Optimisations to Hybrid Monte Carlo for Lattice QCD

∘ *Taylor Ryan Haar* ∘

A thesis submitted for the degree of
DOCTOR OF PHILOSOPHY

DEPARTMENT OF PHYSICS
SCHOOL OF PHYSICAL SCIENCES
FACULTY OF SCIENCES

supervised by
Dr. Waseem KAMLEH, A/Prof. James ZANOTTI

February, 2019

Contents

List of Figures

List of Tables

Abstract

In Lattice Quantum Chromodynamics, we calculate physical quantities on a discrete 4D Euclidean lattice via expectation values, which take the form of path integrals. Due to the high dimensionality of these integrals, the standard technique for evaluating lattice expectation values is Monte Carlo; we generate configurations of gauge fields U and fermion fields ψ distributed according to the lattice action S, then take a weighted average of the observable across the configurations. The most common method used to generate configurations is a Markov Chain technique called Hybrid Monte Carlo. While this technique is functional, it takes a lot of computational resources to generate configurations which are desirably close to the continuum theory.

The object of this work is to investigate a variety of improvements over the basic Hybrid Monte Carlo method, and determine which combinations produce independent configurations at the lowest cost.

We start by performing a systematic study of filtering for double-flavour simulations, comparing polynomial filtering to the common technique of mass filtering. We show that combining these two methods produces optimal speedup with minimal tuning of parameters, which can be a serious concern when multiple filters are involved. During this investigation, we used the novel technique of overlaid integrators for implementing multiple integration time scales, which expands the possible step-size choices.

Next, we investigate improvements to single-flavour simulations, comparing polynomial filtering with a different method that we denote truncated ordered product RHMC. We obtain the best speedup when using truncation filters, but it is highly dependent on the truncation order chosen. To alleviate this problem, we apply a novel integration step-size tuning method called characteristic scale tuning which allows for step-sizes to be better tuned to the energy modes of the system. This improves the performance of our algorithms for a wide range of filter parameters, thus reducing the need to tune filter parameters.

Finally, we extend our single-flavour techniques to Lattice QCD+QED simulations, which include electromagnetic effects via a photon field.

Declaration

I, TAYLOR HAAR, 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.

Date: 17/08/2018

Signature

Acknowledgements

First off, I'd like to thank my supervisors James Zanotti and Waseem Kamleh. They have openly offered their knowledge and expertise in the field, and actively contributed to this work's development. I have had many valuable discussions with them to troubleshoot and crystallise the ideas herein.

I would also like to thank the co-authors of BQCD, Hinnerk Stüben and Yoshifumi Nakamura, for providing the lattice configuration generation code-base BQCD, which much of this work relies upon. It has been a pleasure to work with them to integrate my code modifications into the official version.

This work would have been far more arduous without the support of my fellow students and friends at the University of Adelaide. Thanks to the HPCP class of 2013, the crew in 123a, and the other graduate students I have gotten to know over the years.

Finally, a special thanks to my family for supporting me throughout. Mum, I dedicate this to you.

Introduction

The Standard Model of physics is an incredibly successful theory that describes how three of the four fundamental forces of nature behave, which is through force mediating particles known as gauge bosons that arise from a particular local gauge symmetry. The three forces covered by the Standard Model are electromagnetism, which is mediated by the U(1) photon γ; the weak nuclear force, which is mediated by the $SU(2)_L$ weak bosons $W^{\pm}, Z^0$; and the strong nuclear force, which is mediated by the SU(3) gluon g. In addition, this theory has a Higgs field ϕ which gives a mass to the weak bosons, leptons and quarks. This model agrees strongly with experiment across a wide range of observed particle behaviour.

The part of the Standard Model that describes the strong force is Quantum Chromodynamics (QCD). The defining features of this theory is that the gluon is massless and the SU(3) group is non-Abelian, which gives rise to 3-gluon and 4-gluon self-interactions. This produces a number of unique properties. In particular, the gauge coupling g of the interaction vertex runs with the energy: while it is small at high energies, it is large (~ 1) at typical low energies. The size of the coupling constant invalidates the usual perturbative approaches to calculating observables in the theory: the series expansions involved diverge for some quantities.

Lattice QCD [1] was developed to solve this issue. It is a non-perturbative approach to QCD where we discretise space-time onto an Euclidean 4-D lattice. Calculations are then made via expectation values in the form of path integrals, which are evaluated via Monte Carlo methods: we sample configurations of our gluon and quark fields distributed according to the lattice QCD action, then evaluate expectation values as an ensemble average. Lattice QCD simulations have reliably reproduced the masses of several hadrons as observed in experiment, and provide predictions for the quark masses [2–6], certain decay constants [4, 5] and form factors: see [7] for a recent review.

The most common Monte Carlo method used in Lattice QCD is Hybrid Monte Carlo (HMC). This is where a series of configurations are generated by inte-

gration trajectories that preserve a fictional Hamiltonian, in conjunction with a Metropolis-Hastings acceptance step that ensures that the configurations tend to the desired equilibrium distribution. While this technique is functional, it takes a lot of computation resources to generate configurations of physically interesting lattices which are 'close' to the continuum theory: physical masses, small lattice spacings, and large extents.

In this work, we investigate various optimisations to Hybrid Monte Carlo in order to reduce the computational effort required to generate lattice configurations. First, in chapter 1 we give an overview of Lattice QCD to put lattice configuration generation into context. Then in chapter 2, we discuss Hybrid Monte Carlo at length, along with a variety of improvements that can be made to improve the performance of the algorithm.

The subsequent two chapters are devoted to comparing the performance of certain optimisations to HMC, mainly a class of techniques known as filtering methods. While many of these optimisations already see use in the Lattice QCD community, there are relatively few studies that benchmark different filtering methods. In chapter 3 we compare the cost performance of two different filtering methods used on a double-flavour pseudofermion field, namely polynomial filtering and mass preconditioning. Such a pseudofermion is a combination of two mass degenerate quarks, so it most often used to simulate the up and down quarks. In chapter 4 we consider single-flavour pseudofermions, investigating the performance of polynomial filtering and truncated ordered product rational HMC (tRHMC) which apply in this case. Single-flavour pseudofermions represent a single flavour of quark, and are required for simulating quarks with no identical partner such as the strange quark.

Finally, in chapter 5, we discuss how to incorporate electromagnetism into the Lattice QCD theory to produce Lattice QCD+QED simulations, such that we can calculate electromagnetic corrections to measured observables. This requires the use of single-flavour pseudofermions even for the up and down quarks as they are now differentiated by their charge. Then we investigate the performance of tRHMC for a Lattice QCD+QED simulation.

∘ **1** ∘

Lattice QCD

1.1 Quantum Chromodynamics

Quantum Chromodynamics (QCD) is a quantum field theory in the Standard Model that describes the strong force. It is based on the non-Abelian Lie group SU(3), which is most commonly represented as the set of 3×3 complex unitary matrices with determinant 1. The fundamental fields of this theory are the quark/fermion fields $\psi, \overline{\psi}$ and the gluon/gauge fields A_μ.

The quark fields $\psi, \overline{\psi}$ are elements of the fundamental representation of SU(3), a.k.a. the Lie group SU(3). This means they can have three different charges in SU(3) space, which we denote *colour charge*. The full structure of these fields is given by

$$\psi^f_{a\alpha}(x), \overline{\psi}^f_{a\alpha}(x). \tag{1.1}$$

Thus, quarks are fields of space-time x, Dirac spinors with Dirac indices $\alpha = 0, 1, 2, 3$, and colour vectors with colour index $a = 1, 2, 3$ (alternatively, "red, green, blue"). The index f indicates the flavour of the quark, and in nature we have observed six of these: up, down, strange, charm, top and bottom.

The gluon fields $A_\mu(x)$ are elements of the adjoint representation of SU(3), a.k.a. the Lie algebra $\mathfrak{su}(3)$. This space is 8-dimensional, with elements of the Lie algebra being expressible as real linear combinations of the 8 SU(3) generators $t^{(a)}$, $a = 1, 2, \ldots, 8$; for the gluon field, we can write $A_\mu(x) = A^{(a)}_\mu(x) t^{(a)}$ where $A^{(a)}_\mu(x)$ are eight real-valued fields. This implies there are 8 types of gluons. In matrix notation, elements of the Lie algebra $\mathfrak{su}(3)$ are traceless Hermitian matrices, and the generators are $t^{(a)} = \lambda^a/2$ where λ^a are the Gell-Mann matrices (A.1). The full structure of the gluon field is

$$A^{cd}_\mu(x), \tag{1.2}$$

so it is a Lorentz vector ($\mu = 0, 1, 2, 3$) and a colour matrix ($c, d = 1, 2, 3$).

In nature, we have never observed objects with net colour charge. Motivated by this, the essential idea of QCD is to have quantities which are invariant under local SU(3) gauge transformations, which are defined by the fermion field transformation

$$\psi(x) \to \psi'(x) = G(x)\psi(x) \tag{1.3a}$$

$$\text{and} \quad \overline{\psi}(x) \to \overline{\psi}'(x) = \overline{\psi}(x)G^{-1}(x), \tag{1.3b}$$

where $G(x) \in \mathrm{SU}(3)$. To be concrete about the form of the transformation, we often write $G(x)$ in terms of the generators $t^{(a)}$ of SU(3):

$$G(x) = \exp(i\tau^{(a)}(x)t^{(a)}), \tag{1.4}$$

where $\tau^{(a)}$ are a set of real-valued functions of x

The Lagrangian density for this theory which satisfies gauge invariance is given by

$$\mathcal{L}_{QCD} = \sum_{f,ab,\alpha\beta,\mu\nu} \overline{\psi}^f_{a\alpha}\left(i(\gamma_\mu)_{\alpha\beta}(D_\mu)_{ab} - m^f\,\delta_{ab}\delta_{\alpha\beta}\right)\psi^f_{b\beta} - \frac{1}{2g^2}\,\mathrm{Tr}\left[(F_{\mu\nu})^{ab}(F^{\mu\nu})^{ba}\right]. \tag{1.5}$$

Here, g is the SU(3) gauge coupling, m^f is the bare mass of quark flavour f, γ_μ are the gamma matrices (A.3), D_μ is the covariant derivative and $F_{\mu\nu}$ is the gluon field-strength tensor. We often omit the sums over flavour f, colour $\{a, b\}$, Dirac index $\{\alpha, \beta\}$ and Lorentz index $\{\mu, \nu\}$, and write

$$\mathcal{L}_{QCD} = \overline{\psi}(i\gamma_\mu D_\mu - m)\psi - \frac{1}{2g^2}\,\mathrm{Tr}\left[F_{\mu\nu}F^{\mu\nu}\right]; \tag{1.6}$$

such sums are assumed unless otherwise noted. The form of this Lagrangian can be motivated by the classical equations of motion, which will be discussed in section 1.1.1.

The covariant derivative for QCD is given by

$$D_\mu = \partial_\mu - iA_\mu; \tag{1.7}$$

the addition of the gauge field A_μ to the derivative ensures that $D_\mu\psi$ has the same gauge transformation as ψ and so $\overline{\psi}D_\mu\psi$ is gauge invariant. This implies the gauge transformation

$$A_\mu(x) \to A'_\mu(x) = G(x)A_\mu(x)G^{-1}(x) + i(\partial_\mu G(x))G^{-1}(x). \tag{1.8}$$

The gluon field-strength tensor is given by

$$F_{\mu\nu} = \sum_a F^{(a)}_{\mu\nu} t^{(a)} \tag{1.9}$$

with

$$F^{(a)}_{\mu\nu} = \partial_\mu A^{(a)}_\nu - \partial_\nu A^{(a)}_\mu + f_{abc} A^{(b)}_\mu A^{(c)}_\nu, \tag{1.10}$$

where f_{abc} are the structure constants for SU(3):

$$[t^{(a)}, t^{(b)}] = i f_{abc} t^{(c)}. \tag{1.11}$$

Compared to the electromagnetic field-strength tensor, there is an extra non-Abelian quadratic term $f_{abc} A^{(b)}_\mu A^{(c)}_\nu$. This gives rise to 3-gluon and 4-gluon self-interactions, which causes the gluon gauge coupling g to run (i.e. change) with the energy scale. This is responsible for two of the main features of QCD:

1. *Asymptotic freedom*: At high energies, the coupling constant $\alpha_s \equiv g^2/4\pi$ is small. This gives rise to a gluon-quark plasma with free quarks.

2. *Confinement*: At low energies, the coupling constant $\alpha_s \sim 1$, and we find that free quarks are never observable; they only occur in colourless states known as *hadrons*. Hadrons come in two kinds, corresponding to the ways one can mix the three colour and three anti-colour states into a colourless state: baryons and anti-baryons which consist of three quarks or three anti-quarks, and mesons which consist of a quark and an anti-quark.

1.1.1 Euler-Lagrange equations

In classical mechanics, the equations of motion for a Lagrangian-based theory are the Euler-Lagrange equations. For a particular variable Q in the Lagrangian density $\mathcal{L}(Q, \partial_\mu Q, x)$, these take the form

$$\frac{\partial \mathcal{L}}{\partial Q} = \partial_\mu \left(\frac{\partial \mathcal{L}}{\partial (\partial_\mu Q)} \right), \tag{1.12}$$

where Q and $\partial_\mu Q$ are considered distinct variables. When we construct a Lagrangian for a field theory, we want to ensure that these equations of motion match up with what we expect classically.

The QCD Lagrangian (1.6) has two distinct Euler-Lagrange equations, given by differentiation with respect to ψ or $\overline{\psi}$ and with respect to A_μ. These equations of motion can be shown to match up with results from quantum mechanics.

In the case of the fermion field, the equations of motion are Hermitian conjugates. Thus, we just consider

$$\frac{\partial \mathcal{L}}{\partial \overline{\psi}} = \partial_\mu \left(\frac{\partial \mathcal{L}}{\partial(\partial_\mu \overline{\psi})} \right), \tag{1.13}$$

which can be shown to reduce to

$$i\gamma_\mu(\partial_\mu - iA_\mu)\psi - m\psi = 0. \tag{1.14}$$

This is simply the Dirac equation for a fermion in an external gauge field A_μ.

The equations of motion for A_μ are

$$\frac{\partial \mathcal{L}}{\partial A_\mu} = \partial_\nu \left(\frac{\partial \mathcal{L}}{\partial(\partial_\nu A_\mu)} \right), \tag{1.15}$$

which reduces to

$$\partial_\nu F^{\nu\mu} = g^2 \overline{\psi}\gamma_\mu \psi. \tag{1.16}$$

Comparing with electromagnetism, this equation can be identified as a relativistic wave equation for the gluon field.

1.1.2 Path integral formulation

In order to get predictions for physical quantities from QCD, we must be able to construct observables. There are a variety of ways these are mathematically constructed, but here we discuss the *path integral formulation* [8, 9], as it is the basis for Lattice QCD.

In the path integral formulation, we use the partition function

$$\mathcal{Z}_{QCD} = \int D\psi D\overline{\psi} DA_\mu \exp(iS_{QCD}), \tag{1.17}$$

where $S_{QCD} = \int d^4x \mathcal{L}_{QCD}(x)$ is the QCD action. This integral is a path integral over all possible values of the fields $\psi, \overline{\psi}, A_\mu$ at all points in space-time x. Evaluating this integral is problematic however, as the complex exponential leads to an oscillatory integrand. To avoid this, we Wick rotate from Minkowski space-time (^{M}) with metric tensor

$$\eta^M_{\mu\nu} = \text{diag}(1, -1, -1, -1) \tag{1.18}$$

to Euclidean space-time (^{E}) with metric tensor

$$\eta^E_{\mu\nu} = \text{diag}(1,1,1,1) = \delta_{\mu\nu}. \tag{1.19}$$

Under this transformation, we find that we have 'imaginary' time,

$$t^E = it^M, \tag{1.20}$$

and the partition function becomes

$$\mathcal{Z}^E_{QCD} = \int D\psi D\overline{\psi} DA_\mu \exp(-S^E_{QCD}), \tag{1.21}$$

with

$$\mathcal{L}^E_{QCD} = \overline{\psi}(\gamma_\mu D_\mu + m)\psi + \frac{1}{2g^2}\,\text{Tr}\big[F_{\mu\nu}F_{\mu\nu}\big], \quad D_\mu = \partial_\mu + iA_\mu, \tag{1.22}$$

and all the variables are now in Euclidean space-time. We will use Euclidean space-time for the rest of this thesis. Note that in this space-time, the metric tensor transforms into a delta function, and hence covariant and contra-variant indices are identical. To emphasise the difference, we use space-time indices $\mu = 1,2,3,4$ with $x^4 = t^E$.

We extract observables from this partition function through expectation values, which are given by

$$\langle O \rangle = \frac{1}{\mathcal{Z}_{QCD}} \int D\psi D\overline{\psi} DA_\mu O[\psi, \overline{\psi}, A_\mu] \exp(-S_{QCD}). \tag{1.23}$$

Here, the observable O is evaluated as an explicit functional of the gluon fields A_μ and fermion fields ψ, $\overline{\psi}$ within the path integral. See section 1.3 for a discussion of how these functionals can be constructed. In practice, this integral is evaluated using Monte Carlo methods by taking an average over a set of *gauge configurations*, which we introduce in section 1.3.1.

1.2 Discretisation to Lattice QCD

Lattice QCD comes from continuum QCD after discretising Euclidean space-time into a finite lattice Λ of points with

$$x^\mu = a^\mu n^\mu, \quad n^\mu = 1, \ldots, N_\mu, \tag{1.24}$$

where a^μ is the lattice spacing in each direction. It is common to choose isotropic lattices with a single lattice spacing $a = a^\mu$, the same spatial extent in each direction $N = N_{x,y,z}$ and a larger time extent $N_t > N$, with corresponding lengths $L = aN, L_t = aN_t$. Considering the path integral (1.21), this reduces the size of our phase space from infinite dimensions to a finite set of fields, thus making the evaluation of the path integral computationally tractable. All the operators on the fields also need to be discretised. For example, integrals over space-time are replaced by sums over the lattice

$$\int d^4x \to a^4 \sum_{x\in\Lambda}, \tag{1.25}$$

and derivatives are replaced by finite differences

$$\partial_\mu f(x) \to \frac{1}{2a}[f(x+a\hat{\mu}) - f(x-a\hat{\mu})] \tag{1.26}$$

where $\hat{\mu}$ is the unit vector in the μ direction.

The next step is to find a lattice version of the action S_{QCD}. A guiding principle will be that lattice quantities should match their continuum counterparts in the limit $a \to 0$. To start with, consider the free-particle fermion action where $A_\mu = 0$:

$$S_F^0 = \int d^4x \overline{\psi}(x)(\gamma_\mu \partial_\mu + m)\psi(x). \tag{1.27}$$

The lattice version of the action then seems to be

$$S_F^0 = a^4 \sum_{x\in\Lambda} \overline{\psi}(x)\left[\gamma_\mu \frac{\psi(x+a\hat{\mu}) - \psi(x-a\hat{\mu})}{2a} + m\psi(x)\right]. \tag{1.28}$$

This expression is not gauge invariant, but in a different way to the continuum. For example, consider the term $\overline{\psi}(x)\psi(x+a\hat{\mu})$ under a gauge transformation:

$$\begin{aligned} \overline{\psi}(x)\psi(x+a\hat{\mu}) \to \quad & \overline{\psi}'(x)\psi'(x+a\hat{\mu}) \\ & = \overline{\psi}(x)G^{-1}(x)G(x+a\hat{\mu})\psi(x+a\hat{\mu}). \end{aligned} \tag{1.29}$$

This bi-local transformation can't be fixed by adding a local operator $A_\mu(x)$ as in the continuum. What we require instead is a field $U_\mu(n)$ with directional index μ that transforms like

$$U_\mu(x) \to U'_\mu(x) = G(x)U_\mu(x)G^{-1}(x+a\hat{\mu}), \tag{1.30}$$

for then $\overline{\psi}(x)U_\mu(x)\psi(x+a\hat{\mu})$ is gauge-invariant. We call these bi-local fields $U_\mu(x)$ the *gauge links*, as they are elements of SU(3) Lie group and live on the links between lattice sites.

It is useful to define a gauge link in the negative direction $U_{-\mu}(x)$, which can be related to the usual gauge links by

$$U_{-\mu}(x) = U_\mu^\dagger(x - a\hat{\mu}). \tag{1.31}$$

We can then write a gauge-invariant fermion action called the *naive fermion action*:

$$S_F = a^4 \sum_{x\in\Lambda} \overline{\psi}(x)\left[\gamma_\mu \frac{U_\mu(x)\psi(x+a\hat{\mu}) - U_{-\mu}(x)\psi(x-a\hat{\mu})}{2a} + m\psi(x)\right]. \tag{1.32}$$

It is 'naive' as there is a subtle issue with it, which will be addressed in subsection 1.2.2 by adding an extra term that vanishes in the continuum.

Does (1.32) have the correct continuum limit as $a \to 0$? To take this limit, we require a continuum counterpart to $U_\mu(x)$ that transforms under local SU(3) transformations in the same way. The operator we use is the *gauge generator* $T(x,y)$. It is the path-ordered exponential integral of the gauge field $A_\mu(x)$ along a curve $\mathcal{C}_{xy}$ between two points in space-time

$$T(x,y) = P\exp\left(i\int_{\mathcal{C}_{xy}} A\cdot ds\right). \tag{1.33}$$

It transforms under gauge transformations with

$$T(x,y) \to G(x)T(x,y)G^\dagger(y). \tag{1.34}$$

To connect this to $U_\mu(x)$, we can identify the gauge links in the SU(3) Lie group as exponentials of the gauge fields $A_\mu(x)$ in the SU(3) Lie algebra:

$$U_\mu(x) = \exp(iaA_\mu(x)). \tag{1.35}$$

We thus have $U_\mu(x) = T(x, x+a\hat{\mu}) + \mathcal{O}(a^2)$, approximating the integral from x to $x+a\hat{\mu}$ by the length of the path a multiplied by the value of the field at the starting point $A_\mu(x)$.

Now we are in a position to consider the continuum limit $a \to 0$ of the naive fermion action (1.32). Considering expansions of $U_\mu(x)$ for small a,

$$U_\mu(x) = I + iaA_\mu(x) + \mathcal{O}(a^2) \tag{1.36}$$

$$\text{and} \qquad U_{-\mu}(x) = U_\mu^\dagger(x - a\hat{\mu}) = I - iaA_\mu(x - a\hat{\mu}) + \mathcal{O}(a^2), \tag{1.37}$$

we can substitute into the lattice action and obtain

$$\begin{aligned} S_F = a^4 \sum_{x\in\Lambda} \overline{\psi}(x) \Bigg[& \gamma_\mu \frac{\psi(x + a\hat{\mu}) - \psi(x - a\hat{\mu})}{2a} \\ & + \gamma_\mu \frac{iaA_\mu(x)\psi(x + a\hat{\mu}) + iaA_\mu(x - a\hat{\mu})\psi(x - a\hat{\mu})}{2a} \\ & + m\psi(x) + \mathcal{O}(a) \Bigg] \\ = a^4 \sum_{x\in\Lambda} \overline{\psi}(x) \Bigg[& \gamma_\mu \frac{\psi(x + a\hat{\mu}) - \psi(x - a\hat{\mu})}{2a} + i\gamma_\mu A_\mu(x)\psi(x) \\ & + m\psi(x) + \mathcal{O}(a) \Bigg]. \end{aligned} \tag{1.38}$$

On the last line, we used Taylor series expansions for $\psi(x \pm a\hat{\mu})$ and $A_\mu(x - a\hat{\mu})$, e.g.

$$\psi(x + a\hat{\mu}) = \psi(x) + a\partial_\mu\psi(x) + \mathcal{O}(a^2). \tag{1.39}$$

Taking the limit $a \to 0$ and applying (1.26) then gives the continuum form, namely

$$S_F^{QCD} = \int d^4x \overline{\psi}(x) \big[\gamma_\mu \partial_\mu + i\gamma_\mu A_\mu(x) + m \big] \psi(x). \tag{1.40}$$

As the lattice Λ is finite in extent, it is necessary to consider what to do at the boundaries. Periodic and anti-periodic boundary conditions are usually used as they conserve translational invariance. The gauge field U usually has periodic boundary conditions in all directions. As for the fermion fields $\psi, \overline{\psi}$, it is usual to have periodic boundary conditions in the spatial directions and anti-periodic boundary conditions in the temporal direction. This is done in order to preserve the time reflection symmetry of continuum QCD. Other choices of boundary conditions can help in certain domains however: for example, open boundary conditions in the time direction helps prevent poor sampling of the topological charge [10].

1.2.1 Gauge action

It is useful to consider the action as a sum of two terms

$$S = S_G[U] + S_F[U, \psi, \overline{\psi}], \tag{1.41}$$

consisting of *gluon action* S_G involving just the gauge fields A_μ, and the *fermion action* S_F which describes interactions with the fermion fields $\psi, \overline{\psi}$.

Recalling (1.22), the gluon action in the continuum is given by

$$S_G[U] = \frac{1}{2g^2} \int d^4x \, \mathrm{Tr}\big[F_{\mu\nu}(x)F_{\mu\nu}(x)\big]. \tag{1.42}$$

In order to discretise this term for lattice QCD, we require a gauge-invariant object made entirely out of gauge links. For this task, we can use a more substantial discretisation of the gauge generator $T(x,y)$ (1.33). Consider a path $\mathcal{P}$ of gauge links on the lattice. We can define the ordered product

$$P_{\mathcal{P}}[U] = U_{\mu_0}(x)U_{\mu_1}(x+a\hat{\mu}_1)\cdots U_{\mu_{k-1}}(y-a\hat{\mu}_{k-1}) = \prod_{(x,\mu)\in\mathcal{P}} U_\mu(x), \tag{1.43}$$

noting that each μ_i can be in any of the 8 possible directions. The most important feature of this path is that it has the gauge transformation

$$P[U] \to P[U'] = G(x)P[U]G(y)^{-1}, \tag{1.44}$$

because all the terms between gauge links cancel. It follows that the trace of a closed loop $\mathcal{L}$ of gauge links

$$L[U] = \mathrm{Tr}\left[\prod_{(x,\mu)\in\mathcal{L}} U_\mu(x)\right] \tag{1.45}$$

is gauge invariant:

$$L[U] \to L[U'] = \mathrm{Tr}\left[\prod_{(x,\mu)\in\mathcal{L}} G(x)U_\mu(x)G(x)^{-1}\right] = \mathrm{Tr}\left[\prod_{(x,\mu)\in\mathcal{L}} U_\mu(x)\right] = L[U]. \tag{1.46}$$

The simplest non-trivial gauge loop on the lattice is known as the *plaquette*: this is a product of four link variables

$$\begin{aligned} U_{\mu\nu}(x) &= U_\mu(x)U_\nu(x+a\hat{\mu})U_{-\mu}(x+a\hat{\mu}+a\hat{\nu})U_{-\nu}(x+a\hat{\nu}) \\ &= U_\mu(x)U_\nu(x+a\hat{\mu})U_\mu^\dagger(x+a\hat{\nu})U_\nu^\dagger(x), \end{aligned} \tag{1.47}$$

and is depicted in Figure 1.1. This leads to the simplest gluon action known as the *Wilson gauge action* [1], which is a sum over all plaquettes on the lattice:

$$S_G^{\text{Wilson}}[U] = \frac{2}{g^2} \sum_{x\in\Lambda} \sum_{\mu>\nu} \mathrm{Re}\,\mathrm{Tr}\big[\mathbb{I} - U_{\mu\nu}(x)\big]. \tag{1.48}$$

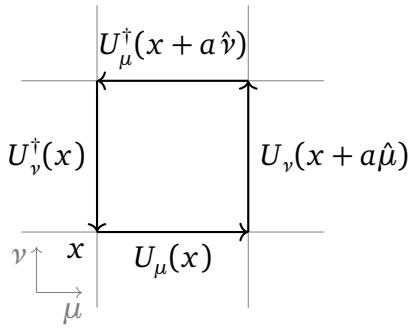

Figure 1.1: The plaquette $U_{\mu\nu}(x)$ on the μ, ν plane of a lattice.

We use this gauge action for several of our lattices as part of the filtering method investigations in chapters 3 and 4.

We will now show that the Wilson gauge action produces the correct continuum value (1.42) in the limit $a \to 0$. This requires an expansion of the plaquette $U_{\mu\nu}$ in terms of the Lie algebra gauge field A_μ

$$U_{\mu\nu}(x) = \exp[iaA_\mu(x)]\exp[iaA_\nu(x+a\hat{\mu})]\exp[-iaA_\mu(x+a\hat{\nu})]\exp[-iaA_\nu(x)]. \tag{1.49}$$

This is a product of matrix exponentials, so the Baker-Campbell-Hausdorff formula is useful:

$$\exp(aA)\exp(aB) = \exp\left(aA + aB + \frac{1}{2}a^2[A,B] + \mathcal{O}(a^3)\right). \tag{1.50}$$

Applying this to the expression (1.49) repeatedly and neglecting $\mathcal{O}(a^3)$ terms, we get

$$\begin{aligned} U_{\mu\nu}(x) = \exp\Big(& iaA_\mu(x) + iaA_\nu(x+a\hat{\mu}) - iaA_\mu(x+a\hat{\nu}) - iaA_\nu(x) \\ & - \frac{a^2}{2}[A_\mu(x), A_\nu(x+a\hat{\mu})] + \frac{a^2}{2}[A_\mu(x), A_\mu(x+a\hat{\nu})] \\ & + \frac{a^2}{2}[A_\nu(x+a\hat{\mu}), A_\mu(x+a\hat{\nu})] + \frac{a^2}{2}[A_\mu(x), A_\nu(x)] \\ & + \frac{a^2}{2}[A_\nu(x+a\hat{\nu}), A_\nu(x)] - \frac{a^2}{2}[A_\mu(x+a\hat{\nu}), A_\nu(x)] + \mathcal{O}(a^3)\Big). \end{aligned} \tag{1.51}$$

Noting that we have gauge fields in terms of shifted arguments, e.g. $A_\mu(x + a\hat{\mu})$, we now insert Taylor series expansions of these fields (1.39) and consider contributions up to $\mathcal{O}(a^2)$. Doing this results in

$$\begin{aligned} U_{\mu\nu}(x) &= \exp\left(ia^2\left(\partial_\mu A_\nu(x) - \partial_\nu A_\mu(x) + i[A_\mu(x), A_\nu(x)]\right) + \mathcal{O}(a^3)\right) \\ &= \exp\left(ia^2 F_{\mu\nu}(x) + \mathcal{O}(a^3)\right), \end{aligned} \tag{1.52}$$

recalling the field strength tensor definition (1.10) in the continuum. We can insert this equation into the Wilson gauge action (1.48) and expand the exponential to get

$$\begin{aligned} S_G^{\text{Wilson}}[U] &= \frac{2}{g^2}\sum_{x\in\Lambda}\sum_{\mu>\nu}\text{Re}\,\text{Tr}\Big[-ia^2F_{\mu\nu}(x)+\frac{1}{2}a^4F_{\mu\nu}^2(x)+\mathcal{O}(a^6)\Big] \\ &= \frac{a^4}{2g^2}\sum_{x\in\Lambda}\sum_{\mu,\nu}\Big(\text{Tr}\Big[F_{\mu\nu}^2(x)\Big]+\mathcal{O}(a^2)\Big). \end{aligned} \tag{1.53}$$

The $\mathcal{O}(a^3)$ terms in the expansion of the exponential are purely imaginary and hence disappear when taking the real part of $\text{Tr}[1-U_{\mu\nu}]$, leaving $\mathcal{O}(a^2)$ errors as noted. Taking the limit $a\to 0$ then produces

$$\lim_{a\to 0} S_G^{\text{Wilson}}[U] = \frac{1}{2g^2}\int d^4x\sum_{\mu,\nu}\text{Tr}[F_{\mu\nu}^2(x)] = S_G[U] \tag{1.54}$$

as desired (cf. (1.42)).

Improved gauge actions

We can improve upon the $\mathcal{O}(a^2)$ errors in the Wilson gauge action by using the Symanzik improvement scheme [11]. The general idea of this scheme is as follows: consider a lattice quantity which approximates some continuum operator. Then, introduce lattice discretisations of higher order terms which are possible corrections to the operator. Finally, combine these terms such that errors up to a certain order disappear.

Given the plaquette is of order 4 (i.e. it has 4 gauge links), we introduce loops of the next possible order, 6. The most general improved action with $\mathcal{O}(a^4)$ errors takes the form [12]

$$\begin{aligned} S_G^{imp}[U] = c_0 S_G^{\text{Wilson}}[U] &+ c_1\frac{2}{g^2}\sum_x\sum_{\text{loops}}\text{Re}\,\text{Tr}\,[I-U_1(x)] \\ &+ c_3\frac{2}{g^2}\sum_x\sum_{\text{loops}}\text{Re}\,\text{Tr}\,[I-U_3(x)], \end{aligned} \tag{1.55}$$

and is known as the *tree-level improved* or the *Lüscher-Weisz* gauge action.

U_1 and U_3 are the gauge loops of order 6 depicted in Figure 1.2, which are summed over all lattice sites in one direction. The other gauge loop with order 6, U_2, introduces $\mathcal{O}(a^2)$ errors to low lying energy terms that are linearly independent from the other gauge loops, so it is disallowed.

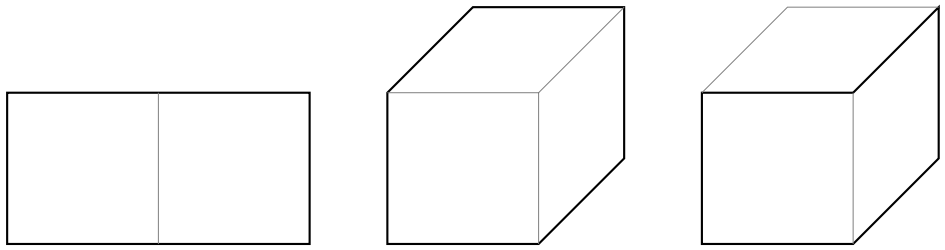

Figure 1.2: All possible gauge loops of order 6. The last two loops U_2, U_3 are three-dimensional.

The real coefficients c_i are normalised such that

$$c_0 + 8c_1 + 16c_3 = 1 \tag{1.56}$$

and must satisfy

$$c_1 - c_3 = -\frac{1}{12} \tag{1.57}$$

to cancel off the $\mathcal{O}(a^2)$ terms. Therefore, these coefficients have one degree of freedom, which we can parametrise using c_3:

$$c_0 = \frac{5}{3} - 24c_3, \tag{1.58a}$$

$$c_1 = c_3 - \frac{1}{12}. \tag{1.58b}$$

We also require the gauge action to be positive, which restricts the values that c_3 can take. A popular choice [13] sets $c_3 = 0$ and just uses the plaquette $U_{\mu\nu}$ and rectangular loops U_1 with $c_0 = \frac{5}{3}, c_1 = -\frac{1}{12}$; we use this choice in the investigation of improving the generation of configurations with electromagnetic corrections in chapter 5.

1.2.2 Fermion action: a problem with doublers

We have noted that the naive fermion action (1.32)

$$S_F = a^4 \sum_{x\in\Lambda} \overline{\psi}(x) \left[\sum_\mu \gamma_\mu \frac{U_\mu(x)\psi(x+a\hat{\mu}) - U_{-\mu}(x)\psi(x-a\hat{\mu})}{2a} + m\psi(x) \right]$$

is problematic. The issue is that this action only couples fermion fields with the gauge field at sites $2a$ apart, leading to two uncoupled fermion fields in each

dimension for a total of $2^4 = 16$ uncoupled fermion fields per fermion flavour. These extra fermion fields, called *fermion doublers*, are unphysical and need to be managed.

Wilson [1] first proposed a solution to this by adding a new term to the fermion action which involves a discretisation of the covariant Laplacian operator:

$$\begin{aligned}\overline{\psi}(x)\Delta\psi(x) &= -\overline{\psi}(x)\sum_{\mu}\frac{U_\mu(x)\psi(x+a\hat{\mu})-2\psi(x)+U_{-\mu}(x)\psi(x-a\hat{\mu})}{2a}\\ &= -\frac{a}{2}\overline{\psi}(x)\sum_{\mu}D_\mu D_\mu\psi(x)+\mathcal{O}(a^2).\end{aligned} \tag{1.59}$$

The factor of a in the continuum limit shows that this term has a higher order 5 than the order 4 terms in the naive fermion action (1.32), and so this extra term disappears in the continuum limit as required. Adding this to the naive fermion action gives the Wilson fermion action

$$\begin{aligned}S_F^{\text{Wilson}} &= a^4\sum_{x\in\Lambda}\overline{\psi}(x)\sum_{\mu}\Big[\gamma_\mu\frac{U_\mu(x)\psi(x+a\hat{\mu})-U_{-\mu}(x)\psi(x-a\hat{\mu})}{2a}\\ &\qquad\qquad + \Delta\psi(x)+m\psi(x)\Big]\\ &= a^4\sum_{x\in\Lambda}\Big[\overline{\psi}(x)\left(m+\frac{4}{a}\right)\psi(x)\\ &\qquad -\frac{1}{2a}\sum_{\mu}(1-\gamma_\mu)U_\mu(x)\psi(x+a\hat{\mu})\psi(x)\\ &\qquad -\frac{1}{2a}\sum_{\mu}(1+\gamma_\mu)U_{-\mu}(x)\psi(x-a\hat{\mu})\psi(x)\Big].\end{aligned} \tag{1.60}$$

As this is bilinear in $\psi, \overline{\psi}$, we can write this in the form

$$S_F^{\text{Wilson}} = a^4\sum_{x,y\in\Lambda}\sum_{a,b,\alpha,\beta}\overline{\psi}(x)_{a\alpha}D_{ab\alpha\beta}(x|y)\psi(y)_{b\beta} \tag{1.61}$$

where $D(x|y)$ is known as the *Dirac operator*. In the case of the Wilson fermion action, we have Dirac operator

$$D_{ab\alpha\beta}(x|y) = \left(m+\frac{4}{a}\right)\delta_{xy}\delta_{ab}\delta_{\alpha\beta}+\sum_{\mu=\pm1}^{\pm4}(1-\gamma_\mu)_{\alpha\beta}U_\mu(x)_{ab}\delta_{x+a\hat{\mu},y}, \tag{1.62}$$

defining $\gamma_{-\mu} = -\gamma_\mu$ for notational convenience. We can simplify this construct by

scaling our fermion fields $\psi \to \frac{1}{\sqrt{2\kappa}}\psi$ where $\kappa = \frac{1}{2am+8}$ is the hopping parameter, such that the Wilson Dirac operator becomes

$$D = 1 - \kappa H \tag{1.63}$$

where

$$H_{ab\alpha\beta}(x|y) = \sum_{\mu=\pm1}^{\pm4} (1-\gamma_\mu)_{\alpha\beta} U_\mu(x)_{ab} \delta_{x+a\hat{\mu},y} \tag{1.64}$$

is known as the hopping matrix. The Wilson fermion action is used in all the lattices in this paper (see chapters 3, 4 and 5), though some lattices have an improvement term which we now discuss.

1.2.3 Fermion action: improved actions

Improved discretisation errors

Adding the Wilson term to the fermion action increases the error to $\mathcal{O}(a)$. To improve the error back to $\mathcal{O}(a^2)$, we use the Symanzik improvement scheme, which leads to the improved action [14]

$$S_F^I = S_F^{\text{Wilson}} + c_{SW} a^5 \sum_{x\in\Lambda} \sum_{\mu<\nu} \overline{\psi}(x) \frac{1}{2} \sigma_{\mu\nu} \hat{F}_{\mu\nu}(x) \psi(x), \tag{1.65}$$

where $c_{SW} \in \mathbb{R}$ is the Sheikholeslami-Wohlert coefficient, $\sigma_{\mu\nu} = [\gamma_\mu, \gamma_\nu]/2i$ and $\hat{F}_{\mu\nu}$ is a discretisation of the field strength tensor. The discretisation is usually taken to be

$$\hat{F}_{\mu\nu}(x) = \frac{-i}{8a^2}(Q_{\mu\nu}(x) - Q_{\nu\mu}(x)) \tag{1.66}$$

where $Q_{\mu\nu}$ is a sum of plaquettes in μ, ν plane

$$Q_{\mu\nu}(x) \equiv U_{\mu\nu}(x) + U_{\nu,-\mu}(x) + U_{-\mu,-\nu}(x) + U_{-\nu,\mu}(x). \tag{1.67}$$

This construct is depicted in Figure 1.3, and is reminiscent of a four-leaf clover. The action term is thus referred to as the *clover term* or *clover improvement*. This improvement is used for some of our lattices in chapters 4 and 5.

