Thesis for the degree of Doctor of Philosophy

JupiterNCSM: A Pantheon of Nuclear Physics

—an implementation of three-nucleon forces in the no-core shell model—

Tor Djärv

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2021

JupiterNCSM: A Pantheon of Nuclear Physics

—an implementation of three-nucleon forces in the no-core shell model—

Tor Djärv

ISBN 978-91-7905-552-3

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5019
ISSN 0346-718X

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

Cover:
Eigenvector continuation emulators can be used for fast and accurate solutions of the quantum many-body problem. The cover image shows the 32×32 subspace-projected matrix for the three-nucleon contact operator used in the construction of the 17-parameter eigenvector-continuation emulator for ^{6}Li discussed in chapter 5. The full Hamiltonian matrix in the $N_{\max} = 10$ model space has the dimension 10^7.

Printed in Sweden by
Reproservice
Chalmers Tekniska Högskola
Göteborg, Sweden, 2021

Abstract

It is well established that three-nucleon forces (3NFs) are necessary for achieving realistic and accurate descriptions of atomic nuclei. In particular, such forces arise naturally when using chiral effective field theories (χEFT). However, due to the huge computational complexity associated with the inclusion of 3NFs in many-body methods they are often approximated or neglected completely. In this thesis, three different methods to include the physics of 3NFs in the ab initio no-core shell-model (NCSM) have been implemented and tested. In the first method, we approximate the 3NFs as effective two-body operators by exploiting Wick's theorem to normal order the 3NF relative a harmonic-oscillator Slater determinant reference state and discarding the remaining three-body term. We explored the performance of this single-reference normal-ordered two-body approximation on the ground-state energies of the two smallest closed-core nuclei, ^{4}He and ^{16}O, in particular focusing on consequences of the breaking of translational symmetry. The second approach is a full implementation of 3NFs in a new NCSM code, named JupiterNCSM, that we provide as an open-source research software. We have validated and benchmarked JupiterNCSM against other codes and we have specifically used it to investigate the effects of different 3NFs on light p-shell nuclei ^{6}He and ^{6}Li. Finally, we implement the eigenvector continuation (EVC) method to emulate the response of ground-state energies of the aforementioned $A = 6$ nuclei to variations in the low-energy constants of χEFT that parametrize the 3NFs. In this approach, the full Hamiltonian is projected onto a small subspace that is constructed from a few selected eigenvectors. These training vectors are computed with JupiterNCSM in a large model space for a small set of parameter values. This thesis provides the first EVC-based emulation of nuclei computed with a Slater-determinant basis. After the training phase, we find that EVC predictions offer a very high accuracy and more than seven orders of magnitude computational speedup. As a result we are able to perform rigorous statistical inferences to explore the effects of 3NFs in nuclear many-body systems.

Errata

- On page 6 in equation (2.4) the quantum numbers should be indexed with i and k such that

$$|\alpha_{i,k}\rangle = |n_{i,k}, l_{i,k}, j_{i,k}, m_{i,k}, t_{z,i,k}\rangle \,. \tag{2.4}$$

- On page 14 in equation (3.11) the frequency $\bar{\omega}$ is just a dummy variable used to define the function $\hat{H}_{\mathrm{CM}}(\bar{\omega})$, and should not be confused with ω or Ω.

- On page 26 in equation (4.11) there is a missing factor of $\frac{1}{4}$ to handle the over counting due to the antisymmetry of the particle permutation, likewise in equation (4.15) there should be a factor of $\frac{1}{36}$.

- On page 28 equation (4.18) the fifth line, concerning the three-nucleon interaction, should be

$$\sum_{i=1}^{D_\pi} \sum_{j=1}^{D_\nu} \sum_{m=1}^{D_\nu} \sum_{x} K_{n(i,j)} M_x^{nnn} \langle \nu_m | \hat{t}_x^{nnn} | \nu_j \rangle \left(|\pi_j\rangle \otimes |\nu_m\rangle \right). \tag{4.18}$$

- On page 29 algorithm 3 and page 30 algorithm 4, there should be two different neutron dimensions one for the ket side and one for the bra side, such that $n' \leftarrow i \cdot D_n + m$ becomes $n' \leftarrow i \cdot D'_n + m$ in algorithm 3 and $n' \leftarrow l \cdot D_n + m$ is $n' \leftarrow l \cdot D'_n + m$.

- On page 44 in equation (5.6) the index i is used both to index the approximate full-space eigenvector and as a summation index. We suggest to change the summation index to k such that the equation is

$$\left|\psi_i^S(\mathbf{c})\right\rangle = \sum_{k=1}^{\mu} v_k(\mathbf{c}) \left|\phi_k\right\rangle \,. \tag{5.6}$$

Publications

This thesis consists of an extended summary and the following appended papers:

Paper I: *"Normal-ordering approximations and translational (non) invariance"*
T. Djärv, A. Ekström, C. Forssén, and G. R. Jansen
Phys. Rev. C.104.024324 (2021)

Paper II: *"Fast & rigorous predictions for $A = 6$ nuclei with Bayesian posterior sampling"*
T. Djärv, A. Ekström, C. Forssén, and H. T. Johansson
Submitted for publication in Phys. Rev. C; arXiv: 2108.13313

Paper III: *"Exploring non-implausible nuclear-matter predictions with Δ-full chiral interactions"*
W. Jiang, C. Forssén, T. Djärv, A. Ekström, G. Hagen, and T. Papenbrock
in preparation (2021)

This thesis is also based on the following research software. The codes are not appended but are available for download via the provided hyperlinks.

Code I: *"JupiterNCSM"*
T. Djärv, C. Forssén, A. Ekström, and H. T. Johansson
License: GPLv2
https://github.com/thundermoose/JupiterNCSM

Code II: *"Anicre"*
D. Sääf, C. Forssén, H. T. Johansson, T. Djärv
License: GPLv2
https://github.com/thundermoose/Anicre

Statement of contribution

Paper I: I developed the cNO2B code used to make the normal-ordered two-body (NO2B) approximated interaction files, based on the pyNO2B code developed by D. Fahlin Strömberg [1] and related works. I also developed the code that generates the H_{CM} using Moshinsky brackets. I contributed with some of the core ideas for the study and performed all the many-body calculations, except for ^{16}O with full 3NFs which where run by C. Forssén. I wrote the first draft of the manuscript, and was involved in all iterations of the text. I also generated most figures.

Paper II: I developed the JupiterNCSM code, performed the NCSM calculations and constructed the EVC-emulators. I also setup and run all the JupiterNCSM calculations for the ^{4}He, ^{6}He and ^{6}Li nuclei. I wrote the parts of the text concerning the NCSM-calculations and the construction and validation of the EVC emulators, and I generated the figures concerning the validity of the EVC emulators.

Paper III: My role was to use JupiterNCSM to construct the EVC-emulators for the $A = 6$ observables. I also generated data for figures.

Code I: I am the primary developer of the entire code and the maintainer of the git-repository. I have implemented the Lanczos algorithm, the matrix-vector multiplication with matrix elements evaluation on the fly, the transformation of the nuclear forces from J-scheme to M-scheme, the projection of the Hamiltonian matrix on to the EVC subspace, and other utility features such as user interfaces. I also parallelized the code using OpenMP.

Code II: While the mechanism for computing the transition densities where already implemented by D. Sääf [2], I modified the code such that it stores the densities as the index lists needed for JupiterNCSM. I also modified parts of the code such that it could be run in parallel.

Acknowledgment

I would like to dedicate this work to the two people that means the most to me, my mother, Margareta Djärv, and my father, Stefan Gustafsson. For their invaluable advice and support I would like to thank my two supervisors, Christian Forssén and Andreas Ekström, without them these projects would have been impossible. I would also like to thank my examiner, Gabriele Ferretti, for the support and interest in my work. For his help and advice on high performance computing and software development in general I would like to thank Håkan T. Johansson. I would extend a thanks to all postdocs and fellow PhD-students I have had the honor to work alongside over the last five years. I thank all the administrative staff at Chalmers for their help with managing the bureaucracy. I would also like to extend a thanks to our international colleagues, in particular at Oakridge National Laboratory Tennessee and at Technishe Universität Darmstadt, for their help with benchmarking our software and insightful discussions about physics. Lastly, I thank Linus Torvalds for the version control system Git, which has saved me multiple times when I accidentally destroyed important parts of my codes or texts.

Contents

Chapter 1

Introduction

One of the greatest discoveries in modern physics is that of the atomic nucleus [3–5]. It was early shown that it consists of electrically positive particles, today known as protons. However, at the time (when Rutherford conducted his scattering experiments in the early years of the 20th century) the only two known fundamental forces of nature where electromagnetism and gravity, and it was understood that the nucleus would disintegrate under these two forces. Therefore, a new fundamental force was needed to explain how atomic nuclei could hold together. This force became known as the strong nuclear force and provides an interaction between protons and an electrically neutral particle, today known as the neutron. However, this force alone could not explain nuclear phenomena such as beta decay [6], and a fourth force was needed, now known as the weak nuclear force. The latter force, however, will not be discussed further in this work.

The most fundamental description of the strong nuclear force today is Quantum-Chromo Dynamics (QCD) [7–9]. It describes the force as interactions between six different fermionic particles known as quarks, where the force is mediated through vector bosons known as gluons. QCD is a non-abelian gauge theory under the $SU(3)$ symmetry group, which has the consequence that the interaction of the theory is strong at low momenta, where we observe atomic nuclei [10], such that perturbation theory cannot be applied. Moreover, at low momenta the quarks and gluons condense into composite particles: mesons (such as pions) and baryons (such as protons and neutrons).

To handle the non-perturbative nature of QCD, it is common to use an effective field theory (EFT) [11–13] that incorporates the degrees of freedom that are more suitable at the low momentum scale, but still respects symmetries (and symmetry breakings) of the underlying theory. The most common class of EFT used in ab inito nuclear physics is chiral EFT (χEFT) [10–15] that incorporates the chiral symmetry of massless quarks, but allows for weak and explicit breaking of this symmetry due to the quark mass terms. The degrees of freedom in χEFT are most commonly protons and neutrons while the force is mediated by pions. However, other degrees of freedom can also be incorporated, for instance in delta-full χEFT, that is used in Paper III, where nucleons are allowed to be excited to delta particles in intermediate states.

The nuclear forces derived from χEFT are ordered in an infinite expansion of increasing powers of $\frac{Q}{\Lambda_\chi}$ where $\Lambda_\chi \sim 1\,\text{GeV}$ is the chiral symmetry breaking scale and Q is the low energy scale of the physics to be described. Usually, it is assumed that $Q = \max(p, m_\pi)$, where p is internal momentum in the Feynman diagrams and m_π is the pion mass [10, 14, 15]. In practice this expansion is truncated at a finite power of $\frac{Q}{\Lambda_\chi}$, and all possible Feynman diagrams up to that order defines the interaction. In this work we have included interactions up to next-to-next-to-leading order (NNLO) in Weinberg power counting [11–13].

An interesting property of χEFT is that it predicts the existence of many-nucleon forces, i.e., forces that only act between multiple nucleons. The main focus of this work is the treatment of three-nucleon forces (3NF) in the no-core shell model (NCSM). These appear at NNLO in χEFT without deltas, as shown in figure 1.1a. In delta-full χEFT a fourth diagram appears, already at next-to-leading order (NLO), depicted in figure 1.1b. Different terms in the chiral expansion are associated with different strength parameters, known as low-energy constants (LEC). For the three NNLO 3NF diagrams, in figure 1.1a, there are in total five LECs that govern their strength. The two-pion exchange diagram connects to three LECs, c_1, c_3 and c_4, which also govern the strength of some two-nucleon force (2NF) diagrams. The strength of the one-pion exchange contact diagram is governed by c_D and, the full 3NF contact diagram's strength is decided by the c_E.

The atomic nucleus is a non-relativistic, quantum-mechanical many-body system and its structure can therefore be described with the many-body Schrödinger equation (MBSE)

$$\hat{H}\,|\Psi\rangle = E\,|\Psi\rangle\,, \tag{1.1}$$

where E is the system energy and $|\Psi\rangle$ is the system state. The Hamiltonian,

$$\hat{H} = \hat{T}_{\text{int}} + \hat{V}_{\text{2NF}} + \hat{V}_{\text{3NF}}, \tag{1.2}$$

is constructed in this work as a sum of an intrinsic kinetic energy term, $\hat{T}_{\text{int}}$ and 2NF and 3NF potentials, $\hat{V}_{\text{2NF}}$ and $\hat{V}_{\text{3NF}}$. As discussed earlier the potentials are derived from χEFT.

To solve the MBSE, we employ the NCSM [16]. The NCSM is related to the nuclear shell model (SM) [17, 18], which models the states of the nucleus analogously to the electronic shells of the atom. Each nucleon, proton or neutron, is assumed to occupy energy shells defined by a single-particle central-potential. In the NCSM, all particles are treated as active degrees of freedom and the basis is constructed from eigenstates of the harmonic-oscillator (HO), but unlike the SM there is no mean-field central potential added to the system Hamiltonian. In this basis the MBSE is turned into a large matrix diagonalization problem that is suitable to be solved by a computer. The NCSM is discussed in detail in chapter 2. Furthermore, the implementation of the NCSM developed in this work, resulting in Code I named JupiterNCSM, is described in chapter 4.

The inclusion of 3NFs in the MBSE comes with a significant penalty in computational complexity , compared to a Hamiltonian with only 2NFs due to the combinatorics of choosing three particles instead of two. Therefore, it is motivated

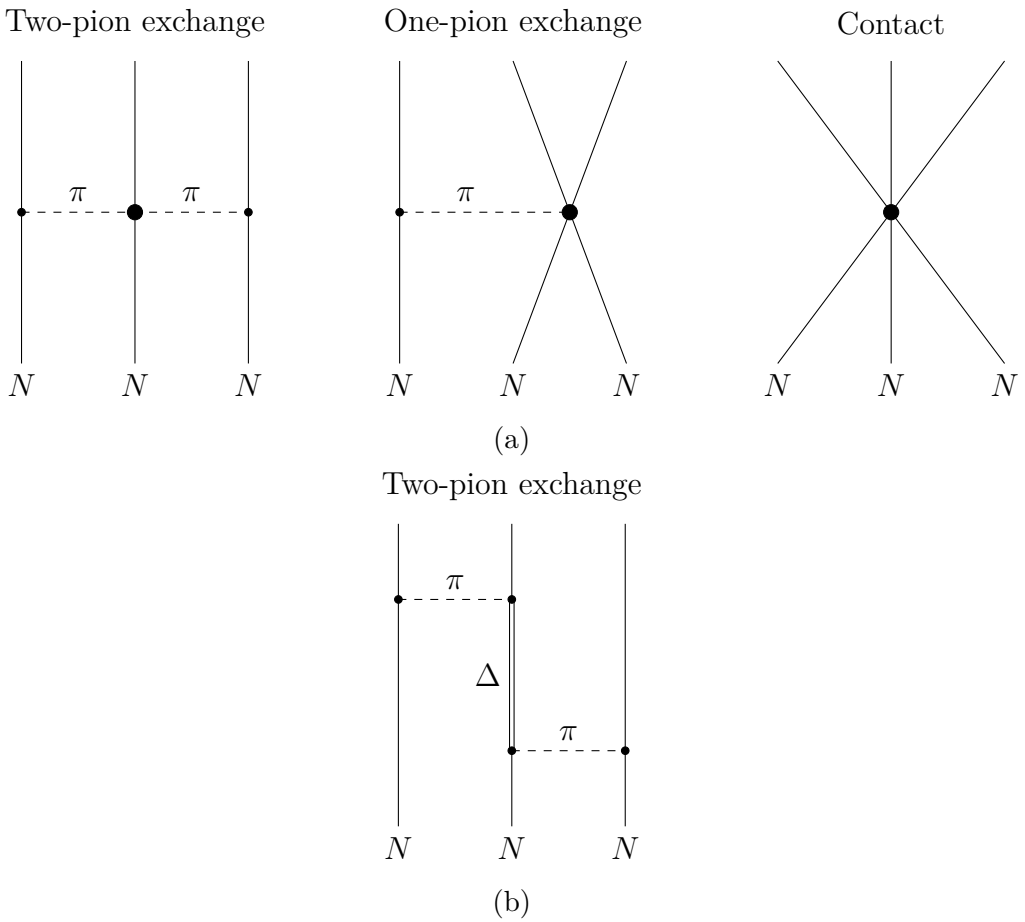

Figure 1.1: (a) The leading 3NF interaction-diagrams that appear at NNLO in the chiral expansion without a delta-degree of freedom. (b) In Delta-full χEFT a leading 3NF diagram appears already at next-to-leading order (NLO).

to find good approximation schemes. One such approximation is the single-reference normal-ordered two-body approximation (NO2B) [19]. In the single-reference NO2B approximation the 3NFs are expanded in a sum of zero-, one-, two- and three-body operators by normal ordering it relative a Slater-determinant reference state using Wick's theorem. The contribution of the normal-ordered three-body operator to the ground-state-energy-expectation value is assumed to be small compared to the contributions from the other operators. Therefore, it is discarded and we are left with an effective two-body force. The single-reference NO2B approximation is the main topic of chapter 3 and Paper I, where we study its performance with a HO-reference state and investigate the consequences of the broken translational symmetry of the NO2B Hamiltonian.

The LECs of χEFT must be determined by fitting them to experimental measurements of various nuclear observables, such as scattering cross sections, binding energies and nuclear radii. This fitting requires repeated solving of the MBSE for many different LEC values, which is extremely computationally demanding with the NCSM. However, by obtaining a few NCSM eigenvectors by solving the MBSE for

a few different LEC parametrizations, it is possible to use eigenvector continuation (EVC) [20–22], to construct emulators that then can be used to compute all necessary observables at much lower computational cost for a large set of LEC values. In chapter 5, EVC is described in more detail, and the method is then used in Paper II to compute six-nucleon observables from a Bayesian analysis of three-body forces resulting in a posterior probability distribution of LEC values [23]. In Paper III I trained EVC emulators for the $A = 6$ nuclei using delta-full χEFT, with 17 free parameters which were used to compare fitting protocols that only include few-body observables with an approach that also includes finite nuclei observables. In particular, we studied the resulting predictions for the saturation of infinite nuclear matter.

Chapter 2

The No-Core-Shell-Model

One of the most common methods to solve the many-body Schrödinger equation (MBSE) for atomic nuclei is the application of the shell model [24, 25]. In this model the nucleons are assumed to occupy the eigenstates of some central potential, similar to how electrons in the shell model for atoms occupy different orbitals due to a Coulomb-potential from the nucleus.

When implementing the shell model in a computer it is common to lock most of the nucleon states in a core, and only consider the dynamics of the valence nucleons. In this work we instead employ the No-Core-Shell-Model (NCSM) [16] where all nucleons are dynamical degrees of freedom. In the NCSM, the central potential that is used for the basis construction is the Harmonic Oscillator (HO). However, unlike the Coulomb-potential in the atom, this HO-potential is not part of the Hamiltonian but is a mathematical tool used to solve the MBSE.

This chapter describes the NCSM method to solve the MBSE in detail. In section 2.1 I introduce and discuss the NCSM-basis; how it relates to the HO-basis and how proton-neutron formalism can be used to separate the proton and neutron bases. Section 2.2 contains a description of how the NCSM-basis is used to construct a matrix eigenvalue problem from the MBSE that can be solved numerically on a computer. Lastly, I discuss the additional computational complexity of the inclusion of 3NFs in the Hamiltonian, compared to just including 2NFs.

2.1 The NCSM-basis

A distinguishing feature of the NCSM is the choice of the total-energy-truncated many-body HO-basis to describe the dynamics of all nucleons. In particular, the basis states, $|\Phi_i\rangle$, for a nucleus with A nucleons, are chosen to be antisymmeterized eigenstates of the HO Hamiltonian

$$\hat{H}_{\mathrm{HO}} = \sum_{i=1}^{A} \left(\frac{\mathbf{p}_i^2}{2m} + \frac{m\Omega^2}{2}\mathbf{r}_i^2 \right). \tag{2.1}$$

In this context, $\hbar\Omega$ is the oscillator frequency and $\mathbf{p}_i$ ($\mathbf{r}_i$) is the momentum (position) of nucleon i in three-dimensions [16]. The mass m is the nucleon mass and varies

between different implementations but is roughly the size of the proton and neutron mass. In addition to the usual HO quantum numbers, the nucleon states are also equipped with spin and iso-spin quantum numbers. Due to the symmetries of $\hat{H}_{\text{HO}}$ it is possible to either use relative (Jacobi)-coordinates or lab-coordinates. In this work I choose the latter since the many-body basis states are easier to antisymmeterize in large systems [26].

The nuclear interactions conserve the total angular momentum. Consequently, J is a good quantum number. To accommodate this symmetry, it is common to use a basis where all particle angular momenta are coupled to a total one, the so-called J-scheme basis. This allows the application of the Wigner-Eckart theorem [27] to reduce the Hamiltonian matrix in number of non-zero matrix elements. However, this is not done in this work, due to the computational complexity of the Slater algebra in this basis. Instead we apply the M-scheme formalism where only single-particle angular momenta are included. It is, however, still possible to reduce the basis based on rotational symmetry, by restricting the M-scheme states to a fixed total M, i.e.

$$\sum_{k=1}^{A} m_k = M, \tag{2.2}$$

where m_k is the azimuthal quantum number of particle k.

The antisymmetrization, due to the fermionic nature of the nucleons, is then easily imposed by requiring that the A-body states are Slater-determinants of the single-nucleon states in equation (2.4),

$$|\Phi_i\rangle = \frac{1}{\sqrt{A!}} \begin{vmatrix} |\alpha_{i,1}\rangle_1 & |\alpha_{i,1}\rangle_2 & \cdots & |\alpha_{i,1}\rangle_A \\ |\alpha_{i,2}\rangle_1 & |\alpha_{i,2}\rangle_2 & \cdots & |\alpha_{i,2}\rangle_A \\ \vdots & \vdots & \ddots & \vdots \\ |\alpha_{i,A}\rangle_1 & |\alpha_{i,A}\rangle_2 & \cdots & |\alpha_{i,A}\rangle_A \end{vmatrix} = \prod_{k=1}^{A} \hat{c}^\dagger_{\alpha_{i,k}} |\rangle \,, \tag{2.3}$$

where $\hat{c}^\dagger_\alpha$ is a second-quantization fermionic creation operator. Here the single-particle states take the form

$$|\alpha_{i,k}\rangle = |n, l, j, m, t_z\rangle \,, \tag{2.4}$$

where n is the HO radial quantum number, l is the orbital angular momentum, j (m) is the total (azimuthal) angular momentum, and t_z is the iso-spin. The outer subscript, k, on the single-particle ket, $|\alpha_{i,j}\rangle_k$, indicates that nucleon k is in state $\alpha_{i,j}$. The subscripts, i j indicates single-particle state j in the ordered list of single-particle states that belong to the many-body state $|\Phi_i\rangle$.

In practice, the many-body basis must be truncated. In the NCSM, the total number of HO-excitations above the A-fermionic HO-ground state is bounded from above, by the inequality

$$\sum_{k=1}^{A} (2n_{i,k} + l_{i,k}) - N_{\min} \leq N_{\max}, \tag{2.5}$$

where $N_{\min}$ is obtained by filling the Z lowest proton states and the $A - Z$ lowest neutron states. The NCSM-basis dimension, $D_{A,Z}(N_{\max})$, is therefore finite. The

truncation parameter $N_{\max}$, combined with the frequency $\hbar\Omega$ are the two main parameters of the NCSM basis.

2.1.1 Proton-Neutron formalism

In the previous discussion protons and neutrons were treated as different states of the same fermionic particle species, the nucleon, only distinguished through the iso-spin quantum number. This view is known as the iso-spin formalism. However, it is possible, and sometimes beneficial to treat these particles as two separate species each living in separate Hilbert spaces, $\mathcal{H}_p$ for protons and $\mathcal{H}_n$ for neutrons.

For each of the two Hilbert spaces we associate anti-symmetrized many-particle bases, $|\pi_i\rangle \in \mathcal{H}_p$ and $|\nu_i\rangle \in \mathcal{H}_n$. Just as $|\Phi_i\rangle$ are Slater determinants in this work, so are $|\pi_i\rangle$ and $|\nu_i\rangle$. Furthermore, we associate two sets of second-quantization operators with the two particle species, such that $|\pi_i\rangle = \prod_{k=1}^{Z} \hat{c}^{\pi\dagger}_{a_{i,k}} |\rangle_p$ and $|\nu_i\rangle = \prod_{k=1}^{N} \hat{c}^{\nu\dagger}_{a_{i,k}} |\rangle_n$. The subscripts $a_{i,k}$ are tuples of all quantum numbers in α except iso-spin. The full Hilbert space, in the proton-neutron formalism, is spaned by the product states

$$|\pi_i\rangle \otimes |\nu_i\rangle \,. \tag{2.6}$$

The relation between the iso-spin formalism and the NCSM states in equation (2.3) is

$$\left|\Phi_{n(i,j)}\right\rangle = \mathcal{A}\,|\pi_i\rangle \otimes |\nu_j\rangle \tag{2.7}$$

where $\mathcal{A}$ is the anti-symmetry operator, intermixing the proton and neutron states.

We still employ the total HO-energy truncation in equation (2.5). This translates to a criterion for the two new bases $N_p + N_n - N_{\min} \leq N_{\max}$, where N_p (N_n) is the HO-excitation of $|\pi_i\rangle$ ($|\nu_j\rangle$).

2.2 The Many-body Schrödinger equation

To solve the MBSE, we project it on the NCSM basis, described in the previous section. This projection results in a matrix eigenvalue problem,

$$\sum_{j=1}^{D(N_{\max})} \langle\Phi_i|\hat{H}|\Phi_j\rangle\, c_j = E_k c_i, \tag{2.8}$$

where the coefficients c_i are the amplitudes of the NCSM-state

$$\left|\Psi^{\mathrm{NCSM}}_k\right\rangle = \sum_{i=1}^{D(N_{\max})} c_i\,|\Phi_i\rangle\,. \tag{2.9}$$

The Hamiltonian in equation (2.8) considered in this work takes the form

$$\hat{H} = \hat{T}_{\mathrm{int}} + \hat{V}_{\mathrm{2NF}} + \hat{V}_{\mathrm{3NF}}, \tag{2.10}$$

where $\hat{T}_{\mathrm{int}}$ is the intrinsic kinetic energy, $\hat{V}_{\mathrm{2NF}}$ ($\hat{V}_{\mathrm{3NF}}$) is the two (three)-nucleon potential. The Hamiltonian is assumed to be translationally invariant. This symmetry

is preserved by the NCSM method because of the energy-conserving property of HO brackets [28] and the total HO energy truncation. Thus, the NCSM eigenstates separate in to an intrinsic part and a center of mass(CM)-part,

$$\left|\Psi_k^{\mathrm{NCSM}}\right\rangle = \left|\Psi_k^{\mathrm{NCSM}}\right\rangle_{\mathrm{int}} \otimes \left|\Psi_k^{\mathrm{NCSM}}\right\rangle_{\mathrm{CM}}, \tag{2.11}$$

where $\left|\Psi_k^{\mathrm{NCSM}}\right\rangle_{\mathrm{CM}}$ is a HO state in the CM coordinate, see reference [16] for further details.

