ADDITION AND REMOVAL ENERGIES VIA THE IN-MEDIUM SIMILARITY RENORMALIZATION GROUP METHOD

By

Fei Yuan

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Physics—Doctor of Philosophy

2018

ABSTRACT

ADDITION AND REMOVAL ENERGIES VIA THE IN-MEDIUM SIMILARITY RENORMALIZATION GROUP METHOD

By

Fei Yuan

The in-medium similarity renormalization group (IM-SRG) is an *ab initio* many-body method suitable for systems with moderate numbers of particles due to its polynomial scaling in computational cost. The formalism is highly flexible and admits a variety of modifications that extend its utility beyond the original goal of computing ground state energies of closed-shell systems.

In this work, we present an extension of IM-SRG through quasidegenerate perturbation theory (QDPT) to compute addition and removal energies (single particle energies) near the Fermi level at low computational cost. This expands the range of systems that can be studied from closed-shell ones to nearby systems that differ by one particle. The method is applied to circular quantum dot systems and nuclei, and compared against other methods including equations-of-motion (EOM) IM-SRG and EOM coupled-cluster (CC) theory. The results are in good agreement for most cases.

As part of this work, we present an open-source implementation of our flexible and easy-to-use J-scheme framework as well as the HF, IM-SRG, and QDPT codes built upon this framework. We include an overview of the overall structure, the implementation details, and strategies for maintaining high code quality and efficiency.

Lastly, we also present a graphical application for manipulation of angular momentum coupling coefficients through a diagrammatic notation for angular momenta (Jucys diagrams). The tool enables rapid derivations of equations involving angular momentum coupling – such as in J-scheme – and significantly reduces the risk of human errors.

In memory of Shichao Yuan (1964-2016) and Erik "Kexie" Gustafsson (1992-2017)

ACKNOWLEDGMENTS

I am forever indebted to my parents, who have made this journey at all possible. My interest in software was strongly inspired by my father's projects, whereas my foundations in mathematics would not have been as solid without the tutoring from my mother. Throughout my studies, they have gone above and beyond to support my education, in spite of the limited resources our family had. I also extend my thanks to my grandparents who cared for me during my youngest years.

On the academic side, I would like to thank my thesis advisor Morten Hjorth-Jensen for his knowledge, wisdom, and, most importantly, support. He had helped reignite my interest in physics by offering a pathway from experimental to computational physics. He encouraged me to research in areas that are most relevant to my interests, allowing me to learn and explore far more than I would have otherwise. I also wish to thank my co-advisor Scott Bogner, who has been very patient and offered valuable assistance and technical discussions. I thank the rest of my committee – Alexandra Gade, Carlo Piermarrochi, and Scott Pratt – for their helpful inputs, as well as my former advisor Chong-Yu Ruan who guided me during both undergraduate research and my first two years of graduate research.

I thank my colleagues for our productive discussions, including Heiko Hergert, Titus Morris, Justin Lietz, as well as Nathan Parzuchowski and Sam Novario, who have contributed substantially to the results of this work. On a personal level, I wish to thank Nathan, my hard-working officemate who is well-versed in the intricacies of IM-SRG and Fortran and has a great sense of humor, Titus, whose mastery in many-body theory, quantum chemistry, and social situations is unmatched, Vinzent Steinberg, with whom many fun discussions and debates in programming have been made, and Tzong-Ru Terry Han, who has been very kind and supportive during my years in the laboratory. I would also like to thank Debbie Barratt and Kim Crosslan for their support

throughout my long stay at Michigan State University.

Lastly, I wish to thank all the friends I have made along the way for their support and encouragement, including my feline companion TipToe who has always had to put up with my chaotic schedule.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

KEY TO SYMBOLS

- $\boldsymbol{x}$ (bold) — matrix or vector
- $\hat{x}$ (hat) — quantum operator
- x^* (superscript asterisk) — complex conjugation
- $x^\dagger$ (superscript dagger) — Hermitian adjoint
- $\mathbf{1}$ — identity matrix
- $\hat{1}$ — identity operator
- $\mathbb{N}$ — nonnegative integers
- $\mathbb{Z}$ — integers
- $\mathbb{H}$ — Hilbert space of a quantum system
- $\mathbb{F}^\pm$ — $\pm$-symmetric Fock space (Sec. 2.2)
- δ_{pq} — Kronecker delta
- ϵ_{ijk} — Levi–Civita symbol
- $\Gamma(x)$ — gamma function
- $L_n^\alpha(x)$ — associated Laguerre polynomial (Eq. 5.3)
- $\bar{L}_n^\alpha(x)$ – normalized associated Laguerre polynomial (Eq. 5.3)
- $Y_{\ell m}(\theta, \varphi)$ — spherical harmonic with Condon–Shortley phase [DLMF]
- $(-)^i$ — shorthand for $(-1)^i$
- $(-)^\sigma$ — sign of a permutation σ
- $\lfloor x \rfloor$ — floor of x
- $[\hat{x}, \hat{y}]_- = [\hat{x}, \hat{y}]$ — commutator
- $[\hat{x}, \hat{y}]_+ = \{\hat{x}, \hat{y}\}$ — anticommutator
- $:x:$ — normal ordering relative to Fermi vaccuum (Sec. 2.5)

- $\vdots x \vdots$ — normal ordering relative to physical vacuum (Sec. 2.5)
- $\mathbb{V} \otimes \mathbb{W}$ — tensor product of vector spaces
- $|a\rangle \otimes |b\rangle$ — tensor product constructor of kets
- $|p \otimes q\rangle$ — tensor product state (Sec. 2.1.1)
- $|pq\rangle^{+}$ — symmetrized state (Sec. 2.1.2)
- $|pq\rangle^{-} = |pq\rangle$ — antisymmetrized state (Sec. 2.1.2)
- $|\varnothing\rangle$ — physical vacuum state (Sec. 2.1.1)
- $|\Phi\rangle$ — reference state (Sec. 2.4)
- $\hat{a}_p$ — physical annihilation operator (Sec. 2.2)
- $\hat{b}_p$ — quasiparticle annihilation operator (Sec. 2.4)
- $\mathcal{S}^{\pm}$, $\mathcal{S}^{(i)}$, $\mathcal{S}$, $\mathcal{A}$ — symmetrization/antisymmetrization symbols (Sec. 2.1.2)
- $\sum_{ij\backslash ab}$ — summation over holes and particles (Eq. 2.3)
- $\prod_{k=m}^{\rightarrow n}$ — ordered product from left ($k = m$) to right ($k = n$)
- $M_j = \{-j, -j+1, \dots, j-1, j\}$ — projection quantum numbers of a multiplet (Eq. 3.4)
- $\check{j}_a = \sqrt{2j_a + 1}$ — (Eq. 3.8)
- $\check{a}$ — time-reversed angular momentum (Eq. 3.15)
- $\begin{pmatrix} & j & \\ m & & m' \end{pmatrix}$ — Herring–Wigner 1-jm symbol (Eq. 3.16)
- $\langle a \| \hat{Q} \| b \rangle$ — reduced matrix element (Sec. 3.6)
- $\langle a, b | ab \rangle = \langle j_a m_a j_b m_b | j_{ab} m_{ab} \rangle$ — Clebsch–Gordan coefficient (Sec. 3.2)
- $(abc) = \begin{pmatrix} j_a & j_b & j_c \\ m_a & m_b & m_c \end{pmatrix}$ — Wigner 3-jm symbol (Sec. 3.3)
- $\begin{Bmatrix} j_a & j_b & j_c \end{Bmatrix}$ — triangular delta (Sec. 3.8.1)
- $\begin{Bmatrix} j_a & j_b & j_c \\ j_d & j_e & j_f \end{Bmatrix}$ — Wigner 6-j symbol (Sec. 3.8.2)

- $\begin{Bmatrix} j_a & j_b & j_c \\ j_d & j_e & j_f \\ j_g & j_h & j_i \end{Bmatrix}$ — Wigner 9-j symbol (Sec. 3.8.3)

Chapter 1

Introduction

Quantum many-body theory is a broad discipline concerned with the behavior of quantum particles in large numbers. It is of critical relevance in many fields ranging from nuclear physics, through quantum chemistry, to condensed matter theory.

The fundamental challenge of quantum many-body theory lies in the difficulty of obtaining accurate results for fermionic systems in an efficient manner, owing to the combinatoric growth of the many-body Hilbert space as the number of particles increases. Certain methods such as full configuration interaction (FCI) theory [SB09] can achieve arbitrarily accurate results, but their cost scales factorially with respect to the basis size and the number of particles, rendering them infeasible for all but the smallest systems. In contrast, methods that scale polynomially must necessarily make certain approximations. This has led to a menagerie of many-body methods that trade varying amounts of accuracy for computational efficiency.

Nuclear science is an area where many-body theory has made substantial breakthroughs over the recent decades. For a long time, theories for nuclear physics have been largely limited to phenomenological methods such as density-functional theories and the nuclear shell model [BW88], which often have limited predictive power beyond the domain in which the parameters were fit.

With advances in experimental technology such as the Facility for Rare Isotope Beams (FRIB) [Tho10], there is a growing demand for more accurate nuclear calculations and wider coverage of the nuclear chart, both of closed-shell and open-shell nuclei, and of stable and exotic ones.

The needs of nuclear astrophysics should also not be understated: accurate models of nuclear systems are critical to the understanding of dense stars and supernovae. Many-body theory is also needed to calculate the rates of processes such as the neutrinoless double-beta decay, which has fundamental relevance in particle physics: whether neutrinos are Majorana fermions or not. Hence, the development of many-body theory is an essential step toward these many goals.

The recent introduction of chiral effective-field theory (EFT) [ME11] and methods based on renormalization group (RG) theory [Wil83; Wei79; Lep05] have dramatically changed the landscape of nuclear theory. Although the derivation of nuclear interactions from quantum chromodynamics (QCD) has not yet been achieved, there has been great progress through lattice QCD approaches [Hoe14; Uka15; IAH07; Ish+12; Iri+16]. At the moment, chiral EFT offers an intermediate, practical solution in which the symmetries of QCD are used to construct an ansatz of the nuclear interaction in a highly systematic way [EHM09]. The development of RG theory and methods have allowed the creation of *soft* interactions that are unitarily equivalent to the original [BFP07]. Such interactions converge much more rapidly with respect to basis size, thereby reducing the cost of computations. The increasing availability of computing power has also played a role in amplifying the progress of nuclear many-body theory.

The fruits of this progress can be seen in Fig. 1.1. Just a decade ago, only a handful of nuclei near or below oxygen-16 could be computed by *ab initio* methods, as shown by the blue squares in the upper chart. The idea of calculating a nucleus as heavy as tin-100 through an *ab initio* method would have been considered absurd.

One particular many-body theory, the so-called in-medium similarity renormalization group method (IM-SRG) [Her+16], has gained significant attention in nuclear theory of late. IM-SRG fuses the flow equation approach of similarity renormalization group (SRG) with the particle-hole operator formalism to reduce computational cost, providing a novel *ab initio* approach for solving

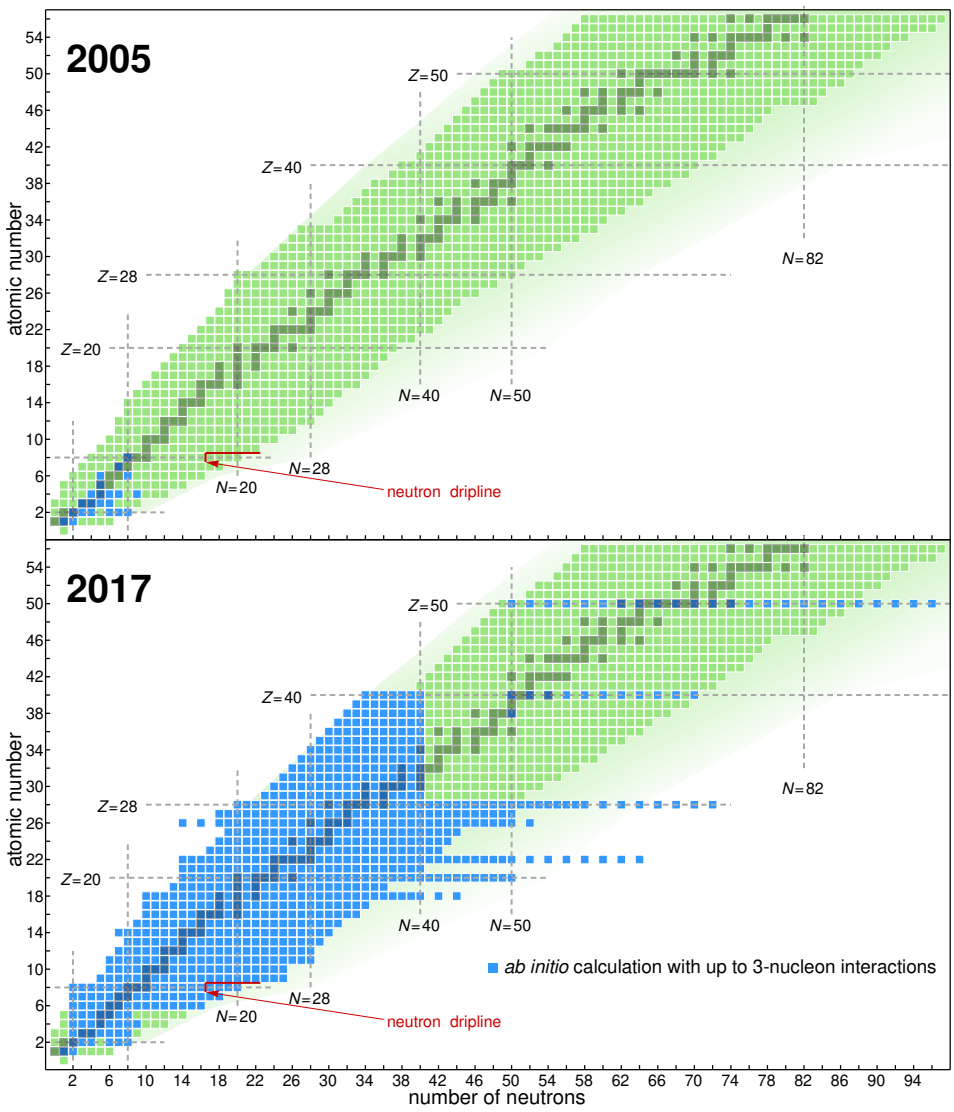

Figure 1.1: Nuclear chart showing the current progress of *ab initio* nuclear structure. Image courtesy of Heiko Hergert [Her+16].

the many-body problem. Over the years, IM-SRG has proven to be a highly flexible and adaptable method. It offers efficient evaluation of observables beyond energy [MPB15; Mor16], the ability to tackle excited states [PMB17; Par17], as well as extensions to open-shell nuclei [Her17].

Unlike coupled-cluster (CC) theory, IM-SRG theory is naturally Hermitian, making it straightforward to utilize its matrix elements as an effective operator for other methods such as the nuclear shell model [Bog+14; Str+16; Str+17]. The renormalizing nature of IM-SRG eliminates many of the couplings between components of the many-body operator, simplifying post-IM-SRG calculations.

In our work, we take advantage of the softening property of IM-SRG to compute single-particle energies (addition and removal energies) via quasidegenerate perturbation theory (QDPT) to third order, also known as open-shell perturbation theory. Our expectation is that the use of IM-SRG ought to improve the overall quality and convergence of the perturbative results.

Compared to more sophisticated approaches such as the equations-of-motions method (EOM), QDPT at third order (QDPT3) is remarkably inexpensive. The ability to cheaply solve systems that are one particle away from a closed shell system can be remarkably useful in practice. Not only does it expand the scope of applicability of closed-shell IM-SRG, it can even permit access to excited states under certain circumstances.

1.1 Contributions

Our main contributions in this work are:

- We have created a graphical tool for performing equality-preserving transformations of angular momentum diagrams [Jucys]. The diagrammatic formalism we use extends the work of [YLV62].
- We have developed an open-source J-scheme codebase with an easy-to-use, flexible, and extensible framework for many-body calculations [Lutario]. With this framework, we have implemented the Hartree–Fock (HF) method, Møller–Plesset perturbation theory at second order (MP2), IM-SRG method with two-body operators (IM-SRG(2)), and QDPT3. Our program supports several quantum systems, including circular quantum dots, homogeneous electron gas, infinite matter, and nuclei.
- We have performed calculations of the quantum dot ground state and single-particle energies using HF, IM-SRG(2), and QDPT3 [Yua+17], benchmarked against similar calculations using EOM and CCSD. The results have been analyzed and extrapolated to the infinite basis limit.

- We have performed calculations of nuclear ground state and single-particle energies using HF, IM-SRG(2), and QDPT3, benchmarked against similar calculations using EOM and CCSD. We discuss the results and some preliminary analysis in this work.

1.2 Outline

The remainder of this thesis is structured as follows:

- We begin with a review of the many-body formalism in chapter 2. The primary purpose of this section is to establish the background theory, terminology, and notational conventions used in this work, as the field of many-body theory tends to be plagued by differences in nomenclature and notation.
- In chapter 3, we discuss the details of angular momentum coupling. This forms a critical part of our J-scheme machinery, needed for efficient nuclear calculations. We also discuss angular momentum diagrams, which are an effective tool for manipulation of angular momentum expressions, as well as the `jucys` software that we have developed to aid simplification of such diagrams.
- In chapter 4, we discuss each of the three major many-body methods that we use in our thesis: Hartree–Fock, IM-SRG, and QDPT. We explain and show all the critical equations that are needed to implement them.
- In chapter 5, we discuss the theoretical background for the main quantum systems that we have chosen to study and analyze: circular quantum dots and nuclei.
- In chapter 6, we provide an overview of our concrete implementation of many-body methods and the quantum systems. We offer explanations, rationale, and discussion of various choices that we have made throughout the evolution of the project.
- In the penultimate chapter 7, we discuss the numerical results obtained from our codes, and

compare them with results of collaborators. We perform analysis and also extrapolation of our results.

- Finally, we conclude in chapter 8 with a review of our main results and perspectives for the future.

An online version of this thesis is available[1] along with any fixes to errors discovered after publication. Issues may be reported through the website. The version of this document is `r44-g58051a3`.

[1] URL: https://github.com/xrf/thesis

Chapter 2

Many-body formalism

In this section we review the fundamentals of many-body theory. While we have aimed to make the presentation fairly pedagogical, the primary goal of this chapter is to define the concepts, terminology, notation, and conventions used throughout this work.

2.1 Many-particle states

In single-particle time-independent quantum mechanics, the Schrödinger equation takes the following form in Dirac notation,

$$\hat{h}|\psi\rangle = \varepsilon|\psi\rangle$$

where

- $\hat{h}$ is the single-particle Hamiltonian operator,
- $|\psi\rangle$ is a state vector, and
- ε is the energy of the state.

The state $|\psi\rangle$ is an abstract ket vector that lives in the Hilbert space $\mathbb{H}$ of the single-particle system. More concretely, we can also represent the state vector by a wave function ψ

$$|\psi\rangle \leftrightarrow \psi(x)$$

where x stands for all degrees of freedom of the particle. For example, an electron with spin in three-dimensional space would have $x = (r_1, r_2, r_3, m_s)$, where (r_1, r_2, r_3) are the three spatial coordinates and m_s is the spin projection quantum number.

To treat systems of multiple particles, one may add additional variables to the wave function to represent the degrees of freedom of the additional particles, that is,

$$|\Psi\rangle \leftrightarrow \Psi(x_1, \dots, x_N)$$

where

- $|\Psi\rangle$ is an N-particle state,
- Ψ is its corresponding wave function,
- N is the number of particles, and
- x_α represents the degrees of freedom of the α-th particle.

With more than one particle, the Schrödinger equation remains conceptually the same,

$$\hat{H}|\Psi\rangle = E|\Psi\rangle$$

but there are many more degrees of freedom and thus more variables. Here, $\hat{H}$ is the N-particle Hamiltonian operator and E is the energy of the N-particle state.

2.1.1 Product states

The simplest multi-particle system that can be solved is that of a non-interacting Hamiltonian of N homogeneous particles,

$$\hat{H}_N^\circ = \sum_{\alpha=1}^{N} \hat{h}(\hat{x}_\alpha, \hat{k}_\alpha)$$

where $\hat{h}(\hat{x}_\alpha, \hat{k}_\alpha)$ is some single-particle Hamiltonian with position $\hat{x}_\alpha$ and momentum[1] $\hat{k}_\alpha = -\mathrm{i}\hat{\nabla}_\alpha$ of the α-th particle. We assume its single-particle Schrödinger equation has a known set of solutions,

$$\hat{h}(\hat{x}, \hat{k})|p\rangle = \varepsilon_p|p\rangle$$

These solutions form a set of **single-particle basis states** for the single-particle Hilbert space $\mathbb{H}$,

$$|p\rangle \leftrightarrow \varphi_p(x)$$

each with energy ε_p and labeled by the quantum numbers p. For example, if $\hat{h}$ is a single-electron atomic system, then we can choose $p = (n, l, m_\ell, m_\mathrm{s})$, which are the four conventional quantum numbers for atomic orbitals. The single-particle basis states $|p\rangle$ would simply be the standard atomic orbital states.

From the single-particle basis, we can define a set of N-particle wave functions

$$|p_1 \otimes \cdots \otimes p_N\rangle \leftrightarrow \Phi_{p_1 \otimes \cdots \otimes p_N}(x_1, \ldots, x_N)$$

[1] We use $\hat{k}$ for momentum to avoid confusion with p, which will be used to denote the quantum numbers of the single-particle basis later on.

via the tensor product construction[2]

$$|p_1 \otimes p_2\rangle = |p_1\rangle \otimes |p_2\rangle$$

$$|p_1 \otimes p_2 \otimes p_3\rangle = |p_1\rangle \otimes |p_2\rangle \otimes |p_3\rangle$$

$$|p_1 \otimes \cdots \otimes p_N\rangle = \bigotimes_{\alpha=1}^{N} |p_\alpha\rangle$$

In terms of wave functions, this is equivalent to the definition

$$\Phi_{p_1 \otimes p_2}(x_1, x_2) = \varphi_{p_1}(x_1)\varphi_{p_2}(x_2)$$

$$\Phi_{p_1 \otimes p_2 \otimes p_3}(x_1, x_2, x_3) = \varphi_{p_1}(x_1)\varphi_{p_2}(x_2)\varphi_{p_3}(x_3)$$

$$\Phi_{p_1 \otimes \cdots \otimes p_N}(x_1, \ldots, x_N) = \prod_{\alpha=1}^{N} \varphi_{p_\alpha}(x_\alpha)$$

These **product states** (also known as **Hartree products**) are eigenstates of the many-body Schrödinger equation and form an orthonormal basis for the N-particle Hilbert space $\mathbb{H}^N$ of the non-interacting N-particle Hamiltonian $\hat{H}_N^\circ$.

Formally, $\mathbb{H}^N$ is defined as the N-th tensor power of the single-particle vector spaces[3]

$$\mathbb{H}^N = \bigotimes_{\alpha=1}^{N} \mathbb{H}$$

Each product state is labeled by the tuple $(p_1, \ldots, p_N)$ and has the following energy in this non-interacting system:

$$E^\circ_{p_1 \otimes \ldots \otimes p_N} = \sum_{\alpha=1}^{N} \varepsilon_{p_\alpha}$$

[2] A ket with $\otimes$ inside denotes a product state. Outside kets, $\otimes$ denotes the *tensor product constructor*, a multilinear operation that constructs maps multiple vectors to a vector in their tensor product space.

[3] To further add to the confusion, this $\otimes$ symbol denotes the *tensor product* of vector spaces.

If we consider the special case $N = 0$, a system with no particles, we find that the tensor product Hilbert space $\mathbb{H}^0$ has precisely one basis state,[4] which we call the **(physical) vacuum state** $|\varnothing\rangle$.

Although the product states form a basis, they span both states in which particles are distinguishable as well as states in which particles are not distinguishable. Thus, unconstrained use of product states would violate the key principle of quantum mechanics that particles are indistinguishable.

Mathematically, indistinguishability means that, under particle exchange, the wave function may only differ by a phase factor s. Consider, say, the exchange of a two-particle wave function Ψ,

$$\Psi(x_2, x_1) = \pm\Psi(x_1, x_2)$$

- For bosons, the phase factor $s = +1$. This means the state is symmetric under particle exchange.
- For fermions, the phase factor $s = -1$. This means the state is antisymmetric under exchange.

To extract the relevant states, we need to either symmetrize or antisymmetrize the product states.

2.1.2 Symmetrization and antisymmetrization

A mathematical object X_{ab} is said to be $\pm$**-symmetric** in the variables a and b if

$$X_{ab} = \pm X_{ba}$$

Note that:

- *+-symmetric* is more commonly known as *symmetric* (without any qualification).

[4] Mathematically, the space $\mathbb{H}^0$ is isomorphic to complex numbers.

- $-$-*symmetric* is more commonly known as *antisymmetric* or *skew-symmetric*.

Objects with more than two variables may be ±-symmetric in many combinations of variables. In particular, if it is ±-symmetric for every pair of variables in $a_1 \dots a_n$, then it is said to be **fully ±-symmetric** in $a_1 \dots a_n$.

If an object is not fully ±-symmetric, we can make it so using the **±-symmetrization symbol** $\mathcal{S}^{\pm}$, defined as

$$\mathcal{S}^{\pm}_{\emptyset} X = X$$

$$\mathcal{S}^{\pm}_{a} X_a = X_a$$

$$\mathcal{S}^{\pm}_{ab} X_{ab} = \frac{1}{2}(X_{ab} \pm X_{ba})$$

$$\mathcal{S}^{\pm}_{abc} X_{abc} = \frac{1}{6}(X_{abc} \pm X_{bac} + X_{bca} \pm X_{cba} + X_{cab} \pm X_{acb})$$

$$\mathcal{S}^{\pm}_{a_1 \dots a_n} X_{a_1 \dots a_n} = \frac{1}{N!} \sum_{\sigma \in S_n} (\pm)^{\sigma} X_{\sigma(a_1 \dots a_n)}$$

where

- $X_{a_1 \dots a_n}$ is an arbitrary formula with free variables $a_1, \dots, a_n$,
- S_n is the symmetric group, which contains all possible permutations of n objects,
- $(-)^{\sigma} = \operatorname{sgn}(\sigma)$, i.e. the sign of the permutation σ,
- $(+)^{\sigma} = 1$, and
- $\sigma(a_1 \dots a_n)$ applies the permutation σ to the sequence of indices $a_1 \dots a_n$.

We will often use the abbreviations $\mathcal{S} = \mathcal{S}^{+}$ and $\mathcal{A} = \mathcal{S}^{-}$. We will also sometimes use the shorthand

$$\mathcal{S}^{(i)} = \mathcal{S}^{(-)^i} = \begin{cases} \mathcal{S}^{+} & \text{if } i \text{ is even} \\ \mathcal{S}^{-} & \text{if } i \text{ is odd} \end{cases}$$

Note that the ±-symmetrization symbol $\mathcal{S}^{\pm}$ is merely a notational shorthand, much like the summation symbol $\sum$. It is not a quantum operator, but we can use it to define one. Let us introduce the N-**particle ±-symmetrizer** $\hat{S}^{\pm}_N$, an operator defined as[5]

$$\hat{S}^{\pm}_0|\varnothing\rangle = |\varnothing\rangle$$

$$\hat{S}^{\pm}_1|p\rangle = |p\rangle$$

$$\hat{S}^{\pm}_2|p_1 \otimes p_2\rangle = \frac{|p_1 \otimes p_2\rangle \pm |p_2 \otimes p_1\rangle}{2}$$

$$\hat{S}^{\pm}_3|p_1 \otimes p_2 \otimes p_3\rangle = \frac{|p_1 \otimes p_2 \otimes p_3\rangle \pm |p_2 \otimes p_1 \otimes p_3\rangle + |p_2 \otimes p_3 \otimes p_1\rangle \pm \cdots}{6}$$

$$\hat{S}^{\pm}_N|p_1 \otimes \cdots \otimes p_N\rangle = \mathcal{S}^{\pm}_{p_1 \ldots p_N}|p_1 \otimes \cdots \otimes p_N\rangle$$

Despite the involvement of a specific single-particle basis $|p\rangle$ in the definition of $\hat{S}^{\pm}_N$, one can show that the operator is actually independent of the basis choice.

The $\hat{S}^{\pm}_N$ operator is idempotent and therefore a projection operator. It projects the N-particle Hilbert space $\mathbb{H}^N$ to the ±-symmetric subspace of $\mathbb{H}^N$. By abuse of notation, we will denote this subspace $\hat{S}^{\pm}_N(\mathbb{H}^N)$.

We can now define N-particle ±**-symmetrized states** in terms of product states,

$$|p_1 p_2\rangle^{\pm} = \sqrt{\frac{2}{C_{p_1 p_2}}}\hat{S}^{\pm}_2|p_1 \otimes p_2\rangle = \frac{|p_1 \otimes p_2\rangle \pm |p_2 \otimes p_1\rangle}{\sqrt{2(1 + \delta_{p_1 p_2})}}$$

$$|p_1 p_2 p_3\rangle^{\pm} = \sqrt{\frac{6}{C_{p_1 p_2 p_3}}}\hat{S}^{\pm}_3|p_1 \otimes p_2 \otimes p_3\rangle$$

$$|p_1 \ldots p_N\rangle^{\pm} = \sqrt{\frac{N!}{C_{p_1 \cdots p_N}}}\hat{S}^{\pm}_N|p_1 \otimes \cdots \otimes p_N\rangle$$

[5]The ±-symmetrizer $\hat{S}^{\pm}$ should not be confused with the spin operator $\hat{S}$ introduced in later sections.

where δ_{ab} denotes the Kronecker delta, and

$$C_{p_1 \dots p_N} = \prod_p n_p! \tag{2.1}$$

is a factor that compensates for overcounting due to multiple particles occupying the same state. Here, n_p is the **occupation number** of the single-particle state $|p\rangle$, which counts the number of times that p appears in the list $p_1 \dots p_N$. In the case of fermions, n_p can never be more than one due to the Pauli exclusion principle, and so $C_{p_1 \dots p_N}$ is always one.

Given a $\pm$-symmetric state $|p_1 \dots p_N\rangle^{\pm}$, the particles are said to **occupy** the single-particle states $|p_1\rangle, \dots, |p_N\rangle$. Any remaining unused single-particle states are said to be **unoccupied**.

These states are solutions of the non-interacting Hamiltonian $\hat{H}^{\circ}$ and they form an orthonormal basis for the N-particle $\pm$-symmetric Hilbert space $\hat{S}_N^{\pm}(\mathbb{H}^N)$. Symmetric states are associated with bosons, and antisymmetric states are associated with fermions. Notationally, we distinguish $\pm$-symmetrized states from product states by the absence of the $\otimes$ symbol in the ket.

In the case of fermions, antisymmetrized states are also known as **Slater determinants**, because their wave functions

$$|p_1 \dots p_N\rangle^{-} \leftrightarrow \Phi^{-}_{p_1 \dots p_N}(x_1, \dots, x_N)$$

can be written as a matrix determinant

$$\Phi^{-}_{p_1 \dots p_N}(x_1, \dots, x_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \varphi_{p_1}(x_1) & \cdots & \varphi_{p_N}(x_1) \\ \vdots & \ddots & \vdots \\ \varphi_{p_1}(x_N) & \cdots & \varphi_{p_N}(x_N) \end{vmatrix}$$

Slater determinants are guaranteed to satisfy the Pauli exclusion principle. One cannot construct a

state in which two particles share the same single-particle state as such states are always projected to zero by the antisymmetrizer.

2.2 Second quantization

In everything we have discussed so far, the number of particles N always appears as an explicit parameter. We will now shift toward the **second quantization** formalism, which avoids explicit mention of N by treating all values of N simultaneously. This offers significant simplifications to the mathematics at the cost of adding a new layer of abstraction. The prior formalism will henceforth be known as **first quantization**.

The first step is to direct sum all of the N-particle $\pm$-symmetric Hilbert spaces $\hat{S}_N^{\pm}(\mathbb{H}^N)$ into a unified space, through the so-called **Fock space construction**,

$$\mathbb{F}^{\pm} = \bigoplus_{N=0}^{\infty} \hat{S}_N^{\pm}(\mathbb{H}^N) = \mathbb{H}^0 \oplus \mathbb{H} \oplus \hat{S}_2^{\pm}(\mathbb{H} \otimes \mathbb{H}) \oplus \hat{S}_3^{\pm}(\mathbb{H} \otimes \mathbb{H} \otimes \mathbb{H}) \oplus \cdots$$

Here, $\mathbb{F}^{\pm}$ is the **$\pm$-symmetric Fock space**.

Next, we introduce a set of creation and annihilation operators, collectively known as the **(physical) field operators**, which serve to connect each N-particle Hilbert space to an adjacent $N \pm 1$-particle Hilbert space. The field operators are dependent on the choice of $\pm$-symmetry but we opt to suppress this detail for clarity.

The **annihilation operator** for the single-particle state $|p\rangle$ is denoted $\hat{a}_p$ and has the effect of removing one of the particles that occupy the $|p\rangle$ state,

$$\hat{a}_p |p p_1 \dots p_N\rangle^{\pm} = \sqrt{n_p} |p_1 \dots p_N\rangle^{\pm}$$

where n_p denotes the occupation number of $|p\rangle$ in $|pp_1 \dots p_N\rangle^{\pm}$, i.e. the number of occurrences of p in $pp_1 \dots p_N$. In particular, if $|p\rangle$ is not occupied in a state $|\Phi\rangle^{\pm}$, then the operator collapses it to zero,

$$\hat{a}_p|\Phi\rangle^{\pm} = 0$$

The **creation operator** for the single-particle state $|p\rangle$ is denoted $\hat{a}_p^\dagger$ and is the Hermitian adjoint of $\hat{a}_p$. It has the effect of adding a new particle that occupies the $|p\rangle$ state,

$$\hat{a}_p^\dagger|p_1 \dots p_N\rangle^{\pm} = \sqrt{1 \pm n_p}|pp_1 \dots p_N\rangle^{\pm}$$

where n_p denotes the occupation number of $|p\rangle$ in $|p_1 \dots p_N\rangle^{\pm}$. For fermions, if $|p\rangle$ is already occupied, then the operator collapses it to zero,

$$\hat{a}_p^\dagger|pp_1 \dots p_N\rangle^{-} = 0$$

A $\pm$-symmetric state can be constructed by a chain of creation operators applied to the vacuum state $|\varnothing\rangle$,

$$|p_1 \dots p_N\rangle = \frac{\hat{a}_{p_1}^\dagger \dots \hat{a}_{p_N}^\dagger|\varnothing\rangle}{\sqrt{C_{p_1 \dots p_N}}}$$

where $C_{p_1 \dots p_N}$ is the compensating factor defined in Eq. 2.1. From the $\pm$-symmetry of the the state, we obtain the commutation relations that define the essential behavior of field operators:

$$[\hat{a}_{p_1}, \hat{a}_{p_2}^\dagger]_{\mp} = \delta_{p_1 p_2} \qquad [\hat{a}_{p_1}, \hat{a}_{p_2}]_{\mp} = [\hat{a}_{p_1}^\dagger, \hat{a}_{p_2}^\dagger]_{\mp} = 0$$

where

- $[\hat{X}, \hat{Y}]_- = [\hat{X}, \hat{Y}]$ denotes the commutator, chosen for symmetric states, and
- $[\hat{X}, \hat{Y}]_+ = \{\hat{X}, \hat{Y}\}$ denotes the anticommutator, chosen for antisymmetric states.

2.3 Many-body operators

From this section onward, we will focus mainly on fermions, so all field operators will be anticommuting.

Given a set of N-particle operators $\hat{Q}_N$ acting on the N-particle Hilbert space, we can direct sum all of them to form a particle-number-agnostic operator $\hat{Q}$ that acts on the entire Fock space,

$$\hat{Q} = \bigoplus_{N=0}^{\infty} \hat{Q}_N$$

This can be applied to any operator from first quantization, including the $\pm$-symmetrizer $\hat{S}_N^{\pm} \mapsto \hat{S}^{\pm}$ and the Hamiltonian $\hat{H}_N \mapsto \hat{H}$.

Many-body operators are classified by **rank**, which determines the maximum number of particles that the operator can couple at any given instant. A rank-k operator is more conventionally known as a k-**body operator**.

2.3.1 Zero-body operators

The most trivial kind of operator are **zero-body operators**, which multiplies a state by a number. Formally, we can write a zero-body operator $\hat{Z}$ in the form

$$\hat{Z} = Z\hat{1}$$

where Z a complex number and $\hat{1}$ is the identity operator, but for simplicity, it suffices to call Z itself the "operator" as the distinction between $\hat{Z}$ and Z is largely irrelevant. The value of Z can be inferred from its expectation value in the vacuum state,

$$Z = \langle \varnothing | Z | \varnothing \rangle$$

although its expectation value in any state will still yield the same result. They do not contribute anything if the bra and ket states differ.

2.3.2 One-body operators

The next simplest kind of operator are **one-body operators**, which are of the form

$$\hat{T} = \sum_{pq} T_{pq} \hat{a}_p^\dagger \hat{a}_q$$

where T_{pq} defines a matrix of complex numbers indexed by p and q, known as the **matrix elements** of $\hat{T}$. In first quantization, one-body operators are of the form,

$$\hat{T}_N = \sum_{\alpha=1}^{N} \hat{t}(\hat{x}_\alpha, \hat{k}_\alpha)$$

Kinetic energy and external potentials are examples of such operators. The matrix elements of such operators may be obtained via the integral,

$$T_{pq} = \langle p | \hat{T} | q \rangle = \int \varphi_p^*(x) \hat{t} \varphi_q(x) \, \mathrm{d}x$$

One-body operators have two notable properties:

- Their expectation value in the vacuum state is zero:

$$\langle \varnothing | \hat{T} | \varnothing \rangle = 0$$

- They do not contribute if the bra and ket states differ in more than one single-particle state. That is, unless the set intersection $\{p_1, \ldots, p_N\} \cap \{q_1, \ldots, q_N\}$ has at least $N - 1$ elements, then

$$\langle p_1 \cdots p_N | \hat{T} | q_1 \cdots q_N \rangle = 0$$

2.3.3 Two-body operators

Two-body operators take the form[6]

$$\hat{V} = \frac{1}{4} \sum_{pqrs} V_{pqrs} \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r = \frac{1}{4} \sum_{pqrs} V_{pqrs} \hat{a}_p^\dagger \hat{a}_r \hat{a}_q^\dagger \hat{a}_s \tag{2.2}$$

where V_{pqrs} denotes an **antisymmetrized matrix element** of $\hat{V}$. Such matrix elements have the following symmetry properties,

$$V_{pqrs} = \mathcal{A}_{pq} \mathcal{A}_{rs} V_{pqrs} \quad \leftrightarrow \quad V_{pqrs} = -V_{qprs} = V_{qpsr} = -V_{pqsr}$$

In first quantization, two-body operators are of the form,

$$\hat{V}_N = \frac{1}{2} \sum_{\alpha=1}^{N} \sum_{\beta=1}^{N} \hat{v}(\hat{x}_\alpha, \hat{x}_\beta, \hat{k}_\alpha, \hat{k}_\beta)$$

[6] Note the reversal of $\hat{a}_r$ and $\hat{a}_s$.

Interactions, such as the Coulomb interactions, are two-body (or higher) operators. The matrix elements of such operators may be obtained via

$$V_{pqrs} = \langle pq|\hat{V}|rs\rangle = 2\mathcal{A}_{rs}V^{\otimes}_{pqrs}$$

where

$$V^{\otimes}_{pqrs} = \langle p \otimes q|\hat{V}|r \otimes s\rangle = \iint \varphi^*_p(x_1)\varphi^*_q(x_2)\hat{v}\varphi_r(x_1)\varphi_s(x_2)\,\mathrm{d}x_1\,\mathrm{d}x_2$$

are the **non-antisymmetrized matrix elements** of $\hat{V}$, which may considered matrix elements in the product state basis. These matrix elements have a weaker symmetry,

$$V^{\otimes}_{pqrs} = \mathcal{S}_{(p,r)(q,s)}V^{\otimes}_{pqrs}$$

In other words, $V^{\otimes}_{pqrs} = V^{\otimes}_{qpsr}$. This arises from the indistinguishability of particles.

Two-body operators have two notable properties:

- Matrix elements of states with one or fewer particles vanish:

$$\langle\varnothing|\hat{V}|\varnothing\rangle = 0 \qquad\qquad \langle p|\hat{V}|q\rangle = 0$$

- They do not contribute if the bra and ket states differ in more than two single-particle states. That is, unless the set intersection $\{p_1, \ldots, p_N\} \cap \{q_1, \ldots, q_N\}$ has at least $N - 2$ elements, then

$$\langle p_1 \cdots p_N|\hat{V}|q_1 \cdots q_N\rangle = 0$$

2.3.4 Three-body operators and beyond

Three-body and, more generally, k-body operators take the form[7]

$$\hat{W} = \frac{1}{36} \sum_{pqrstu} W_{pqrstu} \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r^\dagger \hat{a}_u \hat{a}_t \hat{a}_s = \frac{1}{36} \sum_{pqrstu} W_{pqrstu} \hat{a}_p^\dagger \hat{a}_s \hat{a}_q^\dagger \hat{a}_t \hat{a}_r^\dagger \hat{a}_u$$

$$\begin{aligned}\hat{X} &= \frac{1}{k!^2} \sum_{p_1 \dots p_k q_1 \dots q_k} X_{p_1 \dots p_k q_1 \dots q_k} \hat{a}_{p_1}^\dagger \dots \hat{a}_{p_k}^\dagger \hat{a}_{q_k} \dots \hat{a}_{q_1} \\ &= \frac{1}{k!^2} \sum_{p_1 \dots p_k q_1 \dots q_k} X_{p_1 \dots p_k q_1 \dots q_k} (\hat{a}_{p_1}^\dagger \hat{a}_{q_1}) \dots (\hat{a}_{p_k}^\dagger \hat{a}_{q_k})\end{aligned}$$

with non-antisymmetrized matrix elements

$$W^{\otimes}_{pqrstu} = \langle p \otimes q \otimes r | \hat{W} | s \otimes t \otimes u \rangle$$

$$X^{\otimes}_{p_1 \dots p_k q_1 \dots q_k} = \langle p_1 \otimes \dots \otimes p_k | \hat{X} | q_1 \otimes \dots \otimes q_k \rangle$$

and antisymmetrized matrix elements

$$W_{pqrstu} = \langle pqr | \hat{W} | stu \rangle = 6 \mathcal{A}_{stu} W^{\otimes}_{pqrstu}$$

$$X_{p_1 \dots p_k q_1 \dots q_k} = \langle p_1 \dots p_k | \hat{X} | q_1 \dots q_k \rangle = k! \mathcal{A}_{q_1 \dots q_k} X^{\otimes}_{p_1 \dots p_k q_1 \dots q_k}$$

and symmetries

$$W^{\otimes}_{pqrstu} = \mathcal{S}_{(p,s)(q,t)(r,u)} W^{\otimes}_{pqrstu}$$

$$X^{\otimes}_{p_1 \dots p_k q_1 \dots q_k} = \mathcal{S}_{(p_1,q_1) \dots (p_k,q_k)} X^{\otimes}_{p_1 \dots p_k q_1 \dots q_k}$$

$$W_{pqrstu} = \mathcal{A}_{pqr} \mathcal{A}_{stu} W_{pqrstu}$$

$$X_{p_1 \dots p_k q_1 \dots q_k} = \mathcal{A}_{p_1 \dots p_k} \mathcal{A}_{q_1 \dots q_k} X_{p_1 \dots p_k q_1 \dots q_k}$$

[7] Note the reversal of annihilation operators again.

k-body operators have two notable properties:

- Matrix elements of states with one or fewer particles vanish. That is, if $m < k$, then

$$\langle p_1 \dots p_m | \hat{X} | q_1 \dots q_m \rangle = 0$$

- They do not contribute if the bra and ket states differ in more than k single-particle states. That is, unless the set intersection $\{p_1, \dots, p_N\} \cap \{q_1, \dots, q_N\}$ has at least $N - k$ elements, then

$$\langle p_1 \cdots p_N | \hat{X} | q_1 \cdots q_N \rangle = 0$$

2.4 Particle-hole formalism

Any state in Fock space can be described by a chain of creation operators applied to the vacuum state. In many-body systems, the number of particles can be quite large, leading to long chains of creation operators.

Instead of starting from scratch (i.e. vacuum state), it can be more convenient to start from a pre-existing N-particle Slater determinant $|\Phi\rangle$, called the **reference state**,

$$|\Phi\rangle = \hat{a}^\dagger_{p_1} \cdots \hat{a}^\dagger_{p_N} |\varnothing\rangle$$

Then, to access other states we would create and/or annihilate particles relative to the reference state. For example,

$$\hat{a}^\dagger_p \hat{a}_{p_1} |\Phi\rangle = \hat{a}^\dagger_p \hat{a}^\dagger_{p_2} \cdots \hat{a}^\dagger_{p_N} |\varnothing\rangle$$

In addition to simplifying the algebra, this also provides a coarse measure of "distance" from the reference state, quantified by the number of field operators applied to the reference state that is needed.

Formally, we can construct a set of **quasiparticle (particle-hole) field operators** $\hat{b}_p^\dagger$ and $\hat{b}_p$,

$$\hat{b}_p = \begin{cases} \hat{a}_p & \text{if } p \text{ is unoccupied in the reference state} \\ \hat{a}_p^\dagger & \text{if } p \text{ is occupied in the reference state} \end{cases}$$

In this context, $\hat{b}_p^\dagger$ applied to an occupied state p is said to *create* a **hole state**, whereas $\hat{b}_p^\dagger$ applied to an unoccupied state p is said to create a **particle state**. These operators define the so-called **particle-hole formalism**.

The quasiparticle operators have an algebra analogous to the original field operators $\hat{a}_p^\dagger$ and $\hat{a}_p$:

$$\left\{\hat{b}_p, \hat{b}_q^\dagger\right\} = \delta_{pq} \qquad \left\{\hat{b}_p, \hat{b}_q\right\} = \left\{\hat{b}_p^\dagger, \hat{b}_q^\dagger\right\} = 0 \qquad \hat{b}_p|\Phi\rangle = 0$$

The quasiparticle field operators treat $|\Phi\rangle$ as their "vacuum" state, similar to how original field operators treat $|\varnothing\rangle$ as their vacuum state. For this reason, the reference state $|\Phi\rangle$ is also known as a **Fermi vacuum**.

So far, we have used the letters p, q, r, … to label single-particle states, which contain both hole and particle states relative to the Fermi vacuum. We will continue to use this convention. It is often convenient to sum over only hole states, or only particle states. To this end, we introduce a convention where i, j, k, … are used to label hole states and a, b, c, … label particle states. We

also introduce a special notation for **summation over holes and particles**:

$$\sum_{ijk\ldots\backslash abc\ldots} \cdots \tag{2.3}$$

The backslash serves as an additional reminder that $i, j, k, \ldots$ should be summed over hole states only, and $a, b, c, \ldots$ should be summed over particle states only.

In this formalism, we use the following concise notation to denote states near the reference state,

$$|\Phi_{a_1 \ldots a_k i_1 \ldots i_k}\rangle = \hat{b}^\dagger_{a_1} \ldots \hat{b}^\dagger_{a_k} \hat{b}^\dagger_{i_k} \ldots \hat{b}^\dagger_{i_1} |\Phi\rangle = \hat{a}^\dagger_{a_1} \ldots \hat{a}^\dagger_{a_k} \hat{a}_{i_k} \ldots \hat{a}_{i_1} |\Phi\rangle$$

Note that although this state has $2k$ quasi-particles, it still has exactly N physical particles, because the particles and holes cancel out exactly. We also introduce a shorthand for its energy,

$$E_{\Phi_{a_1 \ldots a_k i_1 \ldots i_k}} = \langle \Phi_{a_1 \ldots a_k i_1 \ldots i_k} | \hat{H} | \Phi_{a_1 \ldots a_k i_1 \ldots i_k} \rangle$$

2.5 Normal ordering

We say a product of field operators is in **normal order** if all creation operators appear before all annihilation operators.

More generally, we may define normal ordering to be an arbitrarily chosen total order subject to the constraint that if both $\hat{\alpha}\hat{\beta}$ and $\hat{\beta}\hat{\alpha}$ are normal ordered then $[\hat{\alpha}, \hat{\beta}]_\mp = 0$. But we will not need this general definition here. Wick's theorem (introduced later) does hold in this general setting, though.

Given a monomial $c\hat{\Pi}$, where c is a numeric coefficient and $\hat{\Pi}$ is a product of field operators,[8] we define its **normal ordering** as

$$:c\hat{\Pi}: = (\pm)^{\sigma} c\, \sigma(\hat{\Pi})$$

where σ is a permutation that rearranges the field operators in $\hat{\Pi}$ such that $\sigma(\Pi)$ is in normal order, and

- for bosons, $(+)^{\sigma} = +$;
- for fermions, $(-)^{\sigma}$ is the sign of the permutation σ.

Although the definition leaves the permutation underdetermined, the definition of normal order ensures that normal ordering will yield the same result no matter which permutation is chosen.

The operation is dependent on the choice of field operators. In this text, we have two choices:

- When normal ordering relative to the *physical vacuum*, the physical field operators $\hat{a}_p^\dagger$ and $\hat{a}_p$ are used.
- When normal ordering relative to the *Fermi vacuum*, the quasiparticle field operators $\hat{b}_p^\dagger$ and $\hat{b}_p$ are used.

We will typically work with normal ordering relative to the Fermi vacuum. For normal ordering relative to the physical vacuum, we use three dots instead of a colon:

$$\vdots c\hat{\Pi} \vdots$$

[8] A product of field operators is sometimes referred to as an *operator string*.

Several notations exist for this operation:

$$:c\hat{\Pi}: \qquad \mathrm{N}[c\hat{\Pi}] \qquad \left\{ c\hat{\Pi} \right\}$$

The colon notation is common in nuclear physics, [PMB17] whereas delimiters prefixed with the letter "N" [SB09] or curly braces [Rei13] are common in quantum chemistry.

As an example, consider the fermionic monomial $-2\hat{b}_p\hat{b}_q^\dagger$ with coefficient $c = -2$ and operators $\hat{\Pi} = \hat{b}_p\hat{b}_q^\dagger$. We can normal order it relative to the Fermi vacuum:

$$:-2\hat{b}_p\hat{b}_q^\dagger: = 2\hat{b}_q^\dagger\hat{b}_p$$

Since this permutation has odd parity, the sign is flipped.

As another example, consider the fermionic monomial $\hat{a}_p\hat{a}_q^\dagger\hat{a}_r^\dagger\hat{a}_s\hat{a}_t$. It has several normal-ordered forms:

$$\vdots\hat{a}_p\hat{a}_q^\dagger\hat{a}_r^\dagger\hat{a}_s\hat{a}_t\vdots = \hat{a}_q^\dagger\hat{a}_r^\dagger\hat{a}_p\hat{a}_s\hat{a}_t = -\hat{a}_q^\dagger\hat{a}_r^\dagger\hat{a}_p\hat{a}_t\hat{a}_s = \hat{a}_r^\dagger\hat{a}_q^\dagger\hat{a}_p\hat{a}_t\hat{a}_s = \cdots$$

They are all equally valid.

Normal ordering is idempotent, and moreover supersedes existing normal orderings:

$$:(\hat{\Pi} :\hat{\Gamma}: \hat{\Lambda}): = :\hat{\Pi}\hat{\Gamma}\hat{\Lambda}:$$

Bosonic/fermionic normal-ordered products are symmetric/antisymmetric under operator exchange, i.e. for any permutation σ,

$$:\hat{\Pi}: = (\pm)^\sigma :\sigma(\hat{\Pi}):$$

2.5.1 Matrix elements relative to the Fermi vacuum

The primary application of normal ordering is in the redistribution of matrix elements between different ranks of operators, as the rank of an operator is dependent on the vacuum used.

Usually, many-body operators are given relative to the physical vacuum. For example, a (0, 1, 2, 3)-body operator $\hat{H}$ has the standard form

$$\hat{H} = E_\varnothing + \hat{H}_1^\varnothing + \hat{H}_2^\varnothing + \hat{H}_3^\varnothing$$

where $E_\varnothing$ is the physical vacuum energy (0-body component) and $\hat{H}_k^\varnothing$ denotes its k-body component relative to the physical vacuum. In other words, the monomials of each component are already in normal order relative to the physical vacuum,

$$\begin{aligned}
\hat{H}_1^\varnothing &= \sum_{pq} H_{pq}^\varnothing \, \vdots \hat{a}_p^\dagger \hat{a}_q \vdots \\
\hat{H}_2^\varnothing &= \frac{1}{4} \sum_{pqrs} H_{pqrs}^\varnothing \, \vdots \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r \vdots \\
\hat{H}_3^\varnothing &= \frac{1}{36} \sum_{pqrstu} H_{pqrstu}^\varnothing \, \vdots \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r^\dagger \hat{a}_u \hat{a}_t \hat{a}_s \vdots
\end{aligned}$$

These definitions are identical to those in Sec. 2.3. The normal ordering notation is present only for emphasis.

