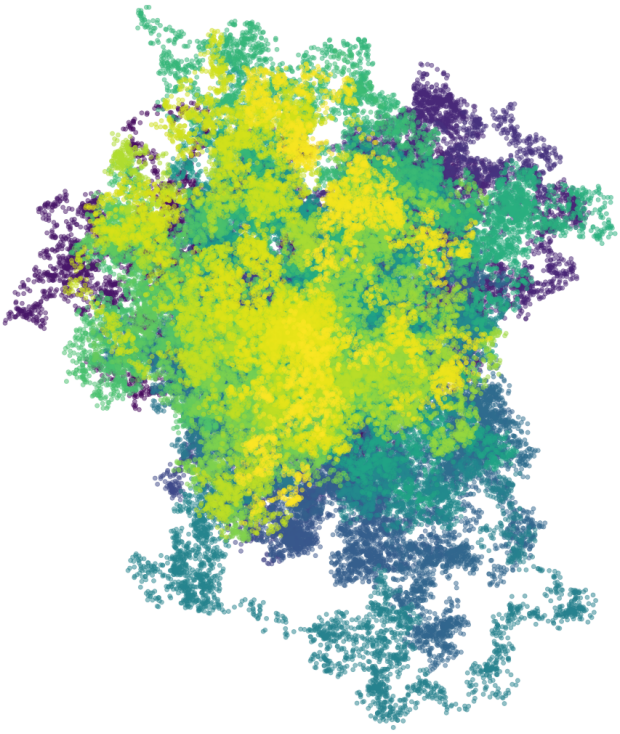
A Lattice Pairing-Field Approach to Ultracold Fermi-Gases

TECHNISCHE UNIVERSITÄT DARMSTADT



Vom Fachbereich Physik der Technischen Universität Darmstadt zur Erlangung des Grades *Doctor rerum naturalium* (Dr. rer. nat.) genehmigte Dissertation von

M. Sc. Florian Ehmann

aus Sindelfingen

1. Gutachten: Prof. Dr. Jens Braun
2. Gutachten: Prof. Dr. Hans-Werner Hammer

Darmstadt 2023

EN: A Lattice Pairing-Field Approach to Ultracold Fermi-Gases
DE: Ein Gitter-Pairing-Field Zugang zu Ultrakalten Fermigasen

Vom Fachbereich Physik der Technischen Universität Darmstadt zur Erlangung des Grades *Doctor rerum naturalium* (Dr. rer. nat.) genehmigte Dissertation von M. Sc. Florian Ehmann aus Sindelfingen

1. Gutachten: Prof. Dr. Jens Braun
2. Gutachten: Prof. Dr. Hans-Werner Hammer

Tag der Disputation: 7. Juni 2023
Jahr der Veröffentlichung: 2023

Abstract

Two-component Fermi gases model the behavior of many systems in different fields of physics, and one of their interesting features is that they condense into superfluids at low temperatures. Ultracold atoms experiments represent one realization of such Fermi gases, and their great flexibility sparked active research into their phase structure. In the literature, there are studies of the phase structure using functional methods, mean-field approximations, and other approaches. In the present work, we aim to perform *ab-initio* Monte-Carlo simulations of the system to probe their phase structure for inhomogeneous phases in the presence of spin imbalance. Such simulations generally require a bosonization of the theoretical description of the system that rewrites the theory in terms of an auxiliary bosonic field. For this auxiliary field, many possible choices achieve this, and previous studies often use a field that corresponds to a density of fermions. In the present work, we develop a novel approach to this problem by bosonizing the system in terms of the so-called *pairing field* which corresponds to the superfluid order parameter. Even in the absence of mass- or spin-imbalance, this approach exhibits a sign problem that presents a problem to many Monte-Carlo methods. To circumvent this sign problem, we base our simulation on the *Complex Langevin* method. The central question of this thesis is if a simulation of a pairing-field-based formalism using the Complex-Langevin method is suitable to study the phase structure of two-component Fermi gases in the presence of spin imbalance. To this end, we develop a lattice theory based on the pairing field by discretizing the continuous Hamiltonian and performing a rigorous derivation of the path integral from there. We pay special attention to the derivation of lattice derivative operators and rescale the theory to be dimensionless. Beyond that, we develop an efficient notation for lattice theories that makes their handling arguably easier than that of continuum theories. With the obtained theory, we derive the Langevin equation for the system together with the expressions we need to sample observables and develop a numerical simulation on that basis. We use the simulation to study $0+1$-dimensional systems as a proof-of-concept. Specifically, we calculate density equations of state and two-point functions for a range of dimensionless couplings and compare them to exact solutions. Additionally, we compare our obtained results for the two-point functions to general discussions of the analytic properties of correlation functions to gain additional insight into the qualitative and numerical behavior of our simulation. We found that our results for both density equations of state and two-point functions are in excellent agreement with the exact solutions. Going forward, we plan to study systems in $d > 0$ spatial dimensions, beginning with simulations of $1+1$-dimensional systems. In $d = 3$ dimensions the pairing field's correspondence to the superfluid order parameter may allow us to efficiently study the spontaneous breaking of the $U(1)$ symmetry of the system. On that basis, we can probe the phase diagram of the system for inhomogeneous phases at finite spin imbalance in future studies.

Kurzfassung

Zweikomponentige Fermigase modellieren das Verhalten vieler Systeme in verschiedenen Feldern der Physik und einer ihrer interessanten Aspekte ist, dass sie bei tiefen Temperaturen zu Superfluiden kondensieren. Experimente mit ultrakalten Atomen stellen eine Realisierung solcher Fermigase dar. Ihre große Flexibilität hat zu aktiver Forschung auf diesem der Phasenstruktur solcher Fermigase geführt. In der Literatur existieren Studien dieser Phasenstruktur auf Basis von funktionalen Methoden, Mean-Field-Näherungen und anderen Ansätzen. In der vorliegenden Arbeit beabsichtigen wir *ab initio* Monte-Carlo Simulationen dieses Systems durchzuführen, um die Phasenstruktur für inhomogene Phasen bei endlicher Spinpolarisierung zu untersuchen. Simulationen dieser Art erfordern in aller Regel eine Bosonisierung der theoretischen Beschreibung des Systems, die die Theorie in Form von bosonischen Hilfsfeldern darstellt. Es gibt viele mögliche Realisierungen für ein solches bosonisches Hilfsfeld und bisherige Studien haben oft ein Feld verwendet, welches Dichten von Fermionen entspricht. In dieser Arbeit hingegen entwickeln wir einen neuen Zugang zu diesem Problem, in dem wir das System durch das sogenannte *Pairig Field* darstellen, das dem superfluiden Ordnungsparameter entspricht. Selbst ohne Massen- oder Spinpolarisierung weist dieser Zugang ein Vorzeichenproblem auf, welches ein Problem für viele Monte-Carlo Methoden darstellt. Um dieses Vorzeichenproblem zu umgehen, basieren wir unsere Simulation auf der *Complex-Langevin*-Methode. Die zentrale Fragestellung dieser Dissertation ist, ob eine Simulation eines solchen Pairing-Field-basierten Formalismus unter Verwendung der Complex-Langevin-Methode geeignet ist, um die Phasenstruktur zweikomponentiger Fermigase in Anwesenheit von Spinpolarisierung zu untersuchen. Zu diesem Zweck entwickeln wir eine Gittertheorie, die auf dem Pairing Field basiert, indem wir den kontinuierlichen Hamiltonoperator diskretisieren und von diesem Punkt aus eine rigorose Herleitung des Pfradintegrals durchführen. Dabei legen wir besonderes Augenmerk auf die Herleitung von Ableitungsoperatoren auf dem Gitter und reskalieren die Theorie zu dimensionslosen Größen. Darüber hinaus entwickeln wir eine effiziente Notation für Gittertheorien, die deren Handhabung unter Umständen einfacher macht als die Handhabung von Kontinuumstheorien. Mit der gewonnenen Theorie leiten wir die Langevingleichung für das System und die Ausdrücke zur Berechnung von Observablen her und entwickeln eine numerische Simulation auf dieser Basis. Wir nutzen diese Simulation, um $0+1$-dimensionale Systeme als Proof of Concept zu studieren. Insbesondere berechnen wir Dichtezustandsgleichungen und Zweipunktfunktionen für eine Reihe von dimensionslosen Kopplungen und vergleichen diese mit exakten Lösungen. Darüber hinaus vergleichen wir unsere Ergebnisse für Zweipunkfunktionen mit allgemeinen Diskussionen von analytischen Eigenschaften von Korrelationsfunktionen, um zusätzliches Verständnis über das qualitative und numerische Verhalten unserer Simulation zu erlangen. Wir fanden heraus, dass sowohl unsere Resultate für Dichtezustandsgleichungen, als auch die für Zweipunktfunktionen exzellent mit exakten Lösungen übereinstimmen. Von hier an planen wir Systeme in $d > 0$ Raumdimensionen zu studieren, beginnend mit Simulationen von $1+1$-dimensionalen Systemen. In $d = 3$ Dimensionen kann die Entsprechung des Pairing Fields zum superfluiden Ordnungsparameter uns erlauben, auf effiziente Weise die spontane Brechung der $U(1)$ Symmetrie des Systems zu

studieren. Auf dieser Basis können wir in zukünftigen Studien das Phasendiagramm des Systems auf inhomogene Phasen in Anwesenheit von Spinpolarisierung hin untersuchen.

Acknowledgments

As I am sitting at my desk, finishing the last parts of my dissertation, as my ability to form sentences comes to an end, so does my time as a PhD student. I cannot end my thesis without a few words of gratitude.

Firstly, I want to thank my supervisor Jens. Jens, I had an incredibly good time working with you. Our discussions have always been a highlight of my day. As productive as it was to discuss intricate details of our work with you, I also greatly appreciate our conversations' tendency to drift to all sorts of topics. Even when you were short on time, you always did your best to be there for your group. As any PhD project does, mine had its fair share of highs and lows, and the pandemic certainly did not make that any less true. I want to thank you for always having an open ear and helping me through the more difficult times. I also want to thank Joaquín. You were always there for discussions about the project, even though you are on the other side of the world and the other side of the day. Due to "the situation", I did not get to do a lot of traveling in my time as a PhD student, but I had a blast when I visited you and your group at UNC for the RPMBT-21 conference. On a more formal note, I also want to thank you and your collaborators, Michael Hoffman, Philip Javernick, Andrew Loheac, William Porter, and Eric Anderson, for sharing your data with me for this thesis. While I want to thank my entire group at the IKP for the time I have had with them, two people deserve a special mention. The first one is Lukas, who co-supervised my master's thesis. You constantly radiate an aura of positivity, and you excel at conveying your passion for your work. Working with you was truly inspiring, and you taught me a great deal about work as a scientist. The other special mention goes to my officemate for the majority of my time as a PhD student, Sebastian. Our random discussions have been the kind of distraction one needs to be truly productive, and I have greatly enjoyed our lunch breaks. You were always ready to provide me with random seeds for my RNGs and my thoughts and that is greatly appreciated. While I am on the topic of work, I want to thank my proofreaders, Tobias, Timon, Marwin, Sebastian, and Vanessa. Without your patience and diligence, this thesis would not be in the shape it is in now.

In my time in Darmstadt, I have met many wonderful people, but of my friends, I want to mention two in particular. Marwin, our years as flatmates have, without a doubt, been the best of my time at the university. I remember the great times we had, wasting entire evenings on the couch, watching silly little videos. It has been a privilege to meet someone who shares my bizarre sense of humor to such a degree. Tobias, you and I have been the last members of a group that once filled entire tables in the canteen. I greatly enjoyed our extended lunch breaks and discussions about physics, academia, software development, and pretty much everything else. I also want to thank my parents and my brother. Even as a child, you have always supported my scientific inclinations and done your best to guide your headstrong son and brother to adulthood, and in my time at university, I could always count on your support.

Furthermore, I cannot end this section without acknowledging the remarkable work of the open-source software community as a whole. Their impact is often unmentioned, but my work and work like mine would simply not be possible without them and, beyond that, for me, they serve as a reminder that there is good in humanity.

To all of you, a thousand thanks – I will always remember the good times I had in Darmstadt, and you made them what they were.

Contents

1 Introduction

In this work, we aim to study systems of two fermion species which we call “up” and “down” with a contact interaction in thermal equilibrium in the context of ultracold atoms experiments. Such a system can model the behavior of many physical systems, not only in the context of ultracold atoms but also nuclear matter and condensed matter physics. A very interesting feature of these systems is the fact that they turn into superfluids at sufficiently low temperatures. The experiments in the field of ultracold atoms, in particular, have gotten to a point at which it is possible to tune the interaction strength between the two fermion species, select different values for mass- and spin-polarization and even confine the system to one or two spatial dimensions, and study the fate of superfluidity under such variations of physical parameters [1–10]. With this much flexibility in the experiment, there is a great deal of synergy between experiments and theoretical calculations in the field of ultracold atoms.

Because of the infinite number of states in the Fock space of this system, it is generally not possible to solve it by means of exact diagonalization. Therefore, we choose to solve the system using ab-initio Monte-Carlo methods. Based on the principle of importance sampling, this approach seems ideally suited for the solution of path integrals, since the majority of the path integral value stems from field configurations around the classical solution of the system. The fermionic nature of the system causes the quantum fields in the action to be Grassmann-valued. This is not ideal for numerical simulations, as computers are far more efficient at handling real or complex numbers if one is at all able to implement calculations of Grassmann numbers directly. To eliminate this problem, we employ a Hubbard-Stratonovich transformation to rewrite the theory in terms of an auxiliary bosonic quantum field. Such a Hubbard-Stratonovich transformation is not unique, and there are many possible choices for such auxiliary fields [11]. A very common choice is an auxiliary field that corresponds to densities of fermions. Such density formalisms have been used to calculate density equations of state and density correlations functions in various spatial dimensions d, see, e.g., Refs. [12, 13]. For the present work, however, we choose to bosonize the system in terms of the so-called *pairing field* that, loosely speaking, describes how two fermions couple to a bosonic pair. Since it directly corresponds to the superfluid order parameter in our system of interest, it seems to be a natural approach to studying the phase structure of this system. While the Hubbard-Stratonovich transformation based on the pairing field is, itself, not new, this represents a novel approach to the phase structure of ultracold gases. Rewriting the system in terms of the pairing field causes a *sign problem* even in the absence of mass- or spin-imbalance. To circumvent this problem, we choose the Complex Langevin (CL) approach to perform the sampling for our Monte Carlo calculations. Based on this formalism, we develop a robust software package that performs simulations of the system and present results of this simulation for $0+1$-dimensional systems that serve as a proof-of-concept of the pairing-field formalism and the CL-based simulation. Furthermore, the details of our novel formalism and the results of our proof-of-concept calculations are currently being published [14].

We begin this thesis by briefly reviewing essential aspects of ultracold atoms. After that, in Chapter 2, we move on to the basic concepts of Monte-Carlo integration in general and stochastic

quantization in particular. Additionally, we review methods of uncertainty estimation in Monte Carlo calculation. In Chapter 3, we develop our formalism in the continuum before we rigorously derive it on the lattice to obtain the lattice theory that serves as a basis for our simulation. Subsequently, in Chapter 4, we derive the Langevin equation for the pairing-field formalism and determine the expressions we need to sample observables of the theory. We also discuss aspects of modern software development to choose appropriate tools and programming languages to implement the simulation. In Chapter 5, we discuss the results of our $0+1$-dimensional proof-of-concept calculations and compare them to exact results.

1.1 Ultracold Atoms

Our system of interest can be realized in the class of so-called *ultracold atoms* experiments. What makes this particular field so interesting is the fact that the behavior of ultracold atoms can be tuned to an incredible degree. For example, as we discuss in Sec. 1.1.2, the interaction strength between the species of a Fermi gas of ultracold atoms can be tuned essentially at will. This creates a unique interplay between experiment and theory, as experimental groups can recreate the systems simulated by theorists for arbitrary interactions. In contrast, in most other fields of physics, theorists can vary the parameters of systems in their calculations but in experiments, one is limited to the parameters and interactions that are realized in nature.

One way to realize different fermion species in ultracold atoms experiments is to prepare neutral atoms in different hyperfine states. Clouds of these atoms are then trapped and cooled to temperatures roughly ranging from $50-400\,\mathrm{nK}$ [15–19] at densities typically around $10^{12}-10^{13}\,\mathrm{atoms/cm^3}$ [15, 19, 20]. At these temperatures, which are only fractions of the Fermi temperature T_F, the system enters the quantum degenerate regime, and the dynamics of the system become dominated by quantum effects. For the aforementioned low densities, the average interparticle spacing $n^{-1/3}$ is large compared to the effective range r_0 of the interaction potential. As such, the microscopic details of the interaction become irrelevant, and the interaction is fully characterized by the scattering length a_s. On the theoretical side, this allows us to simulate these systems using universal and rather simple Hamiltonians.

1.1.1 BEC-BCS Crossover and the Unitary Limit

Historically, most of the interesting quantum effects of ultracold systems appear in one of two limits: the *Bardeen-Cooper-Schrieffer* (BCS) limit or the *Bose Einstein Condensation* (BEC) limit. For Fermi gases, these two limits already describe some very interesting physical effects. The principles of BCS theory have been applied to study the behavior of pairing interactions in atomic nuclei [21] and neutron stars [22]. Aspects of BCS theory have even been applied to the high-density regime of QCD, which may be a color superconductor [23].

With ultracold atoms experiments, it has become possible to not only probe the BCS and BEC limits but also the continuous crossover between them. This has become possible because, in modern ultracold atoms experiments, it is essentially possible to tune the interaction at will, as we discuss in the next section. Figure 1.1 shows a sketch of the phase diagram of a two-component Fermi gas in $d=3$ spatial dimensions. For every sufficiently negative value of $1/(k_F a_s)$, with the Fermi momentum k_F and the scattering length a_s, the system behaves as a normal Fermi liquid for sufficiently high temperatures. As the temperature decreases, the system starts to form compound bosonic pairs of two particles of different species. In the phase diagram, this "crossover region" lies around the curve T^*. As the temperature decreases further and crosses T_c, the pairs condense and form a superfluid. For large negative values of $1/(k_F a_s)$, the dynamics of the system are described by BCS theory, and for large positive values of $1/(k_F a_s)$, the dynamics are

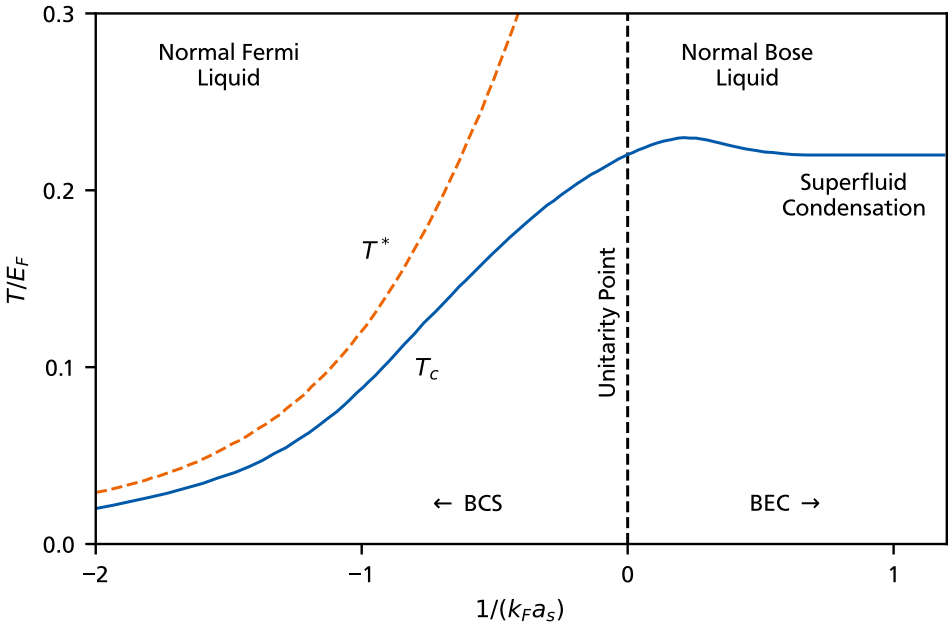


Figure 1.1: Schematic representation of the phase diagram of a two-component Fermi gas in $d = 3$ spatial dimensions with data taken from Ref. [24]. The horizontal axis shows $1/(k_F a_s)$ with the Fermi momentum k_F and the scattering length a_s. The vertical axis shows the temperature in units of the Fermi energy E_F. The dashed orange line shows the temperature T^* around which pairs begin to form with decreasing temperature. The blue line shows the temperature T_c at which superfluid condensation occurs. Along the horizontal axis, the system transitions from the BCS limit at large negative values of $1/(k_F a_s)$ into the BEC limit at large positive values of $1/(k_F a_s)$. At $1/(k_F a_s) = 0$, the system crosses the unitary point.

described by BEC theory. In between the two limits, the system exhibits a continuous crossover from BCS to BEC dynamics. At $1/(k_F a_s) = 0$, the system crosses the so-called *unitary point* with an interesting property. As the scattering length a_s diverges, we find

$$a_s \gg n^{-1/3} \gg r_0\,, \tag{1.1}$$

and the density becomes the only scale left in the system. In this regime, it becomes irrelevant what specific types of fermions and interactions we study, as we observe universal behavior. Ultracold gases of neutral atoms show the same behavior as low-density neutron matter [25] does in this regime. This makes the unitary point particularly interesting for studies of ultracold atoms. The phase diagrams of two-component Fermi gases for $d < 3$ share qualitative similarities with the phase diagram for $d = 3$, although there are certain differences. For example, there is no unitary point in $d = 2$ because pairing appears at arbitrary weak interactions [26–28]. Regardless of the spatial dimension, however, the properties of the system along the crossover from BCS to BEC are determined by the details of the pairing of fermions. In the BCS limit, fermions form weakly localized pairs that are much larger than the average interparticle spacing $n^{-1/3}$ and have zero center-of-mass momentum. On the other side of the crossover, in the BEC limit, the fermions form tightly bound dimers that act like localized composite bosons. As such, describing the system in terms of "effectively fundamental bosons" seems to be a natural approach to studying its phase structure.

While the structure of the phase diagram of the system in $d > 0$ dimensions is the eventual goal of our studies, the present work is focused on developing a new approach and simulation

of the problem. Therefore, in this work, we focus on proof-of-concept calculations of systems in $d = 0$ dimensions.

1.1.2 Interaction Tuning with Feshbach Resonances

In the field of ultracold atoms, one often deals with dilute and very cold systems to make quantum effects observable. In this regime, the thermal wavelength of particles $\lambda_{\mathrm{th}} = (2\pi\beta/m)^{1/2}$, with the particle mass m and the inverse temperature $\beta = 1/k_B T$ is significantly greater than the average interparticle distance $n^{-1/3}$, which makes the microscopic details of the interaction potential between the particles irrelevant and allows us to describe the interaction as a contact interaction between particles of different species with a coupling g. From a scattering perspective, the interaction between the particles is dominated by the s-wave channel. As such, it can be described by a single parameter, the s-wave scattering length a_s. In ultracold atoms experiments, it has actually become possible to tune the scattering length essentially at will using *Feshbach resonances* [29]. An extensive review of this phenomenon can be found in Ref. [30] and a pedagogical discussion of a one-dimensional system in Ref. [31]. As a simple example of a system that exhibits such Feshbach resonances, we consider an elastic scattering problem with two possible final states. One of these states is energetically accessible as a final state because it has the same energy as the incoming state, we call it the *open channel*. The other state has a higher energy than the open channel. Thus, it is energetically inaccessible, and we call it the *closed channel*. For both channels, we find that, through the interaction, the potential energy of the system varies with the particle separation distance. In the case of electrically neutral atoms, these potential curves represent the van der Waals interaction between the atoms. As such, they are attractive for larger separations, have a repulsive core around zero separation, and, in between, feature a local minimum. Possible potential curves for our two-channel model are shown in Fig. 1.2. The energies of the open- and closed channels lie at the thresholds of their respective potential curves. A Feshbach resonance occurs in this system when the well of the potential curve of the minimum is deep enough to feature a bound state and when the energy of that bound state is close to that of the open channel. Despite still being unable to access the closed channel, scattering particles can form virtual bound states in that situation, which varies the scattering length and leads to a diverging scattering length as the energy separation ΔE between open- and closed channel approaches zero.

What makes Feshbach resonances tunable is that the energy separation ΔE can be modified in experiments. In the case of neutral atom experiments, up and down species are usually realized as two different hyperfine states of atoms. To implement the tunability, the open- and closed channels in our example feature a different total spin S. In the case of spin-1/2 particles, the open channel could be a singlet state with $S = 0$, and the closed channel could be a triplet state with $S = 1$. This causes the two channels to experience a different energy shift δE in response to an external magnetic field, as the energy shift is proportional to the total spin of the state:

$$\delta E \propto \mu_{\mathrm{B}}\, \mathbf{S} \cdot \mathbf{B}\,, \tag{1.2}$$

with the Bohr magneton μ_{B}, the total spin $\mathbf{S}$ and the magnetic flux density $\mathbf{B}$. This allows an external magnetic field to control the size of the energy separation ΔE between the two channels and, thus, the scattering length. It is possible to find the following effective relation between the scattering length and the external magnetic flux density [30]:

$$a_s(B) = a_{s,\mathrm{bg}} \left(1 - \frac{\Delta B}{B - B_0}\right)\,. \tag{1.3}$$

The parameter $a_{s,\mathrm{bg}}$ represents the background scattering length we would observe in the absence of the Feshbach resonance, B_0 represents the magnetic flux density at which the energy separa-

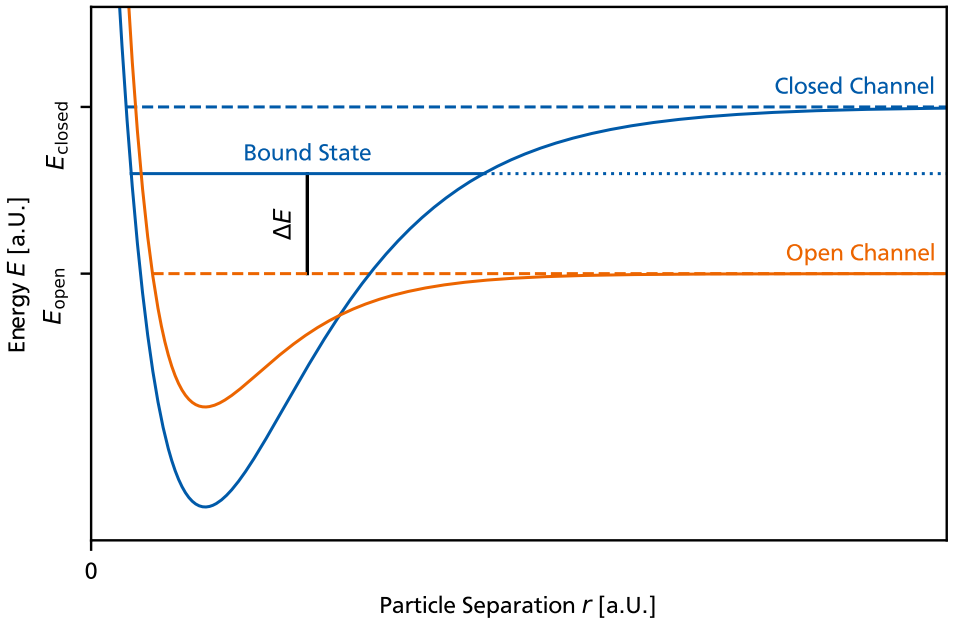


Figure 1.2: Sketch of possible potential curves of open- and closed channels over the particle separation. The closed channel features a deep enough well to create a bound state that lies close to the energy of the open channel. The energy separation ΔE between the open channel and the bound state of the closed channel can be varied via an external magnetic field. When ΔE becomes small, the Feshbach resonance occurs because scattering particles form a virtual bound state.

tion ΔE between the channels vanishes and ΔB describes the width of the resonance. A sketch of this relation is shown in Fig. 1.3. In this figure, we observe that the scattering length diverges at the magnetic flux density B_0. This divergence marks the so-called unitary point of $1/(k_F a_s) = 0$ that we discussed above. Furthermore, we note that the Feshbach resonance allows the scattering length to be tuned to values that put the system in the BCS limit of $1/(k_F a_s) \to -\infty$ or the BEC limit of $1/(k_F a_s) \to +\infty$ and, indeed, in the intermediate regime of the BCS-BEC crossover.

To relate experimental realizations of cold atom gases and their Feshbach resonance controlled interactions to theoretical models, we need to find a connection between the experimentally accessible s-wave scattering length a_s and the coupling parameter g of a Hamiltonian that describes a two-species Fermi-gas with attractive contact interactions. We achieve this by determining the T-matrix for the experimental system, calculating it for the theoretical model, and matching the two expressions. The T-matrix describes how a scattering event affects the wave function of a particle. In the ultracold dilute regime of ultracold atoms, the interaction between particles that leads to the scattering event is effectively given by a contact interaction. As a result, the T-matrix is dominated by the s-wave channel, i.e., the scattering occurs predominantly without angular momentum between the two particles. This simplification allows us to approximate the T-matrix of the experimental system as

$$T^{(+)}\left(k, k; E = \frac{k^2}{2\mu}\right) \approx \frac{4\pi}{\mu} \frac{1}{1/a_s + ik}\,. \tag{1.4}$$

The superscript $+$ in $T^{(+)}$ indicates that we are describing the outgoing, scattered wave, and the argument $E = k^2/2\mu$ indicates that we are considering *on-shell* scattering, i.e., elastic scattering

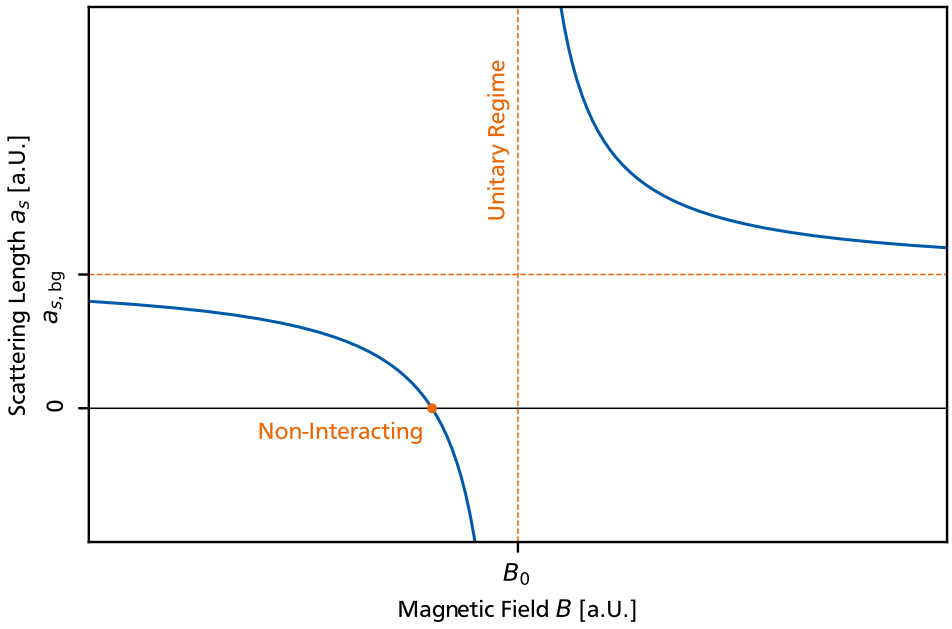


Figure 1.3: Scattering length over the range of an external magnetic field. The scattering length has a pole at the magnetic flux density B_0, where the Feshbach resonance occurs. This divergence of the scattering length marks the unitary point.

between the two particles with the reduced mass

$$\mu = \frac{m_\uparrow m_\downarrow}{m_\uparrow + m_\downarrow} \,. \tag{1.5}$$

Now we need to calculate the T-matrix from a theoretical model of the ultracold atom gas. Below, in Eq. (3.1), we describe the system for an arbitrary number of particles using second quantization. This Hamiltonian serves as the starting point for the development of our theory. For our purposes in this section, it is beneficial to directly consider a system of only two particles for this calculation. Specifically, we describe the system in relative coordinates $\mathbf{r}$, with the particle of the down species resting at $\mathbf{r} = 0$ and the particle of the up species described by the wave function $\psi(\mathbf{r})$. The Hamiltonian of this system in position space reads

$$\hat{H} = -\frac{\nabla^2}{2\mu} + g\,\delta^{(3)}(\mathbf{r}) \,. \tag{1.6}$$

We deduce that the scattering results from a δ-potential at the origin of the relative coordinate space. The T-matrix for this Hamiltonian can be determined by solving the *Lippmann-Schwinger* equation

$$T^{(+)}(\mathbf{k}', \mathbf{k}; E) = V(\mathbf{k}', \mathbf{k}) + \int \frac{\mathrm{d}^3 p}{(2\pi)^3} \frac{V(\mathbf{k}', \mathbf{p})}{E - p^2/2\mu + i\varepsilon}\, T^{(+)}(\mathbf{p}, \mathbf{k}; E) \,, \tag{1.7}$$

with the momentum-space matrix-element of the potential $V(\mathbf{k}', \mathbf{k})$ for the Hamiltonian in Eq. (1.6) simply being the coupling parameter:

$$V(\mathbf{k}', \mathbf{k}) = g \,. \tag{1.8}$$

We can solve this equation by recursively inserting it into itself, resulting in a so-called *Born series*. Because this series is a geometric one, we can determine its limit and find

$$T^{(+)}(\mathbf{k}', \mathbf{k}; E) = \frac{1}{1/g - I(E)} \tag{1.9}$$

with the integral

$$\begin{aligned} I(E) &= \int \frac{\mathrm{d}^3 p}{(2\pi)^3} \frac{1}{E - \mathbf{p}^2/2\mu + i\varepsilon} \\ &= \frac{\mu}{\pi^2} \int \mathrm{d}p \, \frac{p^2}{2\mu E - p^2 + i\varepsilon} \, . \end{aligned} \tag{1.10}$$

The term $+i\varepsilon$ in the denominator of the integrand moves the pole of the integral away from the real axis, and, in practice, we compute this integral as its Cauchy principal value. This integral is linearly divergent. Therefore, we introduce a momentum cutoff Λ to regularize this integral. We find

$$\begin{aligned} I(E) &= \frac{\mu}{\pi^2} \int_0^\Lambda \mathrm{d}p \, \frac{p^2}{2\mu E - p^2 + i\varepsilon} \\ &= -\frac{\mu}{\pi^2}\Lambda + \mathcal{O}\left(k^2/\Lambda^2\right) \, , \end{aligned} \tag{1.11}$$

with $k = \sqrt{2\mu E}$. Inserting this result into the T-matrix of the theoretical model in Eq. (1.9) and matching it to the T-matrix of the experimental system in Eq. (1.7) yields the relation

$$g = \frac{4\pi}{\mu} \frac{1}{1/a_s - 4\Lambda/\pi} \, , \tag{1.12}$$

or, alternatively,

$$a_s = \frac{1}{4\pi/(\mu g) + 4\Lambda/\pi} \tag{1.13}$$

These relations allow us to connect the microscopic coupling g and the experimentally observable scattering length a_s. Note that the coupling g depends on Λ as a consequence of our regularization and renormalization procedure.

1.1.3 Imbalance in Ultracold Atoms Experiments

In our system of interest, we can implement two kinds of imbalance. *Mass imbalance* refers to different masses for the two fermion species. In experiments, mass imbalance can be realized by using different chemical elements or different isotopes of the same chemical element to realize the two components of the Fermi gas. Common examples include ^{161}Dy with ^{162}Dy, see, e.g., Ref. [32], and ^{53}Cr with ^{6}Li, see, e.g., Ref. [33], although this list is by no means meant to be exhaustive. Another form of imbalance is *population imbalance*. In that case, the system consists of a different number of particles for each species. Because the particle species in this context are often called "up" and "down" regardless of how they are realized in the concrete experiment, population imbalance is also commonly referred to as *spin imbalance*. In the experiment, *spin imbalance* can be realized by preparing a system with different particle numbers in each species. If the species are realized as two different hyperfine states of the same atom, this can be achieved by preparing the states accordingly.

The original formulation of BCS theory requires balanced systems for the formation of Cooper pairs, and it is not trivial how imbalance can be included in this formalism. If we continue to assume the BCS pairing mechanism is at work in these systems, there are multiple possible scenarios [34]. On the one hand, the system could separate into a superfluid part and a normal fluid part that is fully or partially polarized. On the other hand, superfluidity could break down altogether.

However, there is also another possible consequence of imbalance. Both mass- and spin-imbalance create a gap between the Fermi surfaces of both fermion species. In this situation,

there could be other types of pairing than the BCS pairing mechanism. One possibility is the pairing of particles with momenta on the opposing sides of their respective Fermi surfaces. This would lead to a pair with non-vanishing center-of-mass momentum $\mathbf{q}$. Pairing of this kind is expected to cause the superfluid order parameter $\Delta(\mathbf{r}) \propto \langle\hat{\psi}_\uparrow(\mathbf{r})\hat{\psi}_\downarrow(\mathbf{r})\rangle$ to lose its translation invariance and form a non-trivial dependence on $\mathbf{r}$. The operators $\hat{\psi}_\uparrow$ and $\hat{\psi}_\downarrow$, in this case, are the field operators of the fermion species. Two popular ansätze for such a dependence are the one by Fulde and Ferrell (FF) [35] with $\Delta(\mathbf{r}) = \Delta e^{i\mathbf{q}\cdot\mathbf{r}}$ and a momentum mode $\mathbf{q}$, and the one by Larkin and Ochinnikov (LO) [36] with $\Delta(\mathbf{r}) = \Delta\cos(\mathbf{q}\cdot\mathbf{r})$ and at least two momentum modes at $\pm\mathbf{q}$. Collectively, these two ansätze are commonly referred to as FFLO. Phases of exotic pairing, such as FFLO, along axes of polarization are likely to become a future subject of study with our formalism. Therefore, we include polarization in the construction of the formalism, even though the calculations in the present work are limited to the balanced case.

1.1.4 Reduced Dimensions

For our proof-of-concept calculations, we limit ourselves to the case of $d = 0$ spatial dimensions for now but plan to move on to $d = 1$ in the future. But even though our everyday life takes place in $d = 3$ dimensions, that does not mean that experimental results are not available for $d < 3$ [37]. In traps for ultracold atoms, the trapping potential around its minimum can be described by a harmonic potential

$$V_{\text{trap}} = \frac{m}{2}(\omega_x^2 x^2 + \omega_y^2 y^2 + \omega_z^2 z^2)\,. \tag{1.14}$$

Instinctively, we may think of a homogeneous trap with $\omega_x = \omega_y = \omega_z$, but that is far from the only possibility. Traps in cold atoms experiments can be modified in a way that essentially confines the atoms to two or fewer spatial dimensions. Consider an initially homogeneous trap with $\omega_x = \omega_y = \omega_z$. As described in, e.g., Ref. [38], we can superimpose this trap with counterpropagating laser beams along the z-direction to modulate the trapping potential along that direction. These counterpropagating laser beams form a standing wave which creates a periodic structure of minima in the trapping potential. In the vicinity of each of those minima, we can again describe the trapping potential as a harmonic potential of the form in Eq. (1.14) with $\omega_z \gg \omega_x, \omega_y$. If the trap frequency ω_z in these minima is so large that the first excited state in z-direction is well above the usual energies of the trapped atoms, i.e., $\omega_z \gg E_F$, then the trap is so narrow, that the motion of the atoms is effectively confined to the xy-plane. Consequently, superimposing the standing wave of counterpropagating laser beams has divided the three-dimensional trap into a stack of two-dimensional traps. One can repeat this procedure with a standing wave along the y-direction of the system to create one-dimensional trap tubes that effectively limit the motion of the atoms to just the x-direction.

In summary, beyond the experiments' astonishing capabilities of tuning the interaction and imbalance parameters of systems, it is even possible to realize systems with less than three spatial dimensions.

2 Methods

In this chapter, we discuss the numerical methods for the simulation of our physical system of interest that we employ in this work. We examine how we can use these methods to calculate observables and how we can estimate their uncertainties. Beyond that, we also briefly touch upon possible future optimizations of the simulation technique employed in this work.

The central problem we need to solve with our simulation is the calculation of observables from some description of a physical system. In the operator formalism of Statistical Physics, the system is described by a Hamiltonian $\hat{H}$, and we find observables

$$\langle \hat{O} \rangle = \frac{1}{Z} \operatorname{tr} \left(\hat{O}\, e^{-\beta(\hat{H} - \mu_\downarrow \hat{N}_\downarrow - \mu_\uparrow \hat{N}_\uparrow)} \right) \tag{2.1}$$

that are calculated from the operator $\hat{O}$ in a grand-canonical ensemble with inverse temperature $\beta = 1/T$ and chemical potentials $\mu_\uparrow$ and $\mu_\downarrow$. The grand-canonical partition function in the operator formalism is given by

$$Z = \operatorname{tr} \left(e^{-\beta(\hat{H} - \mu_\downarrow \hat{N}_\downarrow - \mu_\uparrow \hat{N}_\uparrow)} \right) . \tag{2.2}$$

As an alternative to the operator formalism, we can express the same observable in the path-integral formalism. For most of the present work, the path-integral formalism is the formalism of choice since it interacts nicely with the Langevin method we discuss below; they both rely on an action functional for the description of the physical system. In this representation, we have

$$\langle \hat{O} \rangle = \frac{1}{Z} \int D(\psi^*, \psi)\; \mathcal{O}_O(\psi^*, \psi)\; e^{-S[\psi^*, \psi]} , \tag{2.3}$$

with yet to be defined fermionic fields ψ^* and ψ and an Euclidean action functional S. In the path-integral formalism the partition function is given by

$$Z = \int D(\psi^*, \psi)\; e^{-S[\psi^*, \psi]} . \tag{2.4}$$

The expression $\mathcal{O}_O(\psi^*, \psi)$ in Eq. (2.3) relates the configurations of the fields ψ^* and ψ to the observable and serves the same purpose as the operator $\hat{O}$ in Eq. (2.2).

Both calculations in Eq. (2.2) and Eq. (2.3) are similar in principle: we combine an infinite number of contributions from different states of the system into one observable value. For some systems, there are analytical solutions for these equations. However, when this is not the case, approximating these calculations that span an infinite number of states of the system is quite non-trivial. In general, our system of interest does not have an analytical solution for more than zero spatial dimensions[1], so we need to find a method that can simulate the system in an approximate numerical fashion. Fortunately, we can find such methods in the class of Monte-Carlo methods.

[1] A notable exception it the case of $1 + 1$-dimensional systems where exact solutions for some observables exist [37].

2.1 Monte-Carlo Integration

Monte-Carlo methods get their name from the famous Monte-Carlo district of Monaco that houses the legendary Casino de Monte-Carlo. They carry this name because they rely on random numbers to achieve their goal. One subclass of Monte-Carlo methods is focused on estimating the values of integrals using random numbers. This so-called Monte-Carlo integration can help us to approximate the path integral in calculations of observables such as in Eq. (2.3). For a simple demonstration of the basic principle, we take a look at the *mean-value method* for evaluating the integral

$$I = \int_a^b \mathrm{d}x\, f(x)\,. \tag{2.5}$$

The mean value of the function f on the interval from a to b is given by

$$\langle f \rangle = \frac{1}{b-a} \int_a^b \mathrm{d}x\, f(x)\,. \tag{2.6}$$

which contains the integral I in its definition. Thus, solving the integral I and determining the mean value of f are equivalent tasks, and we have

$$I = (b-a)\langle f \rangle = \langle (b-a) f \rangle\,. \tag{2.7}$$

We can estimate the mean value using random numbers by choosing a random sample of points $x_1, \ldots, x_N$, distributed uniformly at random on the interval $[a,b]$. We then calculate a sample point of our "observable" $f_i = (b-a)f(x_i)$ for each of them. With the observable sample $\{f_i\}$, the integral can be approximated as

$$I \approx \frac{1}{N} \sum_{i=1}^{N} f_i\,. \tag{2.8}$$

That is the fundamental idea of Monte-Carlo integration which allows us to simulate our physical system of interest. This integration approach based on random sampling of a function presents an advantage when performing higher-dimensional integrals. If we sample the function on a uniform lattice, the number of sample points scales with the power of the integral dimension and quickly becomes impractically large. If we choose a coarser lattice to prevent this, we may miss features of the integrand that contribute significantly to the value of the integral. With Monte-Carlo integration, on the other hand, our random sampling of the integrand gives us a random chance to capture fine features that deterministic methods may systematically miss with few sample points. As such, as long as the value of the integral is not predominantly determined by a small part of the integration region, Monte-Carlo integration allows us to obtain a reasonable estimate of the integral value with far less computational effort than deterministic numerical integration methods.

2.1.1 Importance Sampling

With the advantages of Monte-Carlo integration for higher-dimensional integrals in mind, Monte-Carlo integration sounds like the perfect tool for solving path integrals, as they are of incredibly high dimension. There is just one catch: as we stated above, the mean-value method only works if the value of the integral is not largely determined by only a small part of the integration region. The value of a path integral is, in fact, almost entirely determined by the tiny fraction of field configurations that lie close to the classical solution of the system. Therefore, it may seem like Monte-Carlo integration is not such a good choice for our problem after all. Fortunately, there

is an extension of the mean-value method that is not only a workaround to this issue but makes Monte-Carlo integration as a tool a lot more powerful: importance sampling.

The fundamental idea of importance sampling is to sample the integrand at points that are not uniformly distributed at random on the interval of integration but follow a probability distribution $p(x)$. That probability distribution is defined to have a maximum in the region that contributes the most to the value of the integral. To illustrate this idea, we follow the example presented in Ref. [39] and use importance sampling to estimate the value of the integral

$$I = \int_0^1 \mathrm{d}x \, \frac{x^{-1/2}}{e^x + 1} \,. \tag{2.9}$$

Even though this is a simple example, it is already difficult to determine I with deterministic standard methods of integral evaluation. This is due to the fact that the integrand diverges for $x \to 0$, and the bulk of the final value of I comes from a very narrow region around $x = 0$. To solve this integral by using importance sampling, we modify the mean-value method to sample the integrand mainly around the divergence at $x = 0$. This is achieved by using the divergence-creating numerator $x^{-1/2}$ of the integrand as a weight function

$$w(x) = x^{-1/2} \tag{2.10}$$

which we can normalize to obtain the probability distribution

$$p(x) = \frac{w(x)}{\int_0^1 \mathrm{d}x \, w(x)} = \frac{1}{2\sqrt{x}} \,. \tag{2.11}$$

To use the weight function w in the integration, we define a weighted average

$$\langle g \rangle_w = \frac{\int_0^1 \mathrm{d}x \, w(x) g(x)}{\int_0^1 \mathrm{d}x \, w(x)} \tag{2.12}$$

or, using the probability distribution p,

$$\langle g \rangle_p = \int_0^1 \mathrm{d}x \, p(x) g(x) \,. \tag{2.13}$$

When we use the fundamental idea of Monte-Carlo integration in Eq. (2.8), we see that we can also approximate the weighted average $\langle g \rangle_p$ by the sum

$$\langle g \rangle_p \approx \frac{1}{N} \sum_{i=1}^{N} g(x_i) \,, \tag{2.14}$$

if the random sample points $x_1, \ldots, x_N$ follow the probability distribution p. To solve the integral I, we calculate the weighted average of f/p:

$$\left\langle \frac{f}{p} \right\rangle_p = \int_0^1 \mathrm{d}x \, f(x) = I \,, \tag{2.15}$$

which we can approximate by

$$I \approx \frac{1}{N} \sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \,, \tag{2.16}$$

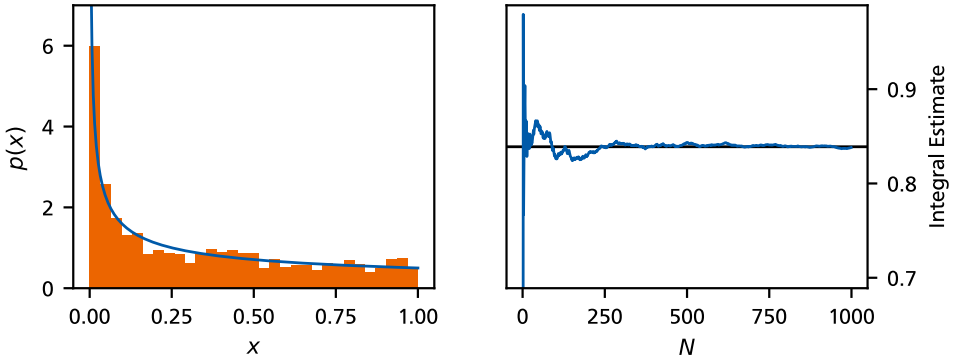


Figure 2.1: Application of importance sampling to solve the integral in Eq. (2.9). The solid blue line in the left-hand figure shows the chosen weight probability distribution p from Eq. (2.11). The orange histogram shows the distribution of $N = 1000$ random points $\{x_i\}$ that are chosen according to p. The solid blue line in the right-hand figure shows the convergence of the integral estimate with the number of used sample points on the horizontal axis. The estimate converges to the exact result, represented by the solid black line.

given a random sample of points $x_1, ..., x_N$ distributed according to p. An importance-sampling-based solution of the integral I in Eq. (2.9) is shown in Fig. 2.1 for up to $N = 1000$ random sample points.

The difficulty of this method is finding random sample points that follow the defined probability distribution p. For our example, we generate the random numbers we see in Fig. 2.1 with a transformation method shown in Ref. [39]. We begin by drawing N random numbers $\{z_i\}$, distributed uniformly at random on the interval $[0, 1]$ with the uniform probability density

$$q(z) = \begin{cases} 1 & 0 \leq z \leq 1\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{2.17}$$

We then search for a transformation $x(z)$ that transforms the random sample $\{z_i\}$ following the probability distribution $q(z)$ into a random sample $\{x_i\}$ following the probability distribution $p(x)$. To this end, we require

$$p(x')\,\mathrm{d}x' = q(z)\,\mathrm{d}z\,. \tag{2.18}$$

We integrate both sides, solve for $x(z)$, and find

$$\begin{aligned} \int_{-\infty}^{x(z)} \mathrm{d}x'\; p(x') &= \int_{-\infty}^{\infty} \mathrm{d}z\; q(z) \\ \Rightarrow \qquad \sqrt{x(z)} &= z \\ \Rightarrow \qquad x(z) &= z^2\,. \end{aligned} \tag{2.19}$$

For the probability distribution in Eq. (2.11), this is a straightforward process. However, if the probability distribution is given by the path integral weight e^{-S}, finding a transformation that turns uniformly random field configurations into field configurations that follow the probability distribution given by the path integral weights requires us to actually solve the path integral. If we could do that, we would not need Monte-Carlo integration, to begin with. For these more complicated situations, there is a range of Markov-chain-based approaches which allow us to generate the required random samples, one of them being the Langevin approach we discuss below. With the random field configurations obtained from the Langevin process provides, we can use the idea of importance sampling to calculate the observables of a simulated system.

2.2 Stochastic Quantization: The Langevin Method

The Langevin method for solving quantum field theories, or *stochastic quantization*, is a method that allows us to calculate quantum mechanical solutions of physical systems. In that regard, it is a quantization method and equivalent to solving the Schrödinger equation or path integral of a system, although it is intimately connected to the latter. From the standpoint of Monte-Carlo integration, the Langevin method performs the importance sampling and provides us with samples of field configurations that we can use to calculate observables. The Langevin method for solving quantum field theories was originally discovered by Parisi and Wu [40], and an extensive review of the subject can be found in Ref. [41]. In this review, stochastic quantization is described as viewing a Euclidean field theory as the equilibrium limit of a statistical system that is coupled to a thermal reservoir. This comparison eludes to the historical origin of the Langevin equation that lies in the description of the Brownian motion of particles [42] (translated from French to English in Ref. [43]). In its original form, the Langevin equation read

$$m\frac{\mathrm{d}\mathbf{v}(t)}{\mathrm{d}t} = -\alpha\,\mathbf{v}(t) + \eta(t)\,. \tag{2.20}$$

It describes the Brownian motion of a particle with mass m and velocity $\mathbf{v}(t)$. In this description, the velocity of the particle is altered by two effects: the friction the particle experiences in its surrounding environment, described with the parameter α, and a stochastic force vector $\eta(t)$. The stochastic noise term models the random collisions a particle may experience in a fluid; a collision with another particle can momentarily change the velocity of the described particle.

To apply this general idea to quantum field theories, we consider a theory described by a Euclidean action $S[\phi]$ that is a functional of some real-valued quantum field ϕ. To formulate a Langevin equation for this system that models a stochastic process, we need some notion of time along which we can evolve the system. To this end, we extend the domain of the quantum field by a new fictitious, non-physical Langevin time t_{CL}:

$$\phi(x) \to \phi(x, t_{\mathrm{CL}})\,, \tag{2.21}$$

wherein x collectively describes the physical spacetime coordinates of the quantum field and we use natural units of $\hbar = k_B = 1$, as we do throughout this work. The Langevin equation of stochastic quantization then reads

$$\boxed{\frac{\partial\phi(x, t_{\mathrm{CL}})}{\partial t_{\mathrm{CL}}} = -\frac{\delta S[\phi]}{\delta\phi(x, t_{\mathrm{CL}})} + \eta(x, t_{\mathrm{CL}})}\,. \tag{2.22}$$

To obtain the correct results, the noise in Eq. (2.22) has to obey the following relations:

$$\begin{aligned} \langle\eta(x, t_{\mathrm{CL}})\rangle_\eta &= 0\,, \\ \langle\eta(x_1, t_{\mathrm{CL},1})\eta(x_2, t_{\mathrm{CL},2})\rangle_\eta &= 2\,\delta^{(n)}(x_1 - x_2)\,\delta(t_{\mathrm{CL},1} - t_{\mathrm{CL},2})\,, \end{aligned} \tag{2.23}$$

with n being the number of spacetime coordinates in x and the average being defined as

$$\langle\bullet\rangle_\eta = \frac{\int D\eta\ (\bullet)\ \exp\left(-\frac{1}{4}\int \mathrm{d}^n x\,\mathrm{d}t_{\mathrm{CL}}\,\eta^2(x, t_{\mathrm{CL}})\right)}{\int D\eta\ \exp\left(-\frac{1}{4}\int \mathrm{d}^n x\,\mathrm{d}t_{\mathrm{CL}}\,\eta^2(x, t_{\mathrm{CL}})\right)}\,. \tag{2.24}$$

In Sec. 2.2.3, we discuss a physical motivation for these requirements for the noise and an exact derivation can be found in the review [41]. The $\delta(t_{\mathrm{CL},1} - t_{\mathrm{CL},2})$ term in the second correlation function in Eq. (2.23) is of particular interest. It states that there is no autocorrelation in the

noise; the value of the noise term at any given Langevin time does not depend on its value at any other Langevin time. By this property, a Langevin process constitutes a Markov chain.

To illustrate the calculation of observables with stochastic quantization, we consider the calculation of an observable O in the path integral formalism. We have

$$O = \langle \mathcal{O}_O(\phi) \rangle = \frac{\int D\phi \; \mathcal{O}_O(\phi) \; e^{-S[\phi]}}{\int D\phi \; e^{-S[\phi]}} \,, \tag{2.25}$$

with an appropriate expression $\mathcal{O}_O(\phi)$ of the field ϕ that leads to the desired observable. In this expression, we can identify a probability density functional

$$P[\phi] = \frac{e^{-S[\phi]}}{\int D\phi \; e^{-S[\phi]}} \,, \tag{2.26}$$

which allows us to rewrite Eq. (2.25) as

$$O = \int D\phi \; P[\phi] \; \mathcal{O}_O(\phi) \,, \tag{2.27}$$

in analogy to Eq. (2.13) from Sec. 2.1.1 on importance sampling.