Because of equation (2.11) spectra of NCSM states will contain both intrinsic and CM excitations. Since only intrinsic excitations are of interest it is desirable to add a Lawson term $\hat{H}_\beta$ to the Hamiltonian [29]. The Lawson term consists of a shifted and scaled CM HO Hamiltonian times a multiplicative factor, i.e.

$$\hat{H}_\beta = \beta \left(\frac{\left(\sum_{k=1}^{A} \mathbf{p}_k\right)^2}{2Am} + \frac{m\Omega^2}{2A} \left(\sum_{k=1}^{A} \mathbf{r}_k\right)^2 - \frac{3}{2}\hbar\Omega \right). \tag{2.12}$$

By choosing sufficiently large β only NCSM states where the CM part is a ground state of $\hat{H}_\beta$ (with energy $E_\beta = 0$) will appear among the low energy states.

The NCSM is a variational method, meaning that the lowest eigenvalue of equation (2.8), $E_{\mathrm{gs}}^{\mathrm{NCSM}}$, is larger than or equal to the true ground state energy, E_{gs} of $\hat{H}$. The two main parameters of the NCSM basis, the oscillator frequency $\hbar\Omega$ and the truncation parameter $N_{\max}$ can be associated with infrared (IR)- and ultraviolet (UV)-momentum cutoffs that yield corresponding length scales, i.e., the longest respective shortest distances the basis can resolve [30–33]. The convergence of $E_{\mathrm{gs}}^{\mathrm{NCSM}}$ towards E_{gs} has been shown [34] to be exponential in the IR-length scale. A study of IR convergence is not part of this work. Nevertheless, the variational nature and tendency at exponential convergence of the NCSM method is illustrated in figure 2.1, where the ground state energy of ^{4}He has been computed with the 2NF part of the N2LO$_{\mathrm{sat}}$ interaction [35].

There are several computer implementations of the NCSM at use by the community. Table 2.1 contains a list of some of the more common implementations that can handle $A > 4$ nuclei. The major part of this work has been focused on developing a new large-scale NCSM code, named JupiterNCSM, that can handle the inclusion of 3NFs in the Hamiltonian. This code is discussed in length in chapter 4. We have used (and benchmarked) NCSD, pAntoine and JupiterNCSM in the research that is part of this thesis. We also use the nsopt code [36], which do NCSM in a relative Jacobi-basis to efficiently use 2NFs and 3NFs. However, it is limited to $A \leq 4$ nuclei.

The eigenvalue problem for the Hamiltonian matrix in equation (2.8) grows quickly in size with increasing $N_{\max}$. However, it is a sparse matrix. Since $\hat{T}_{\mathrm{int}}$ and $\hat{V}_{\mathrm{2NF}}$ are two-nucleon operators they will only have non-zero contribution to $\langle\Phi_i|\,\hat{H}\,|\Phi_j\rangle$ if the states i and j differ with at most two-particle excitations, while $\hat{V}_{\mathrm{3NF}}$ connects also states that differ with three-particle excitations. Even if the matrix is sparse, the number of non-zero matrix elements can be huge; especially with the inclusion of 3NFs. Figure 2.2 illustrates this last point where the total number of matrix elements is being compared to the number of non-zero matrix elements

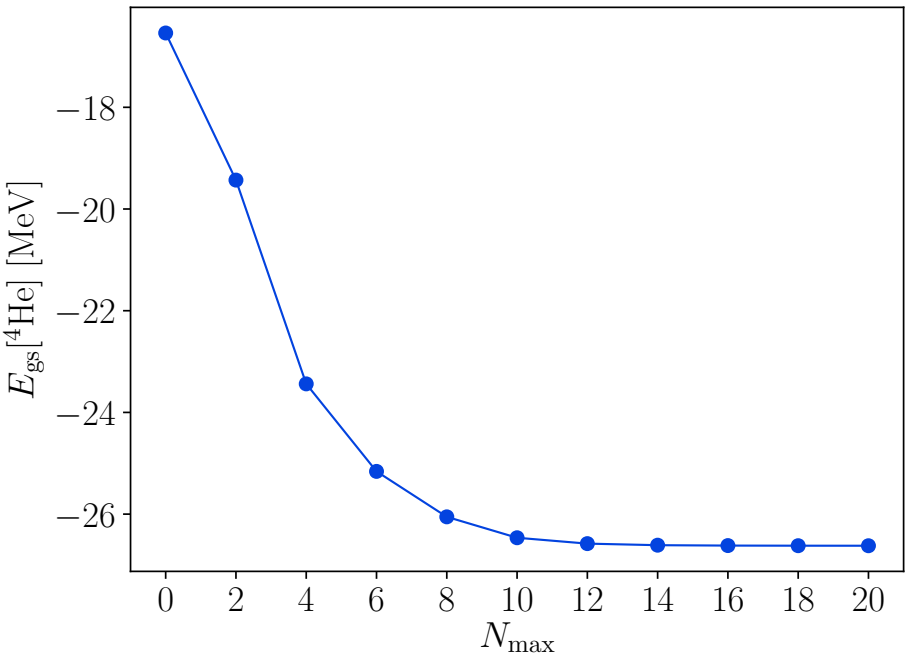

Figure 2.1: The binding energy of ^{4}He computed with only the 2NF part of the N2LO$_{\mathrm{sat}}$ interaction[35] for the basis frequency $\hbar\Omega = 20$.

Table 2.1: List of common NCSM computer programs with capability to handle $A > 4$ nuclei.

Name	Comment	Open source	References
(p)Antoine	Factorized on-the-fly algorithm for computing many-body matrix elements utilizing smaller proton/neutron subspace dimensions. No 3NFs.	no	Forssén et al. [34], Caurier and Nowacki [37], and Navrátil and Caurier [38]
BIGSTICK	Factorized on-the-fly algorithm for computing many-body matrix elements. Both MPI (distributed memory) and OpenMP (shared memory) parallelization.	yes	Johnson et al. [39]
LSU3shell	Symmetry-adapted NCSM code using U(3) and Sp(3,R) many-nucleon basis.	yes	Dytrych et al. [40]
MFDn	Massively parallel code with explicit storage of the Hamiltonian matrix over distributed memory.	no	Sternberg et al. [41]
NCSD	Explicit storage of the Hamiltonian matrix over distributed memory with J-decoupling on the fly.	no	Navratil [42]

for a Hamiltonian with both 2NF and 3NF, and one with only 2NFs for the ^{6}Li nucleus.

It is seen that the number of non-zero matrix elements with inclusion of 3NFs

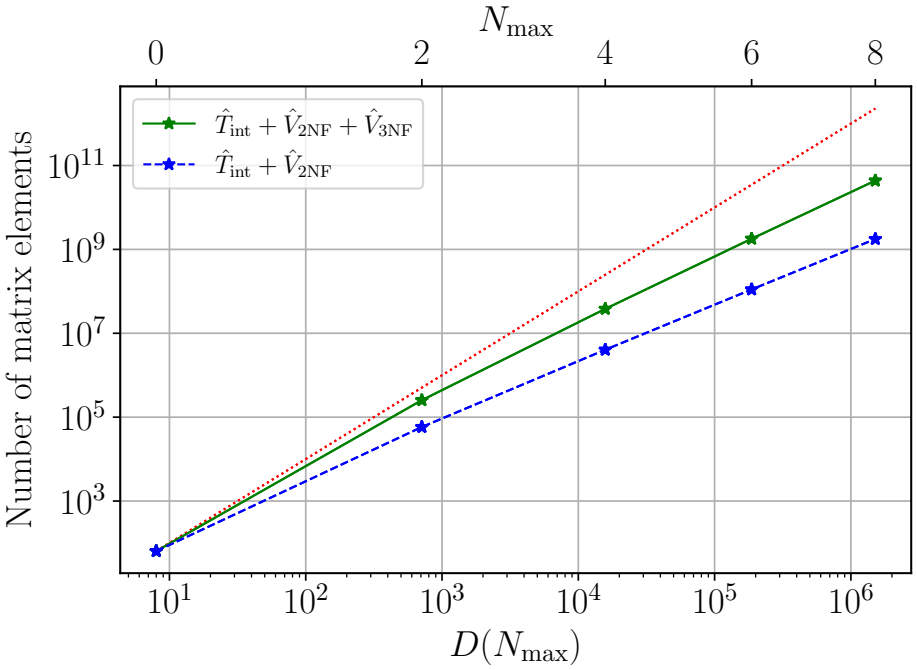

Figure 2.2: The number of non-zero Hamiltonian matrix elements for ^{6}Li, with (green solid line) and without (blue dashed line) 3NFs. These are compared to the total number of elements in the Hamiltonian, $D(N_{\max})^2$ (red dotted line).

increases much faster with $N_{\max}$ than for the 2NF-only case. In fact, for $N_{\max} = 8$ there about 25 times more 3NF matrix elements than 2NF ones. Since the diagonalization time is proportional to the number of non-zero matrix elements the computational cost will be roughly 25 times larger for ^{6}Li at $N_{\max} = 8$ with 3NFs compared to just 2NFs. This difference will increase with particle number A and will effectively put a limit on which A and $N_{\max}$ that can be handled with full inclusion of 3NFs.

Chapter 3

Normal-ordered three-nucleon forces

Approximation schemes could be useful to resolve the problem with the additional computational complexity of the inclusion of 3NFs in the NCSM. In this chapter I explore one such method, the single-reference normal-ordered two-body (NO2B) approximation. This approximation is the topic of Paper I, and was extensively discussed in my Licentiate thesis [43].

In this chapter I will define the NO2B approximation as an application of Wick's theorem [44]. This transforms the 3NF into a sum of residual zero-, one-, two- and three-body operators normal-ordered relative a reference state. If the reference state is a good approximation to the ground state of the system, then the residual thee-body term can be assumed to have vanishing contribution to the ground-state energy and discarded in the calculation, leaving us with an effective 2NF. A more detailed explanation of the approximation is outlined in section 3.1.

The normal-ordering relative a reference state introduces an explicit CM-dependence in the resulting force and thus breaks the translational invariance of the full Hamiltonian. This can induce spurious CM-excitations in the computed NCSM-ground state which affects observable predictions. I discuss this problem in more detail in section 3.2.

In section 3.3 I present the main result of Paper I. Here the NO2B-approximated ground-state energies of the two smallest closed core nuclei ^{4}He and ^{16}O using N2LO$_{\text{sat}}$ interaction [35], as function of the NCSM-basis frequency $\hbar\Omega$, is compared to ground-state energies computed with exact 3NFs. Furthermore, a study of the broken translational symmetry in the NO2B-approximation and its consequence of spurious CM-excitations in the ground states are also presented.

3.1 The NO2B-approximation

The key step of the NO2B-approximation is the application of Wick's theorem on the 3NF part of the Hamiltonian. This theorem states that a product of second quantization operators $\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}\hat{f}$ can be written as a sum of normal-ordered products

of the operators and all possible Wick-contractions, as

$$\begin{aligned}\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}\hat{f} =& \{\hat{a}\hat{b}\hat{c}\hat{d}\hat{e}\hat{f}\} \\ &+ \{\hat{\hat{a}}\hat{\hat{b}}\hat{c}\hat{d}\hat{e}\hat{f}\} + \{\hat{\hat{a}}\hat{b}\hat{\hat{c}}\hat{d}\hat{e}\hat{f}\} + \cdots + \{\hat{a}\hat{b}\hat{c}\hat{d}\hat{\hat{e}}\hat{\hat{f}}\} \\ &+ \{\hat{\hat{a}}\hat{\hat{b}}\hat{c}\hat{d}\hat{e}\hat{f}\} + \cdots + \{\hat{a}\hat{b}\hat{c}\hat{\hat{d}}\hat{\hat{e}}\hat{\hat{f}}\} \\ &+ \{\hat{\hat{a}}\hat{\hat{b}}\hat{c}\hat{\hat{d}}\hat{\hat{e}}\hat{f}\} + \cdots + \{\hat{a}\hat{b}\hat{c}\hat{\hat{d}}\hat{\hat{e}}\hat{\hat{f}}\},\end{aligned} \tag{3.1}$$

where the Wick contractions $\hat{\hat{a}}\hat{\hat{b}}$, defined as a complex number such that $\hat{\hat{a}}\hat{\hat{b}} = \hat{a}\hat{b} - \{\hat{a}\hat{b}\}$, in general are complex numbers. In this context the normal-ordering of a second-quantization operator $\hat{O}$ is done relative a reference state $|\Psi_{\text{ref}}\rangle$ such that the expectation value $\langle\Psi_{\text{ref}}|\{\hat{O}\}|\Psi_{\text{ref}}\rangle = 0$.

In the single-reference NO2B-approximation, the reference state $|\Psi_{\text{ref}}\rangle$ is chosen to be a single Slater-determinant in the basis of choice. In our case this means it is expressed with the same HO-single-particle basis as the NCSM-basis described in chapter 2, i.e., $|\Psi_{\text{ref}}\rangle = \hat{c}^\dagger_{\alpha_1} \cdots \hat{c}^\dagger_{\alpha_A} |0\rangle$. The contractions of pairs of creation and annihilation operators can then be calculated as

$$\hat{\hat{c}}^\dagger_\alpha \hat{\hat{c}}_\beta = \begin{cases} \delta_{\alpha,\beta} & \text{if } \alpha, \beta \in R \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

$$\hat{\hat{c}}_\alpha \hat{\hat{c}}^\dagger_\beta = \begin{cases} \delta_{\alpha,\beta} & \text{if } \alpha, \beta \notin R \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

where $R = \{\alpha_1, \alpha_2, \ldots, \alpha_A\}$ is the reference set. These contractions can be understood through the particle-hole formalism, where the contraction in equation (3.2) involves hole-creation and annihilation operators while the contractions in equation (3.3) involves particle-creation and annihilation operators.

By applying Wick's theorem on a general 3NF potential

$$\hat{V}_{\text{3NF}} = \frac{1}{36} \sum_{\substack{\alpha,\beta,\gamma \\ \alpha',\beta',\gamma'}} \langle \alpha, \beta, \gamma | V_{\text{3NF}} | \alpha', \beta', \gamma' \rangle \, \hat{c}^\dagger_\alpha \hat{c}^\dagger_\beta \hat{c}^\dagger_\gamma \hat{c}_{\gamma'} \hat{c}_{\beta'} \hat{c}_{\alpha'} \tag{3.4}$$

I get the expansion

$$\begin{aligned}\hat{V}_{\mathrm{3NF}} =& \overbrace{\frac{1}{36}\sum_{\substack{\alpha,\beta,\gamma\\ \alpha',\beta',\gamma'}} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha',\beta',\gamma'\rangle \left\{\hat{c}^\dagger_\alpha \hat{c}^\dagger_\beta \hat{c}^\dagger_\gamma \hat{c}_{\gamma'} \hat{c}_{\beta'} \hat{c}_{\alpha'}\right\}}^{\hat{W}^{3b}_{\mathrm{3NF}}} \\ &+ \overbrace{\frac{1}{4}\sum_{\substack{\alpha,\beta\\ \alpha',\beta'}}\sum_{\gamma\in R} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha',\beta',\gamma\rangle \left\{\hat{c}^\dagger_\alpha \hat{c}^\dagger_\beta \hat{c}_{\beta'} \hat{c}_{\alpha'}\right\}}^{\hat{W}^{2b}_{\mathrm{3NF}}} \\ &+ \overbrace{\frac{1}{2}\sum_{\substack{\alpha\\ \alpha'}}\sum_{\beta,\gamma\in R} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha',\beta,\gamma\rangle \left\{\hat{c}^\dagger_\alpha \hat{c}_{\alpha'}\right\}}^{\hat{W}^{1b}_{\mathrm{3NF}}} \\ &+ \overbrace{\frac{1}{6}\sum_{\alpha,\beta,\gamma\in R} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha,\beta,\gamma\rangle}^{W^{0b}_{\mathrm{3NF}}}.\end{aligned} \tag{3.5}$$

The $\hat{W}^{3b}_{\mathrm{3NF}}$ term will only give a small contribution to the ground-state energy if the reference state $|\Psi_{\mathrm{ref}}\rangle$ is a good approximation of the ground-state wave-function. The single-reference NO2B approximated 3NF can therefore be written as

$$\hat{V}^{\mathrm{NO2B}}_{\mathrm{3NF}} \equiv W^{0b}_{\mathrm{3NF}} + \hat{W}^{1b}_{\mathrm{3NF}} + \hat{W}^{2b}_{\mathrm{3NF}}. \tag{3.6}$$

However, to use this approximation in the NCSM method we need to transform it back to a vacuum-normal ordered form. This is done by applying Wick's theorem backwards, which yields

$$\begin{aligned}\hat{V}^{\mathrm{NO2B}}_{\mathrm{3NF}} =& \frac{1}{4}\sum_{\substack{\alpha,\beta\\ \alpha',\beta'}}\sum_{\gamma\in R} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha',\beta',\gamma\rangle\, \hat{c}^\dagger_\alpha \hat{c}^\dagger_\beta \hat{c}_{\beta'} \hat{c}_{\alpha'} \\ &- \frac{1}{2}\sum_{\alpha,\alpha'}\sum_{\beta,\gamma\in R} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha',\beta,\gamma\rangle\, \hat{c}^\dagger_\alpha \hat{c}_{\alpha'} \\ &+ \frac{1}{6}\sum_{\alpha,\beta,\gamma\in R} \langle\alpha,\beta,\gamma|V_{\mathrm{3NF}}|\alpha,\beta,\gamma\rangle\,,\end{aligned} \tag{3.7}$$

an expansion of a two-, one- and zero-body operator [43].

3.2 The Center of Mass Problem

The breaking of the translational symmetry of the Hamiltonian due to the NO2B-approximation is a fundamental problem. In general, the energy spectrum of a nucleus does not depend on its location, unless it is affected by an external potential. Since our reference state is fixed to a specific reference frame, a dependence on the CM coordinate is introduced in the approximated force $\hat{V}^{\mathrm{NO2B}}_{\mathrm{3NF}}$. This breaking of

the translational symmetry of the Hamiltonian can introduce CM excitations that modify the computed energy spectrum.

We employ two different measures, ξ_{CM} and N_{CM}, to measure the presence of CM excitations in the computed ground state, $|\Psi_{\text{gs}}\rangle$. The first measure, ξ_{CM} was introduced by Parzuchowski et al. [45] to analyze CM-excitations in the IM-SRG method [46]. The measure is based on the assumption that the CM-part of the ground-state is a ground-state of a HO with frequency ω * in the CM-coordinate. If this assumption is true then the CM-momentum and CM-position ground-state expectation values with respect to $|\Psi_{\text{gs}}\rangle$ are

$$\langle P^2_{\text{CM}}\rangle = \frac{3\hbar^2}{2b^2} \tag{3.8}$$

$$\langle R^2_{\text{CM}}\rangle = \frac{3b^2}{2}, \tag{3.9}$$

where b is the oscillator length $b = \sqrt{\frac{\hbar}{m\omega}}$, such that the quantity

$$\xi_{\text{CM}} = \frac{\sqrt{\langle P^2_{\text{CM}}\rangle\langle R^2_{\text{CM}}\rangle}}{\hbar} - \frac{3}{2} \tag{3.10}$$

must be equal to zero. However, ξ_{CM} is strictly larger than zero if the CM-state is not a HO-ground state and is therefore a suitable measure for spurious CM-excitations. On the other hand, if the assumption holds true, that the CM-part of the ground state is a HO-ground state, then it is possible to compute the corresponding frequency for that HO-ground-state, $\hbar\omega_\xi = \frac{4}{3}\langle T_{\text{CM}}\rangle$ where the expectation value is with respect to the ground state of the nucleus.

The second measure, N_{CM} was first developed by Hagen, Papenbrock, and Dean [47] for use in the Coupled-Cluster method [48, 49] where CM-excitations can be introduced by the choice of the Hartree-Fock basis. This measure is also based on the assumption that the CM-part of the state $|\Psi_{\text{gs}}\rangle$ is a HO-ground state with a frequency $\hbar\omega_N$. This frequency can be computed from the ground-state expectation value $\langle\hat{H}_{\text{CM}}(\Omega)\rangle \equiv \langle\Psi_{\text{gs}}|\hat{H}_{\text{CM}}(\Omega)|\Psi_{\text{gs}}\rangle$ † where

$$\hat{H}_{\text{CM}}(\bar{\omega}) \equiv \hat{T}_{\text{CM}} + \frac{Am\bar{\omega}^2}{2}\hat{R}^2_{\text{CM}} - \frac{3\hbar\bar{\omega}}{2} \tag{3.11}$$

is a shifted HO-Hamiltonian in the CM-coordinate. From this expectation value a quadratic equation can be constructed which allows to find ω_N with the two solutions

$$\hbar\omega^{\pm} = \hbar\Omega + \frac{2}{3}\langle\hat{H}_{\text{CM}}(\Omega)\rangle \pm \sqrt{\frac{4}{9}\langle\hat{H}_{\text{CM}}(\Omega)\rangle^2 + \frac{4}{3}\hbar\Omega\langle\hat{H}_{\text{CM}}(\Omega)\rangle}. \tag{3.12}$$

The frequency ω_N is determined by

$$\omega_N = \underset{\omega\in\{\omega^{\pm}\}}{\arg\min}\langle\hat{H}_{\text{CM}}(\omega)\rangle. \tag{3.13}$$

*This frequency is not necessarily equal to that of the NCSM basis Ω used in chapter 2.

†The frequency used in this expression is the NCSM basis frequency.

We define the measure $N_{\text{CM}} \equiv \frac{\langle \hat{H}_{\text{CM}}(\omega_N) \rangle}{\hbar\omega_N}$, which indicate how well the initial assumption holds. This measure is zero when $|\Psi_{\text{gs}}\rangle$ is a product state between an intrinsic state and a HO-ground state in the CM-coordinate. In all other cases $N_{\text{CM}} > 0$.

3.3 Results

In Paper I we tested the NO2B approximation for ^{4}He, the smallest closed-core nucleus, by computing its ground-state energy with the N2LO$_{\text{sat}}$ [35] interaction using both a NO2B approximated 3NF and an exact 3NF. The NCSM-code pAntoine, see table 2.1, where utilized to obtain the NO2B ground-energies while the nsopt, discussed in section 2.2, where used for the calculations with exact inclusion of the 3NF. The reference state is chosen to be the HO-ground-state for the system, where all four nucleons fill the lowest $0s_{\frac{1}{2}}$ shell. We then computed the ground-state energy for different NCSM-bases, varied in N_{max} and $\hbar\Omega$.

In the upper panel in figure 3.1 the ground-state energy per nucleon for ^{4}He with a NO2B-approximated and exact 3NF has been plotted. These energies are plotted as functions of increasing NCSM-model space and are computed for nine different basis-frequencies.

To better see the difference between the approximated 3NF and the exact one the lower panel shows the absolute difference between ground-state energy per nucleon for the approximated and exact 3NF. For oscillator frequencies below 20 MeV the NO2B-approximation only has a small discrepancy compared to the exact ground states with a minimal difference for $\hbar\Omega = 20$ MeV. However, for larger frequencies the approximation over-binds the nucleus.

A clue to why the NO2B-approximation works well for $\hbar\Omega \leq 20$ MeV comes from the discussion about CM-excitations. Therefore, we have plotted the ξ_{CM} and the frequency $\hbar\omega_\xi$ in figure 3.2, as function of the basis frequency $\hbar\Omega$ for different N_{max}. As can be seen ξ_{CM} is close to zero for frequencies below 20 MeV while diverging for larger frequencies. An interesting feature is that the frequency of the CM-ground state $\hbar\omega_\xi$ starts to diverge from the basis frequency already for $\hbar\Omega = 16$ MeV, indicating that some CM excitations starts to emerge already here. In Paper I we also present the N_{CM} measure. The N_{CM} analysis is omitted in this work, since it completely agrees with ξ_{CM}, but can be read in detail in Paper I.

In ^{4}He, the additional computational cost of full inclusion of 3NFs is not severe enough to motivate the use of an approximation. Combining this with the large frequency dependence of the approximation error illustrated above, it is motivated to test the NO2B-approximation in a heavier system. In a heavier system the CM require more energy to be excited above the ground-state and thus CM-mixing problem can be less pronounced.

To test this hypothesis, we computed the ground-state energy for ^{16}O in the same way as for ^{4}He. However, the NCSD* code, see table 2.1, used to compute

*The reason that we used NCSD and not JupiterNCSM in this project that the later was not developed at the time.

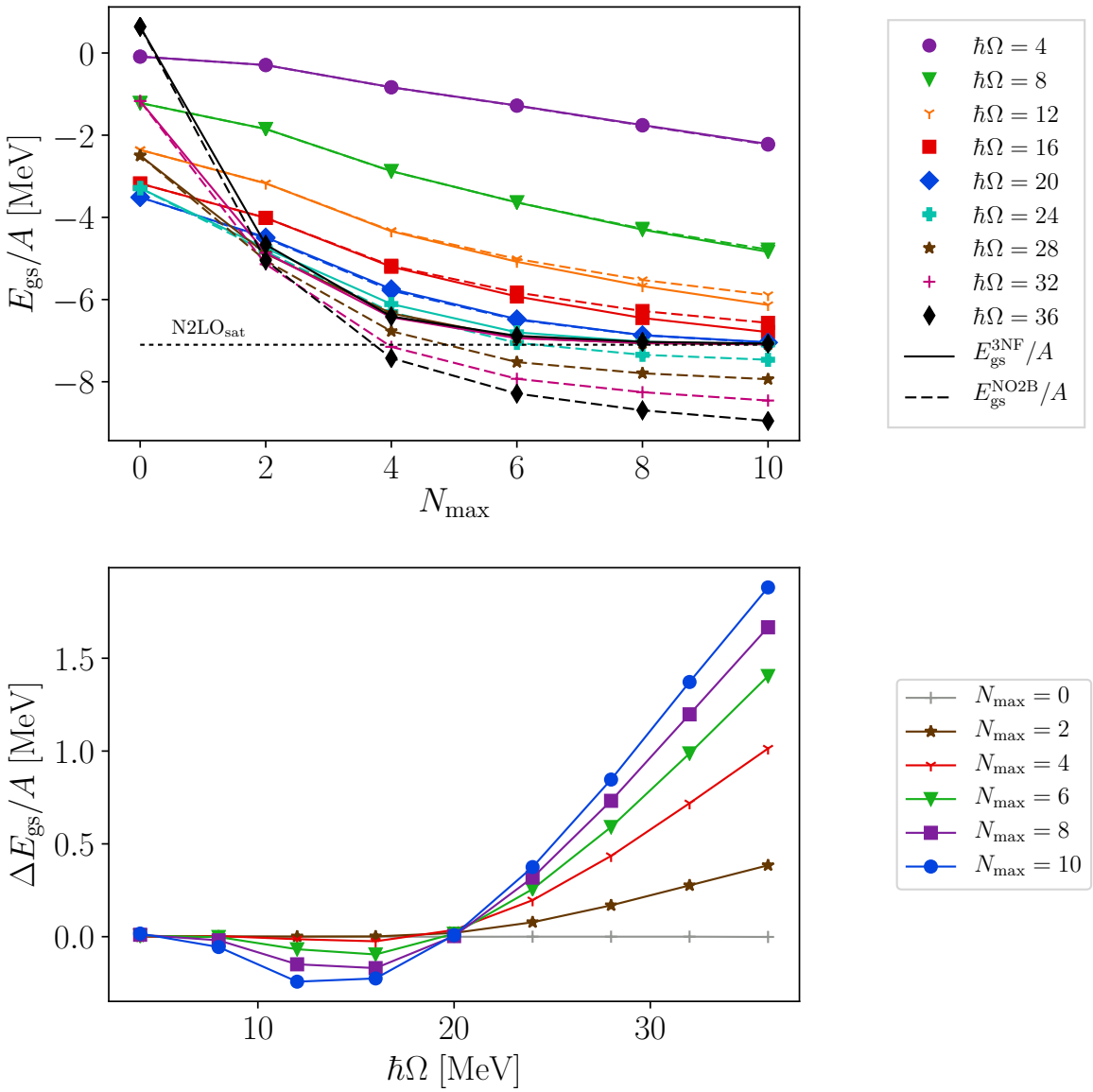

Figure 3.1: Upper panel: The ground-state energy per nucleon of ^{4}He computed with a NO2B-approximated N2LO-sat 3NF, the dashed lines, compared to the same ground-states computed with exact 3NFs. The ground-states are computed for nine different values of the NCSM-basis frequency $\hbar\Omega$, and as function of the truncation parameter N_{max}. Lower panel: The absolute difference in ^{4}He ground-state energy per nucleon between the NO2B-approximated 3NF and the exact one, as function of the basis frequency $\hbar\Omega$.

ground-states with an exact 3NF, was limited $N_{\mathrm{max}} \leq 6$ for ^{16}O. The results of these calculations can be viewed in figure 3.3. Because of the extreme computational cost of the 3NFs calculations, only ground-state energies for the frequencies $\{16, 20, 24, 36\}$ MeV where included, i.e., three frequencies close to the variational minimum and one further away.