If we rearrange their monomials so that they are in normal order relative to the *Fermi* vacuum, portions of higher-rank operators would migrate to lower-rank operators. Take, for example, the two-body component:

$$\vdots \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r \vdots = : \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r : + 4 \mathcal{A}_{pq} \mathcal{A}_{rs} n_q \delta_{qs} : \hat{a}_p^\dagger \hat{a}_r : + 2 \mathcal{A}_{rs} n_p n_q \delta_{pr} \delta_{qs}$$

This leads to a different decomposition of the operators into particle-hole components,

$$\hat{H} = E_\Phi + \hat{H}_1^\Phi + \hat{H}_2^\Phi + \hat{H}_3^\Phi$$

where E_Φ is the Fermi vacuum energy and $\hat{H}_k^\Phi$ denotes[9] its k-body component relative to the Fermi vacuum $|\Phi\rangle$,

$$\begin{aligned}
\hat{H}_1^\Phi &= \sum_{pq} H_{pq}^\Phi : \hat{a}_p^\dagger \hat{a}_q : \\
\hat{H}_2^\Phi &= \frac{1}{4} \sum_{pqrs} H_{pqrs}^\Phi : \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r : \\
\hat{H}_3^\Phi &= \frac{1}{36} \sum_{pqrstu} H_{pqrstu}^\Phi : \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r^\dagger \hat{a}_u \hat{a}_t \hat{a}_s :
\end{aligned}$$

and with matrix elements given by

$$\begin{aligned}
E_\Phi &= E_\varnothing + \sum_{i\backslash} H_{ii}^\varnothing + \frac{1}{2} \sum_{ij\backslash} H_{ijij}^\varnothing + \frac{1}{6} \sum_{ijk\backslash} H_{ijkijk}^\varnothing \\
H_{pq}^\Phi &= H_{pq}^\varnothing + \sum_{i\backslash} H_{piqi}^\varnothing + \frac{1}{2} \sum_{ij\backslash} H_{pijqij}^\varnothing \\
H_{pqrs}^\Phi &= H_{pqrs}^\varnothing + \sum_{i\backslash} H_{pqirsi}^\varnothing \\
H_{pqrstu}^\Phi &= H_{pqrstu}^\varnothing
\end{aligned} \tag{2.4}$$

Note that these particle-hole components $\hat{H}_k^\Phi$ do not conserve the number of quasiparticles, so their algebraic properties are not entirely analogous to the physical components $\hat{H}_k^\varnothing$. Properly rewriting the operators in terms of quasiparticle field operators would lead to tedious results such

[9] In other literature, it is common to use the symbols $\hat{F} = \hat{H}_1^\Phi$, $\hat{\Gamma} = \hat{H}_2^\Phi$, and $\hat{W} = \hat{H}_3^\Phi$.

as:

$$\hat{H}_1^{\Phi} = \sum_{i\backslash a} H_{ai}^{\Phi} : \hat{b}_a^\dagger \hat{b}_i^\dagger : + \sum_{\backslash ab} H_{ab}^{\Phi} : \hat{b}_a^\dagger \hat{b}_b : - \sum_{ij\backslash} H_{ij}^{\Phi} : \hat{b}_j^\dagger \hat{b}_i : + \sum_{i\backslash a} H_{ia}^{\Phi} : \hat{b}_i \hat{b}_a :$$

Fortunately, we do not need to do this.

The matrix elements in Eq. 2.4 can be derived using the definition of normal ordering and the usual commutation relations, more efficiently, using Wick's theorem or many-body diagrams.

2.5.2 Ambiguity of normal ordering on non-monomials

The following expressions are not monomials:

$$1 + \hat{a}_p \hat{a}_q^\dagger \qquad\qquad e^{\hat{a}_p^\dagger \hat{a}_q} \qquad\qquad \hat{H}$$

Normal ordering is *ill-defined* (ambiguous) on such non-monomial operators. The following paradox illustrates the problem: consider the fermionic product $\hat{a}_p^\dagger \hat{a}_q \hat{a}_r$,

$$\begin{aligned}
\hat{a}_p^\dagger \hat{a}_q \hat{a}_r &= \vdots \hat{a}_p^\dagger \hat{a}_q \hat{a}_r \vdots \\
&\stackrel{?}{=} \vdots \delta_{pq} \hat{a}_r - \hat{a}_q \hat{a}_p^\dagger \hat{a}_r \vdots \\
&\stackrel{?}{=} \vdots \delta_{pq} \hat{a}_r \vdots - \vdots \hat{a}_q \hat{a}_p^\dagger \hat{a}_r \vdots \\
&= \delta_{pq} \hat{a}_r + \hat{a}_p^\dagger \hat{a}_q \hat{a}_r
\end{aligned}$$

To eliminate ambiguity one must expand the expression in a very specific way, causing normal ordering of non-monomial expressions to be sensitive to how it is written (its syntactic form). It is possible for two non-monomial expressions that are semantically equal to have semantically unequal normal-ordered results, leading to the paradox above. This paradox does not occur if we

focus solely on monomial operators.

When normal ordering is applied non-monomial expressions, it is conventional to choose the most "direct" expansion without any usage of the ±-commutation relations. For example,

$$\begin{aligned} :1 + \hat{b}_p^\dagger \hat{b}_q: \quad &\longrightarrow \quad 1+ :\hat{b}_p^\dagger \hat{b}_q: \\ :\mathrm{e}^{\hat{b}_p^\dagger \hat{b}_q}: \quad &\longrightarrow \quad 1+ :\hat{b}_p^\dagger \hat{b}_q: + :\tfrac{1}{2}\hat{b}_p^\dagger \hat{b}_q \hat{b}_p^\dagger \hat{b}_q: + \cdots \\ :\hat{H}: \quad &\longrightarrow \quad E+ :\hat{H}_1: + :\hat{H}_2: + \cdots \end{aligned}$$

2.6 Wick's theorem

Wick's theorem [Wic50] is an algebraic theorem for simplifying products of bosonic (commuting) or fermionic (anticommuting) field operators into sums of normal-ordered products. Many of the equations in the previous sections can be derived much more quickly and systematically using this theorem.

To describe the theorem, it is necessary to introduce the concept of Wick contractions.

2.6.1 Adjacent Wick contractions

In the simplest case, a **Wick contraction** between two adjacent field operators $\hat{\alpha}$ and $\hat{\beta}$, denoted by a connecting line, is defined as,

$$\overset{\sqcap}{\hat{\alpha}\hat{\beta}} = \hat{\alpha}\hat{\beta} - :\hat{\alpha}\hat{\beta}: \tag{2.5}$$

For Wick's theorem to apply, we require the result of a contraction to be a number, not an operator.[10]

Intuitively, contractions are the "remainder" of normal ordering. We can elaborate the definition

[10] That is, the result lie in the center of the algebra, commuting with all other elements.

to

$$\hat{\alpha}\hat{\beta} = \begin{cases} 0 & \text{if } \hat{\alpha}\hat{\beta} \text{ is in normal order} \\ \left[\hat{\alpha}, \hat{\beta}\right]_{\mp} & \text{if } \hat{\alpha}\hat{\beta} \text{ is not in normal order} \end{cases} \tag{2.6}$$

where commutator $[,]_- = [,]$ is chosen for bosons and anticommutator $[,]_+ = \{,\}$ is chosen for fermions. Observe that contractions have no effect if a product is already normal ordered.

For fermionic physical operators $\hat{a}_p$, we obtain these cases for contractions relative to the physical vacuum:

$$\underbrace{\hat{a}_p \hat{a}_q^\dagger} = \delta_{pq} \qquad\qquad \underbrace{\hat{a}_p \hat{a}_q} = \underbrace{\hat{a}_p^\dagger \hat{a}_q} = \underbrace{\hat{a}_p^\dagger \hat{a}_q^\dagger} = 0$$

Here, connecting lines are drawn at the bottom of the symbols to indicate that these are contractions relative to the *physical* vacuum. That is,

$$\underbrace{\hat{\alpha}\hat{\beta}} = \hat{\alpha}\hat{\beta} - :\hat{\alpha}\hat{\beta}:$$

We obtain analogous relations for contractions of fermionic quasiparticle operators $\hat{b}_p$ relative to the Fermi vacuum:

$$\overbrace{\hat{b}_p \hat{b}_q^\dagger} = \delta_{pq} \qquad\qquad \overbrace{\hat{b}_p \hat{b}_q} = \overbrace{\hat{b}_p^\dagger \hat{b}_q} = \overbrace{\hat{b}_p^\dagger \hat{b}_q^\dagger} = 0$$

However, contraction of physical operators $\hat{a}_p$ relative to the Fermi vacuum are different:

$$\overbrace{\hat{a}_p \hat{a}_q^\dagger} = (1 - n_p)\delta_{pq} \qquad \overbrace{\hat{a}_p^\dagger \hat{a}_q} = n_p \delta_{pq} \qquad \overbrace{\hat{a}_p \hat{a}_q} = \overbrace{\hat{a}_p^\dagger \hat{a}_q^\dagger} = 0$$

where n_p is equal to 1 if p is occupied in the reference state, 0 otherwise.

2.6.2 Normal-ordered Wick contractions

Next, we consider non-adjacent Wick contractions, which is only defined *within* a normal-ordered product. Note that the contraction operation takes place prior to the normal ordering operation, otherwise the operation would be useless (it would always vanish according to Eq. 2.6).

Let $\hat{\Pi}$ be a product of m field operators. We define the Wick contraction of $\hat{\alpha}$ and $\hat{\beta}$ within a normal ordered product $:\hat{\alpha}\hat{\Pi}\hat{\beta}:$ as

$$:\hat{\alpha}\hat{\Pi}\hat{\beta}: = (\pm)^m \hat{\alpha}\hat{\beta} :\hat{\Pi}:$$

where the sign is (+) for bosonic operators and (−) for fermionic operators.

In effect, we unite the two contracted operators using the ±-symmetries of the normal-ordered product before attempting to evaluate the contraction. Consider this example with fermionic operators,

$$:\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}: = - :\hat{\beta}\hat{\alpha}\hat{\gamma}\hat{\delta}: = -\hat{\alpha}\hat{\gamma} :\hat{\beta}\hat{\delta}:$$

Note that it does not matter whether $\{\hat{\alpha}, \hat{\beta}\}$ is zero, because inside a normal-ordered product all operators behave *as though* their ±-commutators are zero.

2.6.3 Multiple Wick contractions

Given a normal-ordered product of n field operators, there can be multiple ways to contract its contents. Consider the case $n = 4$,

$$:\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}:$$

$$:\overbrace{\hat{\alpha}\hat{\beta}}\hat{\gamma}\hat{\delta}: \quad :\overbrace{\hat{\alpha}\hat{\beta}\hat{\gamma}}\hat{\delta}: \quad :\overbrace{\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}}: \quad :\hat{\alpha}\overbrace{\hat{\beta}\hat{\gamma}}\hat{\delta}: \quad :\hat{\alpha}\overbrace{\hat{\beta}\hat{\gamma}\hat{\delta}}: \quad :\hat{\alpha}\hat{\beta}\overbrace{\hat{\gamma}\hat{\delta}}:$$

$$:\overbrace{\hat{\alpha}\hat{\beta}}\overbrace{\hat{\gamma}\hat{\delta}}: \quad :\overbrace{\hat{\alpha}\overbrace{\hat{\beta}\hat{\gamma}}\hat{\delta}}: \quad :\overbrace{\hat{\alpha}\hat{\beta}\overbrace{\hat{\gamma}\hat{\delta}}}:$$

The first row contains all normal-ordered products with zero contractions. This is the trivial case.

The second row contains all normal-ordered products with exactly one contraction. We shall denote the sum of all entries in the second row by

$$:C^1(\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}): = :\overbrace{\hat{\alpha}\hat{\beta}}\hat{\gamma}\hat{\delta}: + :\overbrace{\hat{\alpha}\hat{\beta}\hat{\gamma}}\hat{\delta}: + :\overbrace{\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}}: + :\hat{\alpha}\overbrace{\hat{\beta}\hat{\gamma}}\hat{\delta}: + :\hat{\alpha}\overbrace{\hat{\beta}\hat{\gamma}\hat{\delta}}: + :\hat{\alpha}\hat{\beta}\overbrace{\hat{\gamma}\hat{\delta}}:$$

The third row contains all normal-ordered products with exactly two contractions. Analogous to the second row, we shall denote the sum of all entries in the third row by

$$:C^2(\hat{\alpha}\hat{\beta}\hat{\gamma}\hat{\delta}): = :\overbrace{\hat{\alpha}\hat{\beta}}\overbrace{\hat{\gamma}\hat{\delta}}: + :\overbrace{\hat{\alpha}\overbrace{\hat{\beta}\hat{\gamma}}\hat{\delta}}: + :\overbrace{\hat{\alpha}\hat{\beta}\overbrace{\hat{\gamma}\hat{\delta}}}:$$

We can generalize this notation. If we have a product $\hat{\Pi}$ of n field operators, the notation $:C^k(\hat{\Pi}):$ denotes the sum of all normal-ordered products with exactly k contractions. In particular:

- If $k = 0$, then there are no contractions. Therefore,

$$:C^0(\hat{\Pi}): = :\hat{\Pi}:$$

- It is impossible to have more than $\lfloor n/2 \rfloor$ contractions. Therefore,

$$2k > n \qquad \Longrightarrow \qquad :C^k(\hat{\Pi}): = 0$$

2.6.4 Statement of Wick's theorem

We are now ready to state **Wick's theorem**: A product of n field operators $\hat{\Pi}$ may be expanded as

$$\hat{\Pi} = \sum_{k=0}^{\infty} :C^k(\hat{\Pi}): = :\hat{\Pi}: + \sum_{k=1}^{\lfloor n/2 \rfloor} :C^k(\hat{\Pi}):$$

That is, any product of field operators may be expanded as a sum of every possible combination of contractions within its normal-ordered product.

The definition of contractions in Eq. 2.5 yields a special case of Wick's theorem for exactly two field operators,

$$\hat{\alpha}\hat{\beta} = :\hat{\alpha}\hat{\beta}: + \overbrace{\hat{\alpha}\hat{\beta}}$$

There is also a **generalized Wick's theorem**: A product of two normal-ordered products $:\hat{\Pi}:$ and $:\hat{\Gamma}:$ may be expanded as

$$:\hat{\Pi}: \times :\hat{\Gamma}: = \sum_{k=0}^{\infty} :C^k(:\hat{\Pi}: \times :\hat{\Gamma}:):$$

where $:C^k(\,:\hat{\Pi}:\times:\hat{\Gamma}:\,):$ is a sum of every possible combination of k contractions between elements of $:\hat{\Pi}:$ and elements of $:\hat{\Gamma}:$ within the normal-ordered product $:(\,:\hat{\Pi}:\times:\hat{\Gamma}:\,):$. Contractions among elements of $:\hat{\Pi}:$ are excluded (and likewise for $:\hat{\Gamma}:$) because contraction between elements that are already in normal order is zero (Eq. 2.6). Despite its name, the generalized Wick's theorem is not actually a generalization but a special case of Wick's theorem when the product is already partly in normal order.

2.6.5 Proof of Wick's theorem

To prove Wick's theorem, we need the following lemma:

$$\hat{\alpha}\hat{\gamma}_1 \dots \hat{\gamma}_j = (\pm)^j \hat{\gamma}_1 \dots \hat{\gamma}_j \hat{\alpha} + \sum_{i=1}^{j} (\pm)^{i-1} \hat{\gamma}_1 \dots \hat{\gamma}_{i-1} [\hat{\alpha}, \hat{\gamma}_i]_{\mp} \hat{\gamma}_{i+1} \cdots \hat{\gamma}_j$$

which describes the process of moving $\hat{\alpha}$ from the left of a product $\hat{\gamma}_1 \dots \hat{\gamma}_j$ to the right. In doing so, it accumulates a series of $\pm$-commutators involving $\hat{\alpha}$.

To prove the lemma by induction, we start with the obvious base case for $j = 0$,

$$\hat{\alpha} = (\pm)^0 \hat{\alpha} + 0$$

For the induction step, assume the lemma holds for j. Then, we can prove it holds for $j + 1$ as well,

$$\begin{aligned}
&\hat{\alpha}\hat{\gamma}_1 \dots \hat{\gamma}_{j+1} \\
&= (\pm)^j \hat{\gamma}_1 \dots \hat{\gamma}_j \hat{\alpha} \hat{\gamma}_{j+1} + \sum_{i=1}^{j} (\pm)^{i-1} \hat{\gamma}_1 \dots \hat{\gamma}_{i-1} [\hat{\alpha}, \hat{\gamma}_i]_{\mp} \hat{\gamma}_{i+1} \cdots \hat{\gamma}_{j+1} \\
&= (\pm)^j \hat{\gamma}_1 \dots \hat{\gamma}_j (\pm \hat{\gamma}_{j+1} \hat{\alpha} + [\hat{\alpha}, \hat{\gamma}_{j+1}]_{\mp}) + \sum_{i=1}^{j} (\pm)^{i-1} \hat{\gamma}_1 \dots \hat{\gamma}_{i-1} [\hat{\alpha}, \hat{\gamma}_i]_{\mp} \hat{\gamma}_{i+1} \cdots \hat{\gamma}_{j+1} \\
&= (\pm)^{j+1} \hat{\gamma}_1 \dots \hat{\gamma}_{j+1} \hat{\alpha} + \sum_{i=1}^{j+1} (\pm)^{i-1} \hat{\gamma}_1 \dots \hat{\gamma}_{i-1} [\hat{\alpha}, \hat{\gamma}_i]_{\mp} \hat{\gamma}_{i+1} \cdots \hat{\gamma}_{j+1}
\end{aligned}$$

thus we have proven the case for $j + 1$. ■

The lemma is very general but also verbose. If we assume the ±-commutators are all numbers, then there is another way to state the lemma using Wick contractions and normal ordering. Suppose we have a product $\hat{\gamma}_1 \dots \hat{\gamma}_j \hat{\alpha} \hat{\gamma}_{j+1} \dots \hat{\gamma}_m$ that is in normal order. Observe that:

- If $i \leq j$, then $\hat{\alpha}^{\sqcap} \hat{\gamma}_i = [\hat{\alpha}, \hat{\gamma}_i]_{\mp}$ because either (a) $\hat{\alpha} \hat{\gamma}_i$ is not in normal order, or (b) both $\hat{\alpha} \hat{\gamma}_i$ and $\hat{\gamma}_i \hat{\alpha}$ are in normal order, in which case both the ±-commutator and contraction are zero.
- If $i > j$, then $\hat{\alpha}^{\sqcap} \hat{\gamma}_i = 0$ because $\hat{\alpha} \hat{\gamma}_i$ is in normal order.

Therefore, we can replace the ±-commutators with Wick contractions and artifically raise the upper limit of the summation from j to m because all those extra ±-commutators are zero, leading to

$$\hat{\alpha} \hat{\gamma}_1 \dots \hat{\gamma}_m = (\pm)^j \hat{\gamma}_1 \dots \hat{\gamma}_j \hat{\alpha} \hat{\gamma}_{j+1} \dots \hat{\gamma}_m + \sum_{i=1}^{m} (\pm)^{i-1} \hat{\alpha}^{\sqcap} \hat{\gamma}_i \hat{\gamma}_1 \dots \hat{\gamma}_{i-1} \hat{\gamma}_{i+1} \dots \hat{\gamma}_m$$

We can rewrite this using our definition of normal ordering and Wick contraction within normal-ordered products,

$$\hat{\alpha} : \hat{\gamma}_1 \dots \hat{\gamma}_m : = : \hat{\alpha} \hat{\gamma}_1 \dots \hat{\gamma}_m : + \sum_{i=1}^{m} : \hat{\alpha}^{\overbrace{}} \hat{\gamma}_1 \dots \hat{\gamma}_i \dots \hat{\gamma}_m :$$

If we define $: \hat{\Gamma} : = s : \hat{\gamma}_1 \dots \hat{\gamma}_m :$ with some sign s, we may rewrite the previous equation as:

$$\frac{1}{s} \hat{\alpha} : \hat{\Gamma} : = \frac{1}{s} : \hat{\alpha} \hat{\Gamma} : + \frac{1}{s} : \hat{\alpha}^{\sqcap} \hat{\Gamma} : \tag{2.7}$$

where $: \hat{\alpha}^{\sqcap} \hat{\Gamma} :$ is a shorthand for

$$: \hat{\alpha}^{\sqcap} \hat{\Gamma} : = \sum_{i=1}^{m} s : \hat{\alpha}^{\overbrace{}} \hat{\gamma}_1 \dots \hat{\gamma}_i \dots \hat{\gamma}_m :$$

Then, Eq. 2.7 simplifies to:

$$\hat{\alpha}\,{:}\hat{\Gamma}{:} = {:}\hat{\alpha}\hat{\Gamma}{:} + {:}\hat{\alpha}\hat{\Gamma}{:} \tag{2.8}$$

Now we may prove Wick's theorem using induction. The base case is obvious,

$$1 = {:}1{:}$$

For the inductive step, we start by assuming that Wick's theorem holds for the product $\hat{\Pi}$ and wish to prove that it holds for $\hat{\alpha}\hat{\Pi}$. From the assumption, we write

$$\hat{\alpha}\hat{\Pi} = \sum_{k=0}^{\infty} \hat{\alpha}\,{:}\hat{C}^k(\hat{\Pi}){:}$$

Now we may use the lemma in Eq. 2.8 with $\hat{\Gamma} = \hat{C}^k(\hat{\Pi})$,

$$\begin{aligned}\sum_{k=0}^{\infty} \hat{\alpha}\,{:}\hat{C}^k(\hat{\Pi}){:} &= \sum_{k=0}^{\infty} {:}\hat{\alpha}\hat{C}^k(\hat{\Pi}){:} + \sum_{k=0}^{\infty} {:}\hat{\alpha}\hat{C}^k(\hat{\Pi}){:} \\ &= {:}\hat{\alpha}\hat{C}^0(\hat{\Pi}){:} + \sum_{k=1}^{\infty} \left({:}\hat{\alpha}\hat{C}^k(\hat{\Pi}){:} + {:}\hat{\alpha}\hat{C}^{k-1}(\hat{\Pi}){:} \right)\end{aligned}$$

Observe that ${:}\hat{C}^0(\hat{\Pi}){:} = {:}\hat{\Pi}{:}$ and

$${:}\hat{C}^k(\hat{\alpha}\hat{\Pi}){:} = {:}\hat{\alpha}\hat{C}^k(\hat{\Pi}){:} + {:}\hat{\alpha}\hat{C}^{k-1}(\hat{\Pi}){:}$$

In other words, k contractions in $\hat{\alpha}\hat{\Pi}$ can be either

- k contractions only within $\hat{\Pi}$, or
- one contraction between $\hat{\alpha}$ and $\hat{\Pi}$ and $k-1$ contractions within $\hat{\Pi}$.

Hence,

$$\hat{\alpha}\hat{\Pi} = :\hat{\alpha}\hat{\Pi}: + \sum_{k=1}^{\infty} :C^k(\hat{\alpha}\hat{\Pi}):$$

■

2.7 Many-body diagrams

Many-body diagrams [Gol57; Hug57] provide a graphical way to apply Wick's theorem in the Fermi vacuum and express summations of matrix elements. They are analogous to Feymann diagrams [Fey49] but with single-particle states in lieu of fundamental particles. We will provide an overview of many-body diagrams, but as it is a fairly substantial topic, we refer interested readers to [SB09] for a more in-depth explanation.

A diagram is composed of a set of **nodes** (vertices) and a set of possibly directed **lines** (edges), arranged in a layout similar to graphs. However, diagrams do have a few differences that distinguish them from graphs in the conventional sense:

- Nodes may not be point-like entities. They may be drawn in various shapes, and the particular arrangement of lines around a given node can be semantically meaningful.
- The ends of a line do not have to be attached to any node. Such lines, where at least one of the ends is dangling, are called **external lines**, whereas lines that are attached to nodes on both ends are called **internal lines**.
- In Feynmann-like diagrams, including many-body diagrams, one of the axes is defined to be the so-called **time axis** and thus the orientation of the diagram can be semantically meaningful. Either the vertical (upward) or horizontal (leftward) axis may be chosen as the time axis depending on convention. We will use the vertical axis as the time axis. Particles

that are created later in time will appear higher in the diagram.

In many-body diagrams, nodes represent operators. Specifically, a node representing a k-body operator has k outgoing (creation) lines and k incoming (annihilation) lines, corresponding to a normal-ordered operator of the form

$$\frac{1}{k!^2} \sum_{p_1 \dots p_k q_1 \dots q_k} X_{p_1 \dots p_k q_1 \dots q_k} :(\hat{a}^\dagger_{p_1} \hat{a}_{q_1}) \dots (\hat{a}^\dagger_{p_k} \hat{a}_{q_k}):$$

There are two varieties of many-body diagrams, which render operators differently:

- In the **Brandow diagrams** (or **antisymmetrized Goldstone diagrams**) [Bra67], nodes are represented by k dots connected by dashed lines. Each dot has exactly one outgoing and one incoming line, thus a dot corresponds to a single creation-annihilation pair $\hat{a}^\dagger_p \hat{a}_q$. The ordering of the dots among themselves is insignificant as creation-annihilation pairs commute with each other within a normal-ordered product.
- In the **Hugenholtz diagrams** [Hug57], nodes are collapsed to a single dot with k outgoing and k incoming lines. Since it is no longer feasible to track the pairing between the outgoing and incoming lines, the sign of Hugenholtz diagrams is ambiguous. To transcribe Hugenholtz diagrams into equations with a definite sign, it is necessary to expand Hugenholtz diagrams to Brandow diagrams by pairing up the creation and annihilation operators in an arbitrary manner.

Lines represent variables that label single-particle states. The direction of a line tells us which end has the creation operator (tail) and which end has the annihilation operator (head). There are three kinds of lines:

- Internal lines (neither end is dangling): these represent Wick contractions and are always directed (have arrows). The orientation of their arrows with respect to the time axis is

significant:

  - If the direction of the arrow goes *along* the time axis (in our convention, if the arrow points up), then it represents a *particle state* a, b, c, … since its creation occurs before its annihilation.
  - If the direction of the arrow goes *against* the time axis (in our convention, if the arrow points down), then it represents a *hole state* i, j, k, … since its creation occurs after its annihilation.

- External lines where one of the ends is dangling: these represent the variables p, q, r, … of uncontracted field operators and their orientation with respect to the time axis is irrelevant. Outgoing lines (lines that leave the node) represent creation operators, and ingoing lines (lines that enter the node) represent annihilation operators. If all operators conserve particle number, then one can often elide the directions of external lines and have them inferred from context.
- External lines where both ends are dangling: this is a degenerate case. Such a line represents a Kronecker delta δ_{pq}, where p and q are the variables on each end of the line. The presence or absence of an arrow is irrelevant.

To illustrate the various parts of a diagram, consider this normal-ordered, partially contracted product $\hat{R}$,

$$\hat{R} = \frac{1}{8} \sum_{\substack{pqrs \\ i \backslash abc}} W_{ipqabc} F_{ai} \Gamma_{bcrs} : (\hat{a}_i^\dagger \hat{a}_a \hat{a}_p^\dagger \hat{a}_b \hat{a}_q^\dagger \hat{a}_c)(\hat{a}_a^\dagger \hat{a}_i)(\hat{a}_b^\dagger \hat{a}_r \hat{a}_c^\dagger \hat{a}_s) : \tag{2.9}$$

where $\hat{W}$ is a three-body operator, $\hat{F}$ is a one-body operator, and $\hat{\Gamma}$ is a two-body operator, all defined relative to the Fermi vacuum. The diagrammatic representation of this expression is shown in Fig. 2.1.

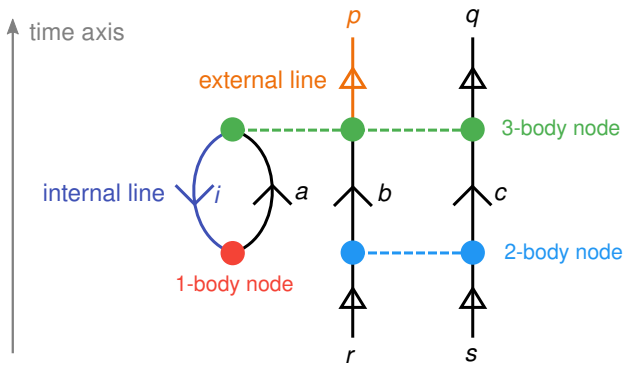

Figure 2.1: An example of a Brandow diagram representing to Eq. 2.9. We have intentionally labeled many parts of this diagram to provide a clear correspondence to the algebraic expression. To emphasize the distinction between internal and external lines, we have drawn the arrows of external lines with a different shape than those of internal lines.

The factor of 1/8 is the **weight** of the diagram. To obtain this number, we examine the symmetries in the Hugenholtz diagram, shown in Fig. 2.2. Observe that $\{p, q\}$ are topologically equivalent, and so are $\{b, c\}$ and $\{r, s\}$. Thus, the factor should be $1/(2! \times 2! \times 2!) = 1/8$.

The Brandow diagram makes it simple to compute the resultant *sign* (phase) of the expression in Eq. 2.9:

1. Pair up each outgoing external line with each incoming external line and connect them with a dotted line. The assignment is arbitrary, but once the choice is made, it fixes the ordering of the operators of the resultant expression.

   For example, if we pick (p, r) and (q, s) in Fig. 2.1, then the resultant expression will have the ordering $:\hat{a}_p^\dagger \hat{a}_r \hat{a}_q^\dagger \hat{a}_s:$.

2. Count (a) number of dotted lines d, (b) the number of internal hole lines h, and (c) the number of loops ℓ, including those that are completed by dotted lines. The sign is equal to

   $$(-)^{d+h+\ell}$$

   In the previous example, we have introduced two dotted lines connecting p to r and q to s.

There is one hole line, and a total three loops (two of which contain dotted lines). The sign is therefore positive.

This leads to the final result:

$$\hat{R} = +\frac{1}{8} \sum_{\substack{pqrs \\ i\backslash abc}} W_{ipqabc} F_{ai} \Gamma_{bcrs} : \hat{a}_p^\dagger \hat{a}_r \hat{a}_q^\dagger \hat{a}_s :$$

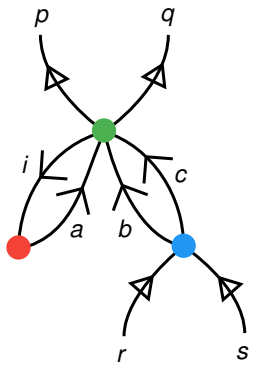

Figure 2.2: A Hugenholtz diagram representing to Eq. 2.9. This diagram is useful for determining the weight.

2.7.1 Perturbative diagrams

A variant of many-body diagrams is used in perturbation theory, which introduces an unusual kind of node called **resolvents** or **energy denominators**, drawn as a horizontal line that cuts across the diagram.

An example of such a diagram is shown on the right-hand side of Fig. 2.3. Note that to interpret such a diagram correctly, all incoming external lines must be **folded** upward, as shown on the right-hand side. For every denominator line, one divides the summation by the following Møller–Plesset denominator (see also Eq. 4.9):

$$\Delta_{q_1 \cdots q_k p_1 \cdots p_k} = \sum_{i=1}^{k} (\varepsilon_{q_i} - \varepsilon_{p_i})$$

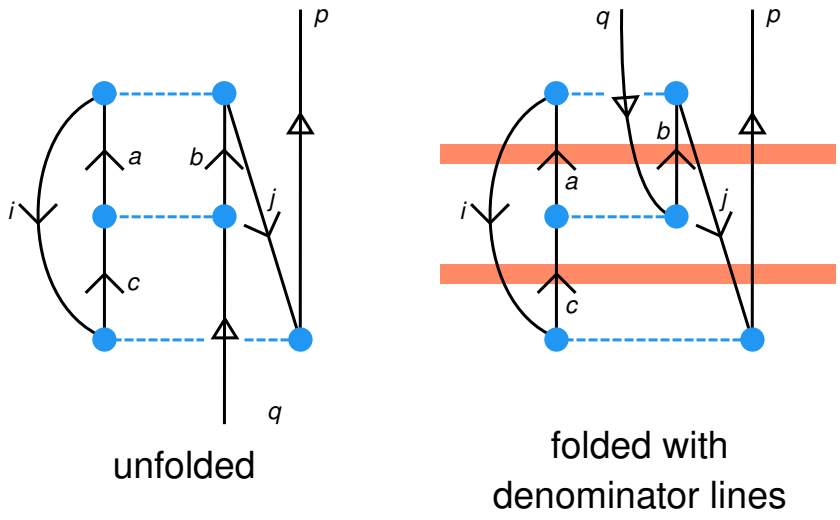

Figure 2.3: Interpretation of a perturbative Goldstone diagram

where $q_1 \ldots q_k$ are all downward lines that cut across the denominator line (including ingoing external lines), $p_1 \ldots p_k$ are all upward lines that cut across the denominator line (including outgoing external lines), and ε_p denotes the energy of the single-particle state p.

In the example, the upper denominator is given by

$$\Delta_{ijqabp} = \varepsilon_i + \varepsilon_j + \varepsilon_q - \varepsilon_a - \varepsilon_b - \varepsilon_p$$

whereas the lower denominator is given by

$$\Delta_{ijcp} = \varepsilon_i + \varepsilon_j - \varepsilon_c - \varepsilon_p$$

For presentation purposes, it is common to omit the denominator lines and to unfold a diagram back into the usual non-perturbative layout, as shown on the left-hand side of Fig. 2.3. To interpret this diagram perturbatively, simply refold the diagram and reinsert denominators between every pair of the $\hat{V}$ interaction nodes.

Chapter 3

Angular momentum coupling

We shall first discuss the details of angular momentum coupling in general, and then more specifically in the context of many-body theory. The objective of this chapter is to lay out the formalism (J-scheme, Sec. 3.11) that one needs to derive the angular-momentum-coupled equations in many-body theory, which are essential for efficient computations in spherically symmetric systems such as nuclei. We also include a brief discussion of our graphical angular momentum software (Sec. 3.10) used for derivations of angular momentum quantities.

3.1 Angular momentum and isospin

In classical physics, **orbital angular momentum** $\boldsymbol{L}$ is defined as $\boldsymbol{L} = \boldsymbol{r} \times \boldsymbol{p}$, where $\boldsymbol{r}$ is position and $\boldsymbol{p}$ is linear momentum. This definition carries over to quantum mechanics with the appropriate replacement of each quantity by their corresponding operator:

$$\hat{\boldsymbol{L}} = \hat{\boldsymbol{r}} \times \hat{\boldsymbol{p}} = -\mathrm{i}\hbar \hat{\boldsymbol{r}} \times \hat{\nabla}$$

In three-dimensional space, the standard eigenstates of orbital angular momentum are the spherical harmonics

$$|\ell m_\ell\rangle \leftrightarrow Y_{\ell m_\ell}(\theta, \varphi)$$

which are labeled by the orbital angular momentum magnitude ℓ and orbital angular momentum projection m_ℓ (also known as *magnetic quantum number* in chemistry), satisfying

$$\ell \in \mathbb{N} \qquad\qquad m_\ell \in M_\ell$$

where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of nonnegative integers and M_ℓ denotes the set

$$\{-\ell, -\ell + 1, -\ell + 2, \ldots, +\ell - 2, +\ell - 1, +\ell\} \tag{3.1}$$

The quantum numbers are related to eigenvalues of $\hat{\boldsymbol{L}}^2$ and $\hat{L}_3$ (z-component of $\hat{\boldsymbol{L}}$):

$$\hat{\boldsymbol{L}}^2 |\ell\, m_\ell\rangle = \hbar\ell(\ell + 1)|\ell\, m_\ell\rangle$$
$$\hat{L}_3 |\ell\, m_\ell\rangle = \hbar m_\ell |\ell\, m_\ell\rangle$$

Notice that these are not eigenstates of $\hat{L}_1$ or $\hat{L}_2$. It is impossible to find eigenstates of all three components, because $\hat{L}_1$, $\hat{L}_2$, and $\hat{L}_3$ do not commute with each other, as evidenced by the non-commuting nature of rotations in three-dimensional space. Instead, they satisfy the commutation relations,

$$[\hat{L}_i, \hat{L}_j] = \mathrm{i}\hbar \sum_{k=1}^{3} \epsilon_{ijk} \hat{L}_k$$

where ϵ_{ijk} is the Levi–Civita symbol.

In quantum mechanics, there is the addition of **spin** $\hat{\boldsymbol{S}}$, a kind of angular momentum intrinsic to each particle. There is an analogous set of standard eigenstates $|s\, m_s\rangle$ labeled by spin magnitude

s and spin projection m_s,

$$\hat{S}^2|sm_s\rangle = \hbar s(s+1)|sm_s\rangle$$

$$\hat{S}_3|sm_s\rangle = \hbar m_s|sm_s\rangle$$

Again, note that these are not eigenstates of $\hat{S}_1$ nor $\hat{S}_2$.

Unlike orbital angular momentum, the quantum numbers of spin are not confined to integers, but could be **half-integers** $\frac{1}{2}\mathbb{Z}$, given by

$$\frac{1}{2}\mathbb{Z} = \left\{ \dots, -\frac{3}{2}, -1, -\frac{1}{2}, 0, +\frac{1}{2}, +2, \frac{3}{2}, \dots \right\} \tag{3.2}$$

They are subject to the following conditions:

$$s \in \frac{1}{2}\mathbb{N} \qquad\qquad m_s \in M_s$$

where $\frac{1}{2}\mathbb{N}$ denotes the set of nonnegative half-integers:

$$\frac{1}{2}\mathbb{N} = \left\{ 0, \frac{1}{2}, 1, \frac{3}{2}, \dots \right\} \tag{3.3}$$

and M_s follows the same definition as Eq. 3.1, but with the argument s extended to support nonnegative integers,

$$M_s = \{-s, -s+1, -s+2, \dots, +s-2, +s-1, +s\} \tag{3.4}$$

Note that if s is a *half-odd* integer – a half-integer that is not also an integer – then M_s does not contain 0.

Spin states are not wave functions of position – they live within their own abstract Hilbert subspace. Most fermions studied in many-body theory, such as electrons or nucleons, are spin-$\frac{1}{2}$ particles, which means s is always $\frac{1}{2}$ and the dimension of this subspace is two. Conventionally, the spin operator for a spin-$\frac{1}{2}$ particle may be represented by a vector of Pauli matrices $\hat{\boldsymbol{\sigma}}$:

$$\hat{\boldsymbol{S}} = \frac{\hbar}{2}\hat{\boldsymbol{\sigma}} = \frac{\hbar}{2}\begin{bmatrix} \hat{\sigma}_1 \\ \hat{\sigma}_2 \\ \hat{\sigma}_3 \end{bmatrix}$$

Each Pauli matrix σ_i acts on the two-dimensional Hilbert subspace of spin.

The spin operator satisfies commutation relations similar to the orbital angular momentum operator,

$$[\hat{S}_i, \hat{S}_j] = \mathrm{i}\hbar \sum_{k=1}^{3} \epsilon_{ijk}\hat{S}_k$$

Spin can be combined with orbital angular momentum to form the **total angular momentum** of a particle,

$$\hat{\boldsymbol{J}} = \hat{\boldsymbol{L}} + \hat{\boldsymbol{S}}$$

Likewise, there is a standard set of total angular momentum eigenstates $|jm_j\rangle$, labeled by total angular momentum magnitude j and total angular momentum projection m_j, with completely

analogous relations,

$$\hat{\boldsymbol{J}}^2|jm_j\rangle = \hbar j(j+1)|jm_j\rangle$$

$$\hat{J}_3|jm_j\rangle = \hbar m_j|jm_j\rangle$$

$$[\hat{J}_i, \hat{J}_j] = \mathrm{i}\hbar \sum_{k=1}^{3} \epsilon_{ijk}\hat{J}_k$$

The quantum numbers are subject to the same constraints as for spin,

$$j \in \frac{1}{2}\mathbb{N} \qquad\qquad m_j \in M_j$$

Lastly, there is a mathematically similar quantity known as **isospin** $\hat{\boldsymbol{I}}$, which arises in the physics of nucleons. However, unlike spin, it is not physically considered as an angular momentum despite the confusing name. The isospin eigenstates may be denoted $|tm_t\rangle$, labeled by isospin magnitude t and isospin projection m_t, with relations just like angular momentum,

$$\hat{\boldsymbol{I}}^2|tm_t\rangle = t(t+1)|tm_t\rangle$$

$$\hat{I}_3|tm_t\rangle = m_t|tm_t\rangle$$

$$[\hat{I}_i, \hat{I}_j] = \mathrm{i} \sum_{k=1}^{3} \epsilon_{ijk}\hat{I}_k$$

and analogous constraints on the quantum numbers as for spin or total angular momentum.

Because isospin is not a kind of angular momentum, the three components of isospin are abstract and have no physical relation to the x-, y-, z-axes of space. They do not transform under spatial rotations.

For neutrons and protons, isospin is mathematically isomorphic to the spin of spin-$\frac{1}{2}$ particles, so $\hat{\boldsymbol{I}}$ too can be defined in terms of Pauli matrices. However, these *isospin* Pauli matrices act on

a different subspace from the *spin* Pauli matrices, so they are conventionally denoted using $\hat{\boldsymbol{\tau}}$ instead of $\hat{\boldsymbol{\sigma}}$:

$$\hat{\boldsymbol{I}} = \frac{1}{2}\hat{\boldsymbol{\tau}} = \frac{1}{2}\begin{bmatrix} \hat{\tau}_1 \\ \hat{\tau}_2 \\ \hat{\tau}_3 \end{bmatrix}$$

Each Pauli matrix $\hat{\tau}_i$ acts on the two-dimensional Hilbert subspace of isospin. The two eigenstates of isospin $|t = \frac{1}{2}, m_t = \pm\frac{1}{2}\rangle$ correspond to neutrons and protons, with two possible conventions:

- $m_t = -\frac{1}{2}$ corresponds to neutrons and $m_t = +\frac{1}{2}$ corresponds to protons (sometimes referred to as the *particle physics convention*)
- $m_t = +\frac{1}{2}$ corresponds to neutrons and $m_t = -\frac{1}{2}$ corresponds to protons (sometimes referred to as the *nuclear physics convention*)

Mathematically, the quantities $\hat{\boldsymbol{L}}$, $\hat{\boldsymbol{S}}$, $\hat{\boldsymbol{J}}$, and $\hat{\boldsymbol{I}}$ are all very similar. Specifically, these operators are representations of the $\mathfrak{su}(2)$ Lie algebra, which are generators of the special unitary group of two-dimensional matrices, SU(2). Elements of the Lie algebra are characterized by commutation relations of the form

$$[\hat{J}_i, \hat{J}_j] \propto \sum_{k=1}^{3} \epsilon_{ijk}\hat{J}_k$$

from which many of the familiar properties follow, including the structure of eigenstates and eigenvalues. The operator $\hat{J}^2$ is known as the *Casimir element* in this context, and commutes with each component $\hat{J}_i$.

Incidentally, the SU(2) group is *locally* similar to the special orthogonal group of 3-dimensional rotations, which means the corresponding $\mathfrak{so}(3)$ Lie algebra is completely isomorphic to $\mathfrak{su}(2)$.

This explains algebraic similarity between $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{S}}$ despite $\hat{\boldsymbol{L}}$ being generators of three-dimensional rotations.

3.2 Clebsch–Gordan coefficients

There are many situations in which angular momentum is added. The total angular momentum $\hat{\boldsymbol{J}} = \hat{\boldsymbol{L}} + \hat{\boldsymbol{S}}$ is one example. Another situation occurs in many-body systems, where the **composite angular momentum** of two (or more) particles is of interest,

$$\hat{\boldsymbol{J}}^{(1,2)} = \hat{\boldsymbol{J}}^{(1)} + \hat{\boldsymbol{J}}^{(2)}$$

where $\hat{\boldsymbol{J}}^{(1)}$ denotes the total angular momentum of particle 1, $\hat{\boldsymbol{J}}^{(2)}$ denotes the total angular momentum of particle 2, $\hat{\boldsymbol{J}}^{(1,2)}$ denotes the total angular momentum of both particles together.

Let us consider a general angular momentum-like quantity defined as

$$\hat{\boldsymbol{J}} = \hat{\boldsymbol{L}} + \hat{\boldsymbol{S}}$$

with $\hat{\boldsymbol{L}}$ and $\hat{\boldsymbol{S}}$ being angular momentum-like quantities as well. The discussion in this section is abstract: we do not assign physical interpretations to these quantities. They simply need to be representations of the $\mathfrak{su}(2)$ Lie algebra satisfying the usual commutation relations. (In particular, $\hat{\boldsymbol{L}}$ need not be restricted to orbital angular momentum.)

Observe that $\hat{\boldsymbol{J}}^2$ commutes with $\hat{\boldsymbol{L}}^2$ and $\hat{\boldsymbol{S}}^2$, but does not commute with $\hat{L}_3$ nor $\hat{S}_3$, because

$$\hat{\boldsymbol{J}}^2 = \hat{\boldsymbol{L}}^2 + 2\hat{\boldsymbol{L}} \cdot \hat{\boldsymbol{S}} + \hat{\boldsymbol{S}}^2$$

The term $2\hat{\boldsymbol{L}} \cdot \hat{\boldsymbol{S}}$ does not commute with $\hat{L}_3$ nor $\hat{S}_3$. This means we can choose the eigenstates of

$(\hat{\boldsymbol{J}}^2, \hat{J}_3)$ to be eigenstates of $\hat{\boldsymbol{L}}^2$ and $\hat{\boldsymbol{S}}^2$ as well, but they cannot not in general be eigenstates of $\hat{J}_3$ nor $\hat{S}_3$. We may label such a state as

$$|jm_j\ell s\rangle$$

This is known in general as a **coupled state**, and if $\hat{\boldsymbol{L}}$ is orbital angular momentum and $\hat{\boldsymbol{S}}$ is spin, then this particular example would be referred to as *LS coupling*. Such states have the following eigenvalues,

$$\hat{\boldsymbol{J}}^2|jm_j\ell s\rangle = \hbar j(j+1)|jm_j\ell s\rangle$$

$$\hat{J}_3|jm_j\ell s\rangle = \hbar m_j|jm_j\ell s\rangle$$

$$\hat{\boldsymbol{L}}^2|jm_j\ell s\rangle = \hbar\ell(\ell+1)|jm_j\ell s\rangle$$

$$\hat{\boldsymbol{S}}^2|jm_j\ell s\rangle = \hbar s(s+1)|jm_j\ell s\rangle$$

In contrast, if we want an eigenstate of $(\hat{J}_3, \hat{\boldsymbol{L}}^2, \hat{L}_3, \hat{\boldsymbol{S}}^2, \hat{S}_3)$, then we can simply construct it as a tensor product of the standard eigenstates of $(\hat{\boldsymbol{L}}^2, \hat{L}_3)$ and $(\hat{\boldsymbol{S}}^2, \hat{S}_3)$, denoted

$$|\ell m_\ell s m_s\rangle = |\ell m_\ell\rangle \otimes |s m_s\rangle$$

with eigenvalues

$$\hat{J}_3|\ell\, m_\ell\, s m_s\rangle = \hbar(m_\ell + m_s)|\ell\, m_\ell\, s m_s\rangle$$

$$\hat{\boldsymbol{L}}^2|\ell\, m_\ell\, s m_s\rangle = \hbar\ell(\ell+1)|\ell\, m_\ell\, s m_s\rangle$$

$$\hat{L}_3|\ell\, m_\ell\, s m_s\rangle = \hbar m_\ell|\ell\, m_\ell\, s m_s\rangle$$

$$\hat{\boldsymbol{S}}^2|\ell\, m_\ell\, s m_s\rangle = \hbar s(s+1)|\ell\, m_\ell\, s m_s\rangle$$

$$\hat{S}_3|\ell\, m_\ell\, s m_s\rangle = \hbar m_s|\ell\, m_\ell\, s m_s\rangle$$

These are known as **uncoupled** states.

The two sets of eigenstates are related by a linear transformation,

$$|j m_j \ell\, s\rangle = \sum_{m_\ell \in M_\ell, m_s \in M_s} |\ell\, m_\ell\, s m_s\rangle\langle \ell\, m_\ell\, s m_s | j m_j\rangle \tag{3.5}$$

where $\langle \ell\, m_\ell\, s m_s | j m_j\rangle$ denotes a set of coefficients known as the **Clebsch–Gordan (CG) coefficients**.

The CG coefficients are subject to a set of constraints – **selection rules** – that we divide into two categories. First, there are the **local selection rules**:

$$\ell, s, j \in \frac{1}{2}\mathbb{N} \qquad m_\ell \in M_\ell \qquad m_s \in M_s \qquad m_j \in M_j$$

These kinds of constraints are intrinsic to each (j, m)-pair and are omnipresent in angular momentum algebra. Thus, to avoid having to repeat ourselves, we will implicitly assume they are satisfied in all equations.

The pairing between the magnitude (j-like) and projection (m-like) variables is established by a notational convention: the subscript on the m-like variable is used to to link to the j-like variable.

For example, m_j pairs with j, and m_ℓ pairs with ℓ. If the j variable itself also has a subscript, then we will typically use the subscript of j directly as a subscript of m, e.g. m_1 pairs with j_1, m_p pairs with j_p, etc.

The second category are the **nonlocal selection rules**, consisting of

$$m_j = m_\ell + m_s$$

which simply states the additive nature of projections, and

$$|\ell - s| \leq j \leq \ell + s$$

This latter constraint is called the **triangle condition** and is equivalent to the geometrical constraint that ℓ, s, and j are lengths of a triangle. It can alternatively be restated as the symmetric combination of constraints,

$$j \leq \ell + s \qquad \ell \leq s + j \qquad s \leq j + \ell$$

It is convenient to define the CG coefficients such that if any of the selection rules are violated, then the CG coefficient is zero. This allows us to omit the constraints on the summation of m_ℓ and m_s in Eq. 3.5 and simply let the selection rule pick the terms that contribute.

In principle, there can be several conventions for CG coefficients. The obvious choice is to limit ourselves to only *real* coefficients, but there remains an arbitrary choice in sign, which is then fixed by the Condon–Shortley phase convention.

In this choice, the linear transformation of CG coefficients is completely symmetric, so we can

use the same coefficients to undo the coupling:

$$|\ell m_\ell s m_s\rangle = \sum_{jm_j} |jm_j \ell s\rangle\langle \ell m_\ell s m_s | jm_j\rangle$$

where summation over j and m_j is constrained by the selection rules of CG.

Hence, the CG coefficients satisfy the orthogonality relations,

$$\sum_{jm_j} \langle \ell m_\ell s m_s | jm_j\rangle\langle \ell m'_\ell s m'_s | jm_j\rangle = \delta_{m_\ell m_\ell}\delta_{m_s m'_s}$$

$$\sum_{m_\ell m_s} \langle \ell m_\ell s m_s | jm_j\rangle\langle \ell m_\ell s m_s | j' m'_j\rangle = \delta_{jj'}\delta_{m_j m'_j}\begin{Bmatrix} \ell & s & j \end{Bmatrix}$$

where $\begin{Bmatrix} \ell & s & j \end{Bmatrix}$ is the **triangular delta**, defined as

$$\begin{Bmatrix} a & b & c \end{Bmatrix} = \begin{cases} 1 & \text{if } |a-b| \le c \le a+b \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

In other words, the triangular delta is the analog of Kronecker delta for the triangle condition.

There are additional symmetry properties of CG coefficients, but it is more convenient to state them indirectly through a similar quantity known as the Wigner 3-jm symbol.

3.3 Wigner 3-jm symbol

The **Wigner 3-jm symbol** is a function of six arguments used for coupling angular momenta with a high degree of symmetry, denoted[1]

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$$

The j_i arguments could be any nonnegative half-integer $\frac{1}{2}\mathbb{N}$ (Eq. 3.3), including both integers and half-odd integers. The m_i arguments are constrained to the M_{j_i} set as defined in Eq. 3.4. These form the local selection rules. The nonlocal selection rules are given by

$$|j_1 - j_2| \le j_3 \le j_1 + j_2 \qquad\qquad m_1 + m_2 + m_3 = 0$$

The 3-jm symbol is related to the Clebsch–Gordan coefficient by the formula,

$$\langle j_1 m_1 j_2 m_2 | j_{12} m_{12} \rangle = (-)^{2j_2 + j_{12} - m_{12}} \breve{j}_{12} \begin{pmatrix} j_1 & j_{12} & j_2 \\ m_1 & -m_{12} & m_2 \end{pmatrix} \tag{3.7}$$

where we introduce the shorthand

$$\breve{j} = \sqrt{2j + 1} \tag{3.8}$$

as the factor appears frequently in angular momentum algebra.

[1] Unfortunately this shares the same notation as 2×3 matrices.

When three angular momentum states are coupled using the 3-jm symbol,

$$\sum_{m_1 m_2 m_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} |j_1 m_1\rangle \otimes |j_2 m_2\rangle \otimes |j_3 m_3\rangle$$

the result is invariant (a spherical scalar) under SU(2) transformations.

The 3-jm symbol is given by the following symmetric formula [Wig93]

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = \sqrt{\Delta(j_1\, j_2\, j_3)} \sum_{k_1 k_2 k_3} \prod_{i=1}^{3} \frac{(-)^{j_1/2+k_i} \sqrt{(j_i - m_i)!(j_i + m_i)!}}{(J/2 - j_i - k_i)!(J/2 - j_i + k_i)!}$$

where:

- $J = j_1 + j_2 + j_3$
- The summation is performed over all half-integers k_i subject to the following constraints:
  1. $m_1 + k_2 - k_3 = m_2 + k_3 - k_1 = m_3 + k_1 - k_2 = 0$
  2. Argument of every factorial involving k_i must be a nonnegative integer.
- $\Delta(j_1\, j_2\, j_3)$ is the **triangle coefficient**:

$$\Delta(j_1\, j_2\, j_3) = \frac{\prod_{i=1}^{3}(J - 2j_i)!}{(J+1)!}$$

The summation over k_i has only one effective degree of freedom. If we break the symmetry by choosing

$$k = \frac{J}{2} - j_3 - k_3$$

we obtain the more conventional form used by Racah [Rac42]

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$$
$$= \delta_{m_1+m_2+m_3,0}(-)^{j_1-j_2-m_3}\sqrt{\Delta(j_1\,j_2\,j_3)\prod_{i=1}^{3}(j_i+m_i)!(j_i-m_i)!}\sum_k$$
$$\frac{(-1)^k}{k!(j_1+j_2-j_3-k)!(j_1-m_1-k)!(j_2+m_2-k)!}$$
$$\times\frac{1}{(j_3-j_2+m_1+k)!(j_3-j_1-m_2+k)!}$$

The summation is performed over all half-integers k such that the argument of every factorial involving k is a nonnegative integer.

The 3-jm symbol is invariant under even permutations of its columns,

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = \begin{pmatrix} j_2 & j_3 & j_1 \\ m_2 & m_3 & m_1 \end{pmatrix} = \begin{pmatrix} j_3 & j_1 & j_2 \\ m_3 & m_1 & m_2 \end{pmatrix}$$

Odd permutations lead to a phase factor,

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-)^{j_1+j_2+j_3}\begin{pmatrix} j_3 & j_2 & j_1 \\ m_3 & m_2 & m_1 \end{pmatrix}$$

The same phase factor arises from time reversal,

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-)^{j_1+j_2+j_3}\begin{pmatrix} j_1 & j_2 & j_3 \\ -m_1 & -m_2 & -m_3 \end{pmatrix}$$

The 3-jm symbol also has an additional set of symmetries called Regge symmetries [Reg58], but

these are seldomly used in angular momentum algebra. They are, however, useful for storage and caching of 3-jm symbols in computations [RY04].

