BERGISCHE UNIVERSITÄT WUPPERTAL

Università degli Studi di Ferrara

Doctoral Thesis in Physics

Multigrid Multilevel Monte Carlo Approaches for Trace Estimation in Lattice QCD

Author:
Mostafa Nasr Khalil

Supervisor:
Prof. Andreas Frommer

Supervisor:
Prof. Raffaele Tripiccione

Supervisor:
Prof. Constantia Alexandrou

Supervisor:
Prof. Sebastiano Schifano

Monday 19th June, 2023

Acknowledgments

First of all, I would like to thank my co-supervisors, Prof. Tripiccione, Prof. Schifano, Prof. Alexandrou, and Prof. Frommer, for helping me with my PhD and giving me advice. I'm especially thankful for Prof. Tripiccione, who was always friendly and helpful but unfortunately is no longer with us. I also want to thank Prof. Schifano for stepping in afterward and continuing to give very helpful guidance. To Prof. Alexandrou for all her help and guidance, and especially for her significant role in making STIMULATE possible.

To Prof. Frommer, who made me feel welcome in his group, creating an inspiring work environment, and providing me with helpful and invaluable guidance throughout the various research aspects. In addition to his deep knowledge, his humbleness, kindness, mentorship, care and willingness to help students all the time is always inspiring to me.

I also want to thank my coleagues in our work group for engaging discussion and providing sound advice. To Daniela, who helped me almost all the time, to Artur, who provided me with the Matlab version of DDαAMG package to use in LQCD experiments and answered my specific questions, and to Gustavo, who was and still is a huge help, especially at the beginning of my PhD. They have made my research work much more enjoyable.

I wish to thank my closest family members, including my parents, siblings, and wife, for their priceless and unwavering support.

Lastly, I would like to acknowledge the members of the physics group at Wuppertal led by Francesco Knechtli for providing me with configurations for the numerical tests and also for organizing seminars related to my research.

Abstract

This thesis addresses the challenging problem of solving large systems of linear equations that arise from the discretization of quantum chromodynamics on the Lattice. The thesis contributes to overcome this challenge by improving stochastic approaches for computing the trace of the inverse, which represents the disconnected loops contributions for certain relevant observables. The main objective is to improve the accuracy of the trace estimator and reduce the computational cost through variance reduction techniques.

The main idea is to take advantage of the multilevel hierarchy in the multigrid solves for the linear systems also for trace estimation. To achieve this goal, we propose a novel stochastic method called multigrid multilevel Monte Carlo (MG-MLMC), which merges both standard multigrid (MG) and multilevel Monte Carlo (MLMC) methods. We also introduce a new trace estimator technique, called multigrid multilevel Monte Carlo++ (MG-MLMC++), which combines the Hutch++ method, a recent inexact deflation technique, with the MG-MLMC approach.

These contributions provide advancements in the field of lattice QCD, improving the accuracy and efficiency of numerical simulations, which are crucial for understanding the behavior of subatomic particles. The methods are designed to handle the increasingly large and ill-conditioned systems of linear equations that arise from the discretization of the QCD equations on a lattice and can be applied in all situations where one has to compute the trace of the inverse of a matrix and when an efficient multigrid hierarchy can be constructed.

Foreword

The work presented in this thesis is in parts based on the following publications:

- A. Frommer, M. N. Khalil, and G. Ramirez-Hidalgo, *A multilevel approach to variance reduction in the stochastic estimation of the trace of a matrix*, SIAM Journal on Scientific Computing, 44 (2022), pp. A2536–A2556
- M. Khalil and A. Frommer, *MGMLMC++ as a Variance Reduction Method for Estimating the Trace of a Matrix Inverse*, PoS, LATTICE2022 (2022), p. 017

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 765048.

We computed all numerical results in this thesis on two machines in our group: *aicomp03* and *aicomp04*.

Contents

Chapter 1

Introduction

Quantum chromodynamics (QCD) is a fundamental theory that describes the interactions of quarks and gluons, which are the constituents of hadrons [19, 52]. However, confinement makes it impossible to isolate quarks in nature, so analytic calculations in QCD typically rely on studying hadrons, which are composed of quarks and are not subject to confinement. Although perturbative expansions are often used in physics calculations, they are not always applicable when dealing with QCD at specific energy levels, so alternative numerical and computational methods must be used.

Quantum chromodynamics on the Lattice (Lattice QCD) is a rapidly growing field in theoretical physics that seeks to understand the properties of subatomic particles. One of the main challenges in Lattice QCD is to accurately determine the properties of these particles, such as their mass and spin, through numerical simulations. To address this challenge, inversion techniques are commonly used to solve large systems of linear equations that arise from the discretization of the QCD equations on a Lattice. The Lattice QCD method involves discretizing the continuous QCD theory on a four-dimensional Lattice using Wick rotations to simulate the theory on a discretized Euclidean space-time [105]. However, this approach presents significant computational and mathematical challenges and is regarded as one of the most demanding computational problems in the world [12, 53]. Nonetheless, Lattice QCD has been successful in generating results that closely match experimental observations.

Solving linear systems of equations is a core aspect of simulating QCD on the Lattice, but this task becomes increasingly difficult as the Lattice parameters are adjusted to match the continuum theory, resulting in more ill-conditioned systems of larger coefficient matrices. This phenomenon is called "critical slowing down," and requires a combination of numerical linear algebra and high-performance

computing methods to overcome. Although traditional methods, such as odd-even preconditioning [26, 32], deflation [68], and domain decomposition [42, 67], have been used to address this challenge, critical slowing still impacts them. Multigrid methods offer a promising alternative as they can converge independently of the conditioning of the linear system. However, using geometric multigrid methods based on underlying PDEs has been difficult in the past due to the random nature of matrices in Lattice QCD simulations.

This thesis contributes to the advancement of Lattice QCD by improving stochastic approaches for computing the trace of the inverse. The trace represents the disconnected loop contributions for certain relevant observables. The main objective is to improve the accuracy of the trace estimator, reduce computational time, and optimize the selection of inversion algorithms. The main idea is to take advantage of the multilevel hierarchy in the multigrid solves for the linear systems also for trace estimation.

Our first contribution is the proposal of a new stochastic method named multigrid multilevel Monte Carlo (MG-MLMC) [41], which merges both standard multigrid and multilevel Monte Carlo methods. The second contribution is a new trace estimator technique, called multigrid multilevel Monte Carlo ++ (MG-MLMC++) [63], which combines the Hutch++ method, a recent inexact deflation technique [70], with the MG-MLMC approach.

The new methods can be applied in all situations where one has to compute the trace of the inverse of a matrix and when an efficient multigrid hierarchy can be constructed.

Our contributions provide in particular advancements in the field of Lattice QCD, as they improve the accuracy and efficiency of numerical simulations, which are crucial for understanding the behavior of subatomic particles. Our methods are designed to handle the increasingly large and ill-conditioned systems of linear equations that arise from the discretization of the QCD equations on the Lattice and can be applied to a wide range of problems in theoretical physics.

We structure this thesis as follows:

In Chapter 2, we introduce important mathematical definitions and concepts used in this thesis. This includes a review of linear algebra, linear systems of equations, and the theory of stochastic processes. We will also cover the estimation of traces using random sampling.

In Chapter 3, we give an overview of QCD on the Lattice and its significance in theoretical physics. We explain the basic concepts of the theory, including the use of Lattice discretization to regulate the theory and the techniques used to extract physical observables from the Lattice.

In Chapter 4, we focus on linear solvers, which are used to solve the large, sparse linear systems of equations.We will introduce different types of solvers and discuss their advantages and disadvantages.

In Chapter 5, we concentrate on stochastic trace estimation, which is a technique used to estimate the trace of an operator. We will explore different methods for performing the estimation and the sources of error in the process.

In Chapter 6, we introduce variance reduction methods, which are essential in this thesis to decrease the error in stochastic trace estimation. We will discuss different techniques, such as deflation, Hutch++ and Adaptive Hutch++ approaches in detail.

In Chapter 7, we introduce the novel concept of MG-MLMC and MG-MLMC++. We will discuss the principles of the methods and their advantages over deflated Hutchinson and plain Hutchinson techniques.

In Chapter 8, we discuss the stopping criteria in the MG-MLMC and MG-MLMC++ approaches. These techniques are used to control when to stop the stochastic estimation process to achieve a balance between precision and computational cost. We will examine different methods for determining when to stop the stochastic estimation process and their advantages and disadvantages.

Finally, in Chapter 9, we present numerical results that demonstrate the effectiveness of the proposed stochastic approaches for trace estimation. These results include a comparison of the performance of different linear solvers and variance reduction methods, as well as a detailed analysis of the behavior of MG-MLMC techniques for standard examples and in the context of QCD simulations.

Chapter 2

Mathematical definitions

In this chapter, we will introduce some important definitions of linear algebra and statistics that are used throughout the thesis. These definitions include concepts for matrices, such as the matrix inverse and the trace of a matrix, as well as concepts such as eigenvalues and eigenvalue decomposition. We will also discuss the singular value decomposition and the matrix condition number. In addition, we will define key concepts in probability and statistics, including probability functions, random vectors, expected value, variance, and estimators. Finally, we will introduce the root mean square error (RMSE), which is a commonly used measure of accuracy in the algorithms.

All of the definitions discussed in this chapter can be found in the following references: [48, 51, 58, 71, 87, 88, 98]

2.1 Notation

In this thesis, all vector spaces and matrices are assumed to be complex and the Euclidean norm $\|\cdot\|$ is used for vector norms without a subscript. The iteration index is denoted by an upper letter $\square^{(k)}$ within parentheses. In general, lowercase letters represent vectors or scalars, while uppercase letters represent matrices. The exception to this is the γ matrices, which are also lowercase letters.

The following notations and abbreviations are used throughout the thesis:

n	problem size
$A = [a_{i,j}]$	the matrix A consists of the elements $a_{i,j}$
$A = [a_1 \vert a_2 \vert \cdots \vert a_n]$	the matrix A consists of vectors $a_1, a_2, \ldots, a_n$ as columns
A^{-1}	inverse of the matrix A
$V = [v_1 \vert \cdots \vert v_k]$	column matrix from a set of k vectors
A^{-H}	short-hand notation $A^{-H} = (A^{-1})^H = (A^H)^{-1}$
$\mathcal{D}$	continuous Dirac operator
$D, D(m), \mathcal{D}_W(m)$ $D_W(m)$	(lattice) Wilson-Dirac operator (with mass m)
D_N	Neuberger overlap operator
γ_i	generator matrices of the Clifford algebra, $i \in \{1, \ldots, 4\}$
γ_5	$\gamma_5 := \gamma_1\gamma_2\gamma_3\gamma_4$
Γ_5	lattice version of γ_5
λ	eigenvalue
Λ	diagonal matrix of eigenvalues
$X = [x_1 \vert \cdots \vert x_n]$	matrix of eigenvectors
$\mathrm{spec}(A)$	spectrum of A
$\mathrm{span}\{\cdots\}$	space of the linear combinations of the vectors or matrices
$\mathrm{tr}(A)$	the trace of the matrix A
$\mathbb{V}$	the variance
ϵ	the relative accuracy

Table 2.1: Notations and abbreviations

The column vector $x \in R^n$ will be denoted as

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

where the elements x_1, x_2, …, x_n are the entries of the column vector.

The matrix $A \in R^{n \times m}$ can be represented in matrix form as:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \ldots & a_{1m} \\ a_{21} & a_{22} & a_{23} & \ldots & a_{2m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \ldots & a_{nm} \end{bmatrix}$$

Here, each element a_{ij} is located in the $i-th$ row and $j-th$ column of the matrix A. The number of rows is n and the number of columns is m.

2.2 Basic definitions

2.2.1 Linear Algebra

The main focus of this thesis is estimating the trace of the matrix inverse that require solving linear systems of equations, which can be represented by matrices. There are characteristics of special matrices with a specific structure that we can take advantage of to create effective algorithms.

Definition 2.1 (Special matrices)**.**
Assume a nonsingular matrix $A \in \mathbb{C}^{n\times n}$, we call A:

1. *sparse, if the number of non-zero entries per row is significantly smaller than n and independent of n.*
2. *symmetric, if $A = A^T$,*
3. *positive definite if $x^T Ax > 0 \quad \forall \quad x \in \mathbb{C}^n, x \neq 0$*
4. *diagonal, if $a_{ij} = 0 \quad \forall \quad i \neq j$.*
5. *Hermitian, if $A = A^H$,*
6. *unitary, if $A^H A = I$,*
7. *normal, if $A^H A = AA^H$,*
8. *a projection, if $A^2 = A$,*
9. *diagonally dominant, if $|a_{ii}| > \sum_{j\neq i} |a_{ij}|$ for $j = 1, \ldots, n$*

We also define:

- *If a matrix A is a projection matrix, then the matrix $(I - A)$ is also a projection matrix, as*

$$(I - A)^2 = I - 2A + A^2 = I - 2A + A = I - A. \tag{2.1}$$

Definitions 2.2, 2.3, 2.4 and 2.5 are introduced in order to provide a foundation for estimating the trace of the inverse of a matrix.

Definition 2.2 (Matrix inverse)**.**
A square matrix A is invertible or non-singular if it has an inverse, which means there exists a matrix B such that $AB = BA = I$. In this case, the matrix B is the inverse of A and is denoted as A^{-1}. It is important to note that not all matrices have an inverse. The following statements are equivalent to a matrix A being non-singular:

- *The determinant of A is non-zero, $det(A) \neq 0$.*
- *The columns of A, denoted $\mathbf{a}_j$, are linearly independent.*
- *The equation $Ax = \mathbf{0}$ has only one solution, $x = \mathbf{0}$.*
- *All eigenvalues of A, denoted λ_i, are non-zero.*

Eigenvalue decomposition: The inverse of a matrix can be computed using its eigenvalue decomposition.

Definition 2.3 (Trace of a matrix)**.**
The trace of a square matrix A is the sum of its diagonal entries, $\mathrm{tr}(A) = \sum_i^n a_{ii}$. For any $n \times n$ matrices A, B, and C, and a scalar α the properties of the trace can be given as follows:

- $\mathrm{tr}(\alpha A + B) = \alpha \mathrm{tr}(A) + \mathrm{tr}(B)$.
- $\mathrm{tr}(AB) = \mathrm{tr}(BA)$.
- $\mathrm{tr}(A) = \sum_{i=1}^n (\lambda_i)$.
- *The trace of a matrix is invariant under cyclic permutations of matrix products, i.e.,* $\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CAB)$.
- *The trace of a matrix is invariant under similarity transformations, i.e.,* $\mathrm{tr}(A) = \mathrm{tr}(PAP^{-1})$ *for any invertible matrix P.*

Definition 2.4 (Vector norm)**.**
A vector norm is a way of measuring the size or length of a vector. For a vector $x \in \mathbb{C}^n$, some widely known norms are:

- *Two-norm: $\|x\|_2$, also known as the Euclidean norm, is defined as*

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2} = \langle x|x \rangle^{1/2} \tag{2.2}$$

- *ρ-norm: $\|x\|_\rho$ is defined as*

$$\|x\|_\rho = \left(\sum_{i=1}^{n} |x_i|^\rho \right)^{1/\rho}. \tag{2.3}$$

- *Infinity norm: $\|x\|_\infty$ is defined as*

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|. \tag{2.4}$$

Definition 2.5 (Matrix norm)**.**
A matrix norm is a way of measuring the size or length of a matrix. For a matrix $A \in \mathbb{C}^{n\times m}$*, some useful norms are:*

- *Induced norm:* $\|A\|_\rho$*, which is based on the way that the matrix* A *transforms a vector* x*. It is defined as*

$$\|A\|_\rho = \max_{\|x\|_\rho=1} \|Ax\|_\rho, \tag{2.5}$$

  where $\|\cdot\|_\rho$ *is a norm in space* $\mathbb{C}^n$

- *Frobenius norm:* $\|A\|_F$*, which does not rely on a vector norm and uses all the components of the matrix. It is defined as*

$$\|A\|_F = \sqrt{\left(\sum_{i=1}^{n}\sum_{j=1}^{m} |a_{ij}|^2\right)}. \tag{2.6}$$

All the previous matrix norms have the following properties:

- $\|A\| \geq 0, \|A\| = 0$ *iff* $A = 0$
- $\|\alpha A\| = |\alpha| A, \alpha \in \mathbb{C}$
- $\|A + B\| \leq \|A\| + \|B\|$
- $\|AB\| \leq \|A\|\|B\|$

We will analyze the computational cost of various algorithms presented in chapters 6, 7 and 8. To do this, we will use a cost model that takes into account not just the number of matrix-vector multiplications (mvms) performed, but also the runtime of the algorithms. Our goal is to find some theoretical explanations for the differences in runtime that we measure for the different methods.

Definition 2.6 (Cost model)**.**
Let $x, y \in \mathbb{C}^n$ *and* $\alpha \in \mathbb{C}$ *be scalars. We will define the following types of operations as elementary vector operations [51]:*

- *axpy:* $y \leftarrow \alpha x + y$*, which performs a vector addition of the scaled vector* αx *to* y*,*
- *scaling:* $y \leftarrow \alpha y$*, which scales the vector* y *by* α*,*
- *assigning:* $y \leftarrow x$*, which copies the contents of* x *to* y*,*
- *swapping:* $y \leftrightarrow x$*, which exchanges the contents of* x *and* y*, and*

- *dot-product:* $\alpha \leftarrow y^H x$*, which computes the dot product between* x *and* y*.*

Using these elementary vector operations, we can describe the cost model for matrix-vector products as follows. Let $x, y \in \mathbb{C}^n$ *and* $A \in \mathbb{C}^{n \times n}$ *be a square matrix. We define the following operations on* x *and* y *as the basic building blocks for the computation of* Ax*:*

- *matrix-vector multiplication:* $y \leftarrow Ax$*, which computes the product of the matrix* A *with the vector* x *using* n^2 *complex scalar operations.*

In our cost model, we assume that all vector operations in Definition 2.6 have the same computational cost of n floating point operations. This assumption is reasonable, especially on modern hardware that supports fused multiply-add floating point operations. When discussing the number of operations in methods involving a matrix $A \in \mathbb{C}^{n \times n}$, we only consider vector operations of size n.

Given that eigenvalues and eigenvectors are crucial concepts for Chapters 6, 7 and 8 we provide a definition that highlights a few concepts of an eigenvalue.

Definition 2.7 (Eigenvalue)**.**
Given a square matrix $A \in \mathbb{C}^{n \times n}$ *we call* $\lambda \in \mathbb{C}$ *an* eigenvalue *of* A *if there exists a nonzero vector* $x \in \mathbb{C}^n$ *such that*

$$Ax = \lambda x. \tag{2.7}$$

Additional characteristics and terms related to eigenvalues:

- x *is called an* eigenvector (belonging to λ)*.*
- *A pair* (λ, x) *of eigenvalue* λ *and its eigenvector* x *is called an* eigenpair*.*
- *The set of all eigenvalues of* A *is called the* spectrum *of* A *and is denoted by* $\operatorname{spec}(A)$*.*
- *The* spectral radius of A *is defined as* $\rho(A) := \max_{\lambda \in \operatorname{spec}(A)}(|\lambda|)$*.*
- *Eigenvalues* λ_i *are the roots of the characteristic polynomial of* A*, i.e.,* $p_A(\lambda) := \det(A - \lambda I)$*.*
- *The multiplicity* m_i *of an eigenvalue in* $p_A(\lambda)$ *is called* algebraic multiplicity *of* λ*.*
- *The* geometric multiplicity *is denoted by* g_i *and is the dimension of the null space of* $(A - \lambda_i I)$*.*

Definition 2.8 (Eigenvalue decomposition)**.**
A square matrix $A \in \mathbb{C}^{n\times n}$ is called diagonalizable *if and only if $g_i = m_i$ for all $\lambda_i \in \Lambda$. We define in this case the* eigenvalue decomposition

$$A = X\Lambda X^{-1}, \tag{2.8}$$

where each column x_i of X contains an eigenvector of A belonging to the eigenvalue $\Lambda_{i,i} = \lambda_i$ of the diagonal matrix Λ.

The singular value decomposition is a generalization of an eigenvalue decomposition to non-square matrices. It allows us to decompose a matrix into its constituent parts, much like an eigenvalue decomposition does for square matrices, and can be defined as follows:

Definition 2.9 (Singular value decomposition)**.**
Given a matrix $A \in \mathbb{C}^{m\times n}$ there exists a matrix decomposition

$$A = U\Sigma V^H, \tag{2.9}$$

where $U \in \mathbb{C}^{m\times m}$ and $V \in \mathbb{C}^{n\times n}$ are unitary matrices and $\Sigma \in \mathbb{C}^{m\times n}$ is a diagonal matrix with non-negative diagonal entries $\sigma_{i,i}$, this decomposition is called the singular value decomposition (SVD). We call the column vectors u_i and v_i left *and* right singular vectors (of $\sigma_{i,i}$)*.*

Definition 2.10 (Power method)**.**
Let A be a matrix with eigenvalues $0 \leq |\lambda_1| \leq \cdots |\lambda_{n-1}| < |\lambda_n|$ and let $v^{(0)}$ be an initial vector such that $\langle v^{(0)}, x_n\rangle \neq 0$. The power method is an iterative procedure that generates a sequence of vectors $v^{(0)}, v^{(1)}, v^{(2)}, \ldots$ and corresponding Rayleigh quotients $\lambda^{(0)}, \lambda^{(1)}, \lambda^{(2)}, \ldots$ as follows:

1. $k \leftarrow 0$.
2. $w \leftarrow Av^{(k)}$.
3. $v^{(k+1)} \leftarrow w/|w|$.
4. $\lambda^{(k+1)} \leftarrow {v^{(k+1)}}^T Av^{(k+1)}$.

If the desired convergence criterion is satisfied, return the approximations $(\lambda^{(k)}, v^{(k)})$ of the largest eigenpair of A. Otherwise, set $k \leftarrow k+1$ and go back to step 2. It can be shown that the sequence of vectors $v^{(k)}$ produced by the power method satisfies

$$v^{(k)} \to x_n, \text{ as } k \to \infty,$$

if we assume $v^{(k)}$ to be normalized after each step. Furthermore, $v^{(k)}$ and its corresponding Rayleigh quotient $\lambda^{(k)}$ satisfy

$$|v^{(k)} - x_n| = \mathcal{O}\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^k\right) \text{ and } |\lambda^{(k)} - \lambda_n| = \mathcal{O}\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|^{2k}\right).$$

The power method can also be modified to find eigenpairs corresponding to other regions of the spectrum by shifting and inverting the matrix A. Specifically, choosing a shift σ sufficiently close the target eigenvalue λ_J, the power method for $(A - \sigma I)^{-1}$ converges to $\frac{1}{\lambda_j - \sigma}$

Definition 2.11 (Condition Number)**.**
The condition number of a matrix A is a way of describing how well or badly the system $Ax = b$ could be approximated and defined as:

$$\kappa(A) = |A|_2 \cdot |A^{-1}|_2,$$

If A is singular, then $\kappa(A) = \infty$. If $\kappa(A)$ is small, the problem is well-conditioned, and if $\kappa(A)$ is large, the problem is rather ill-conditioned. Another expression for the condition number is $\kappa(A) = \sigma_{\max}/\sigma_{\min}$, where $\sigma_{\max}$ and $\sigma_{\min}$ are the maximal and minimal singular values of matrix A. If A is a symmetric matrix, then $\kappa(A) = |\lambda_{\max}/\lambda_{\min}|$, where $\lambda_{\max}$ and $\lambda_{\min}$ denote the largest and smallest eigenvalues of A.

2.2.2 Probability theory

We use the definitions that follow because they are all crucial and central to all of the core algorithms described in this thesis. We only consider discrete stochastic variables here.

Consider two random variable X, Y which take values x_i, y_i where $i = 1, \ldots, n$ for X and Y respectively.

Definition 2.12 (Independent variables)**.**
We call two random variables X, Y are independent if and only if any two of their possible outcomes, x and y hold that:

$$\Pr(X = x \text{ and } Y = y) = \Pr(X = x) \Pr(Y = y). \tag{2.10}$$

Definition 2.13 (Probability function)**.**
The probability $f(x_i)$ of a random variable X can be represented by $\Pr(X = x_i) = \Pr_i = f(x_i)$ according to the axioms of Kolmogorov [16]. These axioms state that:

- *$f(x_i)$ must be nonnegative*
- *The sum of all possible values of $f(x_i)$ equals 1*
- *The probability of the random variable being equal to either x_i or x_j is the sum of their individual probabilities (i.e.* $\Pr(X = x_i \text{ or } X = x_j) = f(x_i) + f(x_j)$*).*

For discrete variables, the following are the three most basic statistics quantities:

Definition 2.14 (Random vectors)**.**
The fundamental concepts of statistics are generalized naturally to random vectors [102]. A column vector $X = (X_1, ...X_n)^T$ with entries $X_i \in \mathbb{C}$ that are random variables is called a random vector.

The variance of a random variable X is a measure of the dispersion or spread of its possible values around the expected value. It is defined as follows:

Definition 2.15 (Expected value)**.**
For a random variable X, the expected value $\mathbb{E}[X]$ is defined as the weighted average of all possible outcomes x_j with probability p_j:

$$\mathbb{E}[X] = \sum_j x_j p_j. \tag{2.11}$$

If all outcomes are equally likely, i.e., the probability $f(x)$ is the same for all possible values of x, then the expected value is equal to the mean μ of the distribution:

$$\mathbb{E}[X] = \mu = \frac{1}{N} \sum_j x_j, \tag{2.12}$$

where N is the number of events. For a random vector $\mathbf{X} = (X_1, X_2, \ldots, X_n)$, the expected value can be represented as a column vector with each component being the expected value of each random variable X_i:

$$\mathbb{E}[\mathbf{X}] = \begin{pmatrix} \mathbb{E}[X_1] & \mathbb{E}[X_2] & \cdots & \mathbb{E}[X_n] \end{pmatrix}. \tag{2.13}$$

Definition 2.16 (Variance)**.**
The variance of a random variable X, denoted as $Var[X]$ or $\mathbb{V}[X]$, is given by the formula:

$$\mathbb{V}[X] = \mathbb{E}[(X - \mathbb{E}[X])^2], \tag{2.14}$$

where $\mathbb{E}[X]$ is the expected value of X, and $\mathbb{E}[(X - \mathbb{E}[X])^2]$ is the expected value of the squared difference between X and its expected value.

Definition 2.17 (Estimator)**.**
An estimator is a statistical tool that allows us to estimate an unknown population parameter using sample data. A common property that we seek in estimators is unbiasedness, which means that the expected value of the estimator is equal to the true value of the population parameter being estimated. Mathematically, an unbiased estimator $\hat{\theta}$ of a population with expected value θ satisfies (2.15)*:*

$$\mathbb{E}(\hat{\theta}) = \theta, \tag{2.15}$$

where $\mathbb{E}(\hat{\theta})$ is the expected value of the estimator and θ is the true value of the population parameter.

Unbiased estimators are characterized by consistency, which means that the estimator becomes more accurate as the sample size increases, i.e., $\lim_{n\to\infty} \mathbb{E}(\hat{\theta}) = \theta$. We are interested in estimators which have small variance as in (2.16) *(or even the samllest possible variance) for the same population parameter.*

$$\mathrm{Var}(\hat{\theta}) = \mathbb{E}(\hat{\theta}^2) - (\mathbb{E}(\hat{\theta}))^2 \tag{2.16}$$

Definition 2.18 (RMSE)**.**
The Root Mean Squared Error (RMSE) is a measure of the difference between a predicted value and the true value. It is widely used as a measure of the accuracy of algorithms that predict. It is defined as the square root of the Mean Squared Error (MSE), which is the average of the squared differences between the predicted values and the true values and is given by:

$$\mathrm{RMSE}(\hat{\theta}) = \sqrt{\mathrm{MSE}(\hat{\theta})} = \sqrt{\mathbb{E}((\hat{\theta} - \theta)^2)}, \tag{2.17}$$

where $\hat{\theta}$ is the predicted value and θ is the true value.

For an unbiased estimator, the RMSE is equal to the standard deviation, which is a measure of the dispersion of a distribution. In this case, the RMSE is given by:

$$\mathrm{RMSE}(\hat{\theta}) = \sqrt{\mathbb{V}[\hat{\theta}]}, \tag{2.18}$$

where $\mathbb{V}[\hat{\theta}]$ is the variance of the predicted values.

Chapter 3

Quantum chromodynamics on the Lattice

In this Chapter, we explain the conceptual foundation and background physics behind the methods discussed in subsequent chapters of this thesis. It is important to understand the connection between new algorithmic developments in applied mathematics and the underlying physical systems in order to gain a better intuition of the limitations and expected behavior of these algorithms. This is particularly relevant in the context of Lattice quantum chromodynamics. Additionally, having a clear understanding of the underlying physics is crucial for understanding the motivation behind the implementations and new developments in this field.

We begin by providing an overview of the QCD and its role in Standard Model in Sec. 3.1. Then, we explore the concept of continuum QCD in Sec. 3.2, which describes the theory in the limit of infinite space and time. Next, we delve into the specific methods used to study QCD on the Lattice in Sec. 3.3, including the Wilson Dirac operator in Sec. 3.3.1 and Clover-improved Wilson Dirac operator in Sec.3.3.2.Lastly, we discuss the Schwinger model in Sec. 3.4, which is a simplified version of QCD that can be solved analytically.

The majority of the information presented in this Chapter comes from the following sources: [8, 9, 44, 62, 69, 78, 83, 96].

3.1 Overview of the Standard Model of particle physics

The Standard Model is a theory that describes the fundamental particles that comprise all known matter in the universe, as well as the forces that govern their

interactions [69]. It is predicated on the premise that all matter is composed of two types of elementary particles: fermions and bosons. Fermions, which include quarks and leptons, are the building components of matter and are characterized by quantum mechanics. Bosons, such as the photon, gluon, W, and Z bosons, are the carriers of the fundamental forces and are described by quantum field theory. The Standard Model can accurately represent a vast array of events, from the behavior of subatomic particles to the characteristics of atoms and molecules. However, it is not a complete theory and does not explain a number of crucial issues, such as the nature of dark matter and the observed matter-antimatter asymmetry in the universe. Particle physics is conducting continuing studies into these and other mysteries [96].

It is a gauge theory based on the gauge group $SU(3)_C \times SU(2)_L \times U(1)_Y$, that describes how elementary particles like fermions and bosons interact with each other via the exchange of gauge bosons. Spinor fields that transform under gauge group representations describe fermions, which are the building blocks of matter. Quarks and leptons are examples of this. Gauge bosons, on the other hand, such as the photon, gluon, W, and Z bosons, are described by vector fields that transform under the gauge group's adjoint representation. It is true that the Standard Model is able to accurately explain a lot of phenomena regarding subatomic particles, atoms, and molecules [78].

For example, one possible way to write the Lagrangian density for the Standard Model is as a sum of four terms, representing the four sub-theories that make up the Standard Model see [78, 96]:

$$\mathcal{L} = \mathcal{L}_{\text{QED}} + \mathcal{L}_{\text{QCD}} + \mathcal{L}_{\text{EW}} + \mathcal{L}_{\text{Higgs}}, \tag{3.1}$$

where:

- $\mathcal{L}_{\text{QED}}$ represents the Lagrangian density of quantum electrodynamics, which describes the electromagnetic force and the interactions of photons with charged particles.
- $\mathcal{L}_{\text{QCD}}$ represents the Lagrangian density of quantum chromodynamics, which describes the strong nuclear force and the interactions of gluons with quarks.
- $\mathcal{L}_{\text{EW}}$ represents the Lagrangian density of the electroweak theory, which describes the weak nuclear force and the interactions of the W and Z bosons with the matter.
- $\mathcal{L}_{\text{Higgs}}$ represents the Lagrangian density of the Higgs boson, which gives mass to the W and Z bosons and, in turn, to all other particles in the Standard Model through the Higgs mechanism.

It's important to note that the Lagrangian density is a mathematical construct, it is not a physical observable. There are many different ways to write the Lagrangian density for the Standard Model, and the specific form used depends on the context and the level of detail required.

3.2 Continuum QCD

The study of QCD is a fascinating and complex field within particle physics that delves into the interactions between quarks and gluons, the fundamental particles that make up protons and neutrons. To fully understand QCD, one must have a deep understanding of modern quantum mechanics, special relativity, and quantum field theory. In some cases, mathematical techniques such as the Taylor series or other asymptotic expansions can be used to simplify calculations, but other situations may require computational methods to fully grasp the model without any analytic approximations. The study of QCD encompasses many areas of physics and mathematics, including group theory, statistical mechanics, and renormalization of non-Abelian group theories. The Large Hadron Collider at CERN has played a vital role in experimental studies of QCD. This section aims to provide an overview of QCD from an applied mathematics perspective, outlining the key concepts and techniques required for simulating QCD on computers [77, 96].

The QCD Lagrangian describes the interactions of quarks and gluons as spin-$\frac{1}{2}$ fermions and vector gauge bosons, respectively. Quantum mechanical operators defined on a four-dimensional space-time describe the quark and gluon fields. These fields are denoted by $\Psi(x)$ and $A^{\mu}(x)$, where x represents a position in space-time and μ is an index ranging from 0 to 3, representing the four dimensions of space-time [66].

The Dirac equation for the skew-adjoint continuum describes this interaction [8, 28, 78, 96].

$$(D + mI)\psi = \eta, \tag{3.2}$$

where $\psi = \psi(x) \in \mathbb{C}^{12}$ and $\eta = \eta(x) \in \mathbb{C}^{12}$ are called *spinors* or *quark fields*. The twelve components $\psi_{c,\sigma}$ label the internal degrees of freedom, the so-called color $c = (1, 2, 3)$ and spin $\sigma = (0, 1, 2, 3)$ of a given spinor at a point $x = (x_0, x_1, x_2, x_3)$ in space-time. The ordering of the degrees of freedom of a spinor is given as

follows:

$$\begin{aligned}\psi(x) = (&\psi_{1,0}(x), \psi_{2,0}(x), \psi_{3,0}(x),\\ &\psi_{1,1}(x), \psi_{2,1}(x), \psi_{3,1}(x),\\ &\psi_{1,2}(x), \psi_{2,2}(x), \psi_{3,2}(x),\\ &\psi_{1,3}(x), \psi_{2,3}(x), \psi_{3,3}(x))^H\end{aligned}$$

The scalar parameter m sets the quark mass of the QCD theory. The Dirac operator $\mathcal{D}$ describes the interaction between the quarks for a given gluon background field and is defined as

$$\mathcal{D} = \sum_{\mu=0}^{3} \gamma_\mu \otimes (\partial_\mu + A_\mu), \tag{3.3}$$

where ∂_μ is a shorthand for $\partial/\partial x_\mu$. The Hermitian and unitary γ-matrices $\gamma_0, \gamma_1, \gamma_2, \gamma_3 \in \mathbb{C}^{4\times 4}$ generate a Clifford algebra [66], satisfying

$$\gamma_\mu\gamma_\nu + \gamma_\nu\gamma_\mu = \begin{cases} 2 \cdot I, & \mu = \nu \\ 0, & \mu \neq \nu \end{cases} \text{ for } \mu, \nu \in \{0, \ldots, 3\} \tag{3.4}$$

and act nontrivially on the spin indices of the spinor and trivially on the color indices:

$$(\gamma_\mu\psi)(x) = (\gamma_\mu \otimes I_3)\psi(x).$$

The gauge field, denoted by $A_\mu(x)$, describes the connection between infinitesimally close space-time points. It is defined by matrices from the Lie algebra of the special unitary group SU(3), namely the skew-Hermitian traceless matrices from SU(3). The gauge field acts trivially on the spin and non-trivially on the color, as represented by (3.5):

$$(A_\mu\psi)(x) = (I_4 \otimes A_\mu(x))\psi(x). \tag{3.5}$$

To ensure that the spinor field, denoted by $\psi(x)$, transforms convariantly under local gauge transformations, a minimal coupling extension of the derivative, referred to as the covariant derivative and denoted by $\partial_\mu + A_\mu$, is used. It is expressed as $((\partial_\mu + A_\mu)\psi)(x)$. In addition, the combination of the covariant derivative and the γ-matrices, denoted by $\mathcal{D}$, ensures that the spinor field transforms in the same way as the space-time transformations of special relativity. These principles of local gauge invariance and special relativity are fundamental to the standard model of elementary particle [39].