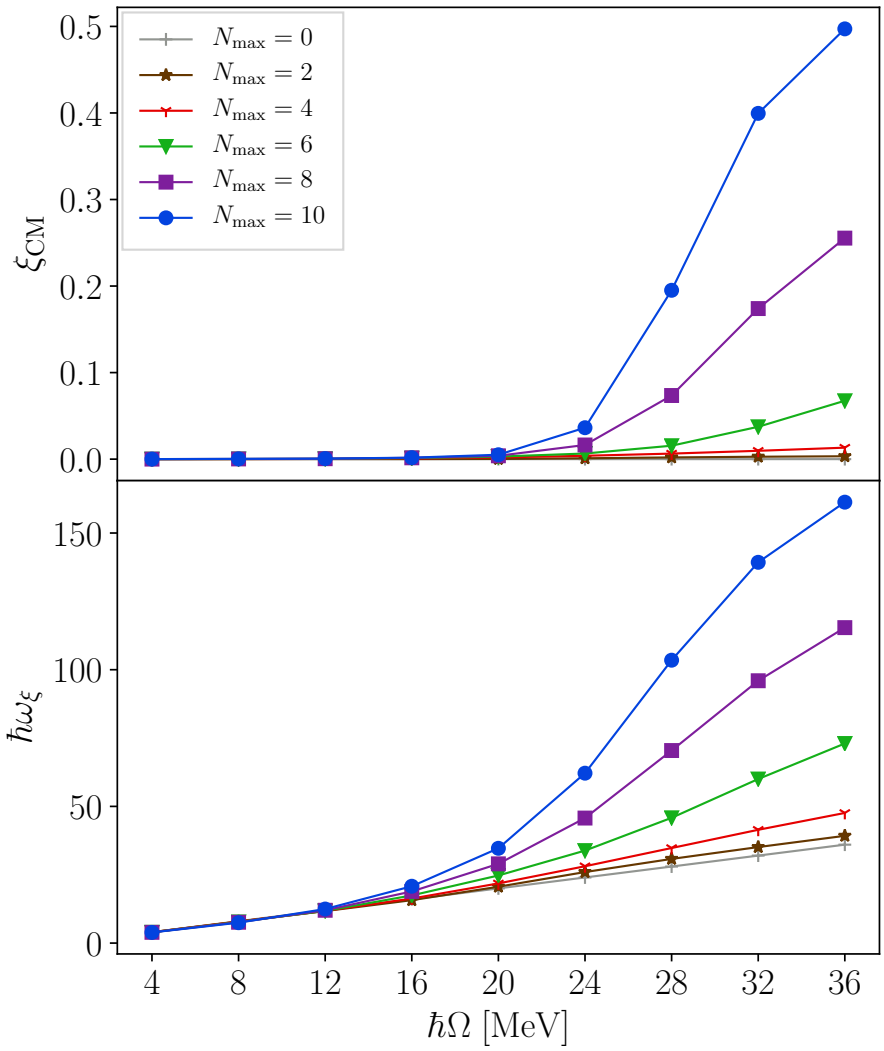

Figure 3.2: Upper panel: The CM-excitation measure ξ_{CM} computed for ^{4}He ground-states with a NO2B-approximated N2LO-sat 3NF as function of the basis frequency $\hbar\Omega$, for different NCSM-truncation. Lower panel: The frequency $\hbar\omega_\xi$, computed for the same ^{4}He ground-state, as function of the NCSM-basis frequency.

The differences between the approximated ground-states and the exact ones are less than 0.1 MeV for all four frequencies, i.e, about 1%. Furthermore, the difference does not show the same strong $\hbar\Omega$-dependence as for ^{4}He. In particular, it does not increase significantly for large frequencies.

We have also computed the ξ_{CM} as function of $\hbar\Omega$ for ^{16}O, which can be viewed in figure 3.4. Notice that the scale on the vertical axis in the ξ_{CM} plot (upper panel) is about a factor of 100 smaller than in figure 3.2.

While there are some CM-excitations in the ^{16}O ground-state, as evident from figure 3.4, it is two orders of magnitude smaller than that of ^{4}He. This seems to confirm the hypothesis that the CM-problem is less of an issue for larger nuclei.

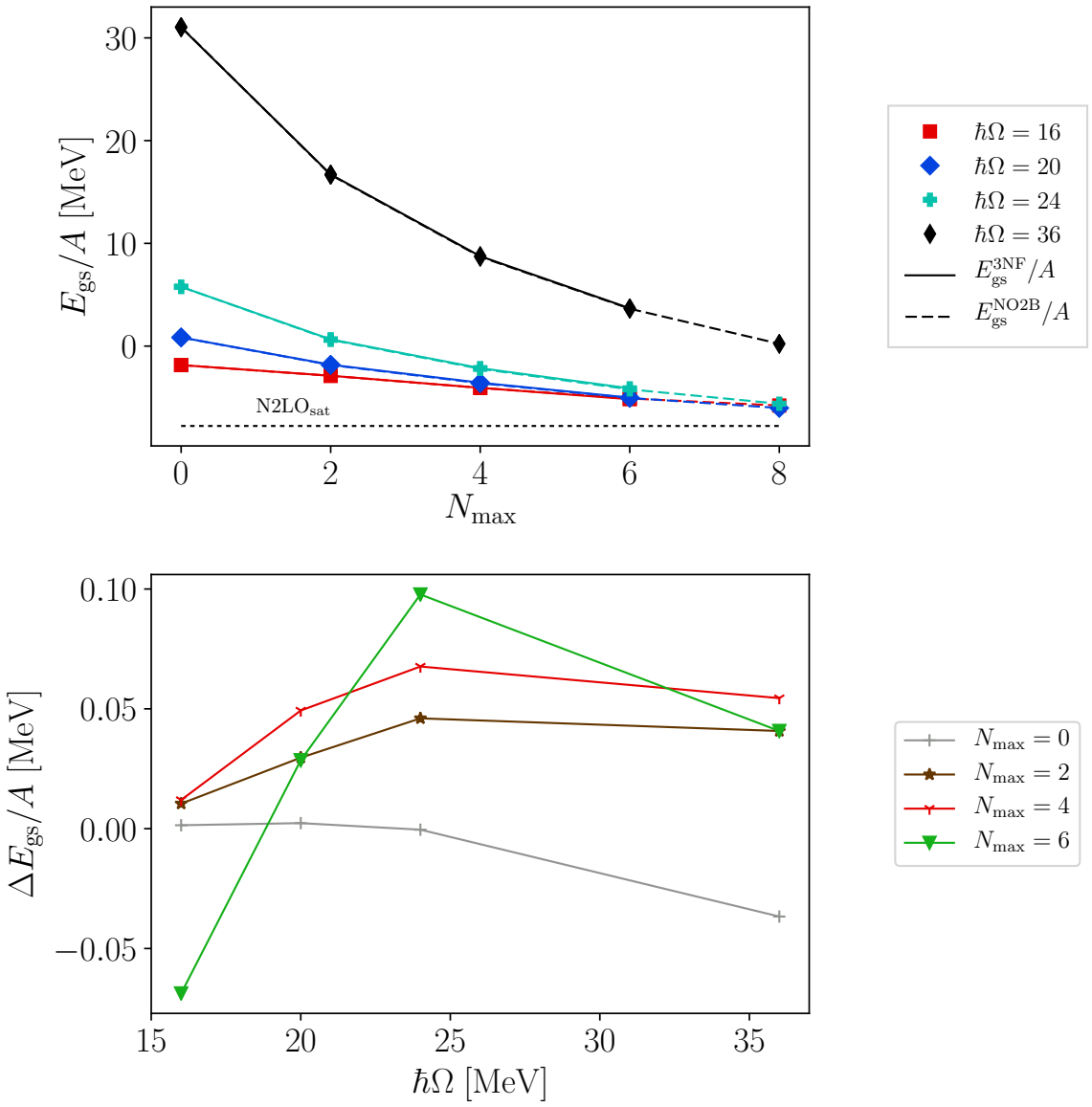

Figure 3.3: Upper panel: The ground-state energy per nucleon of ^{16}O computed with a NO2B-approximated N2LO-sat 3NF, the dashed lines, compared to the same ground-state energy computed with an exact 3NF, solid lines. Due to computational limitations the exact ground-state energies are only computed up to $N_{\mathrm{max}} = 6$ and only four frequencies are computed. Lower panel: The difference between the approximated and exact ground-state energy per nucleon of ^{16}O as function of basis frequency.

There is however a concern regarding the fact that the ^{4}He spectrum is close to be converged while the ^{16}O spectrum is far from converged. This could mean that the CM-problem becomes more pronounced for ^{16}O for higher values of N_{max} than we are able to explore in this work. We elaborate more about this in Paper I.

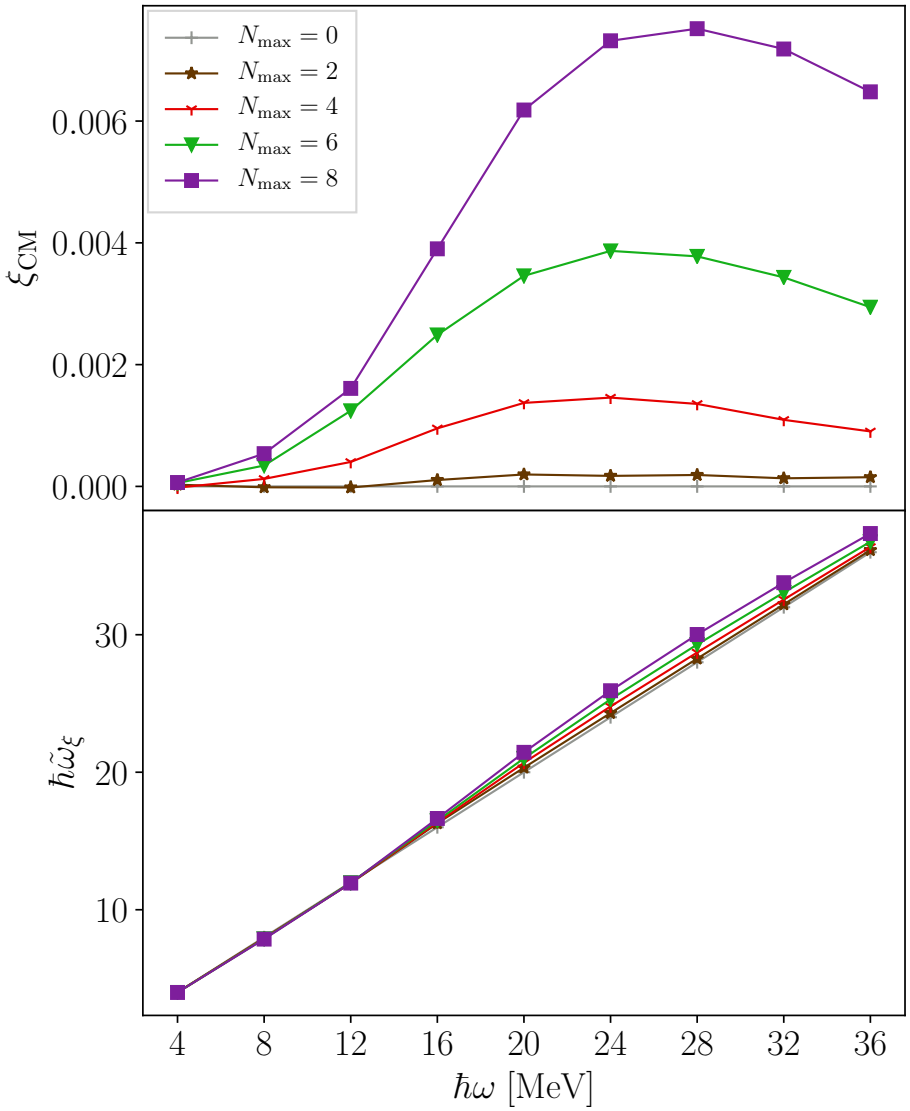

Figure 3.4: Upper panel: The CM-excitation measure ξ_{CM} computed for ^{16}O with a NO2B-approximated N2LO-sat 3NF as function of the basis frequency $\hbar\Omega$, for different NCSM-truncation. Lower panel: The frequency $\hbar\omega_\xi$, computed for the same ^{16}O ground-state, as function of the NCSM-basis frequency.

Chapter 4

The JupiterNCSM code

The main goal of this thesis is to explore different techniques of including 3NFs from realistic interactions in nuclear structure calculations with the NCSM. In this chapter I present Code I, JupiterNCSM, and Code II, Anicre, which together is a computer implementation of the NCSM that can directly handle 3NFs. In appendix A, I present a minimal working example where JupiterNCSM is used to compute the eigenspectrum of ^{4}He for a realistic interaction that includes 3NFs.

This chapter starts with a detailed description of the different modules of JupiterNCSM, with special focus on Bacchus and Minerva. These two modules handle the Lanczos algorithm and the matrix-vector multiplication with matrix element evaluation on the fly, respectively. In section 4.2 I present benchmarks of the code, where results from JupiterNCSM are compared to results from other NCSM codes. In addition, time and memory consumption of the code is presented, and convergence of eigenvectors and eigenvalues in Lanczos diagonalization is studied.

4.1 Program structure

In this section the main structure of JupiterNCSM is described. An overview of JupiterNCSM is shown in figure 4.1. The MBSE matrix eigenvalue problem is solved using the Lanzos algorithm implemented as the code Bacchus, described in section 4.1.1. Each Lanczos iteration needs a matrix-vector multiplication to be performed, which is delegated to the code Minerva, discussed in section 4.1.2. To compute the matrix-vector multiplication efficiently Minerva needs index lists representing transition densities between the basis states. These lists are computed by the code Anicre, explained in section 4.1.3. Furthermore, Minerva also needs the Hamiltonian matrix elements expressed in an M-scheme basis. Since the interactions are stored in a J-scheme format, the codes Neptune, Mercury and Vulcan, described in sections 4.1.4, can be used to transform them from J-scheme to an internal M-scheme format that Minerva can use efficiently. Lastly the code Mars, described in section 4.1.5, instructs Minerva in what order the matrix-vector multiplication is to be performed, with focus on minimizing execution time.*

*Each code, except Anicre, is named after the Roman gods since only a god could truly understand the atomic nucleus.

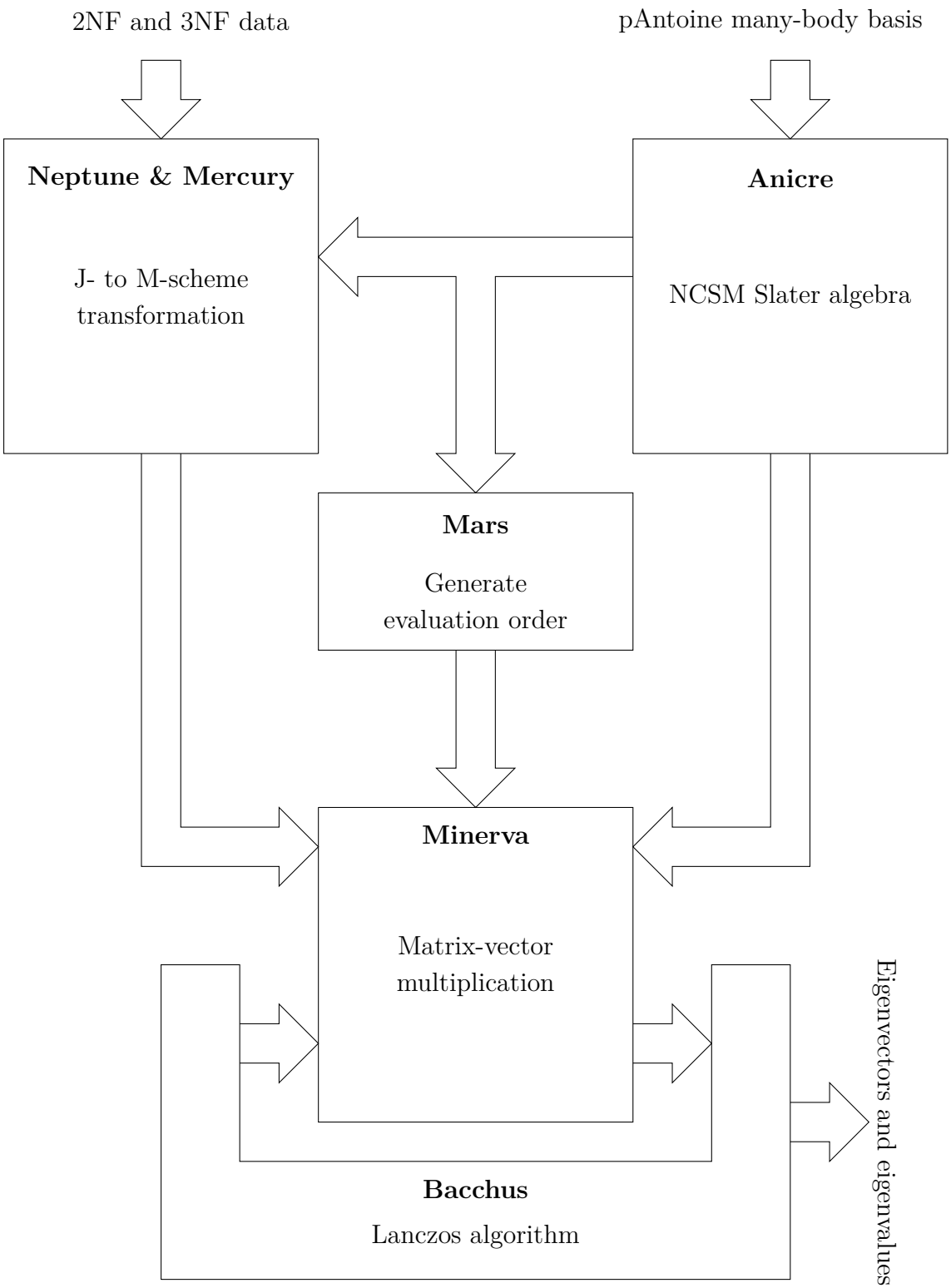

Figure 4.1: The structure of the JupiterNCSM code.

4.1.1 Lanczos diagonalization: Bacchus

Bacchus solves the MBSE (2.8), by applying the Lanczos algorithm [50]. In this section I will describe the Lanczos algorithm, how it is implemented in Bacchus and the user interface of the module.

The Lanczos algorithm transforms a symmetric $D \times D$ matrix $\hat{H}$, in our case the Hamiltonian matrix, to a sequence of symmetric tridiagonal $k \times k$ matrices T_k The eigenvalues of T_k approach the eigenvalues of $\hat{H}$ as k approaches D. Let A_i for $i = 1, \ldots, k$ be the diagonal elements of T_k and B_j for $j = 1, \ldots, k-1$ its off-diagonal elements. Further assume that the $D \times k$ matrix U_k fulfills

$$\hat{H}U_k = U_k T_k, \tag{4.1}$$

and that its columns, $U_k = (|K_1\rangle, \ldots, |K_k\rangle)$, known as Krylov-vectors, are ortho-normal.

To illustrate how the Lanczos algorithm works we first consider $k = 1, 2$ and 3, and then show what happens for any k.

For $k = 1$, equation (4.1) becomes $\hat{H}|K_1\rangle = A_1|K_1\rangle$. In this case there are no off-diagonal elements in T_k and we can easily compute A_1 to be $\langle K_1|\hat{H}|K_1\rangle$.

For $k = 2$, equation (4.1) consists of two rows

$$\begin{cases} \hat{H}|K_1\rangle = A_1|K_1\rangle + B_1|K_2\rangle & \text{(I)} \\ \hat{H}|K_2\rangle = B_1|K_1\rangle + A_2|K_2\rangle & \text{(II)} \end{cases}. \tag{4.2}$$

Because the Krylov vectors $|K_1\rangle$ and $|K_2\rangle$ are ortho-normal, row (I) yields that $A_1 = \langle K_1|\hat{H}|K_1\rangle$, B_1 is the norm of $B_1|K_2\rangle = \hat{H}|K_1\rangle - A_1|K_1\rangle$ and the second Krylov vector is $|K_2\rangle = \frac{1}{B_1}\left(\hat{H}|K_1\rangle - A_1|K_1\rangle\right)$. Row (II) implies that $A_2 = \langle K_2|\hat{H}|K_2\rangle$.

For $k = 3$, we obtain three relations from equation (4.1)

$$\begin{cases} \hat{H}|K_1\rangle = A_1|K_1\rangle + & B_1|K_2\rangle & & \text{(I)} \\ \hat{H}|K_2\rangle = B_1|K_1\rangle + & A_2|K_2\rangle + B_2|K_3\rangle & & \text{(II)} \\ \hat{H}|K_3\rangle = & B_2|K_2\rangle + A_3|K_3\rangle & & \text{(III)}. \end{cases}. \tag{4.3}$$

Row (I) is the same as for $k = 2$ and yields the same result. Row (II) includes one additional term, from which it is possible to obtain that B_2 is the norm of $B_2|K_3\rangle = \hat{H}|K_2\rangle - B_1|K_1\rangle - A_2|K_2\rangle$ and that the third Krylov vector is $|K_3\rangle = \frac{1}{B_2}\left(\hat{H}|K_2\rangle - B1_1|K_1\rangle - A_2|K_2\rangle\right)$. From row (III) it follows that the third diagonal element is $A_3 = \langle K_3|\hat{H}|K_3\rangle$.

Iterating this process to any $k > 1$ gives that

$$\begin{aligned} A_k &= \langle K_k|\hat{H}|K_k\rangle, \\ B_k &= \text{norm}\left(\hat{\mathrm{H}}|\mathrm{K_k}\rangle - \mathrm{B_{k-1}}|\mathrm{K_{k-1}}\rangle - \mathrm{A_k}|\mathrm{K_k}\rangle\right), \\ |K_{k+1}\rangle &= \frac{1}{B_k}\left(\hat{H}|K_k\rangle - B_{k-1}|K_{k-1}\rangle - A_k|K_k\rangle\right). \end{aligned} \tag{4.4}$$

The choice of initial Krylov-vector, $|K_1\rangle$, determines which eigenvectors that can be computed. Assuming that $|\Psi_k\rangle$ for $k = 1, \ldots, N$ (with $N \leq D$) are

eigenvectors to $\hat{H}$ with corresponding eigenvalues E_k such that $E_k \leq E_{k+1}$, and if $|K_1\rangle = \sum_{k=1}^{N} \eta_k |\Psi_k\rangle$ and $\eta_k \neq 0$ for all k, then T_N will have the eigenvalues E_k [50]. What is more useful, is the property that the extremal eigenvalues converge before the internal ones making the algorithm suitable for the study of quantum systems where typically the smallest eigenvalues are of primary interest. Therefore, it is possible to stop the iterations when the eigenvalues of interest have converged. In this work we have been employing two different convergence criteria. Assuming that we are interested in eigenvalues up to E_k, Bacchus can either converge *eigenvalues* or *eigenvectors* in accordance with:

Eigenvalues: Iterate Lanczos until

$$|\lambda_k^{n-1} - \lambda_k^n| < \varepsilon \tag{4.5}$$

where λ_k^n is the kth eigenvalue of T_n.

Eigenvectors: Iterate Lanczos until

$$\|\mathbf{w}_k^{n-1} - \mathbf{w}_k^n\|_2 < \varepsilon \tag{4.6}$$

where $\mathbf{w}_k^n$ is the kth eigenvector of T_n.

For both criteria, ε is a user-defined tolerance.

In theory the Krylov vectors will all be orthogonal to each other. However, due to the finite precision in floating-point arithmetics, this property is not guaranteed in practice. Therefore, it is necessary to include a reorthogonalization step after a number of Lanczos iterations. In JupiterNCSM this is done through a Gram-Schmidt process, currently applied after every Lanczos step. The entire Lanczos procedure is summarized as Algorithm 1.

Algorithm 1 Schematic outlining of the full Lanczos algorithm.

Data: $D \times D$ symmetric matrix H, initial Krylov vector $|K_1\rangle$
Data: $|K_0\rangle = 0$, $B_0 = 0$
Result: Tridiagonal matrix T_k with diagonal elements A_i and off-diagonal elements B_j where $i = 1, \cdots, k$ and $j = 1, \cdots, k-1$
Data: Iteration index k, starts at 1
while *eigen-spectrum of* T_k *is not converged* **do**
  $|w_k\rangle \leftarrow \hat{H} |K_k\rangle$
  $A_k \leftarrow \langle w_k | K_k \rangle$
  $|v_k\rangle \leftarrow |w_k\rangle - A_k |K_k\rangle - B_{k-1} |K_{k-1}\rangle$
  $B_k \leftarrow \sqrt{\langle v_k | v_k \rangle}$
  $|K_{k+1}\rangle \leftarrow \frac{|v_k\rangle}{B_k}$
  $k \leftarrow k + 1$
  Reorthogonalize Krylov vectors if needed.
end

Bacchus expects a configuration file named `bacchus.conf` to be present in the working directory. The configuration file should be structured as follows:

```
interaction:
{
  combination_table_file = "/path/to/comb.txt";
  evaluation_order_file = "/path/to/evaluation_order";
  index_lists_base_directory = "/path/to/index_lists";
  matrix_file_base_directory = "/path/to/m_scheme_interaction";
  num_neutrons = 3;
  num_protons = 3;
}
lanczos:
{
  krylow_vector_directory = "/path/to/krylow_vectors";
  max_num_lanczos_iterations = 100;
  convergence_tolerance = 1.0e-7;
  target_eigenvector = 1;
  converge_eigenvectors = false;
  eigenvector_directory = "/path/to/eigen_vectors";
}
```

The first block, `interaction`, contains six fields that Bacchus will relay to Minerva, the code responsible for the matrix-vector multiplication, which is described in more detail in section 4.1.2. The second block, `lanczos`, also contains six fields. The first one, `krylow_vector_directory`, is the path to store the Krylov vectors that Bacchus generates. The second field, `max_num_lanczos_iterations`, limits the number of Lanczos iterations that will be performed. If the convergence criterion is not reached within this number of Lanczos iterations, Bacchus will terminate anyway. `convergence_tolerance` is ε in equation (4.5) or (4.6). The field `target_eigenvector` instructs Bacchus for which eigenvector or eigenvalue to evaluate the convergence criterion. The field `converge_eigenvectors` tells Bacchus which convergence criterion to use, if it is set to false Bacchus will use the *eigenvalues* criterion , if it is set to true the *eigenvectors* criterion will be used instead. The last field, `eigenvector_directory`, specifies where the computed eigenvectors will be stored.