The orthogonality relations of CG coefficients carry over to 3-jm symbols. The first orthogonality relation is given by

$$\sum_{j_3 m_3} \check{j}_3^2 \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1' & m_2' & m_3 \end{pmatrix} = \delta_{m_1 m_1'} \delta_{m_2 m_2'} \tag{3.9}$$

while the second orthogonality relation is

$$\sum_{m_2 m_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} j_2 & j_3 & j_4 \\ m_2 & m_3 & m_4 \end{pmatrix} = \frac{\delta_{j_1 j_4} \delta_{m_1 m_4}}{\check{j}_1^2} \begin{Bmatrix} j_1 & j_2 & j_3 \end{Bmatrix} \tag{3.10}$$

In the case where one of the angular momenta is zero, the 3-jm symbol has a very simple formula:

$$(-)^{j_1 - m_1} \begin{pmatrix} j_1 & 0 & j_2 \\ -m_1 & 0 & m_2 \end{pmatrix} = \frac{\delta_{j_1 j_2} \delta_{m_1 m_2}}{\check{j}_1} \tag{3.11}$$

There is a special relation that converts a summation over a 3-jm symbol into Kronecker deltas,

$$\sum_{m_2} (-)^{j_2 - m_2} \begin{pmatrix} j_2 & j_1 & j_2 \\ -m_2 & m_1 & m_2 \end{pmatrix} = \delta_{j_1 0} \delta_{m_1 0} \check{j}_2$$

Read in reverse, this means one can also represent Kronecker deltas with zeros as a 3-jm symbol.

3.4 Angular momentum diagrams

Angular momentum diagrams, originally introduced by Jucys (whose name is also translated as Yutsis) [YLV62], provide a graphical way to manipulate expressions of coupling coefficients of angular momentum states. Our presentation of diagrams differs from [YLV62; WP06; BL09] in the treatment of arrows.[2] In other literature, arrows are used to distinguish between covariant and contravariant angular momenta. However, we treat them mechanically as 1-jm symbols (see Sec. 3.4.3). Other differences in presentation are largely superficial. For practical reasons we do not use graphics to describe $\sqrt{2j+1}$ factors unlike [BL09].

3.4.1 Nodes

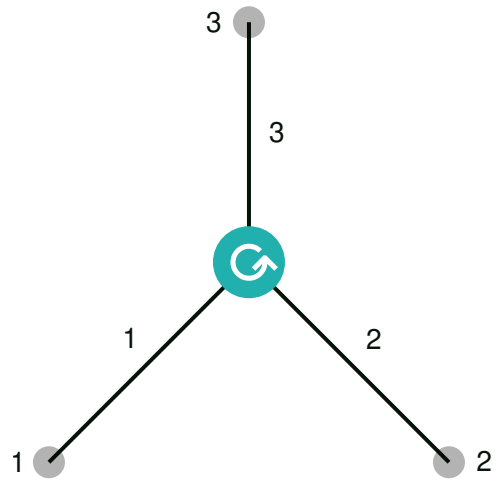

Figure 3.1: Diagram of the 3-jm symbol (123) in Eq. 3.12

The main ingredient of angular momentum algebra are 3-jm symbols. Since they are functions of six variables, one might be tempted to introduce a node with six lines emanating from it.

[2] Conversion from our presentation to the traditional presentation in, say, [WP06] is done by a two-step process: (1) use diagrammatic rules to ensure that every internal line has an arrow and that every arrow on external lines (if any) point *away* from the terminal; (2) on any remaining external lines with no arrows, draw an arrow pointing toward the terminal. Now the diagram can be interpreted in the traditional manner. To convert back, simply revert step (2): delete all arrows on external lines that point toward the terminal.

However, this quickly becomes unwieldy. Instead, it is better to treat each (j, m)-pair as a combined entity.

In Fig. 3.1, we introduce the diagram for the 3-jm symbol,

$$(123) = \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \tag{3.12}$$

for which we have assigned the shorthand (123). Note that 3-jm symbols are the only kind of primitive **node** (vertex) that appears in angular momentum diagrams. Hence, they are the basic building block of such diagrams.

Because 3-jm symbols are invariant under *even* permutations only, it is necessary to assign a definite ordering to the lines. This is denoted by the circular arrow within the node. In other literature, circular arrows are usually replaced by a sign: + for anticlockwise and − for clockwise.

3.4.2 Lines

The **lines** (edges) in angular momentum diagrams serve to link the m-type arguments on both ends of the line. The domain M_j over which the m variable is valid is indicated by the label on the line.[3] For example, if a line is labeled "1", this means the m variable must lie within the domain M_{j_1} of the j_1 variable.

As a convenience (or perhaps a source of confusion), we introduce a special exception to this interpretation when the label is "0". In this case, we instead interpret it to indicate that the domain is $M_0 = \{0\}$, i.e. $m = j = 0$. To alert the reader of this special interpretation, the line is drawn in a faded grey color.

Lines can appear in isolation, as shown in Fig. 3.2. The middle diagram of Fig. 3.2 represents

[3] Because of this, unlike many-body diagrams, labels on lines are *not* optional.

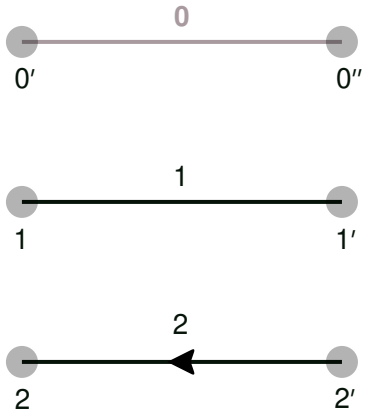

Figure 3.2: Degenerate line diagrams: upper diagram: $(0'0'')$ in Eq. 3.14; middle diagram: $(11')$ in Eq. 3.13; lower diagram: $(\check{2}2')$ in Eq. 3.15

the Kronecker delta,

$$(11') = \delta_{m_1 m_1'} \tag{3.13}$$

The upper diagram is also a Kronecker delta, but with the extra constraint that $m_0' \in M_0$, hence

$$(0'0'') = \delta_{m_0' m_0''} \delta_{m_0' 0} \tag{3.14}$$

3.4.3 Herring–Wigner 1-jm symbol

In the lower diagram of Fig. 3.2, we introduce the notion of an **arrow** on a line. Lines with arrows (*directed* lines) are associated with a $(-)^{j-m}$ phase as well as a sign reversal in m, i.e. the **time-reversal** of angular momentum. More precisely, the diagram represents the quantity:

$$(\check{2}2') = \delta_{m_2,-m_2'} (-)^{j_2 - m_2'} \tag{3.15}$$

This is sometimes referred to as a **Herring–Wigner 1-jm symbol**, denoted by

$$\begin{pmatrix} & j & \\ m & & m' \end{pmatrix} = \check{j} \begin{pmatrix} j & 0 & j \\ m & 0 & m' \end{pmatrix} = \delta_{m,-m'}(-)^{j-m'} \tag{3.16}$$

It acts like a metric tensor for SU(2), since the quantity

$$\sum_{mm'} \begin{pmatrix} & j & \\ m & & m' \end{pmatrix} |jm\rangle \otimes |jm'\rangle$$

is invariant under SU(2) transformations.

3.4.4 Terminals

The **terminals** of lines, highlighted by the grey circles, represent the free m variables (i.e. those that are not summed over). We label the terminals so as to provide a correspondence to the algebraic equations. For example, in Fig. 3.1 we label the terminals "1", "2", and "3" to indicate their correspondence to m_1, m_2, and m_3.

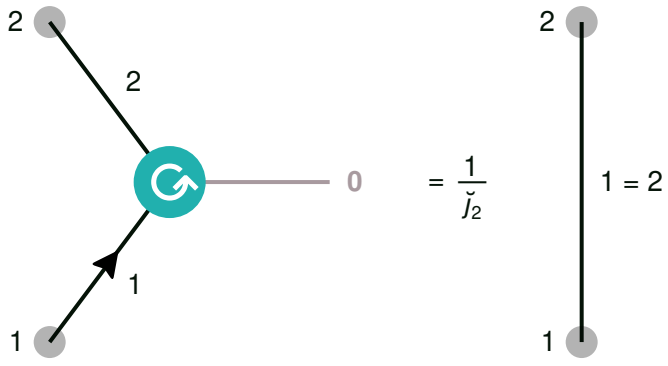

Figure 3.3: 3-jm symbol when an argument is zero: $(\check{1}02) = (12)/\check{j}_1$ in Eq. 3.11

There is one exceptional situation where a terminal may be absent: when the line is zero. This occurs in, for example, the simplification of a 3-jm symbol when one of the angular momenta

is zero in Eq. 3.11. This is depicted diagrammatically in Fig. 3.3, and shown here in shorthand notation:

$$(\check{1}02) = \frac{(12)}{\check{j}_1}$$

3.4.5 Closed diagrams

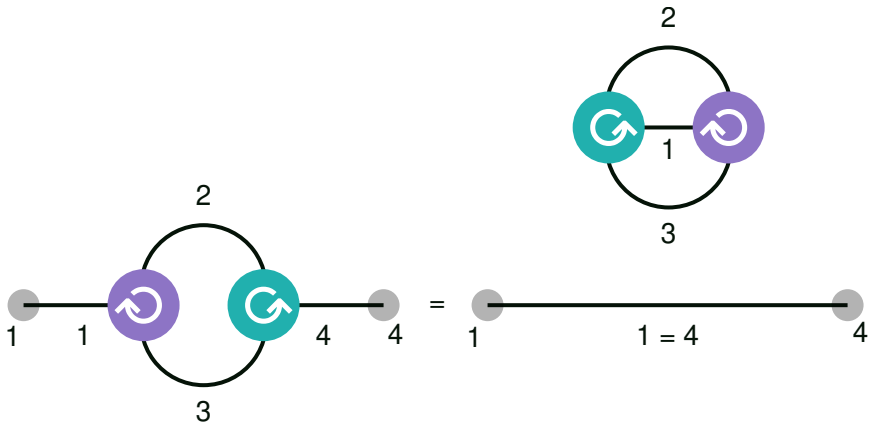

Figure 3.4: Second orthogonality relation for 3-jm symbols: $(123)(234) = (14)(1'23)(1'23)$ in Eq. 3.10

In lines with no terminals – the *internal* lines – their m variables are always summed over. This is exemplified in Fig. 3.4, which depicts the second orthogonality relation for 3-jm symbols in Eq. 3.10 as

$$(123)(234) = (14)(1'23)(1'23) = (14)\{1'23\}$$

On the left-hand side, the m_2 and m_3 lines are both internal and therefore summed over. On the right-hand side, 1 = 4 label indicates the presence of a j-relating Kronecker delta $\delta_{j_1 j_4}$ in addition to the usual $\delta_{m_1 m_4}$. In the upper right, there is a special subdiagram $(1'23)(1'23)$, which is in fact the triangular delta $\begin{Bmatrix} j_1 & j_2 & j_3 \end{Bmatrix}$ as defined in Eq. 3.6.

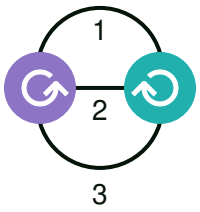

Figure 3.5: Triangular delta: $\{123\} = (123)(123)$ in Eq. 3.17

In diagrammatic notation, the triangular delta is represented by the following sum as shown in Fig. 3.5:

$$\{123\} = (123)(123) = \sum_{m_1 m_2 m_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}^2 = \begin{Bmatrix} j_1 & j_2 & j_3 \end{Bmatrix} \tag{3.17}$$

The triangular delta is the simplest **irreducible closed diagram**: it cannot be broken down into simpler components in a nontrivial way (*irreducible*), and there are no free m-type variables (*closed*). Specifically, we say a diagram is irreducible if it cannot be factorized into subdiagrams without either (a) introducing a summation over a new j-type variable, or (b) introducing another triangular delta. The (b) constraint comes from the fact that a triangular delta can be split (factorized) into a finite number of identical triangular deltas, which is not very interesting.

The triangular delta is a rather degenerate case of an irreducible closed diagram. In Secs. 3.8.2, 3.8.3, we introduce more interesting cases: the 6-j and 9-j symbols. In graph theory, irreducible closed diagrams correspond to cubic graphs that are *cyclically 4-connected*, namely, 3-edge-connected graphs in which every split by the deletion of 3 edges yields at least one disconnected vertex. The triangular delta is unusual in that it is the only non-simple graph of this family.

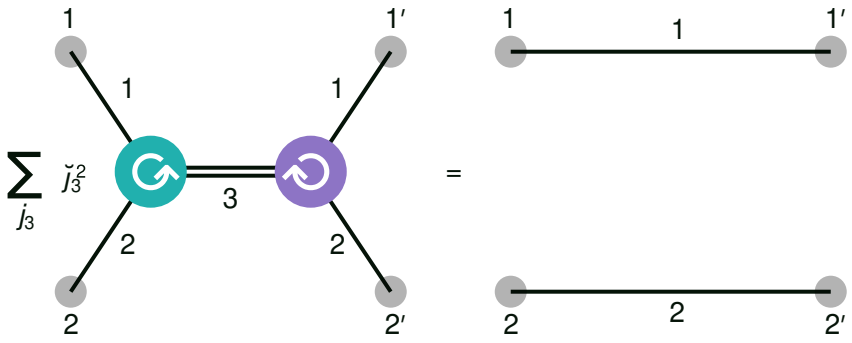

Figure 3.6: First orthogonality relation for 3-jm symbols: $\sum_{j_3} \check{j}_3^2(123)(1'2'3) = (11')(22')$ in Eq. 3.9

3.4.6 Summed lines

There are a few occasions where the j-type variables do need to be summed over. This occurs in the first orthogonality relation in Eq. 3.9, shown diagrammatically in Fig. 3.6 as

$$\sum_{j_3} \check{j}_3^2(123)(1'2'3) = (11')(22')$$

The doubling of the line serves as an additional reminder that j_3 is being summed over.

3.5 Phase rules

Let us begin by considering just one particular angular momentum pair (j_i, m_i) in isolation. In this case, we have the properties

$$(-)^{4j_i} = 1 \qquad\qquad (-)^{2(j_i - m_i)} = 1$$

We may call this the **local phase rules**. Given an arbitrary phase $(-)^{aj_i+bm_i}$ with $a, b \in \mathbb{Z}$, one can always use local rules to *uniquely* decompose the phase as

$$(-)^{aj_i+bm_i} = (-)^{cj_i+d(j_i-m_i)}$$

where $c \in \{0, 1, 2, 3\}$ and $d \in \{0, 1\}$. We will call this the **locally canonical** form of the phase. Hence, phases of a single angular momentum may be represented as a pair (c, d) where c uses modulo-4 arithmetic and d uses modulo-2 arithmetic. There are only 8 unique phases:

	$0j$	$1j$	$2j$	$3j$
$0(j-m)$	0	$+j$	$2j$	$-j$
$1(j-m)$	$j-m$	$+m$	$j+m$	$-m$

The table has a toroidal topology: it wraps around both horizontally and vertically.

Canonicalization provides a mechanical approach for deciding whether two phases are equivalent. Unfortunately, when non-local rules are involved, there is no longer an obvious way to canonicalize phases – the symmetries of the phases become entangled with the topology of the angular momentum diagram. Nonetheless, local canonicalization provides an easy way to eliminate one of the sources of redundancy.

It is common to work with only real recoupling coefficients, thus it is unusual for $(-)^j$ or $(-)^{3j}$ to appear in isolation. They typically appear in groups, such as triplets $(-)^{j_1+j_2+j_3}$ or quadruplets.

Note that the $(-)^{j_i-m_i}$ phase comes from the 1-jm symbol, which are arrows in diagrammatic notation (see Eq. 3.16). The algebraic properties of the 1-jm symbol can be encoded as two arrow

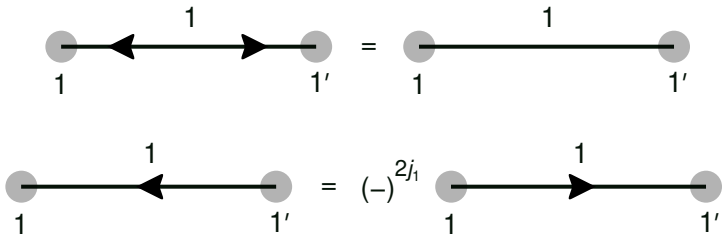

Figure 3.7: Upper diagram: arrow cancellation: $(\check{1}\check{1}')$ = $(11')$ in Eq. 3.18; lower diagram: arrow reversal: $(\check{1}1')$ = $(-)^{2j_1}(1\check{1})$ in Eq. 3.19

rules in diagrammatic theory. The first rule is **arrow cancellation**:

$$\sum_{m_1''} \begin{pmatrix} j & \\ m & m'' \end{pmatrix} \begin{pmatrix} j & \\ m' & m'' \end{pmatrix} = \delta_{m,m'} \tag{3.18}$$

which is depicted in the upper diagram of Fig. 3.7. In the lower diagram, we have the second rule of **arrow reversal**:

$$\begin{pmatrix} j & \\ m & m' \end{pmatrix} = (-)^{2j} \begin{pmatrix} j & \\ m' & m \end{pmatrix} \tag{3.19}$$

Now let us consider the **nonlocal phase rules**, which govern phase triplets related by 3-jm symbols. These arise from the properties of the 3-jm symbol and the selection rules.

One of the nonlocal selection rules of the 3-jm symbol is

$$m_1 + m_2 + m_3 = 0$$

This implies that m_1, m_2, and m_3 are either all integers or one of them is an integer and the rest are half-odd integers. The j_1, j_2, and j_3 variables are constrained by this same condition as a

consequence. Thus we have the **sweeping rule**:

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-)^{2j_1+2j_2+2j_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix}$$

This rule enables $(-)^{2j}$-type phases to be introduced, eliminated, or migrated ("sweeped") around the diagram. In contrast, $(-)^{j}$-type phases by themselves are generally immobile without the aid of Kronecker deltas.

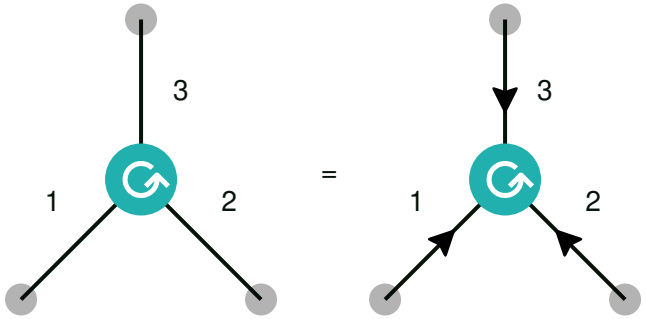

Figure 3.8: Triple arrow rule: (123) = ($\check{1}\check{2}\check{3}$) in Eq. 3.20

The analog of the sweeping rule for arrows is the **triple arrow rule**, which allows three similar arrows to be introduced around any 3-jm node:

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-)^{j_1-m_1+j_2-m_2+j_3-m_3} \begin{pmatrix} j_1 & j_2 & j_3 \\ -m_1 & -m_2 & -m_3 \end{pmatrix} \tag{3.20}$$

Like the sweeping rule, these can be used introduce, eliminate, or migrate arrows around the diagram.

Lastly, it is often necessary to reverse the order of arguments in a 3-jm symbol. This is handled by the **node reversal rule**, which allows the orientation of a 3-jm symbol to be reversed at the

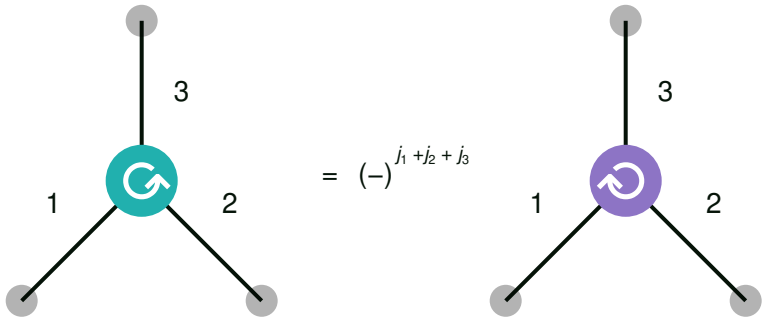

Figure 3.9: Node reversal rule: (123) = $(-)^{j_1+j_2+j_3}$(321) in Eq. 3.21

cost of three $(-)^j$-type phases:

$$\begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} = (-)^{j_1+j_2+j_3} \begin{pmatrix} j_3 & j_2 & j_1 \\ m_3 & m_2 & m_1 \end{pmatrix} \tag{3.21}$$

3.6 Wigner–Eckart theorem

The Wigner–Eckart theorem allows matrix elements of a spherical tensor operator to be factorized into an operator-dependent, m-independent component and an operator-independent, m-dependent factor. The latter factor is composed of a 3-jm symbol (or equivalently a CG coefficient). The theorem is highly advantageous for numerical computations as summations over 3-jm symbols can often be simplified substantially.

The usual statement of the theorem is as follows: if $\hat{T}^{j_T}_{m_T}$ is a rank-j_T spherical tensor operator with components labeled by m_T, then its matrix elements can be factorized in the following manner

$$\langle j_1 m_1 \alpha_1 | \hat{T}^{j_T}_{m_T} | j_2 m_2 \alpha_2 \rangle = (-)^{j_1 - m_1} \begin{pmatrix} j_1 & j_T & j_2 \\ -m_1 & m_T & m_2 \end{pmatrix} \langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle$$

where $\langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle$ is called the **reduced matrix element under the 3-jm convention**. This

is the same convention as the one used in [Rac42].

There are several other conventions. Some conventions differ by a factor of $(-)^{2j_T}$:

$$\begin{aligned}\langle j_1 m_1 \alpha_1 | \hat{T}^{j_T}_{m_T} | j_2 m_2 \alpha_2 \rangle &= (-)^{2j_T + j_1 - m_1} \begin{pmatrix} j_1 & j_T & j_2 \\ -m_1 & m_T & m_2 \end{pmatrix} \langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle' \\ &= \frac{1}{\breve{j}_1} \langle j_2 m_2 j_T m_T | j_1 m_1 \rangle \langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle'\end{aligned}$$

This phase factor is often irrelevant as j_T is commonly an integer. Another convention is to simply use the CG coefficient directly:

$$\langle j_1 m_1 \alpha_1 | \hat{T}^{j_T}_{m_T} | j_2 m_2 \alpha_2 \rangle = \langle j_2 m_2 j_T m_T | j_1 m_1 \rangle \langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle''$$

We call $\langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle''$ the **reduced matrix element under the CG convention**. This convention is convenient for scalar operators where it simplifies to:

$$\langle j_1 m_1 \alpha_1 | \hat{T}^0_0 | j_2 m_2 \alpha_2 \rangle = \delta_{j_1 j_2} \delta_{m_1 m_2} \langle j_1 \alpha_1 \| \hat{T}^0 \| j_2 \alpha_2 \rangle''$$

An unusual way to state the Wigner–Eckart theorem is through the following inverse equation:

$$\langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle = \sum_{m_1' m_T m_2'} (-)^{j_1 - m_1'} \begin{pmatrix} j_1 & j_T & j_2 \\ -m_1' & m_T & m_2' \end{pmatrix} \langle j_1 m_1' \alpha_1 | \hat{T}^{j_T}_{m_T} | j_2 m_2' \alpha_2 \rangle$$

The advantage of this form is that it can be readily translated to diagrams. Of course, in practice

the summation is unnecessary as one could simply compute:

$$\langle j_1 \alpha_1 \| \hat{T}^{j_T} \| j_2 \alpha_2 \rangle = \frac{(-)^{j_1} \langle j_1 0 \alpha_1 | \hat{T}_0^{j_T} | j_2 0 \alpha_2 \rangle}{\begin{pmatrix} j_1 & j_T & j_2 \\ 0 & 0 & 0 \end{pmatrix}}$$

3.7 Separation rules

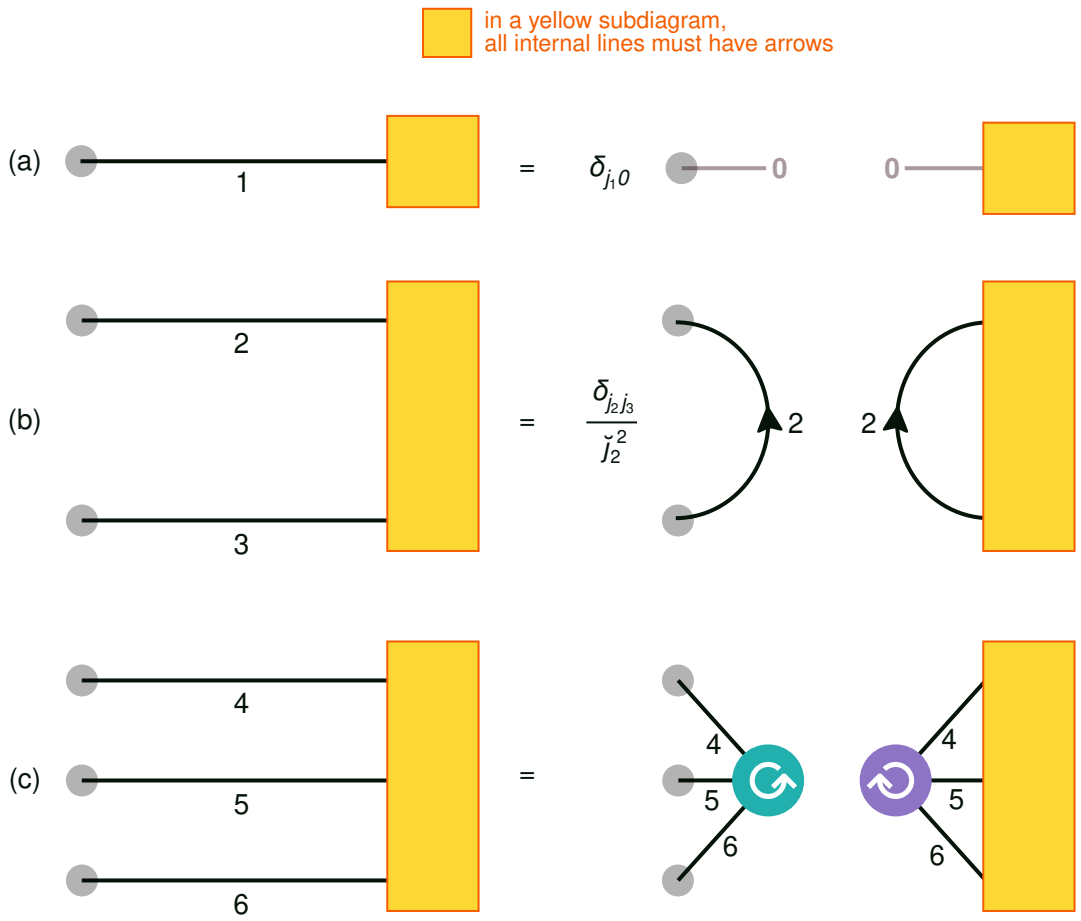

Figure 3.10: Separation rules: (a) single-line separation rule: $f(j_1, m_1) = \delta_{j_1 0} \delta_{m_1 0} f(0, 0)$ in Eq. 3.22; (b) double-line separation rule; (c) triple-line separation rule.

There is a general diagrammatic rule that is closely related to the more specialized Wigner–Eckart theorem. Suppose we have an angular momentum diagram $f(j_1, m_1)$ composed of 3-jm

nodes with exactly one external line and every one of its internal lines has an arrow. Then, we can partition the diagram into two pieces:

$$f(j_1, m_1) = \delta_{j_1 0}\delta_{m_1 0} f(0, 0) \tag{3.22}$$

In other words, f must be invariant (a spherical scalar). We call this the **single-line separation rule** because it allows us to cut the lone external line to separate the diagram into two disconnected pieces. This rule is shown in Fig. 3.10 (a).

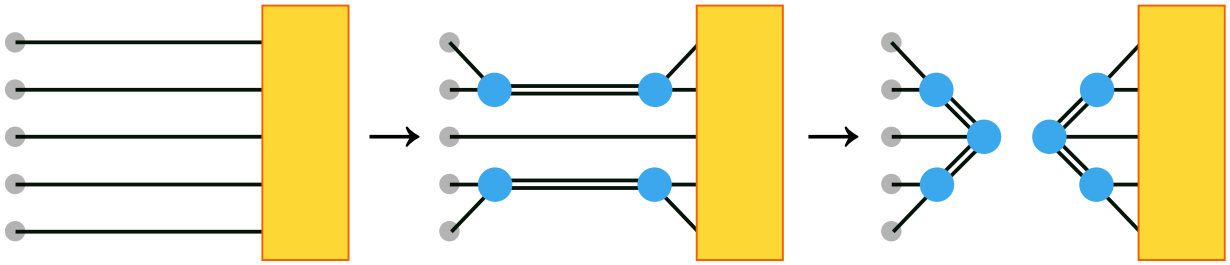

Figure 3.11: A schematic derivation of the separation rule for five lines. The topologies of the diagrams are shown but most details (such as phases or other factors) have been omitted. Double lines indicate summed lines as before (Sec. 3.4.6). The meaning of the yellow rectangles is the same as in Fig. 3.10.

This seemingly simple rule can be used to separate arbitrarily complicated diagrams through a mechanical process (Fig. 3.11) in which lines are repeatedly pairwise combined using the first orthogonality relation (Fig. 3.6) until a single line remains, which can then be cut using Eq. 3.22.

Separation rules for the special cases of two and three lines are shown in Fig. 3.10 (b) and (c) respectively. Both can be derived using (a) and the first orthogonality relation. Analogous separation rules for four or more lines can be derived, but they always introduce new angular momentum variables to be summed over. With six or more lines, there can be multiple non-equivalent separation rules.

Separation rules are used in the derivation of recoupling coefficients. They can also be used to

derive the second orthogonality relation of 3-jm symbols.

3.8 Recoupling coefficients and 3n-j symbols

3.8.1 Triangular delta

Consider the usual CG coupling of $|j_1 m_1\rangle$ and $|j_2 m_2\rangle$ to form the coupled state $|(12)j_{12} m_{12} j_1 j_2\rangle$ as in Eq. 3.5,

$$|(12)j_{12} m_{12} j_1 j_2\rangle = \sum_{m_1 m_2} |j_1 m_1 j_2 m_2\rangle\langle 1, 2|12\rangle$$

Here, we introduce a shorthand for Clebsch–Gordan coefficients:

$$\langle a, b|c\rangle = \langle j_a m_a j_b m_b | j_c m_c\rangle$$

We call this coupling "(12)" because in the CG coefficient angular momentum 1 appears before angular momentum 2. We could have also coupled them in reverse:

$$|(21)j_{12} m_{12} j_1 j_2\rangle = \sum_{m_1 m_2} |j_1 m_1 j_2 m_2\rangle\langle 2, 1|12\rangle$$

This leads to a *different* set of coupled eigenstates, which we call (21). They are still eigenstates of $(\hat{\boldsymbol{J}}^{(12)})^2$, $\hat{J}_3^{(12)}$, $\hat{\boldsymbol{J}}^{(1)}$, and $\hat{\boldsymbol{J}}^{(2)}$, just like the (12) states. Since the two states are bases of the same Hilbert space we expect there to exist a linear transformation between the two:

$$|(21)j_{12} m_{12} j_1 j_2\rangle = \sum_{j'_{12} j'_1 j'_2} |(12)j'_{12} m_{12} j'_1 j'_2\rangle\langle(12)j'_{12} j'_1 j'_2|(21)j_{12} j_2 j_1\rangle$$

The quantity $\langle (12) j'_{12} j'_1 j'_2 | (21) j_{12} j_2 j_1 \rangle$ denotes the **recoupling** coefficient from (12)-coupling to (21)-coupling, one of the simplest recoupling coefficients. From symmetry considerations alone (see Sec. 3.7) we already deduced that the coefficient is both block diagonal in m_{12} and does not depend on m_{12}.

Each recoupling coefficient has a set of selection rules that can be determined in a straightforward manner. In this case, we know that $j'_{12} = j_{12}$, $j'_1 = j_1$, and $j'_2 = j_2$, because they are eigenvalues of the same operators. Thus we find that

$$\langle (12) j'_{12} j'_1 j'_2 | (21) j_{12} j_2 j_1 \rangle = \delta_{j'_{12} j_{12}} \delta_{j'_1 j_1} \delta_{j'_2 j_2} \langle (12) j_{12} j_1 j_2 | (21) j_{12} j_2 j_1 \rangle$$

The remaining part of this particular recoupling coefficient has a very simple formula:

$$\begin{aligned} \langle (12) j_{12} j_1 j_2 | (21) j_{12} j_2 j_1 \rangle &= \frac{1}{\hat{j}_{12}^2} \sum_{m_1 m_2 m_{12}} \langle 1, 2 | 12 \rangle \langle 2, 1 | 12 \rangle \\ &= (-)^{j_1 + j_2 - j_{12}} \begin{Bmatrix} j_1 & j_2 & j_3 \end{Bmatrix} \end{aligned} \tag{3.23}$$

We thus observe that the coupling of states is *not* commutative, even though the addition of angular momenta operators is.

Notice that the recoupling coefficient contains a triangular delta $\begin{Bmatrix} j_1 & j_2 & j_3 \end{Bmatrix}$, which was previously defined in Eq. 3.6 and shown diagrammatically in Fig. 3.5. As we have noted, the triangular delta is the simplest irreducible closed diagram. This is a general property of recoupling coefficients: every recoupling coefficient can be decomposed into a product of irreducible closed diagrams, times simple factor containing phases or $\check{j}$-like quantities.

As we will see in the next few sections, the irreducible closed diagrams are more commonly known as **3n-j symbols**, which contains both 6-j symbols and 9-j symbols. The triangular delta is part of this family too, thus it is fitting to give it the name of a *3-j symbol* by analogy [WP06].

However, we do not use this terminology to avoid the inevitable confusion with *3-jm symbols*.

3.8.2 6-j symbol

Now, consider another case where we have a sum of three angular momenta

$$\hat{\boldsymbol{J}}^{(123)} = \hat{\boldsymbol{J}}^{(1)} + \hat{\boldsymbol{J}}^{(2)} + \hat{\boldsymbol{J}}^{(3)}$$

and we want to find a set of coupled eigenstates for $(\hat{\boldsymbol{J}}^{(123)})^2$ and $\hat{J}_3^{(123)}$. One possibility is to first obtain eigenstates $|j_{12}\, m_{12} j_1 j_2\rangle$ of $(\hat{\boldsymbol{J}}^{(12)})^2$ and $\hat{J}_3^{(12)}$, where $\hat{\boldsymbol{J}}^{(12)}$ is defined as

$$\hat{\boldsymbol{J}}^{(12)} = \hat{\boldsymbol{J}}^{(1)} + \hat{\boldsymbol{J}}^{(2)}$$

and then couple these states with $|j_3\, m_3\rangle$, leading to states of the form

$$|((12)3) j_{123}\, m_{123} j_{12} j_1 j_2 j_3\rangle = \sum_{m_1 m_2 m_{12} m_3} |j_1\, m_1 j_2\, m_2 j_3\, m_3\rangle \langle 1, 2|12\rangle \langle 12, 3|123\rangle$$

which are eigenstates of $(\hat{\boldsymbol{J}}^{(123)})^2$, $\hat{J}_3^{(123)}$, $(\hat{\boldsymbol{J}}^{(12)})^2$, $(\hat{\boldsymbol{J}}^{(1)})^2$, $(\hat{\boldsymbol{J}}^{(2)})^2$, and $(\hat{\boldsymbol{J}}^{(3)})^2$.

It is clear that we have introduced a bias to the $\hat{\boldsymbol{J}}^{(12)}$ here. What if instead we couple $\hat{\boldsymbol{J}}^{(2)}$ to $\hat{\boldsymbol{J}}^{(3)}$, and then couple $\hat{\boldsymbol{J}}^{(1)}$ to that? Then we would obtain the states

$$|(1(23)) j_{123}\, m_{123} j_{23} j_1 j_2 j_3\rangle = \sum_{m_1 m_2 m_3 m_{23}} |j_1\, m_1 j_2\, m_2 j_3\, m_3\rangle \langle 2, 3|23\rangle \langle 1, 23|123\rangle$$

Or we also couple $\hat{\boldsymbol{J}}^{(1)}$ to $\hat{\boldsymbol{J}}^{(3)}$, and then to $\hat{\boldsymbol{J}}^{(2)}$, leading to the states

$$|((13)2) j_{123}\, m_{123} j_{13} j_1 j_2 j_3\rangle = \sum_{m_1 m_3 m_{13} m_2} |j_1\, m_1 j_2\, m_2 j_3\, m_3\rangle \langle 1, 3|13\rangle \langle 13, 2|123\rangle$$

These choices lead to very different sets of eigenstates that are related by nontrivial coefficients. There are also several other ways to couple, such as ((21)3), (1(32)), (2(13)), etc, but they are equivalent to one of the above three up to a phase factor akin to Eq. 3.23.

To convert from, say, ((12)3) to (1(23)), we would require the following m-independent recoupling coefficient:

$$\langle((12)3)j'_{123}j_{12}j'_1j'_2j'_3|(1(23))j_{123}j_{23}j_1j_2j_3\rangle$$

The selection rules tell us that the primed quantities have to match the unprimed quantities. So the only nontrivial elements are:

$$\begin{aligned}&\langle((12)3)j_{123}j_{12}j_1j_2j_3|(1(23))j_{123}j_{23}j_1j_2j_3\rangle\\&=\frac{1}{j_{123}^2}\sum_{m_1m_2m_3m_{12}m_{23}m_{123}}\langle 12,3|123\rangle\langle 1,2|12\rangle\langle 2,3|23\rangle\langle 1,23|123\rangle\end{aligned}$$

Hence, the coupling of states is also not associative, even though the addition of angular momenta operators is.

The recoupling coefficient $\langle((12)3)j_{123}j_{12}j_1j_2j_3|(1(23))j_{123}j_{23}j_1j_2j_3\rangle$ can be expressed in terms of a quantity called the **6-j symbol**, defined as

$$\begin{aligned}\begin{Bmatrix}j_1 & j_2 & j_3\\ j_4 & j_5 & j_6\end{Bmatrix}=\sum_{m_1m_2m_3m_4m_5m_6}(-)^{j_4-m_4+j_5-m_5+j_6-m_6}&\begin{pmatrix}j_1 & j_2 & j_3\\ m_1 & m_2 & m_3\end{pmatrix}\\&\begin{pmatrix}j_1 & j_5 & j_6\\ m_1 & -m_5 & m_6\end{pmatrix}\begin{pmatrix}j_2 & j_6 & j_4\\ m_2 & -m_6 & m_4\end{pmatrix}\begin{pmatrix}j_3 & j_4 & j_5\\ m_3 & -m_4 & m_5\end{pmatrix}\end{aligned}\tag{3.24}$$

Fig. 3.12 shows the diagram for a 6-j symbol, which corresponds to the following diagrammatic

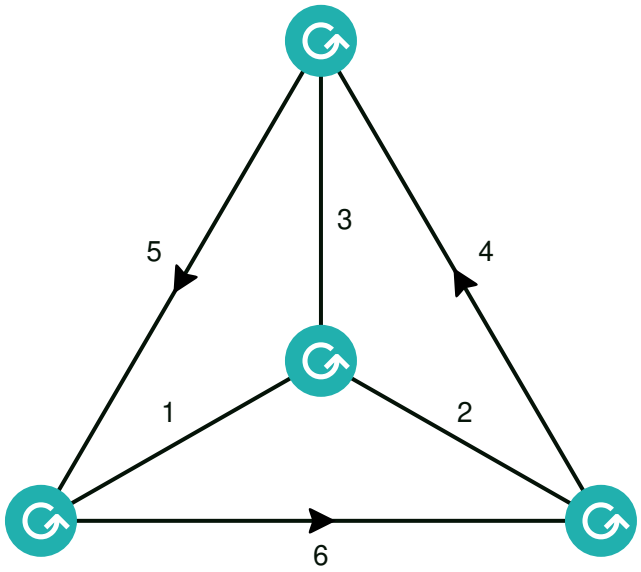

Figure 3.12: 6-j symbol: $\{123456\} = (123)(1\check{5}6)(2\check{6}4)(3\check{4}5)$ in Eq. 3.24

shorthand:

$$\{123456\} = (123)(1\check{5}6)(2\check{6}4)(3\check{4}5)$$

We may now write the aforementioned recoupling coefficient as:

$$\langle((12)3)j_{123}j_{12}j_1j_2j_3|(1(23))j_{123}j_{23}j_1j_2j_3\rangle = (-)^{j_1+j_2+j_3+j_{123}}\check{j}_{12}\check{j}_{23}\begin{Bmatrix} j_1 & j_2 & j_{12} \\ j_3 & j_{123} & j_{23} \end{Bmatrix}$$

The 6-j symbol has the nonlocal selection rules corresponding to those of its 3-jm nodes, which are simply the following triangle conditions:

$$\begin{Bmatrix} j_1 & j_2 & j_3 \end{Bmatrix} \qquad \begin{Bmatrix} j_1 & j_5 & j_6 \end{Bmatrix} \qquad \begin{Bmatrix} j_2 & j_6 & j_4 \end{Bmatrix} \qquad \begin{Bmatrix} j_3 & j_4 & j_5 \end{Bmatrix}$$

Note that Fig. 3.12 is only one out of several ways to draw a 6-j symbol. It has in fact several interesting symmetries that are not immediately obvious. For example, the columns of the 6-j

symbol can be permuted *arbitrarily* (both odd and even permutations):

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \end{Bmatrix} = \begin{Bmatrix} j_2 & j_1 & j_3 \\ j_5 & j_4 & j_6 \end{Bmatrix} = \begin{Bmatrix} j_2 & j_3 & j_1 \\ j_5 & j_6 & j_4 \end{Bmatrix} = \cdots$$

It also has the following tetrahedral symmetries:

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \end{Bmatrix} = \begin{Bmatrix} j_4 & j_5 & j_3 \\ j_1 & j_2 & j_6 \end{Bmatrix} = \begin{Bmatrix} j_4 & j_2 & j_6 \\ j_1 & j_5 & j_3 \end{Bmatrix} = \begin{Bmatrix} j_1 & j_5 & j_6 \\ j_4 & j_2 & j_3 \end{Bmatrix}$$

3.8.3 9-j symbol

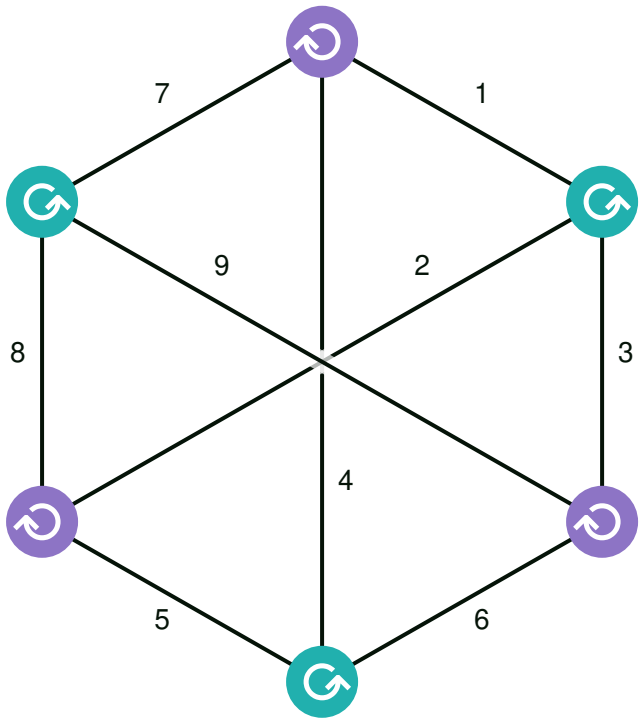

Figure 3.13: 9-j symbol: {123456789} = (123)(456)(789)(147)(258)(369) in Eq. 3.25

Certain recouplings four or more angular momenta can lead to another type of irreducible

diagram known as the **9-j symbol**, defined as:

$$\begin{Bmatrix} j_1 & j_2 & j_3 \\ j_4 & j_5 & j_6 \\ j_7 & j_8 & j_9 \end{Bmatrix} = \sum_{m_1 m_2 m_3 m_4 m_5 m_6 m_7 m_8 m_9} \begin{pmatrix} j_1 & j_2 & j_3 \\ m_1 & m_2 & m_3 \end{pmatrix} \begin{pmatrix} j_4 & j_5 & j_6 \\ m_4 & m_5 & m_6 \end{pmatrix} \begin{pmatrix} j_7 & j_8 & j_9 \\ m_7 & m_8 & m_9 \end{pmatrix} \begin{pmatrix} j_1 & j_4 & j_7 \\ m_1 & m_4 & m_7 \end{pmatrix} \begin{pmatrix} j_2 & j_5 & j_8 \\ m_2 & m_5 & m_8 \end{pmatrix} \begin{pmatrix} j_3 & j_6 & j_9 \\ m_3 & m_6 & m_9 \end{pmatrix} \tag{3.25}$$

This is shown diagrammatically in Fig. 3.13, which can be expressed as the following shorthand:

$$\{123456789\} = (123)(456)(789)(147)(258)(369)$$

The nonlocal selection rules of 9-j symbols are simply triangle conditions on all row and column triplets. They are invariant under reflections about either diagonal, and also invariant under even permutations of rows or columns. An odd permutation would introduce a phase factor of $(-)^{\sum_{i=0}^{9} j_i}$.

3.9 Calculation of angular momentum coefficients

Numerical values of the coupling and recoupling coefficients (i.e. 3-jm, 6-j, and 9-j symbols) can be calculated readily using the formulas as given in this chapter. Due to the presence of large alternating sums, use of arbitrary-precision arithmetic is highly recommended to avoid catastrophic loss of precision.

Optimized variants of the formulas for 3-jm, 6-j, and 9-j symbols have been described in detail in [Wei99; Wei98]. These have been implemented in the `wigner-symbols` software packages in Rust [WSR] and Haskell [WSH], which leverage the GNU Multi Precision (GMP) Arithmetic Library [Gt16] for its highly optimized arbitrary-precision integer and rational arithmetic.

Even with the fastest algorithms, it is often more performant to reuse (re)coupling coefficients that have been previously computed and cached in memory than to recompute them again. For this, the storage scheme devised in [RY04] based on Regge symmetries can help reduce the total memory usage. The storage scheme consists of two main parts:

- A canonicalization scheme that uses the symmetries of the coupling coefficients to link ones that differ by a trivial phase factor.
- An indexing scheme that translates canonicalized angular momenta into an array index, allowing rapid lookup of elements.

In practice, we found the canonicalization scheme most useful for calculations as it provides a guaranteed 1-2 orders of magnitude reduction in memory usage. In contrast, the indexing scheme is not substantially faster than a plain hash-table lookup and comes with the disadvantage of requiring all coefficients to be precomputed up to some limit. This makes it somewhat difficult to use in practice and can result in wasted memory if the limit is overestimated.

3.10 Graphical tool for angular momentum diagrams

We have developed a graphical tool [Jucys] that can be used to perform graphical manipulation of angular momentum coefficients with the diagrammatic technique explained in this chapter, with a few slight modifications. Specifically, non-diagrammatic objects such as phases, $\check{j}$-like factors, Kronecker deltas, or summations over j-type variables are all tracked separately in a **tableau** that is displayed beside the diagram.

The primary motivation of the tool is to eliminate human errors that commonly occur in angular momentum algebra and improve the speed of such derivations. To achieve this, the diagram offers a special *reduction mode* that, when activated, *ensures that all of the user's diagrammatic manipulations preserve equality.* The user modifies the diagram through various gestures and clicks of the mouse cursor. The program is responsible for enforcing the diagrammatic rules, including orthogonal relations, separation rules, various phase rules, etc.

The program comes with a separate *input tool* for writing angular momentum expressions, without which the user would have to manually draw angular momentum diagrams node by node – a tedious and error-prone process. The input tool provides fast means of describing coupling coefficients in text, reducing the room for human error. As an example, the Pandya transformation coefficient for spherical scalars is described by the following input:

```
rel (p + q) (r + s)
rec (p - s) (r - q)
```

Here, `rel` equates the two angular momenta `p + q` and `r + s`. The `rec` equates the two angular momenta `p - s` and `r - q` but also includes an extra $1/\check{j}_{ps}^2$ factor. The plus sign in `p + q` denotes the usual CG coupling

$$\langle p, q | pq \rangle = \langle j_p m_p j_q m_q | j_{pq} m_{pq} \rangle$$

whereas the minus sign in `p - s` denotes coupling with the second angular momentum time-reversed:

$$\langle p, \check{s} | ps \rangle = (-)^{j_s - m_s} \langle j_p, m_p, j_s, -m_s | j_{ps}, m_{ps} \rangle \tag{3.26}$$

After providing this input to the tool, the corresponding 6-j diagram can be rapidly derived along with the associated phases and factors.

As another example, the Pandya transformation coefficient for a spherical tensor $\hat{A}^{j_A}_{m_A}$ is described by

```
wet (p + q) A (r + s)
wet (p - s) A (r - q)
```

Here, `wet` denotes the use of the Wigner–Eckart coupling and the central `A` variable is the rank j_A of the spherical tensor. After providing this input, one can quickly derive the corresponding 9-j diagram with the associated phases and factors.

The tool is a web application written in a combination of JavaScript, HTML, and CSS. It can therefore run in any modern Internet browser and is accessible to users on most desktop platforms. An online version is available for immediate use, but the user can also run the application on their own machine with the appropriate setup. It utilizes SVG technology to display diagrams, making it straightforward to export diagrams as vector images, suitable for use in literature as we have done in this work.

We will not attempt to explain the usage of the program here, as that information will very likely become out of date as the program evolves. Interested users are advised to read the official documentation for usage information.

3.11 Fermionic states in J-scheme

J-scheme is a many-body formalism that takes advantage of angular momentum conservation to reduce the dimensionality of the problem (i.e. the computational cost and size of matrices). In this

context, the usual formalism where we do not take advantage of angular momentum symmetries is dubbed **M-scheme** for contrast.

We use $a, b, c, \ldots$ to label single-particle states in this section. We assume each state has some definite angular-momentum-like quantum numbers: magnitude j and projection m, along with some other quantum number(s) α that are not relevant here.

3.11.1 Two-particle states

A two-particle J-coupled product state is defined as

$$\begin{aligned} |\alpha_a j_a \otimes \alpha_b j_b; j_{ab} m_{ab}\rangle &= \sum_{m_a m_b} |a \otimes b\rangle\langle a, b|ab\rangle \\ &= \sum_{m_a m_b} |\alpha_a j_a m_a \otimes \alpha_b j_b m_b\rangle\langle j_a m_a j_b m_b|j_{ab} m_{ab}\rangle \end{aligned}$$

where $\langle a, b|ab\rangle = \langle j_a m_a j_b m_b|j_{ab} m_{ab}\rangle$ is the Clebsch–Gordan coefficient (Sec. 3.2) and $|a \otimes b\rangle = |\alpha_a j_a m_a \otimes \alpha_b j_b m_b\rangle$ denotes the (non-antisymmetrized) tensor product state (Sec. 2.1.1) in M-scheme. To keep things concise, we will use the following shorthand for coupled product states:

$$|(12)a \otimes b\rangle = |\alpha_a j_a \otimes \alpha_b j_b; j_{ab} m_{ab}\rangle$$

Keep in mind that unlike M-scheme, the states in J-scheme do *not* depend on the individual projections m_a and m_b, only total m_{ab}.

The coupled product states are eigenstates of the total $\hat{J}^2$ of all particles,

$$\hat{J}^2|(12)a \otimes b\rangle = j_{ab}(j_{ab} + 1)|(12)a \otimes b\rangle$$

In contrast, uncoupled states are not eigenstates of $\hat{J}^2$.

For fermionic problems, we can form an antisymmetrized state for J-scheme. The most straightforward way to do this is by coupling the antisymmetrized state,

$$|\alpha_a j_a \alpha_b j_b; j_{ab} m_{ab}\rangle = \frac{1}{\sqrt{N_{ab}}} \sum_{m_a m_b} |ab\rangle\langle j_a m_a j_b m_b | j_{ab} m_{ab}\rangle$$

where the normalization factor is given by

$$N_{ab} = 1 - (-)^{2j_a + j_{ab}} \delta_{\alpha_a \alpha_b} \delta_{j_a j_b} \tag{3.27}$$

If the normalization factor is zero, then the antisymmetrized state does not exist.

Note that N_{ab} depends on only the non-m parts of a and b. If j_a and j_b are always half-odd, then the normalization factor can be further simplified to $N_{ab} = 1 + (-)^{j_{ab}} \delta_{\alpha_a \alpha_b} \delta_{j_a j_b}$, which means if $\alpha_a = \alpha_b$ and $j_a = j_b$, then states with odd j_{ab} do not exist.

As before, we will also introduce a shorthand for the antisymmetrized states,

$$|(12)ab\rangle = |\alpha_a j_a \alpha_b j_b; j_{ab} m_{ab}\rangle$$

which depends on neither m_a nor m_b.

Alternatively, one can also obtain the same state from a J-coupled product state:

$$\begin{aligned} |(12)ab\rangle &= \sqrt{\frac{2}{N_{ab}}} \mathcal{S}^{(1 + j_a + j_b - j_{ab})} |(12) a \otimes b\rangle \\ &= \frac{1}{\sqrt{2N_{ab}}} \left(|(12) a \otimes b\rangle - (-)^{j_a + j_b - j_{ab}} |(12) b \otimes a\rangle \right) \end{aligned}$$

where $\mathcal{S}^{(1+j_a+j_b-j_{ab})}$ is the ±-symmetrization symbol introduced in Sec. 2.1.2,

$$\mathcal{S}^{(1+j_a+j_b-j_{ab})}X_{ab} = \frac{1}{2}\left(X_{ab} + (-)^{1+j_a+j_b-j_{ab}}X_{ba}\right)$$

Note that the antisymmetrizer $\hat{S}^-$ (Sec. 2.1.2) operates differently in J-scheme compared to in M-scheme: matrix elements of the antisymmetrizer $\hat{S}^-$ are not always antisymmetric with respect to (j, α) in J-scheme; instead they depend on the parity of $j_a + j_b - j_{ab}$. This becomes even more complex for 3 or more particles as the matrix elements of the antisymmetrizer may contain 6-j or higher symbols.