With Langevin processes $\phi(x, t_{\mathrm{CL}})$ for $t_{\mathrm{CL}} \in [0, t_{\mathrm{CL,max}}]$ that are the solutions of the Langevin equation, we have to make use of the dependence on the Langevin time to calculate observables. To this end, we define the Langevin time average

$$\langle F(\phi) \rangle_{\mathrm{CL}} = \frac{1}{t_{\mathrm{CL,max}}} \int_0^{t_{\mathrm{CL,max}}} \mathrm{d}t_{\mathrm{CL}} \; F(\phi(x, t_{\mathrm{CL}})) \tag{2.28}$$

over an expression F that depends on a Langevin process $\phi(x, t_{\mathrm{CL}})$. We find that in the limit $t_{\mathrm{CL,max}} \to \infty$, the Langevin time average becomes equal to the path integral weighted by the probability distribution $P[\phi]$ in Eq. (2.27):

$$\lim_{t_{\mathrm{CL,max}} \to \infty} \langle F(\phi) \rangle_{\mathrm{CL}} = \int D\phi \; P[\phi] \; F(\phi) \,. \tag{2.29}$$

A proof of this relation can be found in the reviews [41, 44]. This allows us to calculate an observable O by solving the Langevin equation up to a sufficiently large Langevin time $t_{\mathrm{CL,max}}$:

$$O \approx \langle \mathcal{O}_O(\phi) \rangle_{\mathrm{CL}} \,, \tag{2.30}$$

using the Langevin process $\phi(x, t_{\mathrm{CL}})$ we obtain from the solution of the Langevin equation. As such, solving the Langevin equation has provided us with a sample of field configurations $\phi(x, t_{\mathrm{CL}})$, one field configuration $\phi(x)$ for every point in Langevin time t_{CL}, that follows the probability distribution $P[\phi]$ without the need of solving the path integral explicitly.

2.2.1 Discretizing the Langevin Equation

In general, it is not possible to solve the Langevin equation for continuous time because this requires an analytic treatment of a stochastic differential equation. In practice, we, therefore, discretize the Langevin equation and calculate the Langevin process as a sequence of field configurations at discrete points in Langevin time. To this end, we introduce a Langevin time spacing δt_{CL} and consider the discrete Langevin time steps

$$t_{\mathrm{CL}}^{(i)} = i \cdot \delta t_{\mathrm{CL}} \,, \tag{2.31}$$

for $i \in \{0, ..., N_{\mathrm{CL}}\}$ with

$$t_{\mathrm{CL}}^{(N_{\mathrm{CL}})} = N_{\mathrm{CL}} \cdot \delta t_{\mathrm{CL}} = t_{\mathrm{CL,max}} \,. \tag{2.32}$$

We discretize the Langevin equation by using the Euler method. The first step in this method is to integrate the continuous Langevin equation (2.22) over one of the discrete Langevin time steps from $t_{\mathrm{CL}}^{(i)}$ to $t_{\mathrm{CL}}^{(i+1)} = t_{\mathrm{CL}}^{(i)} + \delta t_{\mathrm{CL}}$:

$$\int_{t_{\mathrm{CL}}^{(i)}}^{t_{\mathrm{CL}}^{(i+1)}} \mathrm{d}t_{\mathrm{CL}} \, \frac{\partial \phi(x, t_{\mathrm{CL}})}{\partial t_{\mathrm{CL}}} = -\int_{t_{\mathrm{CL}}^{(i)}}^{t_{\mathrm{CL}}^{(i+1)}} \mathrm{d}t_{\mathrm{CL}} \, \frac{\delta S}{\delta \phi(x, t_{\mathrm{CL}})} + \int_{t_{\mathrm{CL}}^{(i)}}^{t_{\mathrm{CL}}^{(i+1)}} \mathrm{d}t_{\mathrm{CL}} \, \eta(t_{\mathrm{CL}}) \,. \tag{2.33}$$

The next step is to expand each term around $\delta t_{\mathrm{CL}} = 0$. In the Euler method, we essentially approximate the integrals by assuming the value of the integrand to be constant over the integration region and multiplying it with its length δt_{CL}. We find

$$\phi(x, t_{\mathrm{CL}}^{(i+1)}) - \phi(x, t_{\mathrm{CL}}^{(i)}) = -\frac{\delta S}{\delta \phi(x, t_{\mathrm{CL}}^{(i)})} \delta t_{\mathrm{CL}} + \int_{t_{\mathrm{CL}}^{(i)}}^{t_{\mathrm{CL}}^{(i+1)}} \mathrm{d}t_{\mathrm{CL}} \, \eta(t_{\mathrm{CL}}) \,. \tag{2.34}$$

One may notice that we have not applied the approximation of the integral to the noise term. That is because the noise function $\eta(t_{\mathrm{CL}})$ features a δ-correlation in Langevin, see Eq. (2.23), and a rigorous mathematical treatment of a noise function with a δ-correlation is non-trivial. A well-defined treatment requires us to use a so-called *Wiener process*

$$W(t_{\mathrm{CL}}) = \int_0^{t'_{\mathrm{CL}}} \mathrm{d}t_{\mathrm{CL}} \, \eta(t'_{\mathrm{CL}}) \,, \tag{2.35}$$

which is an integral over the δ-correlated noise function $\eta(t_{\mathrm{CL}})$ and features the correlations

$$\begin{aligned} \langle W(t_{\mathrm{CL}}) \rangle &= 0 \,, \\ \langle W(t_{\mathrm{CL},1}) W(t_{\mathrm{CL},2}) \rangle &= \min(t_{\mathrm{CL},1}, t_{\mathrm{CL},2}) \,. \end{aligned} \tag{2.36}$$

With the Wiener process $W(t_{\mathrm{CL}})$, we have

$$\int_{t_{\mathrm{CL}}^{(i)}}^{t_{\mathrm{CL}}^{(i+1)}} \mathrm{d}t_{\mathrm{CL}} \, \eta(t_{\mathrm{CL}}) = W(t_{\mathrm{CL}}^{(i+1)}) - W(t_{\mathrm{CL}}^{(i)}) \tag{2.37}$$

and using stochastic calculus and the correlations in Eq. (2.36), see also Ref. [41], we find

$$W(t_{\mathrm{CL}}^{(i+1)}) - W(t_{\mathrm{CL}}^{(i)}) = \sqrt{2\delta t_{\mathrm{CL}}} \, \tilde{\eta}(t_{\mathrm{CL}}^{(i)}) \,. \tag{2.38}$$

For each point in Langevin time $t_{\mathrm{CL}}^{(i)}$, the number $\tilde{\eta}(t_{\mathrm{CL}}^{(i)})$ is a random number that is drawn from a normal distribution with a standard deviation of one, also known as a standard normal distribution. This is fortunate for numerical applications, as such numbers can very easily be generated by pseudo-random number generators on computers.

Inserting the integrated noise term into Eq. (2.34), we obtain the discretized Langevin equation

$$\boxed{\phi(x, t_{\mathrm{CL}}^{(i+1)}) = \phi(x, t_{\mathrm{CL}}^{(i)}) - \frac{\delta S}{\delta \phi(x, t_{\mathrm{CL}}^{(i)})} \delta t_{\mathrm{CL}} + \sqrt{2\delta t_{\mathrm{CL}}} \, \tilde{\eta}(t_{\mathrm{CL}}^{(i)})} \,. \tag{2.39}$$

Note that we also reordered the equation to find all terms evaluated at Langevin time step $t_{\mathrm{CL}}^{(i)}$ on the right-hand side. In this form, the Markov property of the Langevin process is clearly visible;

the field configuration at the next step $\phi(x, t_{\mathrm{CL}}^{(i+1)})$ depends only on the field configuration at the current step $\phi(x, t_{\mathrm{CL}}^{(i)})$. We also emphasize that we have only discretized the Langevin time, not the spacetime coordinates x, and the drift term still features a functional derivative. In practice, we generally also discretize the spacetime domain of the field, and the single Langevin equation becomes a set of Langevin equations, one for each point in spacetime. The functional derivative in the drift term then takes the form of a partial derivative with respect to the value of the field at the given point in spacetime.

2.2.2 Zero-Dimensional Example

To illustrate the principles of stochastic quantization, we study a simple zero-dimensional example which is also presented in Ref. [44]. For this example, we consider a system with the action

$$S(\phi) = \frac{\mu}{2}\phi^2 + \frac{\lambda}{4!}\phi^4 \tag{2.40}$$

and a real-valued quantum "field" ϕ. Because a zero-dimensional quantum field is just a single number, the action is a function of ϕ rather than a functional, and we perform no space and no time integral in the action. Beyond that, because the imaginary time represents the temperature in statistical calculations, we cannot define a temperature for this system. Path integrals in zero dimensions are just ordinary integrals over a single integration variable, as we can see in the partition function:

$$Z = \int \mathrm{d}\phi \; e^{-S(\phi)} \,. \tag{2.41}$$

To solve this system using stochastic quantization, we first extend the domain of the quantum field to include the Langevin time:

$$\phi \to \phi(t_{\mathrm{CL}}) \,, \tag{2.42}$$

and formally define the Langevin process as the sequence

$$\left(\phi^{(i)} = \phi(t_{\mathrm{CL}}^{(i)}) \;\middle|\; i = 0, ..., N_{\mathrm{CL}}\right) \,. \tag{2.43}$$

To determine the Langevin equation, we calculate the drift

$$\frac{\partial S}{\partial \phi} = \mu\phi + \frac{\lambda}{3!}\phi^3 \tag{2.44}$$

and replace the field values with the Langevin-time-dependent version:

$$\frac{\partial S}{\partial \phi} \to \left.\frac{\partial S}{\partial \phi}\right|_{\phi^{(i)}} = \mu\phi^{(i)} + \frac{\lambda}{3!}\left(\phi^{(i)}\right)^3 \,. \tag{2.45}$$

Inserting the drift in the discretized Langevin equation (2.39), we find

$$\phi^{(i+1)} = \phi^{(i)} - \left(\mu\phi^{(i)} + \frac{\lambda}{3!}\left(\phi^{(i)}\right)^3\right)\delta t_{\mathrm{CL}} + \sqrt{2\delta t_{\mathrm{CL}}}\,\tilde{\eta}(t_{\mathrm{CL}}^{(i)}) \,. \tag{2.46}$$

This equation can now be used to obtain the Langevin process $(\phi^{(i)})$.

Figure 2.2 shows the action $S(\phi)$, as well as the probability distribution $e^{-S(\phi)}/Z$ of the path integral together with a histogram of the Langevin process $\phi^{(i)}$ for $t_{\mathrm{CL,max}} = 10\,000$ and $\delta t_{\mathrm{CL}} = 0.1$. The plots are shown for two sets of model parameters. In the case of $(\mu, \lambda) = (1.0, 0.4)$ the action features a single minimum at $\phi = 0$ and in the case of $(\mu, \lambda) = (-1.0, 0.4)$ the action features two minima at $\phi \approx \pm 3.87$. For the system with a single minimum in its action, the

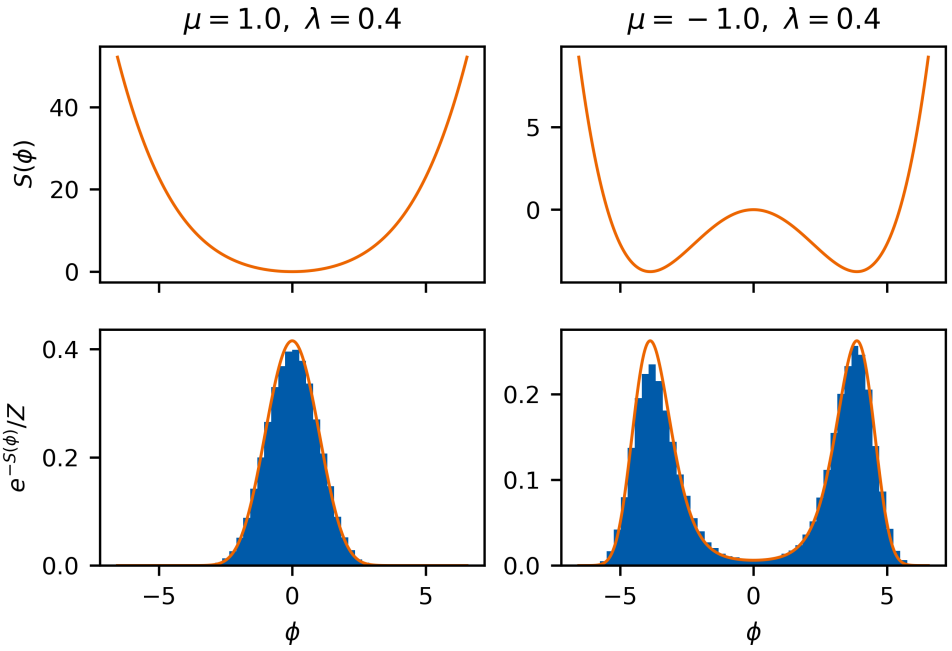


Figure 2.2: Stochastic quantization applied to the example described by the action in Eq. (2.40). The left-hand column shows a system with parameters $(\mu,\lambda) = (1.0, 0.4)$ and the right-hand column shows a system with $(\mu,\lambda) = (-1.0, 0.4)$. The plots in the upper row show the action of the system, and the plots in the lower row show normalized histograms of the Langevin processes $(\phi^{(i)})$ in blue together with the exact probability densities in orange.

Langevin method works as expected. The system with two minima in its action, on the other hand, allows us to study some problematic behavior, which we need to be aware of in practice. We can see that the histograms of the Langevin processes indeed follow the probability distribution of the path integral. Thus, loosely speaking, stochastic quantization does what it is supposed to do.

Next, we calculate some observables for our example systems. We begin with the mean value of the field $\langle\phi\rangle_{\mathrm{CL}}$. We are using a discrete Langevin time, so the integral in the Langevin average $\langle\bullet\rangle_{\mathrm{CL}}$ is given by a discrete sum:

$$\langle\phi\rangle_{\mathrm{CL}} = \frac{1}{\delta t_{\mathrm{CL}} N_{\mathrm{CL}}} \sum_{i=0}^{N_{\mathrm{CL}}} \delta t_{\mathrm{CL}}\, \phi^{(i)} = \frac{1}{N_{\mathrm{CL}}} \sum_{i=0}^{N_{\mathrm{CL}}} \phi^{(i)} = \langle\phi^{(i)}\rangle_i \,. \tag{2.47}$$

Hence, all we need to do here is to average the value of the field ϕ at all Langevin time steps. Note that from the symmetry of the action, we know that the exact result for the mean value of the field has to be zero for all sets of parameters. Another observable that is only slightly more complicated is the mean-squared field value $\langle\phi^2\rangle_{\mathrm{CL}}$. It can be used as a measure of the width of the probability distribution $e^{-S(\phi)}/Z$, and we can compute it as

$$\langle\phi^2\rangle_{\mathrm{CL}} = \frac{1}{N_{\mathrm{CL}}} \sum_{i=0}^{N_{\mathrm{CL}}} \left(\phi^{(i)}\right)^2 \,. \tag{2.48}$$

The exact numerical values for this observable are $\langle\phi^2\rangle \approx 0.82$ for $(\mu,\lambda) = 1.0, 0.4$ and $\langle\phi^2\rangle \approx 13.67$ for $(\mu,\lambda) = -1.0, 0.4$. In Fig. 2.3, we can see the Langevin process itself, as well as the con-

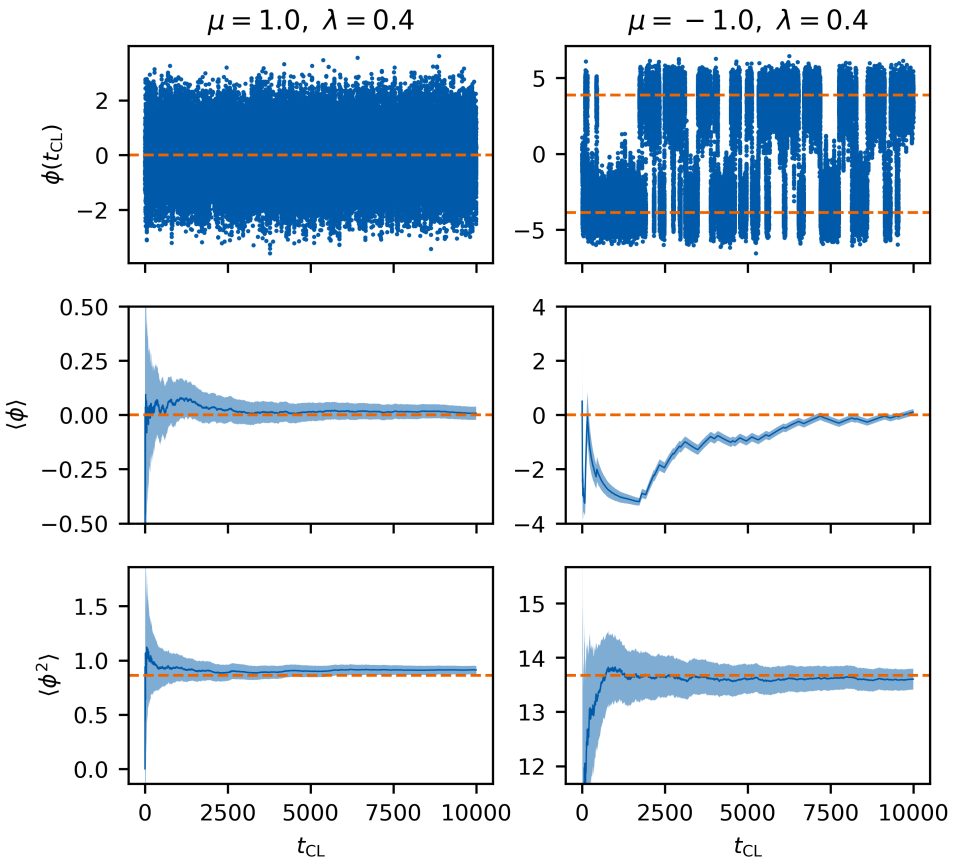


Figure 2.3: Stochastic quantization applied to the example described by the action in Eq. (2.40). The left-hand column shows a system with parameters $(\mu,\lambda) = (1.0, 0.4)$ and the right-hand column shows a system with $(\mu,\lambda) = (-1.0, 0.4)$. The first row shows the Langevin processes in blue, with the dashed orange lines indicating the minima of the action for the respective parameters. The second row shows the convergence of the mean value observable with the dashed orange line representing the exact value. The third row shows the convergence of the mean squared value observable, with the dashed orange line representing the exact value. The shaded blue areas in the second and third rows represent an uncertainty estimate for the observable. We discuss the uncertainty estimate in more detail in Sec. 2.2.4.

vergence of the observables $\langle\phi\rangle$ and $\langle\phi^2\rangle$. To obtain the convergence graphs of the observables, we calculate the Langevin time averages partially; for a given Langevin time t_{CL}, with

$$n = \left\lfloor \frac{t_{CL}}{t_{CL,max}} \right\rfloor \tag{2.49}$$

and the floor function $\lfloor\bullet\rfloor$, we have

$$\langle\phi\rangle_{CL}(t_{CL}) = \frac{1}{n}\sum_{i=0}^{n}\phi^{(i)}\,. \tag{2.50}$$

In the Langevin processes in Fig. 2.3, we see that for the system in the left column with a single minimum in the action, the Langevin process essentially performs a random walk around that minimum. For the system in the right column of the figure, the Langevin process shows two distinct types of behavior. While $\phi^{(i)}$ is close to one of the minima of the action, it performs the

same kind of random walk around it as the system in the left column. At some points in Langevin time, however, the random noise manages to push the system into the other minimum, where it then starts to perform a random walk around it again. We can observe this behavior clearly in the figure.

In the second row of Fig. 2.3, we can see the convergence of the observable $\langle\phi\rangle$. As noted above, the Langevin method only gives us the exact value for an observable in the limit $t_{\mathrm{CL,max}} \to \infty$. As such, all observables for finite values of $t_{\mathrm{CL,max}}$ are merely estimates of the exact value of the observable and converge to it with increasing values of $t_{\mathrm{CL,max}}$. The shaded bands behind the curve represent estimates of the uncertainty of the observable estimates. In Sec. 2.2.4, we discuss how we can obtain such uncertainty estimates, but for now, we focus on the convergence of the observable estimate itself. For the system depicted in the left-hand column of the figure, we can see the expected behavior: after the Langevin process has accumulated a few sample field values, we can estimate the observable somewhere around the exact value, and as we follow the Langevin process further, the estimate for the observable converges to the exact result. For the system depicted in the right-hand column of the figure, the story is a bit different. Until around $t_{\mathrm{CL}} \approx 2000$, the estimate of the observable seems to converge to a wrong value around where one of the minima of the action of the system lies. When studying the Langevin process in the first row of the figure, we can see what is happening: the Langevin process is stuck in the left minimum of the action because the noise is never strong enough to push it over the central barrier into the other minimum of the action. Because of this, the Langevin method "does not see" that the other minimum even exists, and the estimate confidently converges to the wrong value. That "confidence" is also expressed in the uncertainty band for that part of the graph. In Sec. 2.2.4, we discuss the mathematical origin of this problematic behavior in more detail and how we can detect and avoid it in practice. In the third row of Fig. 2.3, we see a convergence of the observable $\langle\phi^2\rangle$, and for this observable, both systems behave as expected. This may come as a surprise, as for the system in the right-hand column of the figure, it is very difficult to determine the mean value of the field $\langle\phi\rangle$, so it may seem paradoxical that the mean squared value $\langle\phi^2\rangle$ can be determined without problems. In fact, we can only estimate $\langle\phi^2\rangle$ because the two minima of the action of that system are symmetric around $\phi = 0$; whether the system is stuck in the left or the right minimum looks the same when we square the field values.

With this zero-dimensional example understood, we should now be equipped to discuss more intricate aspects of stochastic quantization.

2.2.3 Physical Motivation of the Langevin Method

With an understanding of the Langevin method for zero-dimensional "field" theories, we now study a relation that provides us with a physical intuition of how and why the method works. We also motivate the role of the magnitude of the noise in the Langevin equation. We begin our considerations with the continuous Langevin equation (2.22) but without the noise term:

$$\frac{\partial\phi(x,t_{\mathrm{CL}})}{\partial t_{\mathrm{CL}}} = -\frac{\delta S[\phi]}{\delta\phi(x,t_{\mathrm{CL}})}\,. \tag{2.51}$$

Specifically, we are interested in what field configurations the solutions of this modified Langevin equation converge to. The process is converged when $\phi(x,t_{\mathrm{CL}})$ no longer changes with Langevin time, i.e.,

$$\frac{\partial\phi(x,t_{\mathrm{CL}})}{\partial t_{\mathrm{CL}}} = 0\,. \tag{2.52}$$

Comparing this with Eq. (2.51), we see that this modified Langevin equation has a fixed point at field configurations with vanishing drift:

$$\frac{\delta S[\phi]}{\delta \phi(x, t_{\mathrm{CL}})} = 0\,. \tag{2.53}$$

By Hamilton's principle, such field configurations are classical solutions of the system. This connection can nicely be demonstrated in discretized Langevin time with a zero-dimensional field theory with a real-valued quantum "field" ϕ and action function(al) $S(\phi)$. For this system, the discretized Langevin equation without noise reads

$$\phi^{(i+1)} = \phi^{(i)} - \left.\frac{\partial S}{\partial \phi}\right|_{\phi^{(i)}} \delta t_{\mathrm{CL}}\,. \tag{2.54}$$

We now define an adaptive, non-constant step in Langevin time

$$\delta t_{\mathrm{CL}}(t_{\mathrm{CL}}) = \left(\left.\frac{\partial^2 S}{\partial \phi^2}\right|_{\phi^{(i)}} \right)^{-1}\,. \tag{2.55}$$

and rename

$$\begin{aligned} \phi^{(i)} &\to x_i\,, \\ \frac{\partial S}{\partial \phi} &\to f\,. \end{aligned} \tag{2.56}$$

The modified Langevin equation then reads

$$x_{i+i} = x_i - \left.\frac{f}{f'}\right|_{x_i}\,. \tag{2.57}$$

This is precisely the iteration prescription of Newton's method for finding a zero of the function f, which in our case means finding a classical solution of the system. In this sense, in the absence of the noise term, the Langevin equation explicitly pushes a system into its classical solution.

If the Langevin equation with noise gives us the quantum solution of a system and the Langevin equation without noise gives us the classical solution of a system, then the noise must represent the quantum mechanical aspects of nature. Just as the path integral can be understood as a generalization of Hamilton's principle that leads to quantum solutions, the Langevin equation with noise is a generalization of the search for a stationary point of the action that leads to quantum solutions.[2] From this point of view, it also makes sense why we need a specific magnitude of noise for the Langevin method to produce the correct quantum results; the magnitude of the noise represents the "amount of quantumness" in our system. In the example in the right-hand column of Fig. 2.3, the magnitude of noise determines how likely the Langevin process is to "tunnel" through the barrier between the two minima of the action. In this sense, we need to fix the magnitude of the noise to the value that reproduces the "amount of quantumness" we observe in nature. This rationale can serve us as a phenomenological motivation, while the rigorous derivation of the noise correlation functions can be found in the review [41].

[2] There is a certain beauty in the fact that, in the Langevin method, the true randomness of quantum mechanics is included explicitly in the form of random numbers in its calculations.

2.2.4 Uncertainty Estimation Monte-Carlo Processes

When we use the Langevin equation of a system to calculate an observable O, we obtain a sample $\{\mathcal{O}_O^{(i)}\}$ of N observable sample points, which are calculated at each step in Langevin time. We know that we can obtain an estimate for the value of the observable by calculating the arithmetic mean

$$\langle O \rangle = \frac{1}{N} \left(\sum_{i=1}^{N} \mathcal{O}_O^{(i)} \right) . \tag{2.58}$$

To determine how precise this estimate is, we also need to calculate the standard deviation of the sample

$$\sigma_O = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(\mathcal{O}_O^{(i)} - \langle O \rangle \right)^2} . \tag{2.59}$$

This specific expression for the standard deviation accounts for the fact that we also use the data to estimate the mean. For a given Langevin process, the standard deviation of the observable sample converges to a constant value with increasing sample size. It is a measure of the width of the distribution of the sample and depends on the numerical magnitude of noise in the Langevin equation. We can use the standard deviation of the observable sample to calculate the standard deviation of the mean

$$\Delta O = \frac{\sigma_O}{\sqrt{N}} . \tag{2.60}$$

This quantity gives us the desired precision of our estimate of the mean $\langle O \rangle$. We can see that with increasing sample size, the uncertainty of the estimate of the mean decreases by a factor of $\sqrt{N}$. The uncertainty bands in our zero-dimensional example in Fig. 2.3 are given by such standard deviations of the mean. However, the uncertainty bands in the figure are manually scaled up by a factor of ten. With this basic method, we massively underestimate the uncertainty of our estimates for observables. The cause of this is *autocorrelation* in the data, i.e., the fact that subsequent sample points are not independent. The presence of autocorrelation is quite expected in Markov chains. Because any given sample point is generated using only the previous sample point and random numbers, it takes the state of the Markov chain a few steps to become independent of a previous state. We clearly see this behavior in the estimation of the mean value $\langle \phi \rangle$ in the right-hand column of Fig. 2.3. Up until a Langevin time of $t_{\mathrm{CL}} \approx 2000$, the system is stuck in the left minimum of the action. Until the noise is strong enough to push the system over the central barrier, subsequent samples remain close to the left minimum of the action, and we observe strong autocorrelation. For uncertainty estimation, Eq. (2.60), in fact, only holds if the sample points are uncorrelated. Therefore, estimating the standard deviation of estimates of observables with this method is generally not valid. The correct estimate is given by [45]

$$\Delta O = \sqrt{\frac{1 + \tau_a}{N}} \, \sigma_O , \tag{2.61}$$

which is the result for uncorrelated samples scaled up by a factor of $\sqrt{1 + \tau_a}$ with the *integrated autocorrelation time* τ_a. This quantity is given by

$$\tau_a = \frac{\sum_{i=1}^{N_{\mathrm{CL}}} \left(\langle \mathcal{O}_O^{(j)} \mathcal{O}_O^{(j+i)} \rangle_j - \langle O \rangle^2 \right)}{\langle O^2 \rangle - \langle O \rangle^2} \tag{2.62}$$

and tells us how long the correlation of sample points persists within a Markov chain. In this form, it is measured in the number of sample points rather than Langevin time. The integrated autocorrelation time can be calculated directly, but this process can often be tricky. In the present

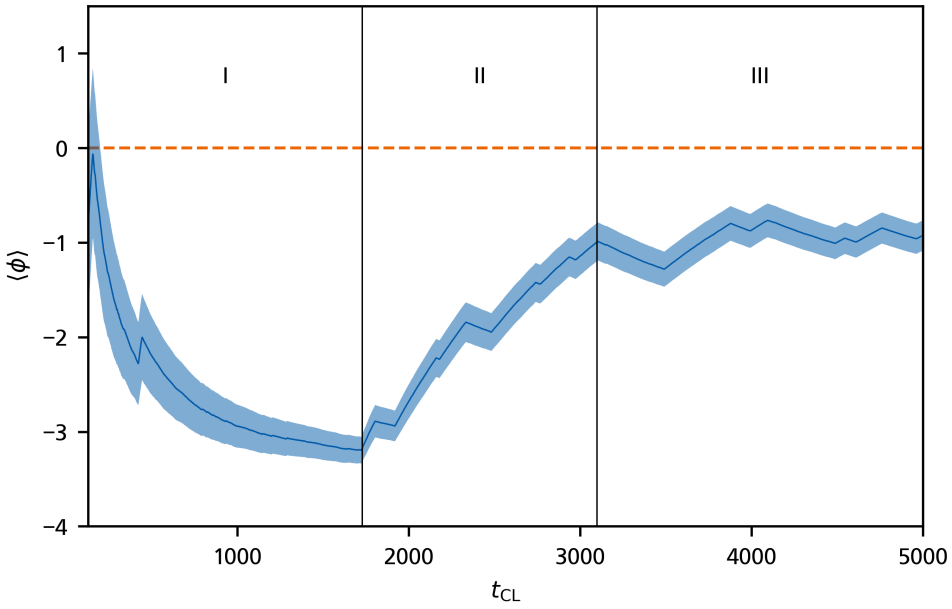


Figure 2.4: Evolution of the estimates for the observable $\langle\phi\rangle$ for the zero-dimensional system from Sec. 2.2.2 with parameters $(\mu,\lambda) = (-1.0, 0.4)$. The graph is divided into sections I, II, and III. In sections I and II, the evolution of the estimate is dominated by the left and right minimum of the action, respectively. In section III, the evolution is balanced between the two minima. The dashed orange line represents the exact value of $\langle\phi\rangle = 0$, and the shaded area represents the unbiased standard deviation of the mean for the observable, manually scaled up by a factor of ten.

work, we choose an arguably more elegant method to obtain a correct, unbiased uncertainty estimate for estimated observables: Jackknife resampling. To motivate this method, we again consider the estimation of the uncertainty of $\langle\phi\rangle$ in our zero-dimensional example with the parameters $(\mu,\lambda) = (-1.0, 0.4)$. Figure 2.4 shows the estimate for the observable with increasing simulation time t_{CL}, as we have already seen in Fig. 2.3. The uncertainty bands are again given by the biased standard deviation of the mean in Eq. (2.59) and are manually scaled up by a factor of ten. In Fig. 2.4, we have separated the graph into three sections. In sections I and II, the convergence of the observable is predominantly determined by one of the minima of the action, and the sample demonstrates strong autocorrelation in these sections. In section III, the system moves frequently between the two minima, and the convergence of the estimate of the observable is balanced between them. The autocorrelation in section III is considerably smaller than in sections I and II. We can indeed identify this just by looking at the convergence graph together with the uncertainty estimate band. At the beginning of section III, the uncertainty band defines a range in which we likely find the true value of the observable. Throughout section III, the estimate of the observable value never strays too far from that initial uncertainty region. In sections I and II, on the other hand, the estimate moves far outside the uncertainty range we find at the beginning of each of the sections. That indicates that the initial uncertainty estimates in these sections are too small because autocorrelation is present, and we need greater uncertainty estimates that account for said autocorrelation. This rationale of considering different subsections of the sample can be formalized into the Jackknife resampling method. In this work, we considered the blocked version of the method, reviewed in Ref. [46]. The Jackknife method is a *resampling* method, meaning it works by estimating statistical parameters on altered versions of the sample. For the blocked Jackknife method, we divide the sample into N_b blocks, each con-

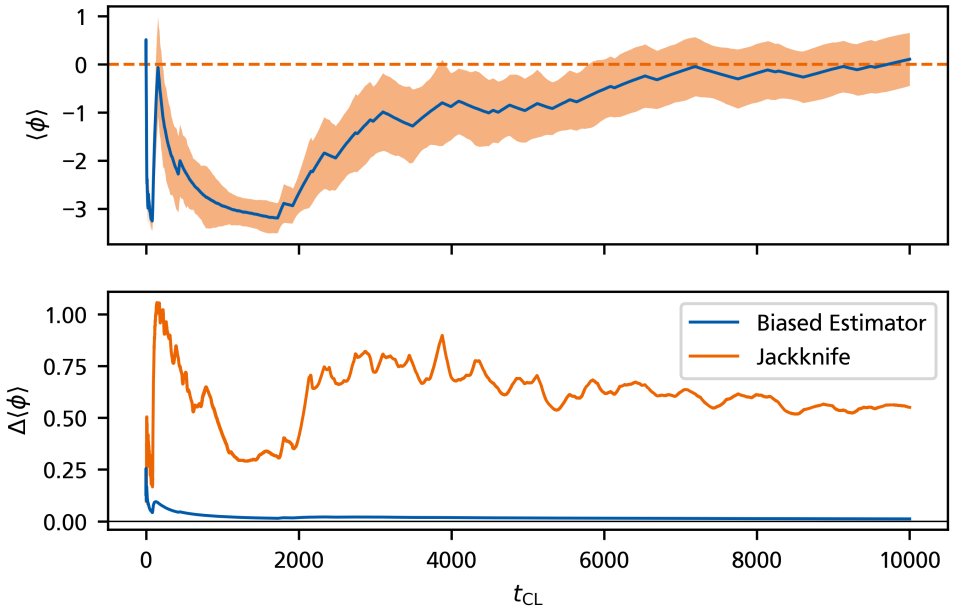


Figure 2.5: Evolution of the estimates for the observable $\langle\phi\rangle$ for the zero-dimensional system from Sec. 2.2.2 with parameters $(\mu,\lambda) = (-1.0, 0.4)$. The upper plot shows the estimate of the observable with an error band which is obtained through the Jackknife method with $N_b = 10$ blocks. The lower plot shows the evolution of uncertainty estimates of both the biased estimator in Eq. (2.60) and the Jackknife.

taining L_b sample points. If the total number of sample points in all blocks $N_b L_b$ is less than the total size of the sample N_{CL}, we discard sample points at the beginning of the Langevin process until N_{CL} is an integer multiple of L_b. The reason we discard sample points at the beginning of the process is that, depending on the system, the Langevin process can require a few samples to equilibrate. If that is the case, the first samples of the Langevin process contain little to no physical information. To estimate the uncertainty of the estimate for an observable O, for each block $b \in \{1, \ldots, N_b\}$ we calculate the mean of all sample points that are *not* inside that block:

$$j_b = \frac{1}{L_b}\left(\sum_{1\le i<(b-1)L_b} \mathcal{O}_O^{(i)} + \sum_{bL_b<i\le N_{\mathrm{CL}}} \mathcal{O}_O^{(i)}\right). \tag{2.63}$$

With these mean values, the Jackknife method estimates the unbiased standard deviation of the mean of the sample $\{\mathcal{O}_O^{(i)}\}$ as

$$\Delta O \approx \frac{1}{\sqrt{N_b}}\,\mathrm{std}\{j_b\}\,. \tag{2.64}$$

Figure 2.5 again shows the evolution of the estimate of the mean value $\langle\phi\rangle$ for our zero-dimensional example with parameters $(\mu,\lambda) = (-1.0, 0.4)$. This time, however, the uncertainty band of the graph is given by the Jackknife estimate with $N_b = 10$ rather than the biased estimate with Eq. (2.60). The lower subplot of the figure shows the evolution of both estimates in comparison. With the Jackknife estimate, we immediately see that the estimated uncertainty is generally considerably greater than the biased estimate. This makes sense because, as discussed above, the data features considerable autocorrelation. As the process approaches the time $t_{\mathrm{CL}} \approx 2000$, with the system stuck in the left minimum of the action for the most part, the uncertainty estimate from the Jackknife method decreases, just as we have seen with the biased estimate. This is to be expected because, up to that point, the process has not "seen" much of the right half of the action

and has no reason for a large uncertainty. It is not until $t_{\mathrm{CL}} \approx 2000$ that we see the unbiased nature of the Jackknife method in action. Once the system moves into the other minimum, the estimate of the mean value moves considerably outside the uncertainty band it has at that point, see Fig. 2.4. The Jackknife picks up on that and the uncertainty estimate increases as a result, something we do not observe with the biased estimator. As soon as the Langevin process is no longer clearly dominated by only one of the minima of the action, the uncertainty estimate of the Jackknife stabilizes and proceeds to decay, demonstrating the expected $1/\sqrt{N_{\mathrm{CL}}}$ behavior. For this example, we chose a number of blocks of $N_b = 10$. Generally, we want this number to be as low as possible, as the computational effort increases with increasing number of blocks. Making it too small, on the other hand, causes the estimate of the uncertainty to fluctuate more. In practice, we need to find a compromise that is acceptable in both steadiness of the estimate and computational effort and generally depends on the problem at hand.

In this example, we have seen that the Jackknife method offers us a computationally inexpensive way to obtain an unbiased estimate of the uncertainty of observables. There is, however, even more, it can do for us. Because it provides us with an unbiased estimate of the standard deviation of the mean of a sample, and because Eq. (2.61) holds, we can even use the Jackknife to obtain an estimate of the integrated autocorrelation time τ_a. For a sample of N observable values $\{\mathcal{O}_O^{(i)}\}$ we find

$$\tau_a \approx N \left(\frac{\Delta O_{\mathrm{jn}}}{\sigma_O} \right)^2 - 1 \tag{2.65}$$

with the Jackknife estimate of the standard deviation of the mean ΔO_{jn} and the standard deviation of the sample σ_O, as given by Eq. (2.59). In this form, τ_a measures the number of steps rather than Langevin time. As discussed above, the integrated autocorrelation time measures how many steps the Markov chain needs to produce uncorrelated sample points again. As such, we can use this quantity to tune the parameters of the Langevin process such that it features as little autocorrelation as possible. After all, if new sample points present barely any new information, why should we bother calculating them? Concretely, we can use the integrated autocorrelation time to tune the Langevin time spacing δt_{CL}. As we see in the discretized Langevin equation (2.39), the amount of noise per step scales with $\sqrt{\delta t_{\mathrm{CL}}}$. Through this behavior, the process becomes more likely to feature larger changes per step with larger values of δt_{CL}. Of course, when doing this, we still need to ensure that δt_{CL} remains sufficiently small. This requirement of the Euler method we used to discretize the Langevin equation still has to be met.

In this work, we use the presented Jackknife method for estimating uncertainties, but for the sake of completeness, we also mention that other variants of the Jackknife exist, see Ref. [46]. Beyond the Jackknife method, there are also other resampling methods, such as *binning* [45] and the *bootstrap* method [47].

Systematic Uncertainties

The uncertainties we discussed in this section have their origin in the random nature of Monte Carlo methods. Therefore we call them *statistical uncertainties*. However, they are not the only source of uncertainty in our simulation. When we take a look at the estimate of $\langle \phi^2 \rangle$ in Fig. 2.3 again, we notice that, even though the statistical uncertainty of the estimate decreases at the end of the simulation, the estimate seems to converge to a value that is visibly different from the exact value. This discrepancy is caused by the presence of *systematic uncertainties* in the simulation. Systematic uncertainties are introduced whenever we make approximations to solve the system. For quantum field theories that involve space and time, one such approximation is the introduction of a spacetime lattice of finite size and finite spacings, as we do below in our pairing-field formalism. Even our simple zero-dimensional example already features two such sources

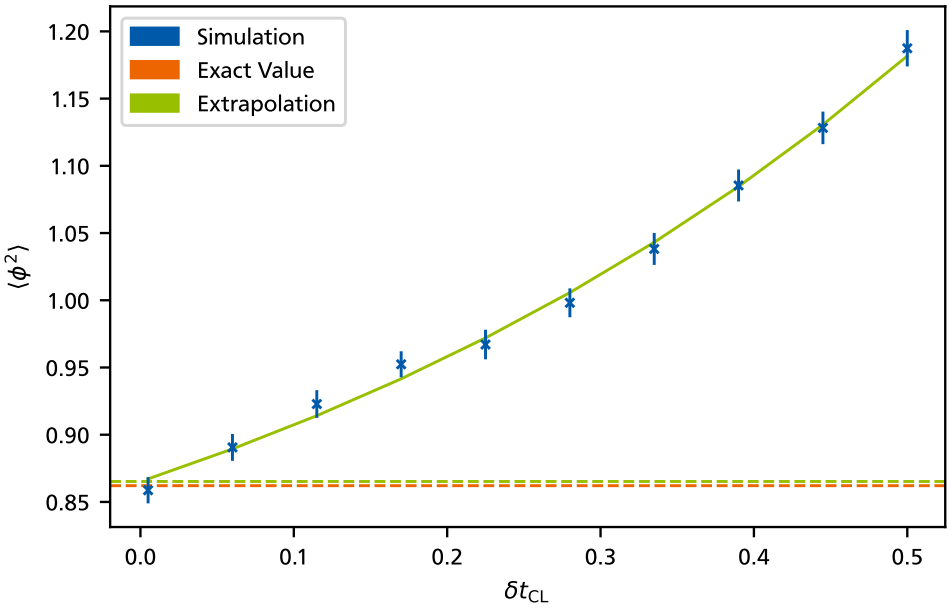


Figure 2.6: Correction of the systematic uncertainty introduced by finite values of δt_{CL} for the system in Sec. 2.2.2 with $(\mu, \lambda) = (1.0, 0.4)$. The dashed orange line represents the exact value of $\langle\phi^2\rangle$. The blue crosses mark simulation runs with uncertainties obtained from the Jackknife method. The green line shows a model fit to the simulation results, and the dashed green line represents the fit result at $\delta t_{\mathrm{CL}} = 0$. It features a $0.4\,\%$ deviation from the exact result.

of systematic uncertainty: the total amount of Langevin time in the simulation $t_{\mathrm{CL,max}}$ and the Langevin time spacing δt_{CL}. When we discretized the Langevin equation above, we used the Euler method that is only exact in the limit $\delta t_{\mathrm{CL}} \to 0$. Beyond that, we stated that the Langevin method only produces exact results for observables in the limit $t_{\mathrm{CL,max}} \to \infty$. Fortunately, we can remove systematic uncertainties from our results, albeit at the cost of increased computational effort. For the total amount of Langevin time $t_{\mathrm{CL,max}}$, we have instinctively already done that in our example. We can simply run the simulation with more Langevin time to get closer to the limit $t_{\mathrm{CL,max}} \to \infty$. This has the added benefit of also decreasing the statistical uncertainty as it scales with $1/\sqrt{t_{\mathrm{CL,max}}} \propto 1/\sqrt{N_{\mathrm{CL}}}$. Decreasing the systematic uncertainty related to the finite Langevin time spacing δt_{CL} is a bit less straightforward. We could just choose very small values for δt_{CL} but that would make the simulation far more computationally costly at constant $t_{\mathrm{CL,max}}$. Moreover, as we have seen in the zero-dimensional example with two minima in its action, smaller values δt_{CL} can lead to more autocorrelation in the Langevin process and potentially slow down the convergence of estimates of observables. So, instead of just decreasing δt_{CL}, we run the simulation multiple times with different values for δt_{CL}. Figure 2.6 shows this approach for the estimate of $\langle\phi^2\rangle$ in our zero dimensional example with $(\mu, \lambda) = (1.0, 0.4)$. We see that the simulation results approach the exact value with decreasing δt_{CL}, so we extrapolate the simulation results to $\delta t_{\mathrm{CL}} = 0$ by fitting a model to the data points. In the figure, we chose the exponential model

$$f(\delta t_{\mathrm{CL}}) = c_1 \cdot e^{c_2 \cdot \delta t_{\mathrm{CL}}} + c_3 \tag{2.66}$$

and obtained the extrapolated results as $f(\delta t_{\mathrm{CL}} = 0)$. The extrapolated result is shown in the figure and only features a $0.4\,\%$ deviation from the exact result. We could increase this deviation further by using more accurate simulations for the fit, but this example already illustrates the basic idea nicely.

2.2.5 The Sign Problem and Complex Langevin

So far, we have been focused on Monte-Carlo methods around some kind of probability distribution, such as the normalized weight e^{-S}/Z in the path integral. This normalized weight, however, can only be interpreted as a probability distribution as long as it is real-valued and non-negative. In fundamental descriptions of physical systems, we generally expect the action to be real-valued, leading to the normalized weight e^{-S}/Z being non-negative for all field configurations. However, there are transformations we can apply to our system that lead to an equivalent description of the system but cause the transformed action to take complex values. This leads to negative or even complex normalized weights e^{-S}/Z. This situation is called the *sign problem*, or *phase problem* in the case of complex path integral weights. It presents an obstacle for many Monte-Carlo methods that rely on the ability to interpret the path integral weights as probability distributions. Fortunately, the Langevin method is not affected by this obstacle, since it does not require non-negative or even real-valued path-integral weights. In fact, the path integral weights do not appear at all in the Langevin method; we merely compute a drift term as the variation of the action and follow where it leads us. As such, where the method is applicable, the sign problem does not prevent stochastic quantization. When we apply stochastic quantization in situations with complex actions, the method is called *Complex Langevin* (CL). Together with other approaches to the sign problem, it has recently been reviewed in Ref. [44]. Complex Langevin requires us to consider some details that do not enter the real-valued case, so we explore the method using a simple example based on the zero-dimensional field theory in Sec. 2.2.2.

Again, we consider a zero-dimensional quantum field theory described by the action

$$S(\phi) = \frac{\mu}{2}\phi^2 + \frac{\lambda}{4!}\phi^4 \,, \tag{2.67}$$

with the real-valued, zero-dimensional quantum "field" $\phi \in \mathbb{R}$ and real-valued parameters μ and λ. The partition function for this system is given by

$$Z = \int_{-\infty}^{\infty} \mathrm{d}\phi \; e^{-S(\phi)} \,. \tag{2.68}$$

Perhaps the most obvious way to turn this action into a complex quantity would be to use a complex value for one of the parameters μ and λ, but instead, we choose another approach that is motivated by a transformation we perform below on our physical system of interest. In the actual study of our system of interest, the quantum field represents fermions and, as such, is Grassmann-valued rather than real-valued. This makes it impossible to solve the path integral with a ϕ^4 term in the action directly as a Gaussian integral. To circumvent this problem, in Sec. 3.1.1, we perform a so-called *Hubbard-Stratonovich transformation* which removes the ϕ^4 term from the action and, instead, describes its effect in the form of a bosonic auxiliary field; we *bosonize* the theory. Due to the Grassmann-valued nature of fermionic fields, the auxiliary field we introduce below is generally complex-valued rather than real-valued. With this motivation, we perform the zero-dimensional equivalent of this Hubbard-Stratonovich transformation on our example system to obtain a complex action and solve the system using Complex Langevin. We begin by inserting a suitably chosen factor of one into the partition function:

$$\begin{aligned} Z &= \int \mathrm{d}\phi \; e^{-S(\phi)} \underbrace{\mathcal{N} \int \mathrm{d}\sigma \; e^{-\frac{\lambda}{4!}\sigma^2}}_{=1} \\ &= \mathcal{N} \int \mathrm{d}\phi \; \mathrm{d}\sigma \; e^{-S^*_{\mathrm{PB}}(\phi,\sigma)} \,. \end{aligned} \tag{2.69}$$

In this expression and for the remainder of this example we suppress the integral bounds, as all integrals range from $-\infty$ to ∞. The variable σ in this factor of one represents our newly introduced auxiliary field and the newly introduced action

$$S^*_{\mathrm{PB}}(\phi,\sigma) = \frac{\mu}{2}\phi^2 + \frac{\lambda}{4!}\phi^4 + \frac{\lambda}{4!}\sigma^2 \tag{2.70}$$

is called the *pre-partially bosonized* action, following the actual transformation in Sec. 3.1.1. Note that the Gaussian integral over σ is only defined for positive values of λ. As such, this particular Hubbard-Stratonovich transformation limits our studies to systems with $\lambda > 0$. The new action S^*_{PB} now depends on the original field ϕ and the auxiliary field σ, but so far we are no closer to removing the ϕ^4 term. To achieve this, we need to shift the integration over σ:

$$\sigma \to \sigma + i\phi^2\,. \tag{2.71}$$

This shift does not change the value of the integral since we have introduced the field σ in a Gaussian integral and, loosely speaking, the value of a Gaussian integral does not depend on the location of the peak of the bell curve. With this shift, the partition function reads

$$Z = \mathcal{N}\int \mathrm{d}\phi\ \mathrm{d}\sigma\ e^{-S_{\mathrm{PB}}(\phi,\sigma)}\,, \tag{2.72}$$

with the *partially-bosonized* action

$$S_{\mathrm{PB}}(\phi,\sigma) = \underbrace{\frac{\mu}{2}\phi^2 + i\frac{\lambda}{12}\sigma\phi^2}_{S_{\mathrm{FC}}(\phi,\sigma)} + \underbrace{\frac{\lambda}{4!}\sigma^2}_{S_{\mathrm{PA}}(\sigma)}\,, \tag{2.73}$$

again named in analogy to the actual study in Sec. 3.1.1. We also define the two parts S_{FC}, which we call the *fermionic contribution* because it contains all terms depending on the original field, and S_{PA}, which we call the *purely-auxiliary* part because it only depends on the auxiliary field σ. With these two parts, the partition function factorizes to:

$$\begin{aligned} Z &= \mathcal{N}\int \mathrm{d}\sigma\ e^{-S_{\mathrm{PA}}(\sigma)} \int \mathrm{d}\phi\ e^{-S_{\mathrm{FC}}(\phi,\sigma)} \\ &= \mathcal{N}\sqrt{\pi}\int \mathrm{d}\sigma\ \left(\frac{\mu}{2} + i\frac{\lambda}{12}\sigma\right)^{-1/2}\ e^{-S_{\mathrm{PA}}(\sigma)} \\ &= \mathcal{N}\int \mathrm{d}\sigma\ e^{-S_{\mathrm{B}}(\sigma)}\,, \end{aligned} \tag{2.74}$$

wherein we have solved the integral over ϕ and redefined the normalization constant $\mathcal{N}$ to absorb the factor $\sqrt{\pi}$. The *bosonized* action S_{B} in the above expression depends purely on σ and is given by

$$S_{\mathrm{B}}(\sigma) = \frac{\lambda}{4!}\sigma^2 + \frac{1}{2}\log\left(\frac{\mu}{2} + i\frac{\lambda}{12}\sigma\right)\,. \tag{2.75}$$

This action depends on the real-valued field σ and the real-valued parameters μ and λ but clearly takes complex values, so it will allow us to study the Complex Langevin method.

To formulate the Langevin equation for this system, we first need to determine the drift term. It is given by the variation of the action with respect to the field of the theory, or, in the zero-dimensional case, the derivative of the action with respect to the field of the theory:

$$\frac{\partial S_{\mathrm{B}}}{\partial\sigma} = \frac{\lambda}{12}\sigma + \frac{i\lambda}{12\mu + i2\lambda\sigma}\,. \tag{2.76}$$

This drift is clearly complex. Considering we obtained it from performing a derivative of a complex quantity, this is not surprising, but it has an interesting consequence. Because we add this drift term to the field in every iteration of the Langevin equation, the field itself has to become a complex quantity. Even though the system is described by a real-valued field σ, when using Complex Langevin to solve the theory, we need to artificially promote the field to complex values to accommodate the complex drift. We call this artificially complexified field the CL field. This holds for all real-valued fields of a theory. If we use Complex Langevin to solve a theory with a complex-valued quantum field ϕ, the theory really depends on the two real-valued fields $\mathrm{Re}(\phi)$ and $\mathrm{Im}(\phi)$ that we would both promote to complex values.