After each Lanczos iteration, Bacchus writes the eigenvalues of T_k to standard output. When Lanczos terminates, Bacchus computes all requested eigenvectors and store them in the specified directory. The format of the computed eigenvectors is the same as the format of the Krylov vectors and will be described in the Minerva section 4.1.2.

4.1.2 Matrix-vector multiplication: Minerva

The matrix-vector multiplication needed for the Lanczos algorithm is implemented in the code Minerva. As mentioned before, we never store the full Hamiltonian matrix but instead generate its matrix elements as they are needed. In this section this matrix-vector multiplication algorithm is explained, and the application-programmers interface (API) is described.

The matrix-vector multiplication is implemented in the proton-neutron formulation of the NCSM, as described in section 2.1.1. We can therefore express a Krylov

vector $|K_k\rangle$ as

$$|K_k\rangle = \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu} K_{n(i,j)}(|\pi_i\rangle \otimes |\nu_j\rangle) \Big|_{\substack{N_{\pi_i}+N_{\nu_j}\leq N_{\max} \\ M_{\pi_i}+M_{\pi_j}=M \\ (-1)^{L_{\pi_i}}\cdot(-1)^{L_{\nu_j}}=\Pi}}, \tag{4.7}$$

where D_π (D_ν) is the dimension of the proton (neutron) basis as defined in section 2.1.1 and K_n the amplitudes and N_{π_i} (N_{ν_j}) is the total HO excitation of state $|\pi_i\rangle$ ($|\nu_j\rangle$). To simplify the notation the conditions of NCSM truncation $N_{\pi_i} + N_{\nu_j} \leq N_{\max}$, conserved azimuthal angular momentum $M_{\pi_i} + M_{\nu_j} = M$, and total parity $(-1)^{L_{\pi_i}} \cdot (-1)^{L_{\nu_j}} = \Pi$ where L_{π_i} (L_{ν_j}) is the total orbital angular momentum of the protons (neutrons), will henceforth be implied but not explicitly written out.

The Hamiltonian, as defined in equation (1.2), can be rewritten as,

$$\begin{aligned}\hat{H} =& \hat{T}_{\text{int}} + \hat{V}_{\text{2NF}} + \hat{V}_{\text{3NF}} \\ =& \hat{T}_{\text{int}} + \overbrace{\hat{V}_{pp} + \hat{V}_{pn} + \hat{V}_{nn}}^{\hat{V}_{\text{2NF}}} \\ &+ \overbrace{\hat{V}_{ppp} + \hat{V}_{ppn} + \hat{V}_{pnn} + \hat{V}_{nnn}}^{\hat{V}_{\text{3NF}}}\end{aligned} \tag{4.8}$$

where $\hat{V}_{pp}$ and $\hat{V}_{ppp}$ ($\hat{V}_{nn}$ and $\hat{V}_{nnn}$) only act on $\mathcal{H}_p$ ($\mathcal{H}_n$) while the other terms act on the full Hilbert space, $\mathcal{H} = \mathcal{H}_p \otimes \mathcal{H}_n$. The kinetic energy term T_{int} is an iso-spin scalar, which in the proton-neutron formalism has the consequence $\langle\pi_i|\hat{T}_{\text{int}}|\nu_j\rangle = 0$ for all i and j.

The matrix-vector multiplication in Algorithm 1 can now be expanded to

$$|w_k\rangle = \hat{H}\,|K_k\rangle = \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu} K_{n(i,j)}\hat{H}(|\pi_i\rangle \otimes |\nu_j\rangle). \tag{4.9}$$

The action of the Hamiltonian on $|\pi_i\rangle \otimes |\nu_j\rangle$ results in the expression

$$\begin{aligned}\hat{H}\,(|\pi_i\rangle \otimes |\nu_j\rangle) =& \sum_{l=1}^{D_\pi}\sum_{m=1}^{D_\nu} (|\pi_l\rangle \otimes |\nu_m\rangle)\,(\langle\pi_l| \otimes \langle\nu_m|)\,H\,(|\pi_i\rangle \otimes |\nu_j\rangle) \\ =& \sum_{l=1}^{D_\pi} (|\pi_l\rangle \otimes |\nu_j\rangle)\ \langle\pi_l|\hat{T}_{\text{int}} + \hat{V}_{pp} + \hat{V}_{ppp}|\pi_i\rangle \\ &+ \sum_{m=1}^{D_\nu} (|\pi_i\rangle \otimes |\nu_m\rangle)\ \langle\nu_m|\hat{T}_{\text{int}} + \hat{V}_{nn} + \hat{V}_{nnn}|\nu_j\rangle \\ &+ \sum_{l=1}^{D_\pi}\sum_{m=1}^{D_\nu} (|\pi_l\rangle \otimes |\nu_m\rangle)\,(\langle\pi_l| \otimes \langle\nu_m|)\hat{V}_{pn} + \hat{V}_{ppn} + \hat{V}_{pnn}(|\pi_i\rangle \otimes |\nu_j\rangle).\end{aligned} \tag{4.10}$$

I will not describe how each and every term is treated. Instead I will focus on the terms $\hat{V}_{pp}$ and $\hat{V}_{ppn}$ since all other terms are treated analogously to one of these two. The operator $\hat{V}_{pp}$ can be written in second quantized form as

$$\hat{V}_{pp} = \sum_{a,a',b,b'} \langle a,b|V_{pp}|a',b'\rangle\ \hat{c}_a^{\pi\dagger}\hat{c}_b^{\pi\dagger}\hat{c}_{b'}^{\pi}\hat{c}_{a'}^{\pi}, \tag{4.11}$$

where $\langle a,b|V_{pp}|a',b'\rangle$ is the two proton matrix element, precomputed by the Mercury and Neptune codes. Therefore, the matrix element of $\hat{V}_{pp}$ needed in equation (4.10) can be expanded as,

$$\langle \pi_l|\hat{V}_{pp}|\pi_i\rangle = \sum_{a,a',b,b'} \langle a,b|V_{pp}|a',b'\rangle \ \langle \pi_l|\hat{c}_a^{\pi\dagger}\hat{c}_b^{\pi\dagger}\hat{c}_{b'}^{\pi}\hat{c}_{a'}^{\pi}|\pi_i\rangle \,. \tag{4.12}$$

To simplify matters, we organize the single-particle quantum numbers, a,b,a',b', in a list and let x be an index that points into that list. We then define $M_x^{pp} = \langle a,b|V_{pp}|a',b'\rangle$, and $\hat{t}_x^{pp} = \hat{c}_a^{\pi\dagger}\hat{c}_b^{\pi\dagger}\hat{c}_{b'}^{\pi}\hat{c}_{a'}^{\pi}$. Using these definitions we can write the sum as

$$\langle \pi_l|\hat{V}_{pp}|\pi_i\rangle = \sum_x M_x^{pp} \langle \pi_l|\hat{t}_x^{pp}|\pi_i\rangle \,. \tag{4.13}$$

The transition density $\langle \pi_l|\hat{t}_x^{pp}|\pi_i\rangle$ can only take the values 0, 1 and -1 since we are using M-scheme many-body states. Just as we precompute the matrix elements, we also precompute the transition densities using the program Anicre, described in section 4.1.3. This results in a list of tuples (l,i,x,φ) with all non-zero transition density elements $\langle \pi_l|\hat{t}_x^{pp}|\pi_i\rangle = (-1)^{\varphi}$. Since this list can be very long we split it into several smaller lists, L_p, that can be used in parallel. Each such list contains all transitions with a specific sum of single particle quantum numbers. The $\hat{T}_{\rm int}$, $\hat{V}_{ppp}$, $\hat{V}_{nn}$ and $\hat{V}_{nnn}$ terms are all treated in similar way, resulting in

$$\begin{aligned}
\langle \pi_l|\hat{T}_{\rm int}|\pi_i\rangle &= \sum_x M_x^{\rm int-pp} \langle \pi_l|\hat{t}_x^{pp}|\pi_i\rangle \,, \\
\langle \pi_l|\hat{V}_{pp}|\pi_i\rangle &= \sum_x M_x^{pp} \langle \pi_l|\hat{t}_x^{pp}|\pi_i\rangle \,, \\
\langle \pi_l|\hat{V}_{ppp}|\pi_i\rangle &= \sum_x M_x^{ppp} \langle \pi_l|\hat{t}_x^{ppp}|\pi_i\rangle \,, \\
\langle \nu_m|\hat{T}_{\rm int}|\nu_j\rangle &= \sum_y M_y^{\rm int-nn} \langle \nu_m|\hat{t}_y^{nn}|\nu_j\rangle \,, \\
\langle \nu_m|\hat{V}_{nn}|\nu_j\rangle &= \sum_y M_y^{nn} \langle \nu_m|\hat{t}_y^{nn}|\nu_j\rangle \,, \\
\langle \nu_m|\hat{V}_{nnn}|\nu_j\rangle &= \sum_y M_y^{nnn} \langle \nu_m|\hat{t}_y^{nnn}|\nu_j\rangle \,.
\end{aligned} \tag{4.14}$$

We can do a similar analysis of the $\hat{V}_{ppn}$ term. The big difference is that there are now second-quantization operators for both protons and neutrons. Therefore, the operator is

$$\hat{V}_{ppn} = \sum_{a,b,c,a',b',c'} \langle a,b,c|V_{ppn}|a',b',c'\rangle \ \hat{c}_a^{\pi\dagger}\hat{c}_b^{\pi\dagger}\hat{c}_c^{\nu\dagger}\hat{c}_{c'}^{\nu}\hat{c}_{b'}^{\pi}\hat{c}_{a'}^{\pi} \tag{4.15}$$

The matrix elements needed for equation (4.10) are rewritten as

$$\begin{aligned}
&(\langle \pi_l| \otimes \langle \nu_m|)\hat{V}_{ppn}(|\pi_i\rangle \otimes |\nu_j\rangle) = \\
&\sum_{a,b,c,a',b',c'} \langle a,b,c|V_{ppn}|a',b',c'\rangle \ \langle \pi_l|\hat{c}_a^{\pi\dagger}\hat{c}_b^{\pi\dagger}\hat{c}_{b'}^{\pi}\hat{c}_{a'}^{\pi}|\pi_i\rangle \ \langle \nu_m|\hat{c}_c^{\nu\dagger}\hat{c}_{c'}^{\nu}|\nu_j\rangle \,.
\end{aligned} \tag{4.16}$$

We see that we can reuse notation from the treatment of $\hat{V}_{pp}$ and recall that $\hat{t}^{pp}_x = \hat{c}^{\pi\dagger}_a \hat{c}^{\pi\dagger}_b \hat{c}^{\pi}_{b'} \hat{c}^{\pi}_{a'}$. Furthermore, analogous to having a list over all possible (a, b, a', b') indexed by x, we can introduce a list over all possible (c, c') indexed by y and and $\hat{t}^{n}_y = \hat{c}^{\nu\dagger}_c \hat{c}^{\nu}_{c'}$. Since the matrix element $\langle a, b, c|V_{ppn}|a', b', c'\rangle$ depends on both (a, b, a', b') and (c, c') it can be relabeled as $M^{ppn}_{x,y} = \langle a, b, c|V_{ppn}|a', b', c'\rangle$. Similar precomputations can be performed for $\hat{V}^{pn}$ and $\hat{V}^{pnn}$, such that these terms can be written as

$$\begin{aligned}
(\langle\pi_l| \otimes \langle\nu_m|)\hat{V}_{pn}(|\pi_i\rangle \otimes |\nu_j\rangle) &= \sum_{x,y} M^{pn}_{x,y} \langle\pi_l|\hat{t}^{p}_x|\pi_i\rangle \ \langle\nu_m|\hat{t}^{n}_y|\nu_j\rangle \,, \\
(\langle\pi_l| \otimes \langle\nu_m|)\hat{V}_{ppn}(|\pi_i\rangle \otimes |\nu_j\rangle) &= \sum_{x,y} M^{ppn}_{x,y} \langle\pi_l|\hat{t}^{pp}_x|\pi_i\rangle \ \langle\nu_m|\hat{t}^{n}_y|\nu_j\rangle \,, \\
(\langle\pi_l| \otimes \langle\nu_m|)\hat{V}_{pnn}(|\pi_i\rangle \otimes |\nu_j\rangle) &= \sum_{x,y} M^{pnn}_{x,y} \langle\pi_l|\hat{t}^{p}_x|\pi_i\rangle \ \langle\nu_m|\hat{t}^{nn}_y|\nu_j\rangle \,.
\end{aligned} \tag{4.17}$$

The matrix vector multiplication, expressed in equation (4.9), can be rewritten as the following

$$\begin{aligned}
|w_k\rangle =& \hat{H}\,|K_k\rangle \\
=& \sum_{i=1}^{D_\pi}\sum_{l=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{x} K_{n(i,j)}(M^{\text{int}-\text{pp}}_x + M^{pp}_x)\,\langle\pi_l|\hat{t}^{pp}_x|\pi_i\rangle\,(|\pi_l\rangle \otimes |\nu_j\rangle) \\
&+ \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{m=1}^{D_\nu}\sum_{x} K_{n(i,j)}(M^{\text{int}-\text{nn}}_x + M^{nn}_x)\,\langle\nu_m|\hat{t}^{nn}_x|\nu_j\rangle\,(|\pi_i\rangle \otimes |\nu_m\rangle) \\
&+ \sum_{i=1}^{D_\pi}\sum_{l=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{x} K_{n(i,j)} M^{ppp}_x\,\langle\pi_l|\hat{t}^{ppp}_x|\pi_i\rangle\,(|\pi_l\rangle \otimes |\nu_j\rangle) \\
&+ \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{m=1}^{D_\nu}\sum_{x} K_{n(i,j)} M^{nnn}_x\,\langle\nu_m|\hat{t}^{nnn}_x|\pi_i\rangle\,(|\nu_j\rangle \otimes |\nu_m\rangle) \\
&+ \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{l=1}^{D_\pi}\sum_{m=1}^{D_\nu}\sum_{x,y} K_{n(i,j)} M^{pn}_{x,y}\,\langle\pi_l|\hat{t}^{p}_x|\pi_i\rangle\ \langle\nu_m|\hat{t}^{n}_y|\nu_j\rangle\,(|\pi_l\rangle \otimes |\nu_m\rangle) \\
&+ \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{l=1}^{D_\pi}\sum_{m=1}^{D_\nu}\sum_{x,y} K_{n(i,j)} M^{ppn}_{x,y}\,\langle\pi_l|\hat{t}^{pp}_x|\pi_i\rangle\ \langle\nu_m|\hat{t}^{n}_y|\nu_j\rangle\,(|\pi_l\rangle \otimes |\nu_m\rangle) \\
&+ \sum_{i=1}^{D_\pi}\sum_{j=1}^{D_\nu}\sum_{l=1}^{D_\pi}\sum_{m=1}^{D_\nu}\sum_{x,y} K_{n(i,j)} M^{pnn}_{x,y}\,\langle\pi_l|\hat{t}^{p}_x|\pi_i\rangle\ \langle\nu_m|\hat{t}^{nn}_y|\nu_j\rangle\,(|\pi_l\rangle \otimes |\nu_m\rangle).
\end{aligned} \tag{4.18}$$

To simplify the continued discussion about this expression we let M^{pp}_x henceforth mean $M^{\text{int}-\text{pp}}_x + M^{pp}_x$ and likewise M^{nn}_y will mean $M^{\text{int}-\text{nn}}_y + M^{nn}_y$. The previously mentioned index lists, for protons L_p and neutrons L_n, and the matrix elements M^{pp}_x, M^{ppp}_x, M^{nn}_y, M^{nnn}_y, $M^{pn}_{x,y}$, $M^{ppn}_{x,y}$ and $M^{pnn}_{x,y}$, are all split into sub-blocks. Exactly how this split is performed will be brought up in section 4.1.3, since Minerva is completely agnostic to this. Note that this ignorance makes it possible to use Minerva, without any modifications, for other applications such as a valence shell model program. The sums in (4.18) are divided into blocks of computations that each uses one of

the three algorithms: Algorithm 2 for the proton-only terms, Algorithm 3 for the neutron-only terms and Algorithm 4 for the mixed proton and neutron interactions.

Algorithm 2 Matrix-vector multiplication of proton-only interactions in Minerva.

Data: L_p index list for protons
Data: M_x matrix elements list
Data: D_n dimension of the neutron space
Data: $\mathbf{K}_k$ in-vector with coefficients K_n
Result: $\mathbf{w}_k$ out-vector with coefficients $w_{n'}$
foreach *index tuple* (l, i, x, φ) *in* L_p **do**
  for $j \leftarrow 0$ **to** $D_n - 1$ **do**
    $n \leftarrow i \cdot D_n + j$
    $n' \leftarrow l \cdot D_n + j$
    $w_{n'} \leftarrow w_{n'} + (-1)^{\varphi} K_n M_x$
  end
end

Algorithm 3 Matrix-vector multiplication of neutron-only interactions in Minerva.

Data: L_n index list for neutrons
Data: M_y matrix elements list
Data: D_p dimension of the proton space
Data: D_n dimension of the neutron space
Data: $\mathbf{K_k}$ in-vector with coefficients K_n
Result: $\mathbf{w_k}$ out-vector with coefficients $w_{n'}$
foreach *index tuple* (m, j, y, φ) *in* L_n **do**
  for $i \leftarrow 0$ **to** $D_p - 1$ **do**
    $n \leftarrow i \cdot D_n + j$
    $n' \leftarrow i \cdot D_n + m$
    $w_{n'} \leftarrow w_{n'} + (-1)^{\varphi} K_n M_y$
  end
end

To perform the matrix vector multiplication Minerva needs several different files representing interaction matrices, vector blocks, and index lists. These are collectively referred to as array files. Furthermore, there are two other files needed: the `comb.txt` file and the evaluation-order file. These two files define in what order the matrix-vector multiplication should be performed and what data needs to be loaded.

The `comb.txt` file contains information on the various arrays and their sizes, and is necessary to decide which arrays to load and unload during the calculation. It is generated by the `cutcomb.pl` script, part of Anicre, described in subsection 4.1.3. The parts of this file that are used directly by Minerva are the headings `*** mp-states ***`, `*** Conn lists ***`, `*** Matrix-elements V (cross p-n)***` and `*** Matrix-elements V (same p/n)***` describing the vector blocks, index lists and the interaction matrices.

Algorithm 4 Matrix-vector multiplication for proton-neutron-mixed interactions in Minerva.

Data: L_p index list for protons
Data: L_n index list for neutrons
Data: $M_{x,y}$ matrix elements matrix
Data: D_n dimension of the neutron space
Data: $\mathbf{K}_k$ in-vector with coefficients K_n
Result: $\mathbf{w}_k$ out-vector with coefficients $w_{n'}$
foreach *index tuple* (l, i, x, φ_p) *in* L_p **do**
  foreach *index tuple* (m, j, y, φ_n) *in* L_n **do**
    $n \leftarrow i \cdot D_n + j$
    $n' \leftarrow l \cdot D_n + m$
    $w_{n'} \leftarrow w_{n'} + (-1)^{\varphi_p}(-1)^{\varphi_n} K_n M_{x,y}$
  end
end

The matrix-vector multiplication is performed in blocks, where each block corresponds to one vector block in, one vector block out, one matrix element file and one or two index lists. The order that these blocks are evaluated is defined in the evaluation order file generated by the Mars code, as discussed in section 4.1.5. Each line corresponds to a single block, consisting of either four or five indices. The two first indices tell which two vector blocks are needed. If there are four indices in total, the third one tells which index list is needed, while if there a total of five indices the next two tells which two index lists are needed. The last index always indicates which matrix block to load.

The matrix-element files correspond to different parts of the M^{pp}, M^{ppp}, etc. objects in the sums in equation (4.18). They are all located in the same directory with a path provided by the user. All files are named `<id-number>_matrix_elements`, where `<id-number>` is defined in the `comb.txt` file. The first 16 bytes contain two 64-bit unsigned integers. The first of the integers, d_n, corresponds to how many neutron states there are in the neutron basis, while the second, d_p, tells how many proton states there are in the proton basis. After these integers comes $8 \cdot d_n$ (in the neutron only case), $8 \cdot d_p$ (in the proton only case) and $8 \cdot d_n \cdot d_p$ (in the mixed case), bytes corresponding to the matrix elements expressed in IEEE double-precision floating-point format. In the mixed case the neutron dimension is the primary dimension, .i.e., there are d_p rows with d_n elements in each.

The vector-block files contain the coefficients K_n of the Krylov vector $|K_k\rangle$. The vector-block files corresponding to the same Krylov vector are all stored in the same directory provided by the user, and must be named `vec_<id-number>` where `<id-number>` is defined in the `comb.txt` file. Each file contains an array of IEEE double-precision floating-point numbers, whose length is dictated by the `comb.txt` file. Minerva requires two sets of vector-block files, one for $|K_k\rangle$ and one for $|K_{k+1}\rangle$, the latter for both reading and writing. Since different matrix-vector blocks might write to the same vector block in $|K_{k+1}\rangle$, Minerva keeps one copy per thread to

eliminate collisions and deadlocks.

The index-list files correspond to the L_p and L_n index lists. These files are all named `index_list_<id-number>`, where `<id-number>` is defined in the `comb.txt` file. They are all supposed to be stored in the same directory, provided by the user. Each file contains a list of triplets of 32-bit integers, (A, B, C), where A and B correspond to l and i if it is a proton list or m and j in the case of a neutron list. The last bit of C corresponds to the phase φ and the first 31 bits corresponds to x if it is a proton list and y if it is a neutron list.

The main interface of Minerva is through the calculation block scheduler, which is represented by a pointer to an opaque struct `scheduler_t`. To create a new instance of the scheduler, use the function

```
scheduler_t new_scheduler(
          evaluation_order_t evaluation_order,
          combination_table_t combination_table,
          const char *index_lists_base_directory,
          const char *matrix_file_base_directory,
          size_t maximum_loaded_memory
);
```

where `evaluation_order_t` represents the evaluation order file. The full interface is discussed later in this section. `combination_table_t` represents the `comb.txt` file, its interface will also be covered later. The pointer `index_list_base_directory` points to the file path where all the index lists are stored while, `matrix_file_base_directory` points to the path where matrix-element files are stored. The last argument defines the maximum memory that all the loaded index lists, matrix-element files and vector-block files may use in RAM at the same time.

To run a matrix-vector multiplication call the function

```
void run_matrix_vector_multiplication(
          const char *output_vector_base_directory,
          const char *input_vector_base_directory,
          scheduler_t scheduler
);
```

where the two pointers point to the file-paths of the input and output vector blocks.

To safely deallocate all memory reserved by the scheduler the user must call

```
void free_scheduler(
          scheduler_t scheduler
);
```

However, `scheduler` may not be `NULL`.

The evaluation order is represented by another opaque struct `evaluation_order_t`, which is created by the command

```
evaluation_order_t read_evaluation_order(
          const char *filename,
          combination_table_t combination_table
);
```

As the name suggests, this function reads the evaluation order from the file located at `filename`. The only other function concerning `evaluation_order_t` the user needs to know about is the corresponding deallocation function

```
void free_evaluation_order(
          evaluation_order_t evaluation_order
);
```

Just as for scheduler this too expects that the argument is not `NULL`.

The `combination_table_t` type represents the `comb.txt` file, and is also an opaque struct. An instance of it is created through

```
combination_table_t new_combination_table(
          const char *filename,
          size_t num_protons,
          size_t num_neutrons
);
```

This function reads the `comb.txt` file with the file path in `filename`. It also requires the number of protons and the number or neutrons, Z and N respectively. Lastly, to correctly deallocate an instance of `combination_table_t` use

```
void free_combination_table(
          combination_table_t combination_table
);
```

which, just as the other deallocation functions expects a non `NULL` value.

4.1.3 Transition densities: Anicre

Code II, Ancire, computes the transition densities $\langle\pi_l|\hat{t}^p_x|\pi_i\rangle$, $\langle\pi_l|\hat{t}^{pp}_x|\pi_i\rangle$, $\langle\pi_l|\hat{t}^{ppp}_x|\pi_i\rangle$, $\langle\nu_l|\hat{t}^n_y|\nu_i\rangle$, $\langle\nu_l|\hat{t}^{nn}_y|\nu_i\rangle$ and $\langle\nu_l|\hat{t}^{nnn}_y|\nu_i\rangle$, each produced as the index lists needed by Minerva. These are generated in an almost identical way. Therefore, I will only describe how the lists of $\langle\pi_l|\hat{t}^{pp}_x|\pi_i\rangle$ are generated. Anicre holds internally a list of all proton states $|\pi_i\rangle = \hat{c}^{\pi\dagger}_{a_1}\hat{c}^{\pi\dagger}_{a_2}\cdots\hat{c}^{\pi\dagger}_{a_Z}|\rangle$. For each of these states Anicre simulates the creation and annihilation operators to evaluate the operation of $\hat{t}^{pp}_x = \hat{c}^{\pi\dagger}_a\hat{c}^{\pi\dagger}_b\hat{c}^{\pi}_{b'}\hat{c}^{\pi}_{a'}$, by picking all combinations of two of the Z single proton states a_1 to a_Z as the a' and b' states in $\hat{t}^{pp}_x$ and then replace them with two new single particle states (that may be the same as those removed), resulting in a different many-proton state $|\pi_z\rangle$ with unknown index z. The phase φ is computed from the permutation needed to get the single proton states in a predefined order. This new many-proton state is looked up in a hash-table of the many-proton basis and its index is identified as $|\pi_l\rangle = |\pi_z\rangle$. Lastly l, i, x and φ is stored to the index list, in the format discussed in Minerva. Besides the index-list files, Anicre also produces the `comb.txt` file, which is described in detail in appendix B.

Anicre also generates a few other files. The connection files contain which one-, two- or three-body transition the xth index in the index list refers to. Consider, for example, an index list that corresponds to two-proton transitions. Then the corresponding connection file contains a list of tuples of indices, i_x, j_x, k_x, l_x into the single-proton basis, such that $\hat{t}^{pp}_x = \hat{c}^{\pi\dagger}_{a_{i_x}}\hat{c}^{\pi\dagger}_{a_{j_x}}\hat{c}^{\pi}_{a_{l_x}}\hat{c}^{\pi}_{a_{k_x}}$. These files are needed by Mercury and Neptune in order to do the transformation from J- to M-scheme of the potentials.