3.3 QCD on the Lattice

In Lattice quantum chromodynamics, a discretization of space-time is necessary for simulations. This is achieved by dividing the four-dimensional Euclidean space-time into a Lattice with, for example, periodic boundary conditions. The Lattice has a size of $n_t \times n_s^3$, where n_t is the number of Lattice points in the time dimension and n_s is the number of Lattice points in each dimension of space. The Lattice sites can be indexed by a four-tuple

$$x = (t, x, y, z) \in \mathcal{L} := (\mathbb{Z}/n_t\mathbb{Z}) \times (\mathbb{Z}/n_s\mathbb{Z})^3$$

The Dirac field, which describes the quarks, is defined on the Lattice sites and is typically denoted by $\psi(x)$. At each Lattice site, the field ψ consists of a combination of four spin and three color components, resulting in a total of 12 independent complex variables. Therefore, ψ is a function $\psi : \mathcal{L} \to \mathbb{C}^{12}$ with $x \mapsto \psi(x)$. Additionally, we define $\psi_\sigma(x) \in \mathbb{C}^3$ to contain only the three color components at the Lattice site x for a given spin index $\sigma \in 0, 1, 2, 3$.

On the other hand, the gluons live on the links between neighboring Lattice sites. To describe the coupling between Lattice sites, we need a few more definitions. First, we define directions $\mu_i \in \mathcal{L}$ for $i = 0, \ldots, 3$ on the Lattice as

$$\mu_0 = (1, 0, 0, 0), \mu_1 = (0, 1, 0, 0), \mu_2 = (0, 0, 1, 0), \text{ and } \mu_3 = (0, 0, 0, 1).$$

Therefore, the 8 neighbors of $x \in \mathcal{L}$ are the distinct points $x \pm \mu_i$ with $i = 0, \ldots, 3$. Second, the continuum gauge fields from QCD can be represented on the Lattice by matrices $U_\mu(x) \in SU(3)$, called gauge links or link variables. Each $U_\mu(x)$ links the Lattice site x with $x + \mu$ and $U_\mu(x)^\dagger = U_\mu(x)^{-1}$ links $x + \mu$ with x vice versa. The set $\mathcal{U} = \{U_\mu(x) : x \in \mathcal{L}, \mu \in \mu_0, \mu_1, \mu_2, \mu_3\}$ containing all gauge links $U_\mu(x)$ is called a configuration.

3.3.1 Wilson Dirac Operator

In the simulation of QCD on the Lattice, it is essential to use a discretized version of the Dirac operator. One widely used discretization of the Dirac operator in Lattice QCD is the Wilson-Dirac operator.

The Wilson-Dirac operator is a simple, local operator that incorporates the interactions of the quarks with the gluons. The definition of the Wilson discretization of the Dirac operator is given as follows:

Definition 3.1.
Given a gauge configuration U on a lattice $\mathcal{L}$ with $n_\mathcal{L}$ sites, lattice spacing a, and

mass parameter m_0, the Wilson-Dirac operator is defined as:

$$\mathcal{D}_W := \frac{m_0}{a} I_{12n_\mathcal{L}} + \frac{1}{2}\sum_{\mu=0}^{3} \left(\gamma_\mu \otimes (\Delta_\mu + \Delta^\mu) - a I_4 \otimes \Delta_\mu \Delta^\mu\right), \tag{3.6}$$

where m_0 sets the quark mass, a is the Lattice spacing, and Δ_μ and Δ^μ denote covariant backward and forward differences, respectively. Their sum describes a discretization of first derivatives, while their product $\Delta_\mu\Delta^\mu$ represents a Laplacian type operator of second derivatives, which is added in (3.6) *as a regularization term to eliminate red-black instabilities [105].*

In explicit matrix notation, the action of $\mathcal{D}_W$ from (3.6) *on a quark field $\psi(x)$ can be written as:*

$$\mathcal{D}_W(\psi(x)) = \frac{1}{2a}\sum_{\mu=0}^{3} \left((I+\gamma_\mu)\otimes U_\mu(x)\right)\psi(x+\mu) + \left((I-\gamma_\mu)\otimes U_\mu^\dagger(x-\mu)\right)\psi(x-\mu) + \frac{m_0}{2a}\psi(x), \tag{3.7}$$

where x denotes a point on the lattice, and $U_\mu(x)$ is the gauge link variable associated with the μ-direction at the point x. Note that we assume a four-dimensional lattice by summing over four directions in (3.7).

3.3.2 Clover-improved Wilson Dirac operator

The Wilson-Dirac operator introduces lattice artifacts and explicitly breaks chiral symmetry, which can cause significant computational difficulties and constraints. Moreover, it does not satisfy the continuum Dirac equation. To address these issues, a clover term is added to the Wilson-Dirac operator, resulting in the clover-improved Wilson-Dirac operator $\mathcal{D}_{\text{clover}}$ [93]:

$$\mathcal{D}_{\text{clover}} = \mathcal{D}_W + c_{sw}\frac{i}{4}\sigma_{\mu\nu}F_{\mu\nu} \tag{3.8}$$

Here, c_{sw} is the clover coefficient, $\sigma_{\mu\nu}$ is the corresponding Pauli matrices, and $F_{\mu\nu}$ is the Lattice field strength tensor.

The addition of the clover term to the Wilson-Dirac operator improves the chiral symmetry properties because the clover term reduces the discretization error from $\mathcal{O}(a)$ to $\mathcal{O}(a^2)$, where a is the lattice spacing. This improvement reduces the explicit chiral symmetry breaking that occurs in the original Wilson-Dirac operator, making the clover-improved operator a better approximation of the continuum Dirac operator on the lattice. The spectrum of $\mathcal{D}_{\text{clover}}$ is symmetric

with respect to the real axis, and the spectrum of $\mathcal{D}_W$ is, in addition, symmetric with respect to the vertical line $\mathrm{Re}(z) = \frac{4+m_0}{a}$. In other words, if λ is an element of the spectrum of $\mathcal{D}_W$, then $2\frac{m_0+4}{a} - \lambda$ is also an element of the spectrum of $\mathcal{D}_W$. The appropriate value of c_{sw} must be determined for different Lattice QCD simulations (see Sheikholeslami and Wohlert, 1985 [93] for more information on this). In practice, m_0 is negative and for physically relevant mass parameters, the spectrum of $\mathcal{D}_{\text{clover}}$ is located in the right half plane (as shown in Figure 3.1).

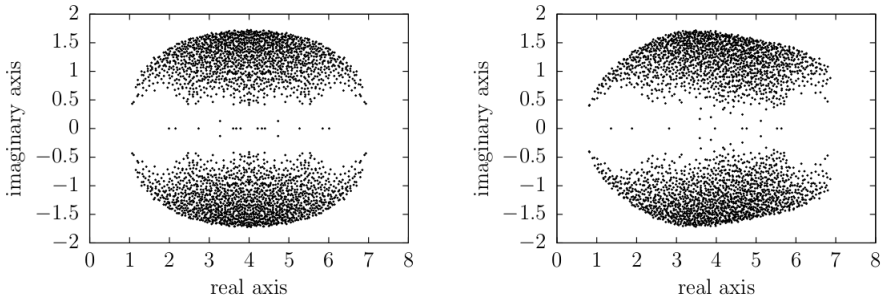


Figure 3.1: Spectrum of 4^4 Wilson-Dirac operator (left) and 4^4 clover improved Wilson-Dirac operator (right) with $m_0 = 0$ and $c_{sw} = 0.1$ respectively. Image adapted from [83].

Fig. 3.1 illustrates the spectrum of a 4^4 Wilson-Dirac operator with $m_0 = 0$ and $c_{sw} = 0$ in the left panel, and the spectrum of a 4^4 clover improved Wilson-Dirac operator with $m_0 = 0$ and $c_{sw} = 1$ in the right panel.

The clover improved Wilson-Dirac operator has been successful in enabling accurate numerical simulations of the QCD theory, and continues to be a valuable tool in the study of the strong interactions [93].

3.4 Schwinger model

The Schwinger model is a description of Quantum Electrodynamics (QED) in two dimensions [91]. It is used as a test bed for the development of algorithms for Lattice QCD because it shares some properties with QCD, such as similar spectral behavior and symmetries. The gauge field of continuum QED in the Schwinger formulation is a real-valued function, and the gauge configurations on the Lattice are a subset of complex numbers with modulus one.

Stabilization of the naive discretization of Lattice QED is necessary in order to suppress the doubling problem. The Wilson-Schwinger operator is therefore

defined as

$$S_W := \frac{m_0}{a} I_{2n_\mathcal{L}} + \frac{1}{2} \sum_{\mu=1}^{2} \left(\sigma_\mu \otimes (\Delta_\mu + \Delta^\mu) - a I_2 \otimes \Delta_\mu \Delta^\mu \right), \tag{3.9}$$

where m_0 is the mass parameter, $n_\mathcal{L}$ is the number of sites on the Lattice, a is the Lattice spacing and σ_1, σ_2 are Pauli matrices $\in \mathbb{C}^{2\times 2}$. The analogue to the Lattice QCD matrix γ_5 in the Schwinger model is $\sigma_3 = i\sigma_1\sigma_2$, and the analogoue to Γ_5 is $\Sigma_3 = \sigma_3 \otimes I_{n_\mathcal{L}}$. The Wilson-Schwinger operator is Σ_3-symmetric, meaning $(\Sigma_3 S_W)^H = \Sigma_3 S_W$.

Chapter 4

Linear Solvers

In this chapter, we will examine various techniques for solving linear systems of equations, including matrix decompositions in Sec. 4.1, iterative methods in Sec. 4.2, and multigrid methods in Sec. 4.3. Matrix decompositions involve breaking the matrix down into simpler matrices that can be used to more efficiently solve the linear system, and can be used to solve the system directly. Iterative methods are used to solve the linear system by repeatedly improving an approximate solution until convergence is reached, and are particularly useful for large-scale problems. However, they may require more iterations to converge and may have a convergence rate that depends on the condition number of the matrix. Multigrid methods are a type of iterative method that use a hierarchy of grids to accelerate convergence and are effective for problems with a wide range of length scales.

We will delve into the specifics of popular iterative methods, including the generalized minimal residual method (GMRES) in Sec. 4.2.3 and the flexible GMRES in Sec. 4.2.5, which are effective for solving large and sparse linear systems. We will also introduce the concepts of preconditioning in Sec. 4.2.4, Krylov subspaces in Sec. 4.2.1, and Arnoldi method in Sec. 4.2.2. The focus of the chapter will be on the mathematical foundations of these techniques and their applications in solving linear systems of equations.

Finally, we will provide an overview of multigrid methods, including topics such as geometric multigrid in Sec. 4.3.1, algebraic multigrid in Sec. 4.3.2, two-level and multilevel multigrid in Sec. 4.3.3, and the application of multigrid methods to Lattice QCD in Sec. 4.3.4.

This chapter draws heavily from sources [15, 58, 83, 87, 88, 95, 98], and proofs and additional information can be found there.

We consider the problem of solving linear systems of equations of the form

$$Ax = b, \tag{4.1}$$

where A is a square matrix, x is the unknown vector, and b is the right-hand side vector.

4.1 Matrix decompositions

Matrix decompositions refer to the process of decomposing a matrix into simpler matrices that can be used to more efficiently solve a linear system of equations. Some common matrix decompositions include LU decomposition, QR decomposition, and Cholesky decomposition [18, 27, 51, 98]. These decompositions can be used as a preprocessing step before applying a direct method, or they can be used to solve the linear system iteratively.

- LU decomposition with partial pivoting: This decomposition factorizes a matrix A into the product of a permutation matrix P, a lower triangular matrix L, and an upper triangular matrix U, i.e.,

  $$A = PLU. \tag{4.2}$$

  This decomposition can be used to solve (4.1) by first solving the system $PLy = b$ for y, and then solving the system $Ux = y$ for x.

- Cholesky decomposition: This decomposition is only applicable to symmetric positive-definite matrices, and factorizes a matrix $A \in \mathbb{C}^{n\times n}$ into the product of a lower triangular matrix L and its transpose L^H, i.e.

  $$A = LL^H, \tag{4.3}$$

  and solving (4.1) can be achieved by first solving $Ly = b$ for y, and then solving the system $L^H x = y$ for x.

- QR decomposition: This decomposition factorizes a matrix $A \in \mathbb{C}^{n\times n}$ into the product of an orthogonal matrix Q and an upper triangular matrix R, i.e.

  $$A = QR. \tag{4.4}$$

  One can utilize this decomposition to solve (4.1) by solving the equation $Rx = Q^H b$ for x. The columns of the matrix Q are an orthonormal basis, meaning that the inner product of q_i and q_j is zero if $i \neq j$ and the norm of $\|q_k\| = 1, \quad \forall \quad k \in 1, ..., n$. The matrix Q can be obtained using the

modified Gram-Schmidt algorithm, which generates a full set of orthonormal vectors $Q = [q_1, ..., q_n]$ that span the space defined by a set of vectors $A = [a_1, ..., a_n]$ if the vectors $a_1, ..., a_n$ are linearly independent and is a numerically more stable version of the regular Gram-Schmidt algorithm.

4.1.1 Modified Gram-Schmidt

Modified Gram-Schmidt procedure is a numerically stable version of the regular Gram-Schmidt algorithm and can be described as follows [98]:

1. Initialize an empty set of orthonormal vectors, $Q = [\quad]$.
2. For each vector a_i in the set of input vectors $A = a_1, a_2, ..., a_n$:
   a) $q_i = a_i$.
   b) For each vector q_j in the set of orthonormal vectors Q:
      i. $q_i = q_i - \frac{q_j^H a_i}{q_j^H q_j} q_j$.
   c) Normalize q_i by setting it to $\frac{q_i}{|q_i|}$.
   d) Add q_i to the set of orthonormal vectors Q.
3. Return the set of orthonormal vectors Q.

The resulting set of orthonormal vectors $Q = [q_1, q_2, ..., q_n]$ defines a basis of the space spanned by the original set of vectors $A = [a_1, a_2, ..., a_n]$.

4.2 Iterative methods

Direct methods like Gaussian elimination can solve linear systems of equations like (4.1), but they tend to be too computationally expensive for large, sparse matrices with a complexity of $\mathcal{O}(n^3)$. Iterative solvers, on the other hand, have a lower complexity of $\mathcal{O}(n)$ for one iteration if the number of nonzeros in A is $\mathcal{O}(n)$ and can be stopped early to provide an approximate solution. They also only require a routine for calculating the matrix-vector product Ax, and do not require storing the entire matrix in memory [87].

To evaluate the quality of an approximation $x^{(k)}$, we can use the residual equation, defined as

$$r^{(k)} = b - Ax^{(k)} \quad \text{and} \quad e^{(k)} = x - x^{(k)}. \tag{4.5}$$

The residual equation states that $Ae^{(k)} = r^{(k)}$. The error of the approximation can be bounded by

$$\|e^{(k)}\| = \|A^{-1}r^{(k)}\| \leq \|A^{-1}\| \cdot \|r^{(k)}\|, \tag{4.6}$$

which shows that if $|A^{-1}|$ is large, the error can still be significant even if the residual is small. Many iterative methods involve updating the iteration vector $x^{(k)}$ in each step, starting from an initial vector $x^{(0)}$, by approximating the error $e^{(k)}$ with $\tilde{e}^{(k)}$ and setting

$$x^{(k+1)} = x^{(k)} + \tilde{e}^{(k)}. \tag{4.7}$$

This process is repeated until some convergence criteria is met, or a maximum number of iterations is reached. Its implementation requires the following steps [87, 98]:

(a) Begin with x_0 as an initial guess solution.

(b) Iterate over the obtained solution x_i until you get an approximation solution which is accurate enough for the exact solution, where $i = 1, \ldots, k$

(c) After k iterations, the error of the approximation is defined as

$$e^{(k)} = x - x^{(k)}, \tag{4.8}$$

and its relative norm is given by:

$$\|e^{(k)}\|_2 = \frac{\|x - x^{(k)}\|_2}{\|x^{(k)}\|_2} \tag{4.9}$$

(d) Since the exact solution is unknown, the error cannot be directly computed. Instead, the accuracy of the k-th approximation can be estimated using the residual, which is defined as the difference between the right-hand side vector b and the product of the coefficient matrix A and the current approximation $x^{(k)}$, i.e., $r^{(k)} = b - Ax^{(k)}$. By solving the linear system $Ae^{(k)} = r^{(k)}$, the error $e^{(k)}$ can be obtained as

$$e^{(k)} = x - x^{(0)} - x^{(1)} - \cdots - x^{(k-1)} = A^{-1}r^{(k)} \tag{4.10}$$

(e) The tolerance ϵ is the stopping criterion for the iterative method, and it is defined as the maximum allowable reduction in the relative residual norm, i.e., the ratio of the current residual norm to the initial residual norm:

$$\frac{\|r^{(k)}\|_2}{\|r^{(0)}\|_2} \leq \epsilon \tag{4.11}$$

only if $x_0 = 0$, otherwise one takes $r^{(0)} = b - Ax^{(0)}$

One method of deriving an iterative method is to split the matrix A into two matrices, M and N, such that $A = M - N$. This allows us to express the original system of equations as follows:

$$Ax = (M - N)x = b, \tag{4.12}$$

where M and N are both $n \times n$ matrices. Substituting this into the linear system equation, we obtain:

$$Mx = Nx + b. \tag{4.13}$$

From this equation, we can construct a basic iterative method as follows:

$$Mx_{k+1} = Nx_k + b \quad \text{or} \quad x_{k+1} = M^{-1}Nx_k + M^{-1}b. \tag{4.14}$$

After $k + 1$ steps, we can compute the iteration x_{k+1} from the previous one, x_k. This equation can also be rewritten as:

$$x_{k+1} = x_k + M^{-1}r_k, \tag{4.15}$$

where r_k is the residual after k iterations, which determines the difference between the exact and iterative solution of the algebraic linear system equation.

Basic Iterative Methods (BIM), such as Jacobi, Gauss-Siedel (GS), and Successive over-relaxation (SOR), can be derived by using the splitting technique

$$A = D - U - L \tag{4.16}$$

where D, U, and L are the diagonal, upper, and lower parts of A respectively. Both matrices M and N depend on this decomposition. The update equations for these BIMs are summarized in Table 4.1.

Method	Matrix Splitting	Update Equation
Jacobi	$M = D, N = L + U$	$x_{k+1} = D^{-1}(b - Lx_k)$
Gauss-Seidel	$M = D - L, N = U$	$x_{k+1} = L^{-1}(b - Ux_k)$
SOR	$M = (1/\omega)D - L, N = ((1 - \omega)/\omega)D - L$	$x_{k+1} = (1 - \omega)x_k + \omega D^{-1}(b - Lx_k)$

Table 4.1: Comparison of the Jacobi, Gauss-Seidel, and SOR methods.

4.2.1 Krylov Subspace

When working with large, sparse matrices, such as those arising from the discretization of partial differential equations, it is often only possible to perform

matrix-vector multiplications with the matrix Ax. This is because direct manipulation of the matrix, such as matrix decompositions, can be too computationally expensive. Krylov subspace methods, which only require matrix-vector multiplications and have low storage requirements, are a good choice for this type of problem because they are efficient and scalable [87]. It can be defined as:

Definition 4.1 (Krylov subspace)**.**
Given a matrix $A \in \mathbb{C}^{n\times n}$ and a vector $r \in \mathbb{C}^n$, the m-th Krylov subspace *is the linear subspace spanned by the set of vectors $r, Ar, A^2r, \ldots, A^{m-1}r$. This subspace is denoted as $\mathcal{K}_m(A, r)$, simply as $\mathcal{K}_m$.*

The Krylov subspace method involves iteratively updating an approximation x_k to the solution of the linear system $Ax = b$ by projecting the residual $r_k = b - Ax_k$ onto the Krylov subspace $\mathcal{K}_m$ and computing a new search direction p_k. The updated approximation is then given by $x_{k+1} = x_k + \alpha_k p_k$, where α_k is a scalar step size.

The dimension of the subspace being considered is determined by the index m. The Krylov subspace methods typically involve projection processes using $\mathcal{K}_m$. These processes are either orthogonal or oblique and are spanned by vectors of the form $p(A)v$, where p is a polynomial defined as

$$\rho(A) = \sum_{i=1}^{m} \rho_i A^{i-1} = \rho_1 + \rho_2 A + \rho_3 A^2 + ... + \rho_m A^{m-1}. \tag{4.17}$$

Therefore, any vector $x \in \mathcal{K}_m$ can be expressed as $x = p(A)r$, as long as the degree of the polynomial is not greater than $m - 1$.

4.2.2 Arnoldi Process

Arnoldi's method [61], described in Algorithm 1 is an iterative method for constructing an orthonormal basis of the Krylov subspace $\mathcal{K}_m$ [87].It is similar to the modified Gram-Schmidt procedure described in Sec. 4.1.1, which is a method for constructing an orthonormal basis from a set of vectors, but it is specifically designed to deal with the ill-conditioning that often occurs when using the Krylov subspace as a basis. The input to the algorithm is a matrix A and an initial vector r, and the output is an orthonormal basis Q_m of the Krylov subspace.

Algorithm 1 Arnoldi Process

Input: matrix $A \in \mathbb{C}^{n \times n}$, vector $r \in \mathbb{C}^n$, number of iterations m
Output: Orthonormal basis $Q_m \in \mathbb{C}^{n \times m+1}$ of the Krylov subspace $\mathcal{K}_{m+1}$ and upper Hessenberg matrix $H_m \in \mathbb{C}^{m+1 \times m}$

$q_1 \leftarrow r/\|r\|_2$
$Q_m \leftarrow [q_1]$
for $j = 1$ to m **do**
    $v \leftarrow AQ_m(:, j)$
    for $i = 1$ to j **do**
        $h_{i,j} \leftarrow Q_m(:, i)^* v$
        $v \leftarrow v - h_{i,j} Q_m(:, i)$
    end for
    $h_{j+1,j} \leftarrow \|v\|_2$
    if $h_{j+1,j} = 0$ **then**
        return Q_m and $H_m = [H_{1:m,1:m}]$
    end if
    $q_{j+1} \leftarrow v/h_{j+1,j}$
    $Q_m \leftarrow [Q_m, q_{j+1}]$
end for
return Q_m and $H_m = [H_{1:m,1:m}]$

In this algorithm, Q_m is an orthonormal basis of the Krylov subspace $\mathcal{K}_{m+1}$, and H_m is an upper Hessenberg matrix that approximates the matrix A in this subspace. The Arnoldi process has a wide range of applications, including solving linear systems of equations, approximating eigenpairs and evaluating matrix functions.

The Arnoldi relation $AQ_m = Q_m H_{m+1,m}$, which is a matrix formulation of Arnoldi's method, can be used to define the Generalized Minimal Residual (GMRES) method for solving linear systems of equations $Ax = b$. GMRES minimizes the residual norm $\|r\| = \|b - Ax\|$ by finding an approximation $x_m = x_0 + Q_m y_m$, where y_m is the solution to an $(m + 1) \times m$ least-squares problem. Accordingly, the residual norm $\|r\|$ can be cheaply updated in each iteration and the iteration stops when $\|r\|$ falls below a given threshold, resulting in the approximate solution x_m [87].

4.2.3 GMRES

GMRES starts with an initial residual and iteratively applies the Arnoldi process to the matrix of the linear system and the residual vector. The solution to the linear system is computed using the orthonormal basis generated by the Arnoldi

process and the residual is updated. The process continues until the residual is below a desired tolerance or the maximum number of iterations is reached. GMRES is efficient for solving large, sparse linear systems and is often used in scientific and engineering applications and it can be described as follows [87]:

Given a matrix $A \in \mathbb{R}^{n \times n}$, a right-hand side vector $b \in \mathbb{R}^n$, and a starting vector $x_0 \in \mathbb{R}^n$, the GMRES method iteratively computes an approximation to the solution x of (4.1) as follows:

1. Set $r_0 = b - Ax_0$
2. Perform m steps of Arnoldi's method starting with r_0. This gives Q_m, H_m in the Arnoldi relation $AQ_{m+1} = Q_m H_m$
3. Solve the least squares problem $y_m = \underset{y \in \mathbb{C}^m}{\operatorname{argmin}} \|H_m y - \|r_0\| e_1\|$, where $e_! = (1, 0, \cdots, 0)^T \in \mathbb{C}^m$
4. Return the iterates $x_m = x_0 + Q_m y_m$

The maximum number of iterations must be specified in advance, but the method may be terminated early if the residual is reduced to a desired tolerance before the maximum number of iterations is reached. This is possible because of the residual norm can be cheaply updated after each iteration of Arnoldi's method without computing the GMRES iterates

4.2.4 Preconditioning

Preconditioning is a technique used to accelerate the convergence of iterative methods for solving linear systems of equations. It is based on the idea of transforming the original problem (4.1) into an equivalent problem with a smaller condition number $\kappa(A)$. This can be done through left or right preconditioning.

1. Left preconditioning involves multiplying the preconditioner M from the left to the original system, as shown in the equation $MAx = Mb$.
2. Right preconditioning involves multiplying the preconditioner from the right, as shown in the equation $AMy = b$, with the solution $x = My$.

where M is a matrix that is "close" to A^{-1} in some sense, such that $\kappa(MA) \approx 1$.

Preconditioning requires the application of the preconditioner matrix M in each iteration. To be effective, M should be "close enough" to A^{-1} to have a significant impact on the condition number, but it should also be cheaper to apply compared to solving linear systems with A. M is often realized through a method that provides a low precision solution of systems with matrix A.

We now discuss two different preconditioners for GMRES and their impact on the convergence speed of linear systems of equations involving the Dirac operator. It is important to note that applying non-stationary iterative schemes as preconditioners for GMRES requires modifying GMRES to maintain its optimality criterion of minimizing the norm of the residual in each step. This modified method is known as flexible GMRES (FGMRES) [85].

4.2.5 FGMRES

The Flexible Generalized Minimum Residual (FGMRES) method is an extension of the GMRES method for solving linear systems of equations. The key difference between GMRES and FGMRES is that FGMRES allows the user to specify a non-stationary preconditioner, which is an iterative scheme to improve the convergence of the algorithm. By applying the preconditioner to the residuals at each iteration, FGMRES can often converge more quickly than GMRES and it can be described as follows [85]:

Given a matrix $A \in \mathbb{R}^{n \times n}$, a right-hand side vector $b \in \mathbb{R}^n$, and a starting vector $x_0 \in \mathbb{R}^n$, the FGMRES method iteratively computes an approximation to the solution x of the linear system $Ax = b$ as follows:

1. Initialize the residual $r_0 = b - Ax_0$ and its norm $\beta = \|r_0\|$, and set the first search direction as $v_1 = r_0/\beta$
2. For $j = 1, \ldots, m$:
   a) Apply the preconditioner to v_j to get z_j
   b) Compute $w_j = Az_j$
   c) For $i = 1, \ldots, j$:
      i. $h_{i,j} \leftarrow \langle w_j, v_i \rangle$
      ii. $w_j \leftarrow w_j - h_{i,j} v_i$
   d) $h_{j+1,j} \leftarrow w_j$
   e) Update the residual norm $\|r\|$
   f) If $\|r\| < tol$, $m \leftarrow j$ and break
   g) $v_{j+1} \leftarrow w_j / h_{j+1,j}$
3. Compute the optimal solution y_m
4. Compute the approximate solution $x_m = x_0 + Z_m y_m$

The FGMRES algorithm has the following important features:

- It maintains the optimality criterion of minimizing the norm of the residual in each step over space Z_m.
- The use of preconditioners can significantly improve the convergence speed of the algorithm and reduce the overall computational cost.
- Preconditioning allows for (a) to be performed in single precision, leading to faster execution times.

4.3 Multigrid Methods

The convergence speed of the iterative methods in Sec.4.2 is still dependent on the conditioning of the matrix, so if no good preconditionar is available, Multigrid methods can be used because the convergence speed does not depend on the condition number [54, 84]. There are two main types of multigrid methods:

- Geometric multigrid (GMG): These methods use geometrically related grids.
- Algebraic multigrid (AMG): These methods construct an algebraic hierarchy of grids and operators based on the structure of the linear system.

Multigrid methods have been found to significantly improve the convergence of iterative solvers in Lattice QCD simulations. However, the choice of method may depend on the specific problem being solved and the available computational resources, see [15]

4.3.1 The Geometry of Multigrid Methods

Multigrid methods were developed to understand iterative methods for solving boundary value problems in a simple geometric interpretation. This interpretation is presented in [15] and [54]. As an example, consider the two-dimensional boundary value problem of finding $u(x, y)$ in the domain $\Omega = (0, 1)^2$ such that $-u_{xx} - u_{yy} = f(x, y)$ and $u(x, y) = 0$ for $(x, y) \in \delta\Omega$, where $\delta\Omega$ is the boundary of Ω.

The discretization of this boundary value problem by second-order finite differences on an equidistant grid with grid spacing h produces a set of linear equations. The grid spacing is equal to $h_x = h_y = \frac{1}{N}$. The resulting system has $m = (N-1)^2$ equations. Each equation has the form

$$\frac{-u_{i-1,j} + 2u_{ij} - u_{i+1,j}}{h^2} + \frac{-u_{i,j-1} + 2u_{ij} - u_{i,j+1}}{h^2} = f_{ij},$$

where $f_{ij} = f(ih_x, jh_y)$ and $i, j = 1, \ldots, N-1$. The boundary conditions are $u_{i0} = u_{im} = u_{0j} = u_{mj} = 0$.

Introducing lexicographically ordered vectors

$$u = \begin{pmatrix} u_{11} & u_{21} & \ldots & u_{N-1,1} & u_{1,2} & \ldots & u_{N-1,N-1} \end{pmatrix}^T$$

and

$$f = \begin{pmatrix} f_{11} & f_{21} & \ldots & f_{N-1,1} & f_{1,2} & \ldots & f_{N-1,N-1} \end{pmatrix}^T$$

this system of linear equations reads

$$Au = h^2 f, \tag{4.18}$$

with A given by

$$A = \begin{pmatrix} B & -I & & & \\ -I & B & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & B & -I \\ & & & -I & B \end{pmatrix},$$

where each block B is given by

$$B = \begin{pmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}.$$

In the discretization of the two-dimensional Poisson equation each variable is only coupled to its direct neighbors as shown in Fig. 4.1. The corresponding linear system can be written in stencil notation as:

$$A = \frac{1}{h^2} \begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}$$

where h is the grid spacing. Solving such sparse linear systems has been a subject of extensive research. One approach to solving these systems is to use a direct solution method like Gaussian elimination. The best direct methods for two-dimensional elliptic equations that use fast Fourier transforms and cyclic reduction schemes have a computational complexity of $\mathcal{O}(N^2 \log(N))$, which is close to the optimal complexity of $\mathcal{O}(N^2)$. However, these methods are limited to two-dimensional problems with constant coefficients.

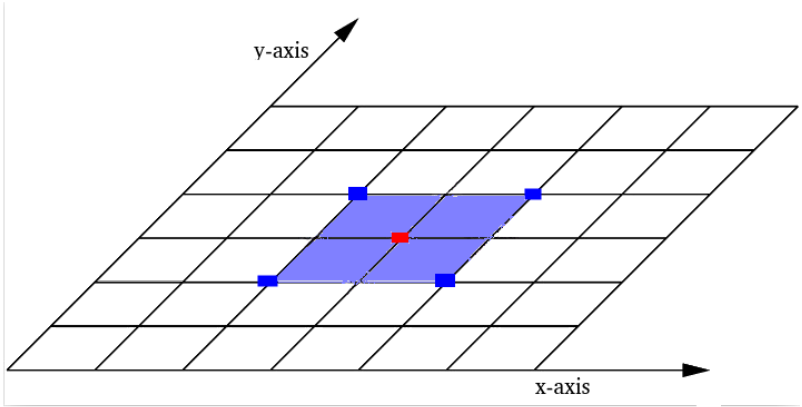


Figure 4.1: The coupling of a typical variable in the discrete equations (4.3.1) on the two-dimensional equidistant grid.

4.3.2 The main idea of Multigrid

Multigrid methods use a hierarchy of grids with progressively finer discretizations to reduce the error in the solution. There are two main components of a multigrid method: a smoother and a coarse grid correction [14, 54, 84, 99]. The smoother is an iterative method used to locally smooth out errors in the solution, while the coarse grid correction involves projecting the smoothed residual onto a coarser grid and solving for the correction there.

One common way to transition between grid levels is through the use of prolongation and restriction operators, which map between grids of different sizes. A full multigrid method involves cycling between multiple grid levels, with the most common method being the V-cycle. This involves starting on the finest grid, progressing through coarser grids, and returning to the finest grid. Another option is the W-cycle, which consists of two V-cycles connected by a single smoothing operation on the finest grid. Fig. 4.2 shows two sketches of different multigrid algorithms. The first sketch (a) shows one cycle of V-cycle multigrid, which consists of a single loop that begins on the coarsest grid and progresses to the finest grid, then back to the coarsest grid. The second sketch (b) shows one cycle of W-cycle of multigrid, which consists of two V-cycles connected by a single smoothing operation on the finest grid.

Algorithm 2 outlines the V-cycle multigrid method. The input includes the initial approximation v_h to the solution u on the fine grid Ω_h, the residual r_h, and the coarser grid approximation e_{2h}. It uses a hierarchy of grids with progressively finer discretizations to reduce the error in the solution. The method consists of two main components: a smoother and a coarse grid correction. The smoother is an iterative method used to locally smooth out errors in the solution, while the coarse grid correction involves projecting the smoothed residual onto a coarser

grid and solving for the correction there. The V-cycle multigrid method starts on the finest grid, progresses through coarser grids, and returns to the finest grid. It uses prolongation and restriction operators to transition between grid levels and applies a few iterations of the smoother at each step [15].

Algorithm 2 Single V-Cycle Multigrid

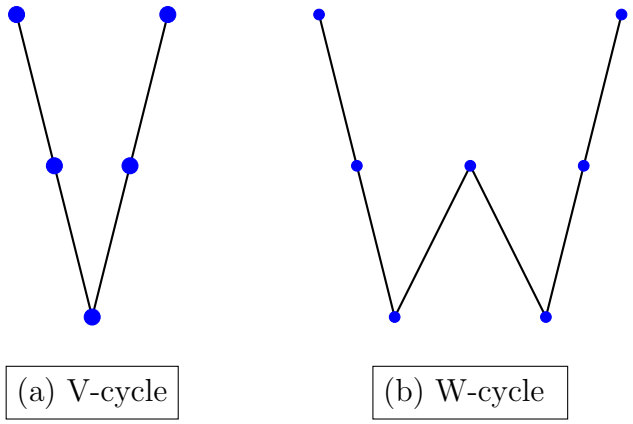
1: Get the approximation $\mathbf{v}_h$ to the solution $\mathbf{u}$ for the problem $A\mathbf{u} = \mathbf{f}$ in the fine grid Ω_h with a few iterations of the smoother.
2: Compute the residual: $\mathbf{r}_h = \mathbf{f}_h - A_h\mathbf{v}_h$.
3: Restrict the residual to the coarser grid: $\mathbf{r}_{2h} = R\mathbf{r}_h$.
4: Approximate the solution for the residual equation $\mathbf{r}_{2h} = A_{2h}\mathbf{e}_{2h}$ in the coarse grid using a few iterations of the smoother.
5: Prolongate this correction to the fine grid Ω_h and correct the fine grid approximation: $\mathbf{v}_h \leftarrow \mathbf{v}_h + P\mathbf{e}_{2h}$.
6: Apply a few iterations of the smoother to the corrected solution in the fine grid.

In the context of Lattice quantum chromodynamics, the Schwarz Alternating Procedure (SAP) method is often used as a smoother in multigrid algorithms. However, the SAP method is not effective at removing error components belonging to the "near kernel," which is the space spanned by the eigenvectors associated with small eigenvalues of the matrix D [83].

To address this issue, an operator D_c is found that behaves similarly to D on the near kernel in terms of both its action on the near kernel and its connection structure and sparsity. The multigrid method is then applied recursively to D_c in order to extend it from a two-grid method to a true multigrid method. This allows the algorithm to effectively remove error components belonging to the near kernel in a smaller subspace with n_c variables, which should accurately approximate the near kernel.

4.3.3 Multilevel AMG

In AMG method, one possibility to define the coarse grid operator D_c is as a Petrov-Galerkin projection of the matrix D, which is applied to the system of linear equations [15]. Let $n_c < n$ with considering a full rank linear restriction and prolongation/interpolation operators $R : \mathbb{C}^n \to \mathbb{C}^{n_c}$ and $P : \mathbb{C}^{n_c} \to \mathbb{C}^n$. The operator D_c is given by $D_c = RDP$. These operators transfer information between grids, with R going from a finer grid to a coarser grid and P going from a coarser grid to a finer grid. The coarse grid correction is then applied to the current approximation, ψ, as follows: $\psi \leftarrow \psi + PD_c^{-1}Rr$, where $r = \eta - D\psi$.