That being said, with the drift term calculated, we need to determine the noise term of the Langevin equation. In the case of Complex Langevin, we have an additional degree of freedom in that regard: we can choose to apply the noise on the real part of the CL field, the imaginary part of the CL field, or on both parts:

$$\sqrt{2N_R\delta t_{\mathrm{CL}}}\,\tilde{\eta}_R(t_{\mathrm{CL}}) + \sqrt{2N_I\delta t_{\mathrm{CL}}}\,\tilde{\eta}_I(t_{\mathrm{CL}}) \tag{2.77}$$

We still need to respect the fluctuation-dissipation theorem that ensures we add the right amount of noise to produce the correct quantum results. This constraint leads to the requirement

$$N_R - N_I = 1\,. \tag{2.78}$$

Other than that, we are free to choose how we distribute the noise between real- and imaginary parts. Because the field σ of the theory is real-valued and only becomes complex in the Complex Langevin solution of the theory, we expect the dynamics of the system to mainly be encoded in the real part of the CL field. As such, purely real noise seems like the most efficient way to solve the theory, as it has the most effect on the part of the CL field that encodes the dynamics of the system. More details about the choice of the noise term in the complex case can be found in Refs. [44, 48]. For this example, we chose purely real noise and find the Langevin equations

$$\begin{aligned}
\sigma_R^{(i+1)} &= \sigma_R^{(i)} - \mathrm{Re}\left(\frac{\lambda}{12}\sigma^{(i)} + \frac{i\lambda}{12\mu + i2\lambda\sigma^{(i)}}\right)\delta t_{\mathrm{CL}} + \sqrt{2\delta t_{\mathrm{CL}}}\,\tilde{\eta}(t_{\mathrm{CL}}) \quad \text{and} \\
\sigma_I^{(i+1)} &= \sigma_I^{(i)} - \mathrm{Im}\left(\frac{\lambda}{12}\sigma^{(i)} + \frac{i\lambda}{12\mu + i2\lambda\sigma^{(i)}}\right)\delta t_{\mathrm{CL}}\,,
\end{aligned} \tag{2.79}$$

with $\sigma_R^{(i)} = \mathrm{Re}(\sigma^{(i)})$ and $\sigma_I^{(i)} = \mathrm{Im}(\sigma^{(i)})$. These Langevin equations allow us to solve the system, but we also need to figure out how to calculate observables, now that our description of the theory no longer contains the original field ϕ. In this example, we want to calculate the observable $\langle\phi\rangle$, which we somehow need to relate to the auxiliary field σ. A general way to do this is to introduce source terms into the original action and propagate them through the entire Hubbard-Stratonovich transformation into the bosonized action of the system. There, they can be used to obtain the expression of the auxiliary field that corresponds to the original field. In Sec. 3.1.2, we do this for our system of fermions to obtain relations between the pairing field and the fermion fields, but in this simple example, we can use a trick instead. In the path-integral expression for the observable $\langle\phi^2\rangle$, we can replace the term ϕ^2 by a derivative with respect to a parameter of the theory and pull the derivative out of the path integral:

$$\begin{aligned}
\langle\phi^2\rangle &= \frac{1}{Z}\int \mathrm{d}\phi\ \phi^2\ e^{-S(\phi)} \\
&= \frac{1}{Z}\int \mathrm{d}\phi\ \left(-2\partial_\mu\right)\ e^{-S(\phi)} \\
&= \frac{1}{Z}\left(-2\partial_\mu\right)\int \mathrm{d}\phi\ e^{-S(\phi)}\,.
\end{aligned} \tag{2.80}$$

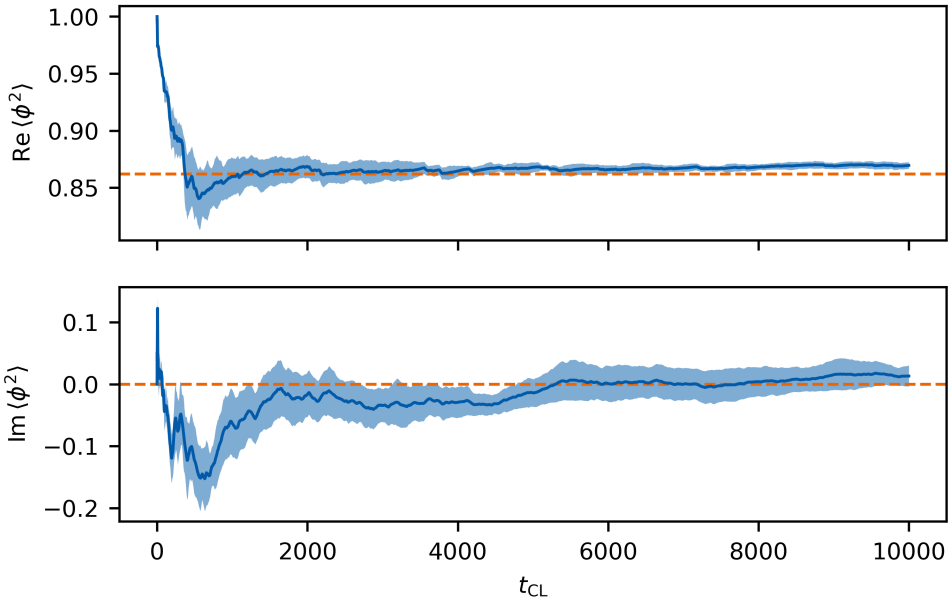


Figure 2.7: Evolution of the estimates of the observable $\langle\phi^2\rangle$ using Complex Langevin. The values of the parameters of the system in this example are $(\mu,\lambda) = (1.0, 0.4)$ and the Langevin equation is solved up to a final Langevin time of $t_{\mathrm{CL,max}} = 10\,000$ with a spacing of $\delta t_{\mathrm{CL}} = 0.05$. The two subplots show the real- and imaginary parts of the estimate, and the dashed orange lines represent the exact values of the observable. The uncertainty bands are obtained using the Jackknife method.

The remaining path integral is the partition function of the system expressed in terms of the original action, and we can replace it with the partition function expressed in terms of the bosonized action. Then we can perform the derivative again and obtain the expression depending on σ that corresponds to ϕ^2 in the original description of the theory:

$$\begin{aligned} &\text{[Continuation of Eq. (2.80)]} \\ &= \frac{1}{Z}\,\mathcal{N}\int \mathrm{d}\sigma\ \left(-2\partial_\mu\right)\ e^{-S_{\mathrm{B}}(\sigma)} \\ &= \frac{1}{Z}\,\mathcal{N}\int \mathrm{d}\sigma\ \left(\frac{6}{6\mu + i\lambda\sigma}\right)\ e^{-S_{\mathrm{B}}(\sigma)}\,. \end{aligned} \tag{2.81}$$

Thus, we can compute the observable $\langle\phi^2\rangle$ as

$$\langle\phi^2\rangle = \left\langle \frac{6}{6\mu + i\lambda\sigma}\right\rangle_{\mathrm{CL}}\,. \tag{2.82}$$

Figure 2.7 shows the evolution of the estimate for the observable $\langle\phi^2\rangle$ using Complex Langevin in both real- and imaginary part for a system with $(\mu,\lambda) = (1.0, 0.4)$ to a final Langevin time $t_{\mathrm{CL,max}} = 10\,000$ with a spacing $\delta t_{\mathrm{CL}} = 0.05$. As expected, the imaginary part of the estimate converges to zero. The real part converges to the exact value even faster than in the real Langevin example in Fig. 2.3. This is likely due to the fact that the auxiliary field σ appears linearly in the observable expression, and, as such, we do not have to square the random process. Either way, for the observable $\langle\phi^2\rangle$, the representation of the system in terms of σ is numerically favorable to the representation in terms of ϕ. The convergence in this example is so good, in fact, that we can now clearly see the systematic deviation of the estimate, discussed in Sec. 2.2.4, caused by the finite spacing δt_{CL}.

2.3 Future Optimizations

Beyond the scope of the present work, there are some optimizations we can apply to the Langevin method and the study of our system of interest in particular. We briefly touch upon some of them in this section.

2.3.1 Adaptive Langevin Step

One future optimization is to make the Langevin time step δt_{CL} adaptive. That means, rather than leaving it at the same value throughout the entire simulation, we adapt its value from step to step based on the current state of the process. This has a few advantages. As we have seen in the above sections, low values of δt_{CL} make the simulation more accurate but also more expensive. High values of δt_{CL} make the simulation less accurate but cheaper, and they also reduce the autocorrelation in the process. So far, we have to weigh the arguments to settle on a value for δt_{CL} that is neither too high nor too low. As it turns out, however, for this particular cake, we can have it and eat it as well. With an adaptive step, we can choose low values of δt_{CL} where the path integral weight e^{-S} changes considerably. This allows us to resolve fine details of the dynamics of the system that are important for obtaining accurate results. In regions where the weight e^{-S} varies less, we can choose larger values for δt_{CL} to sample those regions more efficiently. In fact, we have already used an adaptive stepsize in Sec. 2.2.3. In that section, we introduced the adaptive step to map the Langevin equation to Newton's method for finding zeros, but it ultimately serves the same purpose as it does in Newton's method: Where it can, Newton's method takes larger steps to converge more quickly, and it takes finer steps where they are needed to improve the accuracy of the result.

Beyond that, Ref. [49] discusses certain instabilities which can occur in the Langevin method with complex actions and can be avoided completely by adaptive steps. To employ the analogy to Newton's method again as a motivation, without its adaptive step, it could potentially overshoot and miss the zero it is trying to find.

2.3.2 Stochastic Trace Evaluation

In the simulation of our system of interest we develop below, we need to calculate expressions of the form

$$\mathrm{tr}\left(\mathcal{M}'\mathcal{M}^{-1}\right) , \tag{2.83}$$

with a matrix $\mathcal{M}'$ and an inverse matrix $\mathcal{M}^{-1}$ for the drift term. The matrix $\mathcal{M}$ in this expression depends on the Langevin time. As such, we have to calculate the inverse of $\mathcal{M}$ at every simulation step. This operation is very costly, and in this section, we want to explore an alternative method that approximates the expression in Eq. (2.83) stochastically, sparing us the computational effort of inverting the matrix $\mathcal{M}$. This alternative method consists of two parts: stochastically approximating traces of matrices and using gradient solvers to avoid inverting matrices.

A method of stochastically approximating traces is described in Ref. [50]. For a matrix $M \in \mathbb{C}^{d\times d}$, we generate d vectors

$$\left\{ \chi^{(i)} \,\middle|\, i = 1, ..., d \right\} , \tag{2.84}$$

with $\chi^{(i)} \in \mathbb{C}^d$ and each entry of $\chi^{(i)}$ drawn from a complex normal distribution with a mean of zero and a standard deviation of $1/d$. Due to the chosen standard deviation of the random distribution, all $\chi^{(i)}$ approximately have a norm of one:

$$\chi^{(i)\dagger}\chi^{(i)} \approx 1 . \tag{2.85}$$

Furthermore, because all entries are chosen at random, two different vectors have approximately no overlap:

$$\chi^{(i)\dagger}\chi^{(j)} \overset{i\neq j}{\approx} 0\,. \tag{2.86}$$

These two properties combined imply that the set $\{\chi^{(i)}\}$ approximately form an orthonormal basis of the vector space $\mathbb{C}^d$:

$$\begin{aligned} \delta_{ij} &\approx \chi^{(i)\dagger}\chi^{(j)} \\ \Rightarrow \qquad \mathbb{1} &\approx \sum_{i=1}^{d} \chi^{(i)\dagger}\chi^{(i)}\,. \end{aligned} \tag{2.87}$$

As such, we can use the χ-basis to calculate the trace of M:

$$\operatorname{tr} M \approx \sum_{i=1}^{d} \chi^{(i)\dagger} M \chi^{(i)}\,. \tag{2.88}$$

So far, this is a complete stochastic calculation of the trace rather than an approximation. What allows us to save computational effort is the fact that we do not need to calculate all terms of the sum in Eq. (2.88). Because all entries of each of the vectors $\chi^{(i)}$ are non-zero with a probability of one, every term of the sum in Eq. (2.88) contains contributions by all entries of the matrix M. As a result, each of the terms in the sum is a precise approximation of the trace, and adding them merely increases the estimate's accuracy. For $d' \in \mathbb{N}$ with $d' < d$, we find the estimate

$$\operatorname{tr} M \approx \frac{d}{d'} \sum_{i=1}^{d'} \chi^{(i)\dagger} M \chi^{(i)}\,, \tag{2.89}$$

with the scaling factor in front of the sum accounting for the omitted terms. This approximation can be improved, by explicitly orthonormalizing the χ-basis using the Gram-Schmidt procedure. This improvement allows the trace estimate to converge faster, at the additional cost of the orthonormalization. Whether this decreases the total computational effort depends on the specific problem at hand. For our expression in Eq. (2.83), we find the estimate

$$\operatorname{tr}\left(\mathcal{M}'\mathcal{M}^{-1}\right) = \frac{d}{d'} \sum_{i=1}^{d'} \chi^{(i)\dagger} \mathcal{M}' \mathcal{M}^{-1} \chi^{(i)}\,. \tag{2.90}$$

The second part of estimating the expression in Eq. (2.83) is given by *Conjugate Gradient* (CG) methods [50, 51]. This is the part that actually allows us to not invert the matrix $\mathcal{M}$ at every simulation step. CG methods focus on solving systems of linear equations

$$M\mathbf{x} = \mathbf{b}\,, \tag{2.91}$$

with $M \in \mathbb{C}^{d\times d}$ and $\mathbf{x}, \mathbf{b} \in \mathbb{C}^d$. If the matrix M is not self-adjoint, as is the case in our simulation, we require *Biconjugate Gradient* (BiCG) method that is a generalization of the CG method. For the problem in Eq. (2.91), the matrix M and the vector $\mathbf{b}$ are inputs of the BiCG algorithm, and the result is given by a vector $\mathbf{x}$ satisfying Eq. (2.91):

$$\begin{aligned} M\,\mathrm{BiCG}(M,\mathbf{b}) &= \mathbf{b} \\ \Rightarrow \qquad \mathrm{BiCG}(M,\mathbf{b}) &= M^{-1}\mathbf{b}\,. \end{aligned} \tag{2.92}$$

For the estimate in Eq. (2.90), this allows us to replace the explicit inversion of $\mathcal{M}$:

$$\operatorname{tr}\left(\mathcal{M}'\mathcal{M}^{-1}\right) = \frac{d}{d'} \sum_{i=1}^{d'} \chi^{(i)\dagger} \mathcal{M}'\,\mathrm{BiCG}\left(\mathcal{M}, \chi^{(i)}\right)\,. \tag{2.93}$$

In the above equation, we need to perform the BiCG algorithm for every term in the sum, but because the stochastic estimate of the trace only requires $d' < d$ terms, this calculation can be cheaper than directly inverting the matrix. In our simulation, the size of the matrix $\mathcal{M}$ scales with the power of the spatial dimension. Therefore, this approximation may reduce the computational effort significantly in future studies beyond $0 + 1$-dimensional systems.

3 Pairing-Field Formulation of the System

In this chapter, we will derive a path integral formulation of a grand-canonical partition function $Z(\beta, \mu_\uparrow, \mu_\downarrow)$ for our system of two fermion species with a contact interaction. Moreover, we shall see that it is indeed possible to formulate this path integral purely in terms of auxiliary bosonic degrees of freedom that constitute what we call the *pairing field*.

We begin with a Hamiltonian formulation of our Fermi gas with the two-component species we call "↑" (up) and "↓" (down). Even though we borrow their designations from the theory of spin-1/2 particles, our species do not necessarily have to be two spin projections. In fact, in real-world experiments with ultracold atoms, the species are usually realized as two different hyperfine states of atoms of a given element (e.g. Refs. [2, 52, 53] for ^{6}Li, ^{40}K and ^{167}Er, respectively) or even states of atoms of two different chemical elements or isotopes thereof (e.g. Refs. [32, 33] for ^{162}Dy with ^{161}Dy and ^{53}Cr with ^{6}Li). Regarding this possible realization of the system, we allow our two species to have different masses $m_\uparrow$ and $m_\downarrow$, respectively. We also assign two potentially different chemical potentials $\mu_\uparrow$ and $\mu_\downarrow$ to our species to allow for *population imbalance* or *"spin" imbalance*, again borrowing from the jargon of spin-1/2 particles. However, the chemical potentials do not enter the Hamiltonian description of the system and will appear when we transition to a grand-canonical description of the system, as will the inverse temperature $\beta = 1/T$. The Hamiltonian for this system in second quantization is given by

$$\boxed{\hat{H} = \int \mathrm{d}^d r \, \left(- \sum_{\sigma \in \{\uparrow,\downarrow\}} \hat{\psi}^\dagger_\sigma(\mathbf{r}) \frac{\nabla^2}{2m_\sigma} \hat{\psi}_\sigma(\mathbf{r}) - g \, \hat{\psi}^\dagger_\uparrow(\mathbf{r}) \hat{\psi}_\uparrow(\mathbf{r}) \hat{\psi}^\dagger_\downarrow(\mathbf{r}) \hat{\psi}_\downarrow(\mathbf{r}) \right)} \tag{3.1}$$

with d spatial dimensions, a non-relativistic kinetic energy term, and an attractive contact interaction with the coupling parameter $g \geq 0$. The field operators obey the fermionic anti-commutation relations

$$\{\hat{\psi}_{\sigma,\mathbf{r}_i}, \hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\} = \delta_{\sigma\sigma'} \delta^{(d)}(\mathbf{r}_i - \mathbf{r}_j) \,, \quad \{\hat{\psi}_{\sigma,\mathbf{r}_i}, \hat{\psi}_{\sigma',\mathbf{r}_j}\} = 0 \quad \text{and} \quad \{\hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}, \hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\} = 0 \tag{3.2}$$

with

$$\{\hat{A}, \hat{B}\} = \hat{A}\hat{B} + \hat{B}\hat{A} \,. \tag{3.3}$$

This work is by no means the first to study this Hamiltonian. A variety of approaches can be found in Ref. [20].

We begin by deriving the desired path integral representation of this system in the continuum to avoid the added complications of a lattice at first. We will see that the continuum limit masks some of the finer details and inner workings of the description, which makes the derivation easier in some parts because certain questions simply do not arise in the continuum and harder in others where the continuum masks details that need to be very well understood for correct derivations.

After studying the system in the continuum, we derive a description of our system on a space-time lattice. This description serves as the foundation of the simulation that allows us to obtain numerical results for the behavior of this system. Rather than discretizing the expressions we have obtained in the continuum, we begin anew with the Hamiltonian description of the system. This will ultimately alert us to subtle details in the realm of lattice theories that make a discretization of a continuum action non-trivial and, in fact, incorrect if done naively without paying attention to these details.

3.1 Pairing-Field Formulation in the Continuum

The first step in finding a pairing field formulation of the system is to find a path integral representation of the (grand-canonical) partition function

$$Z(\beta,\mu_\uparrow,\mu_\downarrow) = \operatorname{tr} e^{-\beta(\hat{H}-\mu_\uparrow\hat{N}_\uparrow-\mu_\downarrow\hat{N}_\downarrow)} \tag{3.4}$$

with our system Hamiltonian $\hat{H}$ from Eq. (3.1) and the particle number operators

$$\hat{N}_\sigma = \int \mathrm{d}^d r\, \hat{\psi}^\dagger_{\sigma,\mathbf{r}}\hat{\psi}_{\sigma,\mathbf{r}} \tag{3.5}$$

for $\sigma \in \{\uparrow,\downarrow\}$ as well as the inverse temperature $\beta = 1/T$ and the chemical potentials μ_σ. We then find the path integral representation

$$Z(\beta,\mu_\uparrow,\mu_\downarrow) = \int D(\psi^*,\psi)\; e^{-S_\mathrm{F}[\psi^*,\psi]} \tag{3.6}$$

with the Grassmann-valued fields $\psi^*_\uparrow,\psi_\uparrow,\psi^*_\downarrow,\psi_\downarrow$ and

$$\psi^* = (\psi^*_\uparrow,\psi^*_\downarrow)\,, \quad \psi = (\psi_\uparrow,\psi_\downarrow) \tag{3.7}$$

and the fermionic Euclidean action functional

$$\boxed{S_\mathrm{F}[\psi^*,\psi] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \left[\psi^*(\tau,\mathbf{r})\left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right)\psi(\tau,\mathbf{r}) - g\,(\psi^*_\uparrow\psi_\uparrow\psi^*_\downarrow\psi_\downarrow)(\tau,\mathbf{r})\right]}\,. \tag{3.8}$$

Heuristically, one can obtain this action from the Hamiltonian by replacing the field operators with Grassmann-valued fields with a time dimension and adding a temporal derivative and integration. These rules follow from the general scheme of path integral derivations found in books like Ref. [54] and, indeed, in the Sec. 3.3 we shall see that a rigorous derivation of the path integral for the partition function leads to this continuum action.

3.1.1 Hubbard Stratonovich Transfomation

The fields ψ^* and ψ in the path integral of the partition function in Eq. (3.6) are fermionic fields and thus Grassmann-valued rather than real- or complex-valued like bosonic fields. This presents a problem for the numerical treatment of the system since computers are, by design, more suitable to efficiently work with real or complex numbers. To overcome this, we reformulate our partition function in terms of a path integral over auxiliary bosonic fields that are complex-valued. This will allow us to calculate all the same observables by manipulating complex fields rather than Grassmann-valued fields and fully leverage the advantage of computers in numerics. Such a transformation from a fermionic representation of a system to a bosonic representation of the same system is known as a *Hubbard-Stratonovich transformation* [54, 55].

There are many kinds of Hubbard-Stratonovich transormation. As discussed in Ref. [11], some Hubbard-Stratonovich transformations lead to continuous auxiliary fields, some to discrete auxiliary fields, that can be both bounded or compact and unbounded. The Hubbard-Stratonovich transformation we perform leads to a continuous unbounded auxiliary field that, in its final form, we call the *pairing field*. We begin the Hubbard-Stratonovich transformation by inserting a suitable factor of one into our partition function, namely

$$1 = \int D(\phi^*, \phi)\; e^{-g\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\; \phi^*(\tau,\mathbf{r})\phi(\tau,\mathbf{r})}. \tag{3.9}$$

The fields ϕ^* and ϕ map $[0,\beta)\times\mathbb{R}^d \to \mathbb{C}$ and are considered to be independent. Inserting this factor into Eq. (3.6) we obtain

$$\begin{aligned} Z(\beta,\mu_\uparrow,\mu_\downarrow) &= \int D(\psi^*,\psi)\; e^{-S_{\mathrm{F}}[\psi^*,\psi]} \\ &= \int D(\psi^*,\psi,\phi^*,\phi)\; e^{-S_{\mathrm{F}}[\psi^*,\psi]-g\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\; \phi^*(\tau,\mathbf{r})\phi(\tau,\mathbf{r})} \\ &= \int D(\psi^*,\psi,\phi^*,\phi)\; e^{-S^*_{\mathrm{PB}}[\psi^*,\psi,\phi^*,\phi]} \end{aligned} \tag{3.10}$$

with the *pre-partially-bosonized action*

$$\begin{aligned} S^*_{\mathrm{PB}}[\psi^*,\psi,\phi^*,\phi] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \Bigg[&\psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right)\psi(\tau,\mathbf{r}) \\ &- g\,(\psi^*_\uparrow\psi_\uparrow\psi^*_\downarrow\psi_\downarrow)(\tau,\mathbf{r}) + g\,(\phi^*\phi)(\tau,\mathbf{r}) \Bigg]. \end{aligned} \tag{3.11}$$

The way this new path integration will help us bosonize the action is by the fact that the path integral in the factor of one in Eq. (3.9) is a Gaussian integral. This means we can perform a shift of the integration variables without changing the value of the integral. This fact can be derived rigorously but becomes intuitively evident by considering a one-dimensional Gaussian integral; no matter where the center of the bell curve is, the area under it remains the same, i.e.

$$\int_{-\infty}^{\infty} \mathrm{d}x\; e^{-x^2} = \int_{-\infty}^{\infty} \mathrm{d}x\; e^{-(x-a)^2} = \sqrt{\pi}\,. \tag{3.12}$$

In the context of our action in Eq. (3.11), we can perform a shift on the auxiliary fields ϕ^* and ϕ, such that the shift creates a term that cancels the interaction term. At this point, the fermionic component of the path integral, i.e. the integration over ψ^* and ψ is a Gaussian integral and can be carried out. We perform the shift

$$\phi \to \phi + \psi_\uparrow\psi_\downarrow \quad \text{and} \quad \phi^* \to \phi^* + \psi^*_\downarrow\psi^*_\uparrow\,, \tag{3.13}$$

which creates the countering four-fermion term and Yukawa coupling terms. The latter replace the contact interaction:

$$\begin{aligned} g\,\phi^*\phi &\to g\left(\phi^* + \psi^*_\downarrow\psi^*_\uparrow\right)\left(\phi + \psi_\uparrow\psi_\downarrow\right) \\ &= g\,\phi^*\phi + g\left(\phi^*\psi_\uparrow\psi_\downarrow + \phi\psi^*_\downarrow\psi^*_\uparrow\right) + g\psi^*_\downarrow\psi^*_\uparrow\psi_\uparrow\psi_\downarrow \\ &= g\,\phi^*\phi + g\left(\phi^*\psi_\uparrow\psi_\downarrow - \phi\psi^*_\uparrow\psi^*_\downarrow\right) + g\psi^*_\uparrow\psi_\uparrow\psi^*_\downarrow\psi_\downarrow\,. \end{aligned} \tag{3.14}$$

The path integral with shifted auxiliary fields yields the *partially-bosonized action*

$$S_{\mathrm{PB}}[\psi^*,\psi,\phi^*,\phi] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \Bigg[\psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma \right) \psi(\tau,\mathbf{r}) + g\,(\phi^*\psi_\uparrow\psi_\downarrow - \phi\psi_\uparrow^*\psi_\downarrow^*)(\tau,\mathbf{r}) + g\,(\phi^*\phi)(\tau,\mathbf{r}) \Bigg] \tag{3.15}$$

that is now quadratic in its fermionic degrees of freedom. To perform the fermionic path integral, we split the partially-bosonized action into a *fermionic contribution*

$$S_{\mathrm{FC}}[\psi_{\mathrm{NG}}^*,\psi_{\mathrm{NG}}] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \, \left(\psi_{\mathrm{NG}}^\dagger \, M_{\mathrm{FS}} \, \psi_{\mathrm{NG}} \right)(\tau,\mathbf{r}) \tag{3.16}$$

with spinors $\psi_{\mathrm{NG}}^*, \psi_{\mathrm{NG}}$ and a *fermion matrix* M_{FS} (to be defined below), as well as a *purely-auxiliary* part

$$S_{\mathrm{PA}}[\phi^*,\phi] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \, g(\phi^*\phi)(\tau,\mathbf{r}) \,. \tag{3.17}$$

This allows us to factor out the fermionic part in the partition function

$$Z(\beta,\mu_\uparrow,\mu_\downarrow) = \int D(\phi^*,\phi) \underbrace{\int D(\psi_{\mathrm{NG}}^*,\psi_{\mathrm{NG}}) \, e^{-S_{\mathrm{FC}}}}_{Z_{\mathrm{FC}}} \, e^{-S_{\mathrm{PA}}} \tag{3.18}$$

and solve the Grassman-valued Gaussian integral Z_{FC}, as

$$\begin{aligned} Z_{\mathrm{FC}} &= \mathrm{Det}\left(M_{\mathrm{FS}} \, \delta(\tau-\tau') \, \delta^{(3)}(\mathbf{r}-\mathbf{r}') \right) \\ &\equiv \mathrm{Det}\, M \,, \end{aligned} \tag{3.19}$$

with the capitalized Det indicating that the determinant also includes the matrix structure in spacetime coordinates and not merely the matrix structure of M_{FS} in the 2×2 "field space". To determine the matrix M_{FS}, we need to write the fermionic contribution to the action in terms of *Nambu-Gorkov* spinors

$$\psi_{\mathrm{NG}}^* = \left(\psi_\uparrow^*, \, \psi_\downarrow \right)^\intercal \quad \text{and} \quad \psi_{\mathrm{NG}} = \left(\psi_\uparrow, \, \psi_\downarrow^* \right)^\intercal . \tag{3.20}$$

This particular arrangement of starred and unstarred fields of up- and down-species is necessary to write a bilinear expression $\psi_{\mathrm{NG}}^\dagger M_{\mathrm{FS}} \psi_{\mathrm{NG}}$ that can reproduce the Yukawa terms in the partially-bosonized action in Eq. (3.15). Rewriting the fermionic contribution to the action in terms of these Nambu-Gorkov spinors we find

$$S_{\mathrm{FC}}[\psi_{\mathrm{NG}}^*,\psi_{\mathrm{NG}},\phi^*,\phi] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \left(\psi_{\mathrm{NG}}^\dagger \underbrace{\begin{pmatrix} \partial_\tau - \frac{\nabla^2}{2m_\uparrow} - \mu_\uparrow & -g\phi \\ -g\phi^* & \partial_\tau + \frac{\nabla^2}{2m_\downarrow} + \mu_\downarrow \end{pmatrix}}_{M_{\mathrm{FS}}} \psi_{\mathrm{NG}} \right)(\tau,\mathbf{r}) \,, \tag{3.21}$$

giving us the fermion matrix

$$M = \underbrace{\begin{pmatrix} \partial_\tau - \frac{\nabla^2}{2m_\uparrow} - \mu_\uparrow & -g\phi \\ -g\phi^* & \partial_\tau + \frac{\nabla^2}{2m_\downarrow} + \mu_\downarrow \end{pmatrix}}_{M_{\mathrm{FS}}} \delta(\tau-\tau') \, \delta^{(3)}(\mathbf{r}-\mathbf{r}') \,. \tag{3.22}$$

Note that in the lower-right entry of the "field space" matrix M_{FS}, where we find the free propagation of the down-species, the spatial derivative and chemical potential enter with a different sign than they do in the up-species. This is a consequence of the mixing of fields in the Nambu-Gorkov representation; writing out the terms of this matrix representation has the fields of the up-species in the propagator term in the right order, i.e. $\psi_\uparrow^*(\ldots)\psi_\uparrow$, while in the down-species they are reversed, i.e. $\psi_\downarrow(\ldots)\psi_\downarrow^*$. This transforms the operators in the middle in a non-trivial way, as detailed in App. A.

With the fermion matrix determined from the fermionic contribution to the action, we can define the fully bosonized action S_{B} with

$$Z(\beta,\mu_\uparrow,\mu_\downarrow) = \int D(\phi^*,\phi)\; e^{-S_{\mathrm{B}}[\phi^*,\phi]} \tag{3.23}$$

and

$$\boxed{S_{\mathrm{B}}[\phi^*,\phi] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\; g(\phi^*,\phi)(\tau,\mathbf{r}) - \log\mathrm{Det}\, M}\,. \tag{3.24}$$

Now all dynamics of the system are encoded in an action that depends purely on the bosonic pairing field and lends itself to numerical treatment.

Looking back at the partially bosonized action in Eq. (3.15), we can interpret how the bosonization encodes the interaction in the system. Rather than a term of four fermion fields, we find two Yukawa terms in wich two fermion fields of different species couple to the pairing field. As such, the pairing field mediates the interaction between pairs of fermions.

3.1.2 Correlation Functions of the Pairing Field

We want to examine how expectation values of the pairing field relate to expectation values of the fundamental fields ψ_σ for $\sigma \in \{\uparrow,\downarrow\}$ to get a better understanding of what the pairing field is. To achieve this, we define a generating functional from the fermionic representation of the partition function by including source fields J^* and J for pairs of the fermionic fields[1]

$$Z[J^*,J,\beta,\mu_\uparrow,\mu_\downarrow] = \int D(\psi^*,\psi)\; e^{-S_{\mathrm{F}}[\psi^*,\psi,J^*,J]} \tag{3.25}$$

with

$$\begin{aligned} S_{\mathrm{F}}[\psi^*,\psi,J^*,J] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \Bigg[& \psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right) \psi(\tau,\mathbf{r}) - g\;(\psi_\uparrow^*\psi_\uparrow\psi_\downarrow^*\psi_\downarrow)(\tau,\mathbf{r}) \\ & - g\left(J^*\psi_\downarrow^*\psi_\uparrow^* + J\psi_\uparrow\psi_\downarrow + J^*J\right)(\tau,\mathbf{r})\Bigg]. \end{aligned} \tag{3.26}$$

We now perform the same Hubbard-Stratonovich transformation we have performed on the fermionic action in the previous section to obtain a bosonized theory but also transform the source fields. This will result in a bosonic generating functional that still describes the same physical system. We then can perform functional derivatives with respect to the source fields of the purely fermionic and purely bosonic generating functionals and compare them to identify relations between the expectation values of the fundamental fermionic fields and the pairing field. We begin with the Hubbard-Stratonovich transformation by inserting a suitable factor of one.

[1] The specific sources we choose will lead us to the desired relation, however, when deriving such a relation it can be beneficial to start at the end of the Hubbard-Stratonovich transformation and work backward.

We use the factor found in Eq. (3.9), the same factor we use above to introduce the auxiliary field:

$$\begin{aligned} Z[J^*,J,\beta,\mu_\uparrow,\mu_\downarrow] &= \int D(\psi^*,\psi)\; e^{-S_{\rm F}[\psi^*,\psi,J^*,J]} \\ &= \int D(\psi^*,\psi,\phi^*,\phi)\; e^{-S_{\rm F}[\psi^*,\psi,J^*,J]}\; e^{-\int_{\tau,\mathbf{r}} g(\phi^*\phi)(\tau,\mathbf{r})} \\ &\equiv \int D(\psi^*,\psi,\phi^*,\phi)\; e^{-S^*_{\rm PB}[\psi^*,\psi,\phi^*,\phi,J^*,J]}\,, \end{aligned} \tag{3.27}$$

defining the pre-partially bosonized action

$$\begin{aligned} S^*_{\rm PB}[\psi^*,\psi,\phi^*,\phi,J^*,J] = \int_0^\beta {\rm d}\tau \int {\rm d}^d r \Bigg[& \psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right) \psi(\tau,\mathbf{r}) \\ & - g\,(\psi^*_\uparrow\psi_\uparrow\psi^*_\downarrow\psi_\downarrow)(\tau,\mathbf{r}) + g(\phi^*\phi)(\tau,\mathbf{r}) \\ & - g\left(J^*\psi^*_\downarrow\psi^*_\uparrow + J\psi_\uparrow\psi_\downarrow + J^*J\right)(\tau,\mathbf{r})\Bigg]. \end{aligned} \tag{3.28}$$

Before we perform the shift of the auxiliary fields that eliminates the four-fermion interaction, we need to take one more intermediate step. With the source fields present, we perform yet another shift that couples the auxiliary fields to the source fields. This ensures that we also cancel the source terms of the fermion fields when performing the shift that cancels the four-fermion interaction. This shift assumes the simple form

$$\phi^* \to \phi^* + J \quad \text{and} \quad \phi \to \phi + J^* \tag{3.29}$$

and results in the altered pre-partially-bosonized action

$$\begin{aligned} S^{**}_{\rm PB}[\psi^*,\psi,\phi^*,\phi,J^*,J] &= \int_0^\beta {\rm d}\tau \int {\rm d}^d r \Bigg[\psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right) \psi(\tau,\mathbf{r}) \\ &\qquad - g\,(\psi^*_\uparrow\psi_\uparrow\psi^*_\downarrow\psi_\downarrow)(\tau,\mathbf{r}) \\ &\qquad + g\,((\phi^*+J)(\phi+J^*))\,(\tau,\mathbf{r}) \\ &\qquad - g\left(J^*\psi^*_\downarrow\psi^*_\uparrow + J\psi_\uparrow\psi_\downarrow + J^*J\right)(\tau,\mathbf{r})\Bigg] \\ &= \int_0^\beta {\rm d}\tau \int {\rm d}^d r \Bigg[\psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right) \psi(\tau,\mathbf{r}) \\ &\qquad - g\,(\psi^*_\uparrow\psi_\uparrow\psi^*_\downarrow\psi_\downarrow)(\tau,\mathbf{r}) \\ &\qquad + g\left(J^*\left(\phi^* - \psi^*_\downarrow\psi^*_\uparrow\right) + J\left(\phi - \psi_\uparrow\psi_\downarrow\right)\right)(\tau,\mathbf{r})\Bigg]. \end{aligned} \tag{3.30}$$

From here we can perform the shift we used in the previous section to remove the four-fermion interaction

$$\phi^* \to \phi^* + \psi^*_\downarrow\psi^*_\uparrow \quad \text{and} \quad \phi \to \phi + \psi_\uparrow\psi_\downarrow\,, \tag{3.31}$$

leaving us with the partially-bosonized action

$$S_{\mathrm{PB}}[\psi^*,\psi,\phi^*,\phi,J^*,J] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \Bigg[\psi^*(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma \right) \psi(\tau,\mathbf{r}) - g\, (\psi^*_\uparrow \psi_\uparrow \psi^*_\downarrow \psi_\downarrow)(\tau,\mathbf{r}) + g\, (J^*\phi^* + J\phi)\,(\tau,\mathbf{r}) \Bigg]. \tag{3.32}$$

In this form, we can integrate out the fermion fields in the generating functional resulting in the bosonized action

$$S_{\mathrm{B}}[\phi^*,\phi,J^*,J] = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\, g\, (\phi^*\phi + J^*\phi^* + J\phi)(\tau,\mathbf{r}) - \log \mathrm{Det}\, M \tag{3.33}$$

with the fermion matrix M we defined in Eq. (3.22).

With this at hand, we can finally calculate expectation values and correlation functions of the pairing field. We find

$$\begin{aligned}
\langle \phi(\tau,\mathbf{r}) \rangle &= \frac{1}{Z} \int D(\phi^*,\phi)\; \phi(\tau,\mathbf{r})\, e^{-S_{\mathrm{B}}[\phi^*,\phi]} && (3.34)\\
&= \frac{1}{Z} \int D(\phi^*,\phi) \left(-\frac{1}{g} \frac{\delta}{\delta J(\tau,\mathbf{r})} \right) e^{-S_{\mathrm{B}}[\phi^*,\phi,J^*,J]} \Big|_{J^*=J=0} && (3.35)\\
&= \left(-\frac{1}{g} \frac{\delta}{\delta J(\tau,\mathbf{r})} \right) \frac{Z[J^*,J]}{Z} \Big|_{J^*=J=0} && (3.36)\\
&= \frac{1}{Z} \int D(\psi^*,\psi) \left(-\frac{1}{g} \frac{\delta}{\delta J(\tau,\mathbf{r})} \right) e^{-S_{\mathrm{F}}[\psi^*,\psi,J^*,J]} \Big|_{J^*=J=0} && (3.37)\\
&= \frac{1}{Z} \int D(\psi^*,\psi) \left(-\psi_\uparrow(\tau,\mathbf{r}) \psi_\downarrow(\tau,\mathbf{r}) \right) e^{-S_{\mathrm{F}}[\psi^*,\psi]} && (3.38)\\
&= \left\langle \psi_\downarrow(\tau,\mathbf{r}) \psi_\uparrow(\tau,\mathbf{r}) \right\rangle, && (3.39)
\end{aligned}$$

which justifies the name *pairing field*. For correlation functions, we proceed in a similar manner. For example we have

$$\begin{aligned}
&\langle \phi^*(\tau_1,\mathbf{r}_1) \phi(\tau_2,\mathbf{r}_2) \rangle && (3.40)\\
&= \frac{1}{Z} \int D(\phi^*,\phi)\; \phi^*(\tau_1,\mathbf{r}_1) \phi(\tau_2,\mathbf{r}_2)\, e^{-S_{\mathrm{B}}[\phi^*,\phi]} && (3.41)\\
&= \frac{1}{Z} \int D(\phi^*,\phi) \left(-\frac{1}{g} \frac{\delta}{\delta J^*(\tau_1,\mathbf{r}_1)} \right) \left(-\frac{1}{g} \frac{\delta}{\delta J(\tau_2,\mathbf{r}_2)} \right) e^{-S_{\mathrm{B}}[\phi^*,\phi,J^*,J]} \Big|_{J^*=J=0} && (3.42)\\
&= \frac{1}{Z} \int D(\psi^*,\psi) \left(-\frac{1}{g} \frac{\delta}{\delta J^*(\tau_1,\mathbf{r}_1)} \right) \left(-\frac{1}{g} \frac{\delta}{\delta J(\tau_2,\mathbf{r}_2)} \right) e^{-S_{\mathrm{F}}[\psi^*,\psi,J^*,J]} \Big|_{J^*=J=0} && (3.43)\\
&= \frac{1}{Z} \int D(\psi^*,\psi) \left(-\psi^*_\downarrow(\tau_1,\mathbf{r}_1) \psi^*_\uparrow(\tau_1,\mathbf{r}_1) \right) \left(-\psi_\uparrow(\tau_2,\mathbf{r}_2) \psi_\downarrow(\tau_2,\mathbf{r}_2) \right) e^{-S_{\mathrm{F}}[\psi^*,\psi]} && (3.44)\\
&= \left\langle \psi^*_\downarrow(\tau_1,\mathbf{r}_1) \psi^*_\uparrow(\tau_1,\mathbf{r}_1) \psi_\uparrow(\tau_2,\mathbf{r}_2) \psi_\downarrow(\tau_2,\mathbf{r}_2) \right\rangle. && (3.45)
\end{aligned}$$

Following this technique, we can find expressions for all pairing-related observables in terms of the pairing field ϕ.

3.2 Mean-Field Study of the System

In this section, we perform a mean-field study of our system. That means we study its behavior in the approximation that the pairing field is constant, i.e.

$$\phi(\tau,\mathbf{r}) \equiv \bar{\phi} \quad \text{and} \quad \phi^*(\tau,\mathbf{r}) \equiv \bar{\phi}^* \,. \tag{3.46}$$

To this end, we derive an analytic expression for the partition function with non-constant fermionic fields in momentum space, perform the bosonization and extract physical results from the resulting expression. For simplicity, we shall also limit this mean-field study to the mass-balanced case of

$$m_\uparrow = m_\downarrow = m \,, \tag{3.47}$$

although we shall treat $m_\uparrow$ and $m_\downarrow$ as independent quantities until it is necessary to leverage the mass balance. We do this to obtain more general expressions as long as it is possible.

Fourier-Transforming the Action

We begin by Fourier-transforming the partially-bosonized action in Eq. (3.15) by using the momentum-frequency space fields

$$\tilde{\psi}_\sigma(\omega_n,\mathbf{p}) = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\, \psi_\sigma(\tau,\mathbf{r}) e^{i\omega_n\tau} e^{i\mathbf{pr}} \tag{3.48}$$

with the fermionic Matsubara frequencies

$$\omega_n = \frac{(2n+1)\pi}{\beta} \quad \text{for} \quad n \in \mathbb{Z}. \tag{3.49}$$

This definition together with the identities

$$\int_0^\beta \mathrm{d}\tau\, e^{-i(\omega_n-\omega_{n'})\tau} = \beta\,\delta_{n,n'} \tag{3.50}$$

and

$$\int \mathrm{d}^d r\, e^{-i(\mathbf{p}-\mathbf{p}')\mathbf{r}} = (2\pi)^d\,\delta^{(d)}(\mathbf{p}-\mathbf{p}') \tag{3.51}$$

implies the Fourier representation of the position-space fields

$$\psi_\sigma(\tau,\mathbf{r}) = \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\, \tilde{\psi}_\sigma(\omega_n,\mathbf{p})\; e^{-i\omega_n\tau} e^{-i\mathbf{pr}}. \tag{3.52}$$

For the Fourier transform of the free propagation in the action, we begin by replacing the fields as per Eq. (3.52):

$$\begin{aligned}
&\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\, \psi^*_\sigma(\tau,\mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right) \psi_\sigma(\tau,\mathbf{r}) \\
&= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d} \int \frac{\mathrm{d}^d p'}{(2\pi)^d} \\
&\qquad \tilde{\psi}^*_\sigma(\omega_n,\mathbf{p}) e^{i\omega_n\tau} e^{i\mathbf{pr}} \left(\partial_\tau - \frac{\nabla^2}{2m_\sigma} - \mu_\sigma\right) \tilde{\psi}_\sigma(\omega_{n'},\mathbf{p}') e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}}
\end{aligned} \tag{3.53}$$

At this point, we can replace the operators by scalars and apply the δ-identities in Eq. (3.50) and Eq. (3.51):

$$
\begin{aligned}
&\text{[Continuation of Eq. (3.53)]}\\
&= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\int \frac{\mathrm{d}^d p'}{(2\pi)^d}\\
&\qquad \tilde{\psi}^*_\sigma(\omega_n,\mathbf{p}) e^{i\omega_n\tau} e^{i\mathbf{p}\mathbf{r}} \left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m_\sigma} - \mu_\sigma\right) \tilde{\psi}_\sigma(\omega_{n'},\mathbf{p}') e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}}\\
&= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\int \frac{\mathrm{d}^d p'}{(2\pi)^d}\; e^{-i(\omega_{n'}-\omega_n)\tau} e^{-i(\mathbf{p}'-\mathbf{p})\mathbf{r}}\\
&\qquad \tilde{\psi}^*_\sigma(\omega_n,\mathbf{p}) \left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m_\sigma} - \mu_\sigma\right) \tilde{\psi}_\sigma(\omega_{n'},\mathbf{p}')\\
&= \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\int \frac{\mathrm{d}^d p'}{(2\pi)^d}\; \beta\delta_{n,n'}(2\pi)^d \delta^{(d)}(\mathbf{p}-\mathbf{p}')\\
&\qquad \tilde{\psi}^*_\sigma(\omega_n,\mathbf{p}) \left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m_\sigma} - \mu_\sigma\right) \tilde{\psi}_\sigma(\omega_{n'},\mathbf{p}')\\
&= \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; \tilde{\psi}^*_\sigma(\omega_n,\mathbf{p}) \left(-i\omega_n + \frac{\mathbf{p}^2}{2m_\sigma} - \mu_\sigma\right) \tilde{\psi}_\sigma(\omega_n,\mathbf{p}).
\end{aligned}
\tag{3.54}
$$

The Yukawa-terms are less straight forward because of their pairing of starred and unstarred fields. We need to "massage" the expressions a little to find δ-functions, leading to a slightly different result. To this end, we first consider the simpler expression

$$
\begin{aligned}
&\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r\; \psi_\uparrow(\tau,\mathbf{r})\psi_\downarrow(\tau,\mathbf{r})\\
&= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\int \frac{\mathrm{d}^d p'}{(2\pi)^d}\\
&\qquad \tilde{\psi}_\uparrow(\omega_n,\mathbf{p}) e^{-i\omega_n\tau} e^{-i\mathbf{p}\mathbf{r}} \tilde{\psi}_\downarrow(\omega_{n'},\mathbf{p}') e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}}\\
&= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\int \frac{\mathrm{d}^d p'}{(2\pi)^d}\; e^{-i(\omega_n-(-\omega_{n'}))\tau} e^{-i(\mathbf{p}-(-\mathbf{p}'))\mathbf{r}}\\
&\qquad \tilde{\psi}_\uparrow(\omega_n,\mathbf{p}) \tilde{\psi}_\downarrow(\omega_{n'},\mathbf{p}').
\end{aligned}
\tag{3.55}
$$

Here, we can perform the $\mathbf{r}$ integral to obtain a $\delta^{(d)}(\mathbf{p}+\mathbf{p}')$, effectively flipping the momentum argument of the $\downarrow$-field:

$$
\begin{aligned}
&\text{[Continuation of Eq. (3.55)]}\\
&= \int_0^\beta \mathrm{d}\tau \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\int \frac{\mathrm{d}^d p'}{(2\pi)^d}\; e^{-i(\omega_n-(-\omega_{n'}))\tau}(2\pi)^d\delta^{(d)}(\mathbf{p}-(-\mathbf{p}'))\\
&\qquad \tilde{\psi}_\uparrow(\omega_n,\mathbf{p}) \tilde{\psi}_\downarrow(\omega_{n'},\mathbf{p}')\\
&= \int_0^\beta \mathrm{d}\tau \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; e^{-i(\omega_n-(-\omega_{n'}))\tau} \tilde{\psi}_\uparrow(\omega_n,\mathbf{p}) \tilde{\psi}_\downarrow(\omega_{n'},-\mathbf{p}).
\end{aligned}
\tag{3.56}
$$

For the Matsubara frequencies, we can reverse the summation for the down-species by replacing

$$
n' \to -n' \tag{3.57}
$$

and use the relation

$$\omega_{-n'} = -\omega_{n'-1}\,. \tag{3.58}$$

We also shift the summation index n' by one to make the expression more legible before performing the τ integral to extract the frequency Kronecker-δ:

$$\begin{aligned}
&\text{[Continuation of Eq. (3.56)]}\\
&= \int_0^\beta \mathrm{d}\tau \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; e^{-i(\omega_n-(-\omega_{-n'}))\tau}\tilde{\psi}_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}_\downarrow(\omega_{-n'},-\mathbf{p})\\
&= \int_0^\beta \mathrm{d}\tau \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; e^{-i(\omega_n-\omega_{n'-1})\tau}\tilde{\psi}_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}_\downarrow(-\omega_{n'-1},-\mathbf{p})\\
&= \int_0^\beta \mathrm{d}\tau \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; e^{-i(\omega_n-\omega_{n'})\tau}\tilde{\psi}_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}_\downarrow(-\omega_{n'},-\mathbf{p})\\
&= \frac{1}{\beta}\sum_n \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; \beta\delta_{n,n'}\,\tilde{\psi}_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}_\downarrow(-\omega_{n'},-\mathbf{p})\\
&= \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; \tilde{\psi}_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}_\downarrow(-\omega_n,-\mathbf{p})\,.
\end{aligned} \tag{3.59}$$

Thus, Fourier-transforming products of two starred or unstarred fields flips the sign of the argument of one of them in the momentum-space representation. With that out of the way, we can write the momentum-space representation of the partially-bosonized action as follows:

$$S_{\mathrm{PB}}[\tilde{\psi}^*_\uparrow,\tilde{\psi}_\uparrow,\tilde{\psi}^*_\downarrow,\tilde{\psi}_\downarrow] = \underbrace{g\beta V\bar{\phi}^*\bar{\phi}}_{S_{\mathrm{PA}}} + S_{\mathrm{FC}}[\tilde{\psi}^*_\uparrow,\tilde{\psi}_\uparrow,\tilde{\psi}^*_\downarrow,\tilde{\psi}_\downarrow]\,, \tag{3.60}$$

again divided into a *purely-auxiliary* part S_{PA} and a *fermionic contribution* given by

$$\begin{aligned}
S_{\mathrm{FC}}[\tilde{\psi}^*_\uparrow,\tilde{\psi}_\uparrow,\tilde{\psi}^*_\downarrow,\tilde{\psi}_\downarrow] = \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d} \Bigg[&\tilde{\psi}^*_\sigma\left(-i\omega_n + \frac{\mathbf{p}^2}{2m_\sigma} - \mu_\sigma\right)\tilde{\psi}_\sigma\\
&+ g\,\bar{\phi}^*\tilde{\psi}_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}_\downarrow(-\omega_n,-\mathbf{p})\\
&- g\,\bar{\phi}\tilde{\psi}^*_\uparrow(\omega_n,\mathbf{p})\tilde{\psi}^*_\downarrow(-\omega_n,-\mathbf{p})\Bigg]\,.
\end{aligned} \tag{3.61}$$

Obtaining an Expression for the Partition Function

To integrate out the fermions, we rewrite S_{FC} in terms of Nambu-Gorkov spinors

$$\tilde{\psi}^\dagger = \begin{pmatrix} \tilde{\psi}^*_\uparrow(\omega_n,\mathbf{p}) & \tilde{\psi}_\downarrow(-\omega_n,-\mathbf{p}) \end{pmatrix} \quad \text{and} \quad \tilde{\psi} = \begin{pmatrix} \tilde{\psi}_\uparrow(\omega_n,\mathbf{p}) \\ \tilde{\psi}^*_\downarrow(-\omega_n,-\mathbf{p}) \end{pmatrix}, \tag{3.62}$$

resulting in the expression

$$S_{\mathrm{FC}}[\tilde{\psi}^\dagger,\tilde{\psi}] = \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\; \tilde{\psi}^\dagger \underbrace{\begin{pmatrix} -i\omega_n + \frac{\mathbf{p}^2}{2m_\uparrow} - \mu_\uparrow & -g\bar{\phi} \\ -g\bar{\phi}^* & -i\omega_n - \frac{\mathbf{p}^2}{2m_\downarrow} + \mu_\downarrow \end{pmatrix}}_{M_{\mathrm{FS}}} \tilde{\psi} \tag{3.63}$$

with the fermion matrix

$$M(\omega_n,\mathbf{p},\mu_\uparrow,\mu_\downarrow) = M_{\mathrm{FS}}(\omega_n,\mathbf{p},\mu_\uparrow,\mu_\downarrow)\;\beta\delta_{n',n''}\;(2\pi)^d\delta^{(d)}(\mathbf{p}'-\mathbf{p}''). \tag{3.64}$$

This fermion matrix is understood to have a matrix structure not only in the 2×2 "field space" of M_{FS} but also in the frequency and momentum space. The matrix structure of M beyond the field space is denoted by the Kronecker-δ and δ-function in Eq. (3.64), indicating that M is diagonal in these spaces. They are to be understood as a label of this matrix structure rather than concrete mathematical terms since neither n', n'' nor $\mathbf{p}', \mathbf{p}''$ are actually defined in the expression for S_{FC} in Eq. (3.63).

Performing the path integral over the fermion fields in the partition function, we obtain the expression

$$\begin{aligned} \log \bar{Z}(\bar{\phi}^*, \bar{\phi}, \beta, \mu_\uparrow, \mu_\downarrow) &= \log e^{-S_{\mathrm{PA}}} + \log \mathrm{Det}\, M(\mu_\uparrow, \mu_\downarrow) \\ &= \log e^{-S_{\mathrm{PA}}} + \mathrm{Tr} \log M(\mu_\uparrow, \mu_\downarrow) \end{aligned} \tag{3.65}$$

with the capitalized Det and Tr denoting determinants and traces not only over the 2×2 "field space" but also over frequency and momentum space. The bar above the partition function $\bar{Z}$ indicates that this is not yet the real partition function. The real partition function is obtained by determining the constants $\bar{\phi}$ and $\bar{\phi}^*$. Moreover, note that the expression depends on β, even though M does not. This is a direct consequence of the determinant or trace also acting in frequency space.

The next step is to perform the trace over the fermion matrix. To achieve this, we first nest the field-space trace $\mathrm{tr}_{\mathrm{FS}}$ into the frequency-momentum trace $\mathrm{tr}_{n,\mathbf{p}}$:

$$\mathrm{Tr} \log \left(M_{\mathrm{FS}}\, \beta \delta_{n',n''}\, (2\pi)^d \delta^{(d)}(\mathbf{p}' - \mathbf{p}'') \right) = \mathrm{tr}_{n,\mathbf{p}}\, \mathrm{tr}_{\mathrm{FS}} \log \left(M_{\mathrm{FS}}\, \beta \delta_{n',n''}\, (2\pi)^d \delta^{(d)}(\mathbf{p}' - \mathbf{p}'') \right). \tag{3.66}$$

If we were to write the logarithm as a power series, in every term the Kronecker-δ and δ-function would act as identities due to the trace over frequency and momentum, so we can pull them out of the field-space trace, rewriting this expression as

$$\mathrm{Tr} \log M = \mathrm{tr}_{n,\mathbf{p}} \left((\mathrm{tr}_{\mathrm{FS}} \log M_{\mathrm{FS}})\ \beta \delta_{n',n''}\, (2\pi)^d \delta^{(d)}(\mathbf{p}' - \mathbf{p}'') \right). \tag{3.67}$$

Within the frequency-momentum trace, we express the diagonality of the fermion matrix by replacing

$$\delta_{n',n''} \to 1 \quad \text{and} \quad \delta^{(d)}(\mathbf{p}' - \mathbf{p}'') \to \delta^{(d)}(0) \tag{3.68}$$

resulting in the expression

$$\mathrm{Tr} \log M = \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\ (\mathrm{tr}_{\mathrm{FS}} \log M_{\mathrm{FS}})\ \beta (2\pi)^d \delta^{(d)}(0). \tag{3.69}$$

We can identify the physical meaning of the $\delta^{(d)}(0)$ by employing the Fourier representation of the delta function

$$\begin{aligned} (2\pi)^d \delta^{(d)}(\mathbf{p}' - \mathbf{p}'') &= \int \mathrm{d}^d r\ e^{i(\mathbf{p}' - \mathbf{p}'')\mathbf{r}} \\ \Rightarrow \qquad (2\pi)^d \delta^{(d)}(0) &= \int \mathrm{d}^d r = V \end{aligned} \tag{3.70}$$

and identifying it with the spatial volume V. This is a rather "loose" handling of the δ-distribution, however, within the scope of this work, namely Sec. 3.3, we will explore this theory on the lattice where the matrix nature of the fermion matrix beyond the field space is trivially included and we can confirm our findings by the means of standard determinants and traces of linear algebra. The continuum limit of the expressions we find on the lattice confirms the results we find by employing this interpretation of $\delta^d(0)$.

