Novel Algorithms for Computing Nuclear Correlation Functions in Lattice Quantum Chromodynamics

Nabil Humphrey

The Special Research Centre for the
Subatomic Structure of Matter (CSSM)

School of Physical Sciences

University of Adelaide

February 2022

This dissertation is submitted for the degree of
Master of Philosophy

Abstract

Techniques to attain numerical solutions of Quantum Chromodynamics have developed to the point of beginning to connect aspects of nuclear physics to the underlying degrees of freedom of the Standard Model. There remain deep physical and numerical challenges, including a proliferation of possibilities for interpolating operators that couple to low-energy states, factorial numerical resource scaling, poor signal-to-noise scaling, and potentially dominating floating-point precision errors. The focus of this work is the computational resource scaling associated with numerically evaluating the correlation function for an interpolating operator possessing the quantum numbers of a multi-baryon system in the context of lattice QCD. The naïve computational cost required to compute nuclear correlation functions grows factorially in the number of quarks, however this work develops a set of novel approaches that reduce this cost by exploiting inherent permutation symmetry. A selection of benchmarks demonstrate that the new methods can offer between one and two orders of magnitude in reduced calculation time (excluding hadron block expression evaluation) for correlation functions of light nuclei when compared against the hadron block method alone.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Nabil Humphrey

Acknowledgements

As a result of the preparation of this thesis, I owe many debts of gratitude. Chief among them is to my supervisors, James Zanotti and Ross Young, who have nurtured and inspired significant intellectual growth. I also owe thanks to Phiala Shanahan, Will Detmold, and Artur Avkhadiev for our many helpful and inspiring discussions. My wide support network of friends and family have enabled all the work presented herein.

Preface

The strong interaction plays a central role in high energy physics, and understanding its qualitatively varying behaviour over a vast set of regimes remains a key challenge. Some regimes, such as low-energy nuclear few-body states, are well-described by effective field theories, but the study of more exotic states is likely to require access to the fundamental degrees of freedom of the strong interaction. The principal aim of this work is to extend the feasible range of study of nuclear states via the currently accepted fundamental theory — Quantum Chromodynamics (QCD).

Lattice QCD is a highly successful technique to numerically study the nonperturbative regime of QCD using high-performance computing. A typical lattice QCD calculation proceeds in four stages: gauge field configuration generation, propagator calculation, correlation function calculation, and finally the extraction of physical observables such as the low-lying mass spectrum. The past few decades have seen a highly productive programme of work that has dramatically improved the numerical performance of each of these stages.

The focus of this work is the numerical performance of correlation function calculations, which can dominate the computational budget in the case of many-hadron states due in part to the factorial scaling of the required quark Wick contractions in general. Other numerical challenges for many-hadron states, such as a proliferation of possibilities for interpolating operators that couple to low-energy states, and poor signal-to-noise scaling, are directly impacted by the numerical performance of correlation function calculations.

The primary contributions arising from this thesis are the proposal and evaluation of two novel algorithms for the calculation of nuclear correlation functions, together with a robust framework for the efficient numerical evaluation of tensor expressions with permutation symmetry that arise in many other disciplines in physical science.

Contents

Chapter 1.

Introduction

A key mathematical object in the execution of numerical quantum field theory calculations is the *correlation function*, which is constructed from time-ordered products of field operators to reflect the quantum numbers of a state of interest. After providing a fleeting snapshot of QCD and lattice QFT, Chapter 2 provides an introduction to the construction of correlation functions for multi-hadron states through the application of Wick's Theorem to products of multi-hadron interpolating operators. It will be shown in Chapter 2 that the direct approach to calculating correlation functions scales poorly to the case of nuclear states of many quarks. A review of the algorithmic progress to date will be presented in Chapter 3, which also includes numerical benchmarks that form the foundation for the novel methods developed in the subsequent chapters.

The Standard Model rests upon the representation theory of Lie groups to encode continuous physical symmetries. Permutation group theory is of a sibling mathematical lineage [1], stemming from the 'original' group theory of Jordan [2], Cayley [3], and Galois [4] in the 19^{th} century. Finite permutation group theory is concerned with the groups $Sym(\Omega)$, with elements that permute a finite set Ω and group product given by composition. It is common to select $\Omega = \{1, \ldots, n\}$ for an integer n, in which case $Sym(\{1, \ldots, n\})$ is denoted by the *finite symmetric group* S_n. The finite symmetric group will play a recurring role throughout this thesis. In particular, study of the group S_n forms the basis for *tensor expression canonicalisation* via *graph canonicalisation*, as explored in Chapter 4. Tensor expression canonicalisation is applied directly to the tensor expressions of nuclear correlators to remove redundant isomorphs and hence reduce the computational work necessary to numerically evaluate the correlators. Tensor expression canonicalisation is also a key process in the *tensor e-graph* method developed in Chapter 6.

The tensor expressions of nuclear correlators exhibit a high degree of permutation symmetry, which can be exploited via structures developed in Chapter 5 referred to as *factor trees*. The algorithm developed to evaluate factor trees is engineered so as to optimise cache performance on modern CPU architectures, resulting in significantly faster numerical evaluation of certain choices of nuclear correlation functions. The construction of factor trees requires CPU time and memory that scales exponentially in quark number, limiting the range of applicability and providing motivation for the development of *tensor e-graphs* in Chapter 6.

E-graphs were originally developed in the 1970s to power automated theorem provers [5] by efficiently exploring the space of all possible re-writes of expressions in order to extract the 'optimal' expression via the minimisation of a cost function. Tensor e-graphs are developed in Chapter 6 as a novel e-graph variant that extracts tensor sub-expressions that are maximally re-used throughout the numerical evaluation of a large set of tensor expressions. An exploration of the heuristic parameter space will show that although speed-ups for nuclear correlators are attained from tensor e-graph based optimisation, future work will be necessary to fine-tune the process.

Chapter 7 will present a potential pathway to extend a selection of the techniques developed in Chapter 4 to adiabatic quantum devices. It will be shown that the current maturation of quantum processing hardware is too primitive to execute nuclear correlator calculations that are more performant than high-performance classical computers. However, there is a productive programme of theoretical work that seeks to determine the usefulness (or otherwise) of quantum devices for nuclear correlator calculations, and Chapter 7 endeavours to begin that process.

Chapter 2.

Nuclear Correlation Functions

This chapter will provide a brief snapshot of Quantum Chromodynamics (§ 2.1) and lattice Quantum Field Theory (§ 2.2). The unfamiliar reader is encouraged to consult Refs. [6, 7] for a rigorous and comprehensive account of these broad and intricate subjects. § 2.3 will discuss the construction of the multi-hadron interpolating operators used throughout this thesis. § 2.4 will give a rigorous account of Wick's Theorem, which plays an essential role in the construction of the nuclear correlation functions in § 2.5. Finally § 2.6 will provide a concise selection of the approaches used to extract physical observables from computed correlation functions.

2.1. The Strong Interaction and Quantum Chromodynamics

The formulation of the Quantum Chromodynamics (QCD) lagrangian with a Yang-Mills theory of colour octet gluons by Fritzsch, Gell-Mann, and Leutwyler [8] in 1973 marks an immense achievement that accumulated centuries of scientific triumphs to assert the fundamental degrees of freedom of the strong interaction. The strong interaction, for example, confines fundamental particles known as quarks into hadrons that are further bound to form atomic nuclei amongst more exotic states. There are several notable achievements along the journey to the QCD lagrangian, such as the discovery of the strangeness quantum number through experimental evidence of the K^0 meson [9] and Λ baryon [10], the quantum mechanical description of the strong force by Yukawa [11], the quark model of Gell-Mann [12] and Zweig [13, 14], as well as the proposal of colour charge [15, 16].

The particle content of QCD consists of *quarks*, which are spin-$\frac{1}{2}$ Dirac spinors, and *gluons*, which are spin-1 gauge bosons. The 'core' QCD lagrangian (neglecting CP-symmetry violating and gauge fixing terms for simplicity) is given by [17]:

$$\mathcal{L}_{QCD} = \sum_q \overline{\psi}_q \left(i\gamma^\mu D_\mu - m_q \right) \psi_q - \frac{1}{4} G^a_{\mu\nu} G^{a,\mu\nu}, \tag{2.1}$$

where $\{\gamma^\mu\}$ are the Dirac matrices, m_q is the quark mass, and each of the other symbols are described as follows. $\psi^c_{q,\alpha}(x)$ are quark spinors at spacetime point x with colour index $c \in \{1,2,3\}$, flavour $q \in \{u,d,s,c,b,t\}$, and spinor index $\alpha \in \{1,2,3,4\}$ that transform under the fundamental representation of $SU(3)_c$ as:

$$\psi(x) \xrightarrow{\Omega(x) \in SU(3)_c} \psi'(x) = e^{-ig\theta_a(x)T_a}\psi(x), \tag{2.2}$$

where g is the strong coupling, $\theta_a(x)$ preserves local gauge invariance, and $\{T_a\}_{a=1}^8$ are the generators of $SU(3)_c$. The gluon field strength tensor is given by:

$$G^a_{\mu\nu} = \partial_\mu A^a_\nu - \partial_\nu A^a_\mu - g f_{abc} A^b_\mu A^c_\nu, \tag{2.3}$$

where f_{abc} are the structure constants of $SU(3)_c$, and $A_\mu =: T_a A^a_\mu$ is the gluon field that transforms under the adjoint representation of $SU(3)_c$ as:

$$A_\mu(x) \xrightarrow{\Omega(x) \in SU(3)_c} A'_\mu(x) = \Omega(x) A_\mu(x) \Omega^\dagger(x) + \frac{i}{g} \left(\partial_\mu \Omega(x) \right) \Omega^\dagger(x), \tag{2.4}$$

so that the covariant derivative is $D_\mu = \partial_\mu + igT_a A^a_\mu$.

Two characteristic properties of QCD are *asymptotic freedom* and *colour confinement*. Asymptotic freedom was discovered by Gross, Wilczek [18], and Politzer [19] in 1973, and describes the property whereby the strong coupling g is weak at short distances. Colour confinement is the property that the strong coupling is large at large distances, and necessitates that quarks and gluons are 'confined' to colour-singlet composite objects such as hadrons. One consequence of colour confinement is that there are large swaths of parameter space (where $g \sim \mathcal{O}(1)$) that are inaccessible to perturbation theory. A solution is to use a non-perturbative technique such as lattice Quantum Field Theory, as outlined in the next section.

2.2. Lattice Quantum Field Theory

Lattice Quantum Field Theory (LQFT) is a powerful non-perturbative technique, as pioneered by Wilson [20], to study field theories by harnessing large-scale computing resources. Consider the path integral formalism [21] for the vacuum expectation value of an operator $\hat{O}$ (using QCD as an example, but the generalisation is natural):

$$\langle \hat{O} \rangle = \frac{1}{\mathcal{Z}} \int \mathcal{D}\psi \mathcal{D}\overline{\psi} \mathcal{D}A_\mu \hat{O} e^{iS_{QCD}}, \tag{2.5}$$

where $\mathcal{Z} = \int \mathcal{D}\psi \mathcal{D}\overline{\psi} \mathcal{D}A_\mu e^{iS_{QCD}}$ is the *partition function* and $S_{QCD} = \int d^4x \mathcal{L}_{QCD}$ is the *action*. In order to resolve the highly oscillatory behaviour of $e^{iS_{QCD}}$ for real S_{QCD}, one performs a Wick rotation [22] from $(1+3)$-dimensional Minkowski space to 4-dimensional Euclidean space, so that:

$$S_{QCD} \xrightarrow[rotation]{Wick} S^{(E)}_{QCD} = -iS_{QCD}, \tag{2.6}$$

and therefore:

$$\langle \hat{O} \rangle = \frac{1}{\mathcal{Z}^{(E)}} \int \mathcal{D}\psi \mathcal{D}\overline{\psi} \mathcal{D}A_\mu \hat{O} e^{-S^{(E)}_{QCD}}, \tag{2.7}$$

where $\mathcal{Z}^{(E)} = \int \mathcal{D}\psi \mathcal{D}\overline{\psi} \mathcal{D}A_\mu e^{-S^{(E)}_{QCD}}$. Eqn. 2.7 permits evaluation via importance sampling by identifying $e^{-S^{(E)}_{QCD}}$ as an (un-normalised) probability distribution, provided that $S^{(E)}_{QCD}$ is real and well-behaved (see Refs. [23–26] for a selection of known issues and potential solutions). In order to evaluate Eqn. 2.7 numerically and to introduce an ultraviolet regular, one discretises the field theory using a hypercubic lattice with lattice spacing a, such as the symmetric and isotropic lattice given by:

$$\Lambda = \{(x_1, x_2, x_3, t) \mid 0 \leq x_1, x_2, x_3 < L,\ 0 \leq t < T\}, \tag{2.8}$$

for spatial volume $V = L^3$, and temporal extent T.

After imposing a discrete hypercubic lattice such as in Eqn. 2.8, it remains to discretise the action $S^{(E)}_{QCD}$. This is a topic of rigorous debate that is too broad to relay here, but there are a number of choices that each carry benefits and drawbacks [27]. The simplest (and least effective) approach is known as *naive discretisation*, whereby one applies a finite difference approximation to the derivative together with a finite approximation to the integral, such as for the quark terms of the action with a single

quark field $\psi(x)$:

$$\hat{S}^{(E)}_{QCD} = \int d^4x \overline{\psi}(x) \left(\gamma_\mu \partial_\mu + m\right) \psi(x)$$
$$\xrightarrow[discretisation]{naive} a^4 \sum_{x\in\Lambda} \overline{\psi}(x) \left[\gamma_\mu \left(\frac{\psi(x+\hat{\mu}) - \psi(x-\hat{\mu})}{2a}\right) + m\psi(x)\right], \tag{2.9}$$

where $\hat{\mu}$ is the unit vector for $\mu \in \{1,2,3,4\}$. Discretisation according to Eqn. 2.9 introduces significant artifacts including 15 unphysical poles (known as *doublers*) in the continuum limit. Many of the alternate discretisation methods in effect add terms to Eqn. 2.9 that have advantageous behaviour in the limit $a \to 0$.

For the purposes of the remainder of this thesis, an LQFT calculation proceeds in four stages:

(1) **Gauge field configuration generation**, by importance-sampling representative configurations of the gauge field.

(2) **(Quark) propagator calculation** (as discussed in § 2.4) through inversion of the Dirac operator using the calculated gauge field configurations.

(3) **Correlation function calculation** (as discussed in § 2.5, as well as the remainder of the thesis) using the calculated propagators.

(4) **Extraction of physical observables** (as discussed in § 2.6) through physical point extrapolation.

2.3. Hadronic Interpolating Operators

In order to construct *nuclear correlation functions* $\langle \mathcal{O}' \overline{\mathcal{O}} \rangle$, one must begin by constructing quark-level *interpolating operators* $\mathcal{O}$ with the desired set of quantum numbers. The primary focus here is multi-baryon states of A nucleons, but the techniques developed work equivalently for other nuclear states. A convenient approach for multi-baryon systems is to combine local baryon interpolating operators by tying together uncontracted spinor indices and projecting to achieve the required state. Make a notation change $\psi^c_{q,\alpha} \to q^c_\alpha$ for quark fields $q \in \{u,d,s,c,b,t\}$, and consider the following set of nucleon operators (examples of $\mathcal{O}$) where u,d pairs have spinor indices contracted to

form spin-0 *diquarks*:

$$
\begin{aligned}
p^\alpha(x) &= \epsilon_{abc}(u_a^T(x)(C\gamma_5)d_b(x))u_c^\alpha(x) && \text{(proton)}\\
n^\alpha(x) &= \epsilon_{abc}(d_a^T(x)(C\gamma_5)u_b(x))d_c^\alpha(x) && \text{(neutron)}\\
p_\pm^\alpha(x) &= \epsilon_{abc}(u_a^T(x)(C\gamma_5 P_\pm)d_b(x))u_c^\alpha(x) && \text{(non-relativistic proton)}\\
n_\pm^\alpha(x) &= \epsilon_{abc}(d_a^T(x)(C\gamma_5 P_\pm)u_b(x))d_c^\alpha(x), && \text{(non-relativistic neutron)}
\end{aligned}
\tag{2.10}
$$

where C is the charge conjugation operator and $P_\pm = \frac{1}{2}(1 \pm \gamma_4)$ projects a 4-component spinor onto a 2-component spinor in either the upper two $(+)$ or lower two $(-)$ components of the 4-component spinor. From single baryon operators, construct a selection of multi-baryon operators (examples of $\mathcal{O}$):

$$
\begin{aligned}
D_I(x) &= n^T(x)(C\gamma_3)p(x) && \text{(deuteron I)}\\
D_{II}(x) &= \frac{1}{\sqrt{2}}\left[n^T(x)(C\gamma_3)p(x) - p^T(x)(C\gamma_3)n(x)\right] && \text{(deuteron II)}\\
\widetilde{D}_I(x) &= n^T(x)(C\gamma_5)p(x) && \text{(dinucleon I)}\\
\widetilde{D}_{II}(x) &= \frac{1}{\sqrt{2}}\left[n^T(x)(C\gamma_5)p(x) - p^T(x)(C\gamma_5)n(x)\right] && \text{(dinucleon II)}\\
{}^3He_I^j(x) &= p_-^T(x)(C\gamma_5)n_+(x)p_+^j(x) && \text{(helium-3 I)}\\
{}^3He_{II}^j(x) &= \frac{1}{\sqrt{6}}\Big[p_-^T(x)(C\gamma_5)n_+(x)p_+^j(x) - p_+^T(x)(C\gamma_5)n_+(x)p_-^j(x)\\
&\qquad +n_+^T(x)(C\gamma_5)p_+(x)p_-^j(x) - n_+^T(x)(C\gamma_5)p_-(x)p_+^j(x)\\
&\qquad +p_+^T(x)(C\gamma_5)p_-(x)n_+^j(x) - p_-^T(x)(C\gamma_5)p_+(x)n_+^j(x)\Big] && \text{(helium-3 II [28])}\\
{}^4He_I(x) &= p_+^T(C\gamma_5)n_-(x)p_+^T(x)(C\gamma_5)n_-(x) && \text{(helium-4 I)}\\
{}^4He_{II}(x) &= \frac{1}{4\sqrt{6}}\left[\widetilde{\chi}\eta - \chi\widetilde{\eta}\right], && \text{(helium-4 II [28])}
\end{aligned}
\tag{2.11}
$$

where:

$$\chi = [+-+-] + [-+-+] - [+--+] - [-++-]$$
$$\widetilde{\chi} = [+-+-] + [-+-+] + [+--+] + [-++-] - 2[++--] - 2[--++]$$
$$\eta = [pnpn] + [npnp] - [pnnp] - [nppn]$$
$$\widetilde{\eta} = [pnpn] + [npnp] + [pnnp] + [nppn] - 2[ppnn] - 2[nnpp],$$

under a product structure such that $[+-+-][pnpn] = {}^4He_I(x)$. Note that $D_I(x)$, ${}^3He_I(x)$, and ${}^4He_I(x)$ are chosen to not project spin/isospin quantum numbers so as to provide additional benchmark analysis opportunities in Chapter 4 – Chapter 6. $D_I(x)$, $D_{II}(x)$, $\widetilde{D}_I(x)$, and $\widetilde{D}_{II}(x)$ are relativistic operators (using all four spinor components), whereas the remaining ${}^3He^j_I(x)$, ${}^3He^j_{II}(x)$, ${}^4He_I(x)$, and ${}^4He_{II}(x)$ are non-relativistic.

2.4. Wick's Theorem

The following will report Wick's Theorem as applied to quark fields, and walk through an example of its application in order to make an observation that will lead to the more precise version of Wick's Theorem used for the remainder of the thesis.

Theorem (Wick's Theorem [29])**.** Given a collection of quark field operators $\{q(a_i)\}$, $\{\overline{q}(a'_i)\}$ where $i = 1, \ldots, n_q$, the time-ordered product may be evaluated as:

$$T\left\{q(a_1)\ldots q(a_{n_q})\overline{q}(a'_1)\ldots\overline{q}(a'_{n_q})\right\}$$
$$= :q(a_1)\ldots q(a_{n_q})\overline{q}(a'_1)\ldots\overline{q}(a'_{n_q}) + \text{all possible contractions}:, \tag{2.12}$$

where $:\cdots:$ denotes normal-ordering, and a_i (a'_i) combine flavour f_i (f'_i), colour index c_i (c'_i), spinor index α_i (α'_i), and spacetime point x_i (x'_i).

In order to give a precise meaning to Wick's Theorem, it remains to define "all possible" and "contractions". A *contraction* is the combination of adjacent quark field operators $q, \overline{q}$ to form the quark propagator $S(a_i; a'_j) := \overline{q(a_i)\overline{q}}\,(a'_j)$. Note that $\overline{q(a_i)q}\,(a_j) = 0 = \overline{\overline{q}(a_i)\overline{q}}\,(a_j)$, so will be excluded from the notation of "all possible contractions" for the purposes of Wick's Theorem. Given that QCD interactions conserve flavour quantum numbers, the contraction of distinct flavour quark field operators will vanish, and hence it will be useful to introduce the notation

$S^{f,cc'}_{\alpha\alpha'}(x,x') := \overline{q^{f,c}_{\alpha}(x)\bar{q}}^{f,c'}_{\alpha'}(x')$ for explicit flavour f, colour indices c, c', spinor indices α, α', and spacetime points x, x'. The results reported in Chapter 4 – Chapter 6 will use the isospin symmetric limit where $S^{u,cc'}_{\alpha\alpha'}(x,x') = S^{d,cc'}_{\alpha\alpha'}(x,x')$, but this limit can only be taken after Wick's Theorem has been applied.

In Wick's Theorem, "all possible contractions" is defined as:

$$\sum_{k=1}^{n_q} \{ \text{ all contractions of } k \text{ pairs of quark fields } \}, \tag{2.13}$$

which necessitates the contraction of non-adjacent quark operators. For example, suppose one needs to consider the contraction:

$$\underline{q_1 q_2 \bar{q}_1 \bar{q}}_2, \tag{2.14}$$

where the shorthand $q_i \equiv q(a_i)$ and $\bar{q}_i \equiv \bar{q}(a'_i)$ has been introduced for convenience. In order to re-arrange Eqn. 2.14 such that q_1 and $\bar{q}_2$ are adjacent, consider the anti-commutation relations for $i \neq j$:

$$\{q_i, q_j\} = 0 = q_i q_j + q_j q_i \tag{2.15}$$
$$\{\bar{q}_i, \bar{q}_j\} = 0 = \bar{q}_i \bar{q}_j + \bar{q}_j \bar{q}_i \tag{2.16}$$
$$\{q_i, \bar{q}_j\} = 0 = q_i \bar{q}_j + \bar{q}_j q_i. \tag{2.17}$$

Applying $q_1 q_2 = -q_2 q_1$ from Eqn. 2.15 and $\bar{q}_1 \bar{q}_2 = -\bar{q}_2 \bar{q}_1$ from Eqn. 2.16 successively to Eqn. 2.14 allows the evaluation:

$$\begin{aligned} \underline{q_1 q_2 \bar{q}_1 \bar{q}}_2 &= -q_2 \underline{q_1 \bar{q}_1 \bar{q}}_2 \\ &= (-)^2 q_2 \underline{q_1 \bar{q}}_2 \bar{q}_1 \\ &= S_{1,2}\, q_2 \bar{q}_1, \end{aligned} \tag{2.18}$$

where the shorthand $S_{i,j} := S(a_i; a'_j) = \underline{q_i \bar{q}}_j$ has been introduced.

With the machinery to apply Wick's Theorem, consider for example the case where $n_q = 2$:

$$\begin{aligned}
&T\left\{q_1 q_2 \bar{q}_1 \bar{q}_2\right\} \\
&\quad = :q_1 q_2 \bar{q}_1 \bar{q}_2 + \sum_{k=1}^{2} \{ \text{ all contractions of } k \text{ pairs of quark fields } \} : \\
&\quad = :q_1 q_2 \bar{q}_1 \bar{q}_2 + \underbrace{q_1 q_2 \bar{q}}_{}{}_1 \bar{q}_2 + q_1 \underbrace{q_2 \bar{q}_1 \bar{q}}_{}{}_2 + \underbrace{q_1 q_2 \bar{q}_1 \bar{q}}_{}{}_2 + q_1 \underbrace{q_2 \bar{q}}_{}{}_1 \bar{q}_2 \\
&\qquad + \underbrace{q_1 \overbrace{q_2 \bar{q}_1}^{} \bar{q}}_{}{}_2 + \underbrace{q_1 \overbrace{q_2 \bar{q}}^{} \bar{q}}_{}{}_2 : \\
&\quad = :q_1 q_2 \bar{q}_1 \bar{q}_2 : - S_{1,1} : q_2 \bar{q}_2 : - S_{2,2} : q_1 \bar{q}_1 : + S_{1,2} : q_2 \bar{q}_1 : + S_{2,1} : q_1 \bar{q}_2 : \\
&\qquad - S_{1,1} S_{2,2} + S_{1,2} S_{2,1},
\end{aligned} \tag{2.19}$$

where the final line has used the linearity of normal-ordering. To compute correlators, one is not actually interested in the time-ordered product $T\left\{q_1 q_2 \bar{q}_1 \bar{q}_2\right\}$, but the vacuum matrix element of the time-ordered product, $\langle 0| T\left\{q_1 q_2 \bar{q}_1 \bar{q}_2\right\} |0\rangle$. Vacuum matrix elements of normal-ordered products of free fields vanish [29] since all annihilation operators are pushed to the right, resulting in for example:

$$\langle 0| : q_1 q_2 \bar{q}_1 \bar{q}_2 : |0\rangle = 0 = \langle 0| : q_2 \bar{q}_2 : |0\rangle . \tag{2.20}$$

Applying these identities to Eqn. 2.19 allows the construction of the vacuum matrix element of the time-ordered product for $n_q = 2$:

$$\langle 0| T\left\{q_1 q_2 \bar{q}_1 \bar{q}_2\right\} |0\rangle = S_{1,2} S_{2,1} - S_{1,1} S_{2,2}. \tag{2.21}$$

This observation generalises to all vacuum matrix elements of time-ordered products of quark operators so that the only set of terms in Eqn. 2.13 that contribute are where $k = n_q$, allowing for a precise formulation of Wick's Theorem that will be used henceforth.

Theorem (Wick's Theorem for Vacuum Matrix Elements)**.** Given a collection of quark field operators $\{q_i\}$, $\{\bar{q}_i\}$ where $i = 1, \ldots, n_q$, the vacuum matrix element of the

time-ordered product may be evaluated as:

$$\begin{aligned}\langle q_1 \dots q_{n_q} \overline{q}_1 \dots \overline{q}_{n_q} \rangle &:= \langle 0 | \, T \left\{ q_1 \dots q_{n_q} \overline{q}_1 \dots \overline{q}_{n_q} \right\} | 0 \rangle \\ &= \left(\prod_{l=1}^{n_q - 1} (-)^l \right) \langle 0 | \, T \left\{ q_1 \overline{q}_1 \dots q_{n_q} \overline{q}_{n_q} \right\} | 0 \rangle \\ &= \left(\prod_{l=1}^{n_q - 1} (-)^l \right) \sum_{\vec{k} \in S_{n_q}} \epsilon^{k_1, \dots, k_{n_q}} S_{1,k_1} \dots S_{n_q, k_{n_q}}, \end{aligned} \tag{2.22}$$

where S_{n_q} is the set of the $n_q!$ permutations of $\{1, \dots, n_q\}$, and ϵ is the totally anti-symmetric tensor.