Under particle exchange, the J-scheme antisymmetrized state has the following property:

$$|(12)ab\rangle = -(-)^{j_a+j_b-j_{ab}}|(12)ba\rangle$$

In J-scheme, the two-body antisymmetrized matrix elements are related to the product matrix elements by

$$\begin{aligned}
&\langle(12)ab|\hat{V}|(12)cd\rangle \\
&= \sqrt{\frac{2}{N_{ab}}}\langle(12)a\otimes b|\hat{V}|(12)cd\rangle \\
&= \frac{1}{\sqrt{N_{ab}N_{cd}}}\left(\langle(12)a\otimes b|\hat{V}|(12)c\otimes d\rangle - (-)^{j_c+j_d-j_{cd}}\langle(12)a\otimes b|\hat{V}|(12)d\otimes c\rangle\right)
\end{aligned}$$

3.11.2 Three-particle states

Three-particle states have 3 nontrivially distinct ways of coupling. We will stick to the convention of coupling the first two, then the third, which we call the **standard coupling order**. In this case,

the product state in J-scheme is given by

$$|((12)3)a \otimes b \otimes c\rangle = \sum_{m_a m_b m_c} |a \otimes b \otimes c\rangle\langle a, b|ab\rangle\langle ab, c|abc\rangle$$

As usual, the J-scheme antisymmetrized state is formed by coupling the M-scheme antisymmetrized state,

$$|((12)3)abc\rangle = \frac{1}{\sqrt{N_{(ab)c}}} \sum_{m_a m_b m_c} |abc\rangle\langle a, b|ab\rangle\langle ab, c|abc\rangle$$

where the normalization constant $N_{(ab)c}$ is given by

$$\begin{aligned} N_{(ab)c} = 1 &- (-)^{2j_a+j_{ab}} \delta_{\alpha_a\alpha_b}\delta_{j_a j_b} \\ &- (-)^{2j_{abc}} \check{j}_{ab} \begin{Bmatrix} j_a & j_b & j_{ab} \\ j_{abc} & j_b & j_{ab} \end{Bmatrix} \delta_{\alpha_b\alpha_c}\delta_{j_b j_c} \\ &- (-)^{2j_{abc}} \check{j}_{ab} \begin{Bmatrix} j_b & j_a & j_{ab} \\ j_{abc} & j_a & j_{ab} \end{Bmatrix} \delta_{\alpha_a\alpha_c}\delta_{j_a j_c} \\ &+ 2(-)^{j_{ab}} \check{j}_{ab} \begin{Bmatrix} j_a & j_a & j_{ab} \\ j_{abc} & j_a & j_{ab} \end{Bmatrix} \delta_{\alpha_a\alpha_b}\delta_{j_a j_b}\delta_{\alpha_a\alpha_c}\delta_{j_a j_c} \end{aligned}$$

3.12 Matrix elements in J-scheme

In this work, we do not use normalized J-scheme states: equations tend to be simpler if we use **unnormalized matrix elements** in which the $1/\sqrt{N}$ factor (see Eq. 3.27) is omitted. This convention is used throughout.

3.12.1 Standard-coupled matrix elements

Given an M-scheme two-body matrix $A_{pqrs}^{m_p m_q m_r m_s}$, we can couple p to q and r to s,

$$A_{pqrs}^{j_{pq} m_{pq} j_{rs} m_{rs}(12;34)} = \sum_{m_p m_q m_r m_s} \langle p, q | pq \rangle \langle r, s | rs \rangle A_{pqrs}^{m_p m_q m_r m_s}$$

where $\langle p, q | pq \rangle = \langle j_p m_p j_q m_q | j_{pq} m_{pq} \rangle$ is the CG coefficient (Sec. 3.2). We call this the **standard coupling** for two-body matrix elements and denote it by schematically as $12;34$. We will often omit the $(12;34)$ superscript as we consider this the default coupling scheme.

If the matrix is a spherical scalar, then thanks to the Wigner–Eckart theorem we can omit many of the superscripts:

$$A_{pqrs}^{j_{pq} m_{pq} j_{rs} m_{rs}(12;34)} = \delta_{j_{pq} j_{rs}} \delta_{m_{pq} m_{rs}} A_{pqrs}^{j_{pq}(12;34)}$$

where $A_{pqrs}^{j_{pq}(12;34)}$ denotes the reduced matrix element in the CG convention (Sec. 3.6). Like any reduced matrix element, it is entirely independent of m.

If $\hat{A}$ is a spherical tensor of rank j_A and projection m_A, then it is more convenient to use the reduced matrix element in the 3-jm convention

$$A_{pqrs}^{j_A m_A j_{pq} m_{pq} j_{rs} m_{rs}(12;34)} = (-)^{j_{pq} - m_{pq}} \begin{pmatrix} j_{ps} & j_A & j_{rq} \\ -m_{ps} & m_A & m_{rq} \end{pmatrix} A_{pqrs}^{j_A j_{pq} j_{rs}(12;34)}$$

The standard coupling can be extended for higher-body operators: one simply couples the bra and ket indices in the order as written. For example, a three-body matrix in standard coupling

would be

$$
\begin{aligned}
&A^{j_{pqr} m_{pqr} j_{pq} j_{stu} m_{stu} j_{st}((12)3;(45)6)}_{pqrstu} \\
&= \sum_{m_p m_q m_r m_s m_t m_u} \langle p, q|pq\rangle \langle pq, r|pqr\rangle \langle s, t|st\rangle \langle st, u|stu\rangle A^{m_p m_q m_r m_s m_t m_u}_{pqrstu}
\end{aligned}
$$

This is denoted schematically by (12)3; (45)6. In the case of spherical scalars, we have the following reduced matrix elements in the CG convention:

$$
A^{j_{pqr} m_{pqr} j_{pq} j_{stu} m_{stu} j_{st}((12)3;(45)6)}_{pqrstu} = \delta_{j_{pqr} j_{stu}} \delta_{m_{pqr} m_{stu}} A^{j_{pqr} j_{pq} j_{st}((12)3;(45)6)}_{pqrstu}
$$

3.12.2 Pandya-coupled matrix elements

Besides the standard coupling, two-body operators can be coupled in several other ways. Some are equivalent to 12; 34 up to a phase factor. A nontrivial combination is the **Pandya coupling** [Pan56; Suh07] $1\check{4}; 3\check{2}$:

$$
A^{j_{ps} m_{ps} j_{rq} m_{rq}(1\check{4};3\check{2})}_{psrq} = - \sum_{m_p m_s m_r m_q} \langle p, \check{s}|ps\rangle \langle r, \check{q}|rq\rangle A^{m_p m_q m_r m_s}_{pqrs}
$$

where the $\langle p, \check{s}|ps\rangle$ uses the time-reversed CG notation introduced in Eq. 3.26.

The extraneous minus sign in front of the summation is conventional: if we treat this a recoupling of field operators, we would obtain a minus sign due to antisymmetry since the permutation $1234 \longrightarrow 1432$ is odd. If instead we omit the extraneous minus sign, the coupling is often referred to as **cross-coupling** [Kuo+81] rather than Pandya-coupling.

For spherical scalars, we have the following reduced matrix elements in the CG convention.

$$
A^{j_{ps} m_{ps} j_{rq} m_{rq}(1\check{4};3\check{2})}_{psrq} = \delta_{j_{ps} j_{rq}} \delta_{m_{ps} m_{rq}} A^{j_{ps}(1\check{4};3\check{2})}_{psrq}
$$

They are related to the standard-coupled reduced matrix elements the **Pandya transformation**:

$$A^{j_{ps}(1\check{4};3\check{2})}_{psrq} = -\sum_{j_{pq}} (-)^{2j_{pq}} \check{j}^2_{pq} \begin{Bmatrix} j_p & j_q & j_{pq} \\ j_r & j_s & j_{ps} \end{Bmatrix} A^{j_{pq}(12;34)}_{pqrs}$$

The inverse Pandya transformation is nearly the same:

$$A^{j_{ps}(12;34)}_{pqrs} = -(-)^{2j_{pq}} \sum_{j_{ps}} \check{j}^2_{ps} \begin{Bmatrix} j_p & j_q & j_{pq} \\ j_r & j_s & j_{ps} \end{Bmatrix} A^{j_{ps}(1\check{4};3\check{2})}_{psrq}$$

However, typically when Pandya-coupled matrices are involved, the fermionic antisymmetry is temporarily broken. As a result, in our implementation, Pandya-coupled matrices are not antisymmetrized even though standard-coupled matrices are. To restore the antisymmetry when performing the inverse transformation, we must perform an explicit antisymmetrization during the inverse transformation:

$$A^{12;34}_{pqrs} = -(-)^{2j_{pq}} \mathcal{S}^{(1+j_p+j_q-j_{pq})}_{pq} \mathcal{S}^{(1+j_r+j_s-j_{rs})}_{rs} \sum_{j_{ps}} \check{j}^2_{ps} \begin{Bmatrix} j_p & j_q & j_{pq} \\ j_r & j_s & j_{ps} \end{Bmatrix} \tilde{A}^{1\check{4};3\check{2}}_{psrq}$$

where the tilde symbol ($\tilde{A}$) indicates that the matrix element is not antisymmetrized and $\mathcal{S}^{(i)}$ is the $(-)^i$-symmetrization symbol in Sec. 2.1.2.

For completeness, we also include the Pandya transformation for spherical tensor operators,

$$A^{j_A j_{ps} j_{rq}(1\check{4};3\check{2})}_{psrq} = -\sum_{j_{pq} j_{rs}} \check{j}_{pq} \check{j}_{rs} \check{j}_{ps} \check{j}_{rq} (-)^{j_q+j_s-j_{rs}+j_{rq}} \begin{Bmatrix} j_p & j_q & j_{pq} \\ j_s & j_r & j_{rs} \\ j_{ps} & j_{rq} & j_A \end{Bmatrix} A^{j_A j_{pq} j_{rs}(12;34)}_{pqrs}$$

where we use reduced matrix elements in the 3-jm convention,

$$A^{j_A m_A j_{ps} m_{ps} j_{rq} m_{rq}(1\check{4};3\check{2})}_{psrq} = (-)^{j_{ps}-m_{ps}} \begin{pmatrix} j_{ps} & j_A & j_{rq} \\ -m_{ps} & m_A & m_{rq} \end{pmatrix} A^{j_A j_{ps} j_{rq}(1\check{4};3\check{2})}_{psrq}$$

The inverse transformation is identical except the summation is over j_{ps} and j_{rq}.

3.12.3 Implicit-J convention

To keep J-scheme of scalar operators concise, we will omit explicit mention of composite angular momenta within the matrix elements:

$$A^{j_{pq}}_{pqrs} \rightsquigarrow A_{pqrs}$$

$$A^{j_{pqr} j_{pq} j_{rs}}_{pqrstu} \rightsquigarrow A_{pqrstu}$$

We will also omit mentions of Kronecker deltas between angular momenta as well as triangular deltas. We call this the **implicit-J convention**.

As an example, consider the following scalar equation written in our implicit-J convention:

$$C_{pq} = \frac{1}{2} \sum_{j_{ip}} \sum_{i \backslash ab} \frac{\check{j}^2_{ip}}{\check{j}^2_p} A_{ipab} B_{abiq}$$

To decode this, we follow these steps:

1. We first determine the set of *composite* angular momentum variables. This comes from a combination of (a) the composite angular momenta from the left-hand side (there are none, since C_{pq} is only one-body), (b) the composite angular momenta that are being explicitly

summed over (namely j_{ip}).

$$C_{pq} = \frac{1}{2} \sum_{j_{ip}} \sum_{i \setminus ab} \frac{\check{j}_{ip}^2}{\check{j}_p^2} A^{j_{ip}}_{ipab} B^{?}_{abiq}$$

2. Next, we fill in the remaining slots for composite angular momenta using conservation laws. Since $j_{ip} = j_{ab}$, the missing angular momentum on B is simply j_{ip}:

$$C_{pq} = \frac{1}{2} \sum_{j_{ip}} \sum_{i \setminus ab} \frac{\check{j}_{ip}^2}{\check{j}_p^2} A^{j_{ip}}_{ipab} B^{j_{ip}}_{abiq}$$

3. We may use the conservation laws to determine the Kronecker deltas for the *elementary* angular momenta:

$$C_{pq} = \frac{1}{2} \delta_{j_p j_q} \sum_{j_{ip}} \sum_{i \setminus ab} \frac{\check{j}_{ip}^2}{\check{j}_p^2} A^{j_{ip}}_{ipab} B^{j_{ip}}_{abiq}$$

4. Finally, we use selection rules to restrict the composite angular momenta via triangular deltas:

$$C_{pq} = \frac{1}{2} \delta_{j_p j_q} \sum_{j_{ip}} \sum_{i \setminus ab} \frac{\check{j}_{ip}^2}{\check{j}_p^2} \begin{Bmatrix} j_i & j_p & j_{ip} \end{Bmatrix} \begin{Bmatrix} j_a & j_b & j_{ip} \end{Bmatrix} A^{j_{ip}}_{ipab} B^{j_{ip}}_{abiq}$$

This can be generalized to spherical tensors by omitting the tensor ranks:

$$A^{j_A j_{pq} j_{rs}}_{pqrs} \rightsquigarrow A_{pqrs}$$

$$A^{j_A j_{pqr} j_{pq} j_{rst} j_{rs}}_{pqrstu} \rightsquigarrow A_{pqrstu}$$

The procedure to decode these is analogous: tensor ranks should be treated like composite angular

momenta.

Chapter 4

Many-body methods

We now discuss the many-body methods that form the core of our many-body code. In Sec. 2.1.2, we have noted that antisymmetrized states (Slater determinants) provide solutions for any non-interacting fermionic system. We have not yet discussed how to solve *interacting* systems however, which is the principal focus of many-body theory.

In general, while solutions of the non-interacting Hamiltonian $\hat{H}^\circ$ are often not solutions of any interacting Hamiltonian $\hat{H}$, they do nonetheless provide a useful basis for the Fock space. We expect from basic linear algebra that, if the degrees of freedom (including boundary conditions) are the same between $\hat{H}^{\mathrm{NI}}$ and $\hat{H}$, then any exact N-particle solution $|\Psi\rangle$ can be expanded as a linear combination of antisymmetrized states,

$$|\Psi\rangle = \frac{1}{N!} \sum_{p_1 \cdots p_N} \Psi_{p_1 \cdots p_N} \hat{a}^\dagger_{p_1} \cdots \hat{a}^\dagger_{p_N} |\varnothing\rangle$$

Solving a quantum system in this manner is the central theme of **exact diagonalization** methods, such as **full configuration interaction** (FCI) [Ols+88; KH84] and **no-core shell model** (NCSM) [NVB00; Nav+09].

The key advantage of exact diagonalization is the ability to obtain exact numeric results within the basis (up to machine precision), capturing all the details of the quantum system. However, such methods are very costly as the number of N-particle basis states n_{B} increases rapidly with

the number of particles N and the number of single-particle states n_b, specifically

$$n_\mathrm{B} = \binom{n_\mathrm{b}}{N}$$

where $\binom{n}{k}$ denotes the binomial coefficient. The combinatorial explosion quickly renders such methods unfeasible in systems with even a moderate number of particles, beyond the computational power that exists in the observable universe.

Alternative many-body methods strive to avoid this problem by limiting the N-particle Hilbert space under consideration. It can be particularly beneficial if the non-interacting Hamiltonian $\hat{H}^\circ$ that generates the basis states is to some extent similar to the interacting Hamiltonian $\hat{H}$. That is, one decomposes $\hat{H}$ into

$$\hat{H} = \hat{H}^\circ + \hat{V}$$

where the contributions of the **perturbation** $\hat{V}$ are expected to be small in some sense. In this case, one or a few antisymmetrized states may serve as a good zeroth order approximation to the system.

For now, we will only consider using a single antisymmetrized state as the initial approximation. This limits our consideration to closed-shell systems in which the number of particles coincides with a magic number. This distinguished antisymmetrized state will serve as our *reference state* (Fermi vacuum).

4.1 Hartree-Fock method

The reference state formed by basis states of the non-interacting Hamiltonian may not offer a good approximation of the true ground state (i.e. of the interacting Hamiltonian). The Hartree–Fock (HF) method [Har28; Foc30] provides a way to optimize the basis states such that the reference state provides the best variational estimate of the ground state energy.

4.1.1 Hartree–Fock equations

Using the variational principle, one can compute an approximate ground state $|\Phi\rangle$ by minimizing the energy expectation value (**Hartree–Fock energy**)

$$E_\Phi = \langle \Phi|\hat{H}|\Phi\rangle \tag{4.1}$$

with respect to a reference state $|\Phi\rangle$, subject to the restriction that $|\Phi\rangle$ remains a single Slater determinant constructed from an unknown single-particle basis $|p'\rangle$,[1]

$$|\Phi\rangle = |i'_1 \dots i'_N\rangle$$

where $\{i'_1, \dots, i_N\}$ are an unknown set of occupied state labels drawn from the unknown single-particle basis. The restriction of $|\Phi\rangle$ to a single Slater determinant is what enables the simplicity and efficiency of this method.

To perform numerical calculations, we further assume that each unknown state $|p'\rangle$ is built from a linear combination of known states $|p\rangle$, with an unknown matrix of coefficients C defined

[1] The unknown basis is distinguished from the known basis by the prime symbol $'$ in their labels.

via the transformation equation

$$|p'\rangle = \sum_p |p\rangle C_{pp'}$$

This allows the problem to be reduced from an abstract minimization problem Eq. 4.1 to a concrete numerical problem. The caveat is that the set of known functions must be large enough to capture the relevant behavior of the system.

To ensure orthonormality of the states, we require there to be as many unknown states $|p'\rangle$ as known states $|p\rangle$, and the coefficient matrix $\boldsymbol{C}$ must be unitary,

$$\boldsymbol{C}^\dagger \boldsymbol{C} = \mathbf{1}$$

These conditions are more strict than needed, but they greatly simplify the calculations and allow the states $|p'\rangle$ to act as *optimized* inputs for methods beyond HF (**post-HF** methods). At the end of the calculation, of the set of states $|p'\rangle$ there would be exactly N occupied states that participate in the optimized Slater determinant $|\Phi\rangle$. The remaining unoccupied states serve as the complementary space into which particles can be excited by the interaction $\hat{V}$ during post-HF calculations.

Consider a Hamiltonian $\hat{H}$ that can be decomposed into a set of $(1, 2, 3)$-body operators relative to the physical vacuum,

$$\hat{H} = \hat{H}_1^\varnothing + \hat{H}_2^\varnothing + \hat{H}_3^\varnothing$$

where $\hat{H}_k^\varnothing$ is its k-body component relative to the physical vacuum. (We will omit the $\varnothing$ suffix in

this section.) The goal is to find the coefficients $\boldsymbol{C}$ that minimize the Hartree–Fock energy E_Φ,

$$E_\Phi = \sum_{i'\backslash} \langle i'|\hat{H}_1|i'\rangle + \frac{1}{2}\sum_{i'j'\backslash} \langle i'j'|\hat{H}_2|i'j'\rangle \tag{4.2}$$

where

$$\begin{aligned}\langle p'|\hat{H}_1|q'\rangle &= \sum_{pq} C^*_{pp'}\langle p|\hat{H}_1|q\rangle C_{qq'} \\ \langle p'q'|\hat{H}_2|r's'\rangle &= \sum_{pqrs} C^*_{pp'}C^*_{qq'}\langle pq|\hat{H}_2|rs\rangle C_{rr'}C_{ss'}\end{aligned} \tag{4.3}$$

and $\sum_{i'\backslash}$ denotes a summation over all hole states $|i'\rangle$ in the unknown basis (see Eq. 2.3).

With the method of Lagrange multipliers, the minimization problem can be reduced to the solving of a nonlinear equation – the self-consistent **Hartree–Fock equations**:

$$\boldsymbol{FC} = \boldsymbol{C\varepsilon} \tag{4.4}$$

where the **Fock matrix** $\boldsymbol{F}$ is defined as

$$F_{pq} = \langle p|\hat{H}_1|q\rangle + \sum_{rs}\sum_{i'\backslash} C^*_{ri'}\langle pr|\hat{H}_2|qs\rangle C_{si'} \tag{4.5}$$

with i' ranging over occupied states only, and $\boldsymbol{\varepsilon}$ is a vector of Lagrange multipliers, which serve to constrain the orthonormality of the single-particle basis. Each multiplier $\varepsilon_{p'}$ is associated with a specific single-particle state $|p'\rangle$. Observe that the Fock matrix $\boldsymbol{F}$ contains precisely the matrix elements of the one-body Hamiltonian $\hat{H}_1^\Phi$ relative to the Fermi vacuum $|\Phi\rangle$ in the original basis $|p\rangle$ Eq. 2.4, and the HF energy E_Φ is exactly the zero-body component in Eq. 2.4.

4.1.2 HF equations in J-scheme

In this work, we use the implicit-J convention of Sec. 3.12.3 to describe J-scheme equations.

In J-scheme, the HF energy of Eq. 4.2 is given by

$$E_\Phi = \sum_{i'\backslash} j^2_{i'} \langle i'|\hat{H}_1|i'\rangle + \frac{1}{2}\sum_{j_{i'j'}} \sum_{i'j'\backslash} j^2_{i'j'} \langle i'j'|\hat{H}_2|i'j'\rangle \tag{4.6}$$

and the Fock matrix of Eq. 4.5 is given by

$$F_{pq} = \langle p|\hat{H}_1|q\rangle + \sum_{j_{pr}} {}^{rs} \sum_{i'\backslash} \frac{j^2_{pr}}{j^2_p} C^*_{ri'} \langle pr|\hat{H}_2|qs\rangle C_{si'} \tag{4.7}$$

The transformation equations of Eq. 4.3 remain superficially identical to M-scheme.

4.1.3 Solving HF equations

Aside from trivial cases that are analytically solvable, the HF equation is generally solved numerically using an iterative algorithm. We begin with an initial guess $\boldsymbol{C}^{(k)}$ on the k-th iteration, which is fed into Eq. 4.5 to produce the Fock matrix. This is then used in Eq. 4.4, which leads to a standard eigenvalue problem from which $\boldsymbol{C}^{(k+1)}$ arises as the matrix of eigenvectors and $\boldsymbol{\varepsilon}^{(k+1)}$ as the vector of eigenvalues. This process can be repeated indefinitely until $\boldsymbol{C}$ approaches a fixed point (self-consistency). While in theory it is possible for the solution to never reach a fixed point, or that it may require an unfeasibly large number of iterations, in practice this naive approach can adequately provide solutions for many cases. In other cases where it is insufficient, methods such as direct inversion of the iterative subspace (DIIS) [Pul80; Pul82], Broyden's method [Bro65], or even *ad hoc* linear mixing can improve and accelerate convergence greatly. Therefore, the possibility of slow or non-convergence is generally not a concern in practice.

For the initial guess, we simply use the ground state of our noninteracting Hamiltonian, thus $\boldsymbol{C}^{(0)} = \mathbf{1}$, the identity matrix. At each iteration, we calculate the sum of the Lagrange multipliers $\boldsymbol{\varepsilon}$ as a diagnostic for convergence: as the iteration approaches convergence, the change in the sum per iteration should decrease rapidly.

4.1.4 Post-HF methods

Since HF restricts the ground state to merely a single Slater determinant of single-particle states, it cannot provide an exact solution to a problem where multi-particle correlations are present even if the single-particle basis is not truncated (infinite in size). The discrepancy between the HF energy and the exact ground state energy is often referred to as the **correlation energy**, by definition. The focus of post-HF methods such as IM-SRG or CC is to add corrections beyond mean-field approximations such as HF.

To make use of the HF solution as the reference state for post-HF calculations, we transform the matrix elements via Eq. 4.3. In effect, this means we are no longer operating within the harmonic oscillator single-particle basis, but rather a HF-optimized single-particle basis. However, we will omit the prime symbols as the post-HF methods are generic and can be used in any basis, whether optimized by HF or not.

A commonly used post-HF method is the **Møller–Plesset perturbation theory at second order** (MP2) [MP34], which adds an energy correction to the Hartree–Fock result:

$$\Delta E = \frac{1}{4} \sum_{ij\backslash ab} \frac{V_{ijab} V_{abij}}{\Delta_{ijab}} \tag{4.8}$$

where V_{ijab} are two-body matrix elements of the HF-transformed Hamiltonian, Δ denotes the

Møller–Plesset energy denominators [MP34],

$$\Delta_{q_1 \cdots q_k p_1 \cdots p_k} = \sum_{i=1}^{k} (\varepsilon_{q_i} - \varepsilon_{p_i}) \tag{4.9}$$

and ε_p are HF orbital energies. In J-scheme, the MP2 correction is given by:

$$\Delta E = \frac{1}{4} \sum_{j_{ij}} \sum_{ij \backslash ab} J_{ij}^2 \frac{V_{ijab} V_{abij}}{\Delta_{ijab}} \tag{4.10}$$

The MP2 calculation is extremely simple and cheap, thus it is often used as a diagnostic for estimating the strength of correlations that remain unaccounted for.

A more sophisticated post-HF method is the *coupled-cluster* (CC) method [SB09], in which the N-particle correlated wave function $|\Psi\rangle$ is expressed as the exponential ansatz,

$$|\Psi\rangle = \mathrm{e}^{\hat{T}} |\Phi\rangle$$

Here, $|\Phi\rangle$ is a Slater determinant reference state such as the one from HF and $\hat{T}$ is the *cluster operator*, which is a sum of k-particle-k-hole excitation operators of various k. The Schrödinger equation with this ansatz becomes a set of non-linear algebraic equations (*coupled-cluster equations*) with which one can solve for the matrix elements of $\hat{T}$ and thereby obtain information about $|\Psi\rangle$.

The focus of this work is not on the coupled-cluster method, however, although we will present benchmarks of it for comparison. Our focus is on the in-medium similarity renormalization group (IM-SRG) method.

4.2 Similarity renormalization group methods

4.2.1 Free space SRG

The central theme of similarity renormalization group (SRG) methods is the application of a continuous sequence of unitary transformations on the Hamiltonian to evolve it into a band- or block-diagonal form. This allows the decoupling of a small, designated **model space** from its larger complementary space. The problem can thus be truncated to the small model space while preserving a large amount of information about the system. See for examples [Keh06; Her+16; HLK17] for derivations and calculational details.

The sequence of transformations is parameterized by a continuous variable s known as the **flow parameter**. Without loss of generality, we can define $s = 0$ to be the beginning of this sequence, thus $\hat{H}(0)$ is simply the original Hamiltonian. At any value of s, the evolving Hamiltonian $\hat{H}(s)$ is related to the original Hamiltonian by

$$\hat{H}(s) = \hat{U}(s)\hat{H}(0)\hat{U}^\dagger(s)$$

where $U(s)$ is a unitary operator that describes the product of all such transformations since $s = 0$. Taking the derivative with respect to s, we obtain:

$$\frac{\mathrm{d}}{\mathrm{d}s}\hat{H}(s) = \frac{\mathrm{d}\hat{U}(s)}{\mathrm{d}s}\hat{H}(0)\hat{U}^\dagger(s) + \hat{U}(s)\hat{H}(0)\frac{\mathrm{d}\hat{U}^\dagger(s)}{\mathrm{d}s}$$

If we define the **generator** $\hat{\eta}(s)$ as

$$\hat{\eta}(s) = \frac{\mathrm{d}\hat{U}(s)}{\mathrm{d}s}\hat{U}^\dagger(s) \tag{4.11}$$

we find that it is antihermitian as a result of the unitarity of $\hat{U}(s)$:

$$\hat{\eta}(s) + \hat{\eta}^\dagger(s) = \frac{\mathrm{d}}{\mathrm{d}s}\left(\hat{U}(s)\hat{U}^\dagger(s)\right) = 0$$

From this property we can derive a differential equation known as the **SRG flow equation**:

$$\frac{\mathrm{d}\hat{H}(s)}{\mathrm{d}s} = [\hat{\eta}(s), \hat{H}(s)] \tag{4.12}$$

This equation allows $\hat{H}(s)$ to be evaluated without explicitly constructing the full transformation $\hat{U}(s)$. The focus is instead shifted to the operator $\hat{\eta}(s)$, the generator of the transformation. When $\hat{\eta}(s)$ is *multiplicatively* integrated (**product integral**), the full unitary transformation $\hat{U}(s)$ is recovered:

$$\hat{U}(s') = \lim_{\Delta s \longrightarrow 0} \prod_{i=1}^{\longrightarrow n} \mathrm{e}^{\hat{\eta}(s_i)\Delta s} \tag{4.13}$$

where $s_i = i\Delta s$, $n = \lfloor s'/\Delta s \rfloor$, $\lfloor x \rfloor$ denotes the floor of x, and the product is ordered from left ($i = 1$) to right ($i = n$). This is the formal solution to the linear differential equation Eq. 4.11. The product integral in Eq. 4.13 may also be reinterpreted as "s-ordering" [Rei13] in analogy to time-ordering from quantum field theory.

The power of SRG methods lies in the flexibility of the generator $\hat{\eta}$, which is usually chosen in an s-dependent manner. In particular, it is often dependent on the evolving Hamiltonian $\hat{H}(s)$. The operator $\hat{\eta}$ determines which parts of the Hamiltonian matrix would become suppressed by the evolution, which are usually considered "off-diagonal" in an abstract sense. The "off-diagonal" parts could be elements far away from the matrix diagonal, in which case the evolution drives the matrix towards a band-diagonal form. Or, the "off-diagonal" parts could be elements that couple

the ground state from the excited state, in which case the evolution drives the matrix towards a block-diagonal form that isolates the ground state. Or, the "off-diagonal" could be literally the elements that do not lie on the diagonal, in which case the evolution would simply diagonalize the Hamiltonian. Through different choices of $\hat{\eta}$, the SRG evolution can be controlled and adapted to the features of a particular problem.

4.2.2 In-medium SRG

The SRG flow equation Eq. 4.12 can be solved in the second quantization formalism described in Sec. 2.2, where field operators are defined with respect to the physical vacuum state. However, since the basis of a many-body problem grows factorially with the number of particles and the size of the model space, the applicability of the naive (free-space) SRG method is restricted to comparatively small systems. A more practical approach is to perform the evolution *in medium* [Keh06], i.e. using a many-body Slater determinant as a reference state, which is assumed to be a fair approximation to the true ground state. This gives rise to the in-medium similarity renormalization group (IM-SRG) method [TBS12; Her+16; HLK17].

We begin by decomposing the Hamiltonian $\hat{H}$ into normal-ordered components relative to an appropriately chosen reference state (Fermi vacuum) $|\Phi\rangle$:

$$\hat{H} = E_\Phi + \sum_{pq} H^\Phi_{pq} :\hat{a}^\dagger_p \hat{a}_q: + \frac{1}{4} \sum_{pqrs} H^\Phi_{pqrs} :\hat{a}^\dagger_p \hat{a}^\dagger_q \hat{a}_s \hat{a}_r: + \cdots \tag{4.14}$$

where E_Φ is the energy of the reference state and $H^\Phi_{p_1 \cdots p_k q_1 \cdots q_k}$ are matrix elements of the k-body component $\hat{H}^\Phi_k$. In IM-SRG we work exclusively with matrix elements relative to $|\Phi\rangle$, thus we will omit the Φ suffix in this section.

The use of a Hamiltonian with components relative to the Fermi vacuum may seem like a

triviality – it is still the same $\hat{H}$ after all. However, this makes a critical difference when the operator expressions are *truncated*, i.e. higher-body components discarded from the computation for efficiency reasons. By normal-ordering the components relative to a reference state $|\Phi\rangle$, we preserve a portion of the higher-body contributions within the lower-body operators, significantly decreasing the importance of higher-body operators.

Higher-body operators arise from integrating the flow equations of Eq. 4.12, which is one of the main challenges of the SRG method. With each evaluation of the commutator, the Hamiltonian gains terms of increasingly higher order, and these induced contributions will in subsequent integration steps feed back into terms of lower order. Thus, the higher-body contributions are not irrelevant to the final solution even if only the ground state energy (zero-body component) is desired.

Computationally, higher-body terms rapidly become unfeasible to handle: naive storage of the matrix elements of k-body operator requires an exponentially increasing amount of memory,

$$\mathcal{O}(n_{\mathrm{b}}^{2k})$$

where n_{b} is the number of single-particle basis states. Moreover, the flow equations are capable of generating an infinite number of higher-body terms as the Hamiltonian evolves. To make the method tractable, the IM-SRG flow equations must be closed by truncating the equations to a finite order. We call this **operator truncation**.

In this work, we truncate both $\hat{H}$ and $\hat{\eta}$ at the two-body level, leading to an approach known as **IM-SRG(2)**. This normal-ordered two-body approximation appears to be sufficient in many cases and has yielded excellent results for several nuclei [TBS11; Rot+12; Her+16].

Operator truncation is but one out of the two primary sources of error in this method. The

other source of error comes from **basis truncation**: the size of the single-particle is finite and therefore does not encompass the full infinite-dimensional Hilbert space. This is a concern for any finite-basis approach, including HF, IM-SRG, CC, and many others. This source of error can be reduced by increasing the size of the basis at the expense of greater computational effort, albeit the cost increases much less rapidly in this direction. The CPU cost of IM-SRG methods is polynomial with respect to the number of states in the single-particle basis n_b. For IM-SRG(2) in particular, the CPU cost scales roughly as

$$\mathcal{O}(n_\mathrm{b}^6)$$

This is comparable to coupled cluster singles-and-doubles (CCSD), which also scales as $\mathcal{O}(n_\mathrm{b}^6)$.

The commutator in the flow equations Eq. 4.12 ensures that the evolved state $\hat{U}(s)|\Phi\rangle$ consists of *linked diagrams* only [SB09]. This indicates that IM-SRG is a size-extensive [Bar81] method by construction, even if the operators are truncated.

An accurate and robust solver is required to solve ordinary differential equation (ODE) in Eq. 4.12. In particular, the solver must be capable of handling the stiffness that often arises in such problems. For our numerical experiments, we used a high-order ODE solver algorithm by L. F. Shampine and M. K. Gordon [SG75], which is a multistep method based on the implicit Adams predictor-corrector formulas. Its source code is freely available [ODE; SgOde].

IM-SRG has relations to several other well-known methods of quantum chemistry such as coupled cluster theory [SB09], canonical transformation theory [Whi02; NYC10], the irreducible/anti-Hermitian contracted Schrödinger equation approach [Maz07b; Maz07a], and the driven similarity renormalization group method [Eva14]. These connections are explored in more detail in [Her17].

4.2.3 IM-SRG generators

With an appropriate choice of the generator $\hat{\eta}$, the evolved state $\hat{U}(s)|\Phi\rangle$ will gradually approach a more "diagonal" form. If the "diagonal" form decouples the ground state from the excited states, then $\hat{U}(\infty)|\Phi\rangle$ would yield the exact ground state solution of the problem if no operator or basis truncations are made. In particular, $E_\Phi(\infty)$ would be the exact ground state energy.

The traditional **Wegner generator** [Weg01] is defined as

$$\hat{\eta}^{\mathrm{Wg}} = [\hat{H}^{\mathrm{d}}, \hat{H} - \hat{H}^{\mathrm{d}}] = [\hat{H}^{\mathrm{d}}, \hat{H}]$$

where $\hat{H}^{\mathrm{d}}$ denotes the "diagonal" part of the Hamiltonian and $\hat{H} - \hat{H}^{\mathrm{d}}$ denotes the "off-diagonal" part. This is in the abstract sense described at the end of Section Sec. 4.2.1. Since $\hat{H}$ depends on the flow parameter s, so does $\hat{\eta}$ in general.

Since $\hat{\eta}^{\mathrm{Wg}}$ is a commutator between two Hermitian operators, it is antihermitian as required for a generator. Additionally, it can be shown that the commutator has the property of suppressing off-diagonal matrix elements as the state evolves via the flow equation [Keh06], as we would like. Matrix elements "far" from the diagonal – i.e. where the Hamiltonian couples states with large energy differences – are suppressed much faster than those "close" to the diagonal.

There exist several other generators in literature. One choice, proposed by White [Whi02], makes numerical approaches much more efficient. The problem with the Wegner generator is the widely varying decaying speeds of the Hamiltonian matrix elements. Terms with large energy separations from the ground state are suppressed initially, followed by those with smaller energy separations. This leads to stiffness in the flow equation, which in turn causes numerical difficulties when solving the set of coupled differential equations.

The **White generator** takes an alternative approach, which is well suited for problems where

one is mainly interested in the ground state of a system. Firstly, instead of driving all off-diagonal elements of the Hamiltonian to zero, the generator focuses exclusively on those that are coupled to the reference state $|\Phi\rangle$ so as to decouple the reference state from the remaining Hamiltonian. This reduces the amount of change done to the Hamiltonian, reducing the accuracy lost from the operator truncation. Secondly, the rate of decay in Hamiltonian matrix elements are approximately normalized by dividing the generator matrix elements by an appropriate factor. This ensures that the affected elements decay at approximately the same rate, reducing the stiffness of the flow equations.

The White generator is explicitly constructed in the following way [TBS11; Whi02]:

$$\hat{\eta}^{\mathrm{Wh}} = \hat{\eta}' - \hat{\eta}'^{\dagger} \tag{4.15}$$

where $\hat{\eta}'$ is defined as

$$\hat{\eta}' = \sum_{i\backslash a} \frac{H_{ai}}{\tilde{\Delta}_{ai}} :\hat{a}_a^\dagger \hat{a}_i: + \frac{1}{4} \sum_{ij\backslash ab} \frac{H_{abij}}{\tilde{\Delta}_{abij}} :\hat{a}_a^\dagger \hat{a}_b^\dagger \hat{a}_j \hat{a}_i: + \cdots$$

The symbol $\tilde{\Delta}$ denotes the **Epstein–Nesbet energy denominators** [Eps26; Nes55; SB09], defined as

$$\begin{aligned}
\tilde{\Delta}_{ai} &= E_{\Phi_{ai}} - E_\Phi \\
&= \Delta_{ai} - H_{aiai} \\
\tilde{\Delta}_{abij} &= E_{\Phi_{abij}} - E_\Phi \\
&= \Delta_{abij} + H_{abab} - H_{aiai} - H_{bibi} + H_{ijij} - H_{ajaj} - H_{bjbj} \\
\tilde{\Delta}_{a_1 \ldots a_k i_1 \ldots i_k} &= E_{\Phi_{a_1 \ldots a_k i_1 \ldots i_k}} - E_\Phi
\end{aligned}$$

whereas Δ denotes the Møller–Plesset energy denominators [MP34] defined in Eq. 4.9. White generators can also use Møller–Plesset energy denominators directly in lieu of Epstein–Nesbet energy denominators [Her+16], which leads to a slightly different variant of the White generator. In our calculations, we use exclusively Epstein–Nesbet denominators.

Compared to the Wegner generator, where the derivatives of the final flow equations contain cubes of the Hamiltonian matrix elements (i.e. each term contains a product of 3 one-body and/or two-body matrix elements), the elements in White generators contribute only linearly. This reduces the stiffness in the differential equation, providing a net increase in computational efficiency as stiff ODE solvers tend to be slower and consume more memory.

4.2.4 IM-SRG(2) equations

In the 2-body operator truncation scheme, the generator $\hat{\eta}$ can be written as a generic 2-body operator:

$$\hat{\eta} = \sum_{pq} \eta_{pq} : \hat{a}_p^\dagger \hat{a}_q : + \frac{1}{4} \sum_{pqrs} \eta_{pqrs} : \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r :$$

where η_{pq} and η_{pqrs} respectively are its one- and two-body matrix elements normal ordered relative to $|\Phi\rangle$ and subject to the antihermittivity constraint.

The main complication of the IM-SRG flow equation Eq. 4.12 lies in the commutator,

$$[\hat{\eta}, \hat{H}] = \hat{\eta}\hat{H} - \hat{H}\hat{\eta}$$

By expanding the commutator diagrammatically, we find that all terms where $\hat{\eta}$ and $\hat{H}$ are connected (disconnected diagrams) vanish because they commute. The remaining terms are

simply the linked products between the two operators, which we denote $\hat{C}(\hat{\eta}, \hat{H})$, in either order:

$$[\hat{\eta}, \hat{H}] = \hat{C}(\hat{\eta}, \hat{H}) - \hat{C}(\hat{H}, \hat{\eta})$$

Consider a generic linked product $\hat{C}(\hat{A}, \hat{B})$ where $\hat{C}$ is a (0, 1, 2, 3)-operator given by

$$\hat{C} = C_\Phi + \sum_{pq} C_{pq} : \hat{a}_p^\dagger \hat{a}_q : + \frac{1}{4} \sum_{pqrs} C_{pqrs} : \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_s \hat{a}_r : + \frac{1}{36} \sum_{pqrstu} C_{pqrstu} : \hat{a}_p^\dagger \hat{a}_q^\dagger \hat{a}_r^\dagger \hat{a}_u \hat{a}_t \hat{a}_s :$$

To write out the linked product, we start considering all possible Hugenholtz skeletons[2] $\hat{C}^{cab}$ where a is the rank of the first operator from $\hat{A}$, b is the rank of the second operator from $\hat{B}$, and c is the rank of the product diagram. This leads to the following terms:

$$\hat{C}^0 = \hat{C}^{011} + \hat{C}^{022} \qquad \hat{C}^1 = \hat{C}^{111} + \hat{C}^{112} + \hat{C}^{121} + \hat{C}^{122}$$

$$\hat{C}^2 = \hat{C}^{212} + \hat{C}^{221} + \hat{C}^{222} \qquad \hat{C}^3 = \hat{C}^{322}$$

We can then elaborate on this by considering all possible assignments of the arrows. We classify these diagrams as $\hat{C}^{cabd}$ where d is the number of arrows going from $\hat{B}$ toward $\hat{A}$, which is also

[2] Hugenholtz skeletons are Hugenholtz diagrams without arrows.

the number of particle lines.

$$\hat{C}^{011} = \hat{C}^{0110} \qquad \hat{C}^{022} = \hat{C}^{0220}$$

$$\hat{C}^{111} = \hat{C}^{1110} + \hat{C}^{1111} \qquad \hat{C}^{112} = \hat{C}^{1120}$$

$$\hat{C}^{121} = \hat{C}^{1210} \qquad \hat{C}^{122} = \hat{C}^{1220} + \hat{C}^{1221}$$

$$\hat{C}^{212} = \hat{C}^{2120} + \hat{C}^{2121} \qquad \hat{C}^{221} = \hat{C}^{2210} + \hat{C}^{2211}$$

$$\hat{C}^{222} = \hat{C}^{2220} + \hat{C}^{2221} + \hat{C}^{2222} \qquad \hat{C}^{322} = \hat{C}^{3220} + \hat{C}^{3221}$$

Finally, we write out the diagrams as,

$$C_{\Phi}^{0110} = +\sum_{i\backslash a} A_{ia} B_{ai} \qquad C_{\Phi}^{0220} = +\frac{1}{4} \sum_{ij\backslash ab} A_{ijab} B_{abij}$$

$$C_{pq}^{1110} = -\sum_{i\backslash} A_{iq} B_{pi} \qquad C_{pq}^{1111} = +\sum_{\backslash a} A_{pa} B_{aq}$$

$$C_{pq}^{1120} = +\sum_{i\backslash a} A_{ia} B_{apiq} \qquad C_{pq}^{1210} = +\sum_{i\backslash a} A_{ipaq} B_{ai}$$

$$C_{pq}^{1220} = -\frac{1}{2} \sum_{ij\backslash a} A_{ijaq} B_{apij} \qquad C_{pq}^{1221} = +\frac{1}{2} \sum_{i\backslash ab} A_{ipab} B_{abiq}$$

$$C_{pqrs}^{2120} = -2\mathcal{A}_{rs} \sum_{i\backslash} A_{ir} B_{pqis} \qquad C_{pqrs}^{2121} = +2\mathcal{A}_{pq} \sum_{\backslash a} A_{pa} B_{aqrs}$$

$$C_{pqrs}^{2210} = -2\mathcal{A}_{pq} \sum_{i\backslash} A_{iqrs} B_{pi} \qquad C_{pqrs}^{2211} = +2\mathcal{A}_{rs} \sum_{\backslash a} A_{pqas} B_{ar}$$

$$C_{pqrs}^{2220} = +\frac{1}{2} \sum_{ij\backslash} A_{ijrs} B_{pqij} \qquad C_{pqrs}^{2221} = -4\mathcal{A}_{pq}\mathcal{A}_{rs} \sum_{i\backslash a} A_{iqar} B_{apis}$$

$$C_{pqrs}^{2222} = +\frac{1}{2} \sum_{\backslash ab} A_{pqab} B_{abrs}$$

$$C_{pqrstu}^{3220} = -9\mathcal{A}_{pqr}\mathcal{A}_{stu} \sum_{i\backslash} A_{iqst} B_{priu} \qquad C_{pqrstu}^{3221} = +9\mathcal{A}_{pqr}\mathcal{A}_{stu} \sum_{\backslash a} A_{pqat} B_{arsu}$$

Fig. 4.1 shows these diagrams in diagrammatic form.

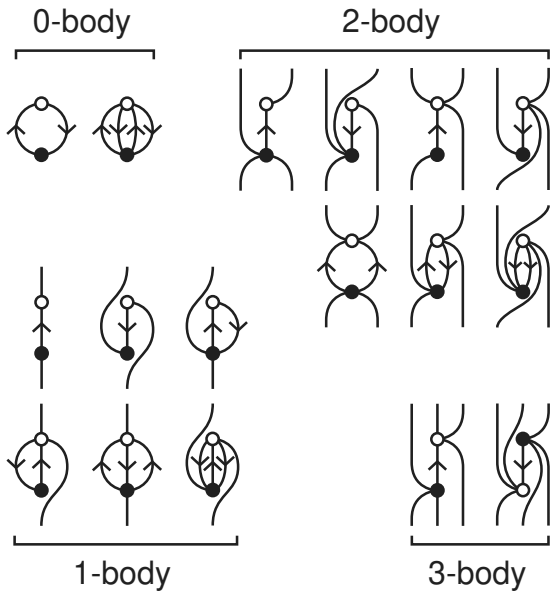

Figure 4.1: Hugenholtz diagrams representing the linked product $\hat{C}(\circ, \bullet)$ in the IM-SRG flow equation, with open circles representing $\hat{A}$ and filled circles representing $\hat{B}$. We omit diagrams that are related by permutations among the external bra lines or among the external ket lines.

4.2.5 IM-SRG(2) equations in J-scheme

Once again, we use the implicit-J convention (Sec. 3.12.3) to write J-scheme equations. Although some equations in J-scheme appear superficially identical to those in M-scheme, they are not interpreted in the same way due to the lack of m-type variables in J-scheme.

The Epstein–Nesbet energy denominators that arise in White generators contain two-body terms that cannot be expressed in J-scheme. As a practical workaround, one could replace occurrences of H_{pqrs} in the denominator with the monopole matrix element

$$H^{\text{mono}}_{pqrs} = \frac{\sum_{j_{pq}} \check{j}^2_{pq} H_{pqrs}}{\sum_{j_{pq}} \check{j}^2_{pq} \left\{ \begin{matrix} j_p & j_q & j_{pq} \end{matrix} \right\}}$$

Unlike the usual matrix element H_{pqrs}, the monopole matrix element H^{mono}_{pqrs} does not depend on j_{pq}.

The replacement by monopole matrix elements leads to the following Epstein–Nesbet energy denominators:

$$\tilde{\Delta}^{\text{mono}}_{ai} = \Delta_{ai} - H^{\text{mono}}_{aiai}$$

$$\tilde{\Delta}^{\text{mono}}_{abij} = \Delta_{abij} + H^{\text{mono}}_{abab} - H^{\text{mono}}_{aiai} - H^{\text{mono}}_{bibi} + H^{\text{mono}}_{ijij} - H^{\text{mono}}_{ajaj} - H^{\text{mono}}_{bjbj}$$

These do result in a different White generator, however. The generator in M-scheme is no longer equivalent to that in J-scheme if monopole matrix elements are used.

Finally, here are the J-scheme IM-SRG(2) equations using the implicit-J convention (Sec. 3.12.3):

$$C^{0110}_{\Phi} = +\sum_{i\backslash a} \check{j}^2_i A_{ia} B_{ai} \qquad C^{0220}_{\Phi} = +\frac{1}{4}\sum_{j_{ij}} \sum_{ij\backslash ab} \check{j}^2_{ij} A_{ijab} B_{abij}$$

$$C^{1110}_{pq} = -\sum_{i\backslash} A_{iq} B_{pi} \qquad C^{1111}_{pq} = +\sum_{\backslash a} A_{pa} B_{aq}$$

$$C^{1120}_{pq} = +\sum_{j_{ap}} \sum_{i\backslash a} \frac{\check{j}^2_{ap}}{\check{j}^2_p} A_{ia} B_{apiq} \qquad C^{1210}_{pq} = +\sum_{j_{ip}} \sum_{i\backslash a} \frac{\check{j}^2_{ip}}{\check{j}^2_p} A_{ipaq} B_{ai}$$

$$C^{1220}_{pq} = -\frac{1}{2}\sum_{j_{ap}} \sum_{ij\backslash a} \frac{\check{j}^2_{ap}}{\check{j}^2_p} A_{ijaq} B_{apij} \qquad C^{1221}_{pq} = +\frac{1}{2}\sum_{j_{ip}} \sum_{i\backslash ab} \frac{\check{j}^2_{ip}}{\check{j}^2_p} A_{ipab} B_{abiq}$$

$$C^{2120}_{pqrs} = -2\mathcal{A}_{rs} \sum_{i\backslash} A_{ir} B_{pqis} \qquad C^{2121}_{pqrs} = +2\mathcal{A}_{pq} \sum_{\backslash a} A_{pa} B_{aqrs}$$

$$C^{2210}_{pqrs} = -2\mathcal{A}_{pq} \sum_{i\backslash} A_{iqrs} B_{pi} \qquad C^{2211}_{pqrs} = +2\mathcal{A}_{rs} \sum_{\backslash a} A_{pqas} B_{ar}$$

$$C^{2220}_{pqrs} = +\frac{1}{2}\sum_{ij\backslash} A_{ijrs} B_{pqij} \qquad \tilde{C}^{2221}_{psrq} = +4\sum_{i\backslash a} \tilde{A}_{iarq} \tilde{B}_{psia}$$

$$C^{2222}_{pqrs} = +\frac{1}{2}\sum_{\backslash ab} A_{pqab} B_{abrs}$$

where the tilde symbol ($\tilde{C}$) denotes non-antisymmetrized Pandya-coupled matrix elements (Sec. 3.12.2).

4.3 Quasidegenerate perturbation theory

The IM-SRG method provides a means to calculate the ground state energy of any system that is reasonably approximated by a single Slater determinant. This works well for closed-shell systems, but it does not provide a direct means to obtain the ground state energy of open-shell systems. While there exist more complicated multi-reference approaches to IM-SRG that seek to tackle the general problem [Her+16], we opted to use a perturbative approach, which is simple, inexpensive, and as we shall see from the results, quite effective for many problems.

Quasidegenerate perturbation theory (QDPT) [Lin74; Kva74] is an extension to the usual perturbation theory framework to support multiple reference states instead of just one. This is useful for solving *open-shell* systems in which there are multiple reference states sharing similar (quasidegenerate) or equal (degenerate) energies. It is particularly useful if the open-shell system is only a few particles away from a closed-shell system. We will focus primarily on states that are one particle different from a closed-shell system. Specifically, we wish to calculate **addition energies** ε_a and **removal energies** ε_i of such systems, defined as

$$\varepsilon_a = E_{\Phi_a} - E_\Phi \qquad\qquad \varepsilon_i = E_\Phi - E_{\Phi_i}$$

respectively.

As usual in perturbation theory, we start by splitting the Hamiltonian $\hat{H}$ into two components,

$$\hat{H} = \hat{H}^\circ + \hat{V}$$

Here $\hat{H}^\circ$ is the zeroth-order **model Hamiltonian** that is easy to solve (typically a non-interacting Hamiltonian) and $\hat{V}$ is the perturbation that makes the problem difficult. We choose *a few* of the

eigenstates of the model Hamiltonian as our set of model states $|u'^{\circ}\rangle$,

$$\hat{H}^{\circ}|u'^{\circ}\rangle = E^{\circ}_{u'}|u'^{\circ}\rangle$$

and we want to solve for the corresponding unknown eigenstates $|u\rangle$ of the full Hamiltonian,

$$\hat{H}|u\rangle = E_u|u\rangle$$

We define a **wave operator** $\hat{\Omega}$ that projects some set of states $|u^{\circ}\rangle$ from the model space to the true ground state $|u\rangle$ (i.e. of the full Hamiltonian):

$$|u\rangle = \hat{\Omega}|u^{\circ}\rangle \tag{4.16}$$

where $|u^{\circ}\rangle$ is taken to be a linear combination of our selection of model states $|u'^{\circ}\rangle$,

$$|u^{\circ}\rangle = \sum_{u'}|u'^{\circ}\rangle C_{u'u}$$

with $C_{u'u}$ being some coefficient matrix.