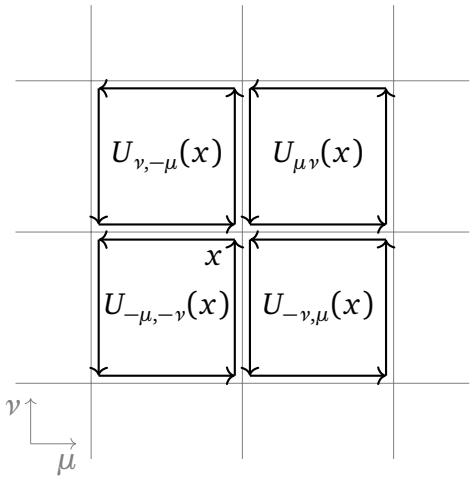

Figure 1.3: The clover term $Q_{\mu\nu}(x)$ on the μ, ν plane of a lattice, which is a sum of 4 plaquettes. The gauge links U are represented by arrows, and are slightly offset to emphasize the structure.

The Sheikholeslami-Wohlert coefficient c_{SW} is set such that the $\mathcal{O}(a)$ order terms in the action are cancelled, and depends non-trivially on the gauge coupling g. It can be calculated directly in Lattice QCD via the partially conserved axial current (PCAC) relation; see e.g. [15] for an explanation of this approach.

Chiral symmetry

The Wilson term, and hence many lattice actions, breaks chiral symmetry in the massless $m_q = 0$ limit. In the continuum, this symmetry in the Lagrangian appears in the massless limit, and is given by the chiral transformation

$$\psi \to \psi' = e^{i\alpha\gamma_5}\psi \quad \text{and} \quad \overline{\psi} \to \overline{\psi}' = \overline{\psi}e^{i\alpha\gamma_5} \tag{1.68}$$

where $\gamma_5 = \gamma_1\gamma_2\gamma_3\gamma_4$. This causes a decoupling of left-handed and right-handed fermion fields ψ_L, ψ_R. This symmetry is spontaneously broken in the continuum, which gives rise to several effects such as the small mass of the pion.

Restoring this chiral symmetry in our fermion action turns out to be impossible without sacrificing more vital properties of the fermion action, as it is part of a no-go theorem [16]. We do not investigate it here, but this can be circumvented by defining a lattice-deformed version of chiral symmetry which has the correct continuum limit; this leads to e.g. overlap fermions [17] and domain wall fermions [18].

1.3 Making measurements

Recall that the key construct for extracting physical information in QCD is the expectation value (1.23). For Lattice QCD, this is given by the path integral

$$\langle O \rangle = \frac{1}{\mathcal{Z}} \int D\psi D\overline{\psi} DU_\mu \, O[\psi, \overline{\psi}, U_\mu] \exp(-S[\psi, \overline{\psi}, U_\mu]). \tag{1.69}$$

This section gives an introduction to how we can measure physical properties using this construct. While we do not make such measurements, understanding them is important for determining the part that gauge configurations play, as generating these is the main focus of this work.

1.3.1 The fermionic integral

Fermions $\psi, \overline{\psi}$ obey Fermi statistics, so the fields anti-commute in all combinations. It follows that the fields are Grassmann numbers, which complicates the evaluation of the expectation value path integrals.

To consider just the fermionic integral, we separate the expectation value into fermionic and gluonic parts

$$\langle A \rangle = \langle \langle A \rangle_F \rangle_G, \tag{1.70}$$

defining

$$\langle A \rangle_F = \frac{1}{\mathcal{Z}_F[U]} \int D\psi D\overline{\psi} \, A[\psi, \overline{\psi}, U] e^{-S_F[\psi, \overline{\psi}, U]} \tag{1.71}$$

with fermionic partition function

$$\mathcal{Z}_F[U] = \int D\psi D\overline{\psi} \, e^{-S_F[\psi, \overline{\psi}, U]}, \tag{1.72}$$

and

$$\langle B \rangle_G = \frac{1}{\mathcal{Z}} \int DU \, B[U] e^{-S_G[U]}. \tag{1.73}$$

Given that the fermion fields are Grassmann numbers, we can evaluate the fermionic partition function by applying the *Matthews-Salam formula* [19], giving

$$\begin{aligned} \mathcal{Z}_F[U] &= \int D\psi D\overline{\psi} \, \exp\left(-\sum_f \overline{\psi}^f D^f \psi^f \right) \\ &= \prod_f \det[D^f]. \end{aligned} \tag{1.74}$$

Note that we have included the implicit sum over quark flavour f in the fermion action for clarity. It follows that for purely gluonic observables $B[U]$, the full expectation value is given by

$$\langle B \rangle = \frac{\int DU \prod_f \det[D^f] B[U] e^{-S_G[U]}}{\int DU \prod_f \det[D^f] e^{-S_G[U]}}. \tag{1.75}$$

This integral is suitable for Monte Carlo methods; if we can generate a set of fields $\{U_i\}$ distributed according to $\frac{1}{Z}\prod_f \det[D^f] e^{-S_G[U]}$, evaluating this integral is simply a matter of taking the average of the operator over U_i,

$$\langle B \rangle \approx \frac{1}{N} \sum_{i=1}^{N} B[U_i]. \tag{1.76}$$

The fields U_i are called *gauge configurations*. Generating such configurations is highly non-trivial in practice, due in part to the size of D^f. This process will be discussed at length in chapter 2.

1.3.2 The Euclidean correlator

The most important kind of expectation value we take is called the *Euclidean correlator*, which consists of two operators $\hat{O}_1$ and $\hat{O}_2$ which only act on lattice sites with Euclidean times t_1 and $t_2 > t_1$ respectively:

$$\langle \hat{O}_2(t_2)\hat{O}_1(t_1)\rangle \equiv \frac{1}{\mathcal{Z}} \int D\psi D\overline{\psi} DU_\mu \, O_2[t = t_2; \psi, \overline{\psi}, U_\mu] O_1[t = t_1; \psi, \overline{\psi}, U_\mu] \times \exp(-S[\psi, \overline{\psi}, U_\mu]). \tag{1.77}$$

On the left hand side we have operators $\hat{O}$, whereas on the right hand side we have functionals O that map the values of the complex fields $\psi, \overline{\psi}, U_\mu$ at all lattice sites $(\vec{x}, t)$ to a complex value. The power of the Euclidean correlator is that it is equal to (see e.g. [20] for a derivation)

$$\langle \hat{O}_2(t_2)\hat{O}_1(t_1)\rangle = \sum_n \langle \Omega|\hat{O}_2|n\rangle \langle n|\hat{O}_1|\Omega\rangle e^{-tE_n}, \tag{1.78}$$

where $t = t_2 - t_1$, $|\Omega\rangle$ is the QCD vacuum state, and E_n and $|n\rangle$ are the eigenvalues and eigenstates of the Hamiltonian operator $\hat{H}$ of our system

$$\hat{H}|n\rangle = E_n|n\rangle \tag{1.79}$$

with $E_0 \leq E_1 \leq \ldots \leq E_k \leq \ldots$. From this construction, it is possible to extract the energies E_n and matrix elements, which then lead to physical properties.

As we are interested in the properties of particles, one of the most common operators used in the Euclidean correlator are creation/annihilation operators. These are operators $\hat{O}$ where $\hat{O}^\dagger$ (mesons)/$\overline{O}$ (baryons) creates a particle and $\hat{O}$ destroys a particle with a particular set of quantum numbers. The operator $\hat{O}$ can also create a state with the quantum numbers of the corresponding anti-particle. These operators are formulated in terms of the fermion fields, combined with gamma matrices and other algebraic objects such that the desired quantum numbers and symmetries of the particle are obtained. The simplest example is the quark field itself: $\psi^f(x)$ annihilates a quark or creates an anti-quark with flavour f at x, while the conjugate $\overline{\psi}^f(x)$ creates a quark or annihilates an anti-quark with flavour f at x.

Creation/annihilation operators can be used in the Euclidean correlator to extract energy levels and matrix elements corresponding to the particle of interest. For example, let $\overline{O}_N(x)$ be an operator which creates a state with the quantum numbers of the neutron at x. Then the Euclidean correlator for a neutron propagating from space-time point 0 to $x = (\vec{x}, t)$ is

$$\langle O_N(\vec{x}, t)\overline{O}_N(\vec{0}, 0)\rangle = \sum_n \langle\Omega|O_N(x)|n\rangle\langle n|\overline{O}_N(0)|\Omega\rangle e^{-tE_n}. \tag{1.80}$$

This can be simplified by noting that the matrix elements will only be non-zero when the state $|n\rangle$ has the same quantum numbers as the neutron. Taking these states to be $|N_i\rangle$ with energy levels $E_i^{(N)}$, the Euclidean correlator is thus equal to

$$\langle O_N(\vec{x}; t)\overline{O}_N(\vec{0}; 0)\rangle = \sum_{N_i} \langle\Omega|O_N(x)|N_i\rangle\langle N_i|\overline{O}_N(0)|\Omega\rangle e^{-tE_i^{(N)}}. \tag{1.81}$$

This construction is known as a *particle correlator*. From this, we can extract mass of the neutron $E_0^{(N)}$ and its excited states $E_i^{(N)}$ along with its matrix elements.

Momentum projection

It is often more convenient to consider operators with a definite momentum $\vec{p}$ rather than definite position $\vec{x}$. To accommodate this, we can Fourier-project the operators

$$\tilde{O}(\vec{p}) = \sum_{\vec{x}} e^{-i\vec{p}\cdot\vec{x}} O(\vec{x}), \tag{1.82}$$

and hence construct the momentum-projected correlator

$$\begin{aligned}\langle \tilde{O}(\vec{p};t)\overline{O}(\vec{x}=\vec{0};0)\rangle &= \sum_{\vec{x}} e^{-i\vec{p}\cdot\vec{x}}\langle O(\vec{x};t)\overline{O}(\vec{0};0)\rangle \\ &= \sum_{n}\langle\Omega|\tilde{O}(\vec{p})|n\rangle\langle n|\overline{O}(\vec{0})|\Omega\rangle e^{-tE_n(\vec{p})},\end{aligned} \tag{1.83}$$

where now the states $|n\rangle$ have definite momentum $\vec{p}$. Note that the creation operator $\overline{O}(0)$ remains in position space.

At zero momentum $\vec{p}=\vec{0}$, the correlator is exactly (1.81) summed over $\vec{x}$. At non-zero momenta $\vec{p}$, the energies $E_i(\vec{p})$ should follow the dispersion relation

$$E(\vec{p}) = \sqrt{E(\vec{0})+|\vec{p}|^2} \tag{1.84}$$

up to discretisation errors.

1.3.3 Extracting the ground state energy

At large times t, exponentials in the Euclidean correlator (1.78) with larger energy terms drop out, leaving the ground state: for a particle correlator,

$$\langle\hat{O}(t)\hat{O}^\dagger(0)\rangle \underset{t \text{ large}}{\to} \langle\Omega|\hat{O}|0\rangle\langle 0|\hat{O}^\dagger|\Omega\rangle e^{-tE_0}. \tag{1.85}$$

To extract the ground state energy, which corresponds to the mass of the particle, we can construct the *effective mass*

$$m_{\text{eff}}\left(t+\frac{a}{2}\right) \equiv \ln\frac{\langle\hat{O}(t)\hat{O}^\dagger(0)\rangle}{\langle\hat{O}(t+a)\hat{O}^\dagger(0)\rangle}, \tag{1.86}$$

which at large t tends to the ground state energy,

$$m_{\text{eff}}(t) \underset{t \text{ large}}{\to} E_0. \tag{1.87}$$

We can thus determine E_0 by fitting to a plateau in the effective mass. Such a fit is generally required because at early times t the higher energy states still dominate while at large times the signal is lost to noise.

For a more detailed discussion, including how to extract higher energy levels and matrix elements, see e.g. [20].

1.3.4 Fermionic operators

When considering observables involving fermions, we can simplify the fermionic expectation value using *Wick's Theorem*:

Theorem 1.1 (Wick's Theorem)**.** Let $a_i, \overline{a}_i$ where $i = 1, \ldots, N$ be a set of $2N$ Grassmann numbers and M be a $N \times N$ complex matrix. Then for some set of n indices $i_k, j_k \in 1, \ldots, N$, we have

$$\begin{aligned}\frac{1}{Z_F}\int \prod_{k=1}^{N}(da_k d\overline{a}_k)\prod_{m=1}^{n}(a_{i_m}\overline{a}_{j_m})\exp\left(\sum_{b,c=1}^{N}\overline{a}_b M_{bc}a_c\right)\\ = (-1)^n \sum_{P(1,2,\ldots,n)} \mathrm{sign}(P)(M^{-1})_{i_1,j_{P_1}}(M^{-1})_{i_2,j_{P_2}}\ldots(M^{-1})_{i_n,j_{P_n}},\end{aligned} \tag{1.88}$$

where the sum on the second line indicates a sum over all possible permutations $P(1,2,\ldots,n)$ of the numbers $1,2,\ldots,n$, which are assigned to the numbers P_i, and $\mathrm{sign}(P)$ is the sign of the permutation.

This theorem with $N = 1$ implies that the fermion expectation value

$$\langle \psi(y)_{a\alpha}\overline{\psi}(x)_{b\beta}\rangle_F = a^{-4}D^{-1}(y|x)_{a\alpha,b\beta}. \tag{1.89}$$

Thus, the Dirac matrix inverse D^{-1} is denoted the *quark propagator*, as it describes a quark propagating from x to y.

For more complicated constructions involving multiple fermions, Wick's Theorem implies that we take a sum over all possible *Wick contractions*. This is where we take all possible rearrangements the product of fermion fields and other operators such that every fermion field $\psi^{(f)}$ is on the left of a matching anti-fermion field $\overline{\psi}^{(f)}$ with the same flavour: if any fields remain unmatched, the observable is unphysical. For each valid permutation, we then factorize the fermionic expectation value into one for each flavour of quark, then replace expectation values with quark propagators (1.89).

For example, a lattice operator that creates a particle with the quantum numbers of the π^+ is

$$O^{\dagger}_{\pi^+}(x) = -\overline{u}(x)\gamma_5 d(x), \tag{1.90}$$

where $f(x) \equiv \psi^{(f)}(x)$ for notational convenience. This has overlap with the continuum operator that actually creates the π^+: in practice, there are many lattice operators with the same quantum numbers. The fermionic part of the

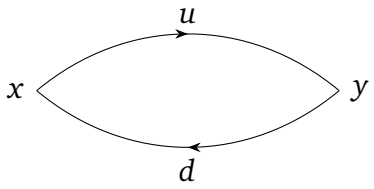

Figure 1.4: The pion correlator

correlator for a propagating π^+ is then given by (using summation notation)

$$\begin{aligned}
\langle O_{\pi^+}(y) O^\dagger_{\pi^+}(x)\rangle_F &= -\langle \overline{d}(y)\gamma_5 u(y)\overline{u}(x)\gamma_5 d(x)\rangle_F \\
&= -(\gamma_5)_{\alpha\beta}(\gamma_5)_{\delta\gamma}\langle \overline{d}(y)_{a\alpha} u(y)_{a\beta}\overline{u}(x)_{b\delta} d(x)_{b\gamma}\rangle_F \\
&= (\gamma_5)_{\alpha\beta}(\gamma_5)_{\delta\gamma}\langle u(y)_{a\beta}\overline{u}(x)_{b\delta} d(x)_{b\gamma}\overline{d}(y)_{a\alpha}\rangle_F \\
&= (\gamma_5)_{\alpha\beta}(\gamma_5)_{\delta\gamma}\langle u(y)_{a\beta}\overline{u}(x)_{b\delta}\rangle_u \langle d(x)_{b\gamma}\overline{d}(y)_{a\alpha}\rangle_d \\
&= (\gamma_5)_{\alpha\beta}(\gamma_5)_{\delta\gamma} D_u^{-1}(y|x)_{a\beta,b\delta} D_d^{-1}(x|y)_{b\gamma,a\alpha} \\
&= \mathrm{Tr}[\gamma_5 D_u^{-1}(y|x)\gamma_5 D_d^{-1}(x|y)]. \qquad (1.91)
\end{aligned}$$

This trace can be visualised as a u quark propagator going from x to y, and a d quark propagator going in the opposite direction, shown in Figure 1.4. Such a visualisation is useful for determining which Wick contractions are required.

The full expectation value for this π^+ correlator in a theory with just the up and down quarks is then given by

$$\begin{aligned}
\langle O_{\pi^+}(y)\overline{O}_{\pi^+}(x)\rangle = -\frac{1}{Z}\int D[U] e^{-S_G[U]} \det[D_u]\det[D_d] \\
\times \mathrm{Tr}[\gamma_5 D_u^{-1}(y|x)\gamma_5 D_d^{-1}(x|y)] \qquad (1.92)
\end{aligned}$$

where

$$Z = \int D[U] e^{-S_G[U]} \det[D_u]\det[D_d]. \qquad (1.93)$$

Supposing we have U_i distributed according to $\frac{1}{Z}\exp(-S_G[U])\det[D_u]\det[D_d]$, which will be discussed in chapter 2, we can evaluate this as an average of the trace over the ensemble of configurations.

1.4 Extrapolation to the physical point

We want to be able to calculate physical quantities from Lattice QCD. Theoretically, this requires quark masses at their physical point, infinitely small lattice

spacing, and infinitely large lattice extents. This is impossible to do in practice, so instead we calculate on lattices with heavy quark masses, finite spacing and finite extent, then extrapolate the observed quantities to the physical point. In this work, we investigate several configuration generation improvements that can help produce lattices closer to the physical point and hence improve predictions.

1.4.1 Quark mass extrapolation: $m_q \to m_{\text{phys}}$

It is only recently that we have been able to simulate Lattice QCD with quark masses at the physical point [2, 4, 5, 21], i.e. with particle masses at their physical value. This is because using such light masses is extremely computationally intensive: improving this situation is the main topic of this thesis. Hence, we still usually have unphysically heavy quarks, and we need to extrapolate the quark masses to the physical point. This is often referred to as *chiral extrapolation*, as in the two-flavour up/down quark case, the quark masses at the physical point are almost zero and hence almost at the continuum chirally-symmetric point $m_q = 0$.

The framework of *chiral perturbation theory* (χPT) is usually used for the physical mass extrapolation. This process involves selecting some reference meson masses that fully represent the quark matter present in the simulation: usually the pion π for the up and down, and the kaon K if strange is involved. Then we expand all others hadron masses in terms of these reference mass parameters about the physical point $m_q = m_{\text{phys}}$, taking into account the symmetries of QCD; this is distinct from pure χPT, where we expand about the massless point $m_q = 0$. The resultant expressions can be fit to mass data from different lattices to determine the coefficients, and hence provide predictions for hadron masses aside from the reference particles.

In two-flavour simulations where the up and down quarks are degenerate, choosing suitable lattices for this extrapolation is straightforward as there is only one bare mass parameter. When we add more quarks with different physical masses, e.g. the strange quark, we must decide on an extrapolation trajectory in quark mass space to the physical point. Many set the strange quark mass to its physical value, then go along a trajectory reducing the up/down quark mass. QCDSF [22] uses a 'symmetric point' extrapolation, where one starts at the point where all three quark masses are equal and have the same sum as in the continuum, then extrapolates towards the physical point. SU(3) flavour singlet quantities, i.e. quantities invariant under a permutation of the quark flavours $\{u, d, s\}$, remain relatively constant along this path, so this simplifies some of the

χPT expressions for hadron masses.

1.4.2 Finite volume extrapolation: $L \to \infty$

We are limited to finite lattice sizes, and quantities observed in a box are different to those observed in free space. Such effects are known as *finite volume effects*. These effects are observed to become small past a certain lattice size, with a common rule-of-thumb being $m_\pi L > 4$. To show that the finite volume effects of an extrapolation are minimal, it is usual to compare the values of several quantities (e.g. meson masses) across different lattice sizes at similar bare quark masses. After a certain size, the quantity should stabilise, indicating where finite volume effects can be neglected.

1.4.3 Finite spacing extrapolation: $a \to 0$

The continuum extrapolation $a \to 0$ is required for physical results. However, we must simultaneously increase the number of lattice sites in order to avoid finite volume effects, thus increasing the computation cost. A common way to take this limit is to sample several values of a at a nearly-fixed volume $V = L^3 \times L_T$, then extrapolate to $a = 0$.

∘ 2 ∘

Gauge configuration generation

In order to effectively evaluate expectation values in Lattice QCD (1.69)

$$\langle O \rangle = \frac{1}{\mathcal{Z}} \int D\psi D\overline{\psi} DU_\mu \, O[\psi, \overline{\psi}, U_\mu] \exp(-S[\psi, \overline{\psi}, U_\mu]),$$

we must have a large set of configurations U distributed according to $\exp(-S)$ (recalling section 1.3.1). This chapter is devoted to describing how we generate such configurations.

2.1 Markov chains

A Markov chain is a series of states

$$X_0 \to X_1 \to X_2 \to \ldots \to X_n \to \ldots \tag{2.1}$$

for some system produced by a Markov process, where each successive state is produced by an update step that only depends on the previous state. The Markov process is characterised by the conditional transitional probability distribution $P(X'|X)$, which reads "the probability of going to state X' given we are at state X". Here, we only consider time-homogeneous Markov chains, so the transition probability is independent of the step number n. By the basic properties of probabilities, $P(X'|X)$ satisfies

$$0 \leq P(X'|X) \leq 1 \quad \text{and} \quad \int dX' \, P(X'|X) = 1. \tag{2.2}$$

If we denote the probability of being in state X at step n as $P_n(X)$, we can write a recursion relation

$$P_{n+1}(X) = \int dX' P(X|X') P_n(X'), \tag{2.3}$$

where the right-hand side is the net probability of transitioning to state X from all possible states X' at step n, including $X' = X$.

In order to generate configurations for Lattice QCD, we want a Markov chain that samples a pre-determined probability distribution, namely $\exp(-S)$. This requires the target distribution to be the *equilibrium distribution* of the Markov chain. Denoting the target distribution as $\Pi(X)$, the required properties are:

- $\Pi(X)$ must be a *stationary distribution*. That is, if $P_n(X) = \Pi(X)$ then $P_{n+1}(X) = \Pi(X)$. Using the recursion relation (2.3), this means that

$$\Pi(X) = \int dX' P(X|X')\Pi(X'). \tag{2.4}$$

  Inserting a complete sum of probabilities (2.2) then gives the *balance equation*

$$\int dX' P(X'|X)\Pi(X) = \int dX' P(X|X')\Pi(X'). \tag{2.5}$$

- For any initial probability distribution $P_0(X)$, we have

$$\lim_{n\to\infty} P_n(X) = \Pi(X). \tag{2.6}$$

  This also implies that $\Pi(X)$ is a unique stationary distribution.

The Markov chain must satisfy certain properties in order for an equilibrium distribution to exist, which will now be described.

Definition 2.1. Denote the probability of a state X reaching state X' for the first time after n steps as $f^{(n)}(X'|X)$. Then the state X is *recurrent* iff

$$\sum_{n=1}^{\infty} f^{(n)}(X|X) = 1, \tag{2.7}$$

i.e. starting from state X, we will always reach X again. A Markov chain is recurrent iff all states X are recurrent.

Definition 2.2. A state X is *positive recurrent* iff the mean recurrence time is finite:

$$\sum_{n=1}^{\infty} n f^{(n)}(X|X) < \infty. \tag{2.8}$$

A Markov chain is positive recurrent iff all states X are positive recurrent.

Definition 2.3. A state X is *aperiodic* iff returns to a particular state X can occur sequentially. Formally,

$$k = \gcd\left\{n > 0 : f^{(n)}(X|X) > 0\right\} = 1 \tag{2.9}$$

where 'gcd' denotes the greatest common divisor. A sufficient condition for aperiodicity is

$$P(X|X) > 0. \tag{2.10}$$

A Markov chain is aperiodic iff all states X are aperiodic.

Definition 2.4. A Markov chain is *ergodic* iff it is both aperiodic and positive recurrent. A sufficient condition for ergodicity is that for every X, X',

$$P(X'|X) > 0. \tag{2.11}$$

Definition 2.5. A Markov chain is *irreducible* iff for each state X and $X' \neq X$, there exists an integer $n > 0$ such that

$$g^{(n)}(X'|X) = \int dX_1 dX_2 \dots dX_{n-1} P(X_n|X_{n-1}) P(X_{n-1}|X_{n-2}) \dots P(X_1|X_0) > 0 \tag{2.12}$$

where $X_0 = X$ and $X_n = X'$. A sufficient condition for irreducibility is that for every X, X',

$$P(X'|X) > 0.$$

Given these definitions, we now present a theorem without proof for the existence of an equilibrium distribution.

Theorem 2.1. If and only if a Markov chain is ergodic and irreducible, then there exists a *unique* equilibrium distribution $\Pi(X)$. That is, $\Pi(X)$ satisfies the balance equation

$$\Pi(X) = \int dX' P(X|X') \Pi(X'), \tag{2.13}$$

and for any initial probability distribution $P_0(X)$,

$$\lim_{n\to\infty} P_n(X) = \Pi(X). \tag{2.14}$$

We can thus generate states X_n from the equilibrium distribution $\Pi(X)$ by using a Markov chain that satisfies Theorem 2.1. The general procedure consists of three stages:

- *Initialisation*: choose an initial state $X_0 = X$, which implies initial probability distribution $P_0(X) = \delta(X - X_0)$.

- *Thermalisation*: apply enough Markov chain updates N_{therm} such that equilibrium is reached.

- *Measurement*: once thermalised, take the generated states as part of the configuration set. Note that these may be correlated.

Techniques using Markov chains in this way, namely to generate Monte Carlo samples of an integral, are known as Monte Carlo Markov Chain (MCMC) methods.

Theorem 2.1 describes how to create a Markov chain with an equilibrium distribution, and this is relatively easily to do in practice: e.g. $P(X'|X) > 0$ is sufficient. The real trick is engineering a particular pre-defined equilibrium distribution $\Pi(X)$. The guide here is the balance equation (2.5). We usually show the sufficient condition of *detailed balance* to prove that a given distribution is the equilibrium one: this is where the balance equation holds term-wise, i.e.

$$P(X'|X)\Pi(X) = P(X|X')\Pi(X'). \tag{2.15}$$

2.1.1 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm [23, 24], a.k.a. the Metropolis algorithm, is a Markov chain update step that can produce a given equilibrium distribution. The Markov update step is as follows:

1. Given the state $X = X_{n-1}$, choose a candidate state X' through some *a priori* probability $P_0(X'|X)$.

2. Accept the new state $X_n = X'$ with probability

$$P_{\text{acc}}(X'|X) = \min\left(1, \frac{P_0(X|X')P(X')}{P_0(X'|X)P(X)}\right), \tag{2.16}$$

   where $P(X)$ is some probability distribution, otherwise recycle $X_n = X$ as the new state.

Theorem 2.2. The Metropolis-Hastings algorithm has the equilibrium distribution $P(X)$ when $P_0(X'|X) > 0$ and $P(X) > 0$ for all X, X'.

Proof. If $P_0(X'|X) > 0$ and $P(X) > 0$, it follows that our total conditional transition probability $P(X'|X) = P_0(X'|X)P_{\text{acc}}(X'|X) > 0$ for all X, X'. Thus, the Markov chain process is ergodic and irreducible as required for Theorem 2.1, and hence there is an equilibrium distribution.

Finally, we show that the given probability distribution $P(X)$ is the equilibrium distribution by proving the detailed balance equation.

When $P_0(X|X')P(X')/P_0(X'|X)P(X) \leq 1$,

$$\begin{aligned} P(X'|X)P(X) &= P_0(X'|X)P_{\text{acc}}(X'|X)P(X) \\ &= P_0(X'|X)\frac{P_0(X|X')P(X')}{P_0(X'|X)P(X)}P(X) \quad [\text{implied by (2.16)}] \\ &= P_0(X|X')P(X') \\ &= P_0(X|X')P_{\text{acc}}(X|X')P(X') \quad \left[\text{as } \frac{P_0(X'|X)P(X)}{P_0(X|X')P(X')} \geq 1\right] \\ &= P(X|X')P(X'). \end{aligned}$$

When $P_0(X|X')P(X')/P_0(X'|X)P(X) \geq 1$, the argument is as above with $X \leftrightarrow X'$. □

2.2 Pure gauge theory

In order to determine how to use the Metropolis algorithm to generate lattice gauge configurations, we first consider the case of pure gauge theory. The generated configurations are U_n and the desired equilibrium distribution is $P(U) = e^{-S[U]}$ where $S[U] = S_G[U]$. We will almost always use an update step that is symmetric, that is $P_0(U'|U) = P_0(U|U')$, so we can write the Metropolis acceptance probability as

$$P_{\text{acc}}(U'|U) = \min\big(1, \exp[-\Delta S[U]]\big) \tag{2.17}$$

where $\Delta S[U] = S[U'] - S[U]$ is the change in the action. The challenge is finding a Metropolis update step that significantly changes the gauge field for good sampling but also provides good acceptance rates $\langle P_{\text{acc}}\rangle$.

We first consider a *local* update step for pure Wilson gauge theory (1.48). This is where a single link variable $U_\mu(x)$ (μ, x fixed) is updated to $U_\mu(x)'$ per Markov chain step.

The resultant change in the action $\Delta S[U]$ can be determined by examining the field in the neighbourhood of the link variable. With the Wilson gauge action

$$S[U] = \frac{2}{g^2}\sum_{x\in\Lambda}\sum_{\mu>\nu}\text{Re}\,\text{Tr}\big[\mathbb{I} - U_{\mu\nu}(x)\big], \tag{2.18}$$

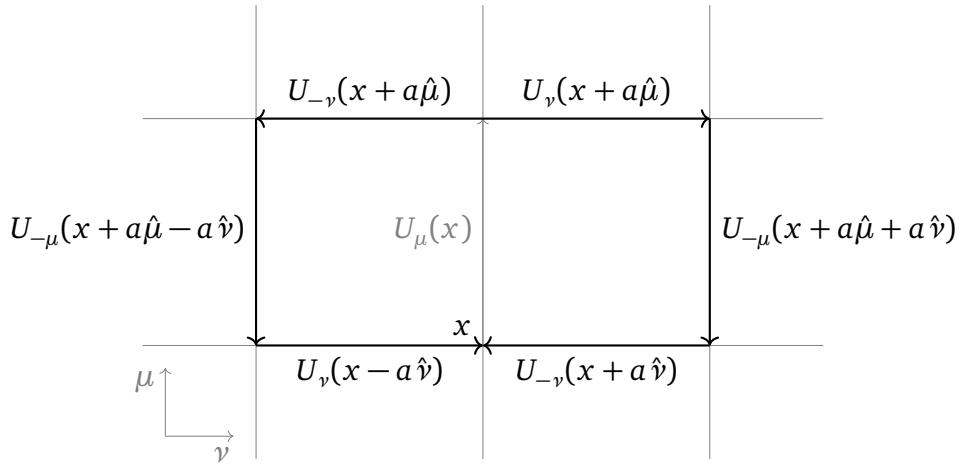

Figure 2.1: Two of the six staples surrounding $U_\mu(x)$, specifically those in the ν and $-\nu$ directions.

the part of the action dependent on the selected link variable is

$$S[U_\mu(x)]_{\text{loc}} = \frac{2}{g^2}\,\text{Re}\,\text{Tr}\big[\mathbb{I} - U_\mu(x)A\big] \tag{2.19}$$

where A is the sum over the so-called *staples* surrounding $U_\mu(x)$:

$$\begin{aligned} A = \sum_{\nu\neq\mu}\big[&U_\nu(x+a\hat{\mu})U_{-\mu}(x+a\hat{\mu}+a\,\hat{\nu})U_{-\nu}(x+a\,\hat{\nu}) \\ &+ U_{-\nu}(x+a\hat{\mu})U_{-\mu}(x+a\hat{\mu}-a\,\hat{\nu})U_\nu(x-a\,\hat{\nu})\big]. \end{aligned} \tag{2.20}$$

These are plaquettes involving $U_\mu(x)$ with $U_\mu(x)$ removed: see Figure 2.1 for a depiction. As the sum over staples does not change when $U_\mu(x)$ changes, it follows that the change in the action when $U_\mu(x) \to U_\mu(x)'$ is given by

$$\Delta S[U_\mu(x)]_{\text{loc}} = -\frac{2}{g^2}\,\text{Re}\,\text{Tr}\big[(U_\mu(x)' - U_\mu(x))A\big]. \tag{2.21}$$

The choice of candidate $U_\mu(x)'$ should be 'close' to $U_\mu(x)$ to ensure a good Metropolis acceptance rate. One way to achieve this is to multiply $U_\mu(x)$ by a random element X of SU(3) near the identity matrix $\mathbb{I}$,

$$U_\mu(x)' = XU_\mu(x), \tag{2.22}$$

chosen such that X and X^{-1} are equally likely and rescaling if necessary to ensure that $U_\mu(x)' \in \text{SU}(3)$. A complete local update step is then as follows

1. Choose a particular gauge link $U_\mu(x)$, and propose a candidate link variable $U_\mu(x)'$ based on e.g. (2.22).
2. Compute the change in the action ΔS via (2.21). Accept the candidate state with probability $\min(1, \exp(-\Delta S))$.

To ensure good sampling of the gauge field distribution, local updates are often repeated across the entire lattice in what are known as *sweeps*.

In many cases however, local update algorithms are not feasible, due to the action containing non-local terms. This is the case for dynamical QCD, where non-local update algorithms, in which large parts of the gauge field are updated at each step, tend to perform substantially better. The most commonly used algorithm in practice is Hybrid Monte Carlo, which will be discussed at length in section 2.4.

2.2.1 A recipe for gauge configuration generation

Now we have the tools to describe a complete gauge configuration generation algorithm. The goal is to generate configurations U_n distributed according to $\exp(-S[U])$. As noted earlier, this process consists of three major stages: *initialisation*, *thermalisation* and *measurement* of the gauge fields.

Initialisation

We can choose any configuration U we like to start with. Two common choices are:

- A *cold start*, where all the gauge links are set to unity $U_\mu(x) = \mathbb{I}$.
- A *hot start*, where all the gauge links are set to random elements of SU(3).

These two choices can be combined to obtain arbitrary initial values of the average gauge field $\langle U_\mu(x) \rangle$.

Thermalisation

Once the gauge fields U are initialised, we then need to thermalise the configuration. This means applying Markov chain updates until we are in equilibrium. The number of updates required is dependent on the lattice size and the gauge coupling g.

One way to determine whether we are in equilibrium is to compare the value of a gluonic quantity between Markov chains with a cold start and with a hot start – once they converge, we are near equilibrium.

Measurement

Supposing the Markov chain has reached equilibrium, we can start to take configurations as part of our ensemble $\{U_i\}$ in order to measure observables via Monte Carlo (1.76),

$$\langle B \rangle \approx \frac{1}{N} \sum_{i=1}^{N} B[U_i].$$

Consecutive configurations in the Markov chain can be highly correlated however, which needs to be taken into account in the statistical analysis of the measured observables. See section B.2 for a discussion. A simple way to deal with such autocorrelations is to only take every N_{corr}-th configuration into our ensemble, ensuring that $N_{\text{corr}} > \tau$ where τ is (an estimate of) the integrated autocorrelation time (B.15).

2.3 Simulating fermions

Now we consider extending Markov Chain Monte Carlo techniques to the full Lattice QCD theory with fermions. Recalling (1.75), the goal is to generate gauge field configurations U_i distributed according to

$$P(U) = \frac{1}{Z} \prod_f \det[D^f[U]] e^{-S_G[U]}. \tag{2.23}$$

For a reasonably sized lattice with Λ lattice sites, evaluating the determinants $\det[D^f]$ directly is prohibitively expensive because D^f is a $12\Lambda \times 12\Lambda$ matrix. In order to evaluate the fermion determinant, we can write the determinant as an integral over complex fields ϕ (Theorem A.1):

$$\det[D^f] \propto \int D\phi D\phi^* \exp\left(-\phi^\dagger (D^f)^{-1} \phi\right). \tag{2.24}$$

This is similar to the Matthews-Salam formula (1.74), but note the matrix inverse. The advantage of this formulation compared to the original integral over fermions fields ψ is that the integral is not Grassmannian, so we can evaluate it by stochastic means. We denote the fields ϕ *pseudofermions*, as they represent fermions but obey bosonic statistics.

With this new form of the determinant (2.24), one can naively write a probability distribution for the gauge fields U_i *and* pseudofermions ϕ_f:

$$P(U, \phi_f) = \frac{1}{Z} \exp\left(-S_G[U] - \sum_f \phi_f^\dagger (D^f[U])^{-1} \phi_f \right). \tag{2.25}$$

Denoting $S_F = \sum_f \phi_f^\dagger (D^f[U])^{-1}\phi$ as the pseudofermion action and $S = S_G + S_F$ as the lattice action, this simplifies to

$$P(U,\phi_f) = \frac{1}{Z} e^{-S[U,\phi_f]}. \tag{2.26}$$

However, for many Dirac matrices D^f (including Wilson), the matrix element $\phi_f^\dagger (D^f[U])^{-1}\phi_f$ is not guaranteed to be non-negative for all ϕ_f, i.e. $D^f[U]$ is not necessarily positive semi-definite. This is problematic, as negative values of S_F cause the Gaussian integral (2.24) to diverge. This causes large instabilities when generating gauge fields. Negative values of S_F also lead to a negative probablity distribution for ϕ_f, which invalidates the use of Monte Carlo importance sampling via (1.76) to evaluate operators. Hence, in order to use (2.26) as a probability distribution in Monte Carlo sampling, we need to modify the pseudofermion action S_F.

The most common way to obtain positive semi-definiteness requires two flavours of fermion with degenerate masses. In practice, these are almost always the up and the down quarks $D^u = D^d = D$, for these have near-degenerate masses in nature. Assuming $\det D \in \mathbb{R}$, we can combine the product of determinants as follows:

$$\det D^u \det D^d = (\det D)^2 = \det D^\dagger \det D = \det(D^\dagger D). \tag{2.27}$$

Then we can apply (2.24) to express this as an integral

$$\det(D^\dagger D) = \int D\phi D\phi^* \exp\left(-\phi^\dagger (D^\dagger D)^{-1}\phi\right), \tag{2.28}$$

which gives rise to the *double-flavour pseudofermion action*

$$S_F = \phi^\dagger (D^\dagger D)^{-1}\phi. \tag{2.29}$$

The matrix $D^\dagger D$ is positive semi-definite by construction, so this fixes the issue. Utilising this pseudofermion action, we can evaluate expectation values by generating configurations (U_i, ϕ_i) distributed according to $\frac{1}{Z}\exp(-S[U,\phi])$ and taking an ensemble average of the observable. The pseudofermion field ϕ is easy to sample as it can be related to a Gaussian distributed $\chi \sim \exp(-\chi^\dagger\chi)$ via $\phi = D^\dagger\chi$, so in practice we do not retain ϕ_i for measurement during the Markov chain. The challenge is in generating appropriate gauge fields U_i.

In the case of the strange quark, there is no obvious same-mass partner, so we cannot use the double-flavour action (2.29). Techniques for simulating singular flavours of fermions are explored in section 2.6.

