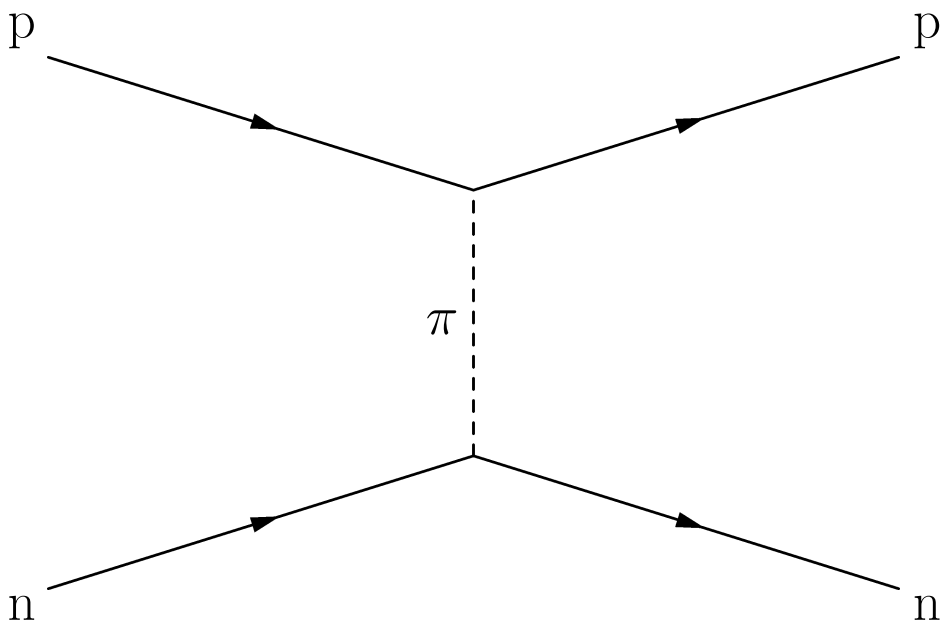
CHALMERS



Uncertainty Quantifications in Chiral Effective Field Theory

Bachelor of Science Thesis for the Engineering Physics Program

Dag Fahlin Strömberg, Oskar Lilja,
Mattias Lindby, Björn Mattsson

Department of Fundamental Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2014

BACHELOR OF SCIENCE THESIS FOR THE ENGINEERING PHYSICS PROGRAM

Uncertainty Quantifications in Chiral Effective Field Theory

Dag Fahlin Strömberg
Oskar Lilja
Mattias Lindby
Björn Mattsson

Department of Fundamental Physics
Division of Subatomic Physics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2014

Uncertainty Quantifications in Chiral Effective Field Theory.
Dag Fahlin Strömberg[a], Oskar Lilja[b], Mattias Lindby[c], Björn Mattsson[d]

Email:
[a]fdag@student.chalmers.se
[b]lioskar@student.chalmers.se
[c]lindby@student.chalmers.se
[d]mabjorn@student.chalmers.se

FUFX02 - Bachelor thesis at Fundamental Physics
Bachelor's thesis FUFX02-14-03

Supervisor: Andreas Ekström, Christian Forssén
Examiner: Daniel Persson

Department of Fundamental Physics
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: +46 (0)31-772 10 00

Cover:
A Feynman diagram illustrating a one-pion exchange between two nucleons.

Chalmers Reproservice
Göteborg, Sweden 2014

Abstract

The nuclear force is a residual interaction between bound states of quarks and gluons. The most fundamental description of the underlying strong interaction is given by quantum chromodynamics (QCD) that becomes nonperturbative at low energies. A description of low-energy nuclear physics from QCD is currently not feasible. Instead, one can employ the inherent separation of scales between low- and high-energy phenomena, and construct a chiral effective field theory (EFT). The chiral EFT contains unknown coupling coefficients, that absorb unresolved short-distance physics, and that can be constrained by a non-linear least-square fitting of theoretical observables to data from scattering experiments.

In this thesis the uncertainties of the coupling coefficients are calculated from the Hessian of the goodness-of-fit measure χ^2. The Hessian is computed by implementing automatic differentiation (AD) in an already existing computer model, with the help of the Rapsodia AD tool. Only neutron-proton interactions are investigated, and the chiral EFT is studied for leading-order (LO) and next-to-leading-order (NLO). In addition, the correlations between the coupling coefficients are calculated, and the statistical uncertainties are propagated to the ground state energy of the deuteron.

At LO, the relative uncertainties of the coupling coefficients are 0.01%, whereas most of the corresponding uncertainties at NLO are 1%. For the deuteron, the relative uncertainties in the binding energies are 0.3% and 0.7% for LO and NLO, respectively. Moreover, there seems to be no obvious obstacles that prevent the extension of this method to include the proton-proton interaction as well as higher chiral orders of the chiral EFT, e.g. NNLO. Finally, the propagation of uncertainties to heavier many-body systems is a possible further application.

Acknowledgements

We wish to thank our supervisors Christian Forssén and Andreas Ekström for their support and guidance during the course of this project. In addition, we would like to express our appreciation to Boris Carlsson for his support with programming issues as well as quick response to our questions.

The Authors, Gothenburg, June 28, 2014

Contents

Chapter 1

Introduction

Ever since the first pion-exchange model was proposed by Yukawa in 1935, attempts to accurately describe the nuclear force have been a recurring theme in nuclear physics. As this force binds nucleons together and thereby gives rise to nuclei, it is of central importance to the field.

The advent of the quark model and the theory of quantum chromodynamics (QCD) meant a radical departure from the Yukawa model and similar meson-exchange theories: The nucleons are now understood to consist of quarks, and the nuclear force is nothing more than a residual interaction of the strong force binding them together. This implies that QCD is the fundamental theory of the nuclear force, but as it is nonperturbative in the low-energy region relevant for nuclear physics, its direct application to the nucleus is extremely computationally demanding. [1]

A modern approach to bypass this problem is by approximating QCD as an effective field theory (EFT). EFTs are used in many different contexts in physics, and utilise the fact that the low-energy behaviour of a theory is insensitive to the details of its short-range nature. Thus, only the long-range dynamics of the original theory need to be resolved exactly in the approximation, whereas the unresolved short-range behaviour is imitated by adding local correction terms to the Hamiltonian of the EFT. The precision of the approximation increases with the addition of more terms. [2] In this thesis, the terms are denoted as leading-order (LO), next-to-leading-order (NLO), next-to-next-to-leading-order (NNLO), and so on. As it is essential that the EFT has the same symmetries as the original theory, our EFT must inherit the chiral symmetry of QCD and it is therefore referred to as chiral EFT. [1]

Each correction term to the Hamiltonian contains a coupling coefficient, which must be determined from experimental data. A computer program, `nsopt`, that uses low-energy scattering measurements has been developed by Andreas Ekström and Boris Carlsson, in an attempt to find the optimal values of these coefficients. This is a part of a larger research collaboration including the Chalmers and Oslo Universities, Oak Ridge National Laboratory, University of Idaho and Argonne National Laboratory. The optimisation in `nsopt` uses a non-linear least-square fit, where the goodness-of-fit measure to be minimised is denoted by χ^2.

The goal of this thesis is to extend `nsopt` to allow for the computation of the first- and second-order partial derivatives of the χ^2-value to be computed with respect to the coupling coefficients. This enables an uncertainty quantification of the optimal coefficient values. To arrive at the derivatives, automatic differentiation (AD) is used. This method is based on the fact that every computer program, no matter how complex, is implemented using a set of simple operations with well-known derivatives. AD uses this in combination with the chain rule to calculate the derivatives through the computational chain of the program, and can do this to machine precision and to an arbitrary derivative order. The AD tool that is used in this project is Rapsodia [3], which combines custom data types with operator overloading to facilitate the differentiation.

1.1 Specific aims

The aims of this thesis can be summarised in the following list:

- Implement AD in the existing `nsopt` code, to allow for the calculation of derivatives.
- Use the computed derivatives to create the covariance matrix and thus find error estimates of the coupling coefficients in the chiral EFT.
- Propagate these uncertainties through the model to different physical observables, e.g. the binding energy of the deuteron.

1.2 Limitations

Due to the limited time frame of the project, not all possibilities could be fully explored. Consequently, only scattering data from neutron-proton interactions have been used for optimisation in the course of this study. This means that we have omitted proton-proton scattering data that would have required implementation of AD in the Coulomb interaction part of `nsopt`. Furthermore, only the LO and NLO terms of the chiral EFT were considered.

1.3 Method

Initially, the project consisted of literature studies in the relevant physics and computational techniques. The subsequent implementation of AD in `nsopt` was performed in a bottom-up manner, starting at the beginning of the computational chain and progressing upwards through the various parts of the software. This gradual approach allowed the AD computed derivatives to be compared with results of numerical differentiation, making errors easier to locate. More details on `nsopt` and the development process can be found in Chapter 6.

1.4 Structure of the thesis

In Chapter 2 a general overview of EFTs and the nuclear force is presented. Chapter 3 concerns scattering theory. The EFT contains a number of parameters that must be determined experimentally. The determination of these parameters is the subject of Chapter 4, which also addresses how generated derivatives can be used to produce covariance matrices and error estimates of the parameters. Chapter 5 explains the principle of AD, as well as the use of the Rapsodia AD tool. The actual implementation of AD in the `nsopt` program is described in Chapter 6. The results are presented in Chapter 7, and the following discussion can be found in Chapter 8. Finally, we summarise our conclusions and recommendations in Chapter 9.

Chapter 2

Effective Field Theories and the Nuclear Force

This chapter serves as a brief introduction to EFT and the nuclear force. The subject is presented here in the context of the computer model at the centre of this thesis, which attempts to fit a chiral EFT to experimental data. Due to the magnitude of the topic of chiral EFTs and their applications to nuclear physics the discussion here is brief. A more formal but still incomplete introduction to chiral symmetry and its breaking is given in Appendix A.

2.1 A history of the nuclear force

The discovery of the nucleus by Rutherford in 1911, and the subsequent detection of the neutron by Chadwick, Curie and Joliet in 1932 implied the existence of an at that time unknown force binding the nucleons together. To counter the repulsive Coloumb interaction between the protons, it had to be strong enough to overcome the electromagnetic force. At the same time, its range would have to be very short, otherwise the nuclei of different atoms would interact and destroy the atomic structure.

In 1935, Yukawa proposed that the nuclear force could be explained by a virtual pion exchange between nucleons. More elaborate theories based on the exchange of pions and other mesons have been developed since then. However, despite their success in describing many aspects of the interaction between nucleons, the discovery of quantum chromodynamics (QCD) in the late 1970s meant that these theories had to be regarded as mere models. QCD is a quantum field theory describing how quarks, which nucleons have been shown to consist of, interact via the so-called strong force. As the nuclear force between nucleons is nothing more than the residue of the strong force between their quarks, QCD is in principle the fundamental theory of the nuclear force. Unfortunately, the strong force is nonperturbative in the nuclear low-energy region, which severely limits the direct use of QCD in nuclear physics.

In the early 90s, Weinberg managed to apply the concept of EFTs to QCD in the low-energy realm, resulting in a chiral EFT without the nonperturbative properties of the original theory. Curiously enough, the chiral EFT describes the nuclear force in terms of pion exchanges just as the original theory of Yukawa, but with the additional constraint of the broken chiral symmetry, which was not known earlier.

2.2 Effective field theories

An EFT is a way of approximating a physical theory without knowing every detail of its behaviour. The theory introduces an energy limit called a cutoff, denoted by Λ. Physics above this energy limit is imitated only to the extent that the resulting approximation will give the right description of the physics that reside below the cutoff. Consequently, the physics below the cutoff is resolved whereas the physics

above is said to be unresolved. The theoretical basis of this process can be found in a technique known as renormalisation [2].

An example of an EFT approximation can be found by imagining a very small current source that radiates electromagnetic waves in an unknown way. If only waves with energy below the cutoff are of interest, only the large-scale structure of the source is significant. This means that distances roughly equal to the wavelength and above need to be described accurately whereas the smaller details can be simplified greatly and still give the correct result for long wavelengths. If the current source is much smaller than the wavelength corresponding to the cutoff the source can be approximated by a set of point sources. This constitutes a significant simplification, since the behaviour of point sources is well understood.

Formally, it can be shown that the EFT can be seen as an expansion in the parameter $\frac{p}{\Lambda}$ where $p < \Lambda$ is the momentum studied. Λ is usually a couple of magnitudes larger than the momentum of interest. To summarise, an EFT needs to:

- Give the correct description of physics at long distances. All features of the long-range behaviour must be known from the underlying theory.
- Introduce a cutoff from an inherent separation of scales in the system studied.
- Imitate short-range behaviour. This is done with correction terms added to the Hamiltonian, forming the contact potential. The potential describes the part of the long-range behaviour that arises from unresolved short-range interactions.

2.3 Chiral symmetry

An object is said to be chiral if it is not identical to its mirror image, the simplest example being the left and right hands of the human body. The hands are mirror images of each other but impossible to superimpose, which becomes evident when trying to fit a right-handed glove on the left hand. Chirality arises in many other parts of nature, including subatomic physics. Chiral transformations in subatomic physics act independently on so called left and right-handed particles. The definition of left and right-handed particles is concerned with how the spin is projected on the direction of motion [4].

In QCD, chiral symmetry would appear if the quarks were massless [5]. As we know, this is not the case, but the concept of chiral symmetry is still useful if it is only broken in a minor way. This is the core; quark masses are small relative to other masses handled by the theory. A more thorough introduction to chiral symmetry is given in Appendix A.

2.4 Chiral EFT

Chiral EFT is based on the fact that in low-energy physics the relevant degrees of freedom for the Lagrangian of QCD are hadrons, contrary to the normal case when they are quarks and gluons. To make use of the ideas from EFTs a cutoff must be introduced. Since we are studying pion exchange the cutoff needs to be larger than the pion mass, a simple choice is to take the cutoff to be in the region of the heavier mesons like the rho meson. To get the EFT an expansion is done with the parameter $\frac{p}{\Lambda}$ as explained in Section 2.2. Along with this a contact potential accounting for short-range behaviour is needed. To model the short-distance interaction in the potential qualitatively, we rely on meson theory. According to this, the short-range behaviour is characterised by heavy meson exchange [1]. The propagator is defined as,

$$\int \mathrm{d}^3p \frac{e^{i\mathbf{p}\cdot\mathbf{r}}}{m^2+p^2} \approx \frac{e^{-mr}}{r}.$$

One property of chiral EFT is that it resides in the low-momentum region. This might cause a problem of describing heavy meson exchange. The issue is avoided by realising that the concerned momentum is much smaller than the mass of the meson, $p \ll m$. This allows for expanding the propagator as,

$$\frac{1}{m^2+p^2} \approx \frac{1}{m^2}\left(1-\frac{p^2}{m^2}+\frac{p^4}{m^4}+\ldots\right).$$

A conclusion is that it should be possible to describe the short range interaction in powers of $\frac{p}{m}$. However, the contact potential also involves terms from renormalisation theory, which is not treated in this thesis. Chiral EFT describes the long-range behaviour using virtual pion exchange, the short range behaviour is described by the contact potential. The contact potential comes with unknown coefficients that need to be fitted against experimental data. The fitting of these parameters is a current optimisation problem in the field.