2.5. Calculation of Two-point Correlation Functions

In the context of Quantum Field Theory, a *correlation function* (or *correlator*) C is a vacuum expectation value of a time-ordered product of field operators constructed, for example, so as to preserve the quantum numbers of a state of interest. After a description of the general form, the following will walk through two examples of the calculation of nuclear two-point correlators to demonstrate a selection of key features and provide motivation for more advanced techniques developed subsequently.

For a given set of quantum numbers, a two-point correlation function takes the form:

$$C = \langle \mathscr{O}' \overline{\mathscr{O}} \rangle = \sum_{(\vec{a}, \vec{a}') \in \mathcal{I}} W^{a_1, \dots, a_{n_q}}_{a'_1, \dots, a'_{n_q}} \left\langle q(a_1) \dots q(a_{n_q}) \overline{q}(a'_1) \dots \overline{q}(a'_{n_q}) \right\rangle, \tag{2.23}$$

where $\vec{a} = (a_1, \dots, a_{n_q})$ combine flavour f_i, colour indices c_i, spinor indices α_i, and spacetime points x_i of quark fields $q(a_i) \equiv q^{f_i, c_i}_{\alpha_i}(x_i)$; $W^{a_1, \dots, a_{n_q}}_{a'_1, \dots, a'_{n_q}}$ combines tensors in $\mathscr{O}$ which project onto the correct set of quantum numbers; and $n_q = n_u + n_d$, where n_u (n_d) denotes the number of up (down) quark fields in operator $\mathscr{O}$. The set of values over which a_i, a'_j range, given by $\mathcal{I} = \{(\vec{a}, \vec{a}') \mid W^{\vec{a}}_{\vec{a}'} \neq 0\}$, is referred to as the *index set* and its cardinality $|\mathcal{I}|$ is referred to as the *index size*.

Consider the proton interpolating operator in Eqn. 2.10 given by:

$$p^\alpha(x) = \epsilon^{abc} \left(u^T_a(x) (C\gamma_5) d_b(x) \right) u^\alpha_c(x), \tag{2.24}$$

where the adjoint operator follows as:

$$\overline{p}^{\alpha'}(x') = \epsilon^{a'b'c'}\overline{u}^{\alpha}_{c'}(x')\left(\overline{d}_{b'}(x')(C\gamma_5)\overline{u}^{T}_{a'}(x')\right). \tag{2.25}$$

Construct the proton correlator with dangling source/sink spinor indices and definite sink momentum $\vec{p}$ from source point x':

$$\begin{aligned}
C_P^{\alpha\alpha'} &:= \left\langle \sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')} p^{\alpha}(x)\overline{p}^{\alpha'}(x') \right\rangle \\
&= \left\langle \sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')} \epsilon^{abc}\left(u_a^T(x)(C\gamma_5)d_b(x)\right) u_c^{\alpha}(x)\epsilon^{a'b'c'}\overline{u}^{\alpha}_{c'}(x')\left(\overline{d}_{b'}(x')(C\gamma_5)\overline{u}^{T}_{a'}(x')\right)\right\rangle \\
&= \sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')}\epsilon^{abc}\epsilon^{a'b'c'}(C\gamma_5)_{\beta\gamma}(C\gamma_5)_{\beta'\gamma'}\times \\
&\quad\times\left\langle \left(u_a^{\beta}(x)u_c^{\alpha}(x)\overline{u}^{\alpha'}_{c'}(x')\overline{u}^{\gamma'}_{a'}(x')\right)\left(d_b^{\gamma}(x)\overline{d}^{\beta'}_{b'}(x')\right)\right\rangle \\
&= \sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')}\epsilon^{abc}\epsilon^{a'b'c'}(C\gamma_5)_{\beta\gamma}(C\gamma_5)_{\beta'\gamma'}\times \\
&\quad\times\left(\underbrace{u_a^{\beta}(x)\overbrace{u_c^{\alpha}(x)\overline{u}^{\alpha'}_{c'}(x')}\overline{u}^{\gamma'}_{a'}(x')}_{} + \underbrace{u_a^{\beta}(x)\overbrace{u_c^{\alpha}(x)\overline{u}^{\alpha'}_{c'}(x')\overline{u}^{\gamma'}_{a'}(x')}}_{}\right)\left(\underbrace{d_b^{\gamma}(x)\overline{d}^{\beta'}_{b'}(x')}_{}\right) \\
&= \sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')}\epsilon^{abc}\epsilon^{a'b'c'}(C\gamma_5)_{\beta\gamma}(C\gamma_5)_{\beta'\gamma'}\times \\
&\quad\times\left(-S^{u,ac'}_{\beta\alpha'}(x,x')S^{u,ca'}_{\alpha\gamma'}(x,x') + S^{u,aa'}_{\beta\gamma'}(x,x')S^{u,cc'}_{\alpha\alpha'}(x,x')\right)S^{d,bb'}_{\gamma\beta'}(x,x').
\end{aligned} \tag{2.26}$$

Next, consider the deuteron II interpolating operator in Eqn. 2.11 given by:

$$D(x) = \frac{1}{\sqrt{2}}\left[n^T(x)(C\gamma_3)p(x) - p^T(x)(C\gamma_3)n(x)\right], \tag{2.27}$$

so that the adjoint is given by:

$$\overline{D}(x') = \frac{1}{\sqrt{2}}\left[-\overline{p}(x)(C\gamma_3)\overline{n}^T(x) + \overline{n}(x)(C\gamma_3)\overline{p}^T(x)\right]. \tag{2.28}$$

Construct the deuteron correlator with total sink momentum projected to $\vec{p}$ (note that this is not projecting the individual nucleons to definite momentum as explored in

Chapter 3) from source point x':

$$\begin{aligned} C_D &= \left\langle \sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')} D(x)\overline{D}(x') \right\rangle \\ &= \frac{1}{2}\sum_x e^{-i\vec{p}\cdot(\vec{x}-\vec{x}')} (C\gamma_3)_{\alpha\beta}(C\gamma_3)_{\alpha'\beta'} \times \\ &\quad \times \Big[\left\langle n^\alpha(x) p^\beta(x) \overline{n}^{\alpha'}(x') \overline{p}^{\beta'}(x') \right\rangle - \left\langle n^\alpha(x) p^\beta(x) \overline{p}^{\alpha'}(x') \overline{n}^{\beta'}(x') \right\rangle \\ &\quad - \left\langle p^\alpha(x) n^\beta(x) \overline{n}^{\alpha'}(x') \overline{p}^{\beta'}(x') \right\rangle + \left\langle p^\alpha(x) n^\beta(x) \overline{p}^{\alpha'}(x') \overline{n}^{\beta'}(x') \right\rangle \Big] . \end{aligned} \tag{2.29}$$

The first of the four terms in Eqn. 2.29 takes the form:

$$\begin{aligned} &\left\langle n^\alpha(x) p^\beta(x) \overline{n}^{\alpha'}(x') \overline{p}^{\beta'}(x') \right\rangle \\ &\quad = \epsilon^{abc}\epsilon^{def}\epsilon^{a'b'c'}\epsilon^{d'e'f'} (C\gamma_5)_{\gamma\delta}(C\gamma_5)_{\sigma\rho}(C\gamma_5)_{\gamma'\delta'}(C\gamma_5)_{\sigma'\rho'} \times \\ &\qquad \times \left\langle d_a^\gamma(x) u_b^\delta(x) d_c^\alpha(x) u_d^\sigma(x) d_e^\rho(x) u_f^\beta(x) \overline{d}_{c'}^{\alpha'}(x') \overline{u}_{b'}^{\gamma'}(x') \overline{d}_{a'}^{\delta'}(x') \overline{u}_{f'}^{\beta'}(x') \overline{d}_{e'}^{\sigma'}(x') \overline{u}_{d'}^{\rho'}(x') \right\rangle \\ &\quad = -\epsilon^{abc}\epsilon^{def}\epsilon^{a'b'c'}\epsilon^{d'e'f'} (C\gamma_5)_{\gamma\delta}(C\gamma_5)_{\sigma\rho}(C\gamma_5)_{\gamma'\delta'}(C\gamma_5)_{\sigma'\rho'} \times \\ &\qquad \times \Big\langle \left(u_b^\delta(x) u_d^\sigma(x) u_f^\beta(x) \overline{u}_{b'}^{\gamma'}(x') \overline{u}_{f'}^{\beta'}(x') \overline{u}_{d'}^{\rho'}(x') \right) \\ &\qquad \times \left(d_a^\gamma(x) d_c^\alpha(x) d_e^\rho(x) \overline{d}_{c'}^{\alpha'}(x') \overline{d}_{a'}^{\delta'}(x') \overline{d}_{e'}^{\sigma'}(x') \right) \Big\rangle . \end{aligned} \tag{2.30}$$

Then each of the bracketed terms in Eqn. 2.30 requires $3! = 6$ Wick contractions, resulting in $6^2 \times 4 = 144$ terms in C_D. Given the impracticality of this approach to nuclear correlators of larger A, Chapter 3 and beyond will explore a set of more efficient approaches.

2.6. Computing Nuclear Properties and Interactions

The purpose of this section is to give some insight into a brief selection of the methodologies used to extract nuclear physics observables from correlators as constructed in § 2.5. It can be shown [30] using the Hamiltonian evolution operator and an insertion of a complete set of states $\{|k\rangle\}$ that a correlator C at a fixed (sink) time t can be written as:

$$C(t) := \langle \mathcal{O}'(t)\mathcal{O}^\dagger(0)\rangle = \sum_k \langle 0|\,\hat{\mathcal{O}}'\,|k\rangle\,\langle k|\,\hat{\mathcal{O}}^\dagger\,|0\rangle\, e^{-aE_k t}. \tag{2.31}$$

Progress by re-writing Eqn. 2.31 as:

$$C(t) = Ze^{-aEt}\left(1 + \mathcal{O}\left(e^{-a\Delta Et}\right)\right), \tag{2.32}$$

where Z is a constant, E is the lowest energy in Eqn. 2.31, and ΔE is the difference between E and the next lowest energy in Eqn. 2.31. Using Eqn. 2.32, it is possible to extract E by defining an *effective energy function* such as:

$$aE_{\text{eff}}(t) := \ln\left(\frac{C(t)}{C(t+1)}\right), \tag{2.33}$$

and formally taking the limit $E = \lim_{t\to\infty} E_{\text{eff}}(t)$, or in practice by (for example) extracting the plateau value of $E_{\text{eff}}(t)$. There are several ways to improve the effective energy function for nuclear states. For example, baryon-baryon correlators (i.e. $A = 2$) without parity projection have forward and backward propagating states with both positive and negative parity, so one expects the correlator to take the form [31]:

$$\begin{aligned} C(t) = {} & Z_1 e^{-aE_A t} + Z_2 e^{-aE_{A'}(T-t)} + Z_3 e^{-aE_N t} e^{-aE_{N'}(T-t)} \\ & + Z_4 e^{-aE_A(T-t)} + Z_5 e^{-aE_{A'}t} + \ldots, \end{aligned} \tag{2.34}$$

where E_A ($E_{A'}$) is the energy of the positive-parity (negative-parity) A-baryon state, E_N ($E_{N'}$) is the energy of the positive-parity (negative-parity) single baryon state, and $\{Z_i\}$ are constant. By forming the ratio:

$$\begin{aligned} \frac{C(t-1) - C(t+1)}{2C(t)} &= \frac{Z_1 e^{-aE_A(t-1)} - Z_1 e^{-aE_A(t+1)}}{2Z_1 e^{-aE_A t}} + \textit{contaminants} \\ &= \sinh(aE_A) + \textit{contaminants}, \end{aligned} \tag{2.35}$$

one arrives at an improved effective energy function[1]:

$$aE'_{\text{eff}}(t) = \operatorname{arcsinh}\left(\frac{C(t-1) - C(t+1)}{2C(t)}\right). \tag{2.36}$$

[1] A similar process for meson correlators gives rise to:

$$aE'_{\text{eff}}(t) = \operatorname{arccosh}\left(\frac{C(t-1) + C(t+1)}{2C(t)}\right).$$

The interactions of baryons, such as (infinite volume) scattering phase shifts and binding energies can be studied in lattice QCD using for example Lüscher's finite volume formula [32,33], the HAL QCD method [34,35], or the Matrix Hamiltonian Model [36,37]. There exist interesting binding energy tensions between Lüscher's formula and the HAL QCD method, as discussed in Ref. [38].

In order to extract the nuclear binding energy $E_B := E_A - AE_N$, where E_A is the energy extracted from an A nucleon correlator $C_A(t)$ in the isospin symmetric limit and E_N is the energy extracted from a single nucleon correlator $C_N(t)$, one can consider the ratio [38,39]:

$$R(t) := \frac{C_A(t)}{[C_N(t)]^A}. \tag{2.37}$$

The extraction via $R(t)$ offers an improvement on systematic errors when compared with the direct calculation from E_A and E_N as $C_A(t)$ and $C_N(t)$ are highly correlated [38,39]. An analogous effective energy function to extract the binding energy E_B can be defined as:

$$a\Delta E''_{\text{eff}}(t) := \ln\left(\frac{R(t)}{R(t+1)}\right). \tag{2.38}$$

Lüscher's finite volume formula [32,33] allows the calculation of, for example, the $A = 2$ (S-wave) scattering phase shift $\delta(k)$ at momentum k using the formula [38]:

$$k\cot\{\delta(k)\} = \frac{1}{\pi(La)} \sum_{\vec{n}\in Z^3} \frac{1}{\vec{n}^2 - q^2}, \tag{2.39}$$

where $q = k(La)/(2\pi)$, and $E_A = 2\sqrt{E_N^2 + k^2}$ are computed at finite volume from the lattice.

The Hamiltonian Matrix Model [36,37] offers an alternate approach to calculating the scattering phase shift from finite volume lattice data. Following the example of $\Delta \to N\pi$ decay from Ref. [36], the method constructs the Hamiltonian $H = H_0 + H_I$,

where [36]:

$$H_0 = \begin{bmatrix} \Delta_0 & 0 & 0 & \dots \\ 0 & \omega_\pi(k_1) & 0 & \dots \\ 0 & 0 & \omega_\pi(k_2) & \\ \vdots & \vdots & & \ddots \end{bmatrix}, \tag{2.40}$$

$$H_I = \begin{bmatrix} 0 & g^{fin}_{\Delta N}(k_1) & g^{fin}_{\Delta N}(k_2) & \dots \\ g^{fin}_{\Delta N}(k_1) & 0 & 0 & \dots \\ g^{fin}_{\Delta N}(k_2) & 0 & 0 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \tag{2.41}$$

with bare resonance energy Δ_0, pion energy $\omega_\pi(k_n) = \sqrt{k_n + m_\pi^2}$, and $g^{fin}_{\Delta N}(k_n)$ obtained from χEFT. The on-shell t-matrix can then be related to the phase shift $\delta(k)$ via [36]:

$$t(k,k;E^+) = -\frac{1}{\pi k \omega_\pi(k)} e^{i\delta(k)} \sin\{\delta(k)\}. \tag{2.42}$$

Solving Eqn. 2.42 allows the extraction of $\delta(k)$. A complete account of the Hamiltonian Matrix Model can be found in Refs. [36, 37].

Chapter 3.

Review of Existing Algorithms

A great deal of algorithmic progress has been made on the subject of the numerical evaluation of nuclear correlators over the preceding years. The purpose of this chapter is to provide a survey of the major accomplishments and insight into the underlying formulations. Some of the algorithms explored here, such as hadron blocks (§ 3.2) and index lists (§ 3.3), will provide an essential bedrock for the algorithms developed in later chapters. Some basic results will be provided of the performance of these algorithms as applied to correlators of the nuclear operators introduced in Chapter 2. The remaining algorithms will be presented as self-contained, and the reader is encouraged to consult the references for a full account of the results obtained using these methods.

3.1. Matrix Determinant Formulation

The primary source for the matrix determinant formulation is Ref. [40], which presents an algorithm to numerically evaluate the quark-level Wick contraction (as introduced in Chapter 2):

$$\left\langle q(a_1)\dots q(a_{n_q})\overline{q}(a_1^{'})\dots\overline{q}(a_{n_q}^{'})\right\rangle, \tag{3.1}$$

where $\vec{a} = (a_1,\dots,a_{n_q})$ combine flavour f_i, colour indices c_i, spinor indices α_i, and spacetime points x_i of quark fields $q(a_i) \equiv q_{\alpha_i}^{f_i,c_i}(x_i)$. The presentation of this method, which follows Ref. [40] closely, begins with the general algebraic expression for the

Wick contraction of quark fields (introduced in Chapter 2):

$$\left\langle q(a_1)\dots q(a_{n_q})\overline{q}(a_1^{'})\dots\overline{q}(a_{n_q}^{'})\right\rangle = \left(\prod_{l=1}^{n_q-1}(-)^l\right)\sum_{\vec{k}\in S_{n_q}}\epsilon^{k_1\dots k_{n_q}}S(a_1;a_{k_1}^{'})\dots S(a_{n_q};a_{k_{n_q}}^{'})$$
$$= (-)^{(n_q-1)n_q/2}\sum_{\vec{k}\in S_{n_q}}\epsilon^{k_1\dots k_{n_q}}\prod_{j=1}^{n_q}S(a_j;a_{k_j}^{'}), \tag{3.2}$$

where the relative sign is introduced by rearranging $q(a_1)\dots q(a_{n_q})\overline{q}(a_1^{'})\dots\overline{q}(a_{n_q}^{'}) = \left(\prod_{l=1}^{n_q-1}(-)^l\right)q(a_1)\overline{q}(a_1^{'})\dots q(a_{n_q})\overline{q}(a_{n_q}^{'})$, and S_{n_q} is the set of all permutations of $[n_q] := \{1,\dots,n_q\}$. In accordance with Ref. [40], define an $n_q\times n_q$ matrix $G^{\vec{a},\vec{a}'}$ with elements $G_{ij}^{\vec{a},\vec{a}'} := S(a_i;a_j^{'})$ and therefore determinant:

$$\det G^{\vec{a},\vec{a}'} := \sum_{\vec{k}\in S_{n_q}}\epsilon^{k_1\dots k_{n_q}}\prod_{j=1}^{n_q}G_{j,k_j}^{\vec{a},\vec{a}'}, \tag{3.3}$$

so that the Wick contractions may be written as:

$$\left\langle q(a_1)\dots q(a_{n_q})\overline{q}(a_1^{'})\dots\overline{q}(a_{n_q}^{'})\right\rangle = (-)^{(n_q-1)n_q/2}\det G^{\vec{a},\vec{a}'}. \tag{3.4}$$

As an example, consider the specific instantiation where $n_q = 3$, with Wick contractions:

$$\begin{aligned}\left\langle q(a_1)q(a_2)q(a_3)\overline{q}(a_1^{'})\overline{q}(a_2^{'})\overline{q}(a_3^{'})\right\rangle&\\ = -S(a_1;a_1^{'})S(a_2;a_2^{'})S(a_3;a_3^{'}) + S(a_1;a_1^{'})S(a_2;a_3^{'})S(a_3;a_2^{'})&\\ + S(a_1;a_2^{'})S(a_2;a_1^{'})S(a_3;a_3^{'}) - S(a_1;a_2^{'})S(a_2;a_3^{'})S(a_3;a_1^{'})&\\ - S(a_1;a_3^{'})S(a_2;a_1^{'})S(a_3;a_2^{'}) + S(a_1;a_3^{'})S(a_2;a_2^{'})S(a_3;a_1^{'}).&\end{aligned} \tag{3.5}$$

Construct $G^{\vec{a},\vec{a}'}$ in the case where $n_q = 3$:

$$G^{\vec{a},\vec{a}'} = \begin{bmatrix} S(a_1;a_1^{'}) & S(a_1;a_2^{'}) & S(a_1;a_3^{'}) \\ S(a_2;a_1^{'}) & S(a_2;a_2^{'}) & S(a_2;a_3^{'}) \\ S(a_3;a_1^{'}) & S(a_3;a_2^{'}) & S(a_3;a_3^{'}) \end{bmatrix}, \tag{3.6}$$

so that the right-hand side of Eqn. 3.4 for $n_q = 3$ is evaluated as:

$$\begin{aligned}
&(-)^{(n_q-1)n_q/2} \det G^{\vec{a},\vec{a}'} \\
&\quad = -\begin{vmatrix} S(a_1;a_1') & S(a_1;a_2') & S(a_1;a_3') \\ S(a_2;a_1') & S(a_2;a_2') & S(a_2;a_3') \\ S(a_3;a_1') & S(a_3;a_2') & S(a_3;a_3') \end{vmatrix} \\
&\quad = -\Big[S(a_1;a_1')\left(S(a_2;a_2')S(a_3;a_3') - S(a_2;a_3')S(a_3;a_2')\right) \\
&\qquad - S(a_1;a_2')\left(S(a_2;a_1')S(a_3;a_3') - S(a_2;a_3')S(a_3;a_1')\right) \\
&\qquad + S(a_1;a_3')\left(S(a_2;a_1')S(a_3;a_2') - S(a_2;a_2')S(a_3;a_1')\right)\Big], \qquad (3.7)
\end{aligned}$$

which agrees with Eqn. 3.5. The primary advantage of Eqn. 3.4 is that $\det G^{\vec{a},\vec{a}'}$ may be evaluated numerically for a particular gauge configuration using LU factorisation [41], which uses $\mathcal{O}(n_q^3)$ operations rather than $\mathcal{O}(n_q!)$. One can further improve this result by considering that interactions in QCD preserve flavour quantum numbers, and so $\underbrace{q_\alpha^{f,c}(x)\overline{q}}_{\alpha'}^{f',c'}(x')$ vanishes for $f \neq f'$. This leads to the factorisation the Wick contraction (assuming two quark flavours, but it naturally generalises) as:

$$\begin{aligned}
&\left\langle q(a_1)\ldots q(a_{n_q})\overline{q}(a_1')\ldots\overline{q}(a_{n_q}')\right\rangle \qquad (3.8)\\
&\quad =: s\left\langle\left(u(a_1^u)\ldots u(a_{n_u}^u)\overline{u}(a_1^{u\,\prime})\ldots\overline{u}(a_{n_u}^{u\,\prime})\right)\left(d(a_1^d)\ldots d(a_{n_d}^d)\overline{d}(a_1^{d\,\prime})\ldots\overline{d}(a_{n_d}^{d\,\prime})\right)\right\rangle,
\end{aligned}$$

where $s \in \{-1, 1\}$ depends on the anti-symmetric rearrangement of quark operators (e.g. $\left\langle d_1u_1d_2\overline{d}_1\overline{d}_2\overline{u}_1\right\rangle = -\left\langle(u_1\overline{u}_1)\left(d_1d_2\overline{d}_1\overline{d}_2\right)\right\rangle$ has $s = -1$). Then, by applying Eqn. 3.4 to each factor, conclude that:

$$\left\langle q(a_1)\ldots q(a_{n_q})\overline{q}(a_1')\ldots\overline{q}(a_{n_q}')\right\rangle = s(-)^{(n_u-1)n_u/2+(n_d-1)n_d/2}\det G_u^{\vec{a},\vec{a}'}\det G_d^{\vec{a},\vec{a}'}, \quad (3.9)$$

where $G_u^{\vec{a},\vec{a}'}$ ($G_d^{\vec{a},\vec{a}'}$) is an $n_u \times n_u$ ($n_d \times n_d$) matrix with elements $G_{u;ij}^{\vec{a},\vec{a}'} := S(a_i^u;a_j^{u\,\prime})$ $\left(G_{d;ij}^{\vec{a},\vec{a}'} := S(a_i^d;a_j^{d\,\prime})\right)$. Through the factorisation in Eqn. 3.9, the number of operations required to compute $\left\langle q(a_1)\ldots q(a_{n_q})\overline{q}(a_1')\ldots\overline{q}(a_{n_q}')\right\rangle$ is reduced from $\mathcal{O}(n_q^3)$ to $\mathcal{O}(n_u^3n_d^3)$.

Consider the general case for a correlation function given by:

$$C = \langle \mathscr{O}' \mathscr{O}^\dagger \rangle = \sum_{(\vec{a},\vec{a}') \in \mathcal{I}} W^{a_1,\ldots,a_{n_q}}_{a'_1,\ldots,a'_{n_q}} \left\langle q(a_1)\ldots q(a_{n_q})\overline{q}(a'_1)\ldots\overline{q}(a'_{n_q}) \right\rangle, \tag{3.10}$$

where $W^{a_1,\ldots,a_{n_q}}_{a'_1,\ldots,a'_{n_q}}$ combines tensors in $\mathscr{O}$ that project onto the correct set of quantum numbers; $n_q = n_u + n_d$, where n_u (n_d) denotes the number of up (down) quark fields in operator $\mathscr{O}$; $\mathcal{I} = \{(\vec{a},\vec{a}') \mid W^{\vec{a}}_{\vec{a}'} \neq 0\}$ is referred to as the *index set* with cardinality, $|\mathcal{I}|$, referred to as the *index size*. In Eqn. 3.10, one must sum over $\mathcal{I}$ as well as evaluate the Wick contractions of the quark fields. The determinant method of Ref. [40] reduces the number of operations required to evaluate Eqn. 3.10 from $\mathcal{O}(|\mathcal{I}|n_u!n_d!)$ to $\mathcal{O}(|\mathcal{I}|n_u^3 n_d^3)$. The $n_u!n_d!$ factor grows faster than $|\mathcal{I}|$, but $|\mathcal{I}|$ dominates $n_u!n_d!$ for light nuclei.