With the Kronecker δ and the δ function out of the way, we obtain the expression

$$\mathrm{Tr}\log M = \beta V \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\ \mathrm{tr}_{\mathrm{FS}}\log M_{\mathrm{FS}}\,, \tag{3.71}$$

in which we can take the field-space trace over $\log M_{\mathrm{FS}}$ by taking the logarithm of the field-space determinant, i.e.

$$\mathrm{tr}_{\mathrm{FS}}\log M_{\mathrm{FS}} = \log\det M_{\mathrm{FS}}. \tag{3.72}$$

To simplify the following steps, we express the per-species chemical potentials in terms of the *average chemical potential*

$$\mu = \frac{\mu_\uparrow + \mu_\downarrow}{2} \tag{3.73}$$

and the *chemical potential imbalance*

$$h = \frac{\mu_\uparrow - \mu_\downarrow}{2} \tag{3.74}$$

resulting in

$$\mu_\uparrow = \mu + h \quad \text{and} \quad \mu_\downarrow = \mu - h. \tag{3.75}$$

Beyond that we define

$$\varepsilon = \frac{\mathbf{p}^2}{2m} \quad \text{and} \quad \Delta^2 = g^2\bar{\phi}^*\bar{\phi}, \tag{3.76}$$

assuming mass balance for the remainder of the mean-field study. Using these shorthands, we calculate the determinant of the field-space fermion matrix as

$$\begin{aligned}
\det M_{\mathrm{FS}} &= \begin{vmatrix} -i\omega_n - h + (\varepsilon-\mu) & -g\bar{\phi} \\ -g\bar{\phi}^* & -i\omega_n - h - (\varepsilon-\mu) \end{vmatrix} \\
&= (-i\omega_n - h + (\varepsilon-\mu))\,(-i\omega_n - h - (\varepsilon-\mu)) - \Delta^2 \\
&= (-i\omega_n - h)^2 - (\varepsilon-\mu)^2 - \Delta^2 \\
&= (-i\omega_n - h)^2 - \sqrt{(\varepsilon-\mu)^2+\Delta^2}^{\,2} \\
&= \left(-i\omega_n - h + \sqrt{(\varepsilon-\mu)^2+\Delta^2}\right)\left(-i\omega_n - h - \sqrt{(\varepsilon-\mu)^2+\Delta^2}\right) \\
&= \prod_{\sigma=\pm1}\left(-i\omega_n - h + \sigma\sqrt{(\varepsilon-\mu)^2+\Delta^2}\right) \\
&= \prod_{\sigma=\pm1}(-i\omega_n + A_\sigma)
\end{aligned} \tag{3.77}$$

defining the new shorthand

$$A_\sigma = -h + \sigma\sqrt{(\varepsilon-\mu)^2+\Delta^2}. \tag{3.78}$$

This product expression for the determinant plays nicely with the logarithm since the logarithm of a product is just a sum of logarithms. We find

$$\mathrm{Tr}\log M = \beta V \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d}\sum_{\sigma=\pm1}\log\left(-i\omega_n + A_\sigma\right). \tag{3.79}$$

To perform the Matsubara sum, we employ a trick. We perform a derivative of $\mathrm{Tr}\log M$ with respect to h, which gives us a simpler Matsubara sum that we can solve and after that, we

integrate the expression with respect to h to obtain the solution of the original Matsubara sum plus a constant. For the derivative we find

$$\frac{\partial \operatorname{Tr} \log M}{\partial h} = \beta V \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^d p}{(2\pi)^d} \sum_{\sigma=\pm 1} \frac{1}{i\omega_n - A_\sigma} \tag{3.80}$$

with the Matsubara sum

$$s = \frac{1}{\beta} \sum_n \frac{1}{i\omega_n - A_\sigma} \,. \tag{3.81}$$

To solve this, we employ a standard technique that is described, e.g., in Ref. [54]: we express the sum in terms of a complex integral and make use of the residue theorem. For this, we define the two meromorphic auxiliary functions

$$u(z) = \frac{\beta}{e^{\beta z} + 1} \tag{3.82}$$

and

$$v(z) = \frac{1/\beta}{iz - A_\sigma}. \tag{3.83}$$

The function v encodes the original terms in the Matsubara sum s, since

$$s = \sum_n v(\omega_n) \tag{3.84}$$

and u provides the connection to complex integrals since it has simple poles at the fermionic Matsubara frequencies ω_n. Because the poles of u are simple, i.e. of order one, we find the residue by making use of L'Hôpital's rule:

$$\begin{aligned} \operatorname*{res}_{z=i\omega_n} u(z)v(-iz) &= \lim_{z\to i\omega_n} (z - i\omega_n) u(z)v(-iz) \\ &= \lim_{z\to i\omega_n} \frac{z - i\omega_n}{(e^{\beta z} + 1)(z - A_\sigma)} \\ &= -\frac{1}{\beta} \frac{1}{i\omega_n - A_\sigma} \,. \end{aligned} \tag{3.85}$$

If we express the Matsubara sum s as of sum of these residues, we can use the residue theorem to express it in terms of a complex integral

$$s = -\sum_n \operatorname*{res}_{z=i\omega_n} u(z)v(-iz) = \frac{i}{2\pi} \oint_{\gamma_1} \mathrm{d}z \; u(z)v(-iz) \tag{3.86}$$

with the integration contour γ_1 running counter-clockwise and enclosing all poles of u on the imaginary axis before connecting at $z = \pm i\infty$.[2] We can deform the integration contour to the new contour γ_2 without changing the value of the integral as long as we do not cross the pole of $v(-iz)$ at $z = A_\sigma$ on the real axis. We chose γ_2, such that it forms a circle of infinite radius with an inset small circle around the pole of $v(-iz)$. Both contours γ_1 and γ_2 are shown in Fig. 3.1. With the contour γ_2, the contribution of the outer circle to the integral vanishes, since the integrand decays sufficiently fast, the connection between the inner and outer circle is made of two identical pieces in opposite directions that cancel each other and the integral over the

[2] More formally, the value of the integral can be understood as the limit of a sequence of integrals with tubular contours successively containing more of the poles of u.

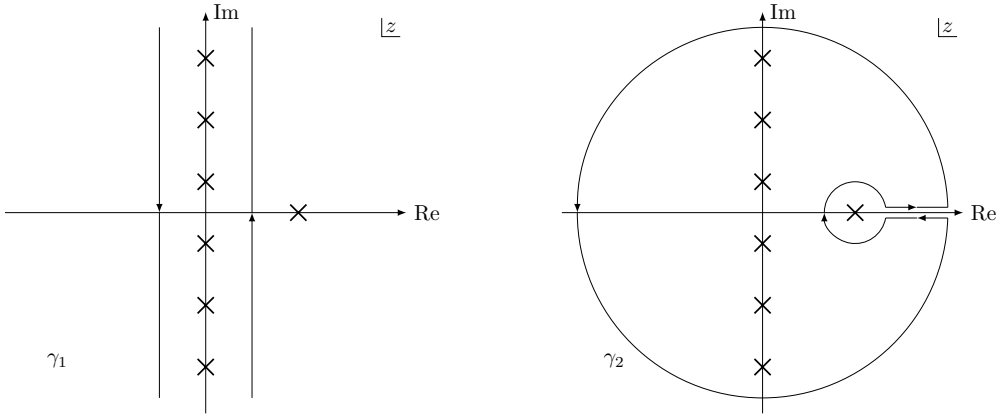


Figure 3.1: Contours γ_1 (left) and γ_2 (right) of the integrals in Eq. (3.87). The poles of the integrand $u(z)v(-iz)$ are shown as crosses in the complex plane with the poles on the imaginary axis corresponding to the fermionic Matsubara frequencies and the single pole on the real axis being caused by $v(-iz)$. The contour γ_1 can be deformed into γ_2 without crossing one of the poles.

inner circle can again be expressed in terms of the residue of the enclosed pole with a change in sign resulting from the opposite direction of the inner circle:

$$\begin{aligned} s &= \frac{i}{2\pi} \oint_{\gamma_1} \mathrm{d}z\, u(z)v(-iz) \\ &= \frac{i}{2\pi} \oint_{\gamma_2} \mathrm{d}z\, u(z)v(-iz) \\ &= \operatorname*{res}_{z=A_\sigma} u(z)v(-iz) \\ &= \lim_{z\to A_\sigma} (z-A_\sigma) \frac{1}{e^{\beta z}+1} \frac{1}{z-A_\sigma} \\ &= \frac{1}{e^{\beta A_\sigma}+1}\,. \end{aligned} \tag{3.87}$$

With this expression for the Matsubara sum, the derivative in Eq. (3.63) reads

$$\frac{\partial \operatorname{Tr}\log M}{\partial h} = \beta V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \sum_{\sigma=\pm 1} \frac{1}{e^{\beta A_\sigma}+1} \tag{3.88}$$

and we can integrate it over h to re-obtain $\operatorname{Tr}\log M$:

$$\begin{aligned} \operatorname{Tr}\log M &= \beta V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \sum_{\sigma=\pm 1} \int \mathrm{d}h\, \frac{1}{e^{\beta A_\sigma}+1} \\ &= \beta V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \sum_{\sigma=\pm 1} \frac{1}{\beta} \left(\log\left(1+e^{\beta A_\sigma}\right) + h + C(\beta,\mu)/2 \right). \end{aligned} \tag{3.89}$$

The expression $C(\beta,\mu)/2$ is the integration constant of the indefinite integral over h. This expression is constant only with respect to h but in general depends on β and μ. Beyond this point,

we shall limit the discussion to the balanced case of $h = 0$.

$$\begin{aligned}
&\text{[Continuation of Eq. (3.89)]}\\
&= V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \left(C(\beta,\mu) + \sum_{\sigma=\pm 1} \log\left(1 + e^{\sigma\beta\sqrt{(\varepsilon-\mu)^2+\Delta^2}}\right)\right)\\
&= V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \left(C(\beta,\mu) + \log\left(2 + 2\cosh\left(\beta\sqrt{(\varepsilon-\mu)^2+\Delta^2}\right)\right)\right)\\
&= V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \left(C(\beta,\mu) + \log\left(\frac{1}{2} + \frac{1}{2}\cosh\left(\beta\sqrt{(\varepsilon-\mu)^2+\Delta^2}\right)\right)\right) ,
\end{aligned} \tag{3.90}$$

where we dropped a constant in the last step. With this expression for $\mathrm{Tr}\log M$ we arrive at the partition function

$$\begin{aligned}
\log \bar{Z}(\bar{\phi}^*, \bar{\phi}, \beta, \mu) = &- g\beta V \bar{\phi}^* \bar{\phi}\\
&+ V \int \frac{\mathrm{d}^d p}{(2\pi)^d} \left(C(\beta,\mu) + \log\left(\frac{1}{2} + \frac{1}{2}\cosh\left(\beta\sqrt{(\varepsilon-\mu)^2+\Delta^2}\right)\right)\right) ,
\end{aligned} \tag{3.91}$$

which coincides with the expression obtained in Ref. [56]. In Sec. 5.1.2 we shall use these results to calculate the density of the system in mean-field approximation.

3.3 Pairing-Field Formulation on the Lattice

In this section, we derive a lattice theory for our system of two interacting fermion species, but rather than discretizing the path integral formulation we derived for the continuum, we begin by defining a lattice and discretizing the Hamiltonian of the system and rigorously derive a fermionic path integral representation from there. As we shall see, this results in a different expression than we would get from naively discretizing the continuous action and ensures that the correct continuum limit is obtained. We then conduct the bosonization of the system by introducing a discrete pairing field. This formalism is also described in Ref. [14].

3.3.1 Defining the Lattice

We begin the discretization of our theory by replacing the continuous domain of the single particle wave functions $\psi_{\mathrm{SP},\sigma}$ of the two fermion species with a discrete one. In the continuum this domain is given by $\mathbb{R}^d$ and on the lattice we replace it with a d-dimensional, periodic (hyper-) cubic lattice of edge length L and a lattice site spacing a_x, such that

$$L = N_x a_x \tag{3.92}$$

with N_x lattice sites in each spatial direction. The total number of lattice sites in this lattice is given by N_x^d.

For being able to denote fermionic states in terms of occupation numbers, as well as the use of the matrix notation defined in Sec. 3.3.7, we require that the lattice sites have a defined order. For this order, it is not actually important how we arrange the lattice sites in a sequence, it is just important that their order is defined. To denote the order we choose, as well as for practical applications within the numerical implementation of this calculation, we define an *index function* α that takes coordinates of a given lattice site as arguments and returns the place that the given lattice site takes in the ordering of the lattice. That means, when representing field configuration on the lattice as a single column vector, the index function α defines which entry of the vector belongs to which point in spacetime.

Since at a later point of the derivation, we will have to add a time dimension to our lattice, we shall define this dimension now and include it in the definition of the index function α. We define our lattice to have a time dimension with edge length β and a lattice site spacing a_τ, such that

$$\beta = N_\tau a_\tau \tag{3.93}$$

for N_τ lattice sites in the temporal direction. The total number of lattice sites in this spacetime lattice is given by

$$N = N_\tau N_x^d \tag{3.94}$$

and for the special case of $N_\tau = 1$, we re-obtain a purely spatial lattice in which all lattice sites share the same value for the time coordinate.

Given these definitions, we define an index function α to have the signature

$$\alpha : [0,\beta) \times [0,L)^d \to \{1,...,N\}\,. \tag{3.95}$$

Such an index function takes a tuple of spacetime coordinates $(\tau, x_1, x_2, ..., x_d)$ and bijectively maps it to a tensor index. Note that we choose to start these index numbers at one rather than zero. This is done to be consistent with tensor notation in mathematics and can potentially clash with the indexing conventions of a programming language used to implement these calculations, constituting a risk for *off-by-one errors*. This is discussed in more detail in Sec. 4.3. It is also useful to define a discrete variant of the index function that takes a tuple of integers as a multi-index rather than a tuple of continuous spacetime coordinates to identify a given lattice site. For this multi-index we define the coordinate indices $i_\tau \in \{1,...,N_\tau\}$ and $i_{x_d} \in \{1,...,N_x\}$ such that they correspond to the spacetime coordinates

$$\tau(i_\tau) = (i_\tau - 1)a_\tau \tag{3.96}$$

and

$$x_d(i_{x_d}) = (i_{x_d} - 1)a_x\,. \tag{3.97}$$

This allows us to define the *discrete index function*

$$\bar{\alpha} : \{1,...,N_\tau\} \times \{1,...,N_x\}^d \to \{1,...,N\} \tag{3.98}$$

with

$$\bar{\alpha}(i_\tau, i_{x_1}, ..., i_{x_d}) = \alpha(\tau(i_\tau),\ x_1(i_{x_1}),\ ...,\ x_d(i_{x_d}))\,. \tag{3.99}$$

Beyond that, we also define the inverses of the continuous and discrete index functions that allow us to obtain the spacetime coordinates or multi-index from a given position within the lattice ordering. For the (continuous) index function α, we define

$$\begin{aligned} \tau^{(\alpha)} &: \{1,...,N\} \to [0,\beta)\,, \\ x_1^{(\alpha)} &: \{1,...,N\} \to [0,L)\,, \\ &... \\ x_d^{(\alpha)} &: \{1,...,N\} \to [0,L)\,, \end{aligned} \tag{3.100}$$

and for the discrete index function $\bar{\alpha}$, we define

$$\begin{aligned} i_\tau^{(\alpha)} &: \{1,...,N\} \to \{1,...,N_\tau\}\,, \\ i_{x_1}^{(\alpha)} &: \{1,...,N\} \to \{1,...,N_x\}\,, \\ &... \\ i_{x_d}^{(\alpha)} &: \{1,...,N\} \to \{1,...,N_x\}\,. \end{aligned} \tag{3.101}$$

For the present work, we make the following concrete choice for the index function α:

$$\alpha(\tau, x_1, x_2, ..., x_d) = \left(\left\lfloor \frac{\tau}{a_\tau} \right\rfloor \bmod N_\tau \right) \cdot N_x^d + \sum_{i=1}^{d} \left(\left\lfloor \frac{x_i}{a_x} \right\rfloor \bmod N_x \right) N_x^{d-i} + 1 \tag{3.102}$$

with $\lfloor \bullet \rfloor$ denoting the floor function and mod denoting modulo division.

3.3.2 Field Operators and Hamiltonian on the Lattice

Now that we have a notion of discrete spacetime, we can discretize the Hamiltonian by introducing discrete field operators. We start by replacing the continuous domain of our single-particle wave functions $\psi_{\mathrm{SP},\sigma}$ for $\sigma \in \{\uparrow, \downarrow\}$ with the set of spatial sites in our lattice

$$\psi_{\mathrm{SP},\sigma} : \mathbb{R}^d \to \mathbb{C} \quad \longrightarrow \quad \psi_{\mathrm{SP},\sigma} : \{0, a_x, ..., N_x a_x\}^d \to \mathbb{C}\,, \tag{3.103}$$

such that for each spatial lattice site $\mathbf{r}$ we find

$$\psi_{\mathrm{SP},\sigma,\mathbf{r}} = \psi_{\mathrm{SP},\sigma}(\mathbf{r}) \tag{3.104}$$

at the time $\tau = 0$. Since these single-particle wave functions constitute bases of Hilbert spaces, we also move from the Hilbert spaces of continuous wave functions $\mathcal{H}_{\sigma,\mathrm{cont}}$ to the Hilbert spaces of discrete wave functions $\mathcal{H}_\sigma$. In analogy to the completeness relation of the Hilbert spaces of continuous wave functions,

$$\mathbb{1}_{\mathcal{H}_{\sigma,\mathrm{cont}}} = \int \mathrm{d}^d r \, |\mathbf{r}\rangle\langle\mathbf{r}| \,, \tag{3.105}$$

we define a completeness relation for the Hilbert spaces of discrete wave functions:

$$\mathbb{1}_{\mathcal{H}_\sigma} = \sum_{\mathbf{r}} a_x^d \, |\mathbf{r}\rangle\langle\mathbf{r}| \,, \tag{3.106}$$

in which the sum $\sum_{\mathbf{r}}$ represents the sum over all spatial lattice sites. This completeness relation implies the state normalization

$$\langle \mathbf{r} | \mathbf{r}' \rangle = \delta^{(d)}_{\mathbf{r},\mathbf{r}'} \tag{3.107}$$

with

$$\delta^{(d)}_{\mathbf{r},\mathbf{r}'} = \frac{1}{a_x^d} \prod_{i=1}^{d} \delta_{\mathbf{r}_i,\mathbf{r}'_i} \,. \tag{3.108}$$

For the single-particle wave functions we choose the normalization

$$\sum_{\mathbf{r}} a_x^d \, \psi^*_{\mathrm{SP},\sigma,\mathbf{r}} \psi_{\mathrm{SP},\sigma,\mathbf{r}} = 1 \,, \tag{3.109}$$

which, together with the completeness relation, implies the projections

$$\langle \mathbf{r} | \psi_{\mathrm{SP},\sigma} \rangle = \psi_{\mathrm{SP},\sigma,\mathbf{r}} \quad \text{and} \quad \langle \psi_{\mathrm{SP},\sigma} | \mathbf{r} \rangle = \psi^*_{\mathrm{SP},\sigma,\mathbf{r}} \tag{3.110}$$

and the normalization

$$\langle \psi_{\mathrm{SP},\sigma} | \psi_{\mathrm{SP},\sigma} \rangle = 1 \,. \tag{3.111}$$

So far we have discretized our one particle Hilbert spaces $\mathcal{H}_\sigma$. In order to be able to describe a system of particles, we need to define discrete field operators that act in a Fock space $\mathcal{F}$ of our system. This Fock space is given by

$$\mathcal{F} = \bigoplus_{N_\uparrow=0}^{N_x^d} \bigoplus_{N_\downarrow=0}^{N_x^d} \left(\mathcal{H}_\uparrow^{N_\uparrow} \otimes \mathcal{H}_\downarrow^{N_\downarrow} \right) \tag{3.112}$$

and its elements can be denoted in occupation number representation. For a state $|\alpha\rangle \in \mathcal{F}$, we have the representation

$$|\alpha\rangle = \left|N_\uparrow; n_{\uparrow,\mathbf{r}_1}, ..., n_{\uparrow,\mathbf{r}_{N_x^d}}\right\rangle \left|N_\downarrow; n_{\downarrow,\mathbf{r}_1}, ..., n_{\downarrow,\mathbf{r}_{N_x^d}}\right\rangle \tag{3.113}$$

for $N_\uparrow$ particles of the up-species and $N_\downarrow$ particles of the down-species. These occupation numbers act as a discrete density field:

$$N_\sigma = \sum_{\mathbf{r}} a_x^d \, n_{\sigma,\mathbf{r}} \,, \tag{3.114}$$

which means we find $a_x^d \, n_{\sigma,\mathbf{r}}$ particles of species σ at the lattice site $\mathbf{r}$. The order of the occupation numbers $n_{\sigma,\mathbf{r}_i}$ in the state is given by the previously defined index function α, i.e.

$$(\mathbf{r_i})_j = x_j^{(\alpha)}(i) \,. \tag{3.115}$$

Before we can introduce the field operators, we need to define the sign exponent that will allow us to capture the sign changes occurring in fermionic states when two particles are interchanged. It is given by

$$s_{\sigma,i} = \sum_{j=1}^{i-1} a_x^d \, n_{\sigma,\mathbf{r}_j} \,. \tag{3.116}$$

Note that $s_{\sigma,i} \in \mathbb{N}_0$ and thus $(-1)^{s_{\sigma,i}} \in \{-1, 1\}$. With this at hand, we can define the creation operators

$$\begin{aligned} \hat{\psi}^\dagger_{\uparrow,\mathbf{r}_i} \left|N_\uparrow; ..., n_{\uparrow,\mathbf{r}_i}, ...\right\rangle \left|N_\downarrow; ...\right\rangle &= (-1)^{s_{\uparrow,i}} \sqrt{n_{\uparrow,\mathbf{r}_i} + a_x^{-d}} \left|N_\uparrow; ..., n_{\uparrow,\mathbf{r}_i} + a_x^{-d}, ...\right\rangle \left|N_\downarrow; ...\right\rangle \,, \\ \hat{\psi}^\dagger_{\downarrow,\mathbf{r}_i} \left|N_\uparrow; ...\right\rangle \left|N_\downarrow; ..., n_{\downarrow,\mathbf{r}_i}, ...\right\rangle &= (-1)^{s_{\downarrow,i}} \sqrt{n_{\downarrow,\mathbf{r}_i} + a_x^{-d}} \left|N_\uparrow; ...\right\rangle \left|N_\downarrow; ..., n_{\downarrow,\mathbf{r}_i} + a_x^{-d}, ...\right\rangle \end{aligned} \tag{3.117}$$

and the annihilation operators

$$\begin{aligned} \hat{\psi}_{\uparrow,\mathbf{r}_i} \left|N_\uparrow; ..., n_{\uparrow,\mathbf{r}_i}, ...\right\rangle \left|N_\downarrow; ...\right\rangle &= (-1)^{s_{\uparrow,i}} \sqrt{n_{\uparrow,\mathbf{r}_i}} \left|N_\uparrow; ..., n_{\uparrow,\mathbf{r}_i} - a_x^{-d}, ...\right\rangle \left|N_\downarrow; ...\right\rangle \,, \\ \hat{\psi}_{\downarrow,\mathbf{r}_i} \left|N_\uparrow; ...\right\rangle \left|N_\downarrow; ..., n_{\downarrow,\mathbf{r}_i}, ...\right\rangle &= (-1)^{s_{\downarrow,i}} \sqrt{n_{\downarrow,\mathbf{r}_i}} \left|N_\uparrow; ...\right\rangle \left|N_\downarrow; ..., n_{\downarrow,\mathbf{r}_i} - a_x^{-d}, ...\right\rangle \,. \end{aligned} \tag{3.118}$$

With these operators, we can define particle number operators $\hat{N}_\sigma$ and local density operators $\hat{n}_{\sigma,\mathbf{r}_i}$ with

$$\hat{N}_\sigma = \sum_{\mathbf{r}_i} a_x^d \, \hat{n}_{\sigma,\mathbf{r}_i} = \sum_{\mathbf{r}_i} a_x^d \, \hat{\psi}^\dagger_{\sigma,\mathbf{r}_i} \hat{\psi}_{\sigma,\mathbf{r}_i} \tag{3.119}$$

and we can use the creation operators to represent the Fock states directly:

$$\left|N_\sigma; n_{\sigma,\mathbf{r}_1}, ..., n_{\sigma,\mathbf{r}_{N_x^d}}\right\rangle = \left(\hat{\psi}^\dagger_{\sigma,\mathbf{r}_1}\right)^{a_x^d \, n_{\sigma,\mathbf{r}_1}} \cdots \left(\hat{\psi}^\dagger_{\sigma,\mathbf{r}_{N_x^d}}\right)^{a_x^d \, n_{\sigma,\mathbf{r}_{N_x^d}}} |0\rangle \,. \tag{3.120}$$

In this representation of the Fock states, we see why it is important to define an order of the lattice sites; interchanging two of the creation operators in the fermionic states introduces a sign change and thus only a defined order of the creation operators can define states with a defined sign. Before we continue to the discrete Hamiltonian, we would like to state the (anti-) commutation relations of the field operators:

$$\left\{\hat{\psi}_{\sigma,\mathbf{r}_i}, \hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\right\} = \delta_{\sigma\sigma'} \delta^{(d)}_{\mathbf{r}_i,\mathbf{r}_j} \quad \text{and} \quad \left\{\hat{\psi}_{\sigma,\mathbf{r}_i}, \hat{\psi}_{\sigma',\mathbf{r}_j}\right\} = \left\{\hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}, \hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\right\} = 0 \,, \tag{3.121}$$

as well as

$$\left[\hat{n}_{\sigma,\mathbf{r}_i}, \hat{\psi}_{\sigma,\mathbf{r}_j}\right] \overset{i\neq j}{=} 0 \overset{i\neq j}{=} \left[\hat{n}_{\sigma,\mathbf{r}_i}, \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right] \tag{3.122}$$

with $[\hat{A}, \hat{B}] = \hat{A}\hat{B} - \hat{B}\hat{A}$.

Discrete Hamiltonian

To write down a discrete representation of the Hamiltonian, we need to define a derivative operator $\hat{D}_\Delta$ that replaces the spatial derivative ∇^2 in the continuous Hamiltonian. To do this, we derive a matrix representation of $\hat{D}_\Delta$ and relate it to well-known finite-difference approximations of derivatives.

For two functions $f, g : \mathbb{R}^d \to \mathbb{C}$ with $f = \Delta g$ and

$$\begin{aligned} \langle \mathbf{r}|f\rangle &= f(\mathbf{r})\,, \\ \langle \mathbf{r}|g\rangle &= g(\mathbf{r})\,, \end{aligned} \tag{3.123}$$

we require

$$\begin{aligned} \langle \mathbf{r}_i|f\rangle &= \langle \mathbf{r}_i|\,\hat{D}_\Delta\,|g\rangle \\ &= \sum_{\mathbf{r}_j} a_x^d \underbrace{\langle \mathbf{r}_i|\,\hat{D}_\Delta\,|\mathbf{r}_j\rangle}_{D'_{\Delta,ij}} \langle \mathbf{r}_j|g\rangle\,, \end{aligned} \tag{3.124}$$

which leads to matrices D'_Δ representing the Laplace operator in the continuum limit $N_x \to \infty$ with $N_x a_x = L = \text{const}$. We write the matrix with a prime here to distinguish it from the space *and* time version of the spatial derivative which we shall define below. As a concrete example for $N_x = 5$ and $d = 1$, when choosing a symmetric finite difference sampling with three points at and around the site of evaluation, i.e.

$$\left.\frac{\partial^2 g}{\partial x^2}\right|_x \approx \frac{g(x-a_x) - 2g(x) + g(x+a_x)}{a_x^2}\,, \tag{3.125}$$

the derivative matrix reads

$$D'_\Delta = \frac{1}{a_x^3}\begin{pmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix}. \tag{3.126}$$

The non-zero entries in the upper right-hand and lower left-hand corner of the matrix are a result of the periodic boundary conditions we have chosen for space on our lattice; it is $g(x+L) = g(x)$. Note that we actually do have a choice in how we approximate the spatial derivative in our theory. As it turns out in the derivation of the path integral representation of the partition function below, this is not the case for the temporal derivative in the action of the system.

Using the matrix elements $D'_{\Delta,ij}$, we can write a second-quantization representation of $\hat{D}_\Delta$:

$$\hat{D}_\Delta = \sum_{\mathbf{r}_i} a_x^d \sum_{\mathbf{r}_j} a_x^d\, D'_{\Delta,ij}\, \hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}\hat{\psi}_{\sigma,\mathbf{r}_j}\,. \tag{3.127}$$

With this last puzzle piece, we can write the discrete Hamiltonian of the system

$$\boxed{\hat{H} = \sum_{\mathbf{r}_i} a_x^d \sum_{\mathbf{r}_j} a_x^d \sum_{\sigma\in\{\uparrow,\downarrow\}} \left(\hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}\frac{D'_\Delta}{2m_\sigma}\hat{\psi}_{\sigma,\mathbf{r}_j}\right) - g\sum_{\mathbf{r}_i} a_x^d\, \hat{\psi}^\dagger_{\uparrow,\mathbf{r}_i}\hat{\psi}_{\uparrow,\mathbf{r}_i}\hat{\psi}^\dagger_{\downarrow,\mathbf{r}_i}\hat{\psi}_{\downarrow,\mathbf{r}_i}}\,, \tag{3.128}$$

which is the discrete-space counterpart to the continuous Hamiltonian in Eq. (3.1).

3.3.3 Discrete Coherent States

With the system represented by the discrete Hamiltonian, we can work towards a path integral representation of the partition function of the system with discrete fields on the lattice. We take the canonical approach to this, which is to define a Fock space basis of coherent states and use it to transform the trace representation of the partition function into a path integral. For our system, we just need to ensure that our coherent basis of the Fock space is simultaneously coherent for both particle species.

We begin by defining the Grassmann-valued fields $\psi_{\sigma,\mathbf{r}}$ and $\psi^*_{\sigma,\mathbf{r}}$ for $\sigma \in \{\uparrow,\downarrow\}$ with

$$\left\{\psi_{\sigma,\mathbf{r}_i},\psi^*_{\sigma',\mathbf{r}_j}\right\} = \left\{\psi_{\sigma,\mathbf{r}_i},\psi_{\sigma',\mathbf{r}_j}\right\} = \left\{\psi^*_{\sigma,\mathbf{r}_i},\psi^*_{\sigma',\mathbf{r}_j}\right\} = 0 \tag{3.129}$$

and

$$\left\{\psi_{\sigma,\mathbf{r}_i},\hat{\psi}_{\sigma',\mathbf{r}_j}\right\} = \left\{\psi_{\sigma,\mathbf{r}_i},\hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\right\} = \left\{\psi^*_{\sigma,\mathbf{r}_i},\hat{\psi}_{\sigma',\mathbf{r}_j}\right\} = \left\{\psi^*_{\sigma,\mathbf{r}_i},\hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\right\} = 0. \tag{3.130}$$

We also define their behavior under Hermitian conjugation as

$$\left(\hat{\tilde{\psi}}\tilde{\psi}\right)^\dagger = \left(\tilde{\psi}^*\hat{\tilde{\psi}}^\dagger\right) \tag{3.131}$$

for every $\hat{\tilde{\psi}} \in \{\hat{\psi}_{\sigma,\mathbf{r}},\hat{\psi}^\dagger_{\sigma,\mathbf{r}}\}$ and every $\tilde{\psi} \in \{\psi_{\sigma,\mathbf{r}},\psi^*_{\sigma,\mathbf{r}}\}$ for arbitrary σ and $\mathbf{r}$.

In the subspace of a single species σ, we can define a coherent state $|\psi_\sigma\rangle$ as

$$|\psi_\sigma\rangle = \exp\left(-\sum_{\mathbf{r}_i} a_x^d\, \psi_{\sigma,\mathbf{r}_i}\hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}\right)|0\rangle\ . \tag{3.132}$$

Such a state is indeed an eigenstate of the annihilation operator:

$$
\begin{aligned}
&\hat{\psi}_{\sigma,\mathbf{r}_i} \exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)|0\rangle \\
&\quad= \hat{\psi}_{\sigma,\mathbf{r}_i} \prod_{\mathbf{r}_j} \exp\left(-a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)|0\rangle \\
&\quad= \hat{\psi}_{\sigma,\mathbf{r}_i} \prod_{\mathbf{r}_j} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)|0\rangle \\
&\quad= \left(\prod_{\mathbf{r}_j \neq \mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right) \hat{\psi}_{\sigma,\mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_i} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}\right)|0\rangle \\
&\quad= \left(\prod_{\mathbf{r}_j} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right) \underbrace{\hat{\psi}_{\sigma,\mathbf{r}_i}|0\rangle}_{=0} + \left(\prod_{\mathbf{r}_j \neq \mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right) \psi_{\sigma,\mathbf{r}_i}|0\rangle \\
&\quad= \left(\prod_{\mathbf{r}_j \neq \mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right) \psi_{\sigma,\mathbf{r}_i}|0\rangle \\
&\quad= \left(\prod_{\mathbf{r}_j \neq \mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right) \left(\psi_{\sigma,\mathbf{r}_i} - a_x^d \underbrace{\psi_{\sigma,\mathbf{r}_i}\psi_{\sigma,\mathbf{r}_i}}_{=0} \hat{\psi}_{\sigma,\mathbf{r}_i}\right)|0\rangle \\
&\quad= \left(\prod_{\mathbf{r}_j \neq \mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right) \psi_{\sigma,\mathbf{r}_i} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_i} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}\right)|0\rangle \\
&\quad= \psi_{\sigma,\mathbf{r}_i} \prod_{\mathbf{r}_j} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)|0\rangle \\
&\quad= \psi_{\sigma,\mathbf{r}_i} \exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)|0\rangle\ ,
\end{aligned}
\tag{3.133}
$$

wherein we used that

$$
\hat{\psi}_{\sigma,\mathbf{r}_i}\left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right) = \begin{cases} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right) \hat{\psi}_{\sigma,\mathbf{r}_i} & (i \neq j) \\ \psi_{\sigma,\mathbf{r}_i} + \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_i} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_i}\right) \hat{\psi}_{\sigma,\mathbf{r}_i} & (i = j) \end{cases}. \tag{3.134}
$$

Since we have

$$
\hat{\psi}_{\sigma',\mathbf{r}_i}\left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right) \overset{\sigma\neq\sigma'}{=} \left(1 - a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right) \hat{\psi}_{\sigma',\mathbf{r}_i} \tag{3.135}
$$

single annihilation operators commute with operator exponentials of different species:

$$
\left[\hat{\psi}_{\sigma',\mathbf{r}_i},\ \exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right] \overset{\sigma\neq\sigma'}{=} 0 \tag{3.136}
$$

and, consequently, even operator exponentials of different species as a whole commute:

$$
\left[\exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\sigma',\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma',\mathbf{r}_j}\right),\ \exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\sigma,\mathbf{r}_j} \hat{\psi}^\dagger_{\sigma,\mathbf{r}_j}\right)\right] \overset{\sigma\neq\sigma'}{=} 0\,. \tag{3.137}
$$

This allows us to use a product of one of these exponential factors for each species acting on the vacuum $|0\rangle$ to create a simultaneous coherent state of both fermion species and define

$$
|\psi_\uparrow, \psi_\downarrow\rangle = \exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\uparrow,\mathbf{r}_j} \hat{\psi}^\dagger_{\uparrow,\mathbf{r}_j}\right) \exp\left(-\sum_{\mathbf{r}_j} a_x^d\, \psi_{\downarrow,\mathbf{r}_j} \hat{\psi}^\dagger_{\downarrow,\mathbf{r}_j}\right)|0\rangle \tag{3.138}
$$

with the corresponding bra

$$\langle \psi_\uparrow, \psi_\downarrow | = \langle 0 | \exp\left(+ \sum_{\mathbf{r}_j} a_x^d \, \psi^*_{\downarrow,\mathbf{r}_j} \hat{\psi}_{\downarrow,\mathbf{r}_j} \right) \exp\left(+ \sum_{\mathbf{r}_j} a_x^d \, \psi^*_{\uparrow,\mathbf{r}_j} \hat{\psi}_{\uparrow,\mathbf{r}_j} \right) . \tag{3.139}$$

These coherent states satisfy the desired eigenvalue relations

$$\hat{\psi}_{\sigma,\mathbf{r}_i} | \psi_\uparrow, \psi_\downarrow \rangle = \psi_{\sigma,\mathbf{r}_i} | \psi_\uparrow, \psi_\downarrow \rangle \quad \text{and} \quad \langle \psi_\uparrow, \psi_\downarrow | \hat{\psi}^\dagger_{\sigma,\mathbf{r}_i} = \langle \psi_\uparrow, \psi_\downarrow | \psi^*_{\sigma,\mathbf{r}_i} , \tag{3.140}$$

as well as the conjugate relations

$$\hat{\psi}^\dagger_{\sigma,\mathbf{r}_i} | \psi_\uparrow, \psi_\downarrow \rangle = -\partial_{\psi_{\sigma,\mathbf{r}_i}} | \psi_\uparrow, \psi_\downarrow \rangle \quad \text{and} \quad \langle \psi_\uparrow, \psi_\downarrow | \hat{\psi}_{\sigma,\mathbf{r}_i} = \partial_{\psi^*_{\sigma,\mathbf{r}_i}} \langle \psi_\uparrow, \psi_\downarrow | . \tag{3.141}$$

Furthermore, they exhibit an orthogonality relation

$$\langle \psi_\uparrow, \psi_\downarrow | \psi_\uparrow, \psi_\downarrow \rangle = \exp\left(\sum_{\mathbf{r}_j} a_x^d \, \psi^*_{\uparrow,\mathbf{r}_j} \psi_{\uparrow,\mathbf{r}_j} \right) \exp\left(\sum_{\mathbf{r}_j} a_x^d \, \psi^*_{\downarrow,\mathbf{r}_j} \psi_{\downarrow,\mathbf{r}_j} \right) \tag{3.142}$$

which implies they form an overcomplete basis of the Fock space:

$$\mathbb{1}_{\mathcal{F}} = \int \mathrm{d}(\psi^*, \psi) \exp\left(- \sum_{\mathbf{r}_i} a_x^d \, \psi^*_{\uparrow,\mathbf{r}_i} \psi_{\uparrow,\mathbf{r}_i} \right) \exp\left(- \sum_{\mathbf{r}_i} a_x^d \, \psi^*_{\downarrow,\mathbf{r}_i} \psi_{\downarrow,\mathbf{r}_i} \right) | \psi_\uparrow, \psi_\downarrow \rangle \langle \psi_\uparrow, \psi_\downarrow | \tag{3.143}$$

with

$$\mathrm{d}(\psi^*, \psi) = \prod_{\mathbf{r}} \mathrm{d}\psi^*_{\uparrow,\mathbf{r}} \mathrm{d}\psi_{\uparrow,\mathbf{r}} \mathrm{d}\psi^*_{\downarrow,\mathbf{r}} \mathrm{d}\psi_{\downarrow,\mathbf{r}} . \tag{3.144}$$

The final feature we need to implement in our coherent states to use them to derive a path integral is a time dependency of the fields. To this end, we give our coherent states a time index

$$| \psi_\uparrow, \psi_\downarrow \rangle \to | \psi_\uparrow, \psi_\downarrow ; i \rangle , \tag{3.145}$$

such that coherent states with a different time index formally are coherent states with different field configurations. We shall refer to the eigenvalues of these states as $\psi_{\sigma,\tau_i,\mathbf{r}_j}$. Coherent states with a time index have the overlap[3]

$$\langle \psi_\uparrow, \psi_\downarrow ; i | \psi_\uparrow, \psi_\downarrow ; j \rangle = \exp\left(\sum_{\mathbf{r}_k} a_x^d \, \psi^*_{\uparrow,\tau_i,\mathbf{r}_k} \psi_{\uparrow,\tau_j,\mathrm{r}_k} \right) \exp\left(\sum_{\mathbf{r}_k} a_x^d \, \psi^*_{\downarrow,\tau_i,\mathbf{r}_k} \psi_{\downarrow,\tau_j,\mathrm{r}_k} \right) \tag{3.146}$$

and for integrals, we denote the time index of the fields in the differential as

$$\mathrm{d}(\psi^*, \psi; i) = \prod_{\mathbf{r}} \mathrm{d}\psi^*_{\uparrow,\tau_i,\mathbf{r}} \mathrm{d}\psi_{\uparrow,\tau_i,\mathbf{r}} \mathrm{d}\psi^*_{\downarrow,\tau_i,\mathbf{r}} \mathrm{d}\psi_{\downarrow,\tau_i,\mathbf{r}} . \tag{3.147}$$

With these “Heisenberg picture” coherent states, we are now ready to derive path integrals for our theory.

[3] This can be shown by using Eq. (3.139) to write the bra and then using the eigenvalue relation of coherent states on the annihilation operator and the ket.

3.3.4 Discrete Fermionic Path Integral

To obtain a path integral formulation of the partition function, we begin with the trace representation of the partition function

$$\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) = \operatorname{tr} e^{-\beta\hat{H}'} \tag{3.148}$$

with

$$\hat{H}' = \hat{H} - \mu_\uparrow\hat{N}_\uparrow - \mu_\downarrow\hat{N}_\downarrow \tag{3.149}$$

and explicitly perform the trace over our coherent basis. To keep the "visual noise" to a minimum in this calculation, we define

$$e^{\varepsilon_{\sigma,i}} = e^{-\sum_{\mathbf{r}} a_x^d\, \psi^*_{\sigma,\tau_i,\mathbf{r}}\psi_{\sigma,\tau_i,\mathbf{r}}} \tag{3.150}$$

and

$$e^{\varepsilon^+_{i,j}} = \left\langle \psi_\uparrow,\psi_\downarrow;i\middle|\psi_\uparrow,\psi_\downarrow;j\right\rangle \ . \tag{3.151}$$

We find

$$\begin{aligned}
\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) &= \operatorname{tr} e^{-\beta\hat{H}'} = \sum_n \langle n|\, e^{-\beta\hat{H}'}\, |n\rangle \\
&= \sum_n \int \mathrm{d}(\psi^*,\psi;1)\; e^{\varepsilon_{\uparrow,1}} e^{\varepsilon_{\downarrow,1}}\, \left\langle n\middle|\psi_\uparrow,\psi_\downarrow;1\right\rangle \left\langle \psi_\uparrow,\psi_\downarrow;1\middle|\, e^{-\beta\hat{H}'}\, |n\right\rangle \\
&= \sum_n \int \mathrm{d}(\psi^*,\psi;1)\; e^{\varepsilon_{\uparrow,1}} e^{\varepsilon_{\downarrow,1}}\, \left\langle -\psi_\uparrow,-\psi_\downarrow;1\middle|\, e^{-\beta\hat{H}'}\, |n\right\rangle \left\langle n\middle|\psi_\uparrow,\psi_\downarrow;1\right\rangle \\
&= \int \mathrm{d}(\psi^*,\psi;1)\; e^{\varepsilon_{\uparrow,1}} e^{\varepsilon_{\downarrow,1}}\, \left\langle -\psi_\uparrow,-\psi_\downarrow;1\middle|\, e^{-\beta\hat{H}'}\, \middle|\psi_\uparrow,\psi_\downarrow;1\right\rangle \ .
\end{aligned} \tag{3.152}$$

We cannot evaluate the matrix elements of $e^{-\beta\hat{H}'}$ in the coherent basis directly because, even though $\hat{H}'$ itself is in normal order, its operator exponential is not. To surmount the problem we divide the imaginary time interval $[0,\beta)$ in N_τ steps of length a_τ, such that

$$\beta = N_\tau a_\tau \tag{3.153}$$

and rewrite

$$e^{-\beta\hat{H}'} = \prod_{i=1}^{N_\tau} e^{-a_\tau\hat{H}'} \ . \tag{3.154}$$

Each of these imaginary time slices is then normal ordered up to a correction of order $\mathcal{O}(a_\tau^2)$ since

$$e^{-a_\tau\hat{H}'} = 1 - a_\tau\hat{H}' + \mathcal{O}(a_\tau^2)\,. \tag{3.155}$$

This means that in our lattice calculations, we can approximate systems for arbitrary values of β by choosing sufficiently small values for a_τ. For convenience, we also define our fields outside the time interval $[0,\beta)$ through

$$\left\langle -\psi_\uparrow,-\psi_\downarrow;1\right| =: \left\langle \psi_\uparrow,\psi_\downarrow;N_\tau+1\right| \ , \tag{3.156}$$

which implies the anti-periodic boundary conditions

$$\psi_{\sigma,\tau_{N_\tau+1},\mathbf{r}} = -\psi_{\sigma,\tau_1,\mathbf{r}} \quad \text{and} \quad \psi^*_{\sigma,\tau_{N_\tau+1},\mathbf{r}} = -\psi^*_{\sigma,\tau_1,\mathbf{r}} \tag{3.157}$$

for the fermions of our theory. With this discretization of the imaginary time interval, we can calculate the transfer-matrix element

$$\left\langle \psi_\uparrow,\psi_\downarrow;i\middle|\, e^{-a_\tau\hat{H}'}\, \middle|\psi_\uparrow,\psi_\downarrow;j\right\rangle = e^{-a_\tau H'(i,j)} e^{\varepsilon^+_{i,j}} + \mathcal{O}(a_\tau^2) \tag{3.158}$$

where the Hamiltonian function is given by

$$H'(i,j) = \left\langle \psi_\uparrow, \psi_\downarrow; i \right| \hat{H}' \left| \psi_\uparrow, \psi_\downarrow; j \right\rangle . \tag{3.159}$$

The concrete expression for this Hamiltonian function can trivially be obtained by utilizing the normal ordering of the Hamiltonian operator; each creation operator can be replaced by its corresponding starred field at time index i and each annihilation operator can be replaced by its corresponding, non-starred field at time index j. Replacing the $e^{-\beta\hat{H}'}$ operator in our path integral with the time-sliced version and inserting the identities of coherent states in Eq. (3.143) in between the slices we obtain

$$\begin{aligned}
\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) &= \int \left(\prod_{i=1}^{N_\tau} \mathrm{d}(\psi^*,\psi;i) \right) \; e^{\varepsilon_{\uparrow,1}} e^{\varepsilon_{\downarrow,1}} \\
&\qquad \cdot \left\langle \psi_\uparrow, \psi_\downarrow; N_\tau + 1 \right| e^{\varepsilon_{\uparrow,N_\tau}} e^{\varepsilon_{\downarrow,N_\tau}} e^{-a_\tau \hat{H}'} \left| \psi_\uparrow, \psi_\downarrow; N_\tau \right\rangle \\
&\qquad \cdot \ldots \cdot \left\langle \psi_\uparrow, \psi_\downarrow; 2 \right| e^{\varepsilon_{\uparrow,2}} e^{\varepsilon_{\downarrow,2}} e^{-a_\tau \hat{H}'} \left| \psi_\uparrow, \psi_\downarrow; 1 \right\rangle \\
&= \int \underbrace{\left(\prod_{i=1}^{N_\tau} \mathrm{d}(\psi^*,\psi;i) \right)}_{=:\mathcal{D}(\psi^*,\psi)} \prod_{i=1}^{N_\tau} e^{\varepsilon_{\uparrow,i}} e^{\varepsilon_{\downarrow,i}} e^{\varepsilon^+_{i+1,i}} e^{-a_\tau H'(i+1,i)} \\
&= \int \mathcal{D}(\psi^*,\psi) \; e^{+\sum_{i=1}^{N_\tau} \left(\varepsilon_{\uparrow,i} + \varepsilon_{\downarrow,i} + \varepsilon^+_{i+1,i} - a_\tau H'(i+1,i) \right)} \\
&= \int \mathcal{D}(\psi^*,\psi) \; e^{-\mathcal{S}_\mathrm{F}[\psi^*,\psi]} .
\end{aligned} \tag{3.160}$$

Here we have introduced the fermionic action

$$\mathcal{S}_\mathrm{F}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} \left(-\varepsilon_{\uparrow,i} - \varepsilon_{\downarrow,i} - \varepsilon^+_{i+1,i} + a_\tau H'(i+1,i) \right) . \tag{3.161}$$

While we now formally have a (lattice) path integral representation with a Euclidean action, its expression will obviously need a bit of "massaging" before we can make any sense of it. So far, all of the terms formally are discrete versions of spatial integrals, except for the Hamiltonian, so we define a discrete Hamiltonian density

$$H'(i,j) =: \sum_{\mathbf{r}} a_x^d \, h'(i,j) \tag{3.162}$$

to be able to factor out the spatial integrations. Doing this and putting the overcompleteness factors ε and the overlaps ε^+ back in, we obtain

$$\begin{aligned}
\mathcal{S}_\mathrm{F}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} a_\tau \sum_{\mathbf{r}} a_x^d \Bigg(& \frac{\psi^*_{\uparrow,\tau_i,\mathbf{r}} \psi_{\uparrow,\tau_i,\mathbf{r}}}{a_\tau} + \frac{\psi^*_{\downarrow,\tau_i,\mathbf{r}} \psi_{\downarrow,\tau_i,\mathbf{r}}}{a_\tau} \\
& - \frac{\psi^*_{\uparrow,\tau_{i+1},\mathbf{r}} \psi_{\uparrow,\tau_i,\mathbf{r}}}{a_\tau} - \frac{\psi^*_{\downarrow,\tau_{i+1},\mathbf{r}} \psi_{\downarrow,\tau_i,\mathbf{r}}}{a_\tau} \\
& + h'(i+1,i) \Bigg) .
\end{aligned} \tag{3.163}$$

In this expression, we can identify discrete derivatives of the fermion fields

$$\mathcal{S}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} a_\tau \sum_{\mathbf{r}} a_x^d \left(\frac{\psi^*_{\uparrow,\tau_i,\mathbf{r}} - \psi^*_{\uparrow,\tau_{i+1},\mathbf{r}}}{a_\tau} \psi_{\uparrow,\tau_i,\mathbf{r}} + \frac{\psi^*_{\downarrow,\tau_i,\mathbf{r}} - \psi^*_{\downarrow,\tau_{i+1},\mathbf{r}}}{a_\tau} \psi_{\downarrow,\tau_i,\mathbf{r}} + h'(i+1,i) \right) . \tag{3.164}$$

Performing a "summation by parts", as described in App. B, we can move the discrete-time derivative to the un-starred fields:

$$\mathcal{S}_{\mathrm{F}}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} a_\tau \sum_{\mathbf{r}} a_x^d \left(\psi^*_{\uparrow,\tau_i,\mathbf{r}} \frac{\psi_{\uparrow,\tau_i,\mathbf{r}} - \psi_{\uparrow,\tau_{i-1},\mathbf{r}}}{a_\tau} + \psi^*_{\downarrow,\tau_i,\mathbf{r}} \frac{\psi_{\downarrow,\tau_i,\mathbf{r}} - \psi_{\downarrow,\tau_{i-1},\mathbf{r}}}{a_\tau} + h'(i+1,i) \right) . \quad (3.165)$$

In this expression, we recognize the discrete form of the expected continuous fermionic action from Eq. (3.8)

$$\begin{aligned} S_{\mathrm{F}}[\psi^*,\psi] &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \left(\psi^*_\uparrow \partial_\tau \psi_\uparrow + \psi^*_\downarrow \partial_\tau \psi_\downarrow + h' \right) \\ &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^d r \left(\psi^*_\uparrow \left(\partial_\tau - \mu_\uparrow\right) \psi_\uparrow + \psi^*_\downarrow \left(\partial_\tau - \mu_\downarrow\right) \psi_\downarrow + h \right) , \end{aligned} \quad (3.166)$$

with spatial derivatives and interaction absorbed into a Hamiltonian density h. However, through the rigorous derivation, we have been given a concrete prescription for the discretization of the time derivative by the construction of the path integral.

We analyze the action further by inserting the expression for the Hamiltonian density

$$\begin{aligned} \sum_{\mathbf{r}_j} a_x^d \, h'(i+1,i) = & \\ & - \sum_{\mathbf{r}_j} a_x^d \sum_{\mathbf{r}_k} a_x^d \sum_\sigma \psi^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \frac{D'_{\Delta,jk}}{2m_\sigma} \psi_{\sigma,\tau_i,\mathbf{r}_k} \\ & + g \sum_{\mathbf{r}_j} a_x^d \, \psi^*_{\uparrow,\tau_{i+1},\mathbf{r}_j} \psi^*_{\downarrow,\tau_{i+1},\mathbf{r}_j} \psi_{\uparrow,\tau_i,\mathbf{r}_j} \psi_{\downarrow,\tau_i,\mathbf{r}_j} \\ & - \sum_{\mathbf{r}_j} a_x^d \sum_\sigma \mu_\sigma \psi^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \psi_{\sigma,\tau_i,\mathbf{r}_j} \end{aligned} \quad (3.167)$$

into the action in Eq. (3.165) and obtain

$$\begin{aligned} \mathcal{S}_{\mathrm{F}}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} a_\tau \sum_{\mathbf{r}_j} a_x^d \Bigg[& \sum_\sigma \psi^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \frac{\psi_{\sigma,\tau_{i+1},\mathbf{r}_j} - (1+a_\tau\mu_\sigma)\psi_{\sigma,\tau_i,\mathbf{r}_j}}{a_\tau} \\ & - \sum_{\mathbf{r}_k} a_x^d \sum_\sigma \psi^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \frac{D'_{\Delta,jk}}{2m_\sigma} \psi_{\sigma,\tau_i,\mathbf{r}_k} \\ & + g \, \psi^*_{\uparrow,\tau_{i+1},\mathbf{r}_j} \psi^*_{\downarrow,\tau_{i+1},\mathbf{r}_j} \psi_{\uparrow,\tau_i,\mathbf{r}_j} \psi_{\downarrow,\tau_i,\mathbf{r}_j} \Bigg] , \end{aligned} \quad (3.168)$$

wherein we suitably combined the $-\mu\hat{N}$ terms with the temporal derivatives. While this expression technically has everything we need for a simulation, working with this plain form is quite tedious. Thus, in the following, we introduce some techniques that make the handling of this discrete theory easier.

