Pyami: A Python Wrapper For The Libami Library

by

© **Michael Burke**

A thesis submitted to the Department of Physics and Physical Oceanography in partial fulfillment of the requirements for the degree of Bachelor of Science.

Department of Physics and Physical Oceanography

Memorial University

April 2023

St. John's, Newfoundland and Labrador, Canada

Abstract

We present `pyami`, a Python library to evaluate the frequency integrals encountered in the evaluation of Feynman diagrams. `pyami` is code bindings for the C++ library `libami`, which implements the Algorithmic Matsubara Integration technique that has been proposed in recent years. By implementing this library into Python, the plethora of mathematical Python libraries are now at one's disposal to evaluate the remaining spatial momentum integrals after the algorithmic Matsubara integration process. Once provided the topologies of the Feynman diagrams of interest, the values can be computed within an interactive Python environment such as a Jupyter Notebook. We then show example calculations using the Python importance sampling package, VEGAS, to evaluate self-energy diagrams on the real frequency axis by a renormalized perturbation theory scheme described in our recent work.

Acknowledgements

Firstly, like to thank my supervisor Dr. James P.F. LeBlanc with whom I held a NSERC USRA position in the spring 2022 semester prior to beginning this Honour's project, giving me all the relevant experience in this subject area of Many-body perturbation theory and lead to my first publication. He also provided me with the opportunity to travel and present my research at the American Physical Society's 2023 March Meeting which helped further my interest in a career in academia. His support has been instrumental in the creation of this thesis.

I would also like to thank Brad McNiven for his help on providing examples of how to format this thesis document.

I also thank Dr. Mykhaylo Evstigneev, honours coordinator, for his guidance and understanding during this irregular semester.

Lastly, I would like my parents for their support and encouragement all throughout my degree.

Table of contents

Title page i

Abstract ii

Acknowledgements iii

Table of contents iv

List of tables vii

List of figures viii

1 Introduction 1

2 Theory 3

2.1 Second Quantization 3

2.1.1 N-body Wave Function 4

2.1.2 Creation and Annihilation Operators 6

2.1.3 Occupation Numbers 7

2.2 Zero Temperature Green's Functions 9

2.2.1 Representations of Quantum Mechanics 10

2.2.2 S-Matrix 14

2.2.3 Green's Functions . . . 16
2.2.4 Wick's Theorem . . . 20
2.2.5 Feynman Diagrams . . . 25
2.2.6 Vacuum Polarization Graphs . . . 27
2.3 Non-zero Temperature Green's Functions . . . 30
2.3.1 Matsubara Green's Functions . . . 34
2.3.2 Retarded and Advanced Green's Functions . . . 37
2.4 The Hubbard Model . . . 43
2.5 Dyson's Equation . . . 45
2.6 Diagrammatic Techniques . . . 47
2.7 Algorithmic Matsubara Integration . . . 55
2.7.1 The `libami` library . . . 57
2.8 Renormalized Perturbation Theory . . . 59

3 Methods **65**
3.1 Writing `pyami` . . . 65
3.1.1 Pybind11 . . . 65
3.1.2 Using `pyami` . . . 66
3.2 Testing . . . 69
3.3 Scaling . . . 69
3.4 Renormalized Perturbation Theory . . . 71
3.4.1 Modifications to Diagram's Topologies . . . 71
3.4.2 Example Calculations . . . 73

4 Results **75**
4.1 Scaling Tests . . . 75
4.2 Renormalized Perturbation Theory Calculations . . . 77

5 Conclusion **81**

5.1 Discussion 81

5.2 Future work 82

A pyami Code Excerpts **83**

A.1 Binding codes with pybind11 83

B Sample Vegas implementation with pyami **89**

B.1 pyami integrand class 89

B.2 pyami-VEGAS integration function 92

Bibliography **95**

List of tables

2.1 Dictionary of the relevant diagrammatic components in the two dimensional Hubbard model using the Matsubara formalism. 48

4.1 `pyami` results for the average time in microseconds to change external parameters and evaluate the integrand of 2^{nd}, 4^{th} and 6^{th} order self-energy integrands 75

4.2 `libami` (C++) results for the average time in microseconds to change external parameters and evaluate the integrand of 2^{nd}, 4^{th} and 6^{th} order self-energy integrands 76

4.3 Time to evaluate fourth order self-energy diagram as a function of Monte Carlo samples. 79

List of figures

2.1 First order diagrams which arise from electron phonon interaction at zero temperature. 26

2.2 Vacuum polarization graphs from the $n = 4$ term of Eq. (2.86). 29

2.3 Partition of the infinite series in Eq. (2.77) where a connected diagram is multiplied by all possible disconnected diagrams. 30

2.4 Two dimensional square lattice depicting hopping energy t from a lattice site to a neighbouring site and the onsite interaction U for filled sites as modelled in the two dimensional Hubbard model. 43

2.5 Examples of self-energy parts. 45

2.6 Examples of proper self-energy parts. 46

2.7 Second order self-energy Feynman diagram. Labeling is specified to match the indices in Eq. 2.166. 49

2.8 Complex plane of ω showing poles in the Residue process of evaluating Matsubara sums. 51

2.9 Counter term self-energy diagrams with $s = 0, 1, 2$ insertions at second order and fourth order which arise in the renormalized perturbation theory scheme. 62

2.10 Schematic representation of the effect on sharp peaks in the AMI integrand due to the introduced numerical regulator α in the renormalized perturbation theory scheme. 64

3.1 Second order self-energy diagram from the renormalized perturbation theory scheme with one insertion 72

4.1 Imaginary part of the fourth order self-energy diagram as a function of the number of Monte Carlo samples using various Monte Carlo methods 77

Chapter 1

Introduction

Many-body perturbation theory is a tool for computing observables of interacting systems and gives rise to Feynman diagrams. Each Feynman diagram depicts a certain sequence of interactions in the many-body system [1]. In these Feynman diagrams, every possible value of internal momentum and frequency must be accounted for, leading to high dimensional integrals. In the last couple of decades, there have been many numerical approaches to evaluating these integrals [2].

Algorithmic Matsubara Integration (AMI) is a method for evaluating the frequency integrals that arise in the evaluation of Feynman diagrams [3]. Once the AMI procedure is applied, the remaining integrals over the spatial frequency degrees of freedom are left to be performed by some numerical approach, typically a Monte Carlo scheme. There is currently a C++ library called `libami` [4] which implements the AMI process. So once given the topology of a Feynman diagram and external parameters of the simulation, the frequency integrals are computed analytically, leaving the integrand for the internal momentum degrees of freedom. This library, once paired with an integration tool for the momentum integrals, produces promising results for applications

in condensed matter physics as described in [5].

However, since `libami` is written in C++, it has a number of limited to external libraries to perform the remaining momentum integrals. By implementing the AMI process into Python, the plethora of Python math libraries will be accessible to perform these remaining integrals and give the user access to interactive environments to preform calculations and view plots such as Jupyter Notebooks.

We present `pyami`, code bindings for the `libami` library, so that the power of the AMI is now available in Python. `pyami` is a complete Python module, so that once provided a Feynman diagram's topology, by using AMI, the momentum integrand is formed and the remaining integrals may be performed by external integration libraries such as the VEGAS importance sampling Python library.

This thesis consists of a full introduction to Feynman diagrams and Algorithmic Matsubara integration, discussion about the original C++ library and Python bindings to create `pyami` and lastly an application where `pyami` is used in conjunction with our resent work [6] for the fast evaluation of Feynman diagrams on the real frequency axis.

Chapter 2

Theory

2.1 Second Quantization

Typically, quantum mechanics is introduced via the concept of a wave-function to describe a particle's probability density and its response to an external potential. This is known as the first quantization, where as a result, the particles and energy of the system are quantized. When we allow the fields that create a potential to be influenced by the particles this is known as the second quantization. Now we essentially quantize the fields as well as the particles, this leads into Quantum field theory [7].

The study of second quantization is important in many-body systems [8]. It allows us to express operators in terms of creation and annihilation operators. This leads to models where we work in the grand canonical ensemble to take statistical mechanics approaches to solve problems in many-body systems. We will follow the discussion provided by Radi Jishi in Ref. [7] to introduce the key ideas in this theory.

2.1.1 N-body Wave Function

We start by looking at a N-body wave-function in a Hilbert space V from first quantization, expressed in an orthonormal and complete basis $|\phi_\nu\rangle$. We have

$$\langle\phi_\alpha|\phi_\beta\rangle = \delta_{\alpha\beta} \text{ (orthonormality)} \qquad \sum_\nu |\phi_\nu\rangle \langle\phi_\nu| = 1 \text{ (complete basis).} \tag{2.1}$$

Here each state, $|\phi_\nu\rangle$, has a index ν which is a N vector with each element containing a particle's quantum numbers. Since each particle will have its own sub-space, V_j with a corresponding orthonormal basis, $|\phi_{\nu_j}\rangle_j$ where only the quantum numbers of the j^{th} particle may vary, we can decompose the states of each particle into a direct product of these spaces. That is, $V = \bigotimes_j V_j$. So the state vector of the many-body system $|\Psi\rangle$ may be expressed as

$$|\Psi\rangle = \sum_{\nu_1,\ldots,\nu_N} C_{\nu_1\nu_2\ldots\nu_N} \bigotimes_i |\phi_{\nu_i}\rangle_i \,. \tag{2.2}$$

The one issue with this decomposition of the wave-functions is that we lose information about the symmetry of the state $|\Psi\rangle$. This information was originally in the product of kets - depending on whether or not the original wave-function is symmetric or antisymmetric under exchange of labels. But now, that information is inside the constant $C_{\nu_1\nu_2\nu_3\ldots\nu_N}$. Meaning whether or not the system is fermionic or bosonic is not as clear. We correct this by symmetrizing or antisymmetrizing the states after we decompose the wave-functions.

For bosons, we sum over the $N!$ permutations $1, 2, 3, ..., N$; so that the basis states

are given by

$$\Phi^B_{\nu_1\nu_2...\nu_N}(1,2,...,N) = \frac{1}{\prod_\mu \sqrt{n_\mu! N!}} \sum_P \phi_{\nu_1}[P(1)]\phi_{\nu_2}[P(2)]...\phi_{\nu_N}[P(N)], \tag{2.3}$$

where $P(i)$ is a permutation of $1, 2, 3, ..., N$ and n_μ is the number of times the index μ appears in the product. The prefactor guarantees that $\Phi^B_{\nu_1\nu_2...\nu_N}(1,2,...,N)$ is normalized.

For fermions, we have a similar formulation but now $n_\mu \in \{0, 1\}$ so the factorial is always 1 and we demand that the wave-function vanishes if there is a repeated state. This corresponds to an extra sign prefactor depending on the parity of the permutation P,

$$\Phi^F_{\nu_1\nu_2...\nu_N}(1,2,...,N) = \frac{1}{\sqrt{N!}} \sum_P (-1)^P \phi_{\nu_1}[P(1)]\phi_{\nu_2}[P(2)]...\phi_{\nu_N}[P(N)]. \tag{2.4}$$

This expression is the summation notation for a determinant as shown in Eq. (2.5) known as the Slater determinant,

$$\Phi^F_{\nu_1\nu_2...\nu_N}(1,2,...,N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_{v_1}(1) & \phi_{v_1}(2) & \dots & \phi_{v_1}(N) \\ \phi_{v_2}(1) & \phi_{v_2}(2) & \dots & \phi_{v_2}(N) \\ \vdots & \vdots & & \vdots \\ \phi_{v_N}(1) & \phi_{v_N}(2) & \dots & \phi_{v_N}(N) \end{vmatrix}. \tag{2.5}$$

Here it is clear that the properties of fermions hold when recalling properties of the determinant. If any state is repeated, this corresponds to repeated row in the determinant and, therefore, vanishes. Also, if we exchange the labels on two of the states, this corresponds to swapping two rows, which in terms of the determinant, has the affect of adding an extra minus sign to the prefactor, which is precisely the antisymmetric

property that we expect when exchanging labels.

2.1.2 Creation and Annihilation Operators

To avoid using determinants and sums over permutations of products, we define operators to deal with the number of particles rather than the explicit wave-functions. We first deal with fermions, where the number of particles whose state is $|\phi_\nu\rangle$ is either 0 or 1 and obey the Pauli exclusion principle. We define a creation operator, $c_\nu^\dagger$, so that once acting upon a state, the number of particles of state $|\phi_\nu\rangle$ increases by 1

$$c_\nu^\dagger \left|\phi_{\nu_1}...\phi_{\nu_N}\right\rangle = \left|\phi_\nu\phi_{\nu_1}...\phi_{\nu_N}\right\rangle . \tag{2.6}$$

So for the state $|\phi_\nu\phi_{\nu_1}...\phi_{\nu_N}\rangle$, the Slater determinant then gains an extra row and column for the extra particle. The Slater determinant then enforces the Pauli exclusion principle so that if the new state was already occupied, the determinant will have a repeated row and, therefore, vanish.