3.2. Hadron Blocks

The formulation of multi-hadron correlation function contractions in terms of single hadron building blocks has been explored in Refs. [40, 42–44]. A hadron block is a tensor, $f^h_{\vec{p}}(x',\vec{\xi})$, constructed from the constituent quarks of a hadron h with colour and spinor indices bundled into $\vec{\xi}$, created at a fixed source spacetime location x'[1] and annihilated as a colour/spinor contracted hadron with definite momentum $\vec{p}$ at the sink. Consider the example case of a single proton block with operator given by:

$$p^\alpha(x) = \epsilon^{abc}\left(u^T_a(x)(C\gamma_5)d_b(x)\right)u^\alpha_c(x), \tag{3.11}$$

where the adjoint operator follows as:

$$\overline{p}^{\alpha'}(x') = \epsilon^{a'b'c'}\overline{u}^\alpha_{c'}(x')\left(\overline{d}_{b'}(x')(C\gamma_5)\overline{u}^T_{a'}(x')\right). \tag{3.12}$$

Compute the necessary Wick contractions using explicit colour and spinor indices for both the hadron block, $f^P_{\vec{p}}(x',\vec{\xi})$ with $\vec{\xi} = (a',\beta',b',\gamma',c',\alpha',\alpha)$, and the quark

[1] One could have each quark at a distinct source spacetime point, but that case won't be considered here.

propagator $\left(S^{f,cc'}_{\alpha\alpha'}(x,x') \equiv \underline{q^{f,c}_{\alpha}(x)}\overline{q}^{f,c'}_{\alpha'}(x')\right)$ via:

$$
\begin{aligned}
f^{P}_{\vec{p}}(x',\vec{\xi}) &\equiv f^{P}_{\vec{p}}(x',a',\beta',b',\gamma',c',\alpha')_{\alpha} \\
&:= \left\langle \sum_{x} e^{-i\vec{p}\cdot\vec{x}} p_{\alpha}(x)\overline{u}^{a'}_{\beta'}(x')\overline{d}^{b'}_{\gamma'}(x')\overline{u}^{c'}_{\alpha'}(x') \right\rangle \\
&= \sum_{x} e^{-i\vec{p}\cdot\vec{x}} \epsilon^{abc}(C\gamma_5)_{\beta\gamma} \left\langle u^{a}_{\beta}(x) d^{b}_{\gamma}(x) u^{c}_{\alpha}(x)\overline{u}^{a'}_{\beta'}(x')\overline{d}^{b'}_{\gamma'}(x')\overline{u}^{c'}_{\alpha'}(x') \right\rangle \\
&= \sum_{x} e^{-i\vec{p}\cdot\vec{x}} \epsilon^{abc}(C\gamma_5)_{\beta\gamma} \left[S^{u,ac'}_{\beta\alpha'}(x,x') S^{u,ca'}_{\alpha\beta'}(x,x') - S^{u,aa'}_{\beta\beta'}(x,x') S^{u,cc'}_{\alpha\alpha'}(x,x') \right] S^{d,bb'}_{\gamma\gamma'}(x,x').
\end{aligned}
\tag{3.13}
$$

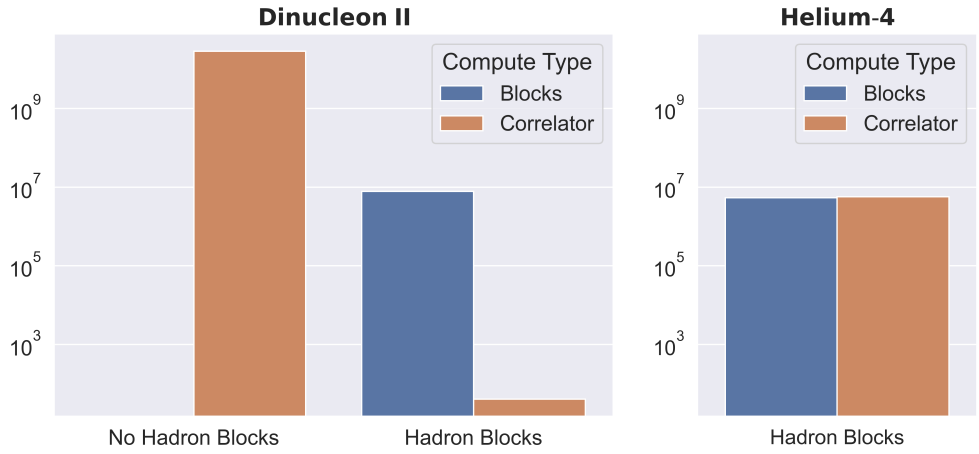


Figure 3.1.: Hadron block benchmark, measuring wall-clock time in milliseconds on a single core of an Intel Xeon Scalable Cascade Lake processor. The helium-4 (I) correlator without hadron blocks was too costly to perform here, but an upper bound on the number of operations required is 10^{22} (equivalent to $\sim 7 \times 10^{19}$ ms by extrapolating the dinucleon II result).

The general form (as will become useful later) for a nuclear correlator of A baryons using hadron blocks is given by:

$$
C = \sum_{k=1}^{N_w} w_k \epsilon_{a_1 b_1 c_1} \dots \epsilon_{a_A b_A c_A} \Gamma^{(1,k)}_{\alpha_1\beta_1} \dots \Gamma^{(N_\Gamma,k)}_{\alpha_{N_\Gamma}\beta_{N_\Gamma}} f^{h_1}_{\vec{p}_{(1,k)}}\left(x'_{(1,k)}, \vec{\xi}_{(1,k)}\right) \dots f^{h_A}_{\vec{p}_{(A,k)}}\left(x'_{(A,k)}, \vec{\xi}_{(A,k)}\right),
\tag{3.14}
$$

where $w_k \in \mathbb{C}$, N_Γ is the number of gamma matrices subject to $A \leq N_\Gamma \leq 2A$, N_w is the number of terms in the contracted expression for C, $\left\{ f^h_{\vec{p}}(x', \vec{\xi}) \right\}$ are sink-momentum projected hadron block functions, $\{h_i\}$ are hadrons determined from the nuclear interpolating operator $\mathcal{O}$, $\left\{ x'_{i,k} \right\}$ are the (possibly different) source spacetime points for the k^{th} term in the correlator, and $\left\{ \vec{\xi}_{(i,k)} \right\}$ are functions of $\{a_i\}$, $\{b_i\}$, $\{c_i\}$, $\{\alpha_i\}$, $\{\beta_i\}$ so that $\left\{ f^h_{\vec{p}}(x', \vec{\xi}) \right\}$ are fully contracted with $\{\epsilon_{abc}\} \cup \{\Gamma_{\alpha\beta}\}$.

The primary advantage of this construction is that block expressions are typically re-used many times during the course of evaluating particular choices of multi-hadron correlators. The factorial number of Wick contractions in the correlator is suppressed by a factor of 2^A for A nucleons, but without block expression re-use this computation cost is merely transferred to the hadron block evaluation. To measure the performance improvement associated with using hadron blocks for atomic nuclei, Figure 3.1 compares wall-clock time for the correlator computation (excluding propagator computation) in the isospin symmetric limit on a 64^3 volume using the dinucleon II and helium-4 I operators defined in Chapter 2. Figure 3.1 demonstrates that hadron blocks offer a clear performance improvement, even for $A = 2$ systems. Note that the cost to compute the block expressions (shown in blue) remains constant[2] in baryon number, whilst the cost to contract these blocks to form the multi-hadron correlator (shown in brown) scales factorially. For light nuclei ($A \lesssim 4$), the cost to compute the hadron blocks (shown in blue) dominates the cost the evaluate the correlator (shown in brown) as a result of the momentum projection cost dependence on the lattice volume. Ref. [39] seeks to reduce this cost using sparse sub-lattices, as explored in § 3.5.

3.3. Index Lists and the Unified Contraction Algorithm

The Unified Contraction Algorithm devised in Ref. [42] pre-computes an *index list*, which is the minimal subset of the *index set* $\mathcal{I}$ (the set over which one has to sum in order to contract all internal indices) of a numerically expensive tensor that has non-vanishing contribution to the correlator. This section will explore two applications to nuclear correlators: the first as presented in Ref. [42], and the second to minimise the cost of evaluating the hadron block tensors as introduced in § 3.2.

[2] Hadron Block cost differs between relativistic (e.g. dinucleon II) and non-relativistic (e.g. helium-4 I) forms

Thus far, correlation functions have been presented such that the Wick contractions permute the contracted indices of the hadron blocks. Equivalently, one may formulate correlation functions where the Wick contractions permute the indices of the colour/spinor projection tensors. For example, a dinucleon I correlator may be written as:

$$C = \sum_{\sigma \in G} \epsilon_{c_{\sigma(1)} c_{\sigma(2)} c_{\sigma(3)}} \epsilon_{c_{\sigma(4)} c_{\sigma(5)} c_{\sigma(6)}} (C\gamma_5)_{\alpha_{\sigma(1)} \alpha_{\sigma(2)}} (C\gamma_5)_{\alpha_{\sigma(4)} \alpha_{\sigma(5)}} (C\gamma_5)_{\alpha_{\sigma(3)} \alpha_{\sigma(6)}} \times \\ \times f^N_{\vec{p}}(x', c_1, \alpha_1, c_2, \alpha_2, c_3, \alpha_3)_\mu (C\gamma_5)_{\mu\nu} f^P_{\vec{p}}(x', c_4, \alpha_4, c_5, \alpha_5, c_6, \alpha_6)_\nu, \tag{3.15}$$

where $G \subset S_6$ permutes the six source quarks according to the hadron block formulation, beginning with the arrangement $(dud)(udu)$. Define a rank-12 tensor $T^{\{c_i\}}_{\{\alpha_i\}}$ with components:

$$T^{\{c_i\}}_{\{\alpha_i\}} = \sum_{\sigma \in G} \epsilon_{c_{\sigma(1)} c_{\sigma(2)} c_{\sigma(3)}} \epsilon_{c_{\sigma(4)} c_{\sigma(5)} c_{\sigma(6)}} (C\gamma_5)_{\alpha_{\sigma(1)} \alpha_{\sigma(2)}} (C\gamma_5)_{\alpha_{\sigma(4)} \alpha_{\sigma(5)}} (C\gamma_5)_{\alpha_{\sigma(3)} \alpha_{\sigma(6)}}. \tag{3.16}$$

Since $T^{\{c_i\}}_{\{\alpha_i\}}$ is independent of gauge configuration, any properties may be pre-computed and then re-used for many gauge configurations. The particular property of interest is the index list of $T^{\{c_i\}}_{\{\alpha_i\}}$, which is defined as follows.

Definition (Index List)**.** Let $T_{i_1,\ldots,i_r}$ be a tensor in a fixed basis with index set $\mathcal{I} =: I_1 \times \cdots \times I_r$ (i.e. each $i_k \in I_k$) that is contracted within the tensor expressions for a correlator C. An *index list* for T is a set $L \subseteq \{(j_1, \ldots, j_r) \in I_1 \times \cdots \times I_r\}$ such that $(j_1, \ldots, j_r) \in L$ if and only if $T_{j_1,\ldots,j_r}$ has a non-vanishing contribution to the correlator (for example, if $T_{j_1,\ldots,j_r} \neq 0$). Represent L efficiently on a computer by constructing bijections $M_k : I_k \to \{0, \ldots |I_k| - 1\}$ so that index list elements $(j_1, \ldots, j_r) \in L$ are represented by:

$$j = M_1(j_1) + (|I_1| - 1)(M_2(j_2) + (|I_2| - 1)(\ldots (M_{r-1}(j_{r-1}) + (|I_{r-1}| - 1)M_r(j_r)) \ldots)).$$

By constructing an index list L of all non-vanishing elements of $T^{\{c_i\}}_{\{\alpha_i\}}$ in Eqn. 3.16, the correlator for dinucleon I may be evaluated as:

$$C = \sum_{(\{c_i\},\{\alpha_i\}) \in L} T^{\{c_i\}}_{\{\alpha_i\}} f^N_{\vec{p}}(x', c_1, \alpha_1, c_2, \alpha_2, c_3, \alpha_3)_\mu (C\gamma_5)_{\mu\nu} f^P_{\vec{p}}(x', c_4, \alpha_4, c_5, \alpha_5, c_6, \alpha_6)_\nu. \tag{3.17}$$

The computational savings associated with evaluating Eqn. 3.17 rather than Eqn. 3.15 are proportional to the ratio of the cardinality of L with the number of elements of $T^{\{c_i\}}_{\{\alpha_i\}}$. This technique may be extended to larger nuclear systems, as presented in Ref. [42], with the caveat that one may have to construct distinct index lists for each of the terms in the correlator that have distinct hadron block factors (for example, different sink momenta or spacetime points). The rank of the tensor $T^{\{c_i\}}_{\{\alpha_i\}}$ grows exponentially in the number of hadrons, so the memory requirement for the index lists restricts the application to light nuclei with current hardware [42].

Next, apply the method of index lists to minimise the cost of evaluating hadron block tensors. Consider a general nuclear correlator after summation over the colour and spinor projection operators from Eqn. 3.14:

$$C = \sum_{k=1}^{\widetilde{N}_w} \widetilde{w}_k f^{h_1}_{\vec{q}_{(1,k)}}\left(\widetilde{x}'_{(1,k)}, \vec{\eta}_{(1,k)}\right) \dots f^{h_A}_{\vec{q}_{(A,k)}}\left(\widetilde{x}'_{(A,k)}, \vec{\eta}_{(A,k)}\right), \tag{3.18}$$

where $\vec{\eta}_{i,k}$ are bundles of fixed colour/spinor values (note the notation change $\vec{\xi} \to \vec{\eta}$ to differentiate contracted and fixed index bundles), $\widetilde{w}_k$ are the new weights after expansion, $\widetilde{N}_w$ is the number of terms in the expanded expression for C, and $\left\{\widetilde{x}'_{i,k}\right\}$ $(\{\vec{q}_{i,k}\})$ are the source spacetime points (sink momenta) of the expanded expression (noting that $\widetilde{x}'_{i,k} \neq x'_{i,k}$ $(\vec{q}_{i,k} \neq \vec{p}_{i,k})$ in general). For each distinct $f^h_{\vec{p}}$, construct an index list $L^h_{\vec{p}}$ of the elements of $f^h_{\vec{p}}$ that have non-vanishing contributions to the correlator. Here, that corresponds to tabulating all distinct $\left(\widetilde{x}'_{(i,k)}, \vec{\eta}_{(i,k)}\right)$ that appear as an argument to any $f^h_{\vec{p}}$ in Eqn. 3.18. Once each distinct $L^h_{\vec{p}}$ has been constructed (independently of gauge configuration), only the elements of $f^h_{\vec{p}}$ that appear in $L^h_{\vec{p}}$ need to be evaluated for each gauge configuration in order to calculate C.

3.4. Recursive Formulations

Ref. [43] builds on the Unified Contraction Algorithm presented in § 3.3 by developing an efficient recursive method to evaluate products of sparse anti-symmetric tensors. In this method, unique elements of a totally antisymmetric tensor $T_{i_1,\dots,i_r}$ are represented by tuples $\vec{A}\left(\{i_k\}\right) = (n(1),\dots,n(l))$, where $n(i_k) \in \{0,1\}$ is the number of occurrences of the value i_k amongst the indices $i_1,\dots,i_r$. The many-to-one relationship is canonicalised by asserting that each $\vec{A}\left(\{i_k\}\right)$ corresponds to the element $T_{i_1,\dots,i_r}$

where $i_{k_1} \leq i_{k_2}$ whenever $k_1 \leq k_2$. Through this notational change, Ref. [43] presents a method to efficiently calculate the antisymmetric product $Z = X \bullet Y$, where X (Y) has r_1 (r_2) indices, as [43]:

$$Z(z) = \sum_{\substack{x,y \\ x+y=z}} X(x)Y(y)\text{sgn}(x \mid y), \tag{3.19}$$

where $x = \vec{A}(i_1, \ldots, i_{r_1})$, $y = \vec{A}(i_{r_1+1}, \ldots, i_{r_1+r_2})$, $z = \vec{A}(i_1, \ldots, i_{r_1+r_2})$, and

$$\text{sgn}(x \mid y) = \prod_{\substack{i>j \\ y_j=1}} (-)^{x_i}. \tag{3.20}$$

Through recursive application of Eqn. 3.19, Ref. [43] computes chains of anti-symmetric tensors, $X^{(n)} = Y_1 \bullet Y_2 \bullet \cdots \bullet Y_n$. By computing an index list for $X^{(n)}$, only a sparse selection of its elements need to be evaluated. Through the repeated application of Eqn. 3.19, the index list is 'propagated' from $X^{(n)}$ to the individual tensors $Y_1, \ldots, Y_n$. Ref. [43] applies this recursive algorithm to construct correlators for atomic nuclei by adding nucleons iteratively to the correlator tensors. The total number of operations needed to calculate these correlators was reported in Ref. [43] to depend on the construction path (e.g. the chain $NNNNPPP$ was much more efficient than $NPPNPNN$ for P (N) a proton (neutron) [43]).

Refs. [45,46] present a comprehensive and general approach for computing correlation functions of multi-meson systems via recursion relations. The full extent of this work is too broad to present here, but it is worth presenting the core ideas applied to a simple case. Consider the correlation function for a system of $n \leq 12$ π^+ mesons:

$$C_{n\pi^+}(t) = \left\langle \left(\sum_{\vec{x}} \pi^+(\vec{x}, t) \right)^n \left(\pi^-(\vec{0}, 0) \right)^n \right\rangle, \tag{3.21}$$

where $\pi^+(\vec{x}, t) = \bar{d}(\vec{x}, t)\gamma_5 u(\vec{x}, t)$ and the restriction $n \leq 12$ arises due to the Pauli Exclusion Principle (with $N_c = 3$ colour indices and $N_s = 4$ spinor indices) [46]. Naïvely, the number of Wick contractions required to compute $C_{n\pi^+}$ is $n_{\bar{d}}! n_u! = (n!)^2$, however many of these are redundant. To proceed, define a colour-spinor matrix (in the isospin symmetric limit with degenerate propagators $S^u = S^d =: S$):

$$A_{ij}(t) = \sum_{\vec{x}} S(\vec{x}, t; \vec{0}, 0)_{ik} S^\dagger(\vec{x}, t; \vec{0}, 0)_{kj}, \tag{3.22}$$

where $i, j = 1, \ldots, 12$ are combined colour-spinor indices. Consider the following identities (derived in Ref. [46]):

$$\det(1 + \lambda A) = \sum_{j=1}^{12} (-)^j \left(\frac{1}{j!}\right) \lambda^j C_{j\pi^+}, \tag{3.23}$$

$$\det(1 + \lambda A) = \sum_{j=1}^{12} \frac{1}{j!} \lambda^j \mathrm{tr}_{C,S} \left[R_j(t)\right], \tag{3.24}$$

where λ is a scalar variable, $\mathrm{tr}_{C,S}$ is the trace over combined colour-spinor indices, and $R_n(t)$ is a partially contracted object that obeys a number of recursion relations. In particular, $R_n(t)$ obeys the relation (derived in Ref. [46]):

$$R_{n+1}(t) = \mathrm{tr}_{C,S} \left[R_n(t)\right] A(t) - n R_n(t) A(t), \tag{3.25}$$

with the boundary case $R_1(t) = A(t)$. By computing $R_n(t)$ through repeated application of the recursion relation in Eqn. 3.25, and by matching coefficients of powers of λ in Eqns. 3.23 and 3.24, one can compute $C_{n\pi^+}$ for any $n \leq 12$. Ref. [46] reports that the number of calculations required to evaluate $C_{12\pi^+}$ using this approach is $\sim 10^3$, as opposed to the naïve approach, which uses $(n!)^2 \approx 2.3 \times 10^{17}$ calculations[3]. Ref. [46] extends the recursive approach to various mesonic systems, including for an arbitrary number of mesons.

3.5. Sparsening Algorithm

As noted in § 3.2, the dominant scaling factor in the cost to evaluate hadron block expressions is the projection onto definite sink momentum, which requires a summation over all lattice sites. Ref. [39] presents an algorithm that exploits the local coherence of QCD by using a sink momentum projection over a sparse lattice. The following will explain briefly the algorithm of Ref. [39], as well as how it has been applied to reduce the cost of computing correlators for multi-hadron states.

For an L^3 spatial lattice, denote the set of lattice sites by:

$$\Lambda_3 := \{(n_1, n_2, n_3) \mid 0 \leq n_i < L\}. \tag{3.26}$$

[3] It is straightforward to re-use the up/down quark contractions in the isospin symmetric limit, resulting in $n! \approx 4.8 \times 10^8$ calculations [46].

The sites corresponding to the N^3 sub-lattice for some $N \in \{1, \ldots, L-1\}$ is defined by:

$$\widetilde{\Lambda}_3(N) := \{(\widetilde{n}_1, \widetilde{n}_2, \widetilde{n}_3) \mid 0 \leq \widetilde{n}_i < L \,;\, \widetilde{n}_i \equiv 0 \,(\text{mod } N)\}\,. \tag{3.27}$$

Then, for example, the 'full' proton block (derived in § 3.2):

$$\begin{aligned} f_{\vec{p}}^{P}(x', a', \beta', b', \gamma', c', \alpha')_{\alpha} \\ = \sum_{\vec{x} \in \Lambda_3} e^{-i\vec{p}\cdot\vec{x}} \epsilon^{abc} \Gamma_{\beta\gamma} \times \\ \times \left[S_{\beta\alpha'}^{u,ac'}(x, x') S_{\alpha\beta'}^{u,ca'}(x, x') - S_{\beta\beta'}^{u,aa'}(x, x') S_{\alpha\alpha'}^{u,cc'}(x, x') \right] S_{\gamma\gamma'}^{d,bb'}(x, x'), \end{aligned} \tag{3.28}$$

may be contrasted with the 'sparsened' proton block of sparsening factor N:

$$\begin{aligned} \widetilde{f}_{\vec{p}}^{P}(x', a', \beta', b', \gamma', c', \alpha')_{\alpha} \\ = \sum_{\vec{x} \in \widetilde{\Lambda}_3(N)} e^{-i\vec{p}\cdot\vec{x}} \epsilon^{abc} \Gamma_{\beta\gamma} \times \\ \times \left[S_{\beta\alpha'}^{u,ac'}(x, x') S_{\alpha\beta'}^{u,ca'}(x, x') - S_{\beta\beta'}^{u,aa'}(x, x') S_{\alpha\alpha'}^{u,cc'}(x, x') \right] S_{\gamma\gamma'}^{d,bb'}(x, x'). \end{aligned} \tag{3.29}$$

The computational cost required to compute Eqn. 3.29 is reduced by a factor of L^3/N^3 in comparison to Eqn. 3.28.

The sparsening process modifies the structure of the interpolating operator at the sink, while maintaining the quantum numbers of the given state. The extracted physical observables in the relevant limits are guaranteed to be independent of the choice of N, but the relative overlap factors are subject to change. Ref. [39] presents a comprehensive set of results in order to verify this claim.

Chapter 4.

Tensor Expression Canonicalisation

When computing nuclear correlation functions by-hand, there are often algebraic simplifications that greatly reduce complexity. To extend this reduction in complexity to systems for which by-hand calculation is impractical, it is useful to formalise the notion of 'algebraic simplification' in order to automate the process. One aspect of algebraic simplification for nuclear correlators is *tensor expression canonicalisation*, whereby contractions of tensors are manipulated into standard forms such that equivalent expressions (i.e. equal expressions up to permutation and relative sign) become equal, so that they may be accumulated. To demonstrate the challenge of this task, consider two expressions: $\epsilon_{ijk}\epsilon_{lmn}T_{ikm}T_{jln}$ and $\epsilon_{ijk}\epsilon_{lmn}T_{nmi}T_{lkj}$, where ϵ is the totally anti-symmetric tensor and T is any rank-3 tensor. These expressions are equivalent, but it would be a non-trivial task to write a set of explicit rules that one could follow to establish that fact. The difficulty arises because there are three interacting permutation symmetries: *label symmetries* (the fact that one can replace the labels $i \leftrightarrow j$), *slot symmetries* (the fact that one can replace $\epsilon_{ijk} \leftrightarrow \epsilon_{kij}$), and *tensor order symmetries* (the fact that one can replace $\epsilon_{ijk}\epsilon_{lmn} \leftrightarrow \epsilon_{lmn}\epsilon_{ijk}$). Canonicalising with respect to any of these symmetries independently is trivial, but canonicalisation with respect to the intertwined symmetries is highly non-trivial.

This chapter will develop the group-theoretic notions required to reason rigorously and efficiently about canonical forms of tensor expressions, and then apply that knowledge to the task of speeding up nuclear correlator calculations. § 4.1 will define graph canonicalisation, and explore its subtleties through McKay's Individualisation-Refinement Algorithm. § 4.2 will build on McKay's algorithm to develop a method that uses graph canonicalisation for tensor expression canonicalisation, and then § 4.3

will apply the developed methodology to provide results of the algebraic properties of a selection of nuclear correlators.

This chapter will rely upon a selection of techniques from Permutation Group Theory, for which Ref. [1] provides an excellent introduction. A particularly useful tool from Permutation Group Theory is the representation of permutations using *cycle notation*, where each cycle, $(n_1\ n_2\ \dots\ n_k)$, represents the permutation where $n_1 \mapsto n_2, n_2 \mapsto n_3, ..., n_k \mapsto n_1$. Any permutation can be written as a product of cycles, where the cycles act right-to-left. For example, the permutation $\sigma = (1\ 5)(1\ 7\ 9)$ represents the map where $1 \mapsto 5, 5 \mapsto 1 \mapsto 7, 7 \mapsto 9$, and $9 \mapsto 1$. A permutation σ may be applied to an object ω via a group action (i.e. a function that preserves the finite symmetric group product structure). For example, the permutation $\sigma = (1\ 5)(1\ 7\ 9)$ applied to the integer tuple $(1,5,7,9)$ can be computed as:

$$(1,5,7,9)^{(1\ 5)(1\ 7\ 9)} = (5,1,7,9)^{(1\ 7\ 9)} = (5,7,9,1), \tag{4.1}$$

from which it can be seen that an equivalent cycle notation expression for σ is given by $(1\ 5\ 7\ 9)$.

4.1. Graph Canonicalisation

4.1.1. Notation, Definitions, and Examples

Graph canonicalisation will play an essential role in tensor expression canonicalisation, which will be explored in the next section. This section will walk through some of the high-level details of the algorithm for graph canonicalisation as developed by McKay in Refs. [47,48]. The primary interest in McKay's algorithm, referred to here as *McKay's Individualisation-Refinement Algorithm*, is his seminal software package *nauty* [48], which was later upgraded to *Traces* [48, 49]. An excellent secondary source that provides greater insight into nauty's internals can be found in Ref. [50]. The following will define *graph isomorphism* (GI), *graph canonicalisation* (GC), and the surrounding notation of ordered partitions.

Definition (Coloured Graph Isomorphism)**.** Given a vertex set V isomorphic to $[n] := \{1, 2, \ldots, n\}$, define *graphs*:

$$\boldsymbol{G}(V) := \{\text{labelled simple (unweighted, undirected, single edge, no loops) graphs on } V\},$$

and *ordered partitions* (i.e. colourings of graphs in $\boldsymbol{G}(V)$):

$$\boldsymbol{\Pi}(V) := \{[V_1, \ldots, V_r] \mid \dot{\cup}_j V_j = V\} =: \{\textit{ordered partitions of } V\},$$

so that the pair $(G, \pi) \in \boldsymbol{G}(V) \times \boldsymbol{\Pi}(V)$ is referred to as a *coloured graph*. Two coloured graphs $(G_1, \pi_1), (G_2, \pi_2) \in \boldsymbol{G}(V) \times \boldsymbol{\Pi}(V)$ are *isomorphic* if $\exists \gamma \in S_n$ such that $G_1^\gamma = G_2$ and $\pi_1^\gamma = \pi_2$, where the action of γ on G_1 (π_1), denoted by G_1^γ (π_1^γ), permutes the labels of G_1 (π_1). Note that (non-coloured) graph isomorphism is the special case where one takes the trivial partition $\pi = [V]$. If $\pi = [V_1, \ldots, V_r], \pi' = [V'_1, \ldots, V'_{r'}] \in \boldsymbol{\Pi}(V)$, then:

(1) V_i is referred to as a *cell* of π

(2) if $|V_i| = 1$, then V_i is referred to as a *singleton cell* of π

(3) if all the cells of π are singleton cells, then π is referred to as a *discrete partition*

(4) if $r = 1$ (i.e. $\pi = [V]$), then π is referred to as the *trivial partition*

(5) if every cell of π is contained in a cell of π' and the cell ordering is *consistent* (i.e. $\forall V_i, V_j \in \pi$ with $i \leq j$, if $V'_k, V'_l \in \pi'$ such that $V_i \subseteq V'_k$ and $V_j \subseteq V'_l$ then $k \leq l$) then π is said to be *finer* than π' and π' is *coarser* than π

(6) π is *strictly finer* (*strictly coarser*) than π' if π is finer (coarser) than π' and $\pi \neq \pi'$.

Definition (Graph Canonicalisation Map)**.** A *graph canonicalisation map*

$$C : \boldsymbol{G}(V) \times \boldsymbol{\Pi}(V) \to \boldsymbol{G}(V)$$

satisfies the property that for any $G_1, G_2 \in \boldsymbol{G}(V)$, and $\pi_1, \pi_2 \in \boldsymbol{\Pi}(V)$, it is the case that $C(G_1, \pi_1) = C(G_2, \pi_2)$ if and only if (G_1, π_1) and (G_2, π_2) are isomorphic.