3.3.5 Restoration of the Silver-Blaze Symmetry

When looking at the action of our system in the continuum, we notice that the temporal derivative of the fermionic field and their chemical potentials group together in a way that is reminiscent of a covariant derivative; we find terms like

$$\psi^*(\tau)(\partial_\tau - \mu)\psi(\tau) , \quad (3.169)$$

ignoring species and space for now. The chemical potential seems to enter the action in the form of a constant temporal gauge field and, indeed, in the continuum we find a simultaneous transformation of the chemical potential and the fermion field that leaves the action invariant, namely transforming

$$\mu \to \mu + i\alpha \quad \text{while} \quad \psi(\tau) \to e^{i\alpha\tau}\psi(\tau)\,. \tag{3.170}$$

We can verify this invariance:

$$\begin{aligned}\psi^*(\tau)e^{-i\alpha\tau}(\partial_\tau - \mu - i\alpha)e^{i\alpha\tau}\psi(\tau) &= \psi^*(\tau)e^{-i\alpha\tau}e^{i\alpha\tau}(i\alpha + \partial_\tau - \mu - i\alpha)\psi(\tau)\\ &= \psi^*(\tau)(\partial_\tau - \mu)\psi(\tau)\,.\end{aligned} \tag{3.171}$$

This property is called the *Silver-Blaze symmetry*; see, e.g., Refs. [57–62] for detailed discussions. In our discretized theory, however, this symmetry is broken. If we look at the derivative operator

$$d_\tau(\mu)\,\psi(\tau) = \frac{\psi(\tau) - \psi(\tau - a_\tau)}{a_\tau} - \mu\psi(\tau - a_\tau) = \frac{\psi(\tau) - (1 + \mu a_\tau)\psi(\tau - a_\tau)}{a_\tau} \tag{3.172}$$

which describes the temporal derivative in the discrete action in Eq. (3.168) and perform a Silver-Blaze transformation on a derivative with this operator, we find

$$\begin{aligned}&\psi^*(\tau)e^{-i\alpha\tau}\,d_\tau(\mu + i\alpha)\,e^{i\alpha\tau}\psi(\tau)\\ &\quad= \psi^*(\tau)e^{-i\alpha\tau}\left(\frac{e^{i\alpha\tau}\psi(\tau) - e^{i\alpha(\tau - a_\tau)}\psi(\tau - a_\tau)}{a_\tau} - (\mu + i\alpha)e^{i\alpha(\tau - a_\tau)}\psi(\tau - a_\tau)\right)\\ &\quad= \psi^*(\tau)\left(\frac{\psi(\tau) - e^{-i\alpha a_\tau}\psi(\tau - a_\tau)}{a_\tau} - (\mu + i\alpha)e^{-i\alpha a_\tau}\psi(\tau)\right)\\ &\quad\neq \psi^*(\tau)\,d_\tau(\mu)\,\psi(\tau)\,.\end{aligned} \tag{3.173}$$

Thus, the introduction of discrete time has indeed destroyed this symmetry. As Ref. [63] states, this is a problem, because it leads to divergences in the continuum limit that scale quadratically with $1/a_\tau$ and, thus, limit the accuracy of our calculations in the regime of small chemical potentials. To remedy this, we can use a modified version of the discrete-time derivative that preserves the Silver-Blaze symmetry. We can obtain such a derivative by replacing the term $(1 + \mu a_\tau)$ in the rightmost expression in Eq. (3.172) with $e^{\mu a_\tau}$. Since $(1 + \mu a_\tau)$ constitutes the first two orders of a power series expansion of $e^{\mu a_\tau}$,

$$e^{\mu a_\tau} - (1 + \mu a_\tau) = \mathcal{O}\left(a_\tau^2\right)\,, \tag{3.174}$$

this replacement removes the aforementioned quadratic divergences in the continuum limit. The Silver-Blaze conserving discrete temporal derivative is given by the operator

$$d_\tau^{(\mathrm{SB})}(\mu)\,\psi(\tau) = \frac{\psi(\tau) - e^{\mu a_\tau}\psi(\tau - a_\tau)}{a_\tau}\,. \tag{3.175}$$

Indeed, we can verify that

$$\begin{aligned}&\psi^*(\tau)e^{-i\alpha\tau}\,d_\tau^{(\mathrm{SB})}(\mu + i\alpha)\,e^{i\alpha\tau}\psi(\tau)\\ &\quad= \psi^*(\tau)e^{-i\alpha\tau}\left(\frac{e^{i\alpha\tau}\psi(\tau) - e^{(\mu + i\alpha)a_\tau}e^{i\alpha(\tau - a_\tau)}\psi(\tau - a_\tau)}{a_\tau}\right)\\ &\quad= \psi^*(\tau)\left(\frac{\psi(\tau) - e^{\mu a_\tau}e^{+i\alpha a_\tau}e^{-i\alpha a_\tau}\psi(\tau - a_\tau)}{a_\tau}\right)\\ &\quad= \psi^*(\tau)\left(\frac{\psi(\tau) - e^{\mu a_\tau}\psi(\tau - a_\tau)}{a_\tau}\right)\\ &\quad= \psi^*(\tau)\,d_\tau^{(\mathrm{SB})}(\mu)\,\psi(\tau)\,.\end{aligned} \tag{3.176}$$

With this correction incorporated into the action in Eq. (3.168), the Silver-Blaze-respecting discrete action reads

$$\begin{aligned}\mathcal{S}_{\mathrm{F}}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} a_\tau \sum_{\mathbf{r}_j} a_x^d \Bigg[&\sum_\sigma \psi^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \frac{\psi_{\sigma,\tau_{i+1},\mathbf{r}_j} - e^{\mu a_\tau}\psi_{\sigma,\tau_i,\mathbf{r}_j}}{a_\tau} \\ &- \sum_{\mathbf{r}_k} a_x^d \sum_\sigma \psi^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \frac{D'_{\Delta,jk}}{2m_\sigma} \psi_{\sigma,\tau_i,\mathbf{r}_k} \\ &+ g\, \psi^*_{\uparrow,\tau_{i+1},\mathbf{r}_j} \psi^*_{\downarrow,\tau_{i+1},\mathbf{r}_j} \psi_{\uparrow,\tau_i,\mathbf{r}_j} \psi_{\downarrow,\tau_i,\mathbf{r}_j} \Bigg] .\end{aligned} \tag{3.177}$$

3.3.6 Dimensionless Formulation

The first step towards improved readability is given by the rescaling of the theory in terms of dimensionless fields and constants. Beyond the "visual" advantages, it also prevents the numerical stability from depending on the choice of the unit system, since the numerical values of dimensionless fields and constants do not change between unit systems. At first, we note that with our conventions of $\hbar = k_{\mathrm{B}} = 1$, actions have a dimension of 1, i.e. they are already dimensionless. As for our lattice constants, we assign the dimensions of time T and length L:

$$[a_\tau] = \mathsf{T} \quad \text{and} \quad [a_x] = \mathsf{L} \tag{3.178}$$

and we define our masses to be dimensionless

$$[m_\uparrow] = [m_\downarrow] = 1 . \tag{3.179}$$

Looking at the spacetime integrals over the time derivative terms in the action, we can conclude the dimensions of the fields in our theory:

$$[\psi_\sigma] = \mathsf{L}^{-d/2} . \tag{3.180}$$

With our definition $[m] = 1$, kinetic terms like

$$\sum a_x^d\, \psi^* \frac{\psi}{a_x^2} , \tag{3.181}$$

which appear in a Hamiltonian, dictate that the dimension of energy E is given by

$$\mathsf{E} = \mathsf{L}^{-2} \tag{3.182}$$

and since we know from the Boltzmann factor $e^{-\beta H}$ that the inverse temperature $\beta = 1/T$ carries dimension $[\beta] = \mathsf{E}^{-1}$, we can conclude that time and length are connected in the following way:

$$\mathsf{L}^2 = \mathsf{T} = \mathsf{E}^{-1} . \tag{3.183}$$

Using this relation between time and space, we can define a dimensionless factor r describing the scale between the temporal and spatial lattice spacings:

$$a_x^2 \cdot r = a_\tau . \tag{3.184}$$

Looking at the dispersion relation of the system in the case of $m = 1$, a value of $r = 1/2$ adjusts the momentum cutoff to be at the momentum that corresponds to the energy cutoff. Therefore, we use a value of $r = 1/2$ for all calculations in this work. For the coupling constant, we find

$$[g] = \mathsf{L}^d \mathsf{T}^{-1} \tag{3.185}$$

and the entries of the Laplace matrix have dimension

$$[D'_{\Delta,jk}] = \mathsf{T}^{-1}\mathsf{L}^{-d} = \mathsf{L}^{-2}\mathsf{L}^{-d} \,. \tag{3.186}$$

We can now multiply the fields and constants with appropriate powers of the lattice spacings a_τ and a_x and obtain the dimensionless fields

$$\tilde{\psi}_\sigma = a_x^{d/2}\psi_\sigma \tag{3.187}$$

as well as the dimensionless constants and matrix entries

$$\tilde{g} = a_\tau a_x^{-d} g \,, \quad \tilde{D'}_{\Delta,jk} = r a_x^2 a_x^d D'_{\Delta,jk} \quad \text{and} \quad \tilde{\mu}_\sigma = a_\tau \mu_\sigma \,. \tag{3.188}$$

In the case of the coupling g and the chemical potentials μ_σ, it is also beneficial to relate the dimensionless constants $\tilde{g}$ and $\tilde{\mu}_\sigma$ to the commonly used dimensionless variants $\lambda = \beta^{1-d/2} g$ and $\beta\mu_\sigma$, as they appear in, e.g., Ref. [12]. We find

$$\tilde{g} = r^{d/2} N_\tau^{d/2-1} \lambda \quad \text{and} \quad \tilde{\mu}_\sigma = \frac{\beta\mu_\sigma}{N_\tau} \,. \tag{3.189}$$

Inserting the dimensionless constants and fields into our dimensionful, discretized, Silver-Blaze-respecting action in Eq. (3.177), we find the dimensionless action

$$\begin{aligned} \mathcal{S}_\mathrm{F}[\psi^*,\psi] = \sum_{i=1}^{N_\tau} \sum_{\mathbf{r}_j} \Bigg[& \sum_\sigma \tilde{\psi}^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \left(\tilde{\psi}_{\sigma,\tau_{i+1},\mathbf{r}_j} - e^{\tilde{\mu}_\sigma} \tilde{\psi}_{\sigma,\tau_i,\mathbf{r}_j} \right) \\ & - \sum_{\mathbf{r}_k} \sum_\sigma \tilde{\psi}^*_{\sigma,\tau_{i+1},\mathbf{r}_j} \frac{\tilde{D'}_{\Delta,jk}}{2m_\sigma} \tilde{\psi}_{\sigma,\tau_i,\mathbf{r}_k} \\ & + \tilde{g} \left(\tilde{\psi}^*_{\uparrow,\tau_{i+1},\mathbf{r}_j} \tilde{\psi}^*_{\downarrow,\tau_{i+1},\mathbf{r}_j} \tilde{\psi}_{\uparrow,\tau_i,\mathbf{r}_j} \tilde{\psi}_{\downarrow,\tau_i,\mathbf{r}_j} \right) \Bigg] \,, \end{aligned} \tag{3.190}$$

which is now free of all lattice spacings and all remaining quantities are dimensionless. As we use the dimensionless formulation of the theory for the remainder of this work, we **omit the tildes** from now on for the sake of readability.

3.3.7 Matrix Notation

Another technique that makes the handling of the lattice theory significantly easier and more efficient is the notation of fields in vectors and operators in matrices. We define the field configuration vectors as column vectors containing the field values at each lattice site:

$$\psi_\sigma = \begin{pmatrix} \psi_{\sigma,(\tau,\mathbf{r})_1} \\ \vdots \\ \psi_{\sigma,(\tau,\mathbf{r})_{N_\tau N_x^d}} \end{pmatrix} . \tag{3.191}$$

The order of the lattice sites in the field configuration vectors is again given by the index function α, as for $(\tau,\mathbf{r})_i$ it is

$$i = \alpha(\tau,\mathbf{r}) \tag{3.192}$$

or inversely

$$\tau = \tau^{(\alpha)}(i) \quad \text{and} \quad \mathbf{r} = \begin{pmatrix} x_1^{(\alpha)}(i) \\ \vdots \\ x_d^{(\alpha)}(i) \end{pmatrix} . \tag{3.193}$$

We define our field configuration vector to connect starred and un-starred fields under Hermitian adjungation:

$$(\psi_\sigma^\dagger)_i = \psi^*_{\sigma,(\tau,\mathbf{r})_i} = (\psi^*_\sigma)_i \,. \tag{3.194}$$

An advantage of this notation is that the scalar product of two such field configuration vectors constitutes the spacetime integral over two fields:

$$\psi_\sigma^\dagger \psi_\sigma = \sum_{i=1}^{N_\tau} \sum_{\mathbf{r}_j} \psi^*_{\sigma,\tau_i,\mathbf{r}_j} \psi_{\sigma,\tau_i,\mathbf{r}_j} \,. \tag{3.195}$$

In the interaction term in the action in Eq. (3.190) we have to bring four field configurations together in a product, which we cannot do by means of a scalar product. For this situation, we use the *Hadamard-* or *element-wise* product $\circ$. The Hadamard-product of two vectors is itself a vector with its entries being the products of the entries of the factor vectors:

$$(\psi_\sigma^{(a)} \circ \psi_\sigma^{(b)})_i = (\psi_\sigma^{(a)})_i \cdot (\psi_\sigma^{(b)})_i \,. \tag{3.196}$$

This way we can write a generic four-fermion term as

$$(\psi^*_\uparrow \circ \psi^*_\downarrow)^\intercal (\psi_\uparrow \circ \psi_\downarrow) = \sum_{i=1}^{N_\tau} \sum_{\mathbf{r}_j} \psi^*_{\uparrow,\tau_i,\mathbf{r}_j} \psi^*_{\downarrow,\tau_i,\mathbf{r}_j} \psi_{\uparrow,\tau_i,\mathbf{r}_j} \psi_{\downarrow,\tau_i,\mathbf{r}_j} \,. \tag{3.197}$$

One may notice that we would never see the above term in a physical action because the construction of the path integral causes starred and un-starred fields within a product to be evaluated at neighboring points in time, rather than the same point. This is where our next important concept comes in: *coordinate shift matrices*. For each coordinate x_k for $k = 0, \ldots, d$, with $x_0 = \tau$ for convenience, we can define a *coordinate advancer matrix* $A_\pm^{(x_k)}$ and a *coordinate retarder matrix* $R_\pm^{(x_k)}$. They are defined to move the field configurations in a vector one step forward (advance) or backward (retard) in the given coordinate direction. The plus or minus denotes, whether they do so respecting periodic (+) or anti-periodic (-) boundary conditions in the given direction.

The coordinate retarder matrices are defined as

$$R^{(x_k)}_{\pm,ij} := \begin{cases} 1 & i^{(\alpha)}_{x_k}(i) = i^{(\alpha)}_{x_k}(j) + 1 \\ \pm 1 & i^{(\alpha)}_{x_k} = 1 \wedge i^{(\alpha)}_{x_k}(j) = \begin{cases} N_\tau & k = 0 \\ N_x & k = 1, \ldots, d \end{cases} \\ 0 & \text{otherwise} \end{cases} \tag{3.198}$$

and, rather than formulating an explicit definition, we can obtain the coordinate advancer matrices by exploiting a relation between coordinate advancer matrices and coordinate retarder matrices. By design, it must be

$$\left(R_\pm^{(x_d)}\right)^{-1} = A_\pm^{(x_d)} = \left(R_\pm^{(x_d)}\right)^\intercal \,, \tag{3.199}$$

since shifting a field configuration one step forward and one step back along a given direction has to result in the original field configuration. The same argument also demonstrates that coordinate shift matrices commute pair-wise. The orthogonality of the coordinate shift matrices can be proven using “summation by parts” (see App. B). For example, in the temporal direction,

for generic fields ξ^* and ξ that respect the same boundary conditions as $A_\pm^{(\tau)}$, we find:

$$\begin{aligned}
\xi^\dagger A_\pm^{(\tau)} \xi &= \sum_{i=1}^{N_\tau} \sum_{\mathbf{r}_j} \xi^*_{\tau_i,\mathbf{r}_j} A_\pm^{(\tau)} \xi_{\tau_i,\mathbf{r}_j} \\
&= \sum_{i=1}^{N_\tau} \sum_{\mathbf{r}_j} \xi^*_{\tau_i,\mathbf{r}_j} \xi_{\tau_{i+1},\mathbf{r}_j} \\
&= \sum_{\mathbf{r}_j} \left(\sum_{i=1}^{N_\tau - 1} \xi^*_{\tau_i,\mathbf{r}_j} \xi_{\tau_{i+1},\mathbf{r}_j} + \xi^*_{\tau_{N_\tau},\mathbf{r}_j} \xi_{\tau_{N_\tau+1},\mathbf{r}_j} \right) \\
&= \sum_{\mathbf{r}_j} \left(\sum_{i=2}^{N_\tau} \xi^*_{\tau_{i-1},\mathbf{r}_j} \xi_{\tau_i,\mathbf{r}_j} + \xi^*_{\tau_{N_\tau},\mathbf{r}_j} \xi_{\tau_{N_\tau+1},\mathbf{r}_j} \right) \\
&= \sum_{\mathbf{r}_j} \left(\sum_{i=2}^{N_\tau} \xi^*_{\tau_{i-1},\mathbf{r}_j} \xi_{\tau_i,\mathbf{r}_j} + \xi^*_{\tau_0,\mathbf{r}_j} \xi_{\tau_1,\mathbf{r}_j} \right) \\
&= \sum_{i=1}^{N_\tau} \sum_{\mathbf{r}_j} \xi^*_{\tau_{i-1},\mathbf{r}_j} \xi_{\tau_i,\mathbf{r}_j} \\
&= \left(R_\pm^{(\tau)} \xi^* \right)^\intercal \xi \\
&= \xi^\dagger \left(R_\pm^{(\tau)} \right)^\intercal \xi
\end{aligned} \tag{3.200}$$

An example of a complete set of coordinate retarder and coordinate advancer matrices for a lattice with $N_\tau = N_x = 3$ and $d = 2$ is given in Fig. 3.2.

Of particular importance for the present work are the coordinate shift matrices in the temporal direction. Therefore, we define

$$R := R_-^{(\tau)} \tag{3.201}$$

as *the retarder matrix* and

$$A := A_-^{(\tau)} \tag{3.202}$$

as *the advancer matrix*. We restrict ourselves to the shift matrices that respect anti-periodic boundary conditions in time since this represents the behavior of the fermionic fields of our theory.

In this handling of our theory, the coordinate shift matrices allow us to easily define derivative operators. We can use coordinate shift matrices to translate a given derivative approximation prescription into a matrix representation of the derivative operator. For example, in spatial directions, we choose to approximate the second derivative as

$$\left. \frac{\partial^2 g}{\partial x^2} \right|_x \approx \frac{g(x - a_x) - 2g(x) + g(x + a_x)}{a_x^2} \,. \tag{3.203}$$

With that, for each coordinate direction k, we can directly translate the approximation prescription into the derivative matrix

$$D_{\Delta_k} = R_+^{(x_k)} - 2 \cdot \mathbb{1} + A_+^{(x_k)} \tag{3.204}$$

with the identity $\mathbb{1}$ and free of any a_x spacings, since we have absorbed them in the rescaling of our fields in Sec. 3.3.6. The matrix representation of the full Laplace operator then reads

$$D_\Delta = \sum_{k=1}^{d} D_{\Delta_k} = -2d \cdot \mathbb{1} + \sum_{k=1}^{d} \left(R_+^{(x_k)} + A_+^{(x_k)} \right) \,. \tag{3.205}$$

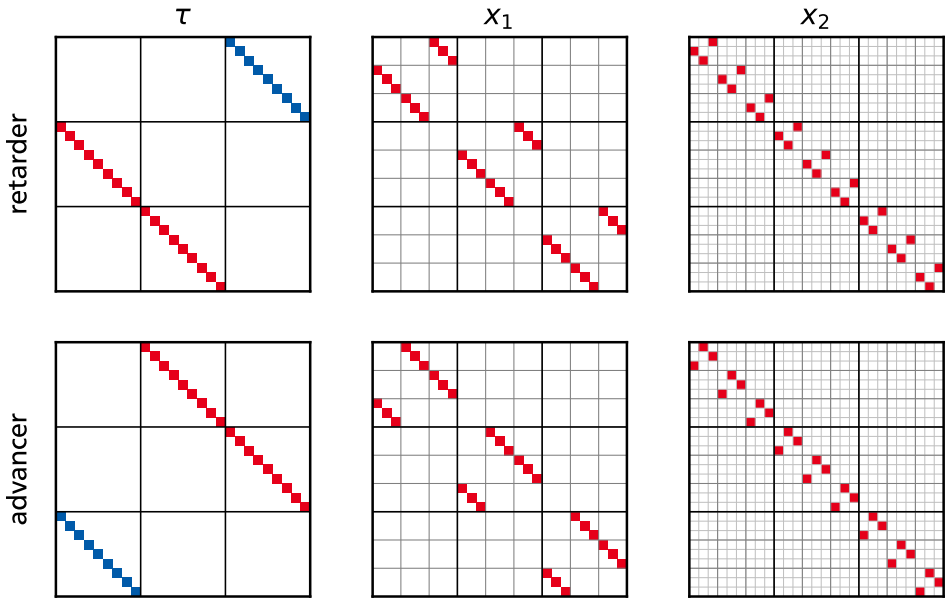


Figure 3.2: Examples for coordinate shift matrices for a lattice with $N_\tau = N_x = 3$ and $d = 2$, resulting in 27×27-matrices. Red squares denote an entry of $+1$, and blue squares an entry of -1. The shift matrices in the temporal direction are calculated for anti-periodic boundary conditions, which results in negative entries. The grids separate regions of constant coordinates. For example, if we multiply a matrix with a field configuration vector, all entries within a grid cell only affect field values at which the coordinate of the column is constant.

For temporal derivatives, we can proceed in an analog manner. In the derivation of the path integral, there naturally emerges a "backward" approximation of the temporal derivative:

$$\left.\frac{\partial g}{\partial \tau}\right|_\tau \approx \frac{g(\tau) - g(\tau - a_\tau)}{a_\tau} \,. \tag{3.206}$$

This prescription can be represented by the matrix

$$D_\tau^{(\mathrm{bw,natural})} = \mathbb{1} - R \,, \tag{3.207}$$

again without the temporal lattice spacing a_τ, as it has been absorbed by the rescaling of our theory in Sec. 3.3.6. In Sec. 3.3.5, we discussed that we need to adjust this derivative to include the chemical potential in an exponential term in order to preserve the Silver-Blaze symmetry of the continuum theory. This corresponds to the derivative approximation prescription

$$\left.\frac{\partial g}{\partial \tau}\right|_\tau - \mu g(\tau) \approx \frac{g(\tau) - e^{\mu a_\tau} g(\tau - a_\tau)}{a_\tau} \,. \tag{3.208}$$

This prescription leads to the derivative matrix

$$D_\tau^{(\mathrm{bw})}(\mu) = \mathbb{1} - e^{\mu} R \,, \tag{3.209}$$

the dimensionless chemical potential μ and, again, without temporal lattice spacing a_τ. A corresponding "forward" derivative would take the form

$$D_\tau^{(\mathrm{fw})}(\mu) = e^{\mu} A - \mathbb{1} \,. \tag{3.210}$$

This representation of the derivative matrices in terms of coordinate shift matrices has a big practical advantage in the implementation of the calculation in code; we only need to define a single function to generate coordinate retarder matrices. All advancer matrices can be obtained from their respective retarder matrices through inversion or transposition and all derivative matrices can be constructed from simple expressions of coordinate shift matrices. This way, all complicated "juggling" of lattice indices and ordering is contained in a single, small, and easy-to-test function, which makes the generation of derivative matrices far less prone to errors in code.

Using the vector representation of field configurations and derivative matrices, we rewrite the dimensionless action in Eq. (3.190) in the compact form

$$\mathcal{S}_{\mathrm{F}}[\psi^*,\psi] = \sum_{\sigma}\left(\psi_\sigma^\dagger D_\tau^{(\mathrm{bw})}(\mu_\sigma)\psi_\sigma - \psi_\sigma^\dagger \frac{RD_\Delta}{2m_\sigma}\psi_\sigma\right) + g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R(\psi_\uparrow \circ \psi_\downarrow)\,. \tag{3.211}$$

In this form, the connection to the continuous action in Eq. (3.8) is very apparent. The two expressions almost read the same, except we have derivative matrices instead of derivative operators and some coordinate shift matrices that are not present in the continuum theory. What makes the expression for the discretized action arguably even more compact than the expression for the continuum action is the fact that we no longer need to write the spacetime integrals. Through the definition of the field configuration vectors, they are encoded in the matrix products of this representation.

3.3.8 Bosonization of the Discrete Theory: The Discrete Pairing Field

With the preparations out of the way, we can now start bosonizing our theory with the action in Eq. (3.211). In principle, we aim to perform a Hubbard-Stratonovich transformation similar to the one we performed in the continuum in Sec. 3.1. However, on the lattice, there are some subtle difficulties we need to deal with. Analogously to the Hubbard-Stratonovich transformation in the continuum, we define a factor of one:

$$1 = \int \mathcal{D}(\boldsymbol{\phi}^*,\boldsymbol{\phi})\; e^{-\boldsymbol{\phi}^\dagger \mathcal{M}_{\boldsymbol{\phi}} \boldsymbol{\phi}} \tag{3.212}$$

with auxiliary complex field configuration vectors $\boldsymbol{\phi}^*$ and $\boldsymbol{\phi}$ and a yet to be defined matrix $\mathcal{M}_{\boldsymbol{\phi}}$. In this expression, we define the path integral differential proportional to the field configuration entry differentials:

$$\mathcal{D}(\boldsymbol{\phi}^*,\boldsymbol{\phi}) \propto \prod_{i=1}^{N} \mathrm{d}\phi_i^* \mathrm{d}\phi_i \tag{3.213}$$

with the lattice size $N = N_\tau N_x^d$ and a proportionality such that Eq. (3.212) holds. When we insert this factor of one into the fermionic action in the path integral

$$\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) = \int \mathcal{D}(\boldsymbol{\psi}^*,\boldsymbol{\psi})\; e^{-\mathcal{S}_{\mathrm{F}}[\boldsymbol{\psi}^*,\boldsymbol{\psi}]} \tag{3.214}$$

and reorganize the terms in the form

$$\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) = \int \mathcal{D}(\boldsymbol{\psi}^*,\boldsymbol{\psi},\boldsymbol{\phi}^*,\boldsymbol{\phi})\; e^{-\mathcal{S}^*_{\mathrm{PB}}[\boldsymbol{\psi}^*,\boldsymbol{\psi},\boldsymbol{\phi}^*,\boldsymbol{\phi}]}\,, \tag{3.215}$$

we find a general pre-partially bosonized action

$$\mathcal{S}^*_{\mathrm{PB}}[\boldsymbol{\psi}^*,\boldsymbol{\psi},\boldsymbol{\phi}^*,\boldsymbol{\phi}] = \mathcal{S}_{\mathrm{kin}}[\boldsymbol{\psi}^*,\boldsymbol{\psi}] + g(\boldsymbol{\psi}_\uparrow^* \circ \boldsymbol{\psi}_\downarrow^*) R(\boldsymbol{\psi}_\uparrow \circ \boldsymbol{\psi}_\downarrow) + \boldsymbol{\phi}^\dagger \mathcal{M}_{\boldsymbol{\phi}} \boldsymbol{\phi} \tag{3.216}$$

with the kinetic terms in $\mathcal{S}_{\text{kin}}$. The Hubbard-Stratonovich transformation is now completed by performing a shift of the integration variables ϕ^* and ϕ:

$$\phi^* \to \phi^* + \phi_S^* \quad \text{and} \quad \phi \to \phi + \phi_S\,, \tag{3.217}$$

leading to the general partially-bosonized action

$$\begin{aligned}\mathcal{S}_{\text{PB}}[\psi^*,\psi,\phi^*,\phi] = \mathcal{S}_{\text{kin}}[\psi^*,\psi] + g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R(\psi_\uparrow \circ \psi_\downarrow) \\ + \phi^\dagger \mathcal{M}_\phi \phi + \phi_S^\dagger \mathcal{M}_\phi \phi + \phi^\dagger \mathcal{M}_\phi \phi_S + \phi_S^\dagger \mathcal{M}_\phi \phi_S\,.\end{aligned} \tag{3.218}$$

The Hubbard-Stratonovich transformation has served its intended purpose if this shift creates a term that counters the four fermion interaction term, i.e.,

$$\phi_S^\dagger \mathcal{M}_\phi \phi_S = -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R(\psi_\uparrow \circ \psi_\downarrow)\,. \tag{3.219}$$

So far, this is no different from the procedure in the continuum. Therefore, one might think that we can simply use the discretized version of the Hubbard-Stratonovich transformation we used in the continuum with the shifts[4] and $\mathcal{M}_\phi$ matrix chosen as follows:

$$\phi_S^* = -(\psi_\uparrow^* \circ \psi_\downarrow^*)\,, \quad \phi_S = (\psi_\uparrow \circ \psi_\downarrow) \quad \text{and} \quad \mathcal{M}_\phi = g\mathbb{1}\,. \tag{3.220}$$

That choice, however, does not succeed in removing the four-fermion interaction term, as

$$\begin{aligned}\phi_S^\dagger \mathcal{M}_\phi \phi_S &= -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal (\psi_\uparrow \circ \psi_\downarrow) \\ &\neq -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R(\psi_\uparrow \circ \psi_\downarrow)\,.\end{aligned} \tag{3.221}$$

This choice of transformation parameters fails to account for the shift in field times we find on the lattice. We might think we can include this shift in our Hubbard-Stratonovich transformation by choosing

$$\phi_S^* = -(\psi_\uparrow^* \circ \psi_\downarrow^*)\,, \quad \phi_S = (\psi_\uparrow \circ \psi_\downarrow) \quad \text{and} \quad \mathcal{M}_\phi = gR\,, \tag{3.222}$$

leading to

$$\phi_S^\dagger \mathcal{M}_\phi \phi_S = -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R(\psi_\uparrow \circ \psi_\downarrow)\,, \tag{3.223}$$

and while it seems this choice leads to the desired result, it has an even bigger problem; the definition of the unity factor that we use to start the Hubbard-Stratonovich in Eq. (3.212) requires the matrix $\mathcal{M}_\phi$ to be positive-definite in order to be well-defined and that is not the case for the retarder matrix R. The solution to this predicament is to shift the auxiliary fields ϕ^* and ϕ to pairs of fermionic field configuration vectors at different times:

$$\phi_S^* = -A(\psi_\uparrow^* \circ \psi_\downarrow^*)\,, \quad \phi_S = (\psi_\uparrow \circ \psi_\downarrow) \quad \text{and} \quad \mathcal{M}_\phi = g\mathbb{1}\,. \tag{3.224}$$

The matrix $\mathcal{M}_\phi = g\mathbb{1}$ is positive-definite, and we can cancel the four-fermion interaction term:

$$\begin{aligned}\phi_S^\dagger \mathcal{M}_\phi \phi_S &= \left(-A(\psi_\uparrow^* \circ \psi_\downarrow^*)\right)^\intercal g\mathbb{1}(\psi_\uparrow \circ \psi_\downarrow) \\ &= -g(\psi_\uparrow^* \circ \psi_\downarrow^*)A^\intercal(\psi_\uparrow \circ \psi_\downarrow) \\ &= -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R(\psi_\uparrow \circ \psi_\downarrow)\,.\end{aligned} \tag{3.225}$$

[4]The minus sign in the shift term ϕ_S^* is present because, compared to the shift term of the continuum Hubbard-Stratonovich transformation, the order of the fermion field in the expression $(\psi_\uparrow^* \circ \psi_\downarrow^*)$ is reversed. This creates a sign because of the anti-commutation relation of the Grassmann-valued fermion fields.

The resulting partially-bosonized action reads

$$\begin{aligned}\mathcal{S}_{\mathrm{PB}}[\psi^*,\psi,\phi^*,\phi] = \sum_\sigma \left(\psi_\sigma^\dagger D_\tau^{(\mathrm{bw})}(\mu_\sigma)\psi_\sigma - \psi_\sigma^\dagger \frac{RD_\Delta}{2m_\sigma}\psi_\sigma\right) \\ + g\phi^\dagger\phi + g\left(\phi^\dagger(\psi_\uparrow\circ\psi_\downarrow) - (\psi_\uparrow^*\circ\psi_\downarrow^*)^\intercal R\phi\right).\end{aligned} \tag{3.226}$$

At first glance, this Hubbard-Stratonovich transformation seems different from the one we use in the continuum. However, the time advancement of the shift of ϕ^* vanishes in the continuum limit, as $a_\tau \to 0$. In fact, seeing the starred fermionic fields $(\psi_\uparrow^*\circ\psi_\downarrow^*)$ evaluated at a point in time advanced with respect to the unstarred fields seems like an expected and natural result given our observations on time shifting in the derivation of the path integral.

To integrate out the fermions, analogously to the continuum, we define Nambu-Gorkov spinors

$$\psi_{\mathrm{NG}}^* = \begin{pmatrix}\psi_\uparrow^*\\ \psi_\downarrow\end{pmatrix} \quad \text{and} \quad \psi_{\mathrm{NG}} = \begin{pmatrix}\psi_\uparrow\\ \psi_\downarrow^*\end{pmatrix}, \tag{3.227}$$

which are $2N_\tau N_x^d$-element spinors that each contain two $N_\tau N_x^d$-element field configuration vectors. This allows us to write

$$\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) = \int \mathcal{D}(\phi^*\phi)\, \underbrace{\int \mathcal{D}(\psi^*,\psi)\, e^{-\mathcal{S}_{\mathrm{FC}}[\psi^*,\psi,\phi^*,\phi]}}_{\mathcal{Z}_{\mathrm{FC}}}\, e^{-\mathcal{S}_{\mathrm{PA}}[\phi^*,\phi]} \tag{3.228}$$

with the purely auxiliary part

$$\mathcal{S}_{\mathrm{PA}}[\phi^*,\phi] = g\phi^\dagger\phi \tag{3.229}$$

and the fermionic contribution

$$\mathcal{S}_{\mathrm{FC}}[\psi^*,\psi,\phi^*,\phi] = \psi_{\mathrm{NG}}^\dagger\, \mathcal{M}\, \psi_{\mathrm{NG}}, \tag{3.230}$$

where $\mathcal{M}$ is the fermion matrix. We find

$$\mathcal{Z}_{\mathrm{FC}} = \det\mathcal{M}, \tag{3.231}$$

which allows us to write

$$\mathcal{Z}(\beta,\mu_\uparrow,\mu_\downarrow) = \int \mathcal{D}(\phi^*,\phi)\, e^{-\mathcal{S}_{\mathrm{B}}[\phi^*,\phi]} \tag{3.232}$$

with the fully bosonized action

$$\mathcal{S}_{\mathrm{B}}[\phi^*,\phi] = g\phi^\dagger\phi - \log\det\mathcal{M}. \tag{3.233}$$

The only thing left to do is to determine the entries of the fermion matrix $\mathcal{M}$. We begin with the ansatz

$$\mathcal{M} = \begin{pmatrix}\mathcal{M}_A & \mathcal{M}_B\\ \mathcal{M}_C & \mathcal{M}_D\end{pmatrix}. \tag{3.234}$$

which results in the fermionic contribution

$$\mathcal{S}_{\mathrm{FC}}[\psi^*,\psi,\phi^*,\phi] = \psi_\uparrow^\dagger\mathcal{M}_A\psi_\uparrow + \psi_\uparrow^\dagger\mathcal{M}_B\psi_\downarrow^* + \psi_\downarrow^\intercal\mathcal{M}_C\psi_\uparrow + \psi_\downarrow^\intercal\mathcal{M}_D\psi_\downarrow^*. \tag{3.235}$$

To determine the matrices $\mathcal{M}_A$, $\mathcal{M}_B$, $\mathcal{M}_C$, and $\mathcal{M}_D$, we compare this expression to the partially-bosonized action in Eq. (3.226), group all terms according to their combinations of fermionic field configuration vectors and rearrange them to match the order of field configuration vectors in Eq. (3.235).

For the matrix $\mathcal{M}_A$, the field configuration vectors are already in the right order and we find

$$\mathcal{M}_A = D_\tau^{(\mathrm{bw})}(\mu_\uparrow) - \frac{RD_\triangle}{2m_\uparrow}\,. \tag{3.236}$$

For the matrix $\mathcal{M}_B$, we find

$$\begin{aligned} -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R\phi &= -g\,\psi_\uparrow^\dagger \,\mathrm{diag}(R\phi)\psi_\downarrow^* \\ \Rightarrow \quad \mathcal{M}_B &= -g\,\mathrm{diag}(R\phi)\,. \end{aligned} \tag{3.237}$$

In this expression, all we had to do to obtain the desired form is to write the retarded pairing field configuration as a diagonal matrix between the fermionic field configuration vectors rather than using the Hadamard product to combine three vectors into a scalar. The equivalence of these two expressions can easily be verified by writing all Hadamard- and matrix products in both expressions as sums:

$$\begin{aligned} -g(\psi_\uparrow^* \circ \psi_\downarrow^*)^\intercal R\phi &= -g\sum_{i=1}^{N} \psi^*_{\uparrow,\tau_i,\mathbf{r}_i}\psi^*_{\downarrow,\tau_i,\mathbf{r}_i}\phi_{\tau_{i-1},\mathbf{r}_i} \\ &= -g\sum_{i=1}^{N} \psi^*_{\uparrow,\tau_i,\mathbf{r}_i}\phi_{\tau_{i-1},\mathbf{r}_i}\psi^*_{\downarrow,\tau_i,\mathbf{r}_i} \\ &= -g\,\psi_\uparrow^\dagger\,\mathrm{diag}(R\phi)\psi_\downarrow^*\,. \end{aligned} \tag{3.238}$$

For the matrix entry $\mathcal{M}_C$, we have

$$\begin{aligned} g(\psi_\uparrow \circ \psi_\downarrow)^\intercal \phi^* &= -g\left(\psi_\downarrow \circ \psi_\uparrow\right)^\intercal \phi^* \\ &= -g\,\psi_\downarrow^\intercal\,\mathrm{diag}(\phi^*)\psi_\uparrow \\ \Rightarrow \quad \mathcal{M}_C &= -g\,\mathrm{diag}(\phi^*)\,. \end{aligned} \tag{3.239}$$

To obtain this block entry of $\mathcal{M}$, we rewrote the expression to contain the pairing field configuration as a diagonal matrix, just like we did for $\mathcal{M}_B$. Before that, we switched the fermionic fields in the Hadamard product, which creates a sign due to them being anti-commutative. This can, again, easily be verified by rewriting the Hadamard product in terms of sums.

For the final block entry $\mathcal{M}_D$ of $\mathcal{M}$, we have

$$\begin{aligned} \psi_\downarrow^\dagger D_\tau^{(\mathrm{bw})}(\mu_\downarrow)\psi_\downarrow - \psi_\downarrow^\dagger\frac{RD_\triangle R}{2m_\downarrow}\psi_\downarrow &= \left(\psi_\downarrow^\dagger D_\tau^{(\mathrm{bw})}(\mu_\downarrow)\psi_\downarrow\right)^\intercal - \left(\psi_\downarrow^\dagger\frac{RD_\triangle}{2m_\downarrow}\psi_\downarrow\right)^\intercal \\ &= \psi_\downarrow^\intercal\left(D_\tau^{(\mathrm{bw})}(\mu_\downarrow)\right)^\intercal\psi_\downarrow^* - \psi_\downarrow^\intercal\frac{D_\triangle^\intercal A}{2m_\downarrow}\psi_\downarrow^*\,. \end{aligned} \tag{3.240}$$

To find the transposes of the derivative matrices, we represent them using coordinate shift matrices and exploit their properties. The transposition of the backward time derivative matrix leads to

$$\begin{aligned} \left(D_\tau^{(\mathrm{bw})}(\mu_\downarrow)\right)^\intercal &= (\mathbb{1} - e^{\mu_\downarrow}R)^\intercal \\ &= \mathbb{1} - e^{\mu_\downarrow}A \\ &= -\left(e^{\mu_\downarrow}A - \mathbb{1}\right) \\ &= -D_\tau^{(\mathrm{fw})}(\mu_\downarrow)\,. \end{aligned} \tag{3.241}$$

The Laplace operator matrix is invariant under this operation, due to our choice of a symmetric approximation of the derivative:

$$\begin{aligned} D_{\Delta}^{\intercal} &= -2d \cdot \mathbb{1}^{\intercal} + \sum_{k=1}^{d} \left(R_{+}^{(x_d)} + A_{+}^{(x_d)} \right)^{\intercal} \\ &= -2d \cdot \mathbb{1} + \sum_{k=1}^{d} \left(R_{+}^{(x_d)} + A_{+}^{(x_d)} \right) \\ &= D_{\Delta} \,. \end{aligned} \tag{3.242}$$

With the transposed derivative matrices, we find

$$\mathcal{M}_D = D_{\tau}^{(\mathrm{fw})}(\mu_{\downarrow}) + \frac{AD_{\Delta}}{2m_{\downarrow}} \,. \tag{3.243}$$

Before we write down the fully bosonized action with the, now determined, fermion matrix, let us take a moment to appreciate how simple it was to determine the entries of the fermion matrix from the partially-bosonized action using our matrix notation. In the continuum, determining the lower right-hand entry of the fermion matrix $\mathcal{M}$ required the lengthy and tedious calculation in App. A, containing multiple Fourier transformations. On the lattice, without the matrix notation, it would have required us to perform "summations by part" described in App. B, for all coordinate directions, but with the matrix notation, all it took were a few transpositions and the use of general properties of the coordinate shift matrices.

The fully bosonized action with the determined entries of the fermion matrix reads

$$\boxed{\mathcal{S}_{\mathrm{B}}[\phi^*, \phi] = g\phi^{\dagger}\phi - \log\det \begin{pmatrix} D_{\tau}^{(\mathrm{bw})}(\mu_{\uparrow}) - \frac{RD_{\Delta}}{2m_{\uparrow}} & -g\,\mathrm{diag}(R\phi) \\ -g\,\mathrm{diag}(\phi^*) & D_{\tau}^{(\mathrm{fw})}(\mu_{\downarrow}) + \frac{AD_{\Delta}}{2m_{\downarrow}} \end{pmatrix}} \,. \tag{3.244}$$

Our system is now fully bosonized and we can move on to the simulation of the system and the calculations of observables.

4 Implementing a Numerical Simulation of the System

With the fully bosonized and discretized formulation of our theory at hand, we can move on to implementing a simulation of our system.

4.1 The Langevin Equation

To avoid the sign problem of the pairing-field formalism, we choose the *Complex-Langevin* (CL) quantization scheme to calculate observables of a system described by the bosonized action $\mathcal{S}_{\mathrm{B}}$. Like in the real Langevin approach, we obtain a quantum solution of the system by solving the Langevin equation. To formulate this equation, we first need to determine the degrees of freedom in our calculations and the drift- and noise terms of the Langevin equation.

4.1.1 Complexified Degrees of Freedom

The CL approach allows us to circumvent the sign problem in the simulation of this system, but it requires us to complexify our real degrees of freedom. The degrees of freedom in our lattice theory are the two complex fields ϕ^* and ϕ or, alternatively, the two real fields $\mathrm{Re}\,\phi$ and $\mathrm{Im}\,\phi$. The latter two real fields are the ones we promote to complex fields in order to use CL. To this end, we define the two *complex* CL fields ϕ_1 and ϕ_2 with

$$\phi = \phi_1 + i\phi_2 \quad \text{and} \quad \phi^* = \phi_1 - i\phi_2\,. \tag{4.1}$$

By this definition, they essentially play the roles of the real- and imaginary part of the complex fields ϕ^* and ϕ. However, since they are, themselves, complex fields, we choose to name them "one" and "two" to avoid possible confusion. In fact, for systems with a mild sign problem, such as the one we study here, we expect the CL fields ϕ_1 and ϕ_2 to be dominated by their real parts.

4.1.2 Drift and Noise Terms

In principle, we find a scalar Langevin equation for every single entry of the CL fields ϕ_1 and ϕ_2. For clarity, we can group them together in two vector equations for the evolution ϕ_1 and ϕ_2, respectively. To calculate the drift in such vector equations, we define a derivative with respect to a field configuration vector by grouping together the derivatives with respect to each entry of the vector:

$$\left(\frac{\partial \mathcal{S}_{\mathrm{B}}}{\partial \phi_{1|2}}\right)_i = \frac{\partial \mathcal{S}_{\mathrm{B}}}{\partial (\phi_{1|2})_i}\,, \tag{4.2}$$

for a field configuration vector $\phi_{1|2}$.

Drift Terms

The bosonized action is given by

$$\mathcal{S}_{\mathrm{B}} = \underbrace{g\,\phi^{\dagger}\phi}_{\mathcal{S}_{\mathrm{PA}}}\underbrace{-\log\det\mathcal{M}}_{\mathcal{S}_{\mathrm{FC}}} \tag{4.3}$$

and we can calculate the drift contributions for the pure-auxiliary contribution to the action $\mathcal{S}_{\mathrm{PA}}$ and the fermionic contribution to the action $\mathcal{S}_{\mathrm{FC}}$ in the above equation separately. We begin by substituting the fields ϕ^* and ϕ in $\mathcal{S}_{\mathrm{PA}}$ for our CL fields:

$$\begin{aligned}\mathcal{S}_{\mathrm{PA}} &= g\,(\phi_1 - i\phi_2)^{\intercal}(\phi_1 + i\phi_2)\\ &= g\,(\phi_1^{\intercal}\phi_1 + \phi_2^{\intercal}\phi_2)\,.\end{aligned} \tag{4.4}$$

This way, we can easily obtain the purely auxiliary contribution to the drift

$$K_{\mathrm{PA}} = \frac{\partial\mathcal{S}_{\mathrm{PA}}}{\partial\phi_{1|2}} = 2g\,\phi_{1|2}\,, \tag{4.5}$$

for ϕ_1 and ϕ_2, respectively.

For the fermionic contributions to the drift, we need to calculate the derivatives of

$$\mathcal{S}_{\mathrm{FC}} = -\log\det\begin{pmatrix} D_{\tau}^{(\mathrm{bw})}(\mu_{\uparrow}) - \frac{RD_{\Delta}}{2m_{\uparrow}} & -g\,\mathrm{diag}\,(R(\phi_1 + i\phi_2)) \\ -g\,\mathrm{diag}(\phi_1 - i\phi_2) & D_{\tau}^{(\mathrm{fw})}(\mu_{\downarrow}) + \frac{AD_{\Delta}}{2m_{\downarrow}}\end{pmatrix}. \tag{4.6}$$

We find

$$\begin{aligned}K_{\mathrm{FC}} &= \frac{\partial\mathcal{S}_{\mathrm{FC}}}{\partial\phi_{1|2}}\\ &= -\frac{\partial}{\partial\phi_{1|2}}\log\det\mathcal{M}\\ &= -\,\mathrm{tr}\left(\frac{\partial}{\partial\phi_{1|2}}\log\mathcal{M}\right)\\ &= -\,\mathrm{tr}\left(\left(\frac{\partial}{\partial\phi_{1|2}}\mathcal{M}\right)\mathcal{M}^{-1}\right)\,,\end{aligned} \tag{4.7}$$

wherein we use

$$\mathrm{tr}\log\mathcal{M} = \log\det\mathcal{M}\,, \tag{4.8}$$

as well as the derivative chain rule for matrix logarithms. Since the derivative of the fermion matrix with respect to a single field value is, itself, a matrix, the vector derivative of the fermion matrix is a tensor of third order:

$$\left(\frac{\partial}{\partial\phi_{1|2}}\mathcal{M}\right)_{ijk} = \left(\frac{\partial}{\partial(\phi_{1|2})_i}\mathcal{M}\right)_{jk}\,. \tag{4.9}$$

What this means is that we have to calculate the trace of a fermion matrix derivative multiplied by the inverse fermion matrix for every lattice site:

$$(K_{\mathrm{FC}})_i = -\,\mathrm{tr}\left(\left(\frac{\partial}{\partial(\phi_{1|2})_i}\mathcal{M}\right)\mathcal{M}^{-1}\right)\,. \tag{4.10}$$

The derivative tensor is given by

$$\frac{\partial}{\partial(\phi_1)_i}\mathcal{M} = \begin{pmatrix} 0 & -g\ \mathrm{diag}(R\,\mathbf{e}_i) \\ -g\ \mathrm{diag}(\mathbf{e}_i) & 0 \end{pmatrix} \tag{4.11}$$

for ϕ_1 and

$$\frac{\partial}{\partial(\phi_2)_i}\mathcal{M} = \begin{pmatrix} 0 & -ig\ \mathrm{diag}(R\,\mathbf{e}_i) \\ ig\ \mathrm{diag}(\mathbf{e}_i) & 0 \end{pmatrix} \tag{4.12}$$

for ϕ_2. Note that these matrices are of dimension $2N_\tau N_x^d \times 2N_\tau N_x^d$ but, both, only contain two non-zero entries. As such, they are predestined to be treated as sparse matrices in a simulation of the system. Moreover, they are constant and do not change between time steps of the simulation. As such, for a given set of simulation parameters, we only need to compute them once. The inverse fermion matrix $\mathcal{M}^{-1}$ depends on the field configurations at the current point in Langevin time and, thus, needs to be recomputed for every step in Langevin time. We can ease the computational effort of this matrix inversion by exploiting the block nature of the fermion matrix. The fermion matrix at Langevin time step i is given by

$$\mathcal{M}^{(i)} = \begin{pmatrix} \mathcal{M}_A & \mathcal{M}_B^{(i)} \\ \mathcal{M}_C^{(i)} & \mathcal{M}_D \end{pmatrix}, \tag{4.13}$$

with the four matrix entries $\mathcal{M}_A$, $\mathcal{M}_B^{(i)}$, $\mathcal{M}_C^{(i)}$ and $\mathcal{M}_D$. The entries $\mathcal{M}_A$ and $\mathcal{M}_D$ are constant and given by

$$\mathcal{M}_A = D_\tau^{(\mathrm{bw})}(\mu_\uparrow) - \frac{RD_\Delta}{2m_\uparrow} \tag{4.14}$$

and

$$\mathcal{M}_D = D_\tau^{(\mathrm{fw})}(\mu_\downarrow) + \frac{AD_\Delta}{2m_\downarrow}\,. \tag{4.15}$$

They may be viewed as the inverse propagators of the fermion fields. The off-diagonal matrix entries $\mathcal{M}_B^{(i)}$ and $\mathcal{M}_C^{(i)}$ are the blocks that contain the pairing field configuration at the current step in Langevin time. They are given by

$$\mathcal{M}_B^{(i)} = -g\,\mathrm{diag}\left(R\left(\phi_1^{(i)} + i\phi_2^{(i)}\right)\right) \tag{4.16}$$

and

$$\mathcal{M}_C^{(i)} = -g\,\mathrm{diag}\left(\phi_1^{(i)} - i\phi_2^{(i)}\right). \tag{4.17}$$

The inverse fermion matrix in this notation can be written as

$$\left(\mathcal{M}^{(i)}\right)^{-1} = \begin{pmatrix} \mathcal{M}_A^{-1} + \mathcal{M}_A^{-1}\mathcal{M}_B^{(i)}F^{(i)}\mathcal{M}_C^{(i)}\mathcal{M}_A^{-1} & -\mathcal{M}_A^{-1}\mathcal{M}_B^{(i)}F^{(i)} \\ -F^{(i)}\mathcal{M}_C^{(i)}\mathcal{M}_A^{-1} & F^{(i)} \end{pmatrix}, \tag{4.18}$$

in which $F^{(i)}$ is the inverse of the Shur complement of block $\mathcal{M}_A$ in $\mathcal{M}^{(i)}$:

$$F^{(i)} = \left(\mathcal{M}_D - \mathcal{M}_C^{(i)}\mathcal{M}_A^{-1}\mathcal{M}_B^{(i)}\right)^{-1}. \tag{4.19}$$

Regarding existence, such a block inversion is possible if the inverses $\mathcal{M}_A^{-1}$ and $F^{(i)}$ exist. For $\mathcal{M}_A$, this can be shown using the representation in coordinate shift matrices, and the inverse matrix $\mathcal{M}_A^{-1}$ can be pre-computed at the beginning of the simulation and reused for every Langevin time step. For $F^{(i)}$, it is conceivable that an unfavorable round of noise makes it impossible to compute $F^{(i)}$ for the problematic step in Langevin time. In that case, the inversion of the fermion matrix

can be performed directly without exploiting its block nature. However, this scenario is highly unlikely.

Performing this block matrix inversion leads to a significant reduction in computational effort for every step in Langevin time because, rather than inverting the fermion matrix $\mathcal{M}^{(i)}$ with dimension $2N_\tau N_x^d$ directly at every step, we invert $(\mathcal{M}_D - \mathcal{M}_C^{(i)}\mathcal{M}_A^{-1}\mathcal{M}_B^{(i)})$ with dimension $N_\tau N_x^d$. In the block inversion of $\mathcal{M}^{(i)}$ we need to perform multiple matrix multiplications of the blocks. However, the blocks $\mathcal{M}_B^{(i)}$ and $\mathcal{M}_C^{(i)}$ are diagonal, which makes matrix multiplications involving them very cheap.

Noise Terms

For the noise term of the Langevin equation, we are free to distribute the noise between the real- and imaginary parts of ϕ_1 and ϕ_2 within the constraints of the fluctuation-dissipation theorem. However, since we expect the sign problem in the system to be mild, the dynamics of the system are mostly carried by the real parts of ϕ_1 and ϕ_2. As such, purely real noise seems like a good choice to efficiently evolve the CL fields into equilibrium. This rationale is formally studied in Ref. [48], which argues that one should use as little imaginary noise as possible, ideally zero, to reduce the uncertainty of calculated observables. With these considerations, we choose the purely real noise terms

$$N_{1|2}(t_{\mathrm{CL}}) = \sqrt{2\delta t_{\mathrm{CL}}}\,\eta_{1|2}(t_{\mathrm{CL}}) \tag{4.20}$$

with noise $\eta_{1|2}(t_{\mathrm{CL}})$ from a standard normal distribution:

$$\langle \eta_{1|2}(t_{\mathrm{CL}})\rangle = 0 \quad \text{and} \quad \langle \eta_{1|2}^2(t_{\mathrm{CL}})\rangle = 1\,. \tag{4.21}$$

Inserting the drift and noise terms into the Langevin equation (2.39), we find

$$\boxed{\phi_{1|2}^{(i+1)} = \phi_{1|2}^{(i)} - \left[2g\,\phi_{1|2}^{(i)} - \mathrm{tr}\left(\left(\frac{\partial}{\partial\phi_{1|2}}\mathcal{M}\right)\left(\mathcal{M}^{(i)}\right)^{-1}\right)\right]\delta t_{\mathrm{CL}} + \sqrt{2\delta t_{\mathrm{CL}}}\,\eta_{1|2}(t_{\mathrm{CL}})}, \tag{4.22}$$

for ϕ_1 and ϕ_2, respectively. Because it is constant, the derivative of the fermion matrix is independent of the Langevin time step i. On the other hand, the fermion matrix at Langevin time step i is assembled using the CL field configurations at that Langevin time step:

$$\mathcal{M}^{(i)} = \begin{pmatrix} D_\tau^{(\mathrm{bw})}(\mu_\uparrow) - \frac{RD_\triangle}{2m_\uparrow} & -g\,\mathrm{diag}\left(R\left(\phi_1^{(i)} + i\phi_2^{(i)}\right)\right) \\ -g\,\mathrm{diag}\left(\phi_1^{(i)} - i\phi_2^{(i)}\right) & D_\tau^{(\mathrm{fw})}(\mu_\downarrow) + \frac{AD_\triangle}{2m_\downarrow} \end{pmatrix}. \tag{4.23}$$

4.2 Langevin Observables

To calculate observables from the Langevin process, we need to relate observable operators to an expression that depends on the fields of the path integral representation of the partition function of the theory. That means, in an expression like

$$\begin{aligned}\langle\hat{O}\rangle &= \frac{1}{\mathcal{Z}}\,\mathrm{tr}\left(\hat{O}\,e^{-\beta(\hat{H}-\mu_\uparrow\hat{N}_\uparrow-\mu_\downarrow\hat{N}_\downarrow)}\right) \\ &= \frac{1}{\mathcal{Z}}\int \mathcal{D}(\phi^*,\phi)\;\mathcal{O}_O(\phi^*,\phi)\,e^{-\mathcal{S}_{\mathrm{B}}}\end{aligned} \tag{4.24}$$

with a given observable operator $\hat{O}$, we need to find the expression $\mathcal{O}_O(\phi^*,\phi)$. This expectation value is approximated by the Langevin process through

$$\frac{1}{\mathcal{Z}}\int \mathcal{D}(\phi^*,\phi)\;\mathcal{O}_O(\phi^*,\phi)\,e^{-\mathcal{S}_{\mathrm{B}}} \approx \langle\mathcal{O}_O\rangle_{\mathrm{CL}} \tag{4.25}$$

with the Langevin expectation value

$$\langle \mathcal{O}_O \rangle_{\mathrm{CL}} = \frac{1}{N_{\mathrm{CL}}} \sum_{i=0}^{N_{\mathrm{CL}}} \mathcal{O}_O^{(i)} \,. \tag{4.26}$$

In the sum over all steps of the Markov chain, $\mathcal{O}_O^{(i)}$ is equivalent to $\mathcal{O}_O(\phi^*, \phi)$ with the fields ϕ^* and ϕ expressed in terms of the CL fields ϕ_1 and ϕ_2 at Langevin time step i.