There is some freedom in the choice of the wave operator $\hat{\Omega}$. We assume it has the following form:

$$\hat{\Omega} = \hat{P} + \hat{Q}\hat{\Omega}\hat{P} \tag{4.17}$$

where

- $\hat{P}$ is a projection operator for the model space,

$$\hat{P} = \sum_u \hat{P}_u$$

- $\hat{P}_u$ is the projection operator for $|u^\circ\rangle$,

$$\hat{P}_u = |u^\circ\rangle\langle u^\circ|$$

- $\hat{Q}$ is the complement of $\hat{P}$,

$$\hat{Q} = 1 - \hat{P}$$

This definition of $\hat{\Omega}$ entails that the exact states $|u\rangle$ are no longer normalized but instead satisfy the so-called **intermediate normalization**,

$$\langle u|u^\circ\rangle = 1$$

From Eqns. 4.16, 4.17 we observe that

$$\hat{\Omega}\hat{H}^\circ = \hat{\Omega}\hat{P}\hat{H}^\circ\hat{P}\hat{\Omega} = \hat{\Omega}\hat{H}^\circ\hat{\Omega}$$

$$\hat{H}\hat{\Omega} = \sum_u \hat{\Omega}E_u\hat{\Omega}\hat{P}_u = \hat{\Omega}\hat{H}\hat{\Omega}$$

These equations can be used to simplify the commutator $[\hat{\Omega}, \hat{H}^\circ]$, leading to the **generalized Bloch equation** [LM86] that defines QDPT:

$$[\hat{\Omega}, \hat{H}^\circ] = (1 - \hat{\Omega})\hat{V}\hat{\Omega}$$

The commutator on the left may be "inverted" using the resolvent approach [SB09, p. 50], resulting in the relation:

$$\hat{Q}\hat{\Omega}\hat{P}_u = \hat{R}_u(1 - \hat{\Omega})\hat{V}\hat{\Omega}\hat{P}_u$$

where $\hat{R}_u$ is the resolvent,[3]

$$\hat{R}_u = \hat{Q}(E_u^\circ - \hat{Q}\hat{H}^\circ\hat{Q})^{-1}\hat{Q}$$

Now define $\hat{\Omega}$ as a series of terms of increasing order, quantified by the exponent (degree) of the perturbation $\hat{V}$,

$$\hat{\Omega} = \sum_{n=0}^{\infty} \hat{\Omega}^{(n)}$$

We can then derive a recursion relation that allows $\hat{\Omega}$ to be calculated to any order

$$\hat{\Omega}^{(n)} = \begin{cases} \hat{P} & \text{if } n = 0 \\ \sum_u \hat{R}_u \left(\hat{V}\hat{\Omega}^{(n-1)} + \sum_{k=1}^{n-1} \hat{\Omega}^{(k)}\hat{V}\hat{\Omega}^{(n-k-1)} \right) \hat{P}_u & \text{if } n > 0 \end{cases}$$

[3] In some literature, the resolvent $\hat{R}_u$ is denoted by

$$\frac{\hat{Q}}{E_u^\circ - \hat{H}^\circ}$$

Up to third order, we have

$$\hat{\Omega}^{(1)}\hat{P}_u = \hat{R}_u\hat{V}\hat{P}_u$$
$$\hat{\Omega}^{(2)}\hat{P}_u = \hat{R}_u\left(\hat{V}\hat{R}_u - \sum_v \hat{R}_v\hat{V}\hat{P}_v\right)\hat{V}\hat{P}_u$$
$$\hat{\Omega}^{(3)}\hat{P}_u = \hat{R}_u\Bigg(\hat{V}\hat{R}_u\hat{V}\hat{R}_u - \hat{V}\hat{R}_u\sum_v \hat{R}_v\hat{V}\hat{P}_v - \sum_v \hat{R}_v\hat{V}\hat{P}_v\hat{V}\hat{R}_u$$
$$- \sum_v \hat{R}_v\hat{V}\hat{R}_v\hat{V}\hat{P}_v + \sum_v \hat{R}_v\sum_w \hat{R}_w\hat{V}\hat{P}_w\hat{V}\hat{P}_v\Bigg)\hat{V}\hat{P}_u$$

We can define an **effective Hamiltonian**

$$\hat{H}^{\text{eff}} = \hat{P}\hat{H}\hat{\Omega}$$

which acts only in the model space but yields the correct eigenvalues of the full space,

$$\hat{H}^{\text{eff}}|u^\circ\rangle = E_u|u^\circ\rangle$$

Thus, the energy corrections are given by:

$$E_u^{(n)} = \begin{cases} \langle u^\circ|\hat{H}^\circ|u^\circ\rangle & \text{if } n = 0 \\ \langle u^\circ|\hat{V}\hat{\Omega}^{(n-1)}|u^\circ\rangle & \text{if } n > 0 \end{cases}$$

The coefficients $C_{u'u}$ are obtained by diagonalizing the effective Hamiltonian through the eigenvalue problem,

$$\sum_{v'}\langle u'^\circ|\hat{H}^{\text{eff}}|v'^\circ\rangle C_{v'u} = C_{u'u}E_u \tag{4.18}$$

4.3.1 QDPT equations

We now consider the application of QDPT to the treatment of addition and removal energies via the particle-hole formalism. Take each reference state to be a Slater determinant constructed by adding or removing a single particle u to an existing closed-shell Fermi vacuum $|\Phi\rangle$,

$$|u^{\circ}\rangle = |\Phi_u\rangle$$

Take note that in QDPT *Fermi vacuum* and *reference state* are no longer synonymous.

We choose u to be close to the Fermi level: it should be a single-particle state within an adjacent shell (**valence shell**). Therefore, the number of reference states in the model space of QDPT is equal to the number of particles in either the lowest unoccupied shell or the highest occupied shell of $|\Phi\rangle$, depending on whether we are considering addition or removal energies, respectively.

We can then express the perturbation expansion in terms of summations over matrix elements as we did for the IM-SRG flow equation. We will restrict ourselves to the case where the perturbation $\hat{V}$ is a two-body operator.

The second-order QDPT corrections of the **left-shift operator** (or **reaction operator**) $\hat{W} = \hat{H}^{\text{eff}} - \hat{H}^{\circ}$ are:

$$W_{pq}^{(2)} = +\frac{1}{2}\sum_{i\backslash ab}\frac{V_{ipab}V_{abiq}}{\Delta_{iqab}} - \frac{1}{2}\sum_{ij\backslash a}\frac{V_{ijaq}V_{apij}}{\Delta_{ijap}}$$

Here, V_{abiq} are matrix elements of the two-body operator $\hat{V}$ and Δ denotes Møller–Plesset denominators as defined in Eq. 4.9. These second-order corrections are depicted as perturbative diagrams (Sec. 2.7.1) in Fig. 4.2.

Third-order QDPT corrections are:

$$
\begin{aligned}
W^{(3)}_{pq} = & \\
&+ \frac{1}{4} \sum_{i\backslash abcd} \frac{V_{ipab} V_{abcd} V_{cdiq}}{\Delta_{iqab} \Delta_{iqcd}} - \frac{1}{4} \sum_{ij\backslash abc} \frac{V_{ijaq} V_{apbc} V_{bcij}}{\Delta_{ijap} \Delta_{ijbc}} - \frac{1}{4} \sum_{ij\backslash abc} \frac{V_{ijab} V_{abcq} V_{cpij}}{\Delta_{ijqabp} \Delta_{ijcp}} \\
&- \frac{1}{4} \sum_{ijkl\backslash a} \frac{V_{ijaq} V_{klij} V_{apkl}}{\Delta_{ijap} \Delta_{klap}} + \frac{1}{4} \sum_{ijk\backslash ab} \frac{V_{ipab} V_{jkiq} V_{abjk}}{\Delta_{iqab} \Delta_{jkab}} + \frac{1}{4} \sum_{ijk\backslash ab} \frac{V_{ijab} V_{kpij} V_{abkq}}{\Delta_{ijqabp} \Delta_{kqab}} \\
&- \frac{1}{2} \sum_{ijk\backslash ab} \frac{V_{ijab} V_{kpjq} V_{abik}}{\Delta_{ijqabp} \Delta_{ikab}} + \frac{1}{2} \sum_{ij\backslash abc} \frac{V_{ijab} V_{bpcq} V_{acij}}{\Delta_{ijqabp} \Delta_{ijac}} + \frac{1}{2} \sum_{ij\backslash abc} \frac{V_{ipaq} V_{ajbc} V_{bcij}}{\Delta_{ia} \Delta_{ijbc}} \\
&+ \frac{1}{2} \sum_{ij\backslash abc} \frac{V_{ijab} V_{abic} V_{cpjq}}{\Delta_{ijqabp} \Delta_{jqcp}} - \frac{1}{2} \sum_{ijk\backslash ab} \frac{V_{ipaq} V_{jkib} V_{abjk}}{\Delta_{ia} \Delta_{jkab}} - \frac{1}{2} \sum_{ijk\backslash ab} \frac{V_{ijab} V_{akij} V_{bpkq}}{\Delta_{ijqabp} \Delta_{kqbp}} \\
&+ \sum_{ij\backslash abc} \frac{V_{ipac} V_{jcbq} V_{abij}}{\Delta_{iqac} \Delta_{ijab}} + \sum_{ij\backslash abc} \frac{V_{ijab} V_{bpjc} V_{aciq}}{\Delta_{ijqabp} \Delta_{iqac}} + \sum_{ij\backslash abc} \frac{V_{ipac} V_{jabi} V_{bcjq}}{\Delta_{iqac} \Delta_{jqbc}} \\
&- \sum_{ijk\backslash ab} \frac{V_{ikaq} V_{ajib} V_{bpjk}}{\Delta_{ikap} \Delta_{jkbp}} - \sum_{ijk\backslash ab} \frac{V_{ikaq} V_{jpbk} V_{abij}}{\Delta_{ikap} \Delta_{ijab}} - \sum_{ijk\backslash ab} \frac{V_{ijab} V_{bkjq} V_{apik}}{\Delta_{ijqabp} \Delta_{ikap}}
\end{aligned}
$$

Perturbative diagrams (Sec. 2.7.1) of third-order corrections are also shown in Fig. 4.2.

One of the benefits of applying QDPT to an IM-SRG-evolved Hamiltonian is that many of the QDPT terms vanish. In IM-SRG, a generator that decouples the ground state energy is required to drive certain classes of matrix elements to zero. Consider for example the White generator, which eliminates matrix elements of the form:

$$V_{ijab} = V_{abij} = 0$$

This means certain kinds of vertices in the diagrams become forbidden, reducing the number of nonzero diagrams at third order from 18 to only four. Out of these four, two of them contribute only to the correction of hole states (removal energies), while the other two contribute only to the correction of the particle states (addition energies).

Note that the final step of diagonalizing the effective Hamiltonian (Eq. 4.18) is usually not

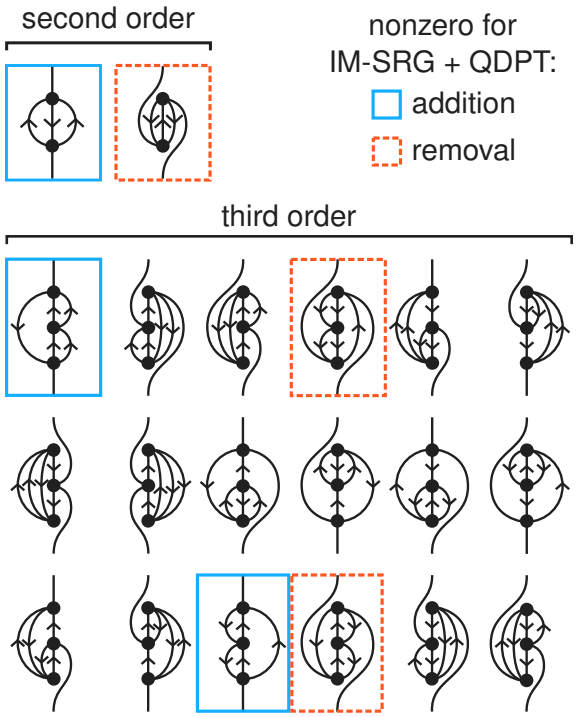

Figure 4.2: Perturbative Hugenholtz diagrams (Sec. 2.7.1) of the second- and third-order QDPT corrections. Denominator lines have been elided. When QDPT is performed on IM-SRG-evolved Hamiltonians, many of the diagrams vanish. The remaining nonvanishing diagrams for addition energy are highlighted in blue and for removal energy are highlighted in red.

needed for the calculation of single-particle energies with one valence shell as the matrix is often already diagonal due to conservation laws of the quantum system.

In J-scheme, the second-order corrections are:

$$W^{(2)}_{pq} = \frac{1}{2}\sum_{j_{ip}}\sum_{i\backslash ab}\frac{\hat{j}^2_{ip}}{\hat{j}^2_p}\frac{V_{ipab}V_{abiq}}{\Delta_{iqab}} - \frac{1}{2}\sum_{j_{ap}}\sum_{ij\backslash a}\frac{\hat{j}^2_{ap}}{\hat{j}^2_p}\frac{V_{ijaq}V_{apij}}{\Delta_{ijap}}$$

As usual, these equations use the implicit-J convention (Sec. 3.12.3).

For efficiency, the third-order corrections in J-scheme make use of the non-antisymmetrized Pandya-transformed matrix elements of $\hat{V}$, which are denoted $\tilde{V}_{psrq}$ (Sec. 3.12.2). Using these

matrix elements, we may write the third-order terms as:

$$
\begin{aligned}
W_{pq}^{(3)} = & \\
& + \frac{1}{4} \sum_{j_{ip}} \sum_{i\backslash abcd} \frac{j_{ip}^2}{j_p^2} \frac{V_{ipab} V_{abcd} V_{cdiq}}{\Delta_{iqab} \Delta_{iqcd}} - \frac{1}{4} \sum_{j_{ap}} \sum_{ij\backslash abc} \frac{j_{ap}^2}{j_p^2} \frac{V_{ijaq} V_{apbc} V_{bcij}}{\Delta_{ijap} \Delta_{ijbc}} \\
& - \frac{1}{4} \sum_{j_{cp}} \sum_{ij\backslash abc} \frac{j_{cp}^2}{j_p^2} \frac{V_{ijab} V_{abcq} V_{cpij}}{\Delta_{ijqabp} \Delta_{ijcp}} - \frac{1}{4} \sum_{j_{ap}} \sum_{ijkl\backslash a} \frac{j_{ap}^2}{j_p^2} \frac{V_{ijaq} V_{klij} V_{apkl}}{\Delta_{ijap} \Delta_{klap}} \\
& + \frac{1}{4} \sum_{j_{ip}} \sum_{ijk\backslash ab} \frac{j_{ip}^2}{j_p^2} \frac{V_{ipab} V_{jkiq} V_{abjk}}{\Delta_{iqab} \Delta_{jkab}} + \frac{1}{4} \sum_{j_{kp}} \sum_{ijk\backslash ab} \frac{j_{kp}^2}{j_p^2} \frac{V_{ijab} V_{kpij} V_{abkq}}{\Delta_{ijqabp} \Delta_{kqab}} \\
& - \frac{1}{2} \sum_{j_{kp} j_{ij}} \sum_{ijk\backslash ab} \frac{j_{kp}^2 j_{ij}^2}{j_p^2 j_j^2} \frac{V_{ijab} V_{kpjq} V_{abik}}{\Delta_{ijqabp} \Delta_{ikab}} + \frac{1}{2} \sum_{j_{bp} j_{ab}} \sum_{ij\backslash abc} \frac{j_{bp}^2 j_{ab}^2}{j_p^2 j_b^2} \frac{V_{ijab} V_{bpcq} V_{acij}}{\Delta_{ijqabp} \Delta_{ijac}} \\
& + \frac{1}{2} \sum_{j_{ip} j_{ij}} \sum_{ij\backslash abc} \frac{j_{ip}^2 j_{ij}^2}{j_p^2 j_i^2} \frac{V_{ipaq} V_{ajbc} V_{bcij}}{\Delta_{ia} \Delta_{ijbc}} + \frac{1}{2} \sum_{j_{cp} j_{ij}} \sum_{ij\backslash abc} \frac{j_{cp}^2 j_{ij}^2}{j_p^2 j_j^2} \frac{V_{ijab} V_{abic} V_{cpjq}}{\Delta_{ijqabp} \Delta_{jqcp}} \\
& - \frac{1}{2} \sum_{j_{ip} j_{ab}} \sum_{ijk\backslash ab} \frac{j_{ip}^2 j_{ab}^2}{j_p^2 j_a^2} \frac{V_{ipaq} V_{jkib} V_{abjk}}{\Delta_{ia} \Delta_{jkab}} - \frac{1}{2} \sum_{j_{bp} j_{ab}} \sum_{ijk\backslash ab} \frac{j_{bp}^2 j_{ab}^2}{j_p^2 j_b^2} \frac{V_{ijab} V_{akij} V_{bpkq}}{\Delta_{ijqabp} \Delta_{kqbp}} \\
& + \sum_{j_{cp}} \sum_{ij\backslash abc} \frac{j_{cp}^2}{j_p^2} \frac{\tilde{V}_{iacp} \tilde{V}_{cqbj} \tilde{V}_{bjia}}{\Delta_{iqac} \Delta_{ijab}} + \sum_{j_{cp}} \sum_{ij\backslash abc} \frac{j_{cp}^2}{j_p^2} \frac{\tilde{V}_{iabj} \tilde{V}_{bjcp} \tilde{V}_{cqia}}{\Delta_{ijqabp} \Delta_{iqac}} \\
& + \sum_{j_{cp}} \sum_{ij\backslash abc} \frac{j_{cp}^2}{j_p^2} \frac{\tilde{V}_{iacp} \tilde{V}_{jbia} \tilde{V}_{cqjb}}{\Delta_{iqac} \Delta_{jqbc}} - \sum_{j_{pk}} \sum_{ijk\backslash ab} \frac{j_{pk}^2}{j_p^2} \frac{\tilde{V}_{iaqk} \tilde{V}_{jbia} \tilde{V}_{pkjb}}{\Delta_{ikap} \Delta_{jkbp}} \\
& - \sum_{j_{pk}} \sum_{ijk\backslash ab} \frac{j_{pk}^2}{j_p^2} \frac{\tilde{V}_{iaqk} \tilde{V}_{pkbj} \tilde{V}_{bjia}}{\Delta_{ikap} \Delta_{ijab}} - \sum_{j_{pk}} \sum_{ijk\backslash ab} \frac{j_{pk}^2}{j_p^2} \frac{\tilde{V}_{iabj} \tilde{V}_{bjqk} \tilde{V}_{pkia}}{\Delta_{ijqabp} \Delta_{ikap}}
\end{aligned}
$$

Chapter 5

Application to quantum systems

The two main systems that we study in this work are quantum dots and nuclei. Quantum dots are a remarkably simple system for studying quantum phenomena and provide a testbed for both theoretical calculations and experimental measurements as the strength of its correlations can be easily tuned by adjusting the width of the external trap. In contrast, nuclei are natural, self-bound systems with a nuclear force that is both complicated and uncertain.

For simplicity, we begin with quantum dots and use it to test the effectiveness of our many-body methods. Ultimately, nuclei are the more challenging and intriguing system to study, and they tend to be more practically relevant.

5.1 Quantum dots

Quantum dots, also known as "artificial atoms", are prototypical quantum systems consisting of electrons confined by an external potential.

5.1.1 Quantum dot Hamiltonian

We will devote our focus on *circular* quantum dots, consisting of a collection of nonrelativistic electrons trapped in a two-dimensional harmonic oscillator potential, interacting through the standard Coulomb interaction. The parameters of the system are:

- N: the number of electrons,

- m: the mass of each electron,
- e: the charge of each electron,
- ϵ: the permittivity of the medium, and
- ω: the angular frequency of the harmonic oscillator potential.

The three basic components of the Hamiltonian are the kinetic energy $t(\boldsymbol{p})$, the potential energy $u(\boldsymbol{r})$, and the Coulomb interaction $v(R)$:

$$t(\boldsymbol{p}) = \frac{\boldsymbol{p}^2}{2m} \qquad u(\boldsymbol{r}) = \frac{m\omega^2 \boldsymbol{r}^2}{2} \qquad v(R) = \frac{e^2}{4\pi\epsilon|R|}$$

where $\boldsymbol{r}$ is the position of an electron relative to the center of the trap, $\boldsymbol{p}$ is its linear momentum, and R is the distance between two electrons. The kinetic and potential energies combine to form the standard harmonic oscillator Hamiltonian $h(\boldsymbol{r}, \boldsymbol{p})$:

$$h(\boldsymbol{r}, \boldsymbol{p}) = t(\boldsymbol{p}) + u(\boldsymbol{r})$$

The quantum many-body problem is described by the Hamiltonian

$$\hat{H} = \hat{H}_1 + \hat{H}_2$$

where $\hat{H}_1$ and $\hat{H}_2$ are respectively its one- and two-body components,[1] namely

$$\begin{aligned} \hat{H}_1 &= \sum_{\alpha=1}^{N} h(\hat{\boldsymbol{r}}_\alpha, \hat{\boldsymbol{p}}_\alpha) \\ \hat{H}_2 &= \sum_{\alpha=1}^{N} \sum_{\beta=1}^{\alpha-1} v(|\hat{\boldsymbol{r}}_\alpha - \hat{\boldsymbol{r}}_\beta|) \end{aligned} \tag{5.1}$$

[1] These components are defined relative to the physical vacuum.

The operator $\hat{\boldsymbol{r}}_\alpha$ is the position operator of the α-th particle,[2] and $\hat{\boldsymbol{p}}_\alpha = -\mathrm{i}\hat{\boldsymbol{\nabla}}_{\boldsymbol{r}_\alpha}$ is its momentum operator.

Even though the model system contains five parameters, many of them are redundant. With an appropriate choice of units, the number of parameters in the system can be reduced to just two. For this system, it is convenient to choose **atomic units** where

$$\hbar = m = e = 4\pi\epsilon = 1$$

Here, Hartree E_h is the unit of energy and Bohr radius a is the unit of length:

$$E_\mathrm{h} = m\left(\frac{e^2}{4\pi\epsilon\hbar}\right)^2 \qquad\qquad a = \frac{4\pi\epsilon\hbar^2}{me^2}$$

This leaves us with (N, ω) as the only two parameters needed to specify the quantum dot system, with ω in units of $E_\mathrm{h}/\hbar$.

5.1.2 Fock–Darwin basis

We will use the noninteracting part of the many-body Hamiltonian $\hat{H}_1$ to define the single-particle basis. The single-particle Hamiltonian is of the form:

$$\hat{h} = \frac{1}{2}\hat{\boldsymbol{\nabla}}^2 + \frac{1}{2}\omega^2\hat{\boldsymbol{r}}^2$$

In Cartesian coordinates, the two-dimensional harmonic oscillator is trivially reducible to the well-known one-dimensional problem. However, to exploit the circular symmetry, we prefer to use **Fock–Darwin states** [Foc28; Dar31], which are written in polar coordinates $\boldsymbol{r} = (r, \varphi)$. Such

[2] The ordering of the particle labels α is unimportant as they exist only for the purpose of bookkeeping.

states have the computationally useful property of conserving orbital angular momentum,

$$\hat{L}_3 = -\mathrm{i}\frac{\hat{\partial}}{\partial\varphi}$$

The Fock–Darwin wave functions can be decomposed into radial and angular components,[Loh10]

$$\begin{aligned} F_{nm_\ell}(r,\varphi) &= \sqrt{\frac{m\omega}{\hbar}} R_{n|m_\ell|}\left(\sqrt{\frac{m\omega}{\hbar}} r\right) A_{m_\ell}(\varphi) \\ R_{n\mu}(\rho) &= \sqrt{2}\mathrm{e}^{-\rho^2/2}\rho^\mu \bar{L}_n^\mu(\varrho^2) \\ A_{m_\ell}(\varphi) &= \frac{1}{\sqrt{2\pi}}\mathrm{e}^{\mathrm{i}m_\ell\varphi} \end{aligned} \tag{5.2}$$

in ordinary units. Here, $\sqrt{\hbar/m\omega}$ is the characteristic length of the harmonic oscillator, $\Gamma(x)$ is the gamma function, and $\bar{L}_n^\alpha(x)$ is the *normalized* variant of the associated Laguerre polynomial $L_n^\alpha(x)$ of degree n and parameter α [DLMF]:

$$\begin{aligned} \bar{L}_n^\alpha(x) &= \sqrt{\frac{n!}{\Gamma(n+\alpha+1)}} L_n^\alpha(x) \\ L_n^\alpha(x) &= \frac{1}{n!} x^{-\alpha}\mathrm{e}^x \frac{\mathrm{d}^n}{\mathrm{d}x^n}(\mathrm{e}^{-x}x^{\alpha+n}) \end{aligned} \tag{5.3}$$

The normalized Laguerre polynomials satisfy the following orthogonality relation:

$$\int_0^\infty u^\alpha \mathrm{e}^{-u} \bar{L}_m^{(\alpha)}(u)\bar{L}_n^{(\alpha)}(u)\,\mathrm{d}u = \delta_{mn}$$

The states are labeled by two quantum numbers: the principal quantum number $n \in \{0, 1, 2, \ldots\}$ and orbital angular momentum projection $m_\ell \in \{\ldots, -2, -1, 0, +1, +2, \ldots\}$. For a wave function, n indicates the degree of the Laguerre polynomial, whereas m_ℓ is the eigenvalue of $\hat{L}_3$.

Since electrons are spin-$\frac{1}{2}$ fermions, they can occupy either of the two possible spin states $\chi_{-\frac{1}{2}}$

or $\chi_{+\frac{1}{2}}$. Thus, every single-particle basis state $|nm_\ell m_s\rangle$ contains both a spatial component (5.2) and a spin component,

$$\langle r\varphi m_s'|nm_\ell m_s\rangle = F_{nm_\ell}(r,\varphi)\delta_{m_s m_s'} \tag{5.4}$$

Here we have introduced spin projection $m_s \in \left\{-\frac{1}{2}, +\frac{1}{2}\right\}$ as the third quantum number, which is the eigenvalue of the spin projection operator $\hat{S}_3$.

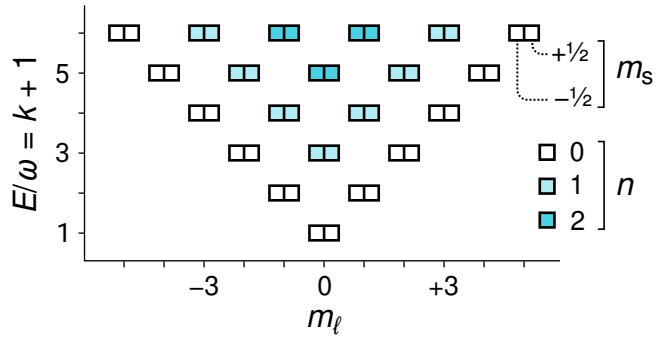

Figure 5.1: The 42 lowest single-particle states (the first 5 shells) in the 2D harmonic oscillator basis. Each box represents a single-particle state arranged by m_ℓ, m_s, and energy, and the up/down arrows indicate the spin of the states. Within each column, the principal quantum number n increases as one traverses upward.

The energy of the single-particle state $|nm_\ell m_s\rangle$ is given by

$$\varepsilon_{nm_\ell m_s} = (2n + |m_\ell| + 1)\hbar\omega \tag{5.5}$$

in ordinary units. These energies are degenerate with respect to the spin projection m_s as our Hamiltonian $\hat{h}$ does not distinguish between them. Additionally, they are degenerate with respect to the number of quanta k, defined as

$$k = 2n + |m_\ell| \tag{5.6}$$

We also call k the **shell index of the two-dimensional harmonic oscillator** as this nonnegative integer labels each shell starting from zero. The shells are equidistant with an energy spacing of $\hbar\omega$. This is depicted graphically in Fig. 5.1.

When the number of particles N satisfies $N = K_{\mathrm{F}}(K_{\mathrm{F}} + 1)$ for some nonnegative integer K_{F}, there would be just enough particles to form a closed-shell Slater determinant, leading to a unique, well-isolated ground state. These specific values of N form the **magic numbers** of this system. We call K_{F} the **number of filled shells** (or "Fermi level"). In particular, a single-particle state is occupied in the ground state Slater determinant if and only if $k < K_{\mathrm{F}}$, where k is the shell index of the single-particle state as defined in Eq. 5.6.

5.1.3 Coulomb interaction in the Fock–Darwin basis

For many-body calculations, we will need matrix elements of the Coulomb interaction in the Fock–Darwin basis that we chose. The antisymmetrized matrix elements, needed for Eq. 2.2, are given by

$$\begin{aligned}
&\langle (nms)_1 (nms)_2 | \hat{H}_2 | (nms)_3 (nms)_4 \rangle \\
&= \langle (nm)_1 (nm)_2 | \hat{H}_2 | (nm)_3 (nm)_4 \rangle \delta_{s_1 s_3} \delta_{s_2 s_4} \\
&\quad - \langle (nm)_1 (nm)_2 | \hat{H}_2 | (nm)_4 (nm)_3 \rangle \delta_{s_1 s_4} \delta_{s_2 s_3}
\end{aligned}$$

where for brevity we have relabeled the quantum numbers with $m = m_\ell$ and $s = m_s$, and

$$\begin{aligned}
&\langle (nm)_1 (nm)_2 | \hat{H}_2 | (nm)_3 (nm)_4 \rangle \\
&= \frac{e^2}{4\pi\epsilon} \iint \frac{F_{(nm)_1}(\boldsymbol{r}) F_{(nm)_2}(\boldsymbol{r}') F_{(nm)_3}(\boldsymbol{r}) F_{(nm)_4}(\boldsymbol{r}')}{|\boldsymbol{r} - \boldsymbol{r}'|} \, \mathrm{d}^2 r \, \mathrm{d}^2 r'
\end{aligned} \tag{5.7}$$

denotes the *non-antisymmetrized* matrix element in ordinary units, and $F_{nm}(\boldsymbol{r})$ denotes a Fock–Darwin wave function.

Analytically, the integral may be evaluated [AM98] as

$$
\begin{aligned}
&\frac{\langle (nm)_1 (nm)_2 | \hat{H}_2 | (nm)_3 (nm)_4 \rangle}{\sqrt{\hbar\omega E_\mathrm{h}}} \\
&= \delta_{m_1+m_2, m_3+m_4} \sqrt{\prod_{i=1}^{4} \frac{n_i!}{(n_i+|m_i|)!}} \sum_{j_1=0}^{n_1} \sum_{j_2=0}^{n_2} \sum_{j_3=0}^{n_3} \sum_{j_4=0}^{n_4} \\
&\quad \left(\prod_{i=1}^{4} \frac{(-)^{j_i}}{j_i!} \binom{n_i + |m_i|}{n_i - j_i} \right) \frac{1}{2^{(G+1)/2}} \sum_{l_1=0}^{\gamma_1} \sum_{l_2=0}^{\gamma_2} \sum_{l_3=0}^{\gamma_3} \sum_{l_4=0}^{\gamma_4} \\
&\quad\quad \delta_{l_1+l_2, l_3+l_4} (-)^{\gamma_2+\gamma_4-l_2-l_4} \Gamma\left(1 + \frac{\Lambda}{2}\right) \Gamma\left(\frac{G-\Lambda+1}{2}\right) \prod_{i=1}^{4} \binom{\gamma_i}{l_i}
\end{aligned}
$$

$$
\begin{aligned}
G &= \sum_{i=1}^{4} \gamma_1 \\
\Lambda &= \sum_{i=1}^{4} l_1 \\
\gamma_1 &= j_1 + j_3 + \frac{|m_1| + m_1}{2} + \frac{|m_3| - m_3}{2} \\
\gamma_2 &= j_2 + j_4 + \frac{|m_2| + m_2}{2} + \frac{|m_4| - m_4}{2} \\
\gamma_3 &= j_3 + j_1 + \frac{|m_3| + m_3}{2} + \frac{|m_1| - m_1}{2} \\
\gamma_4 &= j_4 + j_2 + \frac{|m_4| + m_4}{2} + \frac{|m_2| - m_2}{2}
\end{aligned}
$$

However, the analytic approach is rather inefficient: it has effectively 7 nested summations, which means the computational cost of *each* matrix element grows as $\mathcal{O}(k^7)$ where k is the number of shells. It is also prone to precision losses due to the highly oscillatory terms.

A more effective way to compute the integral is through the technique described in [Kva08] as implemented in the OpenFCI package. By transforming product states $|(nm)_1 \otimes (nm)_2\rangle$ into their

center-of-mass frame, one arrives at a radial integral

$$C^{\mu}_{nn'} = 2(-)^{n+n'} \int_0^{\infty} R^{2\mu} \bar{L}^{\mu}_{n}(R^2) \bar{L}^{\mu}_{n'}(R^2) v(\sqrt{2}R) \mathrm{e}^{-R^2} R \, \mathrm{d}R$$

where $\bar{L}^{\alpha}_{n}(x)$ is the normalized associated Laguerre polynomial defined in Eq. 5.3 and $v(R)$ is our central interaction, although this technique generalizes to many kinds of central interactions. The radial integral may be calculated *exactly* using Gauss–Hermite quadrature of sufficiently high order. The results are transformed back into the laboratory frame using Talmi–Brody–Moshinsky transformation brackets [Tal52; BM67; Mos59].

5.2 Nuclei

The nuclear many-body problem is challenging problem due to the strength of as well as the uncertainty in the interaction.

5.2.1 The nuclear Hamiltonian

A nucleus is a self-bound system of nucleons: neutrons and protons. The nucleons interact with each other through the nuclear interaction. The parameters of the system are:

- A: the number of nucleons,
- N: the number of neutrons,[3]
- Z: the number of protons,
- m: the mass of each nucleon, and
- $\hat{V}$: the nuclear interaction.

Note that for simplicity we treat neutrons and protons as having the same mass, which is

[3]Note the difference in notation compared to quantum dots.

generally adequate given the current levels of accuracy.

The many-body Hamiltonian consists of two components, the relative kinetic energy $\hat{T}^{\text{rel}}$ and the nuclear interaction $\hat{V}$,

$$\hat{H} = \hat{T}^{\text{rel}} + \hat{V}$$

The relative kinetic energy is given by:

$$\hat{T}^{\text{rel}} = \sum_{\alpha=1}^{A} \frac{\hat{\boldsymbol{p}}_\alpha^2}{2m} - \frac{1}{2mA}\left(\sum_{\alpha=1}^{A} \hat{\boldsymbol{p}}_\alpha\right)^2 = \sum_{\alpha=1}^{A}\sum_{\beta=1}^{\alpha-1} \frac{(\hat{\boldsymbol{p}}_\alpha - \hat{\boldsymbol{p}}_\beta)^2}{2m}$$

In second quantization, this can be written as a combination of a one-body and a two-body operator:

$$\hat{t}_1^{\text{rel}}(\boldsymbol{p}) = \left(1 - \frac{1}{A}\right)\frac{\hat{\boldsymbol{p}}^2}{2m} \qquad \hat{t}_2^{\text{rel}}(\boldsymbol{p}, \boldsymbol{p}') = -\frac{\hat{\boldsymbol{p}} \cdot \hat{\boldsymbol{p}}'}{mA}$$

$$\hat{T}_1^{\text{rel}} = \sum_{\alpha=1}^{A} \hat{t}_1^{\text{rel}}(\boldsymbol{p}_\alpha) \qquad \hat{T}_2^{\text{rel}} = \sum_{\alpha=1}^{A}\sum_{\beta=1}^{\alpha-1} \hat{t}_2^{\text{rel}}(\boldsymbol{p}_\alpha, \boldsymbol{p}_\beta)$$

The units we use in nuclear theory are MeV, fm, and combinations thereof. We set the constants $\hbar = c = 1$.

5.2.2 The nuclear interaction

The nuclear interaction $\hat{V}$ is generally quite complicated. Typically it is either a two-body operator, or a combination of two-body and three-body operators. In principle, this interaction could even have higher-body operators than three.

Unlike quantum dots, there are many possible choices for $\hat{V}$, none of which could be considered canonical. This is due to current limitations in the understanding of the nuclear interaction.

So far, the state of the art in nuclear interactions lies in chiral effective field theory (chiral EFT or χ-EFT) [ME11; EHM09], which uses a power-counting scheme to build an effective Lagrangian with nucleons as the degrees of freedom. Most of the coupling constants of this Lagrangian are not known *a priori* and must be determined by fitting experimental data.

Chiral EFT interactions are characterized by the level of truncation in the power-counting scheme, as well as the cut-off momentum Λ used for regularization. A commonly used choice in the literature is the nucleon-nucleon interaction of [EM03] computed to the next-to-next-to-next-to-leading order ($\mathrm{N^3LO}$) with a momentum cutoff at $\Lambda = 500\,\mathrm{MeV}$. This interaction has proven to be quite accurate in practice.

Many kinds of nuclear interactions, including chiral EFT ones, are typically *hard* in that they couple low-momentum states with high-momentum ones, caused by the presence of a strongly repulsive core [BFS10]. This can significantly hinder the convergence of many-body methods, necessitating the use of a large single-particle basis.

To mitigate this, free-space SRG may be used to renormalize the interaction, decoupling the low-momentum states from high-momentum states, thereby *softening* the interaction. This extra preprocessing step confers significant benefits to the convergence of many-body methods, reducing computational cost [BFP07].

The SRG softening is characterized by the flow parameter s_{SRG}, or equivalently by the momentum

$$\lambda_{\mathrm{SRG}} = \frac{1}{\sqrt[4]{s_{\mathrm{SRG}}}}$$

which is not to be confused with the cutoff momentum Λ in chiral EFT. In our calculations, we choose $s_{\mathrm{SRG}} = 0.0625\,\mathrm{fm}^4$ or equivalently $\lambda_{\mathrm{SRG}} = 2\,\mathrm{fm}^{-1}$.

5.2.3 Spherical harmonic oscillator basis

The standard basis used for nuclei is that of the three-dimensional harmonic oscillator, chosen for both its analytic properties and similarity to the nuclear problem. Note unlike quantum dots, the spherical oscillator basis are *not* the eigenstates of the one-body part of the nuclear Hamiltonian.

The frequency ω of the basis is not associated with the physical system, unlike in the case of quantum dots. Therefore, it adds an additional, arbitrary parameter for the nuclear many-body problem. The parameter can be used to evaluate the quality of the result: in a perfect calculation where the results are well converged, there should be no dependence on ω since it is not a physical parameter.

There are several kinds of bases for the three-dimensional harmonic oscillator. We choose the spherical version to take advantage of angular momentum symmetries. They are given by:

$$\psi_{n\ell m_\ell}(r,\theta,\varphi) = \sqrt{2\left(\frac{m\omega}{\hbar}\right)^{\ell+3/2} r^\ell \mathrm{e}^{-m\omega r^2/(2\hbar)} \bar{L}_n^{(\ell+1/2)}\left(\frac{m\omega r^2}{\hbar}\right)} Y_{\ell m_\ell}(\theta,\varphi)$$

where n is the principal quantum number, ℓ is the orbital angular momentum magnitude, m_ℓ is the orbital angular momentum projection, $\bar{L}_n^\alpha$ are normalized associated Laguerre polynomials defined in Eq. 5.3, and $Y_{\ell m_\ell}$ are spherical harmonics.

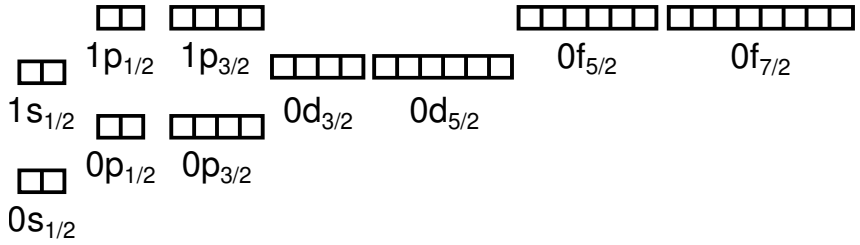

Figure 5.2: Shell structure of the spherical harmonic oscillator

The energy is given by:

$$E_{n\ell m_\ell} = \hbar\omega\left(e + \frac{3}{2}\right)$$

where e is the **shell index of the three-dimensional harmonic oscillator**:

$$e = 2n + \ell \tag{5.8}$$

which counts shells starting at zero. Compare with Eq. 5.6, which is for the two-dimensional harmonic oscillator. These shells are also equidistant with an energy spacing of $\hbar\omega$.

Nucleons are associated with two additional non-spatial quantum numbers: spin projection $m_s = \pm 1/2$ and isospin projection $m_t = \pm 1/2$. This means a proper nucleonic state should have 5 quantum numbers:

$$|n\ell m_\ell m_s m_t\rangle$$

In practice, however, it is more beneficial to use LS coupled states as the nuclear Hamiltonian conserves not $\hat{\boldsymbol{L}}$ but $\hat{\boldsymbol{J}}$:

$$|n\ell j m_j m_t\rangle = \sum_{m_\ell m_s} |n\ell m_\ell m_s m_t\rangle\langle \ell m_\ell \frac{1}{2} m_s | j m_j\rangle$$

The corresponding harmonic oscillator shell structure is shown in Fig. 5.2 using spectroscopic notation $n\ell_j$, where s, p, d, f correspond to $\ell = 0$, $\ell = 1$, $\ell = 2$, and $\ell = 3$.

The nuclear Hamiltonian also conserves parity, thus it can be convenient to use the parity

quantum number $\pi = (-)^{\ell}$ in lieu of ℓ, since from π and j one can recover ℓ:

$$|n\pi j m_j m_t\rangle \simeq |n\ell j m_j m_t\rangle$$

For calculations, it is necessary to truncate the single-particle harmonic oscillator basis. The simplest way is to impose a limit on the number of shells by the **maximum shell index** parameter $e_{\max}$:

$$e \leq e_{\max} \tag{5.9}$$

We can analogously limit n and/or ℓ:

$$n \leq n_{\max} \qquad \ell \leq \ell_{\max} \tag{5.10}$$

On two-particle states constructed from the harmonic oscillator, we may also impose a limit through the $E_{\max}$ parameter (not to be confused with energy):

$$e_1 + e_2 \leq E_{\max} \tag{5.11}$$

Currently, for our calculations we only use the $e_{\max}$ truncation. In future, we may require other forms of truncation in addition to $e_{\max}$ to keep the basis from growing too rapidly as we explore higher numbers of shells.

5.2.4 Matrix elements of kinetic energy

For calculations in a harmonic oscillator basis, one often needs to compute matrix elements of the kinetic energy,

$$\langle n' \ell' m'_\ell | \frac{\hat{\boldsymbol{p}}^2}{2m} | n \ell m_\ell \rangle$$

Since linear momentum squared $\hat{\boldsymbol{p}}^2$ is a scalar, it commutes with angular momentum $\hat{\boldsymbol{L}}$ and therefore matrix elements with differing ℓ and m_ℓ must vanish.

For now, let us use natural units – we will return to ordinary units shortly. To simplify the calculation, observe that

$$\frac{\hat{\boldsymbol{p}}^2}{2} = \hat{H} - \frac{\hat{\boldsymbol{r}}^2}{2}$$

This way, instead of calculating an integral involving a Laplacian, we can get away with an integral involving just r^2.

Using the recurrence relation of Laguerre polynomials,

$$\left(2n + \ell + \frac{3}{2} - r^2\right) L_n^{(\ell+1/2)}(r^2) = (n+1) L_{n+1}^{(\ell+1/2)}(r^2) + \left(n + \ell + \frac{1}{2}\right) L_{n-1}^{(\ell+1/2)}(r^2)$$

we can expand

$$\begin{aligned} \hat{\boldsymbol{r}}^2 | n \ell m_\ell \rangle = &- \sqrt{n\left(n + \ell + \frac{1}{2}\right)} | (n-1) \ell m_\ell \rangle \\ &+ \left(2n + \ell + \frac{3}{2}\right) | n \ell m_\ell \rangle \\ &- \sqrt{(n+1)\left(n + \ell + \frac{3}{2}\right)} | (n+1) \ell m_\ell \rangle \end{aligned}$$

Hence, in ordinary units the matrix elements of potential energy are

$$\langle n'\ell' m'_\ell | \frac{m\omega^2 \hat{\boldsymbol{r}}^2}{2} | n\ell m_\ell \rangle = \frac{\delta_{\ell'\ell}\delta_{m'_\ell m_\ell}\hbar\omega}{2}\left(\left(2n + \ell + \frac{3}{2}\right)\delta_{n'n} - \sqrt{\eta\left(\eta + \ell + \frac{1}{2}\right)}\delta_{|n'-n|1} \right)$$

where $\eta = \max\{n', n\}$ is the larger of the two. From here it is straightforward to compute the matrix elements of kinetic energy,

$$\langle n'\ell' m'_\ell | \frac{\hat{\boldsymbol{p}}^2}{2m} | n\ell m_\ell \rangle = \frac{\delta_{\ell'\ell}\delta_{m'_\ell m_\ell}\hbar\omega}{2}\left(\left(2n + \ell + \frac{3}{2}\right)\delta_{n'n} + \sqrt{\eta\left(\eta + \ell + \frac{1}{2}\right)}\delta_{|n'-n|1} \right) \tag{5.12}$$

with the same η as defined earlier.

Chapter 6

Implementation

We now discuss the details of our specific implementation of the many-body methods that we have discussed. In our experience, we find a dearth of such documentation in scientific literature, potentially leading to the loss of valuable practical knowledge. We hope readers will find this information helpful for either developing their own codes, reproducing our results, or utilizing our code.

The many-body methods in this work are implemented as part of the Lutario project [Lutario], an open-source library written in Rust, dual licensed under the permissive MIT [MIT] and Apache 2.0 licenses [Apache2]. Lutario implements a J-scheme framework for many-body calculations, upon which HF, Møller–Plesset perturbation theory to second order (MP2), IM-SRG(2), and QDPT3 are written. The code supports several systems, including quantum dots and nuclei, whose results we discuss in detail in the next chapter. The code also contains implementations of infinite matter and homogeneous electron gas, but we have not included any of those results in this work.

6.1 Programming language

Rust is a systems programming language focused on memory safety and performance [Rust; RustBook]. It is intended to fulfill a niche similar to those of other close-to-metal languages such as C, C++, or Fortran. These languages are characterized by extremely low overhead on all operations and they offer a high degree of manual control over memory usage and layout. This contrasts with the higher level, garbage-collected languages such as C#, Java, Python, or R, where

the manual memory management is eschewed in favor of an automatic memory management with the aid of a garbage collector (GC) that reclaims unused memory without the programmer's assistance.

Rust differs from mainstream close-to-metal languages like C or C++ in a few critical ways:

- The Rust language is partitioned into *safe* and *unsafe* subsets. While the unsafe subset is as flexible and performant as C, the safe subset sacrifices a bit of flexibility or performance so as to prevent the dreaded *undefined behavior* that plagues similar languages. Use of the safe subset is heavily encouraged by design.
- Among many ideas adopted from research in functional programming languages, it offers a novel *affine* type system augmented with *borrowing* semantics, allowing easy management of scarce resources such as memory and file handles.

In a way, the design choices of Rust is a natural consequence of making safety a top priority and then making pragmatic trade-offs between flexibility and performance.

Nonetheless, our motivation for choosing Rust is not simply because of safety, which is certainly important but not the most important concern in numerical software. Instead, we chose Rust for a combination of reasons:

- Rust includes a subset of the features commonly found in functional programming languages, greatly enhancing productivity. These features include closures, algebraic data types, and traits. While they have also made their way to other languages such as C++, Java, or C#, which were originally object-oriented but have become increasingly multi-paradigm, Rust was originally designed with these features from the outset, and as a result they integrate better into the language's design, whereas older languages have had to retrofit these features.
- Rust comes with an official package manager Cargo [Cargo] with high adoption among the community. It makes it extremely easy to build and install Rust *crates* (packages) from the

Rust package registry [CratesIo], while encouraging sharing and reuse of code.

- Rust has a flourishing and close-knit community from many diverse backgrounds, ranging from system programmers to high-performance computing specialists. This aids adoption of the young language and offers a helpful environment for learners.

With that being said, there are also reasons to not choose Rust:

- Rust remains a very young language by any measure. While the language is officially stable, some portions remain under experimentation. Large parts of the library ecosystem are still in their infancy stages, so there is a high risk of immature, rapidly evolving libraries.
- Rust puts safety above all else. As a result, highly performant but unsafe Rust code can be awkward and non-idiomatic to write. This can often be mitigated with the design of safer data abstractions, but these are also active areas of research.
- The lack of a garbage collector requires significant compromises on abstractions in Rust. For example, closures in Rust are more complex (but also more performant) than those in languages with GC such as Python or JavaScript. One should consider whether these complications are a worthwhile trade-off.

We will discuss the project with a perspective heavily influenced by Rust, but conceptually many of these apply to C and C++ just as well. We will use Rust snippets to illustrate concepts, but we expect any reader familiar with C++ should have little trouble adjusting to the slightly different notation.

In the next two subsections we will provide some motivations to the design of Rust, which can be safely skipped.

6.1.1 Undefined behavior

Undefined behavior (UB) is any behavior that is not defined by the language. Compilers are not obliged to detect whether a program has UB. If a program does have UB, the compiled program is not guaranteed to function correctly at all. It should not be confused with *implementation-defined* behavior, in which the behavior is allowed to vary from platform to platform but must remain documented and predictable.

In C and C++, the list of potential UB is numerous [C11, Annex J.2, p. 557 - 571]. To name a few:

- *buffer overflow*: use of memory beyond the range that was allocated;
- *null pointer dereference*: attempting to dereference an invalid (*null*) pointer;
- *use after free*: use of memory that has already been deallocated;
- *use of uninitialized data*: attempting to read uninitialized variables or arrays
- *data races*: use of the same memory location from multiple threads without proper synchronization, in which at least one of the them is performing a write; and
- *signed arithmetic overflow*: when the result of an arithmetic operation is too large or too negative to fit in the signed integer type.

In software infrastructure, UB can be a bountiful source for security vulnerabilties. In high-performance computing (HPC), the concern of UB lies less so in security, but more in the risk of incorrect computations with possibly subtle and/or non-deterministic effects. This is especially pernicious as many optimizing compilers for C, C++, and Fortran *take for granted that UB never occur*, leading to miscompilations when they do inevitably occur as a result of programmer error. The following C program gives an example of miscompilation due to UB:

```
int main(int argc, char **argv)
{
```

```
    if (argc == 1) {
        int *p;
        return *p;
    }
    return 42;
}
```

If the program is executed with no arguments, then `argc` is 1.[1] In that scenario, the program has UB: one is not permitted to dereference an uninitialized pointer `p`. Naively, one would expect an uninitialized variable to contain a random memory address, which is highly likely to be unallocated. Therefore, one would expect the program to crash with a segmentation fault (memory access violation) with high probability. Indeed this is what typically happens if the program is compiled without optimizations (the so-called `-O0` compiler flag).

However, when compiled *with* optimizations at level 1 (`-O1`) or higher under Clang or GNU C Compiler (GCC),[2] the program would simply exit silently with `42`, despite `argc` being 1. It is as if the compiler had entirely excised the `if` block from the code, treating `argc == 1` to be an unfulfillable condition because `return *p` has undefined behavior, which the compiler assumes is not supposed to ever happen.

In most languages, the programmer may reduce the risk of undefined behavior through appropriate discipline, defensive checks at run time, or the use of safer abstractions. A large fraction of these errors are avoided entirely through automatic memory management.

[1] Note that `argc` counts the name of the program as an additional argument.
[2] This was tested on Clang 5.0.1 and GCC 7.2.1.

6.1.2 Uniqueness and borrowing

Rust tackles UB from a different angle than most languages: it tries to prevent such mistakes from happening through static analysis of the code (the type system, in particular). This is achieved through two unconventional features adapted from its predecessor Cyclone [Jim+02].

The first idea is that of **uniqueness**. Some forms of data are designated as unique, which means once they are consumed or given away they cannot be used again – the compiler assures this. With the guarantee that a piece of data is unique, one effectively has exclusive control over it. Specifically:

- We can modify its contents without the possibility that another agent might accidentally observe the changes. In particular, if we destroy the object, no-one – not us nor anyone else – can use it again, preventing use-after-free bugs.
- We can be certain that its contents will stay the same unless we change it or relinquish control to another part of the program. This is extremely beneficial for not only optimization purposes, but also for readability.
- No other thread has this data, so there is no possibility of data races.

Uniqueness is a powerful guarantee, but it can also be very limiting. To overcome this Rust offers a complementary feature called **borrowing**, where a unique data is temporarily yielded to another agent for a limited amount of time. This duration of time is known as the **lifetime** of the borrow. Borrowing is classified into two kinds:

- When data is *mutably borrowed*, the borrower will be granted temporary but exclusive control over the data. The borrower is free to do anything it pleases with the data, but it must guarantee that under all circumstances the data remains valid when the lifetime ends.[3] During the lifetime of the borrow, the lender is denied all access to the data and cannot lend

[3] It can, for example, destroy the data object, but it must immediately recreate a similar one in its place.

it again until the end of the lifetime.

- Alternatively, one can grant a *shared borrow* of a data, which yields restricted access of the data to the borrower, which typically means the data becomes read-only. During the lifetime of the borrow, the lender can continue granting shared borrows of the data, or access the data directly under the same restrictions.

This leads to a general programming principle referred to as *aliasing XOR mutability*: data should be modified only if it has exclusive control over the data. While this principle is not a hard and fast rule, it can aid both readability and compiler optimizations. With the idea of *exclusive control* encoded within the type system, Rust makes this principle enforceable by the compiler.

The downside of this approach lies in the complications that uniqueness and borrowing introduce to the language and data abstractions. Programming with uniqueness types and borrowing remains fairly novel and under-explored in practice.

6.2 Structure of the program

Calculations in our many-body program follows a linear pipeline:

1. Basis: Set up data structures needed to organize matrix elements.
2. Input: Read and/or compute Hamiltonian matrix elements.
3. HF: Compute coefficient matrix and HF-transformed Hamiltonian.
4. Normal ordering: Obtain Hamiltonian relative to Fermi vacuum.
5. IM-SRG: Evolve Hamiltonian using IM-SRG.
6. QDPT: Compute perturbative corrections to addition and removal energies.

The first two steps are highly dependent on the quantum system involved, whereas the remaining steps are designed to be independent of the details of any particular system.

We emphasize that the design of the program grew out of the needs of the program rather

than some idealistic vision. From our experience, attempting to dictate an "intuitive" structure to programs generally leads to leaky abstractions and lackluster performance. It is more preferable to allow the program to evolve organically to meet its own computational demands, and develop abstractions and models around the natural flow of data to aid human comprehension.