The model used by the computer program in this study utilises the potential from chiral EFT. The one-pion exchange and the contact potential results in the LO potential as,

$$V_{LO} = -\frac{g_A^2}{4f_\pi^2}\tau_{\mathbf{1}}\cdot\tau_{\mathbf{2}}\frac{\sigma_{\mathbf{1}}\cdot\mathbf{q}\sigma_{\mathbf{2}}\cdot\mathbf{q}}{q^2+m_\pi^2} + C_S + C_T\sigma_{\mathbf{1}}\cdot\sigma_{\mathbf{2}}, \tag{2.1}$$

where g_A is the axial-vector coupling constant and f_π is the pion decay constant. m_π is the mass of the pion, $\sigma_{1,2}$ are the spin operators for the two nucleons, $\tau_{1,2}$ are the isospin operators of the nucleons and q is the momentum transfer between the two nucleons. The two constants C_S and C_T are the coupling coefficients of the contact potential. If correction terms of higher orders are added, it will result in a chiral EFT of order (NLO), (NNLO) and so on. A detailed discussion about chiral EFT potentials of LO and higher orders can be found in Ref. [1].

Chapter 3

Scattering Theory

This chapter contains a brief introduction to two-nucleon scattering theory. Only key concepts of relevance to the computer models are presented. The main purpose is to present how experimental observables obtained from nucleon scattering experiments are related to those calculated from a theoretical interaction potential. A more thorough description is given in Ref. [6, 7, 8, 9].

3.1 Nucleon-nucleon states

The nucleus of an atom consists of neutrons and protons. The similarity between these particles makes it possible to define them both as nucleons differing only with an intrinsic property called the isospin, t. Moreover, the isospin projection is $t_z = +1/2$ and $t_z = -1/2$ for the proton and the neutron respectively. The nucleons, with spin $s = 1/2$, are fermions and therefore need to obey the Pauli exclusion principle forcing the total wave function to be antisymmetric. A general two-nucleon wave function can be expressed as a tensor product between the spatial, spin and isospin part,

$$\Psi_{NN} = \psi_{spatial} \otimes \psi_{spin} \otimes \psi_{isospin}.$$

Where the spatial wave function may be decomposed into partial waves as

$$\psi_{spatial}(r,\phi,\theta) = \sum_{L,M} a_L R_L(r) Y_{L,M}(\phi,\theta), \tag{3.1}$$

a_L describing the amplitudes of the partial waves and $Y_{L,M}(\phi,\theta)$ being the spherical harmonics. To describe a particular wave the spectroscopic notation known from atomic physics is used, ${}^{(2S+1)}L_J$ with $\mathbf{J} = \mathbf{L} + \mathbf{S}$, where $\mathbf{J}$ is the total angular momentum.

The nucleon-nucleon interaction contains a tensor-force component. It is still possible to make the partial wave decomposition although now a single J state, or channel, can be a mix of different partial waves. However, the fact that the strong interaction will preserve the total angular momentum J and the parity π of the wave function makes only a limited number of the LS couplings possible [4].

The deuteron is a bound two-nucleon system consisting of a proton and a neutron with ground state $J^\pi = 1^+$ and $T = 0$. The total angular momentum of the system is given by $\mathbf{J} = \mathbf{L} + \mathbf{S}$, where the sum of the two particles intrinsic spin, $\mathbf{S}$, are 0 or 1. The parity of the system is determined by $(-1)^L$ hence only states with even orbital angular momentum is allowed. With $J = 1$ this leads to the conclusion that the only possible S value is 1 while L has to be either 0 or 2. Hence, the only possible LS coupling is given by ${}^3S_1 - {}^3D_1$.

3.2 Nucleon-nucleon scattering formalism

Scattering experiments are used in order to measure the nucleon interaction. Spin-independent scattering will be considered to give a basic understanding of how the measurable observables relate to the wave

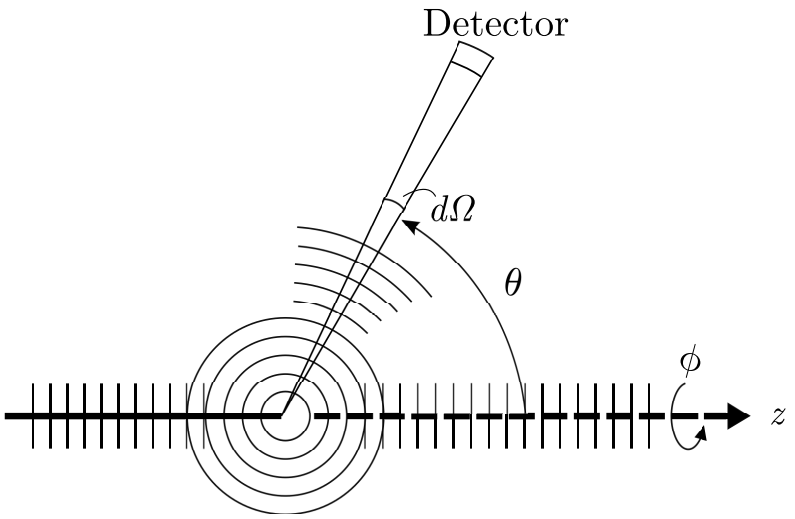


Figure 3.1: Two dimensional projection of a basic scattering experiment. An incident plane wave is scattered against the target resulting in a superposition of spherical and plane waves. A detector is placed far away from the interaction region and subtends the differential cross section $d\Omega$.

functions. Because of the spin-dependence of the nuclear force we will also introduce the spin-scattering matrix needed for a complete spin-dependent treatment of the scattering process. [6]

3.2.1 Cross section

The total scattering cross section, σ, represents the probability of an incident particle to interact and scatter from the target. To measure the cross section in a scattering experiment the incident flux of particles needs to be known and compared with the amount of outgoing scattered particles. However, to measure all scattered particles a spherical detector needs to be used, it is therefore more convenient to measure just a small part of the outgoing particles, which instead will give a differential cross section. A detector with effective area dA at a distance r from the target will subtend the differential solid angle

$$d\Omega = \frac{dA}{r^2},$$

as displayed in Figure 3.1. The number of particles measured will then be $N_r = S_r dA = S_r r^2 d\Omega$, where S_r denotes the probability current of the scattered wave function. Moreover, the differential cross section is defined as the number of particles scattered in a particular solid angle, divided by the incident flux of particles [10] resulting in the following expression,

$$\frac{d\sigma}{d\Omega} = \frac{S_r r^2}{S_i}. \tag{3.2}$$

The total cross section is then given by integrating $\frac{d\sigma}{d\Omega}$ over the entire solid angle.

3.2.2 Scattering amplitude

The scattering of particles is described by the time-dependent Schrödinger equation with boundary conditions suitable for scattering. Assuming non-relativistic conditions it is given as

$$i\hbar \frac{\partial}{\partial t}\Psi(\mathbf{r},t) = H\Psi(\mathbf{r},t). \tag{3.3}$$

The Hamiltonian is most easily expressed in the relative coordinate system of the particles, $H = -\frac{\hbar}{2\mu}\nabla^2 + V$, where μ represents the reduced mass and V is the interaction potential. It is possible to separate the

spatial and time-dependent parts of the wave function as $\Psi(\mathbf{r},t) = \psi(\mathbf{r})e^{-i\frac{Et}{\hbar}}$, which solves Eq. (3.3). Here, the spatial part, $\psi(\mathbf{r})$, is an eigenfunction of the time-independent Schrödinger equation, $E\psi(\mathbf{r}) = H\psi(\mathbf{r})$. [10]

Assuming a spin-independent scattering experiment with incident beams along the z-axis described as plane waves, the outgoing beams will be scattered as spherical waves after interacting with the target as illustrated in Figure 3.1. This implies that for regions far away from the interaction the wave function has to look like a superposition of the incident particles not interacting with the target and those scattered,

$$\psi(\mathbf{r}) = e^{ikz} + f(\theta,\phi)\frac{e^{ikr}}{r},\ r \to \infty. \tag{3.4}$$

With $f(\theta,\phi)$ defined as the scattering amplitude. In general ψ and $f(\theta,\phi)$ also depend on the in and outgoing wave vectors, and the fact that the scattering probability is low enough to not affect the normalisation of the plane wave. [10]

3.2.3 Differential cross section

The probability density current is given by [11]

$$S(\mathbf{r},t) = -\frac{i\hbar}{2\mu}(\Psi^*\nabla\Psi - \Psi\nabla\Psi^*) = \mathcal{R}(\Psi^*\frac{\hbar}{i\mu}\nabla\Psi),$$

where $\mathcal{R}$ denotes the real part. The incident flux of particles will then be

$$S_i = \mathcal{R}\left(e^{-ikz}\frac{\hbar}{i\mu}\frac{d}{dz}e^{ikz}\right) = \frac{\hbar k}{\mu}. \tag{3.5}$$

Given the relation between the wave number and momentum for a free particle, $p = \hbar k$, Eq. (3.5) equals to the velocity of the incoming particle before it reaches the interaction region.

Likewise the outgoing flux from the spherical wave is given by

$$S_r = \mathcal{R}\left(\left(f(\theta)\frac{e^{-ikr}}{r}\right)\frac{\hbar}{i\mu}\frac{d}{dr}\left(f(\theta)\frac{e^{ikr}}{r}\right)\right) = \frac{v|f(\theta)|^2}{r^2} + \frac{\hbar|f(\theta)|^2}{i\mu r^3}.$$

For large r this reduces to approximately $\frac{v}{r^2}|f(\theta)|^2$. The expressions for in and outgoing current densities inserted in Eq. (3.2) relates the differential cross section to the scattering amplitude in the following way

$$\frac{d\sigma}{d\Omega} = \frac{r^2 S_r}{S_i} = |f(\theta)|^2. \tag{3.6}$$

3.2.4 Phase shifts

Above, we have seen how to relate an experimentally measurable quantity, the differential cross section, with the scattering amplitude. This section will show how to express the wave function in Eq. (3.4) so that it can be related to the scattering amplitude in a more direct way. For this a partial wave decomposition as given in Eq. (3.1) is preferable.

If the interaction between the nucleons is central the angular momentum of the system will be conserved. It is then convenient to separate the wave function in angular and radial parts. In a system that is spin-independent and thus spherically symmetric, the solution is independent of the azimuthal angle ϕ. This implies that M is zero in the spherical harmonics $Y_{LM}(\theta,\phi)$ for such a system. Inserting (3.1) in the time-independent Schrödinger equation and remembering that the eigenvalues for the spherical harmonics acting on the angular part is $L(L+1)$ gives

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dR_L}{dr}\right) + \left(\frac{2\mu}{\hbar^2}V(r) - k^2 - \frac{L(L+1)}{r^2}\right)R_L = 0,$$

which is a special case of the Bessel equation with solutions of linear combinations of the Bessel functions j_L and y_L [12]. In the case of a free particle, $V(r) = 0$, the only possible solution is $R_L = a_L j_L$ because y_L diverges at the origin,

$$\psi(r) = \sum_{L=0}^{\infty} a_L j_L(kr) Y_{L0} = \sum_{L=0}^{\infty} A_L P_L(\cos\theta),$$

where P_L represents the Legendre polynomials. In particular we have the plane wave expansion,

$$e^{ikz} = \sum_{L=0}^{\infty} (2L+1) i^L j_L(kr) P_L(\cos\theta).$$

Now consider the case when the potential has a finite range. Outside the region in which the potential acts we have the same case as above, with the exception of no longer being able to discard the y_l. This means that the long-range solution will instead be,

$$\psi(r) = \sum_{L=0}^{\infty} (a_L j_L(kr) + b_L y_L(kr)) P_L(\cos\theta).$$

The long range behaviour of the Bessel functions can be approximated by, [12]

$$j_L \xrightarrow{r\to\infty} \frac{\sin(kr - \frac{1}{2}L\pi)}{kr}$$
$$y_L \xrightarrow{r\to\infty} \frac{\cos(kr - \frac{1}{2}L\pi)}{kr}.$$

Using trigonometric identities we now introduce the angle δ_l defined as the phase shift, which makes it possible to express the wave function as,

$$\psi(r) = \sum_{L=0}^{\infty} \frac{C_L}{kr} \sin\left(kr - \frac{1}{2}L\pi + \delta_L\right) P_L(\cos\theta),$$

with the coefficient C_L being a combination of a_L and b_L. But we already know from Eq. (3.4) what the long range solution from scattering ought to look like. Inserting the plane wave expansion and comparing the coefficients gives,

$$\begin{aligned} f(\theta) &= \sum_{L=0}^{\infty} \frac{(2L+1)}{2ik} \left(e^{2i\delta_L} - 1\right) P_L(\cos\theta) \\ &= \sum_{L=0}^{\infty} (2L+1) f_L P_L(\cos\theta), \end{aligned}$$

with f_L being the partial wave scattering amplitude. Defining the S-matrix as $S_L(k) \equiv e^{2i\delta_L(k)}$ gives the partial wave scattering amplitude,

$$f_L = \frac{S_L(k) - 1}{2ik} = \frac{1}{k} \sin\delta_L(k)\, e^{i\delta_L(k)}.$$

Integrating the relation between the scattering amplitude and the differential cross section in Eq. (3.6) gives the total cross section as a sum over the partial wave amplitudes,

$$\sigma = 4\pi \sum_{L=0}^{\infty} (2L+1) |f_L|^2.$$

Even though it is not possible to measure the phase shift, this shows its direct relation to the observables of a scattering experiment.

3.2.5 The spin-scattering matrix

In the above analysis the spin-dependence of the scattering processes has been neglected. As previously mentioned nucleon-nucleon scattering experiments has to account for the spin of the particles. The main difference compared to spin-independent scattering is that the spin-scattering matrix needs to be defined in terms of the S-matrix. We will not go into details of the spin-dependent treatment but instead present some important relations used in the computer models.

The wave function in Eq. (3.4) has to be modified by redefining the scattering amplitude,

$$\begin{aligned}\boldsymbol{\Psi}_n &= e^{ikz}\boldsymbol{\chi}_n + \mathbf{f}_n\frac{e^{ikr}}{r}\\ \mathbf{f}_n &= M\boldsymbol{\chi}_n.\end{aligned}$$

Where $\boldsymbol{\chi}_n$ is a vector representing the nth initial spin state and $M(\boldsymbol{\sigma}_1, \boldsymbol{\sigma}_2, \mathbf{p}_i, \mathbf{p}_f)$ is a 4x4 matrix called the spin-scattering matrix depending on the in and outgoing momenta and the pauli spin operators. [6] The spin-scattering matrix relates to the S-matrix via the following definition,

$$M(\mathbf{p}_i, \mathbf{p}_f) = \frac{2\pi}{ik}\left\langle\theta_f\phi_f\right| S - 1 \left|\theta_i\phi_i\right\rangle, \tag{3.7}$$

where k is the wave vector for the relative motion of the particles and the bra and ket vectors specify the direction of motion for the in and outgoing nucleons.

The M-matrix elements expressed in partial wave basis for different channels is given in Appendix B.1. The important difference from the previous spin-independent treatment is the existence of several types of partial wave amplitudes and differential cross sections. It is also possible to measure the polarisation of the scattered particles resulting in a wide range of possible observables.

The Saclay representation of the scattering amplitudes, which is used in the computer models to calculate the observables, are given by the elements of the M-matrix as [13]

$$\begin{aligned}a =& \frac{1}{2}(M_{++} + M_{00} - M_{+-})\\ b =& \frac{1}{2}(M_{++} + M_{ss} + M_{+-})\\ c =& \frac{1}{2}(M_{++} - M_{ss} + M_{+-})\\ d =& \frac{(-M_{++} + M_{00} + M_{+-})}{2\cos\theta}\\ e =& \frac{i}{\sqrt{2}}(M_{+0} - M_{0+}).\end{aligned} \tag{3.8}$$

E.g. with this representation it is possible to express the differential cross section of an unpolarised beam and target as

$$\frac{d\sigma}{d\Omega} = \frac{1}{2}(|a|^2 + |b|^2 + |c|^2 + |d|^2 + |e|^2). \tag{3.9}$$

A full list of the different observables and a comprehensive treatment of the spin-dependent scattering formalism of nucleon-nucleon scattering is given in Ref. [6].