2.3.1 Using Metropolis-Hastings

Now that we have a valid probability distribution for the gauge fields in the presence of two degenerate fermions (2.29), we can formulate the appropriate Metropolis-Hastings method:

1. Take an initial guess of the gauge field, U_0

2. Sample $\phi_i = \phi$ from the distribution $\frac{1}{Z}\exp(-S_F[U,\phi])$. This is done by generating Gaussian distributed $\chi \sim \exp\left(-\chi^\dagger\chi\right)$ then assigning $\phi = D^\dagger\chi$, for then $\chi^\dagger\chi = \phi^\dagger D^{-1}(D^\dagger)^{-1}\phi = \phi^\dagger(D^\dagger D)^{-1}\phi$.

3. Update the gauge field from U_{i-1} to U' by some process with the properties given in Theorem 2.2.

4. Accept the new gauge field $U_i = U'$ with probability

$$\min(1, \exp(-\Delta S)), \tag{2.30}$$

   otherwise $U_i = U$.

5. Repeat from step 2.

Using a local update algorithm here is nowhere near as effective as for pure gluodynamics. This is because the fermion action S_F involves the inverse of the Dirac matrix D, so a local update causes broad non-local changes ΔS. Compared to an $\mathcal{O}(1)$ calculation in gluodynamics, determining the change in the action here requires a $\mathcal{O}(\Lambda)$ operation. As a local update algorithm needs to be repeated at each lattice site to be effective, the full algorithmic cost is $\mathcal{O}(\Lambda^2)$. This is prohibitively expensive for most lattices of interest. While it is possible to formulate S_F to be local and thus avoid this issue [25], this is not as efficient to simulate as other methods and few optimisations exist.

We therefore use a global update algorithm when simulating full Lattice QCD. If this update is done completely randomly, however, we will still require a $\mathcal{O}(\Lambda^2)$ cost to control the large changes in the action ΔS that result. Instead, we employ global updates which are not completely random, but rather attempt to go in directions where the action S is roughly conserved. Any potential bias in such methods is corrected by the Metropolis acceptance step.

2.4 Hybrid Monte Carlo

2.4.1 Introduction

The *de facto* standard method for generating gauge configurations in Lattice QCD is Hybrid Monte Carlo (HMC) [26].

In HMC, we introduce momentum fields P conjugate to the gauge fields U. As momentum is a real field, we must consider what parts of the SU(3) gauge field the momentum is conjugate to. It is helpful to express U in terms of the SU(3) generators:

$$U_\mu(x) = \exp\left(i\sum_{i=1}^{8}\omega_\mu^{(i)}(x)t^{(i)}\right) = \exp(iQ), \tag{2.31}$$

where Q is an element of the algebra $\mathfrak{su}(3)$, $\omega_\mu^{(i)}(x)$ are real fields, and $t^{(i)}$ are the generators (cf. (1.4)). We thus introduce momentum fields $P_\mu^{(i)}(x)$ conjugate to the real fields $\omega_\mu^{(i)}(x)$. These eight momentum fields can be neatly combined into a single element of $\mathfrak{su}(3)$ via

$$P_\mu(x) = \sum_{i=1}^{8} P_\mu^{(i)}(x)t^{(i)}, \tag{2.32}$$

which is comparable to Q above.

Using the conjugate momentum fields, we construct the Hamiltonian

$$H[P,U] = \sum \mathrm{Tr}\,[P^2] + S[U,\phi\,], \tag{2.33}$$

where we have used the relation

$$\frac{1}{2}\sum_{x\in\Lambda,\mu,i}\left[P_\mu^{(i)}(x)\right]^2 = \sum_{x\in\Lambda,\mu}\mathrm{Tr}[P_\mu(x)^2]. \tag{2.34}$$

The benefit of this construction is that the Hamiltonian is preserved by Hamilton's equations,

$$\frac{\mathrm{d}\omega^{(i)}}{\mathrm{d}t} = \frac{\partial H}{\partial P^{(i)}} = P^{(i)} \quad \text{and} \quad \frac{\mathrm{d}P^{(i)}}{\mathrm{d}t} = -\frac{\partial H}{\partial\,\omega^{(i)}} \tag{2.35}$$

where t is the simulation time through which we imagine the Hamiltonian is evolved. Now, if we write expectation values in the form

$$\begin{aligned}\langle O\rangle &= \frac{\int D[U]\exp(-S[U])O[U]}{\int D[U]\exp(-S[U])}\\ &= \frac{\int D[U]D[P]\exp\big(-\mathrm{Tr}[P^2]-S[U]\big)O[U]}{\int D[U]D[P]\exp\left(-\mathrm{Tr}[P^2]-S[U]\right)},\end{aligned} \tag{2.36}$$

it suggests that configurations (P,U) from the distribution e^{-H} correspond to gauge configurations U distributed according to e^{-S} as desired for expectation values. Hence, if we engineer a Metropolis update step based upon integrating Hamilton's equations, the approximate preservation of H should allow for high acceptance rates. This forms the central idea of Hybrid Monte Carlo.

2.4.2 Deriving the method

In order to put this idea into practice, we must discretise Hamilton's equations (2.35) to find update steps for U and P. The first differential equation can be rewritten as

$$\begin{aligned}\sum_{i=1}^{8}\frac{\mathrm{d}\omega^{(i)}}{\mathrm{d}t}t^{(i)} &= \sum_{i=1}^{8}P^{(i)}t^{(i)},\\ \implies \frac{\mathrm{d}Q}{\mathrm{d}t} &= P.\end{aligned} \tag{2.37}$$

Upon discretisation with step-size ϵ, we get the integration step

$$\begin{aligned}Q(\epsilon) &= Q(0)+\epsilon P(0),\\ \implies -i\ln U(\epsilon) &= -i\ln U(\epsilon)+\epsilon P(0),\\ \implies U(\epsilon) &= e^{i\epsilon P(0)}U(0).\end{aligned} \tag{2.38}$$

This integration step can be used to evolve the Hamiltonian system, and it is often denoted the *time step*.

The second of Hamilton's equations can be written as

$$\frac{\mathrm{d}P}{\mathrm{d}t} = -\sum_{i=1}^{8}\frac{\partial H}{\partial\omega^{(i)}}t^{(i)}. \tag{2.39}$$

Comparing the right-hand side to a vector derivative

$$\frac{\partial f}{\partial\vec{v}} = \sum_{i}\frac{\partial f}{\partial v_i}\hat{v}_i, \tag{2.40}$$

we can write this equation simply as

$$\frac{\mathrm{d}P}{\mathrm{d}t} = -\frac{\partial H}{\partial Q} = -\frac{\mathrm{d}S}{\mathrm{d}Q}. \tag{2.41}$$

Upon discretisation, this leads to the integration step

$$P(\epsilon) = P(0) - \epsilon\left.\frac{\mathrm{d}S}{\mathrm{d}Q}\right|_{t=0}, \tag{2.42}$$

which is denoted the *space step*. The derivative $\frac{\mathrm{d}S}{\mathrm{d}Q} \equiv F$ is called the *force term*.

We combine these two integration steps into what is known as a molecular dynamics *trajectory* to evolve the system from (P,U) to a new candidate state (P',U'). This new state then undergoes a Metropolis acceptance step, which is slightly modified from the usual form to

$$P_{acc} = \min\big[1, \exp\big(H[P,U] - H[P',U']\big)\big]. \tag{2.43}$$

We use this particular acceptance criteria to ensure that detailed balance (2.15) is satisfied by our desired equilibrium distribution $e^{-S[U]}$. This also requires a molecular dynamics trajectory which is area-preserving and reversible, properties which we now define.

Definition 2.6 (Area-preserving)**.** A molecular dynamics process that updates (P,U) to (P',U') is *area-preserving* if the measure $D[P]D[U]$ remains unchanged. To be precise, the Jacobian has determinant 1:

$$\det \frac{\partial(P',U')}{\partial(P,U)} = 1. \tag{2.44}$$

Definition 2.7 (Reversibility)**.** Denote the probability of a molecular dynamics process producing candidate state (P',U') given initial state (P,U) as

$$T_{md}(P',U'|P,U). \tag{2.45}$$

Then a molecular dynamics process $(P,U) \to (P',U')$ is *reversible* iff

$$T_{md}(P',U'|P,U) = T_{md}(-P,U|-P',U'). \tag{2.46}$$

As the processes we consider are deterministic (given P and U), this is equivalent to saying that reversing the momenta in the final state then performing the same molecular dynamics trajectory produces the initial gauge configuration U with reversed momentum $-P$.

Theorem 2.3 (HMC equilibrium)**.** If the molecular dynamics trajectory in HMC is area-preserving and reversible, then the HMC process has equilibrium distribution $e^{-S[U]}$.

Proof. To show this is the case, we prove the detailed balance equation

$$e^{-S[U]}T(U'|U) = e^{-S[U']}T(U|U'), \tag{2.47}$$

where $T(U'|U)$ is the probability of transitioning to state U' given we were in state U. The total transition probability is given by an integral over all possible initial and final momenta states:

$$T(U'|U) = \int D[P]D[P']T_{md}(P',U'|P,U)T_A(P',U'|P,U)e^{-\mathrm{Tr}[P^2]}, \tag{2.48}$$

where T_{md} is the probability of transition during the molecular dynamics trajectory, T_A is the probability of accepting the state, and $e^{-\mathrm{Tr}[P^2]}$ is the probability of selecting P in the first place. Note that this integral does not have a Jacobian factor because the molecular dynamics trajectory is area-preserving.

Substituting (2.48) into the left-hand side of the detailed balance equation (2.47) gives

$$\begin{aligned} e^{-S[U]}T(U'|U) &= \int D[P]D[P']T_{md}(P',U'|P,U)T_A(P',U'|P,U)e^{-S[U]-\mathrm{Tr}[P^2]} \\ &= \int D[P]D[P']T_{md}(P',U'|P,U)\min\left(1,e^{H[U,P]-H[U',P']}\right)e^{-H[U,P]} \\ &= \int D[P]D[P']T_{md}(P',U'|P,U)\min\left(e^{-S[U]-\mathrm{Tr}[P^2]},e^{-S[U']-\mathrm{Tr}[P'^2]}\right), \end{aligned} \tag{*}$$

where the last step used the positivity of $e^{-H[U,P]}$. Now we apply reversibility:

$$\begin{aligned} e^{-S[U]}T(U'|U) &= \int D[P]D[P']T_{md}(-P,U|-P',U')\min\left(e^{-S[U]-\mathrm{Tr}[P^2]},e^{-S[U']-\mathrm{Tr}[P'^2]}\right) \\ &= \int D[P]D[P']T_{md}(P,U|P',U')\min\left(e^{-S[U]-\mathrm{Tr}[P^2]},e^{-S[U']-\mathrm{Tr}[P'^2]}\right) \\ &= e^{-S[U']}T(U|U'), \end{aligned}$$

where we flipped the sign of P and P', then noted the symmetry $(P,U) \leftrightarrow (P',U')$ with expression (*). □

Theorem 2.3 ensures that the Hybrid Monte Carlo algorithm works as intended. In summary, the steps involved to produce gauge field configurations via HMC are as follows (also see section 2.2.1):

1. Choose some initial state $U = U_0$.
2. Generate pseudofermions ϕ from the Gaussian distributed $\chi \sim \exp(-\chi^\dagger\chi)$. In the case of $S_F = \phi^\dagger(M^\dagger M)^{-1}\phi$, we use $\phi = M^\dagger\chi$.

3. Generate conjugate momenta P which are Gaussian distributed $P^{(i)} \sim \exp(-(P^{(i)})^2)$.

4. Evolve the Hamiltonian system using the integration steps (2.38) and (2.42) to construct a reversible, area-preserving trajectory from (P, U_{i-1}) to candidate state (P', U'). See section 2.4.3 for examples.

5. Accept the candidate state U' as U_i with probability

$$P_{acc} = \min\left(1, e^{-\Delta H}\right) \tag{2.49}$$

where $\Delta H = H[P', U'] - H[P, U]$. Otherwise, recycle the previous state, $U_i = U_{i-1}$.

6. If the system is thermalised, save the configuration U_i for measurements.

7. Go to step 2, until the desired number of configurations are generated.

As Hamilton's equations preserve H, the molecular dynamics process with discretised steps approximately preserves H. This means that the average acceptance rate $\langle P_{acc} \rangle$ should be high, supposing that the integration steps are small enough.

2.4.3 Molecular dynamics integrators

Hybrid Monte Carlo requires molecular dynamics integration schemes built from the two steps (2.38) and (2.42) which are area-preserving and reversible. Another typical requirement is that the sum of the integration step sizes ϵ for both kinds of steps are equal, defining a trajectory length τ, such that we sufficiently change P and U. Here, we prove some theorems that are helpful for constructing suitable schemes, then present a couple of examples.

To begin with, we need a concrete definition of an integration scheme:

Definition 2.8 (Integration scheme)**.** An *integration scheme* $\hat{M}$ (a.k.a. integrator) is a deterministic process which modifies our initial state (P, U) to some final state (P', U'):

$$(P', U') = \hat{M}(P, U). \tag{2.50}$$

Often, an integration scheme can be parametrized by a step-size ϵ, which we will denote $\hat{M}[\epsilon]$.

Using this notation, we denote the two atomic HMC integration steps

$$\hat{S}[\epsilon]: \quad (P, U) \to (P - \epsilon F[U], U) \tag{2.51a}$$

$$\text{and} \quad \hat{T}[\epsilon]: \quad (P, U) \to (P, e^{i\epsilon P} U). \tag{2.51b}$$

We call an integration scheme made up purely of these two steps *symplectic*, because these steps are tangential to planes in phase space where the Hamiltonian is constant.

First, we consider theorems relating to area-preservation:

Theorem 2.4. Let $\hat{A}$ and $\hat{B}$ be area-preserving integration schemes. Then so is $\hat{A}\hat{B}$.

Proof. Let's write the action of the combined scheme on an initial state as

$$\hat{A}\hat{B}(P,U) = \hat{A}(P',U') = (P'',U'').$$

Then the Jacobian for the full scheme is given by

$$\begin{aligned} J_{AB} &= \frac{\partial(P'',U'')}{\partial(P,U)} \\ &= \frac{\partial(P'',U'')}{\partial(P',U')}\frac{\partial(P',U')}{\partial(P,U)} \\ &= J_A J_B. \end{aligned}$$

The determinant of this Jacobian is then $\det J = \det J_A J_B = \det J_A \det J_B = 1$ by assumption. □

Theorem 2.5. The two atomic HMC integration steps are area-preserving, and so, by Theorem 2.4, all symplectic integrators are area-preserving.

Proof. Consider the Jacobian for the space step $\hat{S}[\epsilon]$:

$$J_S = \begin{pmatrix} \frac{\partial P'}{\partial P} & \frac{\partial U'}{\partial P} \\ \frac{\partial P'}{\partial U} & \frac{\partial U'}{\partial U} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \epsilon\frac{\partial F}{\partial U} & 1 \end{pmatrix}$$

Hence, $\det J_S = 1$, and the space step is area-preserving.

Consider the Jacobian for the time step $\hat{T}[\epsilon]$:

$$J_T = \begin{pmatrix} \frac{\partial P'}{\partial P} & \frac{\partial U'}{\partial P} \\ \frac{\partial P'}{\partial U} & \frac{\partial U'}{\partial U} \end{pmatrix} = \begin{pmatrix} 1 & i\epsilon e^{i\epsilon P}U \\ 0 & 1 \end{pmatrix}$$

Hence, $\det J_T = 1$, and the time step is area-preserving. □

In order to determine whether a scheme is reversible, we first write the reversibility condition in our notation. An integration scheme $\hat{M}$ is reversible iff for every (P,U)

$$\hat{M}(-P',U') = (-P,U), \tag{2.52}$$

where $(P',U') = \hat{M}(P,U)$.

Theorem 2.6. Our atomic time and space integration steps are reversible.

Proof. First, let's consider $(P', U') = \hat{S}[\epsilon](P, U) = (P - \epsilon F[U], U)$. Then:

$$\hat{S}[\epsilon](-P', U') = (-P' - \epsilon F[U'], U') = (-(P - \epsilon F[U]) - \epsilon F[U], U) = (-P, U).$$

Hence, the space step is reversible. Next, let's consider $(P', U') = \hat{T}[\epsilon](P, U) = (P, e^{i\epsilon P}U)$. Then:

$$\hat{T}[\epsilon](-P', U') = (-P', e^{-i\epsilon P'}U') = (-P, e^{-i\epsilon P}e^{i\epsilon P}U) = (-P, U).$$

Hence, the time step is reversible. □

We can construct larger integrators from these atomic steps by using the next two theorems.

Theorem 2.7 (Integrator powers)**.** Let $\hat{M}$ be a reversible integration scheme. Then for any $n \in \mathbb{N}$, $\hat{M}^n$ is also reversible.

Proof. This is easily proved by induction. The $n = 1$ case is trivial.

Next, suppose $\hat{M}^k$ is reversible. Then if we write

$$\hat{M}\hat{M}^k(P, U) = \hat{M}(P', U') = (P'', U''),$$

then the reversed integration is

$$\begin{aligned}\hat{M}\hat{M}^k(-P'', U'') &= \hat{M}^k\hat{M}(-P'', U'') \\ &= \hat{M}^k(-P', U') \\ &= (-P, U),\end{aligned}$$

using the reversibility of $\hat{M}$ and $\hat{M}^k$. So $\hat{M}^{k+1}$ is reversible. □

Theorem 2.8 (Symmetric integrators)**.** Let $\hat{M}_i$, $i = 1, \ldots, n$ be reversible integration schemes. Then the product

$$\hat{M} = \prod_{i=1}^{n} \hat{M}_i \tag{2.53}$$

is reversible if $\hat{M}_i = \hat{M}_{n-i+1} \forall i$. We denote such schemes *symmetric*.

Proof. This can be proved by induction with two base cases, $n = 1, 2$, to account for an odd/even number of sub-schemes.

The $n = 1$ case is trivial. For $n = 2$, we have $\hat{M}_2\hat{M}_1 = \hat{M}_1^2$, which is reversible by Theorem 2.7.

Next, suppose that $\prod_{i=1}^{k}\hat{M}_i = \hat{K}$ is reversible for some integer k and reversible $\hat{M}_i = \hat{M}_{k-i+1}$. Then we can write the $k+2$ case as

$$\prod_{i=0}^{k+1}\hat{M}_i = \hat{L}\hat{K}\hat{L}$$

for some reversible $\hat{L} = \hat{M}_0 = \hat{M}_{k+1}$. Let's write the action of the integrator on an initial state as

$$\begin{aligned}\hat{L}\hat{K}\hat{L}(P,U) &= \hat{L}\hat{K}(P',U')\\ &= \hat{L}(P'',U'')\\ &= (P''',U''').\end{aligned}$$

Then the reversed integration is

$$\begin{aligned}\hat{L}\hat{K}\hat{L}(-P''',U''') &= \hat{L}\hat{K}(-P'',U'')\\ &= \hat{L}(-P',U')\\ &= (P,U),\end{aligned}$$

using the reversibility of $\hat{L}$ and $\hat{K}$. Hence, the $n = k+2$ case is true.

Therefore, the $n \in \mathbb{N}$ case is true. □

Combining Theorems 2.5 and 2.8, we therefore find that symmetric symplectic integrators are area-preserving and reversible as required for HMC. Suitable schemes can also be repeated as many times as desired while preserving these properties via Theorem 2.7.

The most basic example of a symmetric symplectic integrator is the *leap-frog integrator*, which has a space-time-space version

$$\hat{S}_{LFSTS}[h] = \hat{S}[h/2]\hat{T}[h]\hat{S}[h/2] \tag{2.54}$$

and a time-space-time version

$$\hat{S}_{LFTST}[h] = \hat{T}[h/2]\hat{S}[h]\hat{T}[h/2]. \tag{2.55}$$

The combined step-size h here is in 'simulation' time, and we can repeat these integration steps n times to form a trajectory of length $\tau = nh$:

$$\hat{M}[\tau] = (\hat{S}_{LPF}[h])^n. \tag{2.56}$$

Leap-frog integration schemes with $n = \tau/h$ steps have discretisation error $\mathcal{O}(h^2)$, and hence they are second order. The structure of (2.56) allows one to easily change the accuracy of the scheme by changing the step count n, which is especially useful for tuning the acceptance rate $\langle P_{\text{acc}} \rangle$.

A class of symmetric symplectic integrators with better integration errors are the *minimal norm integrators*, discovered by Omelyan, Mryglod, and Folk [27]. The simplest example is the second-order minimal norm integrator, which again has both a space-time-space version

$$\hat{S}_{2MNSTS}[h] = \hat{S}[\lambda h]\hat{T}[h/2]\hat{S}[(1-2\lambda)h]\hat{T}[h/2]\hat{S}[\lambda h] \tag{2.57}$$

and a time-space-time version

$$\hat{S}_{2MNTST}[h] = \hat{T}[\lambda h]\hat{S}[h/2]\hat{T}[(1-2\lambda)h]\hat{S}[h/2]\hat{T}[\lambda h]. \tag{2.58}$$

Here $0 \leq \lambda \leq 1$ is a parameter that can be tuned to optimise the integrator error. The second-order minimal norm schemes with $n = \tau/h$ steps have discretisation error $\mathcal{O}(h^2)$, the same order as leap-frog. However, compared to a leap-frog scheme with the same cost (i.e. same number of space steps $\hat{S}$), these schemes have about three times smaller errors.

The next higher order scheme is a fourth-order minimal norm scheme. This can take several forms, but two of note are the 'velocity' version with 5 force evaluations

$$\begin{aligned}\hat{S}_{4MN5FV}[h] = \hat{S}[\theta h]\hat{T}[\rho h]\hat{S}[\lambda h]\hat{T}[\mu h]\hat{S}[(1-2(\lambda+\theta))/2h] \\ \times \hat{T}[(1-2(\mu+\rho))/2h]\hat{S}[(1-2(\lambda+\theta))/2h] \\ \times \hat{T}[\mu h]\hat{S}[\lambda h]\hat{T}[\rho h]\hat{S}[\theta h]\end{aligned} \tag{2.59}$$

and the 'position' version with 4 force evaluations

$$\begin{aligned}\hat{S}_{4MN4FP}[h] = \hat{T}[\rho h]\hat{S}[\lambda h]\hat{T}[\theta h]\hat{S}[(1-2(\lambda+\theta))/2h]\hat{T}[(1-2(\theta+\rho))/2h] \\ \times \hat{S}[(1-2\lambda))/2h]\hat{T}[\theta h]\hat{S}[\lambda h]\hat{T}[\rho h].\end{aligned} \tag{2.60}$$

The four parameters $\theta, \rho, \lambda, \mu$ are set differently in each case to minimise the error terms. Both of these schemes have similar $\mathcal{O}(h^4)$ discretisation errors over n steps. For this reason, it would seem that the position version is superior as there are less force evaluations. Note however that when taking powers of such schemes, any adjacent $\hat{T}$ or $\hat{S}$ steps can be merged into a single step, such that the 4MN5FV scheme only has one more force evaluation than 4MN5FP for any number of steps.

Tuning the acceptance rate

Having discussed the molecular dynamics integrators used to produce candidate states for the Metropolis-Hastings acceptance step, we are now in a position to discuss the acceptance rate $\langle P_{acc} \rangle$.

In HMC simulations, there is a trade-off between good acceptance rates $\langle P_{acc} \rangle$ to minimise autocorrelations and computational cost. The ideal $\langle P_{acc} \rangle$ minimises the cost to generated statistically independent configurations. As the computation is dominated by matrix multiplies by the Dirac matrix D, a suitable cost function for HMC is

$$C = \frac{N_{mat}\tau_a}{\langle P_{\text{acc}} \rangle} \tag{2.61}$$

where N_{mat} is the number of matrix multiplies required per trajectory and τ_a is the integrated autocorrelation time defined in Appendix B. The inclusion of $\langle P_{\text{acc}} \rangle$ accounts for the cost of configurations rejected by the Metropolis acceptance step, which are excluded from the definition of τ_a.

This cost function is not possible to minimise analytically, for we can only determine N_{mat} and τ_a by actually performing HMC. However, with a few assumptions, we can write a cost function in terms of analytical parameters. First, assume that our integrator is of the form $[\hat{I}(h)]^n$ for some fixed integration step $\hat{I}(h)$, e.g. leap-frog or minimal norm, with trajectory length $\tau = nh = 1$. Then N_{mat} is proportional to the number of integration steps n and hence inversely proportional to the step-size h. Second, we assume that on the same lattice, the integrated autocorrelation time does not depend on the particular method used for molecular dynamics integration. Therefore, we can use the cost function

$$C = \frac{1}{\langle P_{acc} \rangle h}. \tag{2.62}$$

It can be shown [28] that for this measure of performance and assuming small h, the optimal acceptance rate only depends on the order m of the integrator used:

$$\langle P_{acc} \rangle_{opt} = \exp\left(-\frac{1}{m}\right). \tag{2.63}$$

For example, the theoretical optimal $\langle P_{acc} \rangle$ for a second-order integrator is 61%. This provides a target acceptance rate when tuning the step-size h in a HMC simulation.

However, problems can arise if the step-size h is set too large. This is because Dirac matrix inversions can be unstable, which occurs more frequently at low quark masses m_q. Instabilities can lead to unexpectedly large forces which cause

'spikes' in the Hamiltonian delta ΔH distribution. This affects the ergodicity of the HMC algorithm, which can change the equilibrium of the Markov chain; thus, one needs to reduce the step-size h to keep this in check. However, an unstable Dirac matrix inversion also requires more iterations to converge, so the computational cost is compounded. Applying improvement techniques, to be discussed in section 2.5, helps to alleviate these issues. Furthermore, in practice, the acceptance rate is often tuned to be higher than that predicted by (2.63) to avoid potential instabilities.

2.4.4 The force term

The most important part of a molecular dynamics trajectory is the evaluation of the force term

$$F = \frac{\mathrm{d}S}{\mathrm{d}Q} = \sum_i \frac{\mathrm{d}S}{\mathrm{d}\omega^{(i)}} t^{(i)} \tag{2.64}$$

at each space step (2.42). This is because, as will be demonstrated, this update is the most computationally intensive part of a HMC simulation for typical lattices.

As the action is written in terms of U, we require a way to evaluate the derivative $\mathrm{d}f/\mathrm{d}\omega^{(i)}$ of some arbitrary scalar function $f(U)$. As U is a member of the SU(3) Lie group, this is a directional derivative in the same direction as $\omega^{(i)}$, namely

$$\frac{\mathrm{d}f(U)}{\mathrm{d}\omega^{(i)}} = \frac{\mathrm{d}}{\mathrm{d}\omega} f(\exp[i \sum_j \omega^{(j)} t^{(j)} + i\omega t^{(i)}]) \Big|_{\omega=0}. \tag{2.65}$$

This can be parametrised as

$$\frac{\mathrm{d}f(U)}{\mathrm{d}\omega^{(i)}} = \frac{\mathrm{d}f(\gamma(\omega))}{\mathrm{d}\omega} \Big|_{\omega=0}, \tag{2.66}$$

where $\gamma(\omega)$ is a path on the SU(3) manifold defined by

$$\gamma(\omega) = \exp[i \sum_j \omega^{(j)} t^{(j)} + i\omega t^{(i)}]. \tag{2.67}$$

This expression (2.65) for the directional derivative is hard to use in practice, as the gauge field U has to be augmented directly. However, it can be shown (see e.g. [29, definition 3.33]) that the directional derivative is the same for any path $\tilde{\gamma}(\omega)$ on the SU(3) manifold with $\tilde{\gamma}(0) = \gamma(0)$ and $\tilde{\gamma}'(0) = \gamma'(0)$. In particular, we can use

$$\gamma(\omega) = \exp[i\omega t^{(i)}]U, \tag{2.68}$$

and write the directional derivative as

$$\frac{\mathrm{d}f(U)}{\mathrm{d}\omega^{(i)}} = \frac{\mathrm{d}}{\mathrm{d}\omega} f(e^{i\omega t^{(i)}} U)\bigg|_{\omega=0}. \tag{2.69}$$

Using this formula, we can now consider the force terms for the gauge and pseudofermion actions.

The Wilson gauge action

Recall the Wilson gauge action (1.48):

$$\begin{aligned} S_G[U] &= \frac{2}{g^2} \sum_{x\in\Lambda} \sum_{\mu>\nu} \mathrm{Re}\,\mathrm{Tr}\Big[\mathbb{I} - U_\mu(x) U_\nu(x+a\hat{\mu}) U_\mu^\dagger(x+a\hat{\nu}) U_\nu^\dagger(x)\Big] \\ &= \frac{1}{g^2} \sum_{x\in\Lambda} \sum_{\mu>\nu} \mathrm{Tr}\Big[2\mathbb{I} - U_\mu(x) U_\nu(x+a\hat{\mu}) U_\mu^\dagger(x+a\hat{\nu}) U_\nu^\dagger(x) \\ &\qquad\qquad - U_\nu(x) U_\mu(x+a\hat{\nu}) U_\nu^\dagger(x+a\hat{\mu}) U_\mu^\dagger(x)\Big] \\ &= \frac{1}{g^2} \sum_{x\in\Lambda} \sum_{\mu,\,\nu\neq\mu} \mathrm{Tr}\Big[\mathbb{I} - U_\mu(x) U_\nu(x+a\hat{\mu}) U_\mu^\dagger(x+a\hat{\nu}) U_\nu^\dagger(x)\Big]. \end{aligned}$$

This involves a sum over all plaquettes in both directions. Upon taking the derivative with respect to $\omega_\mu^{(i)}(x)$, we only act on the terms $U_\mu(x)$ and $U_\mu^\dagger(x)$. Thus, only 6 different plaquettes are involved. Recalling the sum over staples (2.20),

$$A = \sum_{\nu\neq\mu} U_\nu(x+a\hat{\mu}) U_\mu^\dagger(x+a\hat{\nu}) U_\nu^\dagger(x) + \sum_{\nu\neq\mu} U_{-\nu}(x+a\hat{\mu}) U_\mu^\dagger(x-a\hat{\nu}) U_{-\nu}^\dagger(x),$$

we can utilise the cyclic properties of the trace to write the derivative as

$$\begin{aligned} \frac{\partial S_G[U]}{\partial\, \omega_\mu^{(i)}(x)} &= \frac{1}{g^2} \frac{\partial}{\partial\, \omega} \mathrm{Tr}\Big[-e^{i\omega t^{(i)}} U_\mu(x) A - A^\dagger U_\mu^\dagger(x) e^{-i\omega t^{(i)}}\Big] \\ &= -\frac{1}{g^2} \mathrm{Tr}\Big[i t^{(i)} (U_\mu(x) A - A^\dagger U_\mu^\dagger(x))\Big]. \end{aligned} \tag{2.70}$$

The force term for the Wilson gauge action is thus

$$F_G = \frac{\mathrm{d}S_G}{\mathrm{d}Q} = -\frac{1}{g^2} \sum_i t^{(i)}\, \mathrm{Tr}\Big[i t^{(i)} (UA - A^\dagger U^\dagger)\Big]. \tag{2.71}$$

Now, $i(UA - A^\dagger U^\dagger)$ is traceless and Hermitian, and hence is an element of the $\mathfrak{su}(3)$ algebra. Writing an element of $\mathfrak{su}(3)$ in terms of generators and using the identity $\mathrm{Tr}[t^{(i)}t^{(j)}] = \frac{1}{2}\delta_{ij}$, we have

$$\begin{aligned}\sum_i t^{(i)} \mathrm{Tr}\left[t^{(i)} \sum_j c_j t^{(j)} \right] &= \sum_i \sum_j c_j t^{(i)} \mathrm{Tr}[t^{(i)} t^{(j)}] \\ &= \frac{1}{2} \sum_i \sum_j c_j t^{(i)} \delta_{ij} \\ &= \frac{1}{2} \sum_i c_i t^{(i)}. \end{aligned} \tag{2.72}$$

Therefore, the force term for the Wilson gauge action can be simplified to

$$F_G = \frac{\mathrm{d}S_G}{\mathrm{d}Q} = -\frac{1}{2g^2} i(UA - A^\dagger U^\dagger). \tag{2.73}$$

The fermion action

In the case of a double-flavour pseudofermion action (2.29)

$$S_F = \phi^\dagger (D^\dagger D)^{-1} \phi,$$

the corresponding force term is given by

$$\begin{aligned} F = \frac{\mathrm{d}S_F}{\mathrm{d}Q} &= \phi^\dagger \frac{\mathrm{d}}{\mathrm{d}Q} \left[(D^\dagger D)^{-1} \right] \phi \\ &= -\phi^\dagger (D^\dagger D)^{-1} \frac{\mathrm{d}}{\mathrm{d}Q} \left[(D^\dagger D) \right] (D^\dagger D)^{-1} \phi \\ &= -\phi^\dagger (D^\dagger D)^{-1} \left[\frac{\mathrm{d}D^\dagger}{\mathrm{d}Q} D + D^\dagger \frac{\mathrm{d}D}{\mathrm{d}Q} \right] (D^\dagger D)^{-1} \phi. \end{aligned} \tag{2.74}$$

Note that the fermion force term involves inverses of $D^\dagger D$. Due to the size of the matrices involved on a typical lattice, it is impractical to directly calculate the full matrix inverse. Instead, we solve $D^\dagger D \chi = \phi$ for χ with an iterative solver such as conjugate gradient. For a typical lattice, this operation is computationally expensive, which is compounded by the fact the forces have to be calculated at each space step. For these reasons, calculating the fermion force terms is the leading cost of HMC simulations.

In the case of a Wilson fermion (1.63), the derivative of the Dirac operator is fairly simple to write down due to linearity in U:

$$\frac{\mathrm{d}D(x|y)}{\mathrm{d}Q(z)} = -i\kappa \sum_i t^{(i)} \Big[(1-\gamma_\mu) t^{(i)} U_\mu(z) \delta_{x+a\hat{\mu},y} \delta_{x,z} - (1+\gamma_\mu) U_\mu^\dagger(z) t^{(i)} \delta_{x-a\hat{\mu},y} \delta_{y,z} \Big]. \tag{2.75}$$

2.5 HMC improvements

The basic Hybrid Monte Carlo algorithm (page 38) is fully functional for producing gauge field configurations U with fermions. However, when working with large lattices at small quark masses, the calculations required take significantly large amounts of compute power. This has led to the development of several different improvements to the base HMC algorithm, which is the main subject of investigation in this thesis as discussed in chapters 3, 4 and 5.

2.5.1 Even-odd preconditioning

One of the earlier ideas was *even-odd preconditioning* [30]. It is based on the fact that the Wilson Dirac operator D, and many others like it, only couple nearest-neighbour sites. A convenient decomposition of such operators is

$$D = \begin{pmatrix} D_{ee} & D_{eo} \\ D_{oe} & D_{oo} \end{pmatrix}; \tag{2.76}$$

this is a matrix that acts on a fermion vector $\psi = \begin{pmatrix} \psi_e & \psi_o \end{pmatrix}^T$ decomposed into 'even' and 'odd' lattice sites, where D_{ee} is the part of D that connects even sites to even sites, D_{eo} odd to even, et cetera. The 'even' and 'odd' sites are given by a chequerboard decomposition of the lattice Λ, depicted in Figure 2.2.

A Dirac matrix in even-odd form (2.76) can undergo a (block) LDU factorisation

$$\begin{pmatrix} D_{ee} & D_{eo} \\ D_{oe} & D_{oo} \end{pmatrix} = \begin{pmatrix} 1_{ee} & 0 \\ D_{oe} & 1_{oo} \end{pmatrix} \begin{pmatrix} D_{ee} & 0 \\ 0 & D_{oo} - D_{oe} D_{ee}^{-1} D_{eo} \end{pmatrix} \begin{pmatrix} 1_{ee} & D_{eo} \\ 0 & 1_{oo} \end{pmatrix} = L D^* U. \tag{2.77}$$

Now recall that when generating gauge configurations, the fermion contribution comes in the form of the determinant $\det D$ (1.75). Using this LDU factorisation, we find that

$$\det D \propto \det D_{ee} \det \tilde{D} \tag{2.78}$$

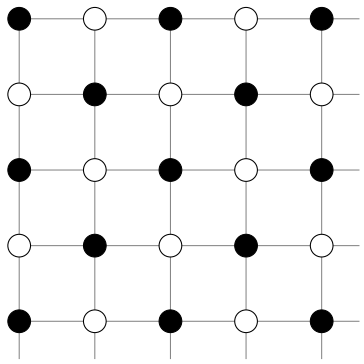

Figure 2.2: Even-odd decomposition of a 2D lattice. Lattice sites with black circles are even, white circles are odd.

where

$$\tilde{D} = D_{oo} - D_{oe} D_{ee}^{-1} D_{eo}. \tag{2.79}$$

This works because L and U are triangular, and D^* is block diagonal. We can therefore replace $\det D$ with $\det D_{ee} \det \tilde{D}$ in the expectation value, as any constant factor disappears in the ratio of path integrals. In the case of an operator that only couples nearest-neighbour sites, D_{ee} and D_{oo} are diagonal with respect to lattice sites. Thus, we can easily evaluate the first determinant via the identity

$$\det D_{ee} = \exp\left(\mathrm{Tr}(\ln D_{ee})\right) \tag{2.80}$$

because the logarithm acts element-wise on a diagonal matrix. The product of determinants in the double-flavour case (2.29) can then be evaluated via

$$(\det D_{ee} \det \tilde{D})^2 = \int d\phi\, d\phi^* \exp\left[2\,\mathrm{Tr}(\ln D_{ee}) - \phi^\dagger (\tilde{D}^\dagger \tilde{D})^{-1} \phi\right] \tag{2.81}$$

and hence gives a fermion action with two terms,

$$S_{\text{even-odd}} = -2\,\mathrm{Tr}(\ln D_{ee}) + \phi^\dagger (\tilde{D}^\dagger \tilde{D})^{-1} \phi. \tag{2.82}$$

The first term is called the *determinant term*, $S_{\text{det}}[U]$, and is relatively cheap to calculate due to $\mathcal{O}(\Lambda)$ scaling. The second term can be treated just like the double-flavour pseudofermion term (2.29) for HMC, but with the advantage that $\tilde{D}$ is a quarter of the size of D.

In the case of a Wilson Dirac operator (1.63), the determinant term can be neglected because $D_{ee} = 1_{ee}$ and hence produces a constant factor in the partition function. The resultant even-odd preconditioned Dirac operator is

$$\tilde{D}_{\text{Wilson}} = 1 - \kappa^2 H_{oe} H_{eo}. \tag{2.83}$$

For the clover improved action, if we denote the clover term (1.65) as T, we get the clover determinant action term

$$S_{\text{det}} = -2\,\text{Tr}\,[\ln(1+T_{ee})] \tag{2.84}$$

and modified Dirac operator

$$\tilde{D}_{\text{Clover}} = (1+T_{oo}) - \kappa^2 H_{oe}(1+T_{ee})^{-1}H_{eo}. \tag{2.85}$$

2.5.2 Multiple time-scales

When using Hybrid Monte Carlo, we often have multiple action terms that have differently behaved forces. The most basic case is a gluon action and a fermion action. The gluon action S_G has a cheap to evaluate force term F_G, so it would be advantageous to use an integration scheme with a fine step-size h for high accuracy. On the other hand, S_F often has an expensive force term F_F, so we would like to use a coarse step-size to reduce the expense. To satisfy these two competing desires, we can use integrations schemes that have multiple integration step-sizes built in. Each level of integration is known as a time-scale.

Nested integrators

One way to construct multiple time-scales is by nesting integration steps within other integrators [31].

