The two-nucleon and three-nucleon systems in three dimensions

Kacper Topolnicki

PHD thesis written under the supervision of prof. Jacek Golak
at the Jagiellonian University, Faculty of Physics, Astronomy
and Applied Computer Science, Kraków,
Friday 14th March, 2014

Contents

Abstract

In this thesis we explore various aspects of a three dimensional treatment of the two- and three-nucleon systems. Starting from a very elementary description of few-nucleon degrees of freedom we work our way up to more complicated calculations. The deuteron bound state and nucleon-nucleon transition operator are treated using a general form of the two-nucleon force. Calculations involving nuclear current operators employ a formalism that allows an extension of these calculations to describe electro-weak processes. The three-nucleon bound state calculations that are discussed in the final chapters utilize, in addition to a two-nucleon force, an operator form of a three-nucleon potential. The presented results have been verified and published [15, 20, 21, 35]. A lot of attention is paid to the practical numerical realization of our calculations. For this purpose, attached to the thesis, are a number of *Mathematica*® [38] notebooks and packages containing tools useful in building a FORTRAN implementation. Additionally, the notation used in the text, especially when defining large linear operators, is specifically chosen to make the translation to codes straightforward.

Chapter 1

Preface

Writing this thesis was a chance to gather in one place the descriptions of the various tools that were used in our three dimensional treatment of two- and three-nucleon systems. The mathematical foundations that govern our treatment of few-nucleon systems are not very complicated. We use non-relativistic quantum mechanics and the isospin formalism to describe the proton and the neutron. However, in order to perform practical computations, complicated analytical expressions resulting (in only a couple steps) from the fundamental equations (Schrödinger, Lippmann-Schwinger, Faddeev) had to be treated in a consistent way. This is where the *Mathematica*® packages and notebooks that were developed in our group and are supplied with this thesis play an important role. We hope that, together with our *Mathematica*® tools, this text will be a practical guide to our calculations that will allow other researchers not only to reproduce our results but also to apply our tools to other problems.

We work in the momentum space, without resorting to partial wave decomposition (PWD). Instead we use three-dimensional momentum eigenstates directly. A very good overview of this three-dimensional (3D) treatment can be found in our paper [1]. We decided to repeat the most important points from this paper below as a part of the introduction.

...

Nucleon-nucleon scattering was treated without PWD already more than twenty years ago. In [2, 3] the time-dependent Schrödinger equation was solved eventually for a one-boson exchange potential. It is worth mentioning that in the latter paper the general form of the potential between two spin-1/2 particles was used to simplify the calculations.

Later in [4] quasielastic electron scattering was investigated and the final-state interaction was taken into account by evaluating the two-body t-matrix directly in 3D for the Malfliet-Tjon (MT III) local spin independent force [5]. More systematically the angular and momentum dependence of the t-matrix was studied in the same 3D approach on as well as off the energy shell in [6], both for positive and negative two-nucleon (2N) energies. In this very informative paper the behaviour of the t-matrix in the vicinity of bound-state pole and resonance poles in the second energy sheet were also investigated for different Malfliet-Tjon-type potentials.

Another alternative to the usual PWD technique was outlined in [7]. There the two-body Lippmann-Schwinger equation was written in a numerically solvable form using helicity theory and taking advantage of the symmetries of the NN interaction. The numerical examples were based on the Bonn OBEPR potential [8]. The helicity formalism was also used in [9] (with slightly modified final equations) for two quite different NN potentials, the Bonn B [10] and the Argonne V18 [36]. The same helicity approach was subsequently used by S. Bayegan *et al.* [11] in 3D calculations of NN bound and scattering states with a chiral N3LO potential [37]. In all these works an excellent agreement with the results based on standard PWD was reported.

Inclusion of the Coulomb interaction on top of a local spin-dependent short-range interaction in two-body scattering was carried out in [12]. The calculations are not performed for the NN system but their implications are important for all results, where the screening and renormalization approach is used to treat the Coulomb interaction.

Parallel to the above mentioned nonrelativistic studies, 3D formulations of the scattering equations were studied also for the relativistic equations. In [13] this was outlined in the case of pion-nucleon and NN scattering treated via the Bethe-Salpeter equation. In [14] a numerical method, based on the Padé summation, was introduced to solve the covariant spectator equation without partial wave decomposition, and applied to the NN system.

Last but not least we would like to mention calculations of the NN t-matrix, which employ directly momentum vectors and use spin-momentum operators multiplied by scalar functions of the momentum vectors. This approach stems from the fact that a general NN force being invariant under time-reversal, parity, and Galileo transformations can depend only on six linearly independent spin-momentum operators. The representation of the NN potential using spin-momentum operators leads to a system of six coupled equations of scalar functions (depending on momentum vectors) for the NN t-matrix, once the spin-momentum operators are analytically calculated by performing suitable trace operations. This treatment, formulated in [15], can be considered as a natural extension for two spin-1/2 particles of the calculations described in [6], In [15] numerical examples for the Bonn B [10] and chiral N2LO potentials [37, 16, 17] were presented. Later in [15] the same approach (with a modified choice of the basis spin-momentum operators) was applied to the Argonne V18 potential [36]. Further variations of this method and inclusion of the Coulomb force can be found in [19, 20]. Finally, the application of this operator based approach to the deuteron electro-disintegration process was discussed in [21].

Next we give an overwiev of calculations related to the bound state of the two- and three-nucleon systems. We start with the deuteron representations formulated without any resort to PWD. In [22] the helicity representation developed previously for NN scattering [23] was applied to the 2N bound state and the deuteron eigenvalue equation in the helicity basis was solved with the Bonn B potential [10]. In the same paper the deuteron wave function in the so-called (momentum space) operator form was also derived. In this representation the whole information about the deuteron is given by two scalar functions, $\phi_1(p)$ and $\phi_2(p)$ (p is the magnitude of the relative momentum between the two nucleons) , which are closely related to the standard S and D components of the deuteron. The direct set of two coupled equations for $\phi_1(p)$ and $\phi_2(p)$ was derived only later in [23]. That derivation included simple trace operations, which helped eliminate spin degrees of freedom and led to analytically given sets of scalar functions depending on momentum vectors only. Numerical examples for the Bonn B [10] and chiral N2LO potentials [37, 16, 17] were published in [15]. Corresponding three dimensional calculations of 2N binding energies with chiral N3LO potentials [37] performed in the helicity formalism were reported in [11].

As already mentioned, the work on NN scattering has very often a preparatory character and further application of the t-matrices are usually planned. This is true also in the case of the 3D calculations. Results of [6] were later used in [24] three-body bound-state calculations without PWD with Malfliet-Tjon-type NN potentials, neglecting spin and isospin degrees of freedom. In the subsequent paper [25] the scheme from [24] was extended to include scalar two-meson exchange three-body forces.

The Teheran group published several papers dealing with 3D solutions of the three-nucleon (3N) and even four-nucleon (4N) bound states [26, 27, 28, 29, 30, 31, 32, 33]. They started with a formulation, which neglected the spin-isospin degrees of freedom [26] and introduced step by step improved dynamical ingredients to their framework, performing calculations with more realistic NN potentials (like the Bonn B one in [27, 28]) and including additionally a 3N force (for example the Tucson-Melbourne 3N potential in [30]). The 3D t-matrices - an input to the systems of coupled equations - were obtained with the helicity representation of [9].

Finally, we list publications dealing with the 3D treatment of the 3N bound state which relies on the general form of the 2N t-matrix and the operator form of the 3N bound state introduced in [34]. The latter consists of eight operators built from scalar products of relative momentum and spin vectors, which are applied to a pure 3N spin 1/2 state. Each of the operators is multiplied by a scalar function of the relative momentum vectors. In [23] one Faddeev equation for identical bosons was replaced by a finite set of coupled equations for scalar functions which depend only on three variables. The inclusion of a 3N force into this 3D Faddeev framework was also discussed. Further elements of this formalism, for example the construction of the full wave function from the Faddeev amplitude, and first numerical results for chiral 2N and 3N N^2LO nuclear forces were provided in [35].

...

A big advantage of our three-dimensional approach is a very explicit way of performing computations. The computing time that is used in PWD calculations is, with the current state of technology, not typically a limiting factor. The cost of human labor that is needed to prepare a numerical realization of PWD calculations is however very significant. The three dimensional approach that is described in the following text requires a larger amount of computing resources but, due to being very direct (and thanks to our *Mathematica*® [38] toolkit), greatly reduces the human workload. Using three dimensional states can make economic sense especially with the exponential growth of available computing power. The cost of erecting and running a new supercomputer is becoming a rival to the cost of raising, training and maintaining a team of physicists. This can be observed when considering that the latter requires decades of heavy investments from the parents and society. We expect that the overall cost of this process will remain constant and the cost of computing time will tend to decrease, at least into the near future. If we are correct, our three dimensional approach will become more and more economically feasible and might even come to replace classical PW calculations.

Preparing the code for three dimensional calculations with the tools that are described in this thesis is very straightforward. Our methods allow the programmer to almost directly implement the algebraic expressions that follow directly from the Schrödinger, Faddeev or Lippmann-Schwinger equations and conceal the complicated (but not very interesting) details of the calculation in automatically generated FORTRAN codes. The linear operators constructed from these automatically generated expressions can be used to calculate the two-nucleon transition operator and bound state with a very general form of the two-nucleon force and the three-nucleon bound state with a very general operator form of the two- and three-nucleon potentials. As was mentioned, the computing resources necessary to perform the three dimensional calculations are large. This is a consequence of the size of linear operators involved. The final parts of this thesis describe methods that can be used in the numerical treatment of large linear operators by reducing the size of the problems.

In the literature one can find a great number of publications that introduce effective two- and three-nucleon potentials. The older ones, for example the two nucleon Bonn B potential - [10] - and the three-nucleon Tucson-Melbourne force - [39, 40, 41, 42] - were based on the one-boson exchange picture. Recently two-nucleon and three-nucleon forces are derived by various groups within the chiral effective field theory approach. For example, the chiral 2N and 3N potentials [17], [43] are derived at different orders of the chiral expansion. The growing number of models is another important motivator for the development of our tools - our methods can be quickly applied to new potentials and provide predictions based on these forces. The growing accuracy of these potentials makes them a great tool in trying to understand the physics of few-nucleon systems. Using classical non-relativistic quantum mechanics also gives a rare opportunity to gain an intuitive understanding of underlying phenomena - this is possible because all calculations can, in only a few steps, be linked to the fundamental equations. This is often difficult when dealing with the complexity of quantum field theory. The flexibility of our calculations and the possibility to extend them to describe new phenomena was a great additional reward in itself.

Chapter 2

Text organization - please read this

Here we will give a brief summary of the contents of the thesis. This chapter can serve as a reading guide since some chapters can be approached independently. We strongly recommend going through the few following paragraphs.

Chapter 4 introduces the physical meaning of equations that will be the subject of our computations. The treatment of those equations will be discussed in more detail in further parts of the text. First we give a short description of the degrees of freedom of the two-nucleon (2N) and three-nucleon (3N) systems in section 4.1 and introduce the set of 2N and 3N states which will be fundamental to our calculations. In section 4.2 we give a very elementary introduction to the calculations involving the transition operator followed by an introduction to the 2N and 3N bound state equations in section 4.3. These two parts give only a very basic outline of the problems extracted from the references given in the text. Section 4.4 contains a more detailed description of the numerical realization of 2N and 3N degrees of freedom and basic operators. The remaining parts of this Chapter contain additional information that might come in handy when working with our formalism.

Chapters 6, 7, 10 concentrate on discussing each of the main equations introduced in Chapter 4 separately with an emphasis on the numerical realization of the calculations. All these chapters are fairly self-contained and can be read independently after going through Chapter 5, where we discuss the general form of the 2N potential operator in the momentum space. After some mathematical manipulations and the incorporation of additional constraints on the operator form of the transition operator as well as on the 2N and 3N bound states, it will turn out that all calculations boil down to large linear problems. In these linear equations operators will act on scalar functions that describe the transition operator, 2N and 3N bound states when the latter are written in their respective operator forms. In Chapter 3 we establish a notation that is meant to make the creation of the numerical implementations of these operators straightforward. This notation is natural when working with a practical implementation of scalar functions (where they will typically be represented by multidimensional arrays inside a computer) and allows for an almost literal translation of the operator definitions to codes. We strongly recommend reading Chapter 3 first, before continuing to the remaining text. Our strategy is to provide expressions for the operators in a way that is convenient for the programmer and to provide tools to create the most complicated parts of the implementation separately. This is taken care of by our *Mathematica*® tools that create FORTRAN codes automatically.

In Chapter 8 we apply the results of the previous sections (deuteron and transition operator calculations) to a description of the $e +^2\mathrm{H} \rightarrow e + p + n$ reaction. A detailed description of the treatment of electromagnetic currents within our framework is presented. The muon induced deuteron disintegration process is treated in a very similar way in Chapter 9.

In the numerical realization of the calculations the scalar functions have to be discretized over a lattice which effectively turns them into finite dimensional vectors. Ideally we would like to be able to create a direct matrix representation of the operators in each case. This is not a straightforward task and is available only for the transition operator and the deuteron bound state. Another problem is the dimension of vectors and operators involved in the calculations. For example, in order to describe the 3N bound state with a reasonable accuracy, 1000000 (or more) dimensional vectors are needed and the calculations would have to involve 1000000×1000000 matrices. Appendix A describes our solution to this problem of large linear operators. We can reduce the 1000000×1000000 problem to a simple (say) 40×40 matrix linear equation with the help of Krylov subspace methods. What's more, when using these methods, we do not need to know the direct matrix representation of the relevant operators. Our procedures require only the calculation of the action of the linear operator on a vector and the computation of a scalar product.

Appendix B contains a description of the *util1N2N3N.m Mathematica*® package that was created to be used with our calculations and is supplied with this thesis. It contains a set of definitions that together with the *FunctionArray.m* package can be used to calculate all building blocks of our calculations. These building blocks are created automatically and often contain very complicated (but otherwise not very interesting) analytical expressions that result from the fundamental equations, thus freeing the physicist to focus on more interesting aspects of the calculations. The *FunctionArray.m* package is described in detail in Appendix C. We suggest reading Appendixes B and C before going on to read Chapters 5 and higher. In this way the reader will be able to use the attached notebooks parallel to reading the text.

Finally, in Apendix D we give a list of *Mathematica*® notebooks that are provided with this thesis and create FORTRAN implementations of all necessary building blocks. The resulting code is documented and can be used to construct a working numerical realization. We do not include the full code only the building blocks because they can be documented in more detail. Additionally, the building blocks can be used more universally with different programing paradigms (on single and parallel machines, using different styles of programming).

Chapter 3

Units and Notation

Calculations presented in this thesis are all based on a set of well defined degrees of freedom of the two and three-nucleon systems (this is described in more detail in section 4.1). It is therefore very convenient to establish a notation that will be used consistently throughout the text.

We adopt the following notation:

1. Operators will be denoted using the " ˇ " symbol, for instance $\check{1}$ is the identity operator.

2. Vectors will be denoted using bold face. For instance $\boldsymbol{p}$, $\check{\boldsymbol{p}}$ will denote the momentum vector and the momentum vector operator respectively.

3. Unit vectors will be denoted using the " ˆ " symbol as in $\hat{\boldsymbol{p}}$.

4. Vector spin operators acting in the space of particle i will be denoted as $\frac{\hbar}{2}\check{\boldsymbol{\sigma}}(i)$ (where $\hbar$ is the reduced Planc constant) and are typically represented using Pauli matrices. $\frac{\hbar}{2}\check{\boldsymbol{\sigma}}(i)_\mu$ will be used to mark the μ component of the spin operator. For cartesian coordinates $\mu = 1, 2, 3$ or $\mu = \hat{\boldsymbol{x}}, \hat{\boldsymbol{y}}, \hat{\boldsymbol{z}}$. For spherical coordinates $\mu = +1, -1, 0$.

5. Vector isospin operators acting in the space of particle i will be denoted using $\frac{1}{2}\check{\boldsymbol{\tau}}(i)$ and are also represented using Pauli matrices. $\frac{1}{2}\check{\boldsymbol{\tau}}(i)_\mu$ will be used to mark the μ component of the isospin operator.

6. Operators and vectors can be placed inside square brackets $[\ldots]$ to mark or remind the reader that their numerical realization is achieved by using a matrix representation. All operators inside $[\ldots]$ can be implemented as matrices using our *util1N2N3N.m Mathematica*® [38] package, described in Appendix B. A FORTRAN implementation of the resulting *Mathematica*® expressions is available through the *FunctionArray.m* package. This package provides implementations compatible both with the free and fixed form of FORTRAN syntax. Both packages are described in the final chapters of this thesis and can be used to quickly piece together a numerical realization of most calculations discussed in the text. In some cases, usually when introducing a new operator, superscripts $[\ldots]^{x\times y}$ or $[\ldots]^{x}$ are used to give additional information on the size of the matrix $(x \times y)$ or vector (x). Optionally $[\ldots]^{\mathrm{1N}}$, $[\ldots]^{\mathrm{2N}}$, $[\ldots]^{\mathrm{3N}}$, $[\ldots]^{\mathrm{1Nspin(isospin)}}$,

$[\ldots]^{\rm 2Nspin(isospin)}$, $[\ldots]^{\rm 3Nspin(isospin)}$ can be used to mark that the matrix representation is in the basis from tables E.1, E.2, E.3, E.4, E.5 or E.6, respectively. The tables are located in Appendix E and details on the matrix representation of states and operators will be given in section 4.4.

7. Symbol $\otimes$ will denote the tensor product. If this symbol is surrounded by expressions inside $[\ldots]$, for example $\left[\breve{A}\right]^{2\times 2} \otimes \left[\breve{B}\right]^{2\times 2}$, then it has an implementation in the *util1N2N3N.m* package. The matrix representation of the tensor product of the two operators $\breve{A}$ and $\breve{B}$ is created using the Kronecker product inside *Mathematica*® . The tensor product of two vectors $[\boldsymbol{a}] \otimes [\boldsymbol{b}]$ also has a natural implementation inside *util1N2N3N.m*. More information on this representation can be found in section 4.4.

8. If not stated otherwise, the capital letter $\boldsymbol{K}$ will be used to denote the total momentum of a system of nucleons. The momenta of individual particles will be referred to using lower-case $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, ... For example $\boldsymbol{K} = \boldsymbol{k}_1 + \boldsymbol{k}_2$ is the total momentum and $\boldsymbol{p} = \frac{1}{2}\left(\boldsymbol{k}_1 - \boldsymbol{k}_2\right)$ is the relative momentum of a 2N system.

9. Functions and operators:

   - Curly brackets as in $t_i^{\{\gamma\}}\left({\scriptstyle \{E\}}, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}, x'\right)$ are used to mark that for any values of the quantities inside ${\scriptstyle \{\}}$ there exists an independent equation, in this case for t. This information can significantly improve the numerical realization of the calculation by reducing the size (and therefore memory requirements) of the (typically large linear) problem.
   - Many parts of the text will introduce linear operators acting on functions of momentum magnitudes, angles etc. We will use notation $\left(\breve{A}t\right)_i^{\{\gamma\}}\left({\scriptstyle \{E\}}, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}, x'\right)$ to mark that operator $\breve{A}$ acts on a scalar function t and we take the resulting function value for a given set of arguments. Additionally, as mentioned above, this notation informs that for each value of $\gamma, E, |\boldsymbol{p}|$ there exists an independent set of linear equations and the dimension of the operator $\breve{A}$ for each subspace is smaller ($\breve{A}$ is constructed using automatically generated FORTRAN implementations with the use of the *util1N2N3N.m* and *FortranFunctionArray.m* packages).

   At first glance this type of notation might seem a little convoluted. We chose this way of writing operators with the programmer in mind. When creating a numerical implementation, functions will typically be replaced by multidimensional arrays and taking the value of a function will be replaced by looking up an index in a multidimensional array. The definitions of the operators will be in a form similar to for example $(\breve{O}f)_i(x, {\scriptstyle \{y\}}) = \ldots \int \ldots \sum_j \ldots A_{ij} f_j(x, {\scriptstyle \{y\}})$, making the conversion to multidimensional arrays natural. Additional information in ${\scriptstyle \{y\}}$ means that f does not need to be discretized over the different values of y.

We chose a unit system in which:

$$\hbar c = 1\,[\hbar\, c] = 197.33\,[\mathrm{MeVfm}]\,,$$

the conversion between units of energy, momentum and mass is done according to:

$$[\text{energy}] = [\text{MeV}] = \frac{1}{197.33}\left[\frac{\hbar\, c}{\text{fm}}\right],$$

$$[\text{momentum}] = \left[\frac{\text{MeV}}{c}\right] = \frac{1}{197.33}\left[\frac{\hbar}{\text{fm}}\right],$$

$$[\text{mass}] = \left[\frac{\text{MeV}}{c^2}\right] = \frac{1}{197.33}\left[\frac{\hbar}{c\,\text{fm}}\right]$$

and typically set $\hbar = c = 1$.

Obtaining the code

We would be happy to share our experience with other researchers. The codes can be obtained on request from

kacper.topolnicki@uj.edu.pl.

We reserve the right to distribute only parts of our software.

The complete package (or parts of the codes) will be compressed in a **zip** file. Please do not redistribute this archive. The current version is available through the e-mail adress given above and contains the latest updates.

The archive contains the master directory

PHD_TOPOLNICKI.

Inside there are two subdirectories: ***TEXT*** contains a hyper-linked PDF with this thesis and ***PROGRAMS*** contains the *Mathematica*® notebooks and codes. Any file paths that might appear in the text are relative to the ***PROGRAMS*** directory. For instance, the full path to

PROGRAMS/util1N2N3N.m

is

<path to PHD_TOPOLNICKI >/PROGRAMS/util1N2N3N.m.

If our code is found to be useful, please give us credit in any resulting publications. This can be done by citing one of the following papers: [21, 20, 15, 35]. For example: "...this paper uses *Mathematica*® and FORTRAN software developed at the Jagiellonian University, the same software was also used in [21]...".

Chapter 4

Introduction

In section 4.1 we discuss the degrees of freedom of the two-nucleon (2N) and three-nucleon (3N) systems. The concept of the nucleon is introduced along with the momentum space representation of the quantum mechanical systems and the form of the free Hamiltonian operators. This chapter is not meant to give a full description, a more complete discussion can be found in section 4.4.

Next section (4.2) deals with the scattering problem and introduces the Lippmann - Schwinger equation for the transition operator. Section 4.3 introduces equations governing two and three-particle bound state calculations. At the end of these two parts, the physical meaning of our numerical calculations from Chapters 6, 7 and 10 should be clear.

The final parts of this chapter give a detailed description of the numerical realization of states and operators discussed earlier. Much attention will be paid to the momentum - isospin - spin representation of the permutation operator which is crucial in the construction of the 3N bound state and 3N scattering calculations.

It should be noted that all calculations in this thesis are done within the realm of classical, non-relativistic quantum mechanics. Because of this, limitations have to be placed on the energies of the 2N and 3N systems considered in our discussions. These energy constraints are not a significant issue when dealing with the bound states, but have to be taken into account when performing scattering calculations.

4.1 Degrees of freedom

The calculations presented in the remaining chapters of this thesis benefit greatly from a well defined set of degrees of freedom (DOF). This well defined set is the subject of this section. Used within the framework of non-relativistic quantum mechanics it allowed us to start all our calculations directly from the most fundamental (Schrödinger, Faddeev and Lippman-Schwinger) equations. Starting from the fundamentals served as a beautiful reminder of how our computations are rooted in reality, this reminder was a great motivator for our work.

Another advantage of using this well defined set of DOF is the possibility of constructing the calculations semi-automatically with the help of symbolic programming. We again remind that Appendixes B and C describe the *util1N2N3N.m* and *FunctionArray.m Mathematica*® [38] packages that can be used to encapsulate the complicated details of the calculation in automatically generated FORTRAN codes. As a result of using these tools it was possible to construct a numerical realization of our calculations by an almost direct implementation of simple expressions (resulting in only a couple steps from the fundamental equations) in FORTRAN. After reading this section and section 4.4 the reader is encouraged to go to Appendixes B and C before reading the remainder of the text. After following the simple tutorials presented there it should be possible to easily construct working FORTRAN implementations of mathematical expressions that make up the bound state and transition operator calculations in parallel to the text. Additionally, a number of notebooks that create FORTRAN implementations of most of the basic building blocks of our computations are distributed with this thesis.

For our purposes we will use a picture in which the proton and the neutron are two different charge states of the same particle - the nucleon. This idea proved to be effective in systems with strong interactions and has roots in the work of Heisenberg. It stems from the observed close similarity of the proton and the neutron properties: they are both spin $\frac{1}{2}$ particles and have similar masses (proton mass: $M_p = 938.272046(21) \left[\frac{\text{MeV}}{c^2}\right]$, neutron mass: $M_n = 939.565378(21) \left[\frac{\text{MeV}}{c^2}\right]$). We will in our later calculations neglect the small difference in their masses and use instead the "nucleon mass", $m = \frac{1}{2}(M_n + M_p)$. What follows is a description of states and basic operators that will be used in the text.

The proton and the neutron belong to the isospin doublet and both have isospin $\frac{1}{2}$. The positively charged proton has the isospin projection $+\frac{1}{2}$ and the neutron has the isospin projection $-\frac{1}{2}$. For example, a system in which the first particle is a proton and the second particle is a neutron can be considered as the following isospin state of the two-nucleon system:

$$\mid \frac{1}{2}\frac{1}{2}\rangle \otimes \mid \frac{1}{2} - \frac{1}{2}\rangle. \tag{4.1}$$

The general spin (isospin) state of any 2N system can be written as a linear combination of the 4 possible tensor product basis states:

$$\mid \frac{1}{2}\nu_1\rangle \otimes \mid \frac{1}{2}\nu_2\rangle, \tag{4.2}$$

where ν_1, ν_2 are the projections of the spin (isospin) of particles 1, 2. A similar basis can be created for the 3N system:

$$\mid \frac{1}{2}\nu_1\rangle \otimes \mid \frac{1}{2}\nu_2\rangle \otimes \mid \frac{1}{2}\nu_3\rangle, \tag{4.3}$$

with ν_3 being the spin (isospin) projection of the third particle.

A detailed discussion of the practical realization of (4.2) and (4.3) with the use of the Kronecker product will be given in section 4.4. This realization allows for a straightforward construction of isospin and spin states and (isospin, spin) operators in the form of vectors and matrices and is crucial in the numerical realization of our calculations.

The classical Hamiltonian for two non-interacting particles with masses M_1, M_2 has the form:

$$H_0^{2N} = \frac{\boldsymbol{k}_1^2}{2M_1} + \frac{\boldsymbol{k}_2^2}{2M_2} \tag{4.4}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$ denote momenta conjugate to the positions $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ of individual particles. The introduction of a different set of coordinates:

$$\begin{aligned} \boldsymbol{r} &= \boldsymbol{x}_1 - \boldsymbol{x}_2, \\ \boldsymbol{R} &= \frac{M_1\boldsymbol{x}_1 + M_2\boldsymbol{x}_2}{M_1+M_2}, \end{aligned} \tag{4.5}$$

and their conjugate momenta:

$$\begin{aligned} \boldsymbol{p} &= \frac{M_2\boldsymbol{k}_1}{M_1+M_2} - \frac{M_1\boldsymbol{k}_2}{M_1+M_2}, \\ \boldsymbol{K} &= \boldsymbol{k}_1 + \boldsymbol{k}_2, \end{aligned} \tag{4.6}$$

leads to a new form of the free Hamiltonian:

$$H_0^{2N} = \frac{\boldsymbol{K}^2}{2\left(M_1+M_2\right)} + \frac{\left(M_1+M_2\right)\boldsymbol{p}^2}{2M_1M_2}. \tag{4.7}$$

In the above equation, the reduced mass of the two particle system $\mu = \frac{M_1M_2}{M_1+M_2}$ is immediately recognizable.

Following the quantization procedure, operators $\check{\boldsymbol{r}}$, $\check{\boldsymbol{p}}$ and $\check{\boldsymbol{R}}$, $\check{\boldsymbol{K}}$ for quantities from (4.5) and (4.6) will fulfill the standard commutation relations for position and momenta. The free Hamiltonian operator of the two particle system can be written using (4.7) in the form:

$$\begin{aligned} \check{H}_0^{2N} &= \frac{\check{\boldsymbol{k}}_1^2}{2M_1} + \frac{\check{\boldsymbol{k}}_1^2}{2M_2} \\ &= \frac{\check{\boldsymbol{K}}^2}{2\left(M_1+M_2\right)} + \frac{\left(M_1+M_2\right)\check{\boldsymbol{p}}^2}{2M_1M_2} \\ &= \frac{\check{\boldsymbol{p}}^2}{m} + \frac{\check{\boldsymbol{K}}^2}{4m}, \end{aligned} \tag{4.8}$$

where (for the 2N system) M_i is either M_p or M_n and in the last line we used our approximation about the identical masses of the proton and neutron ($M_p = M_n = m$). As a consequence we get $\check{\boldsymbol{p}} = \frac{1}{2}\left(\check{\boldsymbol{k}}_1 - \check{\boldsymbol{k}}_2\right)$ for the relative momentum operator of the two-nucleons and $\check{\boldsymbol{K}} = \check{\boldsymbol{k}}_1 + \check{\boldsymbol{k}}_2$ for the total momentum operator. We introduce the following complete set of 2N momentum states that will be used in further calculations:

$$\mid \boldsymbol{p}\boldsymbol{K}\rangle \tag{4.9}$$

States (4.9) obey:

$$\check{\boldsymbol{p}} \mid \boldsymbol{p}\boldsymbol{K}\rangle = \boldsymbol{p} \mid \boldsymbol{p}\boldsymbol{K}\rangle, \tag{4.10}$$

$$\check{\boldsymbol{K}} \mid \boldsymbol{p}\boldsymbol{K}\rangle = \boldsymbol{K} \mid \boldsymbol{p}\boldsymbol{K}\rangle \tag{4.11}$$

together with the completeness relation:

$$\int \mathrm{d}^3\boldsymbol{p}\ \mathrm{d}^3\boldsymbol{K} \mid \boldsymbol{p}\boldsymbol{K}\rangle\langle \boldsymbol{p}\boldsymbol{K} \mid = \check{1} \tag{4.12}$$

and the normalization condition:

$$\langle \boldsymbol{p}'\boldsymbol{K}' \mid \boldsymbol{p}\boldsymbol{K}\rangle = \delta^3(\boldsymbol{p}'-\boldsymbol{p})\delta^3(\boldsymbol{K}'-\boldsymbol{K}). \tag{4.13}$$

Finally, the link to single particle momentum eigenstates is given by:

$$\langle \boldsymbol{k}_1 \boldsymbol{k}_2 \mid \boldsymbol{p} \boldsymbol{K} \rangle = \delta^3 \left(\boldsymbol{p} - \frac{1}{2} \left(\boldsymbol{k}_1 - \boldsymbol{k}_2 \right) \right) \delta^3 \left(\boldsymbol{K} - \left(\boldsymbol{k}_1 + \boldsymbol{k}_2 \right) \right). \tag{4.14}$$

The classical Hamiltonian for three non-interacting particles with masses M_1, M_2, M_3 has the form:

$$H_0^{3N} = \frac{\boldsymbol{k}_1^2}{2M_1} + \frac{\boldsymbol{k}_2^2}{2M_2} + \frac{\boldsymbol{k}_3^2}{2M_3}, \tag{4.15}$$

where $\boldsymbol{k}_1$, $\boldsymbol{k}_2$, $\boldsymbol{k}_3$ denote momenta conjugate to the positions of individual particles $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$. The introduction of a different set of coordinates (see for example [44]):

$$\begin{aligned} \boldsymbol{r}_p &= \boldsymbol{x}_2 - \boldsymbol{x}_1, \\ \boldsymbol{r}_q &= \boldsymbol{x}_3 - \frac{1}{M_1 + M_2} \left(M_1 \boldsymbol{x}_1 + M_2 \boldsymbol{x}_2 \right), \\ \boldsymbol{R} &= \frac{M_1 \boldsymbol{x}_1 + M_2 \boldsymbol{x}_2 + M_3 \boldsymbol{x}_3}{M_1 + M_2 + M_3} \end{aligned} \tag{4.16}$$

and their conjugate momenta (referred to as the Jacobi momenta):

$$\begin{aligned} \boldsymbol{p} &= \frac{M_1 \boldsymbol{k}_2}{M_1 + M_2} - \frac{M_2 \boldsymbol{k}_1}{M_1 + M_2} \\ \boldsymbol{q} &= \frac{\left(M_1 + M_2 \right) \boldsymbol{k}_3 - M_3 \left(\boldsymbol{k}_1 + \boldsymbol{k}_2 \right)}{M_1 + M_2 + M_3} \\ \boldsymbol{K} &= \left(\boldsymbol{k}_1 + \boldsymbol{k}_2 + \boldsymbol{k}_3 \right) \end{aligned} \tag{4.17}$$

again leads to a new form of the Hamiltonian:

$$H_0^{3N} = \frac{\boldsymbol{K}^2}{2 \left(M_1 + M_2 + M_3 \right)} + \frac{\left(M_1 + M_2 \right) \boldsymbol{p}^2}{2 M_1 M_2} + \frac{\left(M_1 + M_2 + M_3 \right) \boldsymbol{q}^2}{2 \left(M_1 + M_2 \right) M_3}. \tag{4.18}$$

As before, the quantization procedure leads to operators $\check{\boldsymbol{r}}_p$, $\check{\boldsymbol{p}}$ and $\check{\boldsymbol{r}}_q$, $\check{\boldsymbol{q}}$ and $\check{\boldsymbol{R}}$, $\check{\boldsymbol{K}}$ for (4.16) and (4.17) that follow the standard commutation relations for position and momenta. The free Hamiltonian of the 3N system can be written using (4.18) in the form:

$$\begin{aligned} \check{H}_0^{3N} &= \frac{\check{\boldsymbol{k}}_1^2}{2M_1} + \frac{\check{\boldsymbol{k}}_2^2}{2M_2} + \frac{\check{\boldsymbol{k}}_3^2}{2M_3} \\ &= \frac{\check{\boldsymbol{K}}^2}{2 \left(M_1 + M_2 + M_3 \right)} + \frac{\left(M_1 + M_2 \right) \check{\boldsymbol{p}}^2}{2 M_1 M_2} + \frac{\left(M_1 + M_2 + M_3 \right) \check{\boldsymbol{q}}^2}{2 \left(M_1 + M_2 \right) M_3} \\ &= \frac{\check{\boldsymbol{p}}^2}{m} + \frac{3 \check{\boldsymbol{q}}^2}{4m} + \frac{\check{\boldsymbol{K}}^2}{6m}, \end{aligned} \tag{4.19}$$

where in the last line we again make the approximation about the masses of the proton and neutron ($M_p = M_n = m$). As a consequence we get $\check{\boldsymbol{K}} = \check{\boldsymbol{k}}_1 + \check{\boldsymbol{k}}_2 + \check{\boldsymbol{k}}_3$ for total momentum operator (for the 3N system) and $\check{\boldsymbol{p}}$ and $\check{\boldsymbol{q}}$ for the Jacobi momentum operators. Jacobi momenta can actually be defined in three ways (using different permutations of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ from (4.16)) see for example [45, 46]:

$$\check{\boldsymbol{p}}_1 = \frac{1}{2} \left(\check{\boldsymbol{k}}_2 - \check{\boldsymbol{k}}_3 \right), \tag{4.20}$$

$$\check{\boldsymbol{q}}_1 = \frac{2}{3}\left(\check{\boldsymbol{k}}_1 - \frac{1}{2}\left(\check{\boldsymbol{k}}_2 + \check{\boldsymbol{k}}_3\right)\right), \tag{4.21}$$

$$\check{\boldsymbol{p}}_2 = \frac{1}{2}\left(\check{\boldsymbol{k}}_3 - \check{\boldsymbol{k}}_1\right), \tag{4.22}$$

$$\check{\boldsymbol{q}}_2 = \frac{2}{3}\left(\check{\boldsymbol{k}}_2 - \frac{1}{2}\left(\check{\boldsymbol{k}}_3 + \check{\boldsymbol{k}}_1\right)\right), \tag{4.23}$$

$$\check{\boldsymbol{p}}_3 = \frac{1}{2}\left(\check{\boldsymbol{k}}_1 - \check{\boldsymbol{k}}_2\right), \tag{4.24}$$

$$\check{\boldsymbol{q}}_3 = \frac{2}{3}\left(\check{\boldsymbol{k}}_3 - \frac{1}{2}\left(\check{\boldsymbol{k}}_1 + \check{\boldsymbol{k}}_2\right)\right). \tag{4.25}$$

We introduce the complete set of 3N momentum states that will be used in our calculations:

$$\mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i, \tag{4.26}$$

that follow:

$$\check{\boldsymbol{p}}_i \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i = \boldsymbol{p}_i \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i, \tag{4.27}$$

$$\check{\boldsymbol{q}}_i \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i = \boldsymbol{q}_i \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i, \tag{4.28}$$

$$\check{\boldsymbol{K}} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i = \boldsymbol{K} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_i. \tag{4.29}$$

Typically we will be using $i = 1$ with particle 1 being the *spectator*. From now on we will assume that whenever Jacobi momenta are used and this index is not specified, they are given by (4.20) and (4.21). The states obey the completeness relation:

$$\int \mathrm{d}^3\boldsymbol{p}\ \mathrm{d}^3\boldsymbol{q}\ \mathrm{d}^3\boldsymbol{K} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle\langle\boldsymbol{p}\boldsymbol{q}\boldsymbol{K} \mid = \check{1} \tag{4.30}$$

and the normalization condition:

$$\langle\boldsymbol{p}'\boldsymbol{q}'\boldsymbol{K}' \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle = \delta^3(\boldsymbol{p}'-\boldsymbol{p})\delta^3(\boldsymbol{q}'-\boldsymbol{q})\delta^3(\boldsymbol{K}'-\boldsymbol{K}). \tag{4.31}$$

Finally, the link between Jacobi momentum states and single particle momentum states is given by (see for example [45, 46]):

$$\begin{aligned} &\langle\boldsymbol{k}_1\boldsymbol{k}_2\boldsymbol{k}_3 \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle = \\ &= \delta^3\left(\boldsymbol{p}_1 - \frac{1}{2}\left(\check{\boldsymbol{k}}_2 - \check{\boldsymbol{k}}_3\right)\right)\delta^3\left(\boldsymbol{q}_1 - \frac{2}{3}\left(\check{\boldsymbol{k}}_1 - \frac{1}{2}\left(\check{\boldsymbol{k}}_2 + \check{\boldsymbol{k}}_3\right)\right)\right) \\ &\quad \delta^3\left(\boldsymbol{K} - \left(\boldsymbol{k}_1 + \boldsymbol{k}_2 + \boldsymbol{k}_3\right)\right). \end{aligned} \tag{4.32}$$

4.2 The transition operator and 2N scattering

This section is written to give a very basic introduction to the quantum mechanical description of the 2N scattering process. It contains a short, simplified sketch of the derivations in [45, 46] that were modified to be consistent with 4.1. These two references can be consulted for more information or to fill any gaps in our discussion. We introduce equations that will be the subject of our three dimensional calculations in Chapter 7. The discussion presented here is limited to 2N systems. The transition to more complex systems of three or more particles is described for example in [47], which gives a detailed description of the possible 3N scattering processes and a way to include 3N forces into the calculations. We would like to again stress that the following discussion is simplified and does not strive to be complete; a good place to look for a better understanding of the underlying physics is [45, 46].

In a typical experimental setup, the initial state of the 2N system is, to some extent, under control. This means that initially the system is prepared to be a wave packet and we can safely assume that the time evolution of this initial (*non-interacting*) quantum mechanical system is governed by the time dependent Schrödinger equation:

$$i\hbar\partial_t \mid \psi(t)\rangle = \check{H}_0 \mid \psi(t)\rangle \tag{4.33}$$

with the free Hamiltonian $\check{H}_0$ (for the case of 2N systems (4.8) is used). More specifically, at some time t_0:

$$\mid \psi(t_0)\rangle = \int \mathrm{d}^3\boldsymbol{p} \mid \boldsymbol{p}\rangle f(\boldsymbol{p}) \tag{4.34}$$

where for 2N systems relative momentum eigenstates $\mid \boldsymbol{p}\boldsymbol{K} = 0\rangle \equiv\mid \boldsymbol{p}\rangle$ from (4.9) are used (we separate the CM motion from the relative motion of the two nucleons since the total momentum of the system is conserved) and $f(\boldsymbol{p})$ is the momentum space representation of $\mid \psi(t_0)\rangle$. Each $\mid \boldsymbol{p}\rangle$ separately is an eigenstate of the free Hamiltonian:

$$\check{H}_0 \mid \boldsymbol{p}\rangle = E \mid \boldsymbol{p}\rangle \tag{4.35}$$

to energy $E = \frac{\boldsymbol{p}^2}{m}$.

This simple picture breaks down after we allow the particles to interact. The time propagation of the *interacting* state is then governed by:

$$i\hbar\partial_t \mid \Psi(t)\rangle = \left(\check{H}_0 + \check{V}\right) \mid \Psi(t)\rangle \equiv \check{H} \mid \Psi(t)\rangle \tag{4.36}$$

with a 2N potential $\check{V}$. In a typical experimental setup, where the initial non-interacting state of the system is (to some extent) known it is necessary to make predictions about the interacting state with the potential $\check{V}$.

It is shown in [45] that a sufficient condition that links the non-interacting state governed by (4.33) - $\mid \psi_0(t)\rangle$ and the interacting state - $\mid \Psi^{(+)}(t)\rangle$ governed by (4.36) is:

$$\lim_{t\to-\infty} \| \mid \Psi^{(+)}(t)\rangle - \mid \psi_0(t)\rangle\| = 0 \tag{4.37}$$

where $\| \mid \psi\rangle\| \equiv \sqrt{\langle\psi \mid \psi\rangle}$.

Condition (4.37) can be reformulated using the unitarity of time propagation operators ($\exp\left(i\check{H}(t-t_0)\right)\exp\left(-i\check{H}(t-t_0)\right) = \check{1}$, $\hbar = 1$) [45]:

$$\begin{aligned} \| \mid \Psi^{(+)}(t)\rangle - \mid \psi_0(t)\rangle\| &= \\ \|\exp\left(-i\check{H}(t-t_0)\right) \mid \Psi^{(+)}(t_0)\rangle - \exp\left(-i\check{H}_0(t-t_0)\right) \mid \psi_0(t_0)\rangle\| &= \\ \| \mid \Psi^{(+)}(t_0)\rangle - \exp\left(i\check{H}(t-t_0)\right)\exp\left(-i\check{H}_0(t-t_0)\right) \mid \psi_0(t_0)\rangle\| &= \end{aligned} \tag{4.38}$$

to:

$$\begin{aligned} \mid \Psi^{(+)}(t_0)\rangle &= \left(\lim_{\tau\to-\infty} \exp\left(i\check{H}\tau\right)\exp\left(-i\check{H}_0\tau\right)\right) \mid \psi_0(t_0)\rangle \\ &\equiv \check{\Omega}^{(+)} \mid \psi_0(t_0)\rangle. \end{aligned} \tag{4.39}$$

This result is a link at time t_0 between the interacting state $\mid \Psi^{(+)}(t_0)\rangle$ (satisfying (4.36) with $\check{V}$) and the non-interacting state $\mid \psi_0(t_0)\rangle$ (satisfying (4.33) without $\check{V}$). The interacting state can be calculated with the use of the Möller wave operator $\check{\Omega}^{(+)}$. This link is crucial because it allows the calculation of scattering observables.

Combining (see for example [45]):

$$\lim_{\tau\to-\infty} g(\tau) = \lim_{\epsilon\to 0^+} \epsilon \int_{-\infty}^{0} \mathrm{d}\tau e^{\epsilon t} g(t)$$

with the assumption (4.35), the Möller wave operator in (4.39) can be rewritten as:

$$\check{\Omega}^{(+)} = \int \mathrm{d}^3\boldsymbol{p} f(\boldsymbol{p}) \check{\Omega}^{(+)}_{E=\frac{p^2}{m}} \mid \boldsymbol{p}\rangle\langle \boldsymbol{p} \mid . \tag{4.40}$$

In this equation we introduce a new operator:

$$\check{\Omega}^{(+)}_E = \lim_{\epsilon\to 0^+} i\epsilon \left(E + i\epsilon - \check{H}\right)^{-1} \tag{4.41}$$

with the energy index E to indicate that we work with free Hamiltonian eigenstates. Additionally from now on it is assumed that whenever ϵ is encountered, the limit $\epsilon \to 0^+$ is to be taken and the symbol of the limit will be dropped.

Introducing two new definitions for the resolvent operators:

$$\check{G}(z) = \left(z - \check{H}\right)^{-1}, \tag{4.42}$$

$$\check{G}_0(z) = \left(z - \check{H}_0\right)^{-1} \tag{4.43}$$

we arrive at [46]:

$$\begin{gathered} \check{G}_0(z)^{-1} - \check{G}(z)^{-1} = z - \check{H}_0 - z + \check{H} = \check{V} \\ \check{G}_0(z)\left(\check{G}_0^{-1}(z) - \check{G}^{-1}(z)\right)\check{G}(z) = \check{G}_0(z)\check{V}\check{G}(z) = \check{G}(z) - \check{G}_0(z) \\ \check{G}(z) = \check{G}_0(z) + \check{G}_0(z)\check{V}\check{G}(z). \end{gathered} \tag{4.44}$$

Equation (4.41) can now be rewritten using (4.44) as:

$$\check{\Omega}^{(+)}_E = \check{1} + i\epsilon \left(E + i\epsilon - \check{H}_0\right)^{-1} \check{V} \left(E + i\epsilon - \check{H}\right)^{-1}. \tag{4.45}$$

When applied to the free state this yields:

$$\begin{aligned} \mid \boldsymbol{p}^{(+)}\rangle &= \check{\Omega}^{(+)}_E \mid \boldsymbol{p}\rangle \\ &= \mid \boldsymbol{p}\rangle + \check{G}_0(E+i\epsilon)\check{V}\left(i\epsilon\check{G}(E+i\epsilon) \mid \boldsymbol{p}\rangle\right) \\ &= \mid \boldsymbol{p}\rangle + \check{G}_0(E+i\epsilon)\check{V} \mid \boldsymbol{p}^{(+)}\rangle \end{aligned} \tag{4.46}$$

where $\mid \boldsymbol{p}^{(+)}\rangle$ is an interacting state generated from the free state $\mid \boldsymbol{p}\rangle$ and in the final line we used (4.41). With a new definition of the transition operator $\check{t}(E)$:

$$\check{V} \mid \boldsymbol{p}^{(+)}\rangle \equiv \check{t}(E) \mid \boldsymbol{p}\rangle \tag{4.47}$$

equation (4.46) takes a very simple form useful for describing scattering experiments and allows the calculation, at time t_0, of an interacting state $\mid \boldsymbol{p}^{(+)}\rangle$ from a non-interacting state $\mid \boldsymbol{p}\rangle$ (satisfying (4.35)):

$$\mid \boldsymbol{p}^{(+)}\rangle = \left(\check{1} + \check{G}_0(E+i\epsilon)\check{t}(E)\right) \mid \boldsymbol{p}\rangle \tag{4.48}$$

For the general non-interacting state (4.34) equation (4.48) takes on the form:

$$\mid \Psi^{(+)}(t)\rangle = \int \mathrm{d}^3\boldsymbol{p} f(\boldsymbol{p}) \exp\left(-i\frac{\boldsymbol{p}^2}{m}(t-t_0)\right) \left(\check{1} + \check{G}_0(\frac{p^2}{m}+i\epsilon)\check{t}(E=\frac{p^2}{m})\right) \mid \boldsymbol{p}\rangle, \tag{4.49}$$

where we explicitly added the time dependence. This expression can be used to construct the interacting state for the 2N system.