Next, we define the annihilation operator c_ν, so that once acting upon a state, the number of particles of state ν decreases by 1

$$c_\nu \left|\phi_\nu\phi_{\nu_1}...\phi_{\nu_N}\right\rangle = \left|\phi_{\nu_1}...\phi_{\nu_N}\right\rangle . \tag{2.7}$$

There is a subtlety here that the state that is being annihilated must be in the left-most position in the ket. Since the fermionic states are antisymmetric, we can rearrange the ket so that the state of interest is in the left-most position, while keeping track of the number of rearrangements by minus signs

$$c_{\nu'} \left|\phi_{\nu_1}\phi_{\nu'}...\phi_{\nu_N}\right\rangle = c_{\nu'}(-\left|\phi_{\nu'}\phi_{\nu_1}...\phi_{\nu_N}\right\rangle) = -\left|\phi_{\nu_1}...\phi_{\nu_N}\right\rangle . \tag{2.8}$$

We further define the annihilation operator for the case that the state $\nu \notin \{\nu_1, \nu_2, ..., \nu_N\}$ to evaluate to zero.

It is not a coincidence that the notation being used for c_ν and $c_\nu^\dagger$ is similar to the usual convention of an operator A and its hermitian conjugate $A^\dagger$, c_ν and $c_\nu^\dagger$ are indeed hermitian conjugates. To see this, let $|\Psi\rangle = |\phi_\nu \phi_{\nu_1} ...\rangle = c_\nu^\dagger |\phi_{\nu_1} ...\rangle$ then

$$\begin{aligned} 1 = \langle \Psi | \Psi \rangle &= (c_\nu^\dagger |\phi_{\nu_1} ...\rangle)^\dagger c_\nu^\dagger |\phi_{\nu_1} ...\rangle \\ &= \langle \phi_{\nu_1} ...| \, (c_\nu^\dagger)^\dagger c_\nu^\dagger |\phi_{\nu_1} ...\rangle \\ &= \langle \phi_{\nu_1} ...| \, ((c_\nu^\dagger)^\dagger |\phi_\nu \phi_{\nu_1} ...\rangle) \end{aligned} \tag{2.9}$$

which the last statement will only hold if the kets, $|\phi_{\nu_1} ...\rangle$ and $((c_\nu^\dagger)^\dagger |\phi_\nu \phi_{\nu_1} ...\rangle)$ are equal. Meaning that $(c_\nu^\dagger)^\dagger = c_\nu$, that is, the creation and annihilation operators are hermitian conjugates.

Lastly, other useful relations are c_ν and $c_\nu^\dagger$'s anti-commutation identities:

$$\{c_\nu, c_{\nu'}^\dagger\} = \delta_{\nu,\nu'} \qquad\qquad \{c_\nu, c_{\nu'}\} = \{c_\nu^\dagger, c_{\nu'}^\dagger\} = 0. \tag{2.10}$$

2.1.3 Occupation Numbers

During the manipulation of the the kets with creation and annihilation operators, we see that they do not act on specific wave-functions in the product, but rather they modify the collection wave-functions present in the product. Meaning that we can reduce our representation to solely the number of particles in a each state j. For example,

$$|\phi_{\nu_1} \phi_{\nu_3}\rangle = |1 \; 0 \; 1\rangle \, . \tag{2.11}$$

This allows us to form Hilbert spaces as follows. Let V^0 be the space that only contains the vacuum state, $|0\ 0\ 0\ \ldots\rangle$, V^1 be the space containing single wave-functions, e.g. $|1\ 0\ 0\ ...\rangle$, $|0\ 1\ 0\ ...\rangle, \ldots$, V^2 be the space containing the quantum states of a 2 body problem e.g. $|1\ 1\ 0\ ...\rangle$, $|1\ 0\ 1\ \ldots\rangle, \ldots$ and so on. Together these spaces, $V^{(N)}$, in a direct sum, will represent the space in which the creation and annihilation operators act. This space is called the Fock space

$$F = \bigoplus_i V^{(i)}. \tag{2.12}$$

So that for some state $|\Psi\rangle \in V^k$, then $c_\nu |\Psi\rangle \in V^{k-1}$ and $c_\nu^\dagger |\Psi\rangle \in V^{k+1}$. We then see that the creation and annihilation operators provide a means of jumping between the sub-spaces.

In this formalism, we can express the operators in a more convenient way

$$c_\nu^\dagger |n_1 ... n_\nu ...\rangle = (-1)^{n_1+n_2+...+n_{\nu-1}} \sqrt{1-n_\nu}\, |n_1\ \ldots\ n_\nu + 1\ \ldots\rangle \tag{2.13}$$

$$c_\nu |n_1 ... n_\nu ...\rangle = (-1)^{n_1+n_2+...+n_{\nu-1}} \sqrt{n_\nu}\, |n_1 ... n_\nu - 1 ...\rangle . \tag{2.14}$$

Very briefly, we can do the same work for bosons. Now we are able to have occupation numbers, $n_i \in \mathbb{Z}_{\geq 0}$ as there is no Pauli exclusion principle. We define the analogous creation and annihilation operators $a_\nu^\dagger$ and a_ν as the following

$$a_\nu^\dagger |n_1 ... n_\nu ...\rangle = \sqrt{n_\nu + 1}\, |n_1 ... n_\nu + 1 ...\rangle \tag{2.15}$$

$$a_\nu |n_1 ... n_\nu ...\rangle = \sqrt{n_\nu}\, |n_1 ... n_\nu - 1 ...\rangle . \tag{2.16}$$

So we can see the similar behaviour as the fermionic case. That is, if the state has no particles in a state $|\nu\rangle$, then the annihilation operator, Eq. (2.16) evaluates to zero.

The last comment to make is that the creation and annihilation operators are typically used in conjunction to determine the number of particles in state $|\phi_\nu\rangle$, n_ν, belonging to a overall state $|\Psi\rangle$, as follows

$$\hat{n}_\nu |\Psi\rangle = c_\nu^\dagger c_\nu |\Psi\rangle = n_\nu |\Psi\rangle \tag{2.17}$$

$$\hat{n}_\nu |\Psi\rangle = a_\nu^\dagger a_\nu |\Psi\rangle = n_\nu |\Psi\rangle . \tag{2.18}$$

This will allow us to express operators in terms of the creation and anhilation operators later on.

2.2 Zero Temperature Green's Functions

Many-body calculations can be formulated for ground state $T = 0$ systems [8]. Although experimental work is never done at these temperatures, most observables that we are interested in are not sensitive to temperatures. For this reason, we work with zero temperature since it can be thought of as the ground state of the many-body system and serves as a jumping off point to acquire non-zero temperature observables. For this section, we will follow Gerald Mahan's discussion in Ref. [8] while using the notation from the previous section.

We start with a Hamiltonian that may be written as

$$H = H_0 + V. \tag{2.19}$$

Here H_0 is a Hamiltonian with a known solution and V is a small modification. This is known as a perturbation method where the goal is to expand the solution of H in terms of H_0's known solutions with corrections given by V in an infinite series. This

is the standard approach to many-body systems where Green's functions are used to acquire solutions. First, we need to define the mathematical machinery required to formally introduce zero temperature Green's functions.

2.2.1 Representations of Quantum Mechanics

First, we must note that there are various representations of quantum mechanics. Of course, these representations must all predict the same values of physical observables and, therefore, be equivalent. However, a clever choice of representation can greatly simplify the math required to acquire the result.

The most common representation is Schrödinger's representation where the wave-function is a function of time governed by the Schrödinger equation (note: we are using $\hbar = 1$)

$$i\frac{\partial \psi}{\partial t} = H\psi \implies \psi(t) = e^{-iHt}\psi(0) \tag{2.20}$$

and may be probed via operators which are static.

Werner Heisenberg provided an alternative point of view where the wave-function is static and the operators are permitted to evolve in time governed by the Hamiltonian

$$i\hbar\frac{\partial}{\partial t}O(t) = [O(t), H] \implies O(t) = e^{iHt}Oe^{-iHt}. \tag{2.21}$$

Quite frequently, we are interested in matrix elements of various operators. As an example of these two representations returning the same result, we can compute the matrix element of an operator, O. In the Schrödinger representation, we can take the static operators to be evaluated at time zero, that is, $O = O(0)$ so we have

$$\langle \psi_m^\dagger(t)O(0)\psi_n(t)\rangle = \langle \psi_m^\dagger(0)e^{iHt}O(0)e^{-iHt}\psi_n(0)\rangle. \tag{2.22}$$

Similarly, for the Heisenberg representation, we can take $\psi = \psi(0)$ and we find

$$\langle \psi_m^\dagger(0) O(t) \psi_n(0) \rangle = \langle \psi_m^\dagger(0) e^{iHt} O(0) e^{-iHt} \psi_n(0) \rangle. \tag{2.23}$$

Another way of looking at our quantum system is under the interaction representation. Here we take both the wave-function and operators to be time dependent. We break the Hamiltonian into H_0 and V as in Eq. (2.19). This allows us to form a perturbation approach where H_0 is the exactly solvable unperturbed part and V is the interaction. Note that here we will use the *hat* notation ($\hat{}$) to denote the time dependence in this representation compared to the previously introduced representations. In this representation, the operators and wave-functions have time dependence

$$\hat{O}(t) = e^{iH_0 t} O e^{-iH_0 t}, \tag{2.24}$$

$$\hat{\psi}(t) = e^{iH_0 t} e^{-iHt} \psi(0). \tag{2.25}$$

We can then check the matrix elements of the operator, O, as before to again acquire the same result

$$\begin{aligned} \langle \hat{\psi}_m^\dagger(t) \hat{O}(t) \hat{\psi}_n(t)^\dagger \rangle &= \langle \psi_m^\dagger(0) e^{iHt} e^{-iH_0 t} (e^{iH_0 t} O e^{-iH_0 t}) e^{iH_0 t} e^{-iHt} \psi_n(0) \rangle \\ &= \langle \psi_m^\dagger(0) e^{iHt} O(0) e^{-iHt} \psi_n(0) \rangle. \end{aligned} \tag{2.26}$$

Next, we can derive the time evolution of the wave-functions as

$$\begin{aligned} \frac{\partial}{\partial t} \hat{\psi}(t) &= i e^{iH_0 t} (H_0 - H) e^{-iHt} \psi(0) \\ &= -i e^{iH_0 t} V e^{-iHt} \psi(0) \\ &= -i e^{iH_0 t} V e^{-iH_0 t} [e^{iH_0 t} e^{-iHt} \psi(0)] \\ &= -i \hat{V}(t) \hat{\psi}(t). \end{aligned} \tag{2.27}$$

Moving forward with this representation, it is easier to define a time evolution operator

$$U(t) = e^{iH_0 t} e^{-iHt}. \tag{2.28}$$

$U(t)$ acts as a mapping from the Schrödinger representation to the interaction representation as

$$\hat{\psi}(t) = U(t)\psi(0). \tag{2.29}$$

Which has the property

$$\frac{\partial}{\partial t} U(t) = -i\hat{V}(t)U(t). \tag{2.30}$$

Iterating the integral of Eq. (2.30) and using $U(0) = 1$ we obtain the series

$$\begin{aligned} U(t) &= 1 - i\int_0^t dt_1 \hat{V}(t_1)U(t_1) \\ &= 1 - i\int_0^t dt_1\, \hat{V}(t_1) + (-i)^2 \int_0^t dt_1 \int_0^{t_1} dt_2\, \hat{V}(t_1)\hat{V}(t_2) + \dots \\ &= \sum_{n=0}^{\infty} (-i)^n \int_0^t dt_1 \int_0^{t_1} dt_2 \dots \int_0^{t_{n-1}} dt_n\, \hat{V}(t_1)\hat{V}(t_2)\dots\hat{V}(t_n). \end{aligned} \tag{2.31}$$

Next, we introduce the time ordering operator, T. When T acts on a group of time dependent operators, it will rearrange the operators from earliest to latest from right to left. For example,

$$T[\hat{V}(t_1)\hat{V}(t_2)\hat{V}(t_3)] = \hat{V}(t_2)\hat{V}(t_3)\hat{V}(t_1) \tag{2.32}$$

for the case that $t_2 > t_3 > t_1$. Another way of looking at the time ordering operator

is by using the Heaviside step function

$$T[\hat{V}(t_1)\hat{V}(t_2)] = \Theta(t_1 - t_2)\hat{V}(t_1)\hat{V}(t_2) + \Theta(t_2 - t_1)\hat{V}(t_2)\hat{V}(t_1). \tag{2.33}$$