Suppose we have the action

$$S = S_G + S_F. \tag{2.86}$$

These can be integrated via the HMC integration steps (2.51a), (2.51b). However, we can separate the space step into updates for each of the action terms:

$$\hat{S}_G[h]: \quad (P,U) \to (P - hF_G, U), \tag{2.87a}$$

$$\text{and} \quad \hat{S}_F[h]: \quad (P,U) \to (P - hF_F, U). \tag{2.87b}$$

These update steps are used in the construction of a multi-scale scheme. Now consider (without loss of generality) the STS leap-frog integrator (2.54) acting just on S_F,

$$\hat{S}_1[h] = \hat{S}_F[h/2]\hat{T}[h]\hat{S}_F[h/2]. \tag{2.88}$$

We can nest another integration scale for S_G within this scheme by replacing the time steps $\hat{T}$ with a reversible and area-preserving integration step for S_G, as this

will always produce a symmetric scheme. The replacement scheme is chosen to have the same step-size, such that the trajectory length τ is consistent between each action term step and the time step. For example, if we write

$$\hat{S}_2[h] = \hat{S}_G[h/2]\hat{T}[h]\hat{S}_G[h/2], \tag{2.89}$$

then we can construct a nested integration step

$$\hat{S}_{\text{nested}}[h] = \hat{S}_F[h/2](\hat{S}_2[h/m])^m\hat{S}_F[h/2], \tag{2.90}$$

where m is a positive integer. This places the gauge action on a fine scale with step-size $h_G = h/m$, and the fermion action on a coarse scale with step-size $h_F = h$.

Such nesting can be applied to all symplectic integrators. The nesting can also be repeated to give a different step-size to each of several action terms, with each finer step-size being a divisor of the previous.

Overlaid integrators

A more flexible way to construct integration schemes with multiple time-scales is to overlay complete schemes for each action term on top of one another. We call these *overlaid integrators* [32].

The basic idea is to first consider the time-steps advancing a time parameter t from 0 to τ. If we construct integration schemes for each action term that advance the partial Hamiltonian $H_i = T + S_i$ where $T = \text{Tr}[P^2]$, then we can superimpose the schemes onto the t axis to produce a single integration scheme that is symmetric and hence suitable for HMC: see Theorem A.3 for the proof. Furthermore, the error in the overlaid scheme has the same order as the constituent integrators.

As an example, let's have an action $S = S_1 + S_2$ with S_1 being integrated with a two-step STS leap-frog scheme (2.54)

$$\hat{V}_1[h] = \hat{S}_1[h/4]\hat{T}[h/2]\hat{S}_1[h/2]\hat{T}[h/2]\hat{S}_1[h/4] \tag{2.91}$$

and S_2 being integrated with a one-step TST second order minimal norm scheme (2.58)

$$\hat{V}_2[h] = \hat{T}[\lambda h]\hat{S}_2[h/2]\hat{T}[(1-2\lambda)h]\hat{S}_2[h/2]\hat{T}[h/2]. \tag{2.92}$$

If we overlay these schemes along the time-step axis t, as shown in Figure 2.3, we end up with the two-scale scheme

$$\begin{aligned}\hat{V}[h] =& \hat{S}_1[h/4]\hat{T}[\lambda h]\hat{S}_2[h/2]\hat{T}[(1/2-\lambda)h]\hat{S}_1[h/2] \\ &\times \hat{T}[(1/2-\lambda)h]\hat{S}_2[h/2]\hat{T}[\lambda h]\hat{S}_1[h/4].\end{aligned} \tag{2.93}$$

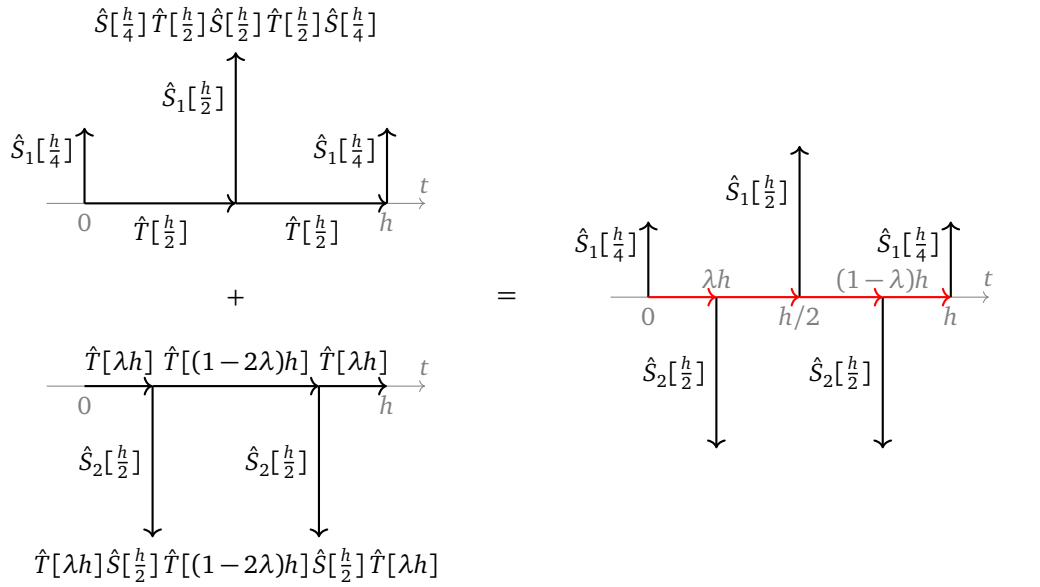

Figure 2.3: Demonstration of overlaid integrators. Overlaying the two-step STS leap-frog scheme (top left) with the one-step TST second order minimal norm scheme (bottom left) produces a two-scale scheme (right).

The advantage of overlaid multiple time-scale integrators is that any number of steps can be used for the constituent schemes, rather than being restricted to multiples as in the nested case. This turns out to be useful when tuning the step-sizes to match the force term distribution, as will be demonstrated when discussing benchmarking results in chapter 3.

2.5.3 Filtering methods

While splitting the gauge action and fermion action onto different integration time scales (section 2.5.2) improves the computational cost, the fermion action in HMC is still expensive to simulate. It is even more expensive when dealing with larger lattices and lighter quark masses, both of which are desirable for reliable physical point extrapolations. To mitigate this, we can split up the standard HMC pseudofermion action into different parts.

Suppose we have a fermion action with matrix kernel L,

$$S_F = \phi^\dagger L \phi. \tag{2.94}$$

We can use the identity

$$\det L^{-1} = \det F^{-1} \det F L^{-1} \tag{2.95}$$

to split up the fermion determinant evaluation into two pseudofermion integrals

$$\det F^{-1} \det F L^{-1} = \int d\phi_1 d\phi_1^* d\phi_2 d\phi_2^* \exp\left[-\phi_1^\dagger F \phi_1 - \phi_2^\dagger F^{-1} L \phi_2\right], \quad (2.96)$$

which gives a two term fermion action

$$S_F = \phi_1^\dagger F \phi_1 + \phi_2^\dagger F^{-1} L \phi_2. \quad (2.97)$$

As the matrix F filters out a portion of the fermion matrix kernel L, we denote such methods *filtering methods*.

The true benefit of filtering is realised when one puts the filter term S_1 and the correction term S_2 on separate time-scales as in section 2.5.2. If we choose a filter F such that [33]

- The filter term S_1 has a cheap to evaluate force term F_1 compared to F_2, and
- The filter term S_1 captures (i.e. encapsulates) the high energy/frequency modes of the system, roughly corresponding to a larger force F_1,

then we can place S_1 on a fine scale h_1 and S_2 on a coarse scale $h_2 > h_1$ while maintaining a good acceptance rate. This works because S_2 only contains lower frequency modes which can be integrated with a coarser step-size. As there are fewer evaluations of the expensive correction term, this leads to a large reduction in computational cost.

Investigating the performance of filtering methods is the main focus of this thesis. This starts with chapter 3, where we compare the performance of the following two filtering methods on a small lattice.

Hasenbusch

One of the most common filtering methods is *Hasenbusch filtering*, a.k.a. *mass preconditioning* [34].

In this method, the filtering action term is similar to the original term except with a heavier quark mass. If we consider the double-flavour pseudofermion action

$$S_F = \phi^\dagger (D^\dagger D)^{-1} \phi, \quad (2.98)$$

then the Hasenbusch filtered action is

$$S_{\text{Hasen}} = \phi_1^\dagger (W^\dagger W)^{-1} \phi_1 + \phi_2^\dagger W (D^\dagger D)^{-1} W^\dagger \phi_2 \quad (2.99)$$

where $W = D[\kappa']$ has a heavier mass $\kappa' < \kappa$. Note that the second term has been rearranged using the cyclic properties of determinants, making its force term a bit easier to evaluate. The filter term S_1 is cheap to evaluate due to the heavier mass, and captures the high energy modes for the same reason. Hence, this filtering method is suitable for speedup via multiple time-scales. In fact, it was such a combination that broke through the 'Berlin Wall' of computational cost [35, 36] and makes physical point calculations feasible today.

The force term for the Hasenbusch filter S_1 is near-identical to that of the pure HMC action,

$$\begin{aligned} F_1 &= -\phi_1^\dagger (W^\dagger W)^{-1} \left[\frac{\mathrm{d}W^\dagger}{\mathrm{d}Q} W + W^\dagger \frac{\mathrm{d}W}{\mathrm{d}Q} \right] (W^\dagger W)^{-1} \phi_1 \\ &= -\phi_1^\dagger (W^\dagger W)^{-1} W^\dagger \frac{\mathrm{d}W}{\mathrm{d}Q} (W^\dagger W)^{-1} \phi_1 + h.c. \end{aligned} \tag{2.100}$$

This requires just one inversion, solving $(W^\dagger W)\chi = \phi_1$ for χ. As for the correction term S_2, we have force

$$\begin{aligned} F_2 &= \frac{\mathrm{d}}{\mathrm{d}Q} \left[\phi_2^\dagger W (D^\dagger D)^{-1} W^\dagger \phi_2 \right] \\ &= \phi_2 \frac{\mathrm{d}W}{\mathrm{d}Q} (D^\dagger D)^{-1} W^\dagger \phi_2 + \phi_2 W (D^\dagger D)^{-1} \frac{\mathrm{d}W^\dagger}{\mathrm{d}Q} \phi_2 + \phi_2^\dagger W \frac{\mathrm{d}(D^\dagger D)^{-1}}{\mathrm{d}Q} W^\dagger \phi_2 \\ &= \phi_2 \frac{\mathrm{d}W}{\mathrm{d}Q} (D^\dagger D)^{-1} W^\dagger \phi_2 - \phi_2^\dagger W (D^\dagger D)^{-1} D^\dagger \frac{\mathrm{d}D}{\mathrm{d}Q} (D^\dagger D)^{-1} W^\dagger \phi_2 + h.c. \end{aligned} \tag{2.101}$$

This also requires just one inversion, solving $(D^\dagger D)\chi = W^\dagger \phi_2$ for χ.

We can easily have multiple Hasenbusch filters, which is often beneficial for particularly light quark masses. For example, the two filter action is

$$S_{2-\mathrm{Hasen}} = \phi_1^\dagger (W_1^\dagger W_1)^{-1} \phi_1 + \phi_2^\dagger W_1 (W_2^\dagger W_2)^{-1} W_1^\dagger \phi_2 + \phi_3^\dagger W_2 (D^\dagger D)^{-1} W_2^\dagger \phi_3 \tag{2.102}$$

where $W_1 = D[\kappa_1]$, $W_2 = D[\kappa_2]$ and $\kappa_1 < \kappa_2 < \kappa$. As before, each of these terms can be placed on separate time-scales $h_1 < h_2 < h_3$ to minimise the cost.

Polynomial

Polynomial filtering [37] is another filtering method. If we define $K = D^\dagger D$, then the filter is a polynomial approximation $P(K)$ to the inverse K^{-1}, giving the action

$$S_{PF} = \phi_1^\dagger P(K) \phi_1 + \phi_2^\dagger [P(K)K]^{-1} \phi_2. \tag{2.103}$$

We usually express the polynomial as a product of roots,

$$P(K) = c_n \prod_{i=1}^{n} (K - z_i) \tag{2.104}$$

where $c_n \in \mathbb{R}$, $z_i \in \mathbb{C}$. Using this form, the force term for the polynomial filter term S_1 is given by

$$\begin{aligned} F_1 &= \phi_1^\dagger \frac{\mathrm{d}}{\mathrm{d}Q} \left[c_n \prod_{i=1}^{n} (K - z_i) \right] \phi_1 \\ &= \sum_{i=1}^{n} \chi_i^\dagger \frac{\mathrm{d}K}{\mathrm{d}Q} \eta_i, \end{aligned} \tag{2.105}$$

where we define

$$\chi_i = \prod_{j=1}^{i-1} (K - z_i^*) \phi_1 \tag{2.106}$$

$$\text{and} \quad \eta_i = c_n \prod_{j=i+1}^{n} (K - z_i) \phi_1. \tag{2.107}$$

Note that this force term has no inverses, which gives it a low cost. Along with the fact that it captures the high energy modes by virtue of approximation, this means polynomial filtering can benefit from multiple time-scales.

The correction term S_2, meanwhile, has force term

$$\begin{aligned} F_2 &= \phi_2^\dagger \frac{\mathrm{d}}{\mathrm{d}Q} \left[c_n^{-1} K^{-1} \prod_{i=1}^{n} (K - z_i)^{-1} \right] \phi_2 \\ &= \phi_2^\dagger \frac{\mathrm{d}}{\mathrm{d}Q} \left[c_n^{-1} \prod_{i=1}^{n+1} (K - z_i)^{-1} \right] \phi_2 \quad [z_{i+1} = 0]. \end{aligned} \tag{2.108}$$

We can express the inverted polynomial in partial fraction form

$$c_n^{-1} \prod_{i=1}^{n+1} (K - z_i)^{-1} = \sum_{i=1}^{n+1} \frac{r_i}{K - z_i}, \tag{2.109}$$

where

$$r_i = \frac{1}{c_n} \prod_{j \neq i} \frac{1}{(z_j - z_i)}. \tag{2.110}$$

Substituting this into the force term gives

$$\begin{aligned} F_2 &= \phi_2^\dagger \frac{\mathrm{d}}{\mathrm{d}Q}\left[\sum_{i=1}^{n+1}\frac{r_i}{K-z_i}\right]\phi_2 \\ &= \sum_{i=1}^{n+1}\phi_2^\dagger (K-z_i)^{-1}\frac{\mathrm{d}K}{\mathrm{d}Q}r_i(K-z_i)^{-1}\phi_2 \\ &= \sum_{i=1}^{n+1}\tilde{\chi}_i^\dagger \frac{\mathrm{d}K}{\mathrm{d}Q}\tilde{\eta}_i, \end{aligned} \tag{2.111}$$

where we define

$$\tilde{\chi}_i = (K-z_i^*)^{-1}\phi_2 \tag{2.112a}$$

$$\text{and} \quad \tilde{\eta}_i = r_i(K-z_i)^{-1}\phi_2. \tag{2.112b}$$

This involves calculating shifted inverses $(K-z_i)^{-1}\phi_2$, which can be done effectively by a multi-shift solver such as multi-shift conjugate gradient with a cost comparable to inverting the zero-shift K. Note that for a polynomial with real coefficients, the roots z_i are either real or come in complex conjugate pairs due to the Fundamental Theorem of Algebra. Thus, the conjugate z_i^* is always equal to some root z_j and so there are only $n+1$ shifted inverses required to evaluate the correction force term.

For two filter polynomial filtering, we can use two approximations $P_1(K)$ and $P_2(K)$ to the inverse K^{-1} with orders $p_1 < p_2$ such that $P_2(K)/P_1(K) = Q(K)$ is also a polynomial. This ensures the intermediate force term remains cheap. This leads to the fermion action

$$S_{2-\text{poly}} = \phi_1^\dagger P_1(K)\phi_1 + \phi_2^\dagger Q(K)\phi_2 + \phi_3^\dagger [P_2(K)K]^{-1}\phi_3. \tag{2.113}$$

One class of polynomials suitable for polynomial filtered HMC (PF-HMC) are Chebyshev approximations to the inverse. The roots of these polynomials are given by

$$z_i = \mu(1-\cos\theta_k) - i\sqrt{\mu^2-\nu^2}\sin\theta_k, \qquad \theta_k = \frac{2\pi k}{n+1}, \tag{2.114}$$

and the normalisation is given by

$$d_n = \frac{1}{\mu\prod_{i=1}^{n}(\mu - z_i)}, \tag{2.115}$$

where μ and $\nu < \mu$ are real parameters. The roots describe an ellipse in the complex plane with semi-major axis μ along the positive real axis and semi-minor

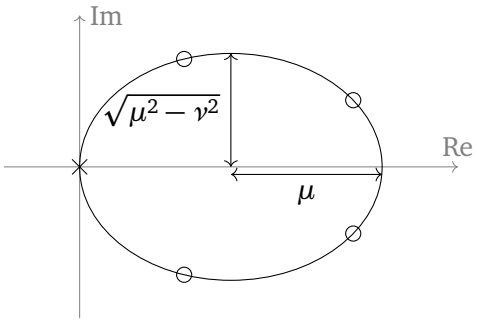

Figure 2.4: The roots of a fourth order Chebyshev polynomial approximation to K^{-1}. The roots are shown as circles, and if the origin (cross) is included they are distributed evenly around an ellipse.

axis $\sqrt{\mu^2 - \nu^2}$. If we add the origin to the set of roots, the set is distributed at equal angles around the ellipse. See Figure 2.4 for a depiction. The approximation is effective within that ellipse, so we require μ and ν to encompass the eigenvalue spectrum of the fermion matrix kernel K. These parameters can be tuned to minimise the size of the filter's force term. At the very least, we ensure $2\mu > \lambda_{max}$ where λ_{max} is the maximum real eigenvalue of K.

A useful property of Chebyshev polynomials is that if $n+1$ divides $m+1$, then the Chebyshev polynomial $P_n(K)$ factorises $P_m(K)$ assuming the same (μ, ν). This is especially useful for constructing multiple levels of polynomial filtering (2.113).

2.5.4 Iterative solvers

The most computational intensive part of a typical HMC calculation is inverting the fermion matrix $K = D^\dagger D$, so the way we perform this inversion has a significant bearing on the computational cost. As this matrix is large and sparse, the most common approach is to solve $K\chi = \phi$ for χ, up to a given precision, via iterative solvers such as conjugate gradient (CG) [38].

The different phases in a HMC simulation have different precision requirements. When generating the pseudofermions ϕ and calculating the action S at the end of a trajectory, we require a high precision to ensure that the ϕ are correctly distributed and that the Metropolis acceptance step is accurate. During a molecular dynamics trajectory, we can safely use a lower precision, because the Metropolis acceptance step corrects for any solver errors. However, a very low precision can result in large changes in the Hamiltonian ΔH and hence low acceptance rates. Large solver errors also affect the reversibility of the molecu-

lar dynamics trajectory. The size of such a reversibility violation can be tested directly: perform a molecular dynamics trajectory, redo it on the resultant state with reversed momentum, then calculate the overall change in the Hamiltonian. As long as these violations are small and ΔH is stable, the precision is safe.

The usual way to change the precision of the iterative solver is by changing the convergence criteria. A common choice is $|K\chi - \phi| < \delta$, where δ is the tolerance which can be tuned to change the precision of the solution χ.

There are advanced iterative solvers such as bi-conjugate gradient stabilised (BICGStab) [39], generalised conjugate residual (GCR) [40], and generalised minimal residual (GMRES) [41]. These improved methods reach a solution quicker than CG when we require high accuracy, which is useful when calculating ϕ and S. However, the precision required during molecular dynamics trajectories is not usually high enough for such methods to be advantageous over CG.

If the required precision is low, one can safely use reduced precision floats, e.g. 32-bit rather than 64-bit, in an iterative solver in order to reduce the memory bandwidth. Taking this idea further, one can utilise low precision solutions with low precision floats to help perform a high precision inversion with high precision floats, leading to *mixed precision solvers*. These often only require a few more matrix operations compared to a normal high precision inversion, but with the advantage of the reduced memory bandwidth from the low precision floats. As iterative solvers on large matrices are often memory-bound (i.e. the performance bottleneck is slow/insufficient memory), this can provide a significant speedup: see e.g. [42].

Another way to speed up the convergence of an iterative solver is to use *chronological inversion* [43]. This is where an initial guess χ_0 of the solution is made based upon recent previous solutions to $K\chi = \phi$. Although K changes between force updates, this change is usually small enough that this technique is still effective. One way to combine the information from previous solutions is minimal residual extrapolation (MRE), where we take our initial guess χ_0 as an average of a small number of previously stored solutions χ_i weighted by the norm of their residuals $r_i = \phi - K\chi_i$. Note that the correlation due to reusing past solutions means that the reversibility violations are larger with this technique. While this forces a higher solver precision to ensure reversibility, it still produces a considerable speedup.

In the case of polynomial filtering (section 2.5.3) and rational HMC (section 2.6.1), we often require *shifted inverses*, which are the solutions to the equations $(K + \sigma_i I)\chi_i = \phi$ with complex shifts σ_i. They can be obtained via multi-shift methods that calculate all shifted inverses simultaneously, such as multi-shift

conjugate gradient (MCG) [44]. Such methods only require a number of matrix K multiplies up to those required for the zero-shift case. Note however that chronological inversion techniques do not work with such methods, as obtaining the solution for all shifts requires starting with the initial guess $\chi_i = 0$.

2.6 Single flavour HMC

Up to this point, we have only considered double-flavour pseudofermions, with fermion action (2.29)

$$S_F = \phi^\dagger (D^\dagger D)^{-1} \phi .$$

These are useful when simulating up and down quarks, as the quarks can usually be set to the same mass without complicating physical point extrapolations. However, in the case of a single flavour such as the strange quark, we do not have a counterpart with the same mass and are hence required to calculate $\det D_s$ on its own. Single flavour fermions are also necessary for incorporating QED effects (see chapter 5). The main barrier to evaluating this is that, recalling section 2.3, using the fermion action $\phi^\dagger D_s^{-1} \phi$ causes numerical instability, so we must find some other way to evaluate $\det D_s$.

2.6.1 Rational Hybrid Monte Carlo (RHMC)

The solution lies in observation that for real and positive $\det D$,

$$\det D = \sqrt{\det D^\dagger D} = \det (D^\dagger D)^{\frac{1}{2}}. \tag{2.116}$$

We cannot calculate a square root of $K = D^\dagger D$ exactly, but we can use a high precision rational polynomial approximation $R(K) \approx K^{-1/2}$. This leads to the fermion action

$$S_{RHMC} = \phi^\dagger R(K) \phi . \tag{2.117}$$

As $R(K)$ is positive definite by construction, this resolves the instability issue. This technique is known as *Rational Hybrid Monte Carlo* [45], or RHMC for short.

For γ_5-Hermitian Dirac operators such as the Wilson Dirac operator where $\gamma_5 D \gamma_5 = D^\dagger$, $\det D$ is guaranteed to be real but could still be negative. If we have configurations where $\det D < 0$, we can still apply RHMC, but we would need to correct the sign $s = \text{sign}(\det D)$ when measuring observables by reweighting. In practice, the mass term $m^{(f)}$ in the Dirac operator is usually large enough to ensure $\det D > 0$.

The use of a rational approximation $R(K)$ instead of D^{-1} introduces a bias in the HMC trajectories. However, this effect can be easily managed. The only places where the equilibrium distribution can be affected are during the Metropolis acceptance step and during the sampling of the pseudofermions ϕ. When calculating S for the Metropolis acceptance step, we can use a different rational approximation $R^*(K) \approx K^{-1/2}$ that has high precision such that the maximum error in the approximation is less than floating-point precision. This ensures that the error in the approximation is comparable with the integration error, and hence does not affect the results. Similarly, to sample ϕ at the start of a trajectory, we use a highly precise approximation $S(K)$ with $S^\dagger(K)S(K) \approx K^{1/2}$, and relate ϕ to a Gaussian distributed χ, $P(\chi) \sim e^{-\chi^\dagger\chi}$, via $\phi = S(K)\chi$. With the bias accounted for, this allows a lower precision, cheaper rational approximation $R(K)$ to be used during the molecular dynamics step. In practice, we do not need a very high order approximation ($n \sim 30$) to achieve the required level of precision in the acceptance step, so we often use the same rational approximation $R(K)$ throughout.

Another consideration is that a rational approximation $R(K)$ typically has an effective range outside of which errors increase dramatically. This range should hence encompass the eigenvalue range $[\lambda_{\min}, \lambda_{\max}]$ of the fermion matrix K in question. While we have not done so in this work, one can scale the rational approximation $R(K)$ dynamically with a factor f via

$$R(K) \approx K^{-1/2} = f^{1/2}(fK)^{-1/2} \approx f^{1/2}R(fK), \tag{2.118}$$

and adjust the effective range to centre geometrically on the eigenvalue range of the fermion matrix.

RHMC can be used to create multiple action terms via the "n-th root trick": using an approximation $R(K) \approx K^{-1/2n}$, we can use the fermion action

$$S_F = \sum_{i=1}^{n} \phi_i^\dagger R(K)\phi_i, \tag{2.119}$$

integrating all terms on the same integration scale. This approach can be adapted to simulate double-flavour pseudofermions by using an approximation $R(K) \approx K^{-1/n}$. From here on out, we will exclusively look at the single flavour case with one term, $R(K) \approx K^{-1/2}$, but many of the methods we discuss extend to these approximations as well.

We usually use the Zolotarev rational approximation $R(K)$ to $K^{-1/2}$ in this work (see e.g. [46]), which has an analytic form in terms of Jacobi elliptic functions and is provably optimal for a rational approximation of given order and

range. Given this approximation, we can construct $S(K)$ with $S^\dagger(K)S(K) \approx K^{1/2}$ as required for generating ϕ: if we express the rational approximation as

$$R(K) = d_n \prod_{i=1}^{n} \frac{K + a_i}{K + b_i} \tag{2.120}$$

where $d_n, a_i, b_i \in \mathbb{R}$, then we can factorise the inverse as

$$R(K)^{-1} = S^\dagger(K)S(K), \tag{2.121}$$

with

$$S(K) = \frac{1}{\sqrt{d_n}} \prod_{i=1}^{n} \frac{M^* + i\sqrt{b_i}}{M^* + i\sqrt{a_i}} \tag{2.122}$$

where $M^* = \gamma_5 M$. Note that M^* is not positive-definite, so conjugate gradient methods for evaluating the shifted inverses $(M^* + i\sqrt{a_i})^{-1}\phi$ may not converge. Instead, we use the multi-shift conjugate residual method (MCR) [44]: as this requires the matrix to be Hermitian, this motivates using M^* in (2.122), as it is Hermitian for Wilson-like Dirac matrices and $(M^*)^\dagger M^* = M^\dagger M = K$.

In the more general case of finding a rational approximation $R(K) \approx K^{\pm 1/n}$, one can use iterative methods such as the Remez algorithm to find appropriate coefficients d_n, a_i, b_i for (2.120) with a given order, range and error.

2.6.2 Force terms

Given the product form of a rational approximation (2.120), the force term for the RHMC action is given by

$$\begin{aligned} F_{RHMC} &= \phi^\dagger \frac{\mathrm{d}}{\mathrm{d}Q} \left[d_n \prod_{i=1}^{n} \frac{K + a_i}{K + b_i} \right] \phi \\ &= \phi^\dagger \frac{\mathrm{d}}{\mathrm{d}Q} \left[d_n \sum_{i=1}^{n} \frac{R_i}{K + b_i} \right] \phi \\ &= \sum_{i=1}^{n} \phi^\dagger (K + b_i)^{-1} \frac{\mathrm{d}K}{\mathrm{d}Q} d_n R_i (K + b_i)^{-1} \phi \\ &= \sum_{i=1}^{n} \chi_i^\dagger \frac{\mathrm{d}K}{\mathrm{d}Q} \eta_i, \end{aligned} \tag{2.123}$$

where R_i are the residues from a partial fractions expansion

$$R_i = \frac{\prod_{j=1}^{n} (-b_i + a_j)}{\prod_{j=1, j\neq i}^{n} (-b_i + b_j)}, \tag{2.124}$$

and

$$\chi_i = (K + b_i)^{-1}\phi, \tag{2.125a}$$
$$\eta_i = d_n R_i (K + b_i)^{-1}\phi. \tag{2.125b}$$

This force term requires n shifted inverses, which can be evaluated effectively by a multi-shift solver such as MCG.

2.6.3 Filtering methods

Just as with the double-flavour pseudofermion action, filtering methods (section 2.5.3) can be applied to the RHMC action (2.117). We investigate the performance of the following methods in chapters 4 and 5.

Polynomial

Polynomial filtering (section 2.5.3) can easily be applied to the RHMC action. If we choose a polynomial $P(K) \approx K^{-1/2}$, then we can use the action

$$S_{PF-RHMC} = \phi_1^\dagger P(K)\phi_1 + \phi_2^\dagger P^{-1}(K)R(K)\phi_2. \tag{2.126}$$

The correction term is just another rational function, so the force term F_2 can be calculated similarly to the pure RHMC case (2.123).

To construct multiple filters, we choose a polynomial approximation $Q(K)$ to $P^{-1}(K)R(K)$, and the action

$$S_{2PF-RHMC} = \phi_1^\dagger P(K)\phi_1 + \phi_2^\dagger Q(K)\phi_2 + \phi_3^\dagger P^{-1}(K)Q^{-1}(K)R(K)\phi_3. \tag{2.127}$$

Choosing such an approximation $Q(K)$ ensures that the correction term's kernel $P^{-1}(K)Q^{-1}(K)R(K) \approx I$.

For this method, we can use Chebyshev polynomial approximations to $K^{-1/2}$ and $P^{-1}(K)R(K)$ for $P(K)$ and $Q(K)$ respectively. However, these do not have nice parametrisations or factorisations as in the double-flavour case (section 2.5.3), so it is easier to set $Q(K)$ as above rather than $P_2(K)$ as was the case for double-flavour polynomial filtering (2.113).

One can calculate Chebyshev polynomial approximations numerically. Suppose we have some function $f(x)$ that we wish to approximate on the range $[-1, 1]$ with an N-th order Chebyshev polynomial approximation of the form

$$P(x) = \sum_{n=0}^{N-1} a_n T_n(x) \tag{2.128}$$

where $a_n \in \mathbb{R}$ and $T_n(x)$ are Chebyshev polynomials of the first kind. We can calculate the optimal coefficients a_n that minimize the size of the relative error $(P(x)-f(x))/f(x)$ numerically via the discrete cosine transform

$$a_n \approx \frac{2-\delta_{0n}}{N}\sum_{k=0}^{N-1}\cos\left(\frac{n\pi(k+1/2)}{N}\right)f(x_k) \tag{2.129}$$

where $x_k = \cos\left(\frac{\pi(k+1/2)}{N}\right)$. To extend this to approximating $f(x)$ on an arbitrary range $[a,b]$, we calculate the optimal coefficients a_n for the shifted function

$$g(x) = f\left(\frac{a-b}{2}x + \frac{a+b}{2}\right), \tag{2.130}$$

then use the approximation

$$P(x) = \sum_{n=0}^{N-1} a_n T_n\left(\frac{2}{a-b}\left(x - \frac{a+b}{2}\right)\right). \tag{2.131}$$

This is method used when we investigate PF-RHMC in chapter 4.

Truncated ordered product RHMC (tRHMC)

A filtering method exclusive to RHMC splits the product form of the rational approximation (2.120). If we order the shifts such that $a_i > a_{i+1}$, $b_i > b_{i+1}$, then the truncated ordered product

$$R_{0,j}(K) = d_n \prod_{i=1}^{j} \frac{K+a_i}{K+b_i} \tag{2.132}$$

also forms an approximation to $K^{-1/2}$. Thus, if we split the rational approximation between action terms,

$$S_{tRHMC} = \phi_1^\dagger R_{0,j}(K)\phi_1 + \phi_2^\dagger R_{j,n}(K)\phi_2 \tag{2.133}$$

where we define

$$R_{i,j}(K) = \prod_{k=i+1}^{j} \frac{K+a_k}{K+b_k} \tag{2.134}$$

for $i \neq 0$ (i.e. without the normalisation d_n), then the filter term S_1 has a cheaper to evaluate force than S_2 due to the larger shifts and captures the high energy modes of the system by virtue of approximation. Therefore, this filtering method can engineer a reduction in cost by placing S_1 on a finer integration scale than

S_2. We denote this method *truncated ordered product RHMC*, or tRHMC for short. Note that as each action term is a rational polynomial, their force terms can be evaluated via (2.123) just as in pure RHMC.

This method easily extends to multiple filters. For example, the 2-filter tRHMC action is

$$S_{2tRHMC} = \phi_1^\dagger R_{0,t_1}(K)\phi_1 + \phi_2^\dagger R_{t_1,t_2}(K)\phi_2 + \phi_3^\dagger R_{t_2,n}(K)\phi_3 \tag{2.135}$$

where $0 < t_1 < t_2 < n$.

There is a similar method [47] that splits the sum over poles expression of the rational approximation,

$$R(K) = d_n \sum_{i=1}^{n} \frac{R_i}{K + b_i}. \tag{2.136}$$

However, tRHMC has the advantage that the filter term $R_{0,j}(K)$ approximates $K^{-1/2}$, and thus acts as a high-pass filter.

∘ 3 ∘

Benchmarking of double-flavour optimisations

The results in this chapter were originally published in Computer Physics Communications [32].

Our first investigation into configuration generation optimisations concerns filtering methods on the double-flavour pseudofermion action, which are often used to simulate the up and down quarks. As there has been widespread use of mass preconditioning, we use this as a benchmark to compare with polynomial filtering for the cost to generate configurations. Refer to subsection 2.5.3 for details of these methods.

3.1 Lattice setup

We used a modified version of the BQCD program [48] to investigate these filtering methods, with the changes now available in version 5. We thermalised a small $16^3 \times 32$ lattice with $n_f = 2$ Wilson fermions with even-odd preconditioning at $\kappa = 0.15825$, giving pion mass $m_\pi \sim 400$ MeV. The gauge coupling is $\beta = 6/g^2 = 5.6$, providing a lattice spacing of $a \sim 0.08$ fm [36]. This was thermalised with 1000 trajectories of length $\tau = 1$, using two Hasenbusch filters. This relatively small and heavy lattice was chosen to ensure that we could accumulate configurations using a wide variety of filters without requiring large amounts of computational resources.

The goal of gauge configuration generation optimisations is to reduce the amount of computational work required to generate statistically independent lattice configurations. Recalling the arguments in subsection 2.4.3, we use the

cost function

$$C = N_{\text{mat}}/P_{\text{acc}}, \tag{3.1}$$

where N_{mat} is the average number of matrix operations per trajectory and P_{acc} is the acceptance rate (dropping the '$\langle\rangle$' notation). In particular, we set N_{mat} to be the average number of fermion matrix $K = M^\dagger M$ (and, for mass preconditioning, $J = W^\dagger W$) multiplies per trajectory, counting D, $D^\dagger$ etc. as half a multiply. This choice of cost function has been used before [49]. Here, we use the uncorrelated standard error (B.8) for the cost function in our results to indicate the variance in the data.

We attempt to tune the acceptance rate to the range $P_{acc} = [0.65, 0.75]$ for consistency. For each set of filters chosen, we used the second-order minimal norm scheme (2.57) with overlaid integrators for multiple time-scales (section 2.5.2). The solver we use for fermion matrix inversions is conjugate gradient, with convergence criteria $|Ax - b| < \delta$ where δ is the tolerance. This tolerance is set to 10^{-8} during molecular dynamics trajectories, and to 10^{-10} when constructing the pseudofermions or performing the Metropolis acceptance step (see section 2.5.4 for why these are set differently). In the case of mass preconditioning, we also use chronological inversion with minimal residual extrapolation from seven prior solutions (section 2.5.4); this reflects what is commonly used in practice.

3.2 Comparing polynomial and mass filtering

We start by comparing the two methods using a single filter. Recalling section 2.5.3, the fermion actions in question are

$$S_{1MP} = \phi_1^\dagger (W^\dagger W)^{-1} \phi_1 + \phi_2^\dagger W (D^\dagger D)^{-1} W^\dagger \phi_2 \tag{3.2}$$

and

$$S_{1PF} = \phi_1^\dagger P(K) \phi_1 + \phi_2^\dagger [P(K)K]^{-1} \phi_2, \tag{3.3}$$

where $K = D^\dagger D$. These filtering schemes have a number of parameters to tune. The mass preconditioned action (1MP) has the free parameter $\kappa' < \kappa$. As for polynomial filtering, we use Chebyshev polynomials (equations (2.114) and (2.115)) and select the ellipse parameters $(\mu, \nu) = (1.2, 0.9)$ to minimise the force term, holding these values fixed for all the runs considered in this chapter. The polynomial action (1PF) then has only one free parameter: p, the polynomial order. Finally, as we are using a multi-scale integrator, both actions have 3 step-sizes $\{h_0 = h_G, h_1, h_2\}$ to tune.

We tune the parameters as follows. We sample a set of appropriate κ' and p, since the optimal values are hard to know *a priori*. The step-sizes $\{h_G, h_1, h_2\}$ are then tuned via *force-balancing*: this is where the force term F_i and the step-size h_i for each action term are set such that

$$F_i h_i \approx \text{constant}. \tag{3.4}$$

In our case, we use the maximal forces for F_i in this equation as this was found to better reflect the energy mode differences between action terms. Equation (3.4) fixes the ratios between step-sizes, leaving the coarsest step-size h_2 to tune. This is set such that the acceptance rate lies in the target range, namely $[0.65, 0.75]$. However, since the gauge term S_G is very cheap to calculate, we ignore this prescription and simply set the step-size h_G to be sufficiently small such that reducing the step-size further does not change the acceptance rate. Here, we used $h_G = 1/480$ for every run.

The resulting parameter choices are given in Tables 3.1 and 3.2; note that we express the step-sizes in terms of the number of steps n_j at each scale, which are related to h_j via $h_j = \tau/n_j$ where τ is the trajectory length ($\tau = 1$ here). We also show the average number of K (and J) multiplications required to evaluate the forces as a basis for comparison between the two methods.

Table 3.1: Single polynomial filter parameters. N_{traj} is the number of trajectories. 'mat/F_i' denotes the average number of matrix multiplications by K to evaluate the force F_i. There is no inversion required for F_1, so the number of matrix multiplications needed is exactly $2.5p-2$ (see section 2.5.3). Otherwise, statistical errors are of the same order as the last significant figure.

p	μ	ν	n_2	n_1	n_0	N_{traj}	mat/F_1	mat/F_2
4	1.2	0.9	48	120	480	2000	8	690
10			36	160	480	2000	23	777
20			24	240	480	2000	48	758

Figure 3.1 shows the cost function C (3.1) for the mass preconditioned and the polynomial filtered actions respectively. Looking at this figure, we see that a single mass filter provides a better overall performance than a single polynomial filter, with a cost of $C = 43,800 \pm 3,500$ at $\kappa' = 0.1545$ compared with $C = 87,500 \pm 7,400$ at $p = 10$. For completeness, plots of the acceptance rate P_{acc} and the number of matrix operations N_{mat} are given in Figures C.1 and C.2 respectively.

Given that the cost to evaluate the filter term F_1 is significantly less for the polynomial filter (Table 3.1) than for the mass filter (Table 3.2), it is worthwhile

Table 3.2: Single mass filter parameters. Refer to Table 3.1 for the legend.

κ'	n_2	n_1	n_0	N_{traj}	mat/F_1	mat/F_2
0.154	8	120	480	2000	84.55	719
0.1545	7	96	480	2000	114.4	671
0.155	7	120	480	2000	112.7	697
0.1555	6	120	480	2000	135.9	754
0.156	5	120	480	2000	173.5	755

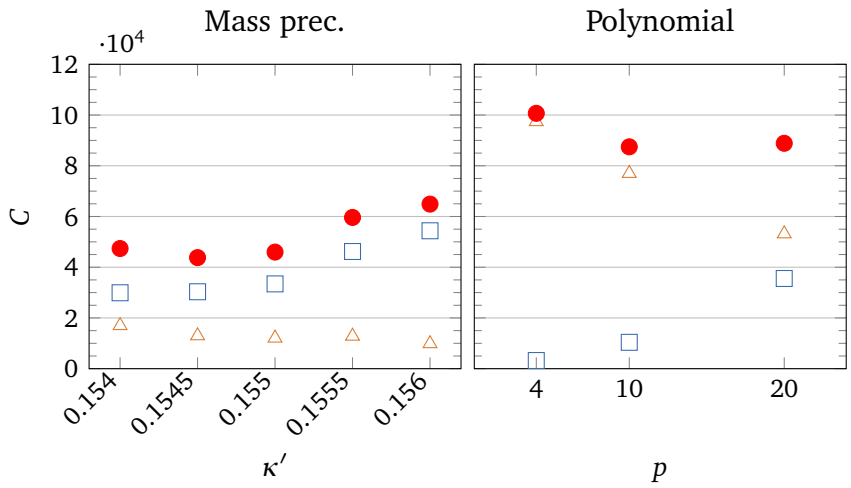

Figure 3.1: Cost function for 1-filter actions. Squares = matrix operations to construct F_1, triangles = F_2 construction, filled circles = total. Statistical errors are smaller than marker size.