6.3 External libraries

We utilize several external libraries in our program. Noteworthy ones include:

- BLAS (Basic Linear Algebra Subprograms [Law+79]) is used for its vector and matrix operations, especially the GEMM (General Matrix-Matrix Multiplication) routine. Note that we do not use the *reference* implementation of BLAS from Netlib. We instead use highly optimized implementations such as OpenBLAS [XQS17; GG08; Wan+13].
- LAPACK (Linear Algebra Package, [And+99]) is used for solving the eigenvalue problem in the Hartree–Fock method.
- The Shampine–Gordon ODE solver [SG75; ODE; SgOde] is used for solving the IM-SRG flow equation.
- The `wigner-symbols` library [Wei99; Wei98; WSR] is used to calculate of angular momentum (re)coupling coefficients needed for J-scheme.
- The non-cryptographic Fowler–Noll–Vo (FNV) [FNV] hash algorithm is used for hash tables. Rust's default choice is more secure, but slower.

6.4 Basis and data layout

A critical question in computational many-body theory is how one should store the matrix elements, which can be numerous. Generally speaking, while there is no one-size-fits-all solution, there are

a few distinct storage layouts that are of use to our three many-body methods. We must also be wary of introducing too many storage layouts, which would introduce bloat and redundancies to our code, reducing compilation speed, and hindering comprehension and maintenance.

One of the first choices lies in the scheme: M-scheme or J-scheme? Since we wish to study nuclei and similar systems, using J-scheme is necessary as the performance of M-scheme code is noticeably worse even for something as small as helium. The divergence in performance will only worsen as one adds more particles and/or shells. But there is also a trade-off: J-scheme code can be more difficult to understand and more difficult to verify. In fact, an easy way to verify J-scheme code would be to perform computations using both J- and M-scheme and compare their results.

One might be tempted to write code that performs both, but fortunately this is mostly unnecessary. J-scheme code can be used to perform *pseudo-M-scheme* calculations by associating each particle with a fictitious `j` quantum number that is always zero, which must not be confused with the physical j quantum number. There are certain optimizations that can be done when `j` is always zero, but this suffices for verifying the correctness of the code.

For codes up to two-body interactions with real matrix elements, we use four separate kinds of operators:

- zero-body operator
- standard-coupled one-body operator
- standard-coupled two-body operator
- Pandya-coupled two-body operator

Zero-body operators are simply floating-point numbers. All other operators are stored as block-diagonal matrices to exploit the sparsity of the matrix. Specifically, because of conservation

laws such as

$$[\hat{H}, \hat{J}_3] = 0$$

our Hamiltonian matrices are guaranteed to have a certain block diagonal form:

$$m_j \neq m'_j \implies \langle m_j \alpha | \hat{H} | m'_j \beta \rangle = 0$$

6.4.1 Matrix types

The most basic data type is that of a matrix. A (dense) matrix containing entries of type `T`, denoted `Mat<T>`, is a combination of three items,[4]

```
struct Mat<T> {
    ptr: *mut T          // data pointer
    num_rows: usize,     // number of rows
    num_cols: usize,     // number of columns
}
```

- Here, `ptr` is declared to have type `*mut T`, namely a mutable pointer to `T`.[5] This is conventional but also somewhat deceptive, because we are actually storing a whole *array* of `T` objects at the location. The pointer simply provides the address to the first entry in this array, assuming the array is at least one element long.
- The dimensions are declared to have type `usize`,[6] which is the pointer-sized unsigned integer type conventionally used to store lengths of data in memory.

[4] We aim to use valid Rust code throughout to illustrate concepts, but the actual implementation may differ slightly in details.

[5] This is equivalent to `(T *)` in C or C++.

[6] This is equivalent to `size_t` in C or C++.

We use zero-based indices throughout the discussion.

There are two common matrix layout conventions: **row-major** (colloquially known as **C order**), where the matrix is laid out row by row, or **col-major** (**Fortran order**), where the matrix is laid out column by column. In both cases, the index $f(i, j)$ of an element within the array at `ptr` is given by the equation

$$f(i, j) = in + j \tag{6.1}$$

What differs is the interpretation of the variables:

- In the row-major convention, i is the row index, n is the number of columns, and j is the column index.
- In the column-major convention, i is the column index, n is the number of rows, and j is the row index.

In our code, we adhere to the row-major convention.

We use the `Mat<T>` data type to represent an *owning* matrix: it has exclusive ownership of its contents. When a `Mat<T>` object is destroyed, its associated memory is automatically deallocated by the destructor we implemented.

To share the matrix or submatrices of it, we introduce two separate types `MatRef<'a, T>` (shared matrix reference) and `MatMut<'a, T>` (mutable matrix reference). The reference types both have a **lifetime parameter** `'a` that determines the lifetime of the borrow. Unlike `Mat<T>`, we introduce an additional field to the contents of `MatRef<T>` and `MatMut<T>`:

```
struct MatRef<T> {        // or MatMut<T>
    ptr: *const T         // data pointer
    num_rows: usize,      // number of rows
```

```
    num_cols: usize,      // number of columns
    stride: usize,        // gap between each row
}
```

The `stride` parameter allows us to extract references of submatrices (incomplete matrices). This is useful because in many situations we only want to operate on, say, the hole states but not the particle states, or vice versa. In this case, the indexing formula in Eq. 6.1 is interpretedly differently: n is now the *stride* of the matrix.

We also use a triangular matrix data type `TriMat<T>`, which can represent not just triangular matrices but also ±-symmetric and ±-Hermitian matrices. In a (non-strict) triangular matrix data type, we store the diagonal and all elements above it, or the diagonal and all elements below it. Because we chose row-major matrices, the latter convention turns out to be more convenient, as we can write the indexing formula as:

$$g(i,j) = \binom{i+1}{2} + \binom{j}{1} = \frac{(i+1)i}{2} + j$$

which is independent of the matrix's dimensions. This formula readily generalizes to higher-rank simplex-shaped tensors.

A block-diagonal matrix is conceptually a matrix composed to square matrices arranged along the diagonal. In terms of data, a block-diagonal matrix is simply an array of matrices. In Rust, this can be represented by the nested type `Vec<Mat<T>>`, where `Vec<M>` denotes a growable array[7] of objects of type `M`.

Each block (or **channel** as we often call them in code) is indexed by l (**channel index**) and each element is indexed by a triplet (l, u, v), with u being the row index within the block and v

[7] This is approximately equivalent to `std::vector<T>` in C++.

being the column index within the block. The indices u and v are known as **auxiliary indices**. The size of blocks may vary within the matrix. Programmatically, block-diagonal matrices are represented by an array of matrices. In Rust, this would be `Vec<Mat<f64>>` where `Mat<T>` denotes our own custom matrix data type with entries of type `T`.

Effectively, all operations on block-diagonal matrices are performed block-wise. For example, matrix multiplication between two block matrices A^l_{uv} and B^l_{uv} can be written as:

$$C^l_{uw} = \sum_v A^l_{uv} B^l_{vw}$$

which is equivalent to

$$\boldsymbol{C}^l = \sum_v \boldsymbol{A}^l \boldsymbol{B}^l$$

This is a very convenient computational property. It provides an avenue for the parallelization of block-diagonal matrix multiplication, since the block operations are independent of each other.

We can generalize block-diagonal matrices such that the blocks are no longer required to be square nor do they have to lie on the diagonal. Instead, they simply need to be arranged so as to touch each other at their top-left/bottom-right corners. Implementation of operations on these generalized block-diagonal matrices remains identical.

There are additional optimizations one can apply to the layout of block-diagonal matrices. One can, for example, pack the contents of all matrices into a single contiguous array and store the matrix dimensions in a separate array along with offsets to each of these blocks. The array of offsets $h(l)$, which we call **block offsets**, is given by the formula

$$h(l) = \sum_{l'=0}^{l-1} e(l')$$

where $e(l)$ is the extent of the l-th block, which for an ordinary rectangular block with dimensions $m(l) \times n(l)$ is simply $e(l) = m(l)n(l)$. For convenience, we allow the indices of $h(l)$ to range from 0 to l rather than 0 to $l - 1$ as would be typical with zero-based indexing. The value of h_l is simply the length of the entire array. To save memory, the block offsets can be shared between matrices that have precisely the same layout of blocks.

6.4.2 Basis charts

A row index pair (l, u) can be considered an abstract label for a left basis vector in this matrix, whereas a column index pair (l, v) can be considered a label for a right basis vector.

For a one-body operators, each index pair corresponds to a one-particle state in J-scheme,

$$|j\kappa\mu\rangle$$

with j being the angular momentum magnitude, κ representing all remaining conserved quantum numbers, and μ representing all remaining non-conserved quantum numbers. Since J-scheme states are reduced states, the angular momentum projection m is absent entirely. The combination of (j, κ) is conserved, thus we have the bijection

$$l \simeq (j, \kappa)$$

in the one-particle basis. In code, we can store this bijection using a special bidirectional lookup table that we call a **chart**, which is a pair of two data structures:

```
struct Chart<T> {
    encoder: HashMap<T, usize>,
```

```
    decoder: Vec<T>,
}
```

The encoder is a hash table that maps from a `T` object into an index, whereas the decoder is a vector that maps from an index to a `T` object. Here, `T` can be any hashable object with an equality relation. In this case, we choose `T = (Half<i32>, K)`, where `Half<i32>` is our custom data type for representing half-integers like j, and `K` is the type of κ.

In our code, we do not store κ directly, but represent κ using another abstract index k isomorphic to κ. This design allows the type of the $l \simeq (j, k)$ chart to be completely independent of the type of κ, avoiding code bloat due to monomorphization. The rationale for this is that most operations in many-body theory only require knowledge of the total angular momentum magnitude j and not of κ.

There is also a bijection between μ and u but it is l-dependent,

$$(l, u) \simeq (l, \mu)$$

This bijection is needed to recover the non-conserved quantum numbers and is needed to interpret matrix elements (e.g. reading input matrix elements, displaying output), but is irrelevant within the core of the many-body methods.

For two-body operators, each index pair corresponds to an unnormalized two-particle state in J-scheme,

$$|j_{12}\kappa_{12}j_1\kappa_1\mu_1 j_2\kappa_2\mu_2\rangle$$

Since $j_{12}\kappa_{12}$ is conserved, we have the bijection,

$$l_{12} \simeq (j_{12}, \kappa_{12})$$

akin to one-particle states.

Notice that the two-particle state can be compressed to

$$|j_{12}p_1p_2\rangle$$

where p is some index isomorphic to (j, κ, μ) that we call the **orbital index**,

$$p \simeq (j, \kappa, \mu)$$

The κ_{12} disappears because it is determined uniquely by the relation

$$\kappa_{12} = \begin{cases} \kappa_1 \dot{+} \kappa_2 & \text{if operator standard-coupled} \\ \kappa_1 \dot{-} \kappa_2 & \text{if operator is Pandya-coupled} \end{cases} \tag{6.2}$$

where the dotted plus sign ($\dot{+}$) denotes the Abelian operation used to combine the conserved quantum numbers and the dotted minus sign ($\dot{-}$) denotes its inverse. Usually, this is simply addition, but for certain multiplicative quantum numbers like parity this would translate to multiplication.

We thus have the bijection

$$(l_{12}, u_{12}) \simeq (j_{12}, p_1, p_2)$$

which allows us to relate two-particle states to one-particle states (*orbitals*) entirely independent of

the concrete quantum numbers. It is not necessary to relate the two-particle index pairs (l_{12}, u_{12}) to the concrete quantum numbers directly.

For convenience, we use a specific choice of orbital index, defined as

$$p(l, u) = M(l) + u$$

where $M(l)$ is an array of **channel offsets**, defined as

$$M(l) = \sum_{l'=0}^{l-1} m(l') \tag{6.3}$$

where $m(l')$ is the number of one-particle states in the one-particle channel l'.

For standard-coupled two-body operators, we impose an additional constraint to save memory:

$$p_1 \geq p_2$$

The antisymmetry of the states allows us to easily invert the order if this constraint is violated:

$$|j_{12} p_2 p_1\rangle = (-)^{j_1 + j_2 - j_{12}} |j_{12} p_1 p_2\rangle$$

In overall, we classify the bijections into two broad categories:

- Bijections that are dependent on the concrete quantum numbers κ and μ are stored in a generic[8] data structure that we call the **atlas**, which necessarily depend on the types of κ and μ.
- Bijections that are independent of the concrete quantum numbers, as well as **layout** information (dimensions and offsets, as in Eqns. 6.3, 6.4) are stored in a non-generic data

[8] Generic data structures are known as templated types in C++.

structure that we call the **scheme**.

The scheme is stored within the atlas for convenience, but most many-body methods do not require the atlas at all; they only need the scheme. The atlas is typically only needed during the input stage where matrix elements are read in, for conversion between J-scheme and pseudo-M-scheme, or for identification of basis states in debugging/display.

6.4.3 Access of matrix elements

Within each block, we partition the states into several contiguous parts, indexed by a part label χ.

- For the one-body operator, states are divided into hole ($\chi = 0$) and particle ($\chi = 1$) parts.
- For the standard-coupled two-body operator, states are divided into hole-hole ($\chi = 0$), hole-particle/particle-hole ($\chi = 1$), and particle-particle ($\chi = 2$) parts.
- For the Pandya-coupled two-body operator, states are divided into hole-hole ($\chi = (0,0)$), hole-particle ($\chi = (0,1)$), particle-hole ($\chi = (1,0)$), and particle-particle ($\chi = (1,1)$) parts.

This partitioning scheme makes it possible to avoid unnecessary iteration over states that do not contribute to a many-body diagram.

Implementing this requires subdividing the channels according to χ, thus we will need analogous quantities to those in Eq. 6.3, such as:

$$M(l,\chi) = \sum_{l'=0}^{l-1} \sum_{\chi'=0}^{\chi-1} m(l',\chi') \tag{6.4}$$

which we call the **part offset**. Here, $m(l',\chi')$ is the number of one-particle states in the one-particle channel l' within part χ'.

While the implicit antisymmetrization of standard-coupled two-particle states saves a significant amount of memory, they do complicate the translation of many-body equations into

code. To reduce this cognitive load, we introduce an **augmented** two-particle state index triplet (t_{12}, l_{12}, u_{12}) that includes an extra permutation parameter t_{12}. The permutation t_{12} can be either 0 or 1 if $p_1 \neq p_2$, or it can only be 0 if $p_1 = p_2$.

- If $t_{12} = 0$, then we interpret (t_{12}, l_{12}, u_{12}) as the usual state $|j_{12}; p_1 p_2\rangle$.
- However, if $t_{12} = 1$, then we interpret (t_{12}, l_{12}, u_{12}) as the permuted state $|j_{12}; p_2 p_1\rangle = (-)^{j_1+j_2-j_{12}}|j_{12}; p_1 p_2\rangle$.

The augmented state (t_{12}, l_{12}, u_{12}) has the advantage of being in a one-to-one correspondence to our intuitive notion of a two-particle state on paper. Thus it offers a useful abstraction that hides the internal complications of implicit antisymmetrization.

When a matrix element is accessed using an augmented state index, we must perform a phase adjustment depending on the value of t,

function get$(V, (t_{12}, l_{12}, u_{12}), (t_{34}, l_{34}, u_{34}))$

if $l_{12} \neq l_{34}$

0

else

$\phi_{12}\phi_{34}V^{l_{12}}_{u_{12}u_{34}}$

where

$$\phi_{ab} = \begin{cases} 0 & \text{if } t_{ab} = 0 \\ -(-1)^{j_a+j_b-j_{ab}} & \text{if } t_{ab} = 1 \end{cases}$$

When a matrix element is *set* using an augmented state index, we perform the same phase

adjustment to the value being set,

function set($V, (t_{12}, l_{12}, u_{12}), (t_{34}, l_{34}, u_{34}), x$)

assert($l_{12} = l_{34}$)

$V^{l_{12}}_{u_{12}u_{34}} \leftarrow \phi_{12}\phi_{34}x$

where the left arrow ($\leftarrow$) denotes array element assignment. If $l_{12} \neq l_{34}$, the operation aborts with an error. However, this setter is a rather leaky abstraction. Suppose we attempt to, say, increment every matrix element by one,

$V_{pqrs} \leftarrow V_{pqrs} + 1$

using the naive algorithm

for pq **in** all_augmented_states

for rs **in** all_augmented_states

set(V, pq, rs, get(V, pq, rs) + 1)

As it turns out, this will cause many of the matrix elements to be incremented *twice* instead of just once. This is because multiple augmented states map to the same unaugmented state. To remedy

this, we introduce a separate abstraction for the addition-assignment operation defined as

function add$(V, (t_{12}, l_{12}, u_{12}), (t_{34}, l_{34}, u_{34}), x)$

assert$(l_{12} = l_{34})$

$$V^{l_{12}}_{u_{12}u_{34}} \leftarrow V^{l_{12}}_{u_{12}u_{34}} + \frac{\phi_{12}\phi_{34}}{N_{12}N_{34}}x$$

where N_{ab} is the normalization factor in Eq. 3.27, which simplifies to $N_{ab} = 2 - \delta_{p_a p_b}$ if non-existent states are excluded. The denominator helps compensate for the overcounting.

6.4.4 Initialization of the basis

6.4.4.1 Input single-particle basis

To set up all the necessary basis structures for many-body theory, we require the following data, all of which are specific to each quantum system:

- We need a list of all single-particle states ("orbitals") in the quantum system.
- For every single-particle state, we need to know its part label χ, which tells us whether it is a hole (occupied) state or a particle (unoccupied) state relative to the Fermi vacuum Slater determinant.
- For every single-particle state, we need to know to its j, κ, and μ quantum numbers. We allow both κ and μ to be of practically any type and leave them as generic type parameters.
- We require κ-type values (1) to be cloneable, (2) to have a total equality relation, (3) to be hashable, and (4) to form an abelian group (i.e. to support the $\dotplus$ and $\dot{-}$ operations in Eq. 6.2). The first three conditions are needed to set up efficient mappings between arbitrary κ values and k indices using hash tables.[9] The last condition is needed to compute conserved

[9] We could have asked for a total ordering instead of hashability, in which case we would use ordered trees instead of hash tables. However, trees are slower and the inherent ordering of trees does not convey any advantages for our

quantum numbers for multi-particle states.

- We require μ-type values (1) to be cloneable, (2) to have a total equality relation, and (3) to be hashable. These conditions are needed to set up efficient mappings between arbitrary μ values and u indices.

Once we have this information, we can construct both the atlas and the scheme data structures for the system.

6.4.4.2 Channelized atlas initialization

The general process for setting up any channelized basis is straightforward. The input is a sequence of (λ, χ, μ) states, where

- λ is the channel,
- χ is the part, and
- μ is any auxiliary information needed to uniquely identify it within all states of the same λ.

The output are various charts, including $l \simeq \lambda$ and $(l, u) \simeq (l, \mu)$, and layout information (arrays of dimensions and offsets, such as $M(l)$ in Eq. 6.3). The process goes as follows:

1. Iterate over each state (λ, χ, μ) and incrementally build the single-particle chart for $l \simeq \lambda$. Store each (l, χ, μ) to a temporary array.
2. Sort the temporary array in lexicographical order, grouping the states by l and then by χ.
3. Iterate over the sorted array to derive the charts $p \simeq (l, u) \simeq (l, \mu)$ and layout information (see Eqns. 6.3, 6.4).

6.4.4.3 Many-body atlas initialization

The channelized atlas initialization procedure is then applied to one-particle, standard-coupled two-particle, and Pandya-coupled two-particle bases:

purposes.

1. To build the one-particle basis for the one-body operator, iterate over the input single-particle basis (Sec. 6.4.4.1) states (χ, j, κ, μ) and incrementally update the single-particle channel chart $k \simeq \kappa$. The states $(\lambda = (j, k), \chi, \mu)$ are then passed to the channelized atlas initialization procedure (Sec. 6.4.4.2) to obtain the necessary charts and layouts.
2. To build the two-particle basis for the standard-coupled two-body operator, iterate over the Cartesian product of the single-particle states with itself, yielding (l_1, u_1, l_2, u_2) per iteration:
   a. Use the single-particle chart to recover p_1, p_2, j_1, j_2, κ_1, κ_2, χ_1, and χ_2.
   b. Skip the iteration if $p_1 < p_2$.
   c. Compute $\kappa_{12} = \kappa_1 \dot{+} \kappa_2$ and $\chi_{12} = \chi_1 + \chi_2$ ($\chi_1, \chi_2 \in \{0, 1\}$ and $\chi_{12} \in \{0, 1, 2\}$).
   d. Update the two-particle channel chart $k_{12} \simeq \kappa_{12}$.
   e. For each j_{12} compatible with $\left\{ j_1 \quad j_2 \quad j_{12} \right\}$, push the state $(\lambda = (j_{12}, k_{12}), \chi = \chi_{12}, \mu = (p_1, p_2))$ into a temporary array.

   Finally, pass the temporary array of states to the channelized atlas initialization procedure (Sec. 6.4.4.2) to obtain the necessary charts and layouts.[10]
3. To build the two-particle basis for the Pandya-coupled two-body operator, iterate over the Cartesian product of the single-particle states with itself, yielding (l_1, u_1, l_4, u_4) per iteration:
   a. Use the single-particle chart to recover p_1, p_4, j_1, j_4, κ_1, κ_4, χ_1, and χ_4.
   b. Compute $\kappa_{14} = \kappa_1 \dot{-} \kappa_4$ and $\chi_{14} = \chi_1 + 2\chi_4$ ($\chi_1, \chi_4 \in \{0, 1\}$ and $\chi_{14} \in \{0, 1, 2, 3\}$).
   c. Update the two-particle channel chart $k_{14} \simeq \kappa_{14}$.
   d. For each j_{14} compatible with $\left\{ j_1 \quad j_4 \quad j_{14} \right\}$, push the state $(\lambda = (j_{14}, k_{14}), \chi = \chi_{14}, \mu = (p_1, p_4))$ into a temporary array.

   Finally, pass the temporary array of states to the channelized atlas initialization procedure

[10] The temporary array isn't technically needed, especially if the language has good support for coroutines/generators, allowing the data to be passed into the generic channelized atlas initialization procedure in parallel. We found it easier to describe the algorithm this way, and regardless the temporary array has a negligible impact on performance.

(Sec. 6.4.4.2) to obtain the necessary charts and layouts.

Note that step 2 and 3 differ in only two aspects: the abelian operation used for κ (addition vs subtraction), and the presence or absence of the implicit antisymmetrization constraint $p_1 \geq p_2$.

6.4.4.4 Basis initialization for quantum dots

The Fock–Darwin basis is used for quantum dot calculations.

In our implementation of the quantum dot system, j is not used, so we can simply set it to zero throughout. The set of conserved quantum numbers are $\kappa = (m_\ell, m_s)$, the projections of orbital angular momentum and spin. This leaves us with $\mu = n$, the principal quantum number.

The abelian group on $\kappa = (m_\ell, m_s)$ is defined in a straightforward manner:

$$\dot{0} = (0, 0)$$

$$(m_\ell, m_s) \dot{+} (m'_\ell, m'_s) = (m_\ell + m'_\ell, m_s + m'_s)$$

$$(m_\ell, m_s) \dot{-} (m'_\ell, m'_s) = (m_\ell - m'_\ell, m_s - m'_s)$$

The occupied states for quantum dots are always selected in complete shells – a shell consists of all states that share the same single-particle energy in Eq. 5.5. Moreover, systems we study always contain complete shells filled from the bottom. These are the N-particle electron configurations we use:

- $N = 2$: $0s^2$
- $N = 6$: $0s^2 0p^4$
- $N = 12$: $0s^2 0p^4 1s^2 0d^4$
- $N = 20$: $0s^2 0p^4 1s^2 0d^4 1p^4 0f^4$
- etc.

Here, the notation $n\ell^i$ means there are i particles in states with n and $|m_\ell| = \ell$ and s $\leftrightarrow$ 0, p $\leftrightarrow$ 1, d $\leftrightarrow$ 2, f $\leftrightarrow$ 3, as usual for spectroscopic notation.

6.4.4.5 Basis initialization for nuclei

The 3D harmonic oscillator basis is used for nuclei calculations.

In J-scheme nuclear calculations, the input j quantum number is simply the total angular momentum magnitude j. The set of conserved quantum numbers are $\kappa = (\pi, m_t)$, parity and isospin projection. This leaves us with $\mu = n$, the principal quantum number.

The abelian group on $\kappa = (\pi, m_t)$ is defined as:

$$\dot{0} = (+, 0)$$

$$(\pi, m_t) \dot{+} (\pi', m_t') = (\pi\pi', m_t + m_t')$$

$$(\pi, m_t) \dot{-} (\pi', m_t') = (\pi\pi', m_t - m_t')$$

In pseudo-M-scheme nuclear calculations, which we use mainly for testing purposes, the input j quantum number[11] is artificially set to zero. The set of conserved quantum numbers are $\kappa = (\pi, m_j, m_t)$, parity and the projections of total angular momentum and isospin. This leaves us with $\mu = (n, j)$, the principal quantum number and total angular momentum magnitude. Note that j cannot be put into κ because we are using uncoupled two-particle states.

[11] Not to be confused with the physical j quantum number, which is put in μ.

The abelian group on $\kappa = (\pi, m_j, m_t)$ is defined as:

$$\dot{0} = (+, 0, 0)$$

$$(\pi, m_j, m_t) \dot{+} (\pi', m_j', m_t') = (\pi\pi', m_j + m_j', m_t + m_t')$$

$$(\pi, m_j, m_t) \dot{-} (\pi', m_j', m_t') = (\pi\pi', m_j - m_j', m_t - m_t')$$

6.5 Input matrix elements

The procurement of input matrix elements varies wildly from system to system and there is not much code that can be shared among them outside of I/O utilities.

6.5.1 Inputs for quantum dots

The one-body matrix for quantum dots is diagonal and can be computed using Eq. 5.5.

Two-body matrix elements are more difficult to compute. We outsource the bulk of the work to the OpenFCI package [Kva08] and precompute the non-antisymmetrized matrix elements of Eq. 5.7 with the frequency-dependence factored out:

$$\frac{\langle (nm)_1 (nm)_2 | \hat{H}_2 | (nm)_3 (nm)_4 \rangle}{\sqrt{\hbar\omega E_\mathrm{h}}}$$

(Recall that m is a shorthand for m_ℓ for quantum dots.)

The elements are stored in a simple binary file format. In this format, the file is contiguous array of 16-byte entries, where the first 8 bytes of each entry contains the quantum numbers $n_1, m_1, n_2, m_2, n_3, m_3, n_4, m_4$ in that order. Each n is an 8-bit unsigned integer and each m is a 8-bit

signed integer. The remaining 8 bytes contain the value of the matrix element as a little-endian 64-bit IEEE 754 double-precision floating-point number. Schematically, we can depict an entry as the following structure:

```
struct Entry {
    n1:    u8,
    m1:    i8,
    n2:    u8,
    m2:    i8,
    n3:    u8,
    m3:    i8,
    n4:    u8,
    m4:    i8,
    value: f64,
}
```

To save space, not all matrix elements are stored. We restrict matrix elements to those that satisfy both of the following canonicalization conditions:

$$(p_1, p_2) \le \mathrm{sort}(p_3, p_4) \qquad\qquad (p_1, p_3) \le (p_2, p_4)$$

where

$$\text{sort}(p_3, p_4) = \begin{cases} (p_3, p_4) & \text{if } p_3 \le p_4 \\ (p_4, p_3) & \text{otherwise} \end{cases}$$

$$p_i = \frac{k_i(k_i + 2) + m_i}{2}$$

$$k_i = 2n_i + |m_i|$$

Note that comparisons such as $(a, b) \le (c, d)$ use lexicographical ordering.

6.5.2 Inputs for nuclei

The one-body matrix for nuclei is the kinetic energy operator, which is not diagonal but can still be easily computed using Eq. 5.12. Note that the equation does *not* include the $(1 - 1/A)$ factor that arises from the center-of-mass kinetic energy subtraction.

The two-body matrix elements consists of two parts. Firstly, there is a two-body component arising from the center-of-mass kinetic energy subtraction:

$$\langle pq| \left(-\frac{\hat{\boldsymbol{p}}_1 \cdot \hat{\boldsymbol{p}}_2}{mA} \right) |pq\rangle$$

This quantity can be computed easily in the center-of-mass frame. The Talmi–Brody–Moshinsky transformation brackets [Tal52; BM67; Mos59] can be used to convert the result back into lab frame.

The second part is the actual nuclear interaction. There are many possible choices here, and they are often available precomputed in a variety of tabular formats.

A common format that used for nuclear matrix elements is the **Darmstadt ME2J format**

[Bin14; Cal14; Lan14]. The chiral-EFT interactions, including [EM03], are often distributed in this format. In this format, all matrix elements are stored in a predefined order without explicitly writing out the quantum numbers. The iteration order is parametrized by $(e_{\max}, n_{\max}, \ell_{\max}, E_{\max})$ as defined in Eqns. 5.9, 5.10, 5.11. The first three parameters constrain the single-particle basis, which is constructed by the following algorithm:

for e **in** $0, \ldots, e_{\max}$
  for λ **in** $0, \ldots, \left\lfloor \frac{e}{2} \right\rfloor$
    let $\ell = 2\lambda + (e \bmod 2)$
    let $n = \frac{e - \ell}{2}$
    if $\ell > \ell_{\max}$
      break
    if $n > n_{\max}$
      continue
    for δ **in** $\left|\ell - \frac{1}{2}\right| - \frac{1}{2}, \ldots, \ell$
      let $j = \delta + \frac{1}{2}$
      yield (e, n, ℓ, j)

The algorithm establishes an *ordering* on the single-particle states, which we denote

$$(e_1, n_1, \ell_1, j_1), (e_2, n_2, \ell_2, j_2), \ldots, (e_{n_\mathrm{b}}, n_{n_\mathrm{b}}, \ell_{n_\mathrm{b}}, j_{n_\mathrm{b}})$$

where n_b is the number of single-particle states. Then, we must iterate over the two-body matrix

elements in the following order, constrained by $E_{\max}$:

for p **in** $1, \ldots, n_{\mathrm{b}}$

  for q **in** $1, \ldots, p$

    if $e_p + e_q > E_{\max}$

      break

    for r **in** $1, \ldots, p$

      for s **in** $1, \ldots,$ (**if** $r < p$ **then** r **else** q)

        if $e_r + e_s > E_{\max}$

          break

        if $(\ell_1 + \ell_2 + \ell_3 + \ell_4) \bmod 2 \neq 0$

          continue

        for J **in** $\max\{|j_p - j_q|, |j_r - j_s|\}, \ldots, \min\{j_p + j_q, j_r + j_s\}$

          for T **in** $0, 1$

            for M_T **in** $-T, \ldots, T$

              yield $(n_p, \ell_p, j_p, n_q, \ell_q, j_q, n_r, \ell_r, j_r, n_s, \ell_s, j_s, J, T, M_T)$

Note that in ME2J, the particle physics convention is used for isospin, so $m_t = -\frac{1}{2}$ is for neutrons and $m_t = +\frac{1}{2}$ is for protons.

6.6 Implementation of HF

The overall structure of our HF program is as follows:

1. Begin with the initial coefficient matrix $\boldsymbol{C}^{(0)}$ set to the identity matrix.
2. Now we loop over i from 1 and terminate at some high cut-off (e.g. 1024):
   a. Compute the Fock matrix $\boldsymbol{F}^{(\text{new})}$ using $\boldsymbol{C}^{(i-1)}$.
   b. Mix $\boldsymbol{F}^{(i-1)}$ and $\boldsymbol{F}^{(\text{new})}$ to obtain $\boldsymbol{F}^{(i)}$ using Eq. 6.6.
   c. Solve the Hartree–Fock equation as a Hermitian eigenvalue problem on $\boldsymbol{F}^{(i)}$. This results in a new set of coefficients $\boldsymbol{C}^{(i)}$ and a vector of eigenvalues (orbital energies) $\boldsymbol{\varepsilon}^{(i)}$. We use `heevr` from LAPACK for this, applied separately to every block of the matrix.
   d. Compute the sum of orbital energies $S^{(i)} = \sum_p j_p^2 \varepsilon_p^{(i)}$ as a diagnostic for convergence.
   e. Adjust the linear mixing factor using Eq. 6.7.
   f. Test how much $S^{(i)}$ has changed compared to $S^{(i-1)}$. If this is within the desired tolerance, break the loop.
3. Report whether the HF calculation has reached convergence (i.e. whether the loop was broken because the tolerance has met). Usually, the program will abort if this fails.
4. Transform the Hamiltonian using the final coefficient matrix.

6.6.1 Calculation of the Fock matrix

From $\boldsymbol{C}$, we compute an auxiliary matrix $\boldsymbol{Q}$ defined as

$$Q_{rs} = \sum_{i\backslash} C^*_{ri'} C_{si'}$$

This summation may be readily computed using GEMM.

Using $\boldsymbol{Q}$, we can reduce the cost of computing the Fock matrix, which is now described by

this equation in J-scheme:

$$F_{pq} = \langle p|\hat{H}_1|q\rangle + \sum_{j_{pr}{}^{rs}} \frac{j_{pr}^2}{j_p^2} Q_{rs} \langle pr|\hat{H}_2|qs\rangle \tag{6.5}$$

Compared with Eq. 4.7, which has a triply-nested sum, this equation only has a doubly-nested sum.

As a demonstration of our J-scheme framework, we include the code used to calculate the two-body contribution to the Fock matrix below.

```
pub fn fock2(
    h2: &OpJ200<f64>,
    q1: &OpJ100<f64>,
    f1: &mut OpJ100<f64>,
)
{
    let scheme = h2.scheme();
    for pr in scheme.states_20(&occ::ALL2) {
        let (p, r) = pr.split_to_10_10();
        for q in p.costates_10(&occ::ALL1) {
            for s in r.costates_10(&occ::ALL1) {
                for qs in q.combine_with_10(s, pr.j()) {
                    f1.add(p, q,
                           pr.jweight(2)
                           / p.jweight(2)
                           * q1.at(r, s)
```

```
                        * h2.at(pr, qs));
                }
            }
        }
    }
}
```

The inputs to this function are the three operator matrices `h2` ($\boldsymbol{H}_2$, the two-body Hamiltonian), `q1` ($\boldsymbol{Q}$), and `f1` ($\boldsymbol{F}$). The output is `f1` ($\boldsymbol{F}$ is mutated in-place).

The function begins by binding a reference of the scheme of `h2` to the `scheme` variable. This is done out of convenience. One could just as well have chosen `q1.scheme()`, or `f1.scheme()`, since all three operators are expected to have the same scheme as a pre-condition.

The outermost loop over iterates over all standard-coupled two-particle states $|j_{pr}; pr\rangle$. Keep in mind that although in storage we deduplicate states related by antisymmetry, the high-level interface here takes great pains to avoid exposing this internal detail.

The two-particle state $|j_{pr}; pr\rangle$ is then split into two single-particle states $|p\rangle$ and $|r\rangle$. Note that this is a many-to-one process: multiple two-particle state can split into the same pair of single-particle states.

The next loop iterates over all $|q\rangle$ that are also **co-states** of $|p\rangle$: these are all states that share the same channel as $|p\rangle$. These states have the same $l \simeq (j, \kappa)$ and thus reside within the same one-body matrix block, allowing us to avoid wasting time on matrix elements that are trivially zero. Another loop iterates over all $|s\rangle$ that are co-states of $|r\rangle$.

In the innermost loop, we combine $|q\rangle$, $|s\rangle$, and j_{pr} to form the state $|j_{pr}; qs\rangle$. This loop is somewhat unusual in that it iterates *at most once*. If for some reason the state $|j_{pr}; qs\rangle$ is forbidden, the innermost loop would do nothing. Otherwise, there can only be one state $|j_{pr}; qs\rangle$ that satisfies

the requirements.

Lastly, we add the appropriate contributions to `f1` using the usual formula. The `add` function and `at` (getter) automatically handle the antisymmetrization phases behind the scenes. Note that the `p.jweight(n)` function computes j_p^n.

This code is written in a rather naive way and utilizes the *high-level* interface of our J-scheme framework. It is certainly not the most efficient way of calculating the Fock matrix, but we consider its simplicity to be an advantage. Compared to other parts of the calculation, this sum is far from being the bottleneck, thus optimizing this code is not a high priority.

6.6.2 *Ad hoc* linear mixing

In some systems, the convergence of Hartree–Fock calculations can sometimes be very slow. This is usually the result of a highly oscillatory convergence. Several methods exist to mitigate this problem, including direct inversion of the iterative subspace (DIIS) [Pul80; Pul82] and Broyden's method [Bro65].

In our code, we implement a very simple *ad hoc* linear mixing strategy that, in practice, can aid convergence in many cases. The general idea is that if the sum of energies S is changing sign, then we attempt to dampen the oscillations by mixing in some portion of the old Fock matrix. If this converging is not oscillatory, then we try to keep most of the new Fock matrix.

The mixing is determined by a factor $c^{(i)}$ that is updated from iteration to iteration. The next Fock matrix to be used $\boldsymbol{F}^{(i)}$ is computed as:

$$\boldsymbol{F}^{(i)} = c^{(i)}\boldsymbol{F}^{(i-1)} + (1 - c^{(i)})\boldsymbol{F}^{(\text{new})} \tag{6.6}$$

where $\boldsymbol{F}^{(\text{new})}$ is the Fock matrix as computed via Eq. 6.5.

At each step, we update the mixing factor $c^{(i)}$ via the logic:

$$c^{(i)} = \begin{cases} \min\{\frac{1}{2}, bc^{(i-1)}\} & \text{if } (S^{(i)} - S^{(i-1)})(S^{(i-1)} - S^{(i-2)}) < 0 \\ \frac{c^{(i-1)}}{b} & \text{otherwise} \end{cases} \tag{6.7}$$

where $b > 1$ is some arbitrary constant that controls how rapid c should respond to the presence or absence of oscillations. We usually choose $b = 2$. The $\min\{\frac{1}{2}, \ldots\}$ prevents the calculation from stalling because too much of the old matrix is being retained.

6.6.3 HF transformation of the Hamiltonian

The HF transformation is describe by Eq. 4.3, which we reproduce here in J-scheme (not that there is any difference):

$$H'_{p'q'} = \sum_{pq} C^*_{pp'} H_{pq} C_{qq'}$$
$$H'_{p'q'r's'} = \sum_{pqrs} C^*_{pp'} C^*_{qq'} H_{pqrs} C_{rr'} C_{ss'}$$

The one-body term is very cheap and can be written in a naive way like the Fock matrix calculation.

The two-body term can be fairly expensive. Naive implementation of the equation would result in an 8-th power scaling, which is unbearably slow. Fortunately, the calculation can be broken down into two 6-th power steps at the cost of a temporary two-body matrix ,

$$\begin{aligned} T_{p'q'rs} &= \sum_{pq} C^*_{pp'} C^*_{qq'} H_{pqrs} \\ H'_{p'q'r's'} &= \sum_{rs} T_{p'q'rs} C_{rr'} C_{ss'} \end{aligned} \tag{6.8}$$

This could be broken down even further into four 5-th power steps, but we generally find this

an unnecessary complication – in particular, it would involve a temporary non-antisymmetrized two-body matrix, which would require introducing yet another matrix data type.

Eq. 6.8 can be programmed naively, which results in a slow but tolerable transformation. Alternatively, one could perform the transformation using GEMM. Our benchmarks indicate that the use of GEMM can provide a two-orders-of-magnitude improvement in speed. Unfortunately, it causes the internal details of implicit antisymmetrization to leak, complicating the phase factor.

In any case, the technique is as follows. Define the following two-body antisymmetrized coefficient matrix G:

$$G_{rsr's'} = N_{rs} S_{rs}^{(1+j_r+j_s-j_{rs})} S_{r's'}^{(1+j_{r'}+j_{s'}-j_{rs})} C_{rr'} C_{ss'}$$

where N_{rs} is the normalization factor in Eq. 3.27 and $S^{(i)}$ is the $(-)^i$-symmetrization symbol of Sec. 2.1.2. Then we can compute the transformation using GEMM:

$$T_{p'q'rs} = \sum_{p \geq q} G^*_{pqp'q'} H_{pqrs}$$

$$H'_{p'q'r's'} = \sum_{r \geq s} T_{p'q'rs} G_{rsr's'}$$

6.7 Implementation of normal ordering

Before performing IM-SRG(2), it is necessary to obtain matrix elements of $\hat{H}$ relative to the Fermi vacuum. This step is often referred to as the *normal ordering* of $\hat{H}$ (the terminology is somewhat overloaded). As part of this step, we also obtain the Hartree–Fock energy.

In the case of two-body operators, there are only two interesting operations here. One is the calculation of the zero-body component (HF energy) using Eq. 4.2 or Eq. 2.4, which we reproduce

here in J-scheme:

$$E_\Phi = E_\varnothing + \sum_{i\backslash} \hat{j}_i^2 H_{ii}^\varnothing + \frac{1}{2} \sum_{j_{ij}} \sum_{ij\backslash} \hat{j}_{ij}^2 H_{ijij}^\varnothing$$

We have omitted the primes because at this point normal ordering itself can be applied independent of HF. This calculation can be done naively as it is very cheap.

The other part is the folding of the two-body component into the one-body component as shown in Eq. 2.4 and reproduced here in J-scheme:

$$H_{pq}^\Phi = H_{pq}^\varnothing + \sum_{j_{pi}} \sum_{i\backslash} \frac{\hat{j}_{pi}^2}{\hat{j}_p^2} H_{piqi}^\varnothing$$

This calculation can also be done naively as it is still very cheap.

6.8 IM-SRG(2) implementation

The overall structure of the IM-SRG implementation is centered around an ODE loop with tests for convergence, much like HF. The inputs to IM-SRG are the normal-ordered, zero-, one-, and two-body operator matrices in their standard-coupled forms.

1. Pack all three standard-coupled Hamiltonian components into a single array $\boldsymbol{y}$ for the ODE solver to consume. In doing so, deduplicate elements that are related by hermitivity.
2. Initialize and maintain a cache of 6-j symbols, needed for the Pandya transformations.
3. Initialize the Shampine–Gordon ODE solver.
4. Now enter the main IM-SRG loop, starting with the flow parameter s set to zero. If the loop exceeds some predefined limit on s, abort.
    a. Request the solver to proceed to $s + \Delta s$, for some Δs specified by the user. Provide

the derivative function $\dot{\boldsymbol{y}} = \boldsymbol{f}(s, \boldsymbol{y})$ to the solver. While the solver is stepping, it will attempt to evaluate our function $\boldsymbol{f}$:

- Unpack $\boldsymbol{y}$ back into the standard-coupled operator matrices $\boldsymbol{H}$.
- Compute the generator $\boldsymbol{\eta}$ from $\boldsymbol{H}$.
- Compute the comutator $\boldsymbol{D} = [\boldsymbol{\eta}, \boldsymbol{H}]$, making use of the 6-j cache.
- Pack the comutator $\boldsymbol{D}$ into the derivative array $\dot{\boldsymbol{y}}$.

b. If the stepping failed, abort.

c. Get the ground state energy E_Φ, which in our packing convention is simply the first element of $\boldsymbol{y}$.

d. Check how much the ground state energy has changed since the previous iteration. If it is less the user-specified tolerance, break the loop with success.

We use the White generator for our IM-SRG calculations, as described in Eq. 4.15. Our implementation supports both Møller–Plesset and Epstein–Nesbet denominators, using monopole matrix elements in the latter case.

6.8.1 Calculation of the IM-SRG(2) commutator

The bulk of the implementation complexity and computational cost lies in the calculation of the commutator $\boldsymbol{D} = [\boldsymbol{\eta}, \boldsymbol{H}]$. At the moment, we implement this as by calculating the linked products of $\hat{\eta}\hat{H}$ and subtracting the linked products of $\hat{H}\hat{\eta}$. This allows us to reuse the linked product code. However, in the future, we may disrupt this symmetry for optimization reasons – to take advantage of $\hat{\eta}$'s higher sparsity as compared to $\hat{H}$.

Since the linked product corresponds to the $\hat{C}$ operator in Secs. 4.2.4, 4.2.5, we will discuss the various terms using the naming convention in that chapter. We will not attempt to discuss all of the terms but instead focus on a few interesting ones.

6.8.1.1 Optimization of terms 2220 and 2222

These are one of the most costly terms in the commutator, but also one the easiest to optimize:

$$C^{2220}_{pqrs} = \frac{1}{2} \sum_{ij\backslash} A_{ijrs} B_{pqij}$$

$$C^{2222}_{pqrs} = \frac{1}{2} \sum_{\backslash ab} A_{pqab} B_{abrs}$$

Calculating these terms using GEMM is quite straightforward and offers orders of magnitude in improvement.

However, a subtlety arises due to the implicit antisymmetrization, which we also encountered earlier in the HF transformation optimization: we must not double-count the $i = j$ (or $a = b$) states. With this taken into account, the equations become

$$C^{2220}_{pqrs} = \frac{1}{2} \sum_{i \geq j\backslash} N_{ij} A_{ijrs} B_{pqij}$$

$$C^{2222}_{pqrs} = \frac{1}{2} \sum_{\backslash a \geq b} N_{ab} A_{pqab} B_{abrs}$$

where N_{ab} denotes the normalization factor in Eq. 3.27. This is an unfortunate consequence of using unnormalized matrix elements; had we used normalized ones, the spurious N_{ab} factor would not appear.

Note that the above two equations are extremely similar and can be implemented as just one function. The difference between the two is the that A and B have been swapped, and that the parts of states being summed over are different.

6.8.1.2 Optimization of terms 2221

The 2221 term is one of the most interesting and costly terms. It is actually described by a three-step process. First, use the Pandya transformation to convert the standard-coupled operators into Pandya coupling (Sec. 3.12.2):

$$\tilde{A}_{psrq} = -\sum_{j_{pq}} (-)^{2j_{pq}} j_{pq}^2 \begin{Bmatrix} j_p & j_q & j_{pq} \\ j_r & j_s & j_{ps} \end{Bmatrix} A_{pqrs}$$

Then, use GEMM to compute the product using Pandya-coupled matrices:

$$\tilde{C}^{2221}_{psrq} = +4 \sum_{i\backslash a} \tilde{A}_{iarq} \tilde{B}_{psia}$$

Finally, convert back into standard coupling using the antisymmetrizing inverse Pandya transformation.

The GEMM part is probably the most straightforward. Unlike 2220 or 2221, there are no unusual phase factors because we do not use implicit antisymmetrization here.

Our benchmarks show that a very significant amount of time is spent on the Pandya transformation, thus it is worthwhile to optimize that operation despite being a roughly 5-th power operation.

One technique is to rewrite the Pandya transformation itself as a GEMM-compatible product:

$$\tilde{A}^{j_p j_q j_r j_s}_{j_{ps};\alpha_p\alpha_q\alpha_r\alpha_s} = -\sum_{j_{pq}} W^{j_p j_q j_r j_s}_{j_{ps};j_{pq}} A^{j_p j_q j_r j_s}_{j_{pq};\alpha_p\alpha_q\alpha_r\alpha_s}$$

where

$$W^{j_p j_q j_r j_s}_{j_{ps} j_{pq}} = (-)^{2j_{pq}} j_{pq}^2 \begin{Bmatrix} j_p & j_q & j_{pq} \\ j_r & j_s & j_{ps} \end{Bmatrix}$$

In this "four-j" matrix layout, each diagonal block of A or $\tilde{A}$ is indexed not by the usual channels, but by (j_p, j_q, j_r, j_s). The right axis of both matrices is labeled by $(\alpha_p, \alpha_q, \alpha_r, \alpha_s)$, where α_p denotes all the non-j quantum numbers of p. The use of this layout, in conjunction with a GEMM-powered Pandya transformation, saves about 40% time.[12] The gains are not as dramatic as in the case of 2220 or 2222 because the matrix-matrix multiplication is only over j_{ps} whose dimensions are usually not very big.

The dominant work is now pushed onto the conversion from the standard- or Pandya-coupled matrices into this four-j layout. This can be expensive due to the large number of hash table lookups for translation between two-particle states and pairs of single-particle states. To mitigate this we cache the translated indices within a separate four-j-layout matrix, bypassing the need for expensive hash table lookups. This saves another 50% time at the cost of extra memory usage.[13]

6.9 QDPT3 implementation

The QDPT3 implementation is fairly tedious as it involves coding about 20 or so terms from Sec. 4.3.1. It is also quite error-prone, much like the IM-SRG commutator, therefore extensive testing is necessary.

Fortunately, all terms up to third order are fairly inexpensive, therefore writing them out naively using the high-level J-scheme interface would suffice as the cost of IM-SRG dwarfs the

[12] Benchmarked using $e_{\max} = 4$ for an ^{16}O-like system.
[13] Benchmarked using $e_{\max} = 4$ for an ^{16}O-like system.

cost of QDPT.

Additionally, a large number of QDPT terms share similar topologies: there is one unique topology at second order (type A), and only three unique topologies at third order (types B, C, and D). Thus, they can reuse with the same code simply by tweaking a the state parts that are being summed over and customizing the denominator. As an example, consider the type B QDPT term:

$$W^{(\mathrm{B})}_{pq}(\chi_r, \chi_{st}, \chi_{uv}, D) = \frac{1}{4} \sum_{r \in \chi_r} \sum_{st \in \chi_{st}} \sum_{uv \in \chi_{uv}} \frac{\breve{J}^2_{rp}}{\breve{J}^2_p} \frac{H_{rpst} H_{stuv} H_{uvrq}}{D(r,s,t,u,v)}$$

All of the first six terms/diagrams at third-order, shown in the equations of Sec. 4.3.1, have this type B topology. The arguments χ_r, χ_{st}, and χ_{uv} indicate which parts of the states should be summed (i.e. hole or particle) for $|r\rangle$, $|st\rangle$, and $|uv\rangle$ respectively.

The code to compute type B terms is shown below:

```
pub fn qdpt_term_b<F>(
    h1: &OpJ100<f64>,
    h2: &OpJ200<f64>,
    p: StateJ10,
    q: StateJ10,

    // these are denoted "chi" in the equations
    r_occ: Occ,
    st_occ: [Occ; 2],
    uv_occ: [Occ; 2],

    // the denominator (closure / function object)
```

```
    denom: F,

) -> f64 where
    F: Fn(StateJ10, StateJ10, StateJ10,
          StateJ10, StateJ10) -> f64,
{
    // make sure p and q are within the same block
    // (i.e. have the same conserved quantum numbers)
    assert_eq!(p.lu().l, q.lu().l);

    let scheme = h1.scheme();
    let mut result = 0.0;
    for r in scheme.states_10(&[r_occ]) {
        for jrp in Half::tri_range(r.j(), p.j()) {
            // combining states can fail,
            // in which case we just continue
            let rp = match r.combine_with_10(p, jrp) {
                None => continue,
                Some(x) => x,
            };
            let rq = match r.combine_with_10(q, jrp) {
                None => continue,
                Some(x) => x,
            };
```

```
            for st in rp.costates_20(&[st_occ]) {
                let (s, t) = st.split_to_10_10();
                for uv in rp.costates_20(&[uv_occ]) {
                    let (u, v) = uv.split_to_10_10();
                    result += 1.0 / 4.0
                        * rp.jweight(2)
                        / p.jweight(2)
                        * h2.at(rp, st)
                        * h2.at(st, uv)
                        * h2.at(uv, rq)
                        / denom(r, s, t, u, v);
                }
            }
        }
    }
    // in Rust syntax, the last expression of a function is
    // implicitly returned if not terminated by semicolon
    result
}
```

Observe that we allow the denominator argument `denom` to be a closure of any type `F`, allowing it to be easily inlined by the compiler for efficiency. This permits the use of anonymously-typed closures, each with a distinct type, making it trivial for the compiler to tell different closures apart. As long as the type of the closure is not erased, the compiler is guaranteed to create distinct copies of the `qdpt_term_b` function for each closure type `F`, greatly improving optimizability. This is an

inherent feature of monomorphization in both Rust and C++.

Here is a snippet of code that demonstrates the use of `qdpt_term_b` to compute the first third-order term shown in the equations of Sec. 4.3.1:

```
qdpt_term_b(
    h1, h2, p, q,
    // notation: I = hole state, A = particle state
    occ::I, occ::AA, occ::AA,
    // we define the denominator using a lambda function;
    |i, a, b, c, d| {
        // the "hd" function extracts the diagonal part
        // of the one-body Hamiltonian
        (hd(i) + hd(q) - hd(a) - hd(b))
            * (hd(i) + hd(q) - hd(c) - hd(d))
    },
)
```

6.10 Testing and benchmarking

The first line of defense for ensuring correct code is through properly designed abstractions and data types. By categorizing values into distinct types one can avoid accidental confusion of quantities.

For example, we have introduced a special data type called `Half` to represent half-integers. Since no native machine type exists for half-integers, the usual workaround is to represent half-integers with twice its effective value. For example, the half-integer $m = 3/2$ would be represented

as `m = 3` on a computer.

This leaves room for human error, if one say, adds a half-integer $m = 3/2$ to a normal integer $n = 1$. The result would be `3 + 1 = 4`, which is incorrect. The `Half` data type, defined below, prevents this problem,

```
pub struct Half(pub i32);
```

It is not possible to add `Half` to an `i32`, because no such operator is defined. Thus the human error becomes a compile error, preventing the incorrect program from compiling.

As another example, we have distinct types for standard-coupled and Pandya-coupled two-body matrices, which prevents us from accidentally using a Pandya-coupled state to look up a standard-coupled matrix.

Types cannot catch all bugs, but with judicious use they certainly catch a lot of the obvious ones. We use tests to catch bugs that cannot be detected at compile time, either because the solution would be too complex, too awkward to use, or downright impossible. Not only do tests ensure that the code is correct *right now*, they also guard against future mistakes as the code evolves. Several kinds of testing strategies are used in Lutario.

The most basic ones are **unit tests**, which are short tests intended to verify basic functionality. These are often suitable for small functions that require little to no setup. For example, we have the following test for our implementation of Euclidean division and modulo:

```
#[test]
fn test_euclid_div_mod() {
    assert_eq!(euclid_div(10, 5), 10 / 5);
    assert_eq!(euclid_mod(10, 5), 10 % 5);
    assert_eq!(euclid_div(-10, 5), -10 / 5);
```

```
    assert_eq!(euclid_mod(-10, 5), -10 % 5);
    assert_eq!(euclid_div(10, -5), 10 / -5);
    assert_eq!(euclid_mod(10, -5), 10 % -5);
    /* etc ... */
}
```

The function attribute `#[test]` informs the Rust compiler that this function is part of the test suite. The test explores all the potential sign errors that could happen in an implementation of Euclidean division and modulo.