The reason for introducing the time ordering operator becomes more clear with its integral properties, which for the n operator case, looks similar to the integrals in Eq. (2.31)

$$\begin{aligned} &\frac{1}{2!}\int_0^t dt_1 \int_0^t dt_2\, T[\hat{V}(t_1)\hat{V}(t_2)] \\ &\quad = \frac{1}{2!}\int_0^t dt_1 \int_0^{t_1} dt_2\, \hat{V}(t_1)\hat{V}(t_2) + \frac{1}{2!}\int_0^t dt_1 \int_0^t dt_2\, \hat{V}(t_2)\hat{V}(t_1). \end{aligned} \tag{2.34}$$

We see by swapping the integration variables, $t_1 \to t_2$ and $t_2 \to t_1$ in each term

$$\frac{1}{2!}\int_0^t dt_1 \int_0^t dt_2\, T[\hat{V}(t_1)\hat{V}(t_2)] = \int_0^t dt_1 \int_0^{t_1} dt_2\, \hat{V}(t_1)\hat{V}(t_2). \tag{2.35}$$

Similarly for the $n = 3$ term in the expansion in Eq. (2.31)

$$\frac{1}{3!}\int_0^t dt_1 \int_0^t dt_2 \int_0^t dt_3\, T[\hat{V}(t_1)\hat{V}(t_2)\hat{V}(t_3)] = \int_0^t dt_1 \int_0^{t_1} dt_2 \int_0^{t_2} dt_3\, \hat{V}(t_1)\hat{V}(t_2)\hat{V}(t_3). \tag{2.36}$$

Inductively, this will hold for all $n \in \mathbb{Z}$ and so we can write Eq. (2.31) as

$$\begin{aligned} U(t) &= 1 + \sum_{n=1}^{\infty} \frac{(-i)^n}{n!} \int_0^t dt_1 \int_0^t dt_2 \ldots \int_0^t dt_n\, T[\hat{V}(t_1)\hat{V}(t_2)\ldots\hat{V}(t_n)] \\ &= T\exp\left(-i\int_0^t dt_1\, \hat{V}(t_1)\right). \end{aligned} \tag{2.37}$$

Now having developed the tools for the interaction representation of Quantum mechanics, we are ready to look into the evolution of the wave-function.

2.2.2 S-Matrix

Recall we had the time evolution of the wave-function given by

$$\hat{\psi}(t) = U(t)\hat{\psi}(0). \tag{2.38}$$

Let us define an operator $S(t, t')$ so that the wave-function at time t' is mapped to time t

$$\hat{\psi}(t) = S(t, t')\hat{\psi}(t'). \tag{2.39}$$

Comparing Eqs. (2.38, 2.39) we can conclude that

$$S(t, t') = U(t)U(t')^{\dagger}. \tag{2.40}$$

One property of the S-matrix is that the identity may be expressed as $S(t', t)S(t, t')$

$$\hat{\psi}(t) = S(t, t')\hat{\psi}(t') \implies \hat{\psi}(t') = S(t', t)S(t, t')\hat{\psi}(t') \implies S(t', t)S(t, t') = 1. \tag{2.41}$$

Another important property of the S-Matrix is that it can be expressed as a time-ordered operator

$$\begin{aligned} \frac{\partial}{\partial t}S(t, t') =& \frac{\partial}{\partial t}U(t)U(t')^{\dagger} = -i\hat{V}(t)U(t)U(t')^{\dagger} = -i\hat{V}(t)S(t, t') \\ &\iff S(t, t') = T\exp\left(-i\int_{t'}^{t} dt_1\, \hat{V}(t_1)\right). \end{aligned} \tag{2.42}$$

Here the exponential is short hand for its Taylor series expansion like in Eq. (2.37).

Recall that we split the Hamiltonian into H_0 and V where the solution to H_0 is known. The goal of the work at zero temperature is to find the ground state wave-function, $\psi(0)$. One method would be to use Green's functions to acquire a

perturbative approximation to the Hamiltonian H. Unfortunately, to do this, we need an exact solution to the ground state $\psi(0)$ to expand about. To escape this circular solution, we need some extra information. Luckily, we know the eigenvalues and eigenvectors of H_0. Let ϕ_0 be the ground state of H_0. Gell-Mann and Low in [9] were the first to relate the ground state of a non-interacting state ϕ_0 and the vacuum state of the interacting system $\psi(0)$ by adiabatically turning the interactions on and off to find that

$$\psi(0) = S(0, -\infty)\phi_0. \tag{2.43}$$

A hand-wavy reasoning is that since $S(0,t)S(t,0) = 1$ we have

$$\hat{\psi}(t) = S(t,0)\psi(0) \implies \psi(0) = S(0,t)\hat{\psi}(t). \tag{2.44}$$

Then taking $t \to -\infty$ and making the assumption that at the dawn of time, the interacting system was not interacting, that is $\psi(-\infty) = \phi_0$, we find

$$\psi(0) = S(0, -\infty)\psi(-\infty) = S(0, -\infty)\phi_0. \tag{2.45}$$

The interpretation of $S(0, -\infty)$ is that it brings the wave-function through time adiabatically so that it has no knowledge of the perturbation, V, and only the original Hamiltonian H_0.

One last result before jumping into Green's functions is in the $t \to \infty$ limit,

$$\hat{\psi}(\infty) = S(\infty, 0)\psi(0). \tag{2.46}$$

Using a similar reasoning as above, we can argue that $\hat{\psi}(\infty)$ is related to ϕ_0 as well.

Namely, they differ by a phase factor, L

$$\begin{aligned} \phi_0 e^{iL} &= \hat{\psi}(\infty) = S(\infty, 0)\psi(0) = S(\infty, \infty)\phi_0 \\ e^{iL} &= \langle \phi_0 | \, |S(\infty, \infty)| \, |\phi_0\rangle \, . \end{aligned} \tag{2.47}$$

Now we are ready to define the Green's functions to proceed in a perturbative fashion to try to find the eigenvalues and eigenvectors of the full Hamiltonian, H, at zero temperature.

2.2.3 Green's Functions

The following procedure has analogs for three types of particles: electrons, phonons and photons. We will follow the case of electrons or fermions in general.

At zero temperature, the fermionic Green's function is defined as

$$G(\nu, t - t') = -i \, \langle | \, T c_\nu(t) c_\nu^\dagger(t') \, | \rangle \, , \tag{2.48}$$

where $c_\nu^\dagger(t)$ and $c_\nu(t)$ are the Heisenberg representations of the fermionic creation and annihilation operators defined in Eqs. (2.13 and 2.14)

$$\begin{aligned} c_\nu^\dagger(t) &= e^{iHt} c_\nu^\dagger e^{-iHt} \\ c_\nu(t) &= e^{iHt} c_\nu e^{-iHt}. \end{aligned} \tag{2.49}$$

Since we are working with zero temperature, only the ground state of H, denoted by the empty ket, $|\rangle$, will be accessible. Here the states ν are the known eigen-states of the unperturbed Hamiltonian H_0, where ν contains the momentum, $\mathbf{p}$, and spin, σ, of the state, $\nu = (\mathbf{p}, \sigma)$.

At the moment, we do not know any states of H and we will use the Green's function to solve them. The Green's function is defined so that it describes two situations as follows. First, look at the case $t > t'$ so the time ordering operator is unity

$$G(\nu, t > t') = -i \langle | \, c_\nu(t) c^\dagger{}_\nu(t') \, | \rangle \, . \tag{2.50}$$

At time t', an excited particle in state ν is created. Then, at a later time t, this same particle is destroyed. Now, if ν was an eigen-state of H, meaning $H c^\dagger{}_\nu(t') \, | \rangle = \epsilon_\nu \, | \rangle$ and $H \, | \rangle = \epsilon_0 \, | \rangle$, the propagator could be written as

$$G(\nu, t > t') = -i \exp\left(-i(t - t')(\epsilon_\nu - \epsilon_0)\right). \tag{2.51}$$

But for the general state, ν, this will not be the case. As a result, the particle in state ν will be scattered, shifted in energy during the time period $t - t'$. Then when we remove the particle, we can see how much energy remains in the state ν.

Alternatively, we can look at the case $t' > t$

$$G(\nu, t' > t) = +i \langle | \, c^\dagger_\nu(t') c_\nu(t) \, | \rangle \, . \tag{2.52}$$

Meaning we destroy a particle in the ground state with momenta and spin $(\mathbf{p}, \sigma)$ at time t. Of course, this is only possible if such a particle exists in the ground state of H at zero temperature. This process usually appears within the terminology of particles and holes in a metal to depict an absence in the Fermi sea. Just like in the $t > t'$ case, the hole will interact and scatter in the system until it is filled in at time t'. This measurement allows us to get information about the hole excitation.

Next, we will take the Green's function in the Heisenberg representation and convert it into the interaction representation. Note the ground state $| \rangle$ is mapped as

follows originally described by [9]

$$|\rangle = S(0,-\infty)\,|\rangle_0\,, \tag{2.53}$$

where $|\rangle_0$ is the ket of the ground state of H_0 in the interaction representation. So we have

$$\begin{aligned} c_\nu(t) =& e^{iHt}e^{-iH_0t}\hat{c}_\nu(t)e^{iH_0t}e^{-iHt} = U^\dagger(t)\hat{c}_\nu(t)U(t) \\ =& S(0,t)\hat{c}_\nu(t)S(t,0) \\ G(\nu,t-t') =& -i\Theta(t-t')_0\,\langle|\, S(-\infty,0)S(0,t)\hat{c}_\nu(t)S(t,0)S(0,t') \\ & \times \hat{c}^\dagger_\nu(t')S(t',0)S(0,-\infty)\,|\rangle_0 \\ & + i\Theta(t'-t)_0\,\langle|\, S(-\infty,0)S(0,t')\hat{c}^\dagger_\nu(t')S(t',0)S(0,t) \\ & \times \hat{c}^\dagger_\nu(t)S(t,0)S(0,-\infty)\,|\rangle_0\,. \end{aligned} \tag{2.54}$$

Recall the phase difference in the non-interacting ground state from Eq. (2.47), so we have

$${}_0\langle|\, S(-\infty,0) = e^{iL}{}_0\langle|\, S(\infty,-\infty)S(-\infty,0) = \frac{{}_0\langle|\, S(\infty,0)}{{}_0\langle|\, S(\infty,-\infty)\,|\rangle_0}. \tag{2.55}$$

So the Green's function may be written as

$$\begin{aligned} G(\nu,t-t') =& -\frac{i}{{}_0\langle|\, S(\infty,-\infty)\,|\rangle_0}[\Theta(t-t')_0\,\langle|\, S(\infty,t)\hat{c}_\nu(t) \\ & \times S(t,t')\hat{c}^\dagger_\nu(t')S(t',-\infty)\,|\rangle_0 \\ & - \Theta(t'-t)_0\,\langle|\, S(\infty,t')\hat{c}^\dagger_\nu(t')S(t',t)\hat{c}^\dagger_\nu(t)S(t,-\infty)\,|\rangle_0]. \end{aligned} \tag{2.56}$$

The first term can be further simplified using the time ordering operator to put all

the parts of S under $S(-\infty,\infty)$

$$\begin{aligned}&\Theta(t-t')_0 \langle|\, S(\infty,t)\hat{c}_\nu(t)S(t,t')\hat{c}^\dagger_\nu(t')S(t',-\infty)\,|\rangle_0 \\ &= \Theta(t-t')_0 \langle|\, T\hat{c}_\nu(t)\hat{c}^\dagger_\nu(t')S(\infty,-\infty)\,|\rangle_0 .\end{aligned} \tag{2.57}$$

So finally, we arrive at the expression for the total Green's function

$$G(\nu,t-t') = -i\frac{{}_0\langle|\, T\hat{c}_\nu(t)\hat{c}^\dagger_\nu(t')S(\infty,-\infty)\,|\rangle_0}{{}_0\langle|\, TS(\infty,-\infty)\,|\rangle_0}. \tag{2.58}$$

We can also define the non-interacting Green's function $G^{(0)}(\nu,t-t')$ for the case that $V=0$ and the S-matrix is unity

$$G^{(0)} = -i\,{}_0\langle|\, T\hat{c}_\nu(t)\hat{c}^\dagger_\nu(t')\,|\rangle_0 . \tag{2.59}$$

This is also known as the unperturbed Green's function or the free propagator.