(a) V-cycle (b) W-cycle

Figure 4.2: Schematic of the various approaches to the multigrid method.

Multigrid methods use restriction and prolongation operators to transfer information between grid levels. The restriction operator R transfers information from the original space to a subspace, while the prolongation operator P brings information back to the original space. In a coarse grid correction, the current residual r is restricted using R to the subspace, and an equation is solved to find the coarse error e_c. This error is then brought back to the original space using P as a correction for the current iterate ψ. One step of coarse grid correction can be written as:

- $\psi \leftarrow (I - PD_c^{-1}RD)\psi + PD_c^{-1}R\eta$, where D_c is the coarse grid operator and η is the current iterate.
- The corresponding error propagator is given by $I - PD_c^{-1}RD$, which is a projection onto the orthogonal complement of the range of RD along the range of P.

The purpose of the coarse grid correction is to complement the action of the smoother by reducing error components that the smoother is unable to effectively remove. To do this, it is important that the range of P approximately contains the near kernel and that the range of RD is approximately orthogonal to the remaining complementary eigenvectors. This second condition is satisfied if the range of R is approximately spanned by the left eigenvectors corresponding to the small eigenvalues of D. This is because left and right eigenvectors are orthogonal.

Mathematically, this can be expressed as:

$$\begin{aligned} \text{range}(P) \approx \text{near kernel}, \quad \text{range}(R) \approx \text{left eigenvectors of } D, \\ \text{range}(RD)^{\perp} \approx \text{complementary right eigenvectors of } D \end{aligned} \tag{4.19}$$

The basic structure of a two-level algorithm using multigrid methods involves

alternating between the application of a smoother and a coarse grid correction. Specifically, the algorithm begins with an initial approximation of the solution, and then applies the smoother to precondition the error. The residual is then computed and transferred to the coarser grid, where it is solved using the coarse grid operator. The solution is then interpolated back to the fine grid and used to correct the error. These steps are repeated until convergence is achieved.

The two-grid method, or two-level V-cycle, is often used as a preconditioner in combination with a Krylov subspace method to solve systems of linear equations. The preconditioner is applied before the Krylov subspace method to reduce the error in the initial guess for the solution, which can improve the convergence rate and efficiency of the overall solution process, see [7].

The multilevel version of Algorithm 2, also known as AMG, involves recursively applying a two-level method to successively coarser grids until a sufficiently small operator D_c is obtained that can be solved directly. The matrices R and P, which transfer information between grids, should be chosen to approximate the left and right near kernel of the original operator D as closely as possible in order to effectively reduce error components in the near kernel. The goal is not to fix the size of the coarser grids, as is done in deflation methods, but rather to find sparse R and P that approximate the near kernel well. To construct the multilevel hierarchy, the following steps are taken [15]:

1. The dimension of the vector space at each level l is denoted by n_l.
2. Interpolation operators $P_l : \mathbb{C}^{n_{l+1}} \to \mathbb{C}^{n_l}$, $l = 1, ..., L-1$, are defined.
3. The restrictors $R_l = P_l^H$ are defined.
4. The coarse grid operators $D_l : \mathbb{C}^{n_l} \to \mathbb{C}^{n_l}$ are defined as the Petrov-Galerkin operator $D_l = P_{l-1} D_{l-1} P_{l-1}^H$, $l = 2, ..., L$.

In AMG, the V-cycles and W-cycles are not always the optimal choice in terms of performance. As a result, the K-cycle method is used, which is a recursive method that uses solutions from a Krylov subspace method. Specifically, at each level l, the solution to the system at level l is approximated using a few iterations of a Krylov method preconditioned by the K-cycle multilevel method from level $l+1$ to L.

To solve the lack of a priori information about the test vectors for the construction of P_l, the AMG method introduces a setup phase which is iterative and adaptive. This phase outputs N_v test vectors using n_{setup} iterations. The iteration begins with a random vector $v_i^{(0)}$ and proceeds as follows:

$$v_i^{(k)} = D^{-1} v_i^{(k-1)} = D^{-k} v_i^{(0)}, \quad i = 1, ..., N_v, \quad k = 1, ..., n_{setup}. \tag{4.20}$$

The inversion of D^{-1} is performed using the SAP smoother and the available K-Cycle on the kth iteration. This approach converges the v_i to the test vectors needed, i.e., to the eigenvectors with the smallest eigenvalues.

In the field of scientific computing, effective P and R operators have been developed for a variety of applications, leading to solvers that do not suffer from critical slowing down. In the following sections we will describe a specific construction of P and R that has been successful in the context of Lattice QCD.

In the following chapters we will be discussing several developments related to stochatic trace estimation of the inverse of a matrix which requires solving linear systems of equations involving the Dirac matrix. These developments are based on the multigrid hierarchy which constructed with a bootstrap setup and aggregation based prolongations as in Domain Decomposition Aggregation-based Adaptive Algebraic Multigrid method (DDαAMG), [38, 40] which is a code package and is used to efficiently solve linear systems of equations.

Moreover, DDαAMG will be utilized in Sec. 9.4 to develop our own algebraic multigrid solver for solving linear systems of equations involving the Schwinger operator, which is used in lattice QED.

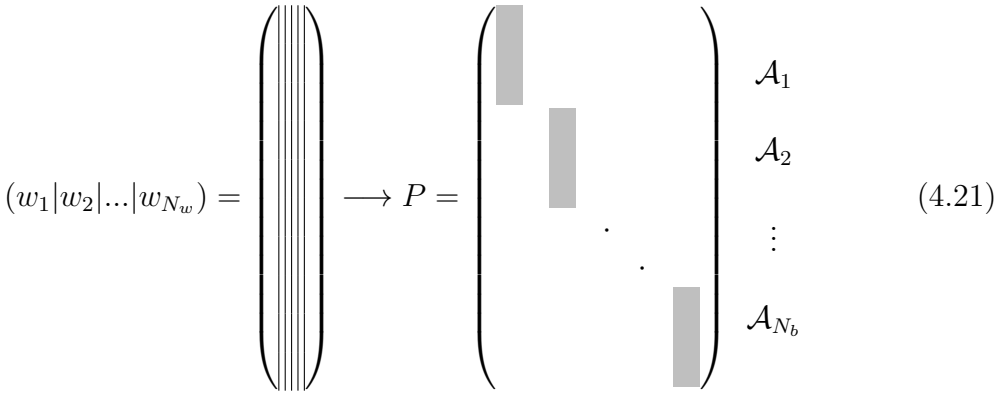
4.3.4 Multigrid in Lattice QCD

In this section, we will discuss the use of adaptive multigrid method to solve linear systems involving the Dirac operator in Lattice QCD. This method has been shown to be effective for various discretizations of the Dirac operator, including the Wilson, Twisted Mass, Staggered, and others [34, 47, 105]. In particular, adaptive (algebraic) multigrid methods have demonstrated superior performance for the Wilson Dirac operator in Lattice QCD, as well as in other Lattice QCD discretizations [7, 13, 36, 40]. These methods offer faster convergence and insensitivity to conditioning compared to conventional Krylov subspace methods. We will describe one construction of efficient multigrid methods specifically for use in Lattice QCD in the following.

The DDαAMG method is a multigrid method that uses domain decomposition and aggregation-based intergrid transfer operators to solve linear systems of equations involving the Dirac operator. It is designed to be adaptive and able to handle a wide range of systems efficiently.A detailed description of DDαAMG , including its implementation, numerical studies, and additional references, can be found in the PhD thesis by M. Rottmann [83] and in parts in [37, 39].

The DDαAMG method starts by generating the eigenvector information for the intergrid operators through a setup procedure. This procedure has two phases and is mainly based on the inverse iteration algorithm. In the first phase, the

algorithm applies several steps of the smoother to a set of n_{tv} of random test vectors $W = w_1, \ldots, w_{n_{tv}}$. Afterwards, the set of vectors W is splitted into aggregates A_i, which can be thought of as a block splitting of the Lattice. The details of this process are described in the following section. Similar to a Lattice block, each aggregate A_i consists of a set of Lattice points, with the exception that variables on a single Lattice site can belong to different aggregates. This leads to a block structure in the matrix W, as shown in Figure 4.3. In this figure, each aggregate A_i is orthogonalized and a "block diagonal" interpolation operator $P = \mathrm{diag}(W)$ is defined, with each diagonal block containing an aggregate A_i.



(4.21)

Figure 4.3: The construction of the interpolation operator P .

The iterative phase of the Bootstrap AMG setup procedure uses the initial multigrid hierarchy to improve the eigenvectors. It does this by applying a full AMG cycle, which includes both the smoother and a coarse grid correction, to each candidate eigenvector. The multigrid hierarchy is then updated and the process is repeated on the next coarser level, see [83].

The Bootstrap AMG setup is a procedure for generating the interpolation and restriction operators P and R that will be used in the DDαAMG method to solve linear systems of equations involving the Dirac operator in Lattice QCD. This algorithm begins by generating a set of random test vectors W, which are then processed using a smoothing operator. The resulting test vectors are divided into aggregates $A_1, \ldots, A_s$ and orthogonalized to form the block diagonal interpolation operator P. The restriction operator R is defined as the transpose of P. The algorithm also includes an iterative phase in which the initial multigrid hierarchy is used to further improve the eigenvectors by performing an AMG cycle on each eigenvector and updating the multigrid hierarchy. The smoothing method used in this algorithm can be chosen between the SAP and GMRES methods, with

numerical studies suggesting that the two color SAP approach is generally the most suitable choice for the Dirac operator.

Chapter 5

Stochastic Trace Estimation

In this chapter, we discuss the problem of estimating the trace of the inverse of a large sparse matrix $D \in \mathbb{C}^{n\times n}$, denoted as $\mathrm{tr}(D^{-1}) = \sum_i^n D_{ii}^{-1}$, occuring in a variety of applications, such as uncertainty quantification [11], computing the log-determinant [90, 103], statistical learning [1, 33, 46], nuclear norm estimation [17], computing the Estrada index of a graph [24]. Further applications are discussed in [100, 101].

Our research is concentrated on Lattice QCD. The trace of the inverse of the discretized Dirac operator is used to determine the disconnected fermion loop contribution to an observable see [92]. With the advancement of simulation techniques, these contributions become more significant. Computing the exact trace of the matrix D^{-1} is prohibitive because of its huge size $n \times n$, and the only way to access information on the entries of D^{-1} is through matrix-vector multiplications $D^{-1}\psi$ where ψ denotes suitable random vectors of length n, i.e. via the solution of linear systems with matrix D.

We first introduce the main idea of estimating the trace of the matrix in Sec. 5.1, and then in sections 5.2 and 5.2.1 we describe the Hutchinson estimator, the standard approach for estimating the trace, and its tail bounds, respectively. We close this chapter by exploring, in detail, the convergence analysis for trace estimation in Sec. 5.2.2.

5.1 Basic idea of trace estimation

The trace of the inverse can be computed directly as a simple task with cubic cost $\mathcal{O}(n^3)$ if the matrix is small enough and given explicitly i.e easy to access the diagonal entities directly. Unlikely, computing the trace of a matrix M could

be extremely difficult when access to M is restricted to matrix-vector products (MVPs), such as M is a function in another matrix $M = f(D)$, and the quadratic form $\psi^H f(D)\psi$ is feasible for an arbitrary vector. We consider here $M = D^{-1}$. Computing the trace of the inverse of the Dirac operator D^{-1} is required in many complicated problems in Lattice QCD [97].

Definition 5.1.
Assume that D admits a spectral decomposition of $D = U\Lambda U^{-1}$ and the function $f(D)$ is defined as:

$$f(D) = Uf(\Lambda)U^{-1}, \quad f(\Lambda) = \begin{bmatrix} f(\lambda_1) & & \\ & \ddots & \\ & & f(\lambda_n) \end{bmatrix} \tag{5.1}$$

Then the trace of $f(D)$ can be written as:

$$\mathrm{tr}(f(D)) = \mathrm{tr}(Uf(\Lambda)U^{-1}) = \mathrm{tr}(f(\Lambda)) = \sum_{i=1}^{n}(f(\lambda_i)) \tag{5.2}$$

In case of $f(D) = D^{-1}$ we have $f(\lambda_i) = 1/\lambda_i$.

Estimating the trace of a matrix $M \in \mathbb{C}^{n\times n}$ can be done by computing quadratic forms $\psi^H M\psi$, through one of the following strategies [6, 29, 60]:

- Estimator for fixed orthogonal basis Q which means choosing uniformly random vector ψ from the elements of an orthonormal matrix Q. The trace is then estimated to be $\psi^H M\psi$.

- Gaussian estimator by choosing the elements of ψ from a zero-mean and unit-variance Gaussian distribution. The elements must be independent and identically distributed (i.i.d). The trace is then estimated to be $\psi^H M\psi$.

- Hutchinson's estimator, the approach considered here, by choosing randomly, independently and identically distributed (i.i.d) the elements of ψ from a Rademacher distribution $\Pr\{\pm 1\}$. The trace is then estimated to be $\psi^H M\psi$.

In the case of $M = D^{-1}$ one can compute $D^{-1}\psi$ using a solver for D and then compute the inner product with ψ.

5.2 Hutchinson Estimator

The standard Monte Carlo approach for estimating the trace of the inverse of a matrix is due to Hutchinson [60]. This method is based on lemma 5.2, which states that the expected value of the inner product of a symmetric matrix M with a random vector ψ is equal to the trace of M:

Lemma 5.2.
Suppose a matrix $M \in \mathbb{C}^{n\times n}$ *and* $\psi = \Big[\psi_1 \dots \psi_n\Big] \in \mathbb{C}^n$ *a random vector with* $\mathbb{E}[\psi^H \psi] = I$. *Then*

$$\mathbb{E}\Big[\psi^H M \psi\Big] = \operatorname{tr}(M). \tag{5.3}$$

Proof. The proof is very simple as in [60]:

$$\mathbb{E}\Big[\psi^H M \psi\Big] = \sum_{i=1}^{n}\sum_{j=1}^{n} M_{ij}\mathbb{E}[\overline{\psi_i}\psi_j] = \sum_{i=1}^{n} M_{ii} = \operatorname{tr}(M) \tag{5.4}$$

where we have used the property $\mathbb{E}[\psi_i^H \psi_j] = \delta_{ij}$. This establishes the desired result (5.3).

□

Typically, one takes the components of ψ to be identically independently distributed (i.i.d.) complex numbers z with

$$\mathbb{E}[z] = 0 \quad \text{and} \quad \mathbb{E}[z^2] = 1. \tag{5.5}$$

A prominent example is the Rademacher vectors, where z takes the values $\{-1, 1\}$ with equal probability. Averaging $\psi^H D^{-1}\psi$ over s independent random vectors ψ gives an unbiased estimator for the trace.

$$\operatorname{tr}_s^H(M) := \frac{1}{s}\sum_{i=1}^{s} \psi^{(i)H} M \psi^{(i)}, \tag{5.6}$$

where s is the number of samples, and if the matrix M is real and symmetric the variance can be given as [60]:

$$\mathbb{V}\Big(\psi^H M \psi\Big) = 2\Big(\|M\|_F^2 - \sum_{i=1}^{n} M_{ii}^2\Big). \tag{5.7}$$

The accuracy of the Hutchinson estimator is proportional to the square root of the variance over the number of samples $\sqrt{\frac{\mathbb{V}(\psi^H M\psi)}{s}}$. Therefore, in order to achieve

a high level of accuracy, it is important to use a large number of samples and to choose the random vectors in such a way that the variance is minimized.

Algorithm 3 outlines the steps for using the Hutchinson estimator to estimate the trace of the inverse of a matrix D. The algorithm requires a tolerance level ϵ for the relative accuracy of the estimate. It involves generating a sequence of random vectors ψ_s that satisfy (5.10). For each vector, its inner product with $D^{-1}\psi_s$ is computed using some method for solving linear systems (multigrid in the algorithm). The estimates τ_s are then averaged to produce the overall estimate τ, and the Variance of the samples is computed. If the variance divided by the number of samples is less than the square of the tolerance level multiplied by the estimate τ, the algorithm terminates and returns the estimate τ as the final result. Otherwise, the process is repeated with the next random vector in the sequence.

Algorithm 3 Hutchinson method for estimating $\mathrm{tr}(D^{-1})$

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ϵ relative accuracy
Output: Approximation τ for $\mathrm{tr}(D^{-1})$

1: **for** $s = 1, 2, \ldots$ **do**
2: generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
3: $\tau_s \leftarrow \psi_s^* D^{-1}\psi_s$ ▷ use mg to solve lin. sys.
4: $\tau = \frac{1}{s}\sum_{i=1}^{s}\tau_i$, $\mathbb{V} = \frac{1}{s-1}\sum_{i=1}^{s}|\tau_i - \tau|^2$ ▷ sample mean and variance
5: **if** $\mathbb{V}/s \leq (\tau\epsilon)^2$ **then**
6: **stop**
7: **end if**
8: **end for**

5.2.1 Tail bounds for the Hutchinson estimator

One of the main advantages of the Hutchinson estimator for trace estimation is that it has well-known tail bounds in the case that D is symmetric and positive semidefinite. These tail bounds provide probabilistic results on the deviation of the estimator from the true trace and can be used to determine the number of samples needed to achieve a desired accuracy. Theorem 5.3 from [6] gives a generalization of these bounds to the allowed probability spaces see [29, 104].

Theorem 5.3.
Assume that the matrix $D \in \mathbb{R}^{n\times n}$ is symmetric and positive semidefinite. Let $\mathrm{tr}_s^H(D)$ denote the Hutchinson estimator with s samples which are Rademacher vectors. Let $\epsilon, \delta \in (0, 1)$. Then, if

$$s \geq \frac{6}{\epsilon^2}\log\frac{n}{\delta} \tag{5.8}$$

one has

$$\mathbb{P}\left(\left|\mathrm{tr}_s^H(D) - \mathrm{tr}(D)\right| \leq \varepsilon\,\mathrm{tr}(D)\right) \geq 1 - \delta; \tag{5.9}$$

In the case of randomized Gaussian vectors instead of Rademacher vectors, (5.9) holds if

$$s > \frac{20}{\epsilon^2}\log\frac{2}{\delta}$$

In [81] the number of samples s for randomized vectors is improved to be

$$s > \frac{6}{\epsilon^2}\log\frac{2}{\delta} \quad \text{and} \quad s > \frac{8}{\epsilon^2}\frac{\|D^{-1}\|_2}{\mathrm{tr}(D^{-1})}\log\frac{2}{\delta}$$

for randomized Rademacher vectors and Gaussian vectors, respectively.

Theorem 5.3 is a quantitative illustration of the crucial drawback of Monte Carlo trace estimation: The accuracy increases only with the square root of the number of samples as $O(1/\sqrt{s})$ and the convergence rate of Monte Carlo becomes very slow, which makes high accuracy estimates practically infeasible unless modifications are found which reduce the variance substantially.

In Chapter 6 we will discuss the most common methods for variance reduction of the Hutchinson estimator based on the projection idea.

5.2.2 Accuracy of trace estimation

The convergence analysis of trace estimation refers to the study of the rate at which the estimate of the trace of a matrix approaches the true trace as the number of samples used in the estimate increases. This is an important consideration in the use of Monte Carlo methods for trace estimation, as the accuracy of the estimate depends on the number of samples used.

Theorem 5.4 is a result about the expected value and variance of the sample mean τ of random variables $\psi^H D\psi$ [41]. It states that the expectation of the trace estimator τ is equal to the true trace of the matrix D. The variance of τ is calculated by taking the expected value of the squared difference between τ and the true trace of D. It is shown that this variance is equal to the sum of the product of the elements in the off-diagonal entries of D with the expected value of the product of four independent components of the random vector ψ.

Theorem 5.4.
Let $\mathbb{P} : \Omega \to [0, 1]$ *be a probability measure on a sample space* Ω *and assume that*

the components ψ_i of the vector $\psi \in \mathbb{C}^n$ are random variables depending on $\omega \in \Omega$ and obey an isotropic distribution, i.e.

$$\mathbb{E}[|\psi_i|^2] = 1, \;\; \mathbb{E}[\psi_i \psi_j] = 0 \text{ for } i, j = 1, \ldots, n, i \neq j. \tag{5.10}$$

Typically, one takes the components to be identically independent distribution (i.i.d.) complex numbers z with $\mathbb{E}[z] = 0$ and $\mathbb{E}[|z^2|] = 1$.

Assume the sample means $\tau = \psi^H D \psi$, then

$$\mathbb{E}[\tau] = \operatorname{tr}(D) \quad \text{and} \quad \mathbb{V}[\tau] = \sum_{\substack{i,j,k,p=1 \\ i \neq j, k \neq p}}^{n} \overline{d}_{ij} d_{kp} \mathbb{E}\big[\psi_i \overline{\psi_j} \overline{\psi}_k \psi_p\big]. \tag{5.11}$$

In particular, if the probability space is such that each component ψ_i is independent from ψ_j for $i \neq j$, then

$$\mathbb{V}[\tau] = \sum_{\substack{i,j \\ i \neq j}}^{n} \overline{d}_{ij} d_{ij} + \sum_{\substack{i,j \\ i \neq j}}^{n} \overline{d}_{ij} d_{ji} \mathbb{E}[\psi_i^2] \mathbb{E}[\overline{\psi}_j^2]. \tag{5.12}$$

The fact that $\mathbb{E}[\tau] = \operatorname{tr}(D)$ was proved in lemma 5.2.

For the variance we have

$$\begin{aligned}
\mathbb{V}[\tau] &= \mathbb{E}\left[\overline{(\tau - \operatorname{tr}(D))}(\tau - \operatorname{tr}(D))\right] \\
&= \mathbb{E}\big[\big(\sum_{\substack{i,j=1 \\ i \neq j}}^{n} \psi_i \overline{d}_{ij} \overline{\psi}_j\big)\big(\sum_{\substack{k,p=1 \\ k \neq p}}^{n} \overline{\psi}_k d_{kp} \psi_p\big)\big] \\
&= \mathbb{E}\big[\sum_{\substack{i,j,k,p=1 \\ i \neq j, k \neq p}}^{n} \overline{d}_{ij} d_{kp} \psi_i \overline{\psi}_j \overline{\psi}_k \psi_p\big] \\
&= \sum_{\substack{i,j,k,p=1 \\ i \neq j, k \neq p}}^{n} \overline{d}_{ij} d_{kp} \mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_k \psi_p].
\end{aligned} \tag{5.13}$$

Since the components ψ_i are assumed to be independent, we have

$$\mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_k \psi_p] = 0 \tag{5.14}$$

except when $i = j, k = p$ (which does not occur in (5.13)) or $i = k, j = p$ or

$i = p, j = k$. This gives

$$\sum_{\substack{i,j,k,p=1 \\ i\neq j, k\neq p}}^{n} \overline{d}_{ij} d_{kp} \mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_k \psi_p] = \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ij} \mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_i \psi_j] + \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ji} \mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_j \psi_i], \quad (5.15)$$

and in the first sum

$$\mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_i \psi_i] = \mathbb{E}[\overline{\psi}_i \psi_i] \mathbb{E}[\overline{\psi}_j \psi_j] = 1, \quad (5.16)$$

by assumption, whereas in the second sum we have

$$\mathbb{E}[\psi_i \overline{\psi}_j \overline{\psi}_j \psi_i] = \mathbb{E}[\psi_i^2] \mathbb{E}[\overline{\psi}_j^2]. \quad (5.17)$$

Note that as a definition for the variance of a complex random variable y we used $\mathbb{E}[(\overline{y - \mathbb{E}(y)})(y - \mathbb{E}[y])]$ rather than $\mathbb{E}[(y - \mathbb{E}[y])^2]$ to keep it real and non-negative.

Standard choices for the probability spaces are to take x with identically and independently distributed (i.i.d.) components as

$$\psi_i \in \{-1, 1\} \text{ with equal probability } \tfrac{1}{2}, \quad (5.18)$$
$$\psi_i \in \{-1, 1, -i, i\} \text{ with equal probability } \tfrac{1}{4}, \quad (5.19)$$
$$\psi_i = \exp(i\theta) \text{ with } \theta \text{ uniformly distributed in } [0, 2\pi], \quad (5.20)$$
$$\psi_i \text{ is } N(0, 1) \text{ normally distributed.} \quad (5.21)$$

For the first choice of probability space in (5.18), we have $\mathbb{E}[\psi_i] = 0$ and $\mathbb{E}[\overline{\psi}_i \psi_i] = 1$, so the first condition in (5.10) is satisfied. In addition, $\mathbb{E}[\overline{\psi}_i \psi_j] = 0$ for $i \neq j$, so the second condition in (5.10) is also satisfied. This means that the probability space defined in (5.18) is valid for use in Theorem 5.4.

For the second choice of probability space in (5.19), we have $\mathbb{E}[\psi_i] = 0$ and $\mathbb{E}[\overline{\psi}_i \psi_i] = 1$, so the first condition in (5.10) is satisfied. In addition, $\mathbb{E}[\overline{\psi}_i \psi_j] = 0$ for $i \neq j$, so the second condition in (5.10) is also satisfied. This means that the probability space defined in (5.19) is valid for use in Theorem 5.4.

For the third choice of probability space in (5.20), we have $\mathbb{E}[\psi_i] = 0$ and $\mathbb{E}[\overline{\psi} i \psi_i] = 1$, so the first condition in (5.10) is satisfied. In addition, $\mathbb{E}[\overline{\psi}_i \psi_j] = 0$ for $i \neq j$, so the second condition in (5.10) is also satisfied. This means that the probability space defined in (5.20) is valid for use in Theorem 5.4.

For the fourth choice of probability space in (5.21), we have $\mathbb{E}[\psi_i] = 0$ and $\mathbb{E}[\overline{\psi}_i \psi_i] = 1$, so the first condition in (5.10) is satisfied. However, $\mathbb{E}[\overline{\psi}_i \psi_j] = 0$ for $i \neq j$ does not hold.Therefore, the probability space defined in (5.21) is not valid for use in Theorem 5.4.

The variance of the random variable $\psi^H D\psi$ for the four different choices of probability spaces for the i.i.d. components ψ_i is given in Corollary 5.5.

Corollary 5.5.
If the components ψ_i are i.i.d. with distribution (5.18)*, then*

$$\mathbb{V}[\psi^H D\psi] = \frac{1}{2}\|\text{offdiag}(D + D^H)\|_F^2, \tag{5.22}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and offdiag *the offdiagonal part of a matrix. If the components are i.i.d. with distribution* (5.19) *or* (5.20)*, then the variance is*

$$\mathbb{V}[\psi^H D\psi] = \|\text{offdiag}(D)\|_F^2. \tag{5.23}$$

In the case of the probability distribution (5.21) *the variance is*

$$\mathbb{V}[\psi^H D\psi] = \frac{1}{2}\|D + D^T\|_F^2. \tag{5.24}$$

Proof. For the distributions (5.18) and (5.21), the components ψ_i have only real values and $\mathbb{E}[\psi_i^2] = 1$. Therefore

$$\begin{aligned}
\sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ij} + \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ji} \mathbb{E}[\psi_i^2]\mathbb{E}[\overline{\psi}_j^2] &= \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ij} + \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ji} \\
&= \frac{1}{2} \sum_{\substack{i,j \\ i\neq j}}^{n} (\overline{d_{ij} + d_{ji}})(d_{ij} + d_{ji}) \\
&= \frac{1}{2}\|\text{offdiag}(D + D^H)\|_F^2. \qquad (5.25)
\end{aligned}$$

For the distributions (5.19) and (5.20) we have $\mathbb{E}[\psi_i^2] = 0$, and thus

$$\sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ij} + \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ji} \mathbb{E}[\psi_i^2]\mathbb{E}[\overline{\psi}_j^2] = \sum_{\substack{i,j \\ i\neq j}}^{n} \overline{d}_{ij} d_{ij} = \|\text{offdiag}(D)\|_F^2. \tag{5.26}$$

□

Remark 5.6.
The variance of the estimator $\psi^H D\psi$ can be used to determine the required number of samples s in order to achieve a certain accuracy. For example, if we want the relative error of the estimator to be less than ϵ, we can set

$$\frac{\mathbb{V}[\psi^H D\psi]}{(\mathbb{E}[\psi^H D\psi])^2} \leq \frac{\epsilon^2}{4}$$

and solve for s. This gives us a bound on the number of samples needed to achieve the desired accuracy.

We can obtain the estimation of $\mathrm{tr}(D)$ by averaging over N samples. In order to evaluate the probability that the computed mean lies within a range of $\sigma, 2\sigma$ or 3σ interval, we calculate the *sample* root mean square deviation alongside the averages and rely on the law of large numbers. As an asymptotic extension of Hutchinson's method, several results have been proposed, which provide tail or concentration bounds; see [6, 21, 81], e.g. Here we provide a summary of the results outlined in [70]. We utilize the *sample* root mean square deviation to evaluate the accuracy of these results.

The following theorem provides a probabilistic bound on the accuracy of approximating the trace of a matrix using the average of $(\psi^{(i)})^H D\psi^{(i)}$ over a number of samples s. It is useful for cases where the exact trace of the matrix is difficult or impossible to compute directly.

Theorem 5.7.
Let the distribution for the i.i.d. components of the random vectors $\psi^{(i)}$ be sub-Gaussian, and let $\epsilon, \delta \in (0,1)$. Then for $s = \mathcal{O}(\log(1/\delta)/\epsilon^2)$ we have that the probability for

$$\left| \frac{1}{s} \sum_{i=1}^{s} (\psi^{(i)})^H D\psi^{(i)} - \mathrm{tr}(D) \right| \leq \epsilon \|D\|_F \tag{5.27}$$

is $\geq 1 - \delta$.

Note that if D is symmetric positive semidefinite with λ_i denoting its (non-negative) eigenvalues, then

$$\|D\|_F = \left(\sum_{i=1}^{n} \lambda_i^2 \right)^{1/2} \leq \sum_{i=1}^{n} \lambda_i = \mathrm{tr}(D), \tag{5.28}$$

implying that (5.27) yields a (probabilistic) relative error bound for the trace. Also note that the real distributions (5.18) and (5.21) are sub-Gaussian; see [70].

One can observe that Theorem 5.3 deals with the Hutchinson estimator, which uses Rademacher or Gaussian random vectors to approximate the trace of a positive semidefinite matrix. On the other hand, Theorem 5.7 concerns the sample mean estimator, which uses sub-Gaussian random vectors to approximate the trace of a matrix. Both theorems provide upper bounds on the error of the estimators with a certain probability guarantee. However, they differ in the specific types of random vectors used and the assumptions made on the matrix.

Chapter 6

Variance Reduction Methods

The variance of the Hutchinson estimator for $\mathrm{tr}(D^{-1})$ is given by

$$\tfrac{1}{2}\|\mathrm{offdiag}(D^{-1} + D^{-H})\|_F^2$$

for Rademacher vectors, and it is $\|\mathrm{offdiag}(D^{-1})\|_F^2$ for Z_4-vectors; see [41], and the heuristics underlying variance reduction techniques typically rely on just reducing $\|D^{-1}\|_F^2$. Reducing the variance can be achieved by using variance reduction techniques to reduce the variance of the Hutchinson estimator and improve the estimate's accuracy for the trace of D^{-1}.

The most common methods for reducing the variance of the plain Hutchinson estimator, in relation to the thesis, are covered in detail in this chapter. We provide an overview of the underlying theory of the deflation approach, which we use as a standard of comparison in our numerical experiments in Sec. 6.1.1.

In sections 6.1.2 and 6.1.3, we explore the two main types of deflation for reducing the variance. Then we discuss in Sec. 6.2, the Hutch++ method [70] as a new estimator since we aim to merge it with our MG-MLMC approach to develop the most efficient algorithm MG-MLMC++ in Sec. 7.3.3. The optimal way to obtain the deflated subspace can be found using the recently adaptive version of Hutch++ [76] which we finally discuss in Sec. 6.3.

6.1 Deflation Approach

The deflation technique is a method for improving the convergence of iterative algorithms by removing certain parts of an operator that make it "ill-conditioned". This technique was first proposed by Nicolaides in 1987 [74] in context of CG

method and has since been widely used in a variety of methods. Deflation can be employed in two different manners:

1. To solve linear systems of equations of the form $D\psi = b$, where D is a matrix. The convergence of these equations is strongly influenced by the distribution of eigenvalues of D. If D has eigenvalues that are close to the origin of the complex plane, the convergence of the linear equations can be slow. Removing the smallest eigenvalues of D can reduce the condition number of the matrix, thus speeding up the convergence. The Arnoldi method is commonly used to construct an orthogonal basis of the Krylov subspace for approximating the eigenpairs of D, especially when D is a large sparse matrix. However, the Restarted Arnoldi (RA) method can lead to slow convergence for more complicated problems, due to small subspaces. Deflated strategies of RA can overcome this issue by removing a few small eigenvalues [72]. Deflation techniques have been observed to improve the convergence rates of iterative methods when adding a few approximate eigenvectors to the Krylov subspaces [20, 25, 30, 64, 73, 86]. For more details, see [89].

2. As a variance reduction method for estimating traces stochastically. Deflation techniques have been used to handle exceptional eigenvalues. The matrix is decomposed into a sum of two parts using a projection on a subspace of small dimension. One part has large rank but reduced the Frobenius norm, thus reducing the effort for stochastic estimation. The other part has a small rank, so that its trace can be computed directly. [42, 43, 45, 80].

However, in large systems such as Lattice QCD simulations, the cost of eigenvector storage is one of the main drawbacks of deflation.

6.1.1 Main idea of the deflation approach

Consider an $n \times n$ algebraic linear system

$$D\psi = b \tag{6.1}$$

where $D \in \mathbb{C}^{n\times n}$ is a general matrix. Assume two projectors are given as

$$R = I - DZ(Y^H DZ)^{-1}Y^H, \tag{6.2}$$
$$Q = I - Z(Y^H DZ)Y^H D \tag{6.3}$$

where the columns of the matrices Z and Y span suitable subspaces [31]. The projectors Q and R have the sizes $n \times n$ and satisfy the following properties:

$$\begin{aligned} R^2 &= R, \quad Q^2 = Q, \quad RD = DQ, \\ & RDZ = Y^H R = 0, \quad Y^H DQ = QZ = 0 \end{aligned} \tag{6.4}$$

These properties are often referred to as the projection properties of Q and R. They ensure that Q and R act as projections onto certain subspaces, as indicated by the zero entries in the last two equations. The first two equations ensure that Q and R are idempotent, meaning that applying them twice is the same as applying them once. The deflation approach allows us to decompose the original linear system (6.1) into two smaller linear systems. The key idea is to use projectors to split the linear system into subproblems that can be solved separately, one system is obtained by applying the projector Q to both sides of the original system, while the other system is obtained by applying the projector $I - Q$ to both sides of the original system.

More specifically, the first smaller linear system is given as

$$(DQ)\psi = (DQ)b \tag{6.5}$$

This system can be derived as follows:

$$\begin{aligned} D\psi &= b \\ DQ\psi + D(I-Q)\psi &= b \\ DQ\psi &= b - D(I-Q)\psi \\ (DQ)\psi &= (D(I-Q))b \end{aligned} \tag{6.6}$$

The last equation shows that the system (6.5) is equivalent to the original system (6.1). In other words, if we can solve the system (6.5), we can obtain the solution to the original system. The second smaller linear system is given as

$$(D(I-Q))\psi = (D(I-Q))b. \tag{6.7}$$

This system can be derived in a similar way:

$$\begin{aligned} D\psi &= b \\ DQ\psi + D(I-Q)\psi &= b \\ D(I-Q)\psi &= b - DQ\psi \\ (D(I-Q))\psi &= (DQ)^{\perp}b \end{aligned} \tag{6.8}$$

where $(DQ)^{\perp}$ denotes the orthogonal complement of the range of DQ. The last equation shows that the system $(D(I-Q))\psi = (D(I-Q))b$ is equivalent to (6.1).