The equation for the $\check{t}(E)$ operator can be found by applying $\check{V}$ to both sides of (4.46) (see for example [46]):

$$\check{V} \mid \boldsymbol{p}^{(+)}\rangle = \check{V} \mid \boldsymbol{p}\rangle + \check{V}\check{G}_0(E+i\epsilon)\check{V} \mid \boldsymbol{p}^{(+)}\rangle,$$

$$\check{t}(E) \mid \boldsymbol{p}\rangle = \left(\check{V} + \check{V}\check{G}_0(E+i\epsilon)\check{t}(E)\right) \mid \boldsymbol{p}\rangle.$$

This is generalized for any $\mid \boldsymbol{p}\rangle$, we can write down an equation for $\check{t}(E)$:

$$\check{t}(E) = \check{V} + \check{V}\check{G}_0(E+i\epsilon)\check{t}(E) \tag{4.50}$$

Relation (4.50) is called the Lippmann-Schwinger equation (LSE) and will be the object of our three dimensional calculations. Further considerations involving $\check{t}$ can lead to expressions for the cross section and other observables. This discussion will not be presented here, the reader is referred to [45] for more information.

The transition operator can also be asociated with an infinite series. It is a simple task to check that the following infinite expansion follows equation (4.50):

$$\check{t}(E) = \check{V} + \check{V}\check{G}_0(E+i\epsilon)\check{V} + \check{V}\check{G}_0(E+i\epsilon)\check{V}\check{G}_0(E+i\epsilon)\check{V} + \ldots. \tag{4.51}$$

We will later encounter this series when constructing the bound state of the three-nucleon system. The series (4.51) known as the Neumann series is not always convergent.

It is important to consider the case of the three-nucleon system with only 2N interactions. In such systems we will typically split the problem and consider a subset with two particles separated from the third "spectator particle". This will allow us to use the 2N transition operator for 3N calculations. For a general 3N system (with states from (4.26) and a non zero total momentum $\boldsymbol{K}$) we define $\check{t}_{3N}(E_{3N})$ as:

$$\langle \boldsymbol{p}'\boldsymbol{q}'\boldsymbol{K}' \mid \check{t}_{3\mathrm{N}}(E_{3N}) \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle = \delta(\boldsymbol{q}'-\boldsymbol{q})\delta(\boldsymbol{K}'-\boldsymbol{K})\langle \boldsymbol{p}' \mid \check{t}(E_{2N} = E_{3N} - \frac{3}{4m}\boldsymbol{q}^2 - \frac{1}{6m}\check{K}^2) \mid \boldsymbol{p}\rangle, \tag{4.52}$$

where $E_{3N(2N)}$ is the energy of the 3N (2N) system. For 3N systems with only two interacting particles series (4.51) can also be associated with $t_{3N}(E_{3N})$. This observation will lead us to identify the transition operator in 3N bound state calculations. Most of the discussion in this chapter dealt with free states that have the energy $E_{2N} > 0$ and with the transition operator that is linked to this energy. For bound states however, we will have to consider cases when

$E_{2N} < 0$ and matrix elements of the transition operator between states with various energies.

The discussion presented here is, as was mentioned at the beginning, meant only to give an overall meaning of equations used in our further discussions. A more complete introduction to quantum mechanical scattering can be found in [45, 46, 47]. In order to solve the LSE additional constraints have to be put on the operator form of the 2N potential, Chapter 7 discusses this in detail. The resulting equations constitute a liner system. Due to the large dimensions of operators involved, special methods have to be used. Appendix A introduces algorithms used in our numerical treatment of large linear operators.

4.3 Equations for the 2N and 3N bound states

This chapter introduces the basic constituents of bound state calculations. We start with a discussion for two particles and then go on to the different forms of bound state integral equations for the three particle system. We will use 2N, 3N to mark two-nucleon and three-nucleon calculations.

The bound state equation for the two-particle system has the familiar form:

$$\left(\check{H}_0 + \check{V}\right) \mid \Psi^{2N}\rangle = E \mid \Psi^{2N}\rangle, \tag{4.53}$$

with $\mid \Psi^{2N}\rangle$ being the two-particle wave function, $\check{H}_0$ being the free Hamiltonian (for example (4.8)), $\check{V}$ being the 2N potential (invariant under the exchange of the particles) and $E < 0$ being the bound state energy. Equation (4.53) can be written in an integral form:

$$\mid \Psi^{2N}\rangle = \left(E - \check{H}_0\right)^{-1} \check{V} \mid \Psi^{2N}\rangle \tag{4.54}$$

or in a more compact form using the free propagator introduced earlier (4.43):

$$\mid \Psi^{2N}\rangle = \check{G}_0(E)\check{V} \mid \Psi^{2N}\rangle \tag{4.55}$$

The solution to (4.55) can be found by considering a similar eigen equation:

$$\check{A}(E) \mid \Psi\rangle = \lambda \mid \Psi\rangle \tag{4.56}$$

with $\check{A}(E) = \check{G}_0(E)\check{V}$ and $\mid \Psi\rangle =\mid \Psi^{2N}\rangle$. If a solution of (4.56) - $\mid \Psi^{2N}_{bound}\rangle$ is found for an energy $E = E^{2N}_{\text{bound}} < 0$ such that $\lambda = 1$ then this solution it is also a solution of (4.53) and (4.55). This method of calculating the bound state is going to be used also for three particle calculations but with a different form of $\check{A}(E)$.

The bound state equation for the 3N system has the form:

$$\left(\check{H}_0 + \sum_{i=1}^{3} \check{V}_i\right) \mid \Psi^{3N}\rangle = E \mid \Psi^{3N}\rangle \tag{4.57}$$

where $\mid \Psi^{3N}\rangle$ is the three particle wave function, $E < 0$ is the bound state energy and H_0 is the free Hamiltonian (for example (4.19)). Finally $\check{V}_i$ are the two-nucleon potential operators (that are symmetric under the exchange of particles j and k such that $i \neq j \neq k \neq i$). Equation (4.57) can be rewritten:

$$\mid \Psi^{3N}\rangle = (E - \check{H}_0)^{-1} \sum_{i=1}^{3} \check{V}_i \mid \Psi^{3N}\rangle \equiv \sum_{i=1}^{3} \mid \psi_i^{3N}\rangle \tag{4.58}$$

where $| \psi_i^{3N} \rangle$ is a Faddeev component of the wave function. With the definition of the free propagator (4.43) the Faddeev component satisfies:

$$| \psi_i^{3N} \rangle \equiv \check{G}_0(E)\check{V}_i \mid \Psi^{3N} \rangle. \tag{4.59}$$

Using the symmetries of $\check{V}_i$ and considering identical particles it can easily be checked that it has the following symmetry properties (see for example [45, 46]):

$$| \psi_2^{3N} \rangle = \check{P}_{12}\check{P}_{23} \mid \psi_1^{3N} \rangle \tag{4.60}$$

$$| \psi_3^{3N} \rangle = \check{P}_{13}\check{P}_{23} \mid \psi_1^{3N} \rangle \tag{4.61}$$

where $\check{P}_{ij}$ is a permutation operator exchanging particles i and j.

Equations (4.60) and (4.61) can be used to rewrite equation (4.59) for a single Faddeev component:

$$\begin{aligned} | \psi^{3N} \rangle &= \check{G}_0(E)\check{V} \left(| \psi^{3N} \rangle + \check{P}_{12}\check{P}_{23} \mid \psi^{3N} \rangle + \check{P}_{13}\check{P}_{23} \mid \psi^{3N} \rangle\right) \\ &\equiv \check{G}_0(E)\check{V} \left(\check{1} + \check{P}\right) \mid \psi^{3N} \rangle \end{aligned} \tag{4.62}$$

where the 1 in $| \psi_1^{3N} \rangle$ and $\check{V}_1$ was dropped and we introduce the permutation operator:

$$\check{P} = \check{P}_{12}\check{P}_{23} + \check{P}_{13}\check{P}_{23}. \tag{4.63}$$

The solution to (4.62) can be found considering (4.56) with with

$$\check{A}(E) = \check{G}_0(E)\check{V} \left(\check{1} + \check{P}\right)$$

and $| \Psi \rangle =| \psi^{3N} \rangle$. If a solution $| \psi^{3N}_{\text{Faddeev}} \rangle$ is found for an energy $E = E^{3N}_{\text{bound}} < 0$ such that $\lambda = 1$ then this solution it is also a solution of (4.59) and the full bound state of the 3N system can be obtained using:

$$| \Psi^{3N}_{bound} \rangle = \left(\check{1} + \check{P}\right) \mid \psi^{3N}_{\text{Faddeev}} \rangle \tag{4.64}$$

The introduction of 3N forces (or three particle forces) transforms the bound state equation:

$$\left(\check{H}_0 + \sum_{i=1}^{3} \check{V}_i + \sum_{i=1}^{3} \check{V}^{(i)}\right) \mid \Psi^{3N} \rangle = E \mid \Psi^{3N} \rangle \tag{4.65}$$

where $\check{V}^{(i)}$ is that part of the 3NF that is symmetric under the exchange of particles j and k such that $i \neq j \neq k \neq i$. Symmetry considerations used in the case without the 3NF can be repeated and result in a new equation for the Faddeev component:

$$| \psi^{3N} \rangle = \check{G}_0(E)\check{V} \left(\check{1} + \check{P}\right) \mid \psi^{3N} \rangle + \check{G}_0(E)\check{V}^{(1)} \left(\check{1} + \check{P}\right) \mid \psi^{3N} \rangle \tag{4.66}$$

The solution to (4.66) can be found using (4.56) with $\check{A}(E) = \check{G}_0(E)\check{V}(\check{1}+\check{P}) + \check{G}_0(E)\check{V}^{(1)}(\check{1} + \check{P})$.

An alternative approach to finding the bound state of the three particle system utilizes the transition operator. This approach is not explored in this thesis but was described in [35]. Equation (4.57) can be written as:

$$| \psi^{3N} \rangle = \left(\check{1} - \check{G}_0(E)\check{V}\right)^{-1} \check{G}_0(E)\check{V}\check{P} \mid \psi^{3N} \rangle \tag{4.67}$$

The right side of this equation can be expanded:

$$\begin{aligned}
\left(\check{1} - \check{G}_0(E)\check{V}\right)^{-1} \check{G}_0(E)\check{V} &= \\
\left(\check{1} + \check{G}_0(E)\check{V} + \check{G}_0(E)\check{V}\check{G}_0(E)\check{V} + \ldots\right) \check{G}_0(E)\check{V} &= \\
\check{G}_0(E) \left(\check{V} + \check{V}\check{G}_0(E)\check{V} + \check{V}\check{G}_0(E)\check{V}\check{G}_0(E)V + \ldots\right) &\equiv \\
\check{G}_0(E)\check{t}_{3N}(E) & \qquad (4.68)
\end{aligned}$$

We identify in (4.68) the same expansion as in (4.51), because we are dealing with the first Faddeev component (particle 1 is the spectator) we substitute $\check{t}_{3N}(E)$ from (4.52) in the last line. In this case however, there is no guarantee that the energy argument of the two-nucleon transition operator from (4.52) will be positive. The transition operator satisfying the LSE (4.50) will be calculated for a negative value of E (see for example [45, 46]), this will in some cases lead to singularities (see for example [46]). Equation (4.67) can be rewritten using $\check{t}_{3N}$:

$$| \psi^{3N} \rangle = \check{G}_0(E)\check{t}_{3N}(E)\check{P} \mid \psi^{3N} \rangle \qquad (4.69)$$

and solved using (4.56) with $A(E) = \check{G}_0(E)\check{t}_{3N}(E)\check{P}$.

The introduction of the three-nucleon force (3NF) into the time independent Schrödinger equation leads to additional terms in (4.69):

$$| \psi^{3N} \rangle = \check{G}_0(E)\check{t}_{3N}(E)\check{P} \mid \psi^{3N} \rangle + \left(\check{1} - \check{G}_0(E)\check{V}\right)^{-1} \check{G}_0(E)\check{V}^{(1)}(\check{1} + \check{P}) \mid \psi^{3N} \rangle \qquad (4.70)$$

Using an expansion similar to (4.68) we arrive at the following version of the bound state equation, with the transition operator:

$$| \psi^{3N} \rangle = \check{G}_0(E)\check{t}_{3N}(E)\check{P} \mid \psi^{3N} \rangle + \left(\check{1} + \check{G}_0(E)\check{t}_{3N}(E)\right) \check{G}_0(E)V^{(1)}(\check{1} + \check{P}) \mid \psi^{3N} \rangle \qquad (4.71)$$

In the above equations for the 3N bound state, the transition operator acts in the three particle space. It is however, constructed entirely from two particle operators and a relation between the 3N and 2N representations of the transition operator can easily be worked out (see for example [45, 46]) and is shown in Chapter 7.

Equations (4.55) , (4.62) and (4.66) will be the basic constituents of our three dimensional calculations. We will not explore equations (4.69) and (4.71) for the bound state calculations that involve the transition operator - this path is described for example in [35]. In order to solve the equations additional constraints have to be put on the operator form of the bound state (both for the 2N and 3N system) - Chapters 6 and 10 discuss this in detail. The resulting equations (similarly as in the case of the transition operator) will constitute a liner system, more precisely a linear eigen system. The dimensions of operators involved will make it necessity to use Krylov subspace methods from Appendix A in order to construct a numerical realization.

4.4 Numerical realization of 2N and 3N states and operators

Having introduced the basic equations governing scattering and bound state calculations, we now consider in detail the matrix representation of the relevant states and operators. As described in Chapter 4.1 the proton and the neutron are assumed to be two different charge states of the same particle - the nucleon. We introduced the general basis isospin (spin) states for the 2N and 3N system in equations (4.2) and (4.3). For 2N systems there are 4 possible spin and 4 possible isospin basis states arising from the different values of the spin and isospin projections ν_1 and ν_2 of the two particles. This allows us to write the full isospin - spin state of the 2N system as a linear combination of $4 \times 4 = 16$ tensor product basis states and the full 3N state as a combination of 64 basis states. States and operators in the joined isospin - spin state space can therefore be implemented as 16 or 64 dimensional vectors and matrices respectively.

This matrix representation can be constructed with the use of the Kronecker Product (KP). The description of this operator can be found in most linear algebra textbooks. For a set of two operators $\left[\check{A}\right]$, $\left[\check{B}\right]$, the KP can be used to construct the matrix representation of the tensor product of two operators $\left[\check{A} \otimes \check{B}\right]$. If

$$\left[\check{A}\right] = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

acts in a space spanned by the two dimensional basis:

$$\left[\hat{\boldsymbol{e}}_1^A\right] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \left[\hat{\boldsymbol{e}}_2^A\right] = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.72}$$

and

$$\left[\check{B}\right] = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

acts in a space spanned by the two dimensional basis:

$$\left[\hat{\boldsymbol{e}}_1^B\right] = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \left[\hat{\boldsymbol{e}}_2^B\right] = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{4.73}$$

then

$$\left[\check{A} \otimes \check{B}\right] = \begin{pmatrix} A_{11}B_{11} & A_{11}B_{12} & A_{12}B_{11} & A_{12}B_{12} \\ A_{11}B_{21} & A_{11}B_{22} & A_{12}B_{21} & A_{12}B_{22} \\ A_{21}B_{11} & A_{21}B_{12} & A_{22}B_{11} & A_{22}B_{12} \\ A_{21}B_{21} & A_{21}B_{22} & A_{22}B_{21} & A_{22}B_{22} \end{pmatrix} \tag{4.74}$$

acting in the space:

$$\left[\hat{\boldsymbol{e}}_1^{A\otimes B}\right] = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \equiv \left[\hat{\boldsymbol{e}}_1^A \otimes \hat{\boldsymbol{e}}_1^B\right] \qquad \left[\hat{\boldsymbol{e}}_2^{A\otimes B}\right] = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} \equiv \left[\hat{\boldsymbol{e}}_1^A \otimes \hat{\boldsymbol{e}}_2^B\right]$$

$$\left[\hat{\boldsymbol{e}}_3^{A\otimes B}\right] = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \equiv \left[\hat{\boldsymbol{e}}_2^A \otimes \hat{\boldsymbol{e}}_1^B\right] \quad \left[\hat{\boldsymbol{e}}_4^{A\otimes B}\right] = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \equiv \left[\hat{\boldsymbol{e}}_2^A \otimes \hat{\boldsymbol{e}}_2^B\right] \tag{4.75}$$

is the matrix representation of the tensor product of operators $\check{A}$ and $\check{B}$. Tensor product states can be constructed in a similar matter. Let

$$[\boldsymbol{x}] = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

be a vector in (4.72) and

$$[\boldsymbol{y}] = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

be a vector in (4.73) then

$$[\boldsymbol{x} \otimes \boldsymbol{y}] = \begin{pmatrix} x_1 y_1 \\ x_1 y_2 \\ x_2 y_1 \\ x_2 y_2 \end{pmatrix}.$$

is the matrix representation of the tensor product of $\boldsymbol{x}$ and $\boldsymbol{y}$ in the basis (4.75). It can be seen how this can be used to construct (4.75) from (4.72) and (4.73).

In some cases it is better to consider only subsets of the full 2N, 3N isospin-spin state. For the 2N and 3N systems we can create a number of different sets of basis states using the KP. We would use a different basis when considering only the spin space of a 2N system such as the deuteron and a different basis when considering the full 3N state with isospin and spin. Different choices for the sets of basis states are listed in the Appendix in tables E.1, E.2, E.3, E.4, E.5, E.6 and will be referred to when necessary. These tables can serve as a template to check the matrix form of operators and states, a useful tool because the KP is sensitive to the order of its operands.

Additional definitions for the scalar and vector products together with the built in *Mathematica*® [38] definitions for the KP or Clebsch-Gordan coefficients made it possible to create a simple tool that can be used to construct all isospin - spin operators and states that are used in this thesis. Appendix B contains a description of the tools in the *util1N2N3N.m Mathematica*® [38] package (supplied with this thesis), this chapter contains a simple tutorial.

Two operators deserve special attention due to their frequent presence in our calculations. The first one is the permutation operator, exchanging two particles. The second operator - the projection operator - will be important when considering the disintegration of the deuteron.

The permutation operator can not be constructed using the KP because it acts simultaniously on the degrees of freedom of all the particles. In isospin (spin) space we will write out its matrix form explicitly (all the following definitions can be obtained using *util1N2N3N.m*). For the 2N system in basis from table E.5 the operator exchanging particles 1 and 2 has the following matrix representation:

$$\left[\check{P}_{12}\right]^{\text{2Nspin}} = \left[\check{P}_{12}\right]^{2Nisospin} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.76}$$

For the 3N system we have three permutation operators, in the basis from table

E.6 they can be impemented as:

$$\left[\check{P}_{12}\right]^{3\text{Nspin}} = \left[\check{P}_{12}\right]^{3\text{Nisospin}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.77}$$

$$\left[\check{P}_{23}\right]^{3\text{Nspin}} = \left[\check{P}_{23}\right]^{3\text{Nisospin}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.78}$$

$$\left[\check{P}_{13}\right]^{3\text{Nspin}} = \left[\check{P}_{13}\right]^{3\text{Nisospin}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.79}$$

The permutation operators should act simultaneously on the spin and isospin of the 2N and 3N systems. 16 and 64 dimensional matrices acting in the joined isospin - spin basis from tables E.2 and E.3 can be constructed using the KP because permutations do not mix isospin and spin degrees of freedom.

The action of the permutation operator in momentum space of the 2N system is straightforward. The exchange of two particles transforms single particle momentum eigenstates (4.14) in the following manner:

$$\check{P}_{12} \mid \boldsymbol{k}_1 \boldsymbol{k}_2 \rangle = \mid \boldsymbol{k}_2 \boldsymbol{k}_2 \rangle \tag{4.80}$$

It follows that for relative and total momentum eigenstates (4.9):

$$\check{P}_{12} \mid \boldsymbol{p} \boldsymbol{K} \rangle = \mid -\boldsymbol{p} \boldsymbol{K} \rangle. \tag{4.81}$$

For 3N systems the action of the permutation operators on single particle momentum eigenstates is still simple:

$$\check{P}_{12} \mid \boldsymbol{k}_1 \boldsymbol{k}_2 \boldsymbol{k}_3 \rangle = \mid \boldsymbol{k}_2 \boldsymbol{k}_2 \boldsymbol{k}_3 \rangle, \tag{4.82}$$

$$\check{P}_{23} \mid \boldsymbol{k}_1 \boldsymbol{k}_2 \boldsymbol{k}_3 \rangle = \mid \boldsymbol{k}_1 \boldsymbol{k}_3 \boldsymbol{k}_2 \rangle, \tag{4.83}$$

$$\check{P}_{13} \mid \boldsymbol{k}_1 \boldsymbol{k}_2 \boldsymbol{k}_3 \rangle = \mid \boldsymbol{k}_3 \boldsymbol{k}_2 \boldsymbol{k}_1 \rangle. \tag{4.84}$$

With Jacobi momentum states the problem is more complicated. For the first set in (4.20), (4.21):

$$\check{P}_{12} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_1 =\mid \frac{1}{4}(2\boldsymbol{p}+3\boldsymbol{q})\ \boldsymbol{p}-\frac{\boldsymbol{q}}{2}\rangle_1, \tag{4.85}$$

$$\check{P}_{23} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_1 =\mid -\boldsymbol{p}\ \boldsymbol{q}\rangle_1, \tag{4.86}$$

$$\check{P}_{13} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_1 =\mid \frac{1}{4}(2\boldsymbol{p}-3\boldsymbol{q})\ -\boldsymbol{p}-\frac{\boldsymbol{q}}{2}\rangle_1. \tag{4.87}$$

For the second set in (4.22), (4.23):

$$\check{P}_{12} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_2 =\mid \frac{1}{4}(2\boldsymbol{p}-3\boldsymbol{q})\ -\boldsymbol{p}-\frac{\boldsymbol{q}}{2}\rangle_2, \tag{4.88}$$

$$\check{P}_{23} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_2 =\mid \frac{1}{4}(2\boldsymbol{p}+3\boldsymbol{q})\ \boldsymbol{p}-\frac{\boldsymbol{q}}{2}\rangle_2, \tag{4.89}$$

$$\check{P}_{13} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_2 =\mid -\boldsymbol{p}\ \boldsymbol{q}\rangle_2. \tag{4.90}$$

For the third set in (4.24), (4.25):

$$\check{P}_{12} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_3 =\mid -\boldsymbol{p}\ \boldsymbol{q}\rangle_3, \tag{4.91}$$

$$\check{P}_{23} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_3 =\mid \frac{1}{4}(2\boldsymbol{p}-3\boldsymbol{q})\ -\boldsymbol{p}-\frac{\boldsymbol{q}}{2}\rangle_3, \tag{4.92}$$

$$\check{P}_{13} \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle_3 =\mid \frac{1}{4}(2\boldsymbol{p}+3\boldsymbol{q})\ \boldsymbol{p}-\frac{\boldsymbol{q}}{2}\rangle_3. \tag{4.93}$$

Package *util1N2N3N.m* contains definitions for all the three sets of Jacobi momenta permutations. Interesting examples with the use of these definitions can be found in Appendix B.

The projection operator $\check{C}_\gamma =\mid \gamma\rangle\langle\gamma \mid$ onto one of the 4 possible 2N isospin states:

$$\begin{aligned} \mid \gamma_1\rangle &= \mid J=0, M=0\rangle, \\ \mid \gamma_2\rangle &= \mid J=1, M=-1\rangle, \\ \mid \gamma_3\rangle &= \mid J=1, M=0\rangle, \\ \mid \gamma_4\rangle &= \mid J=1, M=1\rangle \end{aligned} \tag{4.94}$$

(with the isospin of the two particles coupled to a total isospin J with projection M) will be useful for our few-body calculations. In order to construct a matrix representation of $\check{C}$ it is necessary to first introduce the 4×4 orthogonal basis change matrix $[\beta]^{4\times 4}$ whose rows are simply matrix representations of $[\mid \gamma_{1\ldots 4}\rangle]^4$:

$$[\beta]^{4\times 4} = \begin{pmatrix} [\mid J=0, M=0\rangle] \\ [\mid J=1, M=-1\rangle] \\ [\mid J=1, M=0\rangle] \\ [\mid J=1, M=1\rangle] \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{4.95}$$

and the cast 4×4 matrix $[M(i)]^{4\times 4}$ that is zero everywhere except on the (i,i) position on its diagonal. The matrix representation of $\check{C}_\gamma$ in isospin basis (E.5) is then:

$$\left[\check{C}(\gamma_i)\right]^{\text{2Nisospin}} = [\beta^\dagger][M(i)][\beta]. \tag{4.96}$$

In particular:

$$\left[\check{C}_{\gamma_1}\right]^{\text{2Nisospin}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & -\frac{1}{2} & 0 \\ 0 & -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\left[\check{C}_{\gamma_2}\right]^{\text{2Nisospin}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\left[\check{C}_{\gamma_3}\right]^{\text{2Nisospin}} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\left[\check{C}_{\gamma_4}\right]^{\text{2Nisospin}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The $\check{C}(\gamma)$ operator can also be written in the joined isospin-spin basis from table E.2. It will be an identity operator $(\check{1})$ in spin space. The 16×16 matrix representation can be constructed using the KP:

$$[C(\gamma_i)]^{\text{2N}} = [C(\gamma_i)]^{\text{2Nisospin}} \otimes [\mathbf{1}]^{\text{2Nspin}} \tag{4.97}$$

from two 4×4 matrices using the *Mathematica*® definitions from *util1N2N3N.m*. In a similar fashion using the KP, a representation in the full 3N isospin-spin space can be created.

What remains is to find a momentum space representation of the free propagator $\check{G}_0$ from 4.43. With the definitions from equations (4.8) and (4.19) we can directly write out the matrix elements for the 2N system:

$$\langle \boldsymbol{p}'\boldsymbol{K}' \mid \check{G}_0(E) \mid \boldsymbol{p}\boldsymbol{K}\rangle = \delta^3(\boldsymbol{p}' - \boldsymbol{p})\delta^3(\boldsymbol{K}' - \boldsymbol{K})\frac{1}{E - \frac{\boldsymbol{p}^2}{m} - \frac{\boldsymbol{K}^2}{4m}} \tag{4.98}$$

and for the 3N system:

$$\begin{aligned} &\langle \boldsymbol{p}'\boldsymbol{q}'\boldsymbol{K}' \mid \check{G}_0(E) \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle = \\ &\delta^3(\boldsymbol{p}' - \boldsymbol{p})\delta^3(\boldsymbol{q}' - \boldsymbol{q})\delta^3(\boldsymbol{K}' - \boldsymbol{K})\frac{1}{E - \frac{\boldsymbol{p}^2}{m} - \frac{3\boldsymbol{q}^2}{4m} - \frac{\boldsymbol{K}^2}{6m}}. \end{aligned} \tag{4.99}$$

At this point we would like to again suggest that the reader skip to the final parts of this thesis (Appendixes B and C). The simple tutorials contained in those chapters are aimed to familiarize the reader with the *util1N2N3N.m* and *FunctionArray.m Mathematica*® [38] packages that are supplied with this thesis. The packages can be used to produce working FORTRAN implementations of the building blocks of the calculations that are the subject of the remainder of this thesis and the example notebooks in

PROGRAMS/

Chapter 5

Decomposition of the 2N potential operator

It can be shown [48] that the isospin projected, momentum space matrix elements of the 2N potential (which are still operators in the 2N spin space) can be written as a linear combination of six operators $\check{w}_i(\boldsymbol{p}', \boldsymbol{p})$ and scalar functions $v_i^{tm_t}$:

$$\left[\langle \boldsymbol{p}' \mid \check{V} \mid \boldsymbol{p}\rangle\right]^{2\text{Nspin}} = \sum_{i=1}^{6} v_i^{tm_t}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) \left[\check{w}_i(\boldsymbol{p}', \boldsymbol{p})\right]^{2\text{Nspin}}, \tag{5.1}$$

where the $\check{w}_i$ operators are defined as follows:

$$\begin{aligned}
\check{w}_1(\boldsymbol{p}', \boldsymbol{p}) &= \check{1}, \\
\check{w}_2(\boldsymbol{p}', \boldsymbol{p}) &= \check{\boldsymbol{\sigma}}(1) \cdot \check{\boldsymbol{\sigma}}(2), \\
\check{w}_3(\boldsymbol{p}', \boldsymbol{p}) &= i(\check{\boldsymbol{\sigma}}(1) + \check{\boldsymbol{\sigma}}(2)) \cdot (\hat{\boldsymbol{p}} \times \hat{\boldsymbol{p}}'), \\
\check{w}_4(\boldsymbol{p}', \boldsymbol{p}) &= \check{\boldsymbol{\sigma}}(1) \cdot (\hat{\boldsymbol{p}} \times \hat{\boldsymbol{p}}')\, \check{\boldsymbol{\sigma}}(2) \cdot (\hat{\boldsymbol{p}} \times \hat{\boldsymbol{p}}'), \\
\check{w}_5(\boldsymbol{p}', \boldsymbol{p}) &= \check{\boldsymbol{\sigma}}(1) \cdot (\hat{\boldsymbol{p}}' + \hat{\boldsymbol{p}})\, \check{\boldsymbol{\sigma}}(2) \cdot (\hat{\boldsymbol{p}}' + \hat{\boldsymbol{p}}), \\
\check{w}_6(\boldsymbol{p}', \boldsymbol{p}) &= \check{\boldsymbol{\sigma}}(1) \cdot (\hat{\boldsymbol{p}}' - \hat{\boldsymbol{p}})\, \check{\boldsymbol{\sigma}}(2) \cdot (\hat{\boldsymbol{p}}' - \hat{\boldsymbol{p}})
\end{aligned} \tag{5.2}$$

and the tm_t index indicates that the transition operator matrix element is taken between states in which the individual isospins are coupled to the total isospin t with the projection m_t (the 2N potential operator does not allow isospin mixing). This decomposition (5.2) is not unique and stems from the necessity for the potential to follow the usual symmetries of pairity, time reversal and charge conjugation. The set (5.2) will be used throughout this thesis. A matrix implementation of these operators is available in *util1N2N3N.m Mathematica*$^{®}$ package.

The decomposition given in (5.1) and especially the scalar functions that fully determine the potential serve as a basic input for most of our calculations. For this reason an algorithm is sometimes needed to turn potential operators given in the form of operators into a set of six scalar functions from (5.1). The basic idea behind our method of decomposition is to turn (5.1) into a typical linear problem that can be solved using a matrix representation. In the unmodified form equation (5.1) has operators on both sides. Each operator acts in the 2N spin space and is an element of a vector space spanned by 16 operators. If the matrix representation of these 16 operators is chosen to be:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ldots, \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.3}$$

then the coordinates of any operator from (5.1) in this 16 dimensional basis can be obtained by flattening out the its matrix representation. For example, if

$$\left[\langle \boldsymbol{p}' \mid \check{V} \mid \boldsymbol{p}\rangle\right]^{\text{2Nspin}} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

then the coordinates of the potential in (5.3) are:

$$(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p).$$

With this simple observation, in order to perform the decomposition, all we need to deal with is a straightforward 16 dimensional linear problem.

5.1 Practical implementation

We will present the decomposition procedure using an example but the following method can be used with any potential in operator form of this type. We tried very hard to keep the discussion of *Mathematica*® code limited to the final chapters of the text. For the present moment, however, we will not adhere to this goal. If the reader is not familiar with *Mathematica*® and the packages that are described in the final parts of this thesis, he can feel free to skip this section and come back to it later. For our illustration we will be using the Bonn B potential from Appendix D in [15] and definitions from the *util1N2N3N.m* package. Details on this package along with a short tutorial are available in Appendix B. It is recommended that Appendix B be read before continuing to analyze our decomposition procedure.

The full Bonn B potential is a combination of expressions arising from the exchange of pseudo-scalar (ps), scalar and vector mesons. For demonstration purposes we will use the **ps** part of the potential. The matrix element in momentum space of this fragment is a spin operator in the 2N spin space and has the form:

$$\left[\langle \boldsymbol{p}' \mid \check{V}_{ps} \mid \boldsymbol{p}\rangle\right]^{\text{2Nspin}} = \frac{g_{ps}^2}{(2\pi)^3 4m^2}\sqrt{\frac{m}{E'}}\sqrt{\frac{m}{E}}\frac{F_{ps}^2[(\mathbf{p}'-\mathbf{p})^2]}{(\mathbf{p}'-\mathbf{p})^2+m_{ps}^2}\frac{\left[\check{O}_{ps}\right]^{\text{2Nspin}}}{W'W}, \tag{5.4}$$

with the only non-scalar object in this expression being $\left[\check{O}_{ps}\right]$, it is a tensor product of two spin operators acting in the spaces of the two individual particles:

$$\left[\check{O}\right]_{ps} = 4m^2 W'W \left[\check{\bar{u}}(\boldsymbol{p}')\gamma^5\check{u}(\boldsymbol{p})\right]^{\text{1Nspin}} \otimes \left[\check{\bar{u}}(-\boldsymbol{p}')\gamma^5\check{u}(-\boldsymbol{p})\right]^{\text{1Nspin}}. \tag{5.5}$$

In this context γ are the Dirac matrices. Operator (5.5) with $E = \sqrt{m^2 + \boldsymbol{p}^2}$, $E' = \sqrt{m^2 + \boldsymbol{p'}^2}$, $W = m+E$, $W' = m+E'$ and m being the mass of the nucleon will be the subject of our decomposition. Notation used here will be consistent with [49]. Note however that in our definition of $\check{u}$ the two-dimensional spinors χ_s are stripped off:

$$\check{u}(\boldsymbol{p}) = \sqrt{\frac{E+m}{2m}} \begin{pmatrix} [\check{1}]^{2\text{Nspin}} \\ \left[\frac{\boldsymbol{\sigma}\cdot\boldsymbol{p}}{2m}\right]^{2Nspin} \end{pmatrix}. \tag{5.6}$$

In this way $\check{u}$ becomes a spin operator. As usual, the adjoint operator is:

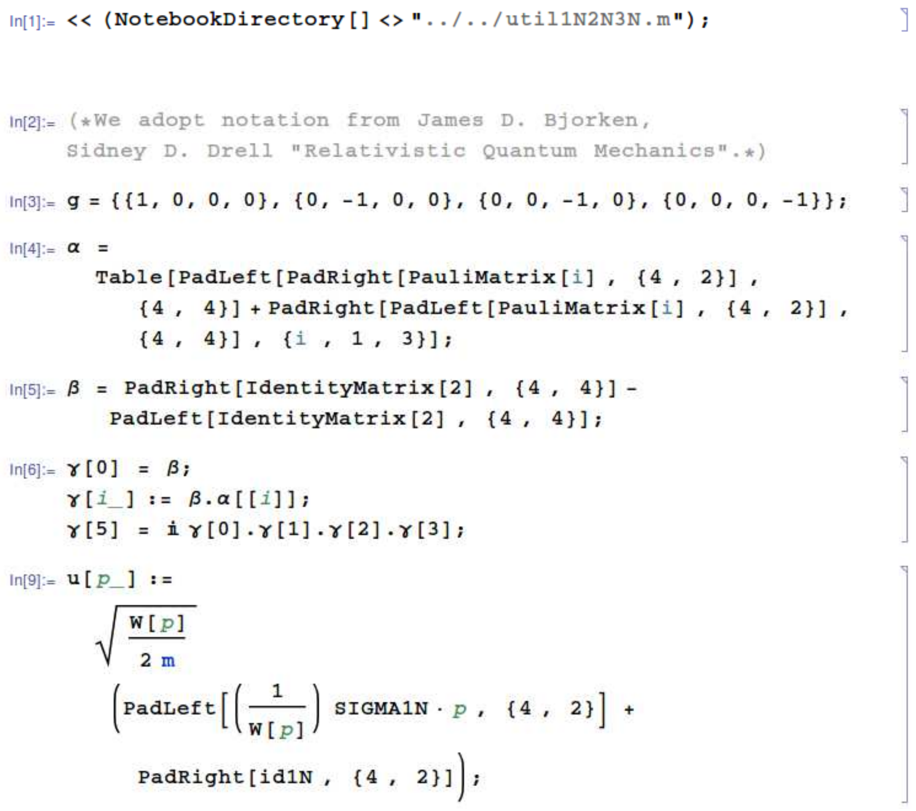
$$\bar{\check{u}} = \check{u}^{\dagger}\gamma_0. \tag{5.7}$$

Codes for the decomposition procedure are available in

PROGRAMS/decomposition/nb/decomposition.nb.

At this point in the text we strongly suggest skipping to the final Appendixes B and C. The simple tutorials that they present can be used to understand the following procedure.

At the start, basic definitions are evaluated in consistency with [49]. Most importantly the spinor `u[p_]` is defined using the `SIGMA1N` and `id1N` definitions from *util1N2N3N.m* (see Appendix B). `SIGMA1N` is a vector, with its components being Pauli matrices and `id1N` is the two dimensional identity matrix. We also give definitions for the metric g, α and γ matrices that can be used to check the properties of `u[p_]`.



```
In[1]:= << (NotebookDirectory[] <> "../../util1N2N3N.m");


In[2]:= (*We adopt notation from James D. Bjorken,
        Sidney D. Drell "Relativistic Quantum Mechanics".*)

In[3]:= g = {{1, 0, 0, 0}, {0, -1, 0, 0}, {0, 0, -1, 0}, {0, 0, 0, -1}};

In[4]:= α =
          Table[PadLeft[PadRight[PauliMatrix[i] , {4 , 2}] ,
             {4 , 4}] + PadRight[PadLeft[PauliMatrix[i] , {4 , 2}] ,
             {4 , 4}] , {i , 1 , 3}];

In[5]:= β = PadRight[IdentityMatrix[2] , {4 , 4}] -
          PadLeft[IdentityMatrix[2] , {4 , 4}];

In[6]:= γ[0] = β;
        γ[i_] := β.α[[i]];
        γ[5] = i γ[0].γ[1].γ[2].γ[3];

In[9]:= u[p_] :=
          Sqrt[W[p]/(2 m)]
           (PadLeft[(1/W[p]) SIGMA1N · p , {4 , 2}] +
             PadRight[id1N , {4 , 2}]);
```

Next additional definitions useful when using *Mathematica*® [38] simplification procedures are evaluated. It is useful to set the components of `P` , `Q` and the normalization `WP` , `WQ` as real. This is achieved by changing the `$Assumptions` variable, when *Mathematica*® attempts to evaluate `FullSimplify` on an expression it will look at this variable and use its definition in the simplification.

```
In[10]:= P = Array[p , {3}];
         Q = Array[q , {3}];
```

```
In[12]:= W[P] = WP;
         W[Q] = WQ;
```

```
In[14]:= W[-P] = WP;
         W[-Q] = WQ;
```

```
In[16]:= $Assumptions =
           (p[1] | p[2] | p[3] | q[1] | q[2] | q[3] | m | WP | WQ) ∈ Reals &&
            m > 0 && WP > 0 && WQ > 0;
```

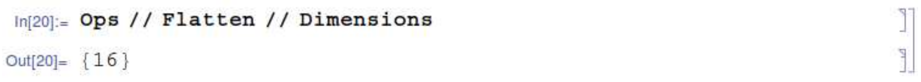
The first step of the decomposition consists of the construction of the matrix representation of $\left[\check{O}_{ps}\right]^{\mathrm{2Nspin}}$ and the set of six operators $[\check{w}_i(\boldsymbol{p}',\boldsymbol{p})]^{2Nspin}$. This is a simple task with the definitions from the *util1N2N3N.m* package as can be seen below:

```
In[17]:= Wtable =
           Table[Flatten[W2N[i , Q , P , 1 , 2]] // FullSimplify ,
            {i , 1 , 6}];
```

```
In[18]:= Ops =
           4 m^2 W[Q] W[P] (ConjugateTranspose[u[Q]].γ[0].γ[5].u[P]) ⊗
              (ConjugateTranspose[u[-Q]].γ[0].γ[5].u[-P]) //
            FullSimplify;
```

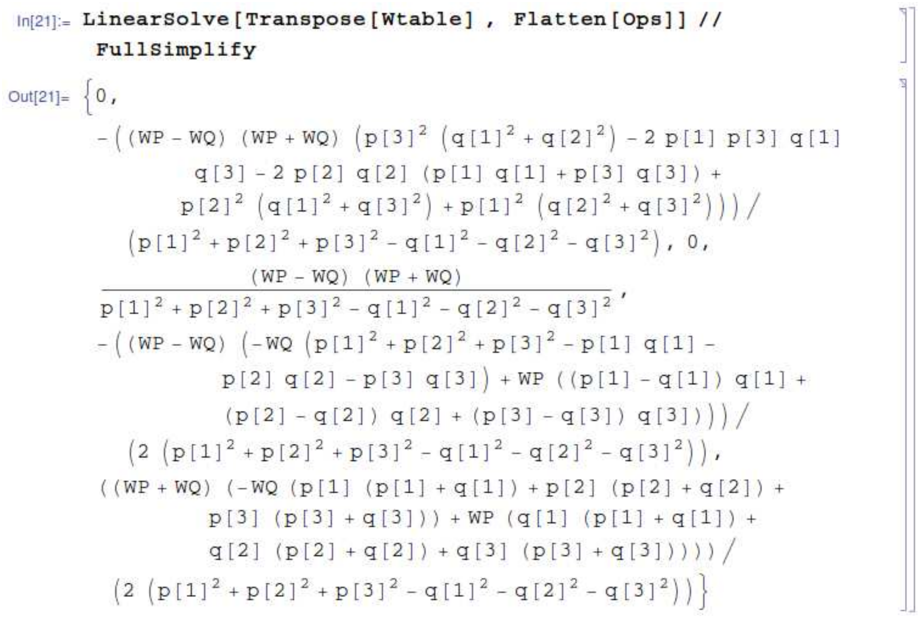
`W2N` gives the matrix representation of $[\check{w}]$ and the potential matrix representation is defined with `Ops`. The resulting matrices are flattened creating from 4×4 matrices 16 dimensional vectors. Flattening the matrices allows the construction of a linear system $[W]^{16\times 6}[x]^6 = [V]^{16}$ for the 6 scalar coefficients of (5.1) in $[x]$, here $[W]$ (symbol `Wtable`) is a matrix created from the 6 columns - flattened $[\check{w}_i(\boldsymbol{q},\boldsymbol{p})]$ matrices and $[V]$ (in *Mathematica*® `Ops//Flatten`) is the flattened $\left[\check{O}_{ps}\right]$ matrix.

```
In[19]:= Wtable // Dimensions
Out[19]= {6, 16}
```



```
In[20]:= Ops // Flatten // Dimensions
Out[20]= {16}
```

The resulting set of linear equations can be solved:



```
In[21]:= LinearSolve[Transpose[Wtable] , Flatten[Ops]] //
          FullSimplify
Out[21]= {0,
         -((WP - WQ) (WP + WQ) (p[3]^2 (q[1]^2 + q[2]^2) - 2 p[1] p[3] q[1]
                 q[3] - 2 p[2] q[2] (p[1] q[1] + p[3] q[3]) +
                p[2]^2 (q[1]^2 + q[3]^2) + p[1]^2 (q[2]^2 + q[3]^2))) /
           (p[1]^2 + p[2]^2 + p[3]^2 - q[1]^2 - q[2]^2 - q[3]^2), 0,
                      (WP - WQ) (WP + WQ)
         ─────────────────────────────────────────────────── ,
         p[1]^2 + p[2]^2 + p[3]^2 - q[1]^2 - q[2]^2 - q[3]^2
         -((WP - WQ) (-WQ (p[1]^2 + p[2]^2 + p[3]^2 - p[1] q[1] -
                  p[2] q[2] - p[3] q[3]) + WP ((p[1] - q[1]) q[1] +
                  (p[2] - q[2]) q[2] + (p[3] - q[3]) q[3]))) /
           (2 (p[1]^2 + p[2]^2 + p[3]^2 - q[1]^2 - q[2]^2 - q[3]^2)),
         ((WP + WQ) (-WQ (p[1] (p[1] + q[1]) + p[2] (p[2] + q[2]) +
                 p[3] (p[3] + q[3])) + WP (q[1] (p[1] + q[1]) +
                 q[2] (p[2] + q[2]) + q[3] (p[3] + q[3])))) /
          (2 (p[1]^2 + p[2]^2 + p[3]^2 - q[1]^2 - q[2]^2 - q[3]^2))}
```

giving, in agreement with [15], the final result. An additional advantage of using this type of procedure is the automatic test of the symmetries of the potential. If the potential does not have the appropriate symmetries (parity, time reversal) then the system of equations does not have a solution and Mathemaica will throw an error message.

The remaining parts of the Bonn B potential are decomposed in

PROGRAMS/decomposition/nb/fulldecomposition.nb.

This notebook contains definitions for all the parts of the Bonn B potential and uses the *FunctionArray.m* package to create FORTRAN code with the scalar coefficients. The scalar coefficients for the Bonn B potential from [15] are also gathered in Appendix E.

Equation (5.1) neglects the isospin degrees of freedom of the 2N system. In a more general case this decomposition has the form:

$$\left[\langle \boldsymbol{p}' \mid \check{V} \mid \boldsymbol{p}\rangle\right]^{2N} = \sum_{i=1}^{6}\sum_{\gamma=1}^{4} v_i^{\gamma}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{p}})\,[\check{w}_i^{\gamma}(\boldsymbol{p}', \boldsymbol{p})]^{2\mathrm{N}} \tag{5.8}$$

with the $[\check{w}_i^{\gamma}]^{2\mathrm{N}}$ operators acting in the full isospin - spin space of the 2N system and γ is one of the 4 possible 2N isospin states (with the isospin of the two particles coupled to a total isospin t with projection m_t):

$$\begin{aligned} \mid \gamma_1\rangle &=\mid t=0, m_t=0\rangle, \\ \mid \gamma_2\rangle &=\mid t=1, m_t=-1\rangle, \\ \mid \gamma_3\rangle &=\mid t=1, m_t=0\rangle, \\ \mid \gamma_4\rangle &=\mid t=1, m_t=1\rangle. \end{aligned} \tag{5.9}$$

The isospin part of $[\breve{w}_i^{\gamma}]^{2\mathrm{N}}$ is a cast operator from (4.96):

$$[\breve{w}_i^{\gamma_i}(\boldsymbol{p}', \boldsymbol{p})]^{2\mathrm{N}} = \left[\check{C}(\gamma_i)\right]^{2\mathrm{Nisospin}} \otimes [\breve{w}_i(\boldsymbol{p}', \boldsymbol{p})]^{2\mathrm{Nspin}} \tag{5.10}$$

The form of (5.8) allows for isospin dependence and enables us to consider each isospin case separatly ($\mid \gamma_1\rangle$ - neutron, proton; $\mid \gamma_3\rangle$ - neutron, proton; $\mid \gamma_2\rangle$ - neutron, neutron and $\mid \gamma_4\rangle$ - proton, proton).