To provide an example of the graph canonicalisation map as provided by the *nauty* package [47], consider the two coloured graphs in Figure 4.1. Under nauty's canonicalisation map C, the two graphs in Figure 4.1 are mapped to the graph in Figure 4.2. According to the definition of the graph canonicalisation map, G_1 and G_2 must

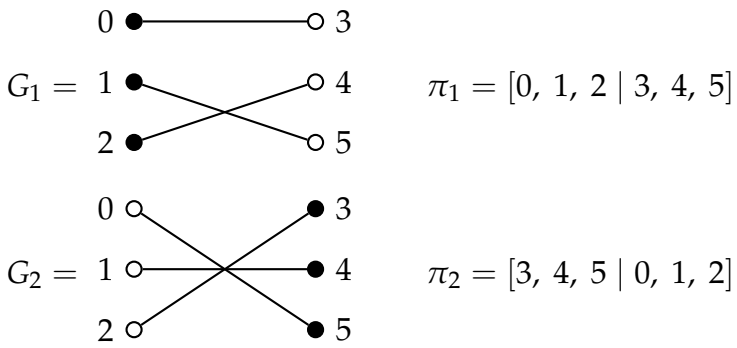


Figure 4.1.: Two coloured graphs (G_1, π_1) and (G_2, π_2). For visual clarity, the colouring of π_1 (π_2) has been applied to G_1 (G_2), but formally G_1 (G_2) has no colouring.

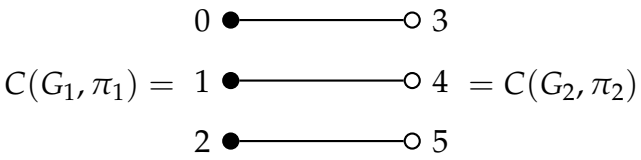


Figure 4.2.: (G_1, π_1) and (G_2, π_2) from Figure 4.1 after canonicalisation map C.

be isomorphic, and indeed they are since one can construct $\delta = (0\,4)(1\,3)(2\,5)$ using cycle notation ($0 \leftrightarrow 4, 1 \leftrightarrow 3, 2 \leftrightarrow 5$) so that $G_1^\delta = G_2$ and $\pi_1^\delta = [4,\ 3,\ 5 \mid 1,\ 0,\ 2] = \pi_2$.

Many applications of graph canonicalisation algorithms (including in the following sections) involve removing redundant isomorphs from a collection of N graphs. Using only isomorphism testing, one has to compare each graph against the remaining collection[1], resulting in $\mathcal{O}(N^2)$ total isomorphism tests. In comparison, one may canonicalise each graph and then use a standard data structure such as a hash map to remove duplicates, requiring only $\mathcal{O}(N)$ canonicalisations[2].

[1] One could minimise the number of total comparisons using a union-find data structure [51], but the computational complexity is certainly still superlinear in all but the trivial case.

[2] Isomorphism tests and canonicalisations are not strictly comparable, but one could reasonably conjecture that they both belong to the same computational complexity class, although it remains an open problem.

4.1.2. McKay's Individualisation-Refinement Algorithm

The following will use McKay's Individualisation-Refinement Algorithm [48] to canonicalise the example coloured graph in Figure 4.3. The general contour of the approach

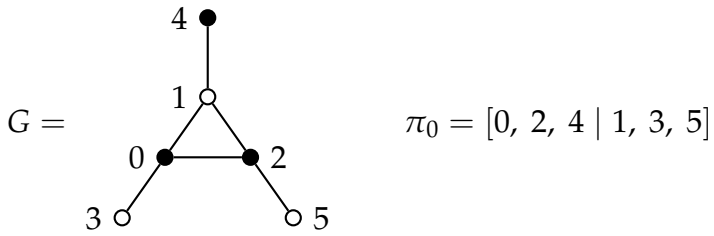


Figure 4.3.: Example coloured graph (G, π_0) to be canonicalised. For visual clarity, the colouring of π has been applied to G, but formally G has no colouring.

is to build a search tree by iteratively refining the ordered partition π_0 until one produces a set of discrete partitions which correspond to isomorphic relabellings of (G, π_0). The search tree for the example graph in Figure 4.3, as constructed throughout this section, is shown in Figure 4.4. Search tree pruning and final branch selection are performed by placing a canonical total ordering on the nodes of the search tree.

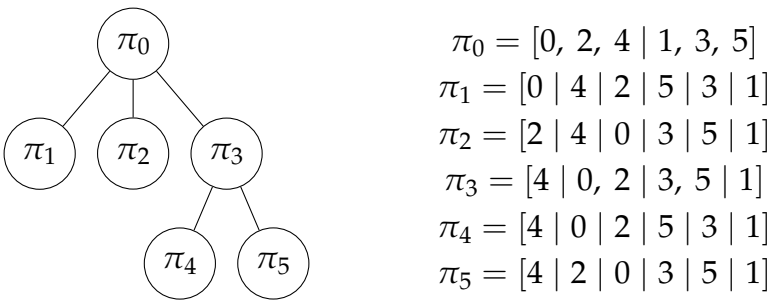


Figure 4.4.: Individualisation-refinement algorithm search tree for example graph (G, π_0) in Figure 4.3.

The first stage of building down the search tree from a *tree element* $\pi_i = [V_1, \ldots, V_r]$ is *individualisation*, whereby a non-singleton cell V_l of π_i is chosen by the *target cell selector*.

Definition (Target Cell Selector)**.** A *target cell selector* (adapted from Ref. [48]) is a function:

$$T : \boldsymbol{G}(V) \times \boldsymbol{\Pi}(V) \times \boldsymbol{\Pi}(V) \to \mathcal{P}(V) \coloneqq \{V' \mid V' \subset V\},$$

such that given any $G \in \boldsymbol{G}(V)$, $\pi_0 \in \boldsymbol{\Pi}(V)$, $\pi_i \in \boldsymbol{\Pi}(V)$:

(a) if π_i is discrete, then $T(G, \pi_0, \pi_i) = \emptyset$

(b) if π_i is not discrete, then $T(G, \pi_0, \pi_i)$ is a non-singleton cell of π_i

(c) $T(G^\gamma, \pi_0^\gamma, \pi_i^\gamma) = T(G, \pi_0, \pi_i)^\gamma$.

A simple (yet perhaps sub-optimal) choice of target cell selector is to select the first smallest non-singleton cell of π_i. Using the running example, one would choose $T(G, \pi_0, \pi_0) = \{0, 2, 4\}$. The procedure is then to create a candidate child tree element for each vertex in the target cell where that vertex becomes a singleton cell. In the example, there would be three candidate child tree elements: $[0 \mid 2, 4 \mid 1, 3, 5]$, $[2 \mid 0, 4 \mid 1, 3, 5]$, and $[4 \mid 0, 2 \mid 1, 3, 5]$. The next step is to refine each child partition into its *coarsest equitable refinement*, which is defined as follows.

Definition (Equitable Partition)**.** An ordered partition $\pi = [V_1, \ldots, V_r] \in \boldsymbol{\Pi}(V)$ is an *equitable partition* (adapted from Ref. [50]) w.r.t $G \in \boldsymbol{G}(V)$ if $deg(v, V_j) = deg(w, V_j)$ $\forall v, w \in V_i \; \forall i, j \in [r] \coloneqq \{1, 2, \ldots, r\}$. Note that $deg(v, V_j)$ is the number of edges from the vertex v to any vertex in the set V_j. An equitable partition π' is the *coarsest equitable refinement* of π if π' is finer than π and there doesn't exist an equitable partition which is finer than π and strictly coarser than π'.

	D_{ij}	1	2	3	4
	1	{ 0 }	{ 0 }	{ 1 }	{1, 1, 0}
j	2	{ 0 }	{ 0 }	{ 0 }	{1, 0, 0}
	3	{ 1 }	{ 0 }	{ 0 }	{1, 0, 1}
	4	{ 2 }	{ 1 }	{ 2 }	{ 0, 0, 0 }

i

Table 4.1.: Partition degree consistency table for $\tau' = [0 \mid 4 \mid 2 \mid 1, 3, 5] = [V_1', V_2', V_3', V_4']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i, j) \in B$.

There are several options to produce the coarsest equitable refinement from an inequitable partition, but presented here is Algorithm 1 as adapted from Ref. [50].

	D_{ij}	1	2	3	4	5
	1	{0}	{0}	{1}	{0}	{1,1}
	2	{0}	{0}	{0}	{0}	{1,0}
j	3	{1}	{0}	{0}	{1}	{1,0}
	4	{0}	{0}	{1}	{0}	{0,0}
	5	{2}	{1}	{1}	{0}	{0,0}

(i = column index)

Table 4.2.: Partition degree consistency table for $\tau' = [0 \mid 4 \mid 2 \mid 5 \mid 1,\ 3] = [V'_1, V'_2, V'_3, V'_4, V'_5]$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

For conciseness, introduce for a partition $\tau' = [V'_1, \ldots, V'_{r'}]$ the set $D_{ij} := \{deg(v, V'_j) \mid v \in V'_i\}$ such that $(i,j) \in B \Leftrightarrow |D_{ij}| > 1$ for B as in Algorithm 1. For illustrative purposes, apply Algorithm 1 to the first child of π_0. First set $\tau' = [0 \mid 2,\ 4 \mid 1,\ 3,\ 5] = [V'_1, V'_2, V'_3]$, and compute $B = \{(2,1),(2,3),(3,1),(3,2)\}$ by computing D_{ij} in Table 4.3. Then by selecting $(k,l) = (2,1)$ (the minimal element of B by lexicographic order), conclude that $[X_1, X_2] = [4 \mid 2]$, and so update $\tau' = [0 \mid 4 \mid 2 \mid 1,\ 3,\ 5]$. Next update $B = \{(4,1),(4,2),(4,3)\}$ by computing D_{ij} in Table 4.1, and by selecting $(k,l) = (4,1)$ conclude $[X_1, X_2] = [5 \mid 1,\ 3]$. By updating $\tau' = [0 \mid 4 \mid 2 \mid 5 \mid 1,\ 3]$, one can update $B = \{(5,2),(5,3)\}$ from Table 4.2, set $X = [3,\ 1]$, and finally update $\tau' = [0 \mid 4 \mid 2 \mid 5 \mid 3 \mid 1]$. As τ' has arrived at a discrete partition (which is equitable by necessity), the process terminates and one concludes that the coarsest equitable refinement of $[0 \mid 2,\ 4 \mid 1,\ 3,\ 5]$ is $\pi_1 = [0 \mid 4 \mid 2 \mid 5 \mid 3 \mid 1]$. The remainder of the worked example to build the search tree in Figure 4.4 can be found in Appendix A.

Algorithm 1: Coarsest Equitable Refinement Procedure [50]

Input: $G \in \boldsymbol{G}(V), \tau = [V_1, \ldots, V_r] \in \boldsymbol{\Pi}(V)$
Output: $\tau' \in \boldsymbol{\Pi}(V)$ (coarsest equitable refinement of τ)
$\tau' = \tau$
$B := \{(i,j) \mid V_j$ 'shatters' $V_i\} = \{(i,j) \mid \exists v, w \in V_i : deg(v, V_j) \neq deg(w, V_j)\}$
while $B \neq \emptyset$ **do**
  $(k,l) := \min_{\preccurlyeq}(B)$ ($\preccurlyeq$ denotes lexicographic order)
  $[X_1, \ldots, X_t] \in \boldsymbol{\Pi}(V_k)$ such that:
    if $v \in X_a, w \in X_b$ then $a < b \Leftrightarrow deg(v, V_l) < deg(w, V_l)$
  $\tau' = [V_1, \ldots, V_{k-1}, X_1, \ldots, X_t, V_{k+1}, \ldots, V_r] =: [V'_1, \ldots, V'_{r'}]$
  $B = \{(i,j) \mid V'_j$ 'shatters' $V'_i\}$
    $= \{(i,j) \mid \exists v, w \in V'_i : deg(v, V'_j) \neq deg(w, V'_j)\}$

	D_{ij}	$i=1$	$i=2$	$i=3$
j	1	$\{0\}$	$\boxed{\{1,0\}}$	$\boxed{\{1,1,0\}}$
	2	$\{1\}$	$\{0,0\}$	$\boxed{\{2,0,1\}}$
	3	$\{2\}$	$\boxed{\{2,1\}}$	$\{0,0,0\}$

Table 4.3.: Partition degree consistency table for $\tau' = [0 \mid 2,\ 4 \mid 1,\ 3,\ 5] = [V'_1, V'_2, V'_3]$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

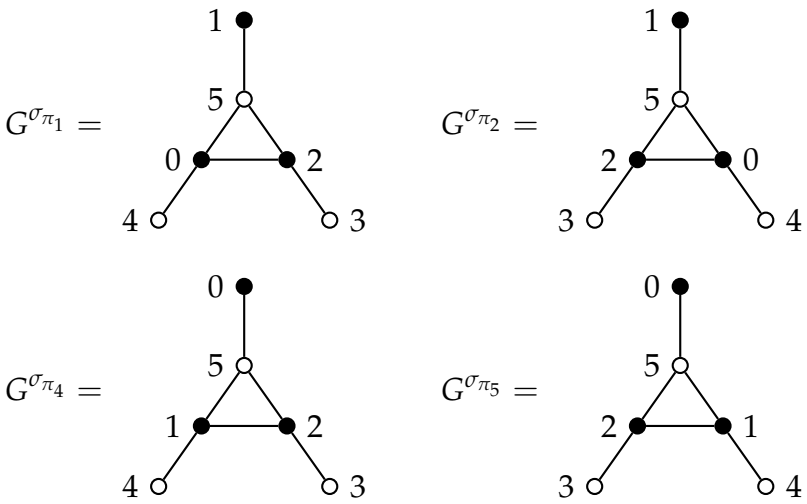


Figure 4.5.: Candidate canonical coloured graphs $G^{\sigma_{\pi_1}}$, $G^{\sigma_{\pi_2}}$, $G^{\sigma_{\pi_4}}$, and $G^{\sigma_{\pi_5}}$, where $G^{\sigma_{\pi_1}} = G^{\sigma_{\pi_2}}$ and $G^{\sigma_{\pi_4}} = G^{\sigma_{\pi_5}}$.

The four leaves of the search tree in Figure 4.4, $\pi_1, \pi_2, \pi_4, \pi_5$, correspond to the four permutations $\sigma_{\pi_1} = (4\ 1\ 5\ 3)$, $\sigma_{\pi_2} = (2\ 0)(4\ 1\ 5)$, $\sigma_{\pi_4} = (4\ 0\ 1\ 5\ 3)$, $\sigma_{\pi_5} = (4\ 0\ 2\ 1\ 5)$. Figure 4.5 shows the original graph G under the four permutations. It can be seen from Figure 4.5 that $G^{\sigma_{\pi_1}} = G^{\sigma_{\pi_2}}$ and $G^{\sigma_{\pi_4}} = G^{\sigma_{\pi_5}}$, so it remains to canonicalically select one of these two options by placing a total ordering on $\boldsymbol{G}(V)$. Using the proposal from Ref. [50], consider lexicographic ordering on *graph binary sequences*.

Definition (Graph Binary Sequence)**.** Let $G \in \boldsymbol{G}(V)$ with $|V| = n$ and adjacency matrix $\mathbf{A}$ with elements $A_{ij} = A_{ji} \in \{0,1\}$ such that $A_{ij} = 1$ if nodes i and j share an

edge, and $A_{ij} = 0$ otherwise. The *graph binary sequence*,

$$i(G) := [A_{12}A_{13}\dots A_{1n}A_{23}A_{24}\dots A_{(n-1)n}],$$

is the binary sequence of length $\binom{n}{2}$ of adjacency matrix elements A_{ij} with $i < j$, ordered lexicographically on (i, j). For example, if $n = 6$ then:

$$i(G) = [A_{12}A_{13}A_{14}A_{15}A_{16}A_{23}A_{24}A_{25}A_{26}A_{34}A_{35}A_{36}A_{45}A_{46}A_{56}].$$

Continuing the running example, the graph binary sequences for $\{G^{\sigma_{\pi_k}}\}$ are given by:

$$i(G^{\pi_1}) = [010110001101000] = i(G^{\pi_2}),$$
$$i(G^{\pi_4}) = [000011011101000] = i(G^{\pi_5}).$$

Under lexicographic order, $i(G^{\sigma_{\pi_4}}) = i(G^{\sigma_{\pi_5}}) \preccurlyeq i(G^{\sigma_{\pi_1}}) = i(G^{\sigma_{\pi_2}})$, so by selecting the maximal candidate graph one may conclude that the canonical isomorph of G under C is $G^{\sigma_{\pi_1}}$.

4.2. Tensor Expression Canonicalisation via Graph Canonicalisation

4.2.1. Tensor Network Notation

In order to use the techniques developed for graph canonicalisation in the previous section for tensor expressions, one must encode the structural information of tensor expressions into graphs such that tensor expression isomorphism is equivalent to the corresponding graph isomorphism. A satisfying presentation of such an encoding uses *tensor network notation* as an initial step. A succinct introduction to tensor network notation is provided here, but an excellent complete presentation can be found in Ref. [52].

In tensor network notation, each tensor is represented in a fixed basis by a geometric shape (circles are used here) with legs corresponding to indices. Figure 4.6 (a) depicts the rank-3 tensor ϵ_{ijk} in tensor network notation. When particular care must be taken

about whether basis vectors belong in the tensor's Hilbert space or the corresponding dual space, it is customary to adopt a convention on leg direction, but here consider it to be implicit. There are several graphical operations used to combine tensors: contraction, tensor product, and index grouping/splitting. Figure 4.6 (b) depicts the contraction with implicit summation of two rank-3 tensors by connecting the legs. Tensor rank is fluid in tensor network notation, so legs contracting common tensors may be 'fused' into a single leg (known as *index grouping*). The 'fused' version of the tensor network in Figure 4.6 (b) is shown in Figure 4.6 (c). Conversely, single legs may be 'expanded' into multiple legs (known as *index splitting*). There is freedom in the choice of basis for index grouping/splitting, but there are several standard choices that can be adopted to make the operation well-defined. The graphical contraction notation extends to the partial trace of a tensor by connecting a leg to itself, as shown in Figure 4.6 (d). The tensor product is represented implicitly by placing tensors next to each other without contraction, as depicted in Figure 4.6 (e).

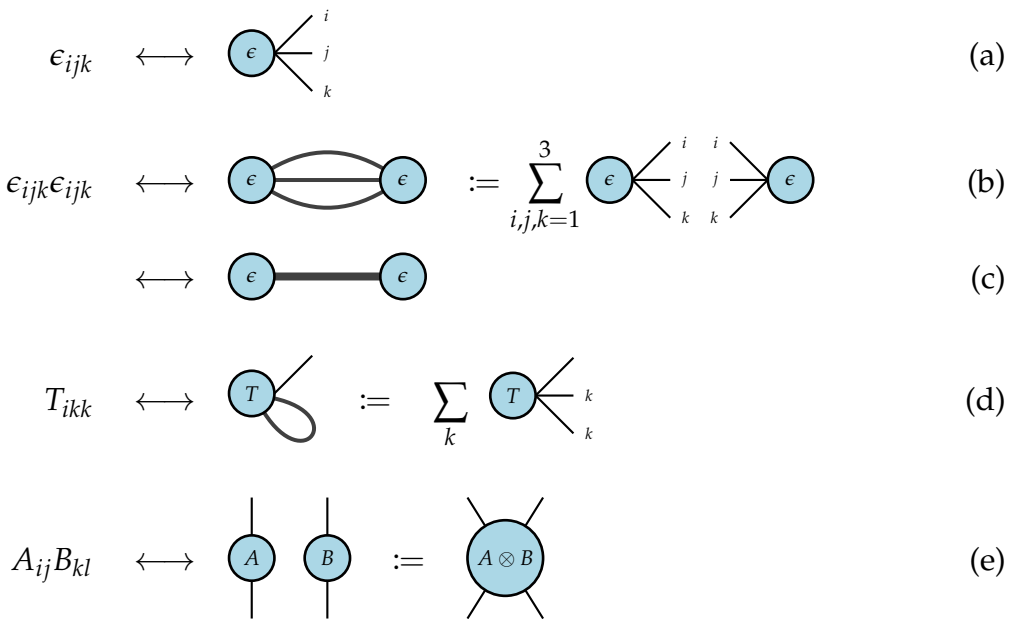


Figure 4.6.: Example algebraic tensor expression (left) and corresponding tensor network representation (right) for single tensor (a), contracted tensors (b), contracted tensors with 'fused' legs (c), partial trace (d), and tensor product (e).

4.2.2. Tensor Expressions as (Coloured) Graphs

In order to harness the machinery of graph canonicalisation, one must work with simple graphs, which are unweighted, undirected, single-edge graphs with no loops. Tensor network notation uses multi-edge graphs, so follow a similar presentation to that found in Ref. [53] to transform tensor networks into colour-partitioned simple graphs. Consider 'single-edge' tensor network notation, whereby each tensor is represented by a collection of nodes: one for each index slot. Examples of this notation for common tensor network operations are depicted in Figure 4.7, where dotted rectangles are used to keep track of which nodes (corresponding to index slots) belong to which tensors. Note that this representation does indeed satisfy all the requirements for a simple graph.

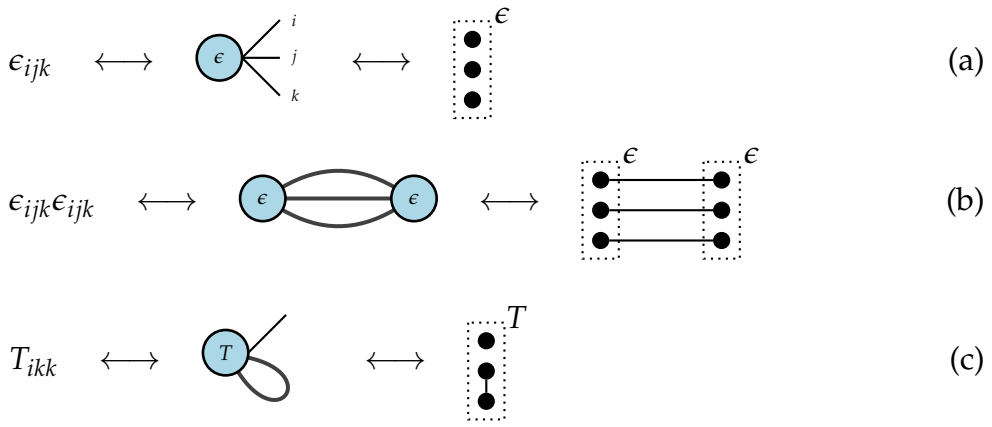


Figure 4.7.: Example algebraic tensor expression (left), tensor network notation (middle), and 'single-edge' tensor network notation (right) for single tensor (a), contracted tensors (b), and partial trace (c).

The next notational transformation is to encode the slot symmetry properties of the individual tensors into the colour partition of the simple graph given by the corresponding 'single-edge' tensor network. Each group of symmetric/anti-symmetric slots (or in fact any set of slots such that any slot permutation yields the same tensor, up to a multiplicative constant) is assigned its own cell in the colour partition. Slots that have no symmetry properties are assigned singleton cells. For example, the tensor ϵ_{ijk} is assigned the colour partition $[0,\ 1,\ 2]$, whereas the rank-3 tensor T_{ijk} with no slot symmetry is assigned $[0 \mid 1 \mid 2]$. Figure 4.8 depicts examples of the full graphical evolution of algebraic tensor expressions to simple graphs with colour partitions such

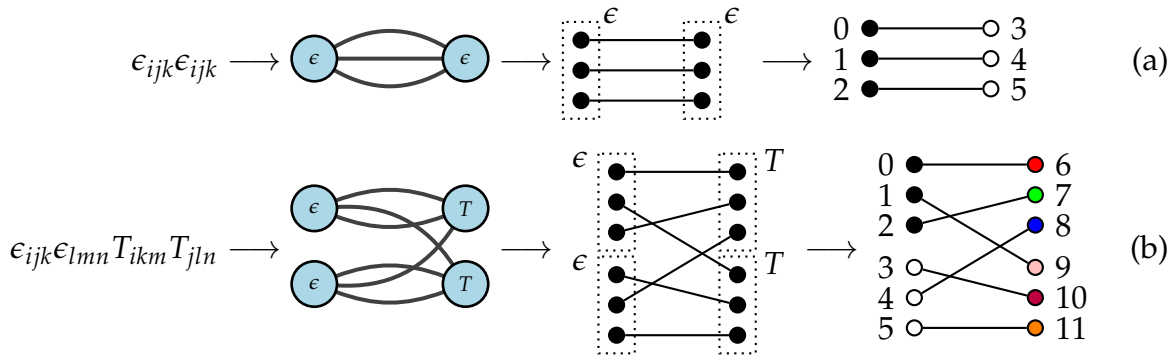


Figure 4.8.: Notational transformation from algebraic tensor expression (left) to tensor network notation (middle left) to 'single-edge' tensor network notation (middle right) to tensor expression graph (simple graph with colour partition) (right) for example expressions (a) and (b), where T is a rank-3 tensor with no slot symmetries.

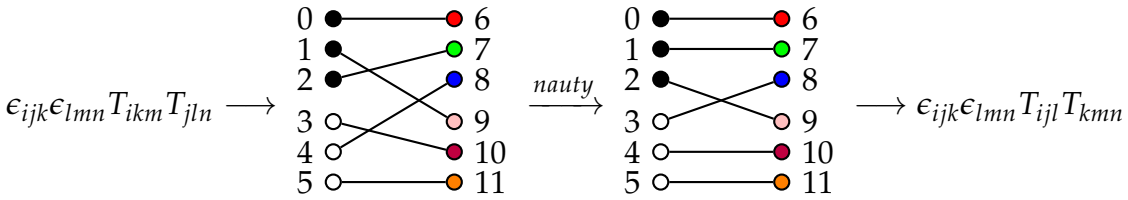


Figure 4.9.: Example tensor expression canonicalisation via graph canonicalisation (GC) using nauty [48] for GC. The tensor expression is the same as in Figure 4.8 (b).

that graph isomorphism is equivalent to tensor expression isomorphism[3]. Figure 4.9 depicts a summary of tensor expression canonicalisation via graph canonicalisation using nauty [48] for an example tensor expression.

4.2.3. Identical Tensor Ordering

The tensor expression canonicalisation methodology developed in the previous section canonicalises tensor expressions with respect to both label symmetries and slot symmetries, but fails to completely canonicalise tensor order. To demonstrate the failure mode, consider the following two tensor expressions:

[3]Up to identical tensor ordering, as explored and resolved in the next section.

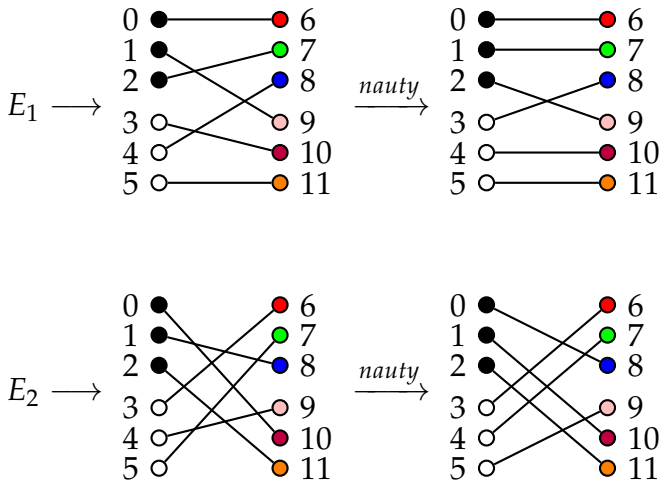


Figure 4.10.: Tensor expressions from Equations 4.2 and 4.3, converted into coloured graphs, and canonicalised using nauty.