4.1.4 Preparing the Hamiltonian

The Hamiltonian needed by Bacchus and Minerva must be in a special binary format that is quick to read in parallel. Furthermore, Minerva works in M-scheme, but the 2NFs and 3NFs used throughout this work are provided in spin-coupled formats: JT-scheme for 2NFs and J-scheme for 3NFs. The following three codes: Mercury, Neptune and Vulcan are used to prepare the Hamiltonian in a format that can be used directly by Minerva.

Mercury

Mercury is the code that schedules the J- to M-scheme and JT- to M-scheme transformations. It divides the potential into 2NFs and 3NFs, since they use data from different input files. Mercury processes the 2NFs first. The 2NFs are read from a binary pAntoine interaction file that uses a JT-basis. Mercury, iterates over the 2NF matrix-element blocks in the `comb.txt` file and calls Neptune's JT- to M-scheme transformation on the JT-potential, read from the pAntoine file, for each of the output matrix-block files.

The 3NFs are processed in a slightly different way. The 3NF matrix-element blocks are combined into clusters such that every block in a cluster has the same difference in HO energy between bra and ket (`dEp` and `dEn`) and the same energy in the ket (`Dp` and `Dn`). For each such cluster, the 3NFs expressed in a J-scheme basis are read from a HDF5 data file (generated by the rel2lab code*), and an M-scheme matrix for that cluster is constructed by calling Neptune's J- to M-scheme transformation. Each block in a cluster is then constructed from that M-scheme matrix and stored to file. The transformations in each cluster are independent of the other clusters and are therefore executed in parallel.

Mercury uses the following call format:

```
1  mercury_J_scheme_to_internal.x <comb.txt> <Anicre output path>
2     <output path> [--num-protons <integer>]
3     [--num-neutrons <integer>]
4     [--single-particle-energy <integer>]
5     [--two-particle-energy <integer>]
6     [--max-loaded-memory <integer>]
7     [--finished-blocks-file <filepath>] [--no-2nf]
8     [--LEC-CE <float>] [--LEC-CD <float>]
9     [--LEC-C1 <float>] [--LEC-C3 <float>]
10    [--LEC-C4 <float>] <interaction path 2nf> [interaction path 3nf]
```

The first arguments are the path to the `comb.txt` file, followed by the path to the output of Anicre and the path to where Mercury can store the resulting matrix-element block files. Mercury also takes `--num-protons` and `--num-neutrons` as optional arguments. The default value is zero for both, which indicates that the kinetic term is to be excluded. The `--single-particle-energy` and `--two-particle-energy` correspond to the maximum HO excitation for the single particle basis and $N_{\max}$ for the two- and three-particle bases.† It is possible for the user to limit how much RAM

*This is an unpublished tranformation code developed by G. Jansen et al.

†The name of the latter is planned to be changed in the future since it is not descriptive enough.

memory that may be used by Mercury to store 3NF J-scheme matrix elements during execution, via the `--max-loaded-memory` flag. The default is 1 GB. The `--finished-blocks-file` argument takes a file path to where it can save information on which blocks that have been completed, providing a simple safeguard in case the program crashes prematurely. The `--no-2nf` flag tells Mercury to only use 3NFs. The 3NF file contains five different 3NF operators corresponding to the LECs c_1, c_3, c_4, c_D and c_E, and it is possible to include them with different LEC values, by using the `--LEC-<lec>` flags. Finally, Mercury takes a path to the 2NF file and a path to the 3NF one.

Neptune

Neptune is the brain of the transformation since the algorithms that are used are implemented in this code. In essence, the transformation is implemented as two back-to-back matrix-matrix multiplications,

$$V_{\text{Output}} = M V_{\text{Input}} M^{\text{T}} \tag{4.19}$$

where M is the transformation matrix, V_{Input} is the matrix to be transformed and V_{Output} is the transformed matrix. Currently, there are two different types of transformation matrices that are implemented: a two nucleon JT- to M-scheme and a three nucleon J- to M-scheme transformation. The matrix elements of the JT- to M-scheme transformation matrix are constructed from Clebsch-Gordon coefficients,

$$\langle \alpha, \beta | M_{JT} | (a,b) J, M, T, T_z \rangle = N_{a,b}^{J,T} \begin{pmatrix} j_a & j_b & J \\ m_\alpha & m_\beta & M \end{pmatrix}_{\text{CG}} \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & T \\ t_{z,\alpha} & t_{z,\beta} & T_z \end{pmatrix}_{\text{CG}} \tag{4.20}$$

where $\alpha = (a, m_\alpha, t_{z,\alpha}) = (n_a, l_a, j_a, m_\alpha, t_{z,\alpha})$, and similar for β and b. The factor $N_{a,b}$ comes from the antisymmetric properties of the fermionic states and can be expressed as

$$N_{a,b}^{J,T} = \begin{cases} 1 & \text{if } a \neq b \\ \sqrt{2} & \text{if } J+T \text{ is odd } \wedge\, a = b \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

The matrix for the three-nucleon J- to M-scheme transformation has matrix elements that are also constructed from Clebsch-Gordon coefficients,

$$\langle \alpha, \beta, \gamma | M_J | ((a,b) J_{ab}, c) J, M \rangle = \begin{pmatrix} j_a & j_b & J_{ab} \\ m_\alpha & m_\beta & m_\alpha + m_\beta \end{pmatrix}_{\text{CG}} \begin{pmatrix} J_{ab} & j_c & J \\ m_\alpha + m_\beta & m_\gamma & M \end{pmatrix}_{\text{CG}} . \tag{4.22}$$

In this case, there is no Pauli-principle normalization constant since it is already included in the provided 3NF matrix elements that we have been using. Note that Neptune is not called directly by the user, it is invoked by Mercury.

Vulcan

Vulcan is used for adding several sets of matrix-element blocks, corresponding to different terms in the Hamiltonian. Each of these terms can be scaled individually,

which is a feature used to construct the Hamiltonian for each training point in the EC method, discussed in chapter 5.

Vulcan reads every matrix-element block file with the same identification number, defined by the `comb.txt` file, for each of the provided terms, and extracts the floating-point array in each of them. These arrays are scaled according to a set of given coefficients and then added elementwise together to form the final array. The resulting array is then stored to file. Vulcan can process several different identification numbers in parallel.

Vulcans takes several command line arguments with the following call format:

```
vulcan.x <output_path> <comb.txt> <num protons> <num neutrons>
   [operator_path [-c/--coefficient <value>]]...
```

The first argument tells Vulcan where to store the produced matrix-element files. The second argument is the path to the relevant `comb.txt` file, and the two following arguments are the proton and neutron numbers of the nucleus. After this follows a list of operators. For each operator, a file path must be given, after which a coefficient is optional. If a coefficient is not provided for an operator the coefficient is assumed to be 1.0.

4.1.5 Evaluation order: Mars

Mars decides in what order the different matrix-vector multiplication blocks should be evaluated. It reads all `CALCBLOCK` lines in `comb.txt` and then finds an approximate solution to the problem of finding the optimal order in which Minerva should evaluate them. This problem is related to the traveling sales person (TSP) problem, where each calculation block can be seen as a town to visit and the distance between each calculation block depends on how much data that must be loaded (and unloaded). However, there are some significant differences compared to the TSP problem. For example, Minerva uses OpenMP parallelization to evaluate several calculation blocks at the same time. This would be equivalent of having multiple sales persons visiting different towns at once. Furthermore, the parallelization introduces a stochastic element in which order each thread picks the next calculation block. While there is an exact solution to the classic TSP problem, albeit very computationally expensive to find, the stochastic nature of Minerva, most likely eliminates the existence of an exact solution all together. Instead, only an approximation solution is sought for by Mars. At the moment, Mars sorts the calculation blocks by the most expensive array (vector block, index list or matrix-element block) to load in descending order.

The command-line interface of Mars follows the format:

```
mars.x <combination table file> <output file>
   [--only-two-nucleon-forces]
```

The first argument is a file path to the `comb.txt` file, the second required argument is where it should save the final evaluation order, in a format that Minerva can read. There is also an optional argument, `--only-two-nucleon-forces`, which can be used to only include calculation blocks involving 2NFs in the final evaluation order. This last option is used when only 2NFs are included in the Hamiltonian.

4.2 Benchmarks for ^{4}He, ^{6}Li

4.2.1 Validation

Results from JupiterNCSM have been validated against output from other NCSM codes for specific test cases. In particular, I have used JupiterNCSM and other codes to compute ground-state energies for ^{4}He and ^{6}Li using the NNLO$_{\text{sat}}$ interaction [35]. The ^{4}He results are presented in figure 4.2. In the left panel, JupiterNCSM:s ability to handle 2NFs is validated by comparing ^{4}He ground-state energies computed with only the 2NF part of the interaction with the same observable computed with pAntoine, see table 2.1. The bottom left panel shows the relative difference between the results, which can actually be traced to the rounding in the printout of pAntoine.

The correct handling of 3NFs is validated by computing the ground-state energy of ^{4}He with the full NNLO$_{\text{sat}}$ interaction, as shown in the right panels of figure 4.2. However, since pAntoine is not capable of handling 3NFs these results are compared instead with nsopt, see section 2.2. The relative difference is shown in panel (d). This difference is a bit larger than in the 2NF-only case but can be fully accounted for due to the use of the isoscalar approximation of the nuclear force [51] that is used in nsopt.

The ^{4}He benchmarks do not test all components of the 3NF. Since there are only two particles of each type, the three-proton and three-neutron pieces of the 3NF is excluded. Furthermore, non-trivial cases for the 2NF that only involves protons or neutrons are also excluded, because there are only two protons and two neutrons. Therefore, the ground-state energy of ^{6}Li was also considered, again using the NNLO$_{\text{sat}}$ interaction. The results of the ^{6}Li benchmarks are presented in figure 4.3. Just as for ^{4}He, the 2NF-only ground-state energies are compared against results obtained with pAntoine, as shown in panel (a). The observed difference, of less than 10^{-6} can also in this case be connected to rounding errors in the output from pAntoine. The 3NF results are compared against the NCSD code, see table 2.1, since the nsopt code is limited to $A \leq 4$ nuclei. The results of these benchmarks are presented in panel (b). As can be seen, the $N_{\max} = 0$ binding energies differ with significantly. This is because the ground state at $N_{\max} = 0$ is actually the $J = 0$ state, whereas for $N_{\max} \geq 2$ it is the $J = 1$ state. The basis that was used in JupiterNCSM for ^{6}Li only include many-body states with a total $M = 1$, and thus the result does not include the correct $J = 0$ ground-state energy. For higher $N_{\max}$ the relative difference is on the order of 10^{-4}. This result is deemed acceptable and we have therefore not investigated the origin of the discrepancy any further.

4.2.2 Time complexity

There are three different time-consuming steps in the execution of JupiterNCSM. These are: 1) the Lanczos-step in Bacchus, 2) J- to M-scheme transformation in Mercury and Neptune, and 3) the computation of the transition densities with Anicre. For the application of EC, discussed in detail in chapter 5, only the Lanczos

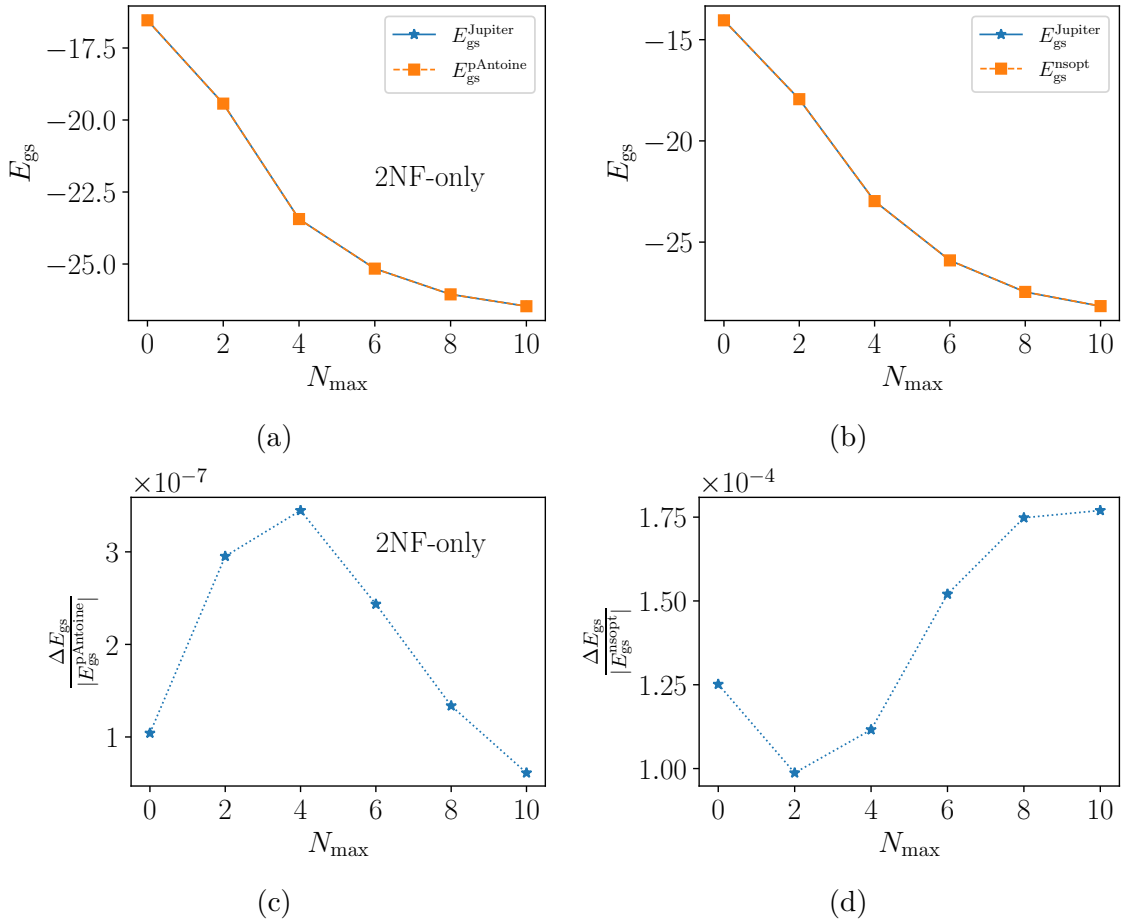

Figure 4.2: The ground-state energy of ^{4}He computed with the NNLO$_{sat}$ interaction [35], computed at $\hbar\Omega = 20$ MeV. In panel (a) only the 2NF part of the interaction is included and JupiterNCSM is compared with pAntoine. In panel (b) both the 2NF part and the 3NF part of the interaction are included and JupiterNCSM is compared with nsopt. The bottom panels, (c) and (d), show the relative differences.

step must be performed for each training vector, while the other two steps only need to be done once per nucleus and model space. The tests below are only for Bacchus and Mercury, since it was not possible to time Anicre consistently on the same machine for all cases, but the most demanding case, ^{6}Li $N_{max} = 10$ took in total approximately 48 hours. The machine used for the test here has two Intel Xeon Gold 6130 CPUs with 16 cores each, 384 GB of RAM and 874 GiB of usable local disk. All tests where run in parallel with 32 threads.

In figure 4.4, the execution time for Bacchus, and the average time per Lanczos iteration, are shown for ^{4}He and ^{6}Li with and without 3NFs. All timings are shown as function of the NCSM truncation. As can be seen, the execution times for ^{6}Li with 3NFs are much larger than the other cases. For $N_{max} = 10$ it takes about 25 hours to reach a converged ground-state. Notably, this run also breaks the almost exponential growth in execution time that is seen for the lower truncations. The reason, is that all matrix-element blocks fit in RAM for the lower truncations, and

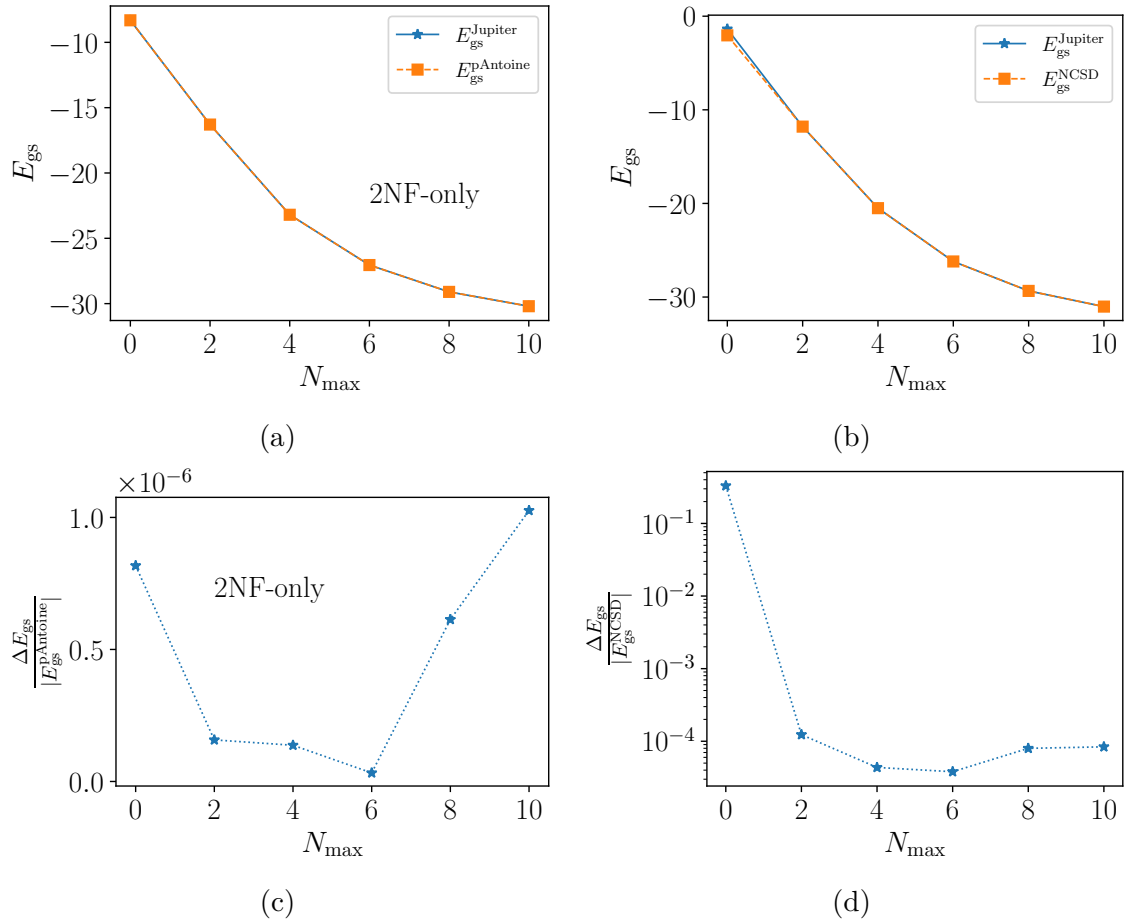

Figure 4.3: The ground-state energy of ^{6}Li computed with the NNLO$_{sat}$ interaction [35], with $\hbar\Omega = 20$. In panel (a) only the 2NF part of the interaction is included and JupiterNCSM is compared with pAntoine. In panel (b) both the 2NF part and the 3NF part of the interaction are included and JupiterNCSM is compared with NCSD. The bottom panels, (c) and (d), show the relative differences.

only needed to be read once. However, for $N_{max} = 10$, Minerva cannot fit all blocks in RAM at the same time. Several blocks therefore needed to be discarded and later read from disk again multiple times per Lanczos iteration.

The total execution time for Mercury as it is used to generate 3NF matrix-elements for ^{6}Li is shown in the left panel of figure 4.5 for different model spaces. As can be seen, the execution time increases faster than exponential. The reading of data from the input 3NF file is more time consuming than the actual transformation. Finally, the total number of generated matrix elements is shown in the right panel of figure 4.5. The number of matrix elements produced by Mercury (blue, solid curve) grows roughly as $D_{A,Z}(N_{max})^{5/4}$. This can be contrasted with the total number of non-zero matrix elements in the full ^{6}Li, $N_{max} = 10$ Hamiltonian (green, dashed curve) which grows as $D_{A,Z}(N_{max})^{3/2}$— also shown as a function of $D_{A,Z}(N_{max})$ in figure 2.2. Therefore, it is clear that we save memory by not storing the full Hamiltonian.

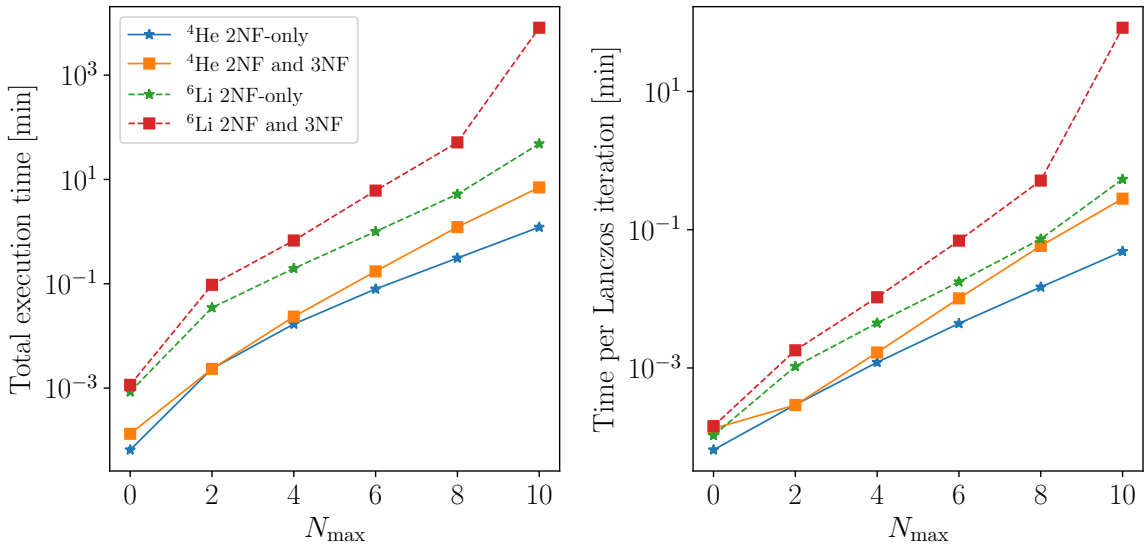

Figure 4.4: Total execution time for the Lanczos diagonalization performed by Bacchus (left panel), and the average time per iteration (right panel).

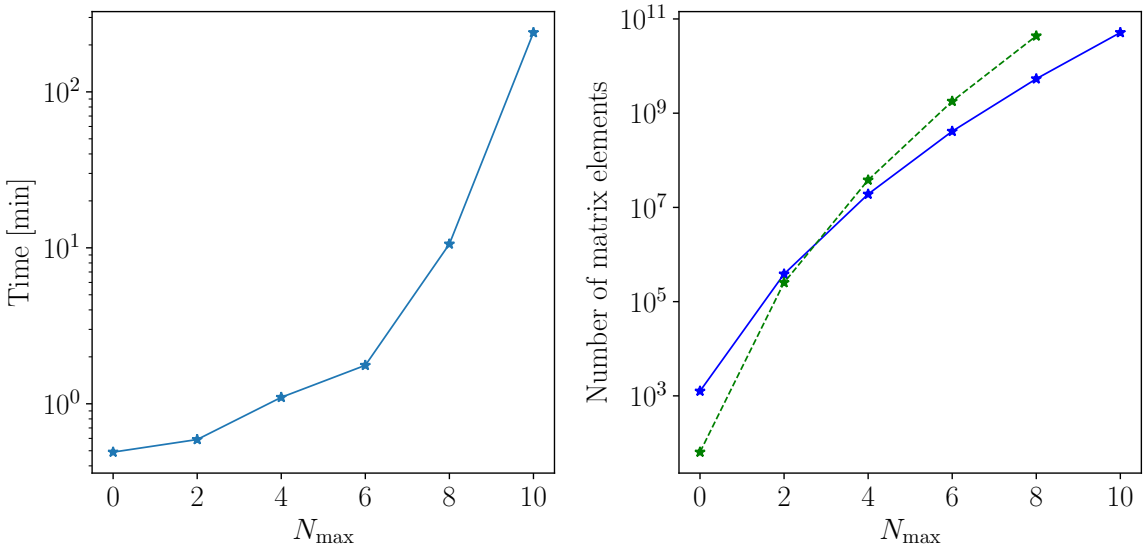

Figure 4.5: Left panel: Execution time for the generation of 2NF and 3NF matrix elements for ^{6}Li with Mercury. Right panel: The total number of matrix elements generated by Mercury (blue, solid curve) and the total number of non-zero matrix elements in the 6Li Hamiltonian (green, dashed curve).

4.2.3 Lanczos convergence

As discussed in section 4.1.1 we use two different convergence criteria: the convergence of eigenvalues or the convergence of eigenvectors. To illustrate the difference

between these two criteria I have plotted the four lowest eigenvalues of the ^{6}Li Hamiltonian as a function of the number of Lanczos iterations in figure 4.6 (top panel). The middle panel shows the difference between eigenvalues from two consecutive iterations for the four lowest states. This measure is used in the eigenvalue convergence criterion where the iterations are stopped when the difference reaches below a specified tolerance. The bottom panel shows the difference in the ground-state eigenvector, computed using the Euclidean norm in the Krylov space and is therefore not a time consuming operation. This difference is used for the eigenvector convergence criteria.

As can be seen, the eigenvalues show signs of level crossings as the iteration number increases. For example, the ground state and the first excited state cross each other between iterations 40 and 50. Therefore, stopping too early, either by setting a too small maximum number of iterations or by setting a too large convergence tolerance, can result in finding the incorrect ground state. Furthermore, we see that the eigenvectors converge much slower than the eigenvalues. Therefore, it is possible to get away with fewer iterations if only the eigenvalues are of interest. However, for the application of generating training-vectors for EVC emulators the convergence of the eigenvectors is of interest.