In other words, if we can solve the system $(D(I-Q))\psi = (D(I-Q))b$, we can obtain the solution to the original system. Thus, by solving the two smaller linear systems

$$(DQ)\psi = (DQ)b \text{ and } (D(I-Q))\psi = (D(I-Q))b, \tag{6.9}$$

we can obtain the solution to the original system (6.1). Now, let's consider the second linear system $(D(I-Q))\psi = (D(I-Q))b$. Using the projection properties of Q and R, we can rewrite this system as

$$D\psi - DQ\psi = b - DQb. \tag{6.10}$$

Substituting $Q\psi$ from the first linear system, we have

$$\begin{aligned} D\psi - DR\psi &= b - DQb \\ D\psi - D(DZ(Y^HDZ)^{-1}Y^H)\psi &= b - D(DZ(Y^HDZ)^{-1}Y^H)b \\ (I - DZ(Y^HDZ)^{-1}Y^H)\psi &= Z(Y^HDZ)^{-1}Y^Hb. \end{aligned} \tag{6.11}$$

Thus, we can compute the first term $(I-Q)\psi$ by solving the linear system

$$(I - DZ(Y^HDZ)^{-1}Y^H)\psi = Z(Y^HDZ)^{-1}Y^Hb. \tag{6.12}$$

Once we have computed both terms $(I-Q)\psi$ and $Q\psi$, we can obtain the solution ψ to the original linear system $D\psi = b$ as $\psi = (I-Q)\psi + Q\psi$.

so, if we take $\Pi = R$ from (6.2) we have $D^{-1}(I-\Pi)b = Z(Y^HDZ)^{-1}Y^Hb$, which requires small subspace with Y^HDZ to be solved and $D^{-1}\Pi b$ is solved iteratively.

This approach can be especially useful when the original matrix has certain structures or properties that can be exploited to make the solution of the smaller systems more efficient.

Theorem 6.1.
Suppose a proper projector $\Pi \in \mathbb{C}^{n\times k}$ applying on a matrix $D \in \mathbb{C}^{n\times n}$, then the inverse of the matrix D^{-1} can be decomposed into two parts as

$$D^{-1} = (I-\Pi)D^{-1} + \Pi D^{-1} \tag{6.13}$$

Proof. We start with the definition of a projection operator: a matrix Π is a projector if it is idempotent, meaning that $\Pi^2 = \Pi$. Note that a projector Π is proper if it is not the identity matrix.

Next, we want to show that D^{-1} can be decomposed into two parts as in equation (6.13), where Π is a proper projector acting on D.

Using the properties of a projection operator, we have:

$$\begin{aligned} \Pi^2 &= \Pi \\ (I-\Pi)^2 &= I - 2\Pi + \Pi^2 = I - \Pi \end{aligned} \tag{6.14}$$

Multiplying the second equation by Π, we get:

$$\Pi(I-\Pi) = 0 \tag{6.15}$$

Now we can apply this to D^{-1}:

$$\begin{aligned} &\Pi(I-\Pi)D^{-1} = 0 \\ \Rightarrow\ &\Pi D^{-1} + (I-\Pi)D^{-1} = D^{-1} \end{aligned} \tag{6.16}$$

This shows that D^{-1} can be decomposed into two parts: $(I-\Pi)D^{-1}$ and ΠD^{-1}.

To see why these two parts add up to D^{-1}, we can compute their sum:

$$\begin{aligned} ((I-\Pi)D^{-1} + \Pi D^{-1})D &= (I-\Pi)DD^{-1} + \Pi D^{-1}D \\ &= D^{-1}(D - \Pi D) + \Pi \\ &= D^{-1}D - \Pi D^{-1}D + \Pi \\ &= I \end{aligned} \tag{6.17}$$

Therefore, we have shown that D^{-1} can be decomposed into $(I-\Pi)D^{-1}$ and ΠD^{-1}, as in equation (6.13), for any proper projector Π acting on D.

□

The success of deflation approaches depends on how many small eigenvectors D has i.e thus large eigenmodes of D^{-1}, and it will become increasingly inefficient if the number of large eigenvectors increase with the dimension of D^{-1} ("volume dependence"). We first present the exact strategy of deflation based on orthogonal projection, the projection Π is built from exact eigenvectors belonging to small eigenvalues [45], wheres in inexact approaches we take possibly quite inaccurate approximations to such eigenvectors [80]. A different approach is thus to construct an appropriate Π as an operator with a relatively high dimension

that (approximately) represents a given percentage of the eigenvectors, see first attempts in [10].

6.1.2 Exact deflation method

The convergence rate of the solution of linear systems can be significantly slowed down if the eigenvalues of the matrix are close to the origin of the complex plane. So exact deflation, widely used in Lattice QCD see [2, 3, 23], is employed to overcome this problem as follows:

Theorem 6.2.
Assume that the random vectors are drawn by using Z_2 noise as in the Hutchinson approach, then the variance of the estimator is given by the square of the Frobenius norm of the off-diagonal part.

$$\mathbb{V}(\psi^H D\psi) = \frac{1}{2}||\text{offdiag}(D + D^T)||_F^2 \tag{6.18}$$

If the singular value decomposition of the non-singular matrix D is given as

$$\begin{aligned} D &= V\Sigma W^H \quad \text{with } V, \Sigma, W \in \mathbb{C}^{n\times n}, V^H V = W^H W = I, && (6.19)\\ & V = [v_1|\cdots|v_n],\ W = [w_1|\ldots|w_n], \\ & \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n), 0 < \sigma_1 \leq \cdots \leq \sigma_n, && (6.20) \end{aligned}$$

where u_i left singular vectors, w_i right singular vectors and positive singular values σ_i which we ordered by increasing value for convenience here. Then the relation between the Frobenius norm of $D \in \mathbb{C}^{n\times n}$ and its singular values as:

$$\|\text{offdiag}(D)\|_F^2 = \sum_{i=1}^{n} \sigma_i^2 - \sum_{i=^1}^{n} |d_{ii}|^2, \tag{6.21}$$

Proof. The proof of Theorem 6.2 relies on the fact that the Frobenius norm of a matrix can be expressed in terms of its singular values. Recall that the Frobenius norm of a matrix $A \in \mathbb{C}^{n\times n}$ is defined as $\|A\|_F = \sqrt{\text{tr}(A^H A)}$. We can express the Frobenius norm of the off-diagonal part of D as follows:

$$\|\text{offdiag}(D)\|_F^2 = \|D\|_F^2 - \sum_{i=1}^{n} \|d_{ii}\|^2, \tag{6.22}$$

Next, we can express $\|D\|_F^2$ in terms of the singular values of D using the singular value decomposition of D given in eq. (6.19).

$$\begin{aligned} |D|_F^2 &= \mathrm{tr}(D^H D) \\ &= \mathrm{tr}(W\Sigma^2 W^H) \\ &= \sum_{i=1}^{n} \sigma_i^2. \end{aligned} \tag{6.23}$$

Substituting this expression into the equation for $\|\mathrm{offdiag}(D)\|_F^2$ gives us the desired result.

$$\begin{aligned} \|\mathrm{offdiag}(D)\|_F^2 &= \sum_{i=1}^{n} \sigma_i^2 - \sum_{i=1}^{n} |d_{ii}|^2 \\ &= \sum_{i=1}^{n} \sigma_i^2 - \sum_{i=1}^{n} |a_{ii}|^2. \end{aligned} \tag{6.24}$$

This completes the proof of Theorem 6.2. □

The exact deflation is given as $D^{-1}v_i = \lambda_i v_i$ where $i = 1, \ldots, s.$ eigenpairs, and the orthogonal projector

$$\Pi = V_s(W_s^H D V_s)^{-1} W_s^H D = V_s V_s^H, \tag{6.25}$$

where $W_s = [w_1|\cdots|v_s]$ and $V_s = [v_1|\cdots|v_s]$ with w_i, v_i the left and right singular vectors for the singular values σ_i of D. Then the trace of D^{-1} can be split as

$$\mathrm{tr}(D^{-1}) = \mathrm{tr}((I - \Pi)D^{-1}) + \mathrm{tr}(\Pi D^{-1}). \tag{6.26}$$

The first term is computed stochastically as in Alg. 4, whereas the second term is computed directly as

$$\mathrm{tr}(\Pi D^{-1}) = \mathrm{tr}(V_s^H D^{-1} V_s) = \sum_{i=1}^{s} \frac{1}{\sigma_i} u_i^H v_i. \quad , \quad V_s^H D^{-1} V_s \in \mathbb{C}^{s\times s} \tag{6.27}$$

If instead W_s and V_s contain the left and right eigenvectors belonging to the smallest eigenvalues λ_i of D, then the oblique projector

$$\Pi = V_s(W_s^H D V_s)^{-1} W_s^H D = V_s W_s^H \tag{6.28}$$

achieves the trace

$$\mathrm{tr}(\Pi D^{-1}) = \mathrm{tr}(W_s^H D^{-1} V_s) = \sum_{i=1}^{s} \frac{1}{\lambda_i}. \tag{6.29}$$

If D is Hermitian and positive definite, the two deflation approaches coincide, since then left and right eigenvectors as well as left and right singular vectors all coincide, and the singular values are the eigenvalues.

The exact deflation method using eigenpairs to estimate the trace of the inverse of a matrix D can be formulated as in Alg. 4. It does so by computing the "low rank" part of the trace, denoted as τ^{lr}, using the exact eigenpairs of the matrix D. The algorithm also computes a stochastic estimate of the trace, denoted as τ, by generating random vectors and projecting them onto the null space of D. The algorithm stops when the variance of the estimate, denoted as $\mathbb{V}$, is less than a certain threshold, which is a function of the relative accuracy parameter ϵ and the sum of the low rank and stochastic estimates of the trace.

Algorithm 4 Exact Deflation

Input: $D \in \mathbb{C}^{n\times n}$, nonsingular, ϵ relative accuracy, k number of deflation vectors
Output: Approximation $\tau^{\mathrm{lr}} + \tau$ for $\mathrm{tr}(D^{-1})$

1: compute eigenpairs $Dq_i = \lambda_i q_i,\ i = 1, \ldots, k$
2: $\Pi \leftarrow Q_k Q_k^H$, $Q_k = [q_1|\cdots|q_d]$ ▷ *orthogonal projector*
3: $\tau^{\mathrm{lr}} \leftarrow \sum_{i=1}^{k} \frac{1}{\lambda_i}$ ▷ *low rank part*
4: **for** $s = 1, 2, \ldots$ **do** ▷ *stochastic part*
5: generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
6: $z_s \leftarrow \psi_s - Q(Q^H \psi_s)$ ▷ projected vector
7: $\tau_s \leftarrow \psi_s^H D^{-1} z_s$ ▷ use mg to solve linear system
8: $\tau \leftarrow \frac{1}{s}\sum_{i=1}^{s} \tau_i$, $\mathbb{V} \leftarrow \frac{1}{s-1}\sum_{i=1}^{s} |\tau_i - \tau|^2$ ▷ sample mean and variance
9: $\rho = (\tau + \tau^{\mathrm{lr}})\epsilon$
10: **if** $\mathbb{V}/s \leq \rho^2$ **then**
11: **stop**
12: **end if**
13: **end for**

6.1.3 Inexact Deflation

The exact deflation method is still quite expensive due to two factors: first, one must compute the singular pairs or eigenpairs in advance; second, we must increase the number of s as the size of the matrix increases in order to maintain the same variance reduction. Inexact deflation can be used as an alternative to

exact deflation, where one can use a large value s to reduce the variance roughly. In [80] they tried to reduce the computational cost of exact deflation and speed up the convergence rate by using an inexact eigensolver on the Hermitian Dirac-Wilson operator to estimate the space of the deflation. In the inexact deflation approach, we replace the projector Π with a more general form.

Definition 6.3.
Given V_s and W_s a number of approximate left and right singular vectors of the largest singular values of D^{-1}, we can form the general projector

$$\Pi = V_s(W_s^H D V_s)^{-1} W_s^H D, \quad V_s, W_s \in \mathbb{C}^{n\times s} \tag{6.30}$$

and split the trace computation into two parts as in (6.13).

Let's consider the projection operator Π defined in Definition 6.3. Since V_s and W_s are approximate singular vectors of D^{-1}, we can write them as a perturbation of the true singular vectors V and W as

$$V_s = V(I + \delta V), \tag{6.31}$$
$$W_s = W(I + \delta W), \tag{6.32}$$

where δV and δW are matrices representing the perturbation.

Substituting these expressions into the definition of the projection operator, we have

$$\begin{aligned}
\Pi =& V_s(W_s^H D V_s)^{-1} W_s^H D \\
=& V_s(I+\delta V_s)(W_s^H(I+\delta W_s) D V_s(I+\delta V_s))^{-1} W_s^H(I+\delta W_s) D \\
=& V_s(I+\delta V_s)(W_s^H D V_s W_s^H D + W_s^H D \delta V_s W_s^H D \\
&+ W_s^H \delta W_s D V_s + W_s^H \delta W_s D \delta V_s)^{-1} W_s^H (I+\delta W_s) D \\
=& V_s(I+\delta V_s)(W_s^H D V_s W_s^H D)^{-1} W_s^H (I+\delta W_s) D \\
&+ V_s(I+\delta V_s)(W_s^H D V_s W_s^H D)^{-1} W_s^H D \delta V_s (W^H D V_s W_s^H D)^{-1} W_s^H (I+\delta W_s) D \\
&+ \cdots
\end{aligned} \tag{6.33}$$

where we have used the identity

$$(A+B)^{-1} = A^{-1} - A^{-1}BA^{-1} + A^{-1}BA^{-1}BA^{-1} - \cdots . \tag{6.34}$$

We can then rearrange the terms in the above expression to obtain

$$\Pi = V_s(I+\delta V_s)(W_s^H D V_s W_s^H D)^{-1} W_s^H (I+\delta W_s) D + \text{higher order terms.} \tag{6.35}$$

Therefore, we can see that the projection operator Π defined in Definition 6.3 is a perturbation of the projection operator defined in Theorem 6.1.

Now we have

$$\mathrm{tr}(D^{-1}) = \mathrm{tr}(\underbrace{D^{-1} - V_s(W_s^H D V_s)^{-1} W_s^H}_{(I-\Pi)D^{-1}}) + \mathrm{tr}(\underbrace{V_s(W_s^H D V_s)^{-1} W_s^H}_{\Pi D^{-1}}) \tag{6.36}$$

Now, $\mathrm{tr}(\Pi D^{-1})$ is not directly available from approximate singular triplets or eigenvalues and there is a difference between the projector $V_s V_s^H$ and the projector $V_s(W_s^H D V_s)^{-1} W_s^H D$ as follows :

Using the projector $V_s V_s^H$ requires s system solves with the large matrix D for

$$\mathrm{tr}(\Pi D^{-1}) = \mathrm{tr}(V_s^H D^{-1} V_s), \quad V_s^H D^{-1} V_s \in \mathbb{C}^{s \times s}, \tag{6.37}$$

whereas in the case of building the projector

$$\Pi = V_s(W_s^H D V_s)^{-1} W_s^H D \tag{6.38}$$

it requires the inversion of a small $s \times s$ matrix,

$$\mathrm{tr}(\Pi D^{-1}) = \mathrm{tr}(V_s(W_s^H D V_s)^{-1} W_s^H). \tag{6.39}$$

If we take larger values for s, we can estimate $\mathrm{tr}(\Pi D^{-1})$ with the projector (6.38) stochastically as in Alg. 5. The inexact deflation approach then becomes a two-level Monte-Carlo method. The decomposition in (6.36) is also used in the Hutch ++ method [70], then the columns of V_s, W_s are taken to be stochastic vectors see section 6.2.

Alg. 5, "Inexact Deflation", is similar to Alg. 4 but instead of using the exact eigenpairs of the matrix D, it uses a projection based on the singular values of the matrix.

6.2 Hutch++

Hutch++ is a method that can be used to estimate the trace of square matrices implicitly through matrix-vector products by merging two phases, the initial phase involves randomized low-rank approximation followed by stochastic trace estimation [70]. The known theory for Hutch++ requires the given matrix D as a symmetric positive semidefinite, but the method can be applied to any given

Algorithm 5 Inexact Deflation

Input: $D \in \mathbb{C}^{n\times n}$, nonsingular, ϵ relative accuracy, s number of deflation vectors
Output: Approximation $\tau^{\text{lr}} + \tau$ for $\text{tr}(D^{-1})$

$\Pi \leftarrow V_s(W_s^H D V_s)^{-1} W_s^H D, \quad V_s, W_s \in \mathbb{C}^{n\times k}$ ▷ *Proj. vec., based on sing. values*
$\tau^{\text{lr}} \leftarrow \text{tr}(V_s^H (W_s^H D V_s)^{-1} W_s^H)$ ▷ *low rank* $V_s = [v_1|\cdots|v_s], W_s = [w_1|\cdots|w_s]$,
for $s = 1, 2, \ldots$ **do** ▷ *stochastic part*
  generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
  $z_s \leftarrow D^{-1}\psi_s - \Pi\psi_s$ ▷ projected vector
  $\tau_s \leftarrow \psi_s^H z_s$ ▷ use mg to solve linear system
  $\tau \leftarrow \frac{1}{s}\sum_{i=1}^{s} \tau_i$, $\mathbb{V} \leftarrow \frac{1}{s-1}\sum_{i=1}^{s} |\tau_i - \tau|^2$ ▷ sample mean and variance
  $\rho = (\tau + \tau^{\text{lr}})\epsilon$
  if $\mathbb{V}/s \le \rho^2$ **then**
    stop
  end if
end for

matrix. Additionally, Hutch++, to approximate the trace within a relative error ϵ, requires $\mathcal{O}(\epsilon^{-1})$ MVPs with a high probability, while using the plain Hutchinson estimator, requires $\mathcal{O}(\epsilon^{-2})$ MVPs. Hutch++ assigns a predetermined, number of MVPs and distributes them to each of the two phases.

The main idea of Hutch++ is to project off the top eigenvalues by using a randomized projection and approximate the trace of the remainder which is due to all the errors. The theory of Hutch++ can be achieved by introducing the lemma [70]:

Lemma 6.4.
Assume $\text{tr}_s^H(D)$ *be the Hutchinson estimator for the matrix* $D \in \mathbb{C}^{n\times n}$*, with probability* $\delta \in (0, 1/2]$*. If* $s \ge c\log(1/\delta)$*, for fixed constants* c, C*, then with probability* $1 - \delta$

$$\Pr(|\,\text{tr}_s^H(D) - \text{tr}(D)|) \le C\sqrt{\frac{\log(1/\delta)}{s}}\|D\|_F. \tag{6.40}$$

Thus, if $s = O(\log(1/\delta)/\varepsilon^2)$ *then, with probability* $1 - \delta$, $|\,\text{tr}_s^H(D) - \text{tr}(D)| \le \varepsilon\|D\|_F$.

In case of D is PSD i.e $D \succeq 0$ then estimation trace given as

$$\text{tr}(D) = \|D\|_F = \|\lambda\|_2 \le \|\lambda\|_1, \text{ with condition } \|\lambda\|_2 \approx \|\lambda\|_1$$

Based on the low-rank approach, another lemma was established as follows:

Lemma 6.5.
Suppose a PSD matrix D and D_d be the best rank approximation, then

$$\|D - D_d\|_F \leq \frac{1}{\sqrt{d}}\mathrm{tr}(D). \tag{6.41}$$

Proof. Since $\lambda_{d+1} \leq \frac{1}{d}\sum_{i=1}^{d}\lambda_i \leq \frac{1}{d}\,\mathrm{tr}(D)$, the proof proceeds as follows:

First, we note that for a PSD matrix, the singular values are equal to the eigenvalues, and that the best rank-d approximation to D can be obtained by truncating the singular value decomposition of D after the d largest singular values:

$$D = U\Sigma U^T \approx U_d\Sigma_d U_d^T = D_d, \tag{6.42}$$

where U is an orthogonal matrix, Σ is a diagonal matrix of singular values, and U_d and Σ_d contain only the first d columns/rows of U and Σ, respectively. Then, we can write the squared Frobenius norm of the difference between D and D_d as follows:

$$\begin{aligned}
\|D - D_d\|_F^2 &= \|U(\Sigma - \Sigma_d)V^T\|_F^2 \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\|u_{ij}(\sigma_i - \sigma_{d,i})v_{ij}\|^2 \\
&= \sum_{i=d+1}^{n}(\sigma_i - \sigma_{d,i})^2 \\
&\leq (\sigma_{d+1})^2\sum_{i=d+1}^{n}1 \\
&= (\sigma_{d+1})^2(n-d) \\
&\leq \frac{1}{d^2}\left(\sum_{i=1}^{n}\sigma_i\right)^2 \\
&= \frac{1}{d^2}\,\mathrm{tr}(D)^2,
\end{aligned} \tag{6.43}$$

where the second line follows from the SVD of D and D_d, the third line follows from squaring and summing over the matrix entries, the fourth line uses the fact that $\sigma_i \geq \sigma_{d+1}$ for $i \geq d+1$ and $|u_{ij}|^2 = |v_{ij}|^2 = 1$, the fifth line uses the fact that there are $n-d$ terms being summed, and the last line follows from using the definition of the trace.

□

A possible algorithm could be developed based on this result with $O(1/\varepsilon)$ complexity. It works as follows: we precompute $y_i := D^{-1}s_i$ for d i.i.d. random vectors s_i. Then the range of the vectors y_i contains, with high probability, good approximations to eigenvectors belonging to large eigenvalues of D^{-1}. Therefore, with $V \in \mathbb{C}^{n\times d}$ denoting the matrix whose columns form an orthonormal basis of the space spanned by the y_i and $Q = VV^H$ the orthogonal projector onto the range of V, we can decompose the inverse of the matrix as

$$D^{-1} = D^{-1}Q + D^{-1}(I - Q) \tag{6.44}$$

The trace of the first term $D^{-1}Q$ can be computed directly using the cyclic property of the trace as

$$\operatorname{tr}(D^{-1}Q) = \operatorname{tr}(V^H D^{-1} V) = \sum_{i=1}^{d} v_i^H D^{-1} v_i, \text{ where } V = [v1|\cdots|v_d]. \tag{6.45}$$

The trace of the second term can be estimated stochastically as in Alg. 6. Due to the fact that $(I - Q)$ approximately deflates the largest eigenpairs of D^{-1}, we anticipate this term to have a smaller variance. The singular value distribution of D^{-1} significantly determine how successfully the two phases of Hutch++ work. It would be sufficient to implement the estimation $\operatorname{tr}(D^{-1}) \approx \operatorname{tr}(D_1^{-1})$ if D^{-1} implies an accurate low-rank approximation and one can skip the second part of eq. (6.45). Whereas reducing the variance through Hutch++ is insignificant if all singular values of D^{-1} are approximately equal and all efforts will be spent on the second phase.

So by combining Lemmas 6.4 and 6.5, the error of the trace estimator $\operatorname{tr}_s^{hpp}(D)$ becomes:

$$\Pr(|\operatorname{tr}_s^{\mathrm{hpp}}(D) - \operatorname{tr}(D)|) \leq C\sqrt{\frac{\log(1/\delta)}{s}}\|D\|_F \tag{6.46}$$

Therefore, for fixed δ, $s = O(1/\varepsilon)$ would be enough for an (ε, δ)-approximator to $\operatorname{tr}(D)$.

Alg. 6 shows how Hutch++ is used for estimating the trace of the inverse of a nonsingular matrix $D \in \mathbb{C}^{n\times n}$. It starts by generating k random vectors and solving k linear systems to compute the matrix Y. Then, it computes the QR factorization of Y and uses the resulting orthogonal matrix Q to compute the low-rank part of the trace estimate. In the stochastic part of the algorithm, it generates additional random vectors, projects them onto the orthogonal complement of the space spanned by the columns of Q, and solves linear systems to compute the trace estimate. The algorithm terminates when the variance of the trace estimate divided by the number of iterations falls below a certain threshold.

Algorithm 6 Hutch++: for estimating $\mathrm{tr}(D^{-1})$

Input: $D \in \mathbb{C}^{n\times n}$, nonsingular, ϵ relative accuracy, k number of deflation vectors
Output: Approximation $\tau^{\mathrm{lr}} + \tau$ for $\mathrm{tr}(D^{-1})$

1: generate k i.i.d. random vectors, ▷ with distribution satisfying (5.10)
2: collect them as columns in $S \in \mathbb{C}^{n\times k}$
3: $Y \leftarrow D^{-1}S$, ▷ $Y \in \mathbb{C}^{n\times k}$, use mg for k lin. systems
4: Compute QR-factoriz. $Y = QR$ ▷ $Q = [q_1|\cdots|q_k] \in \mathbb{C}^{n\times k}$ has orthon. cols
5: $\tau^{\mathrm{lr}} \leftarrow \sum_{i=1}^{k} q_i^H D^{-1} q_i$ ▷ *low rank part*, use mg to solve lin. systems
6: **for** $s = 1, 2, \ldots$ **do** ▷ *stochastic part*
7: generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
8: $z_s \leftarrow \psi_s - Q(Q^H\psi_s)$ ▷ projected vector
9: $\tau_s \leftarrow \psi_s^H D^{-1} z_s$ ▷ solve linear system
10: $\tau \leftarrow \frac{1}{s}\sum_{i=1}^{s}\tau_i$, $\mathbb{V} \leftarrow \frac{1}{s-1}\sum_{i=1}^{s}|\tau_i - \tau|^2$ ▷ sample mean and variance
11: $\rho = (\tau + \tau^{\mathrm{lr}})\epsilon$
12: **if** $\mathbb{V}/s \le \rho^2$ **then**
13: **stop**
14: **end if**
15: **end for**

For symmetric positive definite matrix D, the number of matvecs needed to reach relative accuracy ϵ goes from $\mathbb{O}(\frac{1}{\epsilon^2})$ to $\mathbb{O}(\frac{1}{\epsilon})$ if k is chosen approximately [70].

Alg.. 6 requires $\frac{2s}{3}$ MVPs for the first phase, with D^{-1} : $D^{-1}S$ in line 3 of the Algorithm and $D^{-1}Q$ for computing the trace of the low-rank part $\mathrm{tr}(Q^H D^{-1} Q)$ in line 5. It uses the rest of $\frac{s}{3}$ MVPS for the second phase, which is concerned with estimating

$$\mathrm{tr}(D^{-1} - QQ^H D^{-1}) = \mathrm{tr}((I - QQ^H)D^{-1}(I - QQ^H)) \tag{6.47}$$

stochastically, with D^{-1} to compute $D^{-1}((I - QQ^H)\psi)$ in line 9 of the Algorithm.

6.3 A-Hutch++

In the Hutch++ approach, the number of MVPs is predetermined and distributed between the two phases. The Adaptive Hutch++ (A-Hutch++) method is another version of Hutch++ which proposes two improved ranks [76]. It splits the MVPs in an approximately optimal way among the two phases, to estimate the trace within a certain error tolerance and with a controllable failure probability, with an optimal splitting of matrix-vector products. If the matrix is a symmetric positive semi-definite, they present a special version of Hutch++ called

Nystrom++. In comparison to Hutch++, this algorithm requires only one pass over the matrix, since it uses the Nystrom approximation.

A-Hutch++ approach aims to find an adaptive way of buildings randomized singular value decomposition (SVD) Q_k column-by-column and the user does not have to figure out how many matrix-vector products are necessary to output an estimate of the trace that is within the prescribed error tolerance. It starts with $k = 1, 2, \ldots$ and keeps track stop it when the minimum of $\tilde{m}(k)$ is detected.

A-Hutch++ decomposes the matrix into two parts as $D = D_{lr} - D_{\text{rest}}$ after applying a proper projector $\Pi = Q_k Q_k^H$, where D_{lr} denotes the low-rank part and D_{rest} denotes the rest of the matrix and k is the number of deflated vectors, thus the decomposition of the matrix inverse D^{-1} is given as:

$$D^{-1} := \underbrace{Q_k Q_k^H D^{-1}}_{D_{lr}^{-1}} + \underbrace{(I - Q_k Q_k^H) D^{-1}}_{D_{\text{rest}}^{-1}}, \tag{6.48}$$

and the trace estimation is given as in (6.49) where one can use the cyclic property of the trace and noting that $(I - Q_k Q_k^H)^2 = (I - Q_k Q_k^H)$

$$\operatorname{tr}(D^{-1}) := \operatorname{tr}(Q_k Q_k^H D^{-1}) + \operatorname{tr}((I - Q_k Q_k^H) D^{-1} (I - Q_k Q_k^H)) \tag{6.49}$$

The first part of the eq. (6.49) requires $2k$ matrix-vector products coming from the construction of Q_k which requires k products and we have another k products when we compute $\operatorname{tr}(Q_k Q_k^H D^{-1})$ as $\sum_{i=1}^{k} q_i^h D^{-1} q_i$. While the second term denotes the stochastic part, which can be estimated stochastically as in alg. 7, by requiring $M(k)$ number of matrix-vector products with D^{-1}. Then the total number of matrix-vector products with D^{-1} is

$$m(k) = 2k + M(k). \tag{6.50}$$

Furthermore, A-Hutch++ aims at minimizing $m(k)$ in order to obtain a near-optimal distribution of matrix-vector products between the two phases. That can be achieved by applying Lemma 6.7 on Theorem 6.6 as follows [76]

Theorem 6.6.
Assume a symmetric matrix $A \in \mathbb{C}^{n \times n}$, the tail bound for stochastic trace estimator $\operatorname{tr}_m(A)$ is

$$\mathbb{P}\left(|\operatorname{tr}_m(A) - \operatorname{tr}(A)| \geq \varepsilon\right) \leq 2 \exp\left(-m \frac{\varepsilon^2}{4\|A\|_F^2 + 4\varepsilon \|A\|_2}\right). \tag{6.51}$$

Lemma 6.7.
Let $m \geq \frac{4(1+c)\log(2/\delta)}{c^2\rho(A)}$ *for given* $c > 0$ *. Then the inequality*

$$|\operatorname{tr}_m(A) - \operatorname{tr}(A)| \leq 2\sqrt{1+c}\sqrt{\frac{\log(2/\delta)}{m}}\|A\|_F \tag{6.52}$$

holds with probability at least $1 - \delta$*.*

Proof. By combining the right-hand side of (6.52), $\varepsilon := 2\sqrt{1+c}\sqrt{\frac{\log(2/\delta)}{m}}\|A\|_F$, with (6.51), one can obtain the following results.

$$\begin{aligned}
\mathbb{P}\left(|\operatorname{tr}_m(A) - \operatorname{tr}(A)| \geq \varepsilon\right) &\leq 2\exp\left(-\frac{(1+c)\log(2/\delta)\|A\|_F}{\|A\|_F + 2\sqrt{1+c}\sqrt{\frac{\log(2/\delta)}{m}}\|A\|_2}\right) \\
&\leq 2\exp\left(-\frac{(1+c)\log(2/\delta)\|A\|_F}{(1+c)\|A\|_F}\right) = \delta,
\end{aligned}$$

where the second inequality utilizes

$$c\|A\|_F \geq 2\sqrt{1+c}\sqrt{\frac{\log(2/\delta)}{m}}\|A\|_2,$$

□

We can apply this theorem to bound the probability of the error in the estimate. To do so, we can use the value of $m(k)$ given in equation (6.50) as the number of matrix-vector products in the estimator, and we can use the spectral norm of D^{-1} in place of $\|A\|_2$ in the bound. Based on the symmetry of D^{-1} the Frobenius norm of D_{rest}^{-1} is given as

$$\begin{aligned}
\|D_{\text{rest}}^{-1}\|_F^2 &= \|(I - Q_kQ_k^H)D^{-1}(I - Q_kQ_k^H)\|_F^2 \\
&= \|D^{-1}\|_F^2 + \|Q_k^HD^{-1}Q_k\|_F^2 - 2\|D^{-1}Q_k\|_F^2
\end{aligned} \tag{6.53}$$

and the number of matrix-vector products $M(k)$ of D_{rest}^{-1} is

$$M(k) \approx \underbrace{\frac{1}{\epsilon^2}\log\frac{2}{\delta}}_{C(\epsilon,\delta)}\|(I - Q_kQ_k^H)D^{-1}(I - Q_kQ_k^H)\|_F^2 \tag{6.54}$$

Let

$$C(\varepsilon, \delta) := 4(1 + c)\varepsilon^{-2} \log(2/\delta). \tag{6.55}$$

By Lemma 6.7, number of samples equal to $C(\varepsilon, \delta)\|A\|_F^2$ for suitable small ε, is adequate to attain $|\operatorname{tr}_m(A) - \operatorname{tr}(A)| \leq \varepsilon$ with probability at least $1 - \delta$. In turn, $m(k)$ and the function

$$\tilde{m}(k) := 2k + C(\varepsilon, \delta)\big(\|Q_k^H D^{-1} Q_k\|_F^2 - 2\|D^{-1} Q_k\|_F^2\big) \tag{6.56}$$

have the same minimum, since

$$\begin{aligned} m(k) &:= 2k + C(\epsilon, \delta)\|(I - Q_k Q_k^H) D^{-1} (I - Q_k Q_k^H)\|_F^2 \\ &= \underbrace{2k - C(\epsilon, \delta)(\|Q_k D^{-1} Q_k^H\|_F^2 - 2\|Q_k^H D^{-1}\|_F^2}_{\widehat{m}(k)} + C(\epsilon, \delta)\|D^{-1}\|_F^2 \end{aligned}$$

Recursive updating can be used to compute the latter, without the need for additional matrix-vector products with D^{-1}.

Alg. 7 aims to approximate the trace of the inverse of a given matrix $D \in \mathbb{C}^{n \times n}$ using a combination of low rank and stochastic approximations. The low-rank approximation is obtained by iteratively generating random matrices $Y_k \in \mathbb{C}^{n \times k}$ with i.i.d. entries from the normal distribution, computing an orthonormal basis for the range of Y_k, and updating a running total for the trace of the inverse of the projection of D onto the range of Y_k. The stochastic approximation is obtained by generating random vectors ψ_s with i.i.d. entries from the normal distribution and computing the trace of the inverse of the projection of D onto the orthogonal complement of the range of Y_k. The algorithm terminates when the variance of the stochastic approximation falls below a predetermined threshold.

Algorithm 7 A-Hutch++: for estimating $\mathrm{tr}(D^{-1})$

Input: $D \in \mathbb{C}^{n\times n}$, nonsingular, ϵ relative accuracy, δ failure probability, C
Output: Approximation $\tau^{\mathrm{lr}} + \tau$ for $\mathrm{tr}(D^{-1})$

$k \leftarrow 1$
$Y_k \leftarrow D^{-1}\psi$ where $\psi \in \mathbb{R}^{n\times k}$ has i.i.d. $N(0,1)$ entries.
Obtain orthonormal basis q for range (Y_k).
$Q_k \leftarrow q_k$
$\tau^{\mathrm{lr}} \leftarrow \mathrm{tr}\left(q_k^H\,(D^{-1}q_k)\right)$
Compute $\tilde{m}(k)$.
for $k = 2, 3, \ldots$ **do**
    $Y_k \leftarrow D^{-1}\psi^k$ where $\psi^k \in \mathbb{R}^{n\times k}$ has i.i.d. $N(0,1)$ entries.
    $\widetilde{Q}_k \leftarrow (I - Q_{k-1}Q_{k-1}^H)Y_k$
    Obtain orthonormal basis q_k for range $\left(\widetilde{Q}_k\right)$.
    $Q_k \leftarrow \begin{bmatrix} Q_{k-1} & q_k \end{bmatrix}$
    $\tau^{\mathrm{lr}} \leftarrow \tau^{\mathrm{lr}} + \mathrm{tr}\left(q_k^H\,(D^{-1}q_k)\right)$
    $\tilde{m}(k) :\leftarrow C(\varepsilon,\delta)\left(\|Q_k^H D^{-1} Q_k\|_F^2 - 2\|D^{-1}Q_k\|_F^2\right)$ ▷ Update $\tilde{m}(k)$ recursively.
    if $\tilde{m}(k) \geq \tilde{m}(k-1)$ **then**
        stop
    end if
end for
for $s = 1, 2, \ldots$ **do** ▷ *stochastic part*
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $z_s \leftarrow \psi_s - Q(Q^H\psi_s)$ ▷ projected vector
    $\tau_s \leftarrow \psi_s^H D^{-1} z_s$ ▷ solve linear system
    $\tau \leftarrow \frac{1}{s}\sum_{i=1}^{s}\tau_i$, $\mathbb{V} \leftarrow \frac{1}{s-1}\sum_{i=1}^{s}|\tau_i - \tau|^2$ ▷ sample mean and variance
    $\rho = (\tau + \tau^{\mathrm{lr}})\epsilon$
    if $\mathbb{V}/s \leq \rho^2$ **then**
        stop
    end if
end for

Chapter 7

Multigrid Multilevel Monte Carlo

The main concept underlying our novel method, Multigrid Multilevel Monte Carlo (MG-MLMC), is covered in detail in this Chapter. The name MG-MLMC is coming from merging the Hierachical Multigrid method with the Multilevel Monte Carlo technique. So we begin by reviewing the fundamental facts from Monte Carlo (MC), two-level MC, and multilevel Monte Carlo (MLMC) methods in Sec. 7.1, Sec. 7.2.1, and Sec. 7.2.2, respectively. Then we explore the main idea of the MG-MLMC method, its derivation and how we can use it as a variance reduction method in sections 7.3, 7.3.2.1 and 7.3.2. Furthermore, we examine how to apply Hutch++ and exact deflation approaches on MG-MLMC in Sec. 7.3.3 and Sec. 7.3.4 respectively.