Later, we will show examples of how Feynman diagrams arise in calculations using a specific electron-phonon interaction. So here, we show the basic results of the phonon Green's functions. Although until now, we have focused on the fermionic case, we can also do similar work for phonons to find the Green's function

$$D(\mathbf{q},\lambda,t-t') = -i\langle|\, TA_{\mathbf{q}\lambda}(t)A_{-\mathbf{q}\lambda}(t')\,|\rangle \tag{2.60}$$

$$A_{\mathbf{q}\lambda} = a_{\mathbf{q}\lambda} + a^\dagger_{-\mathbf{q}\lambda}. \tag{2.61}$$

Here $\mathbf{q}$ is the wave-vector and λ refers to the polarization of the phonons. We will assume one type of phonon to drop the λ subscripts. In the interaction representation we find

$$D(\mathbf{q},t-t') = -i\frac{{}_0\langle|\, T\hat{A}_{\mathbf{q}}(t)\hat{A}_{-\mathbf{q}}(t')S(\infty,-\infty)\,|\rangle_0}{{}_0\langle|\, TS(\infty,-\infty)\,|\rangle_0}. \tag{2.62}$$

At zero temperature, no phonons exist. The ground states, $|\rangle_0$ and $|\rangle$ are as previously defined. Note that in a electron-phonon system the notation $|\rangle_0$ refers to the combination of ground states of electrons and phonons. The phonons will have the ground state as its vacuum state so any of the electron's ground states can be used for $|\rangle_0$.

The unperturbed phonon Green's function is defined as

$$\begin{aligned} D^{(0)}(\mathbf{q}, t-t') &= -i\,{}_0\langle|\, T\hat{A}_{\mathbf{q}}(t)\hat{A}_{-\mathbf{q}}(t')\,|\rangle_0 \\ &= -i\,{}_0\langle|\, T(a_{\mathbf{q}}e^{-i\omega_{\mathbf{q}}t} + a^{\dagger}_{-\mathbf{q}}e^{i\omega_{\mathbf{q}}t})(a_{-\mathbf{q}}e^{-i\omega_{\mathbf{q}}t'} + a^{\dagger}_{\mathbf{q}}e^{i\omega_{\mathbf{q}}t'})\,|\rangle_0\,. \end{aligned} \tag{2.63}$$

At zero temperature we find

$$\begin{aligned} &{}_0\langle|\, a_{\mathbf{q}}a^{\dagger}_{\mathbf{q}}\,|\rangle_0 = 1 \\ &{}_0\langle|\, a^{\dagger}_{\mathbf{q}}a_{\mathbf{q}}\,|\rangle_0 = 0 \\ &D^{(0)}(\mathbf{q}, t-t') = -i[\Theta(t-t')e^{-i\omega_{\mathbf{q}}(t-t')} + \Theta(t'-t)e^{i\omega_{\mathbf{q}}(t-t')}]. \end{aligned} \tag{2.64}$$

2.2.4 Wick's Theorem

Next, we will like to evaluate these Green's functions. This will be done using the series expansion of $S(\infty, -\infty)$ as described in Eq. (2.42)

$$\begin{aligned} G(\nu, t-t') = &\sum_{n=0}^{\infty} \frac{(-i)^{n+1}}{n!} \int_{-\infty}^{\infty} dt_1 \ldots \int_{-\infty}^{\infty} dt_n \\ &\times \frac{{}_0\langle|\, T\hat{c}_{\nu}(t)\hat{V}(t_1)\hat{V}(t_2)\ldots\hat{V}(t_n)\hat{c}^{\dagger}_{\nu}(t')\,|\rangle_0}{{}_0\langle|\, S(\infty,-\infty)\,|\rangle_0}. \end{aligned} \tag{2.65}$$

For now, we will ignore the constant ${}_0\langle|\, S(\infty,-\infty)\,|\rangle_0{}^{-1}$ and figure out how to deal with the time-ordered brackets

$$ {}_0\langle|\, T\hat{c}_\nu(t)\hat{V}(t_1)\hat{V}(t_2)\ldots\hat{V}(t_n)\hat{c}^\dagger_\nu(t'))\,|\rangle_0\,. \tag{2.66}$$

Recall that the second quantization formulation allows us to expand the operators $\hat{V}(t_i)$ in terms of creation and annihilation operators so we will have expressions which look like

$$ {}_0\langle|\, T\hat{c}_1(t)\hat{c}^\dagger_{1'}(t_1)\ldots\hat{c}_n(t_n)\hat{c}^\dagger_{n'}(t')\,|\rangle_0\,. \tag{2.67}$$

Inserting the expressions for each $\hat{V}(t_i)$ and then summing of all possible time orderings would be rather cumbersome and so we look for an alternative approach.

First, let us get a general idea of how we evaluate these expressions. By the orthonormality of the kets $|\rangle_0$, the value of Eq. (2.67) will be zero unless the result of all the operators acting on the state is proportional to the state we started with, that is

$$ T\hat{c}_1(t)\hat{c}^\dagger_{1'}(t_1)\ldots\hat{c}_n(t_n)\hat{c}^\dagger_{n'}(t')\,|\rangle_0 \propto |\rangle_0\,. \tag{2.68}$$

Or by breaking the operators into pairs, for any particle created in state ν_i at time t_i there must be an annihilation of the particle in state ν_i at some time $t_{i'} > t_i$ and similarly for annihilation first. This means we can expand the state in pairs as

$$ {}_0\langle|\, T\hat{c}_\alpha(t)\hat{c}^\dagger_\beta(t')\,|\rangle_0 \tag{2.69}$$

which vanishes unless $\alpha = \beta$, while

$$ {}_0\langle|\, T\hat{c}_\alpha(t)\hat{c}^\dagger_\beta(t_1)\hat{c}_\gamma(t_2)\hat{c}^\dagger_\delta(t')\,|\rangle_0 \tag{2.70}$$

vanishes unless $\alpha = \beta, \gamma = \delta$ or unless $\alpha = \delta, \beta = \gamma$. The number of these arrangements leads into a simple combinatorics problem but only a limited number of these cases are physically relevant. We would like to sort the cases to simplify our expression. One of the sorting procedures is known as Wick's theorem [8].

Wick's Theorem states that in making all the possible pairings between creation and annihilation operators, each pairing should be time-ordered. The time ordering of each pair gives the proper time ordering to the entire result. For example,

$$\begin{aligned}
&{}_0\left\langle\right| T\hat{c}_\alpha(t)\hat{c}^\dagger_\beta(t_1)\hat{c}_\gamma(t_2)\hat{c}^\dagger_\delta(t')\left|\right\rangle_0 \\
&= {}_0\left\langle\right| T\hat{c}_\alpha(t)\hat{c}^\dagger_\beta(t_1)\left|\right\rangle_{0\,0}\left\langle\right| T\hat{c}_\gamma(t_2)\hat{c}^\dagger_\delta(t')\left|\right\rangle_0 \\
&- {}_0\left\langle\right| T\hat{c}_\alpha(t)\hat{c}^\dagger_\delta(t')\left|\right\rangle_{0\,0}\left\langle\right| T\hat{c}_\gamma(t_2)\hat{c}^\dagger_\beta(t_1)\left|\right\rangle_0 \\
&= \delta_{\alpha\beta}\delta_{\gamma\delta}\,{}_0\left\langle\right| T\hat{c}_\alpha(t)\hat{c}^\dagger_\alpha(t_1)\left|\right\rangle_{0\,0}\left\langle\right| T\hat{c}_\gamma(t_2)\hat{c}^\dagger_\gamma(t')\left|\right\rangle_0 \\
&- \delta_{\alpha\delta}\delta_{\beta\gamma}\,{}_0\left\langle\right| T\hat{c}_\alpha(t)\hat{c}^\dagger_\alpha(t')\left|\right\rangle_{0\,0}\left\langle\right| T\hat{c}_\gamma(t_2)\hat{c}^\dagger_\gamma(t_1)\left|\right\rangle_0 .
\end{aligned} \tag{2.71}$$

Note that for each pairing of operators, ${}_0\left\langle\right| T\hat{c}_\alpha(t)\hat{c}^\dagger_\beta(t_1)\left|\right\rangle_0$ the T operator is present so it contains 2 cases. Generally, the expectation value with n time ordered pairs will correspond to $n!$ cases.

In practice, the operators $\hat{V}(t)$ will also contain bosonic creation and annihilation operators. But there is no need to panic since these operators commute with the fermionic ones. This means that it does not matter what order they are written in the time ordering expansion and they can be immediately factored out.

In the case that a particle is created and destroyed at the same time, such as

$${}_0\left\langle\right| T\hat{c}^\dagger_\alpha(t)\hat{c}_\beta(t)\left|\right\rangle_0 . \tag{2.72}$$

conventionally, the annihilation operator always goes to the right so

$$ {}_0\langle|\, T\hat{c}^\dagger_\alpha(t)\hat{c}_\beta(t)\,|\rangle_0 = \delta_{\alpha\beta}\,{}_0\langle|\,\hat{c}^\dagger_\alpha(t)\hat{c}_\alpha(t)\,|\rangle_0 = \delta_{\alpha\beta}n_F(\epsilon_\alpha). \tag{2.73} $$

Here $n_F(\epsilon_\alpha)$ is the occupation number for the energy of eigen-state α at time t. This convention follows suit with the convention when Hamiltonians are constructed, that is, the creation operators are always to the left of the annihilation operators.

When two fermionic operators have different time arguments in a pairing, it is conventional to put the creation operator on the right.

$$ {}_0\langle|\, T\hat{c}^\dagger_\alpha(t_1)\hat{c}_\beta(t_2)\,|\rangle_0 = -\delta_{\alpha\beta}\,{}_0\langle|\, T\hat{c}_\alpha(t_2)\hat{c}^\dagger_\beta(t_1)\,|\rangle_0\,. \tag{2.74} $$

This equation above may be recognized as the non-interacting propagator from Eq. (2.59). We may conclude from this that all parings that arise from Wick's theorem may be replaced by non-interacting propagators. For example, Eq. (2.71) may be written as

$$ \begin{aligned} &(-i)^2\,{}_0\langle|\, T\hat{c}_\alpha(t)\hat{c}^\dagger_\beta(t_1)\hat{c}_\gamma(t_2)\hat{c}^\dagger_\delta(t')\,|\rangle_0 \\ &= \delta_{\alpha\beta}\delta_{\gamma\delta}\,G^{(0)}(\alpha, t-t_1)\cdot G^{(0)}(\gamma, t_2-t') \\ &- \delta_{\alpha\delta}\delta_{\beta\gamma}\,G^{(0)}(\alpha, t-t')\cdot G^{(0)}(\gamma, t_2-t_1). \end{aligned} \tag{2.75} $$

In summary, Wick's theorem allows us to expand a time ordered bracket into all of its possible pairings. Each pairing will be either a non-interacting propagator or a number operator and by doing the expansion we get the correct time ordering for each pairing.

Now let us return to the expansion in Eq. (2.65) and focus on the case of the

electron-phonon interaction:

$$V = \sum_{\mathbf{q},\mathbf{k},s} M_{\mathbf{q}} A_{\mathbf{q}} c^{\dagger}_{\mathbf{k}+\mathbf{q},s} c_{\mathbf{k}s}. \tag{2.76}$$

Later in this thesis, we will look at the Hubbard model, but this will serve as an example of how the perturbation expansion is resolved. For the $n = 0$ term in Eq. (2.65), we see that this is the non-interacting propagator, $G^{(0)}(\mathbf{p}, t - t')$. For the $n = 1$ case, we see that there is only one $A_{\mathbf{q}}$ operator present and will lead to a zero expectation value. In fact, this will happen for all of the odd values of n as there will be an odd number of $A_{\mathbf{q}}$. The $n = 2$ case looks like

$$\begin{aligned} G(\mathbf{p}, t - t') = & G^{(0)}(\mathbf{p}, t - t') + \frac{(-i)^3}{2!} \int_{-\infty}^{\infty} dt_1 \int_{-\infty}^{\infty} dt_2 \\ & \times \sum_{\mathbf{q_1},\mathbf{q_2}} M_{\mathbf{q_1}} M_{\mathbf{q_2}} \, {}_0\langle | \, T \hat{A}_{\mathbf{q_1}}(t_1) \hat{A}_{\mathbf{q_2}}(t_2) \, | \rangle_0 \\ & \sum_{\mathbf{k_1},\mathbf{k_2} s s'} {}_0\langle | \, T \hat{c}_{\mathbf{p}\sigma}(t) \hat{c}^{\dagger}_{\mathbf{k_1}+\mathbf{q_1} s}(t_1) \hat{c}_{\mathbf{k_1} s}(t_1) \\ & \times \hat{c}^{\dagger}_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2) \hat{c}_{\mathbf{k_2} s'}(t_2) \hat{c}^{\dagger}_{\mathbf{p}\sigma}(t') \, | \rangle_0 \, . \end{aligned} \tag{2.77}$$