It is generally impossible to check programs over all possible inputs as the input space is usually far too large. One way to ensure that the *interesting* cases are tested is through **code coverage** tools. These tools can track which lines of the source code are executed during the tests. If there are certain lines that are never executed because an if-condition is never true during the tests, then those lines effectively untested. Full code coverage is not a guarantee that the test is exhaustive, however.

When the input space is too large to explore, one could also consider **randomized testing**, in which inputs y are generated randomly and then the test is responsible for verifying the results $y = f(x)$ in some way. One could either (1) compare the results using another function f' that reproduces the result, or (2) check whether certain properties $P(x, y)$ hold (**property testing**). We offer a more concrete example in Sec. 6.10.1.

While it is generally difficult to prove that testing has exhaustively covered all cases, it is nevertheless better to have at lesat one test case than none. These are often called *smoke tests* and they are still remarkably useful in practice.

Larger tests, where multiple components of the program are tested together, are known as **integration tests**. These are used to test the many-body methods, as they are fairly complicated

and require several components working together to achieve a result. We have numerical results for, e.g. ground state energy, from other implementations of the same many-body methods that we can test against.

From time to time, bugs will inevitably slip past the existing tests. Whenever such a bug is discovered, it is important to add additional tests to ensure the bug will not go undetected again in the future – these are known as **regression tests**.

Tests are only useful if they are being run. Unfortunately, tests may require a substantial amount of time to run, which discourages the programmer from running such tests. Frequent runs of tests are important: they ensure that code remains valid at all times, and they allow problems to be discovered at the earliest opportunity.

In Lutario, we have configured a **continuous integration** (CI) system that automatically runs tests on every commit pushed to the repository and notifies failures by email. It ensures that problems are always discovered quickly. Furthermore, it allows us to keep the build process streamlined and reproducible as otherwise the automated testing script, which runs in a clean environment, would fail. It helps prevent environmental problems where the code functions correctly only on the developer's machines, but not on the users'.

6.10.1 Randomized testing of numerical code

The sheer number and tedious nature of the IM-SRG commutator terms and QDPT terms offers a ripe environment for human errors. To mitigate against this, we use tests to verify that the commutators are performing the calculations we expect.

There is a chicken-or-egg problem regarding numerical computations: we generally do not know the answers *a priori* without running a program to calculate it – manual calculations are usually impractical – yet we do not know if the program is calculating the formula we intended.

To break this loop, we have to trust at least one program to compute the result – to "bootstrap" our test suite.

We assign the most trust to the simplest and most naively implementation of the program. Even if it is very slow, it is often sufficient to test just a few small nontrivial cases. This would serve as our *reference* program.

With a reference program in hand, we need some input (matrix elements) to test against. We could use actual matrix elements from physical systems, but it suffices to use a randomly generated set of matrix elements, provided that these matrices satisfy the necessary symmetries and conservation laws for the formula to remain valid. This has the added advantage that as long as the random number generator is deterministic, we only need to store the seed to recover the entire suite of test matrices.

In the predecessor to Lutario, we have generated a set of random test matrices in the quantum dot basis for testing the commutator. They were verified by comparing against an extremely naive implementation that does not take advantage of the sparsity of the matrix. These input matrices, along with the expected output, have now been inherited by Lutario's test suite. They ensure that the new J-scheme implementation remains correct for $j = 0$ (i.e. pseudo-M-scheme).

To test cases where $j \neq 0$ (proper J-scheme), we construct random matrix elements in the nuclei basis in J-scheme and compute the commutator in two different ways:

- We perform the commutator in J-scheme (operation C_{J}), and then convert the resulting matrices to pseudo-M-scheme (operation φ).
- We convert the matrices to pseudo-M-scheme (operation φ), and then perform the commutator in pseudo-M-scheme (operation C_{M}).

We expect the results to be identical in both cases. This is mathematically described by the

commutative square:

$$\varphi \circ C_{\mathrm{J}} = C_{\mathrm{M}} \circ \varphi$$

This is a general approach of testing J-scheme equations that does not require us to know what the correct answers are, as long as the pseudo-M-scheme code is correct.

6.10.2 Linting, static analysis, and dynamic sanitization

Several kinds of automated tools exist to aid the detection of bugs.

Static analyzers read the source code of a program and attempt to look for bugs without actually running it. Due to the halting problem, it is impossible for a static analyzer to avoid false negatives in any Turing-complete language.

Linting tools are a subtype of static analyzers designed to find not only bugs, but also suspicious constructs, non-idiomatic code, and, in some cases, code that does not conform to a particular stylistic convention.

Modern compilers are also capable of giving useful warnings for potentially buggy code. C and C++ compilers by default are very conservative about warnings, but it is possible to request more comprehensive diagnostics using a flag similar to `-Wall`. This alone can catch many common mistakes.

In contrast, the Rust compiler by default issues all warnings. Our code is always written to avoid such warnings, even if the warning is of low priority or not justified. This ensures that if an important warning appears later on, it is not drown out by the deluge of low-priority warnings that had been intentionally ignored. If a warning is a false positive, we can either find a workaround or, failing that, silence the warning at that particular location using an attribute such

as `#[allow(...)]` in Rust.

Dynamic sanitizers are designed to detect errors during the execution of a program. Dynamic sanitizers are naturally better at detecting problems, but aside from the need to actually execute the program, they also introduce some performance overhead. Examples of such tools include Valgrind [Valgrind], as well as various Clang and GCC sanitizer flags (`-fsanitize=...`) .

Sanitizers can be extremely helpful at finding the cause of bugs when there is suspicion of memory errors in a given program. In our experience, Valgrind has been particularly effective at detecting memory errors in our C and C++ projects. The tools can even be used pre-emptively, run routinely as part of the test suite, to reduce the risk of memory errors being introduced as the codebase evolves.

6.10.3 Benchmarks and profiling

For benchmarking, we make use of Rust's official (but unstable) `test` library, which offers a very simple interface:

```
#![feature(test)]
extern crate test; // import the test library


#[bench]
fn my_bench(b: &mut Bencher) {
    b.iter(|| {
        // code goes here
    });
}
```

The code within the closure `|| { ... }` is executed several times by the benchmark runner

to obtain an accurate timing along with an estimated uncertainty. This means the code being benchmarked must avoid irreversibly changing the environment, or else the timing will not be accurate.

It is important to write the benchmarked code in a way such that the compiler cannot delete the code entirely. Consider this example, where one is attempting to benchmark the addition of two integers:

```
b.iter(|| {
    let x = 1 + 2;
});
```

There are several reasons why this naive benchmark would not yield any meaningful results:

- The compiler can easily precompute the result "3" without any effort. This means the input must *not* be predictable to the compiler. To mitigate this, one could either read the input from a file, randomize the input, or use a special `black_box` function to obscure the input from the optimizer, e.g. `black_box(1) + 2`.
- Even if the compiler does not know the inputs, it does know that addition is a pure operation with no side-effects. Thus it would safe to hoist the statement outside the loop (`b.iter(...)` is really a loop in disguise). One could insert `black_box(x)` within the loop to prevent this.
- Moreover, the compiler can easily see that the output variable `x` is not being used. This means the compiler will likely delete the entire calculation through dead-code elimination (DCE). To mitigate this, one could either print the output, or again use a `black_box`.
- Lastly, addition of integers is such a fast operation that the overhead from the benchmark runner as well as environmental noise will heavily skew the results.

We use the nuclei system for benchmarks, but with randomized matrix elements. We split the commutator into groups of terms that are benchmarked separately so that we can analyze the

individual terms separately without the expensive ones drowning out the cheap ones.

To analyze the performance costs of given implementation, we make use of profilers. We generally avoid profilers that instrument code, because the instrumentation itself can easily add a substantial amount of overhead that can completely distort the results. For this reason, we use Perf [Perf], a sampling profiler for Linux that captures stack traces of the program periodically. Similar tools exist other platforms. Perf pairs quite well with flame graphs [Gre16] for visualization, allowing easy identification of hotspots in the code.

Perf can run an optimized program as-is, with no instrumentation. The only extra information needed for sensible output is debugging information (`-g` flag in most compilers), which is tracked separately and does not pessimize the program. Unfortunately, high levels of optimization usually renders the debugging information inaccurate, so it remains important to compare against the assembly code.

6.11 Version control and reproducibility

The codebase for Lutario is stored in a distributed version control system (DVCS) and can be viewed online [Lutario]. We use the Git [Git] as our DVCS but one could equally well have chosen other DVCSes such as Mercurial [Hg].

Version control systems (VCS) are tools designed to store the entire history of a codebase. At a rudimentary level, it may be considered a special kind of database tailored specifically for projects that are dominated by plain text files like code. By recording all changes that occur in a project, it becomes easy to track down the origin of both code and bugs.

VCSes are also designed to support collaboration on projects. Collaborators may work independently on different parts of the project, accumulating their own history of changes. They can periodically synchronize and merge their changes together to form a unified timeline. The merge

can be automated as long as the collaborators work on different parts of the project.

Distributed VCSes are unique in that, unlike centralized VCSes, each repository – i.e. the directory managed by the VCS – maintains its own database of histories and is on equal footing as all other repositories. This means there is no centralized point of failure if any one of the repositories becomes inaccessible, allowing it act as a distributed backup system. The histories need not match either: a repository might store the main history of the project, but may also keep a private fork of this history, or of a new feature, or something entirely unrelated.

VCSes are most useful when they store predominantly relevant, textual data. It is particularly important to avoid storing large files as they can rapidly grow the size of the database. It is also important to avoid storing files that could be easily regenerated, such as executable files, library files, or any non-essential or transient files. Ignore lists are useful for filtering out irrelevant files.

Changes in a VCS are stored in the unit of a *commit*, which should generally perform a single task or add a single feature. Maintaining a clean commit history ensures that if problems appear later on, one can easily isolate the cause down to a single commit, with the aid of strategies such as bisection (e.g. `git-bisect`).

The use of a version control aids in the reproducibility of results. One can accurately refer to a specific version of code within a version-controlled repository using commit identifiers or human-readable tags. In particular, Git and Hg use hashes as the commit identifier, thus knowing the hash one can verify whether the files of the commit are correct with a high degree of confidence, as it is very difficult to forge these hashes.[14]

Of course, knowing that one has the correct code alone does not guarantee that the results will be reproduced perfectly. One may also need to track the version numbers of all transitive

[14]Unfortunately, SHA-1 attacks are becoming more and more possible, which means it may be possible for an attacker to forge commit hashes. Nonetheless, by accident it remains extremely difficult to have two different commits sharing identical commit hashes.

dependencies of the program, the compiler, as well as the environmental conditions under which the program is run. Nondeterminism caused by environmental fluctuations (e.g. in parallel code) can further complicate reproducibility.

6.12 Documentation

There are two orthogonal ways to categorize documentation. On one axis, there is

- external documentation (for users), and
- internal documentation (for developers of the project).

On another axis:

- There is reference documentation (manuals, technical documents, specifications, API[15] documentation), which aims to describe every detail of the program including corner cases. It is usually written to follow the structure of the program (modules, functions, data types). Their target audience are the advanced users who already know their way around the software.
- There is also review documentation (tutorials, guides, overviews), which are usually pedagogical in nature. They are used to teach the important parts of a program, without overwhelming the reader with details. They generally do not follow the structure of the program, but are structured more like a book intended for human consumption. The target audience are the new users who are not yet familiar with the software.

At the moment, external documentation for Lutario is very sparse, given the recency of the project. As the project is still under heavily development, we expect the user-facing interfaces to change substantially. We will consider adding external documentation when a point of stability is reached.

[15] API is a common programming acronym for *application programming interface*, or simply *interface*.

Internal documentation is primarily through this chapter – which may become out of date as the project progresses – as well as the source code comments. This chapter is intended to be an overview of the machinery in Lutario without focus on any one particular aspect. Source code comments may be categorized into two types:

- Documentation comments are designated by the `///` or `//!` prefix in Rust.[16] They are useful for documenting public interfaces (APIs). These special comments can be automatically exported by the Rustdoc documentation generator. The tool outputs a book in HTML format with the comments displayed against the corresponding module, function, data type, etc. Similar tools exist for other languages, including Sphinx [Sphinx] for Python, Haddock [Haddock] for Haskell, and Doxygen [Doxygen] for C, C++, and Fortran.

  They can also be used to provide examples. These snippets of code are automatically tested by the Rust build system, ensuring that the example code does not fall out of date as the code evolves.
- Normal comments (`//` or `/* ... */` in Rust) are used to explain tricky aspects of the code for the developers. They are generally used sparingly, because such comments are meant to convey *important* information that is not evident from the code itself. Excessive use of comments can hinder the readability of code. Moreover, comments – which are not sanity-checked by the compiler – are always at risk of becoming out of sync with the actual code.

Tests themselves can also serve as useful examples, if they are written cleanly. Such dual-purpose tests are extremely useful: they are automatically tested for validity, moreover a learning user can immediately read from the test code what the expected output of the program should be. We currently have three major integration

[16] In other languages, one might find `/**` or `/*!` prefixes for documentation comments.

6.13 Coding style

While source code is to be ultimately consumed by compilers and interpreters, it is equally important for it to be comprehensible to human readers. This reduces mistakes, encourages collaboration, and simplifies maintenance of the program over the long run.

6.13.1 Formatting of code

On a superficial level, code should be formatted neatly and, most importantly, *consistently*. One should adhere to the official style guide of the language (if any), or any prevailing style used by the language community, domain, project, and/or subproject. These style conventions help establish many aspects of formatting, including indentation, spacing, line length, wrapping, and naming conventions.

Some languages such as Fortran, C, C++, or Haskell lack official style guides, but there may exist one or more *de facto* styles from which one can adopt. Others, like Go, Rust, Python have official style guides [Gofmt; FmtRFCs; PEP8] with varying degrees of strictness, which improves collaboration and avoids unnecessary *bikeshedding* (trivial arguments) among the language community. In either case, various automatic reformatting tools are available to help maintain uniformity in coding style, such as `clang-format` for C and C++ [ClFmt], `gofmt` for Go [Gofmt], and `rustfmt` for Rust [Rustfmt].

6.13.2 Coupling and complexity

A major source of complexity in programs arises from *coupling*: an interaction between different components of a system. It is analogous to coupling in physics. A system of non-interacting particles is generally easy to study. As the interactions become stronger and stronger, the system

becomes increasingly difficult to understand.

The same principle applies to programming. Coupling should generally be avoided where possible. But that is unlikely in any non-trivial program. When it is not avoidable, coupling should always be explicitly visible to the reader. This helps avoid "effect at a distance" where, say, a programmer attempts to fix a bug, only to break something else entirely unrelated.

Many kinds of coupling exist. The most common one is **aliasing**: when different parts of a program have a shared reference or pointer to the same variable, and at least one them modifies it.

As discussed in Sec. 6.1.2, if one has exclusive control over a piece of data, they can modify it without other components of the program observing the effects of the modification. If one has shared a piece of data with other components of the program and the data is never modified by anyone, then no-one will know the difference either. However, if the data is mutated, then problems can arise because the value of the variable may unexpectedly change. Thus, aliasing makes it difficult to reason about code.

Without thread synchronization, aliasing generally breaks thread-safety. This is because modern CPUs like to cache data as much as possible. Without some sort of notification to the CPU that data has been modified by another thread, it will by default assume that it has exclusive control over it and continue using the cached value. The only way to ensure this is safe is through synchronization (e.g. memory fences, mutexes), which carries a performance penalty.

Even without threads, there are other performance penalties that arise from aliasing. The compiler can make fewer assumptions about the behavior of an aliased variable, therefore code that involves aliasing may be less well optimized.

Aliasing is not always avoidable. It is generally a good idea to document where aliasing occurs. Rust, in particular, takes a fairly draconian stance with regard to aliasing: all aliasing is forbidden except through one of the alias-enabling wrapper types (e.g. `Cell`, `RefCell`, `Mutex`, `RWLock`).

C and C++ have a set of rules (type-based alias analysis / strict aliasing rule) that determine when aliasing is acceptable and when it is not (in which case aliasing becomes undefined behavior). In C, one may assert the *absence* of aliasing through the `restrict` keyword, but the compiler make no effort to verify that assertion.

Global variables are a special case of aliasing, except more sinister because they are much less obvious, since whether a function accesses a global variable cannot be deduced from the function signature. The only way to tell whether a function uses a global variable is by inspecting the contents of the function and all its transitive dependencies.

Aliasing may be considered a type of **side effect**. A function is said to have side effects if it modifies some kind of state that is externally visible. Aliased variables are examples of such state, but so are things like input/output (I/O), spawning/killing a process, sending/receiving data over network, etc.

Side effects are a form of coupling as well, except the coupling may involve the external environment: other processes, other machines, or even other people. Like any coupling, it is always good idea to document side effects to aid reasoning of programs.

To help manage side effects, one should keep them contained and isolated. In particular, code that has lots of side effects should be kept separate from code with no side effects (**pure** code). Complicated logic are best written without side effects, allowing it to be easily refactored without concern of temporal ordering. Furthermore, pure code is generally more reusable and composable, whereas code with side-effects are usually less flexible. Some languages such as Haskell or PureScript explicitly track side-effects through the type system, encouraging the programmer to manage side-effects in a principled manner.

As an example, in our Lutario codebase, we generally refrain from performing any sort of I/O except in designated functions. In cases where it is not obvious, we name the function with a `do_`

prefix to indicate that it does I/O.

6.13.3 Trade-offs

We advise against blind adherence to any particular paradigm, principle, or pattern in programming. After all, programming is best described as a form of engineering where one must constantly make compromises and trade-offs. There are no hard and fast rules in programming: it is normally a good idea to follow them, but it is even more important to understand their provenance, costs, and benefits so that the programmer can judge whether the pay-offs are worthwhile. Over-engineering can increase the size of the codebase, which can impede understanding just as much as unruly one does.

Chapter 7

Results and analysis

Finally, we discuss the results we obtained with our many-body methods for quantum dots and nuclei.

7.1 Methodology

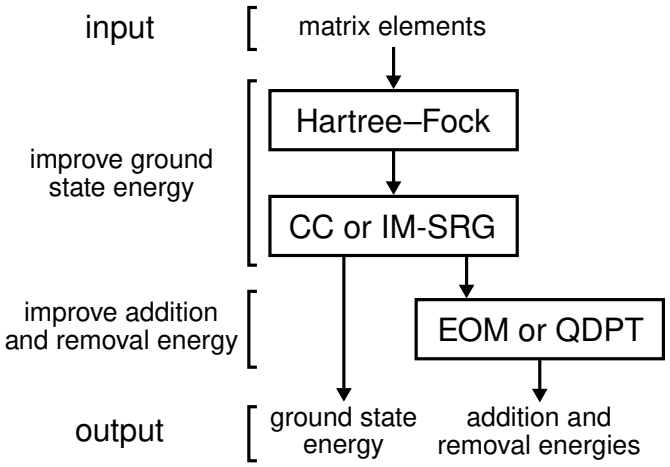

Figure 7.1: A schematic view of the various ways in which many-body methods in this work could be combined to calculate ground state, addition, and removal energies.

There is significant flexibility in the application of many-body methods. The approaches we use are shown in Fig. 7.1. Applying the methods in this order maximizes the benefits of each method: HF acts as an initial, crude procedure to soften the Hamiltonian, followed by IM-SRG or CC (coupled cluster method) to refine the ground state energy, and then finally QDPT or EOM (equations-of-motion method) to refine the addition and removal energies. We expect single-

reference IM-SRG and CC to recover a substantial part of the dynamical correlations, while QDPT and EOM help account for static correlations.

The general process begins with setting up the input matrix elements. Afterward, there are several paths through which one can traverse Fig. 7.1 to obtain output observables. Our primarily focus is on the three combinations:

a. HF + IM-SRG(2) + QDPT3, computed by us,
b. HF + IM-SRG(2) + EOM2, computed and contributed by Nathan M. Parzuchowski, and
c. HF + CCSD + EOM2, computed and contributed by Samuel J. Novario.

It is possible to omit some steps of the process. For example, one can omit HF, but continue with the remaining two steps. While this is doable, from our experience HF significantly improves the results of the later post-HF methods at very low cost compared to the post-HF methods. Therefore, in practice there is little reason to omit HF. We will however investigate the effects of removing one or more of the post-HF methods.

Since every calculation in this work begins with the HF stage, we will not explicitly state *HF* unless there is no post-HF method used at all, in which case we write *HF only*.

All calculations of ground state energy E_N in this work are restricted to cases where the number of particles N is a magic number, i.e. a **closed shell** system. This is a limitation of the many-body methods used in this work and while there are ways to overcome this limit they are beyond the scope of this work (see [Her17]). Addition/removal energies $\varepsilon^{(\pm)}$ are similarly restricted in that we only calculate the energy difference between E_N of a closed shell system and $E_{N\pm1}$ of the same system but with one particle added/removed:

$$\varepsilon^{(+)} = E_{(N+1)} - E_N$$

$$\varepsilon^{(-)} = E_N - E_{(N-1)}$$

7.2 Results for quantum dots

The results in this section have been previously presented in [Yua+17]. There is a difference, however, in the numerical data for HF + QDPT3 presented here. In the original paper, one of the QDPT3 was calculated with the incorrect sign, affecting all HF + QDPT3 results. It has been corrected in this chapter. The conclusions remain unchanged.

Ideally, ground state energies should be characterized entirely by the two system parameters (N, ω), where N is number of particles and ω is the oscillator frequency. However, the methods that we study are limited to a *finite* (truncated) basis and the results depend on the level of truncation. This is characterized by K, the total number of shells in the single-particle basis. Thus, results are generally presented as a graph plotted against K. In Sec. 7.2.4 we discuss how to estimate results as $K \longrightarrow \infty$ (infinite-basis limit) through extrapolations.

The addition and removal energies are similar, but they require an additional parameter: the total orbital angular momentum M_ℓ, defined as the sum of the m_ℓ of each particle. This is due to the presence of multiple states with near-degenerate energies. For this work, we will consider exclusively the addition/removal energies with the lowest $|M_\ell|$ subject to the constraint that the particle added/removed lies within the next/last shell. This means the $N + 1$ states of interest are those with $|M_\ell| = K_\mathrm{F} \bmod 2$ (where mod stands for the modulo operation) where K_F is the number of occupied shells, while the $N - 1$ states of interest are those with $|M_\ell| = 1 - (K_\mathrm{F} \bmod 2)$.

Not all cases are solvable with our selection of many-body methods. Low frequency systems are particularly strenuous for these many-body methods due to their strong correlations, leading to equations that are difficult and expensive to solve numerically. In the tables, *n.c.* marks the cases where IM-SRG(2) or CCSD either diverged or converged extremely slowly. This also affects the extrapolation results in Sec. 7.2.4, as for consistency reasons we chose to extrapolate only when all five points were available.

Numerical calculations in this section are performed with a relative precision of about 10^{-5} or lower. This does not necessarily mean the results are as precise as 10^{-5}, since numerical errors tend to accumulate over the multiple steps of the calculation, thus the precision of the final results is expected to be roughly 10^{-4}.

7.2.1 Ground state energy

Table 7.1: Ground state energy of quantum dots with N particles and an oscillator frequency of ω. For every row, the calculations are performed in a harmonic oscillator basis with K shells. The abbreviation *n.c.* stands for *no convergence*: these are cases where IM-SRG(2) or CCSD either diverged or converged extremely slowly.

N	ω	K	HF	MP2	IM-SRG(2)	CCSD
6	0.1	14	3.8524	3.5449	3.4950	3.5831
6	0.28	14	8.0196	7.6082	7.5731	7.6341
6	1.0	14	20.7192	20.1939	20.1681	20.2000
12	0.1	16	12.9247	12.2460	12.2215	12.3583
12	0.28	16	26.5500	25.6433	25.6259	25.7345
12	1.0	16	66.9113	65.7627	65.7475	65.8097
20	0.1	16	31.1460	29.9674	29.9526	30.1610
20	0.28	16	63.5388	61.9640	61.9585	62.1312
20	1.0	16	158.0043	156.0239	156.0233	156.1243
30	0.1	16	62.6104	60.8265	60.6517	61.0261
30	0.28	16	126.5257	124.1279	124.1041	124.3630
30	1.0	16	311.8603	308.8611	308.8830	309.0300
42	0.1	20	110.7797	108.1350	108.0604	108.5150
42	0.28	20	223.5045	219.9270	220.0227	220.3683
42	1.0	20	547.6832	543.2139	543.3399	543.5423
56	0.1	20	182.6203	179.2370	n.c.	179.6938
56	0.28	20	363.8784	359.1916	359.1997	359.6744
56	1.0	20	885.8539	879.9325	880.1163	880.3781

Fig. 7.2 and Tbl. 7.1 display a selection of ground state energies calculated using HF + IM-SRG(2) and HF + CCSD as described in Sec. 7.1. We include results from Møller–Plesset perturbation theory to second order (MP2), DMC [Høg13], and FCI [Ols13] (see Tbl. 7.2) for comparison where available.

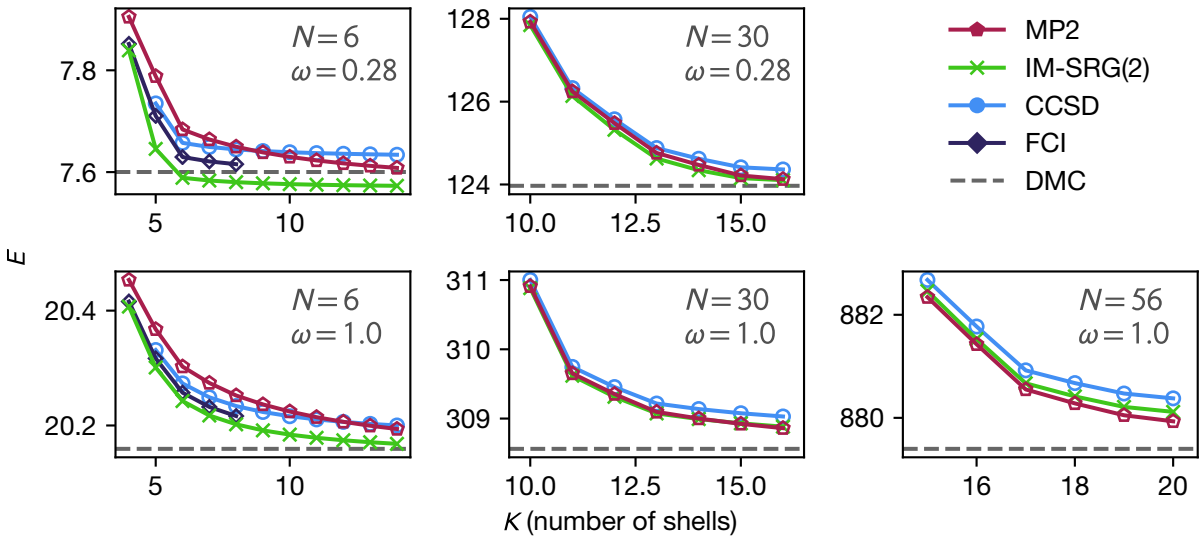

Figure 7.2: Plots of ground state energy of quantum dots with N particles and an oscillatory frequency of ω against the number of shells K. Since DMC does not utilize a finite basis, the horizontal axis is irrelevant and DMC results are plotted as horizontal lines.

We do not include results from *HF only* to avoid overshadowing the comparatively smaller differences between the non-HF results in the plots. Some HF results can be found in Fig. 7.5 instead. Generally, the HF ground state energies differ from the non-HF ones by a few to several percent, whereas non-HF energies tend to differ from each other by less than a percent.

With respect to the number of shells, both IM-SRG(2) and CCSD appear to converge slightly

Table 7.2: Similar to Table 7.1, this table compares the ground state energies of quantum dots calculated using IM-SRG(2), CCSD, and FCI [Ols13].

N	ω	K	IM-SRG(2)	CCSD	FCI
2	0.1	5	n.c.	0.4416	0.4416
2	0.28	5	0.9990	1.0266	1.0266
2	1.0	5	3.0068	3.0176	3.0176
2	0.1	10	n.c.	0.4411	0.4411
2	0.28	10	0.9973	1.0236	1.0236
2	1.0	10	2.9961	3.0069	3.0069
6	0.1	8	3.4906	3.5853	3.5552
6	0.28	8	7.5802	7.6446	7.6155
6	1.0	8	20.2020	20.2338	20.2164

faster than second order perturbation theory (MP2), mainly due to the presence of higher order corrections in IM-SRG(2) and CCSD.

There are a few cases where the IM-SRG over-corrects the result, leading to an energy lower than the quasi-exact DMC results. This is not unexpected given that, unlike the HF results, the IM-SRG method is non-variational in the presence of operator truncations, which in turn results in small unitarity violations. This over-correction tends to occur when the frequency is low (high correlation), or when *few* particles are involved.

7.2.2 Addition and removal energies

Table 7.3: Addition energy of quantum dot systems. See Table 7.1 for details.

N	ω	K	HF +QDPT3	IM-SRG(2) +QDPT3	IMSRG(2) +EOM	CCSD +EOM
6	0.1	14	1.1849	1.2014	1.1809	1.1860
6	0.28	14	2.4737	2.5003	2.4916	2.4833
6	1.0	14	6.4374	6.4546	6.4532	6.4453
12	0.1	16	1.9129	1.9248	1.9094	1.9014
12	0.28	16	3.9266	3.9394	3.9354	3.9205
12	1.0	16	9.9182	9.9256	9.9274	9.9136
20	0.1	16	2.7383	2.7143	2.7149	2.7040
20	0.28	16	5.5552	5.5400	5.5409	5.5226
20	1.0	16	13.7902	13.7799	13.7844	13.7667
30	0.1	16	3.7185	3.6467	3.6536	3.6454
30	0.28	16	7.3230	7.2719	7.2810	7.2615
30	1.0	16	17.9321	17.9022	17.9088	17.8875
42	0.1	20	4.6931	4.5751	4.5867	4.5750
42	0.28	20	9.2021	9.1072	9.1188	9.0963
42	1.0	20	22.3494	22.2941	22.3012	22.2766
56	0.1	20	5.9292	n.c.	n.c.	5.7661
56	0.28	20	11.3123	11.1683	11.1813	11.1518
56	1.0	20	26.9828	26.9033	26.9118	26.8842

The results of our addition and removal energy calculations are summarized in Fig. 7.3 and Fig. 7.4 respectively. The figures show the addition/removal energies for using the approaches

Table 7.4: Removal energy of quantum dot systems. See Table 7.3 for details.

N	ω	K	HF +QDPT3	IM-SRG(2) +QDPT3	IMSRG(2) +EOM	CCSD +EOM
6	0.1	14	1.0073	0.9500	0.9555	1.0054
6	0.28	14	2.0778	2.0346	2.0398	2.0782
6	1.0	14	5.2217	5.1950	5.1970	5.2220
12	0.1	16	1.7518	1.6961	1.7017	1.7503
12	0.28	16	3.5782	3.5334	3.5366	3.5779
12	1.0	16	8.8426	8.8104	8.8102	8.8409
20	0.1	16	2.5610	2.5133	2.5184	2.5670
20	0.28	16	5.2061	5.1639	5.1660	5.2105
20	1.0	16	12.7548	12.7201	12.7185	12.7546
30	0.1	16	3.5161	3.4445	3.4485	3.5113
30	0.28	16	6.9697	6.9282	6.9289	6.9785
30	1.0	16	16.9584	16.9243	16.9215	16.9613
42	0.1	20	4.4481	4.3868	4.3902	4.4451
42	0.28	20	8.8124	8.7765	8.7766	8.8263
42	1.0	20	21.3807	21.3453	21.3421	21.3848
56	0.1	20	5.6254	n.c.	n.c.	5.6341
56	0.28	20	10.8917	10.8471	10.8454	10.8957
56	1.0	20	26.0420	26.0094	26.0056	26.0507

mentioned in Sec. 7.1. Where available, results from diffusion Monte Carlo (DMC) [Ped+11] are shown as a dashed line.

As before, we do not include results from *HF only* in these plots as they are significantly further from the rest. Analogously, we also exclude results from pure IM-SRG (i.e. without QDPT nor EOM) or pure CCSD, as QDPT or EOM both add significant contributions to addition and removal energies. Some HF only and pure IM-SRG results can be seen in Fig. 7.5.

There is strong agreement between IM-SRG(2) + QDPT3 and IM-SRG(2) + EOM2 in many cases, and slightly weaker agreement between the IM-SRG and CCSD families. This suggests that the EOM2 corrections are largely accounted for by the inexpensive QDPT3 method. However, in some cases, most notably with few particles and high correlations (low frequency), the IM-SRG(2) + QDPT3 result differs significantly from both IM-SRG(2) + EOM2 and CCSD + EOM2.

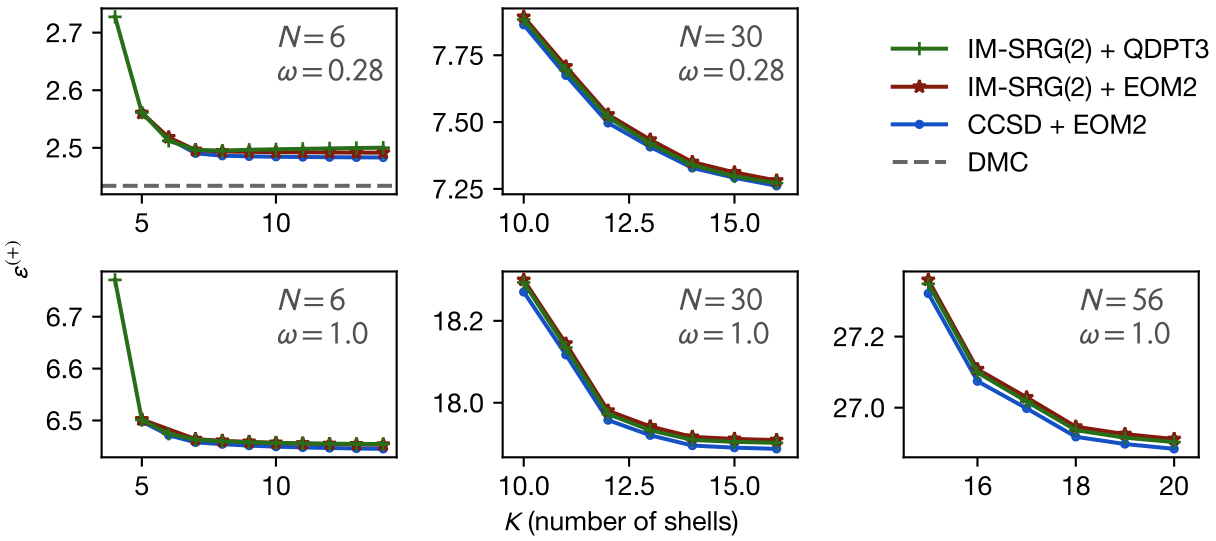

Figure 7.3: Addition energies for a selection of quantum dot parameters. See Fig. 7.2 for details.

7.2.3 Rate of convergence

To analyze the rate of convergence more quantitatively, we define ρ_K as the relative backward difference of the energy (**relative slope**):

$$\rho_K = \frac{\varepsilon_K - \varepsilon_{(K-1)}}{\varepsilon_K}$$

The denominator allows the quantity to be meaningfully compared between different systems. We expect this quantity to become increasingly small as the calculations converge towards the complete basis set limit.

In Fig. 7.6, we plot the ρ_{15} for IM-SRG(2) + QDPT3. The many-body methods were tested against a *modified* Coulomb-like interaction, parametrized by two lengths σ_A and σ_B that characterize the range of the interaction:

$$V_{\sigma_\mathrm{A},\sigma_\mathrm{B}}(r) = \frac{(1+c)^{1-1/c}}{c}\left(1 - \mathrm{e}^{-r^2/(2\sigma_\mathrm{A}^2)}\right)\mathrm{e}^{-r^2/(2\sigma_\mathrm{B}^2)}\frac{1}{r}$$

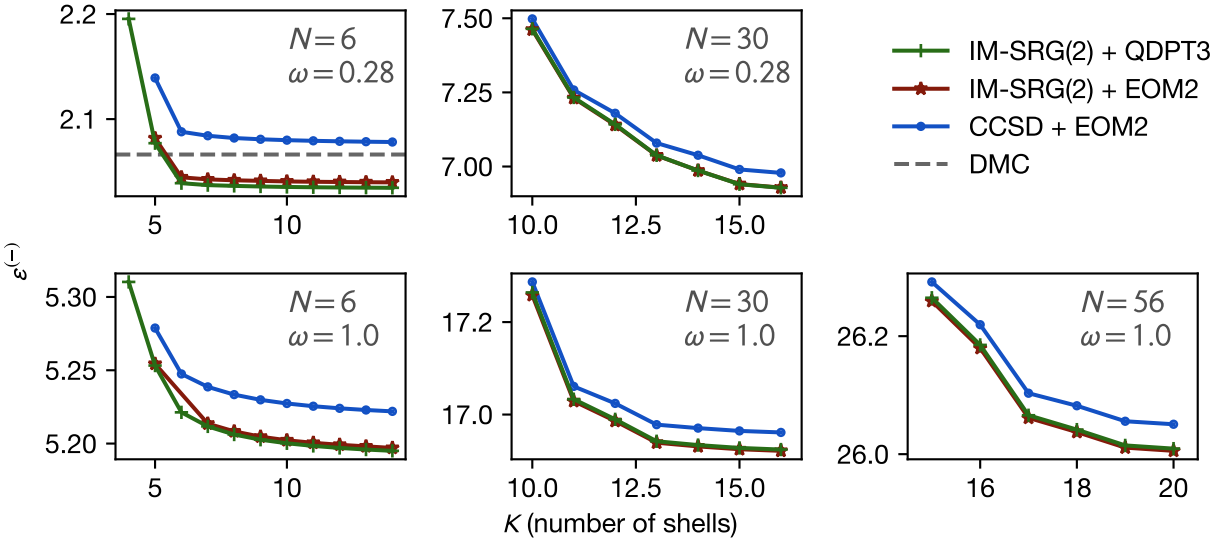

Figure 7.4: Removal energies for a selection of quantum dot parameters. See Fig. 7.3 for details.

where $c = \sqrt{\sigma_B/\sigma_A}$. The coefficient is chosen to ensure the peak of the envelope remains at unity. With $(\sigma_A, \sigma_B) = (0, \infty)$ one recovers the original Coulomb interaction. By increasing σ_A one can truncate the short-range part of the interaction, and analogously by increasing σ_B one can truncate the long-range part of the interaction. For our numerical experiments we considered the following four combinations of (σ_A, σ_B): $(0, \infty)$, $(\frac{1}{2}, \infty)$, $(0, 4)$, $(\frac{1}{2}, 4)$.

Reducing the short-range part of the interaction appears to improve the rate of convergence substantially. Many of the cases have reached the precision of the ODE solver (10^{-5} to 10^{-6}). In contrast, eliminating the long-range part of the interaction had very little effect. This suggests that the main cause of the slow convergence lies in the highly repulsive, short-ranged part of the interaction, which leads to the presence of nondifferentiable cusps (the so-called *Coulomb cusps*) in the exact wave functions that are difficult to reproduce exactly using linear combinations of the smooth harmonic oscillator wave functions.

The convergence is negatively impacted at lower frequencies and, to a lesser extent, by the increased number of particles. Both are expected: lower frequencies increase the correlation in

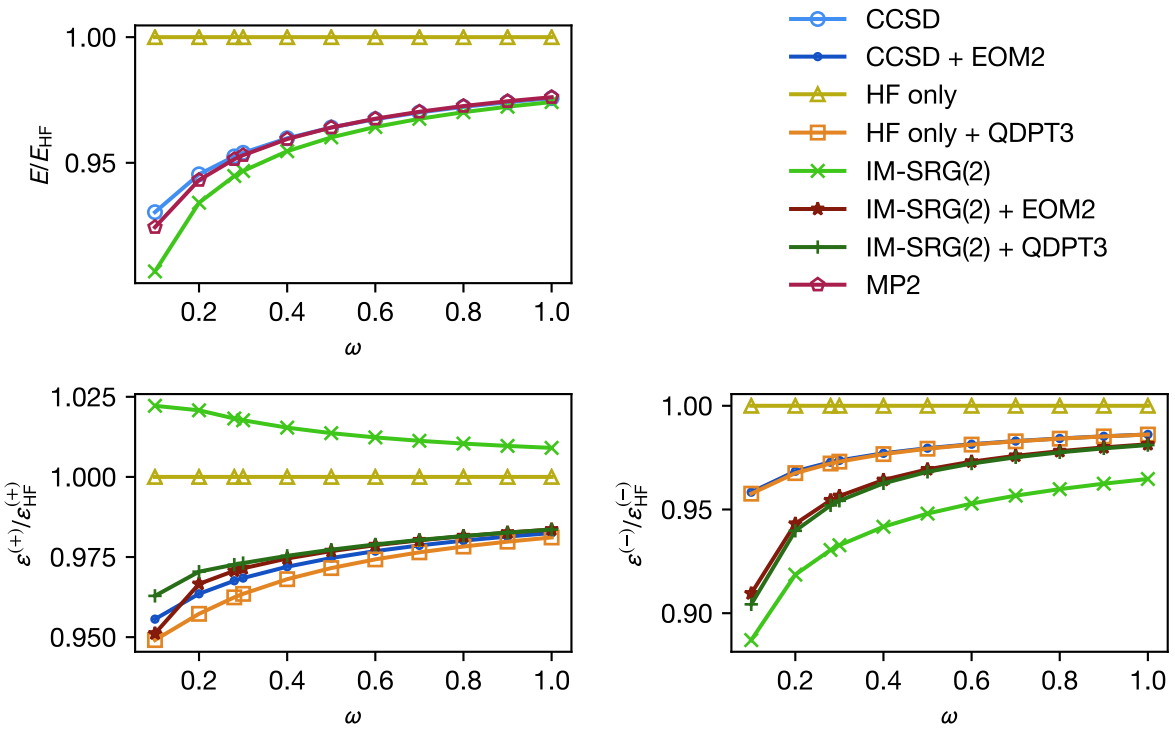

Figure 7.5: The behavior of ground state, addition, and removal energies as a function of the oscillator frequency ω, with $K = 10$ shells in the basis. The energy is normalized with respect to the HF values to magnify the differences. Lower frequency leads to stronger correlations and thereby a more difficult problem.

the system, while higher number of particles naturally require more shells to converge.

In general, there does not appear to be any difference between the convergence behavior of addition energies as compared to that of removal energies.

7.2.4 Extrapolation

To reduce errors from the basis set truncation, one can either use explicitly correlated R12/F12 methods that account for the correct cusp behavior in many-electron wave functions [Kut85; KK87; KBV12], or one can use basis extrapolation techniques. In the present work, we focus on the latter. As derived by Kvaal [Kva09; KHM07], the asymptotic convergence of quantum dot observables in a finite harmonic oscillator basis can be approximately described by a power law

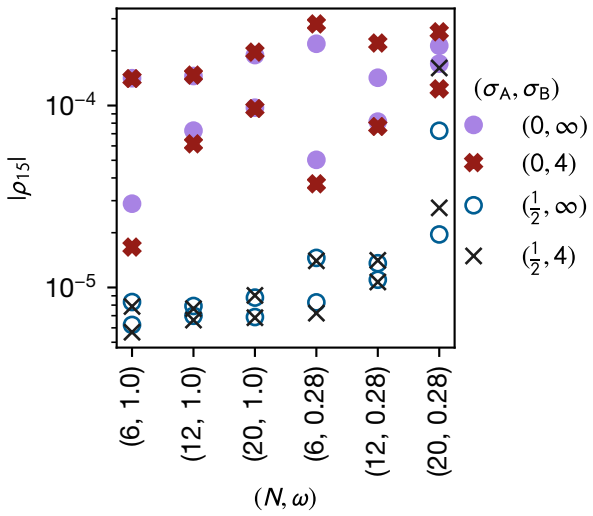

Figure 7.6: The impact of the interaction on convergence of addition and removal energies using IM-SRG(2) + QDPT3. For clarity, the plot does not distinguish between addition and removal energies. The horizontal axis shows the system parameters, where N is the number of particles and ω is the oscillator frequency. The vertical axis shows $|\rho_{15}|$ (*relative slope*), which estimates the rate of convergence at 15 total shells. The lower the value of $|\rho_{15}|$, the faster the convergence. The data points are categorized by the interactions. The trends suggest that the singular short-range part of the interaction has a much stronger impact on the convergence than the long-range tail.

model:

$$\Delta E \propto K^{-\beta}$$

where ΔE is the difference between the finite-basis result and the infinite-basis result, K is the number of shells in the single-particle basis, and β is some positive real exponent. The *smoothness* of the exact wave function determines the rate of the convergence: the more times the exact wave function can be differentiated, the higher the exponent β.

We note that this model was derived under the assumption that all correlations are included in the calculation (i.e. FCI), thus we are making an assumption that our selection of methods approximately obey the same behavior. The validity of this assumption will be assessed at the end of this section.

Table 7.5: Extrapolated ground state energies for quantum dots with fit uncertainties, computed from the approximate Hessian in the Levenberg–Marquardt fitting algorithm. These uncertainties also determine the number of significant figures presented. Extrapolations are done using 5-point fits where the number of shells K ranges between $K_{\text{stop}} - 4$ and K_{stop} (inclusive). The abbreviation *n.c.* stands for *no convergence*: these are extrapolations where, out of the 5 points, at least one of them was unavailable because IM-SRG(2) or CCSD either diverged or converged extremely slowly.

N	ω	K_{stop}	MP2	IM-SRG(2)	CCSD
6	0.1	14	3.5108(4)	3.4963(5)	3.581 83(2)
6	0.28	14	7.5608(3)	7.569 71(2)	7.628 12(7)
6	1.0	14	20.129 98(5)	20.1481(3)	20.1791(3)
12	0.1	16	12.198(7)	12.2217(2)	12.3575(4)
12	0.28	16	25.548(1)	25.6146(1)	25.7190(2)
12	1.0	16	65.627(2)	65.6970(7)	65.7579(8)
20	0.1	16	29.87(5)	29.950(1)	30.13(2)
20	0.28	16	61.88(1)	61.946(3)	62.114(5)
20	1.0	16	155.758(3)	155.912(4)	156.010(3)
30	0.1	16	59.8(2)	n.c.	60.3(2)
30	0.28	16	123.95(8)	124.00(4)	124.26(5)
30	1.0	16	308.80(5)	308.85(3)	309.00(3)
42	0.1	20	106.3(4)	107.0(1)	107.1(4)
42	0.28	20	219.6(2)	219.89(8)	220.2(1)
42	1.0	20	542.686(9)	543.074(3)	543.276(4)
56	0.1	20	172.9(6)	n.c.	174(1)
56	0.28	20	357.3(6)	n.c.	358.1(5)
56	1.0	20	879.86(8)	880.07(7)	880.33(7)

In general, the exponent β cannot be determined *a priori*, thus we will empirically compute β by fitting the following model through our data:

$$E = \alpha K^{-\beta} + \gamma \tag{7.1}$$

As a nonlinear curve fit, it can be quite sensitive to the initial parameters. Therefore, good guesses of the parameters are necessary to obtain a sensible result. For this, we first fit a linear model of

Table 7.6: Extrapolated addition energies for quantum dots with fit uncertainties. The abbreviation *n.c.* has the same meaning as in Table 7.5. The abbreviation *n.f.* stands for *no fit*: this particular extrapolation resulted in unphysical parameters ($\beta \leq 0$). See Table 7.5 for other details.

N	ω	K_{stop}	IM-SRG(2) +QDPT3	IMSRG(2) +EOM	CCSD +EOM
6	0.1	14	1.206(2)	1.180 95(5)	1.185 81(2)
6	0.28	14	2.63(8)	2.490 39(2)	2.482 13(4)
6	1.0	14	6.4536(1)	6.4491(7)	6.440 747(1)
12	0.1	16	n.f.	1.909 274(1)	1.901 39(2)
12	0.28	16	3.925(5)	3.9339(3)	3.918 520(9)
12	1.0	16	9.9235(2)	9.9235(5)	9.9070(1)
20	0.1	16	2.708(4)	2.705(3)	2.682(2)
20	0.28	16	5.539 15(1)	5.5405(3)	5.521 80(7)
20	1.0	16	13.7759(8)	13.779(1)	13.760(2)
30	0.1	16	n.c.	n.c.	3.40(2)
30	0.28	16	7.18(3)	7.18(5)	7.16(4)
30	1.0	16	17.897(4)	17.902(6)	17.880(6)
42	0.1	20	4.19(6)	4.28(4)	4.33(2)
42	0.28	20	9.068(6)	9.08(1)	9.05(1)
42	1.0	20	22.2943(7)	22.301 47(1)	22.2768(9)
56	0.1	20	n.c.	n.c.	3(3)
56	0.28	20	n.c.	n.c.	10.7(3)
56	1.0	20	26.86(4)	26.87(4)	26.84(4)

$\log|\partial E/\partial K|$ against $\log K$:

$$\log\left|\frac{\partial E}{\partial K}\right| = -(\beta + 1)\log K + \log|\alpha\beta|$$

This is useful because linear fits are very robust and will often converge even if the initial parameters are far from their final values. It also provides a means to visually assess the quality of the fit. The derivative is approximated using the central difference:

$$\frac{\partial E}{\partial K} \approx E\left(K + \frac{1}{2}\right) - E\left(K - \frac{1}{2}\right)$$

The process of numerically calculating the derivative can amplify the noise in the data and

Table 7.7: Extrapolated removal energies for quantum dots with fit uncertainties. See Table 7.6 for details.

N	ω	K_{stop}	IM-SRG(2) +QDPT3	IMSRG(2) +EOM	CCSD +EOM
6	0.1	14	0.9509(2)	0.9561(4)	1.004 943(8)
6	0.28	14	2.033 96(1)	2.0387(2)	2.076 20(2)
6	1.0	14	5.188 89(8)	5.186(3)	5.2154(1)
12	0.1	16	1.696 24(8)	1.701 81(6)	1.750 31(7)
12	0.28	16	3.532 236(5)	3.535 12(9)	3.575 27(1)
12	1.0	16	8.8039(4)	8.803 90(1)	8.8331(2)
20	0.1	16	2.5112(6)	2.5163(8)	2.55(1)
20	0.28	16	5.163(1)	5.165(1)	5.208(3)
20	1.0	16	12.7122(4)	12.7101(5)	12.7442(2)
30	0.1	16	n.c.	n.c.	3.35(6)
30	0.28	16	6.88(2)	6.88(2)	6.94(3)
30	1.0	16	16.925(2)	16.923(2)	16.963(2)
42	0.1	20	4.04(8)	4.06(7)	4.1(2)
42	0.28	20	8.73(3)	8.73(3)	8.76(6)
42	1.0	20	21.338(1)	21.335(1)	21.378(2)
56	0.1	20	n.c.	n.c.	5.3(1)
56	0.28	20	n.c.	n.c.	10.75(9)
56	1.0	20	26.008(9)	26.004(8)	26.050(9)

distorts the weights of the data points. Moreover, it does not provide a means to compute γ, the extrapolated energy. Thus a second accurate nonlinear curve fit is necessary.

The parameters α and β are extracted from the linear fit and used as inputs for a power-law fit of E against K. It is necessary to estimate the infinite-basis energy γ as well, which is done by fitting Eq. 7.1 while the parameters α and β are fixed to the initial guesses. The fixing ensures that the fit is still linear in nature and thus highly likely to converge. Afterward, we do a final fit with all three parameters free to vary. All fits are done using the traditional Levenberg–Marquardt (LM) optimization algorithm [Lev44; Mar63] as implemented in `MINPACK` [Mor78; MGH80], with equal weighting of all data points.

There is still one additional tuning knob for this model that is not explicitly part of Eq. 7.1: the range of data points taken into consideration (**fit range**). Since the model describes the *asymptotic*

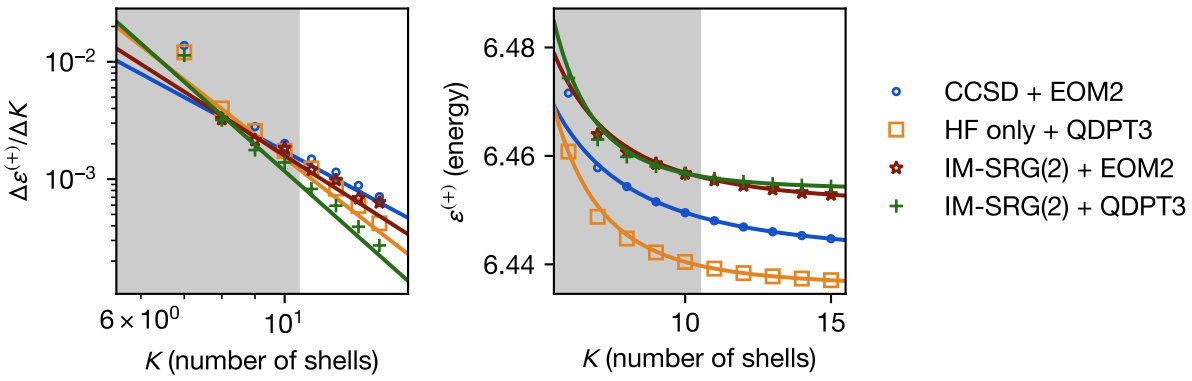

Figure 7.7: A five-point fit of the addition energies of the $(N, \omega) = (6, 1.0)$ system with $K_{\text{stop}} = 15$. The grey shaded region contains the masked data points, which are ignored by the fitting procedure. The left figure plots the central difference of the addition energies $\varepsilon^{(+)}$ with respect to the number of shells K. On such a plot, the power law model should appear as a straight line. The fits are optimized using the procedure described in Sec. 7.2.4. Note that the lines do not evenly pass through the points in the left plot as the fitting weights are tuned for the energy on a linear scale, not the energy differences on a logarithmic scale.

behavior, we do not expect the fit to produce good results when the energy is still very far from convergence. To account for this, we only fit the last few data points within some chosen range. If the range is too large, then the non-asymptotic behavior would perturb the result too much, whereas if the range is too small, there would be more noise and less confidence in whether the trend is legitimate rather than accidental. Empirically, we chose to fit the last 5 points of our available data. The results are shown in Tbls. 7.5, 7.6, and 7.7. A specific example of the fit is shown in Fig. 7.7

The LM fitting procedure also computes uncertainties for the parameters from an approximate Hessian of the model function. It is therefore tempting to use the uncertainty of the fit to quantify the uncertainty of the extrapolated energy. We certainly would not expect this to account for the error due to the operator truncation, but how accurately does it quantify the discrepancy of our extrapolated result from the true infinite-basis energy?