7.1 Standard Monte Carlo method

Monte Carlo method is a computational algorithm that obtains numerical results through repeated random sampling to solve problems that might seem deterministic in essence [65]. Typically, the samples are chosen randomly and independently. The realization variable is estimated based on the basis of those samples.

Monte Carlo (MC) is frequently the preferred method when the total number of free variables is high, so numerous mathematical and physical issues can be solved using the Monte Carlo method, particularly those falling into one of three broad categories: numerical integration, optimization, or generating draws based on probability distributions [65]. It works as follows:

Theorem 7.1.
Suppose $g : R^d \to R$ is a function of a d-dimensional random variable X with probability density ρ, then the expected value of the function $g(x)$ is given as

$$Y = \mathbb{E}[g(x)] = \int_{R^d} g(x)\rho(x)dx \tag{7.1}$$

the plain MC estimator of (7.1) *is given as [50]:*

$$\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} g(X_i) \tag{7.2}$$

where $X_i, i = 1, \ldots, N$ *are independent samples of the random variable* X *and* $\hat{\theta}_N \to \mathbb{E}[g(x)]$ *as* $N \to \infty$ *[82].*

The expected value of the unbiased MC estimator $\hat{\theta}_N$ of Y is:

$$\mathbb{E}(\hat{\theta}_N) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}(g(X_i)) = \mathbb{E}[g(X)] \tag{7.3}$$

and the variance of the estimator $\hat{\theta}_N$ is

$$\mathbb{V}(\hat{\theta}_N) = \mathbb{E}\Big[(\hat{\theta}_N - \theta)^2\Big] = \frac{\sigma^2}{N}. \tag{7.4}$$

where $\sigma^2 = \mathbb{V}(g(X))$, and the root mean square error (RMSE) of the estimator $\hat{\theta}_N$:

$$\mathrm{RMSE}(\hat{\theta}_n) = \frac{\sigma}{\sqrt{N}} \tag{7.5}$$

Eq. (7.5) shows that the convergence order of RMSE of the simple MC estimator is $O(\frac{1}{\sqrt{N}})$ [50]. Furthermore, using the central limit theorem on a finite sample for large enough N the estimator will be approximately normally distributed if variance is finite as:

$$\hat{\theta}_N \approx N(\theta, \frac{\sigma^2}{N}) \tag{7.6}$$

The Monte Carlo techniques build samples using simulations of random processes. However, it is not always possible to simulate random processes precisely.

As a result, one can observe that the MC method has a number of benefits, including being straightforward to comprehend, simple to implement, non-intrusive, trivially parallelizable, and performance independent of the number or dimensionality of the stochastic parameters.

7.1.1 Computational Cost

The sum of all calls to the generator for random numbers is known as computational cost (complexity) [59]. This indicates the cost for the plain MC approach is determined by multiplying the number of instances of random variables by the cost of computing a single random variable. Assume that $\hat{\theta}$ is the approximation of θ thus the mean square error can be expressed as:

$$\text{MSE} = \mathbb{E}\Big[(\hat{\theta} - \theta)^2\Big]. \tag{7.7}$$

$$\begin{aligned} \mathbb{E}\Big[(\hat{\theta} - \theta)^2\Big] &= \mathbb{E}\Big[(\hat{\theta} + \mathbb{E}[\hat{\theta}] - \mathbb{E}[\hat{\theta}] - \theta)^2\Big]. \\ &= \mathbb{E}\Big[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2\Big] + \mathbb{E}\Big[(\mathbb{E}[\hat{\theta}] - \theta)^2\Big] \end{aligned} \tag{7.8}$$

the first term $\mathbb{E}\Big[(\hat{\theta} - \mathbb{E}[\hat{\theta}])^2\Big]$ represents the variance and the second term $\mathbb{E}\Big[(\mathbb{E}[\hat{\theta}] - \theta)^2\Big]$ denotes the bias of the approximation. For a RMSE of order ε, MSE should be $O(\varepsilon^2)$ which is achieved if

$$\text{MSE} = \mathbb{E}\Big[(\hat{\theta} - \theta)^2\Big] = O(\varepsilon^2) \tag{7.9}$$

7.2 Multilevel Monte Carlo

Obviously, standard Monte Carlo is the easiest and most widely used technique for calculating the expectations of random variables. The only weakness of MC is the relatively slow convergence, which can lead to extremely costly computations. Multilevel Monte Carlo, or MLMC for short, is an adaptive method to improve this slow convergence and significantly lower simulation costs. In the following sections, the basic idea of MLMC will be discussed in detail.

7.2.1 Two-level MC

The main idea of the Multilevel Monte Carlo technique was first used to estimate $\mathbb{E}[g(X, \mu)]$, where X is a random vector and μ is a parameter, in the context of parametric integration [56, 57]. Giles generalized the idea to be known as MLMC in his seminal paper [49]. Multilevel Monte Carlo exploits a hierarchy of solving approximations $X_1, X_2, ..., X_L$ at different levels starting with the coarsest

and cheapest approximation X_L and going up to the finest and most expensive approximation X_1.

The simplified version of the MLMC method, the **two levels Monte Carlo**, is where we start our discussion. For a given random variable X_0 assume we take another random variable X_1 and consider: X_1, X_2 and let $X_1 \leq X_2$ then X_2 is given by:

$$X_1 = (X_1 - X_2) + X_2 \tag{7.10}$$

and its expected value as:

$$\mathbb{E}[X_1] = \mathbb{E}[X_1 - X_2] + \mathbb{E}[X_2], \tag{7.11}$$

and we consider the estimator is given by:

$$\hat{\theta}_1 = \frac{1}{N_1}\sum_{i=1}^{N_1}(X_1^{(i)} - X_2^{(i)}) + \frac{1}{N_2}\sum_{i=1}^{N_2} X_2^{(i)} \tag{7.12}$$

$X_1^{(i)} - X_2^{(i)}$ represents a sample of difference $X_1 - X_2$ for the same underlying stochastic sample $w^{(i)}$. The total cost for generating (7.12) is

$$C_T = N_1 C_1 + N_2 C_2 \tag{7.13}$$

where C_2 is the cost for estimating a single sample of X_2 and C_1 represents the cost for estimating a single sample of $X_1 - X_2$. For the variance of the estimator we have

$$\begin{aligned} \mathbb{V}(\frac{1}{N_2}\sum_{i=1}^{N_2} X_2^{(i)}) &= \frac{N_2}{N_1^2}\mathbb{V}_1 = \frac{1}{N_2}\mathbb{V}_1, \\ \mathbb{V}(\frac{1}{N_1}\sum_{i=1}^{N_1}(X_1^{(i)} - X_2^{(i)})) &= \frac{1}{N_1}\mathbb{V}_1 \end{aligned} \tag{7.14}$$

the total variance is

$$\mathbb{V}_T = N_1^{-1}\mathbb{V}_1 + N_2^{-1}\mathbb{V}_2 \tag{7.15}$$

We want to minimize $C(N_1, N_2)$ under the constraint $\mathbb{V}_T = \epsilon^2$, so we consider the Lagrangian function

$$\mathcal{L}(N_1, N_2, \lambda) = C(N_1, N_2) + \lambda(V_T - C^2) \tag{7.16}$$

with the multiplier $\lambda \in \mathbb{R}$. The stationary points of λ satisfy

$$\nabla C[N_1, N_2] + \lambda \nabla \mathbb{V}_T = 0 \quad , \quad (C_1 - \frac{\lambda}{N_1^2}\mathbb{V}_1, C_2 - \frac{\lambda}{N_2^2}\mathbb{V}_2) = 0 \tag{7.17}$$

and

$$\mathbb{V}_T - \epsilon^2 = 0 \quad \frac{1}{N_1}\mathbb{V}_1 + \frac{1}{N_2}\mathbb{V}_2 - \epsilon^2 = 0 \tag{7.18}$$

From (7.17) we get $N_1 = \sqrt{\lambda \frac{\mathbb{V}_1}{C_1}}, N_2 = \sqrt{\lambda \frac{\mathbb{V}_2}{C_2}}$ so that (7.18) given

$$\frac{1}{\sqrt{\lambda}}(\sqrt{\mathbb{V}_1 C_1} + \sqrt{\mathbb{V}_2 C_2}) = \epsilon^2,$$

and then

$$\begin{aligned}
\lambda &= \frac{1}{\epsilon^4}(\sqrt{\mathbb{V}_1 C_1} + \sqrt{\mathbb{V}_2 C_2})^2 \\
N_1 &= \frac{1}{\epsilon^2}\frac{\sqrt{\frac{\mathbb{V}_1}{C_1}}}{(\sqrt{\mathbb{V}_1 C_1} + \sqrt{\mathbb{V}_2 C_2})^2} \\
N_2 &= \frac{1}{\epsilon^2}\frac{\sqrt{\frac{\mathbb{V}_2}{C_2}}}{(\sqrt{\mathbb{V}_1 C_1} + \sqrt{\mathbb{V}_2 C_2})^2},
\end{aligned}$$

where

$$\mathbb{V}(N_1, N_2) = N_1^{-1}\mathbb{V}_1 + N_2^{-1}\mathbb{V}_2 \text{ and } C(N_1, N_2) = N_1 C_1 + N_2 C_2. \tag{7.19}$$

7.2.2 Multilevel Monte Carlo (MLMC) theory

The main goal of MLMC is to estimate the expected values of the quantities that come out from the simulations to reduce the variance of the random variable. Its strategy requires repeating random sampling and taking random samples on different levels of accuracy. only the last few samples are taken with high accuracy with considered high computational costs. Extending the two-level approach of 7.2.1, we continue repeating the process of adding levels until the maximum number of levels L, and decomposing the quantity X_ℓ where $\ell = 1, 2, \cdots, L$, as

$$X_\ell = (X_1 - X_2) + (X_2 - X_3) + \cdots + (X_{L-1} - X_L) + X_L \tag{7.20}$$

where $X_1 = X_\ell$ for all values of ℓ, and it can be written as

$$X_\ell = \sum_{\ell=1}^{L-1}(X_\ell - X_{\ell+1}) + X_L \tag{7.21}$$

since the sequence of random variables $X_1, \dots X_L$ approximates X_ℓ with increasing accuracy but also increasing cost. level L is the coarsest and level 1 is the finest.

The expection value of eq. (7.21) is

$$\mathbb{E}[X_\ell] = \mathbb{E}\Big[\sum_{\ell=1}^{L-1}(X_\ell - X_{\ell+1})\Big] + \mathbb{E}[X_L]. \tag{7.22}$$

Terms of eq. (7.22) can be estimated independently and the multilevel Monte Carlo estimator $\widehat{\theta} = \sum_{\ell=0}^{L} \theta_\ell$ is given as

$$\widehat{\theta} = \sum_{\ell=1}^{L-1}\left(N_\ell^{-1}\sum_{i=1}^{N_\ell}\left(X_\ell^{(i)} - X_{\ell+1}^{(i)}\right)\right) + N_L^{-1}\sum_{i=1}^{N_L} X_L^{(i)}. \tag{7.23}$$

It is the two level case, $X_\ell^i - X_{\ell+1}^i$ stands for one instance of random variable $X_\ell - X_{\ell+1}$ The variance of $\widehat{\theta}$ gives the variance for the random variable $X_\ell - X_{\ell+1}$ [49]

$$\mathbb{V}(\widehat{\theta}) = \sum_{\ell=1}^{L-1} N_\ell^{-1}\mathbb{V}_\ell \tag{7.24}$$

Define the cost of one sample of X_1 as C_1 and Define the cost of one sample of $X_\ell - X_{\ell+1}$ as C_ℓ, then by summing over all levels we obtain the total cost as:

$$C_{\text{tot}} = N_1C_1 + N_2C_2 + \cdots + N_LC_L = \sum_{\ell=1}^{L} N_\ell C_\ell. \tag{7.25}$$

The optimal number of samples at each level difference N_ℓ can be obtained by applying Lagrangian multiplier μ^2 and taking the gradient to find the minimum variance for a fixed cost as

$$\mathcal{L}(N_1, \dots, N_L, \mu^2) = \sum_{\ell=1}^{L}(N_\ell^{-1}\mathbb{V}_\ell + \mu^2 N_\ell C_\ell), \quad \ell = 0, 1, \cdots, L \tag{7.26}$$

$$\frac{\partial}{\partial N_i}\sum_{i=1}^{N_i}(N_i^{-1}\mathbb{V}_i + \mu^2 N_i C_i) = -N_\ell^{-2}\mathbb{V}_\ell + \mu^2 C_\ell) = 0, \tag{7.27}$$

we can reformulate the problem to minimize the quantity

$$\sum_{\ell=1}^{L} N_\ell C_\ell + \mu^2(\frac{V_\ell}{N_\ell} - \frac{\varepsilon^2}{2})$$

After solving eq. (7.27) the number of samples at each level difference is given as

$$N_\ell = \frac{1}{\mu}\sqrt{\frac{\mathbb{V}_\ell}{C_\ell}}, \tag{7.28}$$

and minimal variance of is given as

$$\mathbb{V}(\widehat{\theta}) = \sum_{\ell=1}^{L}\frac{\mathbb{V}_\ell}{N_\ell} = \sum_{\ell=1}^{L}\frac{1}{\mu}\sqrt{\frac{C_\ell}{\mathbb{V}_\ell}}\mathbb{V}_\ell = \frac{1}{\mu}\sum_{\ell=1}^{L}\sqrt{C_\ell\mathbb{V}_\ell} := \varepsilon^{-2} \tag{7.29}$$

From eq. (7.29) we get

$$\mu = \varepsilon^2(\sum_{\ell=1}^{L}\sqrt{C_\ell\mathbb{V}_\ell})^{-1}, \tag{7.30}$$

and the number of samples N_ℓ can be written as

$$N_\ell = \varepsilon^{-2}\sqrt{\frac{\mathbb{V}_\ell}{C_\ell}}\left(\sum_{\ell=1}^{L}\sqrt{C_\ell\mathbb{V}_\ell}\right) \tag{7.31}$$

The total cost of multilevel Monte Carlo is then expressed as

$$C_{\text{mlmc}} = \sum_{\ell=1}^{L} N_\ell C_\ell = \varepsilon^{-2}\left(\sum_{\ell=1}^{L}\sqrt{C_\ell\mathbb{V}_\ell}\right)^2. \tag{7.32}$$

Eq. (7.32) shows that the dominant cost is determined based on varying the product term $\mathbb{V}_\ell C_\ell$ with the level difference ℓ, since we have three different cases for the total cost [49] as:

1. If $\mathbb{V}_\ell C_\ell$ increases with ℓ: then the finest level is responsible for the majority of the cost, and $C_{mlmc} \approx \varepsilon^{-2}V_L C_L$

2. If $\mathbb{V}_\ell C_\ell$ decrease increases with ℓ: then the coarsest level is responsible for the majority of the cost, and $C_{mlmc} \approx \varepsilon^{-2} V_L C_L$
3. $\mathbb{V}_\ell C_\ell$ does not vary: the overall cost $C_{mlmc} \approx \varepsilon^{-2} L^2 V_L C_L \approx \varepsilon^{-2} L^2 V_L C_L$.

7.2.3 Multilevel Monte Carlo for trace estimation

We now turn to estimating the trace of an (implicitly) given matrix $D = D_1$. We know from section 5.2, that the random variable

$$X = \eta^H D \eta, \quad \eta \text{ a Radamacher vector}, \tag{7.33}$$

is an unbaised estimator for $\operatorname{tr} D$. The idea to obtain a MLMC for trace estimation is now to consider a decomposition

$$D_1 = \sum_{\ell=1}^{L-1} (D_\ell - D_{\ell+1}) + D_L, \tag{7.34}$$

so that

$$\operatorname{tr}(D_1) = \sum_{\ell=1}^{L-1} \operatorname{tr}(D_\ell - D_{\ell+1}) + \operatorname{tr}(D_L), \tag{7.35}$$

and introduce the stochastic variable

$$X_\ell - X_{\ell+1} = \eta_\ell^H (D_\ell - D_{\ell+1}) \eta_\ell. \tag{7.36}$$

as well as

$$X_L = \eta_L^H D_L \eta_L \tag{7.37}$$

Then

$$X = \sum_{\ell=1}^{L-1} (X_\ell - X_{\ell+1}) + X_L \tag{7.38}$$

and

$$\operatorname{tr}(X) = \mathbb{E}[X] = \sum_{\ell=1}^{L-1} \mathbb{E}[X_\ell - X_{\ell+1}] + \mathbb{E}[X_L] \tag{7.39}$$

These should be chosen such that $\mathbb{V}(X_\ell - X_{\ell+1})$ is small when $X_\ell - X_{\ell+1}$ is costly to evaluate (typically for the small values of ℓ) and possibly large when they are cheap to evaluate (for the large values of ℓ). Then for a given accuracy, we need only a few samples that are costly to evaluate and possibly many when they are

cheap to evaluate. And it might happen that the expected value of X_L can be computed without any stochastic technique.

The variance ρ^2 for the resulting estimator for $\mathbb{E}[X]$ is the sum of the variances of the estimators for $\mathbb{E}[X_\ell]$. In the *uniform* approach one chooses the number N_ℓ of samples at each level such that

$$\mathbb{V}[X_\ell]/N_\ell = \rho^2/L. \tag{7.40}$$

If one knows the cost C_ℓ for an evaluation of X_ℓ, the problem of minimizing the total cost under the constraint to obtain a variance of ρ^2 is solved for the *optimal* values [49]

$$N_\ell = \frac{1}{\rho^2}\sqrt{\mathbb{V}[X_\ell]/C_\ell}\sum_{j=1}^{L-1}\sqrt{\mathbb{V}[X_j]C_j}. \tag{7.41}$$

The variance of the estimator for X_ℓ with N_ℓ samples is then

$$\mathbb{V}[X_\ell]/N_\ell = \rho^2\sqrt{\mathbb{V}[X_\ell]C_\ell}\ \Bigg/\ \sum_{j=1}^{L-1}\sqrt{\mathbb{V}[X_j]C_j}\ . \tag{7.42}$$

The main goal of the multilevel Monte Carlo method is to reduce the variance by decomposing the main matrix D at different levels. In contrast to the lower levels, which are more expensive to estimate, when evaluating $f(D)$ is more expensive, the variance is small on the higher numbered levels.

7.2.4 Error bounds of multilevel Monte Carlo methods

In this section, we discuss the error bound (guarantees) for multilevel Monte Carlo. Let we have a matrix $D \in C^{n\times n}$ and Rademacher vectors $\psi_\ell^{(i)}$ with iid., then the estimator of the multilevel can be expressed as

$$\mathrm{tr}_s(D) = \sum_{\ell=1}^{L}\sum_{i=1}^{s_\ell}\frac{1}{s_\ell}\psi_\ell^{(i)H}D_\ell\psi_\ell^{(i)}, \tag{7.43}$$

Theorem 7.2 shows how to obtain bounds of the estimator tr_s on the accuracy for single-level samples [22].

Theorem 7.2.
Assume a nonzero symmetric matrix $D \in \mathbb{C}^{n\times n}$ with all-zero diagonal entries and

a Rademacher vector $\psi \in R^n$ *with i.i.d. . Then for all* $\varepsilon > 0$,

$$\Pr\left(|\psi^H D\psi| \geq \varepsilon\right) \leq 2\exp\left(-\frac{\varepsilon^2}{8\|D\|_F^2 + 8\varepsilon\|D\|_2}\right). \tag{7.44}$$

The same proof technique can be used to extend this bound error to a multilevel approach [55] as in theorem 7.3.

Theorem 7.3.
Assume a nonzero symmetric matrix $D \in \mathbb{R}^{n\times n}$. *For multilevel* $\ell = 1, \ldots, L$ *we have* D_ℓ *symmetric matrices where* $D = \sum_{\ell=1}^{L} D_\ell$, *and let* B_ℓ *equal* D_ℓ *but with the diagonal entries set to zero. For sample sizes* $m = m_\ell, \ell = 1, \cdots, L$, *let* $\mathrm{tr}_m(D)$ *be defined as in* (7.43). *Then for all* $\varepsilon > 0$,

$$\Pr\left(|\,\mathrm{tr}_m(D) - \mathrm{tr}(D)| \geq \varepsilon\right) \leq 2\exp\left(\frac{-\varepsilon^2/8}{\sum_{\ell=1}^{L}\|B_\ell\|_F^2/m_\ell + \varepsilon\max\limits_{1\leq\ell\leq L}\|B_\ell\|_2/m_\ell}\right). \tag{7.45}$$

Furthermore, let C_ℓ *denotes the cost of sampling from* B_ℓ *and the variance* $\mathbb{V}_\ell = \|B_\ell\|_F^2 + \varepsilon\|B_\ell\|_2$. *Then if* $m_\ell \geq \mu\sqrt{\frac{\mathbb{V}_\ell}{C_\ell}}$ *where*

$$\mu = 8\varepsilon^{-2}\log(2/\delta)\sum_{\ell=1}^{L}\sqrt{C_\ell\mathbb{V}_\ell}, \tag{7.46}$$

it follows that $\Pr\left(|\,\mathrm{tr}_m(D) - \mathrm{tr}(D)| \geq \varepsilon\right) \leq \delta$.

The total cost of the MLMC estimator for a given pair (ε, δ) can be approximated as

$$C_{mlmc} = 8\varepsilon^{-2}\log(2/\delta)\left(\sum_{\ell=1}^{L}\sqrt{C_\ell\mathbb{V}_\ell}\right)^2. \tag{7.47}$$

Multilevel Monte Carlo estimators can have smaller variances and better (ε, δ)-type error bounds as well.

7.3 Multigrid Multilevel Monte Carlo Method

The basic goal is to combine both standard multigrid (MG) and MLMC techniques together to reduce the variance of trace estimation of the matrix inverse and accelerate the convergence rate. The idea behind **MG-MLMC** is approximating

the trace of the inverse at one level and the next level, restrict the random vector to solve the small system, then prolongate it up again and estimating the trace of this difference.

Using a solver for the finer level i.e Multigrid method, we need to calculate the number of matrix-vector multiplications (MVPs) to find the cost of the required work for each stochastic estimate and can compare it to the plain deflated Hutchinson approach. In the Multigrid Multilevel Monte Carlo approach, we have three different phases:

1. Setup Phase: Based on the multigrid hierarchy, we define the prolongation P_ℓ, restriction R_ℓ and matrices D_ℓ at each level. We use the multigrid method as a solver for each level difference.
2. Difference (fine-coarse) part which estimates the trace of the difference on two matrices corresponding to two consecutive levels
3. Coarsest part which is included in the final matrix D_L (coarsest matrix) that remains after doing the difference part, and its trace of D_L^{-1} might be computed directly if the size of the final operator D_L is small enough, otherwise we can compute it stochastically.

7.3.1 Two-Grid Two-Level Monte Carlo for trace estimation

We are interested in estimating the $\mathrm{tr}(D^{-1})$ where $D \in \mathbb{C}^{n\times n}$. To simplify things we begin our discussion with the simple case, two levels, and use the notation D_1 for the original (finest) system, call it level 1, and D_2 for the coarse system, call it level 2. Moreover, P_1 and R_1 are the prolongation and restriction between levels 1 and 2, so that

$$D_2 = R_1 D_1 P_1.$$

The original matrix decomposes as

$$D_1^{-1} = \underbrace{D_1^{-1} - P_1 D_2^{-1} R_1}_{\text{level 1}} + \underbrace{P_1 D_2^{-1} R_1}_{\text{level 2}}. \tag{7.48}$$

Then the trace of D_1^{-1} is

$$\mathrm{tr}(D_1^{-1}) = \underbrace{\mathrm{tr}(D_1^{-1} - P_1 D_2^{-1} R_1)}_{\text{level 1}} + \underbrace{\mathrm{tr}(P_1 D_2^{-1} R_1)}_{\text{level 2}}. \tag{7.49}$$

In (7.49), we always have to start with a random vector on level 1 when stochastically estimating the various summands. We must also apply R_1 and P_1 if we estimate for the level difference 1.

For simplicity, Fig. 7.1 demonstrates an instance of a level difference between two multigrid hierarchy levels $\ell = 1, 2$; the difference level operator can be displayed as in (7.50)

$$M_1 = D_1^{-1} - P_1 D_2^{-1} R_1 \tag{7.50}$$

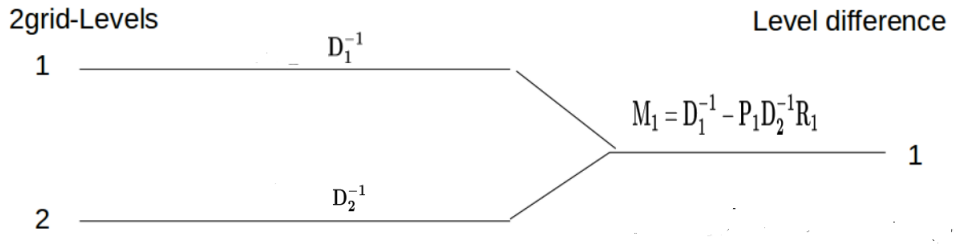


Figure 7.1: shows 2-grid 2-level Monte Carlo example for splitting the original problem into two parts

7.3.2 Multigrid Multilevel Monte Carlo for trace estimation

We can generalize the case in 7.3.1 for levels $\ell = 1, \ldots, L$, and the original matrix D_1 for the number of levels L could split corresponding to the number of levels as $D_1, D_2, \ldots, D_L$ coarsest matrices at different levels. To obtain a multilevel Monte Carlo decomposition we discard the smoother and only consider the coarse grid operators and the intergrid transfer operators which we now describe algebraically.

7.3.2.1 Derivation of Multigrid Multilevel Monte Carlo

The coarse grid operators are given by a sequence of matrices with Petrov-Galerkin construction, i.e.

$$D_\ell \in \mathbb{C}^{n_\ell \times n_\ell}, \ell = 1, \ldots, L,$$

representing the original matrix $D = D_1 \in \mathbb{C}^{n_1 \times n_1}$. On the different levels $\ell = 1, \ldots, L$; the prolongation and restriction operators

$$P_\ell \in \mathbb{C}^{n_\ell \times n_{\ell+1}},\ R_\ell \in \mathbb{C}^{n_{\ell+1} \times n_\ell},\ \ell = 1, \ldots, L-1.$$

transfer data between the levels. Typically, when A is Hermitian, one takes $P_\ell = R_\ell^H$, and for given P_ℓ, R_ℓ the coarse system matrices D_ℓ are often constructed using the Petrov-Galerkin approach

$$D_{\ell+1} = R_\ell D_\ell P_\ell,\ \ell = 1, \ldots, L-1.$$

Using the accumulated prolongation and restriction operators

$$\hat{P}_\ell = P_1 \cdots P_{\ell-1} \in \mathbb{C}^{n \times n_\ell}, \hat{R}_\ell = R_{\ell-1} \cdots R_1 \in \mathbb{C}^{n_\ell \times n},\ \ell = 1, \ldots, L,$$

where we put $\hat{R}_1 = \hat{P}_1 = I \in \mathbb{C}^{n \times n}$ by convention, we regard $\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell$ as the approximation to D^{-1} at level ℓ. Thus the original matrix D decomposes as

$$D = (D_1 - P_1 D_2 P_1^H) + (P_1 D_2 P_1^H - P_1 P_2 D_3 P_2^H P_1^H) \ldots + P_1 \cdots P_{L-1} D_L P_{L-1}^H \cdots P_1^H \tag{7.51}$$

which can be written as:

$$D^{-1} = \sum_{\ell=1}^{L-1} \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \right) + \hat{P}_L D_L \hat{R}_L, \tag{7.52}$$

We thus obtain a multilevel decomposition for the trace of D^{-1} as

$$\mathrm{tr}(D^{-1}) = \sum_{\ell=1}^{L-1} \mathrm{tr} \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \right) + \mathrm{tr}(\hat{P}_L D_L^{-1} \hat{R}_L). \tag{7.53}$$

This gives

$$\mathrm{tr}(D^{-1}) = \sum_{\ell=1}^{L-1} \mathbb{E} \left[(\psi^\ell)^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \right) \psi^\ell \right] + \mathbb{E} \left[(\psi^L)^H \hat{P}_L D_L^{-1} \hat{R}_L \psi^L \right], \tag{7.54}$$

with the components of $\psi^\ell \in \mathbb{C}^n$ being i.i.d. stochastic variables satisfying (5.10). The unbiased multilevel Monte Carlo estimator is then

$$\begin{aligned} \mathrm{tr}(D^{-1}) \ \approx\ & \sum_{\ell=1}^{L-1} \frac{1}{M_\ell} \sum_{m=1}^{M_\ell} \left((\psi^{(m,\ell)})^H \hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi^{(m,\ell)} - (\psi^{(m,\ell)})^H \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \psi^{(m,\ell)} \right) \\ & + \frac{1}{M_L} \sum_{i=1}^{M_L} (\psi^{(m,L)})^H \hat{P}_L D_L^{-1} \hat{R}_L \psi^{(m,L)}, \end{aligned} \tag{7.55}$$

where the vectors $\psi^{(m,\ell)} \in \mathbb{C}^n$ are stochastically independent samples of the random variable $\psi \in \mathbb{C}^n$ satisfying (5.10).

We obtain $\Pi_1 D_1^{-1} = P_1 D_2^{-1} R_1$ based on the projector $\Pi_1 = P_1 D_2^{-1} R_1 D_1$ and similarly for the coarser levels, thus establishing the connection with inexact deflation discussed in the previous chapter in sec. 6.1.3. The prolongations P_ℓ, in the multigrid hierarchy, are precisely built in such a way that they contain accurate approximations to the small eigenmodes or singular triplets of D_ℓ. Since the prolongations $P_{\ell+1}$ are constructed to approximate small eigenpairs or singular triples of D_ℓ, for each level difference we anticipate that the variance of

$$\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1}$$

will be small. A stochastic estimate becomes increasingly less expensive with each level difference because the size of the matrices to invert on each level difference decreases significantly with the level. Depending on the size of the matrix D_L at the coarsest level L, we might be able to compute the trace directly as

$$\sum_{i=1}^{N_L} e_i^T D_L^{-1} \hat{R}_L \hat{P}_L e_i.$$

If we perform it stochastically, we have to invert a matrix with a very small dimension compared to that of D_1.

Computationally, the estimator requires to solve systems of the form $D_\ell y^{(m,\ell)} = z$ with $z = \hat{R}_\ell \psi^{(m,\ell)}$. Since the matrices D_ℓ arise from the multigrid hierarchy, we directly have a multigrid method available for these systems by restricting the method for D_1 to the levels $\ell, \ldots, L$.

7.3.2.2 Coarsest level in the Multigrid Multilevel Monte Carlo method

In the MG-MLMC approach always there are differences all the time so the standard deviation for these differences is small and easy to estimate, but the very last level there is no difference and the variance is expected to be large so we need to take a lot of stochastic samples so that it may be better to compute this directly. Solving the coarsest level directly provides us with a substantial gain, so if we could do it earlier then the gaining will be twice, first solving the coarsest level, and second, we have less accuracy required on the other levels too. At times, even the coarsest matrix in extremely large sparse matrices may still be large, making direct computation of the trace of its inverse expensive. Consequently, a preferable alternative is to compute the trace stochastically. Considering more levels means we have more relative accuracy (be more accurate) on each level because the variance required is the summation of the variances on each level.

Since for any two matrices $A = (a_{ij}) \in \mathbb{C}^{n\times m}$ and $B = (b_{kl}) \in \mathbb{C}^{m\times n}$ the trace of their product does not depend on the order,

$$\mathrm{tr}(AB) = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}b_{ji} = \sum_{j=1}^{m}\sum_{i=1}^{n} b_{ji}a_{ij} = \mathrm{tr}(BA), \tag{7.56}$$

we have

$$\mathrm{tr}(\hat{P}_L D_L^{-1}\hat{R}_L) = \mathrm{tr}(D_L^{-1}\hat{R}_L\hat{P}_L). \tag{7.57}$$

So, instead of estimating the contribution $\mathrm{tr}(\hat{P}_L D_L^{-1}\hat{R}_L)$ in (7.53) stochastically, we can also compute it directly by inverting the matrix $D_L \in \mathbb{C}^{n_L\times n_L}$ and computing the product $D_L^{-1}\hat{R}_L\hat{P}_L$. Note that the matrices $\hat{R}_L$ and $\hat{P}_L$ are usually sparse with a maximum of d, say, non-zero entries per row. The arithmetic work for $D_L^{-1}\hat{R}_L\hat{P}_L$ is thus of order $\mathcal{O}(dn_L^2)$ for the product $\hat{R}_L\hat{P}_L$ plus $\mathcal{O}(n_L^3)$ for the inversion of D_L and the product $D_L^{-1}(\hat{R}_L\hat{P}_L)$. Since the variance of $\psi^H\hat{P}_L D_L^{-1}\hat{R}_L\psi$ is presumably large, this direct computation can be much more efficient than a stochastic estimation, even when we aim at only quite low precision in the stochastic estimate.

7.3.2.3 Aggregation-based case

In the successful multigrid approaches for the Wilson-Dirac matrix or its twisted mass variant, see [4, 7, 13, 40], the restrictions and prolongations are aggregation based with $R_\ell = P_\ell^H$ and R_ℓ orthonormal i.e we have $R_\ell P_\ell = I, \ell = 1, \cdots, L$. Then

$$\mathrm{tr}(\hat{P}_\ell D_\ell^{-1}\hat{R}_\ell) \;=\; \mathrm{tr}(D_\ell^{-1}\hat{R}_\ell\hat{P}_\ell) \;=\; \mathrm{tr}(D_\ell^{-1}), \tag{7.58}$$

and

$$\begin{aligned}\mathrm{tr}(\hat{P}_{\ell+1}D_{\ell+1}^{-1}\hat{R}_{\ell+1}) &= \mathrm{tr}(\hat{P}_\ell P_\ell D_{\ell+1}^{-1}R_\ell\hat{R}_\ell)\\ &= \mathrm{tr}(P_\ell D_{\ell+1}^{-1}R_\ell\hat{R}_\ell\hat{P}_\ell)\\ &= \mathrm{tr}(P_\ell D_{\ell+1}^{-1}R_\ell).\end{aligned} \tag{7.59}$$

This means that instead of the multilevel decomposition (7.53) we can use

$$\mathrm{tr}(D^{-1}) = \sum_{\ell=1}^{L-1}\mathrm{tr}\left(D_\ell^{-1} - P_\ell D_{\ell+1}^{-1}R_\ell\right) + \mathrm{tr}(D_L^{-1}), \tag{7.60}$$

in which the stochastic estimation on level ℓ now involves random vectors from $\mathbb{C}^{n_\ell}$ instead of $\mathbb{C}^n$.

In the general case, we can use the formula as it stands and do Hutch for the matrix

$$D_l^{-1}\hat{R}_l\hat{P}_l - P_l D_{l+1}^{-1}\hat{R}_{l+1}\hat{P}_l \quad \text{from eq. (7.53)}$$

which again requires only random vectors from $\mathbb{C}^{n_\ell}$ Finally, one can consider the MG-MLMC approach as a recursive inexact deflation approach by applying an oblique projector on the matrix D_ℓ^{-1}

Theorem 7.4.
Consider an nonsingular matrix $D \in \mathbb{C}^{n\times n}$, apply projector $\Pi_\ell = P_\ell D_{\ell+1}^{-1} R_\ell D_\ell$ on the matrix inverse D^{-1}. Then, the matrix inverse at each level of the hierarchy $D_\ell^{-1}, \ell = 1, 2, \cdots, L$, can be written as follows:

$$\begin{aligned} D_\ell^{-1} &= (I - \Pi_\ell)D_\ell^{-1} + \Pi_\ell D_\ell^{-1} \\ &= (I_\ell - P_\ell D_{\ell+1}^{-1} R_\ell D_\ell)D_\ell^{-1} + P_\ell D_{\ell+1}^{-1} R_\ell D_\ell D_\ell^{-1} \\ &= (I_\ell - P_\ell D_{\ell+1}^{-1} R_\ell D_\ell)D_\ell^{-1} + P_\ell D_{\ell+1}^{-1} R_\ell \end{aligned} \tag{7.61}$$

7.3.3 Multigrid Multilevel Monte Carlo ++ for trace estimation

The main goal is to find a new trace estimator technique by merging between the Hutch++ method as discussed in section 6.2 and the MG-MLMC approach to reduce the variance of trace estimation of the matrix inverse and speed up the convergence rate of the solver which is used to invert the matrices at different levels [63].