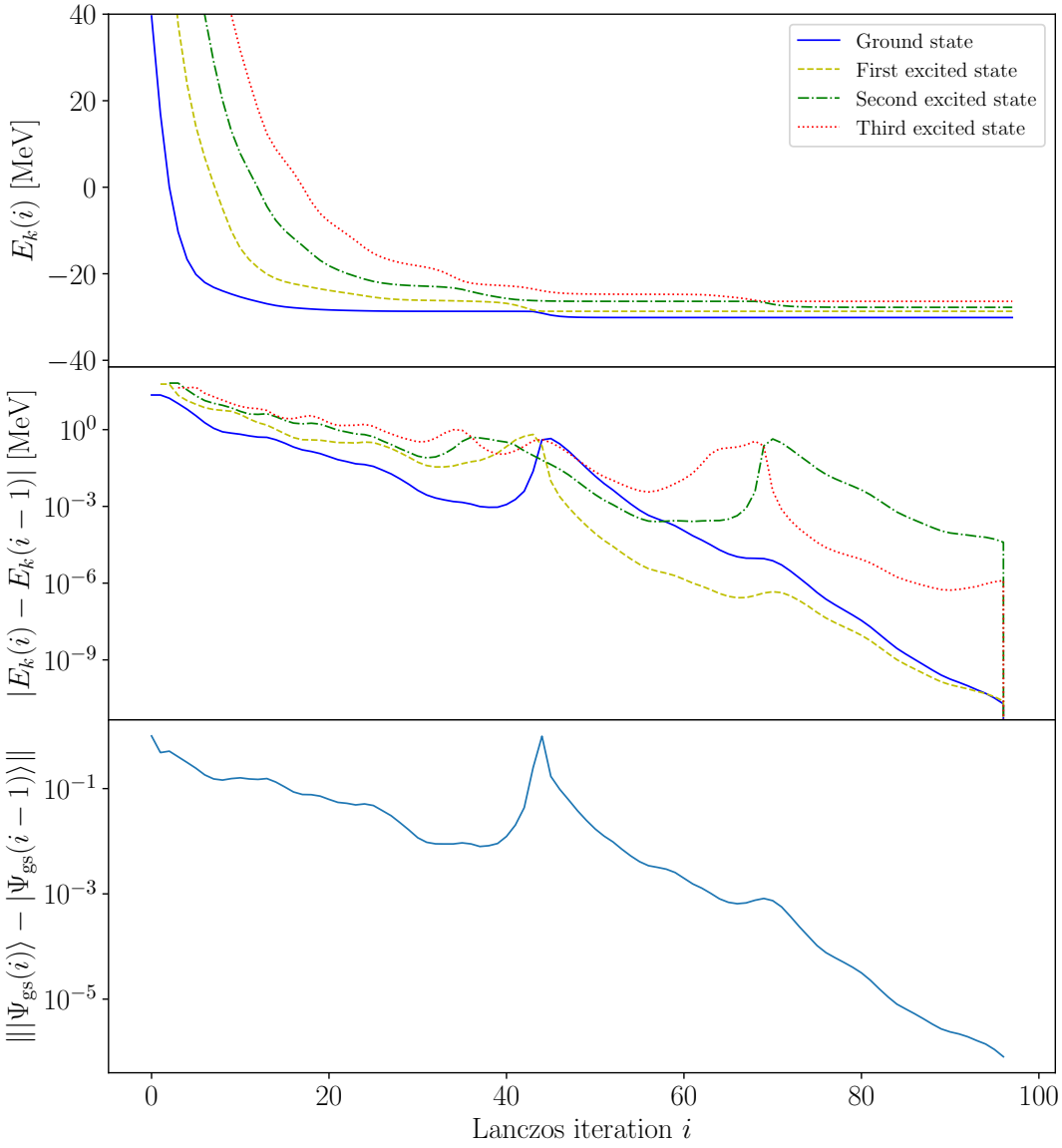

Figure 4.6: Convergence of the four lowest eigenvalues for ^{6}Li at $N_{\max} = 10$ as a function of the number of Lanczos iterations (top panel). The change in eigenvalue between current and previous Lanczos iteration (middle panel), and the change in the ground state wave function (bottom panel).

Chapter 5

Eigenvector continuation

As already established in chapter 2, there is a dilemma between the computational complexity introduced by, and the need for 3NFs, in nuclear structure calculations. In chapter 3 and Paper I this dilemma was addressed by exploring the possibility of approximating 3NFs as effective 2NFs with the NO2B approximation. While this approximation was shown to be potentially useful for larger nuclei, it does introduce spurious CM-excitations which can cause unphysical behavior in observables for light systems. In this chapter, together with Paper II and Paper III, an alternative approach to resolve the dilemma is explored; emulation of many-body observables using eigenvector continuation (EVC) [20–22]. EVC can be used to construct an emulator for eigenvectors and eigenvalues of a parameter-dependent many-body Hamiltonian to very high accuracy by just performing a small number of exact many-body calculations. These emulators can be used to compute approximate many-body observables for a vast set of parameter values at a very low computational cost.

The basic problem that EVC aims to solve is the approximate diagonalization of a linear operator $H(\mathbf{c})$ on some large-dimensional Hilbert space $\mathcal{H}$, with a smooth dependence on m parameters $\mathbf{c} \in \mathbb{C}^m$. Furthermore, it is assumed that $H(\mathbf{c})$ is Hermitian for real parameter values, $\mathbf{c} \in \mathbb{R}^m$. In particular, it is of interest to find specific eigenvalues $E_i(\mathbf{c})$ and eigenvectors $|\psi_i(\mathbf{c})\rangle$ for parameter values in a domain $\Omega \subset \mathbb{R}^m$.

EVC requires a set of known eigenvectors $|\psi_i(\mathbf{c}_k)\rangle$ of the operator $H(\mathbf{c})$, for some μ number of selected parameter values $c_1, \ldots, c_k, \ldots, c_\mu$. These training vectors in this work are obtained through exact diagonalization of the matrix for the selected parameter values, and will be used to construct a subspace $S = \text{span}\{|\psi_\text{i}(\mathbf{c}_1)\rangle, \ldots, |\psi_\text{i}(\mathbf{c}_\mu)\rangle\}$ of $\mathcal{H}$. The operator $H(\mathbf{c})$ is then projected onto the subspace S yielding the subspace operator $\hat{M}$. Then there exists an eigenvalue, λ, of the projected operator $\hat{M}$, that can be taken as an approximation of the eigenvalue $E_i(\mathbf{c})$. Assuming that we are interested in the ground state, the smallest λ is the best approximation due to the variational principle. Since μ can be much smaller than $\dim(\mathcal{H})$, diagonalizing $\hat{M}$ is orders of magnitude less computationally costly than diagonalizing $H(\mathbf{c})$.

EVC is of particular interest if the operator is linearly dependent on the parameters

$$H(\mathbf{c}) = H_0 + c_1 H_1 + c_2 H_2 + \cdots + c_m H_m, \quad \mathbf{c} = (c_1, \ldots, c_m), \tag{5.1}$$

where H_0 to H_m are Hermitian operators. In this case a significant computational speedup can be achieved. Since the operators H_0 to H_m only need to be projected onto the subspace S once. The operator $\hat{M}$ is then a linear combination of these projections for any $\mathbf{c} \in \Omega$.

5.1 Numerical method

In this section, the details of the numerical implementation of EVC are presented. Furthermore a numerical example in python is explored in Appendix C.

Assume first that some eigenvalues and eigenvectors of $H(\mathbf{c})$, for some set of parameter values, can be obtained through numerically converged calculations. Let us say that for the parameter values $\mathbf{c} \in \{\mathbf{c}_1, \ldots, \mathbf{c}_\mu\}$ the following eigenvalue and eigenvector pairs are known,

$$\begin{aligned} |\phi_1\rangle &= |\psi_i(\mathbf{c}_1)\rangle\,, \quad E_i(\mathbf{c}_1) \\ &\vdots \qquad\qquad \vdots \\ |\phi_\mu\rangle &= |\psi_i(\mathbf{c}_\mu)\rangle\,, \quad E_i(\mathbf{c}_\mu). \end{aligned} \tag{5.2}$$

These vectors span a subspace $S \equiv \text{span}\{|\phi_1\rangle, \ldots, |\phi_\mu\rangle\}$. However, there is no guarantee that the vectors $|\phi_1\rangle, \ldots, |\phi_\mu\rangle$ are linearly independent. Therefore, the dimension of the subspace $d_S \equiv \dim(S)$ is less than or equal to μ.

The idea behind EVC is that eigenvectors and eigenvalues of the operator $H(\mathbf{c})$ depend smoothly on the parameters and follow continuous trajectories well approximated by the low dimensional subspace. Eigenvectors for parameter values in the vicinity of $\mathbf{c}_1, \ldots, \mathbf{c}_\mu$ will therefore be close to the subspace S, i.e., the difference between itself and its projection on S is small. Therefore, to obtain approximations of the target eigenvectors and eigenvalues at any parameter vector $\mathbf{c}$, the operator $H(\mathbf{c})$ is projected onto S, forming the $\mu \times \mu$ matrix $\hat{M}(\mathbf{c})$ with matrix elements

$$M_{i,j}(\mathbf{c}) = \langle\phi_i|H(\mathbf{c})|\phi_j\rangle\,. \tag{5.3}$$

As previously mentioned, the vectors $|\phi_1\rangle, \ldots, |\phi_\mu\rangle$, are not necessarily linearly independent. One can account for this by constructing the norm matrix $\hat{N}$ with matrix elements

$$N_{i,j} = \langle\phi_i|\phi_j\rangle\,. \tag{5.4}$$

To compute subspace-projected eigenvectors and eigenvalues of $H(\mathbf{c})$, the generalized eigenvalue problem,

$$\hat{M}(\mathbf{c})\mathbf{v}(\mathbf{c}) = \lambda\hat{N}\mathbf{v}(\mathbf{c}), \tag{5.5}$$

has to be solved for the smallest eigenvalue. The subspace projected eigenvector can then be computed as

$$\left|\psi_i^S(\mathbf{c})\right\rangle = \sum_{i=1}^{\mu} v_i(\mathbf{c})\,|\phi_i\rangle\,, \tag{5.6}$$

where $v_i(\mathbf{c})$ is the i:th component of $\mathbf{v}(\mathbf{c})$. Note that $\left|\psi_i^S(\mathbf{c})\right\rangle \neq |\psi_i(\mathbf{c})\rangle$, but if $\mathbf{c}$ is close to the training points the difference, between these two vectors is small. Furthermore, λ is taken as an approximation of $E_i(\mathbf{c})$.

In the case of linear dependence of $H(\mathbf{c})$, like in equation (5.1), it is trivial to show that also the subspace matrix $\hat{M}$ is linear in the parameter. From the definition of its matrix elements we get

$$\begin{aligned} M_{i,j}(\mathbf{c}) = \langle\phi_i|H(\mathbf{c})|\phi_j\rangle &= \langle\phi_i|H_0|\phi_j\rangle + c_1\langle\phi_i|H_1|\phi_j\rangle + \cdots + c_m\langle\phi_i|H_m|\phi_j\rangle \\ &= M_{0i,j} + c_1 M_{1i,j} + \cdots + c_m M_{mi,j}. \end{aligned} \tag{5.7}$$

Since the matrices $\hat{M}_0$ to $\hat{M}_m$ only need to be computed once, the matrix $\hat{M}(\mathbf{c})$ can then be easily constructed for any c by computing $\hat{M}(\mathbf{c}) = \hat{M}_0 + \sum_{j=1}^{m} c_k \hat{M}_k$ instead of projecting $H(\mathbf{c})$.

5.2 Analytical motivation

The EVC method can be motivated through analytic continuation of the eigenvectors as shown by D.Frame in his PhD thesis [21]. These analytical arguments for EVC are summarized in this section. I will consider the case with a single parameter to keep the notation simple.

Consider the eigenvectors $|\psi_i(c)\rangle$ of the operator $H(c) = H_0 + cH_1$, where H_0 and H_1 are Hermitian. These eigenvectors can be computed using perturbation theory around $c = 0$, in a region such that $|c| < |z|$ as illustrated in figure 5.1

$$|\psi_i(c)\rangle = \sum_{n=0}^{\infty} \left|\psi_i^{(n)}(0)\right\rangle \cdot \frac{c^n}{n!}, \tag{5.8}$$

where $\left|\psi_i^{(n)}(c)\right\rangle$ is the n:th derivative of $|\psi_i(c)\rangle$ with respect to c. The nearest singularities, z and $\bar{z}$, must be complex since both H_0 and H_1 are Hermitian.

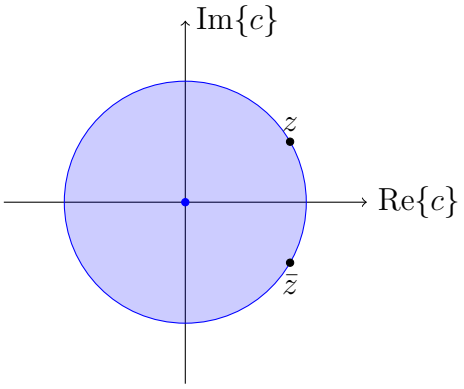

Figure 5.1: The blue shaded area illustrates the region of convergence of the expansion (5.8), limited by the singularities at z and $\bar{z}$.

Assume that eigenvectors for a parameter value c outside the region of convergence is sought for, i.e. $|c| > |z|$. Then perturbation around $c = 0$ will fail. Instead, assume that we can do perturbation around a different parameter value, $c = w$, such

that $|w| < |z|$ and $|c-w| < |z-w|$, as illustrated by the orange region in figure 5.2. In this case we get the expansion

$$|\psi_i(c)\rangle = \sum_{n=0}^{\infty} \left|\psi_i^{(n)}(w)\right\rangle \cdot \frac{(c-w)^n}{n!}. \tag{5.9}$$

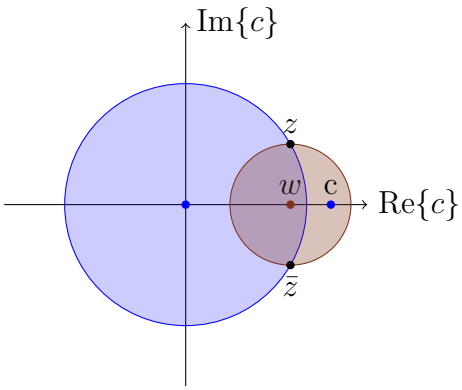

Figure 5.2: The blue shaded region illustrates the region of convergence of the expansions (5.8) while the orange shaded region illustrates the convergence region of expansion (5.9), limited by the singularities at z and $\bar{z}$.

The derivatives of the eigenvectors with respect to w in equation (5.9) can be expanded around 0 since $|w| < |z|$, yielding the expansion

$$\left|\psi_i^{(n)}(w)\right\rangle = \sum_{m=0}^{\infty} \left|\psi_i^{(n+m)}(0)\right\rangle \cdot \frac{w^m}{m!}. \tag{5.10}$$

The combination of, these two expansions result in

$$|\psi_i(c)\rangle = \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} \left|\psi_i^{(n+m)}(0)\right\rangle \frac{(c-w)^n w^m}{n!m!}. \tag{5.11}$$

This last expansion shows that $|\psi_i(c)\rangle$ can be written as a linear combination of derivatives of the eigenvectors evaluated at the parameter value 0, despite $|c| > |z|$.

Frame also considers the case when the singularity z is a branch point of the eigenvector $|\psi_i(c)\rangle$, and whether that will affect the EVC method. He recognizes two cases: a) $E_i(z)$ has multiplicity 1 and b) $E_i(z)$ has multiplicity $k > 1$. In case a) it is clear that as the branch point is circumnavigated once, the eigenvector can only change with a complex factor. Therefore, a new eigenvector can be constructed that is invariant on curves around the branch point.

The case b) is a bit more involved, since as the trajectory circumnavigate the branch point it shifts eigenvector and eigenvalue. To illustrate this feature, the real part of the two eigenvalues of a 2×2 $H(c)$ matrix has been plotted close to a branch point in figure 5.3. Each revolution can be described with a transformation $T(c)$ that cycles through all the k eigenvectors and eigenvalues. It turns out that it is possible to construct vectors $|\gamma_n(c,z)\rangle$ that are invariant under $T(c)$, and as $c \to z$ $|\gamma_n(c,z)\rangle$ becomes eigenvectors of $H(c)$ for $c = z$.

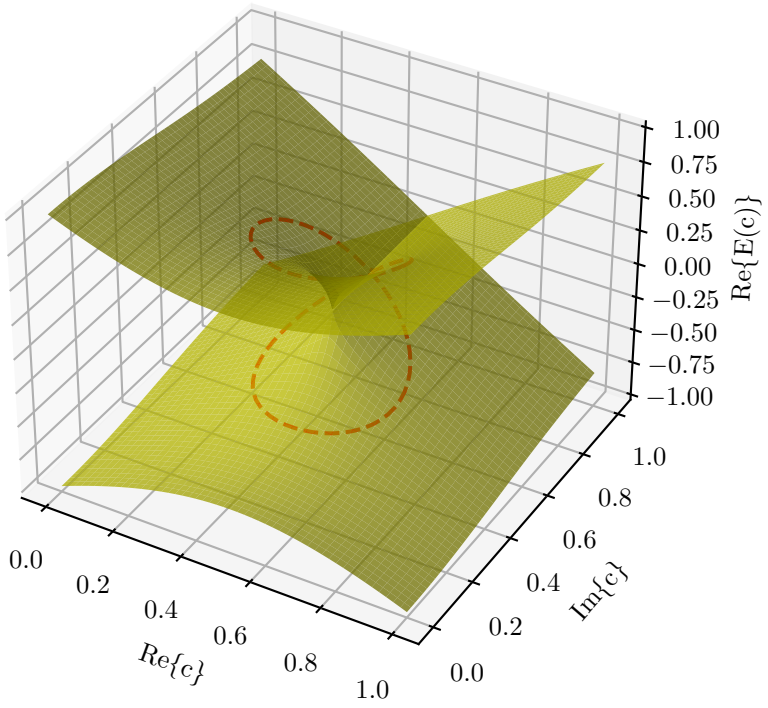

Figure 5.3: A branch point singularity of the eigenvalues of a 2×2 matrix of the form in equation (5.1). The red dashed curve indicates the path around the branch point.

5.3 ^{4}He, ^{6}He and ^{6}Li emulators

In Paper II we developed two-parameter EVC emulators for ^{6}Li and ^{6}He, in which the 3NF LECs c_D and c_E could be varied. We used these emulators to create a posterior predictive distribution of $A = 6$ observables with c_D and c_E drawn from a posterior distribution produced by Wesolowski et al. [23]. I stress that such a statistical study would not have been possible without fast and accurate emulators. The Hamiltonian is written as a sum of three terms

$$\hat{H}(c_D, c_E) = \hat{H}_0 + c_D \hat{H}_1 + c_E \hat{H}_2, \tag{5.12}$$

where the c_D- and c_E-parameter enter linearly and the operators are constructed in the following way

$$\begin{aligned} \hat{H}_0 =& \hat{T}_{\text{int}} + \hat{V}_{\text{2NF}} + \hat{V}_{\text{3NF},2\pi}, \\ \hat{H}_1 =& \hat{V}_{\text{3NF},1\pi-\text{ct}}, \\ \hat{H}_2 =& \hat{V}_{\text{3NF,ct}}. \end{aligned} \tag{5.13}$$

The H_0 operator contains most of the terms in the full system Hamiltonian shown in equation (1.2). However, the 3NF potential has been split up in the three diagrams shown in figure 1.1a.

To construct an emulator for ^{6}Li at $N_{\max} = 10$, we used JupiterNCSM to obtain training vectors from 16 different c_D and c_E values, eight of which were drawn from the posterior in reference [23], while the remaining eight were drawn from a large square: $c_D, c_E \sim \mathcal{U}(-2.5, 2.5)$. To obtain high precision training vectors

$$|\phi_k\rangle = |\psi_{\text{gs}}(c_{D,k}, c_{E,k})\rangle\,, \quad k = 1, 2, \ldots, 16, \tag{5.14}$$

the eigenvector convergence criterion in JupiterNCSM was used with a tolerance $\varepsilon = 10^{-6}$. The emulator was then constructed following the procedure in section 5.1.

To validate the ^{6}Li c_D,c_E emulator we sample an additional 40 c_D- and c_E-values, 20 from the posterior and 20 from the large square. Then we computed the ground-state energy using both the trained emulator and with JupiterNCSM for these new parameter values. Since only eigenvalues where compared for this test we used the eigenvalue convergence criterion with a tolerance $\varepsilon = 10^{-7}$. The validation and training points that are drawn from the posterior are plotted in figure 5.4. The x-axis corresponds to the JupiterNCSM binding energy, while the y-axis corresponds to the relative difference between the emulated ground-state energies and the JupiterNCSM ones.

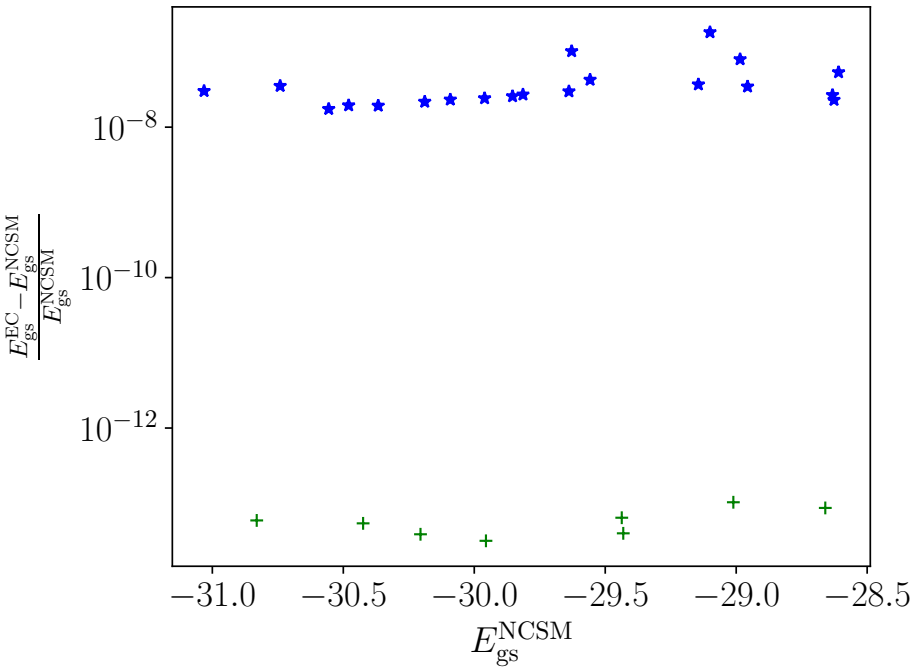

Figure 5.4: Validation of the two-parameter (^{6}Li, $N_{\max} = 10$) EVC emulator. The horizontal axis corresponds to the full NCSM ground-state energy, while the vertical axis is the relative difference between the emulated ground-state energy and the NCSM one. Green crosses corresponds to training points, and blue stars to the validation points.

As can be seen, the relative difference for the validation points is less than $2 \cdot 10^{-7}$. Considering that an eigenvalue convergence criteria with $\varepsilon = 10^{-7}$ was used for the validation points we conclude that our emulator is almost as accurate as full diagonalization within the parameter domain of interest. Within the large square, the accuracy was almost as good, being within 10^{-6} for points with energies close to the interval in the figure. However, for parameter values that give $E_{gs} \approx -100$ MeV, the relative difference increased to about 10^{-2}. The relative difference between prediction for the training points is less than 10^{-12}, which can be due to the numeric error in the floating-point calculations.

This emulator was used to compute ^{6}Li for $2.5 \cdot 10^6$ different c_D, c_E values all drawn from the posterior. This stunning set of computations took about five and a half minute to complete on a normal workstation PC, which can be compared to the 18 hours it takes to generate one validation point on a single compute node using JupiterNCSM.

The method of EVC was also used in Paper III. However, this application was even more challenging as we constructed and emulator using all 17 LECs for all 2NF and 3NF diagrams in delta-full χEFT up to NNLO resulting in the Hamiltonian sum

$$\hat{H}(\mathbf{c}) = H_0 + \sum_{j=1}^{17} c_j \hat{H}_j. \tag{5.15}$$

To generate a (^{6}Li, $N_{\max} = 10$) emulator we used 32 different non-implausible parameter values to obtain the training vectors

$$|\phi_k\rangle = |\psi_{\text{gs}}(\mathbf{c}_k)\rangle\,, \quad k = 1, 2, \ldots, 32. \tag{5.16}$$

Just as before, the emulator was constructed in accordance with section 5.1.

In figure 5.5 I present the validation of this emulator. In this case we used 16 validation points, the blue starts, and the 32 training points are the green stars.

In this case, the relative difference is larger than in figure 5.4 meaning that the emulator is less accurate. Still, while largest relative difference is almost $3 \cdot 10^{-2}$, the median is $2.3 \cdot 10^{-4}$. I did explore, in smaller model spaces (^{6}Li, $N_{\max} = 8$) how the relative difference depends on the number of training points. my findings was that it decreases with about a factor of 10 for every doubling of the number of training points.

These two examples of the EVC method illustrates that very accurate emulators can be produced, and that they can be used to explore huge parameter volumes with low computational cost. However, more training points are needed to accurately capture the parameter dependence of eigenvalues for increasing numbers of parameters in the Hamiltonian.

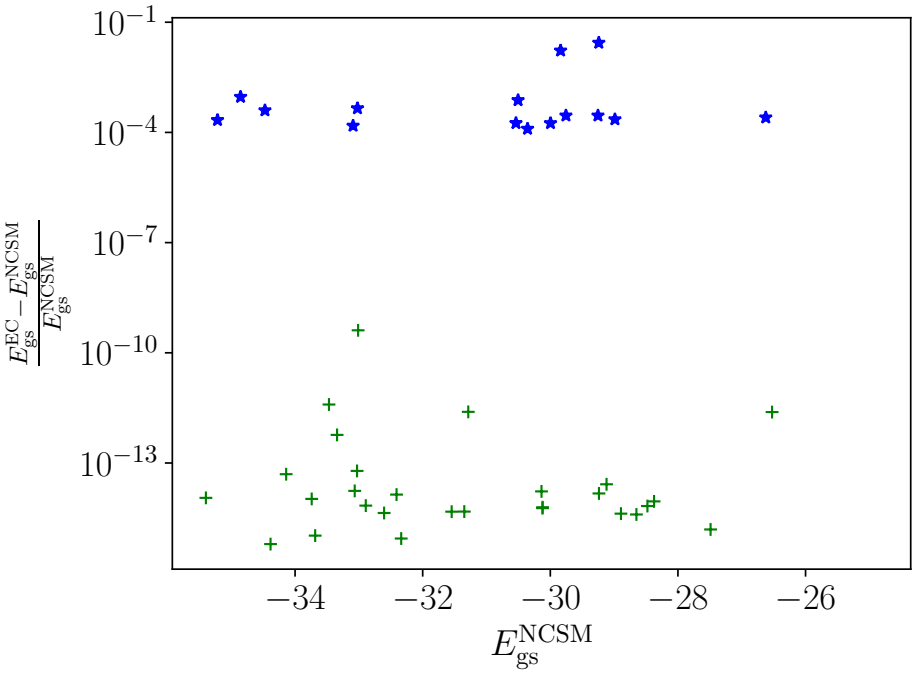

Figure 5.5: Validation plot for the 17 parameter (^{6}Li, $N_{\max} = 10$) emulator for a delta-full χEFT interaction. The horizontal axis corresponds to the full NCSM ground-state energy, while the vertical axis is the relative difference between the emulated ground-state energy and the NCSM one. Green crosses corresponds to training points, and blue stars to the validation points.

Chapter 6

Summary and outlook

The strong nuclear force is described by QCD [7–9]. However, due to QCD being a non-abelian gauge theory it is non-perturbative at low momentum-scales, where atomic nuclei exist. To get around this problem EFTs are used, in particular χEFTs that respect the approximate chiral symmetry (and symmetry breakings) are suitable for atomic nuclei [10–15]. In any EFT derived from QCD 3NFs emerges naturally. In this work I have studied different ways to incorporate 3NFs in the modeling of nuclear many-body systems via the NCSM. The nuclear many-body problem and the NCSM method were introduced in chapter 2. This discussion was continued in chapter 4 in connection with introducing JupiterNCSM, an in-house developed NCSM code that generates matrix elements of the A-nucleon Hamiltonian on-the-fly during iterative matrix diagonalization. These two chapters summarize how exact inclusion of 3NFs can be achieved.