4.2.1 Density

For the particle number N_σ of fermion species σ we find

$$N_\sigma = \frac{1}{\mathcal{Z}} \operatorname{tr}\left(\hat{N}_\sigma \, e^{-\beta(\hat{H} - \sum_\sigma \mu_\sigma \hat{N}_\sigma)} \right) \tag{4.27}$$

in the operator formalism. To find the operator expression $\mathcal{O}_{N_\sigma}(\phi^*, \phi)$ in the path-integral formalism, we make use of a Standard Technique in statistical physics and express the particle number in terms of a derivative with respect to the corresponding chemical potential:

$$\begin{aligned} N_\sigma &= \frac{1}{\mathcal{Z}} \operatorname{tr}\left(\hat{N}_\sigma \, e^{-\beta(\hat{H} - \sum_\sigma \mu_\sigma \hat{N}_\sigma)} \right) \\ &= \frac{1}{\mathcal{Z}} \operatorname{tr}\left(\frac{1}{\beta} \partial_{\mu_\sigma} \, e^{-\beta(\hat{H} - \sum_\sigma \mu_\sigma \hat{N}_\sigma)} \right) \\ &= \frac{1}{\mathcal{Z}} \frac{1}{\beta} \partial_{\mu_\sigma} \operatorname{tr}\left(e^{-\beta(\hat{H} - \sum_\sigma \mu_\sigma \hat{N}_\sigma)} \right) \\ &= \frac{1}{\mathcal{Z}} \frac{1}{\beta} \partial_{\mu_\sigma} \mathcal{Z} \,. \end{aligned} \tag{4.28}$$

In the resulting expression, we can then replace the partition function with its path-integral representation and apply the derivative to the action of the system. Before doing that, however, we need to remind ourselves that we rescaled the theory to use dimensionless fields and the dimensionless chemical potential $\tilde{\mu}_\sigma = a_\tau \mu_\sigma$. The derivatives with respect to the dimensionful and dimensionless chemical potentials are related via

$$\partial_{\mu_\sigma} = a_\tau \partial_{\tilde{\mu}_\sigma} = \frac{\beta}{N_\tau} \partial_{\tilde{\mu}_\sigma} \,. \tag{4.29}$$

Using this relation and omitting tildes again, we find:

$$\begin{aligned} &\text{[Continuation of Eq. (4.28)]} \\ &= \frac{1}{\mathcal{Z}} \frac{1}{N_\tau} \partial_{\mu_\sigma} \int \mathcal{D}(\phi^*, \phi) \, e^{-\mathcal{S}_{\mathrm{B}}} \\ &= \frac{1}{\mathcal{Z}} \int \mathcal{D}(\phi^*, \phi) \, \frac{1}{N_\tau} \partial_{\mu_\sigma} \, e^{-\mathcal{S}_{\mathrm{B}}} \\ &= \frac{1}{\mathcal{Z}} \int \mathcal{D}(\phi^*, \phi) \, \left(-\frac{1}{N_\tau} \partial_{\mu_\sigma} \mathcal{S}_{\mathrm{B}} \right) \, e^{-\mathcal{S}_{\mathrm{B}}} \,. \end{aligned} \tag{4.30}$$

Thus, we have

$$\begin{aligned}\mathcal{O}_{N_\sigma}(\phi^*,\phi) &= -\frac{1}{N_\tau}\partial_{\mu_\sigma}\mathcal{S}_{\mathrm{B}} \\ &= \frac{1}{N_\tau}\partial_{\mu_\sigma}\log\det\mathcal{M} \\ &= \frac{1}{N_\tau}\partial_{\mu_\sigma}\operatorname{tr}\log\mathcal{M} \\ &= \frac{1}{N_\tau}\operatorname{tr}\left(\partial_{\mu_\sigma}\log\mathcal{M}\right) \\ &= \frac{1}{N_\tau}\operatorname{tr}\left(\left(\partial_{\mu_\sigma}\mathcal{M}\right)\mathcal{M}^{-1}\right)\,.\end{aligned} \tag{4.31}$$

For evaluating this expression, we need to determine the μ_σ derivatives of the fermion matrix. Since the chemical potentials only appear in the temporal derivatives of the upper left-hand and lower right-hand blocks of the fermion matrix, we only need to compute the derivatives of the respective temporal derivative matrices. For the up-species in the upper left-hand block, we find

$$\begin{aligned}\partial_{\mu_\uparrow}D_\tau^{(\mathrm{bw})}(\mu_\uparrow) &= \partial_{\mu_\uparrow}\left(\mathbb{1}-e^{\mu_\uparrow}R\right) \\ &= -\partial_{\mu_\uparrow}e^{\mu_\uparrow}R \\ &= -e^{\mu_\uparrow}R\,.\end{aligned} \tag{4.32}$$

For the down-species in the lower right-hand block, we find

$$\begin{aligned}\partial_{\mu_\downarrow}D_\tau^{(\mathrm{fw})}(\mu_\downarrow) &= \partial_{\mu_\downarrow}\left(e^{\mu_\downarrow}A-\mathbb{1}\right) \\ &= \partial_{\mu_\downarrow}e^{\mu_\downarrow}A \\ &= e^{\mu_\downarrow}A\,.\end{aligned} \tag{4.33}$$

This yields the overall μ_σ derivatives of the fermion matrix

$$\partial_{\mu_\uparrow}\mathcal{M} = \begin{pmatrix} -e^{\mu_\uparrow}R & \mathbb{0} \\ \mathbb{0} & \mathbb{0}\end{pmatrix} \tag{4.34}$$

and

$$\partial_{\mu_\downarrow}\mathcal{M} = \begin{pmatrix} \mathbb{0} & \mathbb{0} \\ \mathbb{0} & e^{\mu_\downarrow}A\end{pmatrix}\,, \tag{4.35}$$

with quadratic zero blocks $\mathbb{0}$ of dimension $N_\tau N_x^d$. For the final CL observable expression, we need to use the fermion matrix with the pairing field at the current Langevin time step represented by the CL fields:

$$\mathcal{O}_{N_\sigma}^{(i)} = \frac{1}{N_\tau}\operatorname{tr}\left(\left(\partial_{\mu_\sigma}\mathcal{M}\right)\left(\mathcal{M}^{(i)}\right)^{-1}\right)\,. \tag{4.36}$$

This allows us to calculate the particle number of species σ for a Langevin process:

$$N_\sigma = \left\langle\mathcal{O}_{N_\sigma}\right\rangle_{\mathrm{CL}}\,. \tag{4.37}$$

To turn this particle number into a density, we need to divide it by the volume of the system. However, since we rescaled all parameters of our theory to be dimensionless, we cannot compute dimensionful observables, such as the density, directly. That also makes intuitive sense, as we

never specify the length of our box L. Therefore, we define a dimensionless density by multiplying the density with the thermal wavelength volume λ_{th}^d using the thermal wavelength

$$\lambda_{\text{th}} = (2\pi\beta)^{1/2} \,, \tag{4.38}$$

that is made species-independent by using a mass of one. Using this rescaling, the dimensionless density reads

$$n_\sigma \lambda_{\text{th}}^d = N_\sigma \frac{\lambda_{\text{th}}^d}{V} \,, \tag{4.39}$$

which is the particle number rescaled by a dimensionless ratio $\lambda_{\text{th}}^d / V$. We can express this ratio in terms of the lattice size and spacing parameters:

$$\frac{\lambda_{\text{th}}^d}{V} = \frac{(2\pi N_\tau a_\tau)^{d/2}}{(N_x a_x)^d} = \left(\frac{2\pi N_\tau a_\tau}{N_x^2 a_x^2}\right)^{d/2} = \left(2\pi r \frac{N_\tau}{N_x^2}\right)^{d/2} \,, \tag{4.40}$$

resulting in the dimensionless density

$$n_\sigma \lambda_{\text{th}}^d = N_\sigma \left(2\pi r \frac{N_\tau}{N_x^2}\right)^{d/2} = \left\langle \left(2\pi r \frac{N_\tau}{N_x^2}\right)^{d/2} \mathcal{O}_{N_\sigma} \right\rangle_{\text{CL}} \,. \tag{4.41}$$

Of course, bosonizing the theory by introducing an auxiliary field that represents the pairing of the fermions rather than the particle density is not the best approach for calculating the latter. This is why the calculation of this observable is rather tedious, both in its derivation and in actual numerical studies.

4.2.2 Pair-Correlation Functions

Calculating pair-correlation functions in the pairing field formalism is a lot more straightforward than calculating densities; it is what the pairing field formalism was designed for. In Sec. 3.1.2, we have already seen that we can easily express pair-correlation functions in terms of correlation functions of the pairing field. For example, we found

$$\begin{aligned} G(\tau_1, \mathbf{r}_1, \tau_2, \mathbf{r}_2) &= \left\langle \psi_\downarrow^*(\tau_1, \mathbf{r}_1) \psi_\uparrow^*(\tau_1, \mathbf{r}_1) \psi_\uparrow(\tau_2, \mathbf{r}_2) \psi_\downarrow(\tau_2, \mathbf{r}_2) \right\rangle \\ &= \left\langle \phi^*(\tau_1, \mathbf{r}_1) \phi(\tau_2, \mathbf{r}_2) \right\rangle \,. \end{aligned} \tag{4.42}$$

The associated operator expression $\mathcal{O}_G(\phi^*, \phi)$ used in the CL study can be directly read off from the definition of the observable itself. The continuous pairing fields we used for this derivation were still dimensionful. Therefore, to relate this expression to the discrete fields we employ in the simulation, we need to include the rescaling factors defined in Sec. 3.3.6. We find

$$\mathcal{O}_G(\phi^*, \phi) = a_x^{-d} \phi^*(\tau_1, \mathbf{r}_1) \phi(\tau_2, \mathbf{r}_2) \,. \tag{4.43}$$

4.3 Choice of Tools

In a project concerned with the development of a novel lattice formulation, whether is it calculating observables or processing and visualizing results, we work with code just as much as we work with the mathematical expressions of the theory, if not more. Therefore, we would like to make sure that we select our tools and workflows with care and in a way that is best suited for the particular problem at hand. Beyond that, we would also like to make use of the achievements in the field of modern *Software Development* to achieve the best results we can. In recent

decades, the field of software development has emerged from the world of programming, and while programming itself is part of software development, it also encompasses design, structuring, documenting, testing, and maintaining code. Software development as a field has reached a level of maturity and establishment that allowed standard texts like, e.g. Ref. [64], on the subject to be written. We aim to use the techniques of software development to make our simulation more stable, more verifiable, easier to adapt, and easier to maintain than software that is focused solely on a source code that produces the desired results. However, while we can largely use the techniques of modern software development in an unmodified form, we need to keep in mind that this field usually deals with a scenario that is different from ours; in the world of software development, developers of a piece of software are generally a separate group from its users, whereas, in the case of scientific simulations of active research topics, the people running the simulation and interpreting the results are usually the same people implementing changes based on their observations. In our case, developers and users are the same people. That is something we need to keep in mind when adapting the techniques of software development to our efforts.

For our simulation, we divided the "code" [1] into two major parts. One part consists of auxiliary scripts that, for example, generate sets of input parameters over a given parameter span, as well as the processing and visualization of simulation results. The other part is the simulation core that does the numerical "heavy lifting". The scripts of the first part are loosely connected and usually only take seconds to run while the simulation core is a fully realized software library that performs simulations that can easily take days or weeks.

Because of the short execution time of the pieces of software in the auxiliary part of the code, it does not really matter how performant the chosen libraries and programming languages are. It is far more important to choose fully-fledged solutions that allow the developer to do as much standard work as possible in predefined functions and quickly achieve desired results with compact and easy-to-read code. For this part, the time of the developer is a far greater consideration than the time of the computer. For this particular project, all scripts are implemented in the *Python* programming language [65] using the *matpotlib* [66] for visualization and *NumPy* [67], *pandas* [68] and *SciPy* [69] for data management and manipulation. Most of this work is done interactively using the *Jupyter Lab* [70] environment based on notebooks with discrete code execution units called cells. This cell-based approach is particularly useful when using the library *SymPy* [71], which turns our notebook environment into a powerful computer algebra system for symbolic calculations and derivations.

To choose a programming language for the simulation core, we evaluate candidates under consideration utilizing a range of aspects and criteria. The programming languages under consideration for the simulation code are C/C++, Fortran, Julia [72], and Python. In order to determine the language we will use to implement the simulation core, we compare them regarding their performance, the availability of software development tooling, the availability of mathematical and scientific functions within the core languages and their library ecosystems, their interoperability with the auxiliary part of the simulation, the suitability of their implemented paradigms with regard to the nature of our problem and finally their syntax and ease of use for the developer/user. Before we begin this analysis, however, we shall briefly review keynotes and the history of each of the candidates.

[1] In academic physics, it is common to refer to a collection of software that achieves a given goal as a "code". The author would like to advise against this practice since it perpetuates the notion that merely the code of a software package is enough. This is not the case, because without proper documentation and an extensive set of automated tests, "codes" are tedious to understand, adapt and maintain and only reach a fraction of their potential.

4.3.1 Candidate Languages for the Simulation Core

The *C* programming language was created in the 1970s and is a compiled general-purpose high-level language that was meant to provide human-readable access to the capabilities of the hardware. In 1985, C++[2] was released as an extension of C that added features like support for object orientation and an extended standard library. C is (almost) a subset of C++, which means that in general C code can be compiled by a C++ compiler but not the other way around. Often that leads to libraries not utilizing the language features that are exclusive to C++, in order to be usable by both C and C++ code. Due to this intimate relationship between C and C++, they are often listed together under the name C/C++. The standard libraries of C do not contain any functions for scientific computing. Such capabilities are provided by third-party libraries and for the present discussion we shall focus on the popular *GNU Scientific Library (GSL)* [73].

Fortran is another popular choice among scientists. It was created in the 1950s and is a compiled high-level language for scientific applications. While it is not recorded what the name Fortran exactly means, it is likely a contraction of some variations of the words *for*mula and *trans*lating. Due to its expressed purpose of scientific programming, it has some inbuilt capabilities for the work with tensors up to the multiplication of matrices. Additional capabilities are provided through libraries that implement standards such as *Basic Linear Algebra Subprograms (BLAS)* [74] and the less rudimentary *Linear Algebra Package (LAPACK)* [75]. For LAPACK, we use the *netlib* implementation [75], and for BLAS the *openBLAS* implementation [76], both of which are, themselves, written in Fortran.

The *Julia* language is the youngest entry on this list, being first released in 2012. It is a just-in-time compiled high-level language that is described as a general-purpose language by the authors but has a strong focus on numerical computation and scientific applications. Most essential tools for scientific programming are available in the standard library and its flexible type system and clean purpose-built syntax allow the developer/user to quickly implement calculations.

The *Python* language was first released in 1991 and is a high-level general-purpose scripting language that is usually interpreted rather than compiled. It aims to have a clean and readable syntax which, among other things, is enshrined in the PEP 20 document [77], containing the so-called *Zen of Python*. Python's design along these guiding principles allows developers to solve complex problems in little time with little code. Python has an extensive standard library but numerical and scientific functions are provided through third-party libraries, such as the above-mentioned *NumPy* and *SciPy*.

4.3.2 Performance

The simulation core needs to solve the discretized Langevin equation for potentially very large lattices and a large number of timesteps. This is a process that can easily take weeks on a computer. For that reason, we aim to implement the simulation core in a language that has high performance. To compare the performance of our candidates, we use the micro benchmarks [78] from the Julia website. These micro benchmarks determine the execution times (how long it took the machine to execute the code) of small, specific tasks for multiple languages to compare the performance for this specific workload. These micro benchmarks are relatively small and only feature a single type of workload each. Nevertheless, they can give us a general feeling for the performance of a language. The results for relevant micro benchmarks for our candidate languages are given in Fig. 4.1 and Mathematica is included as a reference, because it is a common tool for physicists, even though it is not being considered for the implementation of

[2] In C syntax, "++" denotes the increment operator. Therefore, the name C++ is to be understood as an "increment" of C: C++ = C + 1. This eludes to C++'s original nature as an extension of C and occasionally, in spoken language, some people pronounce C++ as "C increment".

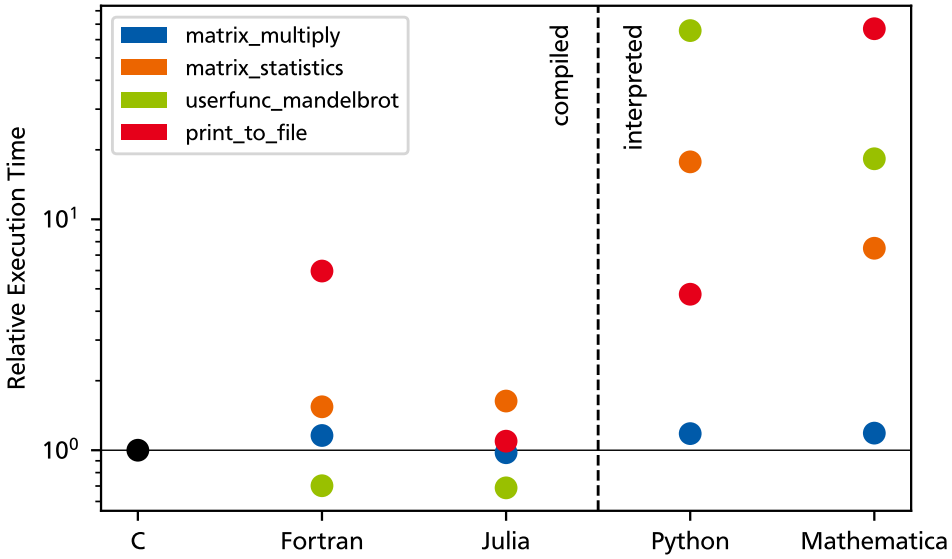


Figure 4.1: Comparison of the execution times of micro benchmarks implemented in different programming languages from the collection [78]. The execution times are normalized to the execution times of the C implementation and depicted on a logarithmic scale. The colors represent the specific micro benchmark that was timed.

the simulation core. Depicted are the execution times normalized to the execution times of the C implementation on a logarithmic scale.

Arguably, the most relevant micro benchmark for our project is "matrix_multiply", since a lot of the numerical work in our simulation is multiplying matrices. In "matrix_multiply" two random $\mathbb{R}^{1000\times1000}$ matrices are generated and their product is calculated. That being said, when we look at the execution times of "matrix_multiply" in Fig. 4.1, we notice something peculiar: all languages, compiled or interpreted, have roughly the same performance. Why is that the case? Above we noted that, of all the candidate languages, Fortran is the oldest and that the rudimentary BLAS and LAPACK standards of the era are usually implemented in Fortran. Because of this, when the other languages and their mathematical libraries were created, efficient implementations of basic operations in linear algebra already existed in the form of Fortran BLAS and LAPACK libraries. [3]. Therefore, most of today's non-Fortran libraries still call Fortran-implemented libraries in the background to perform their basic linear algebra tasks. In fact, all of the examples in Fig. 4.1 do just that and call an implementation of the BLAS function `dgemm` to multiply matrices. Under this aspect, it is not surprising that the execution times of "matrix_multiply" are almost identical for all given languages.

Another important micro benchmark for our simulation is "matrix_statistics". In this micro benchmark random matrices are generated, raised to the fourth power and the mean and standard deviation of sets of traces of the matrix powers are calculated. This micro benchmark describes a more general handling of matrices and is a suitable benchmark for the workload that our simulation generates. Looking at the execution time results in Fig. 4.1, we notice that all compiled languages still perform similarly but interpreted languages are notably slower by around

[3] Among scientists, there sometimes exists the notion that the use of Fortran is necessary to create the most efficient numerical libraries. This is not the case, as any compiled language can in principle generate the same efficient machine code as compiled Fortran. It just so happens that efficient Fortran implementations for many problems already exist, making it unviable for developers to implement their functions in another language rather than just using the existing Fortran implementations.

one order of magnitude. Even though they use efficient libraries to perform most operations on the matrices, their handling in between these basic operations costs interpreted languages a lot of time.

The "userfunc_mandelbrot" micro benchmark takes a section of the complex plane and, for a grid of points within the section, determines whether the points lie within or outside of the Mandelbrot set and it also estimates how quickly the Mandelbrot sequence diverges if it does not remain bounded for a given point. It does so by calculating the Mandelbrot sequence in the body of a loop which also contains a possibly-branching conditional (if) statement to determine if the sequence is unbounded. This is an interesting example that simulates workloads that implement custom algorithms rather than relying on built-in functions of library calls. For compiled languages, this is not really a problem, since, for them, there is no difference whether an algorithm within a library calculates a loop or if the program does it itself. For interpreted languages, on the other hand, manually programming loops rather than using calls to compiled libraries is usually associated with a heavy performance penalty. Indeed, the results in Fig. 4.1 show that the compiled languages have roughly the same performance while the interpreted languages Python and Mathematica are significantly slower. In the case of Python, the execution time is nearly two orders of magnitude larger than that of an equivalent C program. This is a general phenomenon in interpreted languages; seemingly equivalent blocks of code can have starkly different execution times. To achieve the best possible performance, it is, therefore, necessary to use the highest-performing idioms when writing code. This is a bit of a caveat with interpreted languages in general and Python in particular. When starting to use the language, the learning curve is very "shallow" and one can quickly write code that produces the desired results. Later on, however, when one aims to write efficient code, a good amount of knowledge of the inner workings of the language is required to select efficient idioms.

The last micro benchmark we consider for our simulation is "print_to_file". As the name suggests, this micro benchmark prints some data into a file. Specifically, it chooses the virtual `/dev/null` file of a Linux-based system. Therefore, no actual storage media are involved in the process and the execution time is largely determined by the program-under-test itself. Printing results to file is not the most performance-critical part of our simulation, but it does happen a fair bit, so it is worth considering it in the comparison. Again, looking at the results of the benchmarks, the interpreted languages are generally slower than the compiled ones, although their results are still acceptable for our purposes. All things considered, Python's performance is actually really good for an interpreted language; a testament to the decades of optimization by the Python developers. The notable exception in this benchmark is Fortran, which actually performs worse than Python. Since Fortran's performance in this test is still absolutely acceptable, we do not want to spend much time pondering possible causes of this unexpected result, but, as an interesting side note, at the time of Fortran's creation, the modern notion of a computer file did not necessarily exist.

From the point of view of performance, C, Fortran, and Julia seem to be near-equally good choices and superior to Python.

4.3.3 Availability of Software Development Tooling

Since we not only want to create a code that efficiently serves its purpose but want to create a piece of software that is stable, verifiable, maintainable, and easy to understand and adapt, there are certain tools we require to work with a given language. Those tools are a *debugger*, some system of *code documentation*, a framework for *unit testing*, and a sensible system for *package management*.

A debugger is a tool that allows the developer to automatically halt the program at a certain

point or under certain conditions and monitor the variables at that state or even execute lines of code to probe the source of a bug. Debuggers are a staple of software development and debugging interfaces exist in nearly every integrated development environment (IDE).

Code documentation systems are tools that generate a human-readable, searchable, interactive form of documentation of the software. They do this by analyzing the source code and parsing special documentation comments placed in the code by the developer. In a research environment where projects are handed off to other people or shared for collaboration, proper documentation is essential to quickly get new researchers started with the project.

An absolutely mission-critical piece of tooling is the unit testing framework. In unit testing, we create automated tests that check if a given small section of the code, the unit, behaves as intended. To this end the developer writes a unit test that is, itself, a small program, utilizing the unit testing framework. This unit test then calls the unit-under-test and compares its output and behavior to a recorded set of expected outputs and behaviors. These predefined *test cases* test common scenarios of the unit but also edge cases that are usually more likely to result in errors. It is hard to overstate how useful unit testing is for the development of scientific simulations and we would like to demonstrate that fact using an example. Suppose we have implemented the simulation outlined in this chapter and can calculate observables. Now suppose we want to implement an optimization, e.g. in the generation of lattice operators, to produce larger simulation runs for a publication. What if that optimization effort leads to a bug in the code? The best-case scenario is that this bug immediately results in an error that prevents the simulation from running. This way, we know that the bug exists and we can find and fix it. It is also possible, however, that the bug manifests itself without causing a runtime error. For example, it is conceivable that a bug causes deviations of calculated observable values in certain regimes. How would we go about finding the cause of this deviation, if we even notice it at all? This scenario can be prevented by unit testing. When we properly apply the practice, we have a set of unit tests that verify that all lattice-operator-generating functions produce correct results. If we then implement an optimization that introduces a bug, we will immediately see that the unit tests fail and we can resolve the bug. Again, it is hard to overstate how useful unit testing is for scientific simulations.

The last piece of tooling we put special focus on is package management. A package manager is a tool that can download external libraries at a specified version and automatically also download their dependencies and make them available within the project without the need for the developer to perform manual configurations. We use these tools to avoid spending time on something that is not part of the actual simulation.

There are also tools for more advanced tasks we did not list, like, e.g., a *profiler*. We omit this particular tool because fine-tuning the optimization of the code is a task best left to a time when the code is more mature and does not constantly change. Indeed, for simulations of systems under active research, such a time may never come. Should one encounter reusable pieces of code within the simulation that do, in fact, converge to a constant state, it might be appropriate to extract them into a library. One can then perform optimization in the library and even reuse it in different projects.

For our candidate languages, we generally find that modern tooling is available from third parties for the older languages and built-in for the more modern languages. On the debugging side, C and Fortran can be debugged with, e.g., the *GNU Debugger*, Python has a built-in debugger called *pdb* and the Julia community is currently working on a debugger called *Debugger.jl*. For documentation, in C and Fortran, we can use *Doxygen*. For documenting Python, the de-facto standard is *Sphinx*, developed by the Python community, and Julia offers the *Documenter.jl* tool. So far, all tools for all candidate languages operate very similarly, however, when it comes to unit testing we start to see differences. Python and Julia both offer built-in unit testing capabilities,

whereas C and Fortran rely on third-party tools. There is a variety of tools available for each of both languages, which makes it difficult to select one and may require new developers to switch to another framework if they are not used to the one that is used in the project. When it comes to package management, we see the most drastic differences. Python offers the *pip* tool for installing packages and the same can be accomplished in Julia using *Pkg.jl*. Both of these tools can install packages for the system-wide environment of the language or within project-specific virtual environments that allow for an isolated and reproducible state of library installations. C and Fortran have no official package managers and none of the community efforts have managed to establish themselves as de-facto standards. The most common way to install libraries for C and Fortran (under UNIX-like operating systems) is to install system packages that provide library headers and shared runtime libraries to the system. One then needs to provide paths to the libraries to the compiler, as well as configure certain environment variables. This process can lead to conflicts, because other applications may require the system package manager to install the same dependencies at different versions or make modifications to the environment variables. In fact, this package management scheme is so tedious and fragile that it is not uncommon to create a virtual machine, usually some sort of container using Podman [79] or Docker [80], just for the purpose of having an isolated, reproducible environment to compile the code in.

Regarding tooling, the modern choices, Python and Julia, are certainly superior to C and Fortran.

4.3.4 Interoperability with Auxiliary Scripts

Another relevant aspect for the choice of a programming language is its ability to operate with and be operated by external scripts. Our auxiliary scripts for starting the simulation and evaluating results are written in Python, so we would like the simulation core to have a sensible interface to that. For C, the Python community has created libraries such as the *C Foreign Function Interface (CFFI)* [81] that allow us to call functions of C libraries from within Python and perform the necessary conversions of data flowing from Python to C and back. For Fortran, this process is not only possible as well but also easier, since Python's NumPy library contains a utility called *F2PY* that can build Fortran modules and expose them to the Python code in the form of a virtual module. We can then call the functions of this module in the same way we would call Python functions, with the caveat that we need to define our arrays with a memory ordering that is compatible with that of Fortran. If we write the simulation core in Python, it is trivially given that we can call it from the auxiliary Python scripts. Finally, when implementing the simulation core in Julia, we can use the *PyJulia* library for Python to use the Julia module in pretty much the same way we would use ordinary Python modules.

In this comparison, while interoperability between Python and all of the candidate languages is possible, C and Fortran fall short of Python and Julia because they require us to manually define or adjust the way our variables are stored in memory. These additional steps are not difficult to implement but lead to more "cluttered" code that is more difficult to read and understand and contains more potential points of failure.

4.3.5 Suitability of Paradigms and Language Design

When it comes to the design of a language and the paradigms it employs, there are some choices we deem more suitable for writing the simulation core. Regarding *memory management*, we generally would like to avoid having to do it manually. This is the case in C/C++and to a lesser extent also in Fortran. There are multiple problems with the need for manual memory management, first and foremost: it is simply not necessary. We do not care how and where the computer stores our variables in memory, we just want the result to be efficient and modern

languages can do that for us. Moreover, being forced to perform manual memory management can lead to an array of memory-related errors. One of them is the so-called *memory leak*. It occurs when the developer allocates some memory for a variable but never frees or deallocates that memory again. If this happens in a section of the code that is executed over and over again, the memory leak will slowly fill the available memory. This causes the application's performance to drop over time. The memory footprint of the application first becomes too large for the cache storage of the processor, forcing it to be moved in and out of RAM, next it fills the RAM until it becomes so big that it has to be swapped into and out of the mass storage system. Eventually, if the application does not terminate before this point is reached, the memory leak causes either the application or even the entire system to halt. For simulations that can run for weeks, this is not an unrealistic scenario. Another kind of error related to manual memory management is the access of memory that does not belong to a variable. Suppose the developer makes a mistake in the handling of a matrix pointer that causes the program to attempt to read the matrix from a section of memory that does not belong to the variable. In this situation, one of the two following things can happen. The first scenario is that the memory which the program tries to access is outside of the memory space that the operating system has allocated to the program. This will result in a so-called *segmentation fault* and cause the program to halt. The second, and arguably more dangerous, scenario is that the requested section of memory is, in fact, allocated to the program but contains other variables, program code, uninitialized quasi-random values, or a mixture thereof. In this case, rather than reading the values of the matrix, the program would read completely unpredictable and seemingly random values. The dangerous trait of such errors is that they can go unnoticed and cause nonsensical results of the program. Additionally, beyond being a possible source of errors, manual memory management requires us to dedicate code to it, which "clutters" and "dilutes" the calculations we are implementing. This makes the code harder to read and understand. With all that being said, unless there is a specific need for it, it is generally preferable to use a language that does not require the developer to perform manual memory management.

Regarding the *type system* of the candidates, since we are writing the simulation core as a library, we would like to be able to define the types of function arguments and return values. This causes improper use of the library functions to result in type errors. Beyond that, it represents a part of the documentation of the functions that is contained in the code itself and, therefore, does not have to be actively maintained by the developer. Defining these types is possible in all of the candidate languages except for Python, since Python employs the concept of *duck typing*. In duck typing, arguments are of the correct type if the function can perform all operations on them that it wants to perform on them; if it walks like a duck and it quacks like a duck, then it must be a duck. This, however, means that errors resulting from an argument of a wrong type being passed to a function always occur at runtime and at some point within the body of the function. Consequently, it is more difficult to find and remove type errors at the time of programming in languages that practice duck typing. Newer versions of Python offer a feature called *type hinting* that allows the developer to specify types of arguments and return values for the use with static code analysis tools. However, these type hints are not binding and, therefore, require discipline on the part of the developer to be effective. When it comes to the choice between *static typing* and *dynamic typing*, since we are writing a library, we would prefer static type checking, i.e. a system that checks for type errors at compile time rather than runtime. This would increase the reliability of the program because then there could be no library-internal type errors at runtime. However, a dynamic type-checking system, when used in combination with proper unit testing (again, which it absolutely should be), is essentially just as good, because all library-internal type errors that can occur, will occur during the tests. This way we can still discover and fix them before we use the simulation for real calculations. C and Fortran use static type checking while

Julia does not and, due to its just-in-time compiled nature, usually can not check for type errors before runtime.

Another design choice that programming languages make that may seem trivial and inconsequential is whether the language starts array indices at zero or at one. We would like to argue, however, that that choice is not inconsequential at all in the context of scientific programming. This is due to the convention for the notation of tensors in mathematics. Suppose we have a matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \tag{4.44}$$

and a vector

$$\mathbf{v} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} . \tag{4.45}$$

When we reference the upper left-hand entry of the matrix or the uppermost entry of the vector we write A_{11} and $\mathbf{v}_1$ in mathematical notation. In a programming language that starts array indices at zero, however, we would have to refer to these entries with statements like `A[0][0]` and `v[0]`. The solution to this problem seems simple; we just subtract one whenever we want to index an array that represents a tensor. While this solution is certainly correct, in practice indexing arrays that represent tensors is something that we do so often, that errors created from this indexing mismatch are actually very common. Errors that are caused by a discrepancy of one are in fact so common that they have their own name in the jargon of programmers [82]: *off-by-one errors*. Off-by-one errors are usually not hard to detect and usually not hard to fix, however, it is still preferable to use a language that minimizes their occurrence; a language that is specifically built for the purpose of scientific calculations. That is true for both Fortran and Julia and, indeed, they start indexing arrays at one. C and Python are general-purpose languages with no special focus on scientific applications and both of them start indexing arrays at zero. In addition to minimizing off-by-one errors, when using a language that starts indexing arrays at one, the code more closely matches the mathematical expressions we derive on paper. That makes the code easier to read and understand.

Lastly, we briefly touch upon the candidate languages' support of *object-oriented programming*. C, Fortran, and Julia do not include explicit support of classes, although C++does. This is not really a problem since abstracting mathematical calculations into code usually does not involve the notion of one of the operands *owning* a function and the attributes of objects can still be grouped together in custom data structures in all of the candidate languages that do not support classes. Therefore, it does not really matter to us in the scientific context, whether a language supports classes or not.

From the standpoint of language design, Fortran and Julia seem good choices for our simulation, because they allow us to define the types of function arguments and return values and are built for scientific applications. Of these two, Julia does not require the developer to perform manual memory management and, therefore, seems to be the best choice out of our four candidates with regard to language design.

4.3.6 Syntax and Ease of Use

The final aspect under which we compare our four candidate languages is their syntax and ease of use. To this end, we implement a little example program in all four languages and compare their source codes. As an example, we compute the expression

$$D - CA^{-1}B \tag{4.46}$$

using the four matrices

$$A = \begin{pmatrix} 1 & 0 & 1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$
$$C = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ -1 & 0 & -1 \end{pmatrix}. \tag{4.47}$$

This is reminiscent of the inverse of the matrix $F^{(i)}$ that we use above in the block-inversion of the fermion matrix in Eq. (4.18). This example requires us to perform matrix subtraction, multiplication, and inversion and should give us a good idea of how it would feel to implement the simulation core in each of the languages. In the following, we will go through the implementation for each of the four languages. Below that we will then compare them to one another.

We begin by implementing the example calculation in C using the GNU Scientific Library (GSL) for matrix handling and linear algebra operations. The resulting source code is shown in Listing 4.1.

Listing 4.1: Example calculation implemented in C using the GNU Scientific Library (GSL) for matrix operations and linear algebra.

```c
#include <stdio.h>
#include <gsl/gsl_blas.h>
#include <gsl/gsl_matrix.h>
#include <gsl/gsl_linalg.h>

void print_matrix(gsl_matrix *m) {
  for (int row = 0; row < m->size1; row++) {
    for (int col = 0; col < m->size2; col++) {
      printf("%g\t", gsl_matrix_get(m, row, col));
    }
    printf("\n");
  }
}

gsl_matrix *inv(gsl_matrix *matrix) {
  // allocate memory for result matrix and auxiliary variables
  int sign = 0;
  gsl_matrix *matrix_inv = gsl_matrix_alloc(matrix->size1, matrix->size2);
  gsl_permutation *P = gsl_permutation_alloc(matrix->size1);
  gsl_matrix_memcpy(matrix_inv, matrix);

  // perform LU decomposition of copied input data
  gsl_linalg_LU_decomp(matrix_inv, P, &sign);

  // calculate inverse matrix based on LU decomposition
  gsl_linalg_LU_invx(matrix_inv, P);

  // deallocate heap memory of auxiliary variable
  gsl_permutation_free(P);

  return matrix_inv;
}

int main(void)
{
  // define matrices A, B, C and D

```

```
38   int size = 3;
39
40   double A_data[] = {
41     1, 0, 1,
42     -1, 1, 0,
43     0, -1, 1
44   };
45   gsl_matrix_view A = gsl_matrix_view_array(A_data, size, size);
46
47   double B_data[] = {
48     3, 0, 0,
49     0, 2, 0,
50     0, 0, 1
51   };
52   gsl_matrix_view B = gsl_matrix_view_array(B_data, size, size);
53
54   double C_data[] = {
55     -1, 0, 0,
56     0, 3, 0,
57     0, 0, 1
58   };
59   gsl_matrix_view C = gsl_matrix_view_array(C_data, size, size);
60
61   double D_data[] = {
62     -1, 1, 0,
63     0, -1, 1,
64     -1, 0, -1
65   };
66   gsl_matrix_view D = gsl_matrix_view_array(D_data, size, size);
67
68   // allocate memory for intermediate and final result
69   gsl_matrix *intermediate_result = gsl_matrix_calloc(size, size);
70   gsl_matrix *result = gsl_matrix_calloc(size, size);
71
72   // invert A in function
73   gsl_matrix *A_inv = inv(&A.matrix);
74
75   // calculate C A^(-1) B and store it in result
76   gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, &C.matrix, A_inv, 0.0,
77                  intermediate_result);
78   gsl_blas_dgemm(CblasNoTrans, CblasNoTrans, 1.0, intermediate_result,
79                  &B.matrix, 0.0, result);
80
81   // calculate D - result and store it in result
82   gsl_matrix_sub(result, &D.matrix);
83   gsl_matrix_scale(result, -1);
84
85   print_matrix(result);
86
87   // deallocate heap memory of auxiliary variables
88   gsl_matrix_free(intermediate_result);
89   gsl_matrix_free(A_inv);
90   gsl_matrix_free(result);
91
92   return 0;
93 }
```

The program begins execution at the beginning of the `main` function in line 34 and we start by entering the example matrices A, B, C and D. The GSL's documentation suggests doing that by generating matrix views from raw arrays, so we enter the matrix entries in flat arrays and call the

appropriate GSL functions. We begin to prepare the actual calculation in line 69 by allocating memory for the and result, as well as an intermediate result.

In line 73, we make a function call to invert the matrix A. However, GSL does not offer us a function that directly inverts a matrix. Instead, we have to implement such a function ourselves in the form of the `inv` function, starting in line 15. The way we invert matrices using GSL is to perform an LU decomposition of the matrix and call an auxiliary function to compute the inverse matrix based on the decomposition. This is exactly what we do in the `inv` function. In the first few lines, we allocate memory for the resulting inverse matrix and auxiliary variables and then perform the LU decomposition in line 23. To this end, we copied the contents of the input matrix into the variable for the result and passed it as an input to `gsl_linalg_LU_decomp`. We do this because `gsl_linalg_LU_decomp` stores its results by overwriting the contents of its input parameter and we do not want it to overwrite the input parameter of the `inv` function. Beyond that, `gsl_linalg_LU_decomp` gives us a permutation we need to store in order to construct results from the decomposition. This permutation is one of the auxiliary variables we needed to allocate at the beginning of the `inv` function. With the decomposition performed, we then call `gsl_linalg_LU_invx` to create the desired inverse matrix in the return variable. Before we can return this result, however, we need to make sure that we free all memory that we allocated for auxiliary variables in the heap, since otherwise our `inv` function would cause a memory leak. We do this by freeing the memory we allocated for the permutation of the decomposition in line 26.

With matrix A inverted, we can continue the calculation by determining the product $CA^{-1}B$. For performing matrix multiplications, GSL offers us an interface to a BLAS implementation, which means, for calculating the product of two matrices we need the function `gsl_blas_dgemm` that performs the `dgemm` BLAS operation. The `d` in `dgemm` stands for "double", since we perform the operation on matrices that have floating-point numbers with double precision as entries, `ge` stands for "general", because we do not restrict ourselves to a specific kind of matrices, like, e.g., triangular matrices, for which the operation can be further optimized, and `mm` stands for "matrix-matrix", since we perform an operation that involves two matrices. The `dgemm` operation computes the expression

$$P_C = \alpha \operatorname{op}(P_A) \operatorname{op}(P_B) + \beta P_C \tag{4.48}$$

of its parameter matrices P_A, P_B and P_C and its scalar parameters α and β with $\operatorname{op}(P_A)$ being either P_A, $P_A^\intercal$ or $P_A^\dagger$. In line 76, we calculate the product CA^{-1} and store the result in the variable `intermediate_result`. We need neither of the matrices C and A^{-1} to be transposed, so we pass the option `CblasNoTrans` for both operand matrices. Since we want the result to contain only the matrix product and none of what currently is in the `intermediate_result`, we set $\alpha = 1$ and $\beta = 0$. If we did not set β to zero, we would have to take care, that the `intermediate_result` itself is initialized to contain only zeros. When passing the parameter matrices C and A^{-1}, we need to reference the two matrices differently, namely, we reference C through `&C.matrix` and A^{-1} simply through `A_inv`. We shall get back to that below. In line 78, we perform another matrix multiplication to store the result of $CA^{-1}B$ in `intermediate_result`.

After calculating the product $CA^{-1}B$, we need to subtract it from D. To achieve this, we employ GSL's `gsl_matrix_sub` function in line 82. The keen-eyed reader may have noticed, that it is also possible to include this subtraction in the second call of the `dgemm` BLAS routine since we can see in Eq. (4.48) that this routine includes a matrix subtraction if we set $\beta = -1$. However, this requires the use of a second intermediate matrix and makes the code even harder to read than it already is. Since the `gsl_matrix_sub` function stores its result in its first parameter, we need to calculate the difference $CA^{-1}B - D$ and scale it by a factor of -1 to obtain the desired result. This is done in the subsequent line.

With the desired expression calculated, we present the result by printing it to the standard output, i.e. in the console. To do this, in line 85, we call the function `print_matrix` that we

implement in lines 6ff. Unfortunately, GSL does not include a function that prints a matrix to the console in a sensible way. Therefore, we have to implement one ourselves.

Before the program ends by returning from the `main` function, in lines 88f we free the memory that we have allocated for auxiliary variables and the result variable in the `main` function. This is not necessary, since directly after that the program terminates anyway, however, C developers should have a habit of always freeing memory that has previously been allocated. In fact, before we move on to the next example, we shall discuss some more of the memory handling in the C implementation of our example calculation. Firstly, in line 20, we fill the variable holding the result with the entries of the input parameter matrix. We achieve this by invoking the `gsl_matrix_memcpy` function. This is the "most correct" way of doing it since it is the safest; the `gsl_matrix` struct contains information about the size of the matrix and the library function can use it to clone the matrix without causing memory errors. It is, however, also possible to invoke the built-in C function `memcpy` to copy the raw contents of the matrix directly by accessing the `data` field in the matrix variables. Such a practice is commonplace in C, but a bit problematic, since it requires the user to correctly specify the limits of the memory copy operation. This is done by taking the number of elements of the matrix and the size of the data type of the entries into account, in order to clone the matrix completely and without overwriting memory outside of the data of the target matrix. A developer that is not familiar with GSL might resort to this practice, rather than using `gsl_matrix_memcpy`, and introduce an error-prone piece of code to the project. Secondly, as discussed above, we need to manually free memory that we previously manually allocated for variables, in order to not cause memory leaks. In a world where automatic memory management techniques such as garbage collection and borrow checking exist, there is simply no point in taking the risk of manual memory management. Finally, we want to address the syntax that we used in line 76 to reference the parameter matrices to the call of `dgemm`. Because we enter the input matrices of our example in the form of matrix views, rather than referencing `C`, we need to reference the included `gsl_matrix` struct `C.matrix`. However, the `gsl_matrix` struct is usually used in the heap, and, thus, functions operating on matrices usually accept pointers to matrix structs as parameters. Our matrix view is stored in the stack rather than the heap and the `C` variable holds the entire struct instead of a pointer to the struct and, therefore, we need to extract the memory address of `C.matrix` by prefixing it with an ampersand. The same does not hold for `A_inv`, since this matrix is allocated in the heap and the variable already holds a pointer. This results in an awkward mixture of syntax that does not make the already cryptic call of `dgemm` any easier to understand.

Moving on to the second of our candidate languages, we implement our example calculation in Fortran using the a LAPACK library for linear algebra operations. The resulting source code is shown in Listing 4.2.

Listing 4.2: Example calculation implemented in Fortran using LAPACK for linear algebra operations.

```
1  module subroutines
2  use, intrinsic :: iso_fortran_env
3
4  contains
5
6  subroutine print_matrix(n, matrix)
7      implicit none
8      integer, intent(in) :: n
9      real(real64), dimension(n, n), intent(in) :: matrix
10     integer :: row, col
11
12     do row = 1,size(matrix, 1)
13         do col = 1,size(matrix, 2)
```

```
            write ( *, fmt="(f7.2)", advance="no") matrix(row, col)
        end do
        write ( *, *)
    end do
end subroutine

subroutine inv(n, input_matrix, output_matrix)
    implicit none
    integer, intent(in):: n
    real(real64), dimension(n, n), intent(in) :: input_matrix
    real(real64), dimension(n, n), intent(out) :: output_matrix

    ! define auxiliary variables for inversion
    integer, dimension(n) :: ipiv
    real(real64), dimension(n) :: work
    integer info

    ! store input matrix in output variable before beginning to alter it
    output_matrix = input_matrix

    ! perform LU decomposition of copied input data
    call dgetrf(n, n, output_matrix, n, ipiv, info)

    ! use LU decomposition to construct inverse of input matrix in output
    ! variable
    call dgetri(n, output_matrix, n, ipiv, work, n, info)
end subroutine

end module subroutines

program example
    use subroutines
    implicit none
    real(real64), dimension(3, 3) :: A, B, C, D, A_inv, intermediate_result
    integer n
    n = size(A, 1)

    ! define matrices A, B, C and D
    A = reshape((/ &
        1, -1, 0, &
        0, 1, -1, &
        1, 0, 1 &
    /), shape(A))
    B = reshape((/ &
        3, 0, 0, &
        0, 2, 0, &
        0, 0, 1 &
    /), shape(B))
    C = reshape((/ &
        -1, 0, 0, &
        0, 3, 0, &
        0, 0, 1 &
    /), shape(C))
    D = reshape((/ &
        -1, 0, -1, &
        1, -1, 0, &
        0, 1, -1 &
    /), shape(D))

    ! call subroutine to invert matrix A
```

```
74      call inv(n, A, A_inv)
75
76      ! calculate C A^(-1) B
77      intermediate_result = matmul(C, A_inv)
78      intermediate_result = matmul(intermediate_result, B)
79
80      intermediate_result = D - intermediate_result
81
82      call print_matrix(n, intermediate_result)
83  end program example
```

Our program begins execution in line 44 and we begin by declaring the variables of our program in line 47. In lines 52-71 we enter the matrices A, B, C and D of our example calculation. We do this in a similar manner to our C implementation, by entering the matrix entries in flat arrays and reshaping them to fit the desired 3×3 shape. There is, however, a difference in how C and Fortran store matrices: matrices in Fortran are *column-ordered*. That means that, in the flat data array, we need to write all entries of a column before moving on to the next column rather than writing all entries of a row before moving on to the next row. Because of this memory order, the matrices we enter in lines 52-71 in the code all appear to be transposed, but the matrices we store and that we operate on are not. From a technical standpoint, nothing speaks against column-ordering matrices, but from a readability standpoint, it creates a discrepancy between the code we write and the calculation we aim to implement.

In line 74 we invert the matrix A. As is the case in the C implementation of our example calculation, there is no function or subroutine at our disposal that does that directly, therefore, as we do in the C implementation, we implement our own subroutine to invert a matrix. This implementation begins in line 20. As we do in C, we perform an LU decomposition of the input matrix and construct the inverse from the decomposition. We begin by declaring the auxiliary variables for this process in lines 27ff. Unlike in C, calls to Fortran library subroutines usually do not require us to pass pointers as arguments which allows us to pass the variables we declared directly and leads to more consistent code within the subroutine call. Just like we did in C, we write the contents of the input matrix into the output matrix, because the LU decomposition overwrites its input with its results and we cannot (and do not want to) overwrite an input in a subroutine. We perform this copy of data in line 32. Because Fortran is purpose-built for the manipulation of multi-dimensional arrays, this is actually more straightforward than it is in C. To actually perform the LU decomposition, we call the LAPACK subroutine `dgetrf` that stores the result in the `output_matrix` variable and also stores a permutation in the `ipiv` variable, with the name `ipiv` being de-facto standardized by the documentation of the `dgetrf` subroutine [75] and possibly standing for "index pivot". The names of LAPACK subroutines can be decoded similarly to the way we can decode BLAS subroutine names; `d` in `dgetrf` stands for "double", because we work with matrices that have entries with double-precision floating-point numbers as entries, `ge` stands for "general", because we use the variant of the subroutine that can operate on all types of matrices and does not implement optimizations for special types of matrices, like, e.g., symmetric matrices and `trf` represents the operation we want to perform. It is not documented, what the three letters in the code actually stand for, but it appears reasonable that they represent some variation of "triangular factorization". In the subsequent line, we call the LAPACK subroutine `dgetri` with `tri`, possibly standing for "triangular factorization inverse". This subroutine stores its results in place in the `output_matrix` variable, and since this variable is declared as an output of the `inv` subroutine, there is no need for an explicit `return` statement after that. With matrix A inverted, we continue in line 77 and calculate the matrix product CA^{-1}. Fortunately, modern versions of Fortran, starting from Fortran90, have the built-in `matmul` function to perform matrix multiplications. This way, we do not have to find the appropriate BLAS subroutine and construct

the lengthy call. With the subsequent line, we have the product $CA^{-1}B$ calculated. line 78 is where Fortran's purpose-built nature really shines. In this line, we use our result for the product $CA^{-1}B$ to calculate the difference $D-CA^{-1}B$. Instead of requiring lengthy calls to library functions or explicit loops, Fortran offers an intuitive syntax for performing simple arithmetic operations on tensors. Unfortunately, Fortran and LAPACK offer no subroutine to display matrices in a sensible way, hence we have to implement one in line 6 and call it in line 82.

This concludes the implementations with the two older languages in our line-up. The next language we look at is Python. For tensor- and linear algebra operations, we use the library NumPy. A resulting source code for the implementation of our example calculation is shown in Listing 4.3.

Listing 4.3: Example calculation implemented in Python using NumPy for tensor- and linear algebra operations.

```
1  import numpy as np
2
3  if __name__ == '__main__':
4      A = np.array([
5          [1, 0, 1],
6          [-1, 1, 0],
7          [0, -1, 1]
8      ], dtype=np.float64)
9      B = np.diag(np.array([3, 2, 1], dtype=np.float64))
10     C = np.diag(np.array([-1, 3, 1], dtype=np.float64))
11     D = np.array([
12         [-1, 1, 0],
13         [0, -1, 1],
14         [-1, 0, -1]
15     ], dtype=np.float64)
16
17     print(D - C @ np.linalg.inv(A) @ B)
```

In lines 4-15 we enter the matrices A and D as two-dimensional NumPy arrays. For B and C we use NumPy's `diag` function to create diagonal matrices from the values on the diagonal. The entire calculation happens in line 17 with the `@` operator performing matrix multiplication and `np.linalg.inv` inverting the matrix A. Since NumPy's `array` type declares how it wants to be represented in a string by implementing the special `__str__` function, we can simply use Python's built-in `print` function to print a matrix or any other NumPy array to the console.

Julia is similarly concise. The source code of the example calculation implemented in Julia is shown in Listing 4.4.

Listing 4.4: Example calculation implemented in Julia using only the standard library.

```
1  using LinearAlgebra
2
3  function main()
4      A = Matrix{Float64}([
5          1 0 1;
6          -1 1 0;
7          0 -1 1
8      ])
9      B = diagm(Vector{Float64}([3, 2, 1]))
10     C = diagm(Vector{Float64}([-1, 3, 1]))
11     D = Matrix{Float64}([
12         -1 1 0;
13         0 -1 1;
14         -1 0 -1
15     ])
```

```
16
17      display(D - C * inv(A) * B)
18  end
19
20  main()
```

In Julia we do not even need a third-party library, since most linear algebra operations are implemented in the `LinearAlgebra` module that is part of Julia's standard library. In lines 4-15 we enter the matrices A and D as two-dimensional matrices. For B and C, we use `diagm` function from the `LinearAlgebra` module to create diagonal matrices from the values on the diagonal. The entire calculation happens in line 17 with the multiplication operator `*` performing matrix multiplication when applied between matrices and the `inv` function from the `LinearAlgebra` module. In this line, it really shows that Julia was created for scientific calculations; it is almost impossible to make this line more concise and easier to read. The result of the calculation is printed to the console using Julia's built-in `display` function that creates formatted, human-readable output for a whole host of Julia types.

When comparing these four implementations, we make an observation: while all of the four candidate languages are generally considered high-level languages, Python and Julia feature a much higher level of abstraction than C and Fortran, resulting in far shorter code and less need to manually implement auxiliary functions. This allows a developer to implement Python and Julia programs in *significantly* less time and makes the code a lot easier to read and understand. In fact, one may have noticed that the Python and Julia codes of our example calculation feature no comments whatsoever; they are simply not needed when we can just read the code instead. This is a useful feature, because, while comments outside of the documentation of functions, modules, etc. are usually considered good practice in science, in reality, they are merely a workaround for unreadable code. We also notice that Python and Julia allowed us the direct inversion of matrices without forcing us to explicitly perform an LU decomposition. There can be performance benefits in decomposing a matrix in its triangular factors and keeping it in that form throughout calculations, however doing this falls into the category of optimization. In modern software development, the usual order is "make it work, make it pretty, make it fast". This ensures that we only sacrifice readability and maintainability for performance where it is really necessary. As discussed above, in an often-changing simulation under active research, most parts of the code may never get to a point where it is appropriate to make that sacrifice. Therefore, it is desirable for us not to use tools that force that sacrifice onto us. We end the section on the syntax of our candidate languages with the discussion of *boilerplate code*. In programming, the term boilerplate code or just boilerplate, for short, refers to code that is needed to run the program but is not part of the *business logic* of the program. In our case, the business logic is to numerically solve the Langevin equation for our system. A general example of boilerplate code would be the preamble of a program, the part in which we import all the necessary libraries, modules, and functions. Generally, we want our programs to have as little boilerplate code as possible in order to keep them more readable and not "dilute" the physics within them. As we can see in the example source codes above, Python, Julia, and, indeed, many modern languages feature very little boilerplate code and allow us to write concise code. C features more boilerplate code because it requires us to type out instructions for memory management and because we need to call library functions with lengthy names and many arguments to perform basic operations. Fortran can perform some basic operations concisely by employing operators but requires much boilerplate code in many other parts. Consider, for example, the definition of a subroutine in lines 20-24 of Listing 4.2. It consists of four lines and, excluding parameter names and whitespace, 124 characters. That is quite a lot for such a simple task. Therefore, when considering boilerplate code, modern languages such as Python and Julia are a more reasonable choice.

4.3.7 Summary of Comparison and Choice of Language

In this section, we summarize the comparison of the four candidate languages for the simulation core. The results of the comparison are shown in the following table:

Language	Performance	Tooling	Interoperability	Design	Ease of Use	Maturity
C	+	-	-	-	-	+
Fortran	+	-	-	+	-	+
Julia	+	+	+	+	+	-
Python	-	+	+	-	+	+

A plus "+" in this table means that the language is suitable under the given aspect, while a minus "-" indicates that the other languages seem better choices under the given aspect. As we can see, Julia emerges as a clear winner. The only real downside of Julia for our endeavor is its relatively young age; some parts of the language and its tooling are not yet polished. Nevertheless, we pick Julia and use it to implement the simulation core.