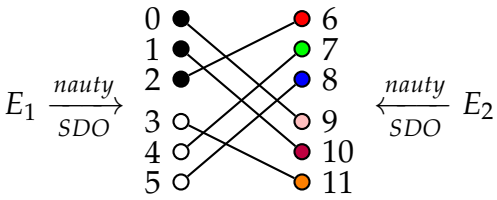


Figure 4.11.: Tensor expressions from Equations 4.2 and 4.3, converted into coloured graphs, and canonicalised using nauty and slot distance order (SDO).

$$E_1 = \epsilon_{ijk}\epsilon_{lmn}T_{ikm}T_{jln}, \tag{4.2}$$

$$E_2 = \epsilon_{ijk}\epsilon_{lmn}T_{lnj}T_{mik}. \tag{4.3}$$

The two expressions in Eqns. 4.2, 4.3 are equivalent, but their canonicalised graphs could be non-identical, as depicted in Figure 4.10. The tensor expression graphs that are supplied to nauty carry no information about the underlying tensors — only their underlying slot symmetry. In the example, ϵ_{ijk} in E_1 plays the structural role of ϵ_{lmn} in E_2, so exchanging $i, j, k \leftrightarrow l, m, n$ would resolve the failure to match after

canonicalisation. Note that for non-identical tensors, it is straightforward to place a total ordering on tensors, for example by asserting that $\epsilon < T$. It remains to order identical tensors using a procedure that doesn't depend on the given tensor order. Here use a method referred to as *slot distance ordering*, where for each index slot of a tensor expression, define the *slot distance* d_s to be the maximum number of index slots to the slot of the repeated index (i.e. distance after re-ordering the left (right) slot's tensor maximally left (right) in its identical tensor group), or zero if no such repeated index occurs. A canonical tensor order may be enforced by sorting identical tensors by lexicographic order on slot distances for slots belonging to each tensor. In the case of E_1, the slot distances (grouped by tensors) are given by:

$$\begin{array}{cccccccccccc} \epsilon & & & \epsilon & & & T & & & T & & \\ (i & j & k) & (l & m & n) & (i & k & m) & (j & l & n) \\ (9 & 8 & 8) & (10 & 10 & 9) & (9 & 8 & 10) & (8 & 10 & 9), \end{array} \tag{4.4}$$

and the slot distances for E_2 are:

$$\begin{array}{cccccccccccc} \epsilon & & & \epsilon & & & T & & & T & & \\ (i & j & k) & (l & m & n) & (l & n & j) & (m & i & k) \\ (10 & 10 & 9) & (9 & 8 & 8) & (9 & 8 & 10) & (8 & 10 & 9). \end{array} \tag{4.5}$$

In the case of Eqn. 4.4, slot distance order dictates that the third and fourth tensors are swapped, resulting in $E_1 \to \epsilon_{ijk}\epsilon_{lmn}T_{jln}T_{ikm}$. In the case of Eqn. 4.5, slot distance order dictates that first and second tensors are swapped, and that the third and fourth tensors are swapped, resulting in $E_2 \to \epsilon_{lmn}\epsilon_{ijk}T_{mik}T_{lnj} = \epsilon_{ijk}\epsilon_{lmn}T_{jln}T_{ikm}$ after relabelling from the left. Both E_1 and E_2 are canonicalised to the same expression, as depicted in Figure 4.11.

4.3. Application to Nuclear Correlation Functions

4.3.1. Software Package for Computing Nuclear Wick Contractions

This chapter has thus far developed techniques to canonicalise tensor expressions through graph canonicalisation. This section will apply tensor expression canonicalisa-

tion to correlators of a selection of nuclear interpolating operators. In order to present the results, a symbolic manipulation program has been written in C++ that generates the tensor expressions (in symbolic form) for a nuclear correlation function given its operator. The program accepts a list of operator terms in one of two operator input modes:

(1) An external tensor expression with a list of source and sink quark operators (specifying colour indices, spinor indices, spacetime indices) that contract with the external tensor expression. The input for a proton operator could be given by the expression $\epsilon_{abc}\Gamma_{\beta\gamma}\epsilon_{a'b'c'}\Gamma_{\beta'\gamma'}$, the source quarks $\overline{u}^{a'}_{\beta'}(x')$, $\overline{d}^{b'}_{\gamma'}(x')$, $\overline{u}^{c'}_{1}(x')$, and the sink quarks $u^{a}_{\beta}(x)$, $d^{b}_{\gamma}(x)$, $u^{c}_{1}(x)$.

(2) An external tensor expression with a list of source quark operators and a list of sink hadron blocks. The input for a proton operator could be given by the expression $\epsilon_{a'b'c'}\Gamma_{\beta'\gamma'}$, the source quarks $\overline{u}^{a'}_{\beta'}(x')$, $\overline{d}^{b'}_{\gamma'}(x')$, $\overline{u}^{c'}_{1}(x')$, and the sink block expression $f^{P}_{\vec{0}}$ as defined in Chapter 3.

There are three modes to generate tensor expressions from operator terms: without hadron blocks, with hadron blocks, and with 'half nucleon' blocks (as explained in the next sub-section).

4.3.2. 'Half Nucleon' Blocks

Nucleon blocks were introduced in Chapter 3, where the proton block was given by:

$$f^{P}_{\vec{p}}(x',a',\beta',b',\gamma',c',\alpha')_{\alpha} = \sum_{x} e^{-i\vec{p}\cdot\vec{x}}\epsilon^{abc}\Gamma_{\beta\gamma}\left[S^{u,ac'}_{\beta\alpha'}(x,x')S^{u,ca'}_{\alpha\beta'}(x,x') - S^{u,aa'}_{\beta\beta'}(x,x')S^{u,cc'}_{\alpha\alpha'}(x,x')\right]S^{d,bb'}_{\gamma\gamma'}(x,x'). \tag{4.6}$$

Similarly, the neutron block is given by:

$$f^{P}_{\vec{p}}(x',a',\beta',b',\gamma',c',\alpha')_{\alpha} = \sum_{x} e^{-i\vec{p}\cdot\vec{x}}\epsilon^{abc}\Gamma_{\beta\gamma}\left[S^{d,ac'}_{\beta\alpha'}(x,x')S^{d,ca'}_{\alpha\beta'}(x,x') - S^{d,aa'}_{\beta\beta'}(x,x')S^{d,cc'}_{\alpha\alpha'}(x,x')\right]S^{u,bb'}_{\gamma\gamma'}(x,x'). \tag{4.7}$$

The calculations here will work in the isospin symmetric limit, where the quark propagators are degenerate: $S^d \to S^u =: S$. Denote by $\widetilde{f}^h$ the hadron block for the hadron h under isospin symmetry. It's clear from Eqns. 4.6 and 4.7 that $\widetilde{f}^P = \widetilde{f}^N$. It remains to argue that in the isospin symmetric limit, it is sensible to define the (sink momentum projected) *'half nucleon' block*:

$$\widetilde{B}_{\vec{p}}(x', a', \beta', b', \gamma', c', \alpha')_\alpha := \sum_x e^{-i\vec{p}\cdot\vec{x}} \epsilon^{abc} \Gamma_{\beta\gamma} S^{aa'}_{\beta\beta'}(x, x') S^{bb'}_{\gamma\gamma'}(x, x') S^{cc'}_{\alpha\alpha'}(x, x'), \tag{4.8}$$

so that the nucleon hadron block may be written in two pieces:

$$\widetilde{f}^{P/N}_{\vec{p}}(x', a', \beta', b', \gamma', c', \alpha')_\alpha = \widetilde{B}_{\vec{p}}(x', c', \alpha', b', \gamma', a', \beta')_\alpha - \widetilde{B}_{\vec{p}}(x', a', \beta', b', \gamma', c', \alpha')_\alpha. \tag{4.9}$$

The consequences are two-fold. First, the cost to compute the full set of 'half nucleon' block expressions is reduced by one-half compared to the full set of isospin symmetric nucleon block expressions. The cost reduction comes at the expense of increasing the number of terms in the correlator by 2^A, which for light nuclei at reasonable lattice volumes is an attractive trade-off since the nucleon block cost dominates the total correlator calculation cost in that regime. Second, there's a much greater propensity for tensor expressions in the correlator to be isomorphic, leading to greater cost savings using tensor expression canonicalisation. This second consequence will be explored numerically in the next sub-section.

4.3.3. Tensor Expression Statistics for Nuclear Operators

The following will explore the number of tensor expressions before and after canonicalisation for a selection of nuclear operators under the three modes of operation: without hadron blocks, with hadron blocks, and with 'half nucleon' blocks. Reported here are five data for each operator: the operator term count, the Wick contraction count (mode-dependent), the uncanonicalised expression count, the canonicalised expression count (output of the symbolic manipulation program), and the ratio of canonicalised and uncanonicalised expression counts (denoted R). The uncanonicalised expression count is always given by $c_O^2 c_W$, where c_O is the operator term count and c_W is the Wick contraction count.

Operator	Operator Term Count	Wick Contraction Count	Uncanonicalised Expession Count	Canonicalised Expression Count (R)
Proton	1	2	2	2 (1)
Dinucleon I	1	36	36	36 (1)
Dinucleon II	2	36	144	72 (0.5)
Helium-3 I	1	2,880	2,880	2,880 (1)

Table 4.4.: Nuclear correlator tensor expression statistics assuming isospin symmetry, and without hadron blocks. Canonicalisation performed via nauty [48].

Without hadron blocks, the Wick contraction count for a multi-nucleon state is given by $n_u!n_d!$, where n_q is the number of flavour q quarks in the operator. Table 4.4 reports the canonicalised expression count for a selection of operators. Note that the Wick contraction count grows factorially, restricting the regime of computable nuclear correlators with currently available hardware.

Operator	Operator Term Count	Wick Contraction Count	Uncanonicalised Expession Count	Canonicalised Expression Count (R)
Proton	1	1	1	1 (1)
Dinucleon I	1	9	9	9 (1)
Dinucleon II	2	9	36	14 (0.39)
Triton I	1	720	720	360 (0.5)
3 Protons	6	540	19,440	4,260 (0.22)
Helium-3 I	1	360	360	360 (1)
Helium-3 II	6	360	12,960	10,622 (0.82)
Helium-4 I	1	32,400	32,400	32,292 (0.997)

Table 4.5.: Nuclear correlator tensor expression statistics assuming isospin symmetry, and with hadron blocks. Wick contraction count is given by Eqn. 4.10. Canonicalisation is performed via nauty [48]. Triton I (3 Protons) are constructed via appropriate nucleon substitution of Helium-3 I (Helium-3 II) operators. The '3 Protons' correlator vanishes (non-trivially) by the Pauli Exclusion Principle.

With hadron blocks, the Wick contraction count is given by:

$$\prod_{q=u,d} \binom{n_q}{N_q^1} \binom{n_q - N_q^1}{N_q^2} \cdots \binom{n_q - N_q^1 - \cdots - N_q^{A-1}}{N_q^A} \tag{4.10}$$

where N_q^i is the number of flavour q quarks in the i^{th} hadron block, n_q is the total number of flavour q quarks, and A is the number of nucleons. Note that Eqn. 4.10 is symmetric under the re-ordering of blocks. For concreteness, in the case of Helium-3 (with nucleons Proton, Neutron, Proton), it is the case that $A = 3$, $n_u = 5$, $n_d = 4$, $N_u^1 = 2$, $N_u^2 = 1$, $N_u^3 = 2$, $N_d^1 = 1$, $N_d^2 = 2$, $N_d^3 = 1$, so the Wick contraction count is:

$$\binom{5}{2}\binom{3}{1}\binom{2}{2}\binom{4}{1}\binom{3}{2}\binom{1}{1} = 360.$$

Table 4.5 reports the canonicalised expression count for a selection of operators. Note here that R tends to be smaller when the operator term count is large.

Operator	Operator Term Count	Wick Contraction Count	Uncanonicalised Expession Count	Canonicalised Expression Count (R)
Proton	1	2	2	2 (1)
Dinucleon I	1	36	36	28 (0.78)
Dinucleon II	2	36	144	37 (0.26)
Triton I	1	5,760	5,760	2,592 (0.45)
3 Protons	6	4,320	155,520	20,550 (0.13)
Helium-3 I	1	2,880	2,880	1,312 (0.46)
Helium-3 II	6	2,880	103,680	37,664 (0.36)
Helium-4 I	1	518,400	518,400	256,832 (0.495)

Table 4.6.: Nuclear correlator tensor expression statistics assuming isospin symmetry, and with 'half' hadron blocks. Wick contraction count is given by Eqn. 4.11. Canonicalisation is performed via nauty [48]. Triton I (3 Protons) are constructed via appropriate nucleon substitution of Helium-3 I (Helium-3 II) operators. The '3 Protons' correlator vanishes (non-trivially) by the Pauli Exclusion Principle.

With 'half nucleon' blocks, the Wick contraction count is given by[4]:

$$2^A \prod_{q=u,d} \binom{n_q}{N_q^1}\binom{n_q - N_q^1}{N_q^2} \cdots \binom{n_q - N_q^1 - \cdots - N_q^{A-1}}{N_q^A}, \tag{4.11}$$

which is a factor of 2^A greater than the contraction count for hadron blocks in Eqn. 4.10. Table 4.6 reports the canonicalised expression count for a selection of operators. Note that R is significantly lower in Table 4.6 than for regular hadron blocks, as reported in

[4]This expression is of course equivalent to $\prod_{q=u,d} n_q!$, but the reported form gives a clearer picture of the actual computational process.

Table 4.5, indicating that 'half nucleon' blocks provide computational benefit beyond the reduced cost to compute the block expressions.

Chapter 5.

Factor Trees

This chapter will present a novel method for computing nuclear correlators through the construction of *factor trees*, which optimally store a 'factorised' gauge configuration independent form of the correlator's tensor expressions. § 5.1 will explore a selection of the permutation symmetry properties of nuclear correlators to motivate the construction of *abstract factor trees* in § 5.2. *Linearised factor trees*, as explored in § 5.3, are an efficient representation of abstract factor trees that are optimised for evaluation using modern CPU architectures. § 5.4 presents a selection of benchmarks of evaluation using linearised factor trees in comparison to the hadron block method as explored in Chapter 3.

5.1. Empirical Symmetry of Nuclear Correlators

Under the hadron block construction for nuclear correlators, the number of independent components of the gauge configuration dependent tensors, $\left\{f^h_{\vec{p}}\right\}$, remains constant[1], while the number of combining operations grows exponentially in hadron number. A natural empirical question arises — how is the re-use of $\left\{f^h_{\vec{p}}\right\}$ components distributed?

[1]The Pauli Exclusion Principle does necessitate that multi-hadron states are constructed with an increasing number of distinct hadron definite momenta, which increases the number of independent components of $\left\{f^h_{\vec{p}}\right\}$ at a rate not relevant to the discussion here.

Begin with the general form for nuclear correlators of A baryons:

$$C = \sum_{k=1}^{N_w} w_k \epsilon_{a_1 b_1 c_1} \dots \epsilon_{a_A b_A c_A} \Gamma^{(1,k)}_{\alpha_1 \beta_1} \dots \Gamma^{(N_\Gamma,k)}_{\alpha_{N_\Gamma} \beta_{N_\Gamma}} f^{h_1}_{\vec{p}_{(1,k)}} \left(x'_{(1,k)}, \vec{\xi}_{(1,k)} \right) \dots f^{h_A}_{\vec{p}_{(A,k)}} \left(x'_{(A,k)}, \vec{\xi}_{(A,k)} \right), \tag{5.1}$$

where $w_k \in \mathbb{C}$, N_Γ is the number of gamma matrices subject to $A \leq N_\Gamma \leq 2A$, N_w is the number of terms in the contracted expression for C, $\left\{ f^h_{\vec{p}}(x', \vec{\xi}) \right\}$ are the sink-momentum projected hadron block functions as constructed in Chapter 3, $\{h_i\}$ are baryons determined from the nuclear interpolating operator $\mathscr{O}$, $\left\{ x'_{i,k} \right\}$ are the (possibly different) source spacetime points for the k^{th} term in the correlator, and $\left\{ \vec{\xi}_{(i,k)} \right\}$ are functions of $\{a_i\}$, $\{b_i\}$, $\{c_i\}$, $\{\alpha_i\}$, $\{\beta_i\}$ so that $\left\{ f^h_{\vec{p}}(x', \vec{\xi}) \right\}$ are fully contracted with $\{\epsilon_{abc}\} \cup \{\Gamma_{\alpha\beta}\}$. When all internal indices $\{a_i\}$, $\{b_i\}$, $\{c_i\}$, $\{\alpha_i\}$, $\{\beta_i\}$ are summed over, one is left with strings of $f^h_{\vec{q}}(\widetilde{x}', \vec{\eta})$ factors:

$$C = \sum_{k=1}^{\widetilde{N}_w} \widetilde{w}_k f^{h_1}_{\vec{q}_{(1,k)}} \left(\widetilde{x}'_{(1,k)}, \vec{\eta}_{(1,k)} \right) \dots f^{h_A}_{\vec{q}_{(A,k)}} \left(\widetilde{x}'_{(A,k)}, \vec{\eta}_{(A,k)} \right), \tag{5.2}$$

where $\vec{\eta}_{i,k}$ are bundles of fixed colour/spinor values (note the notation change $\vec{\xi} \to \vec{\eta}$ to differentiate contracted and fixed index bundles), $\widetilde{w}_k$ are the new weights after expansion, $\widetilde{N}_w$ is the number of terms in the expanded expression for C, and $\left\{ \widetilde{x}'_{i,k} \right\}$ ($\{\vec{q}_{i,k}\}$) are the source spacetime points (sink momenta) of the expanded expression (noting that $\widetilde{x}'_{i,k} \neq x'_{i,k}$ ($\vec{q}_{i,k} \neq \vec{p}_{i,k}$) in general). For example, the correlator for dinucleon I has the $k = 1$ term for Eqn. 5.2 given by:

$$\begin{aligned} & f^P_{\vec{p}=\vec{0}}(x', a' = 1, \beta' = 1, b' = 2, \gamma' = 2, c' = 3, \alpha' = 1)_{\alpha=1} \times \\ & \quad \times f^N_{\vec{p}=\vec{0}}(x', a' = 1, \beta' = 1, b' = 2, \gamma' = 2, c' = 3, \alpha' = 2)_{\alpha=2}. \end{aligned} \tag{5.3}$$

In order to investigate the distribution of strings of $f^h_{\vec{q}}(\widetilde{x}', \vec{\eta})$ factors, canonically order the factors in Eqn. 5.2 using lexicographic order on $(h, \vec{q}, \widetilde{x}', \vec{\eta})$ tuples, and assign multiplicities to identical terms. Figure 5.1 plots these multiplicities for dinucleon I/II and helium-3 I/II. The power-law relationship as depicted in Figure 5.1 suggests that computational savings could be made by computing each degenerate term once, and adjusting the coefficients by multiplicity. Factors trees, as developed throughout the remainder of this chapter, provide an efficient structure to compute correlators by exploiting this symmetry.

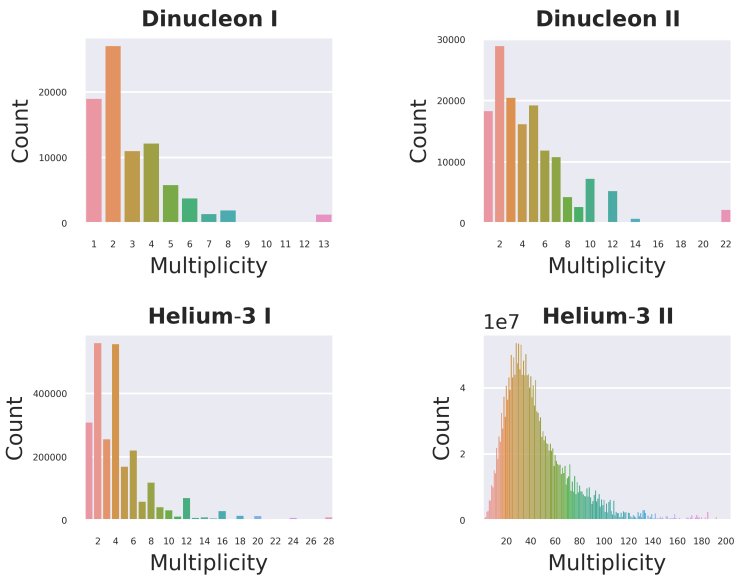


Figure 5.1.: Multiplicity histograms for dinucleon I/II and helium-3 I/II operators, where multiplicities are computed as the number of identical terms in the expansion of Eq. (5.1) after canonical ordering of terms.

5.2. Abstract Factor Trees

In order to walk through some of the developmental details for abstract factor trees as well as provide a basis for their computation, begin with the general expression for a nuclear correlator using hadron blocks after summation as in Eqn. 5.2, and arrange the hadron block factors in an $\widetilde{N}_w \times A$ matrix with weights $\widetilde{w}_k$ annotated on the right-hand side:

$$\begin{bmatrix} f^{h_1}_{\vec{q}_{(1,1)}}\left(\widetilde{x}'_{(1,1)}, \vec{\eta}_{(1,1)}\right) & \cdots & f^{h_A}_{\vec{q}_{(A,1)}}\left(\widetilde{x}'_{(A,1)}, \vec{\eta}_{(A,1)}\right) \\ \vdots & & \vdots \\ f^{h_1}_{\vec{q}_{(1,\widetilde{N}_w)}}\left(\widetilde{x}'_{(1,\widetilde{N}_w)}, \vec{\eta}_{(1,\widetilde{N}_w)}\right) & \cdots & f^{h_A}_{\vec{q}_{(A,\widetilde{N}_w)}}\left(\widetilde{x}'_{(A,\widetilde{N}_w)}, \vec{\eta}_{(A,\widetilde{N}_w)}\right) \end{bmatrix} \begin{matrix} \widetilde{w}_1 \\ \vdots \\ \widetilde{w}_{\widetilde{N}_w}. \end{matrix} \tag{5.4}$$

Next, impose a canonical ordering on rows of the matrix in Eqn. 5.4 by sorting factors according to lexicographic order on $\left(h_i, \vec{q}_{(i,k)}, \widetilde{x}'_{(i,k)}, \vec{\eta}_{(i,k)}\right)$ tuples. Denote by $\sigma_k \in S_A$

the permutation induced by this sorting, so that the matrix in Eqn. 5.4 takes the form:

$$\begin{bmatrix} f^{h_{\sigma_1(1)}}_{\vec{q}_{(\sigma_1(1),1)}}\left(\widetilde{x}'_{(\sigma_1(1),1)},\vec{\eta}_{(\sigma_1(1),1)}\right) & \cdots & f^{h_{\sigma_1(A)}}_{\vec{q}_{(\sigma_1(A),1)}}\left(\widetilde{x}'_{(\sigma_1(A),1)},\vec{\eta}_{(\sigma_1(A),1)}\right) \\ \vdots & & \vdots \\ f^{h_{\sigma_{\widetilde{N}_w}(1)}}_{\vec{q}_{(\sigma_{\widetilde{N}_w}(1),\widetilde{N}_w)}}\left(\widetilde{x}'_{(\sigma_{\widetilde{N}_w}(1),\widetilde{N}_w)},\vec{\eta}_{(\sigma_{\widetilde{N}_w}(1),\widetilde{N}_w)}\right) & \cdots & f^{h_{\sigma_{\widetilde{N}_w}(A)}}_{\vec{q}_{(\sigma_{\widetilde{N}_w}(A),\widetilde{N}_w)}}\left(\widetilde{x}'_{(\sigma_{\widetilde{N}_w}(A),\widetilde{N}_w)},\vec{\eta}_{(\sigma_{\widetilde{N}_w}(A),\widetilde{N}_w)}\right) \end{bmatrix} \begin{matrix} \widetilde{w}_1 \\ \vdots \\ \widetilde{w}_{\widetilde{N}_w}. \end{matrix} \tag{5.5}$$

Next, sort the rows of Eqn. 5.5 by recursive lexicographic order (i.e. the order imposed by comparing each element lexicographically in turn) on

$$\left(\left(h_i,\vec{q}_{(i,k)},\widetilde{x}'_{(i,k)},\vec{\eta}_{(i,k)}\right) \mid k=1,\ldots,\widetilde{N}_w\right) \tag{5.6}$$

tuples so that the identical factors are maximally vertically aligned. Denote by $\rho \in S_{\widetilde{N}_w}$ the permutation induced by this sorting, so that the matrix in Eqn. 5.5 takes the form:

$$\begin{bmatrix} f^{h_{\sigma_{\rho(1)}(1)}}_{\vec{q}_{(\sigma_{\rho(1)}(1),\rho(1))}}\left(\widetilde{x}'_{(\sigma_{\rho(1)}(1),\rho(1))},\vec{\eta}_{(\sigma_{\rho(1)}(1),\rho(1))}\right) & \cdots & f^{h_{\sigma_{\rho(1)}(A)}}_{\vec{q}_{(\sigma_{\rho(1)}(A),\rho(1))}}\left(\widetilde{x}'_{(\sigma_{\rho(1)}(A),\rho(1))},\vec{\eta}_{(\sigma_{\rho(1)}(A),\rho(1))}\right) \\ \vdots & & \vdots \\ f^{h_{\sigma_{\rho(\widetilde{N}_w)}(1)}}_{\vec{q}_{(\sigma_{\rho(\widetilde{N}_w)}(1),\rho(\widetilde{N}_w))}}\left(\widetilde{x}'_{(\sigma_{\rho(\widetilde{N}_w)}(1),\rho(\widetilde{N}_w))},\vec{\eta}_{(\sigma_{\rho(\widetilde{N}_w)}(1),\rho(\widetilde{N}_w))}\right) & \cdots & f^{h_{\sigma_{\rho(\widetilde{N}_w)}(A)}}_{\vec{q}_{(\sigma_{\rho(\widetilde{N}_w)}(A),\rho(\widetilde{N}_w))}}\left(\widetilde{x}'_{(\sigma_{\rho(\widetilde{N}_w)}(A),\rho(\widetilde{N}_w))},\vec{\eta}_{(\sigma_{\rho(\widetilde{N}_w)}(A),\rho(\widetilde{N}_w))}\right) \end{bmatrix} \begin{matrix} \widetilde{w}_{\rho(1)} \\ \vdots \\ \widetilde{w}_{\rho(\widetilde{N}_w)}. \end{matrix} \tag{5.7}$$

Next, omit any string of factors beginning in the first column that is repeated vertically and introduce parent-child arrows between factors that are multiplied to arrive at the *abstract factor tree* for the correlator C. As an example of the process from Eqns. 5.4 – 5.7, consider the following three terms from the correlator for helium-3 II:

$$\begin{aligned} C_3 :=\ & f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 f^{P_-}_{\vec{p}}(x',1,1,1,2,2,1)_1 f^{N_+}_{\vec{p}}(x',2,1,3,2,3,2)_2 \\ & + f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 f^{P_-}_{\vec{p}}(x',2,1,1,2,2,1)_1 f^{N_+}_{\vec{p}}(x',1,2,3,2,3,3)_2 \\ & - f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 f^{N_+}_{\vec{p}}(x',1,1,3,2,3,2)_2 f^{P_-}_{\vec{p}}(x',2,1,1,2,2,1)_1. \end{aligned} \tag{5.8}$$

In the case of C_3, Eqn. 5.4 takes the form:

$$\begin{bmatrix} f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 & f^{P_-}_{\vec{p}}(x',1,1,1,2,2,1)_1 & f^{N_+}_{\vec{p}}(x',2,1,3,2,3,2)_2 \\ f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 & f^{P_-}_{\vec{p}}(x',2,1,1,2,2,1)_1 & f^{N_+}_{\vec{p}}(x',1,2,3,2,3,3)_2 \\ f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 & f^{N_+}_{\vec{p}}(x',1,1,3,2,3,2)_2 & f^{P_-}_{\vec{p}}(x',2,1,1,2,2,1)_1 \end{bmatrix} \begin{matrix} 1 \\ 1 \\ -1. \end{matrix} \tag{5.9}$$

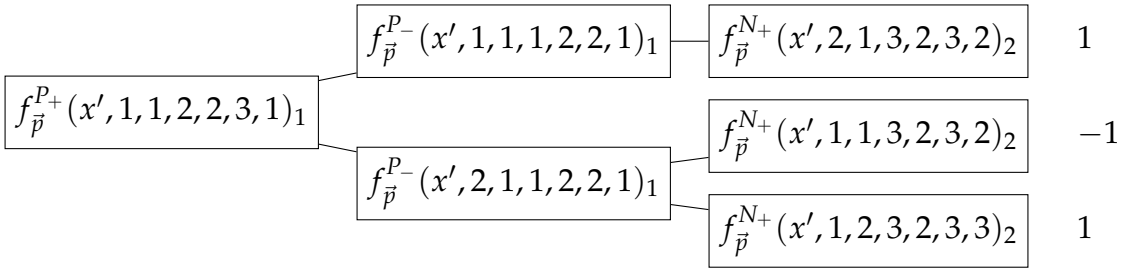


Figure 5.2.: Abstract factor tree for C_3 in Eqn. 5.8 after construction in Eqns. 5.9 – 5.12.