We choose a suitable orthogonal projection

$$\Pi_\ell = Q_\ell^H Q_\ell \quad \in \mathbb{C}^{d_\ell \times d_\ell}, Q_\ell^H Q_\ell = I \tag{7.62}$$

where d_ℓ is the number of deflated vectors at each level difference and $1 \leq \ell < L$, then apply it on the level decomposition operator

$$M_\ell = D_\ell^{-1} - P_\ell D_{\ell+1}^{-1} R_\ell, \quad M_\ell \in \mathbb{C}^{n_\ell \times n_\ell}. \tag{7.63}$$

One obtains a decomposition on each level to reduce the variance as

$$M_\ell = \Pi_\ell M_\ell + (I - \Pi_\ell)M_\ell, \tag{7.64}$$

and

$$\mathrm{tr}(M_\ell) = \mathrm{tr}(\underbrace{\Pi_\ell M_\ell}_{\text{low-rank}}) + \mathrm{tr}(\underbrace{(I - \Pi_\ell) M_\ell}_{\text{stochastically}}) \tag{7.65}$$

In general, the approach can be decomposed into three phases:

1. Low-rank phase: it computes the trace of $\Pi_\ell M_\ell$ directly (no stochastic samples required) see (7.66).
2. Stochastic phase: works in the same way as on a level differences in MG-MLMC, but with an additional projection operator.
3. Coarsest phase: computes the trace of the coarsest level in the same manner as coearsest part in MG-MLMC

The trace of the low-rank part can be computed directly by using the trace cyclic property as

$$\mathrm{tr}(\Pi_\ell M_\ell) = \mathrm{tr}(Q_\ell^H M_\ell Q_\ell), \quad \in \mathbb{C}^{d_\ell \times d_\ell} \tag{7.66}$$

In the MG-MLMC++ approach, the number of deflation vectors d_ℓ for each level difference must be chosen a priori and we have three different phases plus the setup one:

1. Setup phase: the same as in the MG-MLMC approach.
2. Hutch++ phase: we compute the deflated vectors for each level difference and compute its trace directly.
3. Difference (fine-coarse) part: same as in MG-MLMC approach.
4. Coarsest part: similar to MG-MLMC approach.

7.3.4 Deflated MG-MLMC

In the same manner as MG-MLMC++, one can apply an exact projector Π_ℓ corresponding to each level difference ℓ in multigrid multilevel Monte Carlo which requires computing the largest singular vectors of the difference-level operator [43]. The difference level operator of multilevel Monte Carlo is given as:

$$M_\ell = D_\ell^{-1} - P_\ell D_{\ell+1}^{-1} R_\ell \tag{7.67}$$

For the lattice QCD and Wilson Dirace operator and Schwinger model the computation of singular triplets can be eigenpairs. Using the relations, we can build operators of the Hermitian difference level,

$$\Gamma_5^\ell P_\ell = P_\ell \Gamma_5^{\ell+1}, \quad P_\ell^H \Gamma_5^\ell = \Gamma_5^{\ell+1} P_\ell^H, \quad (\Gamma_5^\ell)^H = \Gamma_5^\ell, \quad (\Gamma_5^\ell)^H \Gamma_5^\ell = I, \tag{7.68}$$

which result from the "spin-preserving" algebraic multigrid building projects covered in [40] and which imply

$$Q_\ell := \Gamma_5^\ell D_\ell = D_\ell^H \Gamma_5^\ell = Q_\ell^H.$$

The hermitian operator can be obtained using these relations, which we can use to indirectly extract the singular vectors of M_ℓ as $J_\ell = M_\ell \Gamma_5^\ell$ then

$$\begin{aligned} J_\ell &= D_\ell^{-1}\Gamma_5^\ell - P_\ell D_{\ell+1}^{-1} P_\ell^H \Gamma_5^\ell \\ &= Q_\ell^{-1} - P_\ell D_{\ell+1}^{-1} \Gamma_5^{\ell+1} P_\ell^H \\ &= Q_\ell^{-1} - P_\ell Q_{\ell+1}^{-1} P_\ell^H. \end{aligned} \tag{7.69}$$

Then, the singular vectors of M_ℓ can be extracted using the eigenvectors of J_ℓ

$$J_\ell = X\Lambda X^H \Rightarrow M_\ell = X\Lambda X^H \Gamma_5^\ell \Rightarrow U = X\mathrm{sign}(\Lambda), S = \mathrm{abs}(\Lambda), V = \Gamma_5 X \tag{7.70}$$

7.3.5 A proto-type algorithm for MG-MLMC approaches

In this section, we imply a prototype Alg. 8 that one can use to implement the MG-MLMC approach with fixed accuracies for approximating the trace of the inverse of a square matrix D. In the next chapter, we will discuss further algorithms of MG-MLMC based on different stopping criteria types.

The algorithm operates on a hierarchy of grid levels, with restriction and prolongation operators $\hat{R}_\ell$ and $\hat{P}_\ell$ defining the relationships between levels. On each level ℓ, the algorithm generates a sequence of random vectors x_s distributed as in (5.10), and computes an approximation for the trace using the matrix D_ℓ on that level. The algorithm then calculates the sample mean and variance of the trace approximation, and continues iterating until the variance is within a specified relative accuracy ρ_ℓ for that level. The coarsest level is computed directly, without the use of random vectors. Once all levels have been processed, the algorithm returns an approximation for the trace of D^{-1}.

Algorithm 8 MG-MLMC with fixed accuracies: A proto-type algorithm

Input: $D \in \mathbb{C}^{n\times n}$, nonsingular, ϵ relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, D_ℓ matrix on level ℓ, $\ell = 1, \ldots, L$

Output: Approximation $\sum_{\ell=1}^{L} \tau_\ell$ for $\mathrm{tr}(D^{-1})$

fix relative accuracies ϵ_ℓ s.t. $\sum_{\ell=1}^{L-1} \epsilon_\ell^2 = \epsilon^2$
Compute rough estimate τ_{rough} for the trace ▷ via 5 samples in Algorithm 3
$\rho \leftarrow \epsilon\tau_{\text{rough}}$
for $\ell = 1, \ldots, L-1$ **do** ▷ all level differences
    $\rho_\ell \leftarrow \epsilon_\ell \rho$ ▷ accuracy for level difference ℓ
    for $s = 1, 2, \ldots$ **do**
        generate next random vector x_s ▷ x_s i.i.d. satisfying (5.10)
        $\tau_{s,\ell} \leftarrow x_s^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell x_s - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} x_s \right)$ ▷ two mg solves
        $\tau_\ell = \frac{1}{s} \sum_{i=1}^{s} \tau_{i,\ell}$, $\mathbb{V}_\ell = \frac{1}{s-1} \sum_{i=1}^{s} |\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
        if $\mathbb{V}_\ell / s \leq \rho_\ell^2$ **then**
            stop
        end if
    end for
end for
$\tau_L \leftarrow \sum_{i=1}^{N_L} (e_i^T P_L) D_L^{-1} (R_L e_i)$ ▷ coarsest level is computed directly

Chapter 8

Stopping criteria in Multigrid Multilevel Monte Carlo

One of the key considerations when choosing numerical algorithms is the computational cost, which refers to the number of floating point operations (FLOPS) performed by the algorithm. The cost is usually expressed in terms of big O notation, with the number of FLOPS represented by the exponent p. In Sec. 8.1, we discuss the underlying cost model for all of the numerical results presented in chapter 9. In Sec. 8.2, we examine the two main approaches for distributing the target variance among the difference levels of the multilevel method.

In Sec. 8.2.1, we first provide an overview of the main features of Multigrid Multilevel Monte Carlo (MG-MLMC) algorithms. Based on the type of stopping criteria used, since we can create two versions of each MG-MLMC algorithm. We then delve into a detailed analysis of each algorithm, starting with the uniform MG-MLMC, MG-MLMC++ and Deflated MG-MLMC algorithms in Sec. 8.2.2 and its optimized versions in Sec. 8.2.3. Finally, in Sec. 8.2.4, we discuss how the MG-MLMC approaches can be made less costly by using the skipping technique.

8.1 Cost Model

In this section, we discuss the cost model that underlies all numerical methods in this thesis. This model evaluates the performance of algorithms by counting the number of arithmetic operations in matrix-vector multiplications, including projections, restrictions and prolongations, and smoothing iterations in the multigrid solver. We neglect all operations on vectors and scalars. This number is presented as an indication for the operators at different levels, and is proportional to the

number of non-zeros in the corresponding matrix. We report two main quantities for each experiment: (1) the number of stochastic samples run for the multilevel Monte Carlo method and the number of stochastic samples in the deflated Hutchinson method at each level difference (which always require linear solves at the finest level), and (2) the approximate arithmetic costs for the approach, which are calculated using the cost model. These data provide insight into how the multilevel Monte Carlo method shifts higher variances to coarser level differences.

The computation of the quantity $\psi^H \hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi$ is the main focus of the cost model that we are interested in as follows: in this computation, we only take into account the matrix-vector products. These result from multiplications with the variables $\hat{P}_\ell$ and $\hat{R}_\ell$ as well as from the matrix-vector multiplications we carry out in the multigrid solver that we employ to compute $D_\ell^{-1} y$ at each level ℓ.

We assume a cost of $nnz(B)$, the number of nonzeroes in B, for each matrix-vector product of the generic form Bx. Accordingly, one unit in the cost model roughly equates to a multiplication plus an addition. This applies to the computation of residuals, of prolongations and restrictions and the coarsest grid solve in the multigrid solver as well as to the *global* restrictions and prolongations $\hat{R}_\ell, \hat{P}_\ell$ used in each stochastic sample in multilevel Monte-Carlo.

For all multilevel algorithms, we also count the cost for the direct computation of the trace at the coarsest level, which involves the inversion of the coarsest grid matrix and additional matrix-matrix products.

In case of deflation, we used the k smallest eigenmodes that we precomputed, and then optimized k so as to obtain the smallest overall cost, *excluding* the cost for the eigenvector computation. So the work for deflated Hutchinson is actually higher than what we report. We use our cost model for the following approaches:

1. MG-MLMC method, we are considering the stochastic work is the stochastic samples (on levels differences) plus cost for the direct computational at the coarsest level.

2. MG-MLMC++ method, same as the first one plus the work to compute and apply the projections for the low-rank part on each level difference.

3. Deflated MG-MLMC, is the same as in 2 with the exception of computing the projections , since this uses subtractions for eigenpairs for which we lack a cost model.
   The cost of the low-rank part remains indeterminate in this scenario due to the absence of a cost model that can accurately determine the specific eigenpairs of the deflation projection.

8.1.1 MG-MLMC

Given: number of levels L, difference levels $l = 1, 2, \dots L-1$, and at each level we have operators $D_\ell, P_\ell, R_\ell, \hat{P}_\ell, \hat{R}_\ell$. The level difference operator is given as:

$$M_\ell = \hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \tag{8.1}$$

and its trace is

$$\begin{aligned} \mathrm{tr}(M_\ell) &= \mathrm{tr}(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \underbrace{\hat{P}_{\ell+1}}_{=\hat{P}_\ell P_\ell} D_{\ell+1}^{-1} \hat{R}_{\ell+1}) \\ &= \mathrm{tr}(D_\ell^{-1} \hat{R}_\ell \hat{P}_\ell - P_\ell D_{\ell+1}^{-1} \underbrace{\hat{R}_{\ell+1}}_{R_\ell \hat{R}_\ell} \hat{P}_\ell). \end{aligned} \tag{8.2}$$

We distinguish two cases for $\hat{P}$ and $\hat{R}$.

1. **General case:** we can use the formula as it stands and apply plain Hutchinson or Hutch ++ method for the matrix:

$$D_\ell^{-1} \hat{R}_\ell \hat{P}_\ell - P_\ell D_{\ell+1}^{-1} R_\ell \hat{R}_\ell \hat{P}_\ell \tag{8.3}$$

   (a) This means that the work is the MVPs with $2 \times \hat{P}_\ell, 2 \times \hat{R}_\ell, P_\ell, R_\ell$:

$$W_{Stoch}^\ell = 2 \times \mathrm{nnz}(\hat{P}_\ell) + 2 \times \mathrm{nnz}(\hat{R}_\ell) + \mathrm{nnz}(P_\ell) + \mathrm{nnz}(R_\ell) \tag{8.4}$$

2. **Aggregation case:** we have $\hat{P}_\ell \hat{R}_\ell = I$ and reduce the level differences to

$$D_\ell^{-1} - P_\ell D_{\ell+1}^{-1} R_\ell \tag{8.5}$$

   (a) The stochastic work is MVPs with P_ℓ, R_ℓ:

$$W_{stoch}^\ell = \mathrm{nnz}(P_\ell) + \mathrm{nnz}(P_\ell) \tag{8.6}$$

(b) Compute the work W_{solv}^ℓ for solving the linear systems of D_ℓ and $D_{\ell+1}$

We have levels $\ell = 1, \dots, L,$ where L is the coarsest level, and the $\mathrm{tr}(\hat{P}_L D_L \hat{R}_L)$ can be computed directly using the cyclic property of the trace. The work required for this direct part consists of the work to compute the inverse of the matrix D_L^{-1}, the product $\hat{R}\hat{P}.$ and the product $(\hat{R}\hat{P})D_L^{-1}$.

(c) Then the direct work is given as:

$$W_{dir} = 2 \times n_L^3 + n_L \times nnz(\hat{P}_L). \tag{8.7}$$

The total work is the sum of all parts:

$$W_{mlmc} = W^{\ell}_{stoch} + W^{\ell}_{solve} + W_{dir}. \tag{8.8}$$

8.1.2 MG-MLMC++

When dealing with the MG-MLMC++ case, we construct the matrix Q_ℓ in the projection $I - Q_\ell Q_\ell^H$ from the matrix $S \in \mathbb{C}^{n_\ell \times d_\ell}$. Then the total work will be as before plus the cost of the low-rank part, which is described as:

- the work of the QR factorization:

$$W_{qr} = 2 \times (n_\ell - d_\ell/3) \times d_\ell^2 \times k \tag{8.9}$$

  since k is the number of power iterations, where MG-MLMC++ as introduced do at least ope power iteration.

- the work of computing the low-rank trace directly $\mathrm{tr}(Q_\ell D_\ell^{-1} Q_\ell) \in \mathbb{C}^{d_\ell \times d_\ell}$:

$$W_{dirtr} = n_\ell \times d_\ell + \mathrm{nnz}(P_\ell) + \mathrm{nnz}(R_\ell) \tag{8.10}$$

- then the work for the projection term $Q_\ell(Q_\ell^H x)$ is

$$W_{proj} = 2 \times n_\ell \times d_\ell \tag{8.11}$$

- Compute the work W^{lr}_{solv} for solving the linear systems.

- the low-rank work is given as,

$$W^{\ell}_{LR} = W_{qr} + W_{dirtr} + W_{proj} + W^{\mathrm{lr}}_{solv} \tag{8.12}$$

then the total work would be the sum of all parts:

$$W_{mlmc++} = W^{\ell}_{LR} + W^{\ell}_{stoch} + W^{\ell}_{solve} + W_{dir} \tag{8.13}$$

In the numerical experiments, one can consider time measurements as a cost model rather than FLOPS by measuring the execution times of every particular operation, including solving systems with D_ℓ, deflations, P_ℓ, P_ℓ^H and axpy operations.

8.2 Distributing the variance

In the following, we discuss two different types of variance distribution for MG-MLMC algorithms and show their impact on the convergence speed of linear systems of equations involving the inverse of the matrix D^{-1}. Assume we perform the variance $\mathbb{V}_\ell$ and stochastic samples N_ℓ at each level difference for independent samples

$$w_\ell = \psi^H D_\ell^{-1} \psi, \ell = 1, 2, \cdots, L. \tag{8.14}$$

Then, in order to achieve a target variance of $\rho = \varepsilon \operatorname{tr}(D^{-1})$ in the stochastic estimation, we should have

$$\sum_{\ell=1}^{L-1} \frac{\mathbb{V}_\ell}{N_\ell} \leq \rho^2. \tag{8.15}$$

8.2.1 Main features in algorithms

Before implementing each algorithm in detail, we first outline the most important features that are considered in algorithms in Sec. 8.2.2 and Sec. 8.2.3:

- All algorithms share common assumptions, including the initial matrix $D \in \mathbb{C}^{n \times n}$, the relative accuracy ε, and the total number of hierarchical levels $\ell, \quad \ell = 1, \ldots, L$.
- The multilevel hierarchy operators D_ℓ, P_ℓ and R_ℓ are constructed during the setup phase, and we refer to them as givens. The setup phase, which has been thoroughly covered in Chapter 4, is disregarded in all multilevel algorithms.
- In all multilevel algorithms, there are two main phases that are presented: first, the stochastic phase and followed by the coarsest phase. The low-rank term is an additional phase found in the MG-MLMC++ and Deflated MG-MLMC algorithms.
- The goal is to calculate the trace of the inverse matrix $\operatorname{tr}(D^{-1})$. This is obtained by approximating the trace for each phase and adding them all up to get the estimated trace value in total.
- The target variance is given as $\rho = \varepsilon \operatorname{tr}(D^{-1})$. Since we are unsure of the exact trace, we compute a rough estimate τ_{rough} for the trace via 5 samples as in Algorithm 3 to determine ρ.
- For all algorithms, we generate random vectors ψ_s and use them to compute a stochastic estimate of the trace at each level:

1. **uniform case**: The sample mean and variance of these estimates are used to determine whether the required accuracy has been reached. If the variance of the estimates is less than the required accuracy squared, the level is considered to have reached the required accuracy and the algorithm moves on to the next level. If the variance of the estimates is greater than the required accuracy squared, more samples are taken until the required accuracy is reached.

2. **optimal case**: The sample mean and variance are calculated and used to update the target accuracy for each active level. If the variance of a level difference falls below the required accuracy, the level is marked as inactive, indicating that it has reached the required accuracy. This process continues until all levels are inactive, at which point the algorithm terminates.

- In general, we use the multigrid method to solve the linear systems.
- We usually compute the trace of the coarsest level directly except for the very large matrices at very higher target accuracy because of the size of the coarsest matrix still too large and costly to compute the trace of its inverse directly so we estimate the trace in that case as stochasticly as Plain Hutchinson approach.
- For the MG-MLMC++ and deflated MG-MLMC algorithms, the number of deflation vectors d_ℓ for each level difference must be chosen a priori.
- Deflation vectors for Deflated MG-MLMC algorithm are computed as in the exact deflation approach in 6.1.2
- To measure the cost for a stochastic sample, we consider the cost model in sec. 8.1.1 for the MG-MLMC algorithms and consider the cost model in sec. 8.1.2 for MG-MLMC++ algorithms.
- All the uniform algorithms use the uniform variance distribution in 8.2.2, whereas the optimal algorithms are based on the optimal variance distribution in 8.2.3.
- We describe the algorithms using the decomposition (8.3) with the accumulated prolongations and restrictions. The adaptation to eq. (8.5), should it apply, is straightforward.
- For algorithms based on the optimal variance distribution as discussed in sec. 8.2.3,

    1. we take averages of the cost for each stochastic sample to get an increasingly accurate average cost estimate C_ℓ.

2. With this measured cost and the measured sample variance $\mathbb{V}_\ell$ we determine the optimal target variance from (8.17) for each level difference.

3. This target variance is updated at each additional sample on that level difference.

- For MG-MLMC++ algorithms, Lines 5 and 6 perform one step of the block power iteration, the crucial ingredient of the Hutch++ method. We can perform more than 1, k say, iterations of the block power method by repeating these lines with S_ℓ in the next sweep equal to Q_ℓ from the previous sweep.

8.2.2 Algorithms based on the uniform variance distribution

The uniform variance distribution case is achieved by distributing the target accuracy ρ^2 in equal and asking for $\frac{\mathbb{V}_\ell}{N_\ell} \leq \frac{\rho^2}{L-1}$ for all ℓ, and thus the target accuracy for each level difference is given by:

$$\rho_\ell = \epsilon\tau/\sqrt{L-1} \text{ for all } \ell, \tag{8.16}$$

In practice, this kind of distribution works well and gives good results in simple examples such as $2d$ Laplace and Gauge Laplace matrices [41], however, it is impractical for more advanced problems such as the Schwinger model.So we need to distribute the target variance in an optimal way as in section 8.2.3 see Alg. 12.

Algorithm 9 shows how to implement the MG-MLMC method for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ based on the uniform variance distribution. To obtain the total value of the trace, we first estimate the trace of the fine-coarse levels differences stochastically $\tau_\ell, \ell = 1, \cdots, L-1$, then compute the trace of the coarsest level directly τ_L, and finally, the sum of all trace estimates at each level is returned as the result.

Algorithm 10 explores evaluating the MG-MLMC++ approach for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ relying on the uniform variance distribution, with a fixed number of deflation vectors d_ℓ at each level difference. To determine the total value of the trace, we first decompose the matrix by applying the Hutch++ approach, then compute the trace of the low-rank stage τ_ℓ^{lr}, estimate the trace of the differences between the fine and coarse levels stochastically $\tau_\ell, \ell = 1, \cdots, L-1$, compute the trace of the coarsest level directly τ_L, and sum all values.

Implementing the Deflated MG-MLMC approach for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ based on the uniform variance distribution with

Algorithm 9 MG-MLMC, uniform accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ε relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, D_ℓ matrix on level ℓ, $\ell = 1, \dots, L$

Output: Approximation $\sum_{\ell=1}^{L} \tau_\ell$ for $\mathrm{tr}(D^{-1})$

$\tau_L \leftarrow \sum_{i=1}^{N_L} (e_i^H \hat{P}_L) D_L^{-1} (\hat{R}_L e_i)$ ▷ coarsest level is computed directly
Compute rough estimate τ_{rough} for the trace ▷ via 5 samples in Algorithm 3
$\rho \leftarrow \varepsilon \tau_{\text{rough}}$
for $\ell = 1, \dots, L-1$ **do** ▷ all level differences
  $\rho_\ell \leftarrow \rho/\sqrt{L-1}$ ▷ accuracy for level difference ℓ
  for $s = 1, 2, \dots$ **do**
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $\tau_{s,\ell} \leftarrow \psi_s^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi_s - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \psi_s \right)$ ▷ two mg solves
    $\tau_\ell = \frac{1}{s} \sum_{i=1}^{s} \tau_{i,\ell}$, $\mathbb{V}_\ell = \frac{1}{s-1} \sum_{i=1}^{s} |\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
    if $\mathbb{V}_\ell / s \leq \rho_\ell^2$ **then**
      stop
    end if
  end for
end for

a fixed number of deflation vectors d_ℓ at each level difference is shown in Algorithm 11, taking into account the aforementioned features. To obtain the total value of the trace, we first decompose the matrix by applying the exact deflation, then compute the trace of the deflated part τ_ℓ^{defl}, estimate the trace of the differences between the fine and coarse levels stochastically τ_ℓ, compute the trace of the coarsest level directly as τ_L, and sum all values.

Algorithm 10 MG-MLMC++, uniform accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ϵ relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, $D_\ell \in \mathbb{C}^{n_\ell \times n_\ell}$ matrix on level ℓ, d_ℓ number of deflation vectors on level ℓ, $\ell = 1, \dots, L$.

Output: Approximation $\sum_{\ell=1}^{L-1}(\tau_\ell^{\mathrm{lr}} + \tau_\ell) + \tau_L$ for $\mathrm{tr}(D^{-1})$

for $\ell = 1, \dots, L-1$ **do** ▷ obtain deflation vectors
  generate d_ℓ i.i.d. random vectors $s_i, i = 1, \dots, d_\ell$, ▷ with distribution satisfying (5.10)
  collect them as columns in $S_\ell \in \mathbb{C}^{n\times d_\ell}$
  $Y_\ell \leftarrow \left(\hat{P}_\ell D_\ell^{-1}\hat{R}_\ell - \hat{P}_{\ell+1}D_{\ell+1}^{-1}\hat{R}_{\ell+1}\right) S_\ell$, ▷ $Y_\ell \in \mathbb{C}^{n\times d_\ell}$, $2d_\ell$ mg solves.
  Compute QR-factoriz. $Y_\ell = Q_\ell K_\ell$ ▷ $Q_\ell = [q_1|\cdots|q_{d_\ell}] \in \mathbb{C}^{n\times d_\ell}$ has orthon. cols
  $\tau_\ell^{\mathrm{lr}} \leftarrow \sum_{i=1}^{d_\ell} q_i^H \left(\hat{P}_\ell D_\ell^{-1}\hat{R}_\ell - \hat{P}_{\ell+1}D_{\ell+1}^{-1}\hat{R}_{\ell+1}\right) q_i$ ▷ *low rank part*, use mg to solve lin. sys.
end for
$\tau_L \leftarrow \sum_{i=1}^{N_L}(e_i^H \hat{P}_L) D_L^{-1}(\hat{R}_L e_i)$ ▷ coarsest level is computed directly
Compute rough estimate τ_{rough} for the trace ▷ via 5 samples in Algorithm 3
$\rho \leftarrow \varepsilon\tau_{\mathrm{rough}}$
for $\ell = 1, \dots, L-1$ **do** ▷ all level differences
  $\rho_\ell \leftarrow \rho/\sqrt{L-1}$ ▷ accuracy for level difference ℓ
  for $s = 1, 2, \dots$ **do**
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $z_s \leftarrow \psi_s - Q_{d_\ell}(Q_{d_\ell}^H \psi_s)$ ▷ projected vector
    $\tau_{s,\ell} \leftarrow z_s^H \left(\hat{P}_\ell D_\ell^{-1}\hat{R}_\ell\psi_s - \hat{P}_{\ell+1}D_{\ell+1}^{-1}\hat{R}_{\ell+1}\psi_s\right)$ ▷ two mg solves
    $\tau_\ell = \frac{1}{s}\sum_{i=1}^{s}\tau_{i,\ell}$, $\mathbb{V}_\ell = \frac{1}{s-1}\sum_{i=1}^{s}|\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
    if $\mathbb{V}_\ell/s \le \rho_\ell^2$ **then**
      stop
    end if
  end for
end for

Algorithm 11 Deflated MG-MLMC, uniform accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ϵ relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, $D_\ell \in \mathbb{C}^{n_\ell \times n_\ell}$ matrix on level ℓ, d_ℓ number of deflation vectors on level ℓ, $\ell = 1, \ldots, L$,

Output: Approximation $\sum_{\ell=1}^{L-1}(\tau_\ell^{\text{defl}} + \tau_\ell) + \tau_L$ for $\operatorname{tr}(D^{-1})$

for $\ell = 1, \ldots, L-1$ **do** ▷ obtain deflation vectors
  compute left singular vectors q_i of $\left(\hat{P}_\ell D_\ell \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1} \hat{R}_{\ell+1}\right), i = q_1, \cdots, q_{d_\ell}$
  $Q_{d_\ell} = [q_1 | \cdots | q_{d_\ell}]$
  $\tau_\ell^{\text{defl}} \leftarrow \sum_{i=1}^{d_\ell} q_i^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1}\right) q_i$ ▷ *low rank*, solve lin. systems using mg.
end for
$\tau_L \leftarrow \sum_{i=1}^{N_L} (e_i^H \hat{P}_L) D_L^{-1} (\hat{R}_L e_i)$ ▷ coarsest level is computed directly
Compute rough estimate τ_{rough} for the trace ▷ via 5 samples in Algorithm 3
$\rho \leftarrow \varepsilon \tau_{\text{rough}}$
for $\ell = 1, \ldots, L-1$ **do** ▷ all level differences
  $\rho_\ell \leftarrow \rho/\sqrt{L-1}$ ▷ accuracy for level difference ℓ
  for $s = 1, 2, \ldots$ **do**
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $z_s \leftarrow \psi_s - Q_{d_\ell}(Q_{d_\ell}^H \psi_s)$ ▷ projected vector
    $\tau_{s,\ell} \leftarrow z_s^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi_s - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \psi_s\right)$ ▷ two mg solves
    $\tau_\ell = \frac{1}{s}\sum_{i=1}^{s} \tau_{i,\ell}$, $\mathbb{V}_\ell = \frac{1}{s-1}\sum_{i=1}^{s} |\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
    if $\mathbb{V}_\ell / s \le \rho_\ell^2$ **then**
      stop
    end if
  end for
end for

8.2.3 Algorithms based on the optimal variance distribution

This kind of distribution relies on stopping criteria which use the current sample variances $\mathbb{V}_\ell$ and the current estimates for the costs C_ℓ. If these estimates were indeed the exact variances and the exact cost (per stochastic sample), and if τ were the exact trace, we see that the computation at level ℓ would stop after having performed $N\ell$ stochastic samples with N_ℓ the smallest number for which

$$\mathbb{V}_\ell/N_\ell \leq \left(\sqrt{C_\ell \mathbb{V}_\ell} \Big/ \sum_{j=1}^{L-1} \sqrt{C_j \mathbb{V}_j}\right) \rho^2, \tag{8.17}$$

where $\rho = \varepsilon \operatorname{tr}(D^{-1})$ and ε dentes to the relative accuracy. Thus, up to rounding to the next integer, we have

$$N_\ell = \frac{1}{\rho^2} \sqrt{C_\ell \mathbb{V}_\ell} \sum_{j=1}^{L-1} \sqrt{C_j \mathbb{V}_j}, \tag{8.18}$$

which is the optimal number of samples such that the total cost is minimized, see section 7.2.2.

A variant of the uniform distribution as in Algorithms in Sec. 8.2.2, turned the optimal distribution, updates the parameter ρ_ℓ in a unique way. This approach is based on the following reasoning: for non-active levels, the sample variances $\mathbb{V}_\ell/N_\ell$ are taken as the exact variances for their contribution τ_ℓ to the trace. Therefore, the variances for the remaining active levels should add up to a variance of $\rho^2 - \sum_{j=1,j \text{ non active}}^{L} \rho_j^2$. Algorithm 0 is used to determine the target accuracy to aim for on each of the active levels, in order to achieve minimal cost for this requirement.

$$\rho_\ell \leftarrow \left(\sqrt{C_\ell \mathbb{V}_\ell} \,/\, \textstyle\sum_{j=1,j \text{ active}}^{L-1} \sqrt{C_j \mathbb{V}_j}\right)^{1/2} \cdot \left((\varepsilon\tau)^2 - \textstyle\sum_{j=1,j \text{ non active}}^{L-1} \rho_j^2\right)^{1/2}$$

This optimal approach distributes the target accuracy at the levels differences to ensure that the lowest possible occupancy and fits the difference across each level. On the other hand, we can notice that for each stochastic sample the accuracy dynamically adapts during the experiment to each level difference according to the variance and the cost quantities that we estimated per each sample. Based on the accuracy type, one can display different methods, including:

- Deflated Hutchinson (Exact deflation) 4 which is used as a reference for comparison. We did not use non-deflated Hutchinson, because its performance is by two orders of magnitude worse than that of deflated Hutchinson.

- MG-MLMC with uniform target variances on the levels differences and its modification working with optimal target variances and the corresponding two versions for MG-MLMC++ and Deflated MG-MLMC.

Algorithm 12 describes the implementation of the MG-MLMC approach for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ relying on the optimal variance distribution. To obtain the total value of the trace, we do as same as in Alg. 9.

Algorithm 12 MG-MLMC, optimal accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ε relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, D_ℓ matrix on level ℓ, $\ell = 1, \ldots, L$

Output: Approximation $\sum_{\ell=1}^{L} \tau_\ell$ for $\mathrm{tr}(D^{-1})$

Set all levels ℓ to active ▷ non active levels have reached required accuracy
$\tau_L \leftarrow \sum_{i=1}^{N_L} (e_i^H \hat{P}_L) D_L^{-1} (\hat{R}_L e_i)$ ▷ coarsest level is computed directly
for $s = 1, 2, \ldots$ **until** all levels ℓ are not active **do**
  for $\ell = 1, \ldots, L-1$ **and** ℓ is active **do** ▷ next stoch. est.
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $\tau_{s,\ell} \leftarrow \psi_s^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi_s - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \psi_s \right)$, cost is $C_{s,\ell}$
    $\tau_\ell = \frac{1}{s} \sum_{i=1}^{s} \tau_{i,\ell}$, $\mathbb{V}_\ell = \frac{1}{s-1} \sum_{i=1}^{s} |\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
    $C_\ell = \frac{1}{s} \sum_{i=1}^{s} C_{s,\ell}$ ▷ average cost per sample
  end for
  $\tau = \sum_{\ell=1}^{L} \tau_\ell$
  for $\ell = 1, \ldots, L-1$ **and** ℓ is active **do** ▷ update target accuracies ρ_ℓ
    $\rho_\ell \leftarrow \left(\sqrt{C_\ell \mathbb{V}_\ell} \,/ \sum_{j=1}^{L-1} \sqrt{C_j \mathbb{V}_j} \right)^{1/2} \cdot (\varepsilon \tau)$
    if $\mathbb{V}_\ell / s \le \rho_\ell^2$ **then**
      set level ℓ to inactive ▷ $N_\ell = \mathrm{s}$
    end if
  end for
end for

Algorithm 13 describes the implementation of the MG-MLMC++ approach for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ relying on the optimal variance distribution, with a fixed number of deflation vectors d_ℓ at each level difference. We follow the same procedure as in 10 to obtain the total value of the trace.

Algorithm 13 MG-MLMC++, optimal accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ϵ relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, $D_\ell \in \mathbb{C}^{n_\ell \times n_\ell}$ matrix on level ℓ, d_ℓ number of deflation vectors on level ℓ, $\ell = 1, \ldots, L$,

Output: Approximation $\tau + \tau_L$ for $\mathrm{tr}(D^{-1})$

for $\ell = 1, \ldots, L-1$ **do** ▷ obtain deflation vectors
  generate d_ℓ i.i.d. random vectors $s_i, i = 1, \ldots, d_\ell$, ▷ with distribution satisfying (5.10)
  collect them as columns in $S_\ell \in \mathbb{C}^{n\times d_\ell}$
  $Y_\ell \leftarrow \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1}\right) S_\ell$, ▷ $Y_\ell \in \mathbb{C}^{n\times d_\ell}$, , $2d_\ell$ mg solves.
  Compute QR-factoriz. $Y_\ell = Q_\ell K_\ell$ ▷ $Q_\ell = [q_1 | \cdots | q_{d_\ell}] \in \mathbb{C}^{n\times d_\ell}$ has orthon. cols
  $\tau_\ell^{\mathrm{lr}} \leftarrow \sum_{i=1}^{d_\ell} q_i^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1}\right) q_i$ ▷ *low rank part*, use mg to solve lin. sys.
end for
$\tau_L \leftarrow \sum_{i=1}^{N_L} (e_i^H \hat{P}_L) D_L^{-1} (\hat{R}_L e_i)$ ▷ coarsest level is computed directly
Set all levels ℓ to active ▷ non active levels have reached required accuracy
for $s = 1, 2, \ldots$ **until** all levels ℓ not active **do** ▷ *stochastic part*
  for $\ell = 1, \ldots, L-1$ **and** ℓ is active **do**
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $z_s = \psi_s - Q_\ell (Q_\ell^H \psi_s)$ ▷ projected vector
    $\tau_{s,\ell} \leftarrow z_s^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi_s - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \psi_s\right)$
    $C_{s,\ell} \leftarrow$ cost for lines 11 - 12
    $\tau_\ell = \frac{1}{s} \sum_{i=1}^{s} \tau_{i,\ell}$, $V_\ell = \frac{1}{s-1} \sum_{i=1}^{s} |\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
    $C_\ell = \frac{1}{s} \sum_{i=1}^{s} C_{i,\ell}$ ▷ average cost per sample
  end for
  $\tau = \sum_{\ell=1}^{L} (\tau_\ell + \tau_\ell^{\mathrm{lr}})$
  for $\ell = 1, \ldots, L-1$ **and** ℓ is active **do** ▷ update target accuracies ρ_ℓ
    $\rho_\ell \leftarrow \left(\sqrt{C_\ell V_\ell} \,/\, \sum_{j=1}^{L-1} \sqrt{C_j V_j}\right)^{1/2} \cdot (\epsilon\tau)$
    if $V_\ell / s \le \rho_\ell^2$ **then**
      set level ℓ to inactive
    end if
  end for
end for

Algorithm. 14 illustrates implementing the Deflated MG-MLMC method for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ based on the optimal variance distribution with a fixed number of deflation vectors d_ℓ at each level difference. We follow the same procedure as in 11 to obtain the total value of the trace.