Typically the operator form of 2N potentials is given in literature in a way that makes it easy to project out an appropriate isospin part. When this in not obvious, the decomposition procedure described earlier can still be used. This will require the use of 16×16 dimensional matrices and 256 dimensional flattened vectors instead of the earlier 4×4 matrices and 16 dimensional flattened vectors. Due to the much higher order of the linear equations it can be expected that the time of the computation in *Mathematica*® and memory requirements will grow substantially.

Chapter 6

Calculation of the 2N bound state in three dimensions

Additional constraints (taking into account, along with other properties, the positive parity of the deuteron) on the deuteron wave function can be used to turn equation (4.55) into a form that can directly be used in numerical calculations. The deuteron bound state wave function in the CM (center of mass) reference frame can be written in an operator form (see for example [22]):

$$
\begin{aligned}
&\mid \Psi(m_d)\rangle = \\
&= \int \mathrm{d}^3\boldsymbol{p} \left(\phi_1(|\boldsymbol{p}|) \left[\check{1}\right]^{2\text{Nspin}} + \phi_2(|\boldsymbol{p}|) \left[\check{\boldsymbol{\sigma}}(1)\cdot\boldsymbol{p}\check{\boldsymbol{\sigma}}(2)\cdot\boldsymbol{p} - \frac{1}{3}\boldsymbol{p}^2\check{1} \right]^{2\text{Nspin}} \right) \\
&\qquad \mid \boldsymbol{p}\boldsymbol{K}=0\rangle \otimes \left[\mid 1 m_d\rangle\right]^{2\text{Nspin}} \\
&\equiv \int \mathrm{d}^3\boldsymbol{p} \sum_{k=1}^{2} \phi_k(|\boldsymbol{p}|) \left[\check{b}_k(\boldsymbol{p})\right] \mid \boldsymbol{p}\boldsymbol{K}=0\rangle \otimes \left[\mid 1 m_d\rangle\right], \qquad (6.1)
\end{aligned}
$$

where $\mid 1m_d\rangle$ is a two-nucleon spin state in which the spins of the two nucleons are coupled to the total spin 1 with a projection m_d, $\boldsymbol{p}$ is the relative momentum of the two nucleons (the total momentum $\boldsymbol{K}$ is zero), $\phi_1(|\boldsymbol{p}|)$, $\phi_2(|\boldsymbol{p}|)$ are scalar functions of the relative deuteron momentum magnitude and $\left[\check{b}_k(\boldsymbol{p})\right]$ are operators acting in the spin space of the two nucleons. In the last line of equation (6.1) we introduce the definitions for spin operators $\check{b}_1(\boldsymbol{p})$ and $\check{b}_2(\boldsymbol{p})$. The isospin state of this system is 0 and in the following discussion this information will be suppressed. All operators acting in the spin space of the two particles together with the two particle spin states in (6.1) can easily be implemented as 4×4 matrices and 4 dimensional vectors with the use of the *util1N2N3N.m Mathematica*® package (appendix B gives a description of this package). Further details on notebooks that are supplied with this thesis and that create FORTRAN code that can be used in the numerical realization of deuteron calculations can be found in Appendix D.

Finding a solution to (4.55) for the deuteron bound state is equivalent to calculating the form of ϕ_1, ϕ_2 functions. In order to arrive at the equations for these scalar functions it is necessary to remove the spin dependencies with the use of (5.1). Equation (4.55) rewritten using (6.1) and (5.1) has the following form:

$$\sum_{k'=1}^{2} \phi_{k'}(|\boldsymbol{p}|) \left[\check{b}_{k'}(\boldsymbol{p})\right] [|\, 1m_d\rangle] = \int \mathrm{d}^3\boldsymbol{p}' \sum_{k''=1}^{2} \sum_{j=1}^{6} \frac{1}{E_d - \frac{p^2}{m}}$$
$$\phi_{k''}(|\boldsymbol{p}'|) v_j^{00}(|\boldsymbol{p}|, |\boldsymbol{p}'|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{p}}') \left[\check{w}_j(\boldsymbol{p}, \boldsymbol{p}')\right] \left[\check{b}_{k''}(\boldsymbol{p}')\right] [|\, 1m_d\rangle] \tag{6.2}$$

where E_d is the deuteron binding energy (because the binding energy is negative the limit of $\epsilon \to 0^+$ can be safely carried out). The spin dependency in (6.2) can be removed by projecting from the left with $[\langle 1m_d\,|] \left[\check{b}_k(\boldsymbol{p})\right]$ and summing over m_d. With the introduction of two new functions:

$$A^d_{k'k}(\boldsymbol{p}) = \sum_{m_d=-1}^{1} [\langle 1m_d\,|] \left[\check{b}_{k'}(\boldsymbol{p})\right] \left[\check{b}_k(\boldsymbol{p})\right] [|\, 1m_d\rangle] \tag{6.3}$$

and

$$B^d_{k'jk}(\boldsymbol{p}, \boldsymbol{p}') = \sum_{m_d=-1}^{1} [\langle 1m_d\,|] \left[\check{b}_{k'}(\boldsymbol{p})\right] [\check{w}_j(\boldsymbol{p}, \boldsymbol{p}')] \left[\check{b}_k(\boldsymbol{p}')\right] [|\, 1m_d\rangle]\,, \tag{6.4}$$

equation (6.2) turns into (see also [15]):

$$\sum_{k'=1}^{2} A^d_{kk'}(\boldsymbol{p}) \phi_{k'}(|\boldsymbol{p}|) =$$
$$\frac{1}{E_d - \frac{\boldsymbol{p}^2}{m}} \int \mathrm{d}^3\boldsymbol{p}' \sum_{j=1}^{6} v_j^{00}(|\boldsymbol{p}|, |\boldsymbol{p}'|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{p}}')$$
$$\sum_{k''=1}^{2} B^d_{kjk''}(\boldsymbol{p}, \boldsymbol{p}') \phi_{k''}(|\boldsymbol{p}'|)\ . \tag{6.5}$$

This can be rewritten in a more compact form:

$$\phi_q(|\boldsymbol{p}|) =$$
$$\frac{1}{E_d - \frac{\boldsymbol{p}^2}{m}} \int \mathrm{d}^3\boldsymbol{p}' \sum_{j=1}^{6} v_j^{00}(\boldsymbol{p}, \boldsymbol{p}') \sum_{k''=1}^{2}$$
$$\left(\sum_k \left(A^d(\boldsymbol{p})\right)^{-1}_{qk} B^d_{kjk''}(\boldsymbol{p}, \boldsymbol{p}') \right) \phi_{k''}(|\boldsymbol{p}'|) \tag{6.6}$$

where $\left(A^d(\boldsymbol{p})\right)^{-1}_{qk}$ is defined such that:

$$\sum_k \left(A^d(\boldsymbol{p})\right)^{-1}_{qk} A^d_{kk'}(\boldsymbol{p}) \equiv \delta_{qk'} \tag{6.7}$$

or in the operator form:

$$\boxed{\phi_q(|\boldsymbol{p}|) = ((\check{A}^d)^{-1} \check{B}(E_d)\phi)_q(|\boldsymbol{p}|) \equiv (\check{K}^d(E_d)\phi)_q(|\boldsymbol{p}|).} \tag{6.8}$$

Expressions for $A^d_{k'k}(\boldsymbol{p})$, $B^d_{k'jk}(\boldsymbol{p},\boldsymbol{p'})$ and $\left(A^d(\boldsymbol{p})\right)^{-1}_{qk}$ from equations (6.3),(6.4) and (6.7) have a FORTRAN implementation that is supplied with this thesis and described in section D.1.

Relation (6.8) is a linear eigenequation (in the space of scalar functions of one real variable) for $\phi_1(|\boldsymbol{p}|)$ and $\phi_2(|\boldsymbol{p}|)$ of the type discussed in (4.56) with the operator $\check{A}(E_d) = \check{K}^d(E_d)$ defined by the way it acts on the scalar functions:

$$\left(\check{K}^d(E_d)\phi\right)_q(|\boldsymbol{p}|) = \frac{1}{E_d - \frac{\boldsymbol{p}^2}{m}} \int \mathrm{d}^3\boldsymbol{p'} \sum_{j=1}^{6} v_j^{00}(\boldsymbol{p},\boldsymbol{p'}) \sum_{k''=1}^{2} \left(\sum_k \left(A^d(\boldsymbol{p})\right)^{-1}_{qk} B^d_{kjk''}(\boldsymbol{p},\boldsymbol{p'}) \right) \phi_{k''}(|\boldsymbol{p'}|), \tag{6.9}$$

The realization that in the above equations we use only linear operators is important because Krylov subspace methods, such as the Arnoldi iteration procedure described in section A.2 can be used. For larger problems this will lead to a dramatic decrease in the dimension of the problem. If we were to discretize ϕ_1, ϕ_2 over a grid of, say, 100 points $|\boldsymbol{p}|$, then in order to solve (6.2) we would have to work with 200 dimensional vectors. After the Arnoldi procedure we can reduce this dimension to, say 40. Clearly, in the case of the deuteron bound state, this reduction is not necessary. The 200 linear equations can easily be solved using a modern personal computer and we used this calculation as a test of the Arnoldi iteration scheme. In the case of the 3N bound state, with the dimension of the discretized function-vectors defining the bound state of the order 10^6 a reduction of the problem to, say 40 dimensions is very valuable and allows the use of classical algorithms for linear algebra problems (gathered for example in the LAPACK library). Another advantage of using Krylov methods is that an explicit matrix representation of linear operators is not necessary. For the deuteron and the transition operator such a representation is available, for the three-nucleon bound state the possibility to construct the implementation of the linear operators directly from sums and integrals is crucial as the task of finding the matrix representation is very difficult.

As mentioned before, one way to approach

$$\boxed{\phi = \check{K}^d(E_d)\phi} \tag{6.10}$$

is to construct a Krylov subspace projection of $\check{K}^d$. Using the Arnoldi algorithm described in A.2 we can reduce (6.6) to a (say) 40×40 matrix eigenequation that can be solved using classical methods of linear algebra. Numerous equations (in the from from equation (4.56) - $\check{K}^d(E)\phi = \lambda\phi$) are constructed for different values of E, when a solution with λ sufficiently close to 1 is found, E is an approximation of the bound state energy.

Additional speed up of the calculation can be achieved by considering the following parametrization of vectors in $\check{K}^d(E_d)$ from equation (6.9):

$$\begin{aligned} \boldsymbol{p} &= |\boldsymbol{p}|(0,0,1) \\ \boldsymbol{p'} &= |\boldsymbol{p'}|(cos(\Phi)\sqrt{1-x^2}, sin(\Phi)\sqrt{1-x^2}, x) \end{aligned} \tag{6.11}$$

with $\Phi \in (0, 2\pi)$ and $x \in (-1, 1)$. Using (6.11) the integral over $\boldsymbol{p}'$ turns into:

$$\int \mathrm{d}^3\boldsymbol{p}' \to \int_0^\infty \mathrm{d}|\boldsymbol{p}'||\boldsymbol{p}'|^2 \int_0^{2\pi} \mathrm{d}\Phi \int_{-1}^{1} \mathrm{d}x. \tag{6.12}$$

The integral over Φ can be done immediately as there is no Φ dependence in the integrated function - all arguments depend on $\boldsymbol{p} \cdot \boldsymbol{p}' = x$. Furthermore the integral over x can be done beforehand and stored in a table - $\phi_{k''}(|\boldsymbol{p}'|)$ does not depend on x.

In our calculations we use a chiral NNLO 2N potential [37] with $\Lambda = 600$[MeV] and $\tilde{\Lambda} = 700$[MeV] , the operator form of this potential can be found in Appendix C of [15]. Figure 6.1 contains a plot of the eigenvalues closest to 1 for different test values of E_d. It can be concluded from the calculations that the bound state energy for the deuteron system is -2.2001[MeV]. Figures 6.2 contain plots of scalar functions $\phi_1(|\boldsymbol{p}|)$ and $\phi_2(|\boldsymbol{p}|)$ for different values of the relative momentum magnitude.

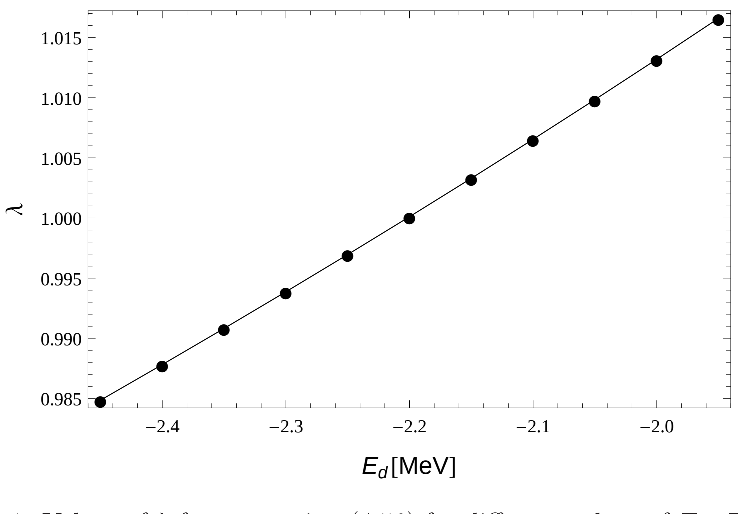


Figure 6.1: Values of λ from equation (4.56) for different values of E_d. The calculations were performed for the chiral NNLO potential [37] with $\Lambda = 600$[MeV] and $\tilde{\Lambda} = 700$[MeV]. The circles mark points used in the calculation, the solid line is interpolated from these points.

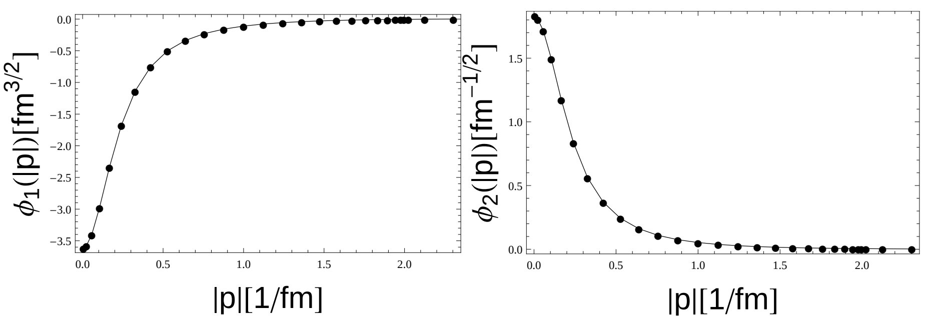


Figure 6.2: ϕ_1, ϕ_2 for $E_d = -2.2001$[MeV]. The calculations were performed for the chiral NNLO potential [37] with $\Lambda = 600$[MeV] and $\tilde{\Lambda} = 700$[MeV]. The circles mark points used in the calculation, the solid line is interpolated from these points.

We also performed calculations with the older Bonn B one-boson-exchange potential from [10]. The results differ slightly from the NNLO case, the calculated energy is lower at -2.2242[MeV]. The range of the potential is larger but the resulting scalar functions are very similar to the NNLO case as can be seen in Figure 6.3.

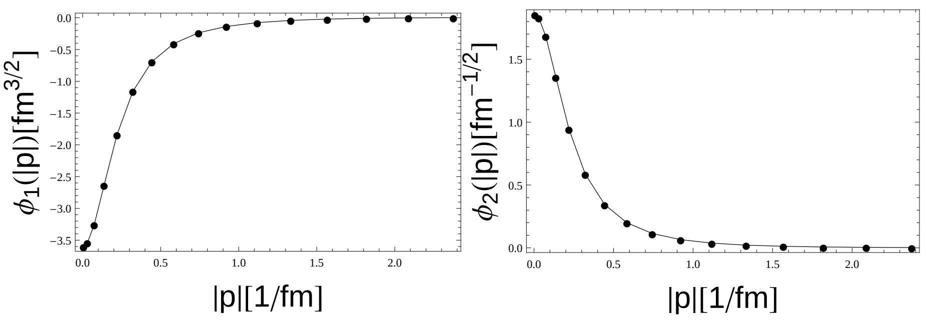


Figure 6.3: ϕ_1, ϕ_2 for $E_d = -2.2242$[MeV]. The calculations were performed for the one-boson-exchange Bonn B potential [10]. The circles mark points used in the calculation, the solid line is interpolated from these points.

The scalar functions ϕ_1, ϕ_2 are directly related to the s (ψ_0) and d (ψ_2) wave component of the deuteron wave function. The relation is [22]:

$$\psi_0(|\boldsymbol{p}|) = \sqrt{4\pi}\phi_1(|\boldsymbol{p}|), \tag{6.13}$$

$$\psi_2(|\boldsymbol{p}|) = \frac{4\sqrt{2}|\boldsymbol{p}|^2}{3}\phi_2(|\boldsymbol{p}|). \tag{6.14}$$

Using this information in Figures 6.4 and 6.5 we compare the three dimensional approach that was the subject of this chapter with partial wave results. A very good agreement is observed between both methods.

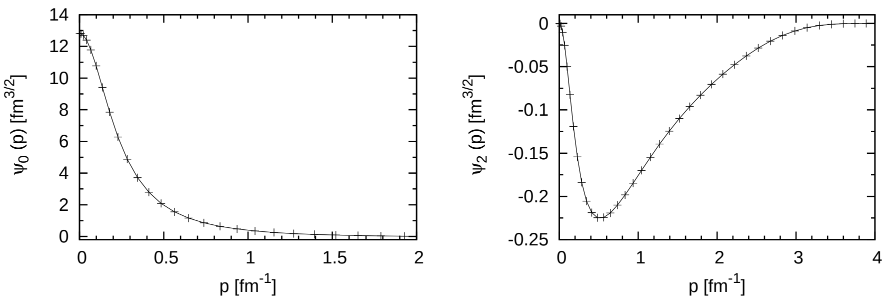


Figure 6.4: The s (left) and d (right) wave component of the deuteron wave function as a function of the relative momentum. Crosses are results obtained using the three-dimensional approach and the solid line represents results obtained using the standard partial wave approach. For this case the chiral NNLO potential [37] was used. Results reprinted from our paper [15].

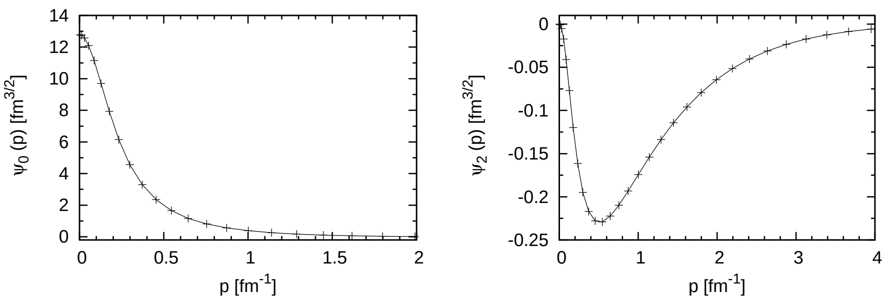


Figure 6.5: The s (left) and d (right) wave component of the deuteron wave function as a function of the relative momentum. Crosses are results obtained using the three-dimensional approach and the solid line represents results obtained using the standard partial wave approach. For this case the Bonn B [10] was used. Results reprinted from our paper [15].

Chapter 7

Calculation of the 2N transition operator in three dimensions

The transition operator $\check{t}$ is, as was shown in section 4.2, an important element of nucleon - nucleon scattering calculations. All relevant observables in the 2N scattering process can be linked to the transition operator (see for example [45]). In some cases, for example when constructing 3N bound state calculations (3N bound state calculations that involve the transition operator are discussed in [23, 35]) there is a need to calculate the 3N matrix elements of the transition operator acting within a 2N system and assuming only 2N interactions (the third particle being a "spectator"). The 3N projected transition operator can be related to the 2N subsystem $\check{t}$ operator. This relation is not hard to work out (see for example [46]), the matrix element in momentum space (operator in isospin - spin space) will have the form in equation (4.52):

$$\langle \boldsymbol{p}'\boldsymbol{q}'\boldsymbol{K}' \mid \check{t}_{3\mathrm{N}}(E) \mid \boldsymbol{p}\boldsymbol{q}\boldsymbol{K}\rangle = \delta(\boldsymbol{q}'-\boldsymbol{q})\delta(\boldsymbol{K}'-\boldsymbol{K})\langle \boldsymbol{p}' \mid \check{t}(E-\frac{3}{4m}\boldsymbol{q}^2-\frac{1}{6m}\check{K}^2) \mid \boldsymbol{p}\rangle, \tag{7.1}$$

where $\check{t}(E-\frac{3}{4m}\boldsymbol{q}^2-\frac{1}{6m}\check{K}^2)$ is the 2N transition operator calculated for the energy of the two particle subsystem in CM.

Looking at the form of the Lippman Schwinger equation (4.50) we can expect the 2N transition operator to follow a decomposition similar to (5.8):

$$\left[\langle \boldsymbol{p}' \mid \check{t}(E) \mid \boldsymbol{p}\rangle\right]^{2\mathrm{N}} = \sum_{\gamma=1}^{4}\sum_{i=1}^{6} t_i^{\gamma}(E,|\boldsymbol{p}'|,|\boldsymbol{p}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{p}})\left[\check{C}(\gamma)\otimes\check{w}_i(\boldsymbol{p}',\boldsymbol{p})\right]^{2\mathrm{N}}, \tag{7.2}$$

where we used the projection $\check{C}(\gamma)$ operator and isospin states ($\gamma = 1,2,3,4$) from (4.96) and assumed that the transition operator does not mix isospin states. In this form the transition operator is fully determined by the 6×4 scalar functions

$$t_i^{\gamma}(E,|\boldsymbol{p}'|,|\boldsymbol{p}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{p}})$$

and the Lippman-Schwinger equation can be written in full detail:

$$
\begin{aligned}
&\sum_{j=1}^{6} t_j^{\gamma}(E, |\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) \left[\breve{w}_j(\boldsymbol{p}', \boldsymbol{p})\right]^{\text{2Nspin}} = \\
&\sum_{j=1}^{6} v_j^{\gamma}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) \left[\breve{w}_j(\boldsymbol{p}', \boldsymbol{p})\right]^{\text{2Nspin}} + \\
&\int \mathrm{d}^3\boldsymbol{p}'' \sum_{j=1}^{6}\sum_{j'=1}^{6} \frac{1}{E - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} v_j^{\gamma}(|\boldsymbol{p}'|, |\boldsymbol{p}''|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}'') t_{j'}^{\gamma}(E, |\boldsymbol{p}''|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}'' \cdot \hat{\boldsymbol{p}}) \\
&\left[\breve{w}_j(\boldsymbol{p}', \boldsymbol{p}'')\right]^{\text{2Nspin}} \left[\breve{w}_{j'}(\boldsymbol{p}'', \boldsymbol{p})\right]^{\text{2Nspin}}
\end{aligned}
\tag{7.3}
$$

where we inserted the full form of the operators in 2N space with the additional assumption about the total momentum of the particles $\boldsymbol{K} = 0$. Additionally we use the property $\check{C}(\gamma)\check{C}(\gamma') = \delta_{\gamma\gamma'}\check{C}(\gamma)$ to separate one isospin case. Removing spin dependencies can be achieved by acting from the left with $[\breve{w}_k(\boldsymbol{p}', \boldsymbol{p})]$ and taking a trace over all 2N spin states. With the introduction of two new functions:

$$
A_{kj}(\boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}', \boldsymbol{p}) = \sum_{s=0}^{1} \sum_{m_s=-s}^{s} \left[\langle s m_s \mid\right] \left[\breve{w}_k(\boldsymbol{p}', \boldsymbol{p})\right] \left[\breve{w}_j(\boldsymbol{p}', \boldsymbol{p})\right] \left[\mid s m_s\rangle\right] \tag{7.4}
$$

$$
\begin{aligned}
&B_{kjj'}(\boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}', \boldsymbol{p}''; \boldsymbol{p}'', \boldsymbol{p}) = \\
&\sum_{s=0}^{1} \sum_{m_s=-s}^{s} \left[\langle s m_s \mid\right] \left[\breve{w}_k(\boldsymbol{p}', \boldsymbol{p})\right] \left[\breve{w}_j(\boldsymbol{p}', \boldsymbol{p}'')\right] \left[\breve{w}_{j'}(\boldsymbol{p}'', \boldsymbol{p})\right] \left[\mid s m_s\rangle\right]
\end{aligned}
\tag{7.5}
$$

(state $\mid s m_s\rangle$ has the spin of both particles coupled to a total spin s with projection m_s, both functions have a simple implementation that can be created using the *util1N2N3N.m* package and is discussed in section D.3) equation (7.3) takes on the form:

$$
\begin{aligned}
&\sum_{j=1}^{6} A_{kj}(\boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}', \boldsymbol{p}) t_j^{\gamma}(E, |\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) = \\
&\sum_{j=1}^{6} A_{kj}(\boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}', \boldsymbol{p}) v_j^{\gamma}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) + \\
&\int \mathrm{d}^3\boldsymbol{p}'' \sum_{j=1}^{6}\sum_{j'=1}^{6} \frac{1}{E - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} v_j^{\gamma}(|\boldsymbol{p}'|, |\boldsymbol{p}''|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}'') t_{j'}^{\gamma}(E, |\boldsymbol{p}''|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}'' \cdot \hat{\boldsymbol{p}}) \\
&B_{kjj'}(\boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}', \boldsymbol{p}''; \boldsymbol{p}'', \boldsymbol{p}).
\end{aligned}
\tag{7.6}
$$

Additional simplification can be achieved by using the rotation invariance of expressions in (7.6). This is utilized by limiting our calculations to the following class of vectors:

$$
\begin{aligned}
\boldsymbol{p} &= |\boldsymbol{p}|(0, 0, 1) \\
\boldsymbol{p}' &= |\boldsymbol{p}'|(\sqrt{1 - x'^2}, 0, x') \\
\boldsymbol{p}'' &= |\boldsymbol{p}''|(\sqrt{1 - x''^2}\cos\phi'', \sqrt{1 - x''^2}\sin\phi'', x'')
\end{aligned}
\tag{7.7}
$$

(with $x' \in (-1,1)$, $x'' \in (-1,1)$, $\phi'' \in (0,2\pi)$) and turns (7.6) into:

$$
\begin{aligned}
&\sum_{j=1}^{6} A_{kj}(|\boldsymbol{p}'|,|\boldsymbol{p}|,x') t_j^{\gamma}(E,|\boldsymbol{p}'|,|\boldsymbol{p}|,x') = \\
&\sum_{j=1}^{6} A_{kj}(|\boldsymbol{p}'|,|\boldsymbol{p}|,x') v_j^{\gamma}(|\boldsymbol{p}'|,|\boldsymbol{p}|,x') + \int_0^{+\infty} \mathrm{d}|\boldsymbol{p}''| \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \\
&\sum_{j=1}^{6}\sum_{j'=1}^{6} \frac{|\boldsymbol{p}''|^2}{E - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} \\
&v_j^{\gamma}(|\boldsymbol{p}'|,|\boldsymbol{p}''|,\sqrt{1-x'^2}\sqrt{1-x''^2}\cos\phi'' + x'x'') t_{j'}^{\gamma}(E,|\boldsymbol{p}''|,|\boldsymbol{p}|,x'') \\
&B_{kjj'}(|\boldsymbol{p}'|,|\boldsymbol{p}|,x',|\boldsymbol{p}''|,x'',\phi''). \qquad (7.8)
\end{aligned}
$$

Analyzing the form of (7.8) it can be concluded that for any value of E, γ, $|\boldsymbol{p}|$ there exists an independent integral equation for $t_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x')$. We will use the curly brackets to mark independent variables. With this notation $t_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x')$ should be treated as a function of k, $|\boldsymbol{p}'|$, x' and the other quantities E, γ, $|\boldsymbol{p}|$ are to be used to label for the equation that the function satisfies. It is also worth noting that the integral over ϕ'' and sum over j can be done beforehand in the numerical realization and stored in a table for further use as $t_{j'}^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}''|,_{\{|\boldsymbol{p}|\}},x'')$ does not depend on ϕ'' or j.

Functions t and v in (7.8) are vectors and are acted upon with linear operators. The left hand side of equation (7.8) is the definition of linear operator $\breve{A}$:

$$
\left(\breve{A}t\right)_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x') = \sum_{j=1}^{6} A_{kj}(|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x') t_j^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x'), \qquad (7.9)
$$

and the right hand side of this equation is the definition of operator $\breve{B}$:

$$
\begin{aligned}
&\left(\breve{B}t\right)_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x') = \\
&\int_0^{+\infty} \mathrm{d}|\boldsymbol{p}''| \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \sum_{j=1}^{6}\sum_{j'=1}^{6} \frac{|\boldsymbol{p}''|^2}{_{\{E\}} - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} \\
&v_j^{\{\gamma\}}(|\boldsymbol{p}'|,|\boldsymbol{p}''|,\sqrt{1-x'^2}\sqrt{1-x''^2}\cos\phi'' + x'x'') \\
&B_{kjj'}(|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x',|\boldsymbol{p}''|,x'',\phi'') t_{j'}^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}''|,_{\{|\boldsymbol{p}|\}},x''). \qquad (7.10)
\end{aligned}
$$

In this notation $\breve{A}t$ is the result of applying operator $\breve{A}$ on the scalar function t and $\left(\breve{A}t\right)_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x')$ means that we take the resulting function value for arguments $_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x')$. This is a natural way of writing operators if we keep in mind that they will be implemented in FORTRAN where taking the value of a function for a set of arguments is typically replaced by taking the value of an array for a set of indexes.

The inverse of operator $\breve{A}$ can also be constructed such that:

$$
\left(\breve{A}^{-1}\breve{A}t\right)_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x') = t_k^{\{\gamma\}}(_{\{E\}},|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x'), \qquad (7.11)
$$

$$
\left(\breve{A}^{-1}\breve{A}v\right)_k^{\{\gamma\}}(|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x') = v_k^{\{\gamma\}}(|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x') \qquad (7.12)
$$

with:

$$\left(\check{A}^{-1}t\right)_k^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') =$$
$$\sum_{r=1}^{6} A_{kr}^{-1}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') t_r^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') \tag{7.13}$$

and

$$\sum_{r=1}^{6} A_{kr}^{-1}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') A_{ri}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') = \delta_{ki}. \tag{7.14}$$

This can be combined with the $\check{B}$ operator:

$$\begin{aligned}
&\left(\check{A}^{-1}\check{B}t\right)_k^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') \equiv \left(\check{\mathcal{B}}t\right)_k^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') = \\
&\int_0^{+\infty} \mathrm{d}|\boldsymbol{p}''| \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \sum_{j=1}^{6}\sum_{j'=1}^{6} \frac{|\boldsymbol{p}''|^2}{{\scriptstyle \{E\}} - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} \\
&v_j^{\{\gamma\}}\ (|\boldsymbol{p}'|, |\boldsymbol{p}''|, \sqrt{1-x'^2}\sqrt{1-x''^2}\cos\phi'' + x'x'') \\
&\left(\sum_{r=1}^{6} A_{kr}^{-1}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') B_{rjj'}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x', |\boldsymbol{p}''|, x'', \phi'')\right) \\
&t_{j'}^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}''|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x'') \\
&\equiv \int_0^{+\infty} \mathrm{d}|\boldsymbol{p}''| \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \sum_{j=1}^{6}\sum_{j'=1}^{6} \frac{|\boldsymbol{p}''|^2}{{\scriptstyle \{E\}} - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} \\
&v_j^{\{\gamma\}}\ (|\boldsymbol{p}'|, |\boldsymbol{p}''|, \sqrt{1-x'^2}\sqrt{1-x''^2}\cos\phi'' + x'x'') \\
&\mathcal{B}_{kjj'}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x', |\boldsymbol{p}''|, x'', \phi'') \\
&t_{j'}^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}''|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x'')
\end{aligned} \tag{7.15}$$

and it turns out that the analytical computation of the sum over the products of

$$\sum_{r=1}^{6} A_{kr}^{-1}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') B_{rjj'}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x', |\boldsymbol{p}''|, x'', \phi'') \equiv \mathcal{B}_{kjj'}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x', |\boldsymbol{p}''|, x'', \phi'') \tag{7.16}$$

and combining the action of $\check{A}^{-1}$ and $\check{B}$ in $\check{\mathcal{B}}$ as opposed to the numerical computation of $\check{B}t$ and then $(\check{A})^{-1}\check{B}t$ - leads to a more stable numerical performance (the 36 A^{-1} coefficients are gathered in Appendix E.4).

Finally the Lippman-Schwinger equation for the transition operator can be written in compact form:

$$\boxed{t = v + (\check{A})^{-1}\check{B}t \equiv v + \check{\mathcal{B}}t} \tag{7.17}$$

or

$$t_k^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') =$$
$$v_k^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x') + \left(\check{\mathcal{B}}t\right)_k^{\{\gamma\}}\ ({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x'). \tag{7.18}$$

With independent equations for different values of γ, E, $|\boldsymbol{p}|$, the calculation can be split into a number of smaller problems. Even with this information the size of linear operators in the numerical realization of the calculations is large. In section A.1 we give a brief description of Krylev methods that allow for a drastic reduction of the size of the problem. In section A.2 we will give a more detailed description of the Arnoldi algorithm that was used in our calculations.

7.1 Positive energies

Many ways of solving the resulting linear equations have been explored. Of particular difficulty is the calculation of the transition operator for energies $E < 0$, it can be worked out that the 2N transition operator will have singular behavior around the deuteron binding energy (see for ex. [46]). In the positive energy range we have to deal with the singularity in the denominator of the propagator (7.15), this can be fixed by using the identity:

$$\begin{aligned}\int_0^{\bar{y}} \mathrm{d}y \frac{y^2}{y_0^2 - y^2 + i\epsilon} f(y) = \\ \int_0^{\bar{y}} \mathrm{d}y \frac{y^2 f(y) - y_0^2 f(y_0)}{y_0^2 - y^2} + \frac{1}{2} y_0 f(y_0) \left(\ln\left(\frac{\bar{y} + y_0}{\bar{y} - y_0} \right) - i\pi \right).\end{aligned} \tag{7.19}$$

The validity of this relation can easily be checked by choosing an appropriate contour for complex integration. For the limit $\epsilon \to 0^+$ in a region close to y_0 the integration should follow an arc in the lower half of the complex plane. The limit then corresponds to taking the radius of this arc equal to 0. Equation (7.19) is used to simplify our calculations (see for example [15]). In our case we introduce a cut-off value for the $|\boldsymbol{p}''|$ integral, $\bar{p}$, and substitute:

$$\begin{aligned}\bar{y} &\to \bar{p}, \\ y_0 &\to p_0(\{E\}\ ), \\ y &\to |\boldsymbol{p}''|, \\ f(y) &\to (\check{f}(|\boldsymbol{p}''|)t)_k^{\{\gamma\}}\ (\{E\}\ , |\boldsymbol{p}'|, \{|\boldsymbol{p}|\}\ , x'),\end{aligned} \tag{7.20}$$

with m being the nucleon mass, $\{E\} \equiv \frac{p_0^2(\{E\}\ )}{m}$ and the $\check{f}$ operator defined as:

$$\begin{aligned}(\check{f}(|\boldsymbol{p}''|)t)_k^{\{\gamma\}}\ (\{E\}\ , |\boldsymbol{p}'|, \{|\boldsymbol{p}|\}\ , x') = \\ m \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \sum_{j=1}^{6} \sum_{j'=1}^{6} \\ v_j^{\{\gamma\}}\ (|\boldsymbol{p}'|, |\boldsymbol{p}''|, \sqrt{1 - x'^2}\sqrt{1 - x''^2} \cos\phi'' + x'x'') \\ \mathcal{B}_{kjj'}(|\boldsymbol{p}'|, \{|\boldsymbol{p}|\}\ , x', |\boldsymbol{p}''|, x'', \phi'') \\ t_{j'}^{\{\gamma\}}\ (\{E\}\ , |\boldsymbol{p}''|, \{|\boldsymbol{p}|\}\ , x'').\end{aligned} \tag{7.21}$$

With (7.19), (7.20) and (7.21) the $\check{\mathcal{B}}$ operator for positive energies, can be written as:

$$
\begin{aligned}
\left(\check{\mathcal{B}}t\right)_k^{\{\gamma\}} ({\scriptstyle \{E\}}, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}, x') &= \\
\int_0^{\bar{p}} \mathrm{d}|\boldsymbol{p}''| \frac{1}{p_0^2({\scriptstyle \{E\}}) - |\boldsymbol{p}''|^2} & \\
(|\boldsymbol{p}''|^2 (\check{f}(|\boldsymbol{p}''|)t)_k^{\{\gamma\}} ({\scriptstyle \{E\}}, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}, x') &- \\
p_0^2({\scriptstyle \{E\}})(\check{f}(p_0({\scriptstyle \{E\}}))t)_k^{\{\gamma\}} ({\scriptstyle \{E\}}, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}, x')) &+ \\
\frac{1}{2} p_0({\scriptstyle \{E\}})(\check{f}(p_0({\scriptstyle \{E\}}))t)_k^{\{\gamma\}} ({\scriptstyle \{E\}}, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}, x') \left(\ln \left(\frac{\bar{p} + p_0({\scriptstyle \{E\}})}{\bar{p} - p_0({\scriptstyle \{E\}})} \right) - i\pi \right) & \quad (7.22)
\end{aligned}
$$

Unlike (7.15), (7.22) is ready for numerical realization. For negative energies the limit $\epsilon \to 0$ in (7.15) can be taken directly for $|\boldsymbol{p}''| = p_0({\scriptstyle \{E\}})$, except for the special case $E = E_d$ - deuteron binding energy. Looking at (7.15) or (7.22) it can be observed that the integral over ϕ'' and the sum over j can be prepared beforehand and stored in a table for future use. This can greatly reduce the time of the computation, as in the numerical realization the operators will be applied many times.

One strategy of solving equation (7.17) with the use of (7.22) involves constructing the matrix representation of $\check{\mathcal{B}}$ explicitly. This can be done separately for each independent case with different values of γ, E, $|\boldsymbol{p}|$ and this approach (among others) was explored in [20]. The complexity of (7.22) makes the task of writing down the explicit matrix representation very difficult, we will not explore this approach here. Another downside of this strategy is the large computational cost. Calculating one independent case can take several hours on a standard PC - this is not acceptable if we require calculating the transition operator for numerous values of the 2N energy.

Another possibility is the use of Krylov subspace methods such as the Arnoldi algorithm described in Appendix A. In this approach a (say) 40×40 dimensional matrix representation of the $\left[\check{\mathcal{B}}\right]^{40\times 40}$ operator is constructed. No explicit matrix representation is necessary for the construction. All that is required is the calculation of the action of $\check{\mathcal{B}}$ on a scalar function t and the computation of a scalar product. The eigenvalues of this matrix correspond to the most extreme eigenvalues of the original operator. This makes the small matrix a good representation of $\check{\mathcal{B}}$ that can be used in solving the original equation (7.17). Finally we end up with a 40 dimensional linear equation that can be solved using standard linear solvers (for example LAPACK or *Mathematica*® linear solvers).

The results presented at the end of this chapter contain a comparison of different methods for calculating the transition operator that were discussed in detail in [20]. From the point of view of this text, the most important are the iterative results - iterative schemes (like the Arnoldi scheme described in the Appendix) can be directly applied to (7.17). Nonetheless, it is interesting to see how the iterative results compare to other methods, especially the inversion of the explicit matrix representation of $\check{\mathcal{B}}$.

In order to provide an example of how the formalism developed in this chapter can be used to describe a physical situation, we will be showing results for nucleon-nucleon scattering. An additional complication arises when calculating the on-shell t-matrix. In this scenario the spin operators corresponding to different scalar functions $t_i(|\boldsymbol{p}|, |\boldsymbol{p}|, x)$ are not linearly independent. One of the operators can be expressed in terms of a linear combination of the other 5 and the on-shell transition operator is typically described by using 5 Wolfenstein parameters. Calculating the transition from t_i to the 5 Wolfenstein parameters a, c, g, h, m is a simple exercise (for example see [45] or [20]). Results for the NNLO potential can be seen in Figure 7.1.

The road from Wolfenstein parameters to physical observables is straightforward. Expressions for the general spin cross-sections can be found in [45] and are functions of the parameters. Observables are then constructed from these cross-sections, selected examples can be seen in Figure 7.2.

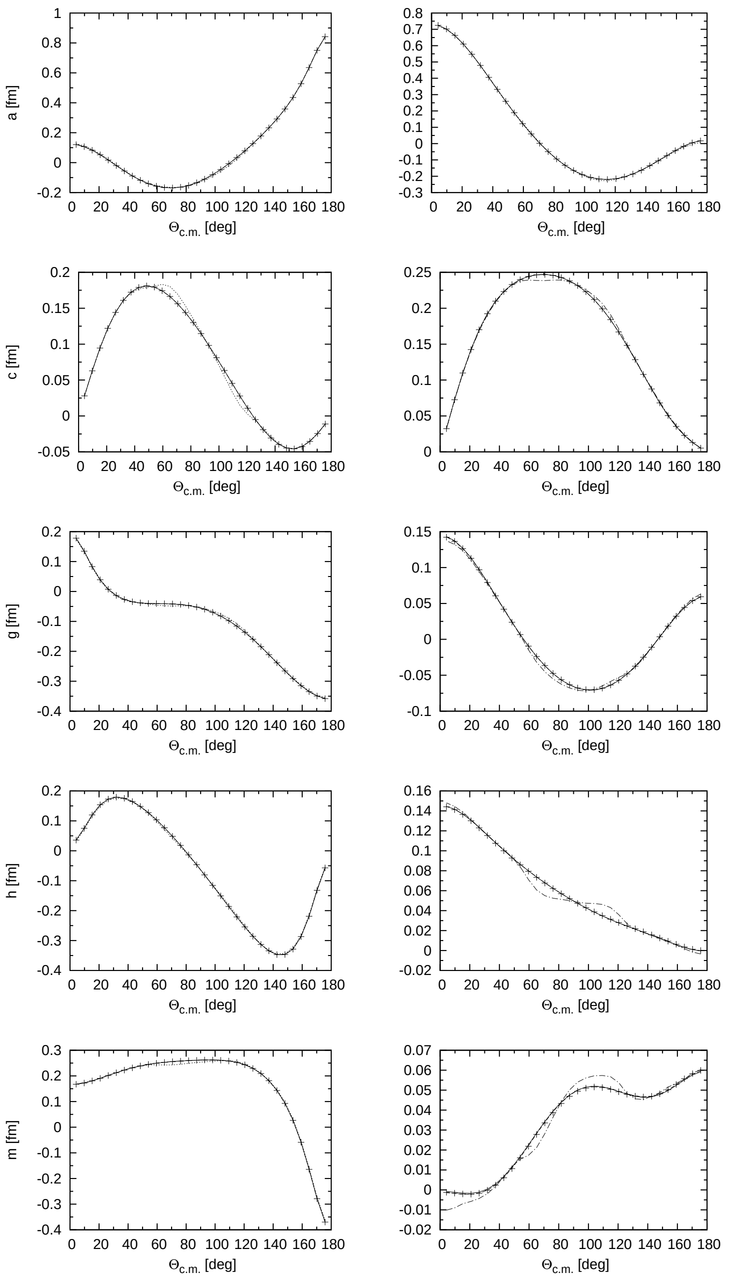


Figure 7.1: Wolfenstein parameters calculated for laboratory kinetic energy 300[MeV] as a functions of the CM angle $\theta_{\mathrm{c.m.}}$ using the NNLO potential [16] for the neutron-proton scattering. Left panel: real part, right panel: imaginary part. Crosses, dashed and solid lines correspond to different methods of solving the three dimensional problem (iterative, via k-matrix and using matrix inversion). These methods are described in detail in [20]. From the point of view of this thesis, the iterative results are most important. Results reprinted from our paper [20].

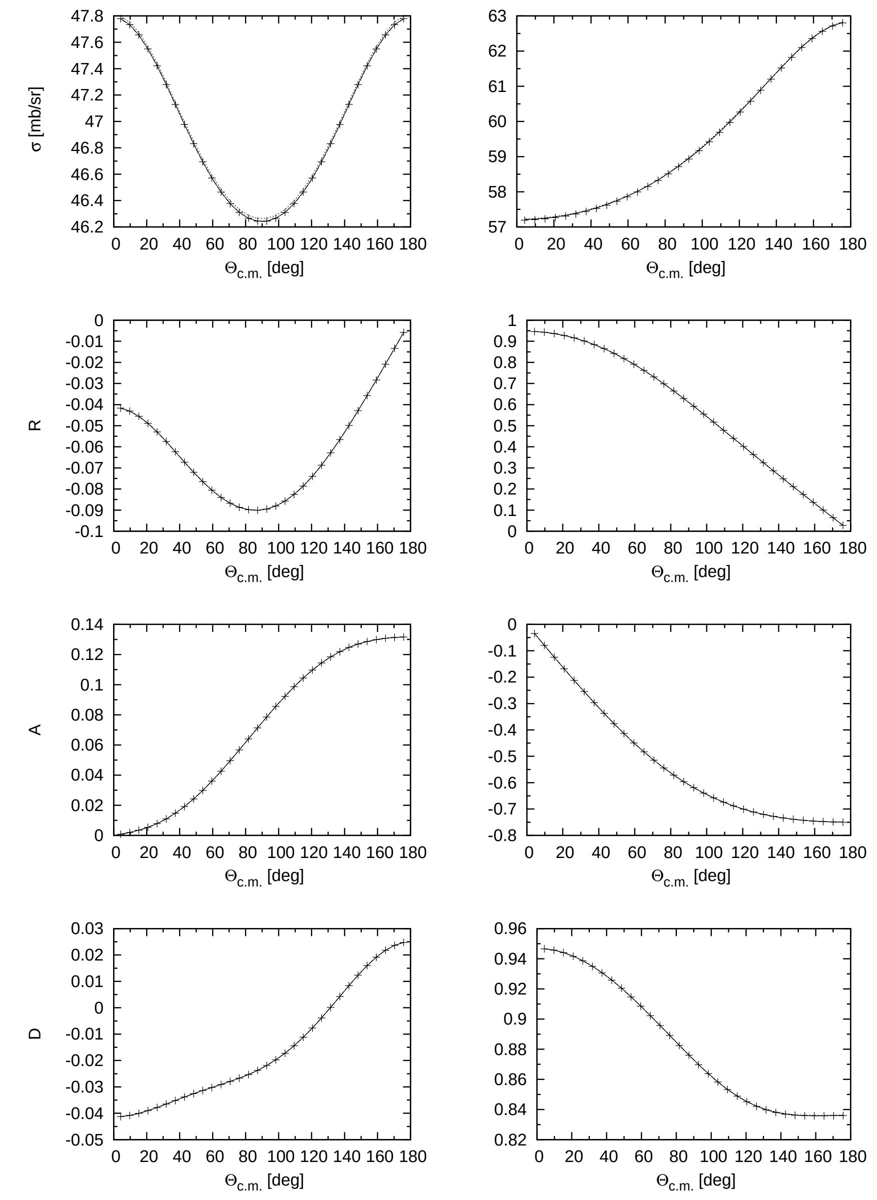


Figure 7.2: Left panel: observables for neutron-neutron scattering, right panel: observables for neutron-proton scattering. Calculations for laboratory kinetic energy 13[MeV] as a functions of the CM angle $\theta_{\mathrm{c.m.}}$ using the NNLO potential [16]. Crosses, dashed and solid lines correspond to different methods of solving the three dimensional problem (iterative, via k-matrix and using matrix inversion). These methods are described in detail in [20]. This thesis describes how to obtain the results based on iterations. Results reprinted from our paper [20].