Sorting the factors row-wise in Eqn. 5.9 with hadron order $P_+ < P_- < N_+ < N_-$ swaps the second and third elements of the third row, resulting in:

$$\begin{bmatrix} f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',1,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',2,1,3,2,3,2)_2 & 1 \\ f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',1,2,3,2,3,3)_2 & 1 \\ f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',1,1,3,2,3,2)_2 & -1. \end{bmatrix} \tag{5.10}$$

Next, sort the rows in Eqn. 5.10 by interchanging the second and third rows:

$$\begin{bmatrix} f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',1,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',2,1,3,2,3,2)_2 & 1 \\ f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',1,1,3,2,3,2)_2 & -1 \\ f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',1,2,3,2,3,3)_2 & 1. \end{bmatrix} \tag{5.11}$$

Omitting repeated factor strings yields:

$$\begin{bmatrix} f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 & f_{\vec{p}}^{P_-}(x',1,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',2,1,3,2,3,2)_2 & 1 \\ & f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 & f_{\vec{p}}^{N_+}(x',1,1,3,2,3,2)_2 & -1 \\ & & f_{\vec{p}}^{N_+}(x',1,2,3,2,3,3)_2 & 1, \end{bmatrix} \tag{5.12}$$

so that one may now form the abstract factor tree for C_3 as depicted in Figure 5.2. To evaluate Figure 5.2, one traverses the tree in depth-first order, which corresponds in

the case of C_3 to:

$$f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 \to f_{\vec{p}}^{P_-}(x',1,1,1,2,2,1)_1 \to f_{\vec{p}}^{N_+}(x',2,1,3,2,3,2)_2 \to$$
$$\to f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 \to f_{\vec{p}}^{N_+}(x',1,1,3,2,3,2)_2 \to f_{\vec{p}}^{N_+}(x',1,2,3,2,3,3)_2.$$

To evaluate an abstract factor tree, one follows a procedure where upon right traversal, one multiplies by the encountered factor, and upon left traversal, one divides by the encountered factor[2]. Upon reaching a leaf, one multiplies by the annotated coefficient and adds the cumulative sum to the overall total. In the case of C_3, this amounts to computing the correlator in the factorised form:

$$f_{\vec{p}}^{P_+}(x',1,1,2,2,3,1)_1 \left[f_{\vec{p}}^{P_-}(x',1,1,1,2,2,1)_1 f_{\vec{p}}^{N_+}(x',2,1,3,2,3,2)_2 + \right.$$
$$\left. + f_{\vec{p}}^{P_-}(x',2,1,1,2,2,1)_1 \left(-f_{\vec{p}}^{N_+}(x',1,1,3,2,3,2)_2 + f_{\vec{p}}^{N_+}(x',1,2,3,2,3,3)_2 \right) \right]. \quad (5.13)$$

A natural way to represent an abstract factor tree that corresponds to the tensor expression C_3 as in Figure 5.2 on a computer might be to represent each node as a structure and to form links between the nodes using pointers. This would be highly inefficient in two senses: first, it doesn't exploit the predictable access pattern (depth-first traversal) that evaluation of the factor trees entails and would as a result be significantly memory access bound; and second, it would have a large memory footprint, taken up almost entirely by pointers representing *parent* $\to$ *child* relationships. A representation that is much more efficient on both counts is given by a *linearised factor tree*, which is explored in the next section.

5.3. Linearised Factor Trees

As shown in Figure 5.3 for the example in Figure 5.2, a linearised factor tree is represented by three arrays: *factors*, *children*, and *coefficients*. The *factors* array is given by the depth-first traversal of the vertices of the abstract factor tree. The *children* array is given by the number of children of the vertices in depth-first traversal order. The *coefficients* array is the list of leaf coefficients in depth-first traversal order. Note here that one may construct Figure 5.3 directly from Eqn. 5.12 without the intermediate construction of the abstract factor tree in Figure 5.2.

[2] In fact, due to the mechanics of floating-point arithmetic, it's much safer (and more efficient) to return to a previously stored value upon left traversal rather than perform a division.

$$
\begin{aligned}
\mathit{factors} &= \Big[f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1, f^{P_-}_{\vec{p}}(x',1,1,1,2,2,1)_1, f^{N_+}_{\vec{p}}(x',2,1,3,2,3,2)_2, \\
&\qquad f^{P_-}_{\vec{p}}(x',2,1,1,2,2,1)_1, f^{N_+}_{\vec{p}}(x',1,1,3,2,3,2)_2, f^{N_+}_{\vec{p}}(x',1,2,3,2,3,3)_2 \Big] \\
\mathit{children} &= [2,1,0,2,0,0] \\
\mathit{coefficients} &= [1,-1,1]
\end{aligned}
$$

Figure 5.3.: Linearised factor tree representation for abstract factor tree for C_3 as depicted in Figure 5.2.

In order to traverse a linearised factor tree, one must maintain three state variables: *position*, *index*, and *leafIdx*. The *position* array maintains the current path (in terms of locations in the *factors*/*children* array) within the tree, such that if one is located at for example the second leaf node in Figure 5.2, then the path is:

$$
f^{P_+}_{\vec{p}}(x',1,1,2,2,3,1)_1 \to f^{P_-}_{\vec{p}}(x',2,1,1,2,2,1)_1 \to f^{N_+}_{\vec{p}}(x',1,1,3,2,3,2)_2, \tag{5.14}
$$

and hence, $\mathit{position} = [0,3,4]$ (i.e. the positions of those three factors in the *factors* array in Figure 5.3). The *index* array keeps track of the child number (i.e. the order ranking of a node in its parent's children). In Figure 5.2, take again the example of the second leaf, which has $\mathit{index} = [1,0]$ (i.e. the second factor in Eqn. 5.14 is the second child of the first factor; and the third factor is the first child of the second factor). The *leafIdx* keeps track of the current row in the matrix given in Eqn. 5.7 so that the current position along the *coeffs* array is maintained.

Algorithm 2 evaluates a linearised factor tree given a set of *values*, which in this case is the set of hadron block elements $\left\{ f^h_{\vec{p}} \right\}$ for a particular gauge configuration. From Algorithm 2, it can be seen that the number of floating-point operations required to evaluate a factor tree of N_F factors and N_L leaves is $2N_L + N_F$. There are of course additional computational resources required to maintain the traversal state, but the evaluation time scales as $\mathcal{O}(2N_L + N_F)$. A key motivating feature of the linearised factor tree design is its cache efficiency. Since the tree variables are stored in traversal order, the $L1$-cache hit-rate is high, as confirmed using the hardware performance counters as provided by PAPI [54] using the C++ codebase developed in order to obtain the results presented in § 5.4.

Algorithm 2: Linearised Factor Tree Evaluation

```
Input: tree{ factors, children, coeffs }
Output: sum := 0
Data: values
index := [ ]
position := [ tree→factors[0] ]
cumulativeProd := [ values[ position[0] ] ]
leafIdx := 0
while True do
    nextPos := position[-1] + 1
    if tree→children[ position[-1] ] == 0 then
        sum += tree→coeffs[leafIdx] × cumulativeProd[-1]
        leafIdx += 1
        success := false
        while ! index→empty() do
            index[-1] += 1
            if index[-1] < tree→children[ position[-2] ] then
                success = true
                position[-1] = nextPos
                cumulativeProd[-1] = cumulativeProd[-2] ×
                      values[ tree→factors[nextPos] ]
                break
            else
                position = position[:-1]
                index = index[:-1]
                cumulativeProd = cumulativeProd[:-1]
                termValues = termValues[:-1]
        if ! success then
            done
    else
        position.append( nextPos )
        index.append( 0 )
        cumulativeProd.append( cumulativeProd[-1] ×
              values[ tree→factors[nextPos] ] )
```

5.4. Results for Nuclear Correlators

The following presents a set of results of the linearised factor tree method applied to a selection of nuclear correlators. Table 5.1 reports computed factor tree statistics for the nucleon, dinucleon I/II, and Helium-3 I/II operators. Also reported is the number

of operations required to evaluate the corresponding linearised factor tree using Algorithm 2. The ratio N_L/N_F gives an indication of the amount of factorisation the factor tree achieves, where a higher ratio corresponds to more factorisation and hence a larger potential speed-up beyond the inherent speed-up of the degeneracy elimination of the factor tree construction. In the limit of no common factors, $N_L/N_F = 1/A$ for A baryons. Table 5.1 reports the ratio N_L/N_F to be substantially larger than $1/A$ for all cases but $A = 1$.

Operator	N_F	N_L	N_L/N_F	Num. Operations ($N_F + 2N_L$)
Nucleon	25	24	0.96	73
Dinucleon I	42,037	37,104	0.88	116,245
Dinucleon II	53,105	48,096	0.91	149,297
Helium-3 I	945,745	722,688	0.76	2,391,121
Helium-3 II	3,765,835	3,296,184	0.87	10,358,203

Table 5.1.: Factor tree statistics for a selection of nuclear operators. The number of operations is given by $N_F + 2N_L$, where N_F is the number of factors (i.e. nodes) in the tree and N_L is the number of terms (i.e. distinct factor strings) in the tree.

To measure the performance improvement associated with using linearised factor trees as evaluated by Algorithm 2, Figure 5.4 compares wall-clock time for correlator computation (excluding propagator and hadron block calculation) using dinucleon I/II and helium-3 I/II operators. Figure 5.4 demonstrates that factor trees offer between one and two orders of magnitude improvement over unoptimised correlator computations for light nuclei using hadron blocks (as explored in Chapter 3). An up-front computational cost for constructing the factor tree, not included in Figure 5.4, is required and scales with the unoptimised correlator computational cost. The extra calculation, however, is only required before the first configuration is analysed and hence the cost may be amortised over the full set of configurations. Although the factor tree method shows promising speed-ups for small nuclei, it should be noted that the memory used will become prohibitive for large nuclei[3].

[3]Constructing factor trees directly to hard-disk could extend the method to larger nuclei. Using MMap [55] would be a particularly efficient approach, but is left for future work.

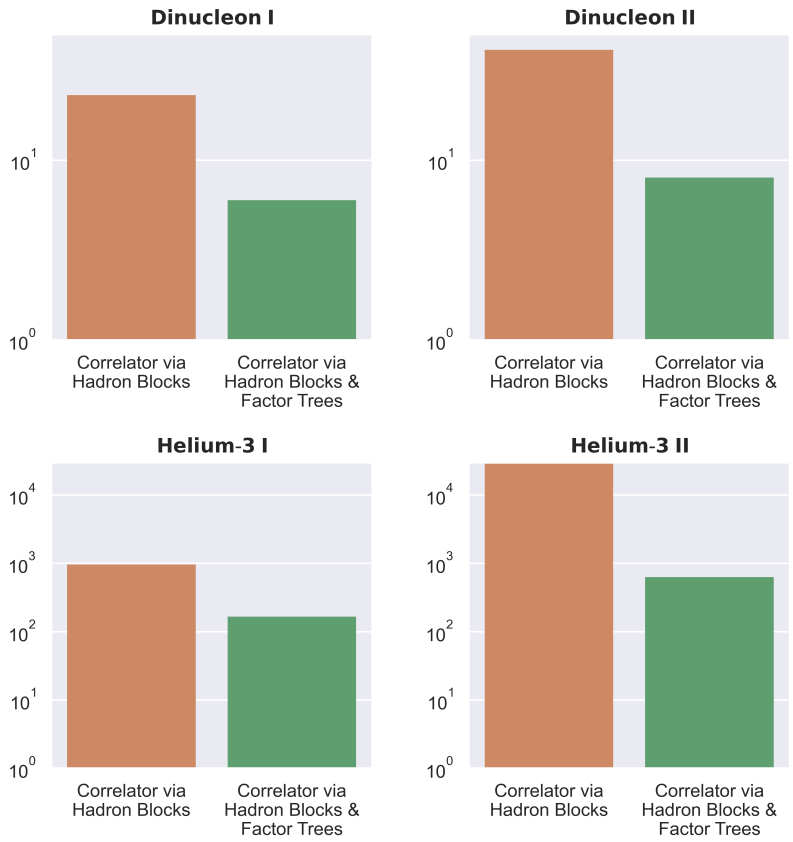


Figure 5.4.: Linearised factor tree benchmark, measuring wall-clock time in milliseconds on a single core of an Intel Xeon Scalable Cascade Lake processor; lattice volume 64^3.

Chapter 6.

Tensor E-graphs

E-graphs (equality graphs) compactly represent and efficiently compute a congruence relation (defined in § 6.2) over a set of expressions. E-graphs were originally developed for automated theorem provers [56, 57], and have seen successful recent application to program optimisation [5], deep learning optimisation [58–60], and improving floating-point accuracy [61].

A key predecessor and core component of e-graphs is the union-find structure as explored in § 6.1. The generic e-graph formulation will be presented in § 6.2, and then § 6.3 will present tensor e-graphs, which is a novel application of the e-graph formulation to tensor expression optimisation. § 6.4 will walk through a selection of the implementation details of tensor e-graphs, and finally § 6.5 will present a set of results for nuclear correlation functions as optimised by tensor e-graphs.

6.1. Equivalence Relations via Union-Find Structures

One of the core components of an e-graph is a *union-find data structure*, as originally devised in Ref. [51], which efficiently computes and represents an equivalence relation (defined below) on a set [51]. The following will explore the process as developed in Ref. [51] and then § 6.2 will apply it to the process of building e-graphs.

Definition (Equivalence Relation)**.** Let X be a set, and define an *equivalence relation* $\sim$ to be a binary relation on X such that $\forall x, y, z \in X$:

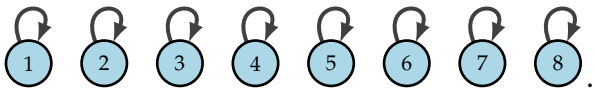
(a) $x \sim x$ (Reflexivity)

(b) $x \sim y$ if and only if $y \sim x$ (Symmetry)

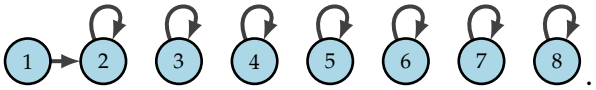
(c) if $x \sim y$ and $y \sim z$, then $x \sim z$ (Transitivity)

Further define the *equivalence class* of $x \in X$ under $\sim$ to be $[x] \coloneqq \{y \in X \mid y \sim x\}$. The set of all equivalence classes partition X (i.e. $X = \bigsqcup_{[x]} [x]$ where $\bigsqcup$ denotes disjoint union).

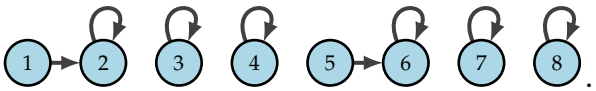
A union-find structure consists of a node for each element in X such that each node points to exactly one node (including possibly itself). Initially, all nodes point to themselves, and represent the the trivial relation where no distinct elements are equivalent. To query whether two elements are members of the same equivalence class, one follows the links of both nodes until they either reach the same node, or terminate at different nodes. To *union* two equivalence classes, one finds the representative nodes (nodes with terminating links), which are the same for each element of the equivalence class, and adjusts the pointer of one to the other. To provide a worked example, consider $X = \{1, \ldots, 8\}$, and construct the initial state with no equivalent elements as:



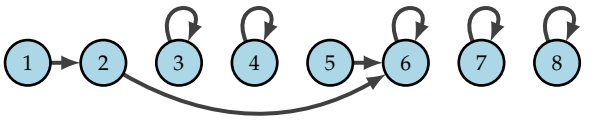
Suppose that $1 \sim 2$, so find the representative nodes of the equivalence classes containing the elements 1 and 2, which are the nodes 1 and 2 respectively, and connect one to the other:



Next suppose $5 \sim 6$, so follow the links and connect $5 \to 6$:

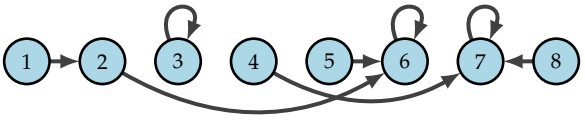


Next suppose $1 \sim 5$, so traverse the chain beginning at 1 to arrive at 2 and traverse the chain beginning at 5 to arrive at 6, and then connect 2 to 6:



.

Finally, suppose $4 \sim 7 \sim 8$, so select any element (choose 7 for example) to follow to find the representative, and then connect the other nodes (4 and 8) to this representative element:



,

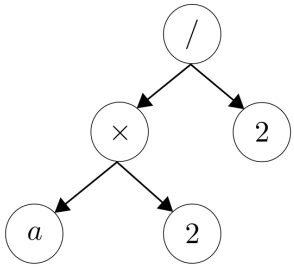
which represents the set of equivalence classes given by:

$$\{[3],[6],[7]\} = \{\{3\},\{1,2,5,6\},\{4,7,8\}\}. \tag{6.1}$$

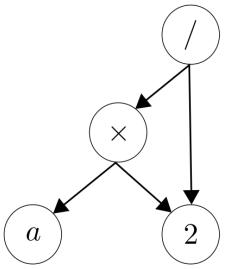
There are many benefits to this computational approach to equivalence relations, but in particular to add a single element, one doesn't need to compare each element against all other elements in X, but only find a single equivalent element. The process isn't linear, since one has to follow the links, but it's asymptotically approximately linear (see Ref. [51] for the precise asymptotic bounds). The next section will explore e-graphs, which rely heavily on this property.

6.2. Congruence Relations via E-graphs

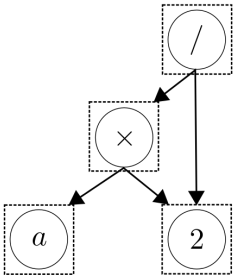
Intuitively, a *congruence relation* is a binary relation $\equiv_f$ defined by a function f on an algebraic space such that $a \equiv_f b$ if and only if $f(a) = f(b)$. The formal definition will vary depending on the particular algebraic space. In the context of an e-graph, a and b are directed acyclic graphs (DAGs) of *e-nodes* (equality nodes), as defined below, representing expressions and f is the evaluation of those expressions. A congruence relation is by necessity an equivalence relation, so is symmetric (i.e. $a \equiv_f b \Leftrightarrow b \equiv_f a$), but in order to give directionality to e-graph construction, that symmetry is broken through the introduction of a *re-write rule*. A re-write rule is a procedure where for a



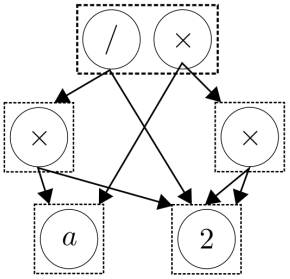
(a) DAG.



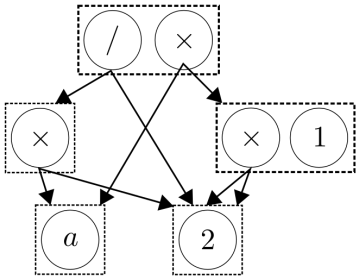
(b) DAG with hash-consing.



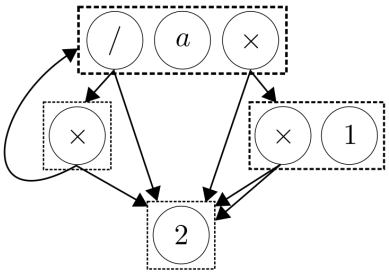
(c) E-graph.



(d) E-graph after $(x \times y)/z \rightarrow x \times (y/z)$ re-write.



(e) E-graph after $x/x \rightarrow 1$ re-write.

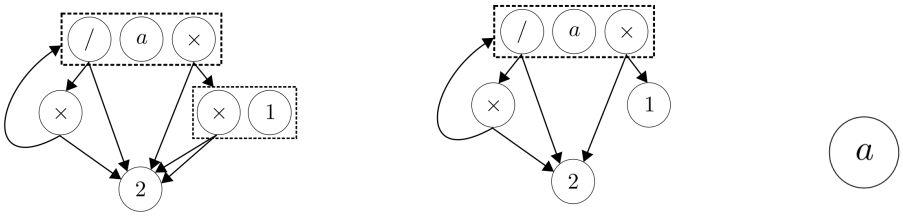


(f) E-graph after $x \times 1 \rightarrow x$ re-write.

Figure 6.1.: E-graph construction for expression $(a \times 2)/2$, where e-nodes are represented by circles and e-classes are represented by dashed rectangles.

subset of pairs related by the congruence relation $a \equiv_f b$, one selects either $a \mapsto b$ or $b \mapsto a$.

Definition (DAG)**.** A *Directed Acyclic Graph* (DAG) is a graph where edges are directional (i.e. they have arrows) and no contiguous path of arrows may be followed to return to the same node (i.e. there are no loops). Note that an e-graph is not a DAG due to hash consing.



(a) E-graph after (trivial) e-node selection in single e-node e-classes.

(b) E-graph after lower right e-class selection.

(c) E-graph after final e-class selection.

Figure 6.2.: E-graph extraction for e-graph as constructed in Figure 6.1. E-classes before selection are depicted by dashed rectangles.

The general procedure is that an e-graph is constructed by successive application of the re-write rules, and then an optimised expression is *extracted* using a supplied cost function. To elucidate this procedure before the formal definition, consider the worked example used throughout Ref. [5] given by the simple expression $(a \times 2)/2$, which may be represented by the DAG in Figure 6.1 (a). A key feature of e-graphs is *hash consing*, whereby sub-graphs are maximally re-used as shown in Figure 6.1 (b). Note that sub-graph identification is performed on the basis of e-nodes, which includes both the nodes and their emanating edges (e.g. $a \times 4$ wouldn't re-use the $\times$ node but would re-use the a node). In order to perform re-writes, the notion of *e-classes* (equivalence classes) is introduced so that all e-nodes within an e-class are congruent. Figure 6.1 (c) depicts e-nodes within e-classes, noting that edges are from e-nodes to e-classes so that re-writes can be cascaded. Figure 6.1 (d) – (f) depicts the e-graph after successive re-writes have been applied.

Figure 6.2 depicts the extraction process of an optimised expression from the e-graph as constructed in Figure 6.1. Extraction is performed by recursively selecting the e-node within each e-class that has the minimum number of sub-DAG nodes. Figure 6.2 (a) shows that selection within single e-node e-classes is trivial. Figure 6.2 (b) – (c) depicts the selection of childless e-nodes to extract the optimal expression $(a \times 2)/2 = a$. Of course, this simple example does little to demonstrate the power of e-graphs, which presents itself where there are many competing and interacting optimisation paths.

Definition (E-graph)**.** Formally, an *e-graph* is a tuple (U, M, H) where:

- U is a union-find data structure (see § 6.1) storing an equivalence relation over *e-class ids* (an *e-class* is a set of equivalent *e-nodes*; an *e-node* is a function symbol paired with a list of children e-classes),
- M is a map from *e-class ids* to *e-classes*; all equivalent *e-class ids* map to the same *e-class*,
- H is a hash-cons (see Ref. [5]) mapping *e-nodes* to *e-class ids*.

A full set of optimised algorithms for working with e-graphs in this formal representation can be found in Ref. [5].

6.3. Tensor E-graphs

The following will introduce *tensor e-graphs*, which is an e-graph variant whereby each e-node is a tensor expression E and the initial e-graph is the set of tensor expressions representing the correlator C. The full tensor e-graph is built by repeatedly applying to each e-node E the re-write rule given in Figure 6.3 to produce candidate common subexpressions $\widetilde{E}_1$, $\widetilde{E}_2$.

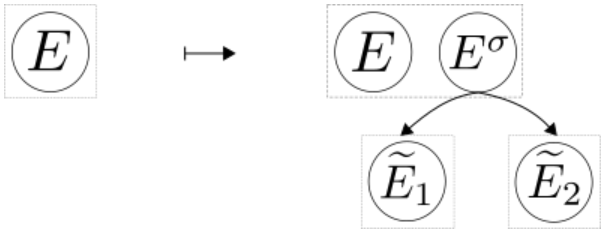


Figure 6.3.: Tensor e-graph re-write rule for a tensor expression E, with $\sigma \in S^{\pm}_{|E|}$ (signed symmetric group[1]) permuting the index slots of E through action $E^\sigma = E_1 E_2$ for subexpressions E_1, E_2 related to child e-nodes through tensor expression canonicalisation $E_k \to \widetilde{E}_k$ (see Chapter 4). E-nodes are notated by circles and e-classes are notated by dashed rectangles.

In order to restrict the rather large class of possible re-writes $E \xrightarrow{\sigma} E^\sigma = E_1 E_2$ to sub-expressions that are likely to result in the identification of redundant operations, introduce two conditions:

1. Both E_1 and E_2 must contain at least one summed index

[1] $S^{\pm}_{|E|}$ is defined by the external wreath product $\mathbb{Z}_2 \wr S_{|E|}$ for symmetric group $S_{|E|}$; see Ref. [62].

2. At least one of E_1 or E_2 must contain at least one fully summed tensor.

For example, take $E = \epsilon_{ijk}\epsilon_{lmn}T_{ijl}T_{nmk}$ for any rank-3 tensor T and consider the sub-expressions $E_1 = \epsilon_{ijk}T_{ijl}$ and $E_2 = \epsilon_{lmn}T_{nmk}$. The canonical forms of these sub-expressions are $\widetilde{E}_1 = \epsilon_{ijk}T_{ijl} = \widetilde{E}_2$, noting that $E_2 \to \widetilde{E}_2$ introduces a relative minus sign from the exchange of anti-symmetric indices, $\epsilon_{lmn} = -\epsilon_{nml}$. The signed permutation σ keeps track of both the rearrangement of tensors $\epsilon\epsilon TT \to \epsilon T\epsilon T$ and the relative sign induced by the tensor expression canonicalisation process. In this case, $\sigma = -(4\,7)(5\,8)(6\,9)(7\,9)$ using (signed) cycle notation. This re-write, as depicted in Figure 6.4, can be used to extract a common sub-expression $B_{kl} = \epsilon_{ijk}T_{ijl}$ so that by permuting the index slots of E by σ (i.e. $ijklmnijlnmk \to -ijkijlnmlnmk$) and discarding the index slots summed in B, one may express $E = -B_{kl}B_{lk}$.