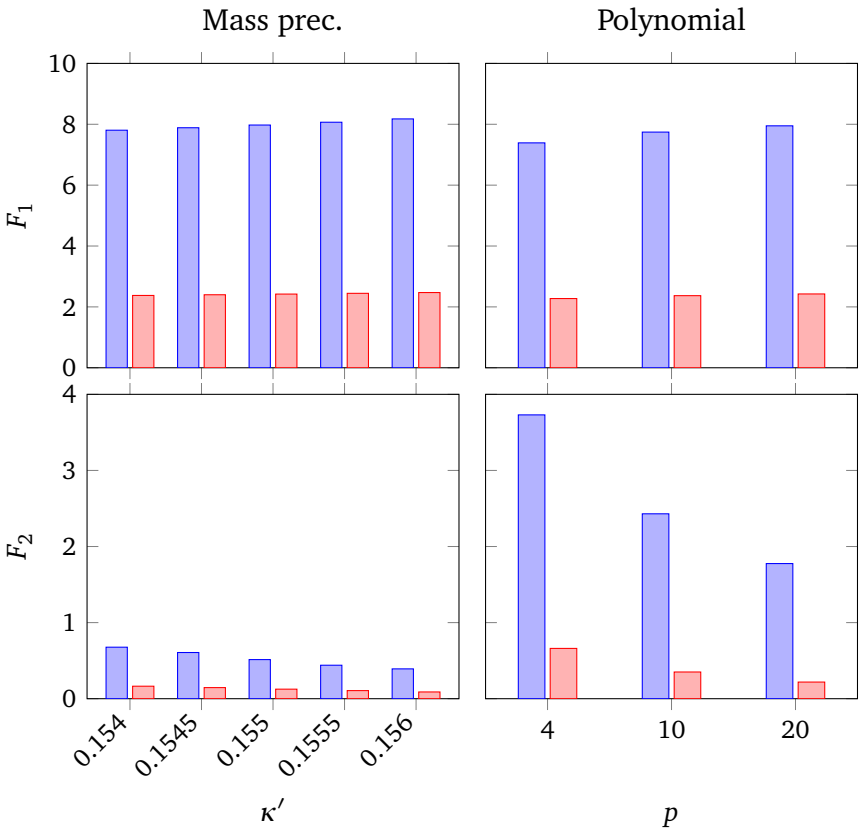

Figure 3.2: 1-filter forces. The left hand plots show the mass preconditioned action's forces whilst the right hand plots show the polynomial filtered action's forces. For each fermion term S_1, S_2, the maximal and average forces are plotted for each choice of κ'/p.

to try to further understand the difference between the two filters. We can do this by considering the force terms shown in Figure 3.2. We see that the force for the filter term F_1 is similar in both cases. However, the average and maximal forces for the correction term F_2 are much larger in the polynomial case than in the mass preconditioning case. This leads to more molecular dynamics steps n_2 via (3.4) for PFHMC (see Table 3.1), and is the main reason for the higher cost. As shown in Table 3.1 and indicated by the squares in the right-hand graph of Figure 3.1, increasing the polynomial order to reduce this force simultaneously increases the cost to calculate F_1, making polynomials of very large order inefficient filters.

The results for a single filter term stand to reason. Given that the Hasenbusch filter is constructing a Krylov-space polynomial to approximate the inverse, a short polynomial term of order 10 cannot capture as much of the dynamics as a Hasenbusch filter that requires 80 or more iterations to invert.

In order to improve the performance of the polynomial filtered action, we can introduce multiple polynomial filters without any additional fine tuning [37]. Here, we consider the two polynomial filter (2PF) action

$$S_{2PF} = \phi_1^\dagger P_1(K)\phi_1 + \phi_2^\dagger Q(K)\phi_2 + \phi_3^\dagger [P_2(K)K]^{-1}\phi_3. \tag{3.5}$$

We set the first polynomial's order to $p_1 = 4$ to keep the cost of F_1 low, then vary the order of the intermediate polynomial $q = p_2 - p_1$. The parameter set is shown in Table 3.3. The cost function for 2PF is shown in Figure 3.3 alongside 1MP for comparison. For completeness, plots of the acceptance rate P_{acc} and the number of matrix operations N_{mat} for 2PF are given in Figures C.3 and C.4 respectively. The minimum of $C = 47,700 \pm 3,700$ here at $q = 50$ is a marked improvement over 1PF's minimum of $C = 87,500 \pm 7,400$, and is quite comparable to 1MP's performance. Looking at the contributions to the cost function, we see that increasing q for 2PF does not increase the cost to evaluate F_2 (triangles in Figure 3.3) nearly as much as the corresponding case of increasing p for 1PF (squares in Figure 3.1).

Table 3.3: Configuration parameters for 2PF

p_1	p_2	μ	ν	N_{traj}	n_3	n_2	n_1	n_0
4	24	1.2	0.9	2000	24	20	108	480
	34			2000	20	16	80	480
	54			2000	16	30	120	480

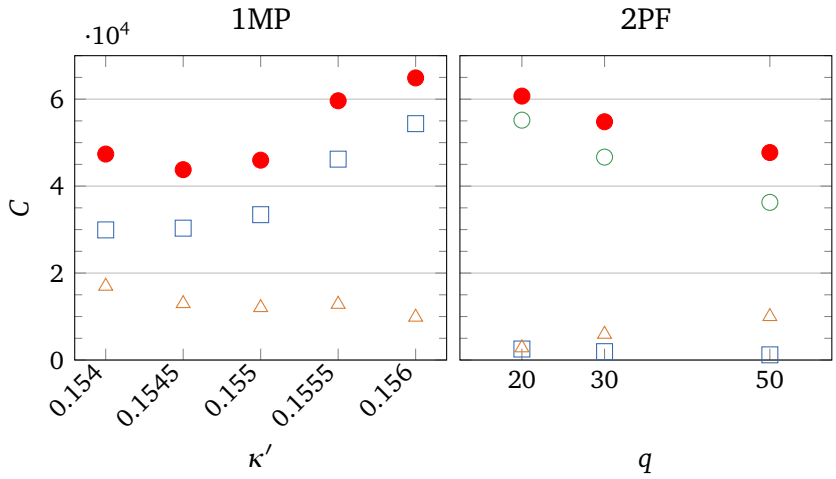

Figure 3.3: Cost function for 1MP versus 2PF. Squares = matrix operations to construct F_1, triangles = F_2 construction, empty circles = F_3 construction, filled circles = total.

3.3 Polynomial-filtered mass-preconditioning

3.3.1 Motivation

At this point it is pertinent to make some remarks comparing the relative efficacy of polynomial filtering and mass preconditioning.

Mass preconditioning (2.99) works best when the difference between the Hasenbusch mass and the target quark mass $\Delta m = m' - m$ is small, as this implies that $J(m')K^{-1}(m) \simeq I$ ($J = W^\dagger W$) and hence the force term is correspondingly reduced. However, when Δm and hence m' is made smaller, the inversion cost to evaluate $J^{-1}\phi$ is increased. At light quark masses, a single Hasenbusch filter is unable to simultaneously satisfy the criteria that the filtered force term F_2 is reduced and the high frequency term F_1 is cheap to evaluate. Hence, to achieve a computationally efficient frequency-splitting scheme, light quark mass simulations introduce multiple mass preconditioning terms [21, 36, 50, 51] that distribute the mass differences across multiple Hasenbusch masses $m < m' < m'' < m''' \ldots$. As it is not possible to know *a priori* the inversion cost for a given term, this requires performing simulations to tune the hierarchy of Hasenbusch mass parameters, which becomes more labour-intensive as more scales are introduced. Previous experience can help guide the choice of parameters, but the extent to which this choice is optimal depends on the ensemble, quark masses and gauge coupling being similar to a past run or another published parameter set.

Meanwhile, the efficacy of polynomial filtering (2.103) depends on two factors: the choice of the polynomial and the spectral range of the matrix whose inverse is being approximated. Specifically, the smaller the spectral range of the matrix K, the smaller the order of the polynomial required to achieve a given accuracy.

In our case, we use a Chebyshev approximation $P(z) \simeq 1/z$ whose roots lie on an ellipse. Choosing the parameters (μ, ν) that determine the ellipse is straightforward: one can simply evaluate the size of the force term while adjusting (μ, ν) and look for a minimum. In practice, one finds that the minimum is relatively shallow and hence fine-tuning of (μ, ν) is not required once a reasonable pair of values has been found.

Once this process has been completed, the only remaining parameter to choose is p, the order of the polynomial approximation. The choice of p allows one to directly determine the cost of the high frequency filter term. As p must be an integer, there is no fine-tuning.

Higher values of p provide a greater reduction in the force for the low frequency correction term F_2, but correspondingly increase the cost for the filter term F_1. Hence it is beneficial to make use of multiple polynomial filtering terms (3.5) to introduce additional frequency scales [37]. An advantage of polynomial filtering over mass-preconditioning is that the introduction of an additional scale simply involves choosing another (integer) polynomial order q and hence does not require additional fine-tuning.

Noting that we could approximate the inverse exactly if we had a polynomial of very high order, we can consider the order of the polynomial filter as a means of interpolating between the high and low frequency scales. The effectiveness of polynomial filtering is best in the high frequency regime, associated with high energy scales. As we move to lower frequency scales, the order of polynomial required to capture the dynamics increases significantly and the Chebyshev approximation becomes inefficient when compared with a Krylov-space construction. On the other hand, at low frequency scales mass preconditioning becomes more effective as Δm becomes smaller and hence $J(m')K(m)^{-1} \sim I$.

This observation leads us to propose applying a polynomial filter (or several) to a mass preconditioned fermion action, giving

$$S_{PF-MP} = \phi_1^\dagger P(J)\phi_1 + \phi_2^\dagger [JP(J)]^{-1}\phi_2 + \phi_3^\dagger W K^{-1} W^\dagger \phi_3. \tag{3.6}$$

As $m' > m$, the condition number and hence spectral range of $J(m')$ is reduced in comparison to that of $K(m)$, and hence the accuracy of the polynomial $P(J)$ is better than that of $P(K)$ at a fixed order. The use of short polynomials then provides a good approximation to the high energy fluctuations and is cheap to

evaluate, simple to tune and provides direct control over the cost of the highest filtering terms. As the highest energy scales are filtered out using polynomials, the filtered mass preconditioner $P^{-1}(J)J^{-1}$ can be placed on a coarse time scale. Hence, the Hasenbusch mass parameter m' can be chosen such that Δm is small to better reduce the force when evaluating the mass preconditioned quark mass term $WK^{-1}W^{\dagger}$. The combined algorithm, which we refer to as polynomial-filtered mass-preconditioned HMC (PF-MP HMC) promises to provide the computational benefit of multiple filters with simpler tuning in comparison to plain mass preconditioning.

3.3.2 Comparison to mass preconditioning

In order to test the performance of the PF-MP scheme, we compared it against using an action with two mass preconditioners (2MP),

$$S_{2MP} = \phi_1^{\dagger} J_1^{-1} \phi_1 + \phi_2^{\dagger} W_1 J_2^{-1} W_1^{\dagger} \phi_2 + \phi_3^{\dagger} W_2 K^{-1} W_2^{\dagger} \phi_3, \tag{3.7}$$

as this type of scheme is relatively common [21, 36, 50, 51]. For the 2MP action we have the Hasenbusch filters $J_1(\kappa_1) = W_1^{\dagger} W_1$ and $J_2(\kappa_2) = W_2^{\dagger} W_2$, with $\kappa_1 < \kappa_2 < \kappa$. For the PF-MP action we have the order p of the polynomial term $P(J)$ and the mass $\kappa' < \kappa$ of the Hasenbusch term $J(\kappa')$. The cheapest filter in each case was fixed — $\kappa_1 = 0.145$ for 2MP and $p = 4$ for PF-MP — and optimisation took place through the choice of intermediate filter κ_2/κ' and the choice of step-sizes $\{h_0, h_1, h_2, h_3\}$. As in the previous section, we tune the step-size ratios such that $F_i h_i \approx$ constant, then tune the coarsest step-size h_3 to the correct acceptance rate. The full range of parameters are detailed in Tables 3.4 and 3.5.

Table 3.4: Configuration parameters for 2MP

κ_1	κ_2	N_{traj}	n_3	n_2	n_1	n_0
0.145	0.154	2000	8	15	120	480
	0.155	2000	7	20	96	480
	0.1555	2000	6	20	96	480
	0.156	2000	5	20	120	480
	0.1565	1000	4	20	120	480

Figure 3.4 shows the forces for the 2MP and PF-MP runs. Whereas for the single-filter actions (section 3.2) the correction term for polynomial filtering has a much greater force variance than that for mass preconditioning, here, the corresponding polynomial correction term S_2 for PF-MP has a maximal force only

Table 3.5: Configuration parameters for PF-MP

p	μ	ν	κ'	N_{traj}	n_3	n_2	n_1	n_0
4	1.2	0.9	0.154	2000	8	20	80	480
			0.155	2000	6	20	120	480
			0.1555	2000	5	20	120	480
			0.156	2000	5	30	120	480
			0.1565	2000	4	30	120	480

slightly larger than that of the Hasenbusch correction term S_2 for 2MP. This supports the prior argument that polynomial filtering (at a fixed order) is more effective on $J(\kappa')$ than on $K(\kappa)$; we are filtering at a heavier mass $\kappa' < \kappa$, with an associated suppression in the long range physics.

Figure 3.5 shows the cost function for the two actions in question. The optimal point for 2MP is at $\kappa_2 = 0.1555$ with cost $C = 31,000 \pm 2,200$, whereas for PF-MP it is at $\kappa' = 0.155$ with cost $C = 29,000 \pm 1,800$. For completeness, plots of the acceptance rate P_{acc} and the number of matrix operations N_{mat} are given in Figures C.3 and C.4 respectively. We see that the PF-MP scheme can perform just as well as mass preconditioning in this instance.

Note that for both 2MP and PF-MP, the cost increases for higher values of the intermediate filter (κ_2/κ'). This is due to the increasing cost to evaluate the correction force terms F_2 (triangles in Figure 3.5), which are caused by S_2 capturing a larger portion of the energy spectrum. This increased dependence on one action term reduces the effectiveness of splitting up the action.

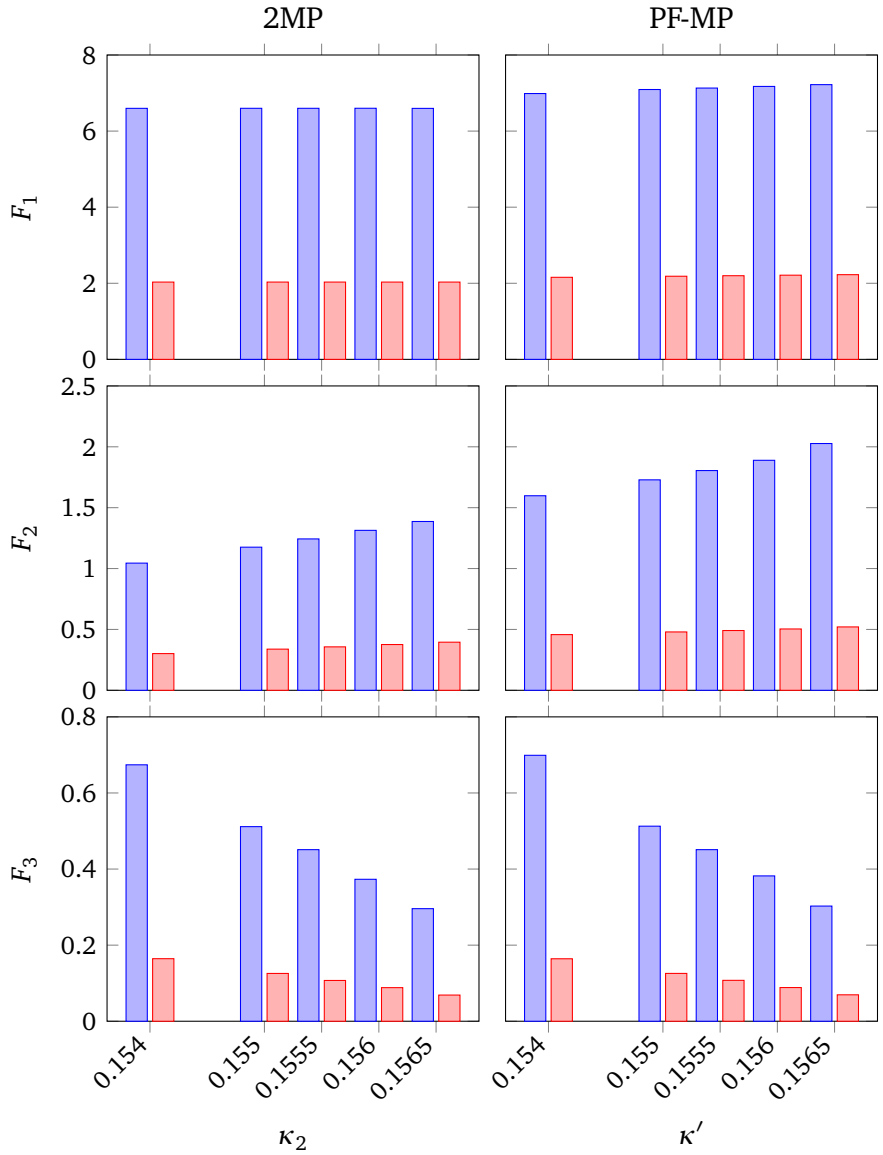

Figure 3.4: 2-filter forces. The left hand plots show the 2MP forces whilst the right hand plots show the PF-MP forces. For each fermion term S_1, S_2, S_3, the maximal and average forces are plotted for each choice of κ'. The third force F_3 is the same in each case because the third action term S_3 is also the same.

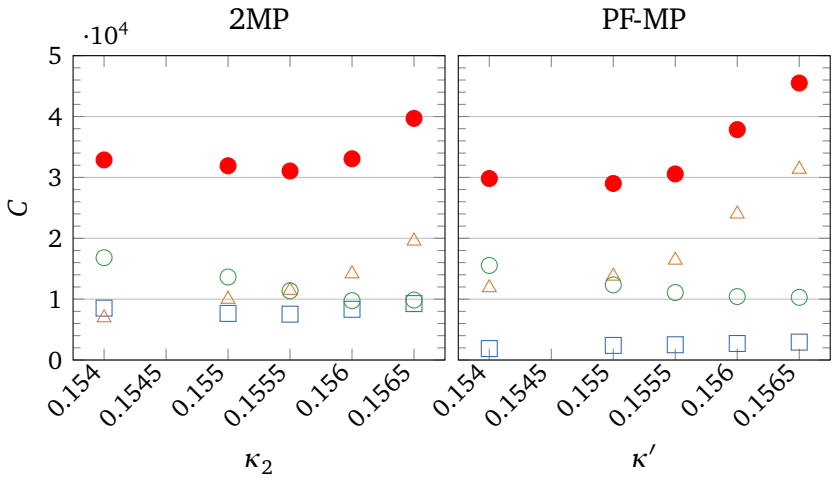

Figure 3.5: Cost function for 2-filter actions. The squares = matrix ops from constructing F_1, triangle = F_2 construction, empty circles = F_3 construction, filled circles = total.

3.3.3 3-filter actions

We have examined the PF-MP action in the case of a single polynomial filter applied to a single mass preconditioner, which for clarity we denote as 1PF-1MP. Within the PF-MP scheme, as for plain polynomial filtering, we use two polynomial terms to see if the introduction of an additional intermediate scale provides any additional benefit. This does not require any additional fine tuning, as the choice of polynomial orders p_1, p_2 provides direct control over the cost and scale of the filter terms, independent of the quark mass. We denote the scheme with two polynomial filters and a single mass preconditioner as 2PF-1MP:

$$\begin{aligned} S_{2PF-1MP} = \phi_1^\dagger P_1(J)\phi_1 + \phi_2^\dagger Q(J)\phi_2 \\ + \ \phi_3^\dagger [J P_2(J)]^{-1}\phi_3 + \phi_4^\dagger W K^{-1} W \phi_4. \end{aligned} \tag{3.8}$$

For completeness, we also examine the 1PF-2MP scheme with a single polynomial filter and two levels of mass preconditioning,

$$\begin{aligned} S_{1PF-2MP} = \phi_1^\dagger P(J_1)\phi_1 + \phi_2^\dagger [J_1 P(J_1)]^{-1}\phi_2 \\ + \ \phi_3^\dagger W_1 J_2^{-1} W_1^\dagger \phi_3 + \phi_4^\dagger W_2 K^{-1} W_2^\dagger \phi_4; \end{aligned} \tag{3.9}$$

however, this does introduce an additional mass parameter that requires fine tuning.

For the 1PF-2MP scheme, we fix the polynomial order at $p = 4$ as with PF-MP, and set κ_1 to 0.145 to match the 2MP runs. For the 2PF-1MP scheme, we set

$p_1 = 4, p_2 = 24$. This leaves a single Hasenbusch parameter κ_2/κ' to tune in both cases. See Tables 3.6 and 3.7 for a full list of the resultant parameters. The forces for 1PF-2MP and 2PF-1MP are shown in Figure 3.6; note that the forces associated with the S_3 term are significantly smaller for 2PF-1MP than for 1PF-2MP. Figure 3.7 shows the cost function. For completeness, plots of the acceptance rate P_{acc} and the number of matrix operations N_{mat} are given in Figures C.5 and C.6 respectively.

Table 3.6: Configuration parameters for 1PF-2MP

p	μ	ν	κ_1	κ_2	N_{traj}	n_4	n_3	n_2	n_1	n_0
4	1.2	0.9	0.145	0.153	2000	11	12	16	96	480
				0.154	2000	8	15	15	96	480
				0.1555	2000	6	20	20	96	480
				0.1565	2000	4	24	20	96	480

Table 3.7: Configuration parameters for 2PF-1MP

p_1	p_2	μ	ν	κ'	N_{traj}	n_4	n_3	n_2	n_1	n_0
4	24	1.2	0.9	0.153	2000	10	5	16	80	480
				0.154	2000	9	6	24	120	480
				0.1555	2000	6	8	20	120	480
				0.1565	2000	4	10	24	120	480

For ease of comparison, the cost function for all the actions considered in this chapter are presented in Figure 3.8, aside from 1PF which has a significantly higher cost than the other actions. Looking at this figure, the three PF-MP schemes all have a similar cost minimum, which is as good as or better than the 2MP benchmark. The differentiator is their dependence on the free mass parameter, κ' or κ_2. We can see that for 2MP, 1PF-1MP and 1PF-2MP that choosing κ'/κ_2 too large can lead to a significant increase in the cost function (see e.g. $\kappa'/\kappa_2 = 0.1565$), where as the 2PF-1MP cost function has only a very weak dependence on the Hasenbusch mass parameter. Comparing the contributions from evaluating F_2 for 2MP, 1PF-1MP (triangles in Figures 3.5) and F_3 for 1PF-2MP, 2PF-1MP (squares in Figures 3.7), we see that the difference lies in the slower increase in the cost of F_3 for 2PF-1MP as we increase the mass parameter κ'. This is due to the smaller force term F_3 which, by force balancing, leads to fewer integration steps for S_3. This demonstrates that no fine tuning of κ' is required for 2PF-1MP to achieve optimal performance.

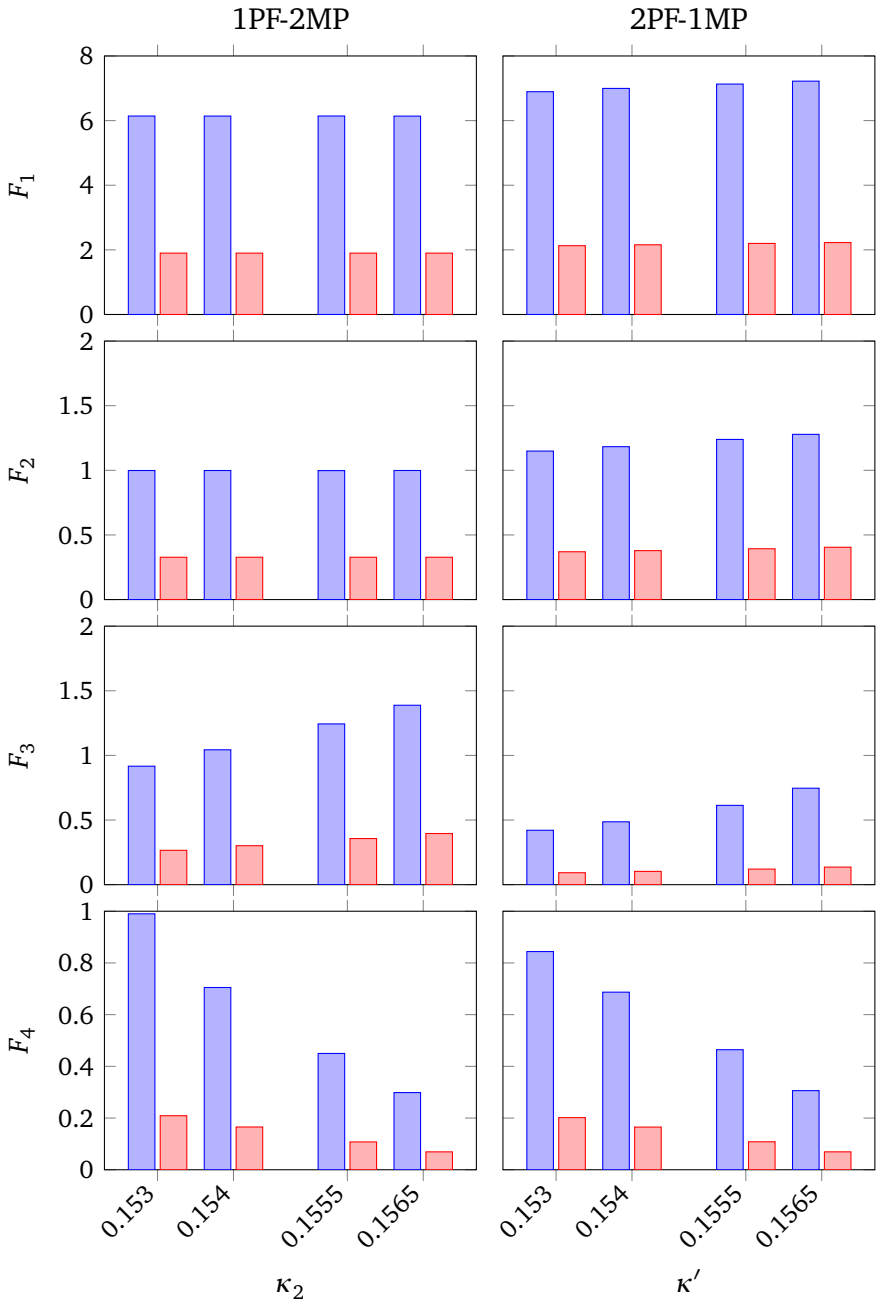

Figure 3.6: Forces for the 3-filter actions. The left hand plots show the 1PF-2MP forces whilst the right hand plots show the 2PF-1MP forces. For each fermion term, the maximal and average forces are plotted for each choice of κ_2/κ'.

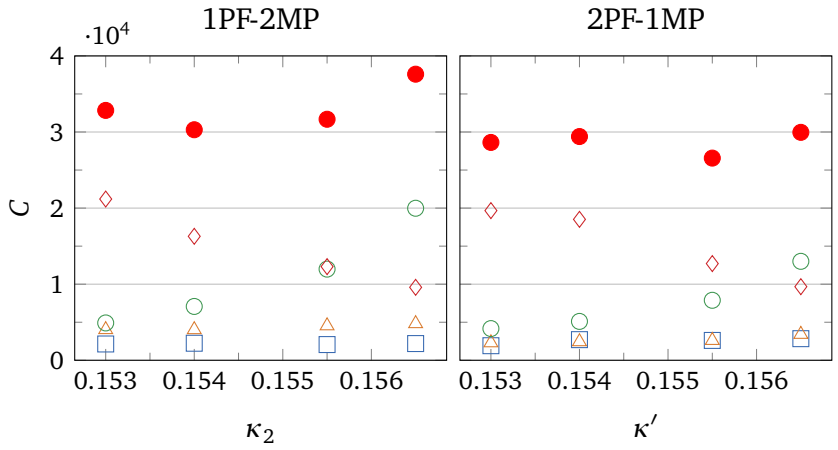

Figure 3.7: Cost function for 3-filter actions. The squares = matrix ops due to constructing the force F_1, triangles = F_2 construction, empty circles = F_3 construction, diamonds = F_4 construction, filled circles = total.

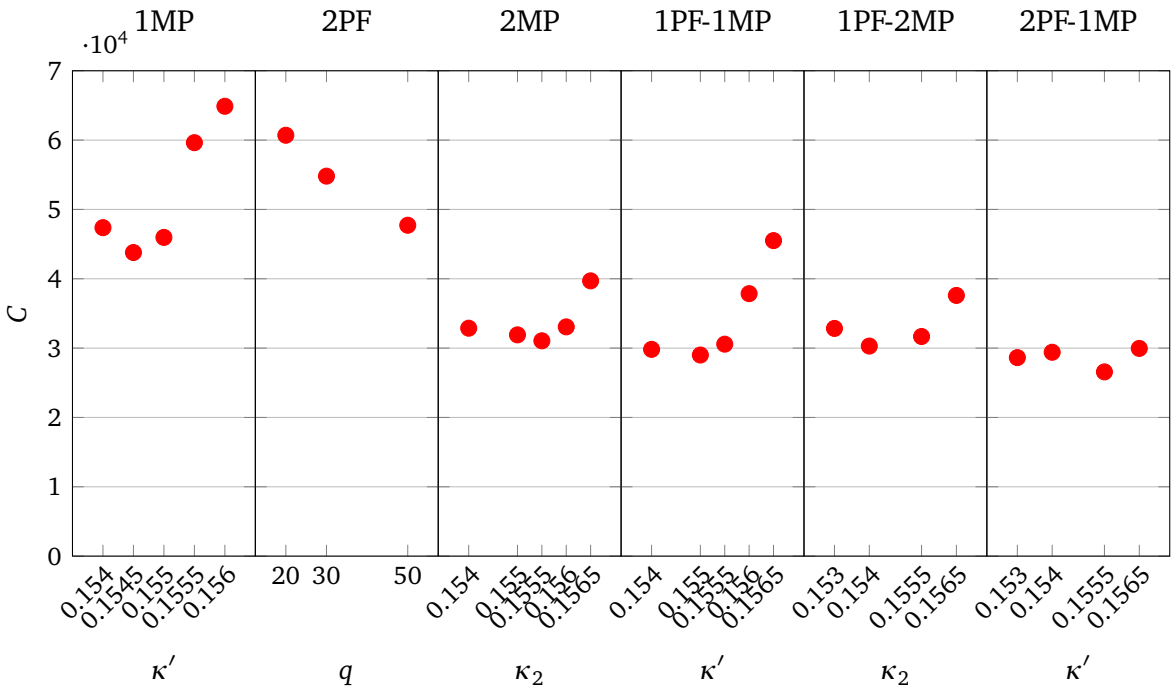

Figure 3.8: Cost function (3.1) for this chapter's actions.

3.4 Concluding remarks

We have compared the polynomial filtered and mass preconditioned HMC algorithms, and found that a 2-level polynomial filter provides a benefit similar to a single mass preconditioner. We proposed combining the two methods to provide a multi-level frequency-splitting scheme with minimal fine tuning of the action parameters. This was partly motivated by noting that the values (μ, ν) determining the Chebyshev polynomial roots produce a shallow minimum in the polynomial force term, and hence do not need fine tuning, leaving the polynomial order p as the only free parameter.

Any form of Sexton–Weingarten integration with a large number of terms requires a sensible choice of the relative time scales to achieve good performance. The tuning of the different time steps for our study of multi-level algorithms was aided by using an overlaid integration scheme, permitting any choice of step-size for each action term. This made it simple to use the force balancing method '$F_i h_i = \text{constant}$' to select the scale for each action term based on its (maximal or average) force.

The polynomial-filtered mass-preconditioned (PF-MP) algorithm was investigated with $n_f = 2$ flavours of dynamical quarks, using several different combinations of polynomial and Hasenbusch filters, and compared to 2-level mass preconditioning (2MP) as a baseline. We found that the 2PF-1MP action yielded a cost function that was as good as or better than the 2MP action, with a significant reduction in the tuning effort required to optimise the overall cost. The 2MP action has two real Hasenbusch parameters κ_1, κ_2 that need to be tuned. In contrast, the 2PF-1MP action did not need any fine tuning: it showed almost no dependence on the Hasenbusch parameter κ', and the orders of the polynomial terms (as integers) were easily chosen to optimise the cost.

This study was performed at an intermediate quark mass $m_\pi \sim 400$ MeV as a proof of the viability of the PF-MP scheme. Simulations at lighter quark masses typically introduce additional filters to further ameliorate the cost of these simulations, with some groups even using 6-level mass preconditioning [51]. At these light quark masses, the PF-MP algorithm can potentially provide an easier path to gain the benefits of multi-level frequency splitting.

∘ 4 ∘

Benchmarking of single-flavour optimisations

The results in this chapter were submitted for publication to Computational Physics Communications on the 19th of June, 2018 [52].

The subject of the next investigation is filtering methods for single-flavour pseudofermions, i.e. pseudofermions simulated using RHMC (section 2.6.1), which are used to simulate the strange quark and for simulations that incorporate electromagnetic effects (chapter 5). Compared to the double-flavour case, there are relatively few investigations in the literature of filtering methods for single-flavour simulations. Here, we compare the performance of polynomial filtered RHMC (PF-RHMC) with truncated ordered product RHMC (tRHMC), which were introduced in section 2.6.3.

4.1 Lattice setup

For this investigation, we reuse the lattice from chapter 3, performing comparisons on the $16^3 \times 32$ lattice with two (degenerate) individual flavours of Wilson fermion, i.e. a $n_f = 2$ simulation performed with $n_f = 1 + 1$ methods. Note that while the two single-flavour fermion actions are treated equally on this lattice, this is not a requirement for the filtering methods studied here: these apply to each fermion flavour individually, and hence allow for the possibility that the fermions have different masses and/or electric charge, the latter of which will be explored in chapter 5. Both the filtering techniques discussed herein have been implemented into version 5 of BQCD [48].

The cost metric is the same as for the double-flavour investigation, namely

(3.1):

$$C = N_{\text{mat}}/P_{\text{acc}}.$$

Here, we use the total cost of both pseudofermions to ease comparison with the results from chapter 3.

4.2 Tuning the step-sizes

Just as in the double-flavour pseudofermion case (chapter 3), we initially tune our step-sizes h_i via *force-balancing*, where the maximal forces F_i satisfy (3.4):

$$F_i h_i \approx \text{constant}.$$

The gluon action step-size $h_0 = h_G$ is excluded from this scheme: as it has a very cheap force term, it is safe to set $n_0 = n_G = 480$ and neglect further tuning.

The degree of freedom remaining after force-balancing is used to tune the acceptance rate P_{acc} to the range $[0.65, 0.75]$. Finding an appropriate coarsest step-size h_n can be aided by the observation that for pure HMC or RHMC, the acceptance rate as a function of the step-size h is approximately given by [53]

$$P_{acc} \approx \text{erfc}\left(\frac{h^2}{c^2}\right), \tag{4.1}$$

where c is a fitting parameter known as the characteristic scale. For the filtering schemes considered here, we find that the final correction term S_n with the 'coarsest' step-size h_n captures most of the dynamics of the system, so to a good approximation we can replace h with h_n in the above expression. To show that this works, we present fits to the complementary error function for RHMC and PF-RHMC ($p = 4$) against step-count $n = 1/h$ in Figures 4.1 and 4.2. While the fits are not perfect, they are superior to guesswork for determining the appropriate step-size h_n for obtaining a given P_{acc}.

4.2.1 Characteristic scale tuning

When we simulate a single pseudofermion flavour with filtering, we find that the force terms for the final correction term F_n can become much smaller than that in the degenerate two-flavour pseudofermion case (see e.g. $t = 6, 7, 8$ tRHMC in Figure 4.3), due to the fact that the fermion matrix is no longer squared. This implies that the fermion matrix inversions for the correction term S_n can take place on a much coarser molecular dynamics integration scale. When we apply

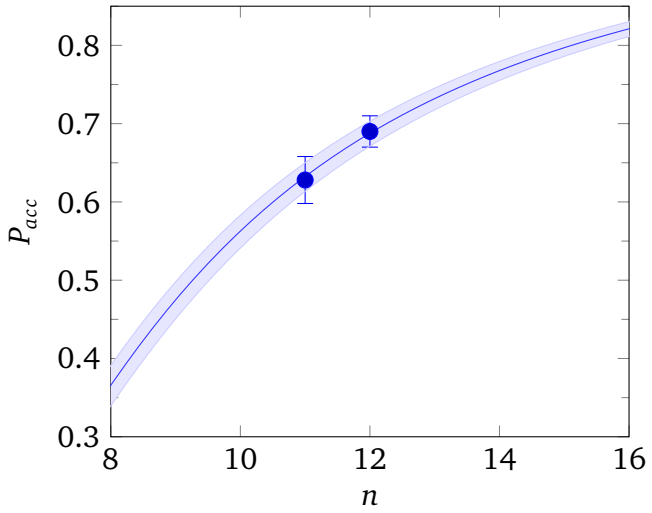

Figure 4.1: Fit of RHMC P_{acc} data on the $16^3 \times 32$ lattice to the complementary error function (4.1). The data is taken from trajectories at the force-balanced point. The fit is shown as a faded band, and has characteristic scale $c = 0.156(4)$.

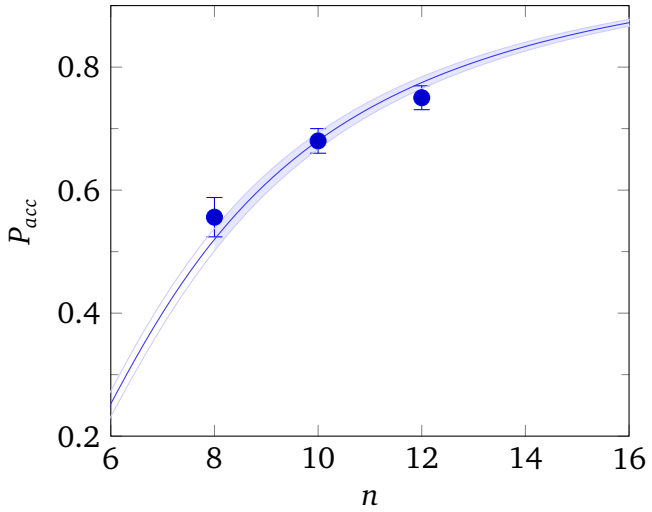

Figure 4.2: Fit of PF-RHMC $p = 4$ P_{acc} data on the $16^3 \times 32$ lattice to the complementary error function (4.1). The data is taken from trajectories at the force-balanced point. The fit is shown as a faded band, and has characteristic scale $c = 0.185(4)$.

force-balancing (3.4), this can be problematic. As the final correction term S_n usually captures most of the dynamics, its force term F_n is quite noisy. When this force term is very small, this leads force-balancing to suggest significantly finer step-sizes for the filter terms than is necessary. This motivates us to try to improve upon force-balancing.