7.2 Negative energies

Calculating the bound state of the 3N system with the use of the transition operator, as is described in [23] and equations (4.69) and (4.71), will reveal the necessity to calculate the 2N transition operator for $E < 0$. For the case where the energy is negative and far from the deuteron binding energy E_b we can use the simpler version of (7.18) with $\check{\mathcal{B}}$ obtained directly from equation (7.15). For this case, the integrated function does not have singular behavior resulting from $_{\{E\}} - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon$, since the energy is negative.

This simple scheme will break down when energies are close to the deuteron binding energy. A more detailed explanation of this problem can be found in [46]; here we repeat only the most important points. As a reminder we rewrite the iterative LSE equation (4.51) below:

$$\check{t}(E) = \check{V} + \check{V}\check{G}_0(E+i\epsilon)\check{V} + \check{V}\check{G}_0(E+i\epsilon)\check{V}\check{G}_0(E+i\epsilon)\check{V} + \ldots \tag{7.23}$$

Equivalently, this equation can be rewritten in the form:

$$\check{t}(E) = \check{V}\left(\check{1} + \check{G}_0(E+i\epsilon)\check{V} + \check{G}_0(E+i\epsilon)\check{V}\check{G}_0(E+i\epsilon)\check{V} + \ldots\right) \tag{7.24}$$

and when acted upon the deuteron bound state satisfying equation (4.55) this equation produces a singularity:

$$\begin{aligned} &\check{t}(E) \mid \Psi^{2N}_{bound}\rangle = \\ &\check{V}\left(\check{1} + \check{G}_0(E+i\epsilon)\check{V} + \check{G}_0(E+i\epsilon)\check{V}\check{G}_0(E+i\epsilon)\check{V} + \ldots\right) \mid \Psi^{2N}_{bound}\rangle = \\ &\check{V}\left(\check{1} + \check{1} + \check{1} + \ldots\right) \mid \Psi^{2N}_{bound}\rangle. \end{aligned} \tag{7.25}$$

This singular behavior is limited to the isospin 0 case (the isospin of the deuteron). The exact form of the singularity at $E = E_b$ can be worked out [46]:

$$\check{t}(E \to E_b) = \check{V} \mid \Psi^{2N}_{bound}\rangle \frac{1}{E - E_b} \langle \Psi^{2N}_{bound} \mid \check{V}. \tag{7.26}$$

In order to arrive at this solution the completeness relation, now with the bound state, has to be inserted into the LSE written in terms of the full propagator from equation (4.42):

$$\check{t} = \check{V} + \check{V}\check{G}(E)\check{V}. \tag{7.27}$$

Using the operator form of the deuteron bound state (6.1) we arrive at an expression for the transition operator at energies close to the deuteron binding energy.

$$\begin{aligned} &\langle \boldsymbol{p}' \mid \check{t}(E \to E_b) \mid \boldsymbol{p}\rangle = \langle \boldsymbol{p}' \mid \left(\check{V} \mid \Psi^{2N}_{bound}\rangle \frac{1}{E - E_b}\langle \Psi^{2N}_{bound} \mid \check{V}\right) \mid \boldsymbol{p}\rangle = \\ &\left(E_b - \frac{\boldsymbol{p}'^2}{m}\right)\langle \boldsymbol{p}' \mid \Psi^{2N}_{bound}\rangle \frac{1}{E - E_b}\langle \Psi^{2N}_{bound} \mid \boldsymbol{p}\rangle \left(E_b - \frac{\boldsymbol{p}'^2}{m}\right) = \\ &\left(E_b - \frac{\boldsymbol{p}'^2}{m}\right)\left(E_b - \frac{\boldsymbol{p}^2}{m}\right) \\ &\sum_{l=1}^{2}\sum_{l'=1}^{2}\sum_{m_d=-1}^{1} \phi_{l'}(|\boldsymbol{p}'|)\phi_l(|\boldsymbol{p}|)^* \left[\check{b}_{l'}(\boldsymbol{p}')\right]\left[\mid 1m_d\rangle\langle 1m_d \mid\right]\left[\check{b}_l(\boldsymbol{p})\right]^\dagger \end{aligned} \tag{7.28}$$

All operators in the above equations act in the 2N spin space and can be calculated using the *util1N2N3N.m* package. One final step has to be taken in order to calculate the six scalar functions that will define the transition operator for energies close to the deuteron binding energy, we can use the procedure described in Chapter 5 to perform the decomposition of (7.28). Appendix D points to a notebook that calculates (7.28) and performs the decomposition.

PROGRAMS/toperator/nb/deu.nb

implements this procedure and produces the `tab` array that can be used as a replacement for t_i when the energy is close to the 2N binding energy for the isospin 0 case.

The results of this substitution (7.28) can be appreciated in Figures 7.3 and 7.4, where we calculate the transition operator multiplied by $(E - E_b)$. Each plot in 7.3 contains a comparison of the transition operator calculated at the deuteron binding energy using (7.28) (for the isospin 0 case) and the transition operator calculated using (7.15) directly for an energy close to the deuteron binding energy. The substitution of (7.28) produces functions that fit smoothly to other energy cases. Figure 7.4 contains plots of the energy dependence of a couple chosen values of the transition operator. Also in this case the results based on the deuteron residue fit nicely to other results.

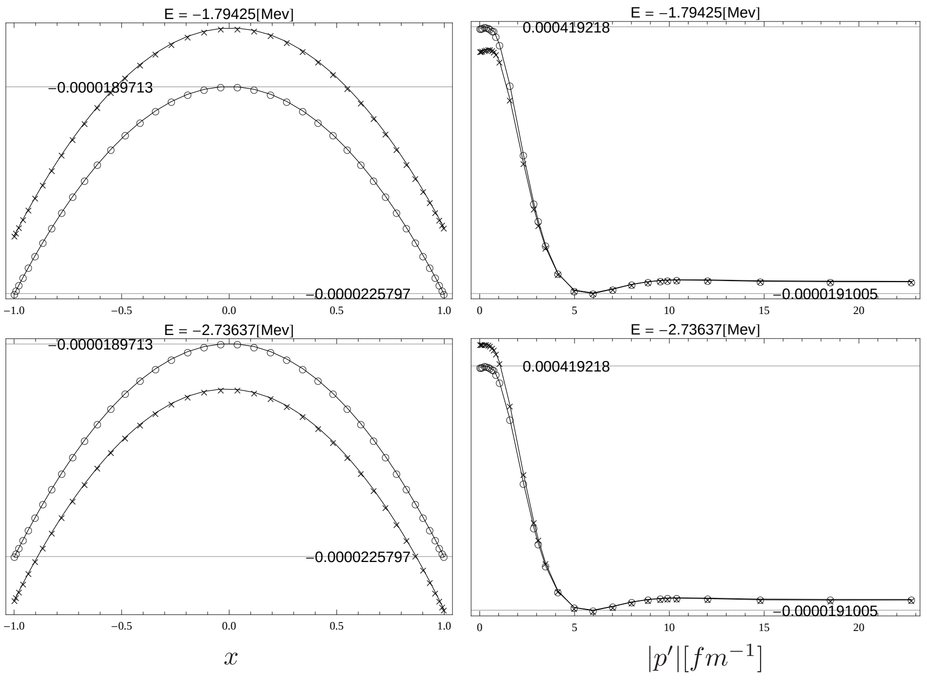
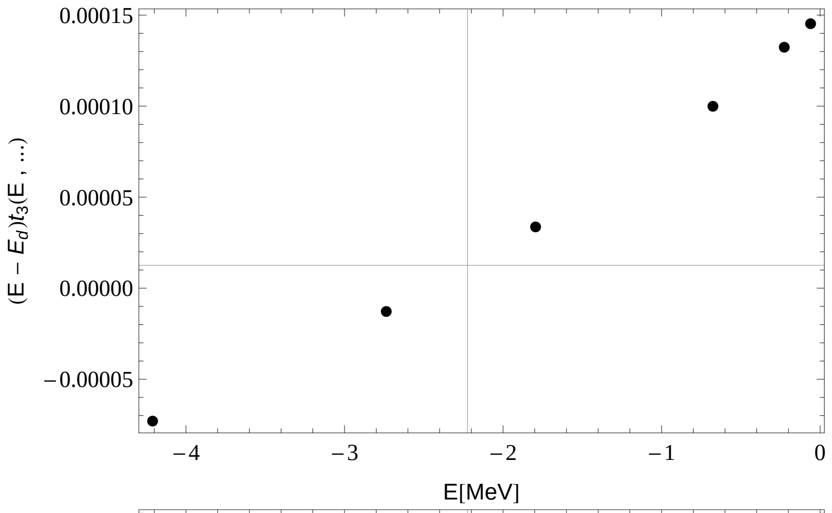


Figure 7.3: Left panel: dependence of $(E - E_d)t_2(E, |\boldsymbol{p}'| = 6.0[fm^{-1}], |\boldsymbol{p}| = 0.3, x = \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}})[fm]$ on x, right panel: dependence of $(E - E_d)t_2(E, |\boldsymbol{p}'|[fm^{-1}], |\boldsymbol{p}| = 0.3, x = \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}} = -0.19)[fm]$ on $|\boldsymbol{p}'|$. Both are calculated for the isospin 0 case and using the Bonn B potential from [10]. Circles mark values calculated from the deuteron bound state at $E_d = -2.2242$[MeV]. Crosses on plots in different rows mark values calculated for various energies - the smooth transition through E_d is visable. The solid lines are interpolated from the calculated points (circles, crosses).



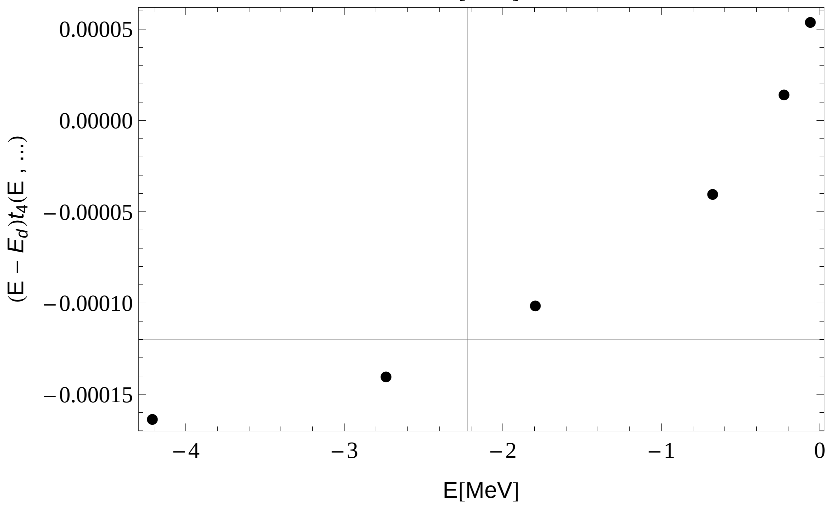


Figure 7.4: Dependence of $(E - E_d)t_i(E, |\boldsymbol{p}'| = 0.26[fm^{-1}], |\boldsymbol{p}| = 0.3, x = \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}} = -0.41)[fm]$ on the energy E for two different choices of i. The crossing point of the horizontal and vertical lines is calculated from the deuteron bound state at $E_d = -2.2242$[MeV]. All plots are calculated for the isospin 0 case and using the Bonn B potential from [10].

Chapter 8

Deuteron electro-disintegration

Having calculated the deuteron bound state and the transition operator it is a good moment to use this in a practical situation. The multitude of energies for the transition operator matrix elements that will be needed in this chapter will reveal the necessity for a quick implementation of transition operator calculations. This can be achieved by using an iterative approach such as the Arnoldi algorithm discussed in Appendix A. The following discussion is taken from our paper on the deuteron electro-disintegration [21]. The notation was changed to be consistent with this thesis and additional detailed information is given on the construction and numerical realization of the resulting expressions. This was not possible in the paper due its natural limitations.

We examine the reaction $e+^2\mathrm{H} \to e+p+n$ in an energy range that allows for a non-relativistic treatment of the nuclear states. The initial state of the system consists of the deuteron with the total momentum $\boldsymbol{K}=0$ and the electron with a momentum value q_e. We will assume that the energy of the electron $E_e \approx q_e$ - due to the small mass of the particle compared to its kinetic energy. In the final state the deuteron is split into a proton and neutron, the electron is scattered by the angle θ_e and changes momentum to q'_e ($E'_e \approx q'_e$). In this process the 2N system (the deuteron in the initial state, the free proton and neutron in the final state) receives a momentum transfer $\boldsymbol{Q} = \boldsymbol{q}'_e - \boldsymbol{q}_e$, whose magnitude can be calculated as:

$$Q = \sqrt{q_e^2 + q_e'^2 - 2q_e q'_e \cos\theta_e} \approx \sqrt{E_e^2 + E_e'^2 - 2E_e E'_e \cos\theta_e}. \tag{8.1}$$

The electromagnetic interaction is mediated by a virtual photon with the four-momentum $(\omega, \boldsymbol{Q}) = (E'_e - E_e, \boldsymbol{Q})$ and we will be working in a reference frame in which $\boldsymbol{Q}$ is parallel to the $\hat{z}$ axis. The energy and momentum is conserved in the reaction making it possible to calculate the total final momentum of the 2N system:

$$\boldsymbol{K}^{\mathrm{f}} = \boldsymbol{k}_1 + \boldsymbol{k}_2 = (0,0,Q) \tag{8.2}$$

and the magnitude of the relative momentum:

$$|\boldsymbol{p}^{\mathrm{f}}| = |\frac{1}{2}(\boldsymbol{k}_1 - \boldsymbol{k}_2) = \frac{1}{2}\sqrt{4m(E_d + E_e - E'_e) - Q^2} \tag{8.3}$$

in the final state. Here E_d is the deuteron binding energy (around -2.2[MeV]) and additionally we make the assumption about the identical mass of the proton and neutron m. We have a freedom of choice for the angle of $\boldsymbol{p}^{\mathrm{f}}$.

Our aim is to find the matrix element $\boldsymbol{M}^{\mu}$:

$$
\begin{aligned}
&\boldsymbol{M}^{\mu}\left(\boldsymbol{p}^{\mathrm{f}}, \boldsymbol{K}^{\mathrm{f}}\right) \\
\equiv_a &\langle \boldsymbol{p}^{\mathrm{f}} \boldsymbol{K}^{\mathrm{f}}, m_1\nu_1, m_2\nu_2 \mid (\check{1} + \check{t}(E) G_0(E)) \check{j}^{\mu}_{2N} \mid \Psi(m_d)\, \boldsymbol{K} = 0 \rangle \\
=_a &\langle \boldsymbol{p}^{\mathrm{f}} \boldsymbol{K}^{\mathrm{f}}, m_1\nu_1, m_2\nu_2 \mid (\check{1} + \check{t}(E) G_0(E)) \\
&(\check{j}^{\mu}(1) + \check{j}^{\mu}(2) + \check{j}^{\mu}(1,2)) \mid \Psi(m_d)\, \boldsymbol{K} = 0 \rangle \\
= 2_a &\langle \boldsymbol{p}^{\mathrm{f}} \boldsymbol{K}^{\mathrm{f}}, m_1\nu_1, m_2\nu_2 \mid \check{j}^{\mu}(2) \mid \Psi(m_d)\, \boldsymbol{K} = 0 \rangle \\
+_a &\langle \boldsymbol{p}^{\mathrm{f}} \boldsymbol{K}^{\mathrm{f}}, m_1\nu_1, m_2\nu_2 \mid \check{j}^{\mu}(1,2) \mid \Psi(m_d)\, \boldsymbol{K} = 0 \rangle \\
+_a &\langle \boldsymbol{p}^{\mathrm{f}} \boldsymbol{K}^{\mathrm{f}}, m_1\nu_1, m_2\nu_2 \mid \check{t} G_0 \check{j}^{\mu}_{2N} \mid \Psi(m_d)\, \boldsymbol{K} = 0 \rangle, \qquad (8.4)
\end{aligned}
$$

where initially there is a deuteron bound state (with $\boldsymbol{K} = 0$ and the $\hat{z}$ projection of the spin m_d) and a free electron with momentum $\boldsymbol{q}_{\mathrm{e}}$. The final state is antisymmetric:

$$
{}_a\langle \boldsymbol{p}\boldsymbol{K}, m_1\nu_1, m_2\nu_2 \mid \equiv \frac{1}{2} \left(\langle \boldsymbol{p}\boldsymbol{K}, m_1\nu_1, m_2\nu_2 \mid - \langle -\boldsymbol{p}\boldsymbol{K}, m_2\nu_2, m_1\nu_1 \mid \right). \qquad (8.5)
$$

The energy value in the transition operator and propagator $E = (\boldsymbol{p}^{\mathrm{f}})^2/m$, m_i (ν_i) being the $\hat{z}$ projection of the spin (isospin) of the individual particles and the particles in the second term have switched places ($m_1 \to m_2$, $m_2 \to m_1$, $\nu_1 \to \nu_2$, $\nu_2 \to \nu_1$) resulting in a change of momentum ($\boldsymbol{p} \to -\boldsymbol{p}$). Equation (8.4) introduces electro-magnetic current operators $\check{j}(1)$, $\check{j}(2)$, $\check{j}(1,2)$ acting on the degrees of freedom of particle 1, 2, both 1 and 2 - respectively. The index $\mu = 0$ marks the charge density operator and $\mu = 1, 2, 3$ mark the vector components of the current. In the following discussion we will drop this index for clarity and assume that a choice of coordinate system and index is available. Additionally when referring to matrix elements of type (8.4) that do not involve the transition operator we will use the term "plane wave" part. The fact that in (8.4) we finally use only $\check{j}(2)$ reflects the antisymmetry of the final 2N state.

Finally it should be noted that the transition operator in (8.4) should satisfy a slightly modified version of the Lippmann Schwinger equation (4.50):

$$
\check{t}(E) = \check{V} + \check{t}(E) \check{G}_0(E + i\epsilon) \check{V},
$$

with the order of operators reversed. This not a big complication, we can use the methods described earlier to calculate the scalar functions that define this "reversed" transition operator. It can be shown in a simple calculation that the "reversed" scalar functions have the initial and final relative momenta switch places with respect to the scalar functions that define the transition operator that follows (4.50).

8.1 Single nucleon current in three dimensions

The matrix elements in momentum space (operators in isospin - spin space) of the single nucleon currents (1N) depend only on the degrees of freedom of one particle. For instance $\check{j}(2)$ produces a matrix element that can be written in terms of the sum and difference of the individual initial ($\boldsymbol{k}_1$, $\boldsymbol{k}_2$) and final ($\boldsymbol{k}_1'$, $\boldsymbol{k}_2'$) momenta of the second particle:

$$\left[\langle \boldsymbol{k}_1'\boldsymbol{k}_2' \mid \check{j}(2) \mid \boldsymbol{k}_1\boldsymbol{k}_2\rangle\right]^{2\mathrm{N}} = \delta\left(\boldsymbol{k}_1' - \boldsymbol{k}_1\right)\left[j(2, \boldsymbol{k}_2' - \boldsymbol{k}_2, \boldsymbol{k}_2' + \boldsymbol{k}_2)\right] \\ \equiv \delta\left(\boldsymbol{k}_1' - \boldsymbol{k}_1\right)\left[j(2, \boldsymbol{k}_-, \boldsymbol{k}_+)\right]. \tag{8.6}$$

This matrix element can also be written in terms of the relative and total momentum eigenstates ($\boldsymbol{p}$, $\boldsymbol{K}$) as:

$$\left[\langle \boldsymbol{p}'\boldsymbol{K}' \mid \check{j}(2) \mid \boldsymbol{p}\boldsymbol{K}\rangle\right] = \delta\left(\frac{1}{2}\boldsymbol{K}' - \frac{1}{2}\boldsymbol{K} + \boldsymbol{p}' - \boldsymbol{p}\right) \\ \left[j(2, \boldsymbol{k}_- = \frac{1}{2}\boldsymbol{K}' - \frac{1}{2}\boldsymbol{K} - \boldsymbol{p}' + \boldsymbol{p}, \boldsymbol{k}_+ = \frac{1}{2}\boldsymbol{K}' + \frac{1}{2}\boldsymbol{K} - \boldsymbol{p}' - \boldsymbol{p})\right]. \tag{8.7}$$

Expressions inside the square brackets [...] have a matrix representation that can be easily constructed using the KP and the *util1N2N3N.m* package, see Apendix B. Using the deuteron bound state in the operator form (6.1) and the initial isospin state of this system ($[\mid 0\,0\rangle]$), the action of $\check{j}(2)$ on the deuteron state at rest can be calculated:

$$\left[\langle \boldsymbol{p}'\boldsymbol{K}' \mid j(2) \mid \Psi(m_d)\,\boldsymbol{K} = 0\rangle\right] = \\ = \sum_{l=1}^{2} \phi_l(|\boldsymbol{p}' + \frac{1}{2}\boldsymbol{K}'|)\left[j(2, \boldsymbol{K}', -2\boldsymbol{p}')\right]^{2\mathrm{N}}\left[B_l(\boldsymbol{p}' + \frac{1}{2}\boldsymbol{K}')\right]^{2\mathrm{N}}\left[\mid 0\,0\rangle \otimes \mid 1 m_d\rangle\right]^{2\mathrm{N}} \\ \equiv \left[O^{1\mathrm{N}}(2, \boldsymbol{p}', \boldsymbol{K}')\right]^{2\mathrm{N}}\left[\mid 0\,0\rangle \otimes \mid 1 m_d\rangle\right]^{2\mathrm{N}}. \tag{8.8}$$

This produces a final state in the $[\ldots]^{2\mathrm{N}}$ matrix representation (table E.2 in the Appendix E). $\left[O^{1\mathrm{N}}\right]$ is the resulting single particle operator. When used as in equation (8.8) it will yield the matrix representation of the final isospin - spin state for the final momenta $\boldsymbol{p}'$, $\boldsymbol{K}'$.

The sum in equation (8.8) is a basic building block of the calculation. A FORTRAN implementation (both fixed format (***.f***) and free format (***.f90***)) of

$$\left[O^{1\mathrm{N}}(2, \boldsymbol{p}', \boldsymbol{K}')\right]\left[\mid 0\,0\rangle \otimes \mid 1 m_d\rangle\right]$$

is available. Details on the notebooks and codes can be found in Appendix D. We use the single nucleon non-relativistic current density:

$$\left[j_0(2, \boldsymbol{k}_-, \boldsymbol{k}_+)\right]^{2\mathrm{N}} = \left[G_E^p \check{\Pi}^p + G_E^n \check{\Pi}^n\right]^{2\mathrm{Nisospin}} \otimes \left[\check{1}\right]^{2\mathrm{Nspin}}, \tag{8.9}$$

and the convection and spin current operators (these two parts are used in the notebooks):

$$\left[j_{\mathrm{conv}}(2, \boldsymbol{k}_-, \boldsymbol{k}_+)\right]^{2\mathrm{N}} = \frac{\boldsymbol{k}_+}{2M_N}\left[G_E^p \check{\Pi}^p + G_E^n \check{\Pi}^n\right]^{2\mathrm{Nisospin}} \otimes \left[\check{1}\right]^{2\mathrm{Nspin}}, \tag{8.10}$$

$$\left[j_{\mathrm{spin}}(2, \boldsymbol{k}_-, \boldsymbol{k}_+)\right]^{2\mathrm{N}} = \left[G_M^p \check{\Pi}^p + G_M^n \check{\Pi}^n\right]^{2\mathrm{Nisospin}} \otimes \left[\frac{i\sigma(2) \times \boldsymbol{k}_-}{2M_N}\right]^{2\mathrm{Nspin}}. \tag{8.11}$$

Here:

$$\left[\check{\Pi}^p\right]^{2\mathrm{Nisospin}} = \frac{1}{2}\left(\left[\check{1}\right] + \left[\tau_3\right]\right) \tag{8.12}$$

is an operator projecting on to the proton subspace and

$$\left[\check{\Pi}^n\right]^{\text{2Nisospin}} = \frac{1}{2}\left(\left[\check{1}\right] - \left[\tau_3\right]\right), \tag{8.13}$$

is an operator projecting on to the neutron subspace. $G_E^p, G_E^n, G_M^p, G_M^n$ are the electric and magnetic proton and neutron form factors. The remaining notation is the same as in [50]. In the notebook we use the spherical components:

$$\check{\boldsymbol{j}}(2)_+ = \frac{1}{\sqrt{2}}\left(-\check{\boldsymbol{j}}(2)_{\hat{\boldsymbol{x}}} - i\check{\boldsymbol{j}}(2)_{\hat{\boldsymbol{y}}}\right),$$

$$\check{\boldsymbol{j}}(2)_- = \frac{1}{\sqrt{2}}\left(\check{\boldsymbol{j}}(2)_{\hat{\boldsymbol{x}}} - i\check{\boldsymbol{j}}(2)_{\hat{\boldsymbol{y}}}\right)$$

and produce codes for $\left[O^{1\text{N}}(2,\boldsymbol{p}',\boldsymbol{K}')\right]^{2\text{N}}\left[\mid 0\,0\rangle \otimes \mid 1 m_d\rangle\right]^{2\text{N}}$ with different values of m_d and for the different types of current operators (and components $+1$, -1).

8.2 2N currents in three dimensions

2N current operator matrix elements in the momentum space (operators in isospin - spin space) can be found in literature. They typically carry the following momentum dependence:

$$\left[\langle \boldsymbol{k}_1'\boldsymbol{k}_2' \mid j(1,2) \mid \boldsymbol{k}_1\boldsymbol{k}_2\rangle\right]^{2\text{N}} = \left[j(1,2,\boldsymbol{k}_1'-\boldsymbol{k}_1,\boldsymbol{k}_2'-\boldsymbol{k}_2)\right]. \tag{8.14}$$

Examples are given in [51, 52, 53]. We restrict our discussion to this class of momentum dependencies. Our formalism can, however, be generalized to include any type of momentum dependence. The classical approach to dealing with (8.14) is to use an expansion into a linear combination of scalar functions (f_i^{jS} , f_i^j) and combinations of operators in isospin space ($\check{T}_i$) and spin space ($\check{O}_{jS}$, $\check{\boldsymbol{O}}_j$):

$$\left[j^0(1,2)\right] = \sum_{i=1}^{5}\sum_{j=1}^{8} f_i^{jS}(\boldsymbol{k}_1'-\boldsymbol{k}_1,\boldsymbol{k}_2'-\boldsymbol{k}_2)\left[\check{T}_i \otimes \check{O}_{jS}\right], \tag{8.15}$$

$$\left[\vec{j}(1,2)\right] = \sum_{i=1}^{5}\sum_{j=1}^{24} f_i^{j}(\boldsymbol{k}_1'-\boldsymbol{k}_1,\boldsymbol{k}_2'-\boldsymbol{k}_2)\left[\check{T}_i \otimes \check{\boldsymbol{O}}_{j}\right], \tag{8.16}$$

with S used to distinguish density and vector parts of operators. This approach can be found for example in [54] where a general operator basis is given. In the present discussion this decomposition is not necessary and we can work directly with the current in the operator form. Equation (8.14) can also be written in terms of the relative and total momenta:

$$\left[\langle \boldsymbol{p}'\boldsymbol{K}' \mid j(1,2) \mid \boldsymbol{p}\boldsymbol{K}\rangle\right] = \left[j(1,2,\frac{1}{2}\boldsymbol{K}'-\frac{1}{2}\boldsymbol{K}+\boldsymbol{p}'-\boldsymbol{p},\frac{1}{2}\boldsymbol{K}'-\frac{1}{2}\boldsymbol{K}-\boldsymbol{p}'+\boldsymbol{p})\right]. \tag{8.17}$$

Again, using the operator form of the deuteron (6.1), the action of $j(1,2)$ can be calculated:

$$
\begin{aligned}
&\left[\langle \boldsymbol{p}'\boldsymbol{K}' \mid j(1,2) \mid \Psi(m_d)\,\boldsymbol{K}=0\rangle\right] \\
&= \int d^3\boldsymbol{p}'' \sum_{l=1}^{2} \phi_l(|\boldsymbol{p}''|) \left[j(1,2,\frac{1}{2}\boldsymbol{K}' + \boldsymbol{p}' - \boldsymbol{p}'', \frac{1}{2}\boldsymbol{K}' - \boldsymbol{p}' + \boldsymbol{p}'') \right] \\
&\qquad [B_l(\boldsymbol{p}'')]\,[\mid 0\,0\rangle \otimes \mid 1 m_d\rangle] \\
&\equiv \left[O^{2\mathrm{N}}(1,2,\boldsymbol{p}',\boldsymbol{K}')\right][\mid 0\,0\rangle \otimes \mid 1 m_d\rangle]\,. \qquad (8.18)
\end{aligned}
$$

$\left[O^{2\mathrm{N}}\right]$ is the resulting two-particle operator. It gives the full isospin - spin state for the final $\boldsymbol{p}'$, $\boldsymbol{K}'$ momenta.

Equation (8.18) is a basic building block of the calculation. A FORTRAN implementation (both fixed format (***.f***) and free format (***.f90***)) of

$$
\left[O^{2\mathrm{N}}(2,\boldsymbol{p}',\boldsymbol{K}')\right][\mid 0\,0\rangle \otimes \mid 1 m_d\rangle]
$$

is available. Details on the notebooks and codes can be found in Appendix D. For our calculations we use 2N currents from [55]. It should be noted that the code produced by the *Mathematica*® notebook should be modified to contain additional definitions for numerical coefficients and scalar functions. Detailed information about the additional definitions can be found in the notebook comments and [55].

8.3 Putting everything together

Using the decomposed form of the transition operator (7.2) and the previously derived operators from (8.8), (8.18) we can write down the detailed form of the rescattering part of M:

$$
\begin{aligned}
&[\langle \mathbf{p}'\mathbf{P}' \mid t(E)G_0(E)j_{2N} \mid \phi_d\, m_d\,\mathbf{P}=0\rangle] \\
&= \int d^3\mathbf{p}\,[\langle \mathbf{p}' \mid t(E) \mid \mathbf{p}\rangle]\,\frac{1}{E - \frac{\mathbf{p}^2}{m} + i\epsilon} \\
&\qquad \times [O(\mathbf{p},\mathbf{P}')]\,[\mid 0\,0\rangle \otimes \chi(m_d)] \\
&= m\int_0^{\bar{p}} \frac{\mathbf{p}^2\,[\mathbf{f}(|\mathbf{p}|)] - \mathbf{p}'^2\,[\mathbf{f}(|\mathbf{p}'|)]}{\mathbf{p}'^2 - \mathbf{p}^2}\,d|\mathbf{p}| \\
&+m\frac{|\mathbf{p}'|\,[\mathbf{f}(|\mathbf{p}'|)]}{2}\left(\ln\left(\frac{\bar{p}+|\mathbf{p}'|}{\bar{p}-|\mathbf{p}'|}\right) - i\pi\right) \\
&\qquad \times [\mid 0\,0\rangle \otimes \chi(m_d)]\,, \qquad (8.19)
\end{aligned}
$$

where we used (7.19). Here $\check{O}$ is either $\check{O}^{1N}$ or $\check{O}^{2N}$, $E = \frac{\mathbf{p}'^2}{m}$, the cut-off value for the integral ($\bar{p}$) was introduced and:

$$
[\mathbf{f}(|\mathbf{p}|)] = \int_0^{2\pi} d\phi \int_{-1}^{1} dx\,[\langle \mathbf{p}' \mid t(E) \mid \mathbf{p}\rangle]\,[O(\mathbf{p},\mathbf{P}')]\,. \qquad (8.20)
$$

The momentum dependence in (8.20) is due to the assumption:

$$\begin{aligned}\boldsymbol{p} = |\boldsymbol{p}| \\ \Big(\cos(\theta^f)\left(\sqrt{1-x^2}\cos(\phi)\cos(\phi^f) + x\sin(\phi^f)\right) - \sqrt{1-x^2}\sin(\phi)\sin(\theta^f), \\ \sqrt{1-x^2}(\cos(\phi)\cos(\phi^f)\sin(\theta^f) + \sin(\phi)\cos(\theta^f)) + x\sin(\phi^f)\sin(\theta^f), \\ x\cos(\phi^f) - \sqrt{1-x^2}\cos(\phi)\sin(\phi^f))\Big)\end{aligned} \tag{8.21}$$

and the integral:

$$\int \mathrm{d}^3\boldsymbol{p} \rightarrow \int_0^{\bar{p}} \mathrm{d}|\boldsymbol{p}||\boldsymbol{p}|^2 \int_0^{2\pi} \mathrm{d}\phi \int_{-1}^{1} \mathrm{d}x. \tag{8.22}$$

This choice of the parametrization of $\boldsymbol{p}$ integration is important to effectively use the previously calculated transition operator. If in (8.21) the angles θ^f, ϕ^f are chosen such that:

$$\boldsymbol{p}' = |\boldsymbol{p}'|(\sin(\phi^f)\cos(\theta^f), \sin(\phi^f)\sin(\theta^f), \cos(\phi^f)), \tag{8.23}$$

then the scalar product $\hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}} = x$. If additionally in the numerical realization the points chosen for x, $|\boldsymbol{p}|$ and $|\boldsymbol{p}'|$ in (8.22) match those that were chosen for the transition operator calculation, then no interpolations are necessary. It should, however, be noted that the gains from this parametrization are present only when working with 1N currents. In the case of 2N currents the additional integration makes it more feasible to use the same set of $|\boldsymbol{p}|$ integration points for all final momenta.

8.4 Results

The kinematic situation is presented in Figure 8.1. Selected observables are calculated under the one-photon exchange approximation [49] using the matrix elements M that were discussed earlier in this chapter. We consider first the exclusive unpolarized cross-section:

$$\mathrm{d}^5\sigma/(\mathrm{d}E_{e'}, \mathrm{d}\Omega_{e'}, \mathrm{d}\Omega_p),$$

with $E_{e'}$ being the energy of the outgoing electron, $\Omega_{e'}$ being the solid angle of the outgoing electron and Ω_p being the solid angle of the proton in the final state. Next we calculate one case of the spin dependent helicity asymmetry:

$$A_{\parallel} = \frac{\sigma(h=+1, \boldsymbol{J}_d \parallel \hat{\boldsymbol{z}}) - \sigma(h=-1, \boldsymbol{J}_d \parallel \hat{\boldsymbol{z}})}{\sigma(h=+1, \boldsymbol{J}_d \parallel \hat{\boldsymbol{z}}) + \sigma(h=-1, \boldsymbol{J}_d \parallel \hat{\boldsymbol{z}})},$$

where h is the initial helicity of the electron and the projection of the deuteron total angular momentum $\boldsymbol{J}_d$ on $\hat{\boldsymbol{z}}$ is $\boldsymbol{m}_d = 1$. Finally, the deuteron analyzing powers are considered - in this case the initial electron is unpolarized and we deal with different polarization states of the initial deuteron. All observables are displayed in Figures 8.2, 8.3, 8.4. We restrict ourselves to only one of the kinematics from [21] - *K6*, with the initial energy of the electron $E_e = 500$[MeV], the final energy of the electron $E'_e = 281.2$[MeV], the magnitude of the final relative momentum of the two-nucleons $p^f = 375.3$[MeV/c] and the momentum transfer $Q = 500$[MeV]. Our main objective is to compare different methods of calculation, thus we did not attempt to compare our results with measured data. Excellent convergence with PWD predictions is observed, thus proving the validity of PWD schemes. More information on the physical significance and calculation of the observables can be found for example in [50].

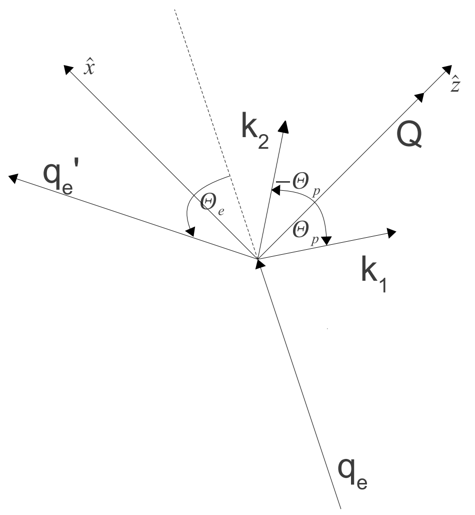


Figure 8.1: The incoming electron has momentum $\boldsymbol{q}_e$, the outgoing electron $\boldsymbol{q}'_e$. Momentum transfer to the 2N system $\boldsymbol{k}_1 + \boldsymbol{k}_2 = \boldsymbol{Q}$ is parallel to the $\hat{\boldsymbol{z}}$ axis.

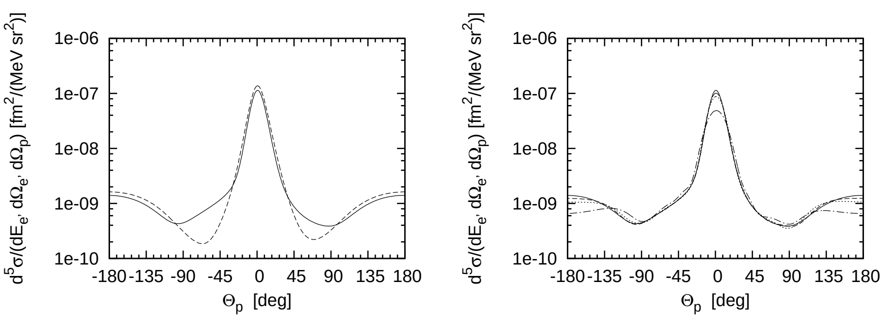
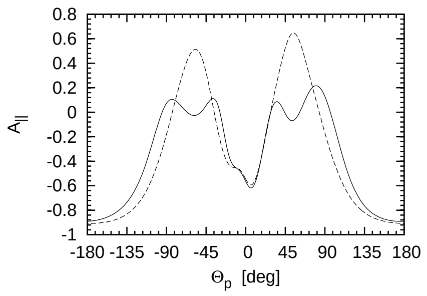


Figure 8.2: The unpolarized cross section as a function of the outgoing proton angle Θ_p from Figure 8.1. Negative values of θ_p correspond to the azimuthal angle $\phi_p = 0°$ and positive to $\phi_p = 180°$. On the left panel plane wave results (dashed) are compared with the full calculation (solid). On the right panel the convergence to the full 3D result (solid) is shown for different numbers of partial waves ($j \leq 4$ - dashed-dotted, $j \leq 7$ - dotted, $j \leq 9$ - dashed). Results reprinted from our paper [21] for kinematics *K6.*



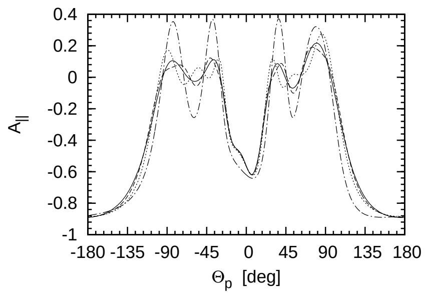


Figure 8.3: The same as in Figure 8.2 but for the selected example of the spin dependent helicity asymmetry.

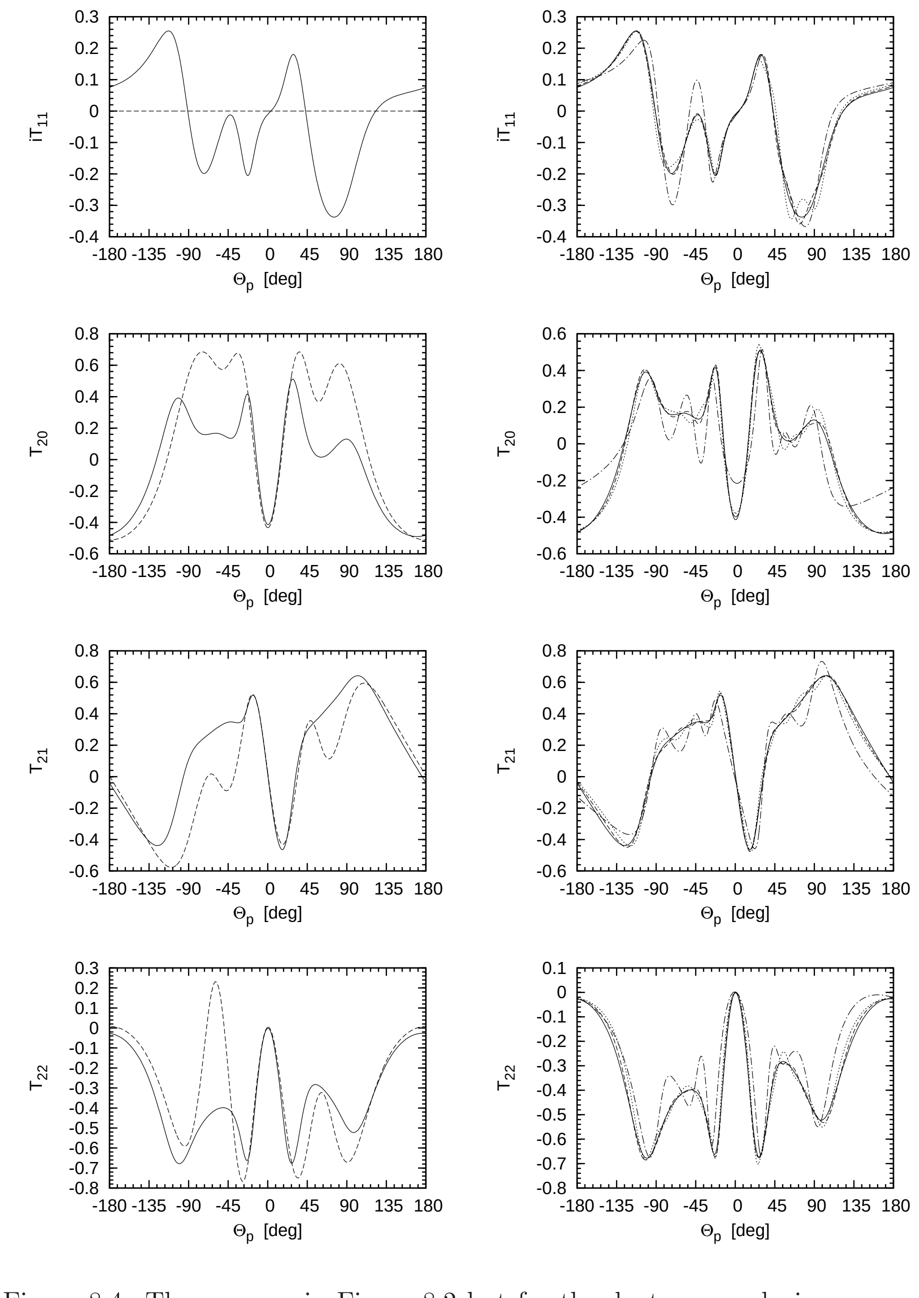


Figure 8.4: The same as in Figure 8.2 but for the deuteron analyzing powers T_{kq} (see for example [50]).

Chapter 9

Muon induced deuteron disintegration

The same methods that were used in the previous chapter to calculate matrix elements of the electromagnetic current operator (and consequently observables) for deuteron electro-disintegration can be applied to the description of muon induced deuteron disintegration: $\mu + d \to \nu_\mu + n + n$. This reaction is treated as the decay of the muonic atom, with the muon initially on the lowest K shell. The muon binding energy in this atom can be safely neglected and in the initial state we deal essentially with the deuteron and muon at rest. We work in a reference frame in which the three-momentum transfer to the 2N system $\boldsymbol{Q} = -\boldsymbol{p}_\nu$, where $\boldsymbol{p}_\nu$ is the outgoing neutrino momentum, antiparallel to $\hat{\boldsymbol{z}}$, as shown in Figure 9.1. Non-relativistic energy and momentum conservation laws lead to the expression for the final relative momentum magnitude of the neutron-neutron (nn) system:

$$|\boldsymbol{p}| = \frac{1}{2}\sqrt{4E_d m + 4m_\mu m - p_\nu(4m + p_\nu)}. \tag{9.1}$$

Here m_μ is the muon mass, $p_\nu \equiv |\boldsymbol{p}_\nu|$ is the magnitude of the outgoing neutrino momentum, m is the nucleon mass and $E_d \approx -2.2[MeV]$ is the deuteron binding energy. Our goal is to calculate the differential and total decay rates.

Using the Fermi approximation, the differential decay rate can be written as:

$$\frac{\mathrm{d}\Gamma}{\mathrm{d}p_\nu} = (2\pi)^2 \frac{(m'\alpha)^3}{\pi} 4\pi \int \mathrm{d}\hat{\boldsymbol{p}} \frac{m|\boldsymbol{p}|}{2}$$
$$\frac{1}{3} \sum_{m_d, m_1, m_2} \frac{1}{2} \sum_{s_\mu, s_\nu} \left| (2\pi)^4 \frac{G}{\sqrt{2}} L_\lambda J^\lambda \right|^2 . \tag{9.2}$$

In the factor $\frac{(m'\alpha)^3}{\pi}$ stemming from the non-relativistic wave function of the hydrogen-like atom $m' = \frac{m_\mu m_d}{m_\mu + m_d}$ is the reduced mass of the deuteron - muon system and $\alpha \approx \frac{1}{137}$ is the fine structure constant. The 4π factor comes from the integration over all neutrino angles. The first sum in (9.2) runs over all spin projections of the initial deuteron and the two final nucleons and the second sum is over the spin projections of the initial muon and the outgoing neutrino. The

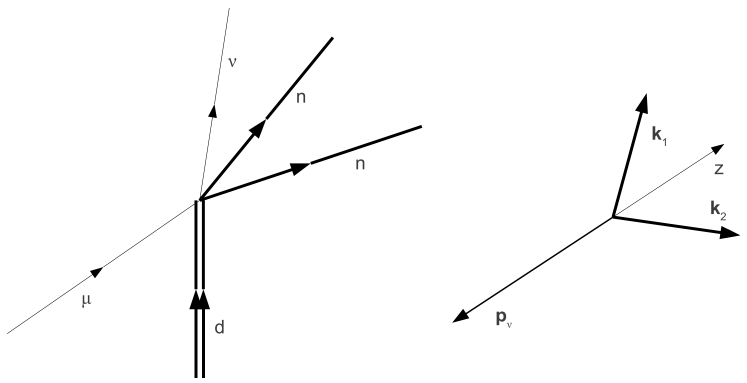


Figure 9.1: Diagrammatic representation and kinematics for the $\mu + d \to \nu_\mu + n + n$ reaction. This process is treated as the decay of the muonic atom, with the muon on the K shell.

integral is over all angles of the final relative momentum of the two nucleons, while the magnitude of this relative momentum is determined by (9.1). Finally, G is the Fermi constant. The weak transition operator T_w is given (up to a factor) by the contraction of the leptonic tensor L_λ and the hadronic tensor J^λ, $T_w = L_\lambda J^\lambda$. The leptonic tensor has the well known $V - A$ form:

$$L_\lambda \equiv \frac{1}{(2\pi)^3} l_\lambda = \frac{1}{(2\pi)^3} \bar{u}(\boldsymbol{p}_\nu, s_\nu)\gamma_\lambda(1-\gamma_5)u(\boldsymbol{p}_\mu, s_\mu)$$

and is given in terms of the Dirac spinors $u(\boldsymbol{p}, s)$ and standard Dirac matrices γ [49]. The hadronic tensor J^λ is just the matrix element of the nuclear weak current operator between the initial 2N bound state (deuteron) and the final 2N scattering state:

$$J^\lambda \equiv \frac{1}{(2\pi)^3} M^\lambda = \frac{1}{(2\pi)^3} \langle \boldsymbol{pK} \mid (1 + tG_0)\check{j}^{\mu}_{2N} \mid \phi_d \boldsymbol{p}_d = 0 \rangle.$$

The equation (9.2) is only the starting point. It has to be modified in order to take into account the hyperfine coupling in the initial muonic atom. That is why in (9.3) the muon spin and the total deuteron angular momentum are coupled (via the Clebsch-Gordan coefficients) to their total spin f.

$$\frac{\mathrm{d}\Gamma}{\mathrm{d}p_\nu} = G^2 \frac{(m'\alpha)^3}{2\pi^2} |\boldsymbol{p}_\nu|^2 \frac{m|\boldsymbol{p}|}{2} \int \mathrm{d}\hat{\boldsymbol{p}} \sum_{m_1, m_2, s_\nu = -1/2}^{1/2}$$

$$\frac{1}{2f+1} \sum_{m_f=-f}^{f} \left| \sum_{m_d=-1}^{1} \sum_{s_\mu=-1/2}^{1/2} C(\frac{1}{2}, 1, f; s_\mu, m_d, m_f) l_\lambda M^\lambda \right|^2 . \tag{9.3}$$

One focuses usually on the dominant decay from the hyperfine doublet states with $f = \frac{1}{2}$. The decay rate from the hyperfine states with $f = \frac{3}{2}$ is expected to be much smaller and is experimentally poorly determined.