3.3 The Lippman-Schwinger equation

Above the connection between the scattering observables and the partial wave expansion of a wave function has been established. Now we present the formal solution to the scattering equation (3.3). This leads to a matrix equation with solutions related to the phase shifts.

In the Heisenberg matrix representation of quantum mechanics the equivalent of the time-independent Schrödinger equation is formulated as,

$$(\hat{H}_0 + \hat{V})\left|\psi_n\right\rangle = E_n\left|\psi_n\right\rangle.$$

Where the Hamiltonian operator $\hat{H}_0$ contains all information except the short range interactions between the nucleons. In the simplest case this will only be the kinetic energy but it could in general also include for example the coulomb interaction in the case of $p-p$ scattering.

The solution of,

$$(\hat{H}_0 - E_n)\,|\psi_n\rangle = \hat{V}\,|\psi_n\rangle\,,$$

gives the eigenfunction to the scattering equation. Assuming the matrix inversion exists the wave function can be expressed as,

$$|\psi_n\rangle = (E_n - \hat{H}_0)^{-1}\hat{V}\,|\psi_n\rangle\,.$$

Considering the free particle solution $\hat{V} = 0$ gives,

$$\hat{H}_0\,|\phi_n\rangle = \omega_n\,|\phi_n\rangle\,.$$

In the case where the Hamiltonian only consists of the kinetic energy, ϕ_n will be the plane wave solution. As the potential goes towards zero this has to be the solution obtained. Therefore, we can suggest the following equation,

$$|\psi_n\rangle = |\phi_n\rangle + (E_n - \hat{H}_0)^{-1}\hat{V}\,|\psi_n\rangle\,,$$

which is equivalent to the Schrödinger equation (3.3) with the suitable boundary conditions for scattering now already included in the the equation.

Multiplying with $\langle\phi_m|\,\hat{V}$ from the left gives

$$\langle\phi_m|\,\hat{V}\,|\psi_n\rangle = \langle\phi_m|\,\hat{V}\,|\phi_n\rangle + \langle\phi_m|\,\hat{V}(E_n - \hat{H}_0)^{-1}\hat{V}\,|\psi_n\rangle\,. \tag{3.10}$$

The wave functions ψ_n are unknown. Instead the transition matrix $\hat{T}$ is defined as the interaction expressed in the partial wave basis,

$$\langle\phi_m|\,\hat{T}\,|\phi_n\rangle = \langle\phi_m|\,\hat{V}\,|\psi_n\rangle\,.$$

Inserted in Eq. (3.10) yields,

$$\langle\phi_m|\,\hat{T}\,|\phi_n\rangle = \langle\phi_m|\,\hat{V}\,|\phi_n\rangle + \langle\phi_m|\,\hat{V}(E_n - \hat{H}_0)^{-1}\hat{T}\,|\phi_n\rangle\,, \tag{3.11}$$

which is one form of the Lippman-Schwinger equation and only dependent on ϕ_n. The completeness relation,

$$\mathbb{1} = \sum_n |\phi_n\rangle\,\langle\phi_n|\,, \quad \langle\phi_n\,|\,\phi_m\rangle = \delta_{n,m},$$

inserted in Eq. (3.11) leads to the integral equation

$$\langle\phi_m|\,\hat{T}\,|\phi_n\rangle = \langle\phi_m|\,\hat{V}\,|\phi_n\rangle + \sum_k \langle\phi_m|\,\hat{V}\,|\phi_k\rangle\,(E_n - \omega_k)^{-1}\,\langle\phi_k|\,\hat{T}\,|\phi_n\rangle\,, \tag{3.12}$$

which can be solved for $\hat{T}$ by matrix inversion.

3.3.1 The Lippman-Schwinger equation in a partial wave basis

Because of the positive energy for the scattering states it is possible to define the reaction matrix $R(k,k')^{J,S}_{LL'}$ as the real part of the complex T-matrix in Eq. (3.12). The solutions for uncoupled waves are given by

$$R^{J,S}_{LL'}(q',q) = V^{J,S}_{LL}(q',q) + \mathcal{P}\int_0^\infty dk\,k^2\frac{m}{q^2-k^2}V^{J,S}_{LL}(q',k)R^{J,S}_{LL}(k,q). \tag{3.13}$$

Where $\mathcal{P}$ indicates the Cauchy principal value. For a complete deduction and the solution of coupled cases see Ref. [8]. The solutions of the R-matrix equation are obtained by numerical matrix inversion techniques. The details of the procedure is given in Ref. [9]. Once the solution is obtained it is possible to express the phase shifts in the $R^{J,S}_{LL'}(k_0,k'_0)$-elements.

3.4 Partial wave decomposition of the interaction

The nucleon-nucleon interaction potential given in Section 2.4 needs to be decomposed in a partial wave basis in momentum space to be able to solve the Lippman-Schwinger R-matrix Eq. (3.13).

The projections on a partial wave basis of the scattering amplitudes of the interaction, $\mathcal{V}_\alpha$, is given as,

$$\mathcal{A}^{J(l)}(p',p) = \pi \int_{-1}^{1} dz \mathcal{V}_\alpha(p',p,z) z^l P_J(z)$$

by integrating over the Legendre polynomials, $P_J(z)$. With the help of these nucleon-nucleon amplitudes it is possible to decompose the interaction in different channels. The results are then used to solve the coupled and uncoupled Lippman-Schwinger R-matrix equations. The result of this decomposition is presented in Appendix B.2

3.5 Obtaining an observable from the chiral EFT potential

Here a short summary is given of how the theory in this chapter is explicitly used to calculate the theoretical observables such as a differential scattering cross section, $\mathcal{O}^{th}$.

If a interaction potential of LO is used, the starting point of the calculation is the interaction potential in Eq. (2.1) for the one pion exchange. This potential is then decomposed in a partial wave basis by projecting the interaction amplitudes on a partial wave basis. The general expressions for different channels are given according to Eq. (B.1). The result is used to solve the Lippman-Schwinger R-matrix equation (3.13). The scattering phase shifts will be determined by this solution and hence it is possible to compute the spin-scattering matrix M according to Eq. (3.7). The theoretically determined differential cross section are for example given by Eq. (3.9) which are dependent of the M-matrix solutions. In the next chapter it will be shown how these are compared with real measurable quantities to determine the value of the coupling coefficients through non-linear least-square fitting.

Chapter 4

Uncertainty Quantifications

In the previous chapters the theoretical model used to calculate physical observables by chiral EFT has been presented. This chapter aims to explain how coupling coefficients can be optimised using non-linear least-square methods, and how the Hessian of the goodness-of-fit measure χ^2 can be used to quantify the coefficient uncertainties.

4.1 Optimisation of the coupling coefficients

The observables of the theoretical model outlined in the previous chapter, such as Eq. (3.9), can be considered as functions denoted by $\mathcal{O}_i^{th}(\mathbf{c})$. Here, the index $i = 1, ..., N_d$ represents different observable quantities that the model tries to reproduce, whereas $\mathbf{c} = (c_1, ..., c_{N_c})$ represents the different coupling coefficients, or parameters, that are part of the model. As previously stated, these coefficients are optimised by using non-linear least-square to fit $\mathcal{O}_i^{th}(\mathbf{c})$ to the experimental $\mathcal{O}_i^{exp}$ values from the given set of N_d experimental data. To gauge the goodness of fit, the following chi-squared value is calculated

$$\chi^2(\mathbf{c}) = \sum_{i=1}^{N_d} \left(\frac{\mathcal{O}_i^{th}(\mathbf{c}) - \mathcal{O}_i^{exp}}{w_i} \right)^2. \tag{4.1}$$

The weight, w_i, corresponds to the uncertainty in the experimental value. Here we do not include the numerical or the theoretical uncertainty, but only the experimental one. The set of optimised coupling coefficients that minimise χ^2 is called $\mathbf{c}_{min}$. A good set of parameters has been found if $\chi^2(\mathbf{c}_{min})$ is approximately as big as N_d (it is a convention in the field to use N_d instead of $N_d - N_c$) [1]. The `nsopt` computer program uses the optimisation library `POUNDerS` [14] to find $\mathbf{c}_{min}$.

4.2 Generating error estimates from the Hessian

The Hessian matrix $\mathbf{H}(\mathbf{c})$ of the $\chi^2(\mathbf{c})$ is defined as

$$\mathrm{H}_{i,j}(\mathbf{c}) = \frac{1}{2} \frac{\partial^2 \chi^2(\mathbf{c})}{\partial c_i \partial c_j}.$$

Assume that the theoretically determined values, $\mathcal{O}_i^{th}(\mathbf{c})$, are only linearly dependent of the coupling coefficients, $\mathbf{c}$, in the region where $\chi^2(\mathbf{c}) \leq \chi^2(\mathbf{c}_{min})(1 + N_d^{-1})$. This gives that the covariance matrix, $\mathbf{C}$, of the coupling coefficients can be approximated by [15][16]

$$\mathbf{C} \approx \frac{\chi^2}{N_d} \mathbf{H}(\mathbf{c}_{min})^{-1}. \tag{4.2}$$

The accuracy of the assumption that $\mathcal{O}_i^{th}(\mathbf{c})$ is linear can be tested by studying $\chi^2(\mathbf{c})$, which should be quadratic if the assumption is correct.

In the covariance matrix, the diagonal elements are the variances whereas the covariances can be found off the diagonal. From the covariance matrix the statistical uncertainties, or the standard deviation, of the coupling coefficients can be calculated as

$$\Delta c_i = \sqrt{\mathrm{C}_{i,i}}. \tag{4.3}$$

The statistical uncertainty of a coefficient describes how much the value of that coefficient deviates from the correct value in average. In addition, the elements in the correlation matrix can be calculated from the covariance matrix as

$$\rho_{i,j} = \frac{\mathrm{C}_{i,j}}{\sqrt{\mathrm{C}_{i,i}\mathrm{C}_{j,j}}}. \tag{4.4}$$

The correlation between two parameters, c_i and c_j, quantify to what extent the values of these parameters correlate.

4.3 Error propagation

The covariance matrix can be used to determine errors of physical observables that are computed by using this model of the potential. To do this we need to propagate the parameter errors all the way to the observable. A multivariate distribution is used for this purpose. The associated probability density function looks like

$$f(\mathbf{c}) = (2\pi)^{-\frac{N_c}{2}} |\mathbf{C}|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{c}-\mathbf{c}_{min})'\mathbf{C}^{-1}(\mathbf{c}-\mathbf{c}_{min})}.$$

Calculating the physical observable for a collection of coupling coefficients generated with this multivariate normal distribution and then taking the standard deviation of the result will generate an uncertainty quantification. This method of propagating errors is easy to implement but it requires that the physical observable is cheap to calculate since it needs to be calculated a large amount of times. This is the method for propagating the errors that is used in this study.

Another way to calculate the errors of an arbitrary observable, A, is by using the following formula [15]

$$\Delta A = \sqrt{\sum_{i,j} \mathrm{G}_i^A \mathrm{C}_{i,j} \mathrm{G}_j^A},$$

where $\mathbf{G}^A$ is defined as

$$\mathbf{G}^A = \partial_{\mathbf{c}} A|_{\mathbf{c}_{min}}.$$

A disadvantage of this method is that it requires the derivatives of A with respect to the coupling coefficients. This method for propagating the errors is not used in this study, but could potentially be implemented with AD.

Chapter 5

Automatic Differentiation

Although the underlying principles of AD might have been used well before the invention of the digital computer, the dawn of modern AD can be traced back to the beginning of the 1960s. Early users of methods that today would be classified as AD included teams at the Lockheed Corporation and the General Electric Company. Both groups wrote computer programs calculating derivatives using so called forward mode AD, which is the conceptually simplest variety. Despite a continuing academic interest in AD into the 70s, especially at the University of Wisconsin-Madison, it failed to receive mainstream attention. However, in the beginning of the 80s, the development of new programming concepts such as operator overloading simplified the task of implementing AD. This in combination with the invention of reverse mode AD set the stage for a revival of the field. Work done by Andreas Griewank and others at the Argonne National Laboratory played an important role in many of the advances of this period. The interest in AD has not waned since then, as evident by the increasing number of applications. [17]

In essence, AD is based on the chain rule of elemental calculus,

$$\frac{dy}{dt} = \frac{dy}{dx}\frac{dx}{dt},$$

which is used to propagate derivatives in parallel with the ordinary calculations. This is possible due to the fact that all mathematical computations expressed in a programming language are implemented using a set of intrinsic functions (e.g. sin, exp, etc.) and operators ($+$, $*$, etc.) with well-known derivatives. In *forward mode* AD, the derivatives are propagated from the input parameters through the various intermediate variables all the way up to the final results. An alternative to this approach is to operate in *reverse mode*, in which, as the name suggests, the computational chain is instead transversed backwards from the final result down to the input variables. In either case, AD is believed [17] to have approximately the same precision as the calculations of the original non-derivative value.

The basic principles of AD as well as the use of the Rapsodia AD tool in Fortran are illustrated in this chapter, but for a more thorough treatment of the theoretical underpinnings of the subject, the reader is referred to other sources such as Ref. [18].

5.1 A simple example of forward mode AD

5.1.1 First-order derivatives

In order to elucidate the concept of AD, a simple expression such as

$$z = \exp(xy) + y\sin(x), \tag{5.1}$$

can be considered. This can be divided into several atomic operations performed on a series of intermediate variables (w_1, w_2, $w_3 \ldots$) in the following fashion:

$$w_1 = x$$

$$
\begin{aligned}
w_2 &= y \\
w_3 &= w_1 w_2 \\
w_4 &= \exp(w_3) \\
w_5 &= \sin(w_1) \\
w_6 &= w_2 w_5 \\
w_7 &= w_4 + w_6 \\
z &= w_7
\end{aligned} \tag{5.2}
$$

When actually implementing the expression in a high-level programming language, a programmer would of course opt for simply writing down (5.1) in code rather than the elaborate construction of (5.2). However, the latter is more general as more complicated computations where several subroutines are involved can be described by this framework, including calculations where the final result cannot be expressed in the input variables as a closed-form formula.

If we would be interested in the derivative of z with respect to some variable t, we would simply have to derive every step using the chain rule and the well known rules for differentiating elemental functions and operators

$$
\begin{aligned}
\dot{w}_1 &= \dot{x} \\
\dot{w}_2 &= \dot{y} \\
\dot{w}_3 &= \dot{w}_1 w_2 + w_1 \dot{w}_2 \\
\dot{w}_4 &= \dot{w}_3 \exp(w_3) \\
\dot{w}_5 &= \dot{w}_1 \cos(w_1) \\
\dot{w}_6 &= \dot{w}_2 w_5 + w_2 \dot{w}_5 \\
\dot{w}_7 &= \dot{w}_4 + \dot{w}_6 \\
\dot{z} &= \dot{w}_7.
\end{aligned} \tag{5.3}
$$

We have here used Newton's notation ($\dot{z} = \frac{dz}{dt}$, $\ddot{z} = \frac{d^2 z}{dt^2}$, etc.) for brevity. x and y might be functions of other variables, and more intermediate variables would in that case be needed to describe the entire computational chain all the way down to the input variables with respect to which we wish to differentiate. Alternatively, we might think of x and y as being supplied by some black box routine returning both the values of x and y as well as their derivatives, $\dot{x}$ and $\dot{y}$. As long as we have values and derivatives to feed in at the bottom of the chain, we will be able to retrieve the value and derivative of the final result.