The phonon term evaluates as a single phonon Green's function

$${}_0\langle | \, T \hat{A}_{\mathbf{q_1}}(t_1) \hat{A}_{\mathbf{q_2}}(t_2) \, | \rangle_0 = i \delta_{\mathbf{q_1}+\mathbf{q_2}} D^{(0)}(\mathbf{q_1}, t_1 - t_2). \tag{2.78}$$

Unfortunately, the electron term is not so simple with six possible pairings using

Wick's theorem. These are simplified using that $\mathbf{q_1} = -\mathbf{q_2}$

$$
\begin{aligned}
&{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\hat{c}_{\mathbf{k_1}s}(t_1)\hat{c}^\dagger_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\hat{c}_{\mathbf{k_2}s'}(t_2)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_0 \\
=&{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_1}s}(t_1)\hat{c}^\dagger_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_2}s'}(t_2)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_0 \\
&+{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_1}s}(t_1)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_2}s'}(t_2)\hat{c}^\dagger_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\,|\rangle_0 \\
&+{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_1}s}(t_1)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\hat{c}^\dagger_{\mathbf{k_2}s'}(t_2)\,|\rangle_0 \\
&+{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_2}s'}(t_2)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\hat{c}^\dagger_{\mathbf{k_1}s}(t_1)\,|\rangle_0 \\
&+{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_{00}\langle|\, T\hat{c}^\dagger_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\hat{c}_{\mathbf{k_1}s}(t_1)\,|\rangle_{00}\langle|\, T\hat{c}^\dagger_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\hat{c}_{\mathbf{k_2}s'}(t_2)\,|\rangle_0 \\
&-{}_0\langle|\, T\hat{c}_{\mathbf{p}\sigma}(t)\hat{c}^\dagger_{\mathbf{p}\sigma}(t')\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_1}s}(t_1)\hat{c}^\dagger_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\,|\rangle_{00}\langle|\, T\hat{c}_{\mathbf{k_2}s'}(t_2)\hat{c}^\dagger_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\,|\rangle_0\,.
\end{aligned}
\tag{2.79}
$$

Then as derived, we can then simplify Eq. (2.79) by inserting non-interacting propagators and number operators

$$
\begin{aligned}
&i^3\delta_{\mathbf{p}=\mathbf{k_2}=\mathbf{k_1}+\mathbf{q_1}}\delta_{s=s'=\sigma}G^{(0)}(\mathbf{p},t-t_1)G^{(0)}(\mathbf{p}-\mathbf{q_1},t_1-t_2)G^{(0)}(\mathbf{p},t_2-t') \\
&+i^2\delta_{\mathbf{p}=\mathbf{k_1}=\mathbf{k_2}-\mathbf{q_1}}\delta_{s=s'=\sigma}G^{(0)}(\mathbf{p},t-t_2)G^{(0)}(\mathbf{p}+\mathbf{q_1},t_2-t_1)G^{(0)}(\mathbf{p},t_1-t') \\
&+i^2\delta_{\mathbf{q_1}=\mathbf{0}}\delta_{\mathbf{p}=\mathbf{k_1}}\delta_{s=\sigma}n_F(\epsilon_{\mathbf{k_2}})G^{(0)}(\mathbf{p},t-t_1)G^{(0)}(\mathbf{p},t_1-t') \\
&+i^2\delta_{\mathbf{q_1}=\mathbf{0}}\delta_{\mathbf{p}=\mathbf{k_2}}\delta_{s'=\sigma}n_F(\epsilon_{\mathbf{k_1}})G^{(0)}(\mathbf{p},t-t_2)G^{(0)}(\mathbf{p},t_2-t') \\
&+i\delta_{\mathbf{q_1}=\mathbf{0}}\delta_{\mathbf{q_2}=\mathbf{0}}n_F(\epsilon_{\mathbf{k_1}})n_F(\epsilon_{\mathbf{k_2}})G^{(0)}(\mathbf{p},t-t') \\
&-i^3\delta_{\mathbf{k_1}=\mathbf{k_2}-\mathbf{q_1}}\delta_{s=s'}G^{(0)}(\mathbf{p},t-t')G^{(0)}(\mathbf{k_1},t_1-t_2)G^{(0)}(\mathbf{k_1}+\mathbf{q_1},t_2-t_1).
\end{aligned}
\tag{2.80}
$$

2.2.5 Feynman Diagrams

Richard Feynman introduced the idea of putting the terms of Eq. (2.80) into drawings [8]. The drawings provide a physical interpretation of each term as a particular situation of the particles interacting. The non-interacting fermionic propagator

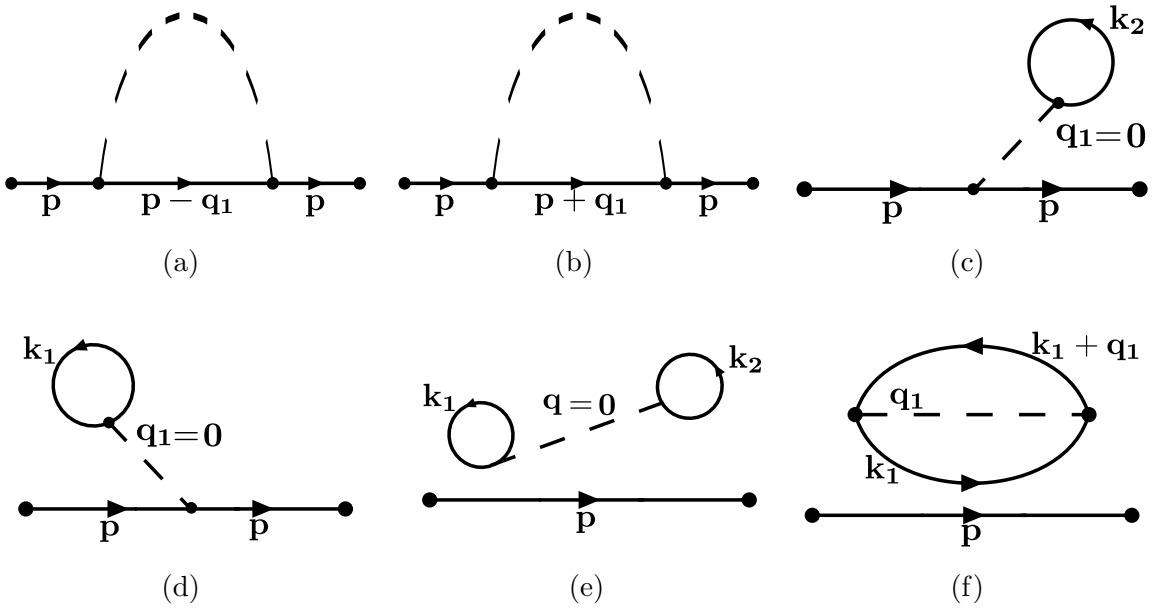


Figure 2.1: First order Feynman diagrams which arise from electron phonon interaction at zero temperature. Each diagram corresponds to a term in Eq. (2.80).

$G^{(0)}(\mathbf{p}, t - t')$ is represented by a solid line with an arrow to denote the time going from t' to t. A phonon's Green's function is represented by a dotted line and does not have an arrow since

$$D^{(0)}(\mathbf{q}, t - t') = D^{(0)}(\mathbf{q}, t' - t), \tag{2.81}$$

so they can be viewed as going in either direction in time. The factor

$$_0\langle| \, c^\dagger_{\mathbf{p}s}(t) c_{\mathbf{p}s}(t) \, |\rangle_0 = n_F(\epsilon_\mathbf{p}) \tag{2.82}$$

is represented by a fermionic propagator that loops back into itself.

By using these rules, we can convert the six terms from Eq. (2.80) into Feynman diagrams as shown in subfigures (a) to (f) in Fig. 2.1. The terms in Figs. (2.1c, 2.1d, 2.1e) are zero since they require the phonon to have $\mathbf{q} = 0$ which implies that the phonon is either a translation of a crystal or strain, but this is not incorporated in our Hamiltonian so we set these to zero. The terms in Figs. (2.1a, 2.1b) are non-zero,

but look very similar. Recall that these terms correspond to

$$\begin{aligned}\frac{1}{2!}\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2\sum_{\mathbf{q}}|M_{\mathbf{q}}|^2D^{(0)}(\mathbf{q},t_1-t_2)\\ \times[G^{(0)}(\mathbf{p},t-t_1)G^{(0)}(\mathbf{p}-\mathbf{q},t_1-t_2)G^{(0)}(\mathbf{p},t_2-t')\\ +G^{(0)}(\mathbf{p},t-t_2)G^{(0)}(\mathbf{p}+\mathbf{q},t_2-t_1)G^{(0)}(\mathbf{p},t_1-t')].\end{aligned} \tag{2.83}$$

We see that these 2 terms are identical under a change of labels. So we may remove the factor of $\frac{1}{2!}$ and use one of the labelling. The term in Fig. 2.1f is also interesting. Here the components of the diagram are topologically disconnected. This leads to the integrals being separable and so the Feynman diagram is evaluated as the product of the disconnected topologies. So it may be written as

$$G^{(0)}(\mathbf{p},t-t')F_1, \tag{2.84}$$

where we define F_1 as

$$\begin{aligned}F_1=-\frac{i}{2}\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2\sum_{\mathbf{kq}}|M_{\mathbf{q}}|^2D^{(0)}(\mathbf{q},t_1-t_2)G^{(0)}(\mathbf{k},t_1-t_2)\\ \times G^{(0)}(\mathbf{k}+\mathbf{q},t_2-t_1).\end{aligned} \tag{2.85}$$

2.2.6 Vacuum Polarization Graphs

Before wrapping up this overview of zero-temperature many-body perturbation theory, we still never dealt with the factor of ${}_0\langle|\, S(\infty,-\infty)\,|\rangle_0$ in the evaluation of the Green's functions

$$\begin{aligned}{}_0\langle|\, S(\infty,-\infty)\,|\rangle_0=\sum_{n=0}^{\infty}\frac{(-i)^n}{n!}\int_{-\infty}^{\infty}dt_1\,\ldots\int_{-\infty}^{\infty}dt_n\\ \times{}_0\langle|\, T\hat{V}(t_1)\hat{V}(t_2)\ldots\hat{V}(t_n)\,|\rangle_0\,.\end{aligned} \tag{2.86}$$

We will take a term by term approach like before. Note that the $n = 1$ term vanishes like in the Green's function expansion. The $n = 2$ term becomes

$$\begin{aligned} {}_0\langle|\, S(\infty,-\infty)\,|\rangle_0 =& 1 + \frac{(-i)^2}{2!}\int_{-\infty}^{\infty} dt_1 \int_{-\infty}^{\infty} dt_2 \\ & \times \sum_{\mathbf{q_1},\mathbf{q_2}} M_{\mathbf{q_1}} M_{\mathbf{q_2}0}\, {}_0\langle|\, T\hat{A}_{\mathbf{q_1}}(t_1)\hat{A}_{\mathbf{q_2}}(t_2)\,|\rangle_0 \\ & \sum_{\mathbf{k_1},\mathbf{k_2}ss'} {}_0\langle|\, T\hat{c}^{\dagger}_{\mathbf{k_1}+\mathbf{q_1}s}(t_1)\hat{c}_{\mathbf{k_1}s}(t_1) \\ & \times \hat{c}^{\dagger}_{\mathbf{k_2}+\mathbf{q_2},s'}(t_2)\hat{c}_{\mathbf{k_2}s'}(t_2)\,|\rangle_0\,. \end{aligned} \tag{2.87}$$

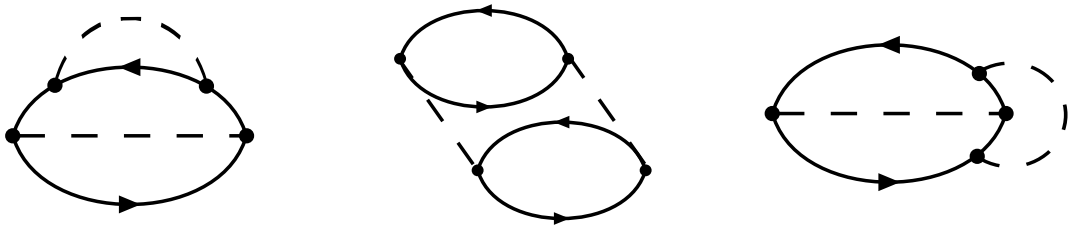
Proceeding with Wick's theorem,

$${}_0\langle|\, T\hat{A}_{\mathbf{q_1}}(t_1)\hat{A}_{\mathbf{q_2}}(t_2)\,|\rangle_0 = i\delta_{\mathbf{q_1}+\mathbf{q_2}} D^{(0)}(\mathbf{q_1}, t_1 - t_2) \tag{2.88}$$

$$\begin{aligned} &{}_0\langle|\, T\hat{c}^{\dagger}_{\mathbf{k_1}+\mathbf{q_1},s}(t_1)\hat{c}_{\mathbf{k_1}s}(t_1)\hat{c}^{\dagger}_{\mathbf{k_2}-\mathbf{q_1},s'}(t_2)\hat{c}_{\mathbf{k_2}s'}(t_2)\,|\rangle_0 \\ &\quad = \delta_{\mathbf{q_1}} n_F(\epsilon_{\mathbf{k_1}}) n_F(\epsilon_{\mathbf{k_2}}) + \delta_{\mathbf{k_1}=\mathbf{k_2}-\mathbf{q_1}} G^{(0)}(\mathbf{k_1}, t_1 - t_2) G^{(0)}(\mathbf{k_1}+\mathbf{q_1}, t_2 - t_1). \end{aligned} \tag{2.89}$$

We have seen the Feynman diagram of Eq. (2.88) before. It is the barbell shape in Fig. 2.1e but now it demands that $\mathbf{q_1} + \mathbf{q_2} = 0$ which again, is not possible, so this term vanishes. We have also seen the Feynman diagram of Eq. (2.89) before, it is the disconnected bubble part in Fig. 2.1f, which in Eq. (2.85), we denoted as F_1. Here we see that F_1 appears whenever the closed bubble occurs, regardless of whether the term arises in the disconnected diagrams of $G(\mathbf{p}, t - t')$ or in the expansion of ${}_0\langle|\, S(\infty,-\infty)\,|\rangle_0$.