Algorithm 14 Deflated MG-MLMC, optimal accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ϵ relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, $D_\ell \in \mathbb{C}^{n_\ell \times n_\ell}$ matrix on level ℓ, d_ℓ number of deflation vectors on level ℓ, $\ell = 1, \ldots, L$,

Output: Approximation $\sum_{\ell=1}^{L-1}(\tau_\ell^{\mathrm{defl}} + \tau_\ell) + \tau_L$ for $\mathrm{tr}(D^{-1})$

for $\ell = 1, \ldots, L-1$ **do** ▷ obtain deflation vectors
  compute left singular vectors q_i of $\left(\hat{P}_\ell D_\ell \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1} \hat{R}_{\ell+1}\right), i = q_1, \cdots, q_{d_\ell}$
  $Q_{d_\ell} = [q_1 | \cdots | q_{d_\ell}]$
  $\tau_\ell^{\mathrm{defl}} \leftarrow \sum_{i=1}^{d_\ell} q_i^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1}\right) q_i$ ▷ *deflated part*, solve lin. systems using mg.
end for
$\tau_L \leftarrow \sum_{i=1}^{N_L} (e_i^H \hat{P}_L) D_L^{-1} (\hat{R}_L e_i)$ ▷ coarsest level is computed directly
Set all levels ℓ to active ▷ non active levels have reached required accuracy
for $s = 1, 2, \ldots$ **until** all levels ℓ not active **do** ▷ *stochastic part*
  for $\ell = 1, \ldots, L-1$ **and** ℓ is active **do**
    generate next random vector ψ_s ▷ ψ_s i.i.d. satisfying (5.10)
    $z_s = \psi_s - Q_\ell (Q_\ell^H \psi_s)$ ▷ projected vector
    $\tau_{s,\ell} \leftarrow z_s^H \left(\hat{P}_\ell D_\ell^{-1} \hat{R}_\ell \psi_s - \hat{P}_{\ell+1} D_{\ell+1}^{-1} \hat{R}_{\ell+1} \psi_s\right)$
    $C_{s,\ell} \leftarrow$ cost for lines 11 - 12
    $\tau_\ell = \frac{1}{s}\sum_{i=1}^{s} \tau_{i,\ell}$, $V_\ell = \frac{1}{s-1}\sum_{i=1}^{s} |\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
    $C_\ell = \frac{1}{s}\sum_{i=1}^{s} C_{i,\ell}$ ▷ average cost per sample
  end for
  $\tau = \sum_{\ell=1}^{L} (\tau_\ell + \tau_\ell^{\mathrm{defl}})$
  for $\ell = 1, \ldots, L-1$ **and** ℓ is active **do** ▷ update target accuracies ρ_ℓ
    $\rho_\ell \leftarrow \left(\sqrt{C_\ell V_\ell} \,/\, \sum_{j=1}^{L-1} \sqrt{C_j V_j}\right)^{1/2} \cdot (\epsilon\tau)$
    if $V_\ell / s \le \rho_\ell^2$ **then**
      set level ℓ to inactive
    end if
  end for
end for

Algorithm 15 describes the implementation of the Adaptive MG-MLMC++ approach for estimating the trace of the inverse of the matrix $\mathrm{tr}(D^{-1})$ relying on the optimal variance distribution, with a dynamical number of deflation vectors

d_ℓ at each level difference. We follow the same procedure as in 13 to obtain the total value of the trace.

Algorithm 15 Adaptive MG-MLMC++, optimal accuracies

Input: $D \in \mathbb{C}^{n\times n}$ nonsingular, ϵ relative accuracy, L number of levels, $\hat{R}_\ell, \hat{P}_\ell$ restriction and prolongation operators between levels 1 and ℓ, $D_\ell \in \mathbb{C}^{n_\ell \times n_\ell}$ matrix on level ℓ, $\ell = 1, \ldots, L$,

Output: Approximation $\sum_{\ell=1}^{L-1}(\tau_\ell^{\text{lr}} + \tau_\ell) + \tau_L$ for $\text{tr}(D^{-1})$

1: Set all levels ℓ to active ▷ non active levels have reached required accuracy
2: **for** $s = 1, 2, \ldots$ **until** all levels ℓ not active **do** ▷ *stochastic part*
3: **for** $\ell = 1, \ldots, L-1$ **and** ℓ is active **do**
4: generate $d_{\ell,s}$ = i.i.d. random vectors
5: collect them as columns in $S_\ell \in \mathbb{C}^{n\times d_{\ell,s}}$
6: $Y_\ell \leftarrow \left(\hat{P}_\ell D_\ell^{-1}\hat{R}_\ell - \hat{P}_{\ell+1}D_{\ell+1}^{-1}\hat{R}_{\ell+1}\right)\eta_\ell$, ▷ $Y_\ell \in \mathbb{C}^{n\times d_\ell}$, , $2d_\ell$ mg solves.
7: Compute QR-factoriz. $Y_\ell = Q_\ell K_\ell$ ▷ $Q_\ell = [q_1|\cdots|q_{d_{\ell,s}}] \in \mathbb{C}^{n\times d_{\ell,s}}$ has orthon. cols
8: $C_{s,\ell} \leftarrow$ cost
9: $\tau_\ell^{\text{lr}} = \frac{1}{s}\sum_{i=1}^{s}\tau_{i,\ell}$, $V_\ell = \frac{1}{s-1}\sum_{i=1}^{s}|\tau_{i,\ell} - \tau_\ell|^2$ ▷ sample mean and variance
10: $C_\ell = \frac{1}{s}\sum_{i=1}^{s} C_{i,\ell}$ ▷ average cost per sample
11: $\rho_\ell \leftarrow \left(\sqrt{C_\ell V_\ell}\,/\,\sum_{j=1}^{L-1}\sqrt{C_j V_j}\right)^{1/2}\cdot(\epsilon\tau^{\text{lr}})$
12: **if** $V_\ell/s \le \rho_\ell^2$ **then**
13: set level ℓ to inactive
14: **end if**
15: **end for**
16: **end for**
17: $\tau_L \leftarrow \sum_{i=1}^{N_L}(e_i^H\hat{P}_L)D_L^{-1}(\hat{R}_L e_i)$ ▷ coarsest level is computed directly
18: do stepts from 9 to 26 in Alg. 13 to compute τ_l

8.2.4 Skipping levels approach

According to numerical experiments, in some MG-MLMC cases such as the Schwinger example, the number of nonzero elements in the projected matrix $D_2 = R_1 D_1 P_1$ is nearly the same as at the second level difference $\ell = 2$ which makes the second level nearly as expensive as the first. Consequently, the concept of skipping levels might be a helpful way to let us get rid of the second-level difference [41].

Assume four-level multigrid hierarchy since the number of levels differences $\ell = 3$ then the operator at each level difference is given as:

$$D_1^{-1} = \underbrace{D_1^{-1} - \hat{P}_1 D_2^{-1} \hat{R}_1}_{\text{1st level diff}} + \underbrace{\hat{P}_1 D_2^{-1} \hat{R}_1 - \hat{P}_2 D_3^{-1} \hat{R}_2}_{\text{2nd level diff}} + \underbrace{\hat{P}_2 D_3^{-1} \hat{R}_2 - \hat{P}_3 D_4^{-1} \hat{R}_3}_{\text{3rd level diff}} + \hat{P}_3 D_4^{-1} \hat{R}_3. \tag{8.19}$$

We can contract the two first terms , skipping the inversion with D_2, to get

$$D_1^{-1} = D_1^{-1} - \hat{P}_2 D_3^{-1} \hat{R}_2 + \hat{P}_2 D_3^{-1} \hat{R}_2 - \hat{P}_3 D_4^{-1} \hat{R}_3 + \hat{P}_3 D_4^{-1} \hat{R}_3. \tag{8.20}$$

and now estimate the trace of these three (instad of four) terms.

As a result of avoiding inversions with D_2 during the multilevel trace expansion, the first term in (8.20) may have a higher variance than the first or second terms in the summation in (7.60). Skipping levels may be caused by increasing the number of samples because the subtraction is less accurate. Although, it could be helpful to apply the Hutch++ or the exact deflation on top of the MG-MLMC approach as in 7.3.3 and 7.3.4 respectively.

Chapter 9

Numerical Results

In this chapter, four different mathematical models used in physics for numerical experiments are presented and analyzed, including the standard discrete 2−dimensional Laplacian in Sec. 9.2, the 2−dimensional gauge Laplacian in Sec. 9.3, the Schwinger model in Sec. 9.4 and the Wilson-Dirac matrix found in Lattice QCD in Sec. 9.5. Each of these models has its own level of complexity and is used in different areas of physics research.

The $2d$ Laplacian is the simplest model among the four and is a symmetric, positive definite matrix with an analytical known trace. It is often used as a basic test problem for numerical methods. The gauge Laplacian is a modification of the $2d$ Laplacian, where the coupling coefficients are complex numbers with random phases. It is used as a first step in modeling gauge field theories in physics. It is a little more complex than the $2d$ Laplacian, but still relatively straightforward.

The Schwinger model is a quantum field theory that describes the behavior of electrons and positrons in the presence of an electromagnetic field. It results in the 2−dimentional Dirac operator and it is considered to be a relatively complex model. The LQCD is the most complex model among the four and requires sophisticated algorithms for its solution.

9.1 Methodology

When deciding which model to use for a particular research question, it is important to consider the available computational resources and the level of complexity required for the research. The $2d$ Laplacian is a good starting point for simple research questions with limited computational resources, while the gauge Laplacian, Schwinger model, and LQCD may be more appropriate for more complex

research questions that require more computational power. Each of these models has its own advantages and disadvantages and the choice of model will depend on the specific research question and desired level of accuracy.

We always comapre the performance of an MG-MLMC algorithms to the plain Hutchinson method, Algorithm 3, and to exactly deflated Hutchinson as described in Algorithm 4.

In this study, we conduct numerical experiments on a single thread of a node with 44 cores Intel Xeon Processor E5-2699 v4 @ 2.20GHz. We are using MATLAB R2021a for the 2-dimensional Laplacian and LQCD, and using Python for the gauge Laplacian and the Schwinger model. Our goal is to achieve a relative accuracy of $\epsilon = 10^{-3}$ in examples 9.2, 9.3 and 9.4 and $\epsilon = 5 \times 10^{-5}$ in example 9.5.

To assess the performance of the algorithms we use the simple cost model presented in section 8.1 which counts the arithmetic operations in all occurring matrix-vector multiplications, i.e. in the projections, the restrictions and prolongations and in the smoothing iteration in the multigrid solver. This arithmetic cost is proportional to the number of non-zeros in the respective matrix.

For each stochastic sample for (7.53) we have to solve linear systems with the matrices D_ℓ and $D_{\ell+1}$. This is done using a multigrid method based on the same prolongations P_ℓ, restrictions R_ℓ and coarse grid operators D_ℓ that we use to obtain our multilevel decomposition 7.53. However, when multigrid is used as a solver, we use the full hierarchy going down to coarse grids of very small sizes, whereas in the multilevel decomposition (7.53) used in multilevel Monte-Carlo we might stop at an earlier level.

In the first two examples, the simplest MG-MLMC (uniform MG-MLMC) Alg. 9 is sufficient to achieve the required accuracy with less cost when compared to deflated Hutchinson [41]. As a result, we will need to conduct additional experiments utilizing another more sophisticated algorithms such as MG-MLMC++ Alg. 13 in those circumstances [63].

9.2 Two-dimensional Laplace

The $2d$ Laplacian is a mathematical operator that describes the distribution of a scalar field over a two-dimensional surface. The discrete $2d$ Laplacian is a matrix representation of this operator for a specific case, where the field is approximated on an equidistant grid in the unit square with Dirichlet boundary conditions. The

matrix, denoted as L^N, is an $N^2 \times N^2$ matrix, with I being the $N \times N$ identity matrix and defined as

$$L^N = B \otimes I + I \otimes B, \;\; B = \begin{bmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{bmatrix} \in \mathbb{C}^{N \times N}$$

The eigenvalues of L^N are explicitly known, which allows for the calculation of the trace of the inverse, which is directly available as equal to the sum of the inverses of the eigenvalues.

For the multigrid hierarchy, the standard bilinear interpolation from a grid of size $N_{\ell+1} \times N_{\ell+1}$ to one of size $N_\ell \times N_\ell$ is chosen as P_ℓ, as described in [99], where N_ℓ is the size of the matrix on level ℓ of a multigrid hierarchy. The number of grid points in one dimension on level ℓ is recursively defined as $N_{\ell+1} = \lfloor \frac{N_\ell}{2} \rfloor$. The coarse grid operators L_ℓ^N are obtained as Galerkin approximations $L_{\ell+1}^N = R_\ell L_\ell^N P_\ell$, and the restrictions P_ℓ are taken as the adjoints of the interpolations P_ℓ. Since $n_\ell = N_\ell^2$, the operator L_ℓ^N at level ℓ is a $n_\ell \times n_\ell$ matrix.

A V-cycle multigrid algorithm with one step of Gauss-Seidel pre- and post-smoothing is used as the solver. The solver starts at the finest level of the hierarchy, performs a series of operations on the various levels, and then reaches the coarsest level of the hierarchy, where the size of the matrix is $N^2 = 7^2$. At this level, the matrix is inverted directly using a Cholesky factorization. For multilevel Monte-Carlo, the maximum number of levels, L, is chosen such that $N_L = 15$. This choice is made because the cost for the direct computation of the trace at level L, which requires the inversion of a matrix of size $15^2 = 225$, is small enough compared to the other costs. The parameters and quantities for this example are summarized in Table 9.1, which includes the matrix size (N), the number of levels (L), the number of eigenvalues deflated (n_{defl}), and the number of nodes (n_ℓ) and non-zero elements ($\text{nnz}(L_\ell^N)$) in the matrix at each level. The sizes of the matrices in this example range from $63^2 \times 63^2 = 3969 \times 3969$ to $511^2 \times 511^2 = 261.121 \times 261.121$. The data in the table reflects that as the matrix size increases, the number of levels and the number of eigenvalues deflated also increases, while the number of nodes and non-zero elements in the matrix decreases at each level.

In Figure 9.1, results are presented for a study that aimed to determine the most time-efficient number of eigenvalues to use in the deflated Hutchinson method for a specific problem with $N = 127$ and a target relative accuracy of $\epsilon = 10^{-3}$. The study found that as the number of deflated eigenpairs is increased, the number of stochastic samples required to reach the target accuracy decreases. The left side

2d Laplace									
N		$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$	$\ell = 6$	n_{defl}	L
63	n_ℓ	63^2	31^2	15^2				92	3
	$\text{nnz}(L_\ell^N)$	1.96e4	8.28e3	1.85e3					
127	n_ℓ	127^2	63^2	31^2	15^2			44	4
	$\text{nnz}(L_\ell^N)$	8.01e4	3.50e4	8.28e3	1.85e3				
255	n_ℓ	255^2	127^2	63^2	31^2	15^2		64	5
	$\text{nnz}(L_\ell^N)$	3.24e5	1.44e5	3.50e4	8.28e3	1.85e3			
511	n_ℓ	511^2	255^2	127^2	63^2	31^2	15^2	76	6
	$\text{nnz}(L_\ell^N)$	1.30e6	5.82e5	1.44e5	3.50e4	8.28e3	1.85e3		

Table 9.1: Parameters and quantities for Example 9.2

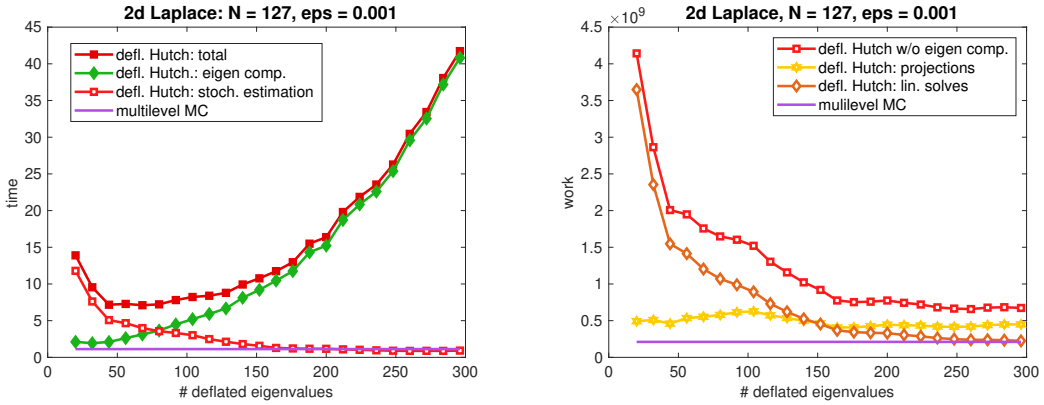


Figure 9.1: 2d Laplace: Comparison of execution times and arithmetic cost for multilevel Monte-Carlo and deflated Hutchinson, varying n_{defl}.

of the figure shows that this results in a significant reduction in execution time up to around 50 deflated eigenpairs. Beyond that point, the cost for computing the eigenpairs becomes increasingly larger, eventually becoming the main factor determining the overall execution time. The bottom horizontal line in Figure 9.1 represents the time required when using the multilevel Monte-Carlo approach with $L = 4$ levels. Even when using the optimal number of deflated eigenvalues, the deflated Hutchinson method takes around 7 times longer than the multilevel Monte-Carlo approach. The right side of the figure shows the arithmetic cost, which does not include the cost for the computation of the eigenpairs, as the details of the specific function used (Matlab's `eigs`) `are not included`. The top line in the right side of the figure corresponds to the line marked with open squares on the left side, and it can be seen that the cost model matches the observed execution times quite accurately.

In Figure 9.1, the cost of the computations is shown for the deflated Hutchinson method and the multilevel Monte-Carlo approach. The cost for the computation

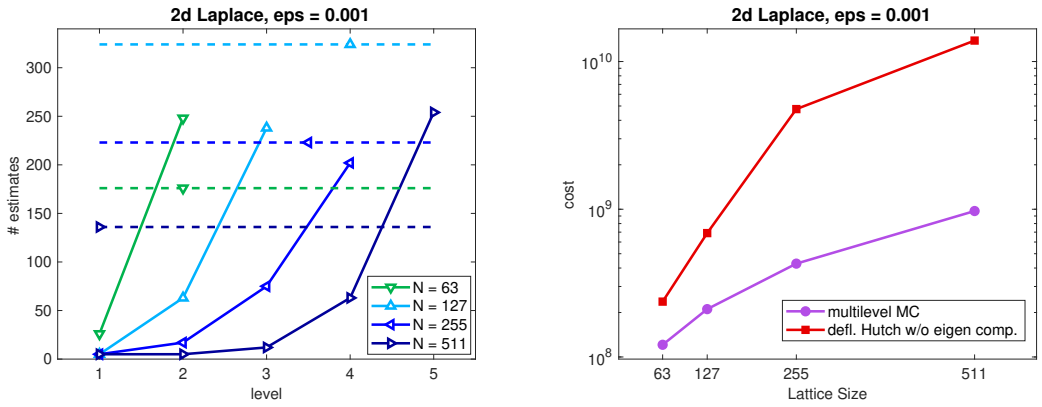


Figure 9.2: Comparison of multilevel Monte-Carlo and optimally deflated Hutchinson for the 2d Laplace matrix: no of stochastic samples on each level difference (7.53) and total cost for different N.

of projections, which is the cost of calculating $(I - UU^*)x = x - U(U^*x)$ where U is the matrix of eigenvectors and n_{defl} is the number of deflated eigenpairs, is shown as one of the lines below the top line, and it is equal to $2nn_{\text{defl}}$. The cost of linear solves, which is the cost of solving the linear systems, is shown as the other line below the top line. The horizontal line represents the cost for multilevel Monte-Carlo approach. It can be observed that when a large number of eigenpairs are deflated, the cost of linear solves in the deflated Hutchinson method becomes similar to that of the multilevel Monte-Carlo approach. This is because both methods require a similar number of stochastic samples to achieve a certain level of accuracy in the solution. However, the cost of computation for projection is still an overhead for deflated Hutchinson.

However, each sample has become significantly more expensive as a result of the additional projections, so we still see a factor of about 3 in favor of multilevel Monte-Carlo. We emphasize that this comparison does not account for the cost of computing the eigenpairs in deflated Hutchinson. None of the timings account for the time spent configuring the multigrid method. Because we need an efficient solver in both approaches, we must set up the multigrid method in both deflated Hutchinson and multilevel Monte-Carlo. prolongations, restrictions, and coarse grid operators for the discrete $2d$ Laplace operator are explicitly known and do not need to be computed, so the multigrid setup time is completely negligible. The adaptive algebraic multigrid construction used in Examples 9.3 and 9.4 below is only marginally different, as the time for the multigrid setup is only on the order of 10 times the time for one subsequent multigrid solve.

In Figure 9.2, an evaluation of the performance of the multilevel Monte-Carlo and deflated Hutchinson method is presented for four different size parameters, $N = 63, 127, 255, 511$. The left portion of the figure depicts the number of stochas-

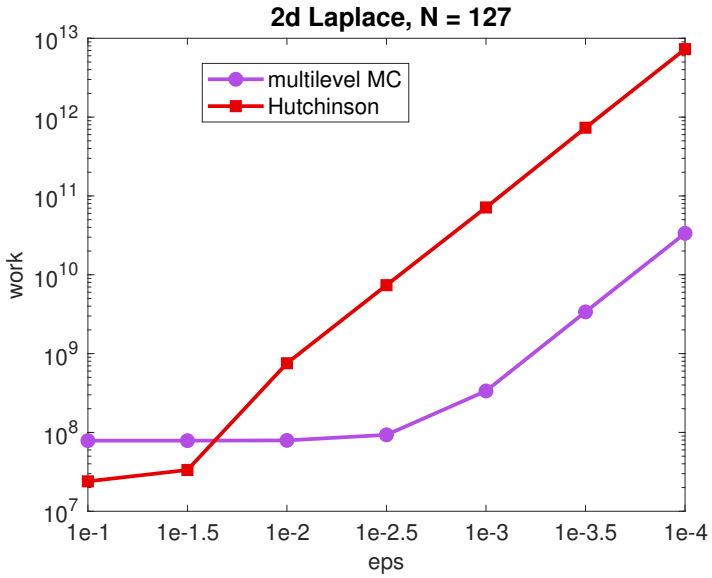


Figure 9.3: Arithmetic cost for multilevel Monte-Carlo and standard Hutchinson as a function of the target accuracy ϵ.

tic samples required by the multilevel Monte-Carlo approach for each of the differences $\hat{P}\ell(L^N\ell)^{-1}\hat{R}\ell - \hat{P}\ell + 1(L^N_{\ell+1})^{-1}\hat{R}\ell + 1$ as stated in equation (7.53). For the purpose of comparison, the number of stochastic samples required by the deflated Hutchinson method, which has been optimized for time, is represented by dashed horizontal lines. The specific values of n_{defl} used in the deflated Hutchinson method can be found in Table 9.1. The right portion of Figure 9.2 illustrates the total cost of the different methods, where the cost associated with the computation of eigenpairs in the deflated Hutchinson method is not taken into account. The figure demonstrates that in the multilevel Monte-Carlo approach, a smaller number of samples are required on the fine and expensive levels and a higher number of samples on the coarse and less expensive levels. The plots on the right side of the figure illustrate the significant reduction in arithmetic cost achieved by the multilevel Monte-Carlo approach, with a difference of approximately one order of magnitude for larger values of N, even without accounting for the cost of computing eigenpairs in the deflated Hutchinson method.

In Figure 9.3, the performance of our multilevel Monte-Carlo approach is evaluated for a specific size parameter, $N = 127$, by examining the relationship between target accuracy, ϵ, and the number of stochastic samples required. The results indicate that the multilevel Monte-Carlo approach exhibits a quadratic scaling with respect to ϵ, which is similar to that of the standard Hutchinson method. This can be observed in the logscale plot, where it is clear that as the target accuracy decreases, the number of samples required increases at a rate that is close to quadratic.

This quadratic scaling can be attributed to the fact that our multilevel Monte-Carlo approach performs a Monte-Carlo sample for each level difference, and as the accuracy requirement becomes stricter, more samples are needed to achieve that accuracy. However, it's worth noting that for larger values of ϵ, the results are less significant as a minimum of at least 5 stochastic samples is always performed for each level difference. This means that the cost for multilevel Monte-Carlo is identical and probably unnecessarily high for $\epsilon = 10^{-1}$ and $\epsilon = 10^{-1.5}$ for all level differences.

Furthermore, it is also worth noting that the arithmetic cost of the standard Hutchinson method is between 2 and 3 orders of magnitude higher than that of the multilevel Monte-Carlo approach for $\epsilon \leq 10^{-2}$. This highlights the efficiency of the multilevel Monte-Carlo approach in comparison to the standard Hutchinson method, especially when a high level of accuracy is required.

9.3 Gauge Laplace

In this example we are studying the gauge Laplacian, denoted as G^N, which is a variation of the standard 2-dimensional discrete Laplace matrix, where the coupling coefficients, also known as gauge links, are complex numbers of modulus one but with a random phase. The gauge Laplacian represents a first step towards the modeling of gauge field theories in physics, where the random coefficients model the fluctuating background gauge field. In this context, the variables u_{ij} represent the values at a specific grid point (ih, jh), where $h = 1/N$ and $i, j = 0, \ldots, N-1$. The coupling between the variables is described by one row of the gauge Laplacian $G^N \in \mathbb{C}^{N^2 \times N^2}$, which is defined as follows:

$$4u_{ij} - e^{\imath\Theta_{ij}} u_{i+1,j} - e^{\imath\Phi_{ij}} u_{i,j+1} - e^{-\imath\Theta_{i-1,j}} u_{i-1,j} - e^{-\imath\Phi_{i,j-1}} u_{i,j-1},$$
$$i, j = 0, \ldots, N-1,$$

where the indices are understood to be $\bmod N$ since periodic boundary conditions are assumed. It is important to note that gauge Laplacians are Hermitian and positive semidefinite, and in most cases, they are even positive definite.

Table 9.2 contains parameters and quantities used in an example related to the gauge Laplacian. The table contains four columns for the size parameter N, the number of levels ℓ, the number of unknowns n_ℓ at each level, and the number of non-zero entries $\mathrm{nnz}(G_\ell^N)$ of the gauge Laplacian at each level. The table also contains two additional columns for the number of deflated eigenpairs n_{defl} and the number of levels used in the multilevel Monte-Carlo method L. For each size parameter, the number of unknowns and non-zero entries decreases as the level

increases. This is due to the coarsening of the grid and the reduction of the number of degrees of freedom at each level. The number of deflated eigenpairs and the number of levels used in the multilevel Monte-Carlo method are constant across all size parameters. The example demonstrates the effectiveness of the multilevel Monte-Carlo method in reducing the computational cost of solving linear systems with the gauge Laplacian.

Gauge Laplace							
N		$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	n_{defl}	L
64	n_ℓ	4096	1354	134		60	3
	$\text{nnz}(G_\ell^N)$	20480	24900	3172			
128	n_ℓ	16384	5440	554		60	3
	$\text{nnz}(G_\ell^N)$	81920	99448	11300			
256	n_ℓ	65536	21802	2348	196	20	4
	$\text{nnz}(G_\ell^N)$	327680	394628	49416	6352		
512	n_ℓ	262144	87296	9562	924	20	4
	$\text{nnz}(G_\ell^N)$	327680	394628	49416	6352		

Table 9.2: Parameters and quantities for Example 9.3

The python pyAMG package, as stated in the reference [75], provides a set of functions to generate gauge Laplacians with specific distributions for the phases Θ_{ij} and Φ_{ij} of the gauge links. These gauge links are complex numbers with modulus 1, which are used to model the fluctuating background gauge fields in physics. In our experiments, we created gauge Laplacians with a grid spacing of $a = 1$ and a temperature of $\beta = 0.009$ for the distribution of the gauge link phases.

The pyAMG package also offers a range of algebraic multigrid methods, which we utilized to construct a multigrid hierarchy for gauge Laplacians. Specifically, we employed the adaptive smoothed aggregation method, in which for the adaptive setup we took the parameters to

$$\texttt{num\_candidates} = 2,\ \texttt{candidate\_iters} = 5,\ \texttt{improvement\_iters} = 8.$$

In order to solve linear systems, we employed the V-cycle multigrid method with one step of Gauss-Seidel pre- and post-smoothing. Table 9.2 presents a summary of important quantities for four different Lattice sizes. The table illustrates that the smoothed aggregation method results in matrices at level 2 that are smaller by a factor of 3, however, they are more dense as they contain more non-zero elements than matrices at level 1. Additionally, it can be observed that for the following levels, the coarsening is quite aggressive as it reduces the sizes by a factor of approximately 10 and similarly for the number of non-zeros.

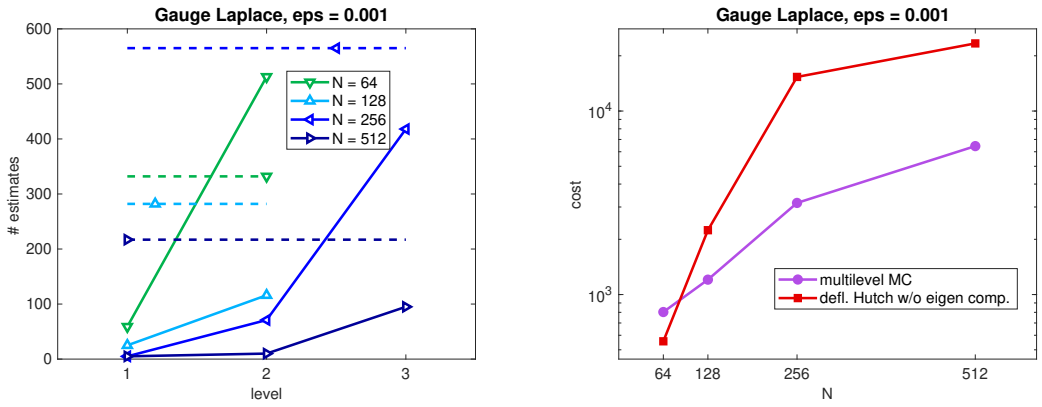


Figure 9.4: Comparison of multilevel Monte-Carlo and optimally deflated Hutchinson for the gauge Laplace matrices: no of stochastic samples on each level difference (7.53) and total cost for different N.

In Figure 9.4, similar to Figure 9.2 for the 2d Laplacian, the performance of the multilevel Monte-Carlo and optimally deflated Hutchinson methods are evaluated for the gauge Laplacian. The figure illustrates the number of samples required at different levels and the arithmetic cost for each method. The results show that the multilevel Monte-Carlo approach allows for fewer samples to be taken on fine levels, resulting in a cost reduction of up to a factor of 5 for larger values of N, such as $N = 256$ and $N = 512$. However, for smaller values of N, such as $N = 64$, the multilevel Monte-Carlo approach may not be as effective as the deflated Hutchinson method. It is also important to note that the arithmetic cost for computing the eigenpairs in deflated Hutchinson is not included in this comparison.

9.4 Schwinger Model

The third example we discuss the Schwinger discretization of the 2-dimensional Dirac operator, as introduced by Schwinger in 1962 [91]. This operator is used to model quantum electrodynamics (QED), which is the quantum field theory of the electromagnetic interaction between charged particles through photons. The Schwinger matrix is similar to the gauge Laplacian in that it has a periodic nearest-neighbor coupling on an equidistant grid in the unit square and the coupling coefficients are based on complex numbers with a modulus of 1 and a random phase. The difference is that for Dirac operators there are several (here: two) variables per grid point, representing different spins. With the Pauli matrices

$$\sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma_2 = \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix},$$

and the understanding that a grid variable u_{ij} has now two components, i.e. $u_{ij} \in \mathbb{C}^2$, representing the different spins at grid point (ih, jh), the periodic coupling is now given as

$$\begin{aligned}(4+m)\cdot u_{ij} &- e^{\imath\Theta_{ij}}(I-\sigma_1)u_{i+1,j} - e^{\imath\Phi_{ij}}(I-\sigma_2)u_{i,j+1} && (9.1)\\ &- e^{-\imath\Theta_{i-1,j}}(I+\sigma_1)u_{i-1,j} - e^{-\imath\Phi_{i,j-1}}(I+\sigma_2)u_{i,j-1}, && (9.2)\\ & i,j = 1,\ldots,N. && (9.3)\end{aligned}$$

Note that the Pauli matrices cross-couple the spins. Thus, if we order spin components such that the first spin componentat each grid location comes first and all second spin components follow, the Schwinger matrix has the form

$$S^N = \begin{bmatrix} G^N & B \\ -B^* & G^N \end{bmatrix} \tag{9.4}$$

where the matrices G^N are gauge Laplacians and the matrix B represents the spin cross-coupling.

The Schwinger matrix used in this study is derived from a thermalized configuration within a Markov process, which ensures that the random gauge links follow a Boltzmann distribution with a specified temperature parameter. The matrix has a size of $2N^2 \times 2N^2$, where $N = 128$, resulting in a matrix of dimensions $32,768 \times 32,768$. To construct the multigrid hierarchy for this matrix, we employed an aggregation-based approach similar to the one used for the 4-dimensional Wilson-Dirac operator; see [7, 40]. This approach involves decreasing the size of the Lattice at each level and representing the operator as a periodic nearest neighbor coupling.

Each Lattice site has several degrees of freedom (dofs) represented by variables of length d. To proceed to the next level, the Lattice is subdivided into smaller subLattices, called aggregates, with each aggregate becoming a single Lattice site on the next level. The restriction operator is approximated by computing the d smallest eigenvectors and assembling them over the aggregates and orthogonalizing the components. This results in restriction operators that are orthonormal, and since we take the prolongations to be their adjoints, we are in a simplified situation for estimating the traces of the differences in multilevel Monte-Carlo.

The Schwinger matrix is not Hermitian, but its eigenvalues come in complex conjugate pairs. This is due to a non-trivial symmetry induced by the spin structure that can be seen from (9.4),

$$JS^N = \left(S^N\right)^* J, \quad \text{where } J = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix},$$

Schwinger model					
L		$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$
4	n_ℓ	$2 \cdot 128^2$	$4 \cdot 32^2$	$8 \cdot 8^2$	$8 \cdot 2^2$
	$\mathrm{nnz}(D_\ell)$	2.94e5	1.64e5	2.46e4	1024
	m_1	m_2	m_3	m_4	m_5
mass value	-0.1320	-0.1325	-0.1329	-0.1332	-0.1333
defl. vects.	384	384	512	512	512

Table 9.3: Parameters used in the Schwinger model and number of deflated eigenvectors chosen in exactly deflated Hutchinson. $\mathrm{nnz}(D_\ell)$ denotes the number of non-zero elements in D_ℓ

so that to each right eigenpair (x, λ) of S^N corresponds a left eigenpair $((Jx)^*, \bar{\lambda})$. This spin symmetry can be preserved on the coarser levels if one doubles the dofs; see [7, 40].

We built a multigrid hierarchy with four levels. For the aggregates, at all levels we always aggregated 4×4 subLattices into one Lattice site on the next level, and we started with 2 dofs for the second level and 4 for all remaining levels. Those dofs are then doubled because we implemented the spin structure preserving approach. Table 9.3 summarizes the most important information on the multigrid hierarchy. It also shows the five different (negative) values for the mass m that we used in our experiments. These values are physically meaningful, and for all of them the spectrum of S^N is contained in the right half plane. As m becomes smaller, S^N becomes more ill-conditioned, so the cost for each stochastic sample increases. When solving linear systems at the various levels, we used one V-cycle of multigrid with two steps of Gauss-Seidel pre- and post-smoothing as a preconditioner for flexible GMRES [87].

Our implementation was done in Python[1,2] with the same configuration and parameters as in [41]. In particular, we use 5 different (negative) masses m to shift the mass-less Schwinger operator by the respective multiple of the identity, thus yielding operators with increasing condition number. The multigrid hierarchy was constructed with a bootstrap setup and aggregation based prolongations as in DDαAMG [40]. Properties of the matrices at the various levels are summarized in the top part of Table 9.3.