If x and y are indeed the input variables, we would wish to retrieve the value of z as well as the values of the partial derivatives z_x and z_y given certain values of x and y. The act of assigning values to $\dot{x}$ and $\dot{y}$ to get the correct derivatives at the end of the chain is known as *seeding*. For example, if we would wish to derive z with respect to x, x will take on the role of the previously mentioned t, and the correct seeding values will thus be $\dot{x} = \frac{dx}{dt} = \frac{dx}{dx} = 1$ and $\dot{y} = \frac{dy}{dt} = \frac{dy}{dx} = 0$. Should we wish to arrive at z_y instead, the correct seeding would be $\dot{x} = 0$ and $\dot{y} = 1$.

It is easy to verify that these two sets of seeding values are indeed valid by inserting them into

$$
\dot{z} = (\dot{x}y + x\dot{y}) \exp(xy) + \dot{y} \sin(x) + y\dot{x} \cos(x),
$$

which does result in the correct partial derivatives

$$
\begin{aligned}
z_x &= y \exp(xy) + y \cos(x) \\
z_y &= x \exp(xy) + \sin(x).
\end{aligned}
$$

Finally, it might be appropriate to yet again emphasise that we do not need the formula (5.1) to arrive at the derivatives: if we want to compute z_x at e.g. $x = 2$ and $y = 7$, we simply input these values as well as the seeding values ($\dot{x} = 1$, $\dot{y} = 0$) into Eq. (5.2) and (5.3) and calculate all the intermediate variables and their derivatives until we arrive at $\dot{z} = \dot{w}_7$.

5.1.2 Second and higher-order derivatives

Computing higher-order derivatives can be done in a manner analogous to the procedure used for first-order derivatives. When differentiating the intermediate variables to the second order, the result is

$$\begin{aligned}
\ddot{w}_1 &= \ddot{x} \\
\ddot{w}_2 &= \ddot{y} \\
\ddot{w}_3 &= \ddot{w}_1 w_2 + 2\dot{w}_1\dot{w}_2 + w_1\ddot{w}_2 \\
\ddot{w}_4 &= \ddot{w}_3 \exp(w_3) + \dot{w}_3{}^2 \exp(w_3) \\
\ddot{w}_5 &= \ddot{w}_1 \cos(w_1) - \dot{w}_1{}^2 \sin(w_1) \\
\ddot{w}_6 &= \ddot{w}_2 w_5 + 2\dot{w}_2\dot{w}_5 + w_2\ddot{w}_5 \\
\ddot{w}_7 &= \ddot{w}_4 + \ddot{w}_6 \\
\ddot{z} &= \ddot{w}_7.
\end{aligned}$$

To calculate z_{xx} the correct seeding values are

$$\begin{aligned}
\dot{x} &= \frac{d}{dx}x = 1 & \dot{y} &= \frac{d}{dx}y = 0 \\
\ddot{x} &= \frac{d^2}{dx^2}x = 0 & \ddot{y} &= \frac{d^2}{dx^2}y = 0,
\end{aligned}$$

and by switching the values of $\dot{x}$ and $\dot{y}$, z_{yy} is produced instead.

It is less evident what seeding values should be used to arrive at the mixed derivative z_{xy}. A naive approach might be to set both $\dot{x} = 1$ and $\dot{y} = 1$ (in addition to $\ddot{x} = \ddot{y} = 0$). When this is inserted into

$$\begin{aligned}
\ddot{z} =& (\ddot{x}y + 2\dot{x}\dot{y} + x\ddot{y})\exp(xy) + (\dot{x}^2y^2 + 2xy\dot{x}\dot{y} + x^2\dot{y}^2)\exp(xy) \\
&+ \ddot{y}\sin(x) + 2\dot{x}\dot{y}\cos(x) + \ddot{x}y\cos(x) - y\dot{x}^2\sin(x),
\end{aligned}$$

the result is

$$\begin{aligned}
&2\exp(xy) + (y^2 + 2xy + x^2)\exp(xy) + 2\cos(x) - y\sin(x) = \\
&y^2\exp(xy) - y\sin(x) + 2(\exp(xy) + xy\exp(xy) + \cos(x)) + x^2\exp(xy),
\end{aligned}$$

but when compared to the real second-order derivatives

$$\begin{aligned}
z_{xx} &= y^2\exp(xy) - y\sin(x) \\
z_{xy} &= \exp(xy) + xy\exp(xy) + \cos(x) \\
z_{yy} &= x^2\exp(xy),
\end{aligned}$$

it can be seen to equal

$$\ddot{z} = z_{xx} + 2z_{xy} + z_{yy}.$$

We have thus arrived at a dilemma: if we want to compute mixed derivatives, we cannot set $\dot{x}$ nor $\dot{y}$ to zero, but this will invariably lead to the inclusion of z_{xx} (due to $\dot{x} \neq 0$) and z_{yy} (due to $\dot{y} \neq 0$). Therefore, we must interpolate the results of several different sets of seeding values in order to extract z_{xy}.

5.2 Reverse mode AD

As mentioned at the start of the chapter, AD can operate in reverse mode in addition to the forward mode described in the previous section. Furthermore, there also exist methods combining the two approaches.

To apply reverse mode AD to Eq. (5.1), the atomic operations in Eq. (5.2) must once again be differentiated, but this time in the opposite order:

$$z = w_7,$$

is first differentiated with respect to w_7, giving rise to the *adjoint*

$$\overline{w}_7 = \frac{\partial z}{\partial w_7} = 1.$$

The adjoints of the intermediate variables w_4 and w_6 immediately preceding $w_7 = w_4 + w_6$ are then given via the chain rule as

$$\begin{aligned}\overline{w}_6 &= \frac{\partial z}{\partial w_6} = \frac{\partial z}{\partial w_7}\frac{\partial w_7}{\partial w_6} = \overline{w}_7 \cdot 1 = \overline{w}_7 \\ \overline{w}_4 &= \cdots = \overline{w}_7,\end{aligned}$$

and the remaining adjoints can be shown to equal

$$\begin{aligned}\overline{w}_5 &= \overline{w}_6 w_2 \\ \overline{w}_3 &= \overline{w}_4 \exp(w_3) \\ \overline{w}_2 &= \overline{w}_6 w_5 + \overline{w}_3 w_1 \\ \overline{w}_1 &= \overline{w}_5 \cos(w_1) + \overline{w}_3 w_2.\end{aligned}$$

Note that the last two adjoints, $\overline{w}_1 = \frac{\partial z}{\partial w_1} = \frac{\partial z}{\partial x}$ and $\overline{w}_2 = \frac{\partial z}{\partial w_2} = \frac{\partial z}{\partial y}$, are the two sought partial derivatives, which have been evaluated with only one reverse sweep through the computational chain. This can be compared with forward AD, where one sweep for each partial derivative would have been needed. As a consequence, reverse mode AD is generally faster for functions

$$f : \mathbb{R}^m \to \mathbb{R}^n,$$

with $m \gg n$ (i.e. many independent and few dependent variables), whereas forward mode is advantageous for functions of the type $m \ll n$. A disadvantage of the reverse mode is the increased memory usage, as the calculated adjoints need to be stored in addition to the other variables.

5.3 Software implementations of AD

AD can be incorporated into software by manually adding derivative-calculating code throughout the program, but it is usually much more convenient to use a pre-existing AD tool. In general, these can be classified as using either source code transformation (SCT) or operator overloading (OO) to implement AD in the target program. [18]

SCT based tools edit the original source code automatically and add additional pieces of code, calculating derivatives, into each relevant subroutine.

In contrast, AD tools relying on operator overloading introduce new numerical data types, allowing variables to store their derivatives in addition to their original values. Intrinsic functions and operators are then overloaded to support the AD data types, with the overloaded methods also calculating derivatives (e.g. the overloaded $\sin(x)$ function does not only set $y = \sin(x)$ but also $\dot{y} = \dot{x}\cos(x)$). The only changes to the original source code the programmer will have to make is changing the data type of all variables in the computational chain from the input variables all the way up to the final result, although this might in itself be a daunting task.

5.4 A simple Fortran example using the Rapsodia AD tool

Rapsodia is an operator overloading-based AD tool, primarily developed by Isabelle Charpentier (currently at the Icube Laboratory in Strasbourg) and Jean Utke (at the Argonne National Laboratory), with

additional contributions from Alexis Malozemoff, Darius Buntinas and Mu Wang. In essence, Rapsodia is a code generator written in Python. Given the maximum order of the derivatives to compute as well as the number of directions (a term that will be discussed shortly), it will generate a Fortran 90 or C++ library. This library contains special numerical data types and the corresponding overloaded intrinsic functions and operators. [3]

In order to demonstrate the use of Rapsodia and similar tools, the simple Fortran program below is used as an example. The program computes the value of the variable z given the input values $x = -1.2$ and $y = 3.2$. Note that z is not expressed directly as a simple formula of x and y, as it takes a detour through the `SUMS` subroutine.

```
SUBROUTINE SUMS(x, y, squaresum, cubesum)
  IMPLICIT NONE
  REAL*8, INTENT(IN) :: x,y
  REAL*8, INTENT(INOUT) :: squaresum, cubesum
  squaresum = x**2+y**2
  cubesum = x**3+y**3
END SUBROUTINE

PROGRAM CALCULATE
  IMPLICIT NONE
  REAL*8 :: x, y, z, squaresum, cubesum

  x=-1.2
  y=3.2

  CALL SUMS(x, y, squaresum, cubesum)

  z=x*y**2+sin(2*x)-squaresum+cubesum
  WRITE(*,*) 'z=', z

END PROGRAM
```

The output of the `CALCULATE` program is

```
 z=   6.3965368745134015
```

In order to calculate the derivatives of z, the source code must be modified slightly:

```
SUBROUTINE SUMS(x, y, squaresum, cubesum)
  INCLUDE 'RAinclude.i90'
  IMPLICIT NONE
  TYPE(RARealD), INTENT(IN) :: x,y
  TYPE(RARealD), INTENT(INOUT) :: squaresum, cubesum
  squaresum = x**2+y**2
  cubesum = x**3+y**3
END SUBROUTINE

PROGRAM CALCULATE
  INCLUDE 'RAinclude.i90'
  IMPLICIT NONE
  TYPE(RARealD) :: x, y, z, squaresum, cubesum
  REAL*8 z_x,z_y,z_xx,z_yy

  x=-1.2
  ! Arguments: RAset(variable, direction, order, value)
```

```
  ! In direction 1, set the first order derivative of x to 1
  CALL RAset(x, 1, 1, 1D0)
  ! In direction 2, set the first order derivative of x to 0
  CALL RAset(x, 2, 1, 0D0)
  y=3.2
  CALL RAset(y, 1, 1, 0D0)
  CALL RAset(y, 2, 1, 1D0)

  CALL SUMS(x, y, squaresum, cubesum)
  z=x*y**2+sin(2*x)-squaresum+cubesum

  CALL RAget(z, 1, 1, z_x)
  CALL RAget(z, 1, 2, z_xx)
  z_xx=2*z_xx
  CALL RAget(z, 2, 1, z_y)
  CALL RAget(z, 2, 2, z_yy)
  z_yy=2*z_yy

  WRITE(*,*) 'z=', z%v
  WRITE(*,*) "z_x=", z_x
  WRITE(*,*) "z_y=", z_y
  WRITE(*,*) "z_xx=", z_xx
  WRITE(*,*) "z_yy=", z_yy

END PROGRAM
```

The new `CALCULATE` program will now display the derivatives z_x, z_{xx}, z_y and z_{yy} in addition to the value of z itself:

```
 z=   6.3965368745134015
 z_x=   15.485213183949114
 z_y=   16.640000400543215
 z_xx=  -6.4981478451910828
 z_yy=   14.800000190734863.
```

In the modified source code, the `INCLUDE 'RAinclude.i90'` statements refer to the library generated by Rapsodia, in which the `RARealD` data type is defined as well as the corresponding overloaded intrinsic functions and operators. This data type is then used for `x` and `y` and all intermediate variables all the way to `z`. `x` and `y` are set to -1.2 and 3.2 as usual, and the `RAset` subroutine is then used to seed the variables. As we want to calculate derivatives both with respect to x and y, we use two different directions, which are nothing more than sets of seeding values: in the first direction we set $\dot{x} = 1$ and $\dot{y} = 0$, whereas $\dot{x} = 0$ and $\dot{y} = 1$ applies in the second one.

When `x` and `y` have been properly seeded, the derivatives will be propagated through the entire computation. Both variables are instances of the `RARealD` type, and the overloaded operators `*`, `**`, `+` and `-` defined in the generated library will thus be used instead of intrinsic Fortran operators for `REAL*8`. The overloaded operators calculate derivatives in parallel with the original computations in accordance with the principles outlined in Section 5.1.

Finally, the `RAget` subroutine is called upon to extract the sought derivatives from the computed `z` variable. As the reader might have noticed, the second-order derivatives returned by `RAget` are multiplied by 2 before being presented to the user. This is done because the values returned by `RAget` (and those sent into `RAset`) are not the derivatives *per se*, but rather coefficients in the corresponding Taylor series,

$$f(a) + f'(a)h + \frac{f''(a)}{2!}h^2 + \frac{f'''(a)}{3!}h^3 + \dots.$$

As the second-order coefficient is actually $\frac{1}{2!} = \frac{1}{2}$ of the second-order derivative, the latter is produced by doubling the coefficient.

In summary, the following steps have to be taken to implement Rapsodia in Fortran programs such as `CALCULATE`:

- Find the input and output values of the calculation to identify the computational chain.
- Switch the data type of the input variables from `REAL` to `RARealD`, and do the same for all variables depending on these through the entire chain up to the output. For computations involving complex numbers, Rapsodia provides the `RAComplexD` data type.
- Seed the input variables with `RAset` with appropriate seeding values.
- Use `RAget` at the end of the chain to retrieve the derivatives.
- Modify other parts of the code, such as the `WRITE` statements in `CALCULATE`, to support the switch of data types.

5.4.1 Mixed derivatives

As previously stated in Section 5.1.2, interpolation must be used to extract mixed higher-order derivatives such as $\frac{d^2z}{dxdy}$. To relieve developers of the burden of implementing their own interpolation code, Rapsodia includes the `HigherOrderTensor` tool. This is based on a strategy for computing derivative tensors of higher orders outlined in [19].

To use this tool in a program in which Rapsodia is already implemented, such as the `CALCULATE` program of the previous section, modifications are only needed in two parts of the program: the seeding of the input variables, and retrieval of the final results.

Given the number of input variables n and the maximum derivative order o, the tool will return the number of directions needed. This number can be shown to equal $\binom{n+o-1}{o}$. To seed the input variables, a matrix generated by `HigherOrderTensor`, containing the appropriate seeding values for each variable in each direction, is used. To get the derivative tensor of the final result, the coefficients extracted from each directions with `RAget` are interpolated using a `HigherOrderTensor` subroutine.

More details on the `HigherOrderTensor` tool can be found in the Rapsodia user manual [20], but as it lacks a Fortran example, a modified version of the program included in the previous section is provided in Appendix C.1.

5.4.2 AD and external library routines

A recurring theme throughout this chapter has been the assumption that all mathematical operations expressed in a programming language only use a set of intrinsic functions and operators with well-known derivatives. In more complex programs, this is usually not entirely true as they often use functions provided by external libraries. Such non-intrinsic functions are usually not overloaded by Rapsodia and similar AD tools, making the implementation of AD in these programs more difficult. A possible solution would be to implement AD in the library itself. However, even if the developers have access to its source code, making such substantial changes in the internal machinery of a third party library is usually outside their field of expertise and is often too laborious to be practical.