When calculating the total decay rate we limit the integration over neutrino momenta to values that produce real values of $|\boldsymbol{p}|$ in (9.1). Alternatively we can use (9.1) to calculate the differential decay rate with respect to the magnitude of the relative momentum of the two outgoing nucleons $p = |\boldsymbol{p}|$. The relation to (9.3) is:

$$\frac{\mathrm{d}\Gamma}{\mathrm{d}p}(p) = \frac{\mathrm{d}\Gamma}{\mathrm{d}p_\nu}(p_\nu(p)) \left. \frac{1}{\left|\frac{\mathrm{d}p}{\mathrm{d}p_\nu}\right|} \right|_{p_\nu = p_\nu(p)} . \tag{9.4}$$

After some simple algebra, it can be seen that in order to calculate the total decay rate from (9.3), matrix elements of the weak 2N current operator in a form very similar to (8.4) need to be calculated. This can be achieved using the formalism described earlier in Chapters 8.1, 8.2 and 8.3. For the chosen kinematics the crucial part of the calculations comes from $\left|l_\lambda M^\lambda\right|^2$. This expression will be expanded to contain nuclear matrix elements of exactly the same type as (8.4) and all elements will carry the dependence on the momenta and spin quantum numbers.

The final element needed for the calculation is the current operator $\check{j}^\mu = (\check{j}^0, \check{\boldsymbol{j}})$. In order to arrive at the weak single nucleon current operator we use a standard approach with the non-relativistic expansion of

$$\begin{aligned} &\langle \boldsymbol{p}', s' \mid \check{j}^\mu(1) \mid \boldsymbol{p}, s\rangle = \\ &\bar{u}(\boldsymbol{p}', s')\Big(\left(g_1^V - 2mg_2^V\right)\gamma^\mu + g_2^V\,(p+p')^\mu \\ &+g_1^A\gamma^\mu\gamma^5 + g_2^A\,(p-p')^\mu\,\gamma^5\Big)\check{\tau}_- u(\boldsymbol{p}, s'), \end{aligned} \tag{9.5}$$

keeping terms up to $\frac{1}{m^2}$ (m being the nucleon mass). The form factors g_1^V, g_1^A, g_2^V, g_2^A of in (9.5) are chosen to agree with [56]. The final result of the expansion is

$$\begin{aligned} &\langle \boldsymbol{p}' \mid \check{j}^0(1) \mid \boldsymbol{p}\rangle = [\check{\tau}_-] \otimes \\ &\Bigg[g_1^V - (g_1^V - 4mg_2^V)\frac{(\boldsymbol{p}'-\boldsymbol{p})^2}{8m^2} + \left(g_1^V - 4mg_2^V\right)\mathrm{i}\frac{(\boldsymbol{p}'\times\boldsymbol{p})\cdot\check{\boldsymbol{\sigma}}}{4m^2} \\ &+g_1^A\frac{\check{\boldsymbol{\sigma}}\cdot(\boldsymbol{p}+\boldsymbol{p}')}{2m} + g_2^A\frac{\left(\boldsymbol{p}'^2-\boldsymbol{p}^2\right)}{4m^2}\check{\boldsymbol{\sigma}}\cdot(\boldsymbol{p}'-\boldsymbol{p})\Bigg], \end{aligned} \tag{9.6}$$

$$\begin{aligned} &\langle \boldsymbol{p}' \mid \check{\boldsymbol{j}}(1) \mid \boldsymbol{p}\rangle = [\check{\tau}_-] \otimes \\ &\Bigg[g_1^V\frac{\boldsymbol{p}+\boldsymbol{p}'}{2m} - \frac{1}{2m}\left(g_1^V - 2mg_2^V\right)\mathrm{i}\check{\boldsymbol{\sigma}}\times(\boldsymbol{p}-\boldsymbol{p}') \\ &+g_1^A\left(1 - \frac{(\boldsymbol{p}+\boldsymbol{p}')^2}{8m^2}\right)\check{\boldsymbol{\sigma}} + \\ &+\frac{g_1^A}{4m^2}\left((\boldsymbol{p}\cdot\check{\boldsymbol{\sigma}})\,\boldsymbol{p}' + (\boldsymbol{p}'\cdot\check{\boldsymbol{\sigma}})\,\boldsymbol{p} + \mathrm{i}\,(\boldsymbol{p}\times\boldsymbol{p}')\right) \\ &g_2^A\,(\boldsymbol{p}-\boldsymbol{p}')\,\frac{\check{\boldsymbol{\sigma}}\cdot(\boldsymbol{p}-\boldsymbol{p}')}{2m}\Bigg], \end{aligned} \tag{9.7}$$

where $\check{\tau}_-$ is the isospin lowering operator. Note that now $\boldsymbol{p}$ ($\boldsymbol{p}'$) is the initial (final) momentum of the single nucleon. Relation (9.6) introduces a single nucleon weak density operator acting in the isospin-spin space of the given nucleon and (9.7) is the corresponding vector part of the single nucleon weak operator acting in the same space. For the moment we restrict ourselves to the single nucleon currents. However, our formalism, allows us to incorporate operators acting on the degrees of freedom of both particles. Incorporation of two-nucleon currents is planned for the near future.

In our calculations we use the Bonn B 2N potential [10] to generate the initial deuteron state and the final 2N scattering states for a number of final neutrino momentum values. Typically we use about 30 values for the integral over p_ν. The angular integrations are not very demanding, either. For the unpolarized situation, there is no dependence on the azimuthal angle of the relative momentum and the integral is only one dimensional. In order to achieve fully convergent results no more than 30 polar angles are needed. After integrating (9.3) over all kinetically allowed neutrino momenta we get the total decay rates for $f = \frac{1}{2}$ and $f = \frac{3}{2}$. We provide two types of results: predictions based on the plane wave approximation (where the $\check{t}G_0$ term is neglected in the final state, see equation 8.4) and full results:

$$\begin{aligned}
\Gamma_{PW}^{f=\frac{1}{2}} &= 363.511[1/s] \\
\Gamma_{full}^{f=\frac{1}{2}} &= 396.118[1/s] \\
\Gamma_{PW}^{f=\frac{3}{2}} &= 10.425[1/s] \\
\Gamma_{full}^{f=\frac{3}{2}} &= 12.231[1/s]
\end{aligned} \tag{9.8}$$

Our value for $\Gamma_{full}^{f=\frac{1}{2}}$ is quite close to the experimentally measured decay rate (see for example [56] and the references therein - the experimental values from different measurements are in the range $365 - 470[\frac{1}{s}]$ their errors amount even to 25%). Before making a more thorough comparison, we would like to include 2N currents in our calculations.

Below in Figure 9.2 we show a comparison between the new three dimensional approach and the classical partial wave method for the differential decay rate with respect to the outgoing neutrino momentum. As can be seen both methods are in a very good agreement. This plot also contains the decay rate calculated with only the plane wave parts of the matrix elements. This illustrates that the effect of the 2N transition operator is significant, especially for higher energies. Figure 9.3 shows the three-dimensional decay rate for the $f = \frac{3}{2}$ case. The results are very small compared the $f = \frac{1}{2}$ case. Figures 9.4 and 9.5 contain similar comparisons but for the differential decay rate with respect to the relative momentum magnitude of the two outgoing neutrons.

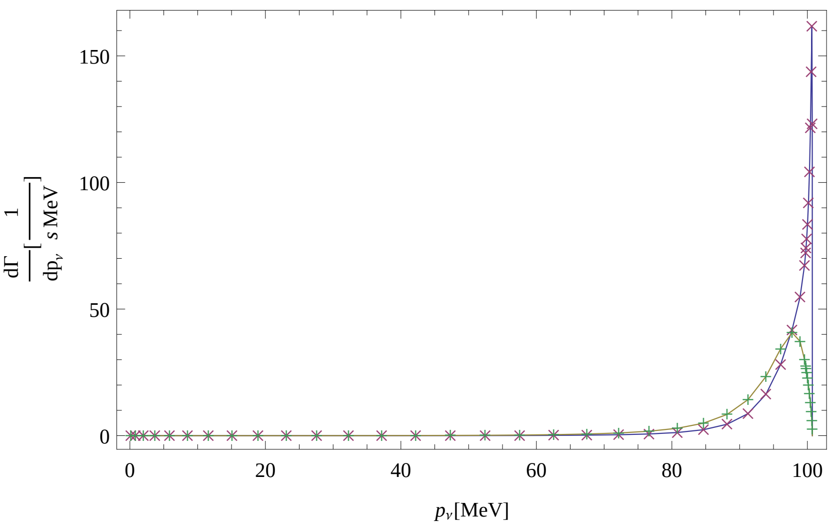


Figure 9.2: The contribution to the total decay rate from various values of the neutrino energy p_ν. The classical partial wave approach (crosses) is in very good agreement with the three dimensional calculations (solid line). The plane wave contribution calculated using the three dimensional approach (solid line) and the partial-wave approach (pluses) is also shown. All plots correspond to the hyperfine doublet state with $f = \frac{1}{2}$.

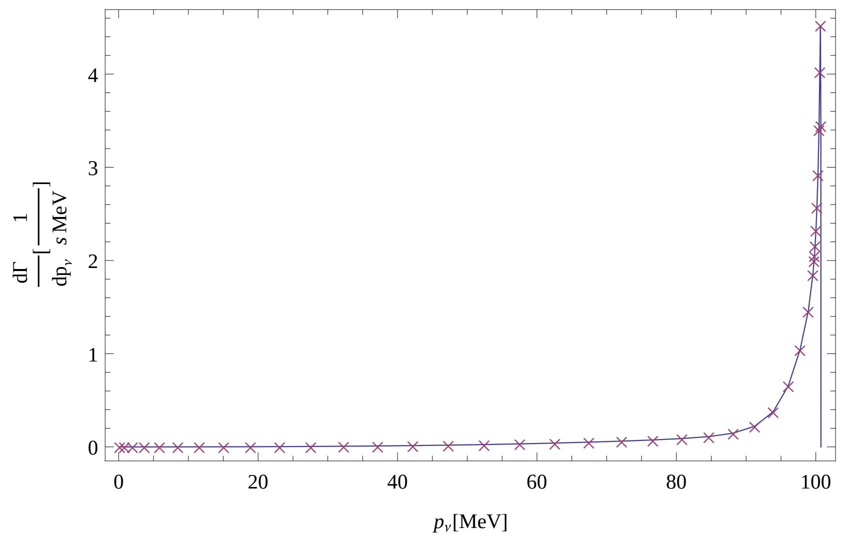


Figure 9.3: The same as in Figure 9.2 for $f = \frac{3}{2}$ but without the plane wave predictions.

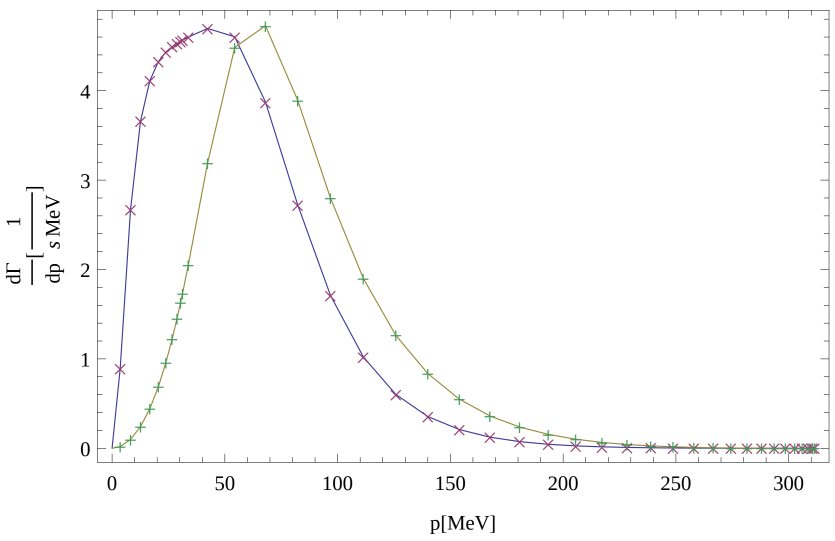


Figure 9.4: The contribution to the total decay rate from various values of the relative nn momentum magnitude p. The classical partial wave approach (crosses) is in a very good agreement with the three dimensional calculations (solid line). The plane wave contribution calculated using the three dimensional approach (solid line) and the partial-wave approach (pluses) is also shown. All plots correspond to the hyperfine doublet state with $f = \frac{1}{2}$.

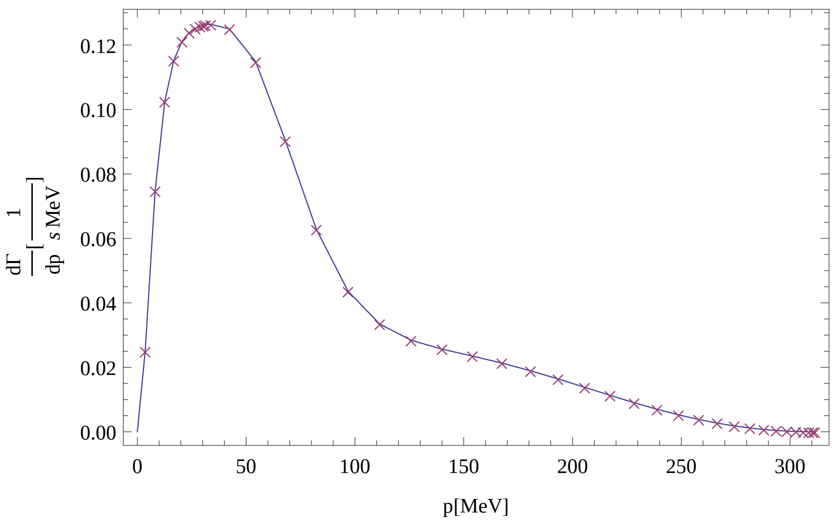


Figure 9.5: The same as in Figure 9.4 for $f = \frac{3}{2}$ but without the plane-wave predictions.

Chapter 10

Calculation of the 3N bound state in three dimensions

Out starting point is equation (4.66) for the Faddeev component of the 3N bound state:

$$\mid \psi^{3N} \rangle = \check{G}_0(E) \check{V} \left(\check{1} + \check{P} \right) \mid \psi^{3N} \rangle + \check{G}_0(E) \check{V}^{(1)} \left(\check{1} + \check{P} \right) \mid \psi^{3N} \rangle,$$

where $\check{V}$ is the 2N potential energy operator acting between particles $2, 3$ and $\check{V}^{(1)}$ is that part of the 3N potential operator that is invariant under the exchange of particles $2, 3$. The full wave function can be constructed from the Faddeev component using permutation operators:

$$\mid \Psi \rangle = \left(\check{1} + \check{P}_{12} \check{P}_{23} + \check{P}_{13} \check{P}_{23} \right) \mid \psi^{3N} \rangle \equiv \left(\check{1} + \check{P} \right) \mid \psi^{3N} \rangle.$$

An alternative form to (4.66) that employs the transition operator (4.71) was also presented in [35]. Here we will only focus on (4.66). Appendix D points to notebooks that create codes necessary in the numerical realization of the calculations presented below.

We introduce the total 3N isospin states:

$$\mid \left(t \frac{1}{2} \right) T \rangle,$$

where the isospins of particles 2 and 3 are coupled to the isospin t and then further with the isospin of particle 1 to the total isospin T. Additional constraints on the 3N matrix elements of the 2N potential [35]:

$$\langle \left(t' \frac{1}{2} \right) T' \mid \check{V} \mid \left(t \frac{1}{2} \right) T \rangle = \delta_{t't} \langle \left(t' \frac{1}{2} \right) T' \mid \check{V} \mid \left(t' \frac{1}{2} \right) T \rangle \tag{10.1}$$

and the 3N force:

$$\langle \left(t' \frac{1}{2} \right) T' \mid \check{V}^{(1)} \mid \left(t \frac{1}{2} \right) T \rangle = \delta_{T'T} \langle \left(t' \frac{1}{2} \right) T' \mid \check{V}^{(1)} \mid \left(t \frac{1}{2} \right) T' \rangle \tag{10.2}$$

can be used with (4.66). Constraints can also be put on the operator form of the 3N Faddeev component. Following [34], $\mid \psi^{3N}\rangle$ can be written as:

$$\begin{aligned}\langle \left(t\frac{1}{2}\right) T \mid \psi^{3N}\rangle = \int \mathrm{d}^3\boldsymbol{p}\ \mathrm{d}^3\boldsymbol{q} \sum_{i=1}^{8} \phi_{tT}^{(i)}(|\boldsymbol{p}|, |\boldsymbol{q}|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}}) \\ \mid \boldsymbol{pq}\rangle \otimes \left(\left[\check{O}_i(\boldsymbol{p},\boldsymbol{q})\right]^{\mathrm{3Nspin}} \left[\mid \chi^m\rangle\right]^{\mathrm{3Nspin}} \right),\end{aligned} \tag{10.3}$$

where the Faddeev component is fully determined by the scalar functions ϕ. $\mid \chi^m\rangle = \mid \left(0, \frac{1}{2}\right) \frac{1}{2} m\rangle$ is a 3N spin state in which the spins of particles 2 and 3 are coupled to 0, then coupled with the spin of the first particle to the total 3N spin $\frac{1}{2}$ with the projection m. The eight 3N spin space operators [34] are defined as $(\check{\sigma}(23) = \frac{1}{2}(\check{\sigma}(2) - \check{\sigma}(3)))$:

$$\begin{aligned}
\check{O}_1(\boldsymbol{p},\boldsymbol{q}) &= \check{1} \\
\check{O}_2(\boldsymbol{p},\boldsymbol{q}) &= \frac{1}{\sqrt{3}}\check{\sigma}(23)\cdot\check{\sigma}(1) \\
\check{O}_3(\boldsymbol{p},\boldsymbol{q}) &= \frac{3}{2}\frac{1}{i}\check{\sigma}(1)\cdot(\hat{\boldsymbol{p}}\times\hat{\boldsymbol{q}}) \\
\check{O}_4(\boldsymbol{p},\boldsymbol{q}) &= \frac{1}{\sqrt{2}}\left(i\check{\sigma}(23)\cdot(\hat{\boldsymbol{p}}\times\hat{\boldsymbol{q}}) - (\check{\sigma}(1)\times\check{\sigma}(23))\cdot(\hat{\boldsymbol{p}}\times\hat{\boldsymbol{q}})\right) \\
\check{O}_5(\boldsymbol{p},\boldsymbol{q}) &= \frac{1}{i}\left(\check{\sigma}(23)\cdot(\hat{\boldsymbol{p}}\times\hat{\boldsymbol{q}}) - \frac{1}{i}(\check{\sigma}(1)\times\check{\sigma}(23))\cdot(\hat{\boldsymbol{p}}\times\hat{\boldsymbol{q}})\right) \\
\check{O}_6(\boldsymbol{p},\boldsymbol{q}) &= \sqrt{\frac{3}{2}}\left(\check{\sigma}(23)\cdot\hat{\boldsymbol{p}}\check{\sigma}(1)\cdot\hat{\boldsymbol{p}} - \frac{1}{3}\check{\sigma}(23)\cdot\check{\sigma}(1)\right) \\
\check{O}_7(\boldsymbol{p},\boldsymbol{q}) &= \sqrt{\frac{3}{2}}\left(\check{\sigma}(23)\cdot\hat{\boldsymbol{q}}\check{\sigma}(1)\cdot\hat{\boldsymbol{q}} - \frac{1}{3}\check{\sigma}(23)\cdot\check{\sigma}(1)\right) \\
\check{O}_8(\boldsymbol{p},\boldsymbol{q}) &= \frac{3}{2}\frac{1}{\sqrt{5}}\left(\check{\sigma}(23)\cdot\hat{\boldsymbol{q}}\check{\sigma}(1)\cdot\hat{\boldsymbol{p}} + \check{\sigma}(23)\cdot\hat{\boldsymbol{p}}\check{\sigma}(1)\cdot\hat{\boldsymbol{q}} - \frac{2}{3}\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}\check{\sigma}(23)\cdot\check{\sigma}(1)\right).
\end{aligned} \tag{10.4}$$

Equation (10.1) together with the general form of the 2N potential (and transition operator) from equation (5.8) and the operator form (10.3) can be used to construct a numerical realization of 3N bound state calculations.

All operators in equations (4.66) and (4.71) are linear. Their form allows us to consider parts of these equations separately. We will first consider the action of the permutation operator on the Faddeev component written in the operator form (10.3). The full permutation operator acts simultaneously in the 3N spin, isospin and momentum spaces. Using (10.3) we can write:

$$\begin{aligned}
&\check{P}_{12}\check{P}_{23} \mid \psi^{3N}\rangle = \\
&\int \mathrm{d}^3\boldsymbol{p}\ \mathrm{d}^3\boldsymbol{q} \sum_{tT}\sum_{i=1}^{8} \phi^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) \\
&\check{P}_{12}\check{P}_{23}\left(\mid \boldsymbol{p}\boldsymbol{q}\rangle \otimes \mid \left(t\frac{1}{2}\right) T\rangle \otimes \left(\check{O}_i(\boldsymbol{p},\boldsymbol{q}) \mid \chi^m\rangle\right)\right) = \\
&\int \mathrm{d}^3\boldsymbol{p}\ \mathrm{d}^3\boldsymbol{q} \sum_{tT}\sum_{i=1}^{8} \phi^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) \\
&\left(\check{P}_{12}\check{P}_{23} \mid \boldsymbol{p}\boldsymbol{q}\rangle\right) \otimes \\
&\left[\check{P}_{12}\right]^{3N} \left[\check{P}_{23}\right]^{3N} \left[\check{1} \otimes \check{O}_i(\boldsymbol{p},\boldsymbol{q})\right]^{3N} \left[\mid \left(t\frac{1}{2}\right) T\rangle \otimes \mid \chi^m\rangle\right]^{3N} .
\end{aligned} \tag{10.5}$$

In the above equations all operators except the momentum space permutation operator have a simple matrix representation that can be created using *util1N2N3N.m* (the representation of the permutation operator acting in the isospin-spin space of the three-nucleon system can be created using the Kronecker Product). Momentum space permutations can also easily be worked out for Jacobi momentum eigenstates (this was described at the end of section 4.4) and have an implementation in the *util1N2N3N.m* package. On the other hand the resulting state will have the same operator form as (10.3):

$$\begin{aligned}
&\check{P}_{12}\check{P}_{23} \mid \psi^{3N}\rangle = \\
&\int \mathrm{d}^3\boldsymbol{p}\ \mathrm{d}^3\boldsymbol{q} \sum_{tT}\sum_{i=1}^{8} \beta^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) \\
&\mid \boldsymbol{p}\boldsymbol{q}\rangle \otimes \mid \left(t\frac{1}{2}\right) T\rangle \otimes \left(\check{O}_i(\boldsymbol{p},\boldsymbol{q}) \mid \chi^m\rangle\right) .
\end{aligned} \tag{10.6}$$

Comparing (10.5) and (10.6) and removing the spin and isospin dependencies by projecting from the left with:

$$\sum_m \langle \boldsymbol{p}'\boldsymbol{q}' \mid \otimes \left[\langle \left(t'\frac{1}{2}\right) T' \mid\right]^{\text{3Nisospin}} \otimes \left[\langle \chi^m \mid \check{O}_k(\boldsymbol{p}',\boldsymbol{q}')\right]^{\text{3Nspin}} \tag{10.7}$$

leads to an equation for $\beta^{(i)}_{tT}(|\boldsymbol{p}|, |\boldsymbol{q}|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}})$:

$$\sum_{tT} \sum_{i=1}^{8} \beta^{(i)}_{tT}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}')$$
$$\sum_{m} \left[\langle \left(t' \frac{1}{2} \right) T' \mid \otimes \langle \chi^m \mid \right]^{3N} \left[\check{1} \otimes \check{O}_k(\boldsymbol{p}', \boldsymbol{q}') \right]^{3N}$$
$$\left[\check{1} \otimes \check{O}_i(\boldsymbol{p}', \boldsymbol{q}') \right]^{3N} \left[\mid \left(t \frac{1}{2} \right) T \rangle \otimes \mid \chi^m \rangle \right]^{3N} =$$
$$\sum_{tT} \sum_{i=1}^{8} \phi^{(i)}_{tT}(|\boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}')|, |\boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}')|, \hat{\boldsymbol{P}}^{2312}(\boldsymbol{p}', \boldsymbol{q}') \cdot \hat{\boldsymbol{Q}}^{2312}(\boldsymbol{p}', \boldsymbol{q}'))$$
$$\sum_{m} \left[\langle \left(t' \frac{1}{2} \right) T' \mid \otimes \langle \chi^m \mid \right] \left[\check{1} \otimes \check{O}_k(\boldsymbol{p}', \boldsymbol{q}') \right] \left[\check{P}_{12} \right] \left[\check{P}_{23} \right]$$
$$\left[\check{1} \otimes \check{O}_i(\boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}'), \boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}')) \right] \left[\mid \left(t \frac{1}{2} \right) T \rangle \otimes \mid \chi^m \rangle \right] \quad (10.8)$$

with $\boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}')$ and $\boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}')$ defined as:

$$\check{P}_{23} \check{P}_{12} \mid \boldsymbol{p}' \boldsymbol{q}' \rangle = \mid \boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}') \boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}') \rangle, \quad (10.9)$$

$$\boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}') = -\frac{1}{4}(2\boldsymbol{p}' + 3\boldsymbol{q}'), \quad (10.10)$$

$$\boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}') = \boldsymbol{p}' - \frac{1}{2}\boldsymbol{q}'. \quad (10.11)$$

The expressions above can be used to translate the 3N bound state problem into a linear problem in the space of the scalar functions ϕ, β. Acting with a permutation on the Faddeev component is equivalent to transforming $\phi \to \beta$. Further parts of this chapter provide similar translations of the remaining parts of (4.66). Once all of the translations are complete, the full form of the 3N operaor $\check{A}$ (used in solving the integral 2N and 3N bound state equations in 4.56) can be constructed.

For the sake of readability we can rewrite equation (10.8) in the following form:

$$\sum_{tT} \sum_{i=1}^{8} \beta^{(i)}_{tT}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}') F^{2233}_{t'T'k;tTi}(\boldsymbol{p}', \boldsymbol{q}') =$$
$$\sum_{tT} \sum_{i=1}^{8} \phi^{(i)}_{tT}(|\boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}')|, |\boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}')|, \hat{\boldsymbol{P}}^{2312}(\boldsymbol{p}', \boldsymbol{q}') \cdot \hat{\boldsymbol{Q}}^{2312}(\boldsymbol{p}', \boldsymbol{q}'))$$
$$F^{1223}_{t'T'k;tTi}(\boldsymbol{p}', \boldsymbol{q}') \quad (10.12)$$

where we introduce two new functions:

$$F^{2233}_{t'T'k;tTi}(\boldsymbol{p}', \boldsymbol{q}') =$$
$$\sum_{m} \left[\langle \left(t' \frac{1}{2} \right) T' \mid \otimes \langle \chi^m \mid \right]^{3N} \left[\check{1} \otimes \check{O}_k(\boldsymbol{p}', \boldsymbol{q}') \right]^{3N}$$
$$\left[\check{1} \otimes \check{O}_i(\boldsymbol{p}', \boldsymbol{q}') \right]^{3N} \left[\mid \left(t \frac{1}{2} \right) T \rangle \otimes \mid \chi^m \rangle \right]^{3N} \quad (10.13)$$

$$F^{1223}_{t'T'k;tTi}(\boldsymbol{p'},\boldsymbol{q'}) = \sum_m \left[\langle\left(t'\frac{1}{2}\right)T' \mid \otimes\langle\chi^m \mid\right] \left[\check{1}\otimes\check{O}_k(\boldsymbol{p'},\boldsymbol{q'})\right]\left[\check{P}_{12}\right]\left[\check{P}_{23}\right] \left[\check{1}\otimes\check{O}_i(\boldsymbol{P}^{2312}(\boldsymbol{p'},\boldsymbol{q'}),\boldsymbol{Q}^{2312}(\boldsymbol{p'},\boldsymbol{q'}))\right]\left[\mid\left(t\frac{1}{2}\right)T\rangle\otimes\mid\chi^m\rangle\right] \tag{10.14}$$

Both can be easily calculated and implemented in FORTRAN (see section D.2). In order to arrive at $\beta^{(i)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}')$ we multiply equation (10.12) from the left by the inverse of the matrix of $F^{2233}_{t'T'k;tTi}$ coefficients $Finv$ (where the set $t'T'k$ denotes the row and the set tTi denotes the column):

$$\sum_{t'T'j} Finv^{2233}_{t''T''k;t'T'j}(\boldsymbol{p'},\boldsymbol{q'})F^{2233}_{t'T'j;tTi}(\boldsymbol{p'},\boldsymbol{q'}) = \delta_{t''t}\delta_{T''T}\delta_{ki}. \tag{10.15}$$

Finally we arrive at:

$$\beta^{(k)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') = \sum_{tT}\sum_{i=1}^{8}\phi^{(i)}_{tT}(|\boldsymbol{P}^{2312}(\boldsymbol{p'},\boldsymbol{q'})|,|\boldsymbol{Q}^{2312}(\boldsymbol{p'},\boldsymbol{q'})|,\hat{\boldsymbol{P}}^{2312}(\boldsymbol{p'},\boldsymbol{q'})\cdot\hat{\boldsymbol{Q}}^{2312}(\boldsymbol{p'},\boldsymbol{q'})) C^{1223}_{t'T'k;tTi}(\boldsymbol{p'},\boldsymbol{q'}), \tag{10.16}$$

where the new function is defined to be:

$$C^{1223}_{t'T'k;tTi}(\boldsymbol{p'},\boldsymbol{q'}) = \sum_{t''T''j} Finv^{2233}_{t'T'k;t''T''j}(\boldsymbol{p'},\boldsymbol{q'})F^{1223}_{t''T''j;tTi}(\boldsymbol{p'},\boldsymbol{q'}) \tag{10.17}$$

and again can be easily implemented using *util1N2N3N.m*. Equation (10.16) introduces a linear operator in the space of scalar functions $\phi^{(k)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}')$, $\beta^{(k)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}')$:

$$\left(\check{P}^{\text{scalar}}_{1223}\phi\right)^{(k)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') = \beta^{(k)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') \tag{10.18}$$

The action of the permutation operator on the Faddeev state is equivalent to the action of (10.18) on the scalar function $\phi^{(k)}_{t'T'}(|\boldsymbol{p'}|,|\boldsymbol{q'}|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}')$.

For the $\check{P}_{13}\check{P}_{23}$ case we analogously introduce $C^{1323}_{t'T'k;tTi}(\boldsymbol{p'},\boldsymbol{q'})$. Momentum space permutations will be given by:

$$\check{P}_{23}\check{P}_{13} \mid \boldsymbol{p}'\boldsymbol{q}'\rangle = \mid \boldsymbol{P}^{2313}(\boldsymbol{p}',\boldsymbol{q}')\boldsymbol{Q}^{2313}(\boldsymbol{p}',\boldsymbol{q}')\rangle \tag{10.19}$$

$$\boldsymbol{P}^{2313}(\boldsymbol{p}',\boldsymbol{q}') = -\frac{1}{4}(2\boldsymbol{p}'-3\boldsymbol{q}') \tag{10.20}$$

$$\boldsymbol{Q}^{2313}(\boldsymbol{p}',\boldsymbol{q}') = -\boldsymbol{p}'-\frac{1}{2}\boldsymbol{q}'. \tag{10.21}$$

Repeating the calculations for a different permutation operator we find:

$$\begin{aligned}
&\beta^{(k)}_{t'T'}(|\boldsymbol{p}'|,|\boldsymbol{q}'|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') = \\
&\sum_{tT}\sum_{i=1}^{8}\phi^{(i)}_{tT}(|\boldsymbol{P}^{2313}(\boldsymbol{p}',\boldsymbol{q}')|,|\boldsymbol{Q}^{2313}(\boldsymbol{p}',\boldsymbol{q}')|,\hat{\boldsymbol{P}}^{2313}(\boldsymbol{p}',\boldsymbol{q}')\cdot\hat{\boldsymbol{Q}}^{2313}(\boldsymbol{p}',\boldsymbol{q}')) \\
&C^{1323}_{t'T'k;tTi}(\boldsymbol{p}',\boldsymbol{q}')
\end{aligned} \tag{10.22}$$

and introduce the operator (note that β used here is a general scalar function, not necessarily the result of (10.18)):

$$\left(\check{P}^{\text{scalar}}_{1323}\phi\right)^{(k)}_{t'T'}(|\boldsymbol{p}'|,|\boldsymbol{q}'|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') = \beta^{(k)}_{t'T'}(|\boldsymbol{p}'|,|\boldsymbol{q}'|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}'). \tag{10.23}$$

The method of calculating the action of permutations that was presented is equivalent to the one described in [35]. Here we use a more direct approach and arrive at a slightly different notation. In particular [35] introduces the geometrical factors $F_{t'tT'}$:

$$\langle\left(t'\frac{1}{2}\right)T' \mid \check{P} \mid \left(t\frac{1}{2}\right)T\rangle = \delta_{T'T}F_{t'tT'}\left(\check{P}^{\text{spin}}_{12}\check{P}^{\text{spin}}_{23} + (-1)^{t'+t}\check{P}^{\text{spin}}_{13}\check{P}^{\text{spin}}_{23}\right), \tag{10.24}$$

while here we combine the action of the permutation in the full isospin - spin space. This approach allows us to simplify the calculation without increasing the computational cost of the numerical implementation.

The next operator to consider is the 2N potential $\check{V}$. We will use the decomposition of the general symmetric 2N potential from equation (5.8) and the operator form of the Faddeev state from equation (10.3). Again using the observation that all operators in (4.66) are linear we can consider the this operator separatly and as before try to arrive at the form of an operator acting in the space of scalar functions that define the Faddeev state (10.3). The calculation in this case is simpler since we will not consider momentum space permutations. We start again from the full Faddeev state:

$$\begin{aligned}
&\check{V} \mid \psi^{3N}\rangle = \\
&\int d^3\boldsymbol{p}\, d^3\boldsymbol{q}\sum_{tT}\sum_{i=1}^{8}\phi^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) \\
&\check{V}\left(\mid \boldsymbol{pq}\rangle\otimes\mid\left(t\frac{1}{2}\right)T\rangle\otimes\left(\check{O}_i(\boldsymbol{p},\boldsymbol{q})\mid\chi^m\rangle\right)\right)
\end{aligned} \tag{10.25}$$

Next we use (10.1) together with the observation that the potential acts in the spaces of particles 2 and 3 (we consider the first Faddeev component):

$$\begin{aligned}
&\left(\langle\boldsymbol{p}'\boldsymbol{q}';\left(t'\frac{1}{2}\right)T'\mid\right)\check{V}\left(\mid\boldsymbol{pq};\left(t\frac{1}{2}\right)T\rangle\right) = \\
&\delta^3(\boldsymbol{q}'-\boldsymbol{q})\delta_{tt'}\left(\left[\check{1}\right]^{\text{1Nspin}}\otimes\left[\langle\boldsymbol{p}'\mid\check{V}^{tT'tT}\mid\boldsymbol{p}\rangle\right]^{\text{2Nspin}}\right)
\end{aligned} \tag{10.26}$$

and (5.8) to write the 3N spin operator as:

$$\left[\langle\boldsymbol{p}'\mid\check{V}^{tT'tT}\mid\boldsymbol{p}\rangle\right]^{\text{3Nspin}} = \sum_{i=1}^{6}v_i^{tT'tT}(|\boldsymbol{p}'|,|\boldsymbol{p}|,\hat{\boldsymbol{p}'}\cdot\hat{\boldsymbol{p}})\left[\check{W}_i(\boldsymbol{p}',\boldsymbol{p})\right]^{\text{3Nspin}}. \tag{10.27}$$

We introduced the capital $\check{W}$ operator acting on the spins of particles 2 and 3. Finally, after inserting all this into the original equation, projecting from the left with (10.7) and we arrive at:

$$\begin{gathered}\int \mathrm{d}^3\boldsymbol{p} \sum_{i=1}^{8} \sum_{T} \sum_{j=1}^{6} \phi^{(i)}_{t'T}(|\boldsymbol{p}|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}}') v_j^{t'T't'T}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) \\ \left(\sum_m \left[\langle \chi^m \mid\right] \left[\check{O}_k(\boldsymbol{p}', \boldsymbol{q}')\right] \left[\check{W}_j(\boldsymbol{p}', \boldsymbol{p})\right] \left[\check{O}_i(\boldsymbol{p}, \boldsymbol{q}')\right] \left[\mid \chi^m\rangle\right] \right) = \\ \sum_{i=1}^{8} \left(\sum_m \left[\langle \chi^m \mid\right] \left[\check{O}_k(\boldsymbol{p}', \boldsymbol{q}')\right] \left[\check{O}_i(\boldsymbol{p}', \boldsymbol{q}')\right] \left[\mid \chi^m\rangle\right] \right) \alpha^{(i)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}'), \end{gathered} \tag{10.28}$$

where we assumed that the resulting state will also have the operator form (10.3) with scalar functions $\alpha^{(i)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}')$. With the introduction of two new functions:

$$\begin{gathered} C_{ki}(\boldsymbol{p}, \boldsymbol{q}) = \\ \sum_m \left[\langle \chi^m \mid\right] \left[\check{O}_k(\boldsymbol{p}, \boldsymbol{q})\right] \left[\check{O}_i(\boldsymbol{p}, \boldsymbol{q})\right] \left[\mid \chi^m\rangle\right], \end{gathered} \tag{10.29}$$

$$\begin{gathered} L_{kji}(\boldsymbol{p}, \boldsymbol{q}; \boldsymbol{p}, \boldsymbol{p}'; \boldsymbol{p}', \boldsymbol{q}) = \\ \sum_m \left[\langle \chi^m \mid\right] \left[\check{O}_k(\boldsymbol{p}, \boldsymbol{q})\right] \left[\check{W}_j(\boldsymbol{p}\boldsymbol{p}')\right] \left[\check{O}_i(\boldsymbol{p}', \boldsymbol{q})\right] \left[\mid \chi^m\rangle\right] \end{gathered} \tag{10.30}$$

and additionally $C^{-1}_{ki}(\boldsymbol{p}, \boldsymbol{q})$ such that

$$\sum_l C^{-1}_{kl}(\boldsymbol{p}, \boldsymbol{q}) C_{li}(\boldsymbol{p}, \boldsymbol{q}) = \delta_{ki}, \tag{10.31}$$

equation (10.28) can be rewritten as:

$$\begin{gathered} \alpha^{(k)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}') = \\ \int \mathrm{d}^3\boldsymbol{p} \sum_{i=1}^{8} \sum_{T} \sum_{j=1}^{6} \phi^{(i)}_{t'T}(|\boldsymbol{p}|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}}') v_j^{t'T't'T}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) \\ \sum_{l=1}^{8} C^{-1}_{kl}(\boldsymbol{p}', \boldsymbol{q}') L_{lji}(\boldsymbol{p}', \boldsymbol{q}'; \boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}, \boldsymbol{q}') \equiv \\ \int \mathrm{d}^3\boldsymbol{p} \sum_{i=1}^{8} \sum_{T} \sum_{j=1}^{6} \phi^{(i)}_{t'T}(|\boldsymbol{p}|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}}') v_j^{t'T't'T}(|\boldsymbol{p}'|, |\boldsymbol{p}|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{p}}) \\ (C^{-1}L)_{kji}(\boldsymbol{p}', \boldsymbol{q}'; \boldsymbol{p}', \boldsymbol{p}; \boldsymbol{p}, \boldsymbol{q}'). \end{gathered} \tag{10.32}$$

Equation (10.32) again introduces a linear operator in the space of scalar functions $\phi^{(k)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}')$, $\alpha^{(k)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}')$:

$$\left(\check{V}^{\mathrm{scalar}}\phi\right)^{(k)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}') = \alpha^{(k)}_{t'T'}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}'). \tag{10.33}$$

The action of the 2N potential on the Faddeev component is equivalent to acting with this operator on the scalar functions that define the Faddeev state in the operator form. Further simplification of the calculation can be achieved by using the following parametrization of vectors in (10.32):

$$\begin{aligned} \boldsymbol{q}' &= |\boldsymbol{q}'|(0,0,1), \\ \boldsymbol{p}' &= |\boldsymbol{p}'|(\sqrt{1-x'^2},0,x'), \\ \boldsymbol{p} &= |\boldsymbol{p}|(\sqrt{1-x^2}cos\phi, \sqrt{1-x^2}sin\phi, x). \end{aligned} \tag{10.34}$$

This transforms the form of the integral in (10.32) to:

$$\int \mathrm{d}^3\boldsymbol{p} \rightarrow \int_0^\infty \mathrm{d}|\boldsymbol{p}||\boldsymbol{p}|^2 \int_0^{2\pi} \mathrm{d}\phi \int_{-1}^{1} \mathrm{d}x \tag{10.35}$$

Using (10.34) the argument of $\phi^{(i)}_{t'T}(|\boldsymbol{p}|,|\boldsymbol{q}'|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}')$ depends only on $|\boldsymbol{p}|$, $|\boldsymbol{q}'|$ and x'. This allows us to perform the integration over ϕ and the summation over j beforehand and store the results in a table for further use.

We now consider the 3N force. This is the most numerically demanding part of the calculation as it will introduce a six-fold integral. Additional difficulties arise from the fact that no integrals or sums can be prepared beforehand for use in further iterations. We will again start with the general expression:

$$\begin{aligned} &\check{V}^{(1)} \mid \psi^{3N}\rangle = \\ &\int \mathrm{d}^3\boldsymbol{p}\mathrm{d}^3\boldsymbol{q} \sum_{tT}\sum_{i=1}^{8} \phi^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) \\ &\check{V}^{(1)}\left(\mid \boldsymbol{pq}\rangle \otimes \mid \left(t\frac{1}{2}\right)T\rangle \otimes \left(\check{O}_i(\boldsymbol{p},\boldsymbol{q}) \mid \chi^m\rangle\right)\right). \end{aligned} \tag{10.36}$$

In principle the three-nucleon force also has a decomposition similar to that of the 2N potential [23]:

$$\begin{aligned} &\langle \boldsymbol{p}'\boldsymbol{q}'; (t'\frac{1}{2})T' \mid \check{V}^{(1)} \mid \boldsymbol{pq}; (t\frac{1}{2})T\rangle = \\ &\delta_{T'T}\sum_l v^l_{t'tT'}(\boldsymbol{p}'\boldsymbol{q}';\boldsymbol{pq})\left[\check{\Omega}(\boldsymbol{p}'\boldsymbol{q}';\boldsymbol{pq})\right]^{3Nspin} \end{aligned} \tag{10.37}$$

but with our representation of 3N spin states and operators it is simpler to consider the 3N operator as a whole, especially because we consider only the case of the N2LO chiral potential (see eg. [37, 43]) with very few 3N spin structures.