We use a deflated Hutchinson method as our reference for comparison. We did not use non-deflated Hutchinson, because its performance is by two orders of magnitude worse than that of deflated Hutchinson. For deflation, we used the k

[1]The MG-MLMC program can be found in the GitHub repository https://github.com/Gustavroot/MLMCTraceComputer

[2]The MG-MLMC++ programs can be found in the GitHub repository https://github.com/khlmtf/MGMLMCpp

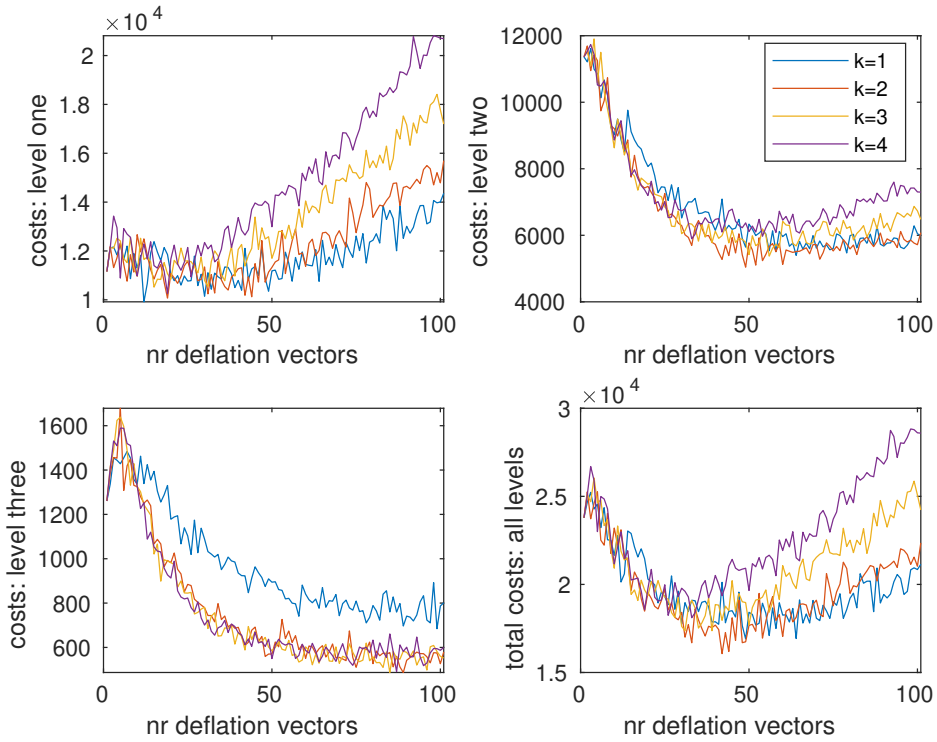


Figure 9.5: Work at each level difference as a function of the number of vectors in the block power iteration and total work when taking the same number on all levels.

smallest eigenmodes that we precomputed, and then optimized k so as to obtain the smallest overall cost, *excluding* the cost for the eigenvector computation. So the work for deflated Hutchinson is actually higher than what we report.

Figure 9.6 reports the arithmetic cost in MFLOPS for five different methods: Deflated Hutchinson for reference, MG-MLMC with uniform target variances on the difference levels and its modification working with optimal target variances, and then the corresponding two versions for MG-MLMC++. Here, we determined the number k of steps of the block power iteration and the number d_ℓ of vectors to be used there by a parameter scan on each level. This scan is reported in Fig. 9.5. We find that $k = 2$ is a better choice than $k = 1$, and that increasing k further does not result in significant further gains. Also, $d_\ell \approx 50$ appears as a good choice on all level differences.

The plot in Fig. 9.6 shows that for all masses considered, the best MG-MLMC method now outperforms deflated Hutchinson (with an optimal number of deflated vectors and without counting the work for computing those). It also shows that with optimal numbers of vectors in the block power iteration, the "++"-enhancement improves MG-MLMC by a factor of 1.5 to 3, with a stronger improvement for the smaller values of m, i.e. the more ill-conditioned matrices. The

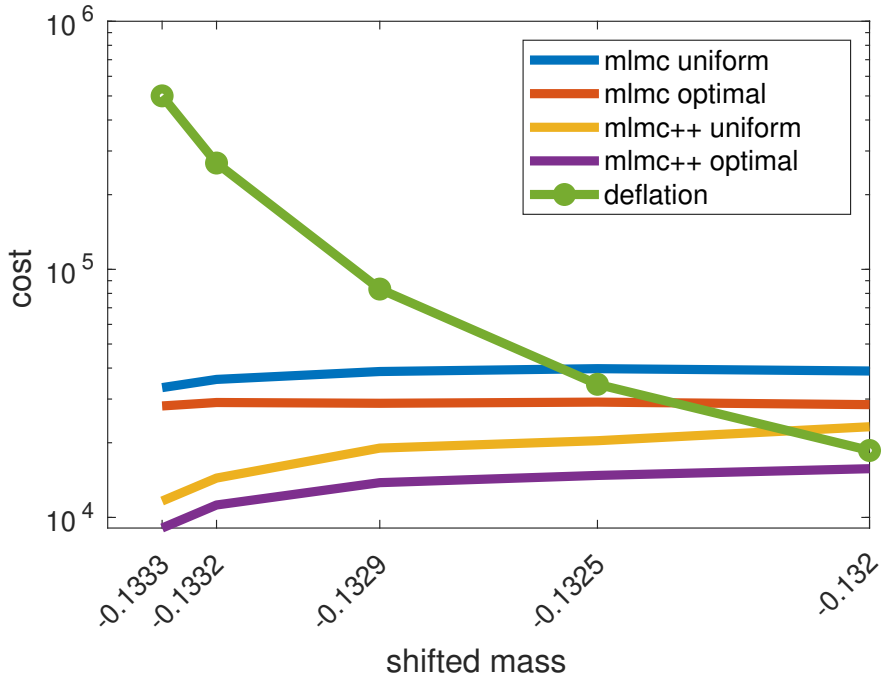


Figure 9.6: MG-MLMCM, MG-MLMC++ and deflated Hutchinson for the Schwinger matrix: total cost for different masses with uniform and the optimized target variances.

influence of the strategy to determine the target variance ("uniform" or "optimized") is, on the other hand, not very significant.

As a supplementary information, Tab. 9.4 reports the number of stochastic samples that were carried out on the different level differences. These numbers directly illustrate the variance reductions achieved in the different approaches. Each stochastic sample involves the solution of two linear systems (with matrices D_ℓ and $D_{\ell+1}$). These are done via multigrid and are thus quite efficient. This is why the numbers of stochastic samples do not reflect the total arithmetic cost of the methods, in which, in particular, performing the projections has a high cost when the deflating subspace becomes larger. Interestingly, there is no visible dependence on the mass parameter for the MG-MLMC approaches as was already observed in [41].

The table presents the results of a numerical experiment comparing different sampling methods for a particular problem. The table shows the number of samples used for each method at different levels of the multigrid hierarchy. The results are reported in three different rows, labeled as "samples/$m_i, i = 1, \cdots, 5$", representing different measurement time.

The deflated hutchinson method uses a constant number of samples across all levels, while the other methods use different numbers of samples at different levels.

method	estimates per mass					
	-0.1320	-0.1325	-0.1329	-0.1333	-0.1333	level
deflation	529	1004	2318	7431	13845	
ML opt.	325	321	315	313	306	$\ell = 1$
	854	873	837	833	791	$\ell = 2$
	4208	4218	4414	4287	4171	$\ell = 3$
ML++ opt.	181	158	143	108	177	$\ell = 1$
	221	221	200	148	111	$\ell = 2$
	278	272	243	162	173	$\ell = 3$

Table 9.4: Number of stochastic samples for different masses at each level ℓ for deflated Hutchinson, MG-MLMC and MG-MLMC++, both with optimized target variances

The "MG-MLMC uniform" method uses a fixed number of samples at each level, while the "MG-MLMC optimal" method uses a variable number of samples at each level in order to achieve a desired level of accuracy. The "MG-MLMC++ uniform" and "MG-MLMc++ optimal" methods are variations of the "MG-MLMC" methods, with the "MG-MLMC++ optimal" method using the fewest number of samples at each level.

The results show that the "MG-MLMC optimal" and "MG-MLMC++ optimal" methods are more efficient than the other methods, using fewer samples at each level to achieve the same level of accuracy. The "MG-MLMC optimal" and "MG-MLMC++ optimal" method also have the lowest sample numbers at each level, indicating that they are more efficient. This can be attributed to their ability to adapt the number of samples to the specific requirements of each level.

9.5 Lattice QCD

In this example, we applied our optimal multigrid multilevel Monte-Carlo approaches as in Alg. 12 and Alg. 13 to compute the trace of the inverse of the four dimentional Dirac operator in Lattice QCD, specifically using the clover-improved Wilson-Dirac discretization. We tested our methods on two Lattices of size $L = 16^4$ with matrix dimension $786,432 \times 786,432$ and $L = 32^4$ with matrix dimension $12,582,912 \times 12,582,912$.

In Lattice QCD simulations, the spacetime is discretized into a four-dimensional grid known as the Lattice configuration. The Wilson-Dirac matrix is defined as $\mathcal{D}_W = mI - D_0$ with $m = 4 + m_0$, where D_0 is the hopping matrix constructed

from the Lattice configuration and m_0 is the quark mass. The parameter κ is often used as a simulation parameter instead of m_0.

The choice of simulation parameters will affect the spectrum of the operator, depending on the Lattice size, boundary conditions, and other parameters. For simulations setup (provided by the Lattice QCD group at Wuppertal university led by Prof. Francesco Knechtli) on the two Lattices with sizes 16^4 and 32^4, a Luescher-Weiss gauge action and three flavors of mass-degenerate Wilson-clover fermions are used with parameters $\beta = 3.55$, $\kappa = 0.137$, $m_0 = -0.350364963503650$, and $c_{sw} = 1.8248654$.

These parameter choices resulted in a Lattice spacing of $a = 0.0643$ fm and a quark mass very close to the average of the physical up, down, and strange quark masses, yielding a resulting pion mass of around 420 MeV. We utilized a Matlab version of the DDαAMG package [3] with parameters as shown in Tab. 9.5 and using v-cycle to solve linear systems at different levels in the multilevel hierarchy for our MG-MLMC and the Hutchinson methods. We used three multigrid levels with aggregation blocks of size 4^4 from the finest level to the first coarse one, and of size 2^4 from the first coarse level to the coarsest.

Parameter	Value
test vectors	20
test vectors type	EVs
setup tol	10^{-5}
smoother	GMRES
GMRES iters	8×3
presmoothing	0
postsmoothing	1
mlmc solver tol	10^{-7}
coarse tol	10^{-1}
levels	3

Table 9.5: Parameters of DDαAMG

In contrast to previous three examples, we are using the plain Hutchinson method as reference in this case, as we have found in practice that there are two main drawbacks of using the deflated Hutchinson method for comparison. Firstly, determining the optimal number of deflated vectors for each matrix requires scanning the expected range of vectors, which can be challenging for very large matrices. Secondly, there is no solid cost model for computing the choosen optimal eigenvectors of the matrix. In the Schwinger model case, this work was neglected, but for Lattice QCD, it is unfair to ignore it, as it can constitute the majority of the total cost for certain matrix sizes. In addition, our optimal MG-MLMC++

[3] https://github.com/mrottmann/DDalphaAMG

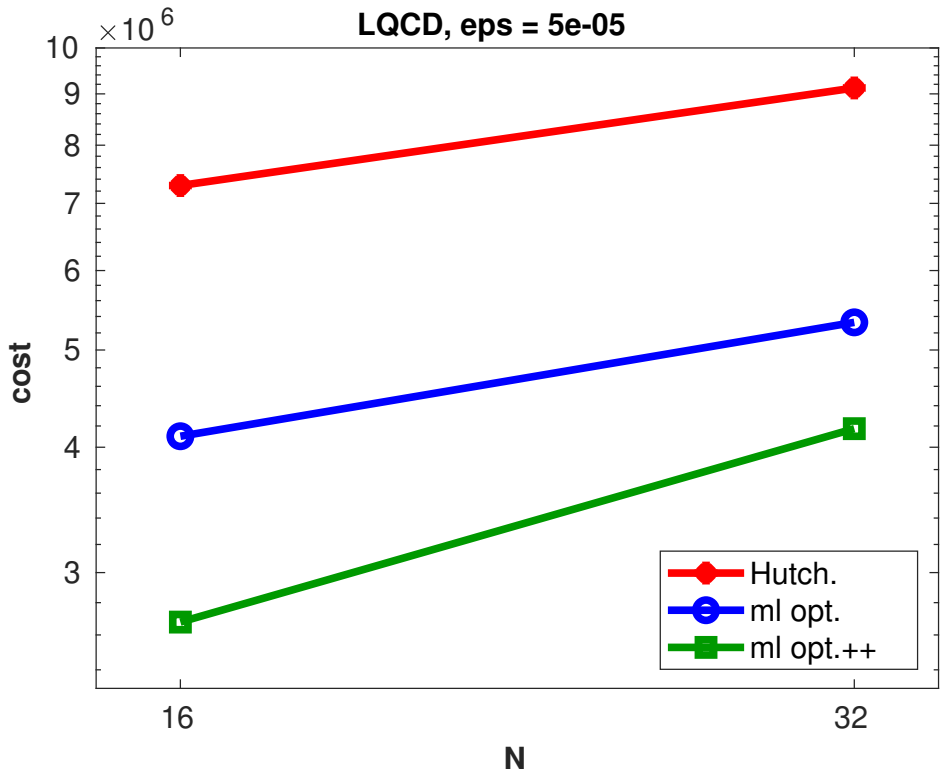


Figure 9.7: MG-MLMC, MG-MLMC++ and plain Hutchinson for the Lattice QCD: total cost for different latice sizes 16^4 and 32^4 with the optimized target variances and relative accuracy $\epsilon = 5e^{-5}$.

approach has been found to outperform deflation method in example 9.4, that we found as a suitable example to compare all MG-MLMC approaches with deflation method.

Typically, we compute the trace of the coarsest level directly. However, for very large matrices (like the current example), it becomes computationally expensive to compute the trace of the inverse directly since the size of the coarsest matrix remains very large. To overcome this limitation, we resort to stochastically computing the trace of the coarsest level using plain Hutchinson. In practice, we observed how this approach enhances the cost model in general for very large matrices with high target accuracy.

The results in Fig. 9.7 report the total arithmetic cost in MFLOPS of three methods: Hutchinson, MG-MLMC and MG-MLMC++ with optimal target variances on the difference levels for the multilevel methods. The plot shows that the multigrid multilevel methods, particularly MG-MLMC++, outperform Hutchinson method and significanly reduce the total cost for Lattice size $L = 16^4$ by factor of 3 to 4 and for $L = 32^4$ by factor to 2 to 3. In addition, the MG-MLMC++ enhancement improves MG-MLMC by a factor of 1.5 to 3.

Figure 9.8 presents a comparison of the sample size obtained using plain Hutchinson, MG-MLMC and MG-MLMC++ methods on the difference levels of two Lattices. The results show that, for both Lattices, the sample size at the first difference level (the most expensive level) of the multilevel methods is consistently lower than that of the plain Hutchinson method. Additionally, even at the second difference level, MG-MLMC++ can significantly reduce the samples and outperform Hutchinson.

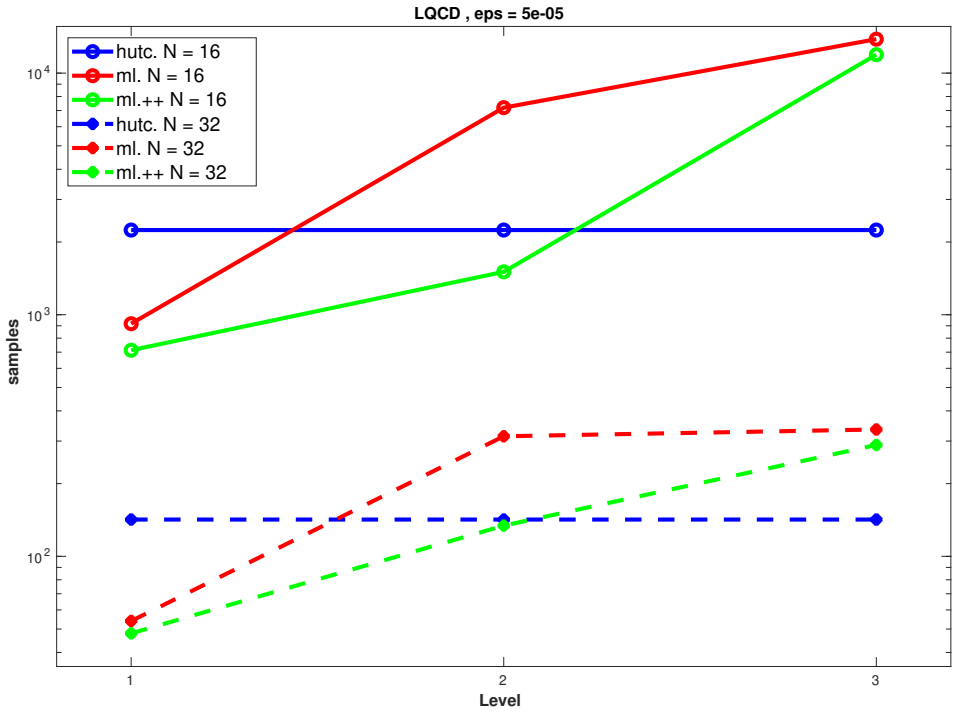


Figure 9.8: MG-MLMC, MG-MLMC++, and Plain Hutchinson for the Lattice QCD: the sample size for different latice sizes 16^4 and 32^4 with the optimized target variances and relative accuracy $\epsilon = 5e^{-5}$.

Additional information is provided in Tab. 9.6, which lists the number of stochastic samples that were run on the various level differences.

9.6 Conclusion and Outlook

The goal of this thesis is to enhance the accuracy of estimating the trace and minimizing computational costs using variance reduction techniques. Two novel stochastic approaches, MG-MLMC and MG-MLMC++, are proposed for achieving this objective by utilizing the multilevel hierarchy in the multigrid solves for

method type	samples nr. per lattice size		
	16^4	32^4	level
Plain Hutchinson	2241	142	
MG-MLMC, optimal	918	54	$\ell = 1$
	7,195	314	$\ell = 2$
	13,820	335	$\ell = 3$
MG-MLMC++, optimal	713	48	$\ell = 1$
	1,505	134	$\ell = 2$
	11,921	289	$\ell = 3$

Table 9.6: Number of stochastic samples for Lattice sizes $L = 16^4$ and $L = 32^4$ for Hutchinson, MG-MLMC and MG-MLMC++ at each level ℓ, both with optimized target variances

linear systems and also for the trace estimation. These methods can be applied in any problem that requires the computation of the trace of the inverse of a matrix.

The numerical results presented in sections 9.2, 9.3 and 9.4 are notable and have encouraged us to conduct additional experiments for Lattice QCD as in 9.5, however we have not compared the performance gains in 9.5 to those achieved by the deflated Hutchinson method, we have observed significant improvements when using our multilevel approach to compute Lattice QCD matrices. These results suggest that our mulilevel approach has great potential for improving the efficiency of Lattice QCD simulations. Moreover, we have not yet applied the Adaptive MG-MLMC++ on the Lattice QCD case and still need to test it for both examples 9.4 and 9.5.

The results in Sec. 9.5 are promising and motivated us to extend the applicability of the methods to more complex systems and gauge theories, where the computational challenges are even greater. To overcome these challenges, we plan to implement the multilevel algorithms in parallel using a suitable programming language. This will enable us to take full advantage of modern computing architectures and significantly improve the efficiency of our methods.

Our MG-MLMC algorithms are very flexible because of the computation of each level difference, including the coarsest level in some cases as shown in 9.5, via plain Hutchinson method. This flexibility allows us to improve the algorithms further if we can utilize the hierarchical probing technique [94], in the same manner of deflation approach 7.3.4, that is used for trace estimation in Lattice QCD. This will all be investigated in upcoming work.

List of Figures

List of Tables

List of Algorithms

Bibliography

[1] R. H. Affandi, E. Fox, R. Adams, and B. Taskar, *Learning the parameters of determinantal point process kernels*, in International Conference on Machine Learning, PMLR, 2014, pp. 1224–1232.

[2] C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, K. Hadjiyiannakou, K. Jansen, G. Koutsou, and A. V. A. Casco, *Proton and neutron electromagnetic form factors from Lattice QCD*, Physical Review D, (2019).

[3] C. Alexandrou, S. Bacchio, M. Constantinou, J. Finkenrath, K. Hadjiyiannakou, K. Jansen, G. Koutsou, H. Panagopoulos, G. Spanoudes, E. T. M. Collaboration, et al., *Complete flavor decomposition of the spin and momentum fraction of the proton using lattice qcd simulations at physical pion mass*, Physical Review D, 101 (2020), p. 094513.

[4] C. Alexandrou, S. Bacchio, J. Finkenrath, A. Frommer, K. Kahl, and M. Rottmann, *Adaptive aggregation-based domain decomposition multigrid for twisted mass fermions*, Physical review letters, 94 (2016), p. 114509. arxiv hep-lat/1610.02370.

[5] C. Alexandrou, M. Constantinou, T. Korzec, and et al., *Twisted mass qcd and its applications*, The European Physical Journal A, 53 (2017), p. 44.

[6] H. Avron and S. Toledo, *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix*, J. ACM, 58 (2011).

[7] R. Babich, J. Brannick, R. C. Brower, M. A. Clark, T. A. Manteuffel, S. F. McCormick, J. C. Osborn, and C. Rebbi, *Adaptive*

multigrid algorithm for the lattice Wilson-Dirac operator, Physical review letters, 105 (2010), p. 201602.

[8] S. G. Bacchio, *Simulating maximally twisted fermions at the physical pointa with multigrid methods*, (2019).

[9] J. Baez and J. Huerta, *The algebra of grand unified theories*, Bulletin of the American Mathematical Society, 47 (2010), pp. 483–552.

[10] G. Bali, S. Collins, A. Frommer, K. Kahl, I. Kanamori, B. Müller, M. Rottmann, and J. Simeth, *(approximate) low-mode averaging with a new multigrid eigensolver*, 2015.

[11] C. Bekas, A. Curioni, and I. Fedulova, *Low cost high performance uncertainty quantification*, in Proceedings of the 2nd Workshop on High Performance Computational Finance, 2009, pp. 1–8.

[12] T. Bergrath, M. Ramalho, R. Kenway, et al., *PRACE scientific annual report 2012*, tech. rep., PRACE, 2012. `http://www.prace-ri.eu/IMG/pdf/PRACE_Scientific_Annual_Report_2012.pdf`, p. 32.

[13] J. Brannick, R. C. Brower, M. A. Clark, J. C. Osborn, and C. Rebbi, *Adaptive Multigrid Algorithm for Lattice QCD*, Physical review letters, 100:041601 (2007).

[14] M. Brezina, T. Manteuffel, S. McCormick, J. Ruge, and G. Sanders, *Towards adaptive smoothed aggregation (αSA) for nonsymmetric systems*, SIAM J. Sci. Comput., 32 (2010), pp. 14–39.

[15] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A multigrid tutorial*, SIAM, 2000.

[16] B. C. Brookes and A. N. Kolmogorov, *Foundations of the theory of probability*, The Mathematical Gazette, 35 (1951).

[17] Z. Bujanovic and D. Kressner, *Norm and trace estimation with random rank-one vectors*, SIAM Journal on Matrix Analysis and Applications, 42 (2021), pp. 202–223.

[18] J. R. Bunch and B. N. Parlett, *Direct methods for solving symmetric indefinite systems of linear equations*, SIAM Journal on Numerical Analysis, 8 (1971), pp. 639–655.

[19] J. Campbell, J. Huston, and F. Krauss, *The black book of quantum chromodynamics: a primer for the LHC era*, Oxford University Press, 2018.

[20] A. Chapman and Y. Saad, *Deflated and augmented krylov subspace techniques*, Numerical linear algebra with applications, 4 (1997), pp. 43–66.

[21] A. Cortinovis and D. Kressner, *On randomized trace estimates for indefinite matrices with an application to determinants*, Foundations of Computational Mathematics, 22 (2022), pp. 875–903.

[22] A. Cortinovis and D. Kressner, *On randomized trace estimates for indefinite matrices with an application to determinants*, Foundations of Computational Mathematics, 22 (2022), pp. 875–903.

[23] Z. Davoudi, W. Detmold, P. Shanahan, K. Orginos, A. Parreno, M. J. Savage, and M. L. Wagman, *Nuclear matrix elements from lattice qcd for electroweak and beyond-standard-model processes*, Physics Reports, 900 (2021), pp. 1–74.

[24] J. A. De La Peña, I. Gutman, and J. Rada, *Estimating the estrada index*, Linear Algebra and its Applications, 427 (2007), pp. 70–76.

[25] E. De Sturler, *Truncation strategies for optimal krylov subspace methods*, SIAM Journal on Numerical Analysis, 36 (1999), pp. 864–889.

[26] T. A. DeGrand and P. Rossi, *Conditioning techniques for dynamical fermions*, Comput. Phys. Commun., 60 (1990), pp. 211–214.

[27] J. Demmel, *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, 1997.

[28] P. A. M. Dirac, *The quantum theory of the electron*, Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character, 117 (1928), pp. 610–624.

[29] S. Dong and K. Liu, *Stochastic estimation with Z_2 noise*, Phys. Lett. B, 328 (1994), pp. 130–136.

[30] J. Erhel, K. Burrage, and B. Pohl, *Restarted gmres preconditioned by deflation*, Journal of computational and applied mathematics, 69 (1996), pp. 303–318.

[31] Y. A. Erlangga and R. Nabben, *Deflation and balancing preconditioners for krylov subspace methods applied to nonsymmetric matrices*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 684–699.

[32] S. Fischer, A. Frommer, U. Glassner, T. Lippert, G. Ritzenhofer, and K. Schilling, *A parallel SSOR preconditioner for Lattice QCD*, Comput. Phys. Commun., 98 (1996), pp. 20–34.

[33] J. Fitzsimons, D. Granziol, K. Cutajar, M. Osborne, M. Filippone, and S. Roberts, *Entropic trace estimates for log determinants*, in Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2017, pp. 323–338.

[34] R. Frezzotti, P. A. Grassi, S. Sint, and P. Weisz, *A local formulation of lattice qcd without unphysical fermion zero modes*, Nuclear Physics B-Proceedings Supplements, 83 (2000), pp. 941–946.

[35] R. Frezzotti, S. Sint, and P. Weisz, *O (a) improved twisted mass lattice qcd*, Journal of High Energy Physics, 2001 (2001), p. 048.

[36] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *Aggregation-based multilevel methods for Lattice QCD*, arXiv preprint arXiv:1202.2462, (2012).

[37] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *An adaptive aggregation based domain decomposition multilevel method for the lattice wilson dirac operator: Multilevel results*, arXiv: High Energy Physics - Lattice, (2013).

[38] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *Adaptive aggregation-based domain decomposition multigrid for the lattice Wilson–Dirac operator*, SIAM J. Sci. Comp., 36 (2014), pp. A1581–A1608.

[39] ——, *Adaptive aggregation based domain decomposition multigrid for the lattice Wilson-Dirac operator*, SIAM J. Sci. Comp., 36 (2014), pp. A1581–A1608.

[40] A. Frommer, K. Kahl, S. Krieg, B. Leder, and M. Rottmann, *Adaptive aggregation-based domain decomposition multigrid for the lattice Wilson-Dirac operator*, SIAM journal on scientific computing, 36 (2014), pp. A1581–A1608.

[41] A. Frommer, M. N. Khalil, and G. Ramirez-Hidalgo, *A multilevel approach to variance reduction in the stochastic estimation of the trace of a matrix*, SIAM Journal on Scientific Computing, 44 (2022), pp. A2536–A2556.

[42] A. Frommer, A. Nobile, and P. Zingler, *Deflation and flexible SAP-preconditioning of GMRES in Lattice QCD simulations*, (2012). arXiv:1204.5463.

[43] A. Frommer and G. Ramirez-Hidalgo, *Deflated Multigrid Multilevel Monte Carlo*, PoS, LATTICE2022 (2022), p. 030.

[44] A. S. Gambhir, *Disconnected diagrams in Lattice QCD*, (2017).

[45] A. S. Gambhir, A. Stathopoulos, and K. Orginos, *Deflation as a method of variance reduction for estimating the trace of a matrix inverse*, SIAM J. on Sci. Comput., 39 (2017), pp. A532–A558.

[46] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, *Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration*, Advances in neural information processing systems, 31 (2018).

[47] C. Gattringer and C. Lang, *Quantum Chromodynamics on the Lattice: an introductory presentation*, vol. 788, Springer Science & Business Media, 2009.

[48] S. Ghahramani, *Fundamentals of probability: With stochastic processes, third edition*, 2015.

[49] M. B. Giles, *Multilevel Monte Carlo methods.*, Acta Numer., 24 (2015), pp. 259–328.

[50] P. Glasserman, *Monte Carlo methods in financial engineering*, vol. 53, Springer, 2004.

[51] G. H. Golub and C. F. V. Loan, *Matrix Computations*, Johns Hopkins University Press, 4th ed., 2013.

[52] W. Greiner, S. Schramm, and E. Stein, *Quantum chromodynamics*, Springer Science & Business Media, 2007.

[53] M. Guest, G. Aloisio, R. Kenway, et al., *The scientific case for HPC in Europe 2012 - 2020*, tech. rep., PRACE, October 2012. `http://www.prace-ri.eu/PRACE-The-Scientific-Case-for-HPC`, p. 75.

[54] W. Hackbusch, *Multi-Grid Methods and Applications*, vol. 4 of Springer Series in Computational Mathematics, Springer, 1985.

[55] E. Hallman and D. Troester, *A multilevel approach to stochastic trace estimation*, Linear Algebra and its Applications, 638 (2022), pp. 125–149.

[56] S. Heinrich, *Monte carlo complexity of global solution of integral equations*, Journal of complexity, 14 (1998), pp. 151–175.

[57] S. Heinrich and E. Sindambiwe, *Monte carlo complexity of parametric integration*, Journal of Complexity, 15 (1999), pp. 317–341.

[58] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[59] J. C. Hull, *Options futures and other derivatives*, Pearson Education India, 2003.

[60] M. F. Hutchinson, *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines*, Comm. Statist. Simulation Comput., 19 (1990), pp. 433–450.

[61] W. Joubert, *On the convergence behavior of the restarted gmres algorithm for solving nonsymmetric linear systems*, Numerical linear algebra with applications, 1 (1994), pp. 427–447.

[62] K. Kahl, *Adaptive algebraic multigrid for Lattice QCD computations*, PhD thesis, Universität Wuppertal, Fakultät für Mathematik und Naturwissenschaften . . . , 2018.

[63] M. Khalil and A. Frommer, *MGMLMC++ as a Variance Reduction Method for Estimating the Trace of a Matrix Inverse*, PoS, LATTICE2022 (2022), p. 017.

[64] S. A. Kharchenko and A. Yu. Yeremin, *Eigenvalue translation based preconditioners for the gmres (k) method*, Numerical linear algebra with applications, 2 (1995), pp. 51–77.

[65] D. P. Kroese, T. Brereton, T. Taimre, and Z. I. Botev, *Why the monte carlo method is so important today*, Wiley Interdisciplinary Reviews: Computational Statistics, 6 (2014), pp. 386–392.

[66] H. B. Lawson and M.-L. Michelsohn, *Spin Geometry (PMS-38), Volume 38*, Princeton university press, 2016.

[67] M. Lüscher, *Solution of the Dirac equation in Lattice QCD using a domain decomposition method*, Comput. Phys. Commun., 156 (2004), pp. 209–220.

[68] M. Lüscher, *Local coherence and deflation of the low quark modes in Lattice QCD*, JHEP, 2007 (2007), p. 081.

[69] R. Mann, *An introduction to particle physics and the standard model*, Taylor & Francis, 2010.

[70] R. A. Meyer, C. Musco, C. Musco, and D. P. Woodruff, *Hutch++: Optimal stochastic trace estimation*, in Symposium on Simplicity in Algorithms (SOSA), SIAM, 2021, pp. 142–155.

[71] A. C. Micheas, *Theory of Stochastic Objects*, 2018.

[72] R. Morgan, *GMRES with deflated restarting*, SIAM J. Sci. Comput., 24 (2002), pp. 20–37.

[73] R. B. Morgan, *A restarted gmres method augmented with eigenvectors*, SIAM Journal on Matrix Analysis and Applications, 16 (1995), pp. 1154–1171.

[74] R. A. Nicolaides, *Deflation of conjugate gradients with applications to boundary value problems*, SIAM Journal on Numerical Analysis, 24 (1987), pp. 355–365.

[75] L. N. Olson and J. B. Schroder, *PyAMG: Algebraic multigrid solvers in Python v4.0*, 2018. Release 4.0.

[76] D. Persson, A. Cortinovis, and D. Kressner, *Improved variants of the hutch++ algorithm for trace estimation*, SIAM Journal on Matrix Analysis and Applications, 43 (2022), pp. 1162–1185.

[77] M. Peskin, *An introduction to quantum field theory*, CRC press, 2018.

[78] M. E. Peskin and D. V. Schroeder, *An Introduction to Quantum Field Theory*, Westview Press, 1995.

[79] J. Rolf and S. Sint, *Twisted mass Lattice QCD*, Progress in Particle and Nuclear Physics, 103 (2018), pp. 74–107.

[80] E. Romero, A. Stathopoulos, and K. Orginos, *Multigrid deflation for lattice QCD*, J. Comput. Phys., 409 (2020), p. 109356.

[81] F. Roosta Khorasani and U. Ascher, *Improved bounds on sample size for implicit matrix trace estimators*, Foundations of Computational Mathematics, 15 (2015), pp. 1187–1212.

[82] S. M. Ross, *Simulation*, Oxford, 2022.

[83] M. Rottmann, *Adaptive Domain Decomposition Multigrid for Lattice QCD*, PhD thesis, Wuppertal U., 2016.

[84] J. Ruge and K. Stüben, *Algebraic multigrid*, in Multigrid Methods, S. F. McCormick, ed., vol. 3 of Frontiers in Applied Mathematics, SIAM, Philadelphia, PA, USA, 1987, pp. 73–130.

[85] Y. Saad, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput, 14 (1992), pp. 461–469.

[86] Y. Saad, *Analysis of augmented krylov subspace methods*, SIAM Journal on Matrix Analysis and Applications, 18 (1997), pp. 435–449.

[87] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, USA, 2nd ed., 2003.

[88] ——, *Numerical Methods for Large Eigenvalue Problems*, SIAM, Philadelphia, PA, USA, 2nd ed., 2011.

[89] Y. Saad, M. Yeung, J. Erhel, and F. Guyomarc'h, *A deflated version of the conjugate gradient algorithm*, SIAM Journal on Scientific Computing, 21 (2000), pp. 1909–1926.

[90] A. K. Saibaba, A. Alexanderian, and I. C. Ipsen, *Randomized matrix-free trace and log-determinant estimators*, Numerische Mathematik, 137 (2017), pp. 353–395.

[91] J. Schwinger, *Gauge invariance and mass II*, Phys. Rev., 128 (1962), p. 2425–2429.

[92] J. Sexton and D. Weingarten, *Systematic expansion for full QCD based on the valence approximation*, 1994.

[93] B. Sheikholeslami and R. Wohlert, *Improved continuum limit Lattice action for QCD with Wilson Fermions*, Nuclear Physics B, 259 (1985), pp. 572–596.

[94] A. Stathopoulos, J. Laeuchli, and K. Orginos, *Hierarchical probing for estimating the trace of the matrix inverse on toroidal lattices*, SIAM J. Sci. Comput., 35 (2013), pp. 299–322.

[95] A. Strebel, *Advanced Applications For Algebraic Multigrid Methods In Lattice QCD*, PhD thesis, Universität Wuppertal, Fakultät für Mathematik und Naturwissenschaften . . . , 2020.

[96] M. Thomson, *Modern Particle Physics*, Cambridge University Press, 2013.

[97] C. Thron, S. Dong, K. Liu, and H. Ying, *Padé-z 2 estimator of determinants*, Physical Review D, 57 (1998), p. 1642.

[98] L. Trefethen and D. Bau, *Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1st ed., 1997.

[99] U. Trottenberg, C. W. Oosterlee, and A. Schüller, *Multigrid. with Guest Contributions by A. Brandt, P. Oswald, K. Stüben*, Academic Press, Orlando, FL, 2001.

[100] S. Ubaru, J. Chen, and Y. Saad, *Fast estimation of* $\mathrm{tr}(f(A))$ *via stochastic Lanczos quadrature*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1075–1099.

[101] S. Ubaru and Y. Saad, *Applications of trace estimation techniques*, in International Conference on High Performance Computing in Science and Engineering, Springer, 2017, pp. 19–33.

[102] R. Vershynin, *High-Dimensional Probability*, 2018.

[103] M. J. Wainwright and M. I. Jordan, *Log-determinant relaxation for approximate inference in discrete Markov random fields*, IEEE transactions on signal processing, 54 (2006), pp. 2099–2109.

[104] W. Wilcox, *Noise methods for flavor singlet quantities*, (1999).

[105] K. G. Wilson, *Confinement of Quarks*, Physical review D, 10 (1974), pp. 2445–2459.