In many cases, a better approach is to try to devise an analytical expression for the derivative of the mathematical operation performed by the library routine and uses this in the code. For example, the `nsopt` program at the centre of this thesis uses a matrix inversion routine from the well known LAPACK library as described in Section 3.3.1. This means that we wish to find the derivative of A^{-1}, which can easily be shown to equal

$$\frac{d}{dt}(A^{-1}) = -A^{-1}\frac{dA}{dt}A^{-1}, \tag{5.4}$$

whereas the second-order derivative can be obtained as

$$\frac{d^2}{dt^2}(A^{-1}) = 2A^{-1}\frac{dA}{dt}A^{-1}\frac{dA}{dt}A^{-1} - A^{-1}\frac{d^2A}{dt^2}A^{-1}. \tag{5.5}$$

If AD has been implemented in the computational chain all the way up to the matrix inversion, $\frac{dA}{dt}$ and $\frac{d^2A}{dt^2}$ are already available and the inverse A^{-1} can be computed using the original library routine. To arrive at the derivatives $\frac{dA^{-1}}{dt}$ and $\frac{d^2A^{-1}}{dt^2}$, we only need to put these values into Eq. (5.4) and (5.5), respectively. The derivatives can then be used to seed the elements of the inverted matrix and thereby starting a new computational chain continuing to propagate derivatives throughout the program.

For the sake of completeness, an example of a subroutine using a library routine for matrix inversion in combination with the analytic derivative approach described above is included in Appendix C.2.

Chapter 6

Method of Implementation

The `nsopt` program calculates a set of observables using chiral EFT (to the LO, NLO, NNLO or N3LO order) with a given set of values of the unknown parameters. These theoretical results are compared with low-energy scattering data. As mentioned in previous chapters, `nsopt` uses a non-linear least-square method, resulting in a goodness-of-fit measure χ^2. Furthermore, `nsopt` also contains components for finding an optimal fit of the parameters (i.e. minimising the χ^2 value) against a certain set of experimental data.

A major part of the work behind this thesis has been to implement automatic differentiation in `nsopt` using Rapsodia, as described in Chapter 5. This allows the computation of first and second order partial derivatives of χ^2 with respect to the different parameters, which can be used to calculate error estimates as described in Chapter 4. In this chapter, the general structure of `nsopt` will be explained as well as the modification required to support the computation of χ^2 derivatives.

6.1 The computational chain of nsopt

`nsopt` is mainly written in Fortran with some more recent parts written in C, although the latter have not been modified in the course of this project. In the following text, the main steps along the computational chain will be described. A figure over the workflow is shown in Figure 6.1.

First, the parameters of the chiral EFT are set to user-specified values and the elements of the potential matrix V (as found in the Lippman-Schwinger equation (3.11)) are computed for a large number of different linear momenta. In the next step, a reaction matrix R is constructed according to Eq. (3.13). This makes it possible to calculate phase shifts using a matrix inversion as described in Section 3.2.4. `nsopt` uses a routine from the LAPACK library to perform the matrix inversion. The partial wave amplitudes are then calculated as outlined in the same section, allowing the spin-scattering matrix to be determined using Eq. (3.7). The program can subsequently evaluate the observables from the scattering matrix using Saclay representation, as described in Eq. (3.8). Finally, these calculated observables are compared to experimental proton-proton and neutron-proton scattering data, resulting in a χ^2 value as defined in Eq. (4.1).

6.2 Implementation of AD in the nsopt program

As seen in Chapter 3 as well as the previous section, the computation of the χ^2 value is a non-trivial process involving a series of intricate steps. Consequently, it was not possible to implement AD in one stroke and a more gradual approach had to be chosen. AD was implemented from the bottom up, one subroutine at a time. This combined with the approach to only add AD support at one chiral order at a time made the implementation consist of small, almost independent steps. This strategy allowed the AD computed derivatives to be cross-checked with finite difference calculations after each step, making errors easier to locate.

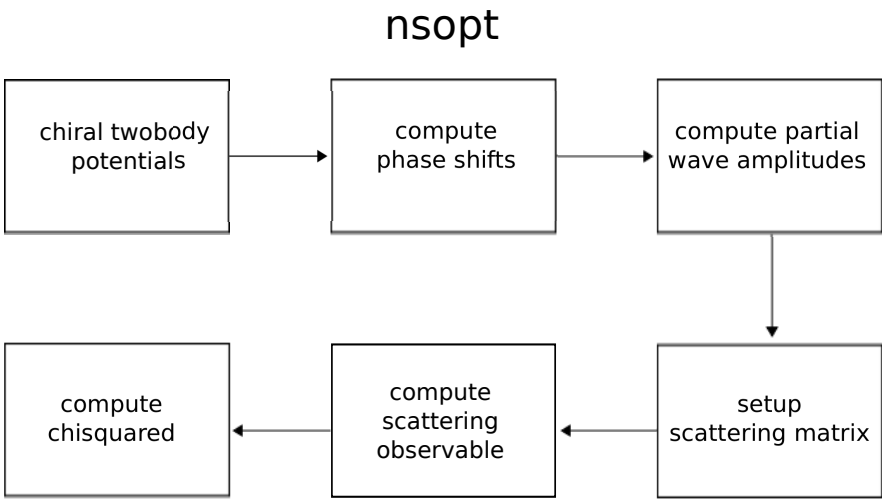


Figure 6.1: A diagram showing the workflow of the computer model in `nsopt` . The names in the boxes refer to different parts or subroutines of the program.

Most of the programming work consisted of the tedious task of changing data types from `REAL*8` to `RARealD` and `COMPLEX*16` to `RAComplexD` throughout the computational chain. A more specific challenge was the implementation of AD support in a matrix inversion routine as described in Section 5.4.2, as well as in routines for computing the absolute value, conjugate and extracting the imaginary parts of `RAComplexD` types.

When first order derivatives of χ^2 could be calculated correctly, adding support for second order derivatives did not present much of a problem.

A final task was the seeding and interpolation of the second order derivatives at the beginning and end of the computational chain, respectively. This required the `HigherOrderTensor` tool described in Section 5.4.1.

6.3 Requirements on the data imposed by the method

Since this thesis aims to calculate derivatives of χ^2 the computer program must consist of functions that are differentiable with respect to the parameters. If data points at a scattering angle of 180° in the centre of mass system are used, the computation of the corresponding theoretical observables will include the absolute value of 0. As the derivative of this function at 0 is not defined, calculations involving such data points will result in `nsopt` returning `NaN` (Not a Number) as the values of the derivatives. Consequently, no calculation at this particular scattering angle can be used in the AD enabled program.

Another obstacle that was encountered during the implementation phase was the difficulty of finding acceptable minima. If there are any cusps at, or very close to the minimum, inverting the Hessian generates covariance matrices with negative variances. The reason for this is that the approximation that is made in Eq. (4.2) is incorrect if the $\chi^2(\mathbf{c})$ is not quadratic.

6.4 Propagating errors to the deuteron

The two-nucleon system of the deuteron is a suitable test for the propagation of errors from the coupling coefficients to many-body physics calculations. The `nsopt` program may also be utilised for the calculation of bound state energies. By sampling the parameter space using a multivariate Gaussian distribution for both the LO and NLO coupling coefficients the binding energy, E_b, was calculated for each set of parameters. From this the standard error, ΔE_b, was determined. Multivariate Gaussian distributions are described in Section 4.3.

Chapter 7

Results

The results detailed in this chapter were derived from Hessians of the χ^2 measure with respect to the coupling coefficients. These Hessians were computed with AD at one LO and two NLO minima according to the principles of Chapter 5 and Chapter 6. The minima were optimised against neutron-proton scattering data, with data points at energies up to 4 MeV and 75 MeV used for LO and NLO, respectively. For LO and one of the NLO minima the cutoff, Λ, was set to 500 MeV. In contrast, $\Lambda = 450$ MeV was used for the other NLO minimum.

Given the Hessians, covariance matrices could be calculated. These matrices were used to obtain statistical uncertainties of, and correlation matrices between, the coupling coefficients. Finally, these uncertainties were propagated to the binding energy of the deuteron. In addition to the results mentioned above, a comparison between AD and numerical differentiation and a demonstration of the linearity of the calculated observables, as required by the method, are also presented in this chapter.

7.1 Statistical uncertainties of the coefficients

The statistical uncertainties in the form of the standard deviations of the coupling coefficients were generated from the Hessians by using Eq. (4.2) and (4.3) for the examined minima.

7.1.1 Leading-order

The statistical uncertainties for the LO minimum and the corresponding coupling coefficients can be seen in Table 7.1. The uncertainties are small for both of the LO parameters.

Table 7.1: The statistical uncertainties of the coupling coefficients for LO.

Constant	c_i	Δc_i	$\lvert\frac{\Delta c_i}{c_i}\rvert$
Ct_1S0	-0.1073	$1.795 \cdot 10^{-5}$	0.0167%
Ct_3S1	-0.0710	$5.024 \cdot 10^{-5}$	0.0708%

7.1.2 Next-to-leading-order

The statistical uncertainties of the NLO minimum with $\Lambda = 500$ MeV can be seen in Table 7.2. Most of the uncertainties are in the order of 0.01. Some of the parameters have larger statistical errors, possibly due to the nature of the used scattering data: If the scattering data points used are less dependent on some of the parameters, these are not as influential in the determination of χ^2. This would presumably result in larger statistical uncertainties.

Table 7.2: Statistical uncertainties of the coupling coefficients at the NLO minimum with $\Lambda = 500\,\text{MeV}$.

Constant	c_i	Δc_i	$\lvert\frac{\Delta c_i}{c_i}\rvert$
Ct_1S0np	-0.1502	$4.057 \cdot 10^{-4}$	0.270%
Ct_3S1	-0.1460	$2.545 \cdot 10^{-3}$	1.743%
C_1S0	1.7016	$4.280 \cdot 10^{-2}$	2.516%
C_3P0	1.4703	$4.864 \cdot 10^{-2}$	3.308%
C_1P1	0.4836	$3.982 \cdot 10^{-2}$	8.235%
C_3P1	-0.0521	$9.687 \cdot 10^{-2}$	185.9%
C_3S1	-0.5893	$1.915 \cdot 10^{-2}$	3.249%
C_3S1-3D1	0.0391	$1.677 \cdot 10^{-2}$	42.94%
C_3P2	-0.1703	$4.639 \cdot 10^{-3}$	2.724%

The statistical errors and values of the coupling coefficients for the NLO minimum with $\Lambda = 450\,\text{MeV}$ are presented in Table 7.3. Comparing this table with that for the other NLO minimum we can see that each pair of coupling coefficients has statistical uncertainties of roughly the same size.

Table 7.3: Statistical uncertainties of the coupling coefficients at the NLO minimum optimised using $\Lambda = 450\,\text{MeV}$.

Constant	c_i	Δc_i	$\lvert\frac{\Delta c_i}{c_i}\rvert$
Ct_1S0np	-0.1560	$7.319 \cdot 10^{-4}$	0.469%
Ct_3S1	-0.1593	$1.949 \cdot 10^{-3}$	1.224%
C_1S0	1.6422	$3.298 \cdot 10^{-2}$	2.008%
C_3P0	1.3717	$3.659 \cdot 10^{-2}$	2.668%
C_1P1	0.4941	$3.476 \cdot 10^{-2}$	7.035%
C_3P1	-0.0149	$8.146 \cdot 10^{-2}$	545.2%
C_3S1	-0.5378	$1.720 \cdot 10^{-2}$	3.199%
C_3S1-3D1	0.0701	$1.534 \cdot 10^{-2}$	21.89%
C_3P2	-0.1899	$5.434 \cdot 10^{-3}$	2.862%

7.2 Correlations of the coupling coefficients

Correlation matrices were calculated from the Hessians by using Eq. (4.2) and (4.4). This was done for both the LO minimum and the two NLO minima.

7.2.1 Leading-order

The correlation between the two parameters at the LO minimum was calculated as 0.79, meaning that the parameters are positively correlated. In Figure 7.1 this correlation is illustrated.

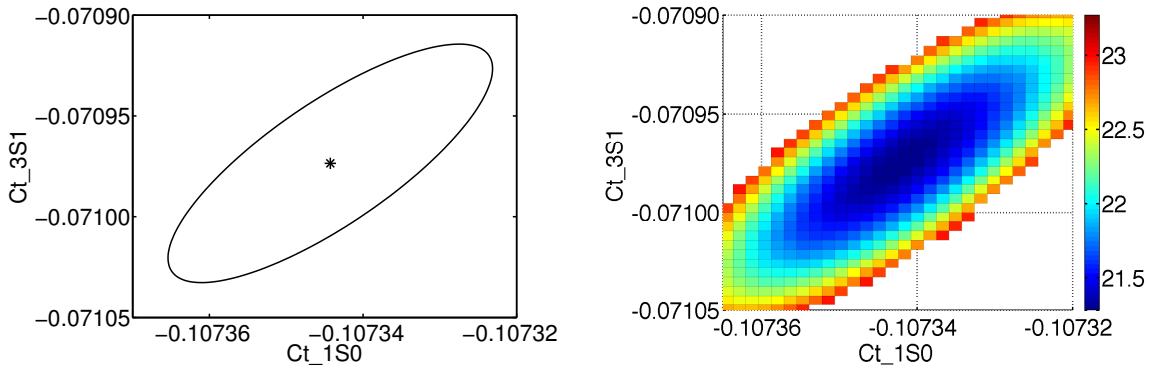


Figure 7.1: The figures illustrate how the Ct_1S0 and Ct_3S1 coupling coefficients correlate for LO. The value of the correlation coefficient is 0.79. The left figure shows the correlation ellipse for the pair of parameters and the right figure displays the variation in χ^2 as the coupling coefficients vary.

Table 7.4: The correlations for the coupling coefficients of the NLO minimum (with $\Lambda = 500\,\text{MeV}$) that is presented in Table 7.2.

Constant	Ct_1S0np	Ct_3S1	C_1S0	C_3P0	C_1P1	C_3P1	C_3S1	C_3S1-3D1	C_3P2
Ct_1S0np	1	0.32	0.98	−0.23	0.49	0.17	−0.26	−0.28	−0.17
Ct_3S1	0.32	1	0.26	0.22	−0.40	0.03	−0.78	−0.89	0.44
C_1S0	0.98	0.26	1	−0.31	0.54	0.24	−0.15	−0.27	−0.21
C_3P0	−0.23	0.22	−0.31	1	−0.61	−0.84	−0.60	0.11	0.58
C_1P1	0.49	−0.40	0.54	−0.61	1	0.52	0.47	0.24	−0.32
C_3P1	0.17	0.03	0.24	−0.84	0.52	1	0.50	−0.41	−0.13
C_3S1	−0.26	−0.78	−0.15	−0.60	0.47	0.50	1	0.40	−0.38
C_3S1-3D1	−0.28	−0.89	−0.27	0.11	0.24	−0.41	0.40	1	−0.36
C_3P2	−0.17	0.44	−0.21	0.58	−0.32	−0.13	−0.38	−0.36	1

7.2.2 Next-to-leading-order

The NLO Hamiltonian has 9 coupling coefficients, resulting in 36 different pairwise correlation coefficients. Table 7.4 shows the correlation matrix at the NLO minimum as in Table 7.2. In Figure 7.2 we can see the correlation between two pairs of coupling coefficients expressed as correlation ellipses.

The correlation between the Ct_1S0np and C_1S0 is close to 1. Thus, at the χ^2 minimum, Ct_1S0np is to a large extent determined by the value of C_1S0, and vice versa. The correlation between these parameters are illustrated in Figure 7.2.