After using (10.2) and comparing the result with (10.3) we get an equation for the scalar functions γ after the action of the 3N potential:

$$\begin{aligned}
&\int \mathrm{d}^3\boldsymbol{p} \int \mathrm{d}^3\boldsymbol{q} \sum_{i=1}^{8} \sum_{tT} \gamma_{tT}^{(i)}(|\boldsymbol{p}|, |\boldsymbol{q}|, \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}}) \\
&\mid \boldsymbol{p}\boldsymbol{q}\rangle \otimes \mid (t\frac{1}{2})T\rangle \otimes \left[\check{O}_i(\boldsymbol{p},\boldsymbol{q})\right] [\mid \chi^m\rangle] = \\
&\int \mathrm{d}^3\boldsymbol{p} \int \mathrm{d}^3\boldsymbol{q} \sum_{i=1}^{8} \sum_{tT} \int \mathrm{d}^3\boldsymbol{p}' \int \mathrm{d}^3\boldsymbol{q}' \sum_{t'} \\
&\phi_{t'T}^{(i)}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}') \mid \boldsymbol{p}\boldsymbol{q}\rangle \otimes \mid (t\frac{1}{2})T\rangle \\
&\left[\langle \boldsymbol{p}\boldsymbol{q}; (t\frac{1}{2})T \mid \check{V}^{(1)} \mid \boldsymbol{p}'\boldsymbol{q}'; (t'\frac{1}{2})T\rangle\right] \left[\check{O}_i(\boldsymbol{p}',\boldsymbol{q}')\right] [\mid \chi^m\rangle]\,.
\end{aligned} \tag{10.38}$$

The next step is projecting (10.38) from the left with (10.7) to get:

$$\begin{aligned}
&\sum_{i=1}^{8} \gamma_{t'T'}^{(i)}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}') \\
&\sum_m [\langle \chi^m \mid] \left[\check{O}_k(\boldsymbol{p}',\boldsymbol{q}')\right] \left[\check{O}_i(\boldsymbol{p}',\boldsymbol{q}')\right] [\mid \chi^m\rangle] = \\
&\sum_{i=1}^{8} \int \mathrm{d}^3\boldsymbol{p}'' \int \mathrm{d}^3\boldsymbol{q}'' \sum_{t''} \\
&\phi_{t''T'}^{(i)}(|\boldsymbol{p}''|, |\boldsymbol{q}''|, \hat{\boldsymbol{p}}'' \cdot \hat{\boldsymbol{q}}'') \sum_m [\langle \chi^m \mid] \left[\check{O}_k(\boldsymbol{p}',\boldsymbol{q}')\right] \\
&\left[\langle \boldsymbol{p}'\boldsymbol{q}'; (t'\frac{1}{2})T' \mid \check{V}^{(1)} \mid \boldsymbol{p}''\boldsymbol{q}''; (t''\frac{1}{2})T'\rangle\right] \left[\check{O}_i(\boldsymbol{p}'',\boldsymbol{q}'')\right] [\mid \chi^m\rangle]\,.
\end{aligned} \tag{10.39}$$

After the introduction of a new scalar function:

$$\begin{aligned}
&E_{ki}^{tt'T}(\boldsymbol{p}\boldsymbol{q}; \boldsymbol{p}\boldsymbol{q}\boldsymbol{p}'\boldsymbol{q}'; \boldsymbol{p}'\boldsymbol{q}') = \sum_m [\langle \chi^m \mid] \left[\check{O}_k(\boldsymbol{p},\boldsymbol{q})\right] \\
&\left[\langle \boldsymbol{p}\boldsymbol{q}; (t\frac{1}{2})T \mid \check{V}^{(1)} \mid \boldsymbol{p}'\boldsymbol{q}'; (t'\frac{1}{2})T\rangle\right] \left[\check{O}_i(\boldsymbol{p}',\boldsymbol{q}')\right] [\mid \chi^m\rangle]
\end{aligned} \tag{10.40}$$

equation (10.39) turns into:

$$\begin{aligned}
&\sum_{i=1}^{8} \gamma_{t'T'}^{(i)}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}') C_{ki}(\boldsymbol{p}',\boldsymbol{q}') = \\
&\sum_{i=1}^{8} \int \mathrm{d}^3\boldsymbol{p}'' \int \mathrm{d}^3\boldsymbol{q}'' \sum_{t''} \phi_{t''T'}^{(i)}(|\boldsymbol{p}''|, |\boldsymbol{q}''|, \hat{\boldsymbol{p}}'' \cdot \hat{\boldsymbol{q}}'') \\
&E_{ki}^{t't''T'}(\boldsymbol{p}'\boldsymbol{q}'; \boldsymbol{p}'\boldsymbol{q}'\boldsymbol{p}''\boldsymbol{q}''; \boldsymbol{p}''\boldsymbol{q}'')
\end{aligned} \tag{10.41}$$

or using (10.31) we can write:

$$\begin{aligned}
&\gamma^{(k)}_{t'T'}(|\boldsymbol{p}'|,|\boldsymbol{q}'|,\hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') = \\
&\sum_{i=1}^{8}\int d^3\boldsymbol{p}''\int d^3\boldsymbol{q}''\sum_{t''}\phi^{(i)}_{t''T'}(|\boldsymbol{p}''|,|\boldsymbol{q}''|,\hat{\boldsymbol{p}}''\cdot\hat{\boldsymbol{q}}'') \\
&\sum_{r=1}^{8}C^{-1}_{kr}(\boldsymbol{p}',\boldsymbol{q}')E^{t't''T'}_{ri}(\boldsymbol{p}'\boldsymbol{q}';\boldsymbol{p}'\boldsymbol{q}'\boldsymbol{p}''\boldsymbol{q}'';\boldsymbol{p}''\boldsymbol{q}'') \equiv \\
&\sum_{i=1}^{8}\int d^3\boldsymbol{p}''\int d^3\boldsymbol{q}''\sum_{t''}\phi^{(i)}_{t''T'}(|\boldsymbol{p}''|,|\boldsymbol{q}''|,\hat{\boldsymbol{p}}''\cdot\hat{\boldsymbol{q}}'')(C^{-1}E)^{t't''T'}_{ki}(\boldsymbol{p}'\boldsymbol{q}'\boldsymbol{p}'\boldsymbol{q}'\boldsymbol{p}''\boldsymbol{q}'';\boldsymbol{p}''\boldsymbol{q}'').
\end{aligned} \tag{10.42}$$

This equation introduces yet another linear operator acting in the space of scalar functions that define the Faddeev state. More precisely:

$$\begin{aligned}
&(\check{V}^{(1)\text{scalar}}\phi)^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) = \\
&\sum_{i=1}^{8}\int d^3\boldsymbol{p}\int d^3\boldsymbol{q}\sum_{t'}\phi^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) \\
&\sum_{r=1}^{8}C^{-1}_{kr}(\boldsymbol{p},\boldsymbol{q})E^{ttT}_{ri}(\boldsymbol{p}\boldsymbol{q};\boldsymbol{p}\boldsymbol{q}\boldsymbol{p}\boldsymbol{q};\boldsymbol{p}\boldsymbol{q}) \equiv \\
&\sum_{i=1}^{8}\int d^3\boldsymbol{p}\int d^3\boldsymbol{q}\sum_{t'}\phi^{(i)}_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,\hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}})(C^{-1}E)^{ttT}_{ri}(\boldsymbol{p}\boldsymbol{q};\boldsymbol{p}\boldsymbol{q}\boldsymbol{p}\boldsymbol{q};\boldsymbol{p}\boldsymbol{q}).
\end{aligned} \tag{10.43}$$

Putting everything together requires constructing:

$$\begin{aligned}
K\check{(E)} = {}& \check{G}_0(E)\check{V}^{\text{scalar}}\left(\check{1}^{\text{scalar}} + \check{P}^{\text{scalar}}_{1223} + \check{P}^{\text{scalar}}_{1323}\right) + \\
& \check{G}_0(E)\check{V}^{(1)\text{scalar}}\left(\check{1}^{\text{scalar}} + \check{P}^{\text{scalar}}_{1223} + \check{P}^{\text{scalar}}_{1323}\right)
\end{aligned} \tag{10.44}$$

from (10.18) , (10.23) , (10.33), (10.43) and the free propagator (4.43). After this operator is constructed (parts of the operator can be applied separately), in order to find the bound state we solve equation (4.56) with $\check{A}(E) = \check{K}(E)$:

$$\boxed{\check{K}(E)\phi = \phi}. \tag{10.45}$$

Results agree well with classical partial wave calculations. We use chiral NNLO 2N and 3N potentials (see eg. [37, 43]). Results for the total binding energy, 2N and 3N kinetic and potential energy expectation values are shown in Table 10.1. Again the three-dimensional results agree very well with the PWD predictions. The effect of the 3N force on the total binding energy is around 9%.

A selected set of slices through the scalar functions that define the full wave function of the three-nucleon system are shown in Figures 10.1 and 10.2. The plots show the $|\boldsymbol{p}|$ and $|\boldsymbol{q}|$ dependence of $\beta^i_{tT}(|\boldsymbol{p}|,|\boldsymbol{q}|,x = \hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}})$ for different values of t and i.

	PWD	3D
λ	1.0	0.99976
$< E_{\text{kin}} >$	33.448	33.412
$< E_{\text{pot}}^{\text{2N}} >$	-41.329	-41.273
$< E_{\text{pot}}^{\text{3N}} >$	-0.765	- 0.770
total energy	-8.646	-8.631

Table 10.1: Expectation values (in $[MeV]$) of different energy operators and λ for the triton. The first column contains the results using partial waves and the second column contains results for the new three dimensional approach. λ is the eigenvalue calculated in equation (4.56) with $A = K$ from (10.44). After calculating the bound state, expectation values for the kinetic, 2N and 3N potential energy can be calculated separately. In our triton calculations we do not allow charge independence breaking. The up to date experimental value for the triton binding energy is $8.481821[MeV]$ [57]. Results from our paper [35].

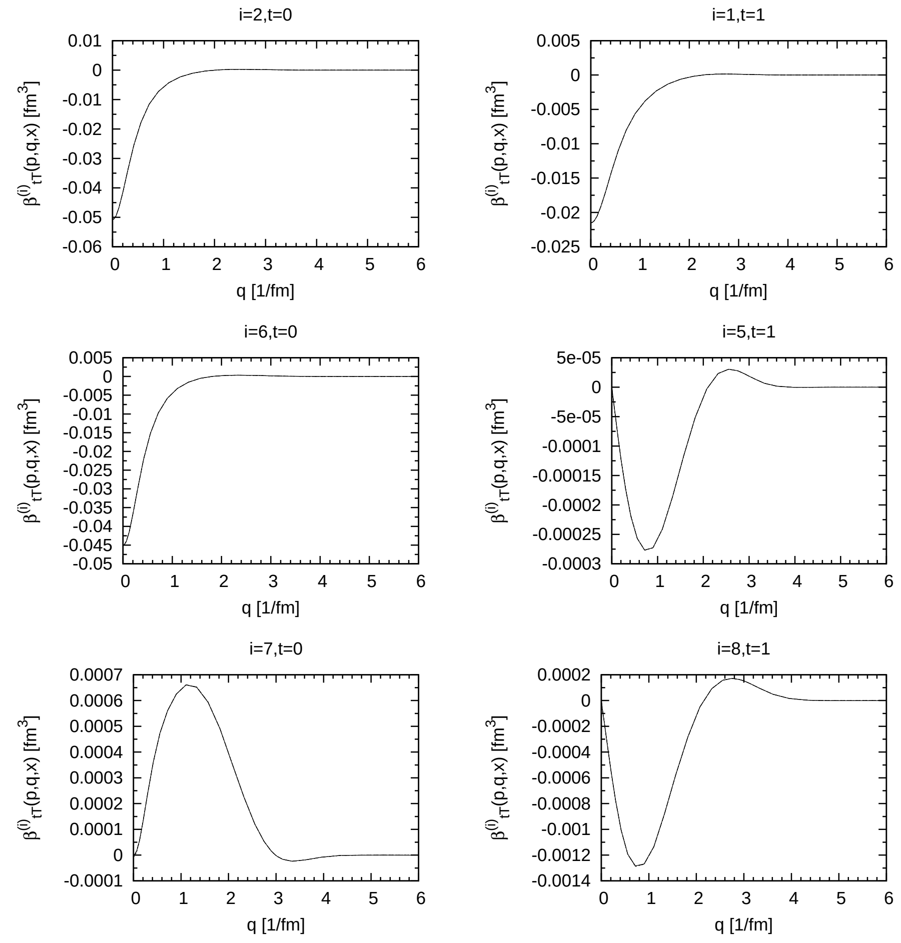


Figure 10.1: Slices through a few selected scalar functions $\beta^i_{tT}(|\boldsymbol{p}|, |\boldsymbol{q}|, x = \hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{q}})$ that describe full 3N bound state are shown. The presented fragments are a function of $|\boldsymbol{q}|$ for $|\boldsymbol{p}| \approx 1.4[fm^{-1}]$, $T = \frac{1}{2}$ and $x \approx 0.55$. Reprinted results from our paper [35]. This paper presents additional iteration schemes that are not discussed in this text but as can be seen all results overlap.

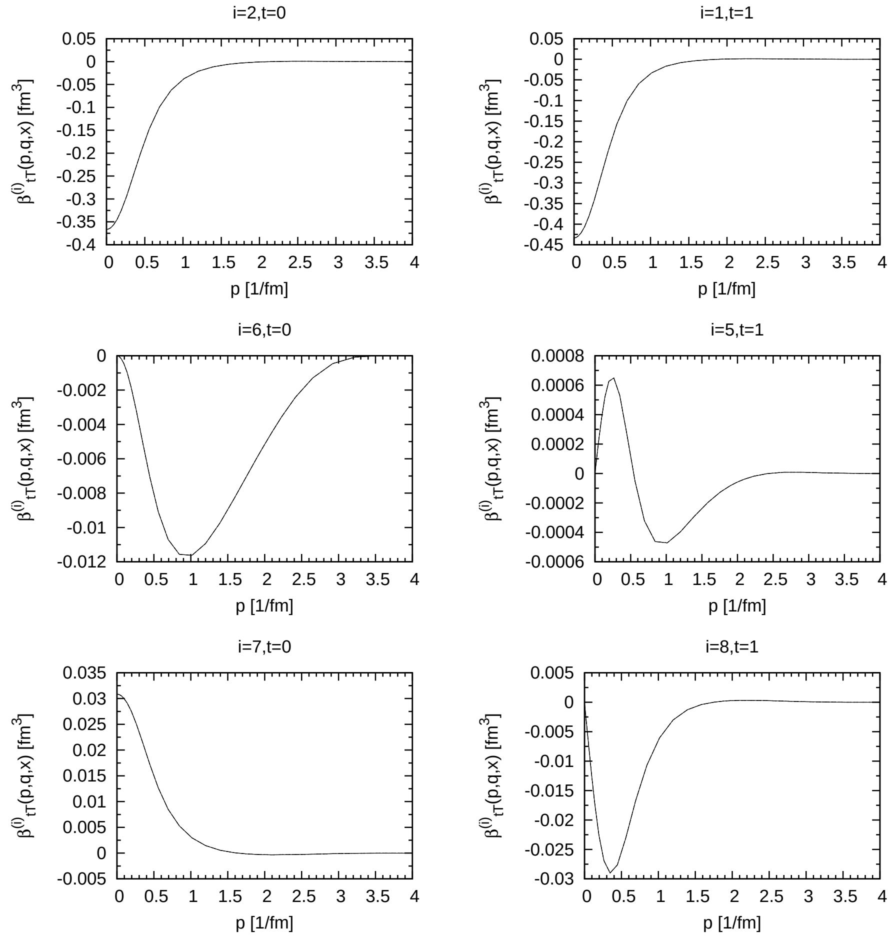


Figure 10.2: The same as in Figure 10.1 but the presented fragments are a function of $|\boldsymbol{p}|$ for $|\boldsymbol{q}| \approx 0.72[fm^{-1}]$, $T = \frac{1}{2}$ and $x \approx -0.61$

Chapter 11

Summary and Outlook

Using the three-dimensional approach (4.9), (4.26) and the isospin formalism to describe the nuclear systems made the path from fundamental equations to practical numerical realizations very direct. This direct path, apart from serving as a beautiful reminder of the roots of our calculations, greatly speeds up the process of adapting our framework to new models. This property makes three dimensional calculations an attractive tool that can be used to test new models. This is especially apparent for higher energies, where the algebra of partial wave projected states and operators becomes increasingly complicated and the problem of convergence emerges.

The text started with an elementary description of the degrees of freedom of 2N and 3N systems in Chapter 4. There we also tried to give a, possibly oversimplified, derivation of the equations that were the subject of our calculations in further chapters. Chapter 4 thus contained a short introduction to the Lippmann-Schwinger equation for the transition operator and the equations for the 2N and 3N bound states. We also gave a description of our treatment of basic 2N and 3N operators. This treatment can be directly implemented in the *util1N2N3N.m Mathematica*$^{®}$ package (Appendix B).

We start our discussions in Chapter 6 with the simple problem - the deuteron wave function. This toy-problem was used to test our Krylov methods and to compare the results of different methods. Our final 3D results are in an excellent agreement with PWD calculations. The deuteron wave function is calculated using a very general form of the 2N force as was published in [15].

In Chapter 7 we presented our approach to calculating the 2N transition operator where we employed a very general form of the 2N potential described in Chapter 5. Chapter 7 considered both the positive and the negative energy case for the transition operator. For positive energies the method described in this text, among others, was published in [20] and can be used to calculate all nucleon-nucleon scattering observables. For the negative energy case, we presented our treatment of the bound-state singularity (at the deuteron binding energy and the isospin 0 case). Our calculation of the residue involves using the deuteron wave function calculated earlier in Chapter 6. This handling of the deuteron residue is necessary for our future calculations of 3N scattering.

Our description of deuteron electro-disintegration given in Chapter 8 was published in [21]. Because the three-dimensional approach employs all partial waves it was possible to recreate PWD results in only one run. The standard

PWD approach however, required many partial waves to be taken into account. This is clearly the case for large momentum transfers to the 2N system as can be observed in the included figures. With a sufficient number of partial-waves, observables calculated using PWD agree very well with the three-dimensional approach.

The formalism, that was developed in Chapter 8 has further applications in the description of muon induced deuteron disintegration. This reaction was the subject of Chapter 9 and is treated as the decay of the muonic atom (with the muon on the K shell) using the Fermi approximation. Results for the total and differential decay rates, calculated using the three-dimensional and PWD approach, agree very well. Although these calculations are not yet published, we managed to compare them with the work of other research groups and found good agreement. We believe that essentially any reaction involving the 2N system (neutrino induced deuteron breakup, pion absorption on $^2H, \ldots$) can be efficiently treated in our three-dimensional formalism.

In Chapter 10 we describe our calculation of the 3N bound state. Our formalism can, not only, employ a very general form of the 2N force (see for example Chapter 5) but additionally a general operator form of the 3N potential. Our results from this chapter were published in [35] and show the effect of the additional 3N force on the bound state energy of the triton.

We would like to emphasize that in all the discussed cases we managed to get a very good agreement with standard PWD calculations.

Natural extensions of the formalisms developed in Chapters 8,9,10 are possible. For example we plan to calculate the ^3He wave function, which would require incorporating the Coulomb proton-proton interaction. This might lead to a more accurate ^3He wave function than those currently available (based on PWD). Later we could treat $\mu +^3 He \rightarrow \nu_\mu +^3 H$ reaction and the β decay of the triton, employing also various models of 2N current operators.

Our formalism can also be used in the partial wave decomposition of new potentials [58, 59, 60]. This was not explored in this text but section E.5 contains a brief description of partial wave states and their relation to the 3D formalism described in this thesis. The resulting complicated algebraic expressions resulting from the isospin and spin matrix elements of operators can also be treated in a consistent way with the use of our *Mathematica*® tools.

In this thesis we put a lot of effort to make the expressions introduced in the previous chapters easily translatable to numerical codes. The following chapters are marked as APPENDIX but are an integral part of the thesis. They contain detailed information on the numerical realization of our calculations. We start in Appendix A by giving a list of all operators that were intruded earlier in the text. With the help of the notation from Chapter 3 we hope that creating a numerical implementation of these expressions should be straightforward. The only components that are needed for the code are (occasionally) interpolations and implementations of the complicated algebraic expressions that result from isospin - spin matrix elements. Creating these expressions is made simple with the use of the *util1N2N3N.m* and *FunctionArra.m* packages that were created for use with this thesis and are described in Appendixes B and C. Appendixes B and C are very *Mathematica*® oriented but require only a very basic understanding of the *Mathematica*® syntax. We hope that the tutorials presented there will be easy to follow. Finally, in Appendix D we document the FORTRAN codes that was created using these two packages and attached to the text.

Appendix A

Numerical methods

A.1 Dealing with large linear operators

In the previous chapters we encountered linear equations for the bound state of the 2N, 3N system and the transition operator. All these equations are constructed from linear operators, that (except for the 2N bound state) have a very large dimension. A crucial step towards solving these equations was to develop methods of dealing with these large operators. Below we give a list of all linear operators that are used in our calculations and the dimension of the problem assuming that all real arguments of the scalar functions are discretized on a grid of 40 points.

- Deuteron operator from (6.9):

$$
\begin{aligned}
&\left(\check{K}^d(E_d)\phi\right)_q(|\boldsymbol{p}|) = \\
&\frac{1}{E_d - \frac{\boldsymbol{p}^2}{m}} \int \mathrm{d}^3\boldsymbol{p}' \sum_{j=1}^{6} v_j^{00}(\boldsymbol{p},\boldsymbol{p}') \\
&\sum_{k''=1}^{2} \left(\sum_k \left(A^d(\boldsymbol{p})\right)^{-1}_{qk} B^d_{kjk''}(\boldsymbol{p},\boldsymbol{p}') \right) \phi_{k''}(|\boldsymbol{p}'|)
\end{aligned}
$$

$$\boxed{\text{80 dimensional vectors and } 80 \times 80 \text{ dimensional operators}}$$

- Transition operator kernel from (7.15):

$$
\begin{aligned}
&\left(\check{\mathcal{B}}t\right)_k^{\{\gamma\}} \left({\scriptstyle \{E\}}\,, |\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x'\right) = \\
&\int_0^{+\infty} \mathrm{d}|\boldsymbol{p}''| \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \sum_{j=1}^{6} \sum_{j'=1}^{6} \frac{|\boldsymbol{p}''|^2}{{\scriptstyle \{E\}} - \frac{|\boldsymbol{p}''|^2}{m} + i\epsilon} \\
&v_j^{\{\gamma\}}\left(|\boldsymbol{p}'|, |\boldsymbol{p}''|, \sqrt{1-x'^2}\sqrt{1-x''^2}\cos\phi'' + x'x''\right) \\
&\mathcal{B}_{kjj'}(|\boldsymbol{p}'|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x', |\boldsymbol{p}''|, x'', \phi'') \\
&t_{j'}^{\{\gamma\}}\left({\scriptstyle \{E\}}\,, |\boldsymbol{p}''|, {\scriptstyle \{|\boldsymbol{p}|\}}\,, x''\right)
\end{aligned}
$$

transition operator $\check{f}$ from (7.21):

$$(\check{f}(|\boldsymbol{p}''|)t)_k^{\{\gamma\}}\left({}_{\{E\}}, |\boldsymbol{p}'|, {}_{\{|\boldsymbol{p}|\}}, x'\right) =$$
$$m \int_{-1}^{1} \mathrm{d}x'' \int_0^{2\pi} \mathrm{d}\phi'' \sum_{j=1}^{6} \sum_{j'=1}^{6}$$
$$v_j^{\{\gamma\}}\left(|\boldsymbol{p}'|, |\boldsymbol{p}''|, \sqrt{1-x'^2}\sqrt{1-x''^2}\cos\phi'' + x'x''\right)$$
$$\mathcal{B}_{kjj'}(|\boldsymbol{p}'|, {}_{\{|\boldsymbol{p}|\}}, x', |\boldsymbol{p}''|, x'', \phi'')$$
$$t_{j'}^{\{\gamma\}}\left({}_{\{E\}}, |\boldsymbol{p}''|, {}_{\{|\boldsymbol{p}|\}}, x''\right)$$

$$\boxed{\text{For a single energy - 160 independent equations of dimension } 9600 \times 9600}$$

- Operators from 3N bound state calculations. Permutation operators from equation (10.16) , (10.22):

$$\left(\check{P}_{1223}^{\text{scalar}}\phi\right)_{t'T'}^{(k)}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') =$$
$$\sum_{tT}\sum_{i=1}^{8} \phi_{tT}^{(i)}(|\boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}')|, |\boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}')|, \hat{\boldsymbol{P}}^{2312}(\boldsymbol{p}', \boldsymbol{q}')\cdot\hat{\boldsymbol{Q}}^{2312}(\boldsymbol{p}', \boldsymbol{q}'))$$
$$C_{t'T'k;tTi}^{1223}(\boldsymbol{p}', \boldsymbol{q}')$$

$$\left(\check{P}_{1323}^{\text{scalar}}\phi\right)_{t'T'}^{(k)}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') =$$
$$\sum_{tT}\sum_{i=1}^{8} \phi_{tT}^{(i)}(|\boldsymbol{P}^{2313}(\boldsymbol{p}', \boldsymbol{q}')|, |\boldsymbol{Q}^{2313}(\boldsymbol{p}', \boldsymbol{q}')|, \hat{\boldsymbol{P}}^{2313}(\boldsymbol{p}', \boldsymbol{q}')\cdot\hat{\boldsymbol{Q}}^{2313}(\boldsymbol{p}', \boldsymbol{q}'))$$
$$C_{t'T'k;tTi}^{1323}(\boldsymbol{p}', \boldsymbol{q}')$$

The 2N potential operator from equation (10.32):

$$\left(\check{V}^{\text{scalar}}\phi\right)_{t'T'}^{(k)}(|\boldsymbol{p}'|, |\boldsymbol{q}'|, \hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}') =$$
$$\int \mathrm{d}^3\boldsymbol{p}' \sum_{i=1}^{8}\sum_{T}\sum_{j=1}^{6} \phi_{tT'}^{(i)}(|\boldsymbol{p}'|, |\boldsymbol{q}|, \hat{\boldsymbol{p}}'\cdot\hat{\boldsymbol{q}}) v_j^{tTtT'}(|\boldsymbol{p}|, |\boldsymbol{p}'|, \hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{p}}')$$
$$(C^{-1}L)_{kji}(\boldsymbol{p}, \boldsymbol{q}; \boldsymbol{p}, \boldsymbol{p}'; \boldsymbol{p}', \boldsymbol{q}).$$

The 3N potential operator from equation (10.43):

$$(\check{V}^{(1)\text{scalar}}\phi)_{tT}^{(i)}(|\boldsymbol{p}|, |\boldsymbol{q}|, \hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) =$$
$$\sum_{i=1}^{8} \int \mathrm{d}^3\boldsymbol{p} \int \mathrm{d}^3\boldsymbol{q} \sum_{t'} \phi_{tT}^{(i)}(|\boldsymbol{p}|, |\boldsymbol{q}|, \hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}})$$
$$\sum_{r=1}^{8} C_{kr}^{-1}(\boldsymbol{p}, \boldsymbol{q}) E_{ri}^{ttT}(\boldsymbol{pq}; \boldsymbol{pqpq}; \boldsymbol{pq}) \equiv$$
$$\sum_{i=1}^{8} \int \mathrm{d}^3\boldsymbol{p} \int \mathrm{d}^3\boldsymbol{q} \sum_{t'} \phi_{tT}^{(i)}(|\boldsymbol{p}|, |\boldsymbol{q}|, \hat{\boldsymbol{p}}\cdot\hat{\boldsymbol{q}}) (C^{-1}E)_{ki}^{ttT}(\boldsymbol{pq}; \boldsymbol{pqpq}; \boldsymbol{pq})$$

$$\boxed{2048000 \text{ dimensional vectors and } 2048000 \times 2048000 \text{ dimensional operators}}$$

Functions $\left(A^d(\boldsymbol{p})\right)^{-1}_{qk}$ **,** $B^d_{kjk''}(\boldsymbol{p},\boldsymbol{p}')$ **,** $\mathcal{B}_{kjj'}(|\boldsymbol{p}'|,_{\{|\boldsymbol{p}|\}},x',|\boldsymbol{p}''|,x'',\phi'')$ **,** $C^{1223}_{t'T'k;tTi}(\boldsymbol{p}',\boldsymbol{q}')$ **,** $C^{1323}_{t'T'k;tTi}(\boldsymbol{p}',\boldsymbol{q}')$ **,** $(C^{-1}L)_{kji}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{p},\boldsymbol{p}';\boldsymbol{p}',\boldsymbol{q})$ **and** $(C^{-1}E)^{ttT}_{ki}(\boldsymbol{pq};\boldsymbol{pqpq};\boldsymbol{pq})$ **were discussed earlier in the text and their implementation is available with the use of the *util1N2N3N.m* and *FunctionArray.m* packages discussed in Appendixes B, C. Notebooks creating the implementations are discussed in Appendix D.**

It is clear from the form of these equations that we will have to deal with large linear systems. This is especially apparent in the 3N bound state case. There is currently no possibility to store 2048000×2048000 dimensional matrices on a single PC. Moreover, the explicit form of the matrices involved in our computations is known only for a limited number of cases. Our methods of dealing with these large equations do not require the knowledge of the matrix representation. Our numerical realization requires only the implementation of the action of appropriate operators onto vectors and some appropriately chosen scalar product.

The calculations are divided into two steps. In the first step we find a subspace of the domain of the large linear operator that is involved in our equations. This subspace is ideally of small dimension (around 40) and contains eigenvalues with the largest absolute values of the linear operator (the matrix representation of the large linear operator in this subspace is calculated simultaneously). In the next step we reconstruct the original equation in this small subspace and solve it using classical methods.

The basic scheme for finding the subspace is as follows. First we start with some arbitrary vector $\boldsymbol{q}_1$, next using our implementation of the action of the large linear operator $\breve{A}$ on a vector we calculate $\breve{A}\boldsymbol{q}_1, \breve{A}\breve{A}\boldsymbol{q}_1, \breve{A}\breve{A}\breve{A}\boldsymbol{q}_1, \ldots$ (this iteration scheme can be different in practical applications, we will typically use the Arnoldi algorithm). From this set of vectors we calculate a new set $\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, \ldots$ of orthogonal vectors that spans the same space

$$span(\boldsymbol{q}_1, \breve{A}\boldsymbol{q}_1, \breve{A}\breve{A}\boldsymbol{q}_1, \breve{A}\breve{A}\breve{A}\boldsymbol{q}_1, \ldots) = span(\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, \ldots) \tag{A.1}$$

and the representation of $\breve{A}$ in this space. It can be deduced from the form of our iterations that, at each step, vector components proportional to the largest eigenvalues of $\breve{A}$ will be amplified. In real applications we use a slightly more sophisticated iteration method - Arnoldi iteration, it will be described in detail in section A.2. The subspace resulting from Arnoldi iterations can be proved to be identical to $span(\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, \ldots)$ but the algorithm is more numerically stable.

A.2 Arnoldi iteration scheme

This section gives a description of the Arnoldi iteration procedure that was successfully used in our calculations. The following simple scheme was fully sufficient for our applications, more advanced Krylov methods are however available, see for example [61]. Let v be some initial vector with norm 1, $\breve{A}$ be a general linear operator acting in the space of v and N be the number of iterations. The Arnoldi algorithm is listed in Program 1.

Program 1 Pseudo - code for Arnoldi algorithm.

```
1     q[1] <- v
2     DO k = 2 , N
3        q[k] <- A q[k - 1]
4        DO j = 1 , k - 1
5           h[j , k - 1] <- (q[j] , q[k])
6           q[k] <- q[k] - h[j , k - 1] q[j]
7        END DO
8        h[k , k - 1] = |q[k]|
9        q[k] = q[k] / h[k , k - 1]
10    END DO
```

In line `3` of Program 1 the action of linear operator `A` on vector `q[k - 1]` is performed and stored in `q[k]`. In line `5` the scalar product is computed and in line `8` the norm calculated.

The result of the algorithm is a $N \times N - 1$ matrix $\left[\check{h}\right]$ with elements `h[i , j]` (h_{ij}) and a set of vectors $\{\boldsymbol{q}_i\}$ (`q[i]`) $i = 1, ..., N$. It can be easily seen that the produced vectors are orthogonal. Following the algorithm and remembering about the normalization in line `9` of Program 1, the first four vectors are:

$$\begin{aligned}
\boldsymbol{q}_1 &= v \\
\boldsymbol{q}_2 &= \breve{A}\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_1)\hat{\boldsymbol{q}}_1 \\
\boldsymbol{q}_3 &= \breve{A}\hat{\boldsymbol{q}}_2 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_2, \breve{A}\hat{\boldsymbol{q}}_2 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_1)\hat{\boldsymbol{q}}_2 \\
\boldsymbol{q}_4 &= \breve{A}\hat{\boldsymbol{q}}_3 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_2, \breve{A}\hat{\boldsymbol{q}}_3 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_1)\hat{\boldsymbol{q}}_2 \\
&- (\hat{\boldsymbol{q}}_3, \breve{A}\hat{\boldsymbol{q}}_3 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_2, \breve{A}\hat{\boldsymbol{q}}_3 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_1)\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_3 \\
&\ldots
\end{aligned} \tag{A.2}$$

Using the properties of the scalar product one can immediately see that the vectors $\boldsymbol{q}_2$ and $\hat{\boldsymbol{q}}_1$ are orthogonal, that is $(\boldsymbol{q}_2, \hat{\boldsymbol{q}}_1) = 0$. Using this it is easy to simplify the expression for $\boldsymbol{q}_3$ and then iterate to arrive at:

$$\begin{aligned}
\boldsymbol{q}_1 &= v \\
\boldsymbol{q}_2 &= \breve{A}\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_1)\hat{\boldsymbol{q}}_1 \\
\boldsymbol{q}_3 &= \breve{A}\hat{\boldsymbol{q}}_2 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_2, \breve{A}\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_2 \\
\boldsymbol{q}_4 &= \breve{A}\hat{\boldsymbol{q}}_3 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_2, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_2 - (\hat{\boldsymbol{q}}_3, \breve{A}\hat{\boldsymbol{q}}_3)\hat{\boldsymbol{q}}_3 \\
&\ldots
\end{aligned} \tag{A.3}$$

or alternatively:

$$\begin{aligned}
\boldsymbol{q}_1 &= v \\
\boldsymbol{q}_2 &= \breve{A}\hat{\boldsymbol{q}}_1 - h_{11}\hat{\boldsymbol{q}}_1 \\
\boldsymbol{q}_3 &= \breve{A}\hat{\boldsymbol{q}}_2 - h_{12}\hat{\boldsymbol{q}}_1 - h_{22}\hat{\boldsymbol{q}}_2 \\
\boldsymbol{q}_4 &= \breve{A}\hat{\boldsymbol{q}}_3 - h_{13}\hat{\boldsymbol{q}}_1 - h_{23}\hat{\boldsymbol{q}}_2 - h_{33}\hat{\boldsymbol{q}}_3 \\
&\ldots
\end{aligned} \tag{A.4}$$

The meaning of elements `h[k , k - 1]` (h_{kk-1}) from line `8` of Program 1 can also be easily calculated. For example the using (A.3):

$$\begin{aligned} |\boldsymbol{q}_3|^2 &= (\boldsymbol{q}_3, \boldsymbol{q}_3) \\ &= (\boldsymbol{q}_3, \breve{A}\hat{\boldsymbol{q}}_2 - (\hat{\boldsymbol{q}}_1, \breve{A}\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_1 - (\hat{\boldsymbol{q}}_2, \breve{A}\hat{\boldsymbol{q}}_2)\hat{\boldsymbol{q}}_2) \\ &= (\boldsymbol{q}_3, \breve{A}\hat{\boldsymbol{q}}_2) \\ &= |\boldsymbol{q}_3|(\hat{\boldsymbol{q}}_3, \breve{A}\hat{\boldsymbol{q}}_2) \end{aligned}$$

In other words when we calculate the norm $|\boldsymbol{q}_3|$ we arrive at $(\hat{\boldsymbol{q}}_3, \breve{A}\hat{\boldsymbol{q}}_2)$. This in addition to (A.4) can be generalized. It becomes clear that:

$$h_{ij} = (\hat{\boldsymbol{q}}_i, \breve{A}\hat{\boldsymbol{q}}_j) \tag{A.5}$$

are matrix elements of the operator $\breve{A}$ projected onto a space spanned by the orthogonal vectors span $(\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, ..., \hat{\boldsymbol{q}}_N)$. It can also be shown that this space is identical to the Krylov space

$$K(\hat{\boldsymbol{v}}, N) = \text{span}\left(\hat{\boldsymbol{v}}, \breve{A}\hat{\boldsymbol{v}}, \breve{A}^2\hat{\boldsymbol{v}}, ..., \breve{A}^N\hat{\boldsymbol{v}}\right) = \text{span}\left(\hat{\boldsymbol{q}}_1, \hat{\boldsymbol{q}}_2, \hat{\boldsymbol{q}}_3, ..., \hat{\boldsymbol{q}}_N\right). \tag{A.6}$$

In order to get a complete set of matrix elements of $[\breve{h}]$ it is necessary to additionally calculate the last column. This can be done using Program 2

Program 2 Pseudo - code for calculating the last column.

```
DO k = 1 , N
   v <- q[n]
   v <- A v
   h[k , n] = (q[k] , v)
END DO
```

or alternatively the last row can be dropped, $\hat{\boldsymbol{q}}_N$ neglected and $N \leftarrow N - 1$ set.

The eigenvectors and eigenvalues of the operator $\breve{h}$ in the Krylov space (A.6) are approximations of the most extreme eigenvectors and values of $\breve{A}$. Reducing the dimension of vectors and matrix operators makes it possible to apply the standard numerical methods of linear algebra to problems that would otherwise be too big due to limited computing resources and time. This reduction is quite substantial. For example, the Faddeev component in 3N bound state calculations can be discretized over its arguments to produce a (typically) 500000 dimensional vector but after using the Arnoldi procedure we work with (typically) 40 dimensional vectors and matrices. More detailed information on the algorithm can be found for example in [61].

This algorithm is in many ways similar to standard Gram-Schmidt orthogonalization procedure, however it proves to be more numerically stable. Calculating the action of $\breve{A}$, the scalar product and the norm can be easily parallelized and implemented to run on a large computing cluster, making Arnoldi iterations an essential part of our numerical performance. In our programs the Arnoldi iteration is used to calculate the bound states of the 2N and 3N systems and to solve linear equations arising in the iterative calculation of transition operator matrix elements. Details can be found in the programs themselves and in Appendix D.

A.3 Implementation on a large computing cluster

As mentioned before the only operations that need to be implemented in order for our iteration schemes to work are the action of a linear operator on a vector and the calculation of a scalar product. Due to the large size of vectors involved it is often necessary to construct parallel code. This section describes our method of implementing these two algebraic operations on a large computing cluster.

In our programs we use the Message Passing Interface (MPI) for parallel computations. For a detailed description of this protocol the reader is referred to a wealth of tutorials and guides available on the Internet. A good starting point for learning the interface is:

`https://computing.llnl.gov/tutorials/mpi/`

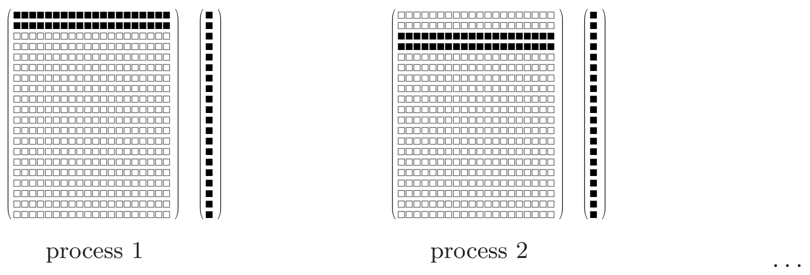
In a typical MPI session the same program is distributed over a large number of computing nodes (processors) and run in parallel. Each running program has access to a unique identification number `id`. We found that in order to implement the two operations (the action of a large linear operator on a vector and scalar product) only two subroutines using the MPI interface are necessary. Below we give the two-line pseudo code that uses these subroutines, can be ran in parallel and used to implement the action of a large linear operator on a vector (Program 3) and scalar product between the two vectors (Program 4).

In Program 3 all variables are vectors and variables ending with `SG_id` hold a fragment of the full vector. The distribution of the full vector between the parallel processes is determined by each processes unique identification number. This distribution is performed in such a way that after running `v <- SGtoFULL(VSG_id)` each process will hold the full version of the vector in `v`. `SGtoFULL(VSG_id)` can be easily implemented using MPI subroutines. It is important to note that vectors from the pseudo code do not have to be implemented as one dimensional tables. When dealing with the 2N, 3N bound states or the transition operator, it is far more advantageous to hold the scalar functions as multidimensional arrays, this way, implementing the action of the large linear operator on the scalar function is more straightforward. Operation `vSG_id <- O(v)` calculates the action of the large linear operator $\check{O}$ on the full vector `v`, only a fragment (the segment is determined by the threads unique `id`) `VSG_id` of the resulting vector is calculated. As a result of running Program 3 the value of the full vector `v` will be changed, the new value is created from the old value with the use of the large operator $\check{O}$. In addition to the full vector and the vector fragment each parallel thread needs to hold in memory the data necessary to use the large linear operator in calculating the vector fragment. For the calculations to be efficient, the number of fragments into which the full vector is split should be such that performing line `1` takes substantially longer than line `2`. Data distribution is a bottle neck on most computational clusters.

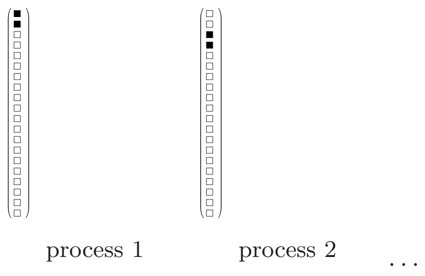
Program 3 Pseudo - code for the parallelized action of the large linear operator on a vector.

```
      <determine the segment of the
      full vector that will be held
      in vSG_id>
1     vSG_id <- O(v)
2     v <- SGtoFULL(vSG_id)
```

To summarize, each process holds information necessary to implement a fragment of the full operator and information necessary to construct the full vector:



Having run Program 3, each process will hold a fragment of the full vector:



and the full vector can be reconstructed from the fragments and used in the next iteration.

The scalar product usually does not require as much computational effort as calculating the action of the large linear operator. Parallelization can be achieved in a straightforward way. In Program 4 `spSG_id` holds part of the scalar product calculated from the threads vector fragments `v1SG_id` and `v2SG_id`. After summing up all contributions from all threads in line `2`, `sp` will hold the full scalar product. Just as before each vector ending with `SG_id` holds a fragment determined by the threads `id`.

Program 4 Pseudo - code for the parallelized scalar product.

```
      <determine the segment of the
      full vectors that will be held
      in v1SG_id , v2SG_id>
1     spSG_id <- SP(v1SG_id , v2SG_id)
2     sp <- SGtoFULLsp(spSG_id)
```

Appendix B

The *util1N2N3N.m* package

In previous chapters many expressions were contained within square brackets [. . .]. This notation was used to mark the existence of a matrix implementation, typically in one of the bases from the Appendix: E.1, E.2, E.3, E.4, E.5 or E.6. Further observation leads to the conclusion that all expressions inside [. . .] can be built from a finite number of construction elements. The *util1N2N3N.m* *Mathematica*® package provides an implementation of these construction elements and tools that can be used to put them together. This package can be used with *FunctionArray.m* discussed in the next chapter in order to create a FORTRAN implementation of any [. . .] expression.

This chapter can be read in parallel to

PROGRAMS/UTIL1N2N3Ntutorial/UTIL1N2N3Ntutorial.nb

After evaluating, the input and output numbers from the notebook schould match those in the text below.

B.1 Loading the package

This can be done by typing:

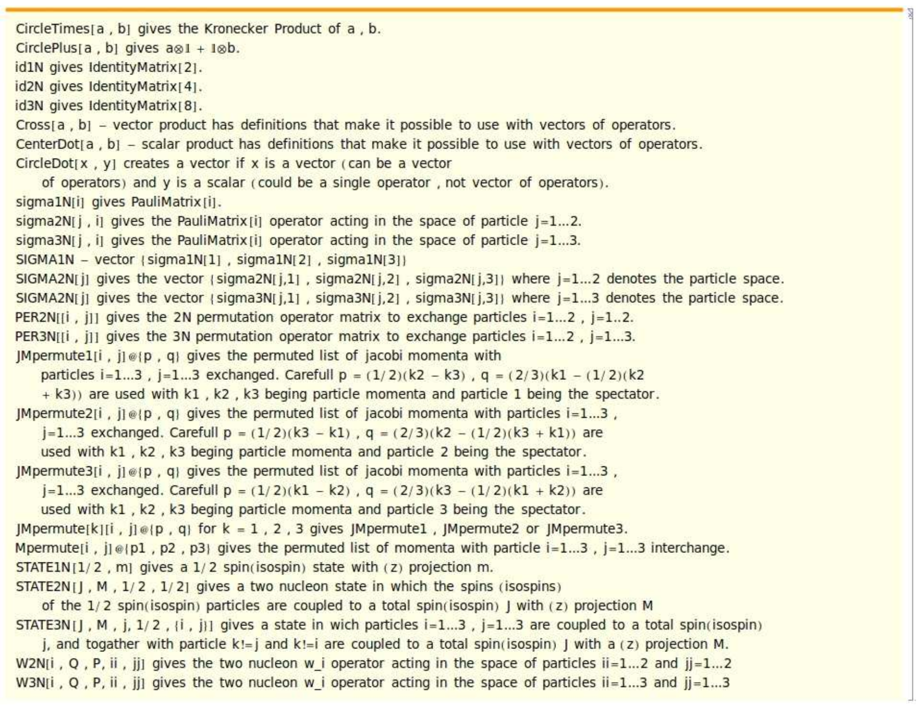
```
In[1]:= << (NotebookDirectory[] <> "../util1N2N3N.m");
```

and uses the `Get` *Mathematica*® function. The argument of this function is a path name pointing to a directory above the tutorial notebook.

B.2 Getting information on the package

Information on any symbols introduced by the package can be accessed by using the `?` command. For example:

```
In[2]:= ? util1N2N3N
```

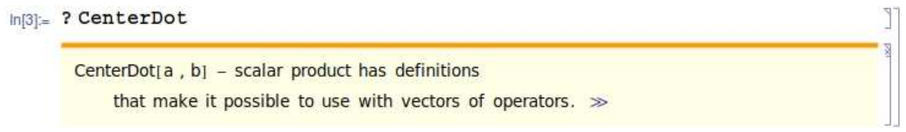


```
CircleTimes[a , b] gives the Kronecker Product of a , b.
CirclePlus[a , b] gives a⊗𝟙 + 𝟙⊗b.
id1N gives IdentityMatrix[2].
id2N gives IdentityMatrix[4].
id3N gives IdentityMatrix[8].
Cross[a , b] – vector product has definitions that make it possible to use with vectors of operators.
CenterDot[a , b] – scalar product has definitions that make it possible to use with vectors of operators.
CircleDot[x , y] creates a vector if x is a vector (can be a vector
    of operators) and y is a scalar (could be a single operator , not vector of operators).
sigma1N[i] gives PauliMatrix[i].
sigma2N[j , i] gives the PauliMatrix[i] operator acting in the space of particle j=1...2.
sigma3N[j , i] gives the PauliMatrix[i] operator acting in the space of particle j=1...3.
SIGMA1N – vector {sigma1N[1] , sigma1N[2] , sigma1N[3]}
SIGMA2N[j] gives the vector {sigma2N[j,1] , sigma2N[j,2] , sigma2N[j,3]} where j=1...2 denotes the particle space.
SIGMA2N[j] gives the vector {sigma3N[j,1] , sigma3N[j,2] , sigma3N[j,3]} where j=1...3 denotes the particle space.
PER2N[[i , j]] gives the 2N permutation operator matrix to exchange particles i=1...2 , j=1..2.
PER3N[[i , j]] gives the 3N permutation operator matrix to exchange particles i=1...2 , j=1...3.
JMpermute1[i , j]@{p , q} gives the permuted list of jacobi momenta with
    particles i=1...3 , j=1...3 exchanged. Carefull p = (1/2)(k2 – k3) , q = (2/3)(k1 – (1/2)(k2
    + k3)) are used with k1 , k2 , k3 beging particle momenta and particle 1 being the spectator.
JMpermute2[i , j]@{p , q} gives the permuted list of jacobi momenta with particles i=1...3 ,
    j=1...3 exchanged. Carefull p = (1/2)(k3 – k1) , q = (2/3)(k2 – (1/2)(k3 + k1)) are
    used with k1 , k2 , k3 beging particle momenta and particle 2 being the spectator.
JMpermute3[i , j]@{p , q} gives the permuted list of jacobi momenta with particles i=1...3 ,
    j=1...3 exchanged. Carefull p = (1/2)(k1 – k2) , q = (2/3)(k3 – (1/2)(k1 + k2)) are
    used with k1 , k2 , k3 beging particle momenta and particle 3 being the spectator.
JMpermute[k][i , j]@{p , q} for k = 1 , 2 , 3 gives JMpermute1 , JMpermute2 or JMpermute3.
Mpermute[i , j]@{p1 , p2 , p3} gives the permuted list of momenta with particle i=1...3 , j=1...3 interchange.
STATE1N[1/2 , m] gives a 1/2 spin(isospin) state with (z) projection m.
STATE2N[J , M , 1/2 , 1/2] gives a two nucleon state in which the spins (isospins)
    of the 1/2 spin(isospin) particles are coupled to a total spin(isospin) J with (z) projection M
STATE3N[J , M , j, 1/2 , {i , j}] gives a state in wich particles i=1...3 , j=1...3 are coupled to a total spin(isospin)
    j, and togather with particle k!=j and k!=i are coupled to a total spin(isospin) J with a (z) projection M.
W2N[i , Q , P, ii , jj] gives the two nucleon w_i operator acting in the space of particles ii=1...2 and jj=1...2
W3N[i , Q , P, ii , jj] gives the two nucleon w_i operator acting in the space of particles ii=1...3 and jj=1...3
```

lists information on all available functionalities.