The terms in the series for ${}_0\langle|\, S(\infty,-\infty)\,|\rangle_0$ are called vacuum polarization terms. Some diagrams for the $n = 4$ case are shown in Fig. 2.2. Here we see that there is a family of similar topologies to F_1 which we will call F_j (take $F_0 = 1$) so that



(a) (b) (c)

Figure 2.2: Vacuum polarization graphs from the $n = 4$ term of Eq. (2.86).

$$ {}_0\langle|\, S(\infty,-\infty)\,|\rangle_0 = \sum_{j=0}^{\infty} F_j. \tag{2.90} $$

But there is an alternative approach rather than brute forcing this series. The next result is quite useful and will also help simplify the current state of the summation for the Green's functions. The theorem is that the vacuum polarization diagrams exactly cancel the disconnected diagrams in the expansion for $G(\mathbf{p}, t-t')$. This means that we only need to concern ourselves with the calculation of connected Feynman diagrams in all our expansions. This corresponds to

$$ {}_0\langle|\, T\hat{c}_{\mathbf{p}}(t)\hat{c}^{\dagger}_{\mathbf{p}}(t')S(\infty,-\infty)\,|\rangle_0 = G_c(\mathbf{p}, t-t')\,{}_0\langle|\, S(\infty,-\infty)\,|\rangle_0\,, \tag{2.91} $$

where $G_c(\mathbf{p}, t-t')$ is the summation of all connected diagrams.

A quick explanation of the theorem is as follows. In the expansion of the S-Matrix, each connected diagram will have high-order terms consisting of all possible disconnected diagrams tagged onto the original connected diagram as shown in Fig. 2.3. In this partition of the series, the connected diagram can be factored from the disconnected parts of the diagrams leaving precisely $\sum_{j=0}^{\infty} F_j = {}_0\langle|\, S(\infty,-\infty)\,|\rangle_0$ by Eq. (2.90). Doing this for all connected topologies, $G_c(\mathbf{p}, t-t')$, we arrive at Eq. (2.91). This means that in the summation for the Green's functions, we are only

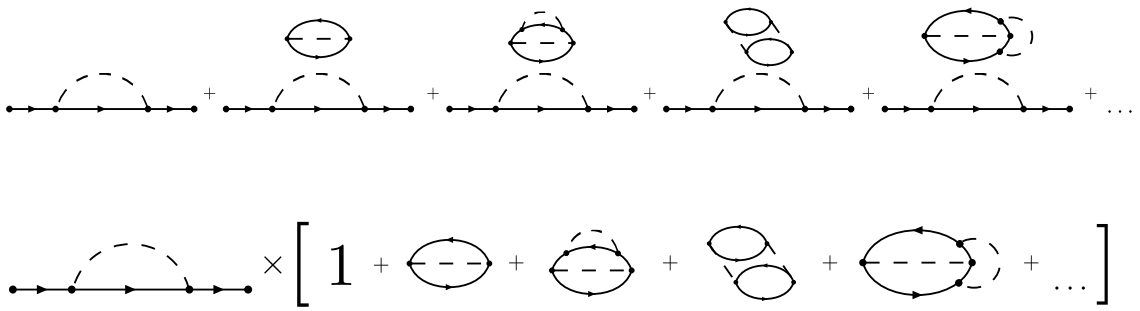


Figure 2.3: Partition of the infinite series in Eq. (2.77) where a connected diagram is multiplied by all possible disconnected diagrams.

concerned with the connected Feynman diagrams

$$G(\mathbf{p}, t-t') = -i\sum_{n=0}^{\infty}\frac{(-i)^n}{n!}\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2\ldots\int_{-\infty}^{\infty}dt_n \times {}_0\langle|\, T\hat{c}_{\mathbf{p}}(t_1)\hat{c}^{\dagger}_{\mathbf{p}}(t')\hat{V}(t_1)\ldots\hat{V}(t_n)\,|\rangle_0 \text{ (connected)}. \tag{2.92}$$

Lastly, since there will be $n!$ identical diagrams which differ by their unique labelling and produce the same answer, we can take any specific labeling and remove the factor of $\frac{1}{n!}$. So we finally arrive at the Green's function at zero temperature given by

$$G(\mathbf{p}, t-t') = -i\sum_{n=0}^{\infty}(-i)^n\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2\ldots\int_{-\infty}^{\infty}dt_n {}_0\langle|\, T\hat{c}_{\mathbf{p}}(t_1)\hat{c}^{\dagger}_{\mathbf{p}}(t') \times \hat{V}(t_1)\ldots\hat{V}(t_n)\,|\rangle_0 \text{ (different connected)}. \tag{2.93}$$

2.3 Non-zero Temperature Green's Functions

Now we will turn to the nonzero temperature case, which luckily is quite similar to the zero temperature case. Now the system will consist of a interacting bath of particles with a non-zero average energy. Due to the sheer number of particles,

the exact configuration of the particles is unknown and, therefore, the exact energy will also be unknown. The best we can do is focus on the average energy which is obtainable since we know the temperature of the system. We will take a Statistical Mechanics approach to average over all possible configurations of the system in a grand canonical ensemble to obtain the non-zero temperature Green's function. We will continue following Gerald Mahan's discussion in Ref. [8].

Recall in the zero temperature case, we only had one state so the Green's function was simply the expectation value of the ground state, Eq. (2.48). Here we will take a grand canonical ensemble average of the system at a inverse temperature $\beta = k_B T^{-1}$

$$\frac{\mathrm{Tr}\, e^{-\beta H} c_{\mathbf{p}\sigma}(t) c^{\dagger}_{\mathbf{p}\sigma}(t')}{\mathrm{Tr}\, e^{-\beta H}} \tag{2.94}$$

$$c_{\mathbf{p}\sigma}(t) = e^{itH} c_{\mathbf{p}\sigma} e^{-itH}. \tag{2.95}$$

Here Tr denotes the trace of an operator or summation over some complete set of states:

$$\mathrm{Tr} = \sum_n \langle n| \ldots |n\rangle \,. \tag{2.96}$$

The issue with Eq. (2.94) is that the Hamiltonian shows up in several places. As before, it shows up in $\exp(\pm iHt)$ but now it also shows up in $\exp(-\beta H)$. So using this definition in its current state would cause a headache with 2 expansions at the same time.

One noteworthy thing about this formula is that the Hamiltonian only appears in exponential factors. Therefore, by interpreting the inverse temperature, β, as a complex time, we see that the two Hamiltonian terms may be replaced with one exponential factor with a complex multiplicative factor on the Hamiltonian. Matsubara [10] took a similar approach but he took time to be a complex temperature so that

t and β are the real and imaginary parts of the complex multiplicative factor on the Hamiltonian. This leaves the Hamiltonian only showing up once in the same fashion as the zero temperature case.

Another motivation for Matsubara's method is using the thermal occupation numbers given by Bose and Fermi-Dirac statistics,

$$n_B(\omega_{\mathbf{q}}) = \frac{1}{e^{\beta\omega_{\mathbf{q}}} - 1} \tag{2.97}$$

$$n_F(\epsilon_{\mathbf{p}} - \mu) = \frac{1}{e^{\beta(\epsilon_{\mathbf{p}} - \mu)} + 1}. \tag{2.98}$$

Using a theorem from complex analysis, we may write the meromorphic functions for these occupation numbers as a summation over their residues evaluated at their poles. Noting that the fermionic case has poles at $\epsilon_{\mathbf{p}} - \mu = \frac{i(2n+1)\pi}{\beta}$ and the bosonic has poles at $\omega_{\mathbf{q}} = \frac{2ni\pi}{\beta}$ for all $n \in \mathbb{Z}$

$$n_F(\epsilon_{\mathbf{p}} - \mu) = \frac{1}{e^{\beta(\epsilon_{\mathbf{p}} - \mu)} + 1} = \frac{1}{2} + \frac{1}{\beta} \sum_{n=-\infty}^{\infty} \frac{1}{(2n+1)i\pi/\beta - \epsilon_{\mathbf{p}} - \mu} \tag{2.99}$$

$$n_B(\omega_{\mathbf{q}}) = \frac{1}{e^{\beta(\epsilon_{\mathbf{p}} - \mu)} - 1} = -\frac{1}{2} + \frac{1}{\beta} \sum_{n=-\infty}^{\infty} \frac{1}{(2n)i\pi/\beta - \omega_{\mathbf{q}}}. \tag{2.100}$$

By letting the poles of each function be $i\omega_n$, the series' have the form

$$\sum_n \frac{1}{i\omega_n - \omega_{\mathbf{q}}} \text{ and } \sum_n \frac{1}{i\omega_n - (\epsilon_{\mathbf{p}} - \mu)}. \tag{2.101}$$

We will see that these are the non-interacting Green's function in the Matsubara method.

Before jumping into Matsubara's complex time approach, we first must recall some useful mathematical relations. We define $\tau = it$ and restrict the Green's functions to

the domain

$$-\beta \leq \tau \leq \beta. \tag{2.102}$$

From Fourier transform theory, if $f(\tau)$ is defined on $-\beta \leq \tau \leq \beta$, its Fourier series expansion is

$$f(\tau) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left[a_n \cos(\frac{n\pi\tau}{\beta}) + b_n \sin(\frac{n\pi\tau}{\beta})\right] \tag{2.103}$$

$$\begin{aligned} a_n &= \frac{1}{\beta}\int_{-\beta}^{\beta} d\tau\, f(\tau)\cos(\frac{n\pi\tau}{\beta}) \\ b_n &= \frac{1}{\beta}\int_{-\beta}^{\beta} d\tau\, f(\tau)\sin(\frac{n\pi\tau}{\beta}). \end{aligned} \tag{2.104}$$

We may also define

$$f(i\omega_n) = \frac{1}{2}\beta(a_n + ib_n) \tag{2.105}$$

and hence

$$f(\tau) = \frac{1}{\beta}\sum_{n=-\infty}^{\infty} e^{-i\pi n\tau/\beta}\, f(i\omega_n) \tag{2.106}$$

$$f(i\omega_n) = \frac{1}{2}\int_{-\beta}^{\beta} d\tau\, e^{i\pi n\tau/\beta}\, f(\tau). \tag{2.107}$$

To further simplify we can use the fact that the bosonic Green's functions have the property

$$\text{bosons: } f(\tau) = f(\tau+\beta) \text{ for } -\beta < \tau < 0 \text{ (and } 0 < \tau+\beta < \beta). \tag{2.108}$$

So then we may rearrange the integral, noting that $f(i\omega_n) = 0$ for odd n, to find for bosons

$$f(i\omega_n) = \int_0^{\beta} d\tau e^{i\omega_n\tau} f(\tau) \tag{2.109}$$

$$f(\tau) = \frac{1}{\beta}\sum_{n} e^{-i\omega_n\tau} f(i\omega_n) \tag{2.110}$$

$$\omega_n = 2n\pi k_B T. \tag{2.111}$$

Similarly, for fermions we have

$$\text{fermions: } f(\tau) = -f(\tau+\beta) \text{ for } -\beta < \tau < 0. \tag{2.112}$$

Then $f(i\omega_n) = 0$ for even n and rearranging the integral gives

$$f(i\omega_n) = \int_0^\beta d\tau e^{i\omega_n \tau} f(\tau) \tag{2.113}$$

$$f(\tau) = \frac{1}{\beta}\sum_n e^{-i\omega_n \tau} f(i\omega_n) \tag{2.114}$$

$$\omega_n = (2n+1)\pi k_B T. \tag{2.115}$$

So the two cases are identical except for the permitted parity of integers in the formulae. Bosons only have even integers while fermions have odd values.