5 0+1 Dimensional Systems

With the simulation implemented, we can now calculate observables, although, for now, our focus is on proof-of-concept calculations. That means we are looking for calculations that do not take too long and whose results can be compared to analytical solutions. Because of this, we focus on $0+1$ dimensional calculations, i.e., calculations with zero spatial dimensions, $d = 0$, and one temporal dimension. At first glance, it seems reasonable that fewer spatial dimensions make it cheaper to run simulations since the lattice needs fewer spatial sites. However, that assumption is not necessarily true, because the dynamics of systems vary strongly and non-trivially between spatial dimensions and, indeed, we often find that systems with fewer spatial dimensions require a finer discretization of (space-)time to converge to acceptable results. Nevertheless, in the $0+1$ dimensional case, we can obtain analytic solutions for observables and compare them against the results of our simulation, making that setting a suitable choice for proof-of-concept calculations.

5.1 Density Equation of State

The first observable we calculate is the density. That may seem like an odd choice since particle densities are actually quite tedious to calculate in the pairing field formalism. Yet, we choose to begin with the calculation of densities for two reasons: densities are simple observable that are straightforward to interpret, and densities are extensively studied for our type of system, and we can compare our results to the literature. Density-focused studies of our system at hand include Ref. [83] in three spatial dimensions, Ref. [12] in one spatial dimension, and Ref. [13] for density-density correlation functions. Density-focused studies for zero spatial dimensions are notably absent in this selection because they are available analytically. Nevertheless, building the machinery for calculating and understanding density equations of state serves us well when we move on to higher-dimensional theories in which analytical results are no longer available for comparison and we can rely on literature like the above-mentioned studies for comparison.

5.1.1 Exact Analytical Solution

To obtain analytical results for the density, we go back to the discretized Hamiltonian in Eq. (3.128). In the case of $d = 0$, the Hamiltonian simplifies to

$$\hat{H} = -g\, \hat{\psi}_\uparrow^\dagger \hat{\psi}_\uparrow \hat{\psi}_\downarrow^\dagger \hat{\psi}_\downarrow \,, \tag{5.1}$$

because without space, there is no kinetic energy and only one spatial lattice site, removing the need for a lattice site index on the field operators. In the absence of space, the local density operators in Eq. (3.119) are (global) particle number operators:

$$\hat{N}_\sigma = \hat{n}_\sigma = \hat{\psi}_\sigma^\dagger \hat{\psi}_\sigma \,, \tag{5.2}$$

and the occupation number representation of the Fock space states for a single species σ we define in Eq. (3.120) becomes

$$|n_\sigma\rangle = \left(\hat{\psi}_\sigma^\dagger\right)^{n_\sigma} |0\rangle \ , \tag{5.3}$$

with the property

$$\hat{n}_\sigma |n_\sigma\rangle = n_\sigma |n_\sigma\rangle \ . \tag{5.4}$$

In this simplified setting, it is beneficial to introduce an occupation number representation for product states of occupation number states of both species:

$$\left|n; n_\uparrow, n_\downarrow\right\rangle = \delta_{n,n_\uparrow+n_\downarrow} \ \left|n_\uparrow\right\rangle \otimes \left|n_\downarrow\right\rangle \ , \tag{5.5}$$

introducing the total density $n = n_\uparrow + n_\downarrow$. With these states we can extract the total density n via the total density operator $\hat{n} = \hat{n}_\uparrow + \hat{n}_\downarrow$:

$$\hat{n} \left|n; n_\uparrow, n_\downarrow\right\rangle = n \left|n; n_\uparrow, n_\downarrow\right\rangle \ , \tag{5.6}$$

as well as the single species densities n_σ:

$$\hat{n}_\sigma \left|n; n_\uparrow, n_\downarrow\right\rangle = n_\sigma \left|n; n_\uparrow, n_\downarrow\right\rangle \ . \tag{5.7}$$

These occupation number states are pairwise orthogonal:

$$\left\langle n; n_\uparrow, n_\downarrow \middle| n'; n'_\uparrow, n'_\downarrow \right\rangle = \delta_{n,n'} \delta_{n_\uparrow,n'_\uparrow} \delta_{n_\downarrow,n'_\downarrow} \tag{5.8}$$

and form an eigenbasis of the Hamiltonian:

$$\hat{H} |0;0,0\rangle = \hat{H} |1;0,1\rangle = \hat{H} |1;1,0\rangle = 0 \quad \text{and} \quad \hat{H} |2;1,1\rangle = -g \, . \tag{5.9}$$

Using this basis, we can write the grand-canonical partition function of the system as

$$\begin{aligned} Z(\beta, \mu_\uparrow, \mu_\downarrow) &= \operatorname{tr} e^{-\beta(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \\ &= \sum_{n_\uparrow=0,1} \sum_{n_\downarrow=0,1} \left\langle n_\uparrow + n_\downarrow; n_\uparrow, n_\downarrow \right| e^{-\beta \hat{H}} \left| n_\uparrow + n_\downarrow; n_\uparrow, n_\downarrow \right\rangle e^{\beta\mu_\uparrow n_\uparrow + \beta\mu_\downarrow n_\downarrow} \\ &= 1 + e^{\beta\mu_\uparrow} + e^{\beta\mu_\downarrow} + e^{\lambda + \beta\mu_\uparrow + \beta\mu_\downarrow} \\ &= (1 + e^{\beta\mu_\uparrow})(1 + e^{\beta\mu_\downarrow}) + e^{\beta\mu_\uparrow + \beta\mu_\downarrow}(e^\lambda - 1) \, , \end{aligned} \tag{5.10}$$

with the dimensionless coupling parameter $\lambda = \beta g$. With this explicit expression of the partition function at hand, it is a straightforward matter to calculate the derivative with respect to μ_σ to obtain the particle density for species σ:

$$\begin{aligned} n_\sigma &= \frac{1}{\beta} \partial_{\mu_\sigma} \log Z(\beta, \beta\mu_\uparrow, \beta\mu_\downarrow) \\ &= \partial_{\beta\mu_\sigma} \log Z(\beta, \beta\mu_\uparrow, \beta\mu_\downarrow) \, . \end{aligned} \tag{5.11}$$

In the above equation, we rewrote the μ_σ derivative in terms of a $\beta\mu_\sigma$ derivative. This makes it easier to keep the dimensionless product $\beta\mu_\sigma$ together. Performing the derivative results in the expression

$$\boxed{n_\sigma(\beta\mu_\uparrow, \beta\mu_\downarrow, \lambda) = \frac{e^{\beta\mu_\sigma} + e^{\lambda + \beta\mu_\uparrow + \beta\mu_\downarrow}}{(1 + e^{\beta\mu_\uparrow})(1 + e^{\beta\mu_\downarrow}) + e^{\beta\mu_\uparrow + \beta\mu_\downarrow}(e^\lambda - 1)}} \, . \tag{5.12}$$

Before comparing this expression to numerical results, we shall make some observations. Firstly, we shall consider the limits $\beta\mu_\sigma \to \infty$ and $\beta\mu_\sigma \to -\infty$, respectively. In the limit $\beta\mu_\sigma \to -\infty$, with $\beta\mu_{\sigma'}$ for $\sigma' \neq \sigma$ and λ fixed, we find

$$\lim_{\beta\mu_\sigma \to -\infty} n_\sigma(\beta\mu_\uparrow, \beta\mu_\downarrow, \lambda) = 0\,, \tag{5.13}$$

as expected. In the limit $\beta\mu_\sigma \to +\infty$, with $\beta\mu_{\sigma'}$ for $\sigma' \neq \sigma$ and λ fixed, we find

$$\lim_{\beta\mu_\sigma \to +\infty} n_\sigma(\beta\mu_\uparrow, \beta\mu_\downarrow, \lambda) = 1\,, \tag{5.14}$$

with one fermion firmly occupying the single lattice site. Secondly, we study the analytical expression for the particle density in the non-interacting case $\lambda = 0$. We find

$$\begin{aligned} n_\sigma(\beta\mu_\uparrow, \beta\mu_\downarrow, \lambda = 0) &= \frac{e^{\beta\mu_\sigma} + e^{\beta\mu_\uparrow + \beta\mu_\downarrow}}{(1 + e^{\beta\mu_\uparrow})(1 + e^{\beta\mu_\downarrow})} \\ &= \frac{1}{e^{-\beta\mu_\sigma} + 1} \\ &= n_{\mathrm{F}}(\beta\mu_\sigma)\,, \end{aligned} \tag{5.15}$$

with the Fermi distribution

$$n_{\mathrm{F}}(x) = \frac{1}{e^{-x} + 1}\,. \tag{5.16}$$

That means that in the non-interacting case $\lambda = 0$, the two species exist as two non-interacting ideal Fermi gases that occupy the same system. Lastly, because we plan to study the density of balanced systems in units of the non-interacting density of balanced systems, comment on the effect of fermion interaction by studying the ratio

$$\begin{aligned} \frac{n}{n_0}(\beta\mu, \lambda) &= \frac{n(\beta\mu_\uparrow = \beta\mu, \beta\mu_\downarrow = \beta\mu, \lambda)}{n(\beta\mu_\uparrow = \beta\mu, \beta\mu_\downarrow = \beta\mu, \lambda = 0)} \\ &= \frac{1 + e^{-\lambda - \beta\mu} + e^{\beta\mu} + e^{-\lambda}}{e^{-\lambda - \beta\mu} + e^{\beta\mu} + 2e^{-\lambda}} \end{aligned} \tag{5.17}$$

with the average chemical potential

$$\beta\mu = \frac{\beta\mu_\uparrow + \beta\mu_\downarrow}{2}\,. \tag{5.18}$$

In the limits $\beta\mu \to \pm\infty$ the ratio n/n_0 becomes one for arbitrary values of λ:

$$\lim_{\beta\mu \to \pm\infty} \frac{n}{n_0}(\beta\mu, \lambda) = 1\,. \tag{5.19}$$

That means the presence of interaction between the two fermion species affects the total density only in a certain region around $\beta\mu = 0$.

5.1.2 Mean Field Results

Even though we determined an exact analytical expression for the particle densities above, we also calculate the particle density in the mean-field approximation introduced in Sec. 3.2. We do this because mean-field studies are often used when exact results are unavailable. Moreover, this gives us a chance to study the limitations of this approximation, i.e., where the full description of our system deviates from a mean-field description.

We start by determining an expression for the particle density with the logarithm of the partition function in Eq. (3.91). For our current purposes of the study of a $0+1$ dimensional system, the logarithm of the partition function simplifies to

$$\log \bar{Z}(\bar{\phi}^*, \bar{\phi}, \beta, \mu) = -g\beta\bar{\phi}^*\bar{\phi} + \log\left(\frac{1}{2} + \frac{1}{2}\cosh\left(\beta\sqrt{\mu^2 + g^2\bar{\phi}^*\bar{\phi}}\right)\right) + C(\beta,\mu)\,. \tag{5.20}$$

To obtain this expression, rather than simply setting $d = 0$ in Eq. (3.91), one needs to follow the derivation in Sec. 3.2 from the top without including space. Before we proceed, we rewrite the logarithm of the partition function in terms of the dimensionless coupling $\lambda = \beta g$ and the dimensionless average chemical potential $\beta\mu$. Furthermore, we utilize the $U(1)$ invariance of the system to reformulate $\log \bar{Z}$ in terms of $\varphi = \mathrm{Re}(\bar{\phi})$ without loss of generality. This yields the more convenient expression

$$\log \bar{Z}(\varphi, \beta, \beta\mu) = -\lambda\varphi^2 + \log\left(\frac{1}{2} + \frac{1}{2}\cosh\sqrt{(\beta\mu)^2 + \lambda^2\varphi^2}\right) + C(\beta,\beta\mu)\,. \tag{5.21}$$

We can obtain the particle number N, which is equal to the particle density n in the absence of space, by calculating

$$\begin{aligned}\bar{n}(\varphi) &= \partial_{\beta\mu}\log\bar{Z}(\varphi,\beta,\beta\mu)\\ &= \frac{\beta\mu\,\sinh\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}}{\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}\left(\cosh\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}+1\right)} + \partial_{\beta\mu}C(\beta,\beta\mu)\,.\end{aligned} \tag{5.22}$$

Note that this expression only becomes the physical density n when we evaluate $\bar{n}(\varphi)$ at a mean pairing field φ that extremizes $\log\bar{Z}$. For a physical system, we expect a density of $n = 0$ in the limit $\beta\mu \to -\infty$. This condition allows us to determine the expression $\partial_{\beta\mu}C(\beta,\beta\mu)$:

$$\begin{aligned}0 &\overset{!}{=} \lim_{\beta\mu\to-\infty}\bar{n}(\varphi)\\ &= \lim_{\beta\mu\to-\infty}\frac{\beta\mu\,\sinh\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}}{\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}\left(\cosh\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}+1\right)} + \partial_{\beta\mu}C(\beta,\beta\mu)\\ &= \left(\lim_{\beta\mu\to-\infty}\frac{\beta\mu}{\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}}\right)\left(\lim_{\beta\mu\to-\infty}\frac{\sinh\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}}{1+\cosh\sqrt{(\beta\mu)^2+\lambda^2\varphi^2}}\right) + \partial_{\beta\mu}C(\beta,\beta\mu)\\ &= -1 + \partial_{\beta\mu}C(\beta,\beta\mu)\,.\end{aligned} \tag{5.23}$$

We find the condition

$$\partial_{\beta\mu}C(\beta,\beta\mu) = 1\,, \tag{5.24}$$

which we can satisfy with the choice

$$\partial_{\beta\mu}C(\beta,\beta\mu) = \beta\mu + C'(\beta)\,. \tag{5.25}$$

Generally, the purely β-dependent part $C'(\beta)$ does not need to be zero. However, it does not contribute to the density. Therefore, we drop it for the remainder of the derivation. With $C(\beta,\beta\mu)$ determined, $\log\bar{Z}$ reads

$$\log \bar{Z}(\varphi, \beta, \beta\mu) = -\lambda\varphi^2 + \log\left(\frac{1}{2} + \frac{1}{2}\cosh\sqrt{(\beta\mu)^2 + \lambda^2\varphi^2}\right) + \beta\mu\,. \tag{5.26}$$

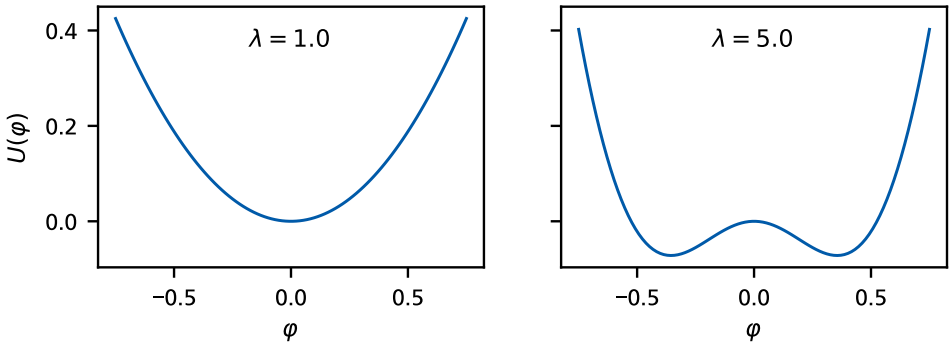


Figure 5.1: Effective potential $U(\varphi) = -\log \bar{Z}$ over the mean pairing field φ. Both figures show the effective potential at $\beta\mu = 0$ and are normalized, such that $U(0) = 0$. In the left figure, U features a single extremum at $\varphi = 0$, and in the right figure, U features a local extremum at $\varphi = 0$ and two minima at $\varphi \neq 0$. These additional minima correspond to a non-physical spontaneous symmetry breaking, i.e., in the $0+1$ dimensional case, such a spontaneous symmetry breaking violates the Mermin-Wagner theorem. Its appearance signals a failure of the mean-field approximation.

A remaining step to calculating the density from this φ-dependent logarithm of the partition function is the undetermined value of the mean pairing field φ. We obtain physical results from the calculation when $\log \bar{Z}$ is extremal with respect to φ, i.e.,

$$\begin{aligned} 0 &\stackrel{!}{=} \partial_\varphi \log \bar{Z}(\varphi, \beta, \beta\mu) \\ &= \frac{\varphi\lambda^2 \ \sinh \sqrt{(\beta\mu)^2 + \lambda^2\varphi^2}}{\sqrt{(\beta\mu)^2 + \lambda^2\varphi^2}\left(1 + \cosh \sqrt{(\beta\mu)^2 + \lambda^2\varphi^2}\right)} - 2\lambda\varphi \,. \end{aligned} \tag{5.27}$$

We observe that the above equation is satisfied in the case $\varphi = 0$, although $\log \bar{Z}$ can have other extrema with respect to φ. Our studies show that $\log \bar{Z}$ forms a second extremum when the coupling exceeds a critical value λ_c, see Fig. 5.1. Such extrema at $\varphi \neq 0$, however, represent non-physical results of the mean-field approximation. This is because a non-zero value of φ corresponds to the spontaneous breaking of the $U(1)$ phase symmetry of the system. However, the Mermin-Wagner theorem [84] states that such a spontaneous symmetry breaking is impossible for our system in spatial dimensions $d \leq 2$. Below, we limit our numerical studies to the range $0 < \lambda \leq 1$, which is well below any value of the aforementioned critical coupling λ_c we have encountered in our mean-field study. Therefore, the non-physical extrema of $\log \bar{Z}$ in the regime $\lambda \geq \lambda_c$ are of no consequence for our present study. For $\lambda < \lambda_c$, the mean-field result for $\log Z$ reads

$$\log \bar{Z}(\varphi = 0, \beta, \beta\mu) = \log Z(\beta, \beta\mu) = \beta\mu + \log\left(\frac{1}{2} + \frac{1}{2}\cosh \beta\mu\right) , \tag{5.28}$$

which results in the density

$$\begin{aligned} n(\beta, \beta\mu) &= \partial_{\beta\mu} \log Z(\beta, \beta\mu) \\ &= 1 + \frac{\sinh(\beta\mu)}{1 + \cosh \beta\mu} = \frac{1 + \cosh \beta\mu + \sinh \beta\mu}{1 + \cosh \beta\mu} \,. \end{aligned} \tag{5.29}$$

It is instructive to rewrite the hyperbolic functions in terms of exponential functions. We obtain:

$$
\begin{aligned}
&\text{[Continuation of Eq. (5.29)]} \\
&= \frac{1+e^{\beta\mu}}{1+\frac{1}{2}\left(e^{\beta\mu}-e^{-\beta\mu}\right)} = \frac{1+e^{\beta\mu}}{\left(e^{2\beta\mu}+2e^{\beta\mu}+1\right)\frac{1}{2e^{\beta\mu}}} \\
&= \frac{1+e^{\beta\mu}}{\left(e^{\beta\mu}+1\right)\frac{e^{\beta\mu}+1}{2e^{\beta\mu}}} = \frac{2e^{\beta\mu}}{e^{\beta\mu}+1}\frac{\frac{e^{\beta\mu}+1}{2e^{\beta\mu}}}{\frac{e^{\beta\mu}+1}{2e^{\beta\mu}}} \\
&= 2\frac{e^{\beta\mu}}{e^{\beta\mu}+1} = 2\frac{1}{e^{-\beta\mu}+1} \\
&= 2n_{\mathrm{F}}(\beta\mu)\,.
\end{aligned}
\tag{5.30}
$$

This means that the density of each of the two fermion species is just the density of a free Fermi gas. In the mean-field solution, the density does not depend on the interaction between the species at all if the interaction coupling λ is sufficiently low. That result is expected, since, as discussed above, the pairing field mediates the interaction between the two fermion species and, as such, fixing the pairing field to zero eliminates interaction between the fermion species in the system. However, above the critical coupling λ_c the densities obtained from the mean-field study deviate from the density of the free gas. Note that the resulting density differs from the exact solution in Eq. (5.17). Beyond our present study, this serves as an educational example of the difficulties of applying mean-field approximations below $d = 3$, as already suggested by the Mermin-Wagner theorem.

5.1.3 Numerical Results

Before we discuss the results of our simulation, we clarify certain details concerning its operation. Our basic unit of performing a simulation is what we call a *run* of the simulation. In a run, we pass *one* set of input parameters to the simulation and it returns *one* data sample for each of the specified observables. Within this run, the simulation solves the Langevin equation iteratively for the given input parameters. At certain points in Langevin time that are determined by the input parameters, the simulation uses the pairing field configuration at the current point in Langevin time to calculate an observable sample point $\mathcal{O}_O^{(i)}$ for observable O. When the simulation has reached its specified maximum Langevin time, it outputs the observable samples $\{\mathcal{O}_O^{(i)}\}$ to a file. The actual Langevin mean, as seen in Eq. (4.26), is performed in processing scripts in the auxiliary part of the simulation. The input parameters for a simulation run can be grouped into three categories:

- **Physical Parameters** – These parameters describe the physical properties of the system. Here, we specify the chemical potentials $\beta\mu_\uparrow$ and $\beta\mu_\downarrow$ for each of the fermion species, their masses $m_\uparrow$ and $m_\downarrow$, and the interaction between the species in the form of the dimensionless interaction parameter $\lambda = \beta^{1-d/2}g$.

- **Lattice Parameters** – These parameters determine the spacetime lattice of the simulation. In this parameter section, we define the temporal lattice extent N_τ, the spatial lattice extent N_x, and the spatial dimension d. These three parameters determine the total number of lattice sites $N = N_\tau N_x^d$. In the case of $d = 0$, the number of lattice sites simplifies to $N = N_\tau$, Beyond that, we also define the dimensionless ratio of the temporal and spatial lattice spacings $r = a_\tau/a_x^2$.

- **Langevin Parameters** – These parameters specify how the simulation solves the Langevin equation of the system. The final Langevin time parameter determines up to which maximum Langevin time the Langevin equation is solved, and the Langevin time spacing parameter δt_{CL} determines the spacing in Langevin time between two consecutive pairing field configurations. Furthermore, we specify an observable decorrelation time. This is a minimum Langevin time we require to pass between two calculations of sample points of observables. These parameters allow us to save computational cost by avoiding calculating sample points that are still correlated to the previous sample points and, thus, do not provide much new information. In this parameter section, we also specify a seed for the pseudo-random number generator. This seed allows us to exactly reproduce a single run by making the Langevin noise deterministic.

Determining Lattice Parameters

In principle it is possible, to just specify the parameters of the simulation lattice N_τ and N_x directly. However, the choices we make here have physical implications; for example, the temporal lattice spacing a_τ defines an energy cutoff that determines which energy values can be resolved on the lattice. Rather than carefully choosing lattice parameters that circumvent such issues, we can use these physical implications explicitly to define criteria that choose the lattice parameters for us, based on the physical parameters of the simulation run. In the following, we want to develop such criteria for the temporal lattice extent N_τ and also for the spatial lattice extent N_x, even though we do not include space in our current calculations. Each of these criteria depends on a control parameter that we have to determine empirically for each observable we calculate. We proceed by determining the values for N_τ and N_x that satisfy each given criterion and then select the smallest values that satisfy all criteria.

For the temporal lattice extent N_τ, we first address the restoration of the Silver-Blaze symmetry, as discussed in Sec. 3.3.5. This restoration makes our temporal derivative deviate from the form that is dictated by the path integral. This deviation is of order $\mathcal{O}(a_\tau^2)$ and, thus, vanishes in the continuum limit but on the lattice, we can limit its effect on the calculation results by imposing an upper limit δ_{SB} on the difference between the derivative factor that respects the Silver-Blaze symmetry and the derivative factor that is produced by the construction of the path integral:

$$e^{a_\tau \mu_{\max}} - (1 + a_\tau \mu_{\max}) < \delta_{\mathrm{SB}} \,. \tag{5.31}$$

In this criterion, we use the chemical potential

$$\mu_{\max} = \max_\sigma \{|\mu_\sigma|\} \,, \tag{5.32}$$

for which the difference in Eq. (5.32) is maximal, since the leading order of this difference is quadratic in $a_\tau \mu_{\max}$, and given the chemical potential imbalance is not too large. Because we can write the chemical potential as

$$a_\tau \mu_{\max} = \frac{\beta \mu_{\max}}{N_\tau} \,, \tag{5.33}$$

we can use the input parameters $\beta\mu_\uparrow$ and $\beta\mu_\downarrow$ to determine a minimum temporal lattice extent N_τ that satisfies the criterion in Eq. (5.32). For the remainder of this work, we fix the empirical constant of this criterion to $\delta_{\mathrm{SB}} = 0.005$, since in most cases the temporal lattice extent is determined by the interaction strength rather than the deviation introduced by the restoration of the Silver-Blaze symmetry. We put this criterion in place mainly to prevent unexpected deviations in results for extreme choices of input parameters.

The second criterion we use to determine the temporal lattice extent is based on the interaction strength of the system. The reciprocal temporal lattice spacing $1/a_\tau$ constitutes an energy

cutoff that limits the energies we can resolve on the lattice. In this spirit, we can define an energy scale

$$E_I = \lambda \frac{\beta^{d/2-1}}{a_x^d}\,, \tag{5.34}$$

which is determined by the interaction parameter λ and requires that this energy scale is less than the cutoff energy imposed by a_τ:

$$\frac{1}{a_\tau} > C_I E_I\,, \tag{5.35}$$

with the empirical constant C_I. Inserting $\beta = N_\tau a_\tau$ into this inequality leads to the criterion

$$N_\tau > \left(\lambda r^{d/2} C_I\right)^{\frac{1}{1-d/2}}\,, \tag{5.36}$$

provided $d < 2$. In the special case of $d = 2$, the coupling parameter g is dimensionless and does not provide us with an energy scale we can use for a criterion, and for $d > 2$, this approach leads to an upper bound for the temporal lattice extent.

To determine the spatial lattice extent N_x, we require that the length of our box $L = N_x a_x$ is sufficiently large compared to the thermal wavelength $\lambda_{\text{th,max}} = (2\pi\beta/m_\sigma)^{1/2}$. To impose the stricter criterion, we chose the lighter of the two species, i.e., for $\sigma' \neq \sigma$, $m_\sigma \leq m_{\sigma'}$, because the lighter species has a greater thermal wavelength. We formulate this requirement for the box length as

$$C_\lambda \lambda_{\text{th,max}} \leq N_x a_x\,, \tag{5.37}$$

with the empirical parameter C_λ. Rewriting all parameters in terms of lattice parameters leads to the criterion

$$N_x \geq C_\lambda \left(\frac{2\pi r}{m_\sigma} N_\tau\right)^{1/2}\,. \tag{5.38}$$

Note that this criterion is conceptually different from the criterion using the interaction energy. While the inverse temperature β is a physical property of our system, the box length $L = N_x a_x$ is an auxiliary quantity that we extrapolate out of observable results. Also note that this criterion tells us how the spatial lattice extent scales with the temporal lattice extent: we find $N_x \propto N_\tau^{1/2}$.

Processing Simulation Runs

When we perform a simulation run, we calculate a sample of observable values, but before we can visualize and interpret the observable, there are some processing steps we need to apply. The first thing to do is to calculate the Langevin mean as shown in Eq. (4.26) to obtain the actual value of the observable calculated by the simulation. Along with the mean, we also estimate the uncertainty of the result using the Jackknife method in Sec. 2.2.4. We then have an observable value that depends on all of the simulation parameters since all of the simulation parameters affect its outcome. Among those are, of course, the physical parameters, such as the chemical potentials, that we expect to affect the calculated observables, but also technical parameters, such as the Langevin parameters and the lattice parameters. For the Langevin parameters, we find that the calculated density values are largely independent of the Langevin time spacing around a range of values that we pick the spacing from and that increasing the final Langevin time just causes the simulation to create larger samples, which decreases the statistical uncertainty of the calculated densities without changing their means beyond the bounds of their statistical uncertainties.

For the lattice parameters, however, we notice that the calculated densities generally show a dependence on the temporal lattice extent N_τ. This dependence is particularly strong for $\beta\mu >$

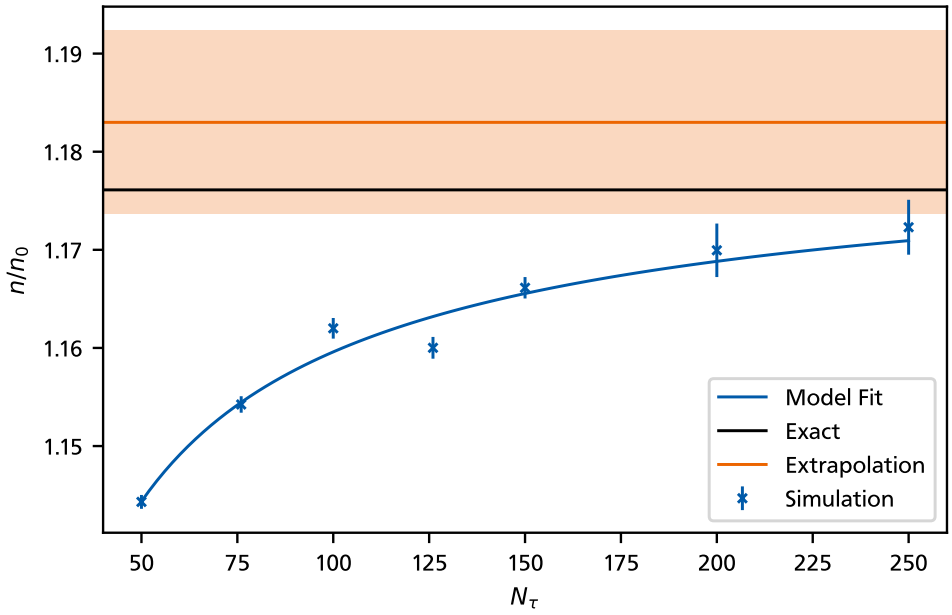


Figure 5.2: Illustration of the extrapolation of finite-lattice effects in the density observable at the point $\lambda = \beta\mu_\uparrow = \beta\mu_\downarrow = 1$. The densities are shown in units of the free density at $\lambda = 0$, and the errorbars indicate the standard deviation of the results. The fit takes the uncertainties of the individual simulation runs into account and results in the shown extrapolated value. The shaded region around the extrapolated value represents the standard deviation of the estimate on either side of the solid line.

-1. This dependence is not physical since the temporal lattice is merely an approximation we introduced to allow us to perform a step in the derivation of the path integral and, as such, the physical result is obtained in the limit $N_\tau \to \infty$. Consequently, this nonphysical dependence is an example of a *finite-lattice artifact*. Figure 5.2 shows the dependence of the calculated densities on the temporal lattice extent for a single point in the physical parameter space at $\lambda = \beta\mu_\uparrow = \beta\mu_\downarrow = 1$ and $m_\uparrow = m_\downarrow = 1$. To remove this nonphysical dependence and formally reach the limit $N_\tau \to \infty$, we fit the calculated densities to a model and use the model fit to determine the density at infinite temporal lattice extent. To this end, we choose the model

$$n_{\text{model}}(N_\tau) = c_1 - c_2 N_\tau^{-c_3} \tag{5.39}$$

that allows us to find the density at infinite temporal lattice extent via

$$\lim_{N_\tau \to \infty} n_{\text{model}}(N_\tau) = c_1 \,. \tag{5.40}$$

An example of this fit and the extrapolation $N_\tau \to \infty$ can also be seen in Fig. 5.2. The fit uses the uncertainties of the single simulation runs as well as the intrinsic uncertainty of fitting data points to a model to determine an appropriate uncertainty for the extrapolated density in the c_1 parameter of the fit.

As mentioned above, this finite-lattice effect is most prominent for $\beta\mu > -1$ and negligible for smaller chemical potentials $\beta\mu < -1$. In that regime, we calculate the density at different reasonably large temporal lattice extents to verify that the dependence is indeed negligible, and then combine them into a single estimate through the use of a maximum-likelihood estimator, in our case, a weighted average that takes the uncertainty of each individual simulation run into account. Suppose we performed N simulation runs for different temporal lattice extents and

obtained the densities $\{n_i\}$ for $i = 1, ..., N$ with standard deviations $\{\sigma_{n_i}\}$, then the maximum-likelihood estimate for the density is the weighted average

$$\bar{n} = \left(\sum_{i=1}^{N} \frac{1}{\sigma_{n_i}^2} \right)^{-1} \sum_{i=1}^{N} \frac{n_i}{\sigma_{n_i}^2} \tag{5.41}$$

with the uncertainty

$$\sigma_{\bar{n}} = \left(\sum_{i=1}^{N} \frac{1}{\sigma_{n_i}^2} \right)^{-1/2} . \tag{5.42}$$

To understand why the maximum-likelihood estimator is realized by an average that is weighted with the uncertainties of the values, let us consider a brief example. Suppose we run the simulation five times and obtain the five densities[1] and their respective standard deviations:

n	1.12	0.94	1.06	2.26	1.16
σ_n	0.03	0.09	0.04	0.94	0.07
σ_n/n	0.027	0.096	0.038	0.416	0.060

One might then think that the density for this set of physical parameters lies around 1.31 ± 0.22 because that are the mean and standard deviation of the mean for this set of densities. However, if we look closer, all but the fourth density paint a relatively consistent picture. Just the fourth density is an extreme outlier that is significantly above the other densities. This outlier also features a much larger uncertainty than the other densities with a relative uncertainty of $41.6\,\%$. Hence, loosely speaking, it is far from certain that the density should actually be that high. Nevertheless, the naive mean does not take that high uncertainty into account, and the outlier drastically increases the total estimate of the density and its uncertainty. The maximum-likelihood estimator is more robust; because the outlier features such a large uncertainty, it has little weight in the sum that determines the estimate for the density. Therefore, the total estimate is dominated by the four more regular densities, and beyond that, the uncertainty of the total density estimate is much lower. One could say that the maximum-likelihood estimator mostly ignores the fourth density because it contains little additional information. With the maximum-likelihood estimator, we find 1.10 ± 0.02 as the total estimate for the density, which is much more in line with the less uncertain densities we calculated. As such, the maximum-likelihood estimator is a suitable tool to combine multiple simulation runs into a total estimate.

Balanced Densities

In this section, we discuss the numerical results obtained from our simulation. We focus on balanced densities with $\beta\mu_\uparrow = \beta\mu_\downarrow = \beta\mu$ and $\beta h = 0$. As a first simple test for our simulation, we use it to calculate the density equation of state for a non-interacting system with $\lambda = 0$. In the non-interacting case, the analytical solution in Eq. (5.12) and the mean-field solution in Eq. (5.30) agree on

$$n(\beta\mu) = 2n_{\mathrm{F}}(\beta\mu) . \tag{5.43}$$

As for our simulation, we need to calculate the expression $\mathcal{O}^{(i)}_{N_\sigma}$ in Eq. (4.36). The only part of the expression $\mathcal{O}^{(i)}_{N_\sigma}$ that depends on the current Langevin time step is the fermion matrix $\mathcal{M}^i$. It is given by

$$\mathcal{M}^{(i)} = \begin{pmatrix} D_\tau^{(\mathrm{bw})}(\mu_\uparrow) - \frac{RD_\triangle}{2m_\uparrow} & -g\,\mathrm{diag}\left(R\left(\phi_1^{(i)} + i\phi_2^{(i)}\right)\right) \\ -g\,\mathrm{diag}\left(\phi_1^{(i)} - i\phi_2^{(i)}\right) & D_\tau^{(\mathrm{fw})}(\mu_\downarrow) + \frac{AD_\triangle}{2m_\downarrow} \end{pmatrix} . \tag{5.44}$$

[1] Recall that in zero-dimensional space densities are identical to particle numbers and dimensionless.

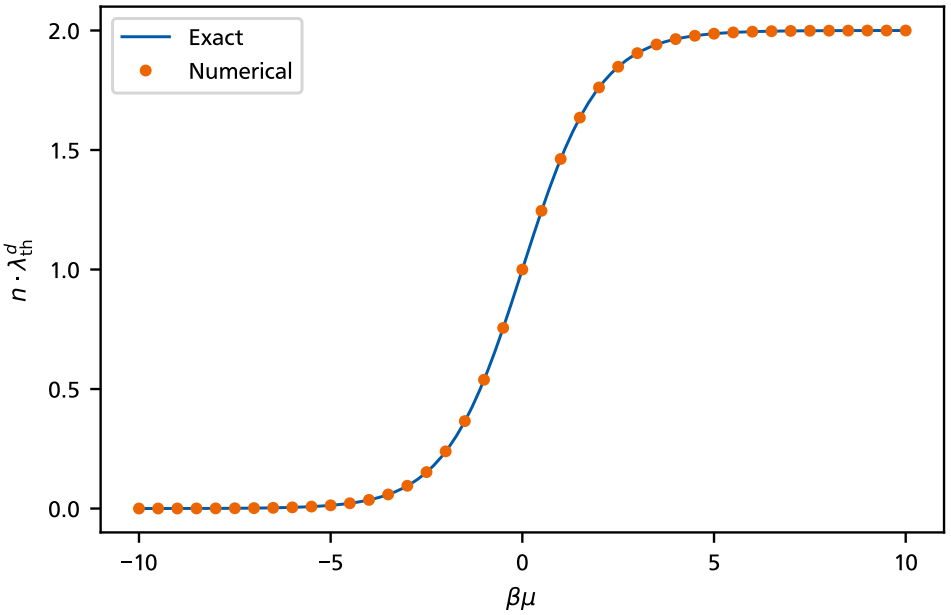


Figure 5.3: Dimensionless density for a non-interacting system, i.e., $\lambda = 0$. No errorbars are shown because in the non-interacting case, no simulation is performed, and the obtained results are numerically exact.

We can see that in the non-interacting case, the coupling constant g becomes zero, and $\mathcal{M}^{(i)}$ reduces to

$$\mathcal{M}^{(i)} = \begin{pmatrix} D_\tau^{(\mathrm{bw})}(\mu_\uparrow) - \frac{RD_\triangle}{2m_\uparrow} & \mathbb{0} \\ \mathbb{0} & D_\tau^{(\mathrm{fw})}(\mu_\downarrow) + \frac{AD_\triangle}{2m_\downarrow} \end{pmatrix}; \tag{5.45}$$

it no longer depends on the pairing-field configuration at all. In fact, the entire expression $\mathcal{O}_{N_\sigma}^{(i)}$ is independent of the Langevin time in the non-interacting case. As such, we can sidestep the entire Langevin process and calculate the observable directly rather than simulating the system. The results of this calculation are shown in Fig. 5.3 and, as expected, they are in perfect agreement with the analytical results.

Our simulation's ability to reproduce the non-interacting density indicates that there are no fundamental problems in the code or the theory. However, the more interesting proof-of-concept lies, of course, in the calculation of interacting densities. Figure 5.4 shows the results of such simulations. To obtain this result, we run the simulation for multiple lattice sizes at each point $(\beta\mu, \lambda)$ in the space of physical parameters and combine these runs into a total estimate. For $\beta\mu \lesssim -1$, we use the maximum-likelihood estimator described above because the finite-lattice artifacts in this regime are negligible. For $\beta\mu \gtrsim -1$, we perform the above-mentioned extrapolation to infinite lattices to remove the finite-lattice effects. The errorbars in the figure represent the standard deviation of the total estimate, and within these uncertainties, our results are in excellent agreement with the exact analytical solution. Thus, our simulation is able to accurately capture the effects of interaction that are completely absent in the mean-field solution.

Beyond our study in $0+1$ dimensions, Ref. [12] contains an analogous study in $1+1$ dimensions. In contrast to our approach, the authors use a density-based auxiliary field to more efficiently calculate densities and related observables. Figure 5.5 shows the density equation of state obtained in said $1+1$-dimensional study. Both studies show the same qualitative behavior as in $0+1$ dimensions. The effects of the interaction concentrated around $\beta\mu = 0$ with $n/n_0 \to 1$

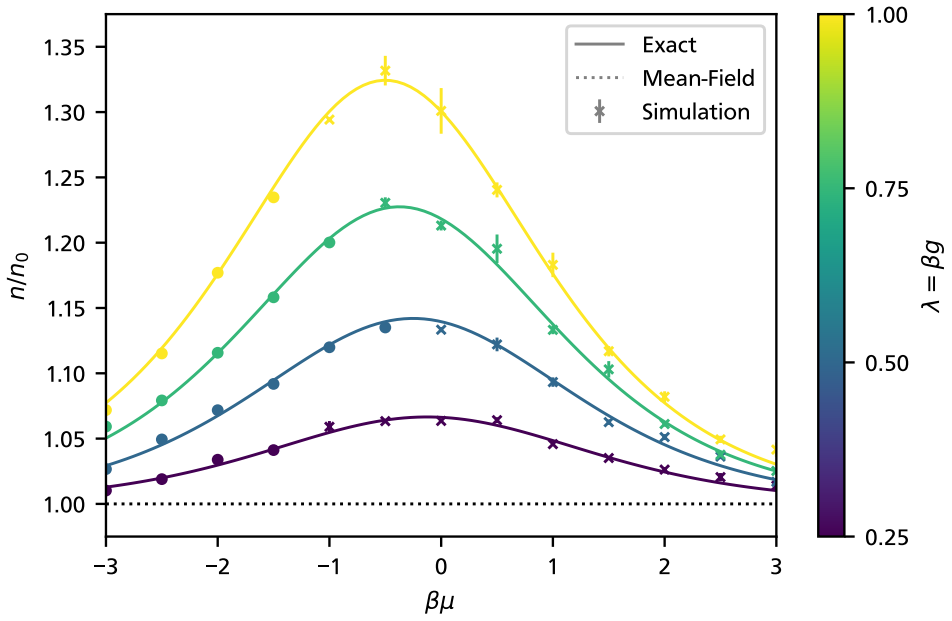


Figure 5.4: Density in units of the non-interacting density for interacting system with $\lambda \in \{0.25, 0.5, 0.75, 1.0\}$. The errorbars represent the statistical uncertainty of the data points in terms of their standard deviation. For data points marked with a cross, finite-lattice effects are extrapolated out, while those marked with a dot are obtained using a maximum-likelihood estimator.

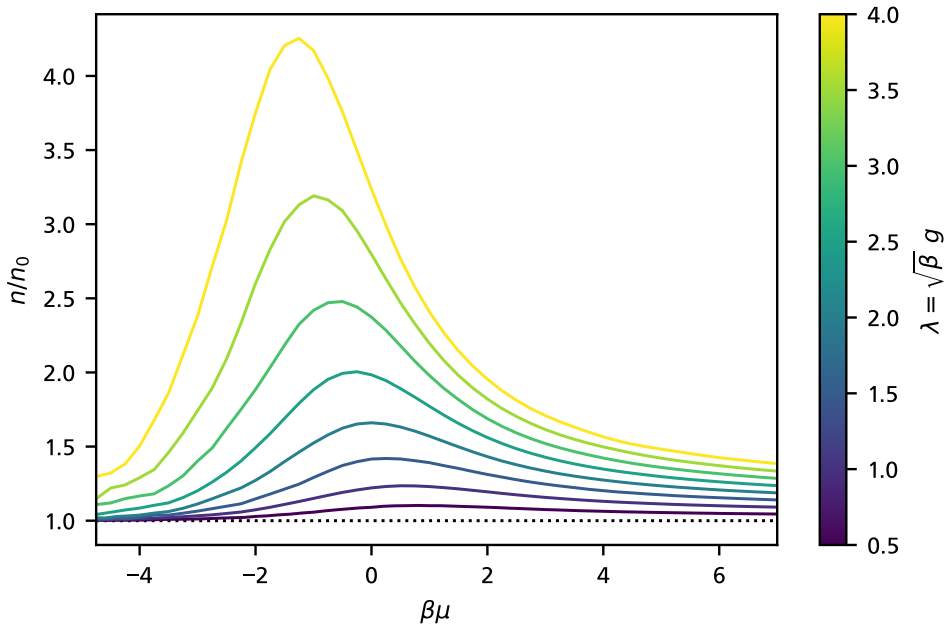


Figure 5.5: Density equation of state for $1+1$-dimensional systems with data taken from Ref. [12] for dimensionless couplings $\lambda \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$. As in the $0+1$-dimensional case, the presence of interaction mainly affects the density around $\beta\mu = 0$.

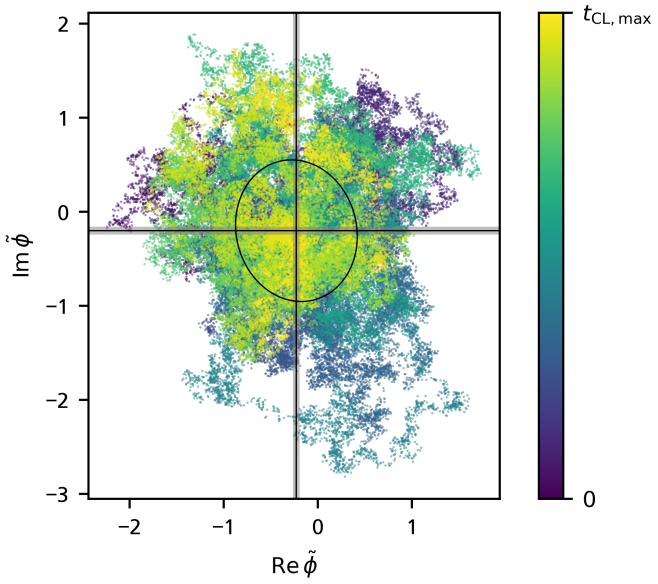


Figure 5.6: Complex Langevin sample points of the time averaged pairing field $\tilde{\phi}$ in $d = 0$ spatial dimensions shown in the complex plane. The color indicates at which Langevin time each sample point is calculated. The simulation ends at the final Langevin time $t_{\mathrm{CL,max}}$. The solid horizontal and vertical lines show the sample mean of the cloud, which lies at $-0.22(03) - i0.20(04)$ with the standard deviation of the sample mean obtained through Jackknife resampling and indicated by the shaded area around the lines. The ellipse shows the principle standard deviations of the sample, i.e., the square roots of the eigenvalues of the covariance matrix of the sample. The semi-axes of the ellipse are the principle standard deviations in length and are oriented along the eigenvectors of the covariance matrix. The simulation is carried out with the physical parameters $\lambda = \beta\mu = 1$ and the lattice size is determined with $C_I = 150$ resulting in a temporal lattice extent of $N_\tau = 150$.

for $|\beta\mu| \to \infty$. Therefore, the underlying physics seem similar, and we can use the results of Ref. [12] as a benchmark in future $1+1$-dimensional studies.

To conclude the discussion of the density equation of state, we note that the simulations that resulted in the results shown in Fig. 5.4 are computationally quite expensive because they run for multiple days. We need relatively large lattices to obtain accurate results for the density, and that strongly affects the duration of simulation runs. That is, however, neither unexpected nor a problem since the pairing-field formalism is designed for pairing-related observables, and the density equation of state serves merely as a proof-of-concept.

5.2 Correlation Functions

In this section, we study quantities that allow us to investigate pairing. Before we get into correlation functions of the pairing field at different points in time, however, we study a simple observable that allows us to verify that our Complex Langevin sampling process is indeed working as intended. An observable which provides us with this insight is the expectation value of the spacetime-averaged pairing field $\langle\phi\rangle_{\tau,\mathbf{r}}$. The expectation value of the pairing field is equal to the expectation value of two fermion fields or, equivalently, two fermionic annihilation operators

$$\langle\phi(\tau,\mathbf{r})\rangle = \langle\hat{\psi}_\downarrow(\tau,\mathbf{r})\hat{\psi}_\uparrow(\tau,\mathbf{r})\rangle\,, \tag{5.46}$$

as shown in Sec. 3.1.2. Hence, we expect this expectation value to be zero in the absence of a term that explicitly breaks the $U(1)$ symmetry. With that being stated, we define the observable as

$$\tilde{\phi} = \lambda_{\text{th}}^{d/2} \langle \phi \rangle_{\tau,\mathbf{r}} = \frac{\lambda_{\text{th}}^{d/2}}{\beta N_x^d a_x^d} \int_0^\beta \mathrm{d}\tau \int_{\mathbf{r}} \mathrm{d}^d r \; \phi(\tau, \mathbf{r}) \,, \tag{5.47}$$

with the continuous pairing-field $\phi(\tau, \mathbf{r})$ in this expression being the same (dimensionful) pairing-field in Eq. (3.24). To calculate this observable with our dimensionless simulation, we rescale the actual average with a power of the species-independent thermal wavelength λ_{th}, defined in Eq. (4.38), to obtain an overall dimensionless observable. The expression $\mathcal{O}_{\tilde{\phi}}^{(i)}$ for this observable is given by

$$\mathcal{O}_{\tilde{\phi}}^{(i)} = (2\pi r N_\tau)^{d/4} \frac{1}{N} \sum_{j=1}^{N} \phi_j^{(i)} \,, \tag{5.48}$$

with $\phi^{(i)}$ being the dimensionless un-starred pairing field configuration vector at Langevin time step i. Apart from the prefactor that scales the observable to satisfy the definition in Eq. (5.47), obtaining the expression in Eq. (5.48) is trivially simple; we only have to average over all components of the field configuration vector we have available within the simulation anyway. A CL sampling process for this observable is shown in Fig. 5.6. We see that we end up around the expected result of zero. Furthermore, we see that the sampling process demonstrates no ill behavior, such as getting stuck or doing large jumps.

As a physically more interesting example of correlation functions, we consider the two-point function of the pairing field

$$G(\tau, \tau') = \langle \phi(\tau) \phi^*(\tau') \rangle \,. \tag{5.49}$$

This function allows us to study the properties of fermion pairs, which are expected to form in a superfluid. Since we limit our discussion to the $0+1$-dimensional case, the two-point function does not come with a dependence on spatial coordinates. Beyond that, the correlation function is already dimensionless, and we do not need to rescale it with the thermal wavelength. This may result in different prefactors between the analytical and numerical solutions, but we can avoid this issue by manually normalizing both results.

5.2.1 Exact Analytical Solution

To obtain an exact analytical solution for the two-point function in Eq. (5.49), we employ the operator formalism. As we noted in Sec. 3.1.2, we can represent the two-point function G in terms of fermionic fields, as

$$G(\tau, \tau') = \langle \phi(\tau) \phi^*(\tau') \rangle = \langle \psi_\uparrow(\tau) \psi_\downarrow(\tau) \psi_\downarrow^*(\tau') \psi_\uparrow^*(\tau') \rangle \,. \tag{5.50}$$

To express this expectation value in terms of fermionic operators rather than fields, we need to define time-dependent fermionic field operators in the imaginary-time Heisenberg picture. We begin by defining time-dependent field operators in real time:

$$\begin{aligned} \hat{\psi}_{\sigma,\text{real}}(t) &= e^{+it(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \; \hat{\psi}_\sigma \; e^{-it(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \quad \text{and} \\ \hat{\psi}_{\sigma,\text{real}}^\dagger(t) &= e^{+it(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \; \hat{\psi}_\sigma^\dagger \; e^{-it(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \,, \end{aligned} \tag{5.51}$$

with the operators from Sec. 3.3.2 and then perform the Wick rotation $t \to -i\tau$ to obtain

$$\begin{aligned} \hat{\psi}_\sigma(\tau) &:= \hat{\psi}_{\sigma,\text{real}}(t = -i\tau) = e^{+\tau(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \; \hat{\psi}_\sigma \; e^{-\tau(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \quad \text{and} \\ \hat{\psi}_\sigma^\dagger(\tau) &:= \hat{\psi}_{\sigma,\text{real}}^\dagger(t = -i\tau) = e^{+\tau(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \; \hat{\psi}_\sigma^\dagger \; e^{-\tau(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \,. \end{aligned} \tag{5.52}$$

Furthermore, we need to remember that the path-integral formalism produces time-ordered correlation functions. In the operator formalism, we need to explicitly include time ordering by means of the time ordering operator[2] $\mathcal{T}$:

$$\begin{aligned} G(\tau,\tau') &= \langle \mathcal{T}(\hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau'))\rangle \\ &= \langle \theta(\tau-\tau')\hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau') + \theta(\tau'-\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau')\hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\rangle\,. \end{aligned} \tag{5.53}$$

We split the analytical calculation of this correlation function into the two cases $\tau > \tau'$ and $\tau < \tau'$ and begin with the former. The expectation value in the operator formalism is given by

$$\begin{aligned} G(\tau,\tau') &= \langle \hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau')\rangle \\ &= \frac{1}{\mathcal{Z}}\,\mathrm{tr}\left(\hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau')\; e^{-\beta(\hat{H}-\mu_\uparrow\hat{n}_\uparrow-\mu_\downarrow\hat{n}_\downarrow)}\right)\,. \end{aligned} \tag{5.54}$$

In this expression, we perform the trace over the basis of states in occupation number representation:

$$\begin{aligned} &\text{[Continuation of Eq. (5.54)]} \\ &= \frac{1}{\mathcal{Z}}\sum_{n_\uparrow=0,1}\sum_{n_\downarrow=0,1}\left\langle n_\uparrow+n_\downarrow; n_\uparrow, n_\downarrow\right| \hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau') \\ &\qquad\qquad \cdot e^{-\beta(\hat{H}-\mu_\uparrow\hat{n}_\uparrow-\mu_\downarrow\hat{n}_\downarrow)}\left|n_\uparrow+n_\downarrow; n_\uparrow, n_\downarrow\right\rangle \\ &= \frac{1}{\mathcal{Z}}\left\langle 0;0,0\right| \hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau')\left|0;0,0\right\rangle \\ &= \frac{1}{\mathcal{Z}}e^{\tau(g+\mu_\uparrow+\mu_\downarrow)}\left\langle 2;1,1|2;1,1\right\rangle e^{-\tau'(g+\mu_\uparrow+\mu_\downarrow)} \\ &= \frac{1}{\mathcal{Z}}e^{\beta(g+\mu_\uparrow+\mu_\downarrow)\frac{\tau-\tau'}{\beta}}\,. \end{aligned} \tag{5.55}$$

In the case $\tau < \tau'$, we proceed analogously. We begin with the definition

$$\begin{aligned} G(\tau,\tau') &= \langle \hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau')\rangle \\ &= \frac{1}{\mathcal{Z}}\,\mathrm{tr}\left(\hat{\psi}_\uparrow(\tau)\hat{\psi}_\downarrow(\tau)\hat{\psi}_\downarrow^\dagger(\tau')\hat{\psi}_\uparrow^\dagger(\tau')\; e^{-\beta(\hat{H}-\mu_\uparrow\hat{n}_\uparrow-\mu_\downarrow\hat{n}_\downarrow)}\right)\,, \end{aligned} \tag{5.56}$$

and perform the trace in the basis of states in occupation number representation. Because the order of creation and annihilation is reversed opposed to the case $\tau > \tau'$, we find a non-zero

[2] Strictly speaking, the time-ordering operator is a super operator since it maps operators to operators.

contribution by a different state:

$$
\begin{aligned}
&\text{[Continuation of Eq. (5.56)]} \\
&= \frac{1}{\mathcal{Z}} \sum_{n_\uparrow=0,1} \sum_{n_\downarrow=0,1} \left\langle n_\uparrow + n_\downarrow; n_\uparrow, n_\downarrow \right| \hat{\psi}_\downarrow^\dagger(\tau') \hat{\psi}_\uparrow^\dagger(\tau') \hat{\psi}_\uparrow(\tau) \hat{\psi}_\downarrow(\tau) \\
&\qquad\qquad \cdot e^{-\beta(\hat{H} - \mu_\uparrow \hat{n}_\uparrow - \mu_\downarrow \hat{n}_\downarrow)} \left| n_\uparrow + n_\downarrow; n_\uparrow, n_\downarrow \right\rangle \\
&= \frac{1}{\mathcal{Z}} \left\langle 2; 1, 1 \right| \hat{\psi}_\downarrow^\dagger(\tau') \hat{\psi}_\uparrow^\dagger(\tau') \hat{\psi}_\uparrow(\tau) \hat{\psi}_\downarrow(\tau) \left| 2; 1, 1 \right\rangle e^{\beta(g+\mu_\uparrow+\mu_\downarrow)} \\
&= \frac{1}{\mathcal{Z}} e^{-\tau'(g+\mu_\uparrow+\mu_\downarrow)} \left\langle 0; 0, 0 | 0; 0, 0 \right\rangle e^{\tau(g+\mu_\uparrow+\mu_\downarrow)} e^{\beta(g+\mu_\uparrow+\mu_\downarrow)} \\
&= \frac{1}{\mathcal{Z}} e^{-\tau'(g+\mu_\uparrow+\mu_\downarrow)} \left\langle 0; 0, 0 | 0; 0, 0 \right\rangle e^{\tau(g+\mu_\uparrow+\mu_\downarrow)} e^{\beta(g+\mu_\uparrow+\mu_\downarrow)} \\
&= \frac{1}{\mathcal{Z}} e^{(g+\mu_\uparrow+\mu_\downarrow)(\tau-\tau')} e^{\beta(g+\mu_\uparrow+\mu_\downarrow)} \\
&= \frac{1}{\mathcal{Z}} e^{\beta(g+\mu_\uparrow+\mu_\downarrow)\left(\frac{\tau-\tau'}{\beta}+1\right)} .
\end{aligned}
\tag{5.57}
$$

In summary, we find the following exact solution for the two-point function

$$
G(\tau, \tau') = \frac{1}{\mathcal{Z}} \begin{cases} e^{\beta(g+\mu_\uparrow+\mu_\downarrow)\frac{\tau-\tau'}{\beta}} & , \tau > \tau' , \\ e^{\beta(g+\mu_\uparrow+\mu_\downarrow)\left(\frac{\tau-\tau'}{\beta}+1\right)} & , \tau < \tau' . \end{cases} \tag{5.58}
$$

We could insert the expression for $\mathcal{Z}$ from Eq. (5.10) into this result but since we manually normalize it anyway, there is no need.