B.3 Scalar products, vector products and spin (isospin) operators

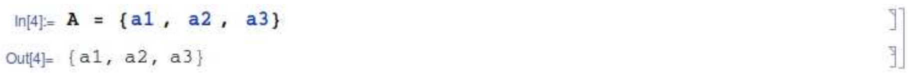
One of the more useful new features is the scalar product (SP). The standard *Mathematica*® implementation uses the . symbol. The new implementation uses `CenterDot` (entered as ⋮.⋮ and displayed as · where ⋮ is the ESCAPE key):



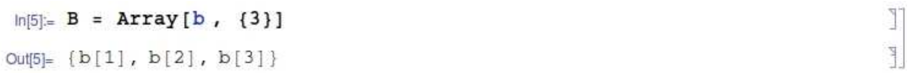
```
In[3]:= ? CenterDot

CenterDot[a , b] – scalar product has definitions
    that make it possible to use with vectors of operators. >>
```

After looking at the following examples it should become apparent that the new implementation has many advantages over the standard method (that uses the lower .).

First we introduce two vectors:



```
In[4]:= A = {a1 , a2 , a3}
Out[4]= {a1, a2, a3}
```



```
In[5]:= B = Array[b , {3}]
Out[5]= {b[1], b[2], b[3]}
```

and calculate the SP:

```
In[6]:=  A · B
Out[6]=  a1 b[1] + a2 b[2] + a3 b[3]
```

Up to this point there is no difference with ".". Now we try to calculate the SP of a known vector A and a new symbol CC:

```
In[7]:=  A · CC
Out[7]=  a1 CC[1] + a2 CC[2] + a3 CC[3]
```

CenterDot assumes that CC is an element of an Array. This assumption can be very useful in some situations. Using A.CC would produce an error.

The next example uses matrix operators. We introduce:

```
In[8]:=  ? SIGMA3N

   SIGMA2N[j] gives the vector {sigma3N[j,1] ,
      sigma3N[j,2] , sigma3N[j,3]} where j=1...3 denotes the particle space.
```

```
In[9]:=  F = SIGMA3N[1];
```

F now holds a vector of matrix operators, more precisely:

$$\left(\check{\sigma}_1 \otimes \check{1} \otimes \check{1}, \check{\sigma}_2 \otimes \check{1} \otimes \check{1}, \check{\sigma}_2 \otimes \check{1} \otimes \check{1}\right)$$

with the spin operator (or the isospin operator, since both are represented using the Pauli matrixes) acting in the space of particle 1. Each component of the vector is an 8×8 dimensional matrix:

```
In[10]:=  F // Dimensions
Out[10]=  {3, 8, 8}
```

CenterDot allows us to quickly calculate what would normally require the use of Sum:

```
In[11]:=  A · F == Sum[A[[i]] F[[i]] , {i , 1 , 3}] // FullSimplify
Out[11]=  True
```

```
In[12]:=  F · F == Sum[F[[i]].F[[i]] , {i , 1 , 3}] // FullSimplify
Out[12]=  True
```

The new definitions for the cross product (KP, entered as ⁝cross⁝ and displayed as ×):

```
In[13]:= ? Cross

   Cross[a , b] - vector product has definitions
       that make it possible to use with vectors of operators. >>
```

have similar properties as the `CenterDot`. We can calculate the CP of two known vectors:

```
In[16]:= A × B
Out[16]= {-a3 b[2] + a2 b[3], a3 b[1] - a1 b[3], -a2 b[1] + a1 b[2]}
```

or the CP of a known vector and an unknown array `CC`:

```
In[17]:= A × CC
Out[17]= {-a3 CC[2] + a2 CC[3], a3 CC[1] - a1 CC[3], -a2 CC[1] + a1 CC[2]}
```

Finally the CP between a vector and a vector of operators and the CP between two vectors of operators works according to expectations:

```
In[21]:= A × F == Table[Sum[LeviCivitaTensor[3][[i , j , k]] A[[j]] F[[k]],
             {j , 1 , 3} , {k , 1 , 3}] , {i , 1 , 3}] // FullSimplify
Out[21]= True
```

```
In[22]:= F × F == Table[Sum[LeviCivitaTensor[3][[i , j , k]] F[[j]].F[[k]],
             {j , 1 , 3} , {k , 1 , 3}] , {i , 1 , 3}] // FullSimplify
Out[22]= True
```

and does not require the manual use of `Sum`.

B.4 Permutations

Permutations within the spin (isospin) space of the 2N and 3N systems are a big part of the calculations presented in the previous chapters. The matrix representation of the permutation operator in the E.5 spin (isospin) basis can be obtained using:

```
In[23]:= PER2N[[1 , 2]] // MatrixForm
Out[23]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Permuting the particles twice produces the identity operator:

```
In[24]:= PER2N[[1 , 2]].PER2N[[1, 2]] // MatrixForm
Out[24]//MatrixForm=
```

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Incidentally the package also provides the identity operator for the 2N spin (isospin) space:

```
In[25]:= id2N // MatrixForm
Out[25]//MatrixForm=
        1 0 0 0
        0 1 0 0
        0 0 1 0
        0 0 0 1
```

With two particles there is only one nontrivial way of permuting them. Permuting particle 1 with particle 1 or particle 2 with particle 2 gives the identity operator:

```
In[26]:= PER2N[[1, 1]] == PER2N[[2, 2]] == id2N // FullSimplify
Out[26]= True
```

Permutation matrices in the joined isospin - spin space can be constructed using the KP. The new definitions for the `CircleTimes` (entered as $\vdots c*\vdots$ and displayed as $\otimes$) can return the KP:

```
In[27]:= PER2N[[1, 2]] ⊗ PER2N[[1 , 2]] // MatrixForm
Out[27]//MatrixForm=
        1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
        0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
        0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
```

The `CircleTimes` definitions will be discussed in more detail in the next section. For 3N systems one can use `PER3N`. Now the permutation operators are implemented using 8×8 matrices in basis E.6.

Permutations within the spin (isospin) spaces of 2N and 3N systems are simple to work out. Careful considerations of E.2 and E.3 lead directly to the matrix form of the permutation operator. Permutations within the momentum space of the 3N system are slightly more complicated. Permutations of single particle momentum eigenstates with particle 1 having momentum `k1`, particle 2 having momentum `k2` and particle 3 having momentum `k3`:

```
In[28]:= {k1 , k2 , k3}
Out[28]= {k1, k2, k3}
```

are still simple. Our package provides the `Mpermute` symbol:

```
In[29]:= Mpermute[1 , 3]@Mpermute[1 , 2]@%
Out[29]= {k3, k1, k2}
```

```
In[30]:= Mpermute[1 , 2]@Mpermute[1 , 3]@%
Out[30]= {k1, k2, k3}
```

which works according to expectations.

Permuting Jacobi momentum eigenstates is a little more tricky. There are three different ways of defining Jacobi momenta (see `?util1N2N3N` for more information), the package introduces the `JMpermute` symbol:

```
In[31]:= P1223 = (JMpermute[1][1 , 2]@JMpermute[1][2 , 3]@#) &;

In[32]:= P1223@{p , q}

           1                       q
Out[32]= { - (-2 p + 3 q), -p - - }
           4                       2
```

which can use the three different definitions (`JMpermute[1...3][...]`). The example above uses the first definition with $\boldsymbol{p} = \frac{1}{2}(\boldsymbol{k}_2 - \boldsymbol{k}_3)$ and $\boldsymbol{q} = \frac{2}{3}(\boldsymbol{k}_1 - \frac{1}{2}(\boldsymbol{k}_2 + \boldsymbol{k}_3))$ and applies $\check{P}_{12}\check{P}_{23}$ to $|\, \boldsymbol{pq}\rangle$. A reverse permutation recreates the original vectors:

```
In[33]:= JMpermute[1][2 , 3]@JMpermute[1][1 , 2]@%

           1  1                      q     1      q    1
Out[33]= { - (- (2 p - 3 q) - 3 (-p - -)), - (p + -) + - (-2 p + 3 q)}
           4  2                      2     2      2    4
```

```
In[34]:= % // FullSimplify
Out[34]= {p, q}
```

B.5 Other useful definitions and examples

Using the built in ⊗ (`CircleTimes` in *Mathematica*® , entered as ⋮$c*$⋮) symbol produces the Kronecker Product matrix representation of the tensor product. It is crucial to underline that the order of operations in the KP is important so an extensive use of brackets is encouraged:

```
In[39]:= SIGMA3N[1] == {(sigma1N[1] ⊗ id1N) ⊗ id1N ,
            (sigma1N[2] ⊗ id1N) ⊗ id1N , (sigma1N[3] ⊗ id1N) ⊗ id1N} //
          FullSimplify
Out[39]= True
```

This is a very powerful function of the *util1N2N3N.m* function and allowed us to glue together expressions acting in the spaces of different nucleons.

In some situations expressions inside [...] consist of vectors acting on scalar quantities. The `CircleDot` symbol (entered using ⋮$c.$⋮ and displayed as ⊙) makes working with such products simple, even when vectors of operators are involved:

```
In[40]:= SIGMA3N[1] ⊙ sigma3N[1, 1] ==
           Table[SIGMA3N[1][[i]].sigma3N[1, 1] , {i , 1 , 3}] //
          FullSimplify
Out[40]= True
```

Here `SIGMA3N[1]` is a vector of operators (three 8×8 dimensional spin (or isospin) operators acting in the space of particle 1) and `sigma3N[1,1]` is a single operator acting in the space of the first particle.

This next example mimics a situation in which the potential operator is given in operator form $\langle \boldsymbol{p}' \mid \check{V} \mid \boldsymbol{p} \rangle$ (where we will use $\boldsymbol{p}' = \texttt{pp}$) and acts in the spin space of the 2N particle system:

```
In[42]:= pp = {pp1 , pp2 , pp3};
         p = {p1 , p2 , p3};
```

```
In[44]:= V =
           i Sum[(SIGMA2N[1][[i]] + SIGMA2N[2][[i]]) Cross[p , pp][[i]] ,
             {i , 1 , 3}] // FullSimplify;
```

Finding the decomposition of this operator into the six $\check{w}$ operators introduced when discussing the deuteron can be achieved using the built definitions for `W2N` (for 3N systems see `?W3N`). First a basis of six flattened $\check{w}$ operators is created:

```
In[45]:= base = Table[W2N[i , pp , p , 1 , 2] // Flatten , {i , 1 , 6}] //
            Transpose;
```

Next a linear system is solved:

```
In[46]:= LinearSolve[base , V // Flatten]
Out[46]= {0, 0, 1, 0, 0, 0}
```

An error in this step might suggest that V does not have a decomposition within the six $\check{w}$ operators.

Single particle spin (isospin) $\frac{1}{2}$ states can be used with:

```
In[47]:= STATE1N[1 / 2 , 1 / 2]
Out[47]= {1, 0}
```

```
In[48]:= STATE1N[1 / 2 , -1 / 2]
Out[48]= {0, 1}
```

Two and three-nucleon spin (isospin) states constructed using Clebsch-Gordan coefficients are also available:

```
In[49]:= STATE2N[1 , 0 , 1 / 2 , 1 / 2]
```

$$\texttt{Out[49]=}\ \left\{0,\ \frac{1}{\sqrt{2}},\ \frac{1}{\sqrt{2}},\ 0\right\}$$

```
In[50]:= STATE2N[1 , 0 , 1 / 2 , 1 / 2] ==
          Sum[
           If[m1 + m2 == 0 , ClebschGordan[{1 / 2 , m1} , {1 / 2 , m2} ,
              {1 , 0}] , 0]
            Flatten[STATE1N[1 / 2 , m1] ⊗ STATE1N[1 / 2 , m2]] ,
           {m1 , -1 / 2 , 1 / 2 , 1} , {m2 , -1 / 2 , 1 / 2 , 1}] //
         FullSimplify

Out[50]= True
```

```
In[51]:= STATE3N[3 / 2 , 1 / 2 , 1 , 1 / 2 , {2 , 3}]
```

Out[51]= $\left\{0, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, 0, \frac{1}{\sqrt{3}}, 0, 0, 0\right\}$

More information on these states is available with ?STATE1N, ?STATE2N, ?STATE2N.

Appendix C

The *FunctionArray.m* package

Below we give an introduction to the *FunctionArray.m Mathematica*® package. With the help of this package it is possible to produce automatic FORTRAN implementations of expressions created using *util1N2N3N.m.* This chapter is designed to be read in parallel to

PROGRAMS/FunctionArraytutorial/FunctionArraytutorial.nb

which contains three examples that are discussed below. After evaluating, the `In[]` numbers from the notebook should match those in this chapter.

C.1 Loading the package

This is very straightforward and one needs simply to type:

```
In[1]:= << (NotebookDirectory[] <> "../FunctionArray.m");
```

into the *Mathematica*® notebook. The package is contained with the files provided with this thesis and located in the directory above ***FunctionArraytutorial.nb***. Other examples also use this location.

C.2 A simple function

Now that the definitions from *FunctionArray.m* are available, we start with a very simple example. We first tell *Mathematica*® that the function `simple` takes one argument, a `double precision` number:

```
In[4]:= ArgumentType[simple] = {{double , {}}};
```

Multiple arguments can be used, for example {{`double , `{`2,2`}},{`complex,`{`2`}}} (two arguments; the first one is a `double precision, dimension(2,2)` matrix and the second one is a `complex, dimension(2)` vector). The first element in each sublist corresponds to an appropriate FORTRAN type and the second element corresponds to the dimension. Empty brackets correspond to a single number. More information on available types is available by running `?FunctionArray`.

Next we need to supply information on the objects returned by the function. In our example `simple` will return a `double precision` number:

```
In[6]:= ReturnType[simple] = {double , {}};
```

Notice the lack of external brackets - only one value can be returned (nothing stands in the way of returning a vector or array).

Many objects that we would like to implement as FORTRAN functions contain a large amount of integer arguments or indexes. The *FunctionArray.m* package makes sure that the implementation of such objects is efficient. This is done using one final piece of information. Apart from the function itself we need to declare that `simple` has one integer index and it can take values from the range $1 \dots 3$:

```
In[8]:= IndexRange[simple] = {3};
```

Finally we define the function itself. The general syntax is:

```
In[10]:= simple@{i_}@{d_} := d^i;
```

where the first element is the name of the function, the second element is a list of integer index arguments (specified by `IndexRange`) and the third element is a list of the remaining arguments (specified by `ReturnType`).

These four definitions (`ArgumentType`, `ReturnType`, `IndexRange` and `simple@{...}@{...}`) are all that *Mathematica*® needs to know in order to produce a FORTRAN implementation. Both `.f` and `.f90` code can be produced. In addition to the function implementations, test programs can be created:

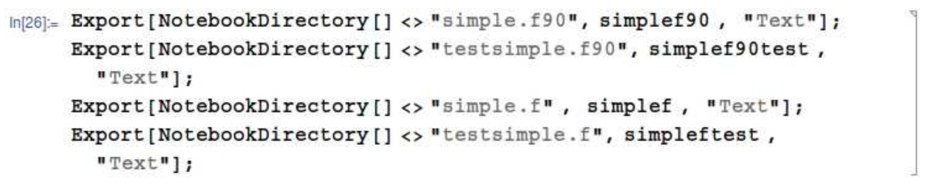
```
In[12]:= (*.f90 code:*)
         simplef90 = functionarray["simple" , simple , N , 100];
         (*.f90 test code:*)
         simplef90test = functionarraytest["simple" , simple , N , 100];
```

```
In[14]:= (*.f code:*)
         simplef = fixedfunctionarray["simple" , simple , N , 70];
         (*.f test code:*)
         simpleftest = fixedfunctionarraytest["simple" , simple ,
            N , 70];
```

The first argument to `functionarray`, `functionarraytest`, `fixedfunctionarray` and `fixedfunctionarraytest` is a string that will be used in the name of the functions in the FORTRAN implementations. The second argument is a symbol for which the necessary definitions (`ArgumentType`, `ReturnType`, ...) have been evaluated. The last argument is the maximal number of columns for the code (100 columns for `.f90` code and 70 for `.f` fixed syntax).

The third argument is a function, in this case `N`. In `functionarraytest` and `fixedfunctionarraytest`, `N` is just a dummy argument. The relevance of this third argument in `functionarray` and `fixedfunctionarray` can be understood after a short introduction to their inner workings. After evaluating `functionarray["simple", simple , N , 100]` or `fixedfunctionarray["simple" , simple , N , 70]` *Mathematica*® creates a list of unique argument symbols according to the specifications in `ArgumentType[simple]`. The list of unique arguments is used as `B` in `simple@{A_List}@{B_List}`, this expression is then evaluated for all possible integer indexes in `A`. This list of evaluated expressions will be turned into FORTRAN code. Before turning the evaluated expressions into a FORTRAN, code `N` will be applied to each expression. If the expression is an array or list, `N` is applied to each element of the array or list separately. This third argument (`N`) thus allows us to make one final adjustment before producing the FORTRAN implementation. It can be especially useful when working with functions that return, say, an array of complex numbers. In this case `N` can be replaced by `DCMPLX[N\[#]]&`. This function will be applied to all elements of the array before turning it into FORTRAN code. Inside the code each element of the array will have `DCMPLX(...)` around it informing the compiler that the elements are of `double complex` type.

Exporting the code can be performed using the `"Text"` argument with `Export`:



```
In[26]:= Export[NotebookDirectory[] <> "simple.f90", simplef90 , "Text"];
         Export[NotebookDirectory[] <> "testsimple.f90", simplef90test ,
           "Text"];
         Export[NotebookDirectory[] <> "simple.f" , simplef , "Text"];
         Export[NotebookDirectory[] <> "testsimple.f", simpleftest ,
           "Text"];
```

The code will be written to the same directory as the notebook. A peek into the new files reveals the method of implementation.

The `simple.f` code uses the `case` construct:

```
      select case(indsum)

      case(1)
       simpleF = simple_1(dp17)
      case(2)
       simpleF = simple_2(dp17)
      case(3)
       simpleF = simple_3(dp17)

      end select
```

Each index case $(1 \dots 3)$ is implemented as a separate function, for example:

```
      function simple_2(dp19)

      double precision :: simple_2
      double precision :: dp19

      simple_2 =
     -dp19**2

      return
      end function simple_2
```

This type of construction works nearly as fast as the function pointer scheme used in the `.f90` implementation that will be discussed later. Using code optimization makes up for the lack of function pointers in FORTRAN77 (making the speed differences, compared to a function pointer implementation, negligible).

In order to use the full implementation, a program should provide an interface. This can be copied from the test program `testsimple.f`:

```
      interface
      function simpleF(ind22, dp21)

      double precision :: simpleF
      double precision :: dp21

      integer :: ind22

      end function simpleF
      end interface
```

Notice that the function name is `simpleF`, it is created from the string "simple" that was supplied to *Mathematica*® and an additional "F". The test program calls `simpleF` for all cases if index values:

```
      print *,simpleF(1, dp20)
      print *,simpleF(2, dp20)
      print *,simpleF(3, dp20)
```

The FORTRAN90 version uses a slightly more refined method of implementation. Each possible index value is assigned a pointer to a function, this scheme is fast even without code optimization. Function pointers are a relatively new addition to FORTRAN and require a compiler that supports this new feature. Most new compilers utilize this novelty (for example `gfortran-4.5` or higher) and there are also many examples of codes available online. File `simple.f90` contains a module:

```
module simple
implicit none
...
end module simple
```

In order to work, this module needs to be initialized by calling:

```
subroutine initializesimple()
        allocate(simpleARRAY(3))
...
        simpleARRAY(1) % simplepointer => simple_1
        simpleARRAY(2) % simplepointer => simple_2
        simpleARRAY(3) % simplepointer => simple_3
        return
end subroutine initializesimple
```

This subroutine allocates an array of function pointers and assigns a function to each element of the array.

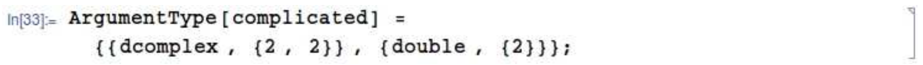
Using the implemented functions is simple, `testsimple.f90` calls each index case:

```
        print * ,simpleARRAY(1) % simplepointer(dp16)
        print * ,simpleARRAY(2) % simplepointer(dp16)
        print * ,simpleARRAY(3) % simplepointer(dp16)
```

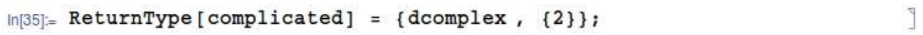
Notice that `simpleARRAY` and `simplepointer` are created from the string "simple" that was supplied to *Mathematica*® and an additional "ARRAY", "pointer". For more complicated functions `simpleARRAY(1)` might be replaced by `moreindexesARRAY(1,2,3)` and will correspond to a different set of indexes.

C.3 A more complicated example

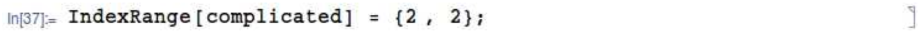
In this example, we implement a function that takes two arguments and returns a 2 dimensional vector of complex numbers. The first argument is a 2×2 matrix and the second one is also a 2 dimensional vector. In addition, we declare two integer indexes that will take values from $1 \ldots 2$. All this information can be supplied to *Mathematica*® using the following three simple expressions:



```
In[33]:= ArgumentType[complicated] =
           {{dcomplex , {2 , 2}} , {double , {2}}};
```

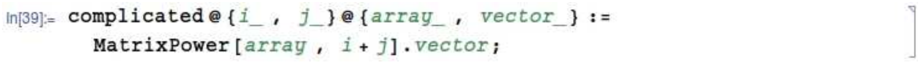


```
In[35]:= ReturnType[complicated] = {dcomplex , {2}};
```



```
In[37]:= IndexRange[complicated] = {2 , 2};
```

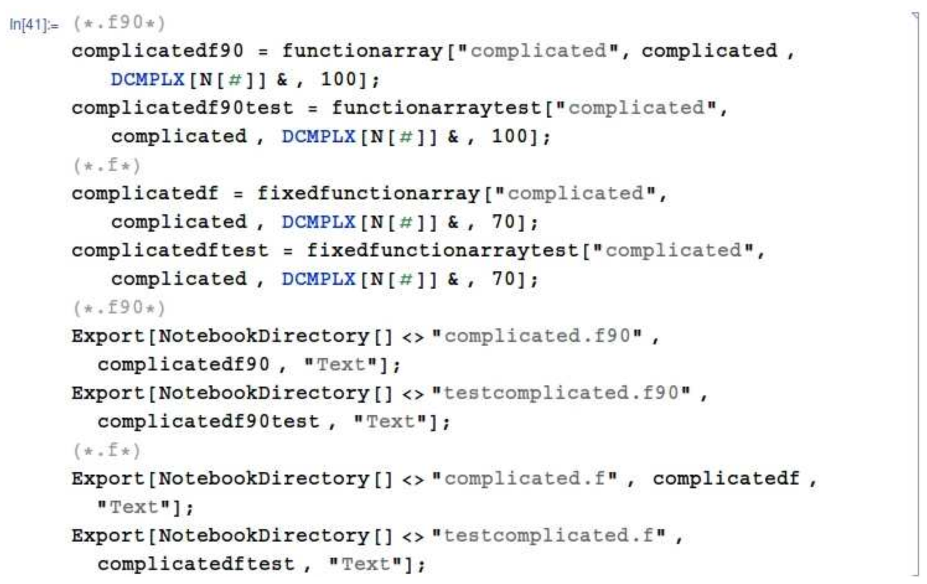
We use a combination of matrix and vector operators to produce the result. The function itself is defined using:



```
In[39]:= complicated@{i_ , j_}@{array_ , vector_} :=
           MatrixPower[array , i + j].vector;
```

with the indexes (`i`, `j`), the matrix argument (`array`) and the vector argument (`vector`).

Code production is the final step:



```
In[41]:= (*.f90*)
         complicatedf90 = functionarray["complicated", complicated ,
            DCMPLX[N[#]] & , 100];
         complicatedf90test = functionarraytest["complicated",
            complicated , DCMPLX[N[#]] & , 100];
         (*.f*)
         complicatedf = fixedfunctionarray["complicated",
            complicated , DCMPLX[N[#]] & , 70];
         complicatedftest = fixedfunctionarraytest["complicated",
            complicated , DCMPLX[N[#]] & , 70];
         (*.f90*)
         Export[NotebookDirectory[] <> "complicated.f90" ,
           complicatedf90 , "Text"];
         Export[NotebookDirectory[] <> "testcomplicated.f90" ,
           complicatedf90test , "Text"];
         (*.f*)
         Export[NotebookDirectory[] <> "complicated.f" , complicatedf ,
           "Text"];
         Export[NotebookDirectory[] <> "testcomplicated.f" ,
           complicatedftest , "Text"];
```

Multiple versions of FORTRAN implementations and test files will be produced. The structure of implementation is analogous to the simple example given above. The `.f` code can be called after supplying an interface (see ***PROGRAMS/FunctionArraytutorial/testcomplicated.f***):

```
      print * ,complicatedF(1, 1, dc35, dp36)
      print * ,complicatedF(1, 2, dc35, dp36)
      print * ,complicatedF(2, 1, dc35, dp36)
      print * ,complicatedF(2, 2, dc35, dp36)
```

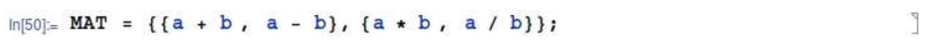
The `.f90` code can be called after initializing the module (see ***PROGRAMS/FunctionArraytutorial/testcomplicated.f90***):

```
        call initializecomplicated()
        ...
        print * ,complicatedARRAY(1, 1) &
                % complicatedpointer(dc27,dp28)
        print * ,complicatedARRAY(1, 2) &
                % complicatedpointer(dc27,dp28)
        print * ,complicatedARRAY(2, 1) &
                % complicatedpointer(dc27,dp28)
        print * ,complicatedARRAY(2, 2) &
                % complicatedpointer(dc27,dp28)
```

The names of the functions and arrays are created, as in the simple example, by adding ”F”, ”ARRAY” or ”pointer” to the string ”complicated” (passed as an argument to `functionarray`, `fixedfunctionarray`, . . .).

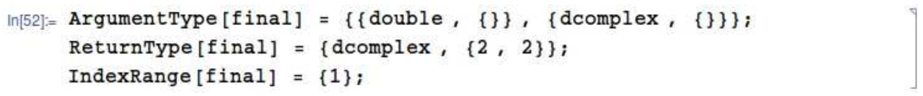
C.4 One more example

In this last example we assume that a lot of time was needed to calculate a matrix:



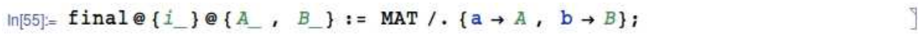
```
In[50]:= MAT = {{a + b , a - b}, {a * b , a / b}};
```

We would like *Mathematica*® to use this array and not calculate its elements on the fly. First we give the standard information:



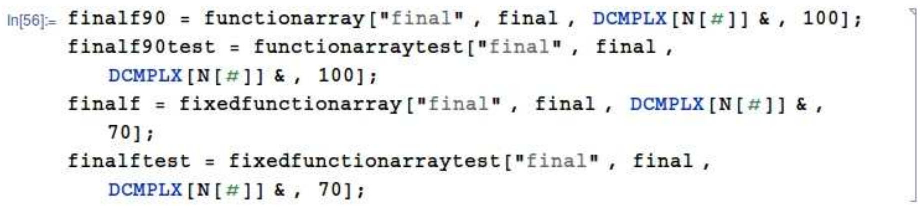
```
In[52]:= ArgumentType[final] = {{double , {}} , {dcomplex , {}}};
         ReturnType[final] = {dcomplex , {2 , 2}};
         IndexRange[final] = {1};
```

Next we define the function itself by using substitution rules:

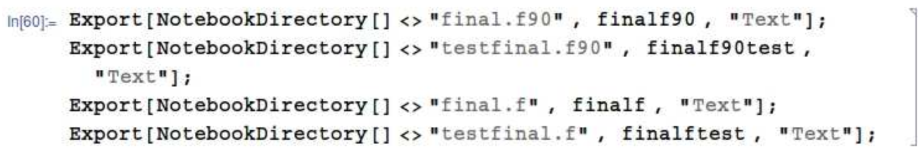


```
In[55]:= final@{i_}@{A_ , B_} := MAT /. {a → A , b → B};
```

Finally we produce the code:



```
In[56]:= finalf90 = functionarray["final" , final , DCMPLX[N[#]] & , 100];
         finalf90test = functionarraytest["final" , final ,
            DCMPLX[N[#]] & , 100];
         finalf = fixedfunctionarray["final" , final , DCMPLX[N[#]] & ,
            70];
         finalftest = fixedfunctionarraytest["final" , final ,
            DCMPLX[N[#]] & , 70];
```



```
In[60]:= Export[NotebookDirectory[] <> "final.f90" , finalf90 , "Text"];
         Export[NotebookDirectory[] <> "testfinal.f90" , finalf90test ,
           "Text"];
         Export[NotebookDirectory[] <> "final.f" , finalf , "Text"];
         Export[NotebookDirectory[] <> "testfinal.f" , finalftest , "Text"];
```

The test functions in this case will print out the 2×2 matrix elements in the default FORTRAN order. A more human readable method of output can be added manually to the test code.

C.5 Final remarks

The test programs use a set of arbitrary numbers as arguments (typically 1.0). In some cases this might produce errors resulting from singularities in the implemented functions. If the output of the test programs produces numerous cases of `NAN` (not a number) this might be an indication that the values of the arguments should be changed.

The package detects errors and throws error messages. If *Mathematica*® refuses to cooperate - do not panic - the messages contain clues as to what went wrong. If the reader notices an error please contact me (*kacper.topolnicki@uj.edu.pl*).

The tutorial notebook was created using *Mathematica*® 8. Opening this notebook in a newer version of *Mathematica*® might produce a warning message. This warning can be ignored since the notebook uses only core *Mathematica*® functions that should be constant from one version to another. The *util1N2N3N.m* package also uses only core language functions and is a simple text file. Employing this package in a newer version of *Mathematica*® should not cause problems.

Appendix D

Code organization

This chapter contains documentation on the notebooks and codes that were supplied with this thesis. Each of the following sections is related to a single problem - the deuteron, the 3N bound state, the transition operator and current calculations. In the previous chapters the *FunctionArray.m Mathematica*® package was introduced. The definitions introduced by the package led us to use a common simple structure for all the following sections. First we introduce the location of notebooks that will be used to create the FORTRAN code. Next we discuss the syntax of the FORTRAN functions, their arguments and relation to previously introduced mathematical expressions.

D.1 Building blocks for the deuteron bound state calculations

The FORTRAN code that can be used in the construction of the 2N bound state calculation is constructed by evaluating:

PROGRAMS/deuteron/nb/deuteron.nb.

The code produced by the notebook will be written to:

PROGRAMS/deuteron/.

Both fixed format (***.f***) and free format (***.f90***) versions are created. Note - in order for the free format code to compile, the compiler must support function pointers (*gfortran-4.5* or higher). We assume that this is fulfilled in this whole chapter.

File ***AdeuteronFUN.f*** contains a function

```
      function AdeuteronF(i , j , p)
       double precision :: AdeuteronFUN
       integer :: i , j
       double precision :: p
```

File ***AdeuteronFUN.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      use Adeuteron
      ...
      AdeuteronARRAY(i , j) % Adeuteronpointer(p)
      ...
```

with the types:

```
       double precision :: p
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeAdeuteron()
```

Both files contain an implementation of (6.3) with $k' =$ `i` , $k =$ `j` and $\boldsymbol{p} = (0, 0, |\boldsymbol{p}|)$ with $|\boldsymbol{p}| =$ `p`.

File ***AinvdeuteronFUN.f*** contains a function

```
      function AinvdeuteronF(i , j , p)
       double precision :: AinvdeuteronFUN
       integer :: i , j
       double precision :: p
```

File ***AinvdeuteronFUN.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      use Ainvdeuteron
      ...
      AinvdeuteronARRAY(i, j) % Ainvdeuteronpointer(p)
      ...
```

with the types:

```
       double precision :: p
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeAinvdeuteron()
```

Both files contain an implementation of $(A^d(\boldsymbol{p}))^{-1}_{qk}$ from (6.7) with $q =$ `i` , $k =$ `j` and $\boldsymbol{p} = (0, 0, |\boldsymbol{p}|)$ with $|\boldsymbol{p}| =$ `p`.

File ***BdeuteronFUN.f*** contains a function

```
      function BdeuteronF(kk , j , k , p ,
     -pp , x)
       double precision :: BdeuteronFUN
       integer :: k , j, kk
       double precision :: p
       double precision :: pp
       double precision :: x
```

File ***BdeuteronFUN.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      use Bdeuteron
      ...
      BdeuteronARRAY(kk , j , k) % Bdeuteronpointer(p , pp , x)
      ...
```

with the types:

```
       double precision :: p
       double precision :: pp
       double precision :: x
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeBdeuteron()
```

Both files contain an implementation of (6.4) with $k' =$ `kk` , $k =$ `k` ,$j =$ `j` and $\boldsymbol{p} = (0, 0, |\boldsymbol{p}|)$ $\boldsymbol{p}' = (\sqrt{1-x^2}|\boldsymbol{p}'|, 0, x|\boldsymbol{p}'|)$ with $|\boldsymbol{p}| =$ `p` , $|\boldsymbol{p}'| =$ `pp` and $x =$ `x`.

Additionally the

PROGRAMS/deuteron/

directory contains test programs that can be used with the code. The programs demonstrate function calling and module initializations. They can be used as a template for new programs.

D.2 Building blocks for 3N bound state calculations

Fortran codes that can be used in the 3N bound state calculations can be built by evaluating:

PROGRAMS/bound3N/nb/CinvL.nb

and

PROGRAMS/bound3N/nb/PER.nb

The code produced by the notebooks will be written to:

PROGRAMS/bound3N/

Both fixed format (***.f***) and free format (***.f90***) versions are created.

File ***C1223.f*** contains a function:

```
      function C1223F(i , j , x , p , q)
       double precision :: C1223FUN
       integer :: i , j
       double precision :: x
       double precision , dimension(3):: p
       double precision , dimension(3):: q
```

File ***C1223.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      C1223ARRAY(i , j) % C1223pointer(x , p , q)
```

with the types:

```
       double precision :: x
       double precision , dimension(3):: p
       double precision , dimension(3):: q
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeC1223()
```

Both files contain an implementation of $C^{1223}_{t'T'k;tTi}(\boldsymbol{p'}, \boldsymbol{q'})$ from equation (10.17) with $\texttt{x} = \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}'$, $\texttt{p} = \boldsymbol{P}^{2312}(\boldsymbol{p}', \boldsymbol{q}')$ and $\texttt{q} = \boldsymbol{Q}^{2312}(\boldsymbol{p}', \boldsymbol{q}')$ (see equations (10.10), (10.11)). Each unique set of $t'T'k$ is numbered by `i` and each unique set of tTi is numbered by `j` - detailed information on the ordering can be easily deduced from the notebook.

File ***C1323.f*** contains a function:

```
      function C1323F(i , j , x , p , q)
       double precision :: C1323FUN
       integer :: i , j
       double precision :: x
       double precision , dimension(3):: p
       double precision , dimension(3):: q
```

File ***C1323.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      C1323ARRAY(i , j) % C1323pointer(x , p , q)
```

with the types:

```
       double precision :: x
       double precision , dimension(3):: p
       double precision , dimension(3):: q
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeC1323()
```

Both files contain an implementation of $C^{1323}_{t'T'k;tTi}(\boldsymbol{p}',\boldsymbol{q}')$ from equation (10.22) with $\texttt{x} = \hat{\boldsymbol{p}}' \cdot \hat{\boldsymbol{q}}'$, $\texttt{p} = \boldsymbol{P}^{2313}(\boldsymbol{p}',\boldsymbol{q}')$ and $\texttt{q} = \boldsymbol{Q}^{2313}(\boldsymbol{p}',\boldsymbol{q}')$ (see equations (10.20), (10.21)). As before, the unique set of $t'T'k$ is numbered by `i` and each unique set of tTi is numbered by `j` - detailed information on the ordering can be found the notebook.

File ***CinvL.f*** contains a function:

```
      function CinvLF(i , j , k ,
     - p , q , x , pp , xx , phiphi)
       double precision :: CinvLFUN
       integer :: ind9, ind10, ind11
       double precision :: p
       double precision :: q
       double precision :: x
       double precision :: pp
       double precision :: xx
       double precision :: phiphi
```

File ***CinvL.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      CinvLARRAY(i , j , k) % &
      &CinvLpointer(p , q , x , pp , xx , phiphi)
```

with the types:

```
       double precision :: p
       double precision :: q
       double precision :: x
       double precision :: pp
       double precision :: xx
       double precision :: phiphi
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeCinvL()
```

Both files contain an implementation of $(C^{-1}L)_{kji}(\boldsymbol{p},\boldsymbol{q};\boldsymbol{p},\boldsymbol{p}';\boldsymbol{p}',\boldsymbol{q})$ from equation (10.32) with:

$$\begin{aligned}\boldsymbol{p} &= |\boldsymbol{p}|(0,0,1)\\ \boldsymbol{q} &= |\boldsymbol{q}|(\sqrt{1-x^2},0,x)\\ \boldsymbol{p}' &= |\boldsymbol{p}'|(\sqrt{1-x'^2}sin(\phi'),\sqrt{1-x'^2}cos(\phi'),x'),\end{aligned}$$

$\texttt{p} = |\boldsymbol{p}|$, $\texttt{q} = |\boldsymbol{q}|$, $\texttt{pp} = |\boldsymbol{p}'|$, $\texttt{x} = x$, $\texttt{xx} = x'$ and $\texttt{phiphi} = \phi'$.

We do not include the notebook that was used in the creation of ***CinvEfun.f*** or ***CinvEfun.f90***. Creating these files requires a lot of time in *Mathematica*®

and additional methods were used in order to increase the speed of computation (parallelization). File ***CinvEfun.f*** contains a function:

```
      function CinvEfunF(i1 , i2 , i3 , k , i
     -, x , Q1 , Q2 , Q3 , thetapp
     -, phipp , thetaqq , phiqq)
       double precision :: CinvEfunFUN
       integer :: i1 , i2 , i3 , k , i
       double precision :: x
       double precision , dimension(3):: Q1
       double precision , dimension(3):: Q2
       double precision , dimension(3):: Q3
       double precision :: thetapp
       double precision :: phipp
       double precision :: thetaqq
       double precision :: phiqq
```

File ***CinvEfun.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      CinvEfunARRAY(i1 , i2 , i3 , k , i) % &
      &CinvEfunpointer(x , Q1 , Q2 , Q3 , &
      &thetapp , phipp , thetaqq , phiqq)
```

with types:

```
       double precision :: x
       double precision , dimension(3):: Q1
       double precision , dimension(3):: Q2
       double precision , dimension(3):: Q3
       double precision :: thetapp
       double precision :: phipp
       double precision :: thetaqq
       double precision :: phiqq
```

Both contain an implementation of $(C^{-1}E)_{ki}^{t't''T'}(\boldsymbol{p}'\boldsymbol{q}'\boldsymbol{p}'\boldsymbol{q}'\boldsymbol{p}''\boldsymbol{q}'';\boldsymbol{p}''\boldsymbol{q}'')$ from (10.42) with a NNLO 3N potential from [43]. $\boldsymbol{k}_1' - \boldsymbol{k}_1 =$ `Q1`, $\boldsymbol{k}_2' - \boldsymbol{k}_2 =$ `Q2`, $\boldsymbol{k}_3' - \boldsymbol{k}_3 =$ `Q3` ($\boldsymbol{k}_1, \boldsymbol{k}_2, \boldsymbol{k}_3$ are the individual particle momenta in the initial state and $\boldsymbol{k}_1', \boldsymbol{k}_2', \boldsymbol{k}_3'$ are individual particle momenta in the final state). Additional angles $\theta_{\boldsymbol{p}'} =$ `thetapp`, $\phi_{\boldsymbol{p}'} =$ `phipp`, $\theta_{\boldsymbol{q}'} =$ `thetaqq`, $\phi_{\boldsymbol{q}'} =$ `phiqq`, parametrize the final Jacobi momenta:

$$\boldsymbol{p}' = |\boldsymbol{p}'|(sin(\theta_{\boldsymbol{p}'})cos(\phi_{\boldsymbol{p}'}), sin(\theta_{\boldsymbol{p}'})sin(\phi_{\boldsymbol{p}'}), cos(\theta_{\boldsymbol{p}'}))$$
$$\boldsymbol{q}' = |\boldsymbol{q}'|(sin(\theta_{\boldsymbol{q}'})cos(\phi_{\boldsymbol{q}'}), sin(\theta_{\boldsymbol{q}'})sin(\phi_{\boldsymbol{q}'}), cos(\theta_{\boldsymbol{q}'})).$$

The integer indexes translate literally $k =$ `k`, $i =$ `i`. The values of `i1`, `i2`, `i3` correspond to diffrent initial and final isospin states:

$$i1 = 1, i2 = 1, i3 = 1 \rightarrow \langle\left(0\frac{1}{2}\right)\frac{1}{2} \mid \dots \mid \left(0\frac{1}{2}\right)\frac{1}{2}\rangle,$$
$$i1 = 2, i2 = 1, i3 = 1 \rightarrow \langle\left(1\frac{1}{2}\right)\frac{1}{2} \mid \dots \mid \left(0\frac{1}{2}\right)\frac{1}{2}\rangle,$$
$$i1 = 1, i2 = 2, i3 = 1 \rightarrow \langle\left(0\frac{1}{2}\right)\frac{1}{2} \mid \dots \mid \left(1\frac{1}{2}\right)\frac{1}{2}\rangle,$$
$$i1 = 2, i2 = 2, i3 = 1 \rightarrow \langle\left(1\frac{1}{2}\right)\frac{1}{2} \mid \dots \mid \left(1\frac{1}{2}\right)\frac{1}{2}\rangle,$$

where in $\langle(t\frac{1}{2})\,T \mid$ the isospins of paricles 2 and 3 is coupled to t then this is coupled with the last particle isospin to the total 3N isospin T.

D.3 Building blocks for transition operator calculations

The FORTRAN codes that can be used in the 2N transition operator calculations is constructed by evaluating:

/PROGRAMS/toperator/nb/toperator.nb.

The code produced by the notebook will be written to:

/PROGRAMS/toperator/.

Both fixed format (***.f***) and free format (***.f90***) versions are created.

File ***AinvBfunctionFUN.f*** contains a function:

```
      function AinvBtoperatorF(k , j , jj , p ,
     -pp , xx , ppp , xxx , phi)
        double precision :: AinvBtoperatorFUN
        integer :: k , j , jj
        double precision :: p
        double precision :: pp
        double precision :: xx
        double precision :: ppp
        double precision :: xxx
        double precision :: phi
```

File ***AdeuteronFUN.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      AinvBtoperatorARRAY(k, j, jj) % &
      &AinvBtoperatorpointer(p , pp , xx , ppp , xxx , phi)
```

with the types:

```
        double precision :: p
        double precision :: pp
        double precision :: xx
        double precision :: ppp
        double precision :: xxx
        double precision :: phi
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeAinvBtoperator()
```

Both files contain an implementation of :

$$\sum_{r=1}^{6} A^{-1}_{kr}(|\boldsymbol{p}'|, _{\{|\boldsymbol{p}|\}}, x')B_{rjj'}(|\boldsymbol{p}'|, _{\{|\boldsymbol{p}|\}}, x', |\boldsymbol{p}''|, x'', \phi'') \tag{D.1}$$

from (7.15) using (7.4) , (7.5) and momenta defined in (7.7) with $|\boldsymbol{p}| =$ `p`, $|\boldsymbol{p}'| =$ `pp`, $x' =$ `xx`, $|\boldsymbol{p}''| =$ `ppp`, $x'' =$ `xxx` and $\phi'' =$ `phi`.

Recall that for energies approaching the deuteron binding energy the transition operator will have a singularity (7.26):

$$\check{t}(E \to E_b) = \check{V} \mid \Psi^{2N}_{bound}\rangle \frac{1}{E - E_b} \langle \Psi^{2N}_{bound} \mid \check{V}.$$

Thus for negative energies we will calculate:

$$\check{t}(E)(E - E_b) \tag{D.2}$$

and expect this operator to have a non singular behavior around the deuteron binding energy.

After evaluation, the notebook

PROGRAMS/toperator/nb/deu.nb

will hold definitions for (D.2) for energies around the deuteron binding energy. More specifically it calculates the matrix representation of:

$$\check{V} \mid \Psi^{2N}_{bound}\rangle\langle\Psi^{2N}_{bound} \mid \check{V} \tag{D.3}$$

and then decomposes this operator according to (5.1). The final result is a table `tab` containing values corresponding to (D.2) for different $|\boldsymbol{p}|$, $\boldsymbol{p}'$, x from (7.7).

The grid of $|\boldsymbol{p}|$, $\boldsymbol{p}'$, x to be used use in ***deu.nb*** is read from:

PROGRAMS/toperator/points10XII2012.

This file can be changed. Its format can easily be inferred from the notebook. Deuteron scalar functions are read from:

PROGRAMS/toperator/d1m2.22400

with the first column being the momentum value $|\boldsymbol{p}|$, the second being $\phi_1(|\boldsymbol{p}|)$ and the third being $\phi_2(|\boldsymbol{p}|)$ (from (6.1)).

D.4 Building blocks for current operators

The FORTRAN code that can be used in calculations involving single nucleon current operators can be constructed by evaluating:

PROGRAMS/currents/nb/Single.nb

The code produced by the notebook will be written to:

PROGRAMS/currents/.

Both fixed format (***.f***) and free format (***.f90***) versions are created.

Several files are produced:

J1spinplusFUN.f90 , ***J1spinplusFUN.f***,

J1spinminusFUN.f90 , ***J1spinminusFUN.f***,

J1convplusFUN.f90 , ***J1convplusFUN.f***,

J1convminusFUN.f90 , ***J1convminusFUN.f***

and here we will describe only the first set, since all other codes is similar. File ***J1spinplusFUN.f*** contains a function:

```
      function J1spinplusFUNF(ind , phi1 , phi2 , pF
     -, ppF)
       double complex , dimension(16):: J1spinplusFUNFUN
       integer :: ind
       double precision :: phi1
       double precision :: phi2
       double precision , dimension(3):: pF
       double precision , dimension(3):: ppF
```

File ***J1spinplus.f90.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      J1spinplusFUNARRAY(ind) % &
      &J1spinplusFUNpointer(phi1 , phi2 , pF , ppF)
```

with the types:

```
       double precision :: phi1
       double precision :: phi2
       double precision , dimension(3):: pF
       double precision , dimension(3):: ppF
```

Before the module can be used, it should be initialized by calling:

```
      subroutine initializeJ1spinplusFUN()
```

Both contain an implementation of

$$\left[O^{1\mathrm{N}}(2,\boldsymbol{p}',\boldsymbol{K}')\right]\left[|\,0\,0\rangle\otimes\,|\,1 m_d\rangle\right]$$

from equation (8.8) with `phi1`, `phi2` being the values of the deuteron scalar functions $\phi_l(|\boldsymbol{p}' + \frac{1}{2}\boldsymbol{K}'|)$ with $l = 1, 2$ calculated for the value of the final total and relative momentum $\boldsymbol{K}'$ = `ppF`, $\boldsymbol{p}'$ = `pF`. The value of `ind` determines m_d: `ind` = 1 - $m_d = -1$, `ind` = 2 - $m_d = 0$, `ind` = 3 - $m_d = 1$.