2.3.1 Matsubara Green's Functions

The fermionic Matsubara Green's function is defined as

$$\mathcal{G}(\mathbf{p}, \tau - \tau') = -\langle T_\tau c_{\mathbf{p}\sigma}(\tau) c^\dagger_{\mathbf{p}\sigma}(\tau')\rangle \tag{2.116}$$

$$\begin{aligned}\mathcal{G}(\mathbf{p}, \tau - \tau') = &- \mathrm{Tr}[e^{-\beta(H-\mu N-\Omega)} T_\tau e^{\tau(H-\mu N)} c_{\mathbf{p}\sigma} e^{-(\tau-\tau')(H-\mu N)} \\ &\times c^\dagger_{\mathbf{p}\sigma} e^{-\tau'(H-\mu N)}]\end{aligned} \tag{2.117}$$

$$e^{-\beta\Omega} = \mathrm{Tr}(e^{-\beta(H-\mu N)}). \tag{2.118}$$

Here $\langle \ldots \rangle$ is the grand canonical ensemble average. We denote Ω as the thermodynamic potential used in the average. Now the Hamiltonian is replaced by $K = H - \mu N$,

where μ is the chemical potential and N is the number operator. Since we are working in complex time, the creation and annihilation operators are replaced with analogous Heisenberg representations with t replaced with $t = \tau/i$ and the time ordering operator is now τ ordered, T_τ.

Apart from working with the grand canonical ensemble average, the workings are mainly the same as the zero temperature case. We find the Green's function's temporal dependence to be solely on the difference $\tau - \tau'$. So we may write the Green's function only in terms of τ

$$\begin{aligned} \mathcal{G}(\mathbf{p},\tau) &= -\langle T_\tau c_{\mathbf{p}\sigma}(\tau) c^\dagger_{\mathbf{p}\sigma}(0)\rangle \\ &= -\operatorname{Tr}\left[e^{-\beta(K-\Omega)} T_\tau (e^{\tau K} c_{\mathbf{p}\sigma} e^{-\tau K} c^\dagger_{\mathbf{p}\sigma})\right]. \end{aligned} \tag{2.119}$$

Next, we confirm the fermionic property in Eq. (2.112)

$$\tau < 0: \quad \mathcal{G}(\mathbf{p},\tau) = \operatorname{Tr}\left[e^{-\beta(K-\Omega)} c^\dagger_{\mathbf{p}\sigma} e^{\tau K} c_{\mathbf{p}\sigma} e^{-\tau K}\right]. \tag{2.120}$$

Using the cyclic property of the trace and noting that $e^{\beta\Omega}$ is a scalar, we can group terms from the time ordering to find

$$\tau < 0: \quad \mathcal{G}(\mathbf{p},\tau) = \operatorname{Tr}\left[e^{-\beta(K-\Omega)} e^{(\tau+\beta)K} c_{\mathbf{p}\sigma} e^{-(\tau+\beta)K} c^\dagger_{\mathbf{p}\sigma})\right]. \tag{2.121}$$

We see the term on the right is $-\mathcal{G}(\mathbf{p},\tau+\beta)$ when $0 < \tau + \beta < \beta$ which implies

$$-\beta < \tau < 0: \quad \mathcal{G}(\mathbf{p},\tau) = -\mathcal{G}(\mathbf{p},\tau+\beta). \tag{2.122}$$

As noted earlier, this property then allows us to expand $\mathcal{G}(\mathbf{p},\tau)$ in a Fourier series as

in Eq. (2.114). So we may write

$$\mathcal{G}(\mathbf{p}, i\omega_n) = \int_0^\beta d\tau\, e^{i\omega_n \tau} \mathcal{G}(\mathbf{p}, \tau) \tag{2.123}$$

$$\mathcal{G}(\mathbf{p}, \tau) = \frac{1}{\beta} \sum_n e^{-i\omega_n \tau} \mathcal{G}(\mathbf{p}, i\omega_n). \tag{2.124}$$

The non-interacting Green's function is obtained using the Hamiltonian

$$H = H_0 = \sum_{\mathbf{p}\sigma} \epsilon_{\mathbf{p}} c^\dagger_{\mathbf{p}\sigma} c_{\mathbf{p}\sigma} \tag{2.125}$$

$$K = K_0 = \sum_{\mathbf{p}\sigma} \xi_{\mathbf{p}} c^\dagger_{\mathbf{p}\sigma} c_{\mathbf{p}\sigma}, \tag{2.126}$$

where $\xi_{\mathbf{p}} = \epsilon_{\mathbf{p}} - \mu$. We find the τ evolution of the operators

$$c_{\mathbf{p}\sigma}(\tau) = e^{\tau K_0} c_{\mathbf{p}\sigma} e^{-\tau K_0} = e^{-\xi_{\mathbf{p}} \tau} c_{\mathbf{p}\sigma} \tag{2.127}$$

$$c^\dagger_{\mathbf{p}\sigma}(\tau) = e^{\tau K_0} c^\dagger_{\mathbf{p}\sigma} e^{-\tau K_0} = e^{\xi_{\mathbf{p}} \tau} c^\dagger_{\mathbf{p}\sigma}. \tag{2.128}$$

Here we used the Baker-Hausdorff theorem

$$e^A C e^{-A} = C + [A, C] + \frac{1}{2!}[A, [A, C]] + \frac{1}{3!}[A, [A, [A, C]]] + \dots\ . \tag{2.129}$$

This leads to the non-interacting Green's function

$$\begin{aligned} \mathcal{G}^{(0)}(\mathbf{p}, \tau) &= -\Theta(\tau) e^{-\xi_{\mathbf{p}} \tau} \langle c_{\mathbf{p}\sigma} c^\dagger_{\mathbf{p}\sigma} \rangle + \Theta(-\tau) e^{-\xi_{\mathbf{p}} \tau} \langle c^\dagger_{\mathbf{p}\sigma} c_{\mathbf{p}\sigma} \rangle \\ &= -e^{-\xi_{\mathbf{p}} \tau} [\Theta(\tau)(1 - n_F(\xi_{\mathbf{p}})) - \Theta(-\tau) n_F(\xi_{\mathbf{p}})] \\ &= -e^{-\xi_{\mathbf{p}} \tau} [\Theta(\tau) - n_F(\xi_{\mathbf{p}})]. \end{aligned} \tag{2.130}$$

Using the fact that in the grand canonical ensemble, the expectation value of the

number operator is the Fermi distribution function

$$\langle c^{\dagger}_{\mathbf{p}\sigma} c_{\mathbf{p}\sigma} \rangle = n_F(\xi_{\mathbf{p}}) = \frac{1}{e^{\beta\xi_{\mathbf{p}}} + 1}. \tag{2.131}$$

Integrating Eq. (2.123), we obtain

$$\mathcal{G}^{(0)}(\mathbf{p}, i\omega_n) = -\frac{(1 - n_F(\xi_{\mathbf{p}}))(e^{\beta(i\omega_n - \xi_{\mathbf{p}})} - 1)}{i\omega_n - \xi_{\mathbf{p}}}. \tag{2.132}$$

Then recalling the fermionic frequencies property,

$$i\beta\omega_n = i(2n+1)\pi \implies e^{i\beta\omega_n} = -1 \tag{2.133}$$

so we find

$$\mathcal{G}^{(0)}(\mathbf{p}, i\omega_n) = \frac{1}{i\omega_n - \xi_{\mathbf{p}}} = \frac{1}{i\omega_n - \epsilon_{\mathbf{p}} + \mu}. \tag{2.134}$$

Which was the predicted formula obtained using the residue composition of the Fermi distribution in Eq. (2.101).

2.3.2 Retarded and Advanced Green's Functions

Up to now, we have looked at Green's functions $G(\mathbf{p}, t - t')$ that are applicable for both cases of time, $t > t'$ and $t < t'$. Since each Green's function is like a response function, it is not physical to allow the latter case. To deal with this, in all physical applications, we break the Green's functions into two parts called the retarded and advanced Green's functions. These are defined to be the regular Green's functions but with a Heaviside step function multiplied to enforce that the retarded Green's function applies to $t > t'$ and the advanced Green's function applies to the case $t < t'$. They come into play as all measurable quantities, such as susceptibilities and

conductivities, are related to retarded Green's functions. There are several ways to go about obtaining these Green's functions, one is using real time Green's functions at non-zero temperature but it is much more efficient to work in the Matsubara formalism and convert back after. It turns out that this process is very simple, we take the Matsubara Green's function and replace $i\omega_n$ by $\omega + i0^+$ in a procedure known as analytic continuation.

The retarded Green's function may be defined for all temperatures as

$$\begin{aligned} G_{\text{ret}}(\mathbf{p}, t-t') &= -i\Theta(t-t')\langle[c_{\mathbf{p}\sigma}(t)c^\dagger_{\mathbf{p}\sigma}(t') + c^\dagger_{\mathbf{p}\sigma}(t')c_{\mathbf{p}\sigma}(t)]\rangle \\ &= -i\Theta(t-t')\,\text{Tr}\,\{e^{-\beta(K-\Omega)}[c_{\mathbf{p}\sigma}(t)c^\dagger_{\mathbf{p}\sigma}(t') + c^\dagger_{\mathbf{p}\sigma}(t')c_{\mathbf{p}\sigma}(t)]\}. \end{aligned} \tag{2.135}$$

Here we see the Heaviside factor of the retarded Green's function in action. That is, it is only active for $t > t'$, which makes it causal. This is analogous to a signal started at time t' and it being measured at $t > t'$. Which is of course the physically correct order of events.

The majority of the operators that we look at in this thesis are from the Hubbard model in Sec. 2.4. We will see that the Hamiltonian consists of terms which look like

$$U = \sum_{ij} M_{ij} C_i^\dagger C_j. \tag{2.136}$$

The operator U is bilinear in the operators C_i and M_{ij} is just some matrix element. This operator is regarded as having bosonic properties, regardless of being built of fermionic or bosonic operators, C (as long as they are both are the same). U is said to be bosonic since it acts as a composite particle. We define a retarded Green's function for this particle

$$\bar{U}_{\text{ret}}(t-t') = -i\Theta(t-t')\langle[U(t)U^\dagger(t') - U^\dagger(t')U(t)]\rangle. \tag{2.137}$$

This Green's function looks similar to the fermionic retarded Green's function in Eq. (2.135) except for the minus sign, which is the case for all bosonic retarded Green's functions. Also, the Fourier transforms are given by

$$G_{\rm ret}(\mathbf{p}, E) = \int_{-\infty}^{\infty} dt\, e^{iE(t-t')} G_{\rm ret}(\mathbf{p}, t-t') \tag{2.138}$$

$$\bar{U}_{\rm ret}(\omega) = \int_{-\infty}^{\infty} dt\, e^{i\omega t} \bar{U}_{\rm ret}(t). \tag{2.139}$$

The advanced Green's functions are defined similarly to the retarded ones

$$G_{\rm adv}(\mathbf{p}, t-t') = i\Theta(t'-t)\langle[c_{\mathbf{p}\sigma}(t)c^{\dagger}_{\mathbf{p}\sigma}(t') + c^{\dagger}_{\mathbf{p}\sigma}(t')c_{\mathbf{p}\sigma}(t)]\rangle \tag{2.140}$$

$$\bar{U}_{\rm adv}(t-t') = i\Theta(t'-t)\langle[U(t)U^{\dagger}(t') - U^{\dagger}(t')U(t)]\rangle. \tag{2.141}$$

From these definitions, we can show that the advanced functions turn out to be the complex conjugate of the corresponding retarded functions. First, the advanced function's Hermitian conjugate can be seen to be the retarded function

$$\bar{U}_{\rm adv}(t'-t)^{\dagger} = -i\Theta(t-t')\langle[U(t)U^{\dagger}(t') - U^{\dagger}(t')U(t)]\rangle = \bar{U}_{\rm ret}(t-t'). \tag{2.142}$$

Then comparing Fourier transforms

$$\bar{U}_{\rm ret}(\omega) = \int_{-\infty}^{\infty} dt\, e^{i\omega(t-t')} \bar{U}_{\rm adv}(t'-t)^{\dagger} = \int_{-\infty}^{\infty} dt_1\, e^{-i\omega t_1} \bar{U}_{\rm adv}(t_1)^{\dagger}, \tag{2.143}$$

and changing the integration variables from $t_1 = t' - t$ leads us to conclude

$$\bar{U}_{\rm ret}(\omega) = \bar{U}^{*}_{\rm adv}(\omega). \tag{2.144}$$

This result holds for all retarded and advanced Green's function pair, meaning once one function is obtained, the other is revealed via a complex conjugation.