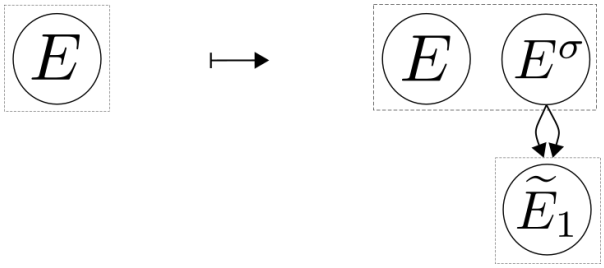


Figure 6.4.: Tensor e-graph re-write rule for tensor expression $E = \epsilon_{ijk}\epsilon_{lmn}T_{ijl}T_{nmk}$ with a rank-3 tensor T. Common sub-expression $\widetilde{E}_1 = \epsilon_{ijk}T_{ijl}$ only appears once in the e-graph, enabling its re-use.

A tensor e-graph is constructed by repeated application of the re-write rule until the *saturation limit*, where no further non-redundant re-writes may be performed, is reached. Often, it is not computationally feasible to reach the saturation limit, and the tensor e-graph construction process terminates after a set e-node limit has been reached, or Algorithm 3 is employed.

After a tensor e-graph has been constructed, it remains to extract the optimal sub-expression decomposition scheme (i.e. the desired computational strategy). One could calculate the number of evaluation operations for each possible sub-expression decomposition scheme, but the computational resources required is formally unbounded (due to loops) and in practice far too large even for correlators of light nuclei. In order to obtain an approximately optimal sub-expression decomposition scheme, consider the cost function parameterised by $(\mathcal{P}, \mathcal{S})$ for an e-node of sub-expression E:

$$cost(E\,; \mathcal{P}, \mathcal{S}) = -N_e \max\left\{N_p - \mathcal{P} - \mathcal{S}N_s\right\}, \tag{6.2}$$

where N_e is the number of operations to evaluate E, N_p is the number of e-node parents from distinct e-classes, N_s is the number of external spinor indices of E, $\mathcal{P}$ is the *parent penalty*, and $\mathcal{S}$ is the *spinor penalty*. To provide some justification for this choice of cost function, consider the tensor e-graph fragment depicted in Figure 6.5, whereby the expressions E and F have been re-written to use the common sub-expressions $\widetilde{E}_1$, $\widetilde{E}_2$, $\widetilde{F}_1$, and $\widetilde{F}_2$. From Figure 6.5, this occurs when the number of excess parents from distinct e-classes $(N_p - 1)$ of an e-node is maximised. In order to incentivise further re-use, § 6.5 investigates the use of a generalised $N_p - \mathcal{P}$ term for parameter $\mathcal{P}$ that filters out e-nodes with only $\mathcal{P}$ parents from distinct e-classes. Further, the cost function in Eqn. 6.2 is sensitive to the number of operations required to evaluate a sub-expressions, since the total operation number savings scales as the product of the number of times a sub-expression is re-used and how many operations it takes to evaluate that sub-expressions. Finally, tensor expressions for correlators of non-relativistic operators may have spinor indices ranging over only two values, rather than four. In this case, the extraction process may 'over-match' a sub-expression with an external spinor index given that the re-use could be using non-overlapping spinor index values. To compensate for this issue, the cost function in Eqn. 6.2 introduces an optional factor $\mathcal{S}$ that penalises any sub-expression with external spinor indices by an amount controlled by $\mathcal{S}$. For relativistic operators, one would assume that $\mathcal{S} = 0$ produces the best results.

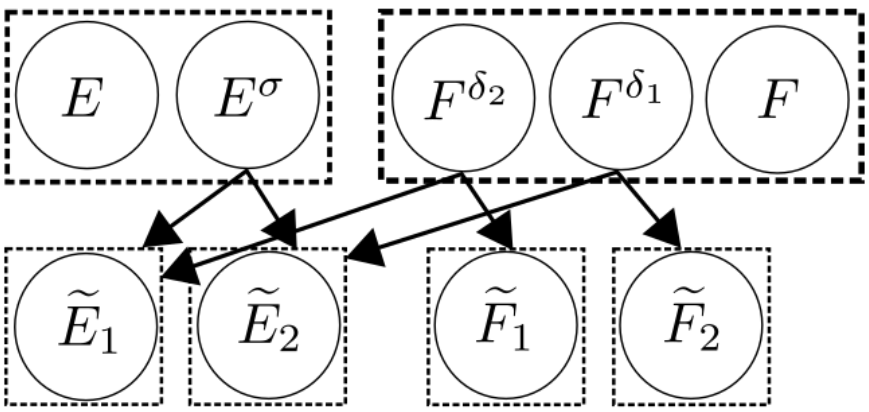


Figure 6.5.: Example e-graph fragment consisting of tensor expressions E and F.

6.4. Tensor E-graph Implementation

The basis for the exploration of e-graphs is Ref. [5] and its corresponding open-source software package written in the programming language Rust. Ref. [5] presents algorithmic improvements over the traditional formulation of e-graphs in the context of re-write driven compiler optimisations and program synthesis, but the software package is extensible to many applications of e-graphs. An implementation of tensor e-graphs has been written in order to obtain the results in § 6.5 using the Rust programming language and linked to the existing C++ codebase using FFI (foreign function interface). Initially, the implementation was written on top of the codebase of Ref. [5], however it was later found that a direct implementation based on the same design could produce higher performance tensor e-graph constructions. It is the latter approach that was used to obtain the results in § 6.5.

It is typically infeasible to construct tensor e-graphs to the saturation limit for all but the lightest nuclei due to computational resource constraints (CPU time and memory). Using the conventional e-graph algorithm, a restriction such as an e-node number limit is enforced so that the e-graph construction process terminates after a set amount of computational resources has been exhausted. Since it is difficult to predict the e-graph size (measured in the number of e-nodes) required to obtain a certain quality of optimised expression, the following proposes an extension to the e-graph algorithm referred to as the *iterated e-graph algorithm* (Algorithm 3).

In particular, Algorithm 3 would allow 'interactive' construction of e-graphs, where a decision of computational resource investment could be made on the basis of the cost function derivative, for example. It is also essential for efficiently obtaining the results exploring $(\mathcal{P}, \mathcal{S})$ space in § 6.5.

6.5. Results for Nuclear Correlators

The following explores a few basic results for the application of tensor e-graphs to nuclear correlators. In particular, to provide a concrete example of the output of the tensor e-graph process, consider the e-graph built from the dinucleon II operator until the saturation limit. The extraction process using the cost function in Eqn 6.2 with $\mathcal{P} = 1$ and $\mathcal{S} = 0$ (since dinucleon II is relativistic) yields for example the following

Algorithm 3: Iterated E-graph Algorithm (Pseudo-code)

```
Input: Initial expression E
Output: Optimal extracted expression Ê
Data: Nodes per generation G_nodes, Max generations G_max, Acceptable cost ĉ
egraph := construct_initial_egraph(E)
next_stop := G_nodes
for i = 1, ..., G_max do
    while True do
        perform_rewrite(egraph)
        if is_saturated(egraph) then
            Ê = extract(egraph)
            stop
        if size(egraph) ≥ next_stop then
            Ê = extract(egraph)
            if cost(Ê) ≤ ĉ then
                stop
            break
    next_stop += G_nodes
```

two common sub-expressions:

$$\mathcal{B}_1(\alpha',\beta',\gamma',\delta') = \epsilon_{abc}\epsilon_{def}(C\gamma_5)_{\alpha\beta}(C\gamma_5)_{\gamma\delta}f^P_{\alpha,\vec{0}}(\vec{0},a,\alpha',d,\beta',b,\gamma')f^N_{\beta,\vec{0}}(\vec{0},e,\gamma,c,\delta',f,\delta), \tag{6.3}$$

$$\mathcal{B}_2(a',b',c',d',e',f') = (C\gamma_5)_{\alpha\beta}(C\gamma_5)_{\gamma\delta}(C\gamma_5)_{\sigma\rho}(C\gamma_5)_{\mu\nu}f^P_{\alpha,\vec{0}}(\vec{0},a',\gamma,b',\sigma,c',\rho)f^N_{\beta,\vec{0}}(\vec{0},d',\mu,e',\delta,f',\nu). \tag{6.4}$$

Note that both $\mathcal{B}_1$ and $\mathcal{B}_2$ satisfy the two re-write restrictions since they both contain the fully summed tensor $(C\gamma_5)_{\alpha\beta}$, for example. The correlator for the dinucleon II operator is optimally computed by first computing $\mathcal{B}_1$ and $\mathcal{B}_2$, and then contracting the remaining indices.

To provide an illustration of some of the generic features of e-graph construction using Algorithm 3, consider the construction of the tensor e-graph for the Helium-3 correlator. By extracting the optimal sub-expression decomposition scheme at each generation, the number of operations to compute the e-graph optimised expressions is plotted in Figure 6.6. A key feature of the tensor e-graph process as shown in Figure

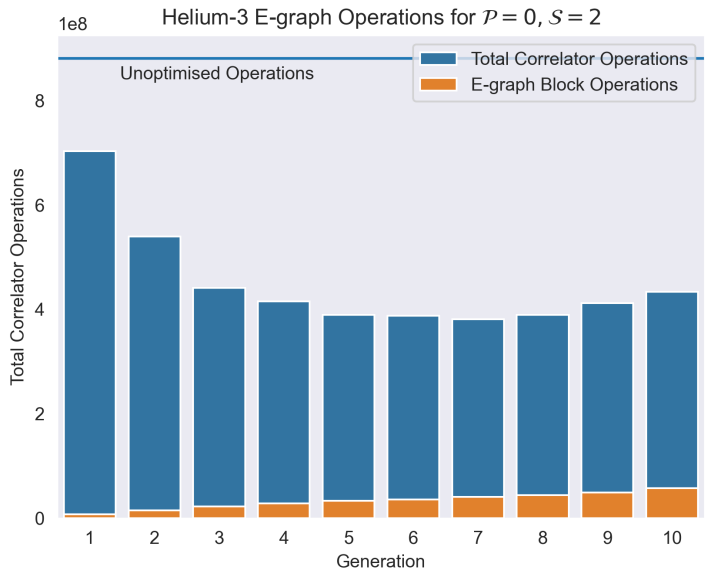


Figure 6.6.: Total correlator operations for Helium-3 as a function of generation using Algorithm 3. Total correlator operations is given by the sum of the number of operations required to compute the e-graph blocks (i.e. the extracted sub-expressions) and the number of operations required to compute the overall correlator using these extracted blocks. The number of operations required to compute the correlator for Helium-3 without e-graphs (i.e. computing the tensor expressions given by the hadron block formulation directly) is shown by the horizontal line as $881,031,168$ operations.

6.6 is that the ratio of e-graph block operations to total correlator operations increases with generation (i.e. with e-graph size) as more re-usable sub-expressions are found.

Figure 6.7 depicts the variation in performance for a selection of values in $(\mathcal{P}, \mathcal{S})$ space. Note here that there are only a limited number of 'reasonable' values for $\mathcal{P}$ and $\mathcal{S}$ without filtering out all potential re-usable sub-expressions. One would expect that the optimal choice of $(\mathcal{P}, \mathcal{S})$ is relatively stable over a wide selection of nuclear correlators, but it is left to future work to demonstrate this rigorously.

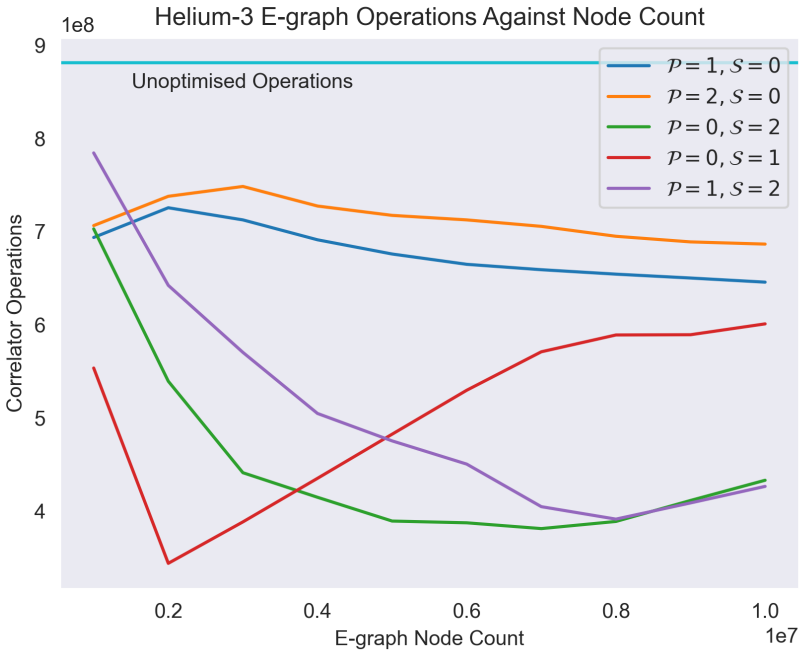


Figure 6.7.: Total correlator operations for Helium-3 as a function of e-graph size at each generation using Algorithm 3 for various $(\mathcal{P}, \mathcal{S})$ values (see Eqn. 6.2). The number of operations required to compute the correlator for Helium-3 without e-graphs (i.e. computing the tensor expressions given by the hadron block formulation directly) is shown by the horizontal line as $881,031,168$ operations.

Chapter 7.

Prospects for Quantum Algorithms

This chapter offers a perspective on the potential applications of quantum combinatorial optimisation to accelerate nuclear correlation function calculations. The purpose is certainly not to provide a complete algorithmic solution, but to provide some modest improvements to the existing literature that may inspire future work. Quantum combinatorial optimisation uses quantum information processing to minimise functions of combinatorial *decision variables*. The reader unfamiliar with quantum information processing is encouraged to consult Refs. [63, 64]. The three primary analog quantum frameworks used to solve optimisation problems are quantum annealing (QA) [65], quantum adiabatic optimisation (QAO) [66], and the quantum approximate optimisation algorithm (QAOA) [67]. All of these frameworks are closely related (see Ref. [68]), and algorithms within one framework can sometimes be formulated in another without redesign.

The approach proposed here is an extension of Chapter 4 that uses a novel QAO algorithm to perform redundant isomorph elimination of tensor expressions (or tensor sub-expressions in the context of tensor e-graphs). § 7.1 provides a brief introduction to quantum adiabatic optimisation and a general approach to formulate classical combinatorial optimisation problems as QAO problems. § 7.2 walks through two formulations of QAO for coloured graph isomorphism, and evaluates their relative strengths and weaknesses. § 7.3 adapts the formulations in § 7.2 to address the additional requirements of tensor expression isomorphism as discussed in Chapter 4. Finally § 7.4 provides a pathway for applications to nuclear correlators, as well as raises some concerns about resourcing and fidelity issues.

7.1. Quantum Adiabatic Optimisation (QAO)

Adiabatic quantum computation has a lineage somewhat distinct from 'circuit model' unitary gate evolution quantum computation, with the first paper of Apolloni, et al. appearing in 1989 [65] (i.e. the same year as Deutsch's 'circuit model' paper [69]). It should be noted that, for example, non-'stoquastic' adiabatic evolution (i.e. Hamiltonians with only non-positive elements in the computational basis) can be simulated in the 'circuit model' with only polynomial overhead [70].

Consider states in a d-dimensional Hilbert space $\mathcal{H}$ with $d < \infty$ and take $H(t)$ to be a slowly varying Hamiltonian. At each fixed t, there are d energy eigenstates of $H(t)$, $\{|E_k(t)\rangle\}_{k=0}^{d-1}$, such that:

$$H(t)\,|E_k(t)\rangle = E_k(t)\,|E_k(t)\rangle\,, \tag{7.1}$$

where $E_0(t) \leq \cdots \leq E_{d-1}(t)$. The quantum adiabatic theorem [70] roughly states that under adiabatic evolution, an initial state $|E_k(0)\rangle$ evolves to the energy eigenstate $|E_k(T)\rangle$ of the 'final' Hamiltonian $H(T)$ in the limit as $T \to \infty$. A more precise statement of this fact can be found in Ref. [70]. One typically chooses $k = 0$ since the convergence rate of an algorithm such as QAO will depend on the energy gap [71]:

$$\min\left\{|E_k(t) - E_{k+1}(t)|, |E_k(t) - E_{k-1}(t)|\right\}. \tag{7.2}$$

Under QAO, one engineers a slowly time-varying Hamiltonian:

$$H(s) = (1-s)H_0 + sH_T, \tag{7.3}$$

where $s : [0,T] \to [0,1]$ is monotonic such that $s(0) = 0$, $s(T) = 1$ (i.e. so that $H(s(0)) = H_0$ and $H(s(T)) = H_T$). H_0 is chosen to be an easily preparable state, and H_T is chosen to have groundstate corresponding to the minimum of a (classical) cost function C. The decision variables of C are encoded into the computational basis states of $\mathcal{H}$ so that the measurement in the computational basis of the prepared state at $s = 1$ allows the argument minimising C to be read-off directly.

7.2. QAO for (Coloured) Graph Isomorphism

As discussed in Chapter 4, coloured graph isomorphism is defined as follows.

Definition (Coloured Graph Isomorphism)**.** Given a vertex set V isomorphic to $[n] \coloneqq \{1, 2, \ldots, n\}$, define *graphs*:

$$\boldsymbol{G}(V) \coloneqq \{\text{labelled simple (unweighted, undirected, single edge, no loops) graphs on } V\},$$

and *ordered partitions* (i.e. colourings of graphs in $\boldsymbol{G}(V)$):

$$\boldsymbol{\Pi}(V) \coloneqq \{[V_1, \ldots, V_r] \mid \dot{\cup}_j V_j = V\} =: \{\textit{ordered partitions of } V\},$$

so that the pair $(G, \pi) \in \boldsymbol{G}(V) \times \boldsymbol{\Pi}(V)$ is referred to as a *coloured graph*. Two coloured graphs $(G_1, \pi_1), (G_2, \pi_2) \in \boldsymbol{G}(V) \times \boldsymbol{\Pi}(V)$ are *isomorphic* if $\exists \gamma \in S_n$ such that $G_1^\gamma = G_2$ and $\pi_1^\gamma = \pi_2$, where the action of γ on G_1 (π_1), denoted by G_1^γ (π_1^γ), permutes the labels of G_1 (π_1). Note that (non-coloured) graph isomorphism is the special case where one takes the trivial partition $\pi = [V]$.

There is a significant body of work [71–77] that has explored quantum algorithms for (non-coloured) graph isomorphism. The following will report the results of the Refs. [71, 72] for (non-coloured) graph isomorphism, and then provide an adaption for coloured graph isomorphism in each case.

In order to formulate an optimisation problem as a QAO problem, there are two central choices: a qubit encoding of the classical decision variables, and a target Hamiltonian H_T with ground-state corresponding to the classical optimum through the qubit encoding. In the case of (coloured or non-coloured) graph isomorphism testing between graphs G_1, G_2, the classical decision variables correspond to a permutation $\sigma \in S_n$ such that $G_1^\sigma = G_2$ if $G_1 \sim G_2$. Two possible qubit encodings of σ are explored in Refs. [71, 72].

The encoding in Ref. [71] begins with the tuple $\sigma = (\sigma^1, \ldots, \sigma^n)$ representing the permutation where $i \to \sigma^i \in \{1, \ldots, n\}$. Each element of the tuple is transformed into

its corresponding binary string via:

$$\sigma_i =: \sum_{j=1}^{\lceil \log_2 n \rceil} \sigma_{ij} 2^j. \tag{7.4}$$

By promoting each element σ_{ij} to a computational basis state, one arrives at the *tuple encoding* of $\sigma \in S_n$ in terms of $n\lceil \log_2 n \rceil$ qubits.

Alternatively, the encoding in Ref. [72] uses binary variables $x_{i,j} \in \{0,1\}$ for $i,j \in \{1,\ldots,n\}$ such that $x_{i,j} = 1$ if $i^\sigma = j$, and $x_{i,j} = 0$ otherwise. This simpler, but less efficient encoding of $\sigma \in S_n$ uses n^2 qubits[1]. After the promotion of each $x_{i,j}$ to a computational basis state, this encoding will be referred to as the *binary encoding*.

Following Ref. [71] and the tuple encoding, a classical cost function, $C_{G_1,G_2}(\sigma)$, may be written in terms of the tuple representation of $\sigma \in S_n$ such that $\min\{C_{G_1,G_2}(\sigma)\} = 0$ iff. $G_1 \sim G_2$ [71]:

$$C_{G_1,G_2}(\sigma) = \sum_{i=1}^{n}\sum_{j=n}^{M} \delta_{\sigma_i,j} + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \delta_{\sigma_i,\sigma_j} + \|\rho(\sigma) A \rho^T(\sigma) - A'\|_p, \tag{7.5}$$

where $M = 2^{\lceil \log_2 n \rceil}$, A (A') is the adjacency matrix of G_1 (G_2), and $\rho(\sigma)$ is the permutation matrix of $\sigma \in S_n$ with elements $\rho_{ij}(\sigma) = \delta_{i,\sigma_j}$. The *entry-wise matrix p-norm* is defined by $\|L\|_p := \left\{\sum_{i=1}^n \sum_{l=1}^n L_{il}^p\right\}^{1/p}$ for an $n \times n$ matrix L with elements L_{il}. The first term penalises $\sigma \notin \{1,\ldots,n\}^n$, the second term penalises $\sigma \notin S_n$, and the third term penalises $G_1^\sigma \neq G_2$. In order to be able to write down an explicit target Hamiltonian, perform the algebraic re-writes:

$$\delta_{\sigma_i,\sigma_j} = \prod_{k=1}^{\lceil \log_2 n \rceil} \left[1 - (\sigma_{ik} - \sigma_{jk})^2\right], \qquad \text{where } \sigma_i =: \sum_{k=1}^{\lceil \log_2 n \rceil} \sigma_{ik} 2^k \tag{7.6}$$

$$\delta_{i,\sigma_j} = \prod_{k=1}^{\lceil \log_2 n \rceil} \left[1 - (i_k - \sigma_{jk})^2\right], \qquad \text{where } i =: \sum_{k=1}^{\lceil \log_2 n \rceil} i_k 2^k, \tag{7.7}$$

[1] The tuple encoding is still quite far away from an optimally parsimonious encoding given that the set of all tuples has cardinality n^n but the set of all permutations has cardinality only $n!$.

so that:

$$\begin{aligned}\left[\rho(\sigma)A\rho^T(\sigma)\right]_{il} &= \rho_{ij}(\sigma)A_{jk}\rho^T_{kl}(\sigma)\\ &= \rho_{ij}(\sigma)A_{jk}\rho_{lk}(\sigma)\\ &= \delta_{i\sigma_j}A_{jk}\delta_{l\sigma_k}\\ &= \prod_{m=1}^{\lceil \log_2 n\rceil}\left[1-(i_m-\sigma_{jm})^2\right]A_{jk}\prod_{m'=1}^{\lceil \log_2 n\rceil}\left[1-(l_{m'}-\sigma_{km'})^2\right],\end{aligned} \tag{7.8}$$

and therefore:

$$\begin{aligned}&\|\rho(\sigma)A\rho^T(\sigma)-A'\|_p\\ &\quad=\left\{\sum_{i=1}^{n}\sum_{l=1}^{n}\left|\left[\rho(\sigma)A\rho^T(\sigma)\right]_{il}-A'_{il}\right|^p\right\}^{\frac{1}{p}}\\ &\quad=\left\{\sum_{i=1}^{n}\sum_{l=1}^{n}\left|\prod_{m=1}^{\lceil \log_2 n\rceil}\left[1-(i_m-\sigma_{jm})^2\right]A_{jk}\prod_{m'=1}^{\lceil \log_2 n\rceil}\left[1-(l_{m'}-\sigma_{km'})^2\right]-A'_{il}\right|^p\right\}^{\frac{1}{p}}.\end{aligned} \tag{7.9}$$

An additional modification required to make the Hamiltonian realisable on quantum hardware is to select $p = 2$ and transform $\|\cdot\|_p \rightarrow \|\cdot\|_p^p$ (i.e. remove the p^{th} root to use $\|L\|_p^p = \sum_{i=1}^{n}\sum_{l=1}^{n}L_{il}^p$) such that the third term of the cost function reads $\|\rho(\sigma)A\rho^T(\sigma)-A'\|_2^2$. The target Hamiltonian therefore takes the form $H_T = H_A + H_B + H_C$, where:

$$H_A = \mathcal{A}\sum_{i=1}^{n}\sum_{j=n}^{M}\prod_{k=1}^{\lceil \log_2 n\rceil}\left[1-(\sigma_{ik}-j_k)^2\right] \tag{7.10}$$

$$H_B = \mathcal{B}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\prod_{k=1}^{\lceil \log_2 n\rceil}\left[1-(\sigma_{ik}-\sigma_{jk})^2\right] \tag{7.11}$$

$$H_C = \mathcal{C}\sum_{i=1}^{n}\sum_{l=1}^{n}\left(\prod_{m=1}^{\lceil \log_2 n\rceil}\left[1-(i_m-\sigma_{jm})^2\right]A_{jk}\prod_{m'=1}^{\lceil \log_2 n\rceil}\left[1-(l_{m'}-\sigma_{km'})^2\right]-A'_{il}\right)^2. \tag{7.12}$$

The constants $\mathcal{A}, \mathcal{B}, \mathcal{C}$ provide some control over the energy spectrum of H_T, as discussed in § 7.4. Alternatively, using the binary encoding described in Ref. [72], it is

straightforward to write down an explicit quadratic cost function [72]:

$$C_{G_1,G_2}(x) = \sum_{i=1}^{n}\left(1-\sum_{j=1}^{n}x_{ij}\right)^2 + \sum_{j=1}^{n}\left(1-\sum_{i=1}^{n}x_{ij}\right)^2 + \sum_{(i,j)\notin E_1}\sum_{(k,l)\in E_2}x_{ik}x_{jl} + \sum_{(i,j)\in E_1}\sum_{(k,l)\notin E_2}x_{ik}x_{jl}, \tag{7.13}$$

where E_1 (E_2) is the edge set for G_1 (G_2). The first two terms penalise x if it doesn't correspond to a permutation matrix, the third term adds a penalty if the permutation maps edges not in G_1 onto G_2, and the final term adds a penalty if the permutation doesn't map edges from G_1 onto G_2. It's then straightforward to write down an explicit Hamiltonian $\widetilde{H}_T = \widetilde{H}_A + \widetilde{H}_B$, where:

$$\widetilde{H}_A = \mathcal{A}\sum_{i=1}^{n}\left(1-\sum_{j=1}^{n}x_{ij}\right)^2 + \mathcal{A}\sum_{j=1}^{n}\left(1-\sum_{i=1}^{n}x_{ij}\right)^2 \tag{7.14}$$

$$\widetilde{H}_B = \mathcal{B}\sum_{(i,j)\notin E_1}\sum_{(k,l)\in E_2}x_{ik}x_{jl} + \mathcal{B}\sum_{(i,j)\in E_1}\sum_{(k,l)\notin E_2}x_{ik}x_{jl}. \tag{7.15}$$

The motivation for including two distinct qubit encodings is that there is a trade-off between qubit resources and target Hamiltonian complexity. The binary encoding uses n^2 qubits and has a quadratic Hamiltonian, whereas the tuple encoding uses $n\lceil\log_2 n\rceil$ qubits and has a target Hamiltonian of degree $2\lceil\log_2 n\rceil > 2$.