Another feature is that the correlation coefficients with large (either positive or negative) values tend to correspond to channels with some quantum numbers in common. For example the Ct_3S1 coupling coefficient has a large negative correlation with C_3S1 and C_3S1-3D1, with values of −0.78 and −0.89 respectively. The coupling coefficients Ct_1S0np and C_1S0 are another example, as well as C_3P0 and C_3P1, which share the same spin and angular momentum quantum state and have a correlation of −0.84. However, there are examples that contradict this pattern: For example, the C_3P1 and C_3P2 coupling coefficients, which have a correlation of merely −0.13.

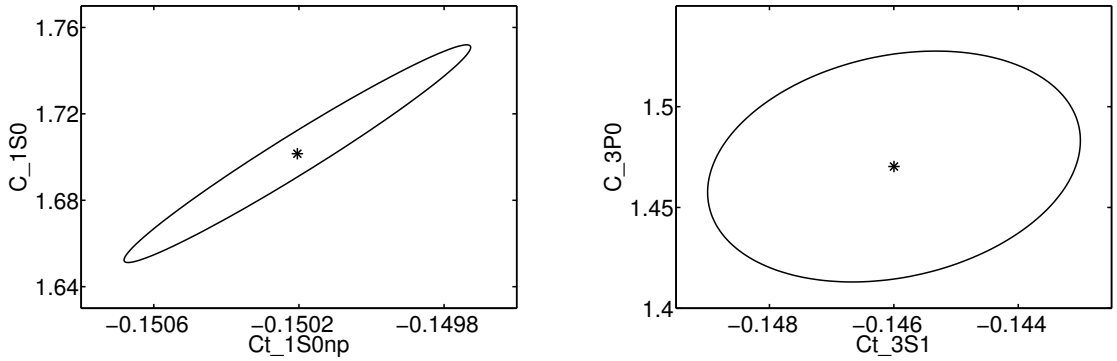


Figure 7.2: The correlation ellipses for two pairs of coupling coefficients for the NLO minimum stated in Table 7.2. The correlation coefficients for the two pairs that are displayed in the two graphs are, from left to right, 0.98 and 0.22.

Table 7.5: The correlation-matrix of the NLO minimum (with $\Lambda = 450\,\text{MeV}$) that is presented in Table 7.3

Constant	Ct_1S0np	Ct_3S1	C_1S0	C_3P0	C_1P1	C_3P1	C_3S1	C_3S1-3D1	C_3P2
Ct_1S0np	1	−0.18	−0.99	0.34	−0.58	−0.27	0.07	0.25	0.22
Ct_3S1	−0.18	1	0.23	0.36	−0.43	−0.11	−0.84	−0.79	0.45
C_1S0	−0.99	0.23	1	−0.31	0.56	0.24	−0.13	−0.26	−0.20
C_3P0	0.34	0.36	−0.31	1	−0.60	−0.82	−0.63	0.08	0.57
C_1P1	−0.58	−0.43	0.56	−0.60	1	0.50	0.46	0.24	−0.31
C_3P1	−0.27	−0.11	0.24	−0.82	0.50	1	0.54	−0.41	−0.08
C_3S1	0.07	−0.84	−0.13	−0.63	0.46	0.54	1	0.34	−0.35
C_3S1-3D1	0.25	−0.79	−0.26	0.08	0.24	−0.41	0.34	1	−0.39
C_3P2	0.22	0.45	−0.20	0.57	−0.31	−0.08	−0.35	−0.39	1

In Table 7.5 the correlation of the other NLO minimum investigated (with $\Lambda = 450\,\text{MeV}$ and coupling coefficients as in Table 7.3) is shown. Comparing the correlation coefficients in Table 7.4 and 7.5 we can see a pattern: For most of the elements the correlation is quite similar between the two different sets of coupling coefficients e.g. C_1S0 and C_3S1. For others the correlation differs with just the sign, for example Ct_1S0 and C_1S0. However, there are also pairs of coupling coefficients that invariably have values far from each other, such as Ct_1S0 and C_3S1.

7.3 Statistical uncertainty in deuteron calculations

After having obtained the standard errors of the chiral EFT coefficients, it is possible to propagate these statistical uncertainties to nuclear observables. The deuteron is the simplest bound nuclear system and therefore constitutes a suitable test.

In Table 7.7 the amount of samples needed to accurately obtain the propagated error was examined. 10 scans were performed with sample sizes of 100, 1000 and 10000 respectively. The samples were generated using a multivariate Gaussian distribution determined by the correlation matrices in Section

7.2. The binding energy E_b was calculated for each set of parameters and the standard error determined by $\Delta E_b = \mathrm{Std}(E_b)$. A sample size for which the standard error is constant, i.e the standard deviation of ΔE_b is small, will display the statistical uncertainty correctly. The results indicate that a sample size above 10^4 is preferable.

Table 7.6: The standard deviation of ΔE_b for different sample sizes. 10 different samples of each size were used to calculate the standard deviation.

Sample size	$\mathrm{Std}(\Delta E_b)$ [keV]	
	LO	NLO
100	$4.652 \cdot 10^{-1}$	$9.415 \cdot 10^{-1}$
1000	$1.733 \cdot 10^{-1}$	$2.829 \cdot 10^{-1}$
10000	$3.434 \cdot 10^{-2}$	$8.759 \cdot 10^{-2}$

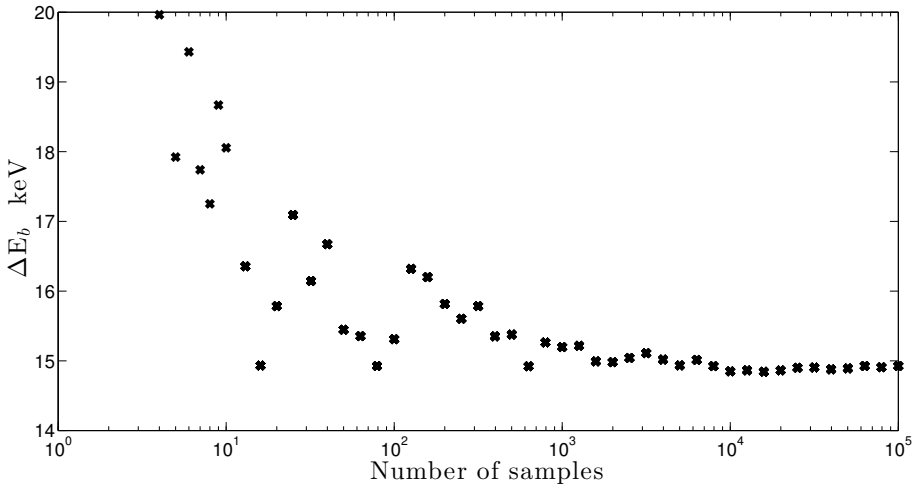


Figure 7.3: The standard deviation of E_b for a 10^5 sample size. The deviation is plotted against the increasing number of samples. Although a point depends on all previous values the standard deviation is constant between 10^4 to 10^5 samples. This indicates that 10^5 samples are enough to obtain the propagated error.

To obtain the statistical uncertainty of the deuteron E_b a sample of 10^5 points was used. Figure 7.3 shows how ΔE_b varies with the amount of samples, supporting our assessment that a sample size over 10^4 points is enough.

The calculated binding energy, $\overline{E}_b$, and standard deviation, ΔE_b, is presented in Table 7.7. For both orders, the relative error are below one percent of $\overline{E}_b$. The deuteron binding energy E_b is known to be 2.22 MeV. However, the discrepancy between this and our results is due to the fact that the interaction potential was not optimised against the binding energy of the deuteron. The results show small deviations indicating a low statistical uncertainty using chiral EFT in deuteron calculations.

At LO the deuteron is only dependent on the ${}^3S_1$-partial wave while at NLO it is dependent on three of the coupling coefficients. The fact that the LO standard error is smaller than the NLO one is most likely due to this fact.

Table 7.7: Calculation of the deuteron binding energy for 10^5 samples and its standard error for both LO and NLO.

Order	N samples	$\overline{E}_b$ [MeV]	ΔE_b [keV]	$\lvert\frac{\Delta E_b}{\overline{E}_b}\rvert$
LO	10^5	-2.113	6.504	0.3078%
NLO	10^5	-2.113	14.93	0.7065%

7.4 AD compared with finite difference methods

To ensure the correctness of the derivatives computed with automatic differentiation, they were compared with results from finite difference methods. Due to the inherent rounding errors of the latter the AD values could never be fully replicated, but the numerical derivatives were close enough to verify that the AD enabled `nsopt` version did indeed give correct derivatives.

Table 7.8 compares AD derivatives at an NLO-minimum with corresponding finite difference derivatives calculated with step lengths h of decreasing size. Initially, the numerical derivatives approach the AD computed values as h decreases, confirming the validity of the AD results. However, when the step length goes below a certain value (10^{-4} and 10^{-6} for the first and second order, respectively) the results of the two methods start to diverge due to rounding errors, showing the precision advantage of automatic differentiation.

It should be mentioned that the numerical derivatives in Table 7.8 were computed with an external Python script reading from the output of `nsopt`, and that some additional rounding errors occur when `nsopt` prints the computed χ^2 values. However, this does not change the fundamental fact that finite difference methods are numerically ill-conditioned for small values of h. This was evident at several stages in the AD implementation in `nsopt` when the derivatives were cross-checked with numerical derivation implemented into the program itself.

Table 7.8: A comparison of χ^2 derivatives in an NLO-minimum computed with AD and by finite difference. The table lists the first and second order derivatives with respect to C_3P0, as well as the mixed second-order derivative with respect to C_3P0 and C_3P1. The finite difference approximation has been computed for various step lengths h, with the approximative values closest to the AD derivatives in bold.

h	First	Second	Mixed
AD	-0.40436387006849017	17381.523865263011	6462.9897938677113
10^{-2}	-2.25177102686	17387.1881932	6465.35799077
10^{-3}	-0.42283434891	17381.5805024	6463.01347297
10^{-4}	-0.404548562756	**17381.5246626**	**6462.98993274**
10^{-5}	-0.404365698614	17381.5465132	6462.99213258
10^{-6}	**-0.404364072892**	17379.5342562	6462.64197712
10^{-7}	-0.404369302487	17394.0861714	6463.09672447
10^{-8}	-0.404520505981	40927.2615798	10231.8153949
10^{-9}	-0.403133526561	2273736.75443	966338.120634
10^{-10}	-0.410409484175	68212102.633	17053025.6582

7.5 The credibility of the linear approximation

As stated in Section 4.2 the theoretical model has to be approximated as a linear function to be able to calculate the covariance matrix using Eq. (4.2). A linear theoretical model results in quadratic χ^2 function. Figure 7.4 displays how χ^2 varies with the Ct_3S1 parameter in the relevant range for the NLO minimum with $\Lambda = 500\,\text{MeV}$, as well as a fitted quadratic curve. The sum of squared errors of prediction (SSE) for the 98 values making up the diagram is $2.34 \cdot 10^{-5}$ and the residual sum of squares (R^2) is 1.00, indicating a very good fit.

This investigation is only done with respect to one of the coupling coefficients. It is nevertheless plausible to suppose that the result would be similar if one conducted the survey with any of the other parameters. The reason for this is that similar, but not so carefully produced charts, of other coupling coefficients all have a quadratic appearance. Similarities in how the coupling coefficients are included in the model also support this reasoning. The conclusion drawn by this examination is that the linear approximation is sound.

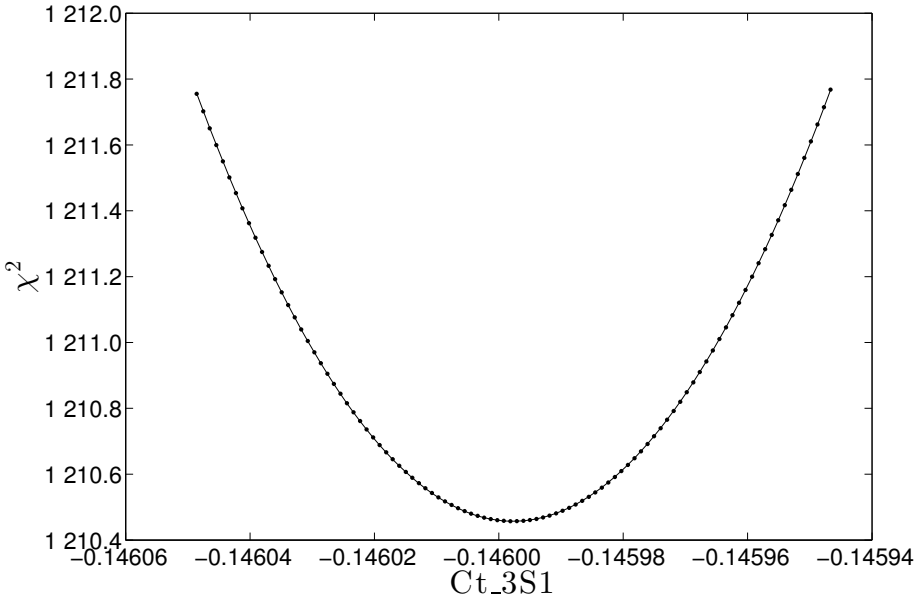


Figure 7.4: The change in χ^2 as Ct_3S1 varies. The dots are the values determined by the chiral EFT model, while the line is a quadratic curve fitting of those values. The similarity between the measured values and the curve fitting shows that the approximation concerning the covariance matrix made in this report is correct.

Chapter 8

Discussion

In this chapter we discuss how the results obtained in this study can be interpreted. Whereas the conclusions drawn in the previous chapter are more specific, often pertaining to a table or a figure, the conclusions presented and substantiated here are more general. A central aim of this study was to investigate whether it is feasible to use AD to obtain statistical uncertainties for the coupling coefficients of a chiral EFT model. This is discussed as well as the limitations of this method. Furthermore, the possibility of extending the computer program further and thus produce additional results is covered. Lastly we suggest how the generated error estimates and derivatives obtained by AD could be used in further applications.

8.1 Computing statistical uncertainties with AD

As previously demonstrated it is indeed possible to obtain error estimates for the parameters of LO and NLO for neutron-proton interactions by using AD. Although we have not implemented this approach for N3LO or proton-proton interactions there are no obvious obstacles preventing such future developments. However, there are a few issues that have to be taken into account when using the outlined method:

- The use of some data records will result in a χ^2 that is non-differentiable, and the computed derivatives will therefore be undefined and set to `NaN` (Not a Number) by the AD enabled `nsopt` program. This seems to be a minor problem and we have only needed to remove two out of over 500 data records.
- The computer program that generates the χ^2 must not contain any inaccuracies that could create cusps in the $\chi^2(\mathbf{c})$ function. These cusps risk not only to complicate the minimisation procedure, but also to prohibit the use of the chosen method of finding covariance matrices, since in doing so the $\chi^2(\mathbf{c})$ needs to be quadratic around the minimum.
- The application of the AD tool Rapsodia to a non-trivial program requires a data type change of all variables from the input parameters to the final result throughout the computational chain. To keep the resulting errors few and easy to locate, it is recommended to implement AD in a stepwise manner and cross-check the resulting derivatives with finite difference methods as often as practically possible.

8.2 Requirements on the data imposed by the method

As discussed in Section 6.3 and the section above there are two requirements of the outlined method for generating covariance matrices that needs to be taken into account when using it: data points at scattering angles of 180° cannot be used, and there must not be any cusps or discontinuities around the minima. This section discusses how these problems might be avoided.

The problem with `NaN` derivatives arise at scattering angles of 180°. At these data records the resulting calculations will include the absolute value of 0, where the absolute value function is not differentiable. Therefore, a singularity will result when `nsopt` tries to calculate its derivative. There is a possibility that this singularity is removable. If this is the case it would be possible to replace all scattering angles at this value with an angle that is very close to 180° and still obtain a correct χ^2 as well as its derivatives.

