ in memory complexity. If just the mean will be calculated then only one solution to the matrix equation needs to be stored, which is $\mathcal{O}(n_t)$. The calculation of the predictive mean requires $\mathcal{O}(n_t n_p)$ memory, but since it is linear in time it can be divided into smaller parts and therefore only requires the memory for storing the results.

Chapter 7

Conclusions and recommendations

The conclusions of this thesis are summarized in this chapter. For a more detailed discussion see Chapters 5 and 6. The chapter ends with an outlook on recommendations for directions of future research.

The great potential of using Gaussian process modeling when emulating nuclear observables has been shown in this thesis. We have demonstrated the possibility of saving thousands of hours in calculation time without adding too much uncertainty to the produced results.

This thesis was done as a pilot study on the possibilities of using Gaussian processes in nuclear physics, and as such it is limited regarding the depth of analysis in every facet of the method. Therefore we present a short description of areas where we consider further research to be needed.

7.1 Covariance functions

Since this was an introductory study we limited the covariance function of use to the squared exponential. Although this is a fairly general covariance function with a broad usability, it would probably be worth studying other functions specifically designed for the problem at hand. If the behavior of the target function is known this could be done by simply choosing a function more resemblant of the target. If instead, as is often the case, the distribution of the target function is unknown, a covariance function can be assembled from the training data.

The Fourier transform of the covariance function, known as its spectral density, provides information about the smoothness of the covariance function. It can be interpreted as a measure of the decay of the eigenfunctions in a spectral decomposition of the covariance function $k(\boldsymbol{x}, \boldsymbol{x}')$. Smooth processes usually have a higher spectral density for lower frequencies, while processes with rapid fluctuations tend to have a higher density for higher frequencies [6].

7.2 Scikit-learn 0.14.1

For this project the implementation of Gaussian processes for regression in Scikit-learn 0.14.1 was used [5]. Using an already existing implementation of the algorithm made it faster and easier to get started, but it also came with some limitations. Version 0.14.1 of Scikit-learn made it harder to make compound covariance functions, while some were even impossible to implement. These deficits in the implementation could be a problem for future work. But since Scikit-learn is an active and still growing project the implementation might be improved in the future.

7.3 Time complexity

When working with large datasets there exist several approximation methods for decreasing the computational complexity of the algorithm. It is also possible to make approximations for the log marginal likelihood function (2.9) and its derivatives. The derivative of the log marginal likelihood function was not used for optimization during this project and would most likely speed up the optimization [6]. This could make Gaussian process more applicable to large datasets.

7.4 Outlook

- It should be possible to calculate more complex observables, e.g. for heavier nuclei or higher chiral orders, where exact calculations are very costly.
- A more effective way to implement suitable covariance functions should be investigated.
- It would be desirable to have a more capable implementation of the Gaussian process, and therefore Scikit-learn should not be considered the default choice for future work.

Bibliography

[1] Salman Habib, Katrin Heitmann, David Higdon, Charles Nakhleh, and Brian Williams. Cosmic calibration: Constraints from the matter power spectrum and the cosmic microwave background. *Phys. Rev. D*, 76:083503, Oct 2007.

[2] Louis-Fran çois Arsenault, Alejandro Lopez-Bezanilla, O. Anatole von Lilienfeld, and Andrew J. Millis. Machine learning for many-body physics: The case of the anderson impurity model. *Phys. Rev. B*, 90:155136, Oct 2014.

[3] J. D. McDonnell, N. Schunck, D. Higdon, J. Sarich, S. M. Wild, and W. Nazarewicz. Uncertainty quantification for nuclear density functional theory and information content of new measurements. *Phys. Rev. Lett.*, 114:122501, Mar 2015.

[4] R. Machleidt and D. R. Entem. Chiral effective field theory and nuclear forces. *Phys. Rept.*, 503:1–75, 2011.

[5] 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.

[6] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[7] H. Georgi. Effective-field theory. *Annual review of nuclear and particle science*, 43:209–252, 1993.

[8] R. Machleidt. High-precision, charge-dependent bonn nucleon-nucleon potential. *Physical Review C*, 63, 2001.

[9] Boris Carlsson, A. Ekstrom, Christian Forssén, Dag Fahlin Strömberg, G. R. Jansen, Oskar Lilja, Mattias Lindby, Björn Mattsson, and K. A. Wendt. Uncertainty analysis and order-by-order optimization of chiral nuclear interactions. *Physical Review X*, 6, 2016.

[10] Boris Carlsson. *Making predictions using χEFT*. Institutionen för fundamental fysik, Chalmers tekniska högskola,, 2015. 89.

[11] Brian R Martin. *Nuclear and Particle Physics*. John Wiley & Sons, Ltd., 2006.

[12] Norio Hoshizaki. Formalism of nucleon-nucleon scattering. *Supplement of the Progress of Theoretical Physics*, 42, 1968.

[13] P. J. Brusaard and I. Glaudemans. *Shell-model applications in nuclear spectroscopy*. North-Holland publishing company, 1977.

[14] W. J. Conover M. D. McKay, R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[15] Jared L. Deutsch and Clayton V. Deutsch. Latin hypercube sampling with multidimensional uniformity. *Journal of Statistical Planning and Inference*, 142(3):763 – 772, 2012.

[16] Kenny Q. Ye, William Li, and Agus Sudjianto. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of statistical planning and inferences*, 90:145–159, 2000.

[17] pydoe: The experimental design package for python. `https://pythonhosted.org/pyDOE/`. Accessed: 2016-03-08.

[18] J Dobaczewski, W Nazarewicz, and P-G Reinhard. Error estimates of theoretical models: a guide. *Journal of Physics G: Nuclear and Particle Physics*, 41(7):074001, 2014.

[19] Benno List. Decomposition of a covariance matrix into uncorrelated and correlated errors.

[20] M. N. Gibbs. *Bayesian Gaussian processes for regression and classification*. University of Cambridge, 1997.