Coloured graph isomorphism is a generalisation of non-coloured graph isomorphism where as well as preserving edge structure in mapping $G_1 \to_\sigma G_2$, there is also a requirement to preserve colour structure: $\pi_1 \to_\sigma \pi_2$. Using the two previously discussed formulations of (non-coloured) graph isomorphism for QAO, it is straightforward to generalise to coloured graph isomorphism by judicious additions to the target Hamiltonians.

For the tuple encoding, consider adding an extra term to the cost function given by:

$$\begin{aligned} C^{\pi_1,\pi_2}_{G_1,G_2}(\sigma)_{colour} &= \|\pi_1^\sigma - \pi_2\|_p^p \\ &= \sum_{i=1}^{n} |[\rho(\sigma)\pi_1]_i - [\pi_2]_i|^p \\ &= \sum_{i=1}^{n} \left|\rho_{ij}(\sigma)\pi_{1j} - \pi_{2i}\right|^p \\ &= \sum_{i=1}^{n} \left|\delta_{i,\sigma_j}\pi_{1j} - \pi_{2i}\right|^p \\ &= \sum_{i=1}^{n} \left| \prod_{k=1}^{\lceil \log_2 n \rceil} \left[1 - (i_k - \sigma_{jk})\right] \pi_{1j} - \pi_{2i} \right|^p , \end{aligned} \tag{7.16}$$

which penalises all $\sigma \in S_n$ such that $\pi_1^\sigma \neq \pi_2$. By making the same choice $p = 2$, the fourth term in the tuple target Hamiltonian can be written as:

$$H_D = \mathcal{D} \sum_{i=1}^{n} \left(\prod_{k=1}^{\lceil \log_2 n \rceil} \left[1 - (i_k - \sigma_{jk})\right] \pi_{1j} - \pi_{2i} \right)^2 . \tag{7.17}$$

For the binary encoding, consider adding an extra term to the cost function given by:

$$C^{\pi_1,\pi_2}_{G_1,G_2}(x)_{colour} = \sum_{i=1}^{n} \left(\pi_{2i} - \sum_{j=1}^{n} x_{ij}\pi_{1j} \right)^2 , \tag{7.18}$$

which again penalises all $\sigma \in S_n$ such that $\pi_1^\sigma \neq \pi_2$. This straightforwardly leads to an additional term in the binary target Hamiltonian given by:

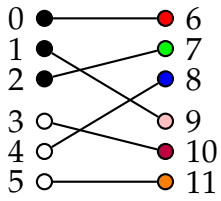
$$\widetilde{H}_C = \mathcal{C} \sum_{i=1}^{n} \left(\pi_{2i} - \sum_{j=1}^{n} x_{ij}\pi_{1j} \right)^2 . \tag{7.19}$$

7.3. QAO for Tensor Expression Isomorphism

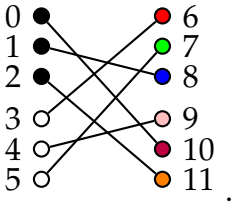
Tensor expression isomorphism is equivalent to a restriction of coloured graph isomorphism where edge structure is preserved from $G_1 \to_\sigma \widetilde{G}_2$, and colour structure is preserved from $\pi_1 \to_\sigma \widetilde{\pi}_2$, and $\widetilde{G}_2$ ($\widetilde{\pi}_2$) is related to G_2 (π_2) by at most the exchange of identical tensors (see Chapter 4 for further details). Consider for example tensor

expressions given by $E_1 = \epsilon_{ijk}\epsilon_{lmn}T_{ikm}T_{jln}$ and $E_2 = \epsilon_{ijk}\epsilon_{lmn}T_{lnj}T_{mik}$. Transform each into their corresponding coloured graph using the process outlined in Chapter 4:

$$E_1 \longrightarrow (G_1, \pi_1) = \tag{7.20}$$



$$E_2 \longrightarrow (G_2, \pi_2) = \tag{7.21}$$



Define the *tensor equivalence tuple*, $\mathcal{T} \in \{1, \ldots, T\}^T$, such that $\mathcal{T}_i = \mathcal{T}_j$ iff. the i^{th} and j^{th} tensor are identical. In the above example, $\mathcal{T} = (0, 0, 1, 1)$. Define the *tensor index tuple*, $\mathcal{I} \in \prod_t \{1, \ldots, n\}^{|\mathcal{I}_t|}$, where $\mathcal{I}_i$ is the set of vertices associated with the i^{th} tensor of E_1. In the above example, $\mathcal{I} = ((0, 1, 2), (3, 4, 5), (6, 7, 8), (9, 10, 11))$. For the binary encoding, define the following extra term in the target Hamiltonian:

$$\widetilde{H}_{tensor} = -\mathcal{B} \sum_{t=1}^{T} \sum_{t'=t+1}^{T} \delta_{\mathcal{T}_t, \mathcal{T}_{t'}} |\mathcal{I}_t| \prod_{i=1}^{|\mathcal{I}_t|} x_{\mathcal{I}_{ti}, \mathcal{I}_{t'i}}. \tag{7.22}$$

This set of Hamiltonian terms is chosen to precisely cancel the terms in $\widetilde{H}_B$ (Eqn. 7.15) that penalise $G_2 \to \widetilde{G}_2$ where $\widetilde{G}_2$ is related to G_2 by the exchange of identical tensors. In the example above, $\widetilde{H}_{tensor} = -3\mathcal{B}[x_{0,3}x_{1,4}x_{2,5} + x_{6,9}x_{7,10}x_{8,11}]$.

7.4. Application to Nuclear Correlation Functions and Open Questions

Sections § 7.2 – § 7.3 have proposed an approach to perform tensor expression isomorphism testing using quantum adiabatic optimisation, with qubit resource scaling that is quadratic in the number of index slots in each expression. For reference, there are

47 index slots in the tensor expressions for helium-3, so the resource requirements are well beyond currently available hardware [78]. One of the key indicators of QAO performance is the density of energy levels of the target Hamiltonian, with sparse energy levels allowing smaller T (or equivalently coarser s discretisation) to achieve the same preparation fidelity. One could conjecture that the inherent permutation symmetry contained in the target Hamiltonians in Eqns. 7.10 – 7.15 will result in highly degenerate energy spectra and hence poor adiabatic performance. Constants $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, and $\mathcal{D}$ offer limited control over energy spectrum density, but it may be possible to engineer degeneracy breaking terms to improve performance. It should be noted that this approach to isomorphism testing is resilient to both preparation and measurement errors since $\sigma \in S_n$ measured in the computational basis can be checked efficiently by computing G_1^σ.

One possible application to nuclear correlators is to follow the procedure proposed in Chapter 4 to perform redundant isomorph elimination on nuclear correlator tensor expressions, however that procedure doesn't present a significant bottleneck in the nuclear correlator pre-processing stage. Algorithmic modification would be required to perform canonicalisation as opposed to isomorphism testing. Tensor e-graph construction, as discussed in Chapter 6, is dominated by tensor expression canonicalisation, so quantum information processing has the potential to accelerate this process, provided that the resource and energy spectrum density issues are resolved.

Chapter 8.

Summary and Outlook

This thesis has explored a number of approaches to accelerate the numerical evaluation of correlation functions of multi-hadron systems in the context of lattice QCD. Chapter 2 introduced a set of multi-nucleon interpolating operators that were used as benchmarks in subsequent chapters. Chapter 3 presented a number of algorithms from the literature, of which the hadron block method and index list technique form a foundation for the novel methods developed subsequently. A set of benchmarks of the hadron block method against the naïve approach of Chapter 2 showed a clear performance benefit for $A \geq 2$, as well as demonstrating that the block expression evaluation time dominates the correlator evaluation CPU time for $A \lessapprox 4$ on a 64^3 lattice volume.

Chapter 4 developed a general approach for removing redundant isomorphs from a large collection of tensor expressions through graph canonicalisation. A high-level overview of McKay's Individualisation-Refinement Algorithm is presented, followed by a general procedure for the transformation of tensor expressions into coloured graphs via tensor network notation. Also described was a software package developed to symbolically canonicalise tensor expressions using graph canonicalisation provided by McKay's nauty software package. The notion of 'half nucleon' blocks, which is the decomposition of nucleon block expressions into two definite-momentum pieces, was proposed and its consequences explored through a presentation of tensor expression statistics.

Chapter 5 presented factor trees, which optimally store a 'factorised' and degeneracy-free form of a nuclear correlator's tensor expressions. Linearised factor trees were presented as a representation of (abstract) factor trees that is optimised for cache performance. A selection of benchmarks was presented for light nuclei that showed

between one and two orders of magnitude improvement over correlator calculation via (only) hadron blocks. Memory usage limits the applicability of this method to heavier nuclei, although future work could extend the current domain significantly.

Chapter 6 developed the tensor e-graph method, which is a novel adaption of e-graph techniques to efficiently extract frequently-used sub-expressions from a large collection of tensor expressions. After a demonstration of the approach used for generic e-graphs, the tensor re-write rule was developed with a set of heuristic restrictions that improve the efficiency of e-graph construction. A parameterised cost function is asserted with motivation, and its extraction performance is measured over a patch of parameter space for a helium-3 correlator. In general, the pre-processing stage for tensor e-graphs requires fewer computational resources in comparison to factor trees, but thus far offers more limited performance improvements for correlator calculations. Future work on extraction heuristics is likely to offer further correlator speed-ups.

Chapter 7 presented a prospective programme of work that endeavours to establish the usefulness (or otherwise) of adiabatic quantum algorithms for nuclear correlator calculations. An algorithm for tensor expression isomorphism testing using quantum adiabatic optimisation was presented. It is left to future work to extend this algorithm to isomorphism partitioning and canonicalisation so that it may be used in, for example, tensor e-graph construction. Basic resource estimates determined that no current hardware would be able to execute the adiabatic tensor expression isomorphism algorithm for nuclear correlator expressions of $A \geq 3$ nucleons.

Appendix A.

Coarsest Equitable Refinements

Continue the worked example in Chapter 4 by applying Algorithm 1 to the second child of π_0. First set $\tau' = [2 \mid 0,\ 4 \mid 1,\ 3,\ 5] = [V_1', V_2', V_3']$, and compute $B = \{(2,1),(2,3),(3,1),(3,2)\}$ by computing D_{ij} in Table A.1. Then by selecting $(k,l) = (2,1)$ (the minimal element of B by lexicographic order), conclude $[X_1, X_2] = [4 \mid 0]$, and so update $\tau' = [2 \mid 4 \mid 0 \mid 1,\ 3,\ 5]$. Next update $B = \{(4,1),(4,2),(4,3)\}$ by computing D_{ij} in Table A.2, and by selecting $(k,l) = (4,1)$ conclude $[X_1, X_2] = [3 \mid 1,\ 5]$. By updating $\tau' = [2 \mid 4 \mid 0 \mid 3 \mid 1,\ 5]$, update $B = \{(5,2),(5,3)\}$ from Table A.3, set $X = [5,\ 1]$, and update $\tau' = [2 \mid 4 \mid 0 \mid 3 \mid 5 \mid 1]$. Since the process has arrived at a discrete partition, the algorithm terminates by concluding that the coarsest equitable refinement of $[2 \mid 0,\ 4 \mid 1,\ 3,\ 5]$ is $\pi_2 = [2 \mid 4 \mid 0 \mid 3 \mid 5 \mid 1]$.

		i		
	D_{ij}	1	2	3
	1	$\{0\}$	$\boxed{\{1,0\}}$	$\boxed{\{1,0,1\}}$
j	2	$\{1\}$	$\{0,0\}$	$\boxed{\{2,1,0\}}$
	3	$\{2\}$	$\boxed{\{2,1\}}$	$\{0,0,0\}$

Table A.1.: Partition degree consistency table for $\tau' = [2 \mid 0,\ 4 \mid 1,\ 3,\ 5] = [V_1', V_2', V_3']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

Next, apply Algorithm 1 to the third child of π_0. First set $\tau' = [4 \mid 0,\ 2 \mid 1,\ 3,\ 5] = [V_1', V_2', V_3']$, and compute $B = \{(3,1),(3,2)\}$ by computing D_{ij} in Table A.4. Then by selecting $(k,l) = (3,1)$ (the minimal element of B by lexicographic order), conclude $[X_1, X_2] = [3,\ 5 \mid 1]$, and so update $\tau' = [4 \mid 0,\ 2 \mid 3,\ 5 \mid 1]$. Next update $B = \emptyset$

	D_{ij}	1	2	3	4
				i	
j	1	$\{0\}$	$\{0\}$	$\{1\}$	$\boxed{\{1,0,1\}}$
	2	$\{0\}$	$\{0\}$	$\{0\}$	$\boxed{\{1,0,0\}}$
	3	$\{1\}$	$\{0\}$	$\{0\}$	$\boxed{\{1,1,0\}}$
	4	$\{2\}$	$\{1\}$	$\{2\}$	$\{0,0,0\}$

Table A.2.: Partition degree consistency table for $\tau' = [2 \mid 4 \mid 0 \mid 1,\ 3,\ 5] = [V_1', V_2', V_3', V_4']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

	D_{ij}	1	2	3	4	5
					i	
	1	$\{0\}$	$\{0\}$	$\{1\}$	$\{0\}$	$\{1,1\}$
	2	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	$\boxed{\{1,0\}}$
j	3	$\{1\}$	$\{0\}$	$\{0\}$	$\{1\}$	$\boxed{\{1,0\}}$
	4	$\{0\}$	$\{0\}$	$\{1\}$	$\{0\}$	$\{0,0\}$
	5	$\{0\}$	$\{1\}$	$\{1\}$	$\{0\}$	$\{0,0\}$

Table A.3.: Partition degree consistency table for $\tau' = [2 \mid 4 \mid 0 \mid 3 \mid 1,\ 5] = [V_1', V_2', V_3', V_4', V_5']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

by computing D_{ij} in Table A.5, and hence terminate and conclude that the coarsest equitable refinement of $[4 \mid 0,\ 2 \mid 1,\ 3,\ 5]$ is $\pi_3 = [4 \mid 0,\ 2 \mid 3,\ 5 \mid 1]$.

	D_{ij}	1	2	3
			i	
	1	$\{0\}$	$\boxed{\{1,0\}}$	$\boxed{\{1,0,1\}}$
j	2	$\{1\}$	$\{0,0\}$	$\boxed{\{2,1,0\}}$
	3	$\{2\}$	$\boxed{\{2,1\}}$	$\{0,0,0\}$

Table A.4.: Partition degree consistency table for $\tau' = [4 \mid 0,\ 2 \mid 1,\ 3,\ 5] = [V_1', V_2', V_3']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

Next, compute the child nodes of $pi_3 = [4 \mid 0,\ 2 \mid 3,\ 5 \mid 1]$ (the only non-discrete child of π_0). Applying the target cell selector to π_3 yields $T(G, \pi_0, \pi_3) = [0,\ 2]$, and so the candidate child nodes for π_3 are $[4 \mid 0 \mid 2 \mid 3,\ 5 \mid 1]$, and $[4 \mid 2 \mid 0 \mid 3,\ 5 \mid 1]$. Apply Algorithm 1 to the first child $\tau' = [4 \mid 0 \mid 2 \mid 3,\ 5 \mid 1]$ by computing $B = \{(4,2),(4,3)\}$

	D_{ij}	1	2	3	4
			i		
j	1	$\{0\}$	$\{0,0\}$	$\{0,0\}$	$\{1\}$
	2	$\{0\}$	$\{1,1\}$	$\{1,1\}$	$\{2\}$
	3	$\{0\}$	$\{1,1\}$	$\{0,0\}$	$\{0\}$
	4	$\{1\}$	$\{1,1\}$	$\{0,0\}$	$\{0\}$

Table A.5.: Partition degree consistency table for $\tau' = [4 \mid 0 \mid 2 \mid 3,\ 5 \mid 1] = [V_1', V_2', V_3', V_4']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

from Table A.6, and then use $X = [5 \mid 3]$ to conclude that $\pi_4 = [4 \mid 0 \mid 2 \mid 5 \mid 3 \mid 1]$. Finally, apply Algorithm 1 to the second child $\tau' = [4 \mid 2 \mid 0 \mid 3,\ 5 \mid 1]$ by computing $B = \{(4,2),(4,3)\}$ from Table A.7, and then use $X = [3 \mid 5]$ to conclude that $\pi_5 = [4 \mid 2 \mid 0 \mid 3 \mid 5 \mid 1]$.

	D_{ij}	1	2	3	4	5
				i		
	1	$\{0\}$	$\{0\}$	$\{0\}$	$\{0,0\}$	$\{1\}$
	2	$\{0\}$	$\{0\}$	$\{1\}$	$\boxed{\{1,0\}}$	$\{1\}$
j	3	$\{0\}$	$\{1\}$	$\{0\}$	$\boxed{\{0,1\}}$	$\{1\}$
	4	$\{0\}$	$\{1\}$	$\{1\}$	$\{0,0\}$	$\{0\}$
	5	$\{1\}$	$\{1\}$	$\{1\}$	$\{0,0\}$	$\{0\}$

Table A.6.: Partition degree consistency table for $\tau' = [4 \mid 0 \mid 2 \mid 3,\ 5 \mid 1] = [V_1', V_2', V_3', V_4', V_5']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

	D_{ij}	1	2	3	4	5
				i		
	1	$\{0\}$	$\{0\}$	$\{0\}$	$\{0,0\}$	$\{1\}$
	2	$\{0\}$	$\{0\}$	$\{1\}$	$\boxed{\{0,1\}}$	$\{1\}$
j	3	$\{0\}$	$\{1\}$	$\{0\}$	$\boxed{\{1,0\}}$	$\{1\}$
	4	$\{0\}$	$\{1\}$	$\{1\}$	$\{0,0\}$	$\{0\}$
	5	$\{1\}$	$\{1\}$	$\{1\}$	$\{0,0\}$	$\{0\}$

Table A.7.: Partition degree consistency table for $\tau' = [4 \mid 2 \mid 0 \mid 3,\ 5 \mid 1] = [V_1', V_2', V_3', V_4', V_5']$. Boxes correspond to $|D_{ij}| > 1 \Leftrightarrow (i,j) \in B$.

Bibliography

[1] J. D. Dixon and B. Mortimer, *Permutation Groups* (Springer-Verlag New York, 1996).

[2] C. Jordan, *Traité des substitutions et des équations algébriques* (Gauthier-Villars, Paris, 1870).

[3] A. Cayley, Philosophical Magazine **7**, 40 (1854).

[4] E. Galois, Bulletin des Sciences Mathématiques **X**III, 271 (1830).

[5] M. Willsey *et al.*, Proceedings of the ACM on Programming Languages **5**, 1–29 (2021).

[6] C. Gattringer and C. B. Lang, *Quantum chromodynamics on the lattice* (Springer, 2010).

[7] M. D. Schwartz, *Quantum Field Theory and the Standard Model* (Cambridge University Press, 2013).

[8] H. Fritzsch, M. Gell-Mann, and H. Leutwyler, Phys. Lett. B **4**7, 365 (1973).

[9] G. D. Rochester and C. C. Butler, Nature **1**60, 855 (1947).

[10] V. D. Hopper and S. Biswas, Phys. Rev. **8**0, 1099 (1950).

[11] H. Yukawa, Progress of Theoretical Physics Supplement **1**, 1 (1955).

[12] M. Gell-Mann, Phys. Lett. **8**, 214 (1964).

[13] G. Zweig, CERN Report 8419/TH.401 (1964).

[14] G. Zweig, Developments in the Quark Theory of Hadrons, A Reprint Collection **1**, 22 (1964).

[15] O. W. Greenberg, Phys. Rev. Lett. **1**3, 598 (1964).

[16] M. Y. Han and Y. Nambu, Phys. Rev. **1**39, B1006 (1965).

[17] M. Illa, Approaching nuclear interactions with lattice qcd, 2021, 2109.10068.

[18] D. J. Gross and F. Wilczek, Phys. Rev. Lett. **3**0, 1343 (1973).

[19] H. D. Politzer, Phys. Rev. Lett. **3**0, 1346 (1973).

[20] K. G. Wilson, Phys. Rev. D **1**0, 2445 (1974).

[21] R. P. Feynman, Rev. Mod. Phys. **2**0, 367 (1948).

[22] G. C. Wick, Phys. Rev. **9**6, 1124 (1954).

[23] G. Aarts, PoS **L**ATTICE2012, 017 (2012), 1302.3028.

[24] W. Detmold, G. Kanwar, H. Lamm, M. L. Wagman, and N. C. Warrington, Physical Review D **1**03 (2021).

[25] N. Warrington, Taming the sign problem in lattice field theory with deformed path integral contours, 2019.

[26] A. Alexandru, G. Basar, P. F. Bedaque, and N. C. Warrington, Complex paths around the sign problem, 2020, 2007.05436.

[27] H. B. Nielsen and M. Ninomiya, Phys. Lett. B **1**05, 219 (1981).

[28] T. Yamazaki, Y. Kuramashi, and A. Ukawa, Phys. Rev. D **8**1, 111504 (2010), 0912.1383.

[29] M. D. Schwartz, *Quantum Field Theory and the Standard Model* (Cambridge University Press, 2013), pp. 100–103.

[30] *Quantum chromodynamics on the lattice* (Springer, Berlin, 2010), p. 124.

[31] NPLQCD, S. R. Beane *et al.*, Phys. Rev. D **8**1, 054505 (2010), 0912.4243.

[32] M. Luscher, Commun. Math. Phys. **1**04, 177 (1986).

[33] M. Luscher, Nucl. Phys. B **3**54, 531 (1991).

[34] N. Ishii, S. Aoki, and T. Hatsuda, Phys. Rev. Lett. **9**9, 022001 (2007), nucl-th/0611096.

[35] S. Aoki, T. Hatsuda, and N. Ishii, Prog. Theor. Phys. **1**23, 89 (2010), 0909.5585.

[36] J. M. M. Hall, A. C. P. Hsu, D. B. Leinweber, A. W. Thomas, and R. D. Young, Phys. Rev. D **8**7, 094510 (2013), 1303.4157.

[37] J. M. M. Hall, A. C. P. Hsu, D. B. Leinweber, A. W. Thomas, and R. D. Young, PoS **L**ATTICE2012, 145 (2012), 1207.3562.

[38] HAL QCD, T. Iritani *et al.*, JHEP **0**3, 007 (2019), 1812.08539.

[39] W. Detmold *et al.*, Phys. Rev. D **1**04, 034502 (2021), 1908.07050.

[40] W. Detmold and K. Orginos, Phys. Rev. D **8**7, 114512 (2013).

[41] G. Lindfield and J. Penny, Chapter 2 - linear equations and eigensystems, in *Numerical Methods (Fourth Edition)*, edited by G. Lindfield and J. Penny, pp. 73–156, Academic Press, , fourth edition ed., 2019.

[42] T. Doi and M. G. Endres, Computer Physics Communications **1**84, 117 (2013).

[43] J. Günther, B. C. Tóth, and L. Varnhorst, Phys. Rev. D **8**7, 094513 (2013).

[44] H. Nemura *et al.*, Proceedings of the 12th International Conference on Hypernuclear and Strange Particle Physics (HYP2015) (2017).

[45] W. Detmold *et al.*, Phys. Rev. D **7**8, 014507 (2008), 0803.2728.

[46] W. Detmold and M. J. Savage, Phys. Rev. D **8**2, 014511 (2010), 1001.2768.

[47] B. D. McKay, Practical graph isomorphism, 1981.

[48] B. D. McKay and A. Piperno, Journal of symbolic computation **6**0, 94 (2014).

[49] A. Piperno, Search space contraction in canonical labeling of graphs, 2011, 0804.4881.

[50] S. G. Hartke and A. J. Radcliffe, Mckay's canonical graph labeling algorithm, 2008.

[51] R. E. Tarjan, J. ACM **2**2, 215–225 (1975).

[52] J. C. Bridgeman and C. T. Chubb, Journal of Physics A: Mathematical and Theoretical **5**0, 223001 (2017).

[53] Z. Li, S. Shao, and W. Liu, Classifications and canonical forms of tensor product expressions in the presence of permutation symmetries, 2018, 1604.06156.

[54] D. Terpstra, H. Jagode, H. You, and J. Dongarra, Collecting performance data with papi-c, in *Tools for High Performance Computing 2009*, edited by M. S. Müller, M. M. Resch, A. Schulz, and W. E. Nagel, pp. 157–173, Berlin, Heidelberg, 2010, Springer Berlin Heidelberg.

[55] M. Kerrisk, *mmap(2) — Linux manual page*, 2021, Available at https://man7.org/linux/man-pages/man2/mmap.2.html.

[56] D. Detlefs, G. Nelson, and J. B. Saxe, J. ACM **5**2, 365–473 (2005).

[57] L. de Moura and N. Bjørner, Z3: An efficient smt solver, in *Tools and Algorithms for the Construction and Analysis of Systems*, edited by C. R. Ramakrishnan and J. Rehof, pp. 337–340, Berlin, Heidelberg, 2008, Springer Berlin Heidelberg.

[58] Z. Jia *et al.*, Taso: Optimizing deep learning computation with automatic generation of graph substitutions, in *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, p. 47–62, New York, NY, USA, 2019, Association for Computing Machinery.

[59] Y. R. Wang, S. Hutchison, J. Leang, B. Howe, and D. Suciu, CoRR **a**bs/2002.07951 (2020), 2002.07951.

[60] Y. Yang *et al.*, CoRR **a**bs/2101.01332 (2021), 2101.01332.

[61] B. Saiki, O. Flatt, C. Nandi, P. Panchekha, and Z. Tatlock, Combining precision tuning and rewriting, in *2021 IEEE 28th Symposium on Computer Arithmetic (ARITH)*, pp. 1–8, 2021.

[62] M. Baake, Journal of Mathematical Physics **2**5, 3171 (1984).

[63] J. A. Bergou, M. Hillery, and M. Saffman, *Quantum Information Processing* (Springer Nature Switzerland AG, 2021).

[64] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press, 2010).

[65] B. Apolloni, C. Carvalho, and D. de Falco, Stochastic Processes and their Applications **3**3, 233 (1989).

[66] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, Quantum computation by adiabatic evolution, 2000, quant-ph/0001106.

[67] E. Farhi, J. Goldstone, and S. Gutmann, A quantum approximate optimization

algorithm, 2014, 1411.4028.

[68] L. T. Brady *et al.*, Behavior of analog quantum algorithms, 2021, 2107.01218.

[69] D. E. Deutsch, Proceedings of the Royal Society of London A **4**25, 73 (1989).

[70] T. Albash and D. A. Lidar, Reviews of Modern Physics **9**0 (2018).

[71] F. Gaitan and L. Clark, Physical Review A **8**9 (2014).

[72] A. Lucas, Frontiers in Physics **2** (2014).

[73] C. Moore, A. Russell, and P. Sniady, Proceedings of the thirty-ninth annual ACM symposium on Theory of computing - STOC '07 (2007).

[74] I. Hen and A. P. Young, Physical Review A **8**6 (2012).

[75] D. Tamascelli and L. Zanetti, Journal of Physics A: Mathematical and Theoretical **4**7, 325302 (2014).

[76] K. Brádler, S. Friedland, J. Izaac, N. Killoran, and D. Su, Special Matrices **9**, 166–196 (2021).

[77] X. Li and H. Chen, The quantum algorithm for graph isomorphism problem, 2019, 1901.06530.

[78] V. Kendon, Interface Focus **1**0 (2020).

List of figures

List of tables