We propose a different step-size tuning technique we denote *characteristic scale tuning*, or *c-scale tuning* for short. This is simply an additional step on top of force balancing. Recalling the force balancing equation (3.4), we drop the h_n term and relax the criteria to

$$F_i h_i \approx \text{constant}, \qquad i = 1, \ldots, n-1, \tag{4.2}$$

allowing the (typically) second-coarsest step-size h_{n-1} to be tuned freely. In practice, we set h_n such that the characteristic scale fit of the acceptance rate (4.1) gives the desired P_{acc}, then we tune h_{n-1} to minimise the cost while remaining within the target acceptance rate range. This extra degree of freedom can help offset an unfavourable force distribution. In the results that follow, we investigate the effects of both tuning methods, and show how much c-scale tuning can improve the performance.

4.3 RHMC

The baseline for comparing our filtering schemes is obviously the cost of a standard RHMC simulation. This has pseudofermion action

$$S_{RHMC}[U, \phi] = \phi^\dagger R(K) \phi. \tag{4.3}$$

In all our simulation results that follow, it is implicitly assumed that there are two copies of the pseudofermion action, one for each degenerate flavour, with independent pseudofermion fields. The rational approximation $R(K)$ we use on our $16^3 \times 32$ lattice is a 20^{th} order Zolotarev approximation (see e.g. [46]) on the interval $[5 \times 10^{-5}, 3]$. The eigenvalue range for this lattice is $[\lambda_{\min}, \lambda_{\max}] = [6.8(1) \times 10^{-5}, 2.18949(5)]$, so this Zolotarev approximation is suitable.

As the gluon step count n_0 has been fixed to 480, there is only one step-size to consider, so tuning is simply a matter of reaching the target acceptance rate range $[0.65, 0.75]$. Using the step counts $(n_1, n_0) = (12, 480)$, we find that the number of matrix operations required per trajectory for pure RHMC is $N_{mat} = 46,960 \pm 120$ with an acceptance rate of $P_{acc} = 0.69(1)$. This gives a normalised cost of

$$C_{RHMC} = 68,100 \pm 1,200, \tag{4.4}$$

setting the benchmark that our filtering methods will have to beat. Note that this is the total cost for *both* pseudofermion flavours. Comparing this with the performance of the double-flavour pseudofermion simulations in chapter 3, we already see an improvement in cost over ordinary HMC, demonstrating the benefits of the n-th root trick (2.119).

4.4 PF-RHMC

For polynomial filtered RHMC (2.126),

$$S_{PF-RHMC} = \phi_1^\dagger P(K)\phi_1 + \phi_2^\dagger P^{-1}(K)R(K)\phi_2,$$

we generate Chebyshev polynomial approximations $P(K)$ to $K^{-1/2}$ using (2.129) on the same range as the rational approximation. This leaves a choice of polynomial order p, so in order to determine which integer value of p is optimal, we sample a broad set of polynomial orders. The forces for all polynomial orders are shown in the middle column of Figure 4.3, and the resultant step-sizes are shown in Table 4.1.

Table 4.1: PF-RHMC configurations on the $16^3 \times 32$ lattice, using force balancing

p	n_2	n_1	n_0	P_{acc}	N_{traj}
4	10	18	480	0.68(1)	2000
10	10	26	480	0.70(1)	2000
16	11	35	480	0.75(1)	2000

The cost function for PF-RHMC with force-balanced points is shown in the middle plot of Figure 4.4. The optimal point is at $p = 4$ with $C = 62,100 \pm 1,100$, which is only 9% cheaper than plain RHMC (4.4). It's surprising that the cost increases as we increase the polynomial order. The main contributor to this is the cost of evaluating the correction term's forces F_2 (empty circles in Figure 4.4), which does not go down as we increase p. The computational cost of each force term F_2 evaluation is relatively constant in p due to the use of a multi-shift solver (see section 2.6.2). While Figure 4.3 shows that the magnitude of F_2 does decrease as p increases, the corresponding step count n_2 required to maintain an acceptance rate in the range $[0.65, 0.75]$ only changes marginally (see Table 4.1).

Looking at the relative error of $P(K)$ in Figure 4.5, we can see that increasing the polynomial order p does not greatly improve the approximation, especially at

small K. This is likely due to the inherent divergence of $K^{-1/2}$ at the origin, which is hard to approximate with polynomials. Hence, this indicates that S_1 does not capture significantly more of the action as we increase p, leaving the brunt of the dynamics and required computational work at the expensive correction term S_2.

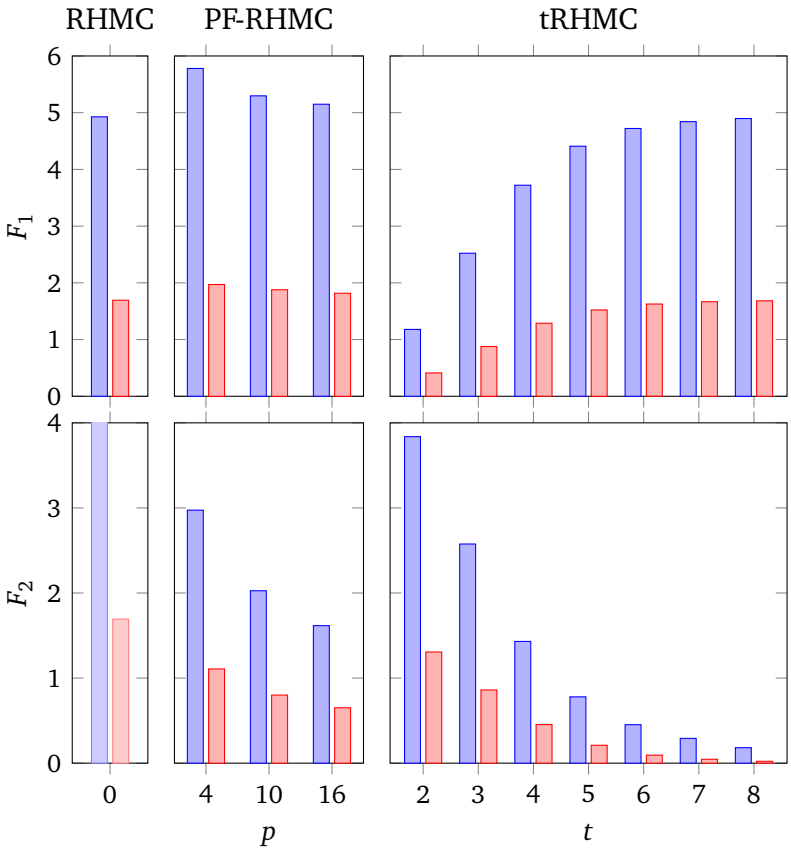

Figure 4.3: Forces for the 1-filter actions on the $16^3 \times 32$ lattice. Left-hand bars show the maximal force, while right-hand bars show the average. The single force term for plain RHMC is included for comparison on both force terms.

The cost for PF-RHMC with c-scale tuning is shown on the left in Figure 4.6, which shows no significant improvement over force-balancing. The resulting step-sizes under c-scale tuning are shown in Table 4.2, and are not very different from those of force balancing (Table 4.1). As discussed earlier, most of the computational effort here is in the expensive correction term S_2, so using c-scale tuning to optimise the filter term's step-size h_1 has little effect.

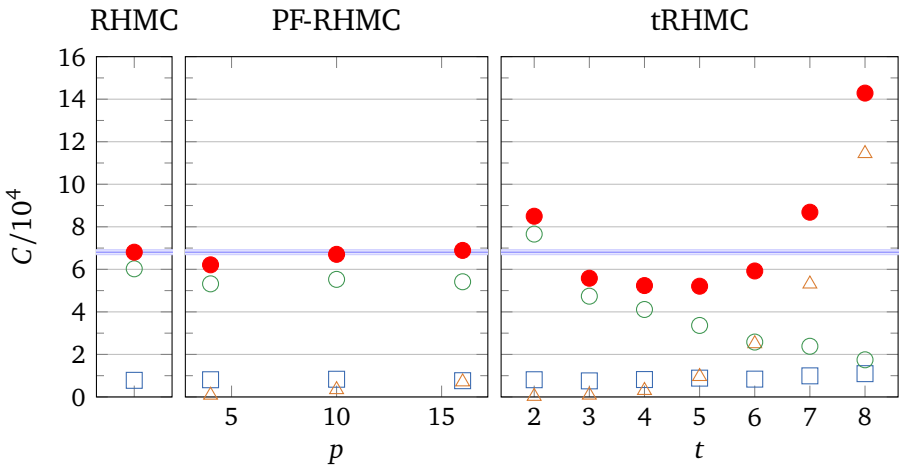

Figure 4.4: Cost function for RHMC and the single-filter actions on the $16^3 \times 32$ lattice, using force balancing. Filled circles are the total cost. Empty squares are the component of the total cost due to action initialisation. For RHMC, empty circles are the cost component due to calculating the force term F. Otherwise, empty triangles are the cost component due to calculating F_1, and empty circles due to calculating F_2. Statistical errors are smaller than the marker size. The faded band is the cost of plain RHMC, included for ease of comparison.

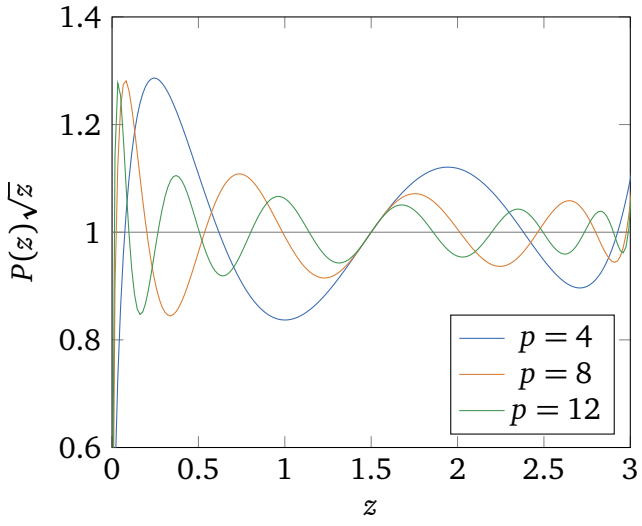

Figure 4.5: $P(z)\sqrt{z}$ for various Chebyshev polynomials with order p, and range $[10^{-5}, 3]$. Closeness to unity indicates how well $P(K)$ approximates $K^{-1/2}$.

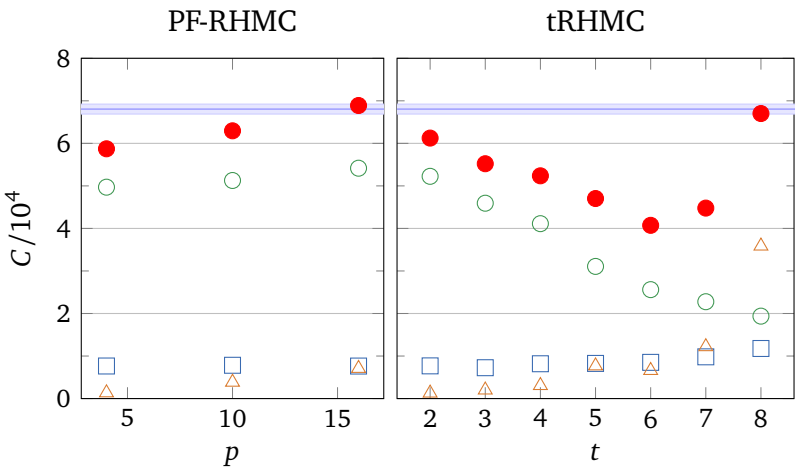

Figure 4.6: C-scale tuned cost for the single-filter actions on the $16^3 \times 32$ lattice. Refer to Figure 4.4 for the legend.

Table 4.2: C-scale tuned configurations for PF-RHMC on the $16^3 \times 32$ lattice

p	n_2	n_1	n_0	P_{acc}	N_{traj}
4	10	30	480	0.71(1)	2000
10	10	30	480	0.72(1)	2000
16	11	35	480	0.75(1)	2000

4.5 tRHMC

For truncated RHMC (2.133),

$$S_{tRHMC} = \phi_1^\dagger R_{0,t}(K)\phi_1 + \phi_2^\dagger R_{t,n}(K)\phi_2,$$

the only novel tunable parameter is the truncation order t. Just as with PF-RHMC, we sample a range of values for t, then tune the step-sizes with force balancing and acceptance rate fitting. The forces are shown in the right-hand column of Figure 4.3, and the resultant configuration points are shown in Table 4.3.

Table 4.3: tRHMC configurations on the $16^3 \times 32$ lattice, using force balancing

t	n_2	n_1	n_0	P_{acc}	N_{traj}
2	15	5	480	0.68(1)	2000
3	10	10	480	0.72(1)	2000
4	8	20	480	0.67(1)	2000
5	6	35	480	0.65(1)	2000
6	5	60	480	0.73(1)	2000
7	4	70	480	0.70(1)	2000
8	3	85	480	0.68(1)	2000

The cost function is shown in the right-hand plot of Figure 4.4, and has a minimum at $t = 5$ with $C = 52,100 \pm 1,000$. This is a more significant improvement than PF-RHMC: it is 24% cheaper than plain RHMC. However, note that the cost increases dramatically if we choose a truncation too low at $t = 2$, or too high at $t = 7, 8$. These filter parameters correspond to unbalanced force term hierarchies (see Figure 4.3) where either the correction term has a larger force than the filter term, or the correction term has a very small force. Thus, it is unsurprising that force-balancing does not perform well here.

The effect of c-scale tuning on tRHMC is shown on the right in Figure 4.6, with the corresponding step-sizes in Table 4.4. Compared to PF-RHMC, the improvement is much more pronounced, particularly at the lower and higher truncation orders where we noted sub-optimal force hierarchies. The overall minimum in cost is significantly better than the best force-balanced tRHMC point $t = 4$ (compare Figure 4.4); the new optimal point is $t = 6$ with cost $C = 40,700 \pm 700$, now 30% cheaper than plain RHMC. Note that, due to tuning P_{acc} via the characteristic scale fit rather than brute force, we are able to use a

larger step-size for the $t = 2$ case under c-scale tuning than with force balancing (compare Table 4.3 with Table 4.4) to achieve a similar P_{acc}.

Table 4.4: C-scale tuned configurations for tRHMC on the $16^3 \times 32$ lattice

t	n_2	n_1	n_0	P_{acc}	N_{traj}
2	11	25	480	0.71(1)	2000
3	10	25	480	0.76(1)	2000
4	8	20	480	0.67(1)	2000
5	6	30	480	0.70(1)	2000
6	5	15	480	0.71(1)	2000
7	4	15	480	0.67(1)	2000
8	3	25	480	0.65(1)	2000

More significantly, we engineer a significant reduction in the cost across a range of truncation values, namely $t = 2, 5, 6, 7, 8$. Consequently, the computational cost under c-scale tuning for $t = 4$–7 is similar, and the cost for $t = 8$ is vastly improved over force balancing. This demonstrates that characteristic scale tuning also helps to reduce filter parameter sensitivity. In practice, this means that using c-scale tuning should reduce the need for re-tuning the filter parameter(s) when generating a new lattice with different physical parameters (such as the quark mass).

Note that (refering to Tables 4.3 and 4.4) the acceptance rate P_{acc} for the c-scale tuned point $t = 5$, 0.70(1), is higher than the force-balanced point, 0.65(1). This is despite the fact that the c-scale tuned point has fewer integration steps ($n_1 = 30$ versus $n_1 = 35$).

This behaviour is most likely due to the use of overlaid integrators, which can have different discretisation error behaviour than nested integrators: see section A.3 for a full discussion. In short, because the c-scale tuned point with step counts $(n_2, n_1) = (6, 30)$ uses step sizes with a small odd factor between them, the resultant discretisation error (A.18) is smaller than what would be expected in the nested case (A.17) and hence the acceptance rate P_{acc} improves. Furthermore, the force-balanced point $(n_2, n_1) = (6, 35)$ has step sizes which are not multiples of each other, and hence has a larger discretisation error than what would be expected in the nested case. These effects combine to give the c-scale tuned point a better P_{acc} than the force-balanced point.

4.6 Two-filter results

We now investigate using multiple filters to see if they will improve the computational performance just as in chapter 3. We also want to measure the effectiveness of c-scale tuning for such lattice actions.

We first consider polynomial filtering. For two polynomial filters (2PF-RHMC), we select a pair of polynomials $P(K), Q(K)$ such that $P(K) \approx K^{-1/2}$ and $Q(K) \approx P(K)^{-1}K^{-1/2}$. This yields the action

$$S_{2PF-RHMC} = \phi_1^\dagger P(K)\phi_1 + \phi_2^\dagger Q(K)\phi_2 + \phi_3^\dagger [P(K)Q(K)]^{-1}R(K)\phi_3, \qquad (4.5)$$

where the correction term S_3 has matrix kernel $\approx I$. The 2PF-RHMC fermion action has two polynomial orders p, q and three step-sizes $\{h_1, h_2, h_3\}$ to tune. We fix $p = 4$, vary q, and then tune the step-sizes with force balancing. This gives the configurations in Table 4.5. Starting from this parameter set, we perform c-scale tuning by varying the second-coarsest step-size h_2 while keeping h_2/h_1 fixed. The resultant configuration choices are given in Table 4.6.

Table 4.5: 2PF-RHMC configurations on the $16^3 \times 32$ lattice, using force balancing

p	q	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	10	7	33	62	480	0.68(1)	2000
	16	8	51	98	480	0.66(1)	2000
	20	9	65	125	480	0.70(1)	2000

Table 4.6: C-scale tuned 2PF-RHMC configurations on the $16^3 \times 32$ lattice

p	q	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	10	7	33	62	480	0.68(1)	2000
	16	8	15	27	480	0.65(1)	2000
	20	9	20	38	480	0.72(1)	2000

For two-level truncation filtering (2tRHMC), we use a pair of truncation orders $t < t'$, giving fermion action

$$S_{2tRHMC} = \phi_1^\dagger R_{0,t}(K)\phi_1 + \phi_2^\dagger R_{t,t'}(K)\phi_2 + \phi_3^\dagger R_{t',n}(K)\phi_3. \qquad (4.6)$$

We set $t = \{4, 5\}$, vary $t' > t$, and apply force balancing and c-scale tuning to the step-sizes. The resultant configuration choices are given in Table 4.7 (force balancing) and Table 4.8 (c-scale tuning).

Table 4.7: 2tRHMC configurations on the $16^3 \times 32$ lattice, using force balancing

t	t'	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	5	6	6	29	480	0.69(1)	2000
	6	5	12	40	480	0.72(1)	2000
	7	4	17	50	480	0.71(1)	2000
	8	3	25	70	480	0.66(1)	2000
5	7	4	8	64	480	0.67(1)	2000
	8	3	10	70	480	0.66(1)	2000
	10	2	22	129	480	0.77(1)	2000

Table 4.8: C-scale tuned 2tRHMC configurations on the $16^3 \times 32$ lattice

t	t'	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	5	6	6	29	480	0.69(1)	2000
	6	5	12	40	480	0.72(1)	2000
	7	4	12	35	480	0.69(1)	2000
	8	3	10	30	480	0.67(1)	2000
5	7	4	6	47	480	0.71(1)	2000
	8	3	5	35	480	0.63(1)	2000
	10	2	5	30	480	0.68(1)	2000

The cost function for 2PF-RHMC and 2tRHMC using force balancing is shown in Figure 4.7, and the corresponding forces are shown in Figure 4.8. Comparing with the one-filter case Figure 4.4, we only see a small additional benefit to using two polynomial filters versus just one, with a minima at $(p,q) = (4,10)$. Looking at the 2tRHMC results, using two truncations with $t = 4$ keeps the cost at a level similar to the best 1tRHMC point for $t' = 5,6,7$. However, as in the single filter case, using force-balancing to set the step sizes is far from optimal for higher polynomial orders, see $q = 16,20$, or at larger truncation orders, see $(t,t') = (4,8)$ or any of the $t = 5$ actions.

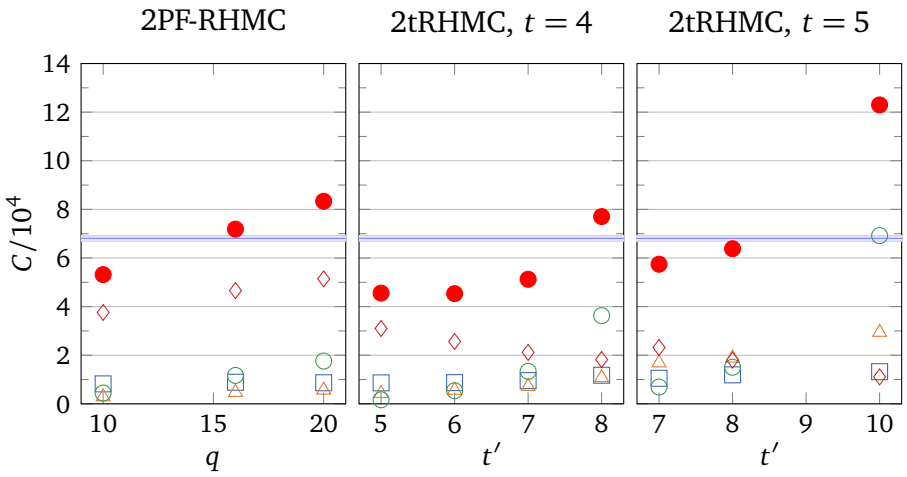

Figure 4.7: The cost function for 2PF-RHMC and 2tRHMC on the $16^3 \times 32$ lattice using force balancing. Filled circles are the total cost. Empty squares are the component of the total cost due to action initialisation, empty triangles due to calculating F_1, empty circles due to calculating F_2, and empty diamonds due to calculating F_3. Statistical errors are smaller than the marker size. The faded band is the cost of plain RHMC, included for ease of comparison.

Next we consider c-scale tuning for the two-filter actions, with cost functions shown in Figure 4.9. Comparing this with the force balanced case (Figure 4.7), 2PF-RHMC gets some improvement in the case of higher order polynomials $q = 16,20$, but overall there is only a relatively small benefit. The optimal point $(p,q) = (4,10)$ has a cost $C = 53,200 \pm 900$, yielding a 22% improvement over plain RHMC.

The main contributor to the 2PF-RHMC cost is, as in the single-filter case, the cost of evaluating the correction term F_3 (empty diamonds in Figure 4.9), which is symptomatic of an increasing step count n_3 (see Table 4.6). Looking at the relative error of $Q(K)$ in approximating $P^{-1}(K)z^{-1/2}$ in Figure 4.10, we find that, just as in the single filter case, increasing the polynomial order q does not

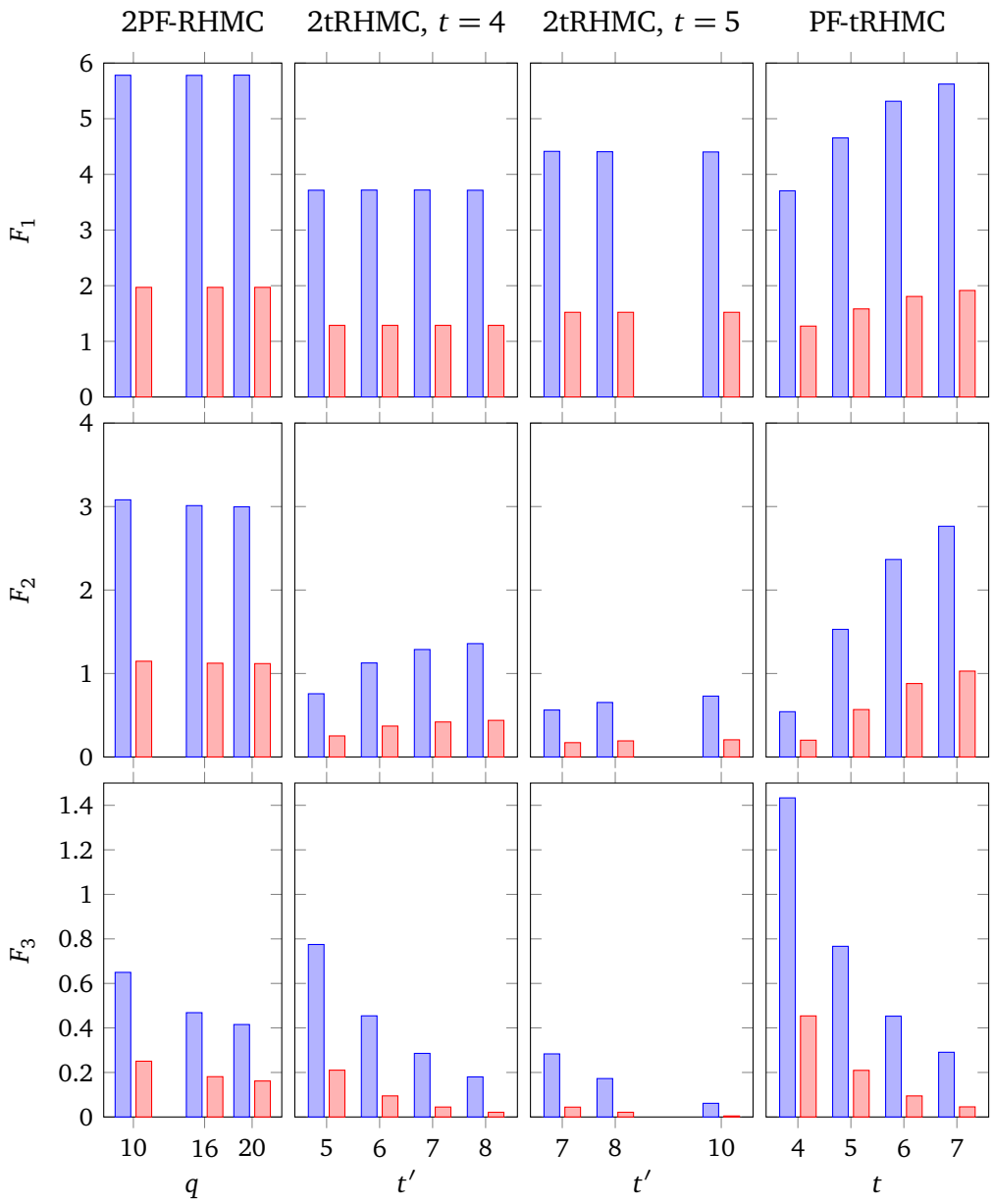

Figure 4.8: Forces for the 2-filter actions on the $16^3 \times 32$ lattice. Left-hand bars show the maximal force, while right-hand bars show the average.

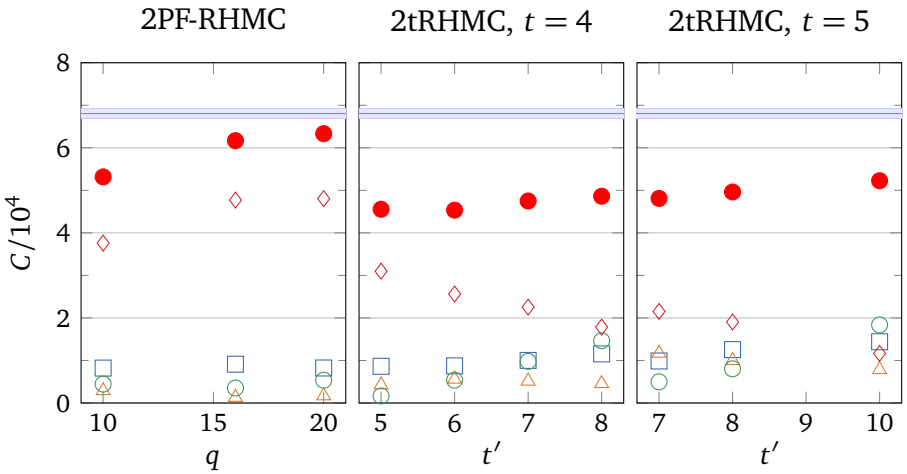

Figure 4.9: C-scale tuned cost for 2PF-RHMC and 2tRHMC on the $16^3 \times 32$ lattice. Refer to Figure 4.7 for the legend.

greatly improve the approximation. This behaviour shows that using a Chebyshev polynomial approximation for PF-RHMC is ineffective. A different class of polynomials could have better approximation behaviour at lower values, but we do not investigate this here.

The optimal point for 2tRHMC with c-scale tuning is at $(t, t') = (4, 6)$ with cost $C = 45,300 \pm 700$, a 32% improvement over plain RHMC. The optimal cost is only marginally improved over the force balanced results, and is actually slightly worse than that in the single truncation filter case. However, note that the cost function under c-scale tuning is now consistent over all the filter choices shown. This behaviour demonstrates that c-scale tuning reduces the sensitivity of the cost to the truncation order in the case of multiple tRHMC filters. In practice, this means that a set of reasonable truncation parameters (t, t') for a particular order rational approximation should work near-optimally across a variety of lattices.

4.6.1 Combining the filters: PFtRHMC

It is also possible to combine the two filtering techniques we consider here, applying both a polynomial filter and a truncation filter. For our PF-tRHMC tests, we place a $p = 4$ polynomial filter on top of the tRHMC action,

$$S_{PF-tRHMC} = \phi_1^\dagger P(K)\phi_1 + \phi_2^\dagger P(K)^{-1} R_{0,t}(K)\phi_2 + \phi_3^\dagger R_{t,n}(K)\phi_3. \tag{4.7}$$

This form suggests using a polynomial $P(K)$ that approximates $R_{0,t}(K)^{-1}$. As

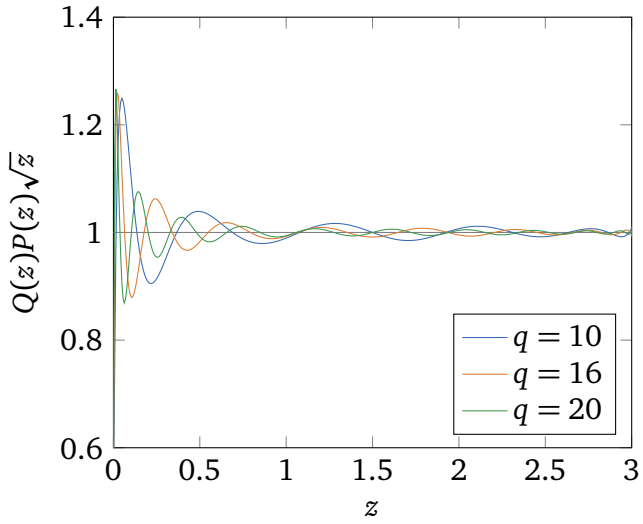

Figure 4.10: $Q(z)P(z)\sqrt{z}$ for various Chebyshev polynomials with orders q and $p = 4$, and range $[10^{-5}, 3]$. Closeness to unity indicates how well the intermediate filter $Q(K)$ approximates $P^{-1}(K)K^{-1/2}$.

before, we make use of a Chebyshev approximation. The configuration choices are shown in Tables 4.9 and 4.10 for force balancing and c-scale tuning respectively, the corresponding forces on the right-hand side in Figure 4.8, and the cost function in Figure 4.11.

Table 4.9: PF-tRHMC configurations on the $16^3 \times 32$ lattice, using force balancing

p	t	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	4	15	6	39	480	0.82(1)	2000
	5	6	12	36	480	0.75(1)	2000
	6	5	26	59	480	0.75(1)	2000
	7	4	38	79	480	0.69(1)	2000

PF-tRHMC with force balancing is very sensitive to the choice of truncation order t, with inflated cost at $t = 4$ and $t = 7$. This is appears to be due to a rapidly varying distribution of forces between the polynomial and truncated terms in the action (see Figure 4.8).

When we apply c-scale tuning, PF-tRHMC performs as well as (or slightly better than) 2tRHMC, with a relatively consistent cost improvement over the range of filters tested $t = 4, 5, 6, 7$. The optimal filter choice is $(p, t) = (4, 5)$ with cost $C = 42,600 \pm 700$, which is a small improvement on the optimal 2tRHMC

Table 4.10: C-scale tuned PF-tRHMC configurations on the $16^3 \times 32$ lattice

p	t	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	4	8	8	42	480	0.73(1)	2000
	5	6	10	30	480	0.69(1)	2000
	6	5	12	27	480	0.69(1)	2000
	7	4	16	32	480	0.70(1)	2000

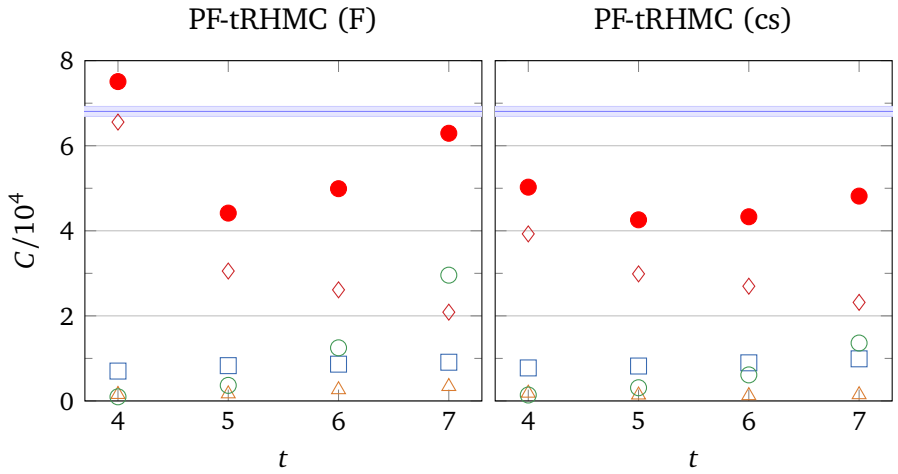

Figure 4.11: The cost function for PF-tRHMC on the $16^3 \times 32$ lattice. The left hand plot shows the force-balanced data, while the right hand plot shows the c-scale tuned data. Refer to Figure 4.7 for the legend.

result. The superior behaviour of PF-tRHMC, compared with 2PF-RHMC, is due to the improved performance of the polynomial filter correction term (empty circles in Figure 4.11). This occurs because the polynomial filter is small ($p = 4$) and does not have to approximate the full inverse square-root. However, we note that 2tRHMC provides a similar benefit whilst being simpler to implement because it only has one type of filtering.

4.7 Larger lattice tests

4.7.1 Setup

In order to determine whether the single-flavour improvement techniques described here scale to more physical lattices, we also run comparison tests on a $24^3 \times 48$ lattice with two degenerate single-flavour clover-improved fermions,

pion mass $m_\pi \approx 300$ MeV and lattice spacing ~ 0.07 fm [54]. Based on the relative merits of the $16^3 \times 32$ results, we do not consider polynomial filtering on the larger volume, and only compare plain RHMC, tRHMC and 2tRHMC. The cost of simulating the $24^3 \times 48$ lattice is much larger than that for the $16^3 \times 32$ lattice, so we only have 100 trajectories per filter set. Nonetheless, these statistics are sufficient to compare the different filtering methods to the baseline result.

For the rational approximation on this lattice, we use the 30th order Zolotarev approximation on the range $[10^{-6}, 3]$. When we measure the eigenvalue spectrum of the fermion matrix, we find that the distribution extends slightly outside this range: $[\lambda_{\text{min}}, \lambda_{\text{max}}] = [3.3(1) \times 10^{-6}, 3.078(4)]$. Nonetheless, the rational function is still valid as the error of the Zolotarev approximation remains within the desired tolerance well above the upper bound for this range.

Note that using clover-improved fermions under even-odd preconditioning adds an extra term to the fermion action, namely the determinant term $S_{\text{det}} = -2\,\text{Tr}(\ln(D_{ee}))$, which must be placed on an integration scale (see section 2.5.1). This term is relatively cheap to calculate, so for this investigation it is always integrated on the second-finest integration scale, i.e. n_1.

4.7.2 Results

Using $n_1 = 35$ steps for the pseudofermion and determinant terms, and $n_0 = 480$ for the gauge term, plain RHMC for the $24^3 \times 48$ lattice takes $N_{\text{mat}} = 628,000 \pm 4,000$ matrix operations per trajectory with an acceptance rate of $P_{acc} = 0.65(4)$. This gives a baseline cost of

$$C = 971,000 \pm 65,000 \tag{4.8}$$

for comparison.

The cost functions for tRHMC and 2tRHMC (with $t = 4$) using c-scale tuning are shown in Figure 4.12. The baseline RHMC point is shown as a faded band. The corresponding forces are shown in Figure 4.13, and the configuration data in Tables 4.11 and 4.12.

We find that the results with c-scale tuning are qualitatively similar to the $16^3 \times 32$ lattice results. Once again we see that tRHMC provides the best result, with the minimum cost at $t = 10$ with $C = 412,000 \pm 27,000$, an impressive 58% improvement over plain RHMC. As was the case for the $16^3 \times 32$ lattice, we also see that using two truncation filters does not improve the minimum cost – the optimal cost for 2tRHMC with $t = 4$ is at $t' = 8$, with $C = 451,000 \pm 32,000$. Most importantly, we see that the cost is relatively flat across all filter

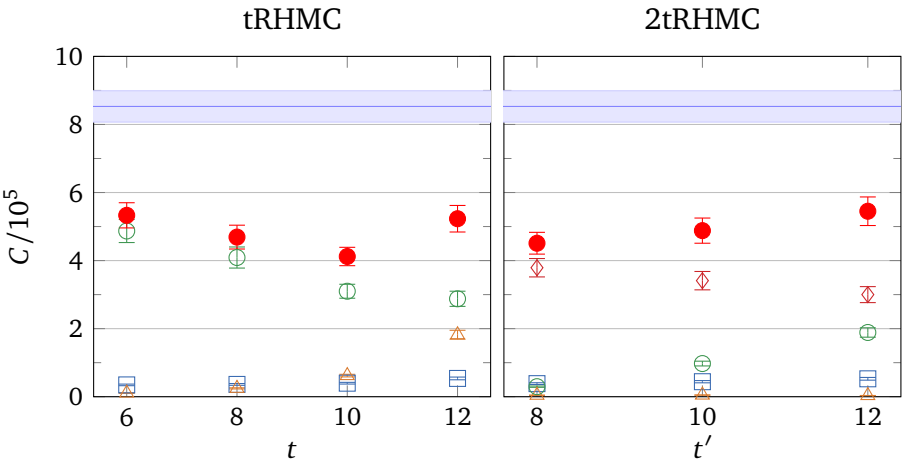

Figure 4.12: C-scale tuned cost of tRHMC and 2tRHMC on the $24^3 \times 48$ lattice. Filled circles are the total cost. Empty squares are the component of the total cost due to action initialisation, empty triangles due to calculating F_1, empty circles due to calculating F_2, and empty diamonds due to calculating F_3. The faded band is the cost of plain RHMC, included for ease of comparison.