Exact implementation of 3NFs. While JupiterNCSM has been used with success in this thesis, it is a very new code and still lacks important features. The list below contains my outlook of the most important future improvements of JupiterNCSM:

- *Implementation of other observables.* Currently the ground-state energy is the only observable that JupiterNCSM can compute. However, it would be very useful if other observables, such as radii and angular momentum, could be computed as well. The implementation of additional observables would not need much extra code, and could use most of the mechanisms already in place in the current implementation. The only needed extra code would generate the matrix-elements corresponding to the observable of interest in a one-, two- or three-nucleon basis. Then the same code for the matrix-vector multiplication, in Minerva, used by the Lanczos step, in Bacchus, can be utilized to compute expectation values for any given one-, two- or three-nucleon operator. In addition to this, if Anicre is extended to also compute four-particle, or arbitrary k-particle transition densities, in principle any four-, or k-particle observable could be computed.

- *Parallel diagonalization for multiple LEC parametrizations.* In the application of generating training vectors for EVC, it is necessary to run JupiterNCSM

repeatedly for different parameter values. This could be made more efficient if Lanczos iterations for different parameter values could be run in parallel on the same machine. This is because the Hamiltonian is constructed from the same operators, only scaled by different input parameters. The matrix-vector multiplication implemented in Minerva is limited by input and output to disk. Since the χEFT Hamiltonian used in this work consists of a sum of operators, each multiplied by a combination of the LECs, it is possible to load each operator separately and then construct the full matrix element on the fly for each vector of LECs. One set of Krylov-vectors could then be stored for each LEC vector. Furthermore, since modern Intel and AMD processors support single-instruction multiple-data (SIMD) parallelism, multiple matrix-vector multiplications can be performed at the same time.

- *MPI parallelization.* The study of larger A-nuclei (and larger model spaces) with JupiterNCSM can be made possible with the implementation of MPI parallelization. Currently, only single-node parallelization using OpenMP has been implemented. The list that define the evaluation order of the matrix-vector multiplication blocks, could be easily divided into several smaller lists, one per node, that could be processed independently. Therefore, MPI parallelization is not a difficult task. However, finding the optimal evaluation order for efficient parallelization will be a challenge due to its stochastic nature and relation to the TSP problem.

- *Remove the pAntoine dependence by generating an internal many-body basis.* Currently, JupiterNCSM requires an input many-body basis generated by pAntoine to compute the transition-densities. This dependence limits the useability of JupiterNCSM since pAntoine is not openly distributed. The structure of this basis is already known and could therefore be implemented directly in Anicre where it is used. This has not been done in the current work, but should be prioritized in the near future.

- *Implement a Lawson term.* The usual way of removing CM-excitations from the spectrum is to add a Lawson term to the Hamiltonian [29], see chapter 2 for more details. The study of excited states with JupiterNCSM will require a Lawson term. The NCSM states can be excited due to CM- and intrinsic-excitations. However, only the latter are of interest. Currently this is not implemented but it would be a minor extension of existing code.

- *Use JupiterNCSM for valence-space SM calculations with 3NFs.* Currently, JupiterNCSM can only do NCSM calculations. However, the technology used in Minerva to make the matrix-vector multiplication step in Lanczos is completely agnostic of any NCSM-specific details. Therefore, the code could be used to do valence-space SM calculations without any modifications, assuming that an M-scheme basis is used. What is needed is a new code to generate the transition-density-index lists and matrix-element files. With only minor modifications to Minerva, non-M-scheme bases, such as J-scheme or Jacobi-bases,

could be used as well. Specifically, the transition-density phase, φ, needs to be changed from a single bit to a floating-point number.

The NO2B approximation. The computational complexity associated with exact inclusion of 3NFs severely limits which nuclei can be studied. Therefore, I investigated an approximation scheme, known as the NO2B approximation, which replaces the full 3NF by an effective 2NF. This method was discussed in chapter 3 and is the main topic of Paper I. In particular, we investigated the breaking of translational symmetry caused by the approximation and how it induces mixing of intrinsic and CM excitations in the resulting ground states. We found that the problem of CM-mixing is present in both ^{4}He and ^{16}O. However, the consequences are much worse in the smaller nucleus, ^{4}He, while the NO2B approximation works better in the larger one, ^{16}O.

The NO2B project illustrated the existence of a potential problem with the approximation. However, this work can be extended. I have listed below a few more studies that can improve our understanding of the NO2B approximation:

- *The study of translation symmetry breaking of NO2B in other single-particle bases.* In the NO2B study performed in chapter 3 and Paper I we only used the HO-basis to construct the Slater-determinant reference state. However, other single-particle bases, such as the Hartree-Fock basis, might eliminate some of the problems associated with the breaking of translational symmetry. A Hartree-Fock basis constructed from a HO-single-particle basis will have less dependence on $\hbar\Omega$ as the maximal single-particle HO energy increases. Therefore, the expectation is that the ground-state energy contribution of the discarded three-body term in equation (3.5), $\hat{W}^{\text{3b}}_{\text{3NF}}$, will become independent of the basis frequency at sufficiently large basis truncation. Since the NO2B-approximation with normal-ordering in a Hartree-Fock basis is common in Coupled-Clusters calculations [31, 52, 53], I think such a study is motivated.

- *Investigate CM-mixing in the multi-reference NO2B approximation.* Instead of using a reference state based on a single Slater-determinant it is possible to use one based on a linear combination of such states. This approach is known as multi-reference NO2B. In this case, a ground-state computed in a smaller N_{max}-model space, or computed with a different many-body method, could be used as a reference state to incorporate more information about the system in the approximation. It is quite possible that the breaking of translational symmetry becomes smaller if the reference state is closer to the real ground-state. Furthermore, single-reference NO2B is mostly restricted to closed-core nuclei. However, with multi-reference NO2B there is no such restriction and the CM-mixing effect could then be studied in a larger set of nuclei.

- *How does the CM-mixing affect other observables?* The NO2B Hamiltonian contains an explicit CM-dependency, in contrast to all other nuclear observables that only depends on intrinsic and translationally invariant properties of the nucleus. Therefore, it is possible that the observed mixing of CM and

intrinsic excitations does not affect other observables in the same way as it does for the binding energy. The study in Paper I was mainly focused on the ground-state energy, and only briefly demonstrated the effect on point-proton radii.

Fast and accurate emulators. Nuclear interaction models constructed from χEFT involves many parameters, e.g. LECs, that must be determined by fitting to experiments. This fitting requires the ability to evaluate nuclear observables for many different parameter values. Doing this directly with the NCSM is a computationally demanding and extremely time-consuming process. Instead, it is desirable to construct nearly exact emulators for the desired observables from training eigenvectors obtained from just a few full NCSM calculations. EVC is a promising method to construct such emulators that are both fast and accurate [20–22]. This method is described in chapter 5 and utilized in Paper II and Paper III.

The working principle of emulators based on EVC is to use eigenvectors of the linearly-parameterized matrix evaluated at just a few parameter values to span a much smaller subspace. The matrix eigenvalue problem is then projected on to this subspace where it becomes much less computationally demanding to solve. The emulators constructed for Paper II can be used to compute the ground-state energy for ^{6}Li at one million different parameter values in a few minutes on a normal work station PC. Computing the same energy with JupiterNCSM at just one set of parameter values takes more than 24 hours on a dedicated compute node. What makes this even more impressive is that the emulators are just as accurate as the full NCSM calculations for c_D and c_E parameter values drawn from the posterior pdf from Wesolowski et al. [23].

The EVC emulators generated in this work have been shown to work extremely well. However, there are still several aspects that have not yet been explored:

- *Study excited states with EVC.* There is no restriction of the EVC method to just work with ground states. It is possible to include also excited states [20, 54]. However, in the work done in this thesis, we only used ground states for training and predictions due to the missing Lawson term in the JupiterNCSM code. I expect that emulating excited states using EVC will yield just as accurate results as we saw for ground states, since this has been the case in previous studies [20].

- *EVC for different nuclear observables.* It is possible to emulate other observables than energies with the EVC method. However, the subspace eigenvector does not in general correspond to an exact eigenvector of the full Hamiltonian. Therefore, computing other expectation values than energy eigenvalues can potentially show a strong dependence on the difference between emulated and exact eigenstate. However, previous studies have shown that this concern is not very serious and that other observables, such as nuclear radii and β-decay transitions, exhibit just as accurate results as energies [22, 23].

Bibliography

[1] Dag Fahlin Strömberg. "Three-Nucleon Forces Through Normal-Ordered Approximations". 44. MSc. thesis. 2016. URL: https://odr.chalmers.se/handle/20.500.12380/244698.

[2] Daniel Sääf. "Bridging scales in nuclear physics, Microscopic description of clusterization in light nuclei". PhD thesis. Chalmers, 2016. URL: https://research.chalmers.se/publication/236353.

[3] Hans Geiger and Ernest Rutherford. "On the scattering of the α-particles by matter". In: *P. R. Soc. Lond. A-Conta.* 81.546 (1908), pp. 174–177. DOI: 10.1098/rspa.1908.0067.

[4] H. Gegier, Ernest Marsden, and Ernest Rutherford. "On a diffuse reflection of the α-particles". In: *P. R. Soc. Lond. A-Conta.* 82.557 (1909), pp. 495–500. DOI: 10.1098/rspa.1909.0054.

[5] Hans Geiger and Ernest Rutherford. "The scattering of α-particles by matter". In: *P. R. Soc. Lond. A-Conta.* 83.565 (1910), pp. 492–504. DOI: 10.1098/rspa.1910.0038.

[6] E. Fermi. "Trends to a Theory of beta Radiation. (In Italian)". In: *Nuovo Cim.* 11 (1934), pp. 1–19. DOI: 10.1007/BF02959820.

[7] M. Gell-Mann. "Aschematic model of baryons and mesons". In: *Phys. Lett.* (1964). DOI: 10.1016/S0031-9163(64)92001-3.

[8] David J. Gross and Frank Wilczek. "Ultraviolet Behavior of Nonabelian Gauge Theories". In: *Phys. Rev. Lett.* 30 (1973). Ed. by J. C. Taylor, pp. 1343–1346. DOI: 10.1103/PhysRevLett.30.1343.

[9] H. David Politzer. "Reliable Perturbative Results for Strong Interactions?" In: *Phys. Rev. Lett.* 30 (1973). Ed. by J. C. Taylor, pp. 1346–1349. DOI: 10.1103/PhysRevLett.30.1346.

[10] R. Machleidt and D.R. Entem. "Chiral effective field theory and nuclear forces". In: *Physics Reports* 503.1 (2011), pp. 1 –75. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2011.02.001.

[11] Steven Weinberg. "Phenomenological Lagrangians". In: *Physica A* 96.1-2 (1979). Ed. by S. Deser, pp. 327–340. DOI: 10.1016/0378-4371(79)90223-1.

[12] Steven Weinberg. "Nuclear forces from chiral Lagrangians". In: *Phys. Lett. B* 251 (1990), pp. 288–292. DOI: 10.1016/0370-2693(90)90938-3.

[13] Steven Weinberg. "Effective chiral Lagrangians for nucleon - pion interactions and nuclear forces". In: *Nucl. Phys. B* 363 (1991), pp. 3–18. DOI: 10.1016/0550-3213(91)90231-L.

[14] Evgeny Epelbaum, Hans-Werner Hammer, and Ulf-G. Meissner. "Modern Theory of Nuclear Forces". In: *Rev. Mod. Phys.* 81 (2009), pp. 1773–1825. DOI: 10.1103/RevModPhys.81.1773. arXiv: 0811.1338 [nucl-th].

[15] H. W. Hammer, S. König, and U. van Kolck. "Nuclear effective field theory: status and perspectives". In: *Rev. Mod. Phys.* 92.2 (2020), p. 025004. DOI: 10.1103/RevModPhys.92.025004. arXiv: 1906.12122 [nucl-th].

[16] Bruce R. Barrett, Petr Navratil, and James P. Vary. "Ab initio no core shell model". In: *Prog. Part. Nucl. Phys.* 69 (2013), pp. 131–181. DOI: 10.1016/j.ppnp.2012.10.003.

[17] J. Hans, D. Jensen, and M. Goeppert-Mayer. "Electromagnetic Effects Due to Spin-Orbit Coupling". In: *Phys. Rev.* 85 (6 1952), pp. 1040–1041. DOI: 10.1103/PhysRev.85.1040.2.

[18] E. Caurier et al. "The Shell Model as Unified View of Nuclear Structure". In: *Rev. Mod. Phys.* 77 (2005), pp. 427–488. DOI: 10.1103/RevModPhys.77.427. arXiv: nucl-th/0402046.

[19] Robert Roth et al. "Medium-Mass Nuclei with Normal-Ordered Chiral NN+3N Interactions". In: *Phys. Rev. Lett.* 109 (5 2012), p. 052501. DOI: 10.1103/PhysRevLett.109.052501.

[20] Dillon Frame et al. "Eigenvector continuation with subspace learning". In: *Phys. Rev. Lett.* 121.3 (2018), p. 032501. DOI: 10.1103/PhysRevLett.121.032501. arXiv: 1711.07090 [nucl-th].

[21] Dillon K. Frame. "Ab Initio Simulations of Light Nuclear Systems Using Eigenvector Continuation and Auxiliary Field Monte Carlo". PhD thesis. May 2019. arXiv: 1905.02782 [nucl-th].

[22] S. König et al. "Eigenvector Continuation as an Efficient and Accurate Emulator for Uncertainty Quantification". In: *Phys. Lett. B* 810 (2020), p. 135814. DOI: 10.1016/j.physletb.2020.135814. arXiv: 1909.08446 [nucl-th].

[23] S. Wesolowski et al. "Fast & rigorous constraints on chiral three-nucleon forces from few-body observables". In: (Apr. 2021). arXiv: 2104.04441 [nucl-th].

[24] Maria Goeppert Mayer. "Nuclear Configurations in the Spin-Orbit Coupling Model. I. Empirical Evidence". In: *Phys. Rev.* 78 (1 1950), pp. 16–21. DOI: 10.1103/PhysRev.78.16.

[25] P. Federman and S. Pittel. "Unified shell-model description of nuclear deformation". In: *Phys. Rev. C* 20 (1979), pp. 820–829. DOI: 10.1103/PhysRevC.20.820.

[26] P. Navratil, G. P. Kamuntavicius, and B. R. Barrett. "Few nucleon systems in translationally invariant harmonic oscillator basis". In: *Phys. Rev. C* 61 (2000), p. 044001. DOI: 10.1103/PhysRevC.61.044001. arXiv: nucl-th/9907054.

[27] J. J. Sakurai and J. Napolitano. *Modern Quantum Mechanics*. Addison-Wesley, Pearson Education, Inc, 1994. ISBN: 9780321503367.

[28] Marcos Moshinsky. "Transformation brackets for harmonic oscillator functions". In: *Nucl. Phys.* 13.1 (1959), pp. 104–116. DOI: 10.1016/0029-5582(59)90143-9.

[29] D. H. Gloeckner and R. D. Lawson. "Spurious center-of-mass motion". In: *Phys. Lett. B* 53 (1974), pp. 313–318. DOI: 10.1016/0370-2693(74)90390-6.

[30] I. Stetcu, B. R. Barrett, and U. van Kolck. "No-core shell model in an effective-field-theory framework". In: *Phys. Lett. B* 653 (2007), pp. 358–362. DOI: 10.1016/j.physletb.2007.07.065. arXiv: nucl-th/0609023.

[31] G. Hagen et al. "Ab initio coupled-cluster approach to nuclear structure with modern nucleon-nucleon interactions". In: *Phys. Rev. C* 82 (2010), p. 034330. DOI: 10.1103/PhysRevC.82.034330. arXiv: 1005.2627 [nucl-th].

[32] R. J. Furnstahl, G. Hagen, and T. Papenbrock. "Corrections to nuclear energies and radii in finite oscillator spaces". In: *Phys. Rev. C* 86 (2012), p. 031301. DOI: 10.1103/PhysRevC.86.031301. arXiv: 1207.6100 [nucl-th].

[33] Sidney A. Coon et al. "Convergence properties of \it ab initio calculations of light nuclei in a harmonic oscillator basis". In: *Phys. Rev. C* 86 (2012), p. 054002. DOI: 10.1103/PhysRevC.86.054002. arXiv: 1205.3230 [nucl-th].

[34] C. Forssén et al. "Large-scale exact diagonalizations reveal low-momentum scales of nuclei". In: *Phys. Rev. C* 97.3 (2018), p. 034328. DOI: 10.1103/PhysRevC.97.034328. arXiv: 1712.09951 [nucl-th].

[35] A. Ekström et al. "Accurate nuclear radii and binding energies from a chiral interaction". In: *Phys. Rev. C* 91.5 (2015), p. 051301. DOI: 10.1103/PhysRevC.91.051301. arXiv: 1502.04682 [nucl-th].

[36] A. Ekström et al. "Optimized Chiral Nucleon-Nucleon Interaction at Next-to-Next-to-Leading Order". In: *Phys. Rev. Lett.* 110.19 (2013), p. 192502. DOI: 10.1103/PhysRevLett.110.192502. arXiv: 1303.4674 [nucl-th].

[37] Etienne Caurier and Frederic Nowacki. "Present Status of Shell Model Techniques". In: *Act. Phys. Pol. B* 30 (Mar. 1999), p. 705. URL: http://adsabs.harvard.edu/cgi-bin/nph-data_query?bibcode=1999AcPPB..30..705C&link_type=ABSTRACT.

[38] P. Navrátil and E. Caurier. "Nuclear structure with accurate chiral perturbation theory nucleon-nucleon potential: Application to ^{6}Li and ^{10}B". In: *Phys. Rev. C* 69 (1 2004), p. 014311. DOI: 10.1103/PhysRevC.69.014311.

[39] Calvin W. Johnson et al. *BIGSTICK: A flexible configuration-interaction shell-model code*. Jan. 2018. arXiv: 1801.08432 [physics.comp-ph].

[40] T. Dytrych et al. "Efficacy of the SU(3) scheme for ab initio large-scale calculations beyond the lightest nuclei". In: *Comput. Phys. Commun.* 207 (2016), pp. 202–210. DOI: 10.1016/j.cpc.2016.06.006. arXiv: 1602.02965 [nucl-th].

[41] Philip Sternberg et al. "Accelerating Configuration Interaction Calculations for Nuclear Structure". In: *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*. SC '08. Austin, Texas: IEEE Press, 2008. ISBN: 9781424428359.

[42] P. Navratil. *No-core shell model slater determinant code (NCSD) Unpublished.* 2011.

[43] Tor Djärv. *Three-nucleon forces in nuclear physics simulations*. Chalmers, 2019. URL: https://research.chalmers.se/publication/512770.

[44] W. H. Dickhoff and D. Van Neck. *Many-Body theory exposed! Propagator Description of Quantum Mechanics*. 2nd ed. Vol. 1. 5 Toh Tuck Link, Singapore 596224: World Scientific Publishing Co. Pte. Ltd., 2008. ISBN: 13 978-981-281-379-4.

[45] N. M. Parzuchowski et al. "Ab initio electromagnetic observables with the in-medium similarity renormalization group". In: *Phys. Rev. C* 96.3 (2017), p. 034324. DOI: 10.1103/PhysRevC.96.034324. arXiv: 1705.05511 [nucl-th].

[46] H. Hergert et al. "The In-Medium Similarity Renormalization Group: A Novel Ab Initio Method for Nuclei". In: *Phys. Rept.* 621 (2016), pp. 165–222. DOI: 10.1016/j.physrep.2015.12.007. arXiv: 1512.06956 [nucl-th].

[47] G. Hagen, T. Papenbrock, and D. J. Dean. "Solution of the center-of-mass problem in nuclear structure calculations". In: *Phys. Rev. Lett.* 103 (2009), p. 062503. DOI: 10.1103/PhysRevLett.103.062503. arXiv: 0905.3167 [nucl-th].

[48] D. J. Dean and M. Hjorth-Jensen. "Coupled cluster approach to nuclear physics". In: *Phys. Rev. C* 69 (2004), p. 054320. DOI: 10.1103/PhysRevC.69.054320. arXiv: nucl-th/0308088.

[49] G. Hagen et al. "Coupled-cluster computations of atomic nuclei". In: *Rept. Prog. Phys.* 77.9 (2014), p. 096302. DOI: 10.1088/0034-4885/77/9/096302. arXiv: 1312.7872 [nucl-th].

[50] C. Lanczos. "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators". In: *J. Res. Nat. Bur. Stand.* (1950). DOI: 10.6028%2Fjres.045.026.

[51] G. P. Kamuntavičius et al. "Isoscalar Hamiltonians for light atomic nuclei". In: *Phys. Rev. C* 60 (4 1999), p. 044304. DOI: 10.1103/PhysRevC.60.044304.

[52] D. J. Dean and M. Hjorth-Jensen. "Coupled-cluster approach to nuclear physics". In: *Phys. Rev. C* 69 (5 2004), p. 054320. DOI: 10.1103/PhysRevC.69.054320.

[53] G. Hagen et al. "Neutron and weak-charge distributions of the ^{48}Ca nucleus". In: *Nature Phys.* 12.2 (2015), pp. 186–190. DOI: 10.1038/nphys3529. arXiv: 1509.07169 [nucl-th].

[54] Avik Sarkar and Dean Lee. "Convergence of Eigenvector Continuation". In: *Phys. Rev. Lett.* 126.3 (2021), p. 032501. DOI: 10.1103/PhysRevLett.126.032501. arXiv: 2004.07651 [nucl-th].

Appendix A

Minimal working example for JupiterNCSM

To illustrate how to use Code I, JupiterNCSM, to compute the energy spectrum of a nucleus I have constructed an example for ^{4}He using the NNLO$_{\text{sat}}$ interaction for $N_{\max} \leq 6$. The example consists of a shell script named `example_4_helium.sh` located in the sub directory example in the JupiterNCSM git repository. A set of data is also needed to run the example. Download instructions can be found in the readme file. The following commands illustrates a run of the example:

```
>> cd example
>> ./example_4_helium.sh
Usage ./example_4_helium.sh <nmax> [<k-particle-forces>]
    [--max-loaded-memory <memstr>]
>> ./example_4_helium.sh 6 3 --max-loaded-memory 4GB
...
Diagonalization end after 935722 mu s
eigensystem:
{
        num_eigenvalues = 11;
        eigenvalues = ( -25.90722535033397, 17.521187442926934,
           23.599734543083045, 35.906134844316433,
           53.329043345505333, 80.410359916207241,
           104.87999820268072, 119.21925859073369,
           129.67968655743007, 142.2911766330169,
           167.74531258016918 );
}
Bacchus ends after 1.02472e+06 mu s
```

As can be seen, the resulting spectrum is printed at the end of the run. However, it is also stored in a file named `example_run_2nf_and_3nf_nmax6/bacchus_results`. The final spectrum is stored at the very end of this file and can be viewed by the following command:

```
>> tail -n 6 example_run_2nf_and_3nf_nmax6/bacchus_results
eigensystem:
{
        num_eigenvalues = 11;
        eigenvalues = ( -25.90722535033397, 17.521187442926934,
           23.599734543083045, 35.906134844316433,
```

```
            53.329043345505333, 80.410359916207241,
            104.87999820268072, 119.21925859073369,
            129.67968655743007, 142.2911766330169,
            167.74531258016918 );
}
Bacchus ends after 1.02472e+06 mu s
```

As can be seen in the example above, if the script is run without any arguments, it will print a usage line and then exit. There is one obligatory argument to the script and that is `<nmax>` representing the NCSM basis truncation $N_{\max}$ that will be used. In this example it must take one of the values $\{0, 2, 4, 6\}$. In addition, there are two optional arguments:

- `<k-particle-forces>`: This argument can either take the values 2 if only 2NFs should be included or 3 if 2NFs and 3NFs should be included in the Hamiltonian. The default value of this argument is 2.

- `--max-loaded-memory <memstr>`: This argument allows the user to limit the amount of RAM that Bacchus and Mercury may use. This allows JupiterNCSM to run on systems with relatively low amount of RAM, however there must be enough memory to store the largest matrix-element block in RAM. The default value is `16GB` but in this example it will never need more than `1GB`.

The script needs to complete five different tasks to run JupiterNCSM. The first task is to unpack the test data, in case it is not already unpacked. This data contains the index lists generated by Anicre*, and the 2NFs and 3NFs from the N2LO_{sat} interaction. Once the test data is unpacked, the script runs Mercury in order to transform the 2NFs and 3NFs into the internal M-scheme format. If `<k-particle-forces>` is 2 then only the 2NFs will be transformed. To generate the order in which the matrix-vector multiplication should be performed, the script runs the Mars code. While Anicre generates which matrix-vector-multiplication blocks needs to be executed, Mars decides the evaluation order of the matrix-vector multiplication. Anicre is not fully aware of the naming of the index lists it generates, so the code Aurora is used to rename the index lists in accordance with the `comb.txt` file. Before the script can run Bacchus, in order to do the Lanczos iterations, it generates the `bacchus.conf` file:

```
interaction:
{
    combination_table_file = "/absolute/path/to/JupiterNCSM/example
       /example_4he_data/anicr_runs/4He_nmax6.jtd/comb.txt";
    evaluation_order_file = "evaluation_order";
    index_lists_base_directory = "index_lists";
    matrix_file_base_directory = "m_scheme_interaction";
    num_neutrons = 2;
    num_protons = 2;
}
lanczos:
```

*Since Anicre is currently a free-standing project and needs input in form of a pAntoine wavefunction file, I have chosen to pre-compute the index lists for this example.

```
{
    krylow_vector_directory = "krylow_vectors";
    max_num_lanczos_iterations = 20;
    convergence_tolerance = 1.0e-7;
    target_eigenvector = 1;
    converge_eigenvectors = false;
    eigenvector_directory  = "eigen_vectors";
}
```

The `interaction:` section contains information about the nuclear forces and the target nucleus, in this case ^{4}He. Most of the fields point to folders that are generated in the working directory of the script. The `lanczos:` section contains the settings for the Lanczos algorithm. In this particular case, Bacchus will never make more than twenty iterations, but if the lowest eigenvalue changes with less than 10^{-7}MeV between two consecutive iterations, it will stop earlier. As indicated by setting `converge_eigenvectors` to `false`, Bacchus will use the *eigenvalue* convergence criteria, and as indicated by setting `target_eigenvector` to 1 it converges the lowest eigenvector eigenvalue pair. Lastly the script runs Bacchus and pipes its standard output to the file `bacchus_results` and the script standard output, using `tee`. For more details on each step I refer to the script.

Appendix B

The "comb.txt" file

The code Anicre constructs the `comb.txt` file, which contains important meta-data about all the other files. The `comb.txt` file starts with a section with the title `*** mp-states ***` listing the blocks that the full NCSM-basis is divided in to. Each row represents a block and takes the following format:

```
<Ep>  <Mp> <En>  <Mn>  <#mp-p>  <#mp-n>  <#mp>  # ARRAYMP:
    <array_id>=  <#bytes>
```

where `Ep` (`En`) and `Mp` (`Mn`) are the HO-energy and total azimuthal quantum number, respectively, of the proton (neutron) states, `#mp-p` (`#mp-n`) is the number of proton (neutron) states in the block, `#mp` is the total number of many-body states in the block, `array_id` is the identification number for the vector block file corresponding to the basis block, and `#bytes` is the file size in bytes needed to store the vector block as IEEE double precision floating-point numbers.