The problem with cusps in the $\chi^2(\mathbf{c})$ function is more difficult. The only way to proceed is to somehow remove these cusps. This could be done by removing the problematic data records and then minimising the χ^2 again. It can also be done by trying to find a new local minimum that preferably is smaller, with the same data records.

8.3 Possible speed gain by using reverse-mode AD

As noted in Section 5.2, reverse-mode AD tend to be faster than forward-mode AD when seeking partial derivatives of few dependent variables with respect to many input parameters. `nsopt` calculates a single χ^2 value as a function of the parameters in the chiral EFT, which are numerous, especially at higher orders. It is therefore possible that the program execution time might be significantly reduced by implementing reverse mode AD in `nsopt`. The Rapsodia user manual [20] mentions reverse-mode AD without offering any examples of its use, and the difficulty of applying this approach to `nsopt` thus remains uncertain.

8.4 Alternative ways to generate the covariance matrix

In this study, Eq. (4.2) has been used to approximate the covariance matrix. There are two other approximations that can be used. One of them only needs first-order derivatives, whereas the other one uses both first- and second-order derivatives. For further information, see Ref. [16].

These two other ways of approximating the covariance matrix have not been investigated in this survey due to the limited time available, but this could be an interesting future goal for two reasons: Firstly, it could result in a faster method for generating the covariance matrices, since only first-order derivatives needs to be calculated for one of them. Secondly, it might also add some extra insight concerning the validity of these approximations, since a comparison between the results generated could be conducted.

8.5 AD derivatives for optimisation

As mentioned in Chapter 4 the `nsopt` program uses the optimisation tool `POUNDerS` to find a minimum of the χ^2 value. This optimisation algorithm does not use derivatives since they were previously not available to the developers. One interesting idea could be to employ the derivatives computed with AD to both speed up the optimising process as well as use it to find new and possibly better minima. A technique that uses derivatives in the optimising process is the Levenberg–Marquardt algorithm [21]. This method makes use of the derivatives found in this thesis, and would probably improve the optimising process.

8.6 Statistical uncertainty in many-body physics observables

Obtaining error estimates of the chiral EFT coefficients presents the possibility of propagating statistical uncertainties to nuclear many-body observables. As exemplified in the deuteron case this is indeed possible. However, the methods used in this thesis for the error propagation might not be optimal. Our conclusion that 10^5 samples, maybe even fewer, are enough to obtain a statistical certainty in the solution was obtained by scanning the nine-dimensional parameter space. This process will be computationally demanding for more complex systems, or for higher orders as NNLO and N3LO. Therefore, other methods

such as error propagation using derivatives (briefly explained in Section 4.3) or machine learning need to be examined.

Chapter 9

Conclusions and Recommendations

This chapter summarises the conclusions drawn in this thesis. More details concerning the conclusions, as well as justifications, can be found in Chapters 7 and 8. The conclusions presented are divided into three different categories: those discussing the possibility of implementing AD in chiral EFT, more general conclusions about chiral EFT, and lastly an outlook for what this method can contribute to in the future.

9.1 The feasibility of the method

- AD can be implemented in the chiral EFT computer model to obtain derivatives of χ^2 with respect to the coupling coefficients.
- By using the Hessian obtained by AD, it is possible to calculate the covariance matrix. This covariance matrix can be used to calculate error estimates of, as well as correlations between, the coupling coefficients.
- The calculation of the covariance matrix requires the minimum to be free from cusps. If not, the approximation made is invalid resulting in unreasonable variances.
- It is possible to propagate the error estimates up to physical observables, such as the binding energy of the deuteron, by sampling the parameter space with the covariance matrix.

9.2 Discovered features of chiral EFT

- The error estimates for LO coupling coefficients are of the order 0.01%.
- NLO coupling coefficients have statistical uncertainties in the order of 1%, although some of the parameters deviate from this.
- The uncertainty of the deuteron binding energy was calculated to be 0.3% for an LO chiral EFT and 0.7% for an NLO one.
- There seems to be a higher correlation between coupling coefficients describing scattering channels with some quantum numbers in common.

9.3 Outlook

- It is most likely possible to obtain the Hessian for proton-proton data as well as for higher chiral orders.
- There exists alternative ways to approximate the covariance matrix, one of them only requires first-order derivatives which would decrease the computational time.

- A technique that could be investigated is to use the derivatives for speeding up the optimisation process. An algorithm that probably would fit this purpose is the Levenberg–Marquardt algorithm.
- By implementing reverse-mode AD instead of forward-mode AD it is probably possible to shorten the computational time.

References

[1] R. Machleidt, D. R. Entem, Chiral effective field theory and nuclear forces, Phys. Reports 503 (1-75).

[2] P. Lepage, How to renormalize the schrodinger equation, arXiv preprint nucl-th/9706029.

[3] Rapsodia website, `http://www.mcs.anl.gov/Rapsodia/`, accessed: 2014-05-03.

[4] B. Martin, Nuclear and particle physics: An introduction, John Wiley & Sons, 2006.

[5] V. Koch, Introduction to chiral symmetry (nucl-th/9512029. LBL-38000) (1995) 52 p.

[6] N. Hosihizaki, Formalism of nucleon-nucleon scattering, Suppl. of the Progress of Theoretical Physics No. 43.

[7] Momentum space calculations and helicity formalism in nuclear physics, Nuclear Physics A 176 (2) (1971) 413 – 432.

[8] Nuclear saturation and the smoothness of nucleon-nucleon potentials, Nuclear Physics A 158 (1) (1970) 1 – 42.

[9] R. Machleidt, High-precision, charge-dependent bonn nucleon-nucleon potential, Phys. Rev. C 63 (2001) 024001.

[10] S. S. M. Wong, Introductory Nuclear Physics, 2nd Edition, John Wiley & Sons, Inc., New York, 1998.

[11] C. Nordling, J. Österman, Physics Handbook: For Science and Engineering, Studentlitteratur, 2006.

[12] G. B. Folland, Fourier Analysis and Its Applications, American Mathematical Society, Providence, 2009.

[13] J. Bystricky, F. Lehar, P. Winternitz, Formalism of Nucleon-Nucleon Elastic Scattering Experiments, J.Phys.(France) 39 (1978) 1.

[14] T. Munson, J. Sarich, S. Wild, S. Benson, L. C. McInnes, Tao 2.1 users manual.
URL `www.mcs.anl.gov/tao/docs/tao_manual.pdf`

[15] J. Dobaczewski, W. Nazarewicz, P.-G. Reinhard, Error estimates of theoretical models: a guide, arXiv preprint arXiv:1402.4657.

[16] J. R. Donaldson, R. B. Schnabel, Computational experience with confidence regions and confidence intervals for nonlinear least squares, Technometrics 29 (1) (1987) 67–82.

[17] L. Rall, Perspectives on automatic differentiation: Past, present, and future?, in: M. Bücker, G. Corliss, U. Naumann, P. Hovland, B. Norris (Eds.), Automatic Differentiation: Applications, Theory, and Implementations, Vol. 50 of Lecture Notes in Computational Science and Engineering, Springer Berlin Heidelberg, 2006, pp. 1–14.

[18] A. Griewank, A. Walther, Evaluating derivatives: principles and techniques of algorithmic differentiation, Siam, 2008.

[19] A. Griewank, J. Utke, A. Walther, Evaluating higher derivative tensors by forward propagation of univariate taylor series, Math. Comput. 69 (231) (2000) 1117–1130.

[20] I. Charpentier, J. Utke, Rapsodia: User manual, Tech. rep., Argonne National Laboratory, available online at `http://www.mcs.anl.gov/Rapsodia/userManual.pdf` (2014).

[21] H. Gavin, The levenberg-marquardt method for nonlinear least squares curve-fitting problems, Department of Civil and Environmental Engineering, Duke University.

Appendix A

Chiral Symmetry

In this appendix a more formal definition of chiral symmetry is given in terms of transformations that act on a field. We also explain why massive particles break the chiral symmetry.

A.1 Useful matrices

To understand and define chiral symmetry a set of matrices is needed, these are presented below and will be used later on.

A.1.1 Gamma matrices

The gamma matrices is a set of four 4×4 matrices with special commutation and anti-commutation relations. We will first define these matrices and then give a brief introduction to the anti-commutation relations to the extent that is needed for this thesis. The contravariant gamma matrices are labelled $\gamma^0, ..., \gamma^3$ and are defined as,

$$\gamma^0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

$$\gamma^1 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}$$

$$\gamma^2 = \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}$$

$$\gamma^3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

These matrices satisfy certain anti-commutation relations that follows,

$$\begin{cases} \{\gamma^i, \gamma^j\} = 0 \, i \neq j \\ \{\gamma^i, \gamma^i\} = 1 \, i = 0 \\ \{\gamma^i, \gamma^i\} = -1 \, i = 1,2,3. \end{cases}$$

At last we define the product of the four gamma matrices to be γ^5 which, despite its name, is not a gamma matrix.

$$\gamma^5 = i\gamma^0\gamma^1\gamma^2\gamma^3 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

As can be checked by the reader γ^5 anticommutes with all the four gamma matrices.

A.2 Quantum field theory

To describe what chiral symmetry implies, an object that is invariant under the corresponding symmetry operation is needed. In this case we will use the Lagrangian for massless fermions, given by [5].

$$\mathcal{L} = i\bar{\psi}\partial_\mu\gamma^\mu\psi. \tag{A.1}$$

Chiral symmetry implies that the field is left invariant under the two specific operations of vector and axial transformation, referred to as Λ_V and Λ_A respectively, and defined as,

$$\begin{cases} \Lambda_V : \psi \longrightarrow e^{-i\frac{\vec{\tau}}{2}\vec{\theta}}\psi \approx \left(\mathbf{1} - i\frac{\vec{\tau}}{2}\vec{\theta}\right)\psi \\ \Lambda_A : \psi \longrightarrow e^{-i\gamma_5\frac{\vec{\tau}}{2}\vec{\theta}}\psi \approx \left(\mathbf{1} - i\gamma_5\frac{\vec{\tau}}{2}\vec{\theta}\right)\psi. \end{cases}$$

In the above equations, $\vec{\tau}$ is a isospin vector consisting of the gamma matrices [5]. The complex conjugate of the fields transform like,

$$\begin{cases} \Lambda_V : \bar{\psi} \longrightarrow e^{i\frac{\vec{\tau}}{2}\vec{\theta}}\bar{\psi} \approx \left(\mathbf{1} + i\frac{\vec{\tau}}{2}\vec{\theta}\right)\bar{\psi} \\ \Lambda_A : \bar{\psi} \longrightarrow e^{-i\gamma_5\frac{\vec{\tau}}{2}\vec{\theta}}\bar{\psi} \approx \left(\mathbf{1} - i\gamma_5\frac{\vec{\tau}}{2}\vec{\theta}\right)\bar{\psi}. \end{cases}$$

For the axial transform the minus sign in the exponent remains because of the anti-commutation relations between the gamma matrices,

$$\overline{-i\gamma_5\frac{\vec{\tau}}{2}\vec{\theta}} = i\frac{\vec{\tau}}{2}\vec{\theta}\gamma_5 = -i\gamma_5\frac{\vec{\tau}}{2}\vec{\theta}.$$

With these definitions we can check that the Lagrangian given in Eq. (A.1) really is invariant under Λ_V and Λ_A:

$$\Lambda_V : i\bar{\psi}\partial_\mu\gamma^\mu\psi \longrightarrow i\bar{\psi}\partial_\mu\gamma^\mu\psi - i\vec{\theta}\left(\bar{\psi}i\partial_\mu\gamma^\mu\frac{\vec{\tau}}{2}\psi - \bar{\psi}\frac{\vec{\tau}}{2}i\partial_\mu\gamma^\mu\psi\right) = i\bar{\psi}\partial_\mu\gamma^\mu\psi$$

$$\Lambda_A : i\bar{\psi}\partial_\mu\gamma^\mu\psi \longrightarrow i\bar{\psi}\partial_\mu\gamma^\mu\psi - i\vec{\theta}\left(\bar{\psi}i\partial_\mu\gamma^\mu\gamma_5\frac{\vec{\tau}}{2}\psi + \bar{\psi}\gamma_5\frac{\vec{\tau}}{2}i\partial_\mu\gamma^\mu\psi\right) = i\bar{\psi}\partial_\mu\gamma^\mu\psi.$$

For the axial transform the terms in the parenthesis cancel each other out since γ^μ and γ_5 anti commutes. Since the Lagrangian is preserved for both operators it is said to have chiral symmetry.

A.3 Explicit breaking of the symmetry

To understand why the chiral symmetry breaks for massive particles we will introduce a mass term into the Lagrangian and check if it still is invariant under the symmetry operations. The mass term is introduced in the Lagrangian as [5],

$$\mathcal{L} = i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi.$$

The symmetry operation Λ_V acts on it in the following way,

$$\Lambda_V : i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi \longrightarrow i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi - m\vec{\theta}\left(\bar{\psi}i\frac{\vec{\tau}}{2}\psi - \bar{\psi}i\frac{\vec{\tau}}{2}\psi\right) = i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi.$$

As seen the Lagrangian is still invariant under the vector symmetry. Applying the axial symmetry operation on the Lagrangian results in,

$$\Lambda_A : i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi \longrightarrow i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi$$
$$-m\vec{\theta}\left(\bar{\psi}\gamma_5 i\frac{\vec{\tau}}{2}\psi + \bar{\psi}\gamma_5 i\frac{\vec{\tau}}{2}\psi\right) = i\bar{\psi}\partial_\mu\gamma^\mu\psi - m\bar{\psi}\psi - 2m\vec{\theta}\left(\bar{\psi}i\frac{\vec{\tau}}{2}\gamma_5\psi\right),$$

showing that the Lagrangian no longer is invariant under the axial transformation, meaning that the chiral symmetry breaks down. The extent of the symmetry breaking is seen to be proportional to the mass of the particle described by the Lagrangian. If the mass of the particle is small compared to the working mass scale, the Lagrangian is almost invariant and chiral symmetry can be said to apply approximately. The fact that the symmetry is almost satisfied may allow for the use of perturbation theory, which can be a very effective approach to describe the nuclear force. When done in a systematic way, it is known as chiral EFT.

The explicit breaking of the symmetry described above should not be confused with the spontaneously breaking of the chiral symmetry. The spontaneously breaking means that the Lagrangian is not symmetric in its ground state. For more information on this subject consider reference [5].

Appendix B

Scattering Theory

This Appendix contains more complete expressions for some of the presented concepts in Chapter 3.