Table 4.11: C-scale tuned configurations for tRHMC on the $24^3 \times 48$ lattice

t	n_2	n_1	n_0	P_{acc}	N_{traj}
6	20	30	480	0.71(4)	100
8	16	25	480	0.69(5)	100
10	12	25	480	0.73(4)	100
12	10	25	480	0.66(5)	100

Table 4.12: C-scale tuned configurations for 2tRHMC on the $24^3 \times 48$ lattice

t	t'	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
4	8	15	30	32	480	0.70(4)	100
	10	12	35	34	480	0.66(5)	100
	12	11	25	23	480	0.65(5)	100

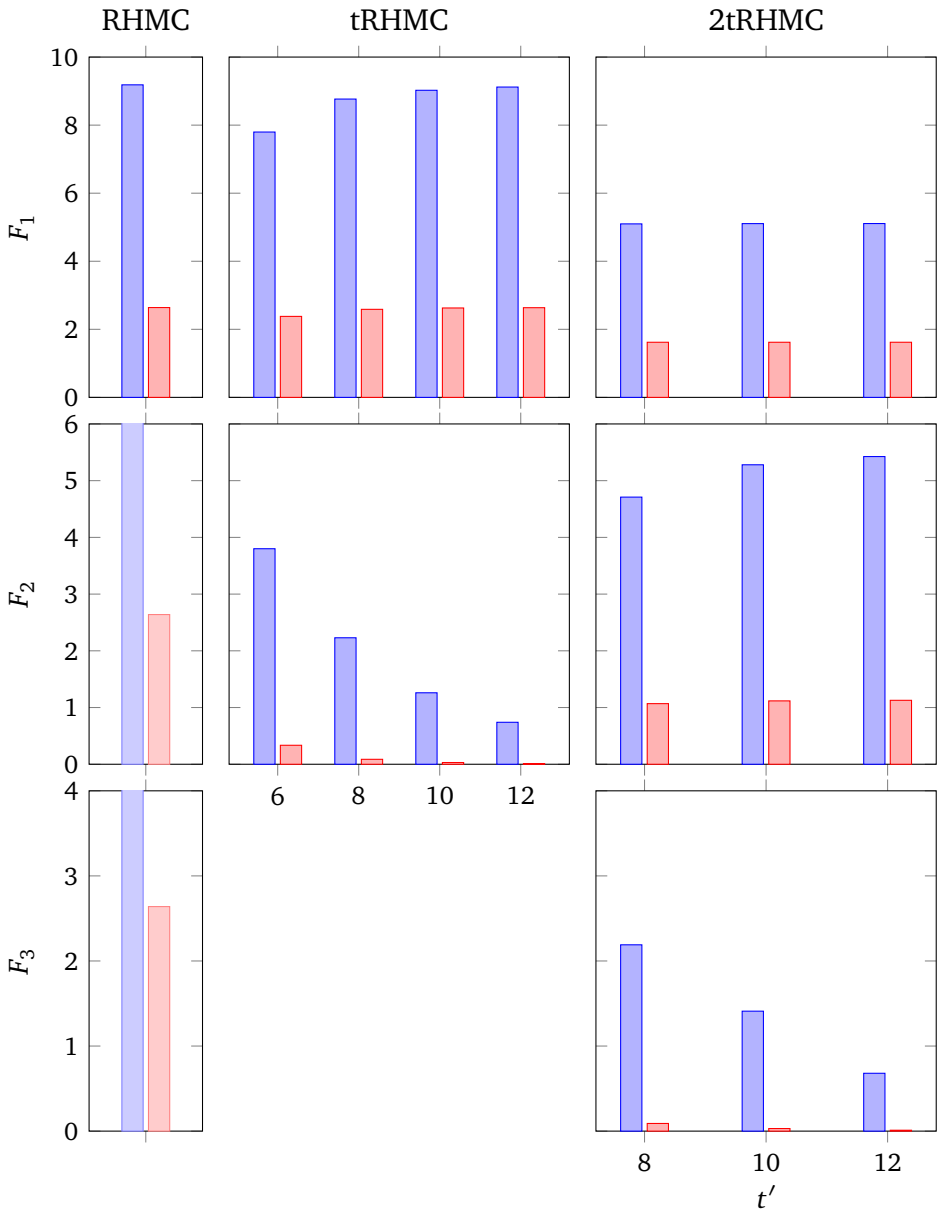

Figure 4.13: Forces for RHMC, tRHMC and 2tRHMC on the $24^3 \times 48$ lattice. Left-hand bars show the maximal force, while right-hand bars show the average. The single force term for plain RHMC is included for comparison on all force terms.

parameters shown, demonstrating again that c-scale tuning reduces the need for filter parameter tuning.

4.8 Concluding remarks

We have studied two types of filtering that can be applied to RHMC: polynomial filtering (PF-RHMC) and truncated ordered product filtering (tRHMC). Comparing tRHMC and PF-RHMC for single-flavour pseudofermion simulations on a $16^3 \times 32$ lattice, we find that tRHMC performs better overall, providing a 40% improvement in cost over the plain RHMC case.

We tested both one and two levels of filtering, finding that adding a second filter did not improve the performance of tRHMC. When testing tRHMC on a larger $24^3 \times 48$ lattice at a lighter quark mass we find a similar improvement: tRHMC reduces the cost significantly, but two truncations are not better than one. A possible explanation for this is that, by simulating a single flavour only (rather than two as a single pseudofermion), there is already a significant reduction in the force variance [55], and hence there is less work for the filter to do.

The superior benefits of tRHMC over PF-RHMC are possibly due to the Chebyshev polynomial filter being a relatively poor approximation to the high-energy modes of the single-flavour system. A different polynomial approximation might provide better performance, but given the simplicity of implementing tRHMC we did not consider investigating this here. The combined PF-tRHMC filtering method performs as well as tRHMC, but again this combination is more complex to implement than tRHMC.

An important consideration for the filtering methods investigated here is choosing the filter parameters and integration step-sizes. The most common way to tune the integration step-sizes is to choose step-sizes h_i such that the molecular dynamics forces F_i satisfy $F_i h_i \simeq$ constant. However, when using this force balancing method to tune the step-sizes, we found that the cost function is highly dependent on the choice of filter parameters – choose these too low or too high, and the performance compared with ordinary RHMC is only marginally better or even worse. This is partially due to small numbers of integration steps at the coarsest scale.

To mitigate the filter parameter dependence, we introduced a novel way of tuning the integration step-sizes, which we refer to as c-scale tuning. This technique uses the characteristic scale to set the coarsest step size, then holds that fixed while tuning the next step size, with the remaining step sizes set using the

force balancing method. For both filtering methods and both lattices, we found that while employing this technique did not improve the lowest achievable cost, it did make the cost function significantly less sensitive to the filter parameters.

The advantage of this is particularly relevant for larger and more physical lattices, where generating configurations is so expensive that we cannot tune the filter parameters by brute-force within a reasonable time-frame. When we use multiple filters, which is usually necessary for such lattices, then the benefits of c-scale tuning should prove significant. One can just choose a reasonable set of filters, calculate the forces, then tune two step-sizes to achieve near-optimal costs.

Implementing tRHMC and c-scale tuning is straightforward. In particular, tRHMC only requires a small modification to regular RHMC code to allow for multiple rational polynomial terms. Applying c-scale tuning only requires information about the force terms in a HMC simulation. Thus, these techniques can be quickly deployed in order to reduce the computational cost in simulating single-flavour pseudofermions on the lattice. Here, the lightest quark mass we considered was $m_\pi \approx 300$ MeV. At lighter masses, the relative benefits of single-flavour tRHMC filtering will increase. This should prove particularly useful for dynamical Lattice QCD+QED configurations, where the up and the down quark must be simulated separately: this is the subject of the next investigation in chapter 5.

∘ 5 ∘

Electromagnetic effects

We now discuss how to dynamically incorporate electromagnetic effects into Lattice QCD simulations, which is denoted *Lattice QCD+QED*. These effects are important for many physical quantities of interest, such as the neutron-proton mass difference, and the current precision of Lattice QCD simulations is such that even $\sim 5\%$ corrections from EM make a noticeable difference.

5.1 Quantum Electrodynamics

First, we consider pure electrodynamics for a system of quarks, which is governed by the theory of *Quantum Electrodynamics* (QED). The QED action looks very similar to that of QCD in the continuum (1.22): in Euclidean space-time, it takes the form

$$S_{QED} = \overline{\psi}(\gamma_\mu D_\mu^{(QED)} + m)\psi + \frac{1}{4}F_{\mu\nu}^{(QED)}F_{\mu\nu}^{(QED)}. \tag{5.1}$$

The vital difference here is that all of the gauge objects are based on the Abelian Lie group U(1), rather than SU(3) as for QCD.

The Lie group U(1) is the unitary group of dimension one, which is usually represented as a complex phase: $g \in \mathrm{U}(1)$ if

$$g = e^{i\chi}, \chi \in \mathbb{R}. \tag{5.2}$$

Thus, we have just one kind of U(1) charge, the *electric charge*. The quark fields $\psi, \overline{\psi}$ are elements of the fundamental representation of U(1), which means that under local U(1) gauge transformations, we have

$$\psi(x) \to \psi'(x) = e^{i\chi(x)}\psi(x) \tag{5.3a}$$

$$\text{and} \quad \overline{\psi}(x) \to \overline{\psi}'(x) = e^{-i\chi(x)}\overline{\psi}(x), \tag{5.3b}$$

where $\chi(x)$ is a real-valued function of x.

The covariant derivative is defined in terms of the U(1) gauge field $B_\mu(x)$

$$D_\mu^{(QED)}(x) = \partial_\mu + iQ_f B_\mu(x). \tag{5.4}$$

The parameter Q_f here is the charge of the quark f in question, so for the six physical quarks we have $Q_u = Q_c = Q_t = \frac{2}{3}e$ and $Q_d = Q_s = Q_b = -\frac{1}{3}e$ where e is the magnitude of the charge on the electron. The gauge field $B_\mu(x)$ is an element of the adjoint representation of U(1), namely the Lie algebra $\mathfrak{u}(1)$. U(1) only has one generator, $t = 1$, so the adjoint set is simply the set of all real numbers. Hence, $B_\mu(x)$ is just a real-valued function of x with Lorentz index μ. This field represents the *photon*.

Just as in QCD, it is convenient to absorb the main factor of the gauge coupling e into the definition of the fields, so we use the action

$$S_{QED} = \overline{\psi}(\gamma_\mu D_\mu^{(QED)} + m)\psi + \frac{1}{4e^2} F_{\mu\nu}^{(QED)} F_{\mu\nu}^{(QED)} \tag{5.5}$$

with covariant derivative

$$D_\mu^{(QED)}(x) = \partial_\mu + iq_f B_\mu(x). \tag{5.6}$$

where $q_f = Q_f/e$ is a real number that quantifies the relative strength of the coupling: $2/3$ for the up-type quarks $\{u, c, t\}$; $-1/3$ for the down-type quarks $\{d, s, b\}$.

In contrast to SU(3), U(1) is Abelian: multiplication of group elements commute. Hence, the electromagnetic field strength tensor takes the form

$$F_{\mu\nu}^{(QED)} = -i[D_\mu^{(QED)}(x), D_\nu^{(QED)}(x)] = \partial_\mu B_\nu(x) - \partial_\nu B_\mu(x) \tag{5.7}$$

and we do not have self-interactions of photons. Because of this, the gauge coupling of QED e remains relatively constant across energy scales. The coupling is often expressed by the fine-structure constant $\alpha = e^2/4\pi \approx \frac{1}{137}$. This coupling is small, so, unlike QCD, QED can be readily examined using perturbation theory.

5.2 Lattice QCD+QED

We want to incorporate electromagnetic effects into our lattice calculations. If we combine the QED action (5.5) with the QCD action (1.22), we obtain

$$S_{QCD+QED} = S_G + S_A + S_F^{QCD+QED} \tag{5.8}$$

where S_G is the SU(3)/gluon gauge action

$$S_G = \frac{1}{2g^2}\,\mathrm{Tr}\big[F_{\mu\nu}(x)F_{\mu\nu}(x)\big], \tag{5.9}$$

S_A is the U(1)/photon gauge action

$$S_A = \frac{1}{4e^2}F^{(QED)}_{\mu\nu}(x)F^{(QED)}_{\mu\nu}(x), \tag{5.10}$$

and $S_F^{QCD+QED}$ is the combined QCD+QED fermion action

$$S_F^{QCD+QED} = \overline{\psi}(\gamma_\mu D^*_\mu + m)\psi \tag{5.11}$$

where the combined covariant derivative is

$$D^*_\mu = \partial_\mu + iA_\mu(x) + iq_f B_\mu(x). \tag{5.12}$$

In order to discretise this action onto a lattice, we must decide upon the essential gauge field for the QED interaction from which we construct all other QED operators. As the U(1) theory is Abelian, we have two choices:

1. A *compact* representation: use the U(1) fundamental group gauge fields, $U'_\mu(x) = \exp\big(iq_f B_\mu(x)\big)$, or

2. A *non-compact* representation: use the $\mathfrak{u}(1)$ algebra gauge fields, $B_\mu(x)$.

As QED has no self-interactions, it turns out that the non-compact representation is the most useful. With this decision, we are now in a position to formulate lattice operators for the new QED parts of the action.

First, consider the discretisation of S_A. Expanding the electromagnetic field-strength tensor gives

$$S_A = \frac{1}{4e^2}\int d^4x \sum_{\mu,\nu}\big(\partial_\mu B_\nu(x) - \partial_\nu B_\mu(x)\big)^2. \tag{5.13}$$

This can be discretised by replacing the derivative $\partial_\mu B_\nu(x)$ with the forward difference $[B_\nu(x+a\hat{\mu}) - B_\nu(x)]/a$, and replacing the integral with a sum. This leads to the lattice photon action

$$S_A = \frac{a^2}{4e^2}\sum_{x\in\Lambda}\sum_{\mu,\nu}\big(B_\mu(x) + B_\nu(x+a\hat{\mu}) - B_\mu(x+a\,\hat{\nu}) - B_\nu(x)\big)^2. \tag{5.14}$$

This is accurate to order $\mathcal{O}(a^2)$. This is comparable to the Wilson gauge action (1.48), where instead of the plaquette $U_{\mu\nu}$ we have the sum $B_\mu(x)+B_\nu(x+a\hat{\mu})-B_\mu(x+a\hat{\nu})-B_\nu(x)$. If better discretisation errors are required, we can apply the Symanzik improvement scheme just as in the QCD case.

Next, consider the combined fermion action S_F. The only difference from the QCD action is that the covariant derivative has the extra term $iq_f B_\mu(x)$. Considering the derivation of a lattice fermion action in section 1.2, we simply need to replace all instances of the gluon gauge field $U_\mu(x)$ with $e^{iq_f B_\mu(x)}U_\mu(x)$. Therefore, the Wilson lattice action with QED effects is

$$S_F^{\text{Wilson+}QED} = a^4 \sum_f \sum_{x,y\in\Lambda} \overline{\psi}^f(x)\left[\delta_{x,y} - \kappa_f H^f(x|y)\right]\psi^f(y) \tag{5.15}$$

where the hopping matrix is now given by

$$\begin{aligned} H^f_{ab\alpha\beta}(x|y) = &\sum_{\mu=1}^{4}(1-\gamma_\mu)_{\alpha\beta}e^{iq_f B_\mu(x)}U_\mu(x)_{ab}\delta_{x+a\hat{\mu},y} \\ &+\sum_{\mu=1}^{4}(1+\gamma_\mu)_{\alpha\beta}e^{-iq_f B_\mu(x)}U_{-\mu}(x)_{ab}\delta_{x-a\hat{\mu},y}. \end{aligned} \tag{5.16}$$

The QED component of this action is accurate to $\mathcal{O}(a\alpha)$, which is usually sufficient for typical Lattice QCD+QED simulations.

5.2.1 Other points to consider

In Lattice QCD+QED, the electromagnetic gauge needs to be fixed in order for physical quantities to be measured. This is not a consideration in Lattice QCD; confinement ensures that all observable particles are flavour singlets and thus QCD gauge-invariant, so expectation values are always well-defined and computable without gauge-fixing. In contrast, we have several electrically charged particles in Lattice QCD+QED (e.g. the π^+), which are inherently QED gauge-variant. One popular choice of gauge for Lattice QCD+QED simulations is the Landau gauge, given by the condition

$$\sum_\mu \partial_\mu B_\mu(x) = 0. \tag{5.17}$$

Another potential choice is the Coulomb gauge, given by

$$\sum_{i=1,2,3} \partial_i B_i(x) = 0. \tag{5.18}$$

These can be applied on the lattice by numerical optimisation of a gauge fixing function, based on the idea that at the local minima and maxima of a function f, we have $\partial f = 0$.

The photon field $B_\mu(t, \vec{x})$ can have a non-physical zero-momentum mode

$$\tilde{B}_\mu(t, \vec{k} = 0) = \sum_{\vec{x}} B_\mu(t, \vec{x}), \tag{5.19}$$

which causes an IR divergence in the continuum. This makes extrapolating quantities to the physical point impossible. The cause of this phenomenon is that the classical equations of motion for QED are inconsistent with a single charged particle in a periodic box. To fix this issue, we can remove the zero-mode explicitly by setting

$$\tilde{B}_\mu(t, \vec{k} = 0) = \sum_{\vec{x}} B_\mu(t, \vec{x}) = 0, \quad \forall t, \mu. \tag{5.20}$$

While this constraint violates the hypercubic symmetry of the lattice action, the symmetry is restored in the continuum. See [56] for a detailed discussion of this zero-mode removal.

In practice, the smallness of the electromagnetic coupling $\alpha \sim \frac{1}{137}$ compared to the gluon gauge coupling $\alpha_s \sim 1$ makes measuring electromagnetic contributions to expectation values difficult, because they are often drowned out by the statistical noise of the QCD contribution. To alleviate this, we can simulate with non-physical large EM couplings α to boost the signal, then interpolate to the physical value by also measuring quantities on plain Lattice QCD configurations where $\alpha = 0$ [57, 58].

Finite volume effects for Lattice QCD+QED simulations are much more pronounced than those in pure Lattice QCD. This is because the electromagnetic force is a long-range interaction, so, unlike QCD, the strength of the electromagnetic force does not quickly drop off as we move to the boundaries. Therefore, the finite-volume effects should always be considered when using Lattice QCD+QED simulations for continuum extrapolations.

5.3 Benchmarking tRHMC

In Lattice QCD+QED, the up and down quarks have different charges which necessitates using single-flavour pseudofermions to simulate them. Given the success of filtering methods for single-flavour pseudofermions in chapter 4, we now try to determine whether we can achieve a similar improvement when generating Lattice QCD+QED configurations.

For this benchmarking, we use a lattice from the QCDSF collaboration near the flavour-symmetric point [58]. This is a $24^3 \times 48$ lattice with a tree-level improved gluon action (1.55), and $n_f = 1+1+1$ SLiNC fermions [59] with hopping parameters $\kappa_u = 0.124362, \kappa_d = \kappa_s = 0.121713$ and quark charges $q_u = 2/3\ e, q_d = q_s = -1/3\, e$. These parameters are chosen such that, after taking into account the electromagnetic effects, the masses of the quarks are about the same. Electromagnetic effects are implemented via the addition of a non-compact photon action (5.14) and the modification of the Wilson part of the fermion action (5.15), using the Landau gauge (5.17). The electromagnetic coupling is set unphysically large to $\alpha \sim 0.1$. The pion mass on this lattice is $m_\pi \approx 330$ MeV, and the lattice spacing is $a = 0.068$ fm. For further details of this lattice, refer to the original literature [58].

Given the performance of tRHMC for single-flavour pseudofermions (chapter 4), we only compare RHMC (2.117) and tRHMC (2.133), using the same cost function as before (3.1):

$$C = N_{mat}/P_{acc}.$$

Here, we use a 30th order Zolotarev approximation on the range $[10^{-6}, 5]$. For each choice of filter parameters, we perform 100 trajectories using BQCD [48] and apply c-scale tuning in order to see if we can provide an improvement in simulation cost.

The forces terms for RHMC and various tRHMC filters are shown in Figure 5.1. Note that the force terms for the up quark and the down-type quarks $\{d, s\}$ are similar, demonstrating that the quark masses are also similar.

The choice of step-sizes under force balancing are shown in Table 5.1, while the c-scale tuned points are shown in Table 5.2. Here, we use the same truncation order t for each quark, with corresponding step-sizes $\{h_2, h_3\}$ for the filter and correction term respectively. As in the previous investigations, the Wilson gauge action is very cheap to calculate, so we fix its step-size to $h_0 = 1/480$. Meanwhile, the tree-level improvement to the gauge action and the clover determinant term are a little more expensive to simulate, so we put them on a coarser scale h_1 which is fixed to $1/80$.

The cost function for RHMC and tRHMC is shown in Figure 5.2, with both force balancing and c-scale tuning. The cost of RHMC is plotted as a faded band, and takes the value

$$C_{RHMC} = 923,000 \pm 62,000. \tag{5.21}$$

Under force-balancing, the optimal cost with tRHMC is at $t = 8$ with $C = 711,000 \pm 45,000$, which is a 23% improvement over RHMC. Hence, tRHMC is effective for reducing the cost of generating QCD+QED configurations. We

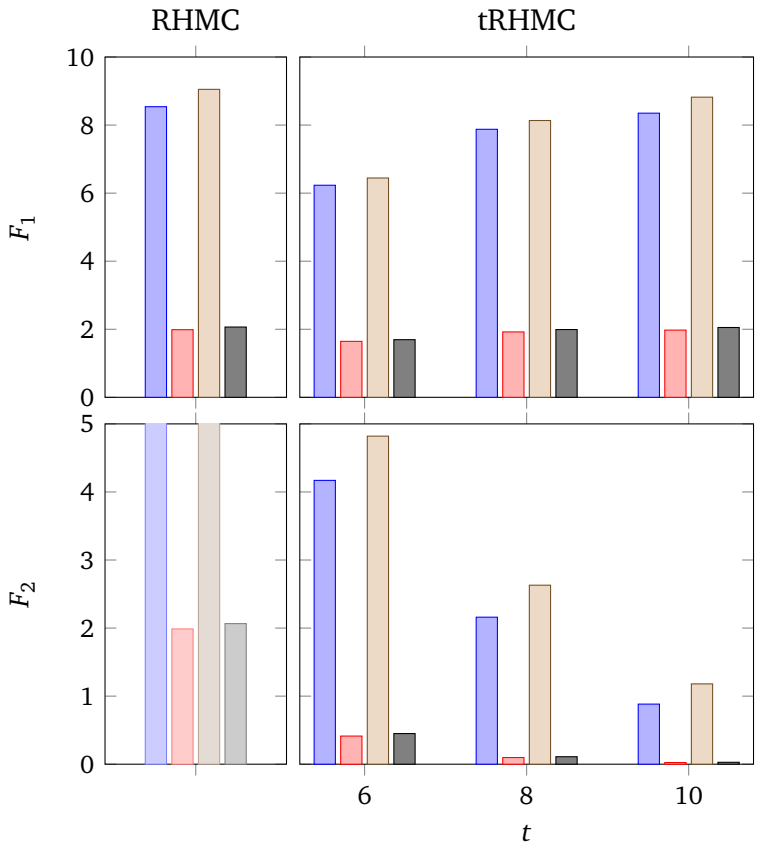

Figure 5.1: Forces for RHMC and tRHMC on the $24^3 \times 48$ lattice with QED. For each filter set, the first two bars represent the maximal and average forces for the down-type quarks $\{d, s\}$, while the second two bars represent the maximal and average forces for the up quark u. Note that the vertical scale differs between F_1 and F_2. The single fermion force term F for RHMC is included for comparison on both force terms F_1, F_2 (left-hand plots).

Table 5.1: tRHMC configurations on the $24^3 \times 48$ lattice with QED, using force balancing. The configuration for plain RHMC is included as '$t = 0$'.

t	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
0		30	80	480	0.72(4)	100
6	16	40	80	480	0.72(4)	100
8	20	80	80	480	0.74(4)	100
10	26	145	80	480	0.74(4)	100

Table 5.2: tRHMC configurations on the $24^3 \times 48$ lattice with QED, using c-scale tuning. The configuration for plain RHMC is included as '$t = 0$'.

t	n_3	n_2	n_1	n_0	P_{acc}	N_{traj}
0		30	80	480	0.72(4)	100
6	16	40	80	480	0.72(4)	100
8	20	80	80	480	0.74(4)	100
10	26	40	80	480	0.71(4)	100

note however that the cost is highly dependent on the truncation order, with far less optimal costs for $t = 6$ and $t = 10$. Looking at the force terms involved (Figure 5.1), this is due to a rapidly varying force distribution between the filter and correction terms.

The situation is improved with c-scale tuning: while the only improvement we see is at $t = 10$, it is quite substantial, producing a reduced cost of $C = 679,000 \pm 47,000$. This behaviour demonstrates that our c-scale tuning is still effective at reducing filter parameter sensitivity.

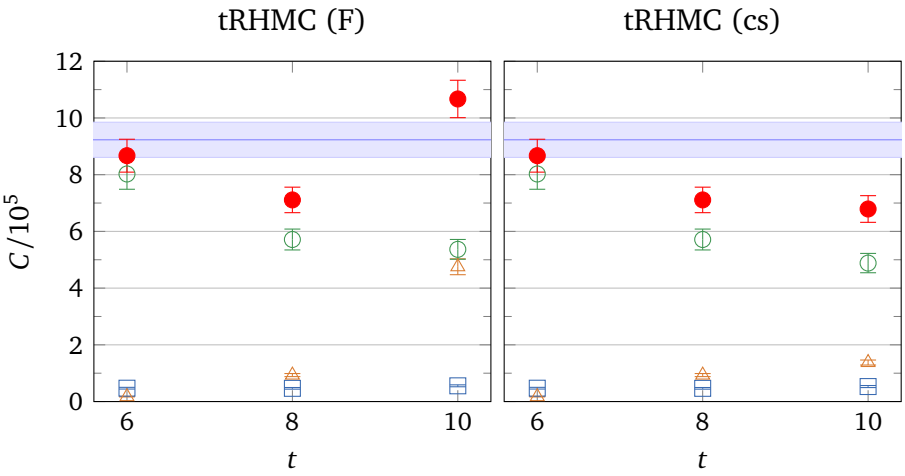

Figure 5.2: Cost function for tRHMC on the $24^3 \times 48$ lattice with QED. The left-hand plot shows the force-balanced data, while the right hand plot shows the c-scale tuned data. Filled circles are the total cost. Empty squares are the component of the total cost due to action initialisation, empty triangles due to calculating F_1 for all three fermions, empty circles due to calculating F_2 for all three fermions. The faded band is the cost of plain RHMC, included for ease of comparison.

Conclusion

Lattice QCD simulations have come a long way from their inception, with current lattices reaching physical masses and continuum extrapolations with errors as low as 1%. This achievement is partially due to improvements in computational capabilities, but it is also due to significant improvements in simulation techniques. In this work, we have mainly focussed on a particular class of improvement techniques known as filtering methods, which work by splitting the fermion action into different energy-mode components that can be integrated separately in Hybrid Monte Carlo. We have shown that these techniques can provide significant cost benefits.

First, we looked at polynomial filtering and mass preconditioning for double-flavour pseudofermions, showing that a combination of the filters produces significant computational cost improvements without the need for filter parameter fine-tuning. This should prove useful for production lattice simulations, where the lattices are often so computationally intensive that the effort required to finely tune such parameters is significant.

Next, we considered single-flavour pseudofermions, with polynomial filtering and truncated ordered product RHMC. There, we found that tRHMC produced the biggest improvement in cost. This technique is also very simple to implement into existing codebases with RHMC. Furthermore, by applying a novel step-size tuning technique we denote c-scale tuning, we can significantly reduce the dependency of the cost on the truncation order. This should further reduce the tuning effort required for production lattice simulations.

Finally, we tried the tRHMC technique on a dynamical Lattice QCD+QED simulation, showing that we can achieve cost improvements there as well. The c-scale tuning technique also provides a similar reduction in filter parameter dependency. The combination of tRHMC and c-scale tuning should prove useful for dynamical Lattice QCD+QED simulations near the physical point, such that important quantities like the neutron-proton mass difference can be accurately calculated in the Standard Model. Discrepancies with experimental values would then provide evidence for physics beyond the Standard Model.

∘ **A** ∘

Algebraic toolbox

This appendix describes several pieces of algebra that appear throughout the thesis.

A.1 Matrix definitions

The generators of SU(3) span the $\mathfrak{su}(3)$ algebra, which is usually represented as the set of traceless Hermitian 3×3 matrices. In this representation, the generators are given by $t^{(a)} = \lambda_a/2$ where λ_a are the Gell-Mann matrices:

$$\lambda_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \lambda_2 = \begin{pmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad \lambda_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$
$$\lambda_4 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad \lambda_5 = \begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix}, \quad \lambda_6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix},$$
$$\lambda_7 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix}, \quad \lambda_8 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}. \tag{A.1}$$

The gamma matrices γ_μ are a set of four matrices which satisfy the anti-commutation relation

$$\{\gamma_\mu, \gamma_\nu\} = 2\eta_{\mu\nu} I \tag{A.2}$$

where $\eta_{\mu\nu}$ is the metric tensor. The smallest matrix size which can satisfy this relation are 4×4, and within that there are many possible representations. In Lattice QCD, we are mainly concerned with Euclidean space-time where $\eta_{\mu\nu} =$

$\delta_{\mu\nu}$. A common representation of the gamma matrices in this space-time is the chiral representation, where

$$\gamma_1 = \begin{pmatrix} 0 & 0 & 0 & i \\ 0 & 0 & i & 0 \\ 0 & -i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}, \quad \gamma_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix},$$
$$\gamma_3 = \begin{pmatrix} 0 & 0 & i & 0 \\ 0 & 0 & 0 & -i \\ i & 0 & 0 & 0 \\ 0 & -i & 0 & 0 \end{pmatrix}, \quad \gamma_4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. \tag{A.3}$$

We also define another matrix, γ_5, where

$$\gamma_5 \equiv \gamma_1\gamma_2\gamma_3\gamma_4 = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{(chiral)}. \tag{A.4}$$

A.2 Theorems

Theorem A.1 (Matthews-Salam for bosonic fields)**.** Let a_i, $i = 1, \ldots, N$ be a set of complex numbers and M_{ij} be a positive-definite complex matrix. Then

$$\int da_1 da_1^* \ldots da_N da_N^* \exp\left(-\sum_{i,j=1}^{N} a_i^* (M)_{ij}^{-1} a_j\right) = \pi^N \det M. \tag{A.5}$$

Proof. Consider the transformed co-ordinates

$$a_i' = \sum_{i=1}^{N} (M)_{ij}^{-1} a_j.$$

It follows that with this change of variables, we have

$$da_1' \ldots da_N' = |\det M^{-1}| da_1 \ldots da_N,$$
$$\implies \quad \det M da_1' \ldots da_N' = da_1 \ldots da_N,$$

using the fact that $\det M > 0$. Hence, we can write the integral as

$$\begin{aligned}
&\int da_1 da_1^* \dots da_N da_N^* \exp\left(-\sum_{i,j=1}^{N} a_i^* (M)_{ij}^{-1} a_j\right) \\
&= \det M \int da_1' da_1^* \dots da_N' da_N^* \exp\left(-\sum_{i=1}^{N} a_i^* a_i'\right) \\
&= \det M \prod_{i=1}^{N} \int da_i' da_i^* \exp(-a_i^* a_i') \\
&= \det M \pi^N. \qquad \square
\end{aligned}$$

A.2.1 Symmetry of overlaid integrators

In order to show that the overlaid multi-scale integrators are time-reversible, we first define a useful construct for expressing integrators.

Definition A.1 (Time-step inserted product)**.** Let $\hat{A}_i[\mu = b_i], i = 1, \dots n$ be a set of integration steps $\hat{A}_i$ augmented by computational time parameters $\mu = b_i$. Then the time-step inserted product of these operators from $\mu = 0$ to $\mu = \tau$

$$\hat{M} = \mathcal{T}_0^{\tau}\left(\prod_{i=1}^{n} \hat{A}_i[\mu = b_i]\right) \tag{A.6}$$

is defined as

$$\hat{M} = \hat{T}[b_{n+1}' - b_n'] \prod_{i=1}^{n} (\hat{A}_i \hat{T}[b_i' - b_{i-1}']), \tag{A.7}$$

where

- $\hat{T}[h]$ is the atomic time-step (2.38),
- the b_i have been rearranged to b_i' such that $b_{i+1}' \geq b_i'$, with any steps at the same time $b_i' = b_{i+1}'$ remaining in the same order,
- $b_0' = 0$, $b_{n+1}' = \tau$, and
- The product Π is left-ordered, such that $\Pi[a_i] = a_n a_{n-1} \cdots a_2 a_1$.

This offers a convenient way to express symplectic integrators of trajectory length τ. As we will usually use the limits $(0, \tau)$, we denote $\mathcal{T} = \mathcal{T}_0^{\tau}$. For

example, a one-step second-order minimal norm space-time-space scheme of length $\tau = h$ can be expressed as

$$\hat{M}_{2MNSTS}[h] = \mathcal{T}\left(\hat{S}[\lambda h; \mu = h]\hat{S}[(1-2\lambda)h; \mu = h/2]\hat{S}[\lambda h; \mu = 0]\right). \quad \text{(A.8)}$$

It is even more convenient for expressing overlaid integrators: given a set of N integration schemes that conserve the partial Hamiltonians $H_i = T + S_i$

$$\hat{M}_i = \mathcal{T}\left(\prod_{j=1}^{n_i} \hat{S}_i[b_j^i; \mu = c_j^i]\right), \quad \text{(A.9)}$$

the corresponding overlaid integration scheme is simply

$$\hat{M}_{\text{overlaid}} = \mathcal{T}\left(\prod_{i=1}^{N}\prod_{j=1}^{n_i} \hat{S}_i[b_j^i; \mu = c_j^i]\right). \quad \text{(A.10)}$$

Note that as $[\hat{S}_i[h_1], \hat{S}_i[h_2]] = 0$, the constituent schemes can always be written such that the c_j^i are distinct for each scheme i and $b_j^i \neq 0$.

The parameter μ can be thought of as the cumulative total of time-step updates, ranging from 0 to τ. Note that this parameter does not modify the action of an integration step in any way – it is only used to specify the action of the time-step inserted product above.

For time-step inserted products of operators, the symmetry condition becomes as follows.

Theorem A.2 (Symmetry of a time-step inserted product)**.** Suppose we have an integration scheme

$$\hat{M} = \mathcal{T}\left(\prod_{i=1}^{n} \hat{A}_i[\mu = b_i]\right). \quad \text{(A.11)}$$

where $\hat{A}_i \neq \hat{I}$ are reversible and the b_i are distinct. Then $\hat{M}$ is symmetric, and hence reversible, iff for every operator $\hat{A}[\mu = b]$ in the product, there is also an operator of the form $\hat{A}[\mu = \tau - b]$.

Proof. Under the action of the time-step insertion operator, we can rearrange the operators $\hat{A}_i$ such that $b_i < b_{i+1}$. Expanding the time-step insertion operator then gives

$$\hat{M} = \hat{T}[b_{n+1} - b_n]\prod_{i=1}^{n}\left(\hat{A}_i\hat{T}[b_i - b_{i-1}]\right),$$

where $b_0 = 0$ and $b_{n+1} = \tau$, and only the operators at either end, $\hat{T}[b_1 - b_0]$ and $\hat{T}[b_{n+1} - b_n]$, can possibly be equal to the identity $\hat{I}$.

By Theorem 2.8, this product is symmetric iff

$$\hat{A}_i = \hat{A}_{n-i+1} \tag{A.12}$$

and

$$b_i - b_{i-1} = b_{n-i+2} - b_{n-i+1} \quad \forall i = 0, \dots, n+1.$$

Rearranging the second condition gives

$$\begin{aligned} b_i + b_{n-i+1} &= b_{i-1} + b_{n-i+2} \\ &= b_{i-2} + b_{n-i+3} \\ &= \dots \\ &= b_0 + b_{n+1} = \tau, \end{aligned}$$

so

$$b_i + b_{n-i+1} = \tau \quad \forall i = 0, \dots, n+1. \tag{A.13}$$

The two conditions (A.12) and (A.13) together are equivalent to saying that for each the operator $\hat{A}_i[\mu = b_i]$ in the product, we also have $\hat{A}_{n-i+1}[\mu = b_{n-1+1}] = \hat{A}_i[\mu = \tau - b_i]$. □

Given the reversibility condition Theorem A.2, we can then systematically show reversibility for overlaid integrators.

Theorem A.3 (Reversibility of overlaid integrators)**.** If the constituent symplectic integrators of an overlaid multi-scale integrator are symmetric, then the overlaid multi-scale integrator is also symmetric and hence time-reversible.

Proof. Consider the overlaid integrator with N schemes, $\hat{M}_{\text{overlaid}}$ (A.10), expressing the constituent schemes in minimal form such that the c_j^i are distinct for each scheme i.

By assumption, the constituent schemes are symmetric. Hence, by Theorem A.2, it follows that each $\hat{S}_i[b, \mu = c]$ has a mirror $\hat{S}_i[b, \mu = \tau - c]$ for all of the schemes $i = 1, \dots, N$.

Now suppose that for some $\mu = c$, we have a set of $M \leq N$ space-steps of the form $\hat{S}_j[b_j, \mu = c]$. Upon application of the time-step insertion operator, these form the left-ordered product $\hat{S}_M \cdots \hat{S}_1$ in the integration scheme, which we define as a single operator $\hat{S}'$ to ensure only one operator in the scheme has $\mu = c$. As each constituent scheme is symmetric, the constituent steps of $\hat{S}'$ have mirrors $\hat{S}_j[b_j, \mu = \tau - c]$, which upon application of the time-step insertion operator are *also* left-ordered $\hat{S}_M \cdots \hat{S}_1$ in the product. Thus, $\hat{S}'[\mu = c]$ has a

mirror $\hat{S}'[\mu = \tau - c]$. However, we must demonstrate that $\hat{S}'$ is reversible, which by Theorem 2.8 requires

$$\hat{S}_1[b_1]\hat{S}_2[b_2]\cdots\hat{S}_M[b_M] = \hat{S}_M[b_M]\cdots\hat{S}_2[b_2]\hat{S}_1[b_1]. \tag{A.14}$$

As $[\hat{S}_i[a], \hat{S}_j[b]] = 0$ for every i, j, a, b, this equation is satisfied.

Thus, we can write $\hat{M}_{\text{overlaid}}$ in a form such that every operator $\hat{M}[\mu = c]$ in the product has a mirror $\hat{M}[\mu = \tau - c]$, every operator is reversible, and no two operators have the same μ. Hence, by Theorem A.2, $\hat{M}_{\text{overlaid}}$ is symmetric and thus reversible. □

A.3 Integration scheme errors

In this section, we consider the discretisation errors of the integration schemes used in Hybrid Monte Carlo, which quantify how accurately an integration scheme preserves the Hamiltonian and hence determines the acceptance rate P_{acc}.

For a classical system described by the Hamiltonian operator $H = T + S$, the operator $e^{\tau H}$ can be used to advance the system state by time τ. In HMC, $e^{\tau H}$ does not have a linear form, and can't be used to advance the system (P, U) directly. However, we do have linear forms for the operators e^{hT} and e^{hS}; these are simply the time $\hat{T}[h]$ and space $\hat{S}[h]$ atomic integration steps (2.51). When we combine these steps into an integration scheme of trajectory length τ, we are effectively applying an operator $e^{\tau\tilde{H}}$ for some $\tilde{H}$ which is known as the *shadow Hamiltonian* for the scheme. The difference $\tilde{H} - H$ is the *discretisation error* of the integration scheme or the *integration error*, and we would like to keep this small in order to maintain a high acceptance rate P_{acc}.

To calculate the discretisation error for a given scheme, we can start by writing the full integration scheme in terms of the T and S operators: for the integrators in subsection 2.4.3, we replace the time step $\hat{T}[a]$ with e^{aT} where a is the step-size, and the space step $\hat{S}[a]$ with e^{aS}. If we rewrite the resultant product of exponentials as a single exponential factor, we determine $\tilde{H}$ and hence the error. This can achieved by repeatedly applying the Baker-Campbell-Hausdorff formula for a symmetric product,

$$\ln\left(e^{h\hat{A}}e^{h\hat{B}}e^{h\hat{A}}\right) = h(2A + B) - \frac{h^3}{6}([B,[A,B]] + [A,[A,B]]) + \mathcal{O}(h^5), \tag{A.15}$$

assuming a symmetric integrator as we use for Hybrid Monte Carlo. For example, the discretisation error of a second-order minimal norm space-time-space

(2MNSTS) integration step of size h (2.57) is

$$\tilde{H}_{2MN} - H = h^2 \left(\frac{6\lambda^2 - 6\lambda + 1}{12} [S,[S,T]] + \frac{1-6\lambda}{24} [T,[S,T]] \right) + \mathcal{O}(h^4). \quad \text{(A.16)}$$

In the practical case of applying n such steps to form a trajectory of length $\tau = nh$, the error is given by $n(\tilde{H}_{2MN} - \hat{H})$.