After the list of basis blocks, follows several sections corresponding to lists of different combinations of basis blocks, index lists and matrix-element files that need to be evaluated to do a full matrix-vector multiplication. This information is the needed input to Mars that determins the evaluation order. Each of these sections has a title with the format: `*** Connections <type> ***` where `type` indicates if it involves protons-only, (`p-dia`), neutrons only (`n-dia`), or protons and neutrons, (`p-n`), (`p-nn`) and (`pp-n`). The rows that involve both protons and neutrons have the format:

```
<Ep1> <Mp1> <En1> <Mn1> <Dp1> <Ep2> <Mp2> <En2> <Mn2> <Dn1>
    <#mp1> <#mp2> <#conn-p> <#conn-n> <conn>
    # CALCBLOCK:<force>: <bra_vector_block_id>
    <ket_vector_block_id> <index_list1_id>
    <index_list2_id> <matrix_block_id>
```

Here the `Ep1`, `Mp1`, `En1` and `Mn1` refer to which many-body basis block that is needed for the bra side, while `Ep2`, `Mp2`, `En2` and `Mn2` refer to the ket side. `Dp1` and `Dn1` refer to how much HO-energy is removed from the ket side to be replaced by new single-particle states. Moreover, `#mp1` and `#mp2` tell how many many-body states there are in the bra and ket blocks respectively, `#conn-p` and `#conn-n` tell how many transition densities there are for the the proton and neutron spaces, respectively, and `conn` is the total number of combined transition densities. Non of this information before

CALCBLOCK is acctually used by any program in JupiterNCSM, but is included for debuging and statistics reasons. The five fields at the end, however, are important for Mars. These tell which vector blocks, index lists and matrix-element blocks to be used for each matrix-vector multiplication block in Minerva. The <force> field tells if the block describes a 1NF, 2NF or 3NF. The proton-only and neutron-only sections are very similar, but have only one index-list id field instead of two. I will describe how these blocks are used in section 4.1.5.

The index lists are described in section *** Conn lists ***. Each index list is a row with the following format:

```
1 <type> <E1> <M1> <E2> <M2> <D1> <len> # ARRAY: <id>= <#bytes>
```

The first field, <type>, can take any of the following values: n, nn, nnn, p, pp or ppp. It indicates if the list represents one-, two-, or three- neutron, or proton transition densities. The subsequent four fields describe how much energy and total azimuthal angular momentum the bra and ket side of the transition involves. The D1 field indicates how much HO-energy in the ket should be replaced. The next field, len, tells how many transition densities that are included in the index list. The id of the index list, used in the CALCBLOCKS, is described by id. Finally, the size (in bytes), needed to store the index list is given by #bytes.

The next two sections represent the matrix-element blocks, and have titles of the format *** Matrix-elements V (<type>)***. Here, type indicates if the potential involves both protons and neutrons, (cross p-n), or just either proton or neutrons (same p/n). Each row represents a matrix-element block. If type is cross p-n each line follows the format:

```
1 <prt_p> <dEp> <dMp> <Dp> <prt_n> <dEn> <dMn> <Dn>
2     <#comb-p> <#comb-n> <#comb> # ARRAY: <id>= <#bytes>
```

Here prt_p (prt_n) refers to how many protons (neutrons) there are in the interaction, it can either take the value p (n) or pp (nn). dEp (dEn) is how much the HO-energy of the protons (neutrons) changes due to the interaction. dMp and dMn are the changes in the azimuthal quantum number of protons and neutrons, respectively. Since the total azimuthal quantum number is conserved, these two always sum to zero. id is the identification of the corresponding matrix-element file. If type is instread same p/n the format is instead:

```
1 <prt> <dEx> <dMx> <Dx> <#comb-x> <#comb> # ARRAY: <id>= <#bytes>
```

Here prt can take the values: p, pp, ppp, n, nn or nnn. The rest is analogous with the cross p-n case.

Anicre includes a few more lines in the comb.txt, but these are ignored by JupiterNCSM and will not be discussed.

Appendix C

Eigenvector continuation example

To test the performance of the EVC method, we have implemented a small numerical example using python. The matrix $H(c)$ is constructed as in Eq. (5.1), where the matrices H_0 and H_1 are both random positive-definite matrices plus a constant.

In Program 1 the python class, **RandomHamiltonian**, is defined to represent $H(c)$. The member function **create_subspace_hamiltonian** takes a list of μ parameter values for which it then generates the vectors $|\phi_i\rangle$ for $i = 1, \ldots, \mu$, one for each parameter value. It then generates the norm matrix $\hat{N}$, and the subspace matrices $\hat{M}_0$ and $\hat{M}_1$. These matrices are then used to produce an instance of the class **SubspaceHamiltonian** defined in program 3.

The class **SubspaceHamiltonian** represents the subspace-projected matrix. It has a method, **diagonalize**, that solves the generalized eigenvalue problem in Eq. (5.5). This equation is solved through the **scipy.linalg.eigh** function that uses QR factorization through linpack.

In Fig. C.1 the lowest eigenvalue of a 1000×1000 matrix has been approximated using the python EVC implementation discussed above. The blue stars are computed by exact diagonalization, i.e. by using the **diagonalize** method in **RandomHamiltonian**, while the green curve is obtained by diagonalizing the subspace projected matrix, i.e. by calling the **diagonalize** method in **SubspaceHamiltonian**. For this particular case there are five training vectors obtained by diagonalizing the full-space matrix. The corresponding parameter values are indicated by the five red stars.

As can be seen, the EVC approximation seems to capture the general behavior of the lowest eigenvalue of the full-space matrix. In fact, it reproduces the exact results (blue stars) with an average relative difference of $3.7 \cdot 10^{-5}$ excluding the exact result at $c = 0$ for which the relative difference is 1954. The smallest difference of $5.8 \cdot 10^{-12}$ is obtained for the parameter value 2000, which is one of the training points.

The most likely explanation for the huge error at $c = 0$ is that there are singularities close to the real axis between $c = 0$ and the (red) training points. To investigate this hypothesis further, EVC is applied to two random 10×10 matrices, $A(c)$ and $B(c)$, which can be found in appendix D. These matrices are chosen such that EVC works well for $A(c)$ using training points $c \in \{5, 5.25, 5.5, 5.75, 6\}$ while the method works less well for $B(c)$ using the same training points. EVC results for

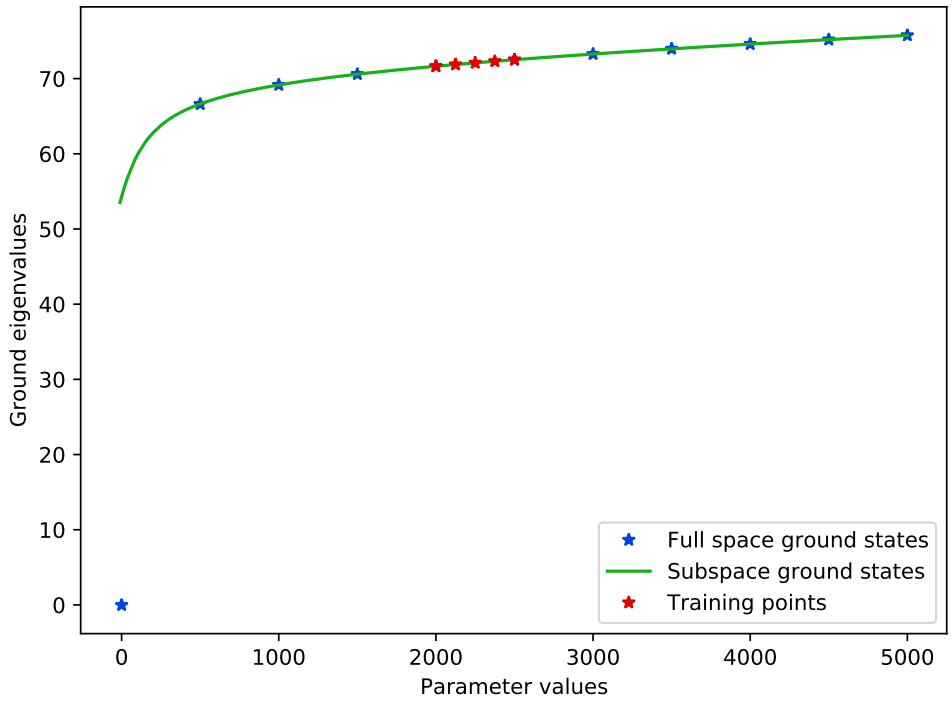

Figure C.1: The lowest eigenvalue as a function of the parameter c.of a random 1000×1000 matrix of the form in Eq. (5.1) compared with the EVC approximated lowest eigenvalue. The blue stars correspond to by direct diagonalization of the full matrix, $H(c)$, the green curve is obtained by solving the generalized eigenvalue problem in Eq. (5.5) and the red stars indicate the training parameter values, i.e. for which parameter values the vectors $|\phi_{1\cdots\mu}\rangle$ are computed.

these two matrices using the aforementioned training points are shown in the upper row in figure C.2.

To study the singularities of the eigenvalue problem, we look at where the Riemann surface of the lowest eigenvalue touches the surface of the second lowest eigenvalue. At these locations branch points can occur. In the middle row, the difference between the two Riemann surfaces for the two matrices can be viewed. The colormap is logarithmic and goes from 10^{-5} (dark orange), to 1 and above (yellow).

For both matrices the two lowest eigenvalues approach each other at points on either side of the real axis for $\mathrm{Re}\{c\} < 1$. However, $B(c)$ displays more pronounced features closer to the real axis at about $\mathrm{Re}\{c\} \approx 0.7$, while the eigenvalues of $A(c)$ only get closer than 1 in a small area for $\mathrm{Re}\{c\} \approx 0.5$ and $|\,\mathrm{Im}\{c\}| > 1$.

The features of matrix $B(c)$ corresponds to actual branch points as discussed in section 5.2, while for $A(c)$, the two eigenvalues touch, but never swap order. For both $A(c)$ and $B(c)$ the training points are all larger than one and thus, are all on the same side of the singularities. This could explain why the method fails for $c < 0.5$ for $A(c)$ and $c < 0.7$ for $B(c)$. To capture the true parameter dependence for $c < 1$ we move the training point at 5 to 0.5 and redo EVC for the new training points for both matrices. The results of the new training can be viewed in the bottom row

of Fig. C.2. As can be seen the subspace approximated eigenvalue, the blue curve, matches the exact eigenvalue, the orange curve.

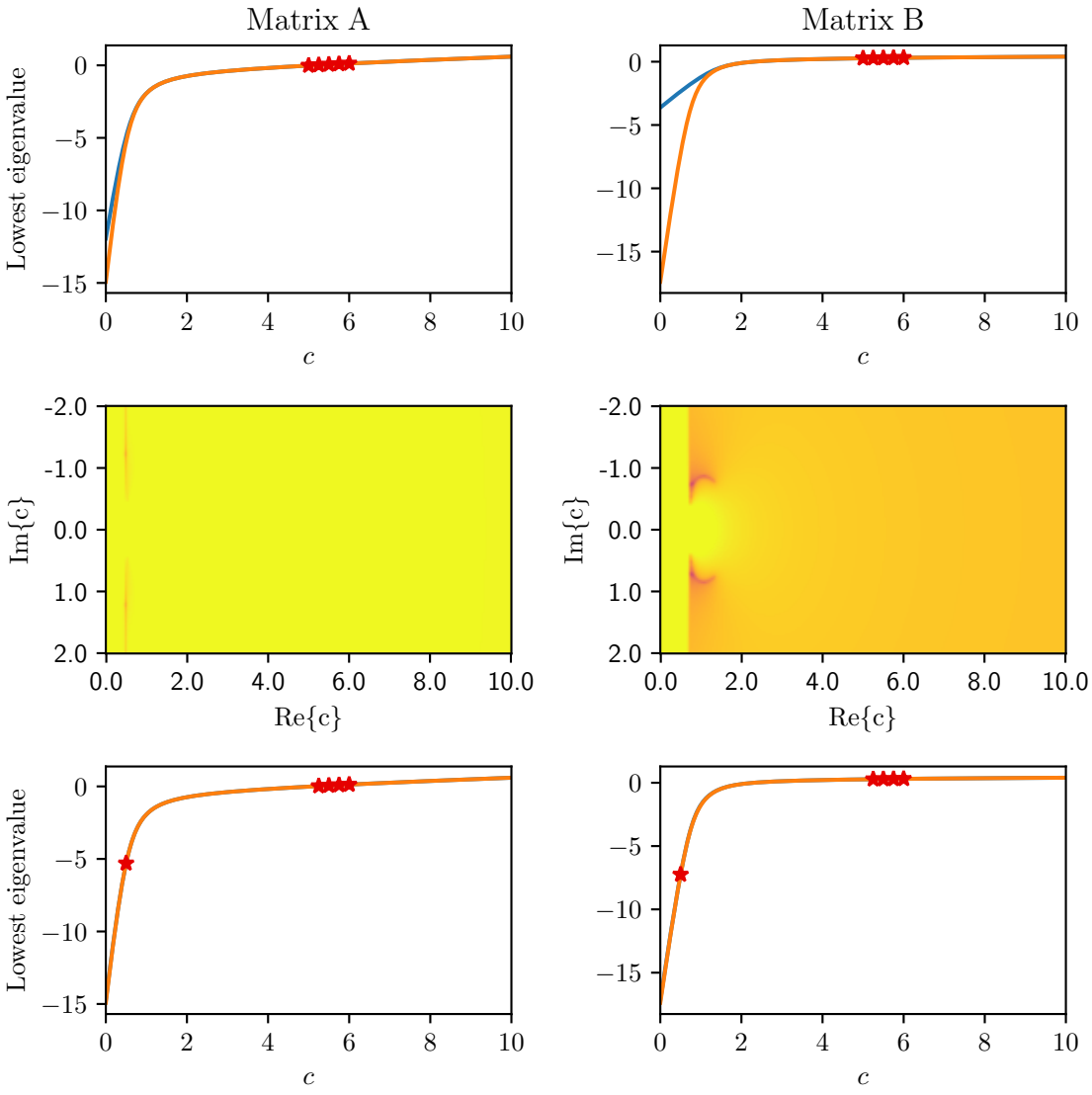

Figure C.2: Upper row: EVC results for two random 10×10 matrices, $A(c)$ and $B(c)$, where the red stars indicate training points, the orange curve is the true lowest eigenvalue and the blue curve is the approximated one. Middle row: The absolute difference between the lowest and second lowest eigenvalue as function of complex parameter values. The color mapping is logarithmic, where yellow indicate larger than one and the darkest orange is 10^{-5}. Bottom row: EVC results for $A(c)$ and $B(c)$ with the leftmost training point moved to the left of the singular region.

Based on the obtained knowledge from the study of the matrices $A(c)$ and $B(c)$ above, we can now improve the EVC approximation of the 1000×1000 matrix used in Fig. C.1. Between the exact eigenvalue points, the blue stars, located at $c = 0$ and $c = 500$ the curve changes dramatically. It is therefore possible that there are some singularities close to the real axis in this region. By moving the training point

at $c = 2000$ to $c = 5$ it is possible to capture the behavior of the eigenvalue for small c. The result of this calculation can be viewed in figure C.3. As can be seen the eigenvalue of the subspace-projected operator for $c = 0$ is very close to the exact value.

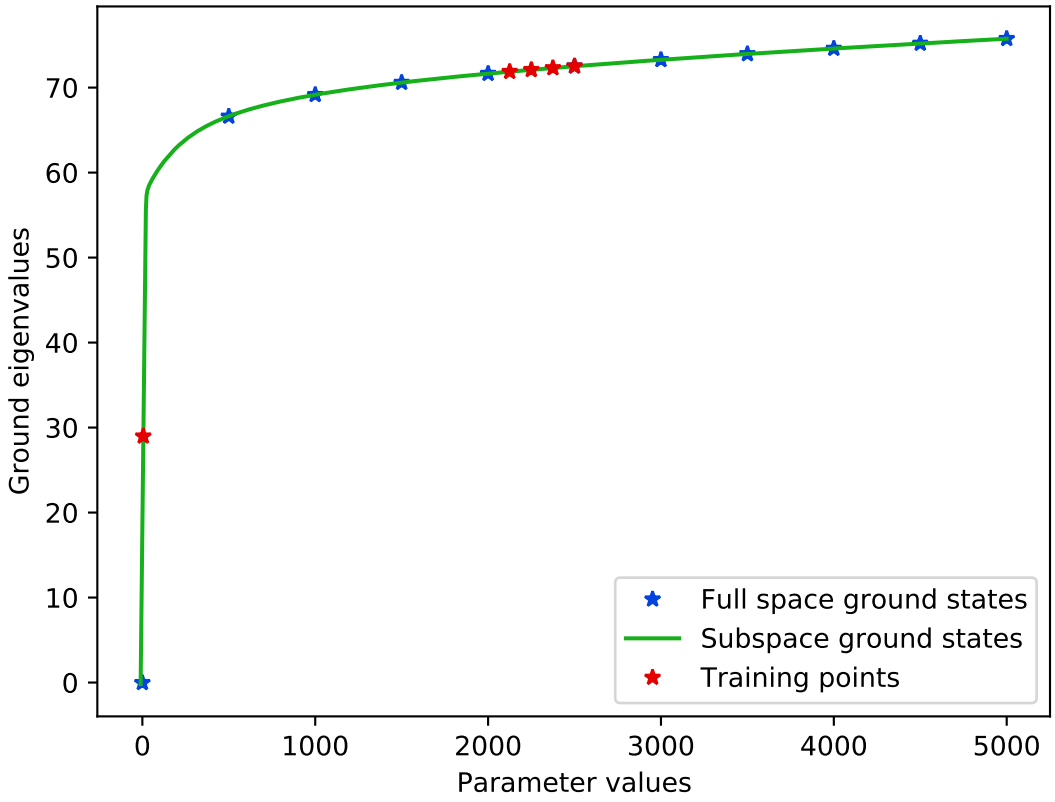

Figure C.3: EVC applied to the same 1000×1000 matrix as in Fig. C.1, the difference is that the left most training point has been moved from 2000 to 5.

Program 1 A class to represent a parameterized random Hamiltonian.

```
import numpy as np
class RandomHamiltonian:
        def __init__(self, side = 1000):
                self.side = side
                self.H0 =
                  random_positive_definite_matrix(side) - 4
                self.H1 =
                  random_positive_definite_matrix(side) - 0.1
                self.training_points = None
        def __call__(self,parameter):
                return self.H0 + parameter*self.H1

        def create_subspace_hamiltonian(self,
                                        training_param_values,
                                        target_vector):
                sub_space_dimension = len(training_param_values)
                training_vectors = np.zeros((self.side,
                   sub_space_dimension))
                self.training_points = np.zeros((
                   sub_space_dimension,2))
                # Generating the training vectors
                for i,parameter_value in enumerate(
                   training_param_values):
                        print(f'Generating training vector {i} for
                           paramter value {parameter_value}')
                        eigen_values,eigen_vectors = np.linalg.eigh
                           (self(parameter_value))
                        training_vectors[:,i] = eigen_vectors[:,
                           target_vector]
                        self.training_points[i,0] = parameter_value
                        self.training_points[i,1] = eigen_values[
                           target_vector]
                # Projecting H0 and H1 on the subspace
                M0 = np.dot(np.transpose(training_vectors),
                            np.dot(self.H0,training_vectors))
                M1 = np.dot(np.transpose(training_vectors),
                            np.dot(self.H1,training_vectors))
                # Creating the norm matrix
                N = np.dot(np.transpose(training_vectors),
                           training_vectors)
                return SubspaceHamiltonian(M0,M1,N,
                                   training_vectors,
                                   training_param_values)
        def diagonalize(self,parameter = 0):
                return np.linalg.eigh(self(parameter))
        def get_training_points(self):
                return self.training_points
```

Program 2 This function generates a random positive definite matrix needed by RandomHamiltonian in program 1.

```
def random_positive_definite_matrix(side):
        m = np.random.rand(side,side)
        h = np.dot(m,np.transpose(m))
        return h
```

Program 3 A class to represent a parameterized subspace Hamiltonian

```
import numpy as np
import scipy.linalg as sp
class SubspaceHamiltonian:
        def __init__(self,M0,M1,N,training_vectors,
                     training_param_values):
                self.M0 = M0
                self.M1 = M1
                self.N = N
                self.training_param_values = training_param_values
                self.training_vectors = training_vectors
        def __call__(self,parameter):
                return self.M0 + parameter*self.M1
        def diagonalize(self,parameter = 0):
                # Solving the genralized eigenvalue problem
                # for the subspace Hamiltonian
                return sp.eigh(self(parameter),
                               self.N,
                               eigvals_only=False)
```

Appendix D

Example matrices

$$A(c) = \begin{pmatrix} -0.48 & -1.05 & -2.32 & -0.99 & -1.37 & -2.58 & -1.27 & -1.9 & -1.0 & -1.9 \\ -1.05 & -0.51 & -2.2 & -0.75 & -1.27 & -2.18 & -1.46 & -1.4 & -0.75 & -0.8 \\ -2.32 & -2.2 & -1.33 & -0.98 & -2.44 & -2.98 & -1.65 & -1.65 & -1.63 & -1.17 \\ -0.99 & -0.75 & -0.98 & 0.6 & -0.86 & -2.4 & -0.75 & -0.51 & -0.22 & 0.14 \\ -1.37 & -1.27 & -2.44 & -0.86 & -0.96 & -2.46 & -1.36 & -1.59 & -1.05 & -1.14 \\ -2.58 & -2.18 & -2.98 & -2.4 & -2.46 & -2.46 & -2.31 & -2.66 & -1.95 & -2.22 \\ -1.27 & -1.46 & -1.65 & -0.75 & -1.36 & -2.31 & -0.65 & -1.66 & -0.9 & -1.06 \\ -1.9 & -1.4 & -1.65 & -0.51 & -1.59 & -2.66 & -1.66 & -0.68 & -0.93 & -0.84 \\ -1.0 & -0.75 & -1.63 & -0.22 & -1.05 & -1.95 & -0.9 & -0.93 & 0.21 & -0.26 \\ -1.9 & -0.8 & -1.17 & 0.14 & -1.14 & -2.22 & -1.06 & -0.84 & -0.26 & 1.26 \end{pmatrix}$$

$$+c \begin{pmatrix} 2.79 & 2.02 & 1.79 & 1.3 & 2.69 & 2.64 & 1.97 & 2.3 & 2.96 & 3.06 \\ 2.02 & 2.62 & 2.11 & 0.95 & 2.6 & 1.99 & 2.1 & 2.43 & 2.49 & 2.98 \\ 1.79 & 2.11 & 2.52 & 1.03 & 2.43 & 1.9 & 1.89 & 2.51 & 2.71 & 2.67 \\ 1.3 & 0.95 & 1.03 & 1.07 & 1.64 & 1.52 & 0.92 & 1.38 & 1.7 & 1.9 \\ 2.69 & 2.6 & 2.43 & 1.64 & 3.87 & 2.75 & 2.44 & 2.97 & 3.46 & 3.55 \\ 2.64 & 1.99 & 1.9 & 1.52 & 2.75 & 3.18 & 2.52 & 2.75 & 3.56 & 3.45 \\ 1.97 & 2.1 & 1.89 & 0.92 & 2.44 & 2.52 & 3.24 & 3.23 & 3.3 & 2.84 \\ 2.3 & 2.43 & 2.51 & 1.38 & 2.97 & 2.75 & 3.23 & 4.2 & 3.43 & 3.53 \\ 2.96 & 2.49 & 2.71 & 1.7 & 3.46 & 3.56 & 3.3 & 3.43 & 4.87 & 4.06 \\ 3.06 & 2.98 & 2.67 & 1.9 & 3.55 & 3.45 & 2.84 & 3.53 & 4.06 & 4.74 \end{pmatrix}$$

$$B(c) = \begin{pmatrix} -0.66 & -1.27 & -2.01 & -1.08 & -1.0 & -2.46 & -2.1 & -2.35 & -1.74 & -0.37 \\ -1.27 & -0.29 & -1.31 & -0.97 & -1.14 & -1.8 & -1.9 & -1.9 & -1.66 & -0.24 \\ -2.01 & -1.31 & -1.34 & -1.2 & -2.07 & -2.22 & -2.22 & -2.03 & -2.19 & -1.28 \\ -1.08 & -0.97 & -1.2 & -0.17 & -1.3 & -1.91 & -2.1 & -1.92 & -1.3 & -0.84 \\ -1.0 & -1.14 & -2.07 & -1.3 & -0.41 & -1.88 & -2.13 & -2.22 & -1.71 & -0.17 \\ -2.46 & -1.8 & -2.22 & -1.91 & -1.88 & -1.8 & -2.82 & -2.3 & -2.08 & -1.71 \\ -2.1 & -1.9 & -2.22 & -2.1 & -2.13 & -2.82 & -1.95 & -2.74 & -2.86 & -1.47 \\ -2.35 & -1.9 & -2.03 & -1.92 & -2.22 & -2.3 & -2.74 & -2.12 & -2.19 & -1.67 \\ -1.74 & -1.66 & -2.19 & -1.3 & -1.71 & -2.08 & -2.86 & -2.19 & -1.45 & -1.45 \\ -0.37 & -0.24 & -1.28 & -0.84 & -0.17 & -1.71 & -1.47 & -1.67 & -1.45 & 0.99 \end{pmatrix}$$

$$+c \begin{pmatrix} 1.57 & 1.91 & 1.45 & 1.58 & 1.46 & 1.45 & 1.25 & 1.44 & 1.8 & 0.74 \\ 1.91 & 4.35 & 2.15 & 3.28 & 2.92 & 3.29 & 3.1 & 3.07 & 3.52 & 2.43 \\ 1.45 & 2.15 & 2.56 & 1.9 & 1.48 & 2.13 & 1.7 & 2.01 & 2.71 & 1.04 \\ 1.58 & 3.28 & 1.9 & 2.92 & 2.04 & 2.81 & 1.98 & 2.29 & 2.72 & 1.47 \\ 1.46 & 2.92 & 1.48 & 2.04 & 2.39 & 2.23 & 2.45 & 2.11 & 2.25 & 1.8 \\ 1.45 & 3.29 & 2.13 & 2.81 & 2.23 & 3.87 & 2.46 & 2.49 & 2.81 & 1.74 \\ 1.25 & 3.1 & 1.7 & 1.98 & 2.45 & 2.46 & 3.1 & 2.6 & 2.68 & 2.2 \\ 1.44 & 3.07 & 2.01 & 2.29 & 2.11 & 2.49 & 2.6 & 2.61 & 3.01 & 1.78 \\ 1.8 & 3.52 & 2.71 & 2.72 & 2.25 & 2.81 & 2.68 & 3.01 & 3.9 & 1.87 \\ 0.74 & 2.43 & 1.04 & 1.47 & 1.8 & 1.74 & 2.2 & 1.78 & 1.87 & 1.81 \end{pmatrix}$$