B.1 Spin-scattering matrix solutions

The singlet-triplet spin-space matrix is given by

$$M = \begin{bmatrix} M_{++} & M_{+0} & M_{+-} & 0 \\ M_{0+} & M_{00} & M_{0-} & 0 \\ M_{-+} & M_{-0} & M_{--} & 0 \\ 0 & 0 & 0 & M_{ss} \end{bmatrix}$$

Without any detailed deduction equation (3.7) can be solved and gives the following result for neutron-proton scattering.

$$\begin{aligned}
M_{ss} &= \sum_L (2L+1) f_L P_L(\cos\theta) \\
M_{++} &= \sum_L \Bigg[\frac{L+2}{2} f_{L,L} + \frac{L-1}{2} f_{L,L-1} - \frac{1}{2}\sqrt{(L+1)(L+2)} f^{L+1} \\
&\quad - \frac{1}{2}\sqrt{(L-1)L} f^{L-1} \Bigg] P_L(\cos\theta) \\
M_{00} &= \sum_L \Big[(L+1) f_{L,L+1} \\
&\quad L f_{L,L-1} \sqrt{(L+1)(L+2)} f^{L+1} \sqrt{(L-1)L} f^{L-1} \Big] P_L(\cos\theta) \\
M_{0+} &= \sum_L \Bigg[-\frac{L+2}{\sqrt{2}(L+1)} f_{L,L+1} + \frac{2L+1}{\sqrt{2}L(L+1)} f_{LL} + \frac{L-1}{\sqrt{2}L} \\
&\quad f_{L,L-1} \sqrt{\frac{L+2}{2(L+1)}} f^{L+1} - \sqrt{\frac{L-1}{2}} f^{L-1} \Bigg] P_L^1(\cos\theta) \\
M_{+0} &= \sum_L \left[\frac{1}{\sqrt{2}} f_{L,L+1} - \frac{1}{\sqrt{2}} f_{L,L+1} \sqrt{\frac{L+2}{2(L+1)}} f^{L+1} - \sqrt{\frac{L-1}{2}} f^{L-1} \right] \\
M_{+-} &= \sum_L \left[\frac{1}{2(L+1)} f_{L,L+1} - \frac{2L+1}{2L(L+1)} f_{LL} + \frac{1}{2L} f_{L,L-1} \right.
\end{aligned}$$

$$-\frac{1}{2\sqrt{(L+1)(L+2)}}f^{L+1}-\frac{1}{2\sqrt{(L-1)L}}f^{L-1}\Bigg]P_L^2(\cos\theta)$$

with the partial wave amplitudes given as

$$\begin{aligned}
f_L =&\frac{1}{2ik_0}\left(e^{2i\delta_L-1}\right)\\
f_{LJ} =&\frac{1}{2ik_0}\left(e^{2i\delta_{LJ}}-1\right)\\
f_{J\pm1,J} =&\frac{1}{2ik_0}\left(\cos(2\epsilon_J)e^{2i\delta_{J\pm1,J}}-1\right)\\
f^J =&\frac{1}{2k}\sin(2\epsilon_J)e^{i(\delta_{J-1,J}+\delta_{J+1,J})}
\end{aligned}$$

With these expressions it is possible to calculate observables for different types of polarised scattering experiments.

B.2 Interaction formulas

The partial wave decomposition of the interaction in momentum space is given by $\langle LSJ\,|\,V(p,p')\,|\,LS'J\rangle$. The partial wave projections discussed in Section 3.4 is given in six channels by the following expressions.

$$\begin{aligned}
W_1 =& \mathcal{A}_C^{J,0}-3\mathcal{A}_S^{J,0}+p^2p'^2(\mathcal{A}_{\sigma L}^{J,2}-\mathcal{A}_{\sigma L}^{J,0})-(p^2+p'^2)\mathcal{A}_T^{J,0}+2p'p\mathcal{A}_T^{J,1}\\
W_2 =& \mathcal{A}_C^{J,0}+\mathcal{A}_S^{J,0}+\frac{pp'}{2J+1}\left(\mathcal{A}_{LS}^{J+1,0}-\mathcal{A}_{LS}^{J-1,0}\right)+\\
&p^2p'^2\left(-\mathcal{A}_{\sigma L}^{J,0}+\frac{J-1}{2J+1}\mathcal{A}_{\sigma L}^{J+1,1}+\frac{J+2}{2J+1}\mathcal{A}_{\sigma L}^{J-1,1}\right)+\\
&\left((p'^2+p^2)\mathcal{A}_T^{J,0}-\frac{2pp'}{2J+1}\left(J\mathcal{A}_T^{J+1,0}+(J+1)\mathcal{A}_T^{J-1,0}\right)\right)\\
W_3 =& \mathcal{A}_C^{J+1,0}+\mathcal{A}_S^{J+1,0}+pp'\frac{J+2}{2J+3}\left(\mathcal{A}_{LS}^{J+2,0}-\mathcal{A}_{LS}^{J,0}\right)+\\
&p^2p'^2\left(\frac{2J+3}{2J+1}\mathcal{A}_{\sigma L}^{J+1,0}-\frac{2}{2J+1}\mathcal{A}_{\sigma L}^{J,1}-\mathcal{A}_{\sigma L}^{J+1,2}\right)+\\
&\frac{1}{2J+1}\left(2pp'\mathcal{A}_T^{J,0}-(p^2+p'^2)\mathcal{A}_T^{J+1,0}\right)\\
W_4 =& \mathcal{A}_C^{J-1,0}+\mathcal{A}_S^{J-1,0}+pp'\frac{J-1}{2J-1}\left(\mathcal{A}_{LS}^{J-2,0}-\mathcal{A}_{LS}^{J,0}\right)\\
&p^2p'^2\left(\frac{2J-1}{2J+1}\mathcal{A}_{\sigma L}^{J-1,0}+\frac{2}{2J+1}\mathcal{A}_{\sigma L}^{J,1}-\mathcal{A}_{\sigma L}^{J-1,2}\right)\\
&\frac{1}{2J+1}\left(-2pp'\mathcal{A}_T^{J,0}+(p^2+p'^2)\mathcal{A}_T^{J-1,0}\right)\\
W_5 =& -2p^2p'^2\frac{\sqrt{J(J+1)}}{(2J+1)^2}\left(\mathcal{A}_{\sigma L}^{J+1,0}-\mathcal{A}_{\sigma L}^{J-1,0}\right)\\
&-2\frac{\sqrt{J(J+1)}}{2J+1}\left(p^2\mathcal{A}_T^{J+1,0}+p'^2\mathcal{A}_T^{J-1,0}-2pp'\mathcal{A}_T^{J,0}\right)\\
W_6 =& -2p^2p'^2\frac{\sqrt{J(J+1)}}{(2J+1)^2}\left(\mathcal{A}_{\sigma L}^{J+1,0}-\mathcal{A}_{\sigma L}^{J-1,0}\right)\\
&-2\frac{\sqrt{J(J+1)}}{2J+1}\left(p^2\mathcal{A}_T^{J-1,0}+p'^2\mathcal{A}_T^{J+1,0}-2pp'\mathcal{A}_T^{J,0}\right)
\end{aligned}\tag{B.1}$$

Appendix C

Additional Rapsodia Examples

C.1 The HigherOrderTensor tool and mixed derivatives

The following code is a modified version of the `CALCULATE` program of 5.4, using the `HigherOrderTensor` tool to calculate mixed second order derivatives.

```
SUBROUTINE SUMS(x, y, squaresum, cubesum)
  INCLUDE 'RAinclude.i90'
  IMPLICIT NONE
  TYPE(RARealD), INTENT(IN) :: x,y
  TYPE(RARealD), INTENT(INOUT) :: squaresum, cubesum
  squaresum = x**2+y**2
  cubesum = x**3+y**3
END SUBROUTINE

PROGRAM CALCULATE
  INCLUDE 'RAinclude.i90'
  USE higherOrderTensorUtil
  IMPLICIT NONE

  TYPE(higherOrderTensor) :: T
  INTEGER, DIMENSION(:,:), ALLOCATABLE :: SeedMatrix
  REAL*8, DIMENSION(:,:), ALLOCATABLE :: TaylorCoefficients
  REAL*8, DIMENSION(:,:), ALLOCATABLE :: CompressedTensor
  INTEGER :: i, k, N, O, DIRS

  TYPE(RARealD) :: x, y, z, squaresum, cubesum
  REAL*8 z_x,z_y,z_xx,z_yy

  N=2 ! Two variables
  O=2 ! We want derivatives up to the second order

  CALL setNumberOfIndependents(T,N)
  CALL setHighestDerivativeDegree(T,O)
  DIRS = getDirectionCount(T) ! DIRS will be set to 3

  ALLOCATE(SeedMatrix(N, DIRS))
  CALL getSeedMatrix(T, SeedMatrix)
```

```
! SeedMatrix will have the following content:
!          x'     y'
! DIR 1:   0      2
! DIR 2:   1      1
! DIR 3:   2      0

x=-1.2
y=3.2

! Seed
DO i= 1,DIRS
  CALL RAset(x, i, 1, REAL(SeedMatrix(1,i), KIND=RAdKind))
  CALL RAset(y, i, 1, REAL(SeedMatrix(2,i), KIND=RAdKind))
ENDDO

! Calculate z
CALL SUMS(x, y, squaresum, cubesum)
z=x*y**2+sin(2*x)-squaresum+cubesum

! Save Taylor coefficient of z in the various directions
ALLOCATE(TaylorCoefficients(0, DIRS))
DO k=1,0
  DO i= 1,DIRS
    CALL RAget(z, i, k, TaylorCoefficients(k, i))
  ENDDO
ENDDO

! Compute derivatives via interpolation of Taylor coefficients
ALLOCATE(CompressedTensor(0, DIRS))
CALL setTaylorCoefficients(T, TaylorCoefficients)
! First order
CALL getCompressedTensor(T, 1, CompressedTensor(1,:))

! Second order
CALL getCompressedTensor(T, 2, CompressedTensor(2,:))

WRITE(*,*) 'z=', z%v

WRITE(*,*) 'First␣order␣derivatives:'
DO i= 1,DIRS
  WRITE (*,'(A,I1,A,I1,A)',ADVANCE='NO') '[', SeedMatrix(1, i), &
          '][', SeedMatrix(2, i), ']:'
  WRITE (*,*) CompressedTensor(1,i)
ENDDO

WRITE(*,*) 'Second␣order␣derivatives:'
DO i= 1,DIRS
  WRITE (*,'(A,I1,A,I1,A)',ADVANCE='NO') '[', SeedMatrix(1, i), &
          '][', SeedMatrix(2, i), ']:'
  WRITE (*,*) CompressedTensor(2,i)
ENDDO

DEALLOCATE(SeedMatrix)
```

```
  DEALLOCATE(TaylorCoefficients)
  DEALLOCATE(CompressedTensor)

END PROGRAM
```

As described in Section 5.4.1, the `HigherOrderTensor` tool will compute the number of needed directions as function of the number of independent variables `N` and the maximum derivative order `O`. To compute second order derivatives with respect to two variables, 3 directions are needed, corresponding to the number of unique elements in a 2x2 Hessian. Based on `N` and `O`, the tool will then create a `SeedMatrix` with seeding values for each direction. When the computation has finished, the Taylor coefficients are extracted with `RAget` and put into the 2D array `TaylorCoefficients`. Finally, the Taylor coefficients are interpolated with `getCompressedTensor`, which then associates each direction with a computed first and second order derivative.

The output of the modified program is

```
 z=   6.3965368745134015
 First order derivatives:
[0][2]:   16.640000400543215
[1][1]:   15.485213183949114
[2][0]:   0.0000000000000000
 Second order derivatives:
[0][2]:   14.800000190734863
[1][1]:   6.4000000953674308
[2][0]:  -6.4981478451910828
```

where the numbers in the square brackets refer to seeding values of `x` and `y`, respectively, in the direction at hand. When compared with the output of the original program

```
 z=   6.3965368745134015
 z_x=   15.485213183949114
 z_y=   16.640000400543215
 z_xx=  -6.4981478451910828
 z_yy=   14.800000190734863
```

we clearly see that all previously computed values are reproduced, and that the second order derivatives follow interesting pattern: $z_{yy} = \frac{\partial^2 z}{\partial y^2}$ is printed in the direction seeded with `[0][2]`, $z_{xx} = \frac{\partial^2 z}{\partial x^2}$ is printed in the `[2,0]` direction and the last value, which must be $z_{xy} = \frac{\partial^2 z}{\partial x \partial y}$, is printed in the `[1][1]` direction. This means that the seeding values of `x` and `y` in a certain direction corresponds to the number of times the variables appear in the denominator of the second order derivative associated with that direction. The order of the first order derivatives are less obvious however, and some manual checks might be needed to determine which derivative appears where.

As previously mentioned, further information on the `HigherOrderTensor` tool can be found in the Rapsodia user manual. [20]

C.2 AD enabled matrix inversion using external library routines

In the code listed below, a `matrix_inversion_RA` subroutine is defined which inverts a `RARealD` matrix and calculates the appropriate derivatives in an AD manner. It is a slightly simplified version of an actual routine implemented in the `nsopt` program. `matrix_inversion_RA` starts the computation by inverting the non-derivative part of the matrix using `matrix_inversion`, which uses `LAPACK` routines to perform ordinary matrix inversion. The derivatives of the matrix inverse are then calculated using the formulas (5.4) and (5.5) described in Section 5.4.2 and attached to the inverted non-derivative matrix.

```
SUBROUTINE matrix_inversion_RA(N, A)
  ! NOTE: THIS SUBROUTINE ONLY SUPPORTS FIRST
  ! AND SECOND ORDER DERIVATIVES
  IMPLICIT NONE
  INTEGER, INTENT(INOUT) :: N
  TYPE(RARealD), INTENT(INOUT) :: A(N,N)
  TYPE(RARealD), DIMENSION(:,:), ALLOCATABLE :: A_backup
  REAL*8, DIMENSION(:,:), ALLOCATABLE :: A_real, A_derv, &
  A_derv2, A_inv_derv, A_inv_derv2
  REAL*8 coeff
  INTEGER :: dir, i, k, num_dir
  num_dir = AD_DIR ! preprocessor variable defined in makefile

  ALLOCATE(A_backup(N,N))
  ALLOCATE(A_real(N,N))
  ALLOCATE(A_derv(N,N))
  ALLOCATE(A_derv2(N,N))
  ALLOCATE(A_inv_derv(N,N))
  ALLOCATE(A_inv_derv2(N,N))

  ! Save the non-derivative values to an ordinary REAL*8 array
  A_real = A%v

  ! Invert the derivative stripped A using
  ! the ordinary library routine
  CALL matrix_inversion(N, A_real)
  A_backup = A
  A = A_real

  ! A now holds the inverted matrix inv(A), but not its derivatives
  ! which we still need to compute for each direction
  DO dir=1,num_dir
    ! Get the first order derivatives of each element in A_backup(N,N)
    DO i=1,N
      DO k=1,N
        CALL RAget(A_backup(i,k),dir,1,coeff)
        A_derv(i,k) = coeff
      ENDDO
    ENDDO

    ! Compute the first order derivative of inv(A) according
    ! to the formula d(inv(A))/dt=-inv(A)*dA/dt*inv(A)
    A_inv_derv = -MATMUL(MATMUL(A_real,A_derv),A_real)

    ! Save these derivatives to A
    DO i=1,N
      DO k=1,N
        CALL RAset(A(i,k),dir,1,A_inv_derv(i,k))
      ENDDO
    ENDDO

    ! Get the second order derivatives
    DO i=1,N
```

```
      DO k=1,N
        CALL RAget(A_backup(i,k),dir,2,coeff)
        ! The second order derivative is twice
        ! the Taylor coefficient f''/2!
        A_derv2(i,k) = 2*coeff
      ENDDO
    ENDDO

    ! Compute the second order derivative of inv(A) according
    ! to the formula d^2(inv(A))/dt=
    ! =2*inv(A)*dA/dt*inv(A)*dA/dt*inv(A)-inv(A)*d^2A/dt^2*inv(A)
    A_inv_derv2 = 2*MATMUL(MATMUL(MATMUL(MATMUL(A_real,A_derv), &
    A_real),A_derv),A_real) - MATMUL(MATMUL(A_real,A_derv2),A_real)

    ! The second order Taylor coefficient is half the derivative
    A_inv_derv2 = A_inv_derv2/2

    ! Add the second order derivatives to A
    DO i=1,N
      DO k=1,N
        CALL RAset(A(i,k),dir,2,A_inv_derv2(i,k))
      ENDDO
    ENDDO
  ENDDO

  DEALLOCATE(A_backup)
  DEALLOCATE(A_real)
  DEALLOCATE(A_derv)
  DEALLOCATE(A_derv2)
  DEALLOCATE(A_inv_derv)
  DEALLOCATE(A_inv_derv2)

END SUBROUTINE matrix_inversion_RA
```