For a multi-scale integration scheme, the Hamiltonian operator is given by $H = T + \sum_i S_i$ where we sum over several action terms S_i. The error of such an integration scheme has terms of the form $[S_i,[S_j,T]]$ and $[T,[S_i,T]]$ at first order. For symmetric symplectic integration schemes, the terms of the form $[T,[S_i,T]$ and $[S_i,[S_i,T]]$ for a particular S_i are identical to what would be the case if the integration scheme only had the T and S_i operators, e.g. (A.16). This can be shown by considering all possible scenarios when applying the symmetric Baker-Campbell-Housdorff formula (A.15) recursively. The cross terms $[S_i,[S_j,T]]$, $i \neq j$ are thus a distinguishing feature of multi-scale schemes, and they typically reduce the benefit of placing one action term on a finer time-scale than another.

For a nested 2MNSTS scheme with step-size h for S_1 and nesting factor a for S_2 (i.e. each $\hat{T}$ step was replaced by a 2MNSTS steps for S_2), the error is

$$\begin{aligned} \tilde{H}_{\text{nested}} - \hat{H} = {} & h^2 \left(\frac{6\lambda^2 - 6\lambda + 1}{12} [S_1,[S_1,T]] + \frac{1-6\lambda}{24} [T,[S_1,T]] \right) \\ & + \frac{h^2}{2a} \left(\frac{6\lambda^2 - 6\lambda + 1}{12} [S_2,[S_2,T]] + \frac{1-6\lambda}{24} [T,[S_2,T]] \right) \\ & + h^2 \frac{1-6\lambda}{24} [S_1,[S_2,T]] + \mathcal{O}(h^4), \end{aligned} \quad \text{(A.17)}$$

Note that this scheme is exactly equivalent to an overlaid 2MNSTS scheme with step-sizes $h_1 = h$ and $h_2 = h/2a$.

When we use overlaid integrators, we have access to a variety of different step-sizes (h_1, h_2) that are not possible with nesting. In such cases, the integration error generally differs from the nested case.

Consider an overlaid 2MNSTS scheme with step-sizes $h_1 = h$ and $h_2 = h/a$ where a is odd; the case of a even corresponds to a nested scheme with nesting

factor $a/2$. The error of this scheme is given by

$$\begin{aligned}\tilde{H}_{\text{nested}} - \hat{H} = h^2 &\left(\frac{6\lambda^2-6\lambda+1}{12}[S_1,[S_1,T]] + \frac{1-6\lambda}{24}[T,[S_1,T]]\right) \\ &+ \frac{h^2}{a}\left(\frac{6\lambda^2-6\lambda+1}{12}[S_2,[S_2,T]] + \frac{1-6\lambda}{24}[T,[S_2,T]]\right) \\ &+ h^2\left[\frac{1}{a^2}\frac{6\lambda^2-6\lambda+1}{6} + \left(\frac{a^2-1}{a^2}\right)\frac{1-6\lambda}{24}\right][S_1,[S_2,T]] + \mathcal{O}(h^4).\end{aligned} \tag{A.18}$$

Hence, depending on a and λ, the cross-term coefficient can be larger or smaller than the nested case (A.17).

In this thesis, we use $\lambda \approx 0.193183$ which minimises the Euclidean norm of the two error terms in the nested case (A.17). That is, it minimises $\sqrt{\alpha^2+\beta^2}$ where $\alpha = (6\lambda^2-6\lambda+1)/12$ and $\beta = (1-6\lambda)/24$. Hence, the coefficient of the cross-term in the nested case is

$$\frac{1-6\lambda}{24} \approx -0.006629. \tag{A.19}$$

Denoting the coefficient of the cross-term in (A.18) as $c(a)$, we have:

$$\begin{aligned} c(1) &= \frac{6\lambda^2-6\lambda+1}{6} \approx 0.01080, \\ c(3) &\approx -0.004692, \\ c(5) &\approx -0.005932, \\ c(7) &\approx -0.006273, \\ &\cdots \\ c(\infty) &= \frac{1-6\lambda}{24} \approx -0.006629. \end{aligned}$$

Therefore, for odd factors $a > 1$, the cross-term has a smaller coefficient than in the nested case. If we were to interpolate the nested integrator case for arbitrary factors a, this particular integration scheme would have a smaller error than the interpolation. This effect reduces as we increase a.

The other case of a two-level overlaid 2MNSTS integrator is where one level has n steps of size h/n and the other has $m > n$ steps of size h/m where m is not a multiple of n. In such cases, the discretisation error is generally worse than an interpolation of the nested case.

∘ **B** ∘

Statistical analysis

In this appendix, we outline the various statistical tools used to obtain meaningful information from Lattice QCD configurations.

B.1 Uncorrelated data

Let's consider analysing a set of n measurements X_i of some observable X. In the frequentist approach, these sample a true normal population distribution of X measurements with mean $E(X) = \mu$ and variance $\mathrm{Var}(X) = \sigma^2$. While we do not know these parameters in practice, we can construct effective estimators based upon our sample. An unbiased estimator of the true mean value μ is

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \tag{B.1}$$

while the true variance σ^2 of the observable can be estimated by

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n} (\overline{X} - X_i)^2. \tag{B.2}$$

Often, we want to know the mean value of X and an estimate on how accurate this mean value is. For the latter, we use the square root of the variance of the mean,

$$\sigma_{SEM} = \sqrt{\mathrm{Var}(\overline{X})}, \tag{B.3}$$

which is known as the *standard error of the mean*. To construct an estimator for the standard error, we consider each measurement X_i a random variable and

expand (B.3) in terms of these variables:

$$\begin{aligned}\mathrm{Var}(\overline{X}) &= \mathrm{Var}\left(\frac{1}{n}\sum_{i=1}^{n} X_i\right) \\ &= \mathrm{Cov}\left(\frac{1}{n}\sum_{i=1}^{n} X_i, \frac{1}{n}\sum_{j=1}^{n} X_j\right) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Cov}(X_i, X_j),\end{aligned} \tag{B.4}$$

where $\mathrm{Cov}(X,Y)$ is the covariance

$$\mathrm{Cov}(X,Y) = (E[X]-X)(E[Y]-Y), \tag{B.5}$$

and $\mathrm{Var}(X) \equiv \mathrm{Cov}(X,X)$. The last equality uses the fact that the covariance is bilinear. In the case of uncorrelated data, all the covariances where $i \neq j$ vanish and we are left with

$$\mathrm{Var}(\overline{X}) = \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(X_i). \tag{B.6}$$

If we assume the data is equally distributed with variance σ^2, the variance of the mean $\overline{X}$ of the data is then given by

$$\mathrm{Var}(\overline{X}) = \frac{\sigma^2}{n}. \tag{B.7}$$

It follows that we can estimate the standard error of the mean by

$$\hat{\sigma} = \frac{s}{\sqrt{n}}, \tag{B.8}$$

and so we quote the mean value of our observable X as

$$\langle X \rangle = \overline{X} \pm \hat{\sigma}. \tag{B.9}$$

B.2 Correlated data

We often encounter data which has correlations. This means that the covariance $\mathrm{Cov}(X_i, X_j)$ is non-zero for some $i \neq j$. In this case, the variance of the mean is given by

$$\begin{aligned}\mathrm{Var}(\overline{X}) &= \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\mathrm{Cov}(X_i, X_j) \\ &= \frac{1}{n^2}\sum_{i=1}^{n}\mathrm{Var}(X_i) + \frac{2}{n^2}\sum_{1\leq i<j\leq n}\mathrm{Cov}(X_i, X_j).\end{aligned} \tag{B.10}$$

Let's define the *autocorrelation* as the covariance between two measurements X_i and X_j normalised by their variances:

$$R(i,j) = \frac{\mathrm{Cov}(X_i, X_j)}{\sigma_i \sigma_j}. \tag{B.11}$$

For stationary processes (e.g. HMC), this function only depends on the *lag* $t = j - i$ between the measurements in computational 'time'. The autocorrelation can thus be written as

$$R(t) = \frac{\mathrm{Cov}(X_i, X_{i+t})}{\sigma^2}. \tag{B.12}$$

Substituting into the variance of the mean gives

$$\begin{aligned} \mathrm{Var}(\overline{X}) &= \frac{1}{n}\sigma^2 + \frac{2}{n^2}\sum_{t=1}^{n-1}\sigma^2(n-\tau)R(t) \\ &= \frac{\sigma^2}{n}\left(1 + 2\sum_{t=1}^{n-1}\frac{n-t}{n}R(t)\right). \end{aligned} \tag{B.13}$$

For large n, a good approximation for this variance is

$$\mathrm{Var}(\overline{X}) \approx \frac{\sigma^2}{n}\tau \tag{B.14}$$

where τ is the *autocorrelation time*, defined as

$$\tau = 1 + 2\sum_{i=1}^{\infty} R(t). \tag{B.15}$$

The error in the mean is thus

$$\hat{\sigma}_{\mathrm{corr}} = \sqrt{\frac{\tau}{n}}\sigma. \tag{B.16}$$

Comparing this to the uncorrelated case (B.7), we are effectively using a sample size

$$n_{\mathrm{eff}} = \frac{n}{\tau}. \tag{B.17}$$

Thus, the autocorrelation time τ can be thought of as a measure of how much lag is required between measurements to remove correlations.

As all of the data sample the same distribution, a suitable estimator of the covariance is

$$\mathrm{Cov}(X_i, X_j) \approx (\overline{X} - X_i)(\overline{X} - X_j). \tag{B.18}$$

The autocorrelation can then be estimated by

$$\hat{R}(t) = \frac{1}{n-t}\sum_{i=1}^{n-t}\frac{(\overline{X}-X_i)(\overline{X}-X_{i+\tau})}{s^2}. \tag{B.19}$$

In practice, the estimate for the autocorrelation becomes unreliable for sufficiently large lags t. Thus, to estimate the autocorrelation time we often choose a suitable cutoff T and write

$$\tau \approx 1 + 2\sum_{i=1}^{T}\hat{R}(t). \tag{B.20}$$

A fairly large number of measurements are required for a reliable estimate of the autocorrelation, $\sim 1000\tau$.

In Lattice QCD, we are interested in a wide range of observables on a set of gauge configurations. Each of these have different autocorrelation times. As the autocorrelation time is hard to measure in practice, we usually use the autocorrelation time of some representative observable O in all cases, $\tau = \tau_O$. An example of an observable with typical autocorrelation time is the plaquette. Note however that observables with non-local structure such as the topological charge often have larger autocorrelation times.

We often cannot obtain a reliable estimate of the autocorrelation time. In this case, we can employ other methods to extract an uncorrelated set of data from our sample, such that we can estimate the standard error via (B.7). One such method is *data blocking*, where we bin our data into n/m bins of size m, then take the mean value of the observable in each bin

$$Y_i = \frac{1}{m}\sum_{j=(m-1)i}^{mi-1} X_j \tag{B.21}$$

as our new data set. Consider the covariance of this new measurement Y_i:

$$\begin{aligned}\mathrm{Cov}(Y_i, Y_j) &= \mathrm{Cov}\left(\frac{1}{m}\sum_{k=(m-1)i}^{mk-1} X_k, \frac{1}{m}\sum_{l=(m-1)j}^{li-1} X_l\right)\\ &= \frac{1}{m^2}\sum_{k=(m-1)i}^{mk-1}\sum_{l=(m-1)j}^{li-1}\mathrm{Cov}(X_k, X_l).\end{aligned} \tag{B.22}$$

For $i \neq j$, the covariance in X_i has an average lag of $|i-j|m$. Hence, for large enough bins m, the covariance for $i \neq j$ vanishes (corresponding to where $R(m) \to 0$), and the blocked data set is uncorrelated. More simply, we can choose to only take every $N_{\mathrm{corr}}^{\mathrm{th}}$ data point into our data set, using some upper bound estimate $N_{\mathrm{corr}} > \tau$ to remove autocorrelations.

B.3 Small data sets

Due to the inherent computational cost in generating configurations and calculating observables in Lattice QCD, we often only have small sets of uncorrelated data. In such cases, we cannot reliably estimate the variance of the data and thus the standard error in the usual manner.

Surprisingly, we can artificially inflate the number of measurements by resampling our existing uncorrelated sample and thus obtain better estimators. One way we can do this is *statistical bootstrap*, where we create an arbitrarily-sized bootstrap sample data set of size N_{boot} (at least 1000) by randomly sampling our set of n measurements with replacement. Then we can use the usual statistical estimators on this new data set to find reliable estimates on the variance and standard error, "as if pulling oneself up by their bootstraps".

B.4 The binomial distribution

An important quantity for determining the performance of Hybrid Monte Carlo is the Metropolis-Hastings acceptance rate $\langle P_{\text{acc}} \rangle$. This is the number of successfully accepted configurations at the acceptance step divided by the number of steps, so it is related to a *binomial distribution*.

A binomial distribution characterises a series of independent yes/no experiments with the same success probability, and is parametrised by the number of trials n and the probability of success p. The probability mass function, which is the probability of exactly k successes from n trials, is given by

$$Pr(k; n, p) = \frac{n!}{k!(n-k)!} p^k (1-p)^{n-k}. \tag{B.23}$$

The mean of this distribution is given by

$$E(X) = np, \tag{B.24}$$

while the variance is given by

$$\text{Var}(X) = np(1-p). \tag{B.25}$$

The acceptance rate follows the binomial distribution, but is normalised by the number of experiments n. In practice, we do not know what p is, but it can be estimated by

$$\hat{p} = \frac{s}{n} \tag{B.26}$$

when we observe s accepted configurations after n steps. An effective estimator of the variance of the acceptance rate is then given by

$$\hat{\sigma}^2 = \frac{n\hat{p}(1-\hat{p})}{n^2} = \frac{\hat{p}(1-\hat{p})}{n}. \tag{B.27}$$

For sufficiently large n, the binomial distribution tends to the normal distribution, where we can quote the acceptance rate as

$$\langle P_{\text{acc}} \rangle = \hat{p} \pm \hat{\sigma}. \tag{B.28}$$

∘ C ∘

Extra data for chapter 3

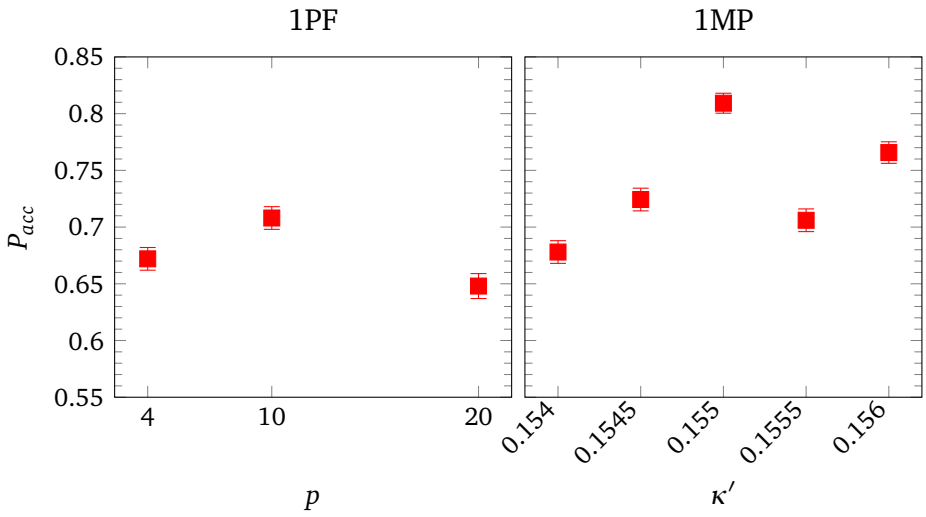

Figure C.1: Acceptance rates for the double-flavour 1-filter actions with the parameters given in Tables 3.1 and 3.2.

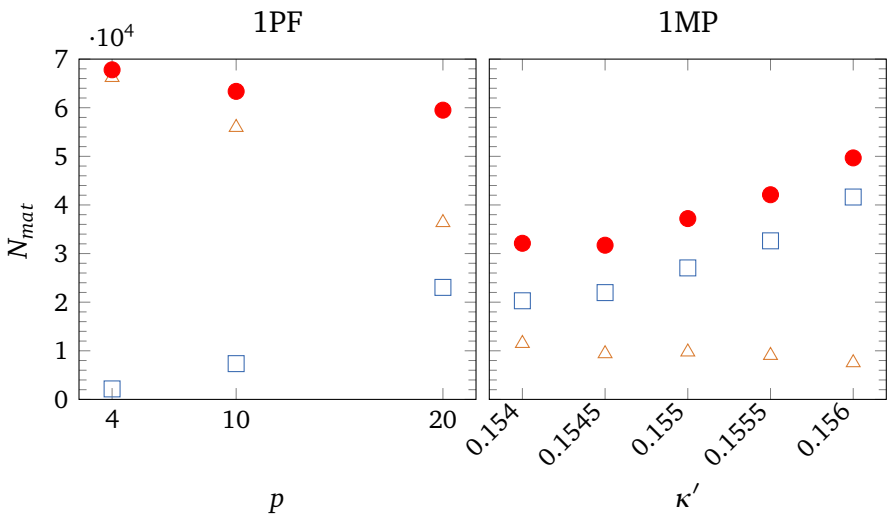

Figure C.2: Matrix operation counts for double-flavour 1-filter actions with the parameters given in Tables 3.1 and 3.2. Squares = matrix operations to construct F_1, triangles = F_2 construction, filled circles = total. Statistical errors are smaller than marker size.

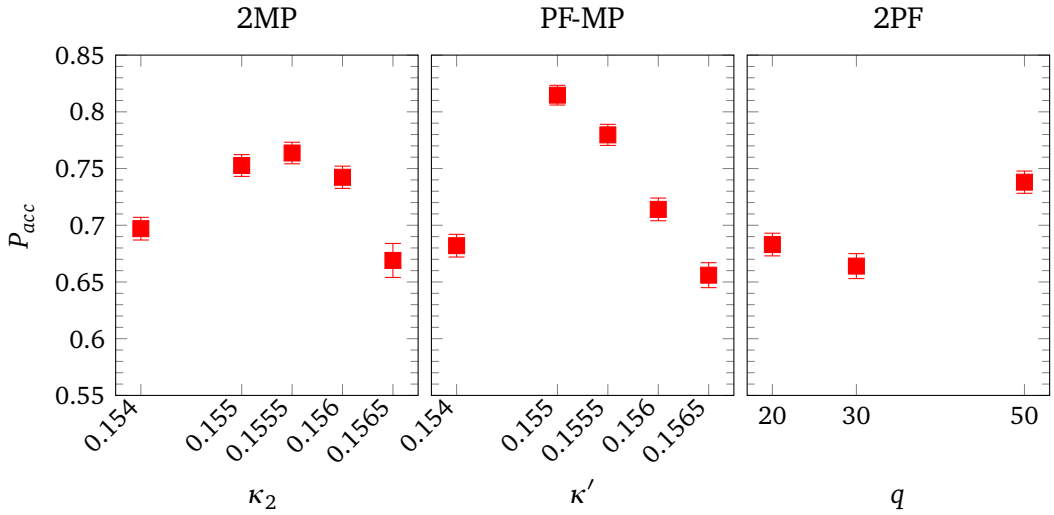

Figure C.3: Acceptance rates for the double-flavour 2-filter actions with the parameters given in Tables 3.3, 3.4 and 3.5.

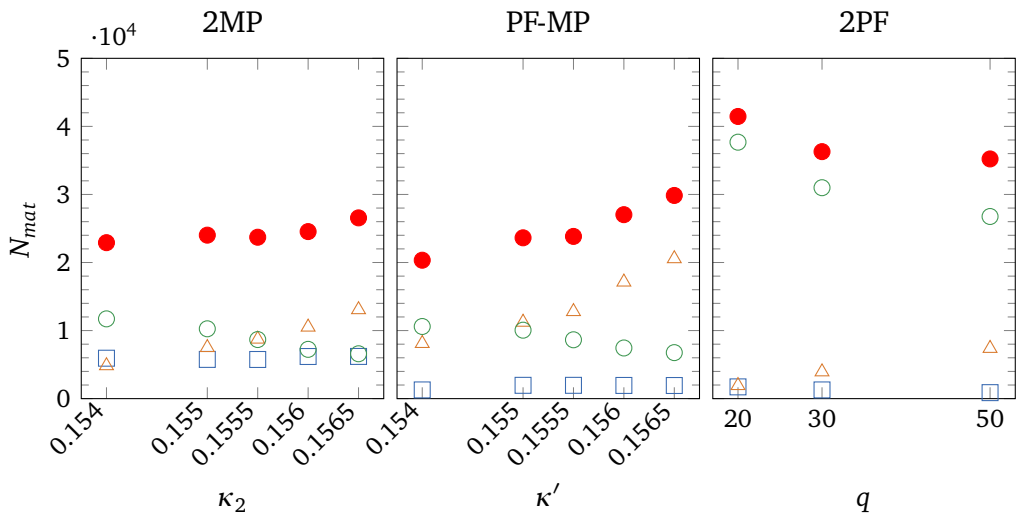

Figure C.4: Matrix operation counts for double-flavour 2-filter actions with the parameters given in Tables 3.3, 3.4 and 3.5. Squares = matrix operations to construct F_1, triangles = F_2 construction, empty circles = F_3 construction, filled circles = total. Statistical errors are smaller than marker size.

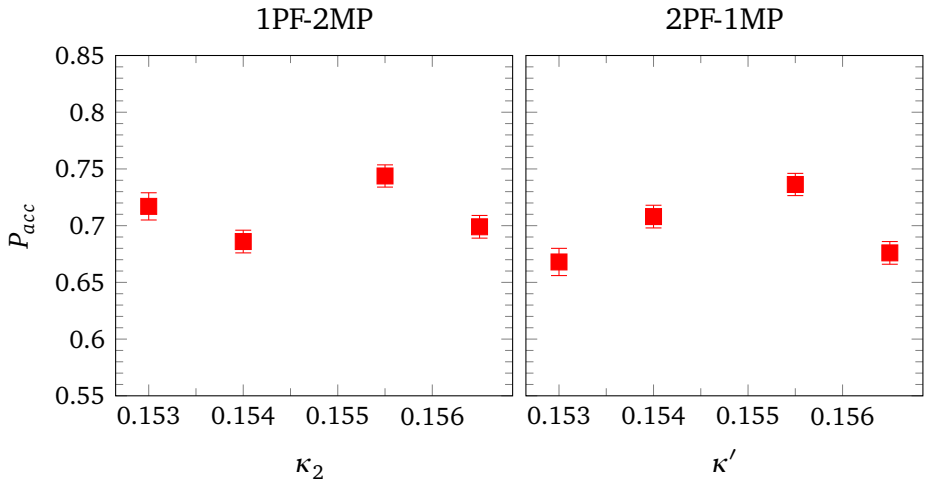

Figure C.5: Acceptance rates for the double-flavour 3-filter actions with the parameters given in Tables 3.6 and 3.7.

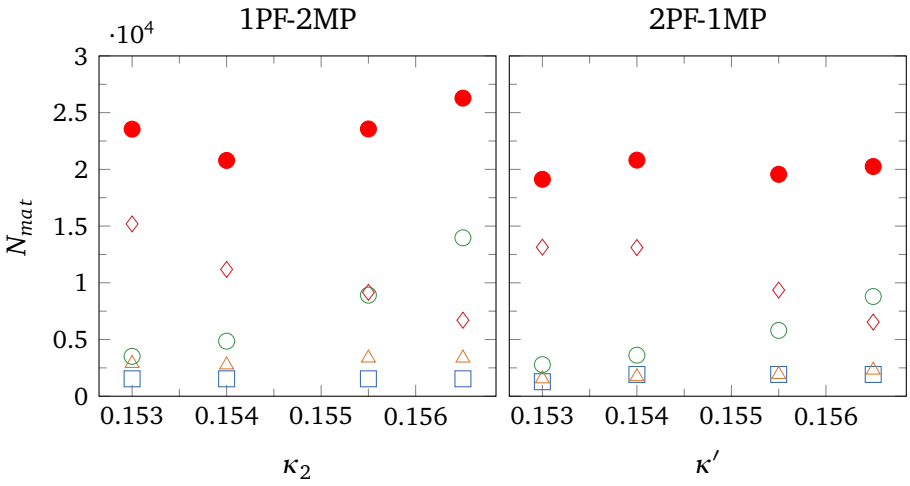

Figure C.6: Matrix operation counts for double-flavour 3-filter actions with the parameters given in Tables 3.6 and 3.7. Squares = matrix operations to construct F_1, triangles = F_2 construction, empty circles = F_3 construction, diamonds = F_4 construction, filled circles = total. Statistical errors are smaller than marker size.

Bibliography

[1] Kenneth G. Wilson. "Confinement of Quarks". In: *Phys. Rev.* D10 (1974), pp. 2445–2459. DOI: 10.1103/PhysRevD.10.2445.

[2] S. Durr et al. "Lattice QCD at the physical point: Simulation and analysis details". In: *JHEP* 08 (2011), p. 148. DOI: 10.1007/JHEP08(2011)148. arXiv: 1011.2711 [hep-lat].

[3] Patrick Fritzsch et al. "The strange quark mass and Lambda parameter of two flavor QCD". In: *Nucl. Phys.* B865 (2012), pp. 397–429. DOI: 10.1016/j.nuclphysb.2012.07.026. arXiv: 1205.5380 [hep-lat].

[4] T. Blum et al. "Domain wall QCD with physical quark masses". In: *Phys. Rev.* D93.7 (2016), p. 074505. DOI: 10.1103/PhysRevD.93.074505. arXiv: 1411.7017 [hep-lat].

[5] A. Bazavov et al. "Charmed and light pseudoscalar meson decay constants from four-flavor lattice QCD with physical light quarks". In: *Phys. Rev.* D90.7 (2014), p. 074509. DOI: 10.1103/PhysRevD.90.074509. arXiv: 1407.3772 [hep-lat].

[6] Bipasha Chakraborty et al. "High-precision quark masses and QCD coupling from $n_f = 4$ lattice QCD". In: *Phys. Rev.* D91.5 (2015), p. 054508. DOI: 10.1103/PhysRevD.91.054508. arXiv: 1408.4169 [hep-lat].

[7] S. Aoki et al. "Review of lattice results concerning low-energy particle physics". In: *Eur. Phys. J.* C77.2 (2017), p. 112. DOI: 10.1140/epjc/s10052-016-4509-7. arXiv: 1607.00299 [hep-lat].

[8] Paul A. M. Dirac. "The Lagrangian in quantum mechanics". In: *Phys. Z. Sowjetunion* 3 (1933), pp. 64–72.

[9] R. P. Feynman. "Space-time approach to nonrelativistic quantum mechanics". In: *Rev. Mod. Phys.* 20 (1948), pp. 367–387. DOI: 10.1103/RevModPhys.20.367.

[10] Martin Lüscher and Stefan Schaefer. "Lattice QCD without topology barriers". In: *JHEP* 07 (2011), p. 36. DOI: 10.1007/JHEP07(2011)036. arXiv: 1105.4749 [hep-lat].

[11] K. Symanzik. "Continuum Limit and Improved Action in Lattice Theories. 1. Principles and phi**4 Theory". In: *Nucl. Phys.* B226 (1983), pp. 187–204. DOI: 10.1016/0550-3213(83)90468-6.

[12] M. Lüscher and P. Weisz. "On-Shell Improved Lattice Gauge Theories". In: *Commun. Math. Phys.* 97 (1985). [Erratum: Commun. Math. Phys. 98, 433 (1985)], p. 59. DOI: 10.1007/BF01206178.

[13] G. Curci, P. Menotti, and G. Paffuti. "Symanzik's Improved Lagrangian for Lattice Gauge Theory". In: *Phys. Lett.* 130B (1983). [Erratum: Phys. Lett. 135B, 516 (1984)], p. 205. DOI: 10.1016/0370-2693(83)91043-2.

[14] B. Sheikholeslami and R. Wohlert. "Improved Continuum Limit Lattice Action for QCD with Wilson Fermions". In: *Nucl. Phys.* B259 (1985), p. 572. DOI: 10.1016/0550-3213(85)90002-1.

[15] Karl Jansen and Rainer Sommer. "O(a) improvement of lattice QCD with two flavors of Wilson quarks". In: *Nucl. Phys.* B530 (1998). [Erratum: Nucl. Phys. B643, 517 (2002)], pp. 185–203. DOI: 10.1016/S0550-3213(98)00396-4. arXiv: hep-lat/9803017 [hep-lat].

[16] Holger Bech Nielsen and M. Ninomiya. "No Go Theorem for Regularizing Chiral Fermions". In: *Phys. Lett.* 105B (1981), pp. 219–223. DOI: 10.1016/0370-2693(81)91026-1.

[17] Herbert Neuberger. "Exactly massless quarks on the lattice". In: *Phys. Lett.* B417 (1998), pp. 141–144. DOI: 10.1016/S0370-2693(97)01368-3. arXiv: hep-lat/9707022 [hep-lat].

[18] Vadim Furman and Yigal Shamir. "Axial symmetries in lattice QCD with Kaplan fermions". In: *Nucl. Phys.* B439 (1995), pp. 54–78. DOI: 10.1016/0550-3213(95)00031-M. arXiv: hep-lat/9405004 [hep-lat].

[19] P. T. Matthews and A. Salam. "Propagators of quantized field". In: *Il Nuovo Cimento (1955-1965)* 2.1 (1955), pp. 120–134. ISSN: 1827-6121. DOI: 10.1007/BF02856011.

[20] C. Gattringer and C.B. Lang. *Quantum Chromodynamics on the Lattice: An Introductory Presentation*. Vol. 788. Lecture Notes in Physics. Springer Berlin Heidelberg, 2010. ISBN: 978-3-642-01849-7. DOI: 10.1007/978-3-642-01850-3.

[21] S. Aoki et al. “Physical Point Simulation in 2+1 Flavor Lattice QCD”. In: *Phys. Rev.* D81 (2010), p. 074503. DOI: 10.1103/PhysRevD.81.074503. arXiv: 0911.2561 [hep-lat].

[22] W. Bietenholz et al. “Tuning the strange quark mass in lattice simulations”. In: *Phys. Lett.* B690 (2010), pp. 436–441. DOI: 10.1016/j.physletb.2010.05.067. arXiv: 1003.1114 [hep-lat].

[23] N. Metropolis et al. “Equation of state calculations by fast computing machines”. In: *J. Chem. Phys.* 21 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114.

[24] W. K. Hastings. “Monte Carlo Sampling Methods Using Markov Chains and Their Applications”. In: *Biometrika* 57 (1970), pp. 97–109. DOI: 10.1093/biomet/57.1.97.

[25] Martin Lüscher. “A New approach to the problem of dynamical quarks in numerical simulations of lattice QCD”. In: *Nucl. Phys.* B418 (1994), pp. 637–648. DOI: 10.1016/0550-3213(94)90533-9. arXiv: hep-lat/9311007 [hep-lat].

[26] S. Duane et al. “Hybrid Monte Carlo”. In: *Phys. Lett.* B195 (1987), pp. 216–222. DOI: 10.1016/0370-2693(87)91197-X.

[27] I.P. Omelyan, I.M. Mryglod, and R. Folk. “Symplectic analytically integrable decomposition algorithms: classification, derivation, and application to molecular dynamics, quantum and celestial mechanics simulations”. In: *Comput. Phys. Commun.* 151.3 (2003), pp. 272–314. ISSN: 0010-4655. DOI: 10.1016/S0010-4655(02)00754-3.

[28] Tetsuya Takaishi. “Choice of integrator in the hybrid Monte Carlo algorithm”. In: *Comput. Phys. Commun.* 133 (2000), pp. 6–17. DOI: 10.1016/S0010-4655(00)00161-2. arXiv: hep-lat/9909134 [hep-lat].

[29] Kristopher Tapp. *Differential Geometry of Curves and Surfaces*. Undergraduate Texts in Mathematics. Springer International Publishing, 2016. ISBN: 978-3-319-39799-3. DOI: 10.1007/978-3-319-39799-3.

[30] Thomas A. DeGrand and Pietro Rossi. “Conditioning Techniques for Dynamical Fermions”. In: *Comput. Phys. Commun.* 60 (1990), pp. 211–214. DOI: 10.1016/0010-4655(90)90006-M.

[31] J. C. Sexton and D. H. Weingarten. “Hamiltonian evolution for the hybrid Monte Carlo algorithm”. In: *Nucl. Phys.* B380 (1992), pp. 665–677. DOI: 10.1016/0550-3213(92)90263-B.

[32] Taylor Haar et al. “Applying polynomial filtering to mass preconditioned Hybrid Monte Carlo”. In: *Comput. Phys. Commun.* 215 (2017), pp. 113–127. DOI: 10.1016/j.cpc.2017.02.020. arXiv: 1609.02652 [hep-lat].

[33] Mike J. Peardon and James Sexton. “Multiple molecular dynamics time scales in hybrid Monte Carlo fermion simulations”. In: *Nucl. Phys. Proc. Suppl.* 119 (2003), pp. 985–987. DOI: 10.1016/S0920-5632(03)01738-9. arXiv: hep-lat/0209037 [hep-lat].

[34] M. Hasenbusch and K. Jansen. “Speeding up lattice QCD simulations with clover improved Wilson fermions”. In: *Nucl. Phys.* B659 (2003), pp. 299–320. DOI: 10.1016/S0550-3213(03)00227-X. arXiv: hep-lat/0211042 [hep-lat].

[35] A. Ukawa. “Computational cost of full QCD simulations experienced by CP-PACS and JLQCD Collaborations”. In: *Nucl. Phys. Proc. Suppl.* 106 (2002), pp. 195–196. DOI: 10.1016/S0920-5632(01)01662-0.

[36] C. Urbach et al. “HMC algorithm with multiple time scale integration and mass preconditioning”. In: *Comput. Phys. Commun.* 174 (2006), pp. 87–98. DOI: 10.1016/j.cpc.2005.08.006. arXiv: hep-lat/0506011 [hep-lat].

[37] Waseem Kamleh and Mike Peardon. “Polynomial Filtered HMC: An Algorithm for lattice QCD with dynamical quarks”. In: *Comput. Phys. Commun.* 183 (2012), pp. 1993–2000. DOI: 10.1016/j.cpc.2012.05.002. arXiv: 1106.5625 [hep-lat].

[38] M.R. Hestenes and E. Stiefel. “Methods of Conjugate Gradients for Solving Linear Systems”. In: *Journal of Research of the National Bureau of Standards* 49.6 (1952), 409–436. ISSN: 0091-0635. DOI: 10.6028/jres.049.044.

[39] H. van der Vorst. “Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 13.2 (1992), pp. 631–644. DOI: 10.1137/0913035.

[40] S. Eisenstat, H. Elman, and M. Schultz. “Variational Iterative Methods for Nonsymmetric Systems of Linear Equations”. In: *SIAM Journal on Numerical Analysis* 20.2 (1983), pp. 345–357. DOI: 10.1137/0720023.

[41] Y. Saad and M. Schultz. “GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems”. In: *SIAM Journal on Scientific and Statistical Computing* 7.3 (1986), pp. 856–869. DOI: 10.1137/0907058.

[42] M. A. Clark et al. “Solving Lattice QCD systems of equations using mixed precision solvers on GPUs”. In: *Comput. Phys. Commun.* 181 (2010), pp. 1517–1528. DOI: 10.1016/j.cpc.2010.05.002. arXiv: 0911.3191 [hep-lat].

[43] R. C. Brower et al. “Chronological inversion method for the Dirac matrix in hybrid Monte Carlo”. In: *Nucl. Phys.* B484 (1997), pp. 353–374. DOI: 10.1016/S0550-3213(96)00579-2. arXiv: hep-lat/9509012 [hep-lat].

[44] Beat Jegerlehner. “Krylov space solvers for shifted linear systems”. In: (1996). arXiv: hep-lat/9612014 [hep-lat].

[45] M.A. Clark and A.D. Kennedy. “The RHMC algorithm for two flavors of dynamical staggered fermions”. In: *Nucl. Phys. Proc. Suppl.* 129 (2004), pp. 850–852. DOI: 10.1016/S0920-5632(03)02732-4. arXiv: hep-lat/0309084 [hep-lat].

[46] Ting-Wai Chiu et al. “A Note on the Zolotarev optimal rational approximation for the overlap Dirac operator”. In: *Phys. Rev.* D66 (2002), p. 114502. DOI: 10.1103/PhysRevD.66.114502. arXiv: hep-lat/0206007 [hep-lat].

[47] M.A. Clark and A.D. Kennedy. “Accelerating dynamical fermion computations using the rational hybrid Monte Carlo (RHMC) algorithm with multiple pseudofermion fields”. In: *Phys. Rev. Lett.* 98 (2007), p. 051601. DOI: 10.1103/PhysRevLett.98.051601. arXiv: hep-lat/0608015 [hep-lat].

[48] Taylor Ryan Haar, Yoshifumi Nakamura, and Hinnerk Stüben. “An update on the BQCD Hybrid Monte Carlo program”. In: *Proceedings, 35th International Symposium on Lattice Field Theory (Lattice 2017): Granada, Spain, June 18-24, 2017*. Vol. 175. 2018, p. 14011. DOI: 10.1051/epjconf/201817514011. arXiv: 1711.03836 [hep-lat].

[49] A. Ali Khan et al. “Accelerating the hybrid Monte Carlo algorithm”. In: *Phys. Lett.* B564 (2003), pp. 235–240. DOI: 10.1016/S0370-2693(03)00703-2. arXiv: hep-lat/0303026 [hep-lat].

[50] Mattia Bruno et al. “Simulation of QCD with $N_f = 2 + 1$ flavors of non-perturbatively improved Wilson fermions”. In: *JHEP* 02 (2015), p. 43. DOI: 10.1007/JHEP02(2015)043. arXiv: 1411.3982 [hep-lat].

[51] R. Arthur et al. “Domain Wall QCD with Near-Physical Pions”. In: *Phys. Rev.* D87 (2013), p. 094514. DOI: 10.1103/PhysRevD.87.094514. arXiv: 1208.4412 [hep-lat].

[52] Taylor Haar et al. “Single flavour optimisations to Hybrid Monte Carlo”. In: (2018). arXiv: 1806.04350 [hep-lat].

[53] Sourendu Gupta et al. “The Acceptance Probability in the Hybrid Monte Carlo Method”. In: *Phys. Lett.* B242 (1990), pp. 437–443. DOI: 10.1016/0370-2693(90)91790-I.

[54] G. S. Bali et al. “Nucleon mass and sigma term from lattice QCD with two light fermion flavors”. In: *Nucl. Phys.* B866 (2013), pp. 1–25. DOI: 10.1016/j.nuclphysb.2012.08.009. arXiv: 1206.7034 [hep-lat].

[55] M. A. Clark and A. D. Kennedy. “Accelerating dynamical fermion computations using the rational hybrid Monte Carlo (RHMC) algorithm with multiple pseudofermion fields”. In: *Phys. Rev. Lett.* 98 (2007), p. 051601. DOI: 10.1103/PhysRevLett.98.051601. arXiv: hep-lat/0608015 [hep-lat].

[56] Masashi Hayakawa and Shunpei Uno. “QED in finite volume and finite size scaling effect on electromagnetic properties of hadrons”. In: *Prog. Theor. Phys.* 120 (2008), pp. 413–441. DOI: 10.1143/PTP.120.413. arXiv: 0804.2044 [hep-ph].

[57] Sz. Borsanyi et al. “Ab initio calculation of the neutron-proton mass difference”. In: *Science* 347 (2015), pp. 1452–1455. DOI: 10.1126/science.1257050. arXiv: 1406.4088 [hep-lat].

[58] R. Horsley et al. “Isospin splittings of meson and baryon masses from three-flavor lattice QCD + QED”. In: *J. Phys.* G43.10 (2016), 10LT02. DOI: 10.1088/0954-3899/43/10/10LT02. arXiv: 1508.06401 [hep-lat].

[59] N. Cundy et al. “Non-perturbative improvement of stout-smeared three flavour clover fermions”. In: *Phys. Rev.* D79 (2009), p. 094507. DOI: 10.1103/PhysRevD.79.094507. arXiv: 0901.3302 [hep-lat].