Now to see how these retarded Green's functions are expressed in the Matsubara formalism, we take a Statistical Mechanics type of argument for a general representation of the system. Here we assume there exists a complete set of states, $|m\rangle$, which are exact eigen-states of $K = H - \mu N$. We do not need specifics of the states, only that they exist with corresponding eigenvalues E_m

$$K\,|m\rangle = E_m\,|m\rangle\,. \tag{2.145}$$

This complete set of states will allow us to evaluate the thermodynamic average as in Eq. (2.96).

$$\bar{U}_{\rm ret}(t-t') = -i\Theta(t-t')e^{\beta\Omega}\sum_n \langle n|\, e^{-\beta K}[U(t)U^\dagger(t') - U^\dagger(t')U(t)]\,|n\rangle\,. \tag{2.146}$$

Inserting unity between the U operators

$$\begin{aligned}\bar{U}_{\rm ret}(t-t') = -i\Theta(t-t')e^{\beta\Omega}\sum_{m,n} e^{-\beta E_n}[&\langle n|\, U(t)\,|m\rangle\,\langle m|\, U^\dagger(t')\,|n\rangle \\ &- \langle n|\, U^\dagger(t')\,|m\rangle\,\langle m|\, U(t)\,|n\rangle].\end{aligned} \tag{2.147}$$

Evaluating the terms like

$$\langle n|\, U(t)\,|m\rangle = \langle n|\, e^{iKt}Ue^{-iKt}\,|m\rangle = \langle n|\, U\,|m\rangle\, e^{it(E_n-E_m)}. \tag{2.148}$$

We get the following by rearranging the dummy variables

$$\begin{aligned}\bar{U}_{\text{ret}}(t-t') &= -i\Theta(t-t')e^{\beta\Omega}\sum_{m,n}e^{-\beta E_n}[e^{i(t-t')(E_n-E_m)}|\langle n|U(t)|m\rangle|^2 \\ &\quad - e^{-i(t-t')(E_n-E_m)}|\langle m|U(t')|n\rangle|^2] \\ &= -i\Theta(t-t')e^{\beta\Omega}\sum_{m,n}|\langle n|U|m\rangle|^2 e^{i(t-t')(E_n-E_m)}[e^{-\beta E_n}-e^{-\beta E_m}].\end{aligned} \tag{2.149}$$

Taking the Fourier transform to get the frequency function,

$$\begin{aligned}\bar{U}_{\text{ret}}(\omega) &= -i\int_0^\infty e^{it(\omega+i\Gamma)}dt\, e^{\beta\Omega}\sum_{m,n}|\langle n|U|m\rangle|^2 e^{it(E_n-E_m)}[e^{-\beta E_n}-e^{-\beta E_m}] \\ &= e^{\beta\Omega}\sum_{m,n}|\langle n|U|m\rangle|^2\frac{e^{-\beta E_n}-e^{-\beta E_m}}{\omega+E_n-E_m+i\Gamma}.\end{aligned} \tag{2.150}$$

Here $i\Gamma$ is added to the frequency to ensure convergence at large times ($\Gamma \to 0^+$).

Now we take equivalent Matsubara function for the operator U defined as $\mathcal{U}$

$$\mathcal{U}(\tau) = -\langle T_\tau U(\tau)U^\dagger(0)\rangle \tag{2.151}$$

$$\mathcal{U}(i\omega_n) = \int_0^\beta d\tau e^{i\omega_n\tau}\mathcal{U}(\tau). \tag{2.152}$$

Using the $|n\rangle$ representation for $\tau > 0$ we have

$$\begin{aligned}\tau > 0:\ \mathcal{U}(\tau) &= -e^{\beta\Omega}\sum_{n,m}\langle n|e^{-\beta K}U(\tau)|m\rangle\langle m|U^\dagger(0)|n\rangle \\ \mathcal{U}(\tau) &= -e^{\beta\Omega}\sum_{n,m}|\langle n|U|m\rangle|^2 e^{-\beta E_n}e^{\tau(E_n-E_m)}.\end{aligned} \tag{2.153}$$

Then taking the frequency transform

$$\begin{aligned}\mathcal{U}(i\omega_n) &= -e^{\beta\Omega}\sum_{n,m}|\langle n|U|m\rangle|^2 e^{-\beta E_n}\int_0^\beta d\tau\, e^{i\omega_n\tau}e^{\tau(E_n-E_m)} \\ &= e^{\beta\Omega}\sum_{n,m}|\langle n|U|m\rangle|^2\frac{e^{-\beta E_n}-e^{-\beta E_m}}{i\omega+E_n-E_m}.\end{aligned} \tag{2.154}$$

Here again, $\exp(\beta i\omega_n) = 1$ for bosons. Now comparing Eqs. (2.150, 2.154), we see that they only differ by the frequencies in the denominator of the integrands as the Matsubara method has $i\omega_n$ and the retarded function has $\omega + i\Gamma$. So the Matsubara function can be changed to a retarded one with the alteration:

$$\mathcal{U}(i\omega_n \to \omega + i\Gamma) = \bar{U}_{\rm ret}(\omega). \tag{2.155}$$

This alteration is called an analytic continuation, the same process can be done for the other Green's functions

$$\mathcal{G}(\mathbf{p}, i\omega_n \to \omega + i\Gamma) = G_{\rm ret}(\mathbf{p}, \omega). \tag{2.156}$$

This very simple relation is what makes the Matsubara formalism so useful. All the physical quantities that are found through the retarded Green's functions now may be obtained by analytically continuing the Matsubara Green's function.

Also note that the advanced Green's functions may also be obtained by an analytic continuation $i\omega_n \to \omega - i\Gamma$ and $\Gamma \to 0^+$ as before. Note that this is expected since the retarded and advanced Green's functions are complex conjugates (Eq. (2.144)).

Now having introduced the Green's functions, we will now describe the 2D Hubbard model, where we will perform sample calculations in this thesis.

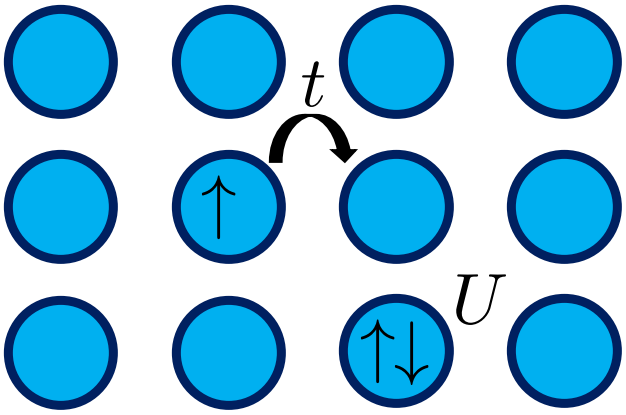


Figure 2.4: Two dimensional square lattice depicting hopping energy t from a lattice site to a neighbouring site and the onsite interaction U for filled sites as modelled in the two dimensional Hubbard model.

2.4 The Hubbard Model

The Hubbard model [11] is a simplistic model of a correlated electron system. It was formulated by Hubbard in 1963, originally created to approximate properties of transition and rare earth metals with their partially filled conduction bands [11]. This model has often been referred to as a cornerstone of condensed matter physics [12] as it has since been used in all sorts of applications where a similar band filling to the transition and rare earth metals occurs. While this model has only been solved in one and infinite dimensions, this model has been of great interest in numerical approaches to acquire approximate solutions [2, 12]. In this model, shown in Fig. 2.4, the electrons are confined to a lattice where they are permitted to hop to neighbouring sites with energy t and have an onsite interaction U when a lattice site is occupied by both an up and down spin. Specifically to this thesis, calculations will be done using the single band Hubbard model on a two dimensional square lattice with Hamiltonian

$$H = \underbrace{\sum_{ij\sigma} t_{ij}(c^{\dagger}_{i\sigma}c_{j\sigma} + c^{\dagger}_{j\sigma}c_{i\sigma})}_{H_0} + \underbrace{U\sum_{i} n_{i\uparrow}n_{i\downarrow}}_{H_v}. \tag{2.157}$$

Here, only nearest neighbour hopping is permitted and each direction has the same energy, $t_{ij} = t$. As before, $c^\dagger_{i,\sigma}$ ($c_{i,\sigma}$) are the creation (annihilation) operators at lattice site i and the spin, $\sigma \in \{\uparrow, \downarrow\}$ and $n_{i\sigma}$ is the number operator.

In this work, we use this model strictly as a testing case for numerical approaches to evaluating Feynman diagrams. Further to this, we will work in units of hopping energy $t = 1$ and consider the half filled problem, $\mu = 0$, which is most computationally challenging with perturbative methods.

From the Hubbard Hamiltonian Eq. (2.157), we see that the parts are labeled

$$H_0 = \sum_{ij\sigma} t_{ij}(c^\dagger_{i\sigma}c_{j\sigma} + c^\dagger_{j\sigma}c_{i\sigma}) \tag{2.158}$$

$$H_v = U\sum_i n_{i\uparrow}n_{i\downarrow}. \tag{2.159}$$

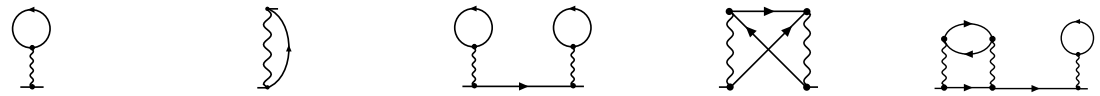
As described in section 2.2.3 this is the Hamiltonian split into a known Hamiltonian, H_0, and an interaction or perturbation, H_v. The known Hamiltonian leads to the general non-interacting Matsubara Green's functions

$$G_0^{-1}(k, i\omega_n) = i\omega_n - \epsilon_k + \mu. \tag{2.160}$$

Here $\epsilon_{\mathbf{k}}$ is the dispersion of the 2D tight-binding model given by

$$\epsilon_{\mathbf{k}} = -2t(\cos(k_x) + \cos(k_y)). \tag{2.161}$$

This dispersion can be derived by finding the eigenvalues of H_0 as in [13]. H_v determines the details of the Feynman diagrams that are summed in the expansion of the S-matrix. Recall from Sec. 2.3.2, H_v is composed of an even number of fermionic operators and so it is regarded as having bosonic properties. For this reason, the



(a) (b) (c) (d) (e)

Figure 2.5: Examples of self-energy parts. Note the stumps on each component of a Feynman diagrams showing where they would be attached to external propagators.

Feynman diagrams in the 2D Hubbard model will have interactions that are depicted by wavy bosonic lines, see Fig. 2.5. Further in this model, each bosonic line will correspond to a constant interaction U in the integral representation of each Feynman diagram.

2.5 Dyson's Equation

One question that can be made about all the diagrams shown so far is "What stops each propagator from interacting with itself while in transit to another interaction?". The answer is nothing. In fact, all Feynman diagrams will have a corresponding infinite series of Feynman diagrams where itself is repeated on the internal propagators. These diagrams are contained in the full perturbation expansion of the S-Matrix and should be summed. This is exploited in what is known as the Dyson's equation [1].

Like the vacuum polarization diagrams, we will look at diagrams that are constructed from simpler diagrams. First, let us define a few components of the topology of Feynman diagrams.

We define the self-energy part of a diagram to be a diagram without external (i.e., incoming and outgoing) lines, which can be inserted into a propagator. Some examples are shown in Fig. 2.5, note we are using wavy lines for the bosonic interactions as in the Hubbard model. We define the proper self-energy part or irreducible self-energy

part to be a self-energy part that cannot be broken into two unconnected self-energy parts by removing a propagator. Some examples are shown in Fig. 2.6.

(a) (b) (c) (d)

Figure 2.6: Examples of proper self-energy parts. Note the stumps on each part, showing where they would be attached to external propagators.

So we see that the self-energy parts in Figs. 2.5c, 2.5e are non-Proper self-energy parts since we can break one propagator and recover two self-energy parts.

Returning to the observation that any propagator could interact with itself while in transit to the next interaction, this would be possible for every self-energy part. But as seen in Figs. 2.5c and 2.5e we can construct the self-energy parts with the irreducible self-