5.2.2 Numerical Results

For the numerical calculations of correlation functions, we can in principle use criteria like we defined in Sec. 5.1.3 to determine the lattice parameters based on the physical parameters of the system. However, we observe that far smaller lattices are sufficient to produce acceptable results for correlation functions compared to densities. Therefore, we fix the lattice size to $N_\tau = 26$ for our numerical studies of correlation functions, regardless of the interaction λ and the chemical potentials $\beta\mu_\sigma$. Furthermore, we limit our exploratory study of correlation functions to the chemical potentials $\beta\mu_\uparrow = \beta\mu_\downarrow = 0$. In the calculations, we focus on larger final Langevin times to increase the precision of the results. Concretely, we choose a final Langevin time of $t_{\mathrm{CL,max}} = 45\,000$ with a spacing of $\delta t_{\mathrm{CL}} = 0.05$ for all runs of the study. The temporal lattice of our simulation has N_τ sites that correspond to the times

$$
(\tau_i = i\, a_\tau \mid i \in \{0, ..., N_\tau - 1\}) . \tag{5.59}
$$

That means that we can compute the correlation function in the form of an $N_\tau \times N_\tau$-matrix (G_{ij}) with the entries

$$
G_{ij} = G(\tau_i, \tau_j) . \tag{5.60}
$$

It also means that the maximum numerically accessible values of τ and τ' in $G(\tau, \tau')$ are given by the temporal extent β of the simulation lattice. Numerically, we are limited to the case $\tau, \tau' < \beta$ while the exact solution in Eq. (5.58) is valid for all real values. Figure 5.7 shows correlation function matrices of the type shown in Eq. (5.60) for multiple values of the interaction λ. In the plot, we manually set the diagonal values of all two-point function matrices to zero to exclude

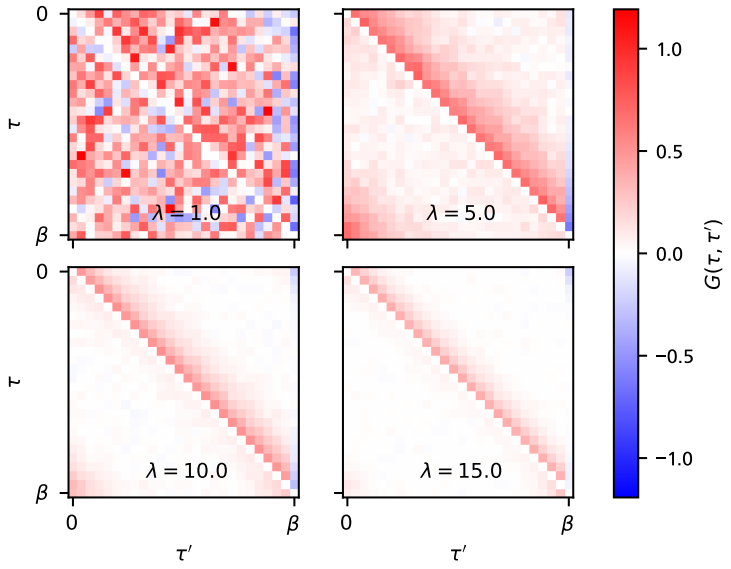


Figure 5.7: Two-point functions for multiple interactions λ. The subplots show matrices defined in Eq. (5.60) which show the two-point function over all sites of the temporal lattice of the simulation. The diagonal entries of all matrices are manually set to zero to exclude the case of $\tau = \tau'$ in the plot. All simulations are performed on a lattice with $N_\tau = 26$ temporal lattice sites at $\beta\mu_\uparrow = \beta\mu_\downarrow = 0$. The Langevin equation is solved up to $t_{\mathrm{CL,max}} = 45\,000$ with a spacing of $\delta t_{\mathrm{CL}} = 0.05$.

the equal time case $\tau = \tau'$ from the plot. We do this because, as we can see in the exact solution in Eq. (5.58), the two-point function features a discontinuity at that point. In numerical results, this discontinuity takes the form of large values on the diagonal. These large values make it difficult to discern the off-diagonal features of the two-point function if we do not remove them manually. In Fig. 5.7, we see that for a given fixed lattice size the signal-to-noise ratio of the two-point function increases with increasing interaction λ. For the case of $\lambda = 1.0$ the structure of the two-point function is barely visible in the statistical noise, and we do not include it in our analyses below; it would seem larger lattices are required to access this region of weaker interaction. Furthermore, we see that in all matrices the structure of the two-point function is more pronounced in the triangle above the diagonal. That means that in the upper triangle that corresponds to the case $\tau < \tau'$ we find a higher signal-to-noise ratio. For this reason, we focus the remainder of the study on this case. Beyond that, we also see the two-point function increases in the upper right-hand and lower left-hand corners of the matrices. This behavior is an artifact of the finite lattice size, and we do not include it in our further analyses, i.e., we limit our analyses to the region $|\tau - \tau'| \leq \beta/2$.

To compare the numerical results to the analytical solution in Eq. (5.58), we study the function $G(0, \tau)$. Considering the matrices in Fig. 5.7, this corresponds to taking the first row of data. In the exact solution, we see that the two-point function $G(\tau, \tau')$ has a translation invariance and only depends on the difference of τ and τ'. By averaging multiple rows of the two-point function matrices, we could, in principle, use that invariance to obtain a more accurate estimate of $G(0, \tau)$. In Fig. 5.7, we can see that this invariance is realized along the diagonal, and we could average over multiple rows at constant $\tau - \tau'$. The constraint $|\tau - \tau'| \leq \beta/2$ that is imposed by finite-lattice artifacts would cause us to average over more values the closer they are to the diagonal. In any case, for the purpose of a proof-of-concept calculation, we refrain from this procedure. Figure 5.8 shows the comparison of our numerical results with the exact solution.

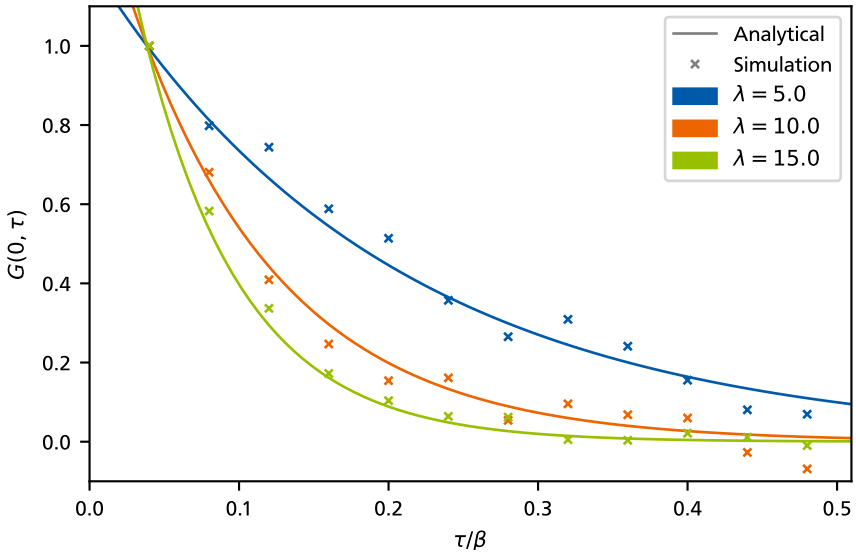


Figure 5.8: Two-point function $G(0,\tau)$ for multiple values of λ. The crosses represent the numerical results, whereas the solid lines show the exact solution from Eq. (5.58). The plot only contains the times $0 < \tau < \beta/2$ to omit finite-lattice artifacts and the value at the discontinuity point $\tau = 0$. The statistical uncertainties obtained through Jackknife resampling are not shown because they are smaller than the markers. This indicates that the present deviations are of systematic nature.

We see that the numerical results clearly reproduce the exponential structure of the exact solution even without further averaging. Additionally, we find that the agreement of the numerical results with the exact solution increases with increasing interactions λ.

Beyond the time-domain function $G(0,\tau)$, it interesting to consider its Fourier transform $\tilde{G}(1/\tau)$. Figure 5.9 shows the magnitude $|\tilde{G}(1/\tau)|^2$ and the phase $\arg \tilde{G}(1/\tau)$ for the time-domain data in Fig. 5.8. Similarly to the time domain, we find a generally good agreement between the numerical results and the exact solution, and the agreement increases for increasing values of λ. This could allow us to analytically extend the data to real times in future studies. With such an analytical extension, we could access interesting quantities such as the real-time spectral function of the system.

Additionally, we can use the analytical analyses in Refs. [54, 85] to interpret our results further. For a $0+1$-dimensional system, the two-point function can be written as

$$G(0,\tau) = \sum_{n\geq 0} c_n e^{-(E_n - E_0)\tau}\,, \tag{5.61}$$

with E_n being eigenenergies of the Hamiltonian and expansion coefficients c_n. The Hamiltonian for our system in $0+1$ dimensions in Eq. (5.1) has only one eigenstate above the ground state. Consequently, the two-point function in this case is given by just one exponential function:

$$G(0,\tau) = c_1 e^{-(E_1 - E_0)\tau}\,. \tag{5.62}$$

This is consistent with the exact solution we obtained above in Eq. (5.58). Inversely, we could argue that since our two-point function is given by one single exponential function, our system must have only one excited state. In frequency space, that means that the real-time spectral function has just one δ peak at a non-zero frequency. Furthermore, Eq. (5.62) shows us that

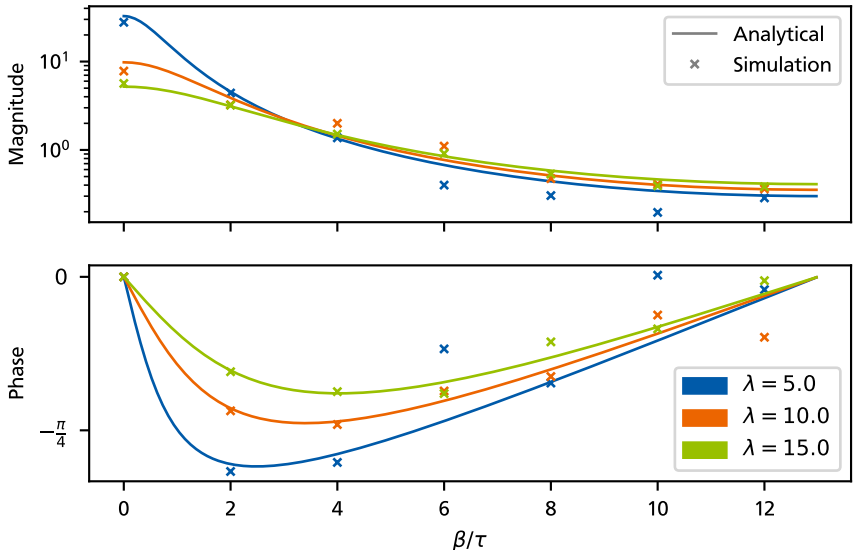


Figure 5.9: Fourier transform of $G(0,\tau)$. The crosses represent the numerical results, while the solid lines show the Fourier transform of the analytical solution from Eq. (5.58). The upper plot shows the squared absolute value of the Fourier transform, and the lower plot shows its complex argument. Only points with $0 < \tau < \beta/2$ are used as input for the Fourier transform to avoid including finite-lattice artifacts and the value at the discontinuity point $\tau = 0$.

the steepness of the exponential function depends on the difference $E_1 - E_0$. For our system, at $\beta\mu = 0$, this difference is given by $g = \lambda/\beta$. This is consistent with the exact solution in Eq. (5.58) and our observation in Fig. 5.8. As we see in Fig. 5.7, shallow correlation functions are numerically unfavorable because they feature a lower signal-to-noise ratio. However, in grand-canonical systems, the steepness of the two-point function is not given by the energy difference $E_1 - E_0$ alone. As we can see in our exact solution in Eq. (5.58), even for $\lambda = 0$, we obtain non-vanishing exponents by increasing $|\beta\mu|$. On the other hand, however, we also find that for every value of λ, there is a chemical potential $\beta\mu \leq 0$ at which the exponent vanishes. That means for every value of λ there is a critical chemical potential around which we can only simulate the two-point function at increased computational effort. This critical chemical potential moves to smaller values as λ increases.

These proof-of-concept calculations for $0+1$-dimensional field theories show that the pairing-field formalism and our simulation of it are capable of producing correlation functions that agree well with exact solutions. With these results, our simulation should be able to calculate correlation functions, including spatial coordinates in higher dimensions, and allow us to study pairing and the formation of a superfluid ground state in the phase diagram of ultracold Fermi gases.

5.3 Sign Problem

Our simulation of the system exhibits a sign problem, and while the CL approach is unhindered by it, it may be of interest to study the sign problem quantitatively. The source of the sign problem in our calculations lies in the pairing-field configurations in the fermion matrix $\mathcal{M}$, found in Eq. (3.244). Even without spin- or mass imbalance, as soon as we evaluate the determinant of the fermion matrix for non-uniform pairing-field configurations, we generally end up with

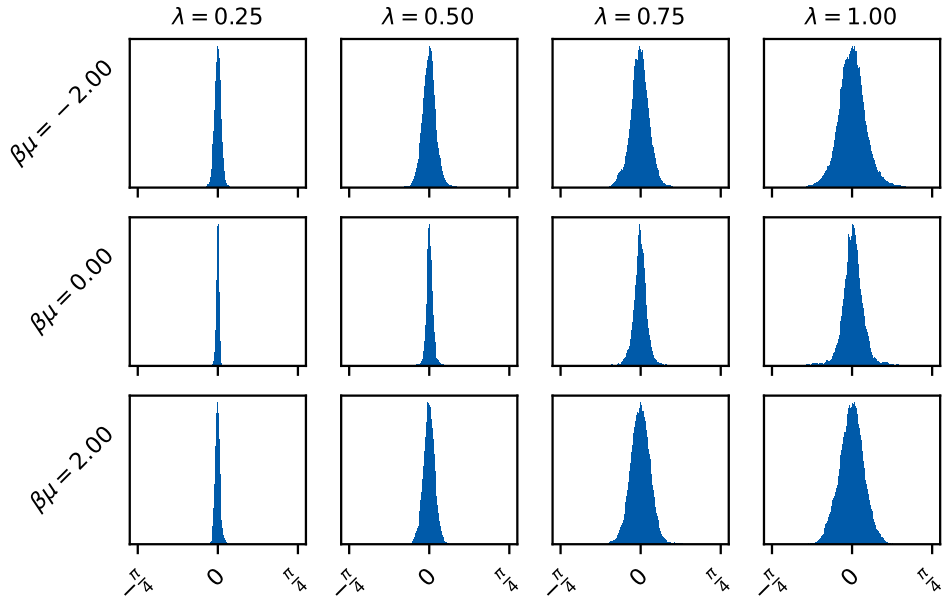


Figure 5.10: Distribution histograms of the complex path integral weights e^{-S_B} in the Complex Langevin sampling process. Sample points of the weights are recorded at every step in Langevin time, and the Langevin time spacing is $\delta t_{CL} = 0.05$. The process was calculated up to a final Langevin time of 15 000, and the first 5 % of sample points have been discarded as "warm up". The lattice size for each pair of chemical potential and interaction parameters is determined by the above-mentioned criteria with $C_I = 100$, resulting in lattices up to $N_\tau = 100$ for $\lambda = 1.0$.

complex fermion determinants. This results in complex weights e^{-S_B} of the path integral for all interacting systems. Figure 5.10 shows distributions histograms for the complex argument of path integral weights e^{-S_B} for a selection of chemical potentials $\beta\mu$ and interaction parameters λ. We generally observe narrow distributions around an argument of zero. That means that the weights generally lie close to the real axis. As such, the sign problem in our simulation can be characterized as a *weak phase problem*. The width of the distributions in Fig. 5.10, and, as such, the severity of the phase problem, increases with increasing values of λ. This comes as no surprise since the phase problem originates from the pairing-field configurations entering the fermion matrix, where they are scaled with the interaction coupling. To study this trend more quantitatively, we determine the width of the argument distributions by calculating the standard distributions of the samples $\{\arg e^{-S_B}\}$. We then fit these widths to the model

$$f(\lambda) = c_1 \cdot \lambda^{c_2} \,. \tag{5.63}$$

The results of these fits can be seen in Fig. 5.11. We choose this specific model because we know that our calculations are free of a phase problem in the non-interacting case. The model reflects that behavior as $f(\lambda = 0) = 0$.

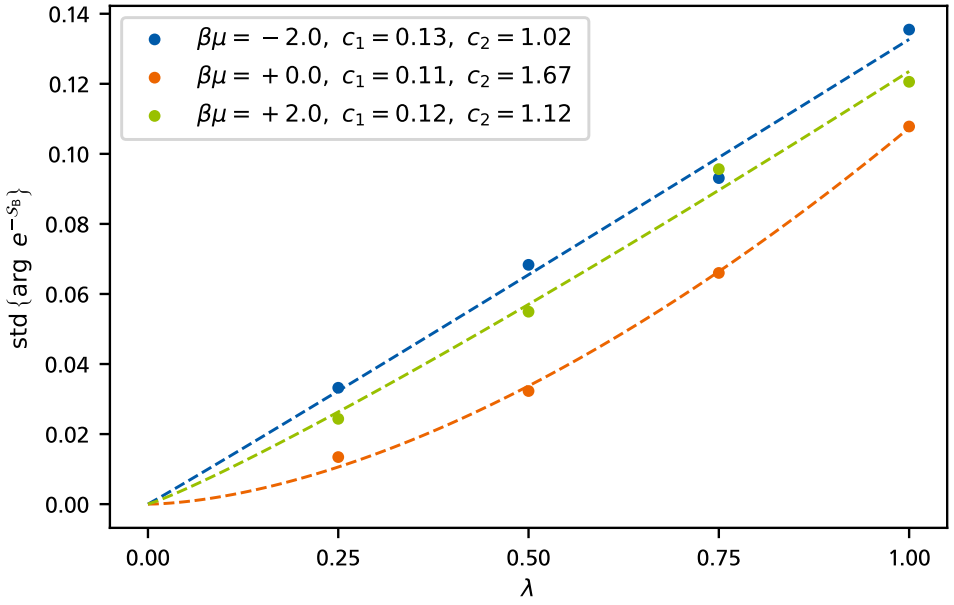


Figure 5.11: Sample standard deviation of the samples shown in Fig. 5.10. The dashed lines show fits of the data points to the model in Eq. (5.63), and the determined fit parameters are shown in the legend. The width of the argument distributions increases with increasing interaction parameters λ and the exponent of the growth depends on the chemical potential $\beta\mu$.

6 Conclusion and Outlook

In this work, we set out to construct a formalism and simulation that is suited for the study of the phase diagram of two-component Fermi gases using Monte-Carlo methods in an ab-initio fashion in the future. To this end, we chose to bosonize the underlying fermionic model with the aid of a Hubbard-Stratonovich transformation that reformulates the system in terms of the pairing field. There are plenty of other choices for a Hubbard-Stratonovich transformation but since the details of the pairing mechanism determine the behavior of the system at low temperatures, the pairing field seems like the most natural approach, especially when considering its correspondence to the superfluid order parameter. To circumvent the sign problem that plagues many Monte-Carlo methods in this particular problem, we use the Complex Langevin method to implement a simulation of this system. The main goal of this thesis is to develop the pairing-field formalism and develop a software package that allows for calculations of density equations of state and correlation functions. We have indeed achieved this goal and demonstrated the application of our novel approach by computing density equations of state and two-point functions of the pairing field.

Before we summarize our findings, we quickly recapitulate the central aspects of our work. In Chapter 3, we derived the pairing field formalism for the simulation. At first, we bosonized the system in the continuum and used the resulting theory to derive relations between the pairing field and the fermionic fields of the original theory. We then used the continuum theory to perform a mean-field study of the system that we later used as a reference for calculated density equations of state. In Sec. 3.3, we started to rigorously derive the formalism on the lattice, beginning with the fermionic Hamiltonian. We constructed a basis of coherent states and used it to derive the path integral of the fermionic lattice theory. At this point, we modified the resulting discretized temporal derivative to restore the Silver-Blaze symmetry, which is important for the calculation of densities. To improve the general handling of the theory and, more importantly, its numerical properties in the simulation, we rescaled all constants and fields to be dimensionless. With the dimensionless lattice theory, we introduced field configuration vectors and coordinate shift matrices to represent the action of the theory more compactly. In this compact dimensionless form of the theory, we performed the lattice equivalent of the pairing field transformation, taking care to include all subtleties that we discovered in the rigorous discretization of the theory. After that, we finally obtained the bosonized lattice theory that serves as the foundation of our simulation. In Chapter 4, we used the lattice theory to derive the Langevin equation for the simulation by determining the drift terms. Furthermore, we determined the expressions we need to sample observables of the system. Additionally, we discussed aspects of software development to determine the tools and programming languages we used to create the simulation. In Chapter 5, we finally focused on the simulation of $0 + 1$-dimensional systems. As a proof-of-concept, we first focused on density equations of state. We calculated an exact solution for the density, which we compared to the results of the simulation for different dimensionless couplings. After that, we started an exploratory study of two-point correlation functions. Again, we started by determining the exact solution and compared it to the results of our simulation. In both cases,

we find excellent agreement, which demonstrates the power of our approach.

Now, let us summarize the main findings of this work. In the derivation of the formalism, we discovered that the lattice theory is not just an arbitrary discretization of the continuum theory. The rigorous derivation of the lattice path integral imposes rules for the temporal derivative that also influence the details of the lattice pairing field transformation. We found a highly concise notation for the lattice theory that allowed us to leverage matrix properties to speed our derivations up considerably.[1] Beyond that, in the construction of lattice derivative operators, the matrix notation allows us to offload the complicated and error-prone lattice-ordering aspect completely into the construction of the coordinate shift matrices. With the coordinate shift matrices at hand, the process of constructing lattice derivative operators from finite differences is almost trivial. Furthermore, the use of coordinate shift matrices makes the theory more robust against errors in derivations, because the most error-prone part can be thoroughly tested in isolation from the rest of the lattice theory. We were able to create a Complex Langevin simulation that successfully solves systems in $0 + 1$ dimensions in an ab-initio fashion for attractive interactions. Through careful application of principles of modern software development, the simulation is modular and robust. It features an extensive set of unit tests that make it easy to modify even integral parts of the simulation without the risk of breaking existing functionality. In addition, the modularity of our simulation makes it easy to extend, as new observables can be added completely without disrupting the existing simulation. Our simulation produces density equations of states that are in excellent agreement with exact solutions. We emphasize that the mean-field approximation fully fails at this point. Because of the qualitative similarities between density equations of state in $0 + 1$-dimensional and $1 + 1$-dimensional systems, we are confident that our simulation will produce similarly good results in future studies of $1 + 1$-dimensional systems. In addition to density equations of state, we have demonstrated that our simulation is capable of calculating correlation functions of the pairing field that will be instrumental in future studies of the phase structure of two-component Fermi gases. Our simulation results for the two-point function of the system agree well with the available exact solutions. Additionally, we found that the pairing field formalism is capable of producing precise results for correlation functions of the pairing field on smaller lattices than it requires to produce precise results for densities. Therefore, it indeed seems to be an efficient approach to simulating pairing-related quantities. Using analytic studies of the properties of correlation functions, we identified ranges of physical parameters that lead to favorable numerical conditions in the simulation. With this insight gained, the simulation is ready to tackle correlation functions in higher spatial dimensions.

In conclusion, our novel lattice pairing field formalism is successful in simulating two- component Fermi gases in $0 + 1$ dimensions and looks like a promising approach to studying the phase structure of the system in $d \neq 0$ dimensions. In particular, it could be an effective way to access inhomogeneous phases in spin-polarized systems.

The next step for the pairing field formalism is to study systems in $1 + 1$ dimensions. For proof-of-concept calculations of density equations of state, we can use the numerical results in

[1] We want to share an anecdote to illustrate that fact. The lattice theory presented in this thesis is actually the third incarnation of the pairing field theory. For the first lattice theory, we naively discretized an action we bosonized in the continuum, as it is often done in the literature. Getting that theory to the point where we can simulate it took us over a year. Eventually, we figured out that the naive discretization produced incorrect results. Therefore, we derived the theory again, this time starting from the Hamiltonian. We had already done a lot of the preparations for the lattice, therefore, deriving the second incarnation only took us about five months. We discovered that there was still a problem in the discretized pairing field, so we had to derive the formalism for a third time. However, at that point, we had discovered how to use coordinate shift matrices to formulate lattice theories. With them, we could derive the third theory in twenty minutes on two pages of paper! Because their usefulness translates directly to the implementation of the simulation, even implementing this new formalism in the simulation took no more than an hour.

Ref. [12]. We note that further improvements to our software package may be worthwhile to implement. For example, adaptive steps for the Langevin solver, as discussed in Sec. 2.3.1, present an improvement that can not only increase the performance of our simulation but potentially also make it more robust against certain classes of numerical problems that may occur. Beyond $d = 0$, the lattices of our simulation may become so large that the direct inversion of the fermion matrix in the calculation of the drift becomes unviable. In that case, we can implement the scheme discussed in Sec. 2.3.2 to approximate the drift stochastically. Regardless, already in its present version our novel approach is ready to move towards higher dimensions until in $d = 3$ dimensions, the correspondence of the pairing field to the superfluid order parameter allows us to study the spontaneous breaking of the $U(1)$ symmetry.

A Manipulating Field Bilinears

Rewriting the action in Eq. (3.8) in terms of Nambu-Gorkov spinors requires flipping the order of fermion fields around a central operator in a bilinear expression. To illustrate how this is achieved, let us consider the expression

$$A = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r \; \psi^*(\tau, \mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m} - \mu \right) \psi(\tau, \mathbf{r}) \tag{A.1}$$

with the Grassmann-valued fields ψ^* and ψ. Our goal is to reverse the order of the fields and obtain a new bilinear representation of A of the form

$$A = \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r \; \psi(\tau, \mathbf{r}) \, (\quad ? \quad) \, \psi^*(\tau, \mathbf{r}) \tag{A.2}$$

and the unknown operator in its center. What initially prevents us from just swapping the fields around is the fact, that we have a differential operator in the center that acts on the right-hand field. Thus, a reasonable first step is to Fourier transform the fields and replace the differential operators by scalars so that the fields can be moved around the center:

$$\begin{aligned} A &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r \; \psi^*(\tau, \mathbf{r}) \left(\partial_\tau - \frac{\nabla^2}{2m} - \mu \right) \psi(\tau, \mathbf{r}) \\ &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r \; \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3} \; \frac{1}{\beta} \sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3} \cdot \\ &\qquad e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}} \, \tilde{\psi}^*(\omega_n, \mathbf{p}) \left(\partial_\tau - \frac{\nabla^2}{2m} - \mu \right) \tilde{\psi}(\omega_{n'}, \mathbf{p}') \; e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \end{aligned} \tag{A.3}$$

$$\begin{aligned} &= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r \; \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3} \; \frac{1}{\beta} \sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3} \cdot \\ &\qquad e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}} \, \tilde{\psi}^*(\omega_n, \mathbf{p}) \left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m} - \mu \right) \tilde{\psi}(\omega_{n'}, \mathbf{p}') \; e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \end{aligned}$$

At that point the central expression only contains scalars and we can move the fields past it:

$$\begin{aligned}
&\text{[Continuation of Eq. (A.3)]}\\
&= -\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r\, \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \cdot \\
&\qquad e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}}\, \tilde{\psi}(\omega_{n'}, \mathbf{p}')\left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m} - \mu\right) \tilde{\psi}^*(\omega_n, \mathbf{p})\, e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \\
&= -\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r\, \frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \cdot \\
&\qquad e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}} e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \quad \tilde{\psi}(\omega_{n'}, \mathbf{p}')\left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m} - \mu\right) \tilde{\psi}^*(\omega_n, \mathbf{p})\, .
\end{aligned} \tag{A.4}$$

The fields have now switched order, so what we would like to do is reverse the Fourier transform to go back to the position space fields. For this we need to regain the differential operators in the center operator. This, however, requires us to rewrite the center operator in terms of ω and $\mathbf{p}$, the arguments of the field that is now on the right-hand side. This can be achieved by using the time and space integrations together with the exponential functions as deltas via the relations

$$\begin{aligned}
\int_0^\beta \mathrm{d}\tau\, e^{-i(\omega_{n'}-\omega_n)\tau} &= \beta\delta_{n',n} \quad \text{and} \\
\int \mathrm{d}^3 r\, e^{-i(\mathbf{p}'-\mathbf{p})\mathbf{r}} &= (2\pi)^3 \delta^{(3)}(\mathbf{p}'-\mathbf{p})\, .
\end{aligned} \tag{A.5}$$

We continue:

$$\begin{aligned}
A = &-\frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \beta\delta_{n',n}(2\pi)^3\delta^{(3)}(\mathbf{p}'-\mathbf{p})\, \cdot \\
&\tilde{\psi}(\omega_{n'}, \mathbf{p}')\left(-i\omega_{n'} + \frac{\mathbf{p}'^2}{2m} - \mu\right) \tilde{\psi}^*(\omega_n, \mathbf{p}) \\
= &-\frac{1}{\beta}\sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta}\sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \beta\delta_{n',n}(2\pi)^3\delta^{(3)}(\mathbf{p}'-\mathbf{p})\, \cdot \\
&\tilde{\psi}(\omega_{n'}, \mathbf{p}')\left(-i\omega_n + \frac{\mathbf{p}^2}{2m} - \mu\right) \tilde{\psi}^*(\omega_n, \mathbf{p})\, .
\end{aligned} \tag{A.6}$$

At this point, frequency and momentum in the center operator correspond to the arguments of the right-hand field, so we can go back to position space. To do this we rewrite the deltas as exponential functions[1], absorb the leading sign into the center operator, reconstruct the differential

[1] You might think that you can move the sign in the exponential functions between $e^{i\omega_n\tau}$ and $e^{-i\omega_{n'}\tau}$ to change the sign of the time derivative in the new center operator. While it is true, that you can move the sign due to the symmetry of the Kronecker delta, you cannot perform the correct reverse Fourier transform if you do so and, thus, the sign of the time derivative cannot be changed in the resulting transformed central operator.

operators and perform the reverse Fourier transform:

$$
\begin{aligned}
A = -\int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r\, \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta} \sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \cdot \\
e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \tilde{\psi}(\omega_{n'}, \mathbf{p}') \left(-i\omega_n + \frac{\mathbf{p}^2}{2m} - \mu \right) e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}} \tilde{\psi}^*(\omega_n, \mathbf{p}) \\
= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r\, \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta} \sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \cdot \\
e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \tilde{\psi}(\omega_{n'}, \mathbf{p}') \left(i\omega_n - \frac{\mathbf{p}^2}{2m} + \mu \right) e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}} \tilde{\psi}^*(\omega_n, \mathbf{p}) \\
= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r\, \frac{1}{\beta} \sum_n \int \frac{\mathrm{d}^3 p}{(2\pi)^3}\, \frac{1}{\beta} \sum_{n'} \int \frac{\mathrm{d}^3 p'}{(2\pi)^3}\, \cdot \\
e^{-i\omega_{n'}\tau} e^{-i\mathbf{p}'\mathbf{r}} \tilde{\psi}(\omega_{n'}, \mathbf{p}') \left(\partial_\tau + \frac{\nabla^2}{2m} + \mu \right) e^{i\omega_n \tau} e^{i\mathbf{p}\mathbf{r}} \tilde{\psi}^*(\omega_n, \mathbf{p}) \\
= \int_0^\beta \mathrm{d}\tau \int \mathrm{d}^3 r\, \psi(\tau, \mathbf{r}) \left(\partial_\tau + \frac{\nabla^2}{2m} + \mu \right) \psi^*(\tau, \mathbf{r})\, .
\end{aligned}
\tag{A.7}
$$

Thus, we obtained the desired representation of the field bilinear A with the previously unknown center operator $\left(\partial_\tau + \frac{\nabla^2}{2m} + \mu\right)$.

B Summation by Parts

Summation by parts is what we call a lattice equivalent to integration by parts in the continuum. The priciple idea behind this technique is to interpret a sum over lattice sites as an integral and shift the summation indices in an appropriate manner. This creates boundary terms of field values outside of the lattice, that we reconcile with the sum by utilizing the boundary conditions of the field. In this section we want to demonstrate this technique for a temporal derivative of both bosonic and fermionic fields.

In the bosonic case we consider the bosonic field values $\{\phi_i^*\}$ and $\{\phi_i\}$ for $i \in \{1, ..., N\}$ with the periodic boundary conditions

$$\phi_{N+1}^* = \phi_1^* \quad \text{and} \quad \phi_{N+1} = \phi_1 \,. \tag{B.1}$$

These field values represent the field configurations of a $0+1$-dimensional theory with N temporal lattice sites. A common problem in these theories is essentially the same as the continuum problem we tackle in App. A: In the derivation of the theory we can find expressions like

$$\sum_{i=1}^{N} (\phi_i^* - \phi_{i+1}^*)\, \phi_i \,, \tag{B.2}$$

which is a discretization of

$$\int \mathrm{d}\tau \;\; (-\partial_\tau \phi^*)\, \phi \,. \tag{B.3}$$

However, we would much rather have an expression like

$$\int \mathrm{d}\tau \; \phi^* \, (?) \, \phi \tag{B.4}$$

that has an operator acting on ϕ to the right of ϕ^*. In the continuum we solve this problem through the means of integration by parts and on the lattice, the "summation by parts" is performed as follows:

$$\begin{aligned} \sum_{i=1}^{N} (\phi_i^* - \phi_{i+1}^*)\, \phi_i &= \sum_{i=1}^{N} \phi_i^* \phi_i - \sum_{i=1}^{N} \phi_{i+1}^* \phi_i \\ &= \sum_{i=1}^{N} \phi_i^* \phi_i - \sum_{i=2}^{N+1} \phi_i^* \phi_{i-1} \\ &= \sum_{i=1}^{N} \phi_i^* \phi_i - \sum_{i=2}^{N} \phi_i^* \phi_{i-1} - \phi_{N+1}^* \phi_N \,. \end{aligned} \tag{B.5}$$

In this expression, the rightmost term is similar to the boundary terms in continuous integration by parts. We can use the boundary conditions of the field in Eq. (B.1) to reabsorb this term in

the incomplete lattice time summation:

$$\begin{aligned}
&\text{[Continuation of Eq. (B.5)]}\\
&= \sum_{i=1}^{N} \phi_i^* \phi_i - \sum_{i=2}^{N} \phi_i^* \phi_{i-1} - \phi_1^* \phi_0\\
&= \sum_{i=1}^{N} \phi_i^* \phi_i - \sum_{i=1}^{N} \phi_i^* \phi_{i-1}\\
&= \sum_{i=1}^{N} \phi_i^* \left(\phi_i - \phi_{i-1}\right) .
\end{aligned} \tag{B.6}$$

The resulting expression we found here corresponds to the continuum expression

$$\int \mathrm{d}\tau \; \phi^* \partial_\tau \phi \,, \tag{B.7}$$

so we succeeded in finding the desired form of the expression.

For the fermionic case we consider the discrete fermionic, i.e. Grassmann-valued fields $\{\psi_i^*\}$ and $\{\psi_i\}$ for $i \in \{1, \dots, N\}$ with the, now anti-periodic, boundary conditions

$$\psi_{N+1}^* = -\psi^* 1 \quad \text{and} \quad \psi_{N+1} = -\psi_1 \,. \tag{B.8}$$

Again, we consider the expression

$$\sum_{i=1}^{N} \left(\psi_i^* - \psi_{i+1}^*\right) \psi_i \,. \tag{B.9}$$

In the "summation by parts" of the fermionic expression we begin the same way as for bosonic fields:

$$\begin{aligned}
\sum_{i=1}^{N} \left(\psi_i^* - \psi_{i+1}^*\right) \psi_i &= \sum_{i=1}^{N} \psi_i^* \psi_i - \sum_{i=1}^{N} \psi_{i+1}^* \psi_i\\
&= \sum_{i=1}^{N} \psi_i^* \psi_i - \sum_{i=2}^{N+1} \psi_i^* \psi_{i-1}\\
&= \sum_{i=1}^{N} \psi_i^* \psi_i - \sum_{i=2}^{N} \psi_i^* \psi_{i-1} - \psi_{N+1}^* \psi_N \,,
\end{aligned} \tag{B.10}$$

and use the fermionic boundary conditions to reabsorb the boundary term in the incomplete lattice time summation:

$$\begin{aligned}
&\text{[Continuation of Eq. (B.10)]}\\
&= \sum_{i=1}^{N} \psi_i^* \psi_i - \sum_{i=2}^{N} \psi_i^* \psi_{i-1} - (-\psi_1^*)(-\psi_0)\\
&= \sum_{i=1}^{N} \psi_i^* \psi_i - \sum_{i=1}^{N} \psi_i^* \psi_{i-1}\\
&= \sum_{i=1}^{N} \psi_i^* \left(\psi_i - \psi_{i-1}\right) .
\end{aligned} \tag{B.11}$$

Because sums like this always contain products of two fermionic fields, we apply the boundary conditions twice and the emerging signs cancel. This leaves us with the same behavior as in the bosonic case.

Bibliography

[1] C. H. Schunck, Y. Shin, A. Schirotzek, M. W. Zwierlein, and W. Ketterle. *"Pairing Without Superfluidity: The Ground State of an Imbalanced Fermi Mixture"*. Science **316**, 867–870 (2007).

[2] Y.-i. Shin, C. H. Schunck, A. Schirotzek, and W. Ketterle. *"Phase Diagram of a Two-Component Fermi Gas with Resonant Interactions"*. Nature **451**(7179), 689–693 (2008).

[3] S. Nascimbène, N. Navon, K. J. Jiang, F. Chevy, and C. Salomon. *"Exploring the Thermodynamics of a Universal Fermi Gas"*. Nature **463**(7284), 1057–1060 (2010).

[4] S. Nascimbène, N. Navon, F. Chevy, and C. Salomon. *"The Equation of State of Ultracold Bose and Fermi Gases: A Few Examples"*. New Journal of Physics **12**, 103026 (2010).

[5] A. Sommer, M. Ku, G. Roati, and M. W. Zwierlein. *"Universal Spin Transport in a Strongly Interacting Fermi Gas"*. Nature **472**(7342), 201–204 (2011).

[6] S. Nascimbène, N. Navon, S. Pilati, F. Chevy, S. Giorgini, A. Georges, and C. Salomon. *"Fermi-Liquid Behavior of the Normal Phase of a Strongly Interacting Gas of Cold Atoms"*. Physical Review Letters **106**, 215303 (2011).

[7] M. J. H. Ku, A. T. Sommer, L. W. Cheuk, and M. W. Zwierlein. *"Revealing the Superfluid Lambda Transition in the Universal Thermodynamics of a Unitary Fermi Gas"*. Science **335**, 563–567 (2012).

[8] Y. Sagi, T. E. Drake, R. Paudel, R. Chapurin, and D. S. Jin. *"Breakdown of the Fermi Liquid Description for Strongly Interacting Fermions"*. Physical Review Letters **114**, 075301 (2015).

[9] M. Horikoshi, M. Koashi, H. Tajima, Y. Ohashi, and M. Kuwata-Gonokami. *"Ground-State Thermodynamic Quantities of Homogeneous Spin-1/2 Fermions from the BCS Region to the Unitarity Limit"*. Physical Review X **7**, 041004 (2017).

[10] P. A. Murthy, M. Neidig, R. Klemt, L. Bayha, I. Boettcher, T. Enss, M. Holten, G. Zürn, P. M. Preiss, and S. Jochim. *"High-Temperature Pairing in a Strongly Interacting Two-Dimensional Fermi Gas"*. Science **359**, 452–455 (2018).

[11] J. E. Drut and A. N. Nicholson. *"Lattice Methods for Strongly Interacting Many-Body Systems"*. Journal of Physics G: Nuclear and Particle Physics **40**, 043101 (2013).

[12] M. D. Hoffman, P. D. Javernick, A. C. Loheac, W. J. Porter, E. R. Anderson, and J. E. Drut. *"Universality in One-Dimensional Fermions at Finite Temperature: Density, Pressure, Compressibility, and Contact"*. Physical Review A **91**, 033618 (2015).

[13] L. Rammelmüller, J. E. Drut, and J. Braun. *"Pairing Patterns in One-Dimensional Spin- and Mass-Imbalanced Fermi Gases"*. SciPost Physics **9**, 014 (2020).

[14] F. Ehmann, J. E. Drut, and J. Braun. *"A Lattice Pairing-Field Approach to Ultracold Fermi Gases"*. arxiv:2212.12298.

[15] S. L. Campbell, R. B. Hutson, G. E. Marti, A. Goban, N. Darkwah Oppong, R. L. McNally, L. Sonderhouse, J. M. Robinson, W. Zhang, B. J. Bloom, and J. Ye. *"A Fermi-degenerate Three-Dimensional Optical Lattice Clock"*. Science **358**, 90–94 (2017).

[16] G. Modugno, M. Modugno, F. Riboli, G. Roati, and M. Inguscio. *"Two Atomic Species Superfluid"*. Physical Review Letters **89**, 190404 (2002).

[17] A. G. Truscott, K. E. Strecker, W. I. McAlexander, G. B. Partridge, and R. G. Hulet. *"Observation of Fermi Pressure in a Gas of Trapped Atoms"*. Science **291**, 2570–2572 (2001).

[18] F. Schreck, L. Khaykovich, K. L. Corwin, G. Ferrari, T. Bourdel, J. Cubizolles, and C. Salomon. *"Quasipure Bose-Einstein Condensate Immersed in a Fermi Sea"*. Physical Review Letters **87**, 080403 (2001).

[19] Z. Hadzibabic, C. A. Stan, K. Dieckmann, S. Gupta, M. W. Zwierlein, A. Görlitz, and W. Ketterle. *"Two-Species Mixture of Quantum Degenerate Bose and Fermi Gases"*. Physical Review Letters **88**, 160401 (2002).

[20] W. Zwerger. *"The BCS-BEC Crossover and the Unitary Fermi Gas"*. Berlin, Heidelberg: Springer (2012).

[21] A. Bohr, B. R. Mottelson, and D. Pines. *"Possible Analogy between the Excitation Spectra of Nuclei and Those of the Superconducting Metallic State"*. Physical Review **110**, 936–938 (1958).

[22] A. B. Migdal. *"Superfluidity and the Moments of Inertia of Nuclei"*. Nuclear Physics **13**, 655–674 (1959).

[23] M. G. Alford, A. Schmitt, K. Rajagopal, and T. Schäfer. *"Color Superconductivity in Dense Quark Matter"*. Reviews of Modern Physics **80**, 1455–1515 (2008).

[24] C. A. R. Sá de Melo, M. Randeria, and J. R. Engelbrecht. *"Crossover from BCS to Bose Superconductivity: Transition Temperature and Time-Dependent Ginzburg-Landau Theory"*. Physical Review Letters **71**, 3202–3205 (1993).

[25] S. Huth, P. T. H. Pang, I. Tews, T. Dietrich, A. Le Fèvre, A. Schwenk, W. Trautmann, K. Agarwal, M. Bulla, M. W. Coughlin, and C. Van Den Broeck. *"Constraining Neutron-Star Matter with Microscopic and Macroscopic Collisions"*. Nature **606**(7913), 276–280 (2022).

[26] M. Randeria, J.-M. Duan, and L.-Y. Shieh. *"Superconductivity in a Two-Dimensional Fermi Gas: Evolution from Cooper Pairing to Bose Condensation"*. Physical Review B **41**, 327–343 (1990).

[27] M. Drechsler and W. Zwerger. *"Crossover from BCS-superconductivity to Bose-condensation"*. Annalen der Physik **504**, 15–23 (1992).

[28] M. Randeria, J.-M. Duan, and L.-Y. Shieh. *"Bound States, Cooper Pairing, and Bose Condensation in Two Dimensions"*. Physical Review Letters **62**, 981–984 (1989).

[29] H. Feshbach. *"Unified Theory of Nuclear Reactions"*. Annals of Physics **5**, 357–390 (1958).

[30] C. Chin, R. Grimm, P. Julienne, and E. Tiesinga. *"Feshbach Resonances in Ultracold Gases"*. Reviews of Modern Physics **82**, 1225–1286 (2010).

[31] J. Taron. *"Feshbach Resonance: A One Dimensional Example"*. American Journal of Physics **81**, 603–609 (2013).

[32] M. Lu, N. Q. Burdick, and B. L. Lev. *"Quantum Degenerate Dipolar Fermi Gas"*. Physical Review Letters **108**, 215301 (2012).

[33] E. Neri, A. Ciamei, C. Simonelli, I. Goti, M. Inguscio, A. Trenkwalder, and M. Zaccanti. *"Realization of a Cold Mixture of Fermionic Chromium and Lithium Atoms"*. Physical Review A **101**, 063602 (2020).

[34] P. F. Bedaque, H. Caldas, and G. Rupak. *"Phase Separation in Asymmetrical Fermion Superfluids"*. Physical Review Letters **91**, 247002 (2003).

[35] P. Fulde and R. A. Ferrell. *"Superconductivity in a Strong Spin-Exchange Field"*. Physical Review **135**(3A), A550–A563 (1964).

[36] A. I. Larkin and I. U. N. Ovchinnikov. *"Inhomogeneous State of Superconductors"*. Soviet Physics-JETP **20**, 762–769 (1965).

[37] X.-W. Guan, M. T. Batchelor, and C. Lee. *"Fermi Gases in One Dimension: From Bethe Ansatz to Experiments"*. Reviews of Modern Physics **85**, 1633–1691 (2013).

[38] I. Bloch. *"Ultracold Quantum Gases in Optical Lattices"*. Nature Physics **1**(1), 23–30 (2005).

[39] M. E. J. Newman. *"Computational Physics"*. Michigan (2013).

[40] G. Parisi and Y.-S. Wu. *"Perturbation Theory without Gauge Fixing"*. Scientia Sinica **24**, 35 (1981).

[41] P. H. Damgaard and H. Hüffel. *"Stochastic Quantization"*. Physics Reports **152**, 227–398 (1987).

[42] P. Langevin. *"Sur la théorie du mouvement brownien"*. Comptes rendus de l'Académie des Sciences **146**, 530–533 (1908).

[43] D. S. Lemons and A. Gythiel. *"Paul Langevin's 1908 Paper "On the Theory of Brownian Motion" ["Sur La Théorie Du Mouvement Brownien," C. R. Acad. Sci. (Paris) 146, 530–533 (1908)]"*. American Journal of Physics **65**, 1079–1081 (1997).

[44] C. E. Berger, L. Rammelmüller, A. C. Loheac, F. Ehmann, J. Braun, and J. E. Drut. *"Complex Langevin and Other Approaches to the Sign Problem in Quantum Many-Body Physics"*. Physics Reports , (2020).

[45] V. Ambegaokar and M. Troyer. *"Estimating Errors Reliably in Monte Carlo Simulations of the Ehrenfest Model"*. American Journal of Physics **78**, 150–157 (2010).

[46] R. G. Miller. *"The Jackknife-a Review"*. Biometrika **61**, 1–15 (1974).

[47] B. Efron. *"Nonparametric Estimates of Standard Error: The Jackknife, the Bootstrap and Other Methods"*. Biometrika **68**, 589–599 (1981).

[48] G. Aarts, E. Seiler, and I.-O. Stamatescu. *"Complex Langevin Method: When Can It Be Trusted?"*. Physical Review D **81**, 054508 (2010).

[49] G. Aarts, F. A. James, E. Seiler, and I.-O. Stamatescu. *"Adaptive Stepsize and Instabilities in Complex Langevin Dynamics"*. Physics Letters B **687**, 154–159 (2010).

[50] C. Gattringer and C. B. Lang. *"Quantum Chromodynamics on the Lattice"*. Berlin, Heidelberg: Springer Berlin Heidelberg (2010).

[51] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *"Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods"*. Society for Industrial and Applied Mathematics (1994).

[52] B. DeMarco and D. S. Jin. *"Onset of Fermi Degeneracy in a Trapped Atomic Gas"*. Science **285**, 1703–1706 (1999).

[53] A. Frisch, K. Aikawa, M. Mark, F. Ferlaino, E. Berseneva, and S. Kotochigova. *"Hyperfine Structure of Laser-Cooling Transitions in Fermionic Erbium-167"*. Physical Review A **88**, 032508 (2013).

[54] A. Altland and B. D. Simons. *"Condensed Matter Field Theory"*. Cambridge University Press (2010).

[55] J. Hubbard. *"Calculation of Partition Functions"*. Phys. Rev. Lett. **3**, 77–78 (1959).

[56] J. Braun, J.-W. Chen, J. Deng, J. E. Drut, B. Friman, C.-T. Ma, and Y.-D. Tsai. *"Imaginary Polarization as a Way to Surmount the Sign Problem in Ab Initio Calculations of Spin-Imbalanced Fermi Gases"*. Physical Review Letters **110**, 130404 (2013).

[57] T. D. Cohen. *"Functional Integrals for QCD at Nonzero Chemical Potential and Zero Density"*. Physical Review Letters **91**, 222001 (2003).

[58] G. Markó, U. Reinosa, and Z. Szép. *"Bose-Einstein Condensation and Silver Blaze Property from the Two-Loop Φ-Derivable Approximation"*. Physical Review D **90**, 125021 (2014).

[59] N. Khan, J. M. Pawlowski, F. Rennecke, and M. M. Scherer. *The Phase Diagram of QC2D from Functional Methods*. Dec. 11, 2015. URL: `http://arxiv.org/abs/1512.03673` (visited on 02/28/2023). preprint.

[60] W.-j. Fu and J. M. Pawlowski. *"Relevance of Matter and Glue Dynamics for Baryon Number Fluctuations"*. Physical Review D **92**, 116006 (2015).

[61] J. Braun, M. Leonhardt, and M. Pospiech. *"Fierz-Complete NJL Model Study: Fixed Points and Phase Structure at Finite Temperature and Density"*. Physical Review D **96**, 076003 (2017).

[62] J. Braun, T. Dörnfeld, B. Schallmo, and S. Töpfel. *"Renormalization Group Studies of Dense Relativistic Systems"*. Physical Review D **104**, 096002 (2021).

[63] P. Hasenfratz and F. Karsch. *"Chemical Potential on the Lattice"*. Physics Letters B **125**, 308–310 (1983).

[64] R. C. Martin. *"Clean Code"*. Philadelphia, PA: Prentice Hall (2008).

[65] G. Van Rossum and F. L. Drake. *"Python 3 Reference Manual"*. Scotts Valley, CA: CreateSpace (2009).

[66] J. D. Hunter. *"Matplotlib: A 2D Graphics Environment"*. Computing in Science & Engineering **9**, 90–95 (2007).

[67] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. *"Array Programming with NumPy"*. Nature **585**, 357–362 (2020).

[68] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by S. van der Walt and Jarrod Millman. 2010, pp. 56–61.

[69] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, . Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. *"SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python"*. Nature Methods **17**, 261–272 (2020).

[70] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. "Jupyter Notebooks – a Publishing Format for Reproducible Computational Workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87–90.

[71] A. Meurer, C. P. Smith, M. Paprocki, O. Certík, S. B. Kirpichev, M. Rocklin, Am. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, Š. Roucka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz. *"SymPy: Symbolic Computing in Python"*. PeerJ Computer Science **3**, e103 (2017).

[72] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. *"Julia: A Fresh Approach to Numerical Computing"*. SIAM Review **59**, 65–98 (2017).

[73] *GSL - GNU Scientific Library - GNU Project - Free Software Foundation*. URL: `https://www.gnu.org/software/gsl/` (visited on 03/21/2023).

[74] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. *"Basic Linear Algebra Subprograms for Fortran Usage"*. ACM Transactions on Mathematical Software **5**, 308–323 (1979).

[75] *LAPACK — Linear Algebra PACKage*. URL: `https://netlib.org/lapack/` (visited on 03/21/2023).

[76] *OpenBLAS : An Optimized BLAS Library*. URL: `https://www.openblas.net/` (visited on 03/21/2023).

[77] *PEP 20 – The Zen of Python*. URL: `https://peps.python.org/pep-0020/` (visited on 03/21/2023).

[78] *Julia Micro-Benchmarks*. URL: `https://julialang.org/benchmarks/` (visited on 03/21/2023).

[79] *Podman*. URL: `https://podman.io/` (visited on 03/23/2023).

[80] *Docker: Accelerated, Containerized Application Development*. May 10, 2022. URL: `https://www.docker.com/` (visited on 03/23/2023).

[81] *CFFI Documentation — CFFI 1.15.1 Documentation*. URL: `https://cffi.readthedocs.io/en/latest/index.html` (visited on 03/23/2023).

[82] *The Jargon File*. URL: `http://catb.org/jargon/html/index.html` (visited on 03/23/2023).

[83] A. Bulgac, J. E. Drut, and P. Magierski. *"Spin 1 / 2 Fermions in the Unitary Regime: A Superfluid of a New Type"*. Physical Review Letters **96**, 090404 (2006).

[84] N. D. Mermin and H. Wagner. *"Absence of Ferromagnetism or Antiferromagnetism in One- or Two-Dimensional Isotropic Heisenberg Models"*. Physical Review Letters **17**, 1133–1136 (1966).

[85] J. Zinn-Justin. *"Path Integrals in Quantum Mechanics"*. Oxford University Press (2004).