The FORTRAN code that can be used in calculations involving two-nucleon current operators can be constructed by evaluating:

PROGRAMS/currents/nb/Double.nb

The code produced by the notebook will be written to:

PROGRAMS/currents/.

Both fixed format (***.f***) and free format (***.f90***) versions are created.

A number of files can be produced by the notebooks, here we use:

J2NPRC1plusFUN.f90 , ***J2NPRC1plusFUN.f***

that implements the +1 spherical component of a current from [55]. All other codes is similar and can be created by slight modifications to the notebook (all definitions - for other components - are ready after notebook evaluation, all that needs to be added is a couple `Export` directives). File ***J2NPRC1plusFUN.f*** contains a function:

```
      function J2NPRC1plusFUNF(ind , phi1 , phi2 , pF
     - , ppF , pI , ppI)
       double complex , dimension(16):: J2NPRC1plusFUNFUN
       integer :: ind
       double precision :: phi1
       double precision :: phi2
       double precision , dimension(3):: pF
       double precision , dimension(3):: ppF
       double precision , dimension(3):: pI
       double precision , dimension(3):: ppI
```

File ***J2NPRC1plusFUN.f90*** contains a module with an array of function pointers. The individual functions can be called using:

```
      J2NPRC1plusFUNARRAY(ind) % &
      &J2NPRC1plusFUNpointer(phi1 , phi2 , pF , ppF , pI , ppI)
```

with the types:

```
        double precision :: phi1
        double precision :: phi2
        double precision , dimension(3):: pF
        double precision , dimension(3):: ppF
        double precision , dimension(3):: pI
        double precision , dimension(3):: ppI
```

Before the module can be used, it should be initialized by calling:

```
       subroutine initializeJ2NPRC1plusFUN()
```

Both contain an implementation of

$$\left[O^{2\mathrm{N}}(2, \boldsymbol{p}', \boldsymbol{K}')\right] [|\,0\,0\rangle \otimes \mid 1 m_d\rangle]$$

from equation (8.18) with `phi1`, `phi2` being the values of the deuteron scalar functions $\phi_l(|\boldsymbol{p}''|)$ with $l = 1, 2$ calculated for the value of the magnitude of the integral momentum $\boldsymbol{p}'' =$ `pI`. Final total and relative momentum $\boldsymbol{K}' =$ `ppF`, $\boldsymbol{p}' =$ `pF`. The value of `ind` determines m_d as before. In these functions `ppI` is a dummy argument.

Appendix E

Other details

E.1 Reference tables for the different basis vectors

i	$\nu_1^{\text{isospin}}(i)$	$\nu_1^{\text{spin}}(i)$
1	$-\frac{1}{2}$	$-\frac{1}{2}$
2	$-\frac{1}{2}$	$\frac{1}{2}$
3	$\frac{1}{2}$	$-\frac{1}{2}$
4	$\frac{1}{2}$	$\frac{1}{2}$

Table E.1: Reference list of 1N basis states
$\mid \frac{1}{2}, \nu_1^{\text{isospin}}(i)\rangle^{\text{isospin}} \otimes \mid \frac{1}{2}, \nu_1^{\text{spin}}(i)\rangle^{\text{spin}}$
quantum numbers. i is the number of the basis state and $\nu_j^{\text{spin(isospin)}}(i)$ is the projection of the spin (isospin) of particle i.

i	$\nu_1^{\text{isospin}}(i)$	$\nu_2^{\text{isospin}}(i)$	$\nu_1^{\text{spin}}(i)$	$\nu_2^{\text{spin}}(i)$
1	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
2	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
3	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
4	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
5	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
6	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
7	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
8	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
9	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
10	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
11	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
12	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
13	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
14	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
15	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
16	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

Table E.2: Reference list of 2N basis states
$\mid \frac{1}{2}, \nu_1^{\text{isospin}}(i)\rangle^{\text{isospin}} \otimes \mid \frac{1}{2}, \nu_2^{\text{isospin}}(i)\rangle^{\text{isospin}} \otimes \mid \frac{1}{2}, \nu_1^{\text{spin}}(i)\rangle^{\text{spin}} \otimes \mid \frac{1}{2}, \nu_2^{\text{spin}}(i)\rangle^{\text{spin}}$
quantum numbers. i is the number of the basis state and $\nu_j^{\text{spin(isospin)}}(i)$ is the projection of the spin (isospin) of particle i.

i	$\nu_1^{\text{isospin}}(i)$	$\nu_2^{\text{isospin}}(i)$	$\nu_3^{\text{isospin}}(i)$	$\nu_1^{\text{spin}}(i)$	$\nu_2^{\text{spin}}(i)$	$\nu_3^{\text{spin}}(i)$
1	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
2	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
3	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
4	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
5	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
6	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
7	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
8	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
9	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
10	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
11	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
12	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
13	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
14	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
15	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
16	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
17	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
18	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
19	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
20	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
21	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
22	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
23	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
24	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
25	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
26	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
27	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
28	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
29	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
30	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
31	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
32	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
33	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
34	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
35	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
36	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
37	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
38	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
39	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
40	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
41	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
42	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
43	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
44	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
45	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
46	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
47	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
48	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
49	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
50	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
51	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
52	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
53	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
54	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
55	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
56	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
57	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
58	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
59	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
60	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
61	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
62	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
63	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
64	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

Table E.3: Reference list of 3N basis states
$\mid \frac{1}{2}, \nu_1^{\text{isospin}}(i)\rangle^{\text{isospin}} \otimes \mid \frac{1}{2}, \nu_2^{\text{isospin}}(i)\rangle^{\text{isospin}} \otimes \mid \frac{1}{2}, \nu_3^{\text{isospin}}(i)\rangle^{\text{isospin}} \otimes$
$\mid \frac{1}{2}, \nu_1^{\text{spin}}(i)\rangle^{\text{spin}} \otimes \mid \frac{1}{2}, \nu_2^{\text{spin}}(i)\rangle^{\text{spin}} \otimes \mid \frac{1}{2}, \nu_3^{\text{spin}}(i)\rangle^{\text{isospin}}$
quantum numbers. i is the number of the basis state and $\nu_j^{\text{spin(isospin)}}(i)$ is the projection of the spin (isospin) of particle i.

i	$\nu_1^{\text{spin(isospin)}}(i)$
1	$-\frac{1}{2}$
2	$\frac{1}{2}$

Table E.4: Reference list of 1N spin(isospin) basis states
$\mid \frac{1}{2}, \nu_1^{\text{spin(isospin)}}(i)\rangle^{\text{spin(isospin)}}$
quantum numbers. i is the number of the basis state and $\nu_j^{\text{spin(isospin)}}(i)$ is the projection of the spin (isospin) of particle i.

i	$\nu_1^{\text{spin(isospin)}}(i)$	$\nu_2^{\text{spin(isospin)}}(i)$
1	$-\frac{1}{2}$	$-\frac{1}{2}$
2	$-\frac{1}{2}$	$\frac{1}{2}$
3	$\frac{1}{2}$	$-\frac{1}{2}$
4	$\frac{1}{2}$	$\frac{1}{2}$

Table E.5: Reference list of 2N spin(isospin) basis states
$\mid \frac{1}{2}, \nu_1^{\text{spin(isospin)}}(i)\rangle^{\text{spin(isospin)}} \otimes \mid \frac{1}{2}, \nu_2^{\text{spin(isospin)}}(i)\rangle^{\text{spin(isospin)}}$
quantum numbers. i is the number of the basis state and $\nu_j^{\text{spin(isospin)}}(i)$ is the projection of the spin (isospin) of particle i.

i	$\nu_1^{\text{spin(isospin)}}(i)$	$\nu_2^{\text{spin(isospin)}}(i)$	$\nu_3^{\text{spin(isospin)}}(i)$
1	$-\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
2	$-\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
3	$-\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
4	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
5	$\frac{1}{2}$	$-\frac{1}{2}$	$-\frac{1}{2}$
6	$\frac{1}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$
7	$\frac{1}{2}$	$\frac{1}{2}$	$-\frac{1}{2}$
8	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$

Table E.6: Reference list of 3N spin(isospin) basis states
$\mid \frac{1}{2}, \nu_1^{\text{spin(isospin)}}(i)\rangle^{\text{spin(isospin)}} \otimes \mid \frac{1}{2}, \nu_2^{\text{spin(isospin)}}(i)\rangle^{\text{spin(isospin)}} \otimes$
$\mid \frac{1}{2}, \nu_3^{\text{spin(isospin)}}(i)\rangle^{\text{spin(isospin)}}$
quantum numbers. i is the number of the basis state and $\nu_j^{\text{spin(isospin)}}(i)$ is the projection of the spin (isospin) of particle i.

E.2 Expansion functions for the Bonn B potential

The scalar functions for the different parts of the Bonn B potential are (all necessary definitions were given in Chapter 6) [15]:

$$\begin{aligned} O_{ps} &= \left(1+\frac{2m}{E'+E}\right)\left[(\mathbf{p}'\cdot\mathbf{p})^2 - p'^2p^2\right]w_2 + \left(1+\frac{2m}{E'+E}\right)w_4 \\ &\quad +\frac{1}{4}\left\{-(W'-W)^2+\left(1+\frac{2m}{E'+E}\right)\left[p'^2+p^2-2(\mathbf{p}'\cdot\mathbf{p})\right]\right\}w_5 \\ &\quad +\frac{1}{4}\left\{-(W'+W)^2+\left(1+\frac{2m}{E'+E}\right)\left[p'^2+p^2+2(\mathbf{p}'\cdot\mathbf{p})\right]\right\}w_6 \end{aligned} \tag{E.1}$$

$$O_s = -[W'W - (\mathbf{p}'\cdot\mathbf{p})]^2 w_1 - [W'W - (\mathbf{p}'\cdot\mathbf{p})]w_3 + w_4 \tag{E.2}$$

$$\begin{aligned}
O_{vv} = {} & \left\{[W'W + (\mathbf{p}'\cdot\mathbf{p})]^2 + W'^2p^2 + W^2p'^2 + 2W'W(\mathbf{p}'\cdot\mathbf{p})\right\} w_1 \\
& + \Bigg\{ -\frac{1}{2}\left(W'^2 + W^2\right)\left(p'^2 + p^2\right) + 2W'W(\mathbf{p}'\cdot\mathbf{p}) \\
& + \frac{1}{2}\left(1 + \frac{2m}{E'+E}\right)\left[p'^4 + p^4 - 2(\mathbf{p}'\cdot\mathbf{p})^2\right]\Bigg\} w_2 \\
& - [3W'W + (\mathbf{p}'\cdot\mathbf{p})]w_3 - \left(2 + \frac{2m}{E'+E}\right) w_4 \\
& - \frac{1}{4}\left\{-(W'-W)^2 + \left(1 + \frac{2m}{E'+E}\right)[p'^2 + p^2 - 2(\mathbf{p}'\cdot\mathbf{p})]\right\} w_5 \\
& - \frac{1}{4}\left\{-(W'+W)^2 + \left(1 + \frac{2m}{E'+E}\right)[p'^2 + p^2 + 2(\mathbf{p}'\cdot\mathbf{p})]\right\} w_6
\end{aligned} \tag{E.3}$$

$$\begin{aligned}
O_{vt} = {} & \Bigg\{ W'^2p^2 + W^2p'^2 - \frac{W'-W}{2m}\left(W'^2p^2 - W^2p'^2\right) \\
& + 2W'W\left[2W'W + (\mathbf{p}'\cdot\mathbf{p})\right] - \frac{W'+W}{m}\left[W'^2W^2 - (\mathbf{p}'\cdot\mathbf{p})^2\right]\Bigg\} w_1 \\
& + \Bigg\{ 2W'W(\mathbf{p}'\cdot\mathbf{p}) - \frac{1}{2}\left(W'^2 + W^2\right)\left(p'^2 + p^2\right) \\
& + \frac{1}{2}\left(1 + \frac{2m}{E'+E}\right)\left[p'^4 + p^4 - 2(\mathbf{p}'\cdot\mathbf{p})^2\right] \\
& - \frac{1}{2m(E'+E)}\left[W'^2p^4 + W^2p'^4 - \left(W'^2 + W^2\right)(\mathbf{p}'\cdot\mathbf{p})^2\right]\Bigg\} w_2 \\
& - \left[2W'W + \frac{W'+W}{m}(\mathbf{p}'\cdot\mathbf{p})\right] w_3 \\
& + \left\{ -\frac{W'+W}{m} + \frac{1}{2m(E'+E)}\left[W'^2 + W^2 - 2m(W'+W)\right]\right\} w_4 \\
& - \frac{1}{4}\Bigg\{ -(W'-W)^2 + \left(1 + \frac{2m}{E'+E}\right)[p'^2 + p^2 - 2(\mathbf{p}'\cdot\mathbf{p})] \\
& - \frac{1}{m(E'+E)}[W'^2p^2 + W^2p'^2 - \left(W'^2 + W^2\right)(\mathbf{p}'\cdot\mathbf{p})]\Bigg\} w_5 \\
& - \frac{1}{4}\Bigg\{ -(W'+W)^2 + \left(1 + \frac{2m}{E'+E}\right)[p'^2 + p^2 + 2(\mathbf{p}'\cdot\mathbf{p})] \\
& - \frac{1}{m(E'+E)}[W'^2p^2 + W^2p'^2 + \left(W'^2 + W^2\right)(\mathbf{p}'\cdot\mathbf{p})]\Bigg\} w_6
\end{aligned} \tag{E.4}$$

$$\begin{aligned}
O_{tt} = {} & \Bigg\{ [W'W + (\mathbf{p}'\cdot\mathbf{p})]^2 + 2\left(2 - \frac{W'+W}{m}\right)\left[W'^2W^2 - (\mathbf{p}'\cdot\mathbf{p})^2\right] \\
& + \left\{3 + \frac{3[W'W - m(W'+W)] + (\mathbf{p}'\cdot\mathbf{p})}{2m^2}\right\}[W'W - (\mathbf{p}'\cdot\mathbf{p})]^2 \\
& + \left[1 + \frac{(W'-W)^2}{4m^2}\right]\left(W'^2p^2 + W^2p'^2\right) + 2\left[1 - \frac{(W'-W)^2}{4m^2}\right]W'W(\mathbf{p}'\cdot\mathbf{p}) \\
& - \frac{W'-W}{m}\left(W'^2p^2 - W^2p'^2\right)\Bigg\} w_1
\end{aligned}$$

$$
\begin{aligned}
&+\Bigg\{2\left[1-\frac{(W'-W)^2}{4m^2}\right]W'W(\mathbf{p}'\cdot\mathbf{p})-\left[1+\frac{(W'-W)^2}{4m^2}\right]\left(1+\frac{2m}{E'+E}\right)(\mathbf{p}'\cdot\mathbf{p})^2\\
&-\left[1+\frac{(W'-W)^2}{4m^2}\right]\left(\frac{m^2+E'E}{E'+E}+m\right)(W'p^2+Wp'^2)\\
&-\frac{1}{m(E'+E)}\left[W'^2p^4+W^2p'^4-\left(W'^2+W^2\right)(\mathbf{p}'\cdot\mathbf{p})^2\right]\Bigg\}w_2\\
&+\Bigg\{-W'W-(\mathbf{p}'\cdot\mathbf{p})+2\left(2-\frac{W'+W}{m}\right)(\mathbf{p}'\cdot\mathbf{p})-2\left[1-\frac{(W'-W)^2}{4m^2}\right]W'W\\
&+\left\{3+\frac{3[W'W-m(W'+W)]+(\mathbf{p}'\cdot\mathbf{p})}{2m^2}\right\}[W'W-(\mathbf{p}'\cdot\mathbf{p})]\Bigg\}w_3\\
&-\Bigg\{4+\frac{3[W'W-m(W'+W)]+(\mathbf{p}'\cdot\mathbf{p})}{2m^2}-2\left(2-\frac{W'+W}{m}\right)\\
&+\left[1+\frac{(W'-W)^2}{4m^2}\right]\left(1+\frac{2m}{E'+E}\right)-\frac{1}{m(E'+E)}\left(W'^2+W^2\right)\Bigg\}w_4\\
&-\frac{1}{2}\Bigg\{\left[1-\frac{(W'-W)^2}{4m^2}\right]W'W-\frac{1}{m(E'+E)}\left[W'^2p^2+W^2p'^2-\left(W'^2+W^2\right)(\mathbf{p}'\cdot\mathbf{p})\right]\\
&+\frac{1}{2}\left[1+\frac{(W'-W)^2}{4m^2}\right]\left[\left(1+\frac{2m}{E'+E}\right)[p'^2+p^2-2(\mathbf{p}'\cdot\mathbf{p})]-W'^2-W^2\right]\Bigg\}w_5\\
&-\frac{1}{2}\Bigg\{-\left[1-\frac{(W'-W)^2}{4m^2}\right]W'W-\frac{1}{m(E'+E)}\left[W'^2p^2+W^2p'^2+(W'^2+W^2)(\mathbf{p}'\cdot\mathbf{p})\right]\\
&+\frac{1}{2}\left[1+\frac{(W'-W)^2}{4m^2}\right]\left[\left(1+\frac{2m}{E'+E}\right)[p'^2+p^2+2(\mathbf{p}'\cdot\mathbf{p})]-W'^2-W^2\right]\Bigg\}w_6.
\end{aligned}
\tag{E.5}
$$

E.3 Scalar coefficients for the deuteron bound state equation

$$
\begin{aligned}
A^d_{1,1}&=3\\
A^d_{1,2}&=0\\
A^d_{2,1}&=0\\
A^d_{2,2}&=\frac{8p^4}{3}
\end{aligned}
$$

$$B^d_{1,1,1} = 3$$
$$B^d_{1,1,2} = 0$$
$$B^d_{1,2,1} = 3$$
$$B^d_{1,2,2} = 0$$
$$B^d_{1,3,1} = 0$$
$$B^d_{1,3,2} = 0$$
$$B^d_{1,4,1} = -p^2\left(x^2-1\right)p'^2$$
$$B^d_{1,4,2} = \frac{4}{3}p^2\left(x^2-1\right)p'^4$$
$$B^d_{1,5,1} = p^2 + 2pxp' + p'^2$$
$$B^d_{1,5,2} = \frac{4}{3}p'^2\left(p^2\left(3x^2-1\right)+2p'\left(p'+2px\right)\right)$$
$$B^d_{1,6,1} = p^2 - 2pxp' + p'^2$$
$$B^d_{1,6,2} = \frac{4}{3}p'^2\left(p^2\left(3x^2-1\right)+2p'\left(p'-2px\right)\right)$$
$$B^d_{2,1,1} = 0$$
$$B^d_{2,1,2} = \frac{4}{3}p^2\left(3x^2-1\right)p'^2$$
$$B^d_{2,2,1} = 0$$
$$B^d_{2,2,2} = \frac{4}{3}p^2\left(3x^2-1\right)p'^2$$
$$B^d_{2,3,1} = 0$$
$$B^d_{2,3,2} = 8p^3x\left(x^2-1\right)p'^3$$
$$B^d_{2,4,1} = \frac{4}{3}p^4\left(x^2-1\right)p'^2$$
$$B^d_{2,4,2} = -\frac{4}{9}p^4\left(9x^4-14x^2+5\right)p'^4$$
$$B^d_{2,5,1} = \frac{4}{3}p^2\left(2p^2+\left(3x^2-1\right)p'^2+4pxp'\right)$$
$$B^d_{2,5,2} = -\frac{4}{9}p^2p'^2\left(p^2\left(3x^2-1\right)+\left(3x^2-1\right)p'^2+4pxp'\right)$$
$$B^d_{2,6,1} = \frac{4}{3}p^2\left(2p^2+\left(3x^2-1\right)p'^2-4pxp'\right)$$
$$B^d_{2,6,2} = \frac{4}{9}p^2p'^2\left(p^2\left(1-3x^2\right)+\left(1-3x^2\right)p'^2+4pxp'\right)$$

$$A^{-1d}_{1,1} = \frac{1}{3}$$
$$A^{-1d}_{1,2} = 0$$
$$A^{-1d}_{2,1} = 0$$
$$A^{-1d}_{2,2} = \frac{3}{8p^4}$$

E.4 Scalar coefficients for the 2N Lippmann - Schwinger equation

Here we list only the A^{-1} coefficients, the remaining A, B were used in the same form as in [15].

$$A^{-1}_{1,1} = \frac{1}{4}$$
$$A^{-1}_{1,2} = 0$$
$$A^{-1}_{1,3} = 0$$
$$A^{-1}_{1,4} = 0$$
$$A^{-1}_{1,5} = 0$$
$$A^{-1}_{1,6} = 0$$
$$A^{-1}_{2,1} = 0$$
$$A^{-1}_{2,2} = \frac{|\boldsymbol{p}|^4 - 2|\boldsymbol{p}|^2|\boldsymbol{p}'|^2 x'^2 + |\boldsymbol{p}'|^4}{4\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{2,3} = 0$$
$$A^{-1}_{2,4} = \frac{|\boldsymbol{p}|^4 - 2|\boldsymbol{p}|^2|\boldsymbol{p}'|^2 x'^2 + |\boldsymbol{p}'|^4}{4|\boldsymbol{p}'|^2\left(x'^2 - 1\right)\left(|\boldsymbol{p}|^3 - |\boldsymbol{p}||\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{2,5} = -\frac{|\boldsymbol{p}|^2 - 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{2,6} = -\frac{|\boldsymbol{p}|^2 + 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$

$$A^{-1}_{3,1} = 0$$
$$A^{-1}_{3,2} = 0$$
$$A^{-1}_{3,3} = \frac{1}{8|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(x'^2-1\right)}$$
$$A^{-1}_{3,4} = 0$$
$$A^{-1}_{3,5} = 0$$
$$A^{-1}_{3,6} = 0$$
$$A^{-1}_{4,1} = 0$$
$$A^{-1}_{4,2} = \frac{|\boldsymbol{p}|^4 - 2|\boldsymbol{p}|^2|\boldsymbol{p}'|^2x'^2 + |\boldsymbol{p}'|^4}{4|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^3 - |\boldsymbol{p}||\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{4,3} = 0$$
$$A^{-1}_{4,4} = \frac{|\boldsymbol{p}|^4 - |\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(x'^2+1\right) + |\boldsymbol{p}'|^4}{2|\boldsymbol{p}|^4|\boldsymbol{p}'|^4\left(x'^2-1\right)^2\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{4,5} = -\frac{|\boldsymbol{p}|^2 - 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^3 - |\boldsymbol{p}||\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{4,6} = -\frac{|\boldsymbol{p}|^2 + 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^3 - |\boldsymbol{p}||\boldsymbol{p}'|^2\right)^2}$$

$$A^{-1}_{5,1} = 0$$
$$A^{-1}_{5,2} = -\frac{|\boldsymbol{p}|^2 - 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{5,3} = 0$$
$$A^{-1}_{5,4} = -\frac{|\boldsymbol{p}|^2 - 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^3 - |\boldsymbol{p}||\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{5,5} = -\frac{\left(|\boldsymbol{p}|^2 - 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2\right)^2}{32|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{5,6} = \frac{|\boldsymbol{p}|^4 + 2|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(2x'^2-3\right) + |\boldsymbol{p}'|^4}{32|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{6,1} = 0$$
$$A^{-1}_{6,2} = -\frac{|\boldsymbol{p}|^2 + 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{6,3} = 0$$
$$A^{-1}_{6,4} = -\frac{|\boldsymbol{p}|^2 + 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2}{8|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^3 - |\boldsymbol{p}||\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{6,5} = \frac{|\boldsymbol{p}|^4 + 2|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(2x'^2-3\right) + |\boldsymbol{p}'|^4}{32|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$
$$A^{-1}_{6,6} = -\frac{\left(|\boldsymbol{p}|^2 + 2|\boldsymbol{p}||\boldsymbol{p}'|x' + |\boldsymbol{p}'|^2\right)^2}{32|\boldsymbol{p}|^2|\boldsymbol{p}'|^2\left(x'^2-1\right)\left(|\boldsymbol{p}|^2 - |\boldsymbol{p}'|^2\right)^2}$$

E.5 Link to partial wave states

Here we give expressions that can be used to link (4.9) and (4.26) to the partial wave basis states. The formulas below are taken from [45] and are repeated for convenience. Whenever partial wave states are mentioned, this section can provide detailed information.

E.5.1 Two nucleons

We define:

$$[\mid sm_s\rangle]^{\text{2Nspin}} \quad , \quad [\mid tm_t\rangle]^{\text{2Nisospin}}$$

to be states in which the spins (isospins) of the two nucleons are coupled to the total spin (isospin) $s(t) = 0, 1$ with the projection $m_s(m_t) = -s(t), \ldots, s(t)$. Additionally:

$$\mid plm_l\rangle$$

is a state with the relative momentum magnitude p, the relative orbital angular momentum l with the projection m_l. The link between this state with and the three-dimensional vector $\mid \boldsymbol{p}'\rangle$ is given by:

$$\langle \boldsymbol{p}' \mid plm_l\rangle = \frac{\delta(p - |\boldsymbol{p}'|)}{p^2} Y_{lm_l}(\hat{\boldsymbol{p}}'), \tag{E.6}$$

where $Y_{lm}(\hat{\boldsymbol{p}}')$ is the spherical harmonic.

The partial wave states have the spin and the relative angular momentum coupled to the total angular momentum j with its projection m:

$$\mid p(ls)jmtm_t\rangle = \mid p\gamma\rangle = \sum_{m_l m_s} C(lsj, m_l m_s m) \mid plm_l\rangle \otimes [\mid sm_s\rangle] \otimes [\mid tm_t\rangle] \,. \tag{E.7}$$

In numerical calculations all states up to a given $j = j_{max}$ are considered and γ denotes the set of discrete quantum numbers.

E.5.2 Three nucleons

We define

$$\left[\mid (s\frac{1}{2})SM_S\rangle\right]^{\text{3Nspin}}$$

to be a state in which the spin of the subsystem of particles 2 and 3 (s) is coupled with the spin of the particle 1 ($\frac{1}{2}$) to the total spin S with the projection M_S. The 3N isospin state:

$$\left[\mid (t\frac{1}{2})TM_T\rangle\right]^{\text{3Nisospin}}$$

is defined in a similar manner but with the isospin of the subsystem t, the first particle isospin $\frac{1}{2}$ and the total isospin T with the projection M_T.

Additionally to partial-wave states $\mid plm_l\rangle$ describing the relative motion within the subsystem (2,3) we introduce also partial wave states of the relative motion between particle 1 and the (2,3) subsystem: $\mid q\lambda m_\lambda\rangle$. Here q is the magnitude of the relative momentum and λ is the corresponding angular momentum with the projection m_λ. These states are naturally coupled to give the states of the total angular momentum L with the projection M_L:

$$\mid pq(l\lambda)LM_L\rangle .$$

The overlap between the product of three-dimensional momenta $\mid \boldsymbol{p}'\boldsymbol{q}'\rangle$ and the states $\mid pq(l\lambda)LM_L\rangle$ is given by:

$$\langle \boldsymbol{p}'\boldsymbol{q}' \mid pq(l\lambda)LM_L\rangle = \frac{\delta(p-|\boldsymbol{p}'|)}{p|\boldsymbol{p}'|}\frac{\delta(q-|\boldsymbol{q}'|)}{q|\boldsymbol{q}'|}\mathcal{Y}^{LM_L}_{l\lambda}(\hat{\boldsymbol{p}}',\hat{\boldsymbol{q}}'), \tag{E.8}$$

where $\mathcal{Y}^{LM}_{l\lambda}(\hat{\boldsymbol{p}}',\hat{\boldsymbol{q}}')$ is a linear combination of spherical harmonics (see for example [62]):

$$\mathcal{Y}^{LM_L}_{jj'}(\hat{\boldsymbol{p}}',\hat{\boldsymbol{q}}') = \sum_{m=-j}^{j}\sum_{m=-j'}^{j'} C(jj'L, mm'M_L)Y_{jm}(\hat{\boldsymbol{p}}')Y_{j'm'}(\hat{\boldsymbol{q}}'). \tag{E.9}$$

Finally, we define the following set of states:

$$\begin{aligned} \mid pq(l\lambda)L(s\tfrac{1}{2})S(LS)JM(t\tfrac{1}{2})TM_T\rangle \equiv \mid pq\beta\rangle = \\ \sum_{M_LM_S} C(LSJ, M_LM_SM) \\ \mid pq(l\lambda)LM_L\rangle \otimes \left[\mid (s\tfrac{1}{2})SM_S\rangle\right] \otimes \left[\mid (t\tfrac{1}{2})TM_T\rangle\right], \end{aligned} \tag{E.10}$$

where we couple the total orbital angular momentum L and the total spin S to the total 3N angular momentum J with the projection M. In (E.10) β denotes the whole set of discrete quantum numbers.

An alternative set of states can be created by coupling l and s to the total $(2,3)$ subsystem angular momentum j. Parallel to that, the orbital angular momentum λ is coupled with the spin of particle 1 to its total angular momentum I. Finally, j and I are coupled to the total 3N angular momentum J with the projection M. Such states are denoted as $\mid pq\alpha\rangle$, where α comprises information about $l, s, j, \lambda, I, J, M, t, T, M_T$. The link between (E.10) and the new states are given by [63]:

$$\begin{aligned} \mid pq(ls)j(\lambda\tfrac{1}{2})I(jI)JM(t\tfrac{1}{2})T\rangle \equiv \mid pq\alpha\rangle = \\ \sum_{LS}\sqrt{(2j+1)(2I+1)(2L+1)(2S+1)} \\ \left\{\begin{matrix} l & s & j \\ \lambda & \frac{1}{2} & I \\ L & S & J \end{matrix}\right\} \mid pq(l\lambda)L(s\tfrac{1}{2})S(LS)JM(t\tfrac{1}{2})T\rangle, \end{aligned} \tag{E.11}$$

where $\{\ldots\}$ is the Wigner $9j$ symbol. Since J, M and the pairity $\Pi = (-1)^{l+\lambda}$ are constants of motion, in numerical calculations states with given J, M and Π can be considered separately. For numerical implementations channels with $J \leq J_{max}$ and 2N channels with $j \leq j_{max}$ are considered.

The partial wave decomposition of new forces and current operators is quite challenging. Our *Mathematica*® framework can be used to calculate the complicated isospin - spin matrix elements making the decomposition easier [58, 59, 60].

E.5.3 Connection with our three-dimensional calculations

As mentioned earlier, the S and D components ψ_0, ψ_2 of the deuteron wave function are directly related to the scalar functions ϕ_1, ϕ_2 from Chapter 6. The link is [22]:

$$\psi_0(|\boldsymbol{p}|) = \sqrt{4\pi}\phi_1(|\boldsymbol{p}|) \tag{E.12}$$

$$\psi_2(|\boldsymbol{p}|) = \frac{4\sqrt{2}|\boldsymbol{p}|^2}{3}\phi_2(|\boldsymbol{p}|). \tag{E.13}$$

The relation of our three-dimensional 3N bound state results and the partial-wave projected triton using (E.11) is more complicated. The link between the two representations was worked out in [35], the final expression being:

$$\begin{aligned}
&\langle pq\alpha|\Psi m m_t\rangle = \\
&\delta_{M_T,m_t} \sum_{L,S} \sqrt{(2j+1)(2I+1)(2L+1)(2S+1)} \left\{ \begin{array}{ccc} l & s & j \\ \lambda & \frac{1}{2} & I \\ L & S & J \end{array} \right\} \\
&\times \sum_{M_L=-L}^{L} C(LSJ; M_L, M-M_L, M) \int_0^{\pi} d\theta_p \sin\theta_p \int_0^{2\pi} d\phi_p \int_0^{\pi} d\theta_q \sin\theta_q \int_0^{2\pi} d\phi_q \\
&\times \mathcal{Y}^{*\,LM_L}_{l\lambda}(\hat{\boldsymbol{p}}, \hat{\boldsymbol{q}}) \sum_{i=1}^{8} \phi^{(i)}_{tT}(\boldsymbol{p}, \boldsymbol{q}) \\
&\times \langle \left(s\frac{1}{2}\right) SM-M_L| \left[\check{O}_i(\boldsymbol{p}, \boldsymbol{q})\right] |\chi^m\rangle,
\end{aligned} \tag{E.14}$$

where $|\ \Psi m m_t\rangle$ is the full wave function of the 3N bound state (m is the spin projection and m_t is the isospin projection) and ϕ are the scalar functions that define the 3N bound state in operator form (10.3). The integrals are over all possible directions of $\boldsymbol{p}$ and $\boldsymbol{q}$.

The transition operator calculated using the 3D approach in Chapter 7 can also be transformed to the partial-wave basis from (E.7). The link was given in [58]:

$$\begin{aligned}
&\langle p'(l's)jm_j \mid V^{tm_t} \mid p(ls)jm_j\rangle = \\
&\quad = \int d\boldsymbol{p}' \int d\boldsymbol{p} \sum_{m_l'} C(l', s, j; m_l', m_j - m_l', m_j) \\
&\quad \times \sum_{m_l} C(l, s, j; m_l, m_j - m_l, m_j)\ Y^*_{l'\,m_l'}(\hat{\boldsymbol{p}}')\ Y_{l\,m_l}(\hat{\boldsymbol{p}}) \\
&\quad \times \langle s\, m_j - m_l' \mid \left[V^{tm_t}(\boldsymbol{p}', \boldsymbol{p})\right] \mid s\, m_j - m_l\rangle,
\end{aligned} \tag{E.15}$$

first for the 2N potential but the same expression can be used for the transition operator. $V^{tm_t}(\boldsymbol{p}', \boldsymbol{p})$ is a matrix element of the 2N isospin projected ($|\ tm_t\rangle$) potential (or transition operator) between $|\ \boldsymbol{p}'\rangle$ and $|\ \boldsymbol{p}\rangle$. Actually the above expression can be greatly simplified, see [58].

List of Figures

Acknowledgments

INNOWACYJNA GOSPODARKA
NARODOWA STRATEGIA SPÓJNOŚCI

UNIWERSYTET JAGIELLOŃSKI W KRAKOWIE

FNP Fundacja na rzecz Nauki Polskiej

UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ ROZWOJU REGIONALNEGO

INTERNATIONAL PHD PROJECTS IN APPLIED NUCLEAR PHYSICS AND INNOVATIVE TECHNOLOGIES

This project is supported by the Foundation for Polish Science – MPD program, co-financed by the European Union within the European Regional Development Fund

We acknowledge support by the Foundation for Polish Science - MPD program, co-financed by the European Union within the European Regional Development Fund. Part of the numerical calculations have been performed on the Supercomputing cluster of the JSC, Jülich, Germany.

Kacper Topolnicki would like to thank professor *Jacek Golak* for a fruitful collaboration over the course of his PHD studies. Finally, we would also like to mention and thank all our collaborators: *Roman Skibiński*, *Henryk Witała*, *Andreas Nogga*, *Hiroyuki Kamada* and last but not least *Laura Marcucci*.

Bibliography

[1] H. Witała, J.Golak, R. Skibiński, and K. Topolnicki. Calculations of three-nucleon reactions with N^3LO chiral forces: achievements and challenges. *arXiv 1310.0198, accepted for publication in Journal of Physics G.*

[2] J. Holz and W. Glöckle. Time-independent and time-dependent potential scattering without angular momentum decomposition. *Journal of Computational Physics*, 76(1):131 – 158, 1988.

[3] J. Holz and W. Glöckle. Nucleon-nucleon scattering in a time-dependent treatment. *Physical Review C*, 37:1386–1402, 1988.

[4] D. Hüber, W. Glöckle, and A. Bömelburg. Quasielastic electron scattering on a two-nucleon model system: Scaling and cumulant expansion of the structure function. *Physical Review C*, 42:2342–2357, 1990.

[5] R.A. Malfliet and J.A. Tjon. Solution of the Faddeev equations for the triton problem using local two-particle interactions. *Nuclear Physics A*, 127(1):161 – 168, 1969.

[6] Ch. Elster, J. H. Thomas, and W. Glöckle. Two-body t-matrices without angular-momentum decomposition: Energy and momentum dependences. *Few-Body Systems*, 24(1):55–79, 1998.

[7] R.A. Rice and Y.E. Kim. Formulation of few-body equations without partial waves. *Few-Body Systems*, 14(3):127–148, 1993.

[8] R. Machleidt, K. Holinde, and Ch. Elster. The Bonn meson-exchange model for the nucleonnucleon interaction. *Physics Reports*, 149(1):1 – 89, 1987.

[9] I. Fachruddin, Ch. Elster, and W. Glöckle. Nucleon-nucleon scattering in a three dimensional approach. *Physical Review C*, 62:044002, 2000.

[10] R. Machleidt. *The Meson Theory of Nuclear Forces and Nuclear Structure*, volume 19 of *Advances in Nuclear Physics*. Springer US, 1989.

[11] S. Bayegan, M. A. Shalchi, and M. R. Hadizadeh. Three dimensional calculations of NN bound and scattering states with a chiral potential up to N^3LO. *Physical Review C*, 79:057001, 2009.

[12] M. Rodriguez Gallardo, A. Deltuva, E. Cravo, R. Crespo, and A. C. Fonseca. Two-body scattering without angular-momentum decomposition. *Physical Review C*, 78:034602, 2008.

[13] G.L. Caia, V. Pascalutsa, and L.E. Wright. Solving potential scattering equations without partial wave decomposition. *Physical Review C*, 69:034003, 2004.

[14] G. Ramalho, A. Arriaga, and M. T. Peña. Solution of the spectator equation for relativistic nn scattering without partial wave expansion. *Few-Body Systems*, 39(3-4):123–157, 2006.

[15] J. Golak, W. Glöckle, R. Skibiński, H. Witała, D. Rozpędzik, K. Topolnicki, I. Fachruddin, Ch. Elster, and A. Nogga. Two-nucleon systems in three dimensions. *Physical Review C*, 81:034006, 2010.

[16] Evgeny Epelbaum. Few-nucleon forces and systems in chiral effective field theory. *Progress in Particle and Nuclear Physics*, 57(2):654 – 741, 2006.

[17] E. Epelbaum, H.-W. Hammer, and Ulf-G. Meißner. Modern theory of nuclear forces. *Review of Modern Physics*, 81:1773–1825, 2009.

[18] S. Veerasamy, Ch. Elster, and W.N. Polyzou. Two-nucleon scattering without partial waves using a momentum space argonne V18 interaction. *Few-Body Systems*, 54(12):2207–2225, 2013.

[19] R. Skibiński, J. Golak, D. Rozpędzik, K. Topolnicki, H. Witała, W. Glöckle, A. Nogga, E. Epelbaum, H. Kamada, Ch. Elster, and I. Fachruddin. Recent developments of a three-dimensional description of the nn system. *Few-Body Systems*, 50(1-4):279–281, 2011.

[20] J. Golak, R. Skibiński, H. Witała, K. Topolnicki, W. Glöckle, A. Nogga, and H. Kamada. Different methods for the two-nucleon t-matrix in the operator form. *Few-Body Systems*, 53(3-4):237–252, 2012.

[21] K. Topolnicki, J. Golak, R. Skibiński, A.E. Elmeshneb, W. Glöckle, A. Nogga, and H. Kamada. Deuteron disintegration in three dimensions. *Few-Body Systems*, 54(12):2233–2253, 2013.

[22] I. Fachruddin, Ch. Elster, and W. Glöckle. New forms of deuteron equations and wave function representations. *Physical Review C*, 63:054003, 2001.

[23] W. Glöckle, Ch. Elster, J. Golak, R. Skibiński, H. Witała, and H. Kamada. A new treatment of 2N and 3N bound states in three dimensions. *Few-Body Systems*, 47(1-2):25–38, 2010.

[24] Ch. Elster, W. Schadow, A. Nogga, and W. Glöckle. Three body bound state calculations without angular momentum decomposition. *Few-Body Systems*, 27(2):83–105, 1999.

[25] H. Liu, Ch. Elster, and W. Glöckle. Model study of three-body forces in the three-body bound state. *Few-Body Systems*, 33(4):241–258, 2003.

[26] M. R. Hadizadeh and S. Beyagan. Proceedings of the 3rd Asia- Pacific Conference, Nakhon Ratchasima, Thailand, July 2005 (World Scientific, Singapore, 2007). page 16.

[27] S. Bayegan, M. R. Hadizadeh, and M. Harzchi. Three-nucleon bound state in a spin-isospin dependent three dimensional approach. *Physical Review C*, 77:064005, 2008.

[28] S. Bayegan, M. R. Hadizadeh, and M. Harzchi. A realistic three-dimensional calculation of the 3H binding energy. *Few-Body Systems*, 44(1-4):65–67, 2008.

[29] M. R. Hadizadeh, L. Tomio, and S. Beyagan. AIP Conference Proceedings 1265 (2010). page 84.

[30] M. R. Hadizadeh and S. Bayegan. Four-body bound-state calculations in three-dimensional approach. *Few-Body Systems*, 40(3-4):171–191, 2007.

[31] M. R. Hadizadeh, L. Tomio, and S. Bayegan. Solutions of the bound-state Faddeev-Yakubovsky equations in three dimensions by using NN and 3N potential models. *Physical Review C*, 83:054004, 2011.

[32] M.R. Hadizadeh and S. Bayegan. Bound-state calculations of the three-dimensional Yakubovsky equations with the inclusion of three-body forces. *The European Physical Journal A*, 36(2):201–209, 2008.

[33] S. Bayegan, M.R. Hadizadeh, and W. Glöckle. A realistic formalism for 4N bound state in a three-dimensional Yakubovsky scheme. *Progress of Theoretical Physics*, 120(5):887–916, 2008.

[34] I. Fachruddin, W. Glöckle, Ch. Elster, and A. Nogga. Operator form of 3H (3He) and its spin structure. *Physical Review C*, 69:064002, 2004.

[35] J. Golak, K. Topolnicki, R. Skibiński, W. Glöckle, H. Kamada, and A. Nogga. A three-dimensional treatment of the three-nucleon bound state. *Few-Body Systems*, 54(12):2427–2446, 2013.

[36] R. B. Wiringa, V. G. J. Stoks, and R. Schiavilla. Accurate nucleon-nucleon potential with charge-independence breaking. *Physical Review C*, 51:38–51, 1995.

[37] E. Epelbaum, W. Glöckle, and Ulf-G. Meißner. The Two-nucleon system at next-to-next-to-next-to-leading order. *Nuclear Physics A*, 747:362–424, 2005.

[38] Wolfram Research Inc. Mathematica version 8.0. *Wolfram Research Inc. Champaign, Illinois*, 2010.

[39] S.A. Coon, M.D. Scadron, P.C. McNamee, B.R. Barrett, D.W.E. Blatt, and B.H.J. McKellar. The two-pion-exchange three-nucleon potential and nuclear matter. *Nuclear Physics A*, 317(1):242 – 278, 1979.

[40] R.G. Ellis, S.A. Coon, and B.H.J. McKellar. π- and ρ-exchange three-nucleon potentials (i). *Nuclear Physics A*, 438(34):631 – 668, 1985.

[41] S. A. Coon and W. Glöckle. Two-pion-exchange three-nucleon potential: Partial wave analysis in momentum space. *Physical Review C*, 23:1790–1802, 1981.

[42] S. A. Coon and M. T. Peña. Momentum and coordinate space three-nucleon potentials. *Physical Review C*, 48:2559–2575, 1993.

[43] E. Epelbaum, A. Nogga, W. Glöckle, H. Kamada, Ulf-G. Meißner, and H. Witała. Three-nucleon forces from chiral effective field theory. *Physical Review C*, 66:064001, 2002.

[44] E. Belbruno. *Capture Dynamics and Chaotic Motions In Celestial Mechanics.* Princton University Press, Princton, 2004.

[45] W. Glöckle. *The Quantum Mechanical Few-Body Problem.* Springer-Verlag, Berlin-Heidelberg, 1983.

[46] Ch. Elster. Lectures on few body systems. Available online at `www.phy.ohiou.edu/~elster/`.

[47] W.Glöckle, H.Witała, D.Hüber, H.Kamada, and J.Golak. The three-nucleon continuum: achievements, challenges and applications. *Physics Reports*, 274(34):107 – 285, 1996.

[48] L. Wolfenstein. Possible triple-scattering experiments. *Physical Review*, 96:1654–1658, 1954.

[49] S.D. Drell J.D. Bjorken. *Relativistic Quantum Mechanics.* McGraw-Hill Book Company, 1964.

[50] D. Rozpędzik. Phd thesis, Jagiellonian University, Cracow. *unpublished*, 2010.

[51] D.O. Riska. Isovector electromagnetic exchange currents and the nucleon - nucleon interaction. *Physica Scripta*, 31:471, 1985.

[52] R. Schiavilla, V. R. Pandharipande, and D. O. Riska. Magnetic form factors of the trinucleons. *Physical Review C*, 40:2294–2309, 1989.

[53] R. Schiavilla, V. R. Pandharipande, and D. O. Riska. Charge form factors of the three- and four-body nuclei. *Physical Review C*, 41:309–317, 1990.

[54] S. Kölling, E. Epelbaum, H. Krebs, and Ulf-G. Meißner. Two-pion exchange electromagnetic current in chiral effective field theory using the method of unitary transformation. *Physical Review C*, 80:045502, 2009.

[55] S. Kölling, E. Epelbaum, H. Krebs, and Ulf-G. Meißner. Two-nucleon electromagnetic current in chiral effective field theory: One-pion exchange and short-range contributions. *Physical Review C*, 84:054008, 2011.

[56] L.E. Marcucci, M. Piarulli, M. Viviani, L. Girlanda, A. Kievsky, et al. Muon capture on deuteron and ^{3}He. *Physical Review C*, 83:014002, 2011 and references therein.

[57] G. Audi and A.H. Wapstra. The 1995 update to the atomic mass evaluation. *Nuclear Physics A*, 595(4):409 – 480, 1995.

[58] J. Golak, D. Rozpędzik, R. Skibiński, K. Topolnicki, H. Witała, W. Glöckle, A. Nogga, E. Epelbaum, H. Kamada, Ch. Elster, and I. Fachruddin. A new way to perform partial-wave decompositions of few-nucleon forces. *The European Physical Journal A*, 43(2):241–250, 2010.

[59] R. Skibiński, J. Golak, K. Topolnicki, H. Witała, H. Kamada, W. Glöckle, and A. Nogga. The Tucson-Melbourne three-nucleon force in the automatized partial-wave decomposition. *The European Physical Journal A*, 47(4):1–16, 2011.

[60] R. Skibiński, J. Golak, K. Topolnicki, H. Witała, E. Epelbaum, W. Glöckle, H. Krebs, A. Nogga, and H. Kamada. Triton with long-range chiral N^3LO three-nucleon forces. *Physical Review C*, 84:054005, 2011.

[61] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS Publishing Company, 1996.

[62] V. F. Weisskopf J. M. Blatt. *Theoretical Nuclear Physics.* Springer-Verlag, New York Heidelberg Berlin, 1979.

[63] A. R. Edmonds. *Angular Momentum in Quantum Mechanics.* Princton University Press, New Jersey, 1957.