We investigated this idea by performing a fit over *all possible* 5-point fit ranges $[K_{\text{stop}} - 4, K_{\text{stop}}]$.

By comparing the extrapolated results at varying values of K_{stop} with the extrapolated result at the highest possible K_{stop} and treating the latter as the "true" infinite-basis result, we can statistically assess whether the fit uncertainties are a good measure of the discrepancy from the true infinite-basis result. Our results show a somewhat bimodal distribution: when the relative fit uncertainty is higher than $10^{-3.5}$, the fit uncertainty quantifies the discrepancy well; otherwise, the fit uncertainty underestimates the discrepancy by a factor of 10 or less.

Unlike the other methods, HF energies are somewhat unusual in that they generally do not conform to the power-law model. In fact, the plots indicate an *exponential* convergence with respect to the number of shells, which has also been observed in molecular systems [Hal+99]. We surmise that HF is insensitive to the Coulomb cusp.

Nonetheless, despite the poor fits that often arise, the extrapolated energies are often quite good for HF. This is likely due to its rapid convergence, which leaves very little degree of freedom even for a poorly chosen model. Moreover, we found that the fit uncertainties of the energy are fairly good measures of the true discrepancy.

Not all fits yield a positive value of β for addition and removal energies, which suggests that the data points do not converge, or require a very high number of shells to converge. This affects exclusively IM-SRG(2) + QDPT3 for systems with few particles and low frequencies, indicating that perturbation theory is inadequate for such systems.

7.3 Results for nuclei

In this section, we provide a few selected results for nuclear systems as a proof of concept. Due to time constraints, we have not been able to run calculations with higher values of e_{max} (Eq. 5.9) or to explore a greater span of the oscillator frequency ω of the basis. We expect to gather a more diverse collection of results in a future publication [Yua+].

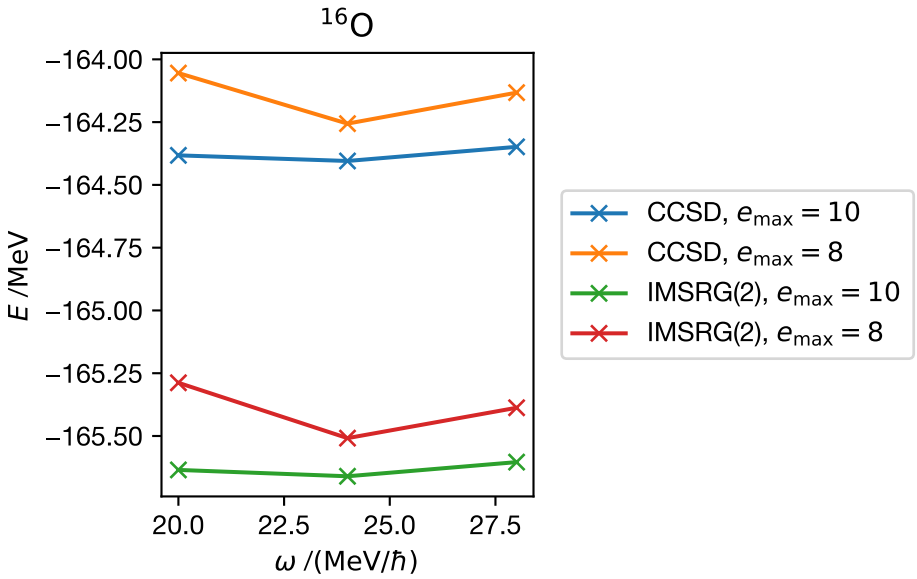

Figure 7.8: Ground state of ^{16}O, computed using IM-SRG(2) and CCSD with the N^{3}LO(Λ = 500 MeV, λ_{SRG} = 2 fm^{-1}) interaction

Fig. 7.8 shows the computed ground state energies of oxygen-16 plotted as a function of the basis oscillator frequency ω. The results were computed using two many-body methods: HF + IM-SRG and HF + CCSD (coupled cluster singles and doubles). The interaction we use is the N^{3}LO(Λ = 500 MeV) interaction of [EM03], softened by SRG evolution to λ_{SRG} = 2 fm^{-1}. We use a strictly two-body nuclear interaction here.

We observe that both methods agree with each other to about 1 MeV. Furthermore, we see that the results are very close to convergence with respect to e_{max}. The cup-shaped curve suggests the existence of a local minimum near ω = 24 MeV/$\hbar$. The curve is quite flat, which again suggests that the results are nearly converged.

For reference, the experimental value is about −128 MeV. The reason for this large discrepancy is that the SRG softening of the interaction has introduced a significant three-body component to the nuclear interaction that we are neglecting. With the appropriate treatment of this three-body component, one can achieve values much closer to experimental data. Readers interested in more

elaborate ground state energy calculations using IM-SRG may consult [Her+13].

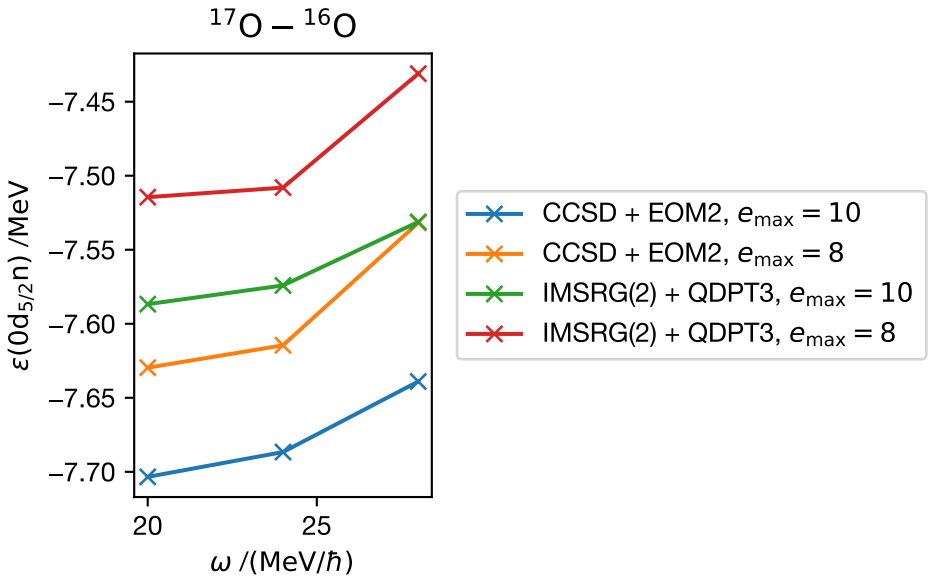

Figure 7.9: Addition energy from ^{16}O to ^{17}O, computed using IM-SRG(2) + QDPT3 and CCSD + EOM2 with the N^{3}LO($\Lambda = 500\,\text{MeV}, \lambda_{\text{SRG}} = 2\,\text{fm}^{-1}$) interaction

We now consider the addition energy going from oxygen-16 to oxygen-17, achieved by adding a neutron to the $0d_{5/2}$ state. This is presented in Fig. 7.9. The energies were calculated using HF + IM-SRG(2) + QDPT3 and HF + CCSD + EOM2 with the same nuclear interaction as before.

Both methods agree with each other to about 0.2 MeV. With respect to e_{max}, both curves are converging at a rate of 0.05 MeV per shell, which is also quite good. Unlike the ground state however, the curve is no longer cup-shaped, but increasing with the frequency. This is not entirely unusual, as such shapes have been observed in other non-energy observables. It may also be possible that there is a local minimum to the left side of the graph, though we consider this unlikely given our removal energy results in the next figure.

For comparison, the experimental value is about −4.1 MeV. We suspect that the absence of three-body forces is a significant reason for this discrepancy.

The removal energy going from oxygen-16 to nitrogen-15 is achieved by removing a proton

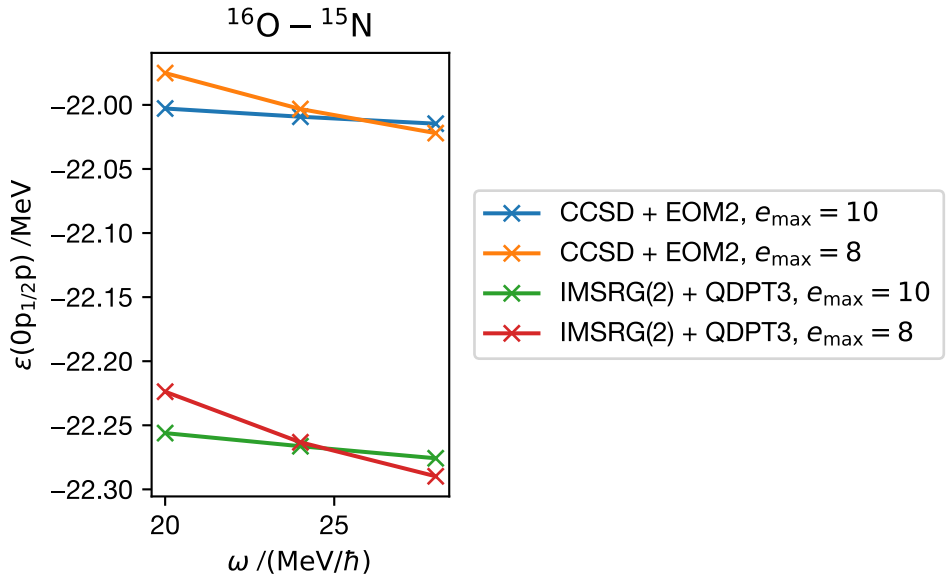

Figure 7.10: Removal energy from ^{16}O to ^{15}N, computed using IM-SRG(2) + QDPT3 and CCSD + EOM2 with the N^{3}LO(Λ = 500 MeV, λ_{SRG} = 2 fm^{-1}) interaction

from the $0p_{1/2}$ state. This is presented in Fig. 7.10. The energies were calculated using HF + IM-SRG(2) + QDPT3 and HF + CCSD + EOM2 with the same nuclear interaction as before.

Both methods agree with each other to about 0.3 MeV. With respect to e_{max}, both curves are converging at a rate of 0.02 MeV per shell, which is really good considering the magnitude of the removal energy. Like the addition energies, the curve is not cup-shaped, but leans to the side. However, in this case we clearly see a point where the curves at different e_{max} values cross each other, at around ω = 25 MeV/$\hbar$.

For comparison, the experimental value is about −12 MeV. Again, our values are significantly different, likely due to missing three-body contributions.

For reference, we have also attached the results for nitrogen-15 in the excited state $J^\pi = \frac{3}{2}^-$ in Fig. 7.11, oxygen-23 in Fig. 7.12, and oxygen 21 in Fig. 7.13. All these results were calculated using the same approach as before. In all cases, the difference between the two methods is very small and the convergence with respect to e_{max} appears to be quite good.

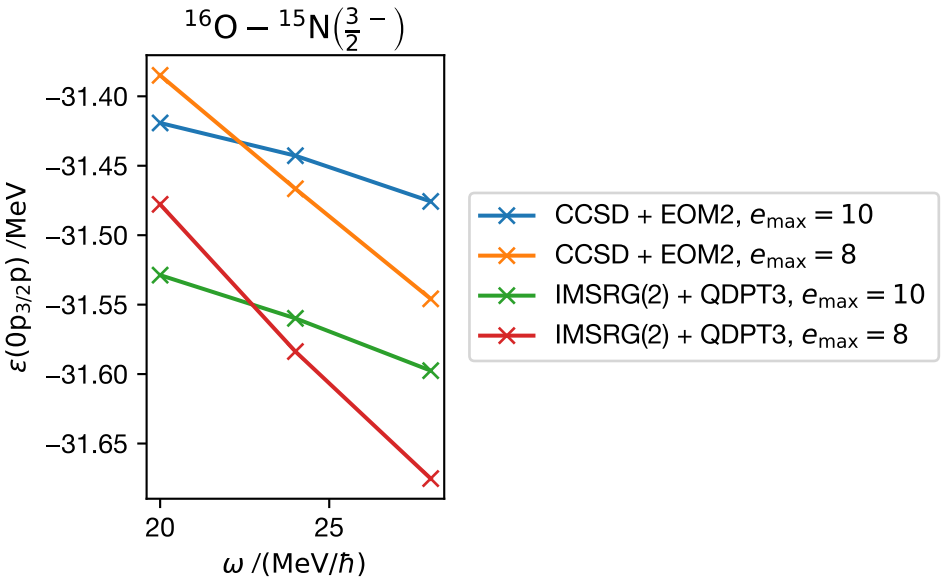

Figure 7.11: Removal energy from ^{16}O to ^{15}N in the excited $J^\pi = \frac{3}{2}^-$ state, computed using IM-SRG(2) + QDPT3 and CCSD + EOM2 with the N^{3}LO($\Lambda = 500\,\text{MeV}, \lambda_{\text{SRG}} = 2\,\text{fm}^{-1}$) interaction

From the preliminary results so far, we see that our perturbative results agree quite well with the EOM results. In conjunction with the testing and verification described in Sec. 6.10, this helps confirm the correctness of our J-scheme implementation. We believe these results indicate a promising start for more extensive studies of nuclei through this approach.

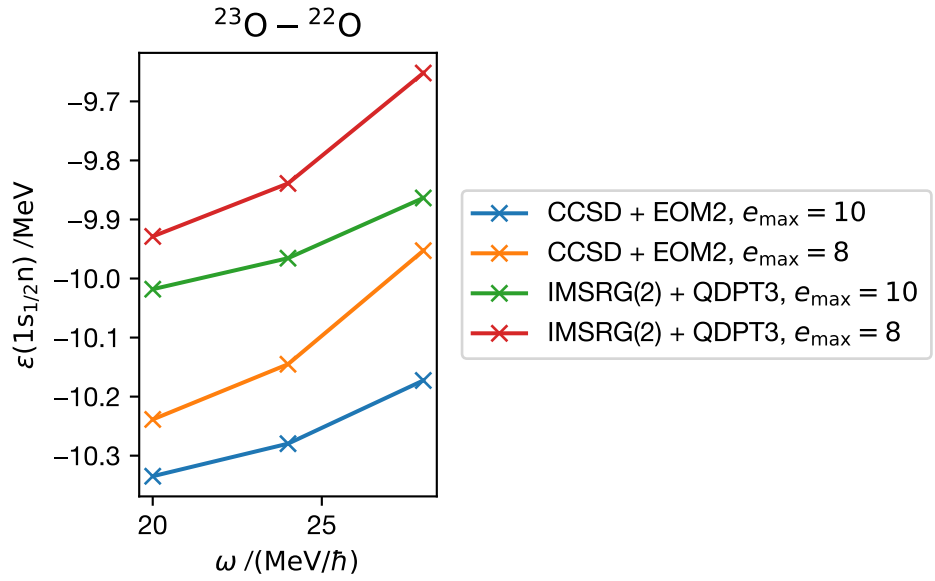

Figure 7.12: Addition energy from ^{22}O to ^{23}O, computed using IM-SRG(2) + QDPT3 and CCSD + EOM2 with the N^{3}LO(Λ = 500 MeV, λ_{SRG} = 2 fm^{-1}) interaction

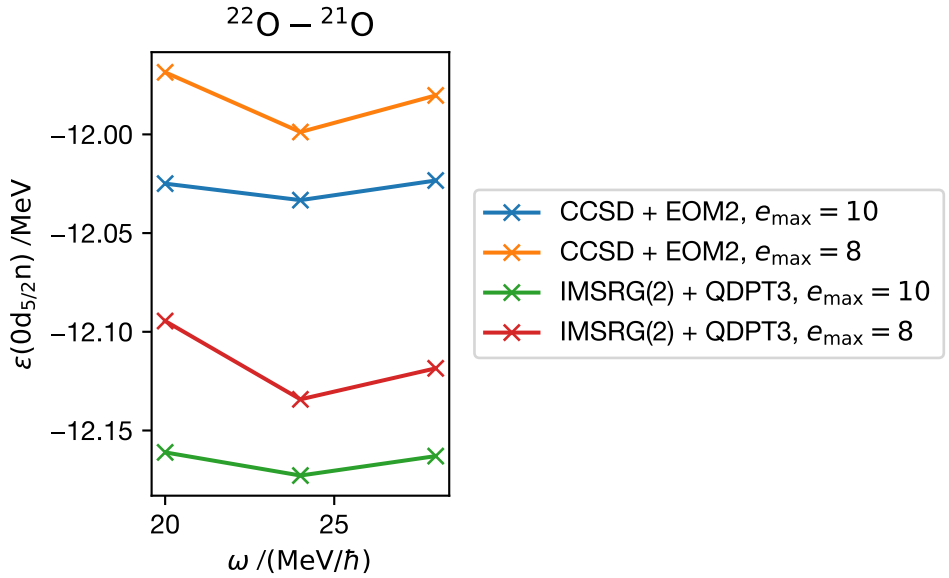

Figure 7.13: Removal energy from ^{22}O to ^{21}O, computed using IM-SRG(2) + QDPT3 and CCSD + EOM2 with the N^{3}LO(Λ = 500 MeV, λ_{SRG} = 2 fm^{-1}) interaction

Chapter 8

Conclusions

In this work, our focus has been on the calculation of single-particle energies (addition and removal energies) of quantum dots and nuclei using a combination of Hartree–Fock (HF) theory, in-medium similarity renormalization group theory up to two-body operators (IM-SRG(2)), and quasidegenerate perturbation theory at third order (QDPT3). We have compared the results to other methods like equations-of-motion up to two-particle excitations (EOM2) and coupled cluster with singles and doubles (CCSD) and found good agreement in the majority of the systems. Thus, we have a reasonably effective and inexpensive way to compute energies of states near closed-shell nuclei.

To achieve this calculation, we have developed an open-source J-scheme implementation of the three major many-body methods verified by a variety of tests. It is capable of calculating various quantum systems, including quantum dots and nuclei. The framework of the code is highly flexible: one can readily add additional quantum systems simply by writing a module that supplies the appropriate input single-particle basis (Sec. 6.4.4.1).

In concert with the J-scheme implementation, we have also developed a graphical tool for painless manipulation of angular momentum coupling diagrams. This greatly reduces the effort required to derive J-scheme equations and eliminates many sources of human error. We expect this to be particularly useful in theories where spherical tensor operators occur.

8.1 Future perspectives

There are many directions in which our current work can be improved upon. The most immediate extension is the exploration of additional parameters for our nuclear calculations, including additional oscillator frequencies ω, additional values of $e_{\max}$ (maximum shell index, Eq. 5.9), and of course additional nuclear isotopes. There are numerous possibilities here.

After obtaining nuclear results with more parameters, we could perform a more detailed analysis of the convergence patterns with respect to both $e_{\max}$ and ω. We can also compute extrapolations using, for example, the prescription in [Her+16].

As we have already implemented systems like infinite nuclear matter [HLK17] and homogeneous electron gas [SG13], we could explore these systems and analyze the quality our method in these systems. Neutron drop calculations can also be readily achieved with our code since it uses essentially the same basis as nuclei.

It is possible to construct valence shell model Hamiltonians using only QDPT [HKO95], but concerns were raised about its convergence due to the strength of the nuclear interaction. Given our preliminary but promising results, it may be possible to use IM-SRG + QDPT to construct valence shell model Hamiltonians and operators with comparable quality to those from EOM-based approaches, which are significantly more expensive.

The inclusion of three-body force is likely a necessity for results that are comparable with experimental data. We can introduce a large fraction of its contribution through the three-body normal-ordering process, which is computationally tractable, unlike IM-SRG(3). We could also upgrade the HF framework to include three-body forces.

We could improve the IM-SRG(2) approximation by incorporating some of the truncated higher-body terms in the commutator through approximate techniques such as those described in [Her+16; Mor16]. It may also be worth evaluating additional QDPT terms at fourth order

for greater accuracy. Since the nature of IM-SRG can eliminate a large number of QDPT terms, QDPT4 may be feasible. Some classes of diagrams could even be summed to infinite order through resummation techniques.

Our J-scheme implementation is not yet fully optimized. Some of the expensive commutator terms are still coded fairly naively and could be improved. We have also made very little use of parallelization at either the shared-memory or distributed-memory level – currently, parallelization occurs primarily within the external GEMM implementation, which is limited to threads. The Shampine–Gordon ODE solver library that we use is very much designed for distributed-memory parallelization – if we enable this feature, it would help distribute both the computational and memory load across multiple nodes. Several of the expensive GEMM calculations could be distributed between nodes on a block-by-block basis similar to [Akt+11].

REFERENCES

REFERENCES

[Akt+11] H. M. Aktulga, C. Yang, E. G. Ng, P. Maris, and J. P. Vary. "Large-scale parallel null space calculation for nuclear configuration interaction". *2011 International Conference on High Performance Computing Simulation*. July 2011, pp. 176–185. DOI: 10.1109/HPCSim.2011.5999822.

[AM98] E. Anisimovas and A. Matulis. "Energy spectra of few-electron quantum dots". *J. Phys. Condens. Mat.* 10.3 (1998), p. 601. DOI: 10.1088/0953-8984/10/3/013.

[And+99] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Third. Philadelphia, PA: Society for Industrial and Applied Mathematics, 1999. ISBN: 0-89871-447-8 (paperback). DOI: 10.1137/1.9780898719604. URL: http://www.netlib.org/lapack/lug/.

[Apache2] *Apache License, Version 2.0*. Apache Software Foundation. URL: https://www.apache.org/licenses/LICENSE-2.0.

[Bar81] R. J. Bartlett. "Many-body perturbation-theory and Coupled Cluster Theory for electron correlation in molecules". *Annu. Rev. Phys. Chem.* 32 (1981), p. 359. DOI: 10.1146/annurev.pc.32.100181.002043.

[BFP07] S. K. Bogner, R. J. Furnstahl, and R. J. Perry. "Similarity renormalization group for nucleon-nucleon interactions". *Phys. Rev. C* 75 (2007), p. 061001.

[BFS10] S.K. Bogner, R.J. Furnstahl, and A. Schwenk. "From low-momentum interactions to nuclear structure". *Progress in Particle and Nuclear Physics* 65.1 (2010), pp. 94–147. ISSN: 0146-6410. DOI: 10.1016/j.ppnp.2010.03.001. arXiv: 0912.3688.

[Bin14] Sven Binder. "Coupled-Cluster Theory for Nuclear Structure". PhD thesis. Darmstadt: Technische Universität, Feb. 2014. URL: https://tuprints.ulb.tu-darmstadt.de/3946/.

[BL09] Ewald Balcar and Stephen W. Lovesey. *Introduction to the Graphical Theory of Angular Momentum. Case Studies*. 1st ed. Vol. 234. Springer Tracts in Modern Physics. Berlin, Heidelberg: Springer, 2009. ISBN: 978-3-642-03117-5. DOI: 10.1007/978-3-642-03118-2.

[BM67] T.A. Brody and M. Moshinsky. *Tables of transformation brackets for nuclear shell-model calculations*. 2nd ed. Mexico, 1967.

[Bog+14] S. K. Bogner, H. Hergert, J. D. Holt, A. Schwenk, S. Binder, A. Calci, J. Langhammer, and R. Roth. "Nonperturbative Shell-Model Interactions from the In-Medium Similarity Renormalization Group". *Phys. Rev. Lett.* 113 (14 Oct. 2014), p. 142501. DOI: 10.1103/PhysRevLett.113.142501. arXiv: 1402.1407.

[Bra67] Baird H. Brandow. “Linked-Cluster Expansions for the Nuclear Many-Body Problem”. *Rev. Mod. Phys.* 39 (4 Oct. 1967), pp. 771–828. DOI: 10.1103/RevModPhys.39.771.

[Bro65] Charles G Broyden. “A class of methods for solving nonlinear simultaneous equations”. *Math. Comput.* 19.92 (1965), pp. 577–593. DOI: 10.1090/S0025-5718-1965-0198670-6.

[BW88] B A Brown and B H Wildenthal. “Status of the Nuclear Shell Model”. *Annual Review of Nuclear and Particle Science* 38.1 (1988), pp. 29–66. DOI: 10.1146/annurev.ns.38.120188.000333.

[C11] *ISO International Standard ISO/IEC 9899:2011 – Programming languages C*. Standard. Geneva, Switzerland: International Organization for Standardization, Dec. 2011. URL: http://open-std.org/jtc1/sc22/wg14/www/standards.html#9899.

[Cal14] Angelo Calci. “Evolved Chiral Hamiltonians at the Three-Body Level and Beyond”. PhD thesis. Darmstadt: Technische Universität, June 2014. URL: https://tuprints.ulb.tu-darmstadt.de/4069/.

[Cargo] *Cargo: The Rust package manager*. GitHub. URL: https://github.com/rust-lang/cargo.

[ClFmt] *ClangFormat*. LLVM. 2017. URL: https://clang.llvm.org/docs/ClangFormat.html.

[Dar31] C. G. Darwin. “The Diamagnetism of the Free Electron”. *Mathematical Proceedings of the Cambridge Philosophical Society* 27.1 (1931), pp. 86–90. DOI: 10.1017/S0305004100009373.

[DLMF] *NIST Digital Library of Mathematical Functions*. Release 1.0.14 of 2016-12-21. F. W. J. Olver, A. B. Olde Daalhuis, D. W. Lozier, B. I. Schneider, R. F. Boisvert, C. W. Clark, B. R. Miller and B. V. Saunders, eds. 2016. URL: http://dlmf.nist.gov.

[Doxygen] Dimitri van Heesch. *Doxygen*. 2017. URL: http://doxygen.org.

[EHM09] E. Epelbaum, H.-W. Hammer, and Ulf-G. Meißner. “Modern theory of nuclear forces”. *Rev. Mod. Phys.* 81 (4 Dec. 2009), pp. 1773–1825. DOI: 10.1103/RevModPhys.81.1773. arXiv: 0811.1338.

[EM03] D. R. Entem and R. Machleidt. “Accurate charge-dependent nucleon-nucleon potential at fourth order of chiral perturbation theory”. *Phys. Rev. C* 68 (4 Oct. 2003), p. 041001. DOI: 10.1103/PhysRevC.68.041001. arXiv: nucl-th/0304018.

[Eps26] Paul S. Epstein. “The Stark Effect from the Point of View of Schroedinger’s Quantum Theory”. *Phys. Rev.* 28 (4 Oct. 1926), pp. 695–710. DOI: 10.1103/PhysRev.28.695.

[Eva14] Francesco A. Evangelista. “A driven similarity renormalization group approach to quantum many-body problems”. *J. Chem. Phys.* 141.5 (2014), p. 054109. DOI: 10.1063/1.4890660.

[Fey49] R. P. Feynman. “The Theory of Positrons”. *Phys. Rev.* 76 (6 Sept. 1949), pp. 749–759. DOI: 10.1103/PhysRev.76.749.

[FmtRFCs] *Rust code formatting RFCs*. GitHub. URL: https://github.com/rust-lang-nursery/fmt-rfcs.

[FNV] Glenn Fowler, Landon Curt Noll, Kiem-Phong Vo, Donald Eastlake, and Tony Hansen. *The FNV Non-Cryptographic Hash Algorithm*. Internet-Draft draft-eastlake-fnv-14. IETF Secretariat, Dec. 2017. URL: https://tools.ietf.org/html/draft-eastlake-fnv-14.

[Foc28] V. Fock. “Bemerkung zur Quantelung des harmonischen Oszillators im Magnetfeld”. *Z. Phys.* 47.5 (May 1928), pp. 446–448. ISSN: 0044-3328. DOI: 10.1007/BF01390750.

[Foc30] V. Fock. “Näherungsmethode zur Lösung des quantenmechanischen Mehrkörperproblems”. *Z. Phys.* 61.1 (Jan. 1930), pp. 126–148. ISSN: 0044-3328. DOI: 10.1007/BF01340294.

[GG08] Kazushige Goto and Robert A. van de Geijn. “Anatomy of High-performance Matrix Multiplication”. *ACM Trans. Math. Softw.* 34.3 (May 2008), 12:1–12:25. ISSN: 0098-3500. DOI: 10.1145/1356052.1356053. URL: http://www.cs.utexas.edu/~flame/pubs/GotoTOMS_revision.pdf.

[Git] *Git*. 2017. URL: https://git-scm.com.

[Gofmt] *Command gofmt*. Build version go 1.9.2. Google. URL: https://golang.org/cmd/gofmt/.

[Gol57] Jeffrey Goldstone. “Derivation of the Brueckner many-body theory”. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 239.1217 (1957), pp. 267–279. ISSN: 0080-4630. DOI: 10.1098/rspa.1957.0037. URL: http://rspa.royalsocietypublishing.org/content/239/1217/267.

[Gre16] Brendan Gregg. “The Flame Graph”. *Commun. ACM* 59.6 (May 2016), pp. 48–57. ISSN: 0001-0782. DOI: 10.1145/2909476. URL: http://queue.acm.org/detail.cfm?id=2927301.

[Gt16] Torbjörn Granlund and the GMP development team. *GNU MP: The GNU Multiple Precision Arithmetic Library*. 6.1.2. 2016. URL: https://gmplib.org.

[Haddock] Simon Marlow. *Haddock: A Haskell Documentation Tool*. 2017. URL: https://www.haskell.org/haddock/.

[Hal+99] Asger Halkier, Trygve Helgaker, Poul Jørgensen, Wim Klopper, and Jeppe Olsen. “Basis-set convergence of the energy in molecular Hartree–Fock calculations”. *Chem.*

Phys. Lett. 302.5 (1999), pp. 437–446. ISSN: 0009-2614. DOI: 10.1016/S0009-2614(99)00179-7.

[Har28] D. R. Hartree. "The Wave Mechanics of an Atom with a Non-Coulomb Central Field. Part I. Theory and Methods". *Math. Proc. Cambridge* 24.1 (1928), pp. 89–110. DOI: 10.1017/S0305004100011919.

[Her+13] H. Hergert, S. K. Bogner, S. Binder, A. Calci, J. Langhammer, R. Roth, and A. Schwenk. "In-medium similarity renormalization group with chiral two- plus three-nucleon interactions". *Phys. Rev. C* 87 (3 Mar. 2013), p. 034307. DOI: 10.1103/PhysRevC.87.034307. arXiv: 1212.1190.

[Her+16] H. Hergert, S. K. Bogner, T. D. Morris, A. Schwenk, and K. Tsukiyama. "The In-Medium Similarity Renormalization Group: A novel ab initio method for nuclei". *Phys. Rep.* 621 (2016), p. 165. DOI: 10.1016/j.physrep.2015.12.007.

[Her17] H Hergert. "In-medium similarity renormalization group for closed and open-shell nuclei". *Phys. Scr.* 92.2 (2017), p. 023002. DOI: 10.1088/1402-4896/92/2/023002.

[Hg] *Mercurial*. 2017. URL: https://www.mercurial-scm.org.

[HKO95] Morten Hjorth-Jensen, Thomas T. S. Kuo, and Eivind Osnes. "Realistic effective interactions for nuclear systems". *Physics Reports* 261.3 (1995), pp. 125–270. ISSN: 0370-1573. DOI: 10.1016/0370-1573(95)00012-6.

[HLK17] Morten Hjorth-Jensen, Maria Paola Lombardo, and Ubirajara van Kolck. *An Advanced Course in Computational Nuclear Physics: Bridging the Scales from Quarks to Neutron Stars*. Vol. 936. Lecture Notes in Physics. Cham: Springer International Publishing, 2017. ISBN: 978-3-319-53336-0. DOI: 10.1007/978-3-319-53336-0_1. URL: https://github.com/ManyBodyPhysics/LectureNotesPhysics.

[Hoe14] Christian Hoelbling. "Lattice QCD: Concepts, Techniques and Some Results". *Acta Physica Polonica B* 45.12 (2014), p. 2143. DOI: 10.5506/APhysPolB.45.2143. arXiv: 1410.3403.

[Høg13] Jørgen Høgberget. "Quantum Monte-Carlo Studies of Generalized Many-body Systems". MA thesis. University of Oslo, 2013. URL: https://www.duo.uio.no/handle/10852/37167.

[Hug57] N. M. Hugenholtz. "Perturbation theory of large quantum systems". *Physica* 23.1 (1957), pp. 481–532. ISSN: 0031-8914. DOI: 10.1016/S0031-8914(57)92950-6.

[IAH07] N. Ishii, S. Aoki, and T. Hatsuda. "Nuclear Force from Lattice QCD". *Phys. Rev. Lett.* 99 (2 July 2007), p. 022001. DOI: 10.1103/PhysRevLett.99.022001. arXiv: nucl-th/0611096.

[Iri+16] T. Iritani, T. Doi, S. Aoki, S. Gongyo, T. Hatsuda, Y. Ikeda, T. Inoue, N. Ishii, K. Murano, H. Nemura, and K. Sasaki. "Mirage in temporal correlation functions for baryon-baryon interactions in lattice QCD". *Journal of High Energy Physics* 1610.10 (Oct. 2016), p. 101. ISSN: 1029-8479. DOI: 10.1007/JHEP10(2016)101. arXiv: 1607.06371.

[Ish+12] Noriyoshi Ishii, Sinya Aoki, Takumi Doi, Tetsuo Hatsuda, Yoichi Ikeda, Takashi Inoue, Keiko Murano, Hidekatsu Nemura, and Kenji Sasaki. "Hadron–hadron interactions from imaginary-time Nambu–Bethe–Salpeter wave function on the lattice". *Physics Letters B* 712.4 (2012), pp. 437–441. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.04.076. arXiv: 1203.3642.

[Jim+02] Trevor Jim, J. Gregory Morrisett, Dan Grossman, Michael W. Hicks, James Cheney, and Yanling Wang. "Cyclone: A Safe Dialect of C". *USENIX Annual Technical Conference, General Track*. 2002. URL: http://www.cs.umd.edu/~mwh/papers/cyclone-cuj.pdf.

[Jucys] *Jucys*. GitHub. URL: https://github.com/Rufflewind/jucys.

[KBV12] Liguo Kong, Florian A. Bischoff, and Edward F. Valeev. "Explicitly Correlated R12/F12 Methods for Electronic Structure". *Chem. Rev.* 112.1 (2012). PMID: 22176553, pp. 75–107. DOI: 10.1021/cr200204r.

[Keh06] S. Kehrein. *The Flow Equation Approach to Many-Particle Systems*. Springer Tracts in Modern Physics. Springer, 2006. DOI: 10.1007/3-540-34068-8.

[KH84] P. J. Knowles and N. C. Handy. "A new determinant-based full configuration interaction method". *Chemical Physics Letters* 111.4 (1984), pp. 315–321. ISSN: 0009-2614. DOI: 10.1016/0009-2614(84)85513-X.

[KHM07] S. Kvaal, M. Hjorth-Jensen, and H. Møll Nilsen. "Effective interactions, large-scale diagonalization, and one-dimensional quantum dots". *Phys. Rev. B* 76 (2007), p. 085421. DOI: 10.1103/PhysRevB.76.085421.

[KK87] Wim Klopper and Werner Kutzelnigg. "Møller-plesset calculations taking care of the correlation CUSP". *Chem. Phys. Lett.* 134.1 (1987), pp. 17–22. ISSN: 0009-2614. DOI: 10.1016/0009-2614(87)80005-2.

[Kuo+81] T. T. S Kuo, J. Shurpin, K. C. Tam, E. Osnes, and P. J. Ellis. "A simple method for evaluating Goldstone diagrams in an angular momentum coupled representation". *Annals of Physics* 132.2 (1981), pp. 237–276. ISSN: 0003-4916. DOI: 10.1016/0003-4916(81)90068-3.

[Kut85] Werner Kutzelnigg. "r12-Dependent terms in the wave function as closed sums of partial wave amplitudes for large l". *Theor. chim. acta* 68.6 (Dec. 1985), pp. 445–469. ISSN: 1432-2234. DOI: 10.1007/BF00527669.

[Kva08] S. Kvaal. "Open source FCI code for quantum dots and effective interactions". *ArXiv e-prints* (Oct. 2008). arXiv: 0810.2644.

[Kva09] Simen Kvaal. "Harmonic oscillator eigenfunction expansions, quantum dots, and effective interactions". *Phys. Rev. B* 80 (2009), p. 045321. DOI: 10.1103/PhysRevB.80.045321.

[Kva74] V. Kvasnička. "Construction of model hamiltonians in framework of Rayleigh-Schrödinger perturbation theory". *Czech. J. Phys. Sect. B* 24.6 (June 1974), pp. 605–615. ISSN: 1572-9486. DOI: 10.1007/BF01587295.

[Lan14] Joachim Langhammer. "Chiral Three-Nucleon Interactions in Ab-Initio Nuclear Structure and Reactions". PhD thesis. Darmstadt: Technische Universität, Feb. 2014. URL: https://tuprints.ulb.tu-darmstadt.de/3945/.

[Law+79] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. "Basic Linear Algebra Subprograms for Fortran Usage". *ACM Trans. Math. Softw.* 5.3 (Sept. 1979), pp. 308–323. ISSN: 0098-3500. DOI: 10.1145/355841.355847.

[Lep05] G. P. Lepage. "What is Renormalization?" *ArXiv* (June 2005). arXiv: hep-ph/0506330.

[Lev44] Kenneth Levenberg. "A Method for the Solution of Certain Non-Linear Problems in Least Squares". *Q. Appl. Math.* 2.2 (1944), pp. 164–168. ISSN: 0033569X, 15524485. URL: https://www.jstor.org/stable/43633451.

[Lin74] I. Lindgren. "The Rayleigh-Schrodinger perturbation and the linked-diagram theorem for a multi-configurational model space". *J. Phys. Pt. B Atom. M. P.* 7.18 (1974), p. 2441. DOI: 10.1088/0022-3700/7/18/010.

[LM86] Ingvar Lindgren and John Morrison. *Atomic Many-Body Theory*. 2nd ed. Vol. 3. Springer Series on Atomic, Optical, and Plasma Physics. Berlin, Heidelberg: Springer, 1986. ISBN: 9783642616402. DOI: 10.1007/978-3-642-96614-9.

[Loh10] Magnus Pedersen Lohne. "Coupled-cluster studies of quantum dots". MA thesis. University of Oslo, 2010. URL: https://www.duo.uio.no/handle/10852/10966.

[Lutario] Fei Yuan. *Lutario*. GitHub. URL: https://github.com/xrf/lutario.

[Mar63] Donald W. Marquardt. "An Algorithm for Least-Squares Estimation of Nonlinear Parameters". *J. Soc. Ind. Appl. Math.* 11.2 (1963), pp. 431–441. DOI: 10.1137/0111030.

[Maz07a] David A. Mazziotti. "Anti-Hermitian Formulation of the Contracted Schrödinger Theory". *Reduced-Density-Matrix Mechanics: With Application to Many-Electron Atoms and Molecules*. John Wiley& Sons, Inc., 2007, pp. 331–342. ISBN: 9780470106600. DOI: 10.1002/9780470106600.ch12.

[Maz07b] David A. Mazziotti. “Anti-Hermitian part of the contracted Schrödinger equation for the direct calculation of two-electron reduced density matrices”. *Phys. Rev. A* 75 (2 Feb. 2007), p. 022505. DOI: 10.1103/PhysRevA.75.022505.

[ME11] R. Machleidt and D. R. Entem. “Chiral effective field theory and nuclear forces”. *Physics Reports* 503.1 (2011), pp. 1–75. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2011.02.001. arXiv: 1105.2919.

[MGH80] J J Moré, B S Garbow, and K E Hillstrom. *User guide for MINPACK-1.* ANL-80-74. Argonne, IL, Aug. 1980. URL: https://www.mcs.anl.gov/~more/ANL8074b.pdf.

[MIT] *MIT License.* Open Source Initiative. URL: https://opensource.org/licenses/MIT.

[Mor16] Titus Dan Morris. “Systematic improvements of *ab-initio* in-medium similarity renormalization group calculations”. PhD thesis. Michigan State University, 2016. URL: https://publications.nscl.msu.edu/thesis/Morris_2016_387.pdf.

[Mor78] Jorge J. Moré. “The Levenberg-Marquardt algorithm: Implementation and theory”. *Numerical Analysis: Proceedings of the Biennial Conference Held at Dundee, June 28–July 1, 1977.* Ed. by G. A. Watson. Berlin, Heidelberg: Springer, 1978, pp. 105–116. ISBN: 978-3-540-35972-2. DOI: 10.1007/BFb0067700.

[Mos59] Marcos Moshinsky. “Transformation brackets for harmonic oscillator functions”. *Nuclear Physics* 13.1 (1959), pp. 104–116. ISSN: 0029-5582. DOI: 10.1016/0029-5582(59)90143-9.

[MP34] C. Møller and M. S. Plesset. “Note on an Approximation Treatment for Many-Electron Systems”. *Phys. Rev.* 46 (Oct. 1934), pp. 618–622. DOI: 10.1103/PhysRev.46.618.

[MPB15] T. D. Morris, N. M. Parzuchowski, and S. K. Bogner. “Magnus expansion and in-medium similarity renormalization group”. *Phys. Rev. C* 92 (3 Sept. 2015), p. 034331. DOI: 10.1103/PhysRevC.92.034331.

[Nav+09] Petr Navrátil, Sofia Quaglioni, Ionel Stetcu, and Bruce R. Barrett. “Recent developments in no-core shell-model calculations”. *Journal of Physics G: Nuclear and Particle Physics* 36.8 (2009), p. 083101. DOI: 10.1088/0954-3899/36/8/083101. arXiv: 0904.0463.

[Nes55] R. K. Nesbet. “Configuration interaction in orbital theories”. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 230.1182 (1955), pp. 312–321. ISSN: 0080-4630. DOI: 10.1098/rspa.1955.0134. eprint: http://rspa.royalsocietypublishing.org/content/230/1182/312.full.pdf.

[NVB00] P. Navrátil, J. P. Vary, and B. R. Barrett. “Large-basis ab initio no-core shell model and its application to ^{12}C”. *Phys. Rev. C* 62 (5 Oct. 2000), p. 054311. DOI: 10.1103/PhysRevC.62.054311.

[NYC10] Eric Neuscamman, Takeshi Yanai, and Garnet Kin-Lic Chan. "A review of canonical transformation theory". *Int. Rev. Phys. Chem.* 29.2 (2010), pp. 231–271. DOI: 10.1080/01442351003620540.

[ODE] Lawrence Shampine and Marilyn Gordon. *ODE: Ordinary differential equation initial-value problem solver*. URL: http://www.netlib.org/ode/ode.f.

[Ols+88] Jeppe Olsen, Björn O. Roos, Poul Jørgensen, and Hans Jørgen Aa. Jensen. "Determinant based configuration interaction algorithms for complete and restricted configuration interaction spaces". *The Journal of Chemical Physics* 89.4 (1988), pp. 2185–2192. DOI: 10.1063/1.455063.

[Ols13] Veronica K. Berglyd Olsen. "Full Configuration Interaction Simulation of Quantum Dots". MA thesis. University of Oslo, 2013. URL: https://www.duo.uio.no/handle/10852/34217.

[Pan56] Sudhir P. Pandya. "Nucleon-Hole Interaction in jj Coupling". *Phys. Rev.* 103 (4 Aug. 1956), pp. 956–957. DOI: 10.1103/PhysRev.103.956.

[Par17] Nathan Michael Parzuchowski. "Nuclear Spectroscopy with the In-Medium Similarity Renormalization Group". PhD thesis. Michigan State University, 2017. URL: https://d.lib.msu.edu/etd/4630.

[Ped+11] M. Pedersen Lohne, G. Hagen, M. Hjorth-Jensen, S. Kvaal, and F. Pederiva. "*Ab initio* computation of the energies of circular quantum dots". *Phys. Rev. B* 84 (2011), p. 115302.

[PEP8] Guido van Rossum, Barry Warsaw, and Nick Coghlan. *Style Guide for Python Code*. Python Enhancement Proposal 8. July 2001. URL: https://www.python.org/dev/peps/pep-0008/.

[Perf] *perf: Linux profiling with performance counters*. URL: https://perf.wiki.kernel.org.

[PMB17] N. M. Parzuchowski, T. D. Morris, and S. K. Bogner. "Ab initio excited states from the in-medium similarity renormalization group". *Phys. Rev. C* 95 (4 Apr. 2017), p. 044304. DOI: 10.1103/PhysRevC.95.044304. arXiv: 1611.00661.

[Pul80] Péter Pulay. "Convergence acceleration of iterative sequences. the case of scf iteration". *Chem. Phys. Lett.* 73.2 (1980), pp. 393–398. ISSN: 0009-2614. DOI: 10.1016/0009-2614(80)80396-4.

[Pul82] P. Pulay. "Improved SCF convergence acceleration". *J. Comput. Chem.* 3.4 (1982), pp. 556–560. ISSN: 1096-987X. DOI: 10.1002/jcc.540030413.

[Rac42] Giulio Racah. "Theory of Complex Spectra. II". *Phys. Rev.* 62 (9-10 Nov. 1942), pp. 438–462. DOI: 10.1103/PhysRev.62.438.

[Reg58] T. Regge. “Symmetry properties of Clebsch-Gordon’s coefficients”. *Il Nuovo Cimento* 10.3 (Nov. 1958), pp. 544–545. ISSN: 1827-6121. DOI: 10.1007/BF02859841.

[Rei13] Sarah Reimann. “Quantum-mechanical systems in traps and Similarity Renormalization Group theory”. MA thesis. University of Oslo, 2013. URL: https://www.duo.uio.no/handle/10852/37161.

[Rot+12] Robert Roth, Sven Binder, Klaus Vobig, Angelo Calci, Joachim Langhammer, and Petr Navrátil. “Medium-Mass Nuclei with Normal-Ordered Chiral NN+3N Interactions”. *Phys. Rev. Lett.* 109 (2012), p. 052501.

[Rust] *Rust*. URL: https://www.rust-lang.org.

[RustBook] The Rust Project Developers. *The Rust Programming Language*. Ed. by Steve Klabnik. 2nd ed. 2017. URL: https://github.com/rust-lang/book.

[Rustfmt] *rustfmt*. GitHub. URL: https://github.com/rust-lang-nursery/rustfmt.

[RY04] J. Rasch and A. C. H. Yu. “Efficient Storage Scheme for Precalculated Wigner 3j, 6j and Gaunt Coefficients”. *SIAM Journal on Scientific Computing* 25.4 (2004), pp. 1416–1428. DOI: 10.1137/S1064827503422932.

[SB09] I. Shavitt and R. J. Bartlett. *Many-Body Methods in Chemistry and Physics: MBPT and Coupled-Cluster Theory*. Cambridge Molecular Science. Cambridge University Press, 2009.

[SG13] James J. Shepherd and Andreas Grüneis. “Many-Body Quantum Chemistry for the Electron Gas: Convergent Perturbative Theories”. *Phys. Rev. Lett.* 110 (22 May 2013), p. 226401. DOI: 10.1103/PhysRevLett.110.226401. arXiv: 1310.6059.

[SG75] L. F. Shampine and M. K. Gordon. *Computer Solution of Ordinary Differential Equations: The Initial Value Problem*. Freeman, 1975.

[SgOde] Lawrence Shampine, Marilyn Gordon, and Fei Yuan. *Parallelizable Shampine-Gordon ODE solver*. GitHub. URL: https://github.com/xrf/sg-ode.

[Sphinx] Georg Brandl. *Sphinx: Python Documentation Generator*. 2018. URL: http://www.sphinx-doc.org.

[Str+16] S. R. Stroberg, H. Hergert, J. D. Holt, S. K. Bogner, and A. Schwenk. “Ground and excited states of doubly open-shell nuclei from ab initio valence-space Hamiltonians”. *Phys. Rev. C* 93 (5 May 2016), p. 051301. DOI: 10.1103/PhysRevC.93.051301. arXiv: 1511.02802.

[Str+17] S. R. Stroberg, A. Calci, H. Hergert, J. D. Holt, S. K. Bogner, R. Roth, and A. Schwenk. "Nucleus-Dependent Valence-Space Approach to Nuclear Structure". *Phys. Rev. Lett.* 118 (3 Jan. 2017), p. 032502. DOI: 10.1103/PhysRevLett.118.032502. arXiv: 1607.03229.

[Suh07] Jouni Suhonen. *From Nucleons to Nucleus. Concepts of Microscopic Nuclear Theory*. 1st ed. Theoretical and Mathematical Physics. Berlin, Heidelberg: Springer, 2007. ISBN: 978-3-540-48861-3. DOI: 10.1007/978-3-540-48861-3.

[Tal52] Igal Talmi. "Nuclear Spectroscopy with Harmonic Oscillator Wave-Functions". *Helvetica Physica Acta* 25.3 (1952), pp. 185–234. ISSN: 0018-0238. DOI: 10.5169/seals-112307.

[TBS11] K. Tsukiyama, S. K. Bogner, and A. Schwenk. "In-Medium Similarity Renormalization Group For Nuclei". *Phys. Rev. Lett.* 106 (2011), p. 222502.

[TBS12] K. Tsukiyama, S. K. Bogner, and A. Schwenk. "In-medium similarity renormalization group for open-shell nuclei". *Phys. Rev. C* 85 (2012), p. 061304. DOI: 10.1103/PhysRevC.85.061304.

[Tho10] M. Thoennessen. "Plans for the Facility for Rare Isotope Beams". *Nuclear Physics A* 834.1 (2010). The 10th International Conference on Nucleus-Nucleus Collisions (NN2009), pp. 688c–693c. ISSN: 0375-9474. DOI: 10.1016/j.nuclphysa.2010.01.125.

[Uka15] Akira Ukawa. "Kenneth Wilson and Lattice QCD". *Journal of Statistical Physics* 160.5 (Sept. 2015), pp. 1081–1124. ISSN: 1572-9613. DOI: 10.1007/s10955-015-1197-x. arXiv: 1501.04215.

[Valgrind] Valgrind Developers. *Valgrind*. 2017. URL: http://valgrind.org.

[Wan+13] Qian Wang, Xianyi Zhang, Yunquan Zhang, and Qing Yi. "AUGEM: Automatically Generate High Performance Dense Linear Algebra Kernels on x86 CPUs". *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. SC '13. Denver, Colorado: ACM, 2013, 25:1–25:12. ISBN: 978-1-4503-2378-9. DOI: 10.1145/2503210.2503219. URL: https://xianyi.github.io/paper/augem_SC13.pdf.

[Weg01] Franz J. Wegner. "Flow equations for Hamiltonians". *Phys. Rep.* 348 (2001), pp. 77–89. DOI: 10.1016/S0370-1573(00)00136-8.

[Wei79] Steven Weinberg. "Phenomenological Lagrangians". *Physica A: Statistical Mechanics and its Applications* 96.1 (1979), pp. 327–340. ISSN: 0378-4371. DOI: 10.1016/0378-4371(79)90223-1.

[Wei98] Liqiang Wei. "New formula for 9-j symbols and their direct calculation". *Computers in Physics* 12.6 (1998), pp. 632–634. DOI: 10.1063/1.168745.

[Wei99] Liqiang Wei. “Unified approach for exact calculation of angular momentum coupling and recoupling coefficients”. *Computer Physics Communications* 120.2 (1999), pp. 222–230. ISSN: 0010-4655. DOI: 10.1016/S0010-4655(99)00232-5.

[Whi02] S. R. White. “A numerical canonical transformation approach to quantum many body problems”. *J. Chem. Phys.* 117 (2002), p. 7472.

[Wic50] G. C. Wick. “The Evaluation of the Collision Matrix”. *Phys. Rev.* 80 (2 Oct. 1950), pp. 268–272. DOI: 10.1103/PhysRev.80.268.

[Wig93] E. P. Wigner. “On the Matrices Which Reduce the Kronecker Products of Representations of S. R. Groups”. *The Collected Works of Eugene Paul Wigner: Part A: The Scientific Papers*. Ed. by Arthur S. Wightman. Berlin, Heidelberg: Springer, 1993, pp. 608–654. ISBN: 978-3-662-02781-3. DOI: 10.1007/978-3-662-02781-3_42.

[Wil83] Kenneth G. Wilson. “The renormalization group and critical phenomena”. *Rev. Mod. Phys.* 55 (3 July 1983), pp. 583–600. DOI: 10.1103/RevModPhys.55.583. eprint: https://www.nobelprize.org/nobel_prizes/physics/laureates/1982/wilson-lecture.pdf.

[WP06] Paul E. S. Wormer and Josef Paldus. “Angular Momentum Diagrams”. Ed. by J. R. Sabin and E. Brändas. Vol. 51. Advances in Quantum Chemistry Supplement C. Academic Press, 2006, pp. 59–123. DOI: 10.1016/S0065-3276(06)51002-0. URL: http://www.theochem.ru.nl/files/dbase/aqc-51-59-2006.pdf.

[WSH] *wigner-symbols*. 2015. URL: https://hackage.haskell.org/package/wigner-symbols.

[WSR] *wigner-symbols*. 2017. URL: https://crates.io/crates/wigner-symbols.

[XQS17] Zhang Xianyi, Wang Qian, and Werner Saar. *OpenBLAS*. 2017. URL: http://www.openblas.net.

[YLV62] A. P. Yutsis, I. B. Levinson, and V. V. Vanagas. *Mathematical Apparatus of The Theory of Angular Momentum*. Trans. by A. Sen and R. N. Sen. Translated from Russian. Jerusalem: Israel Program for Scientific Translations, 1962. URL: https://archive.org/details/nasa_techdoc_19630001624.

[Yua+] Fei Yuan, Samuel J. Novario, Nathan M. Parzuchowski, S. K. Bogner, and Morten Hjorth-Jensen. “Addition and removal energies of nuclei”. in preparation.

[Yua+17] Fei Yuan, Samuel J. Novario, Nathan M. Parzuchowski, Sarah Reimann, S. K. Bogner, and Morten Hjorth-Jensen. “Addition and removal energies of circular quantum dots”. *The Journal of Chemical Physics* 147.16 (2017), p. 164109. DOI: 10.1063/1.4995615. eprint: 1707.00229. URL: https://doi.org/10.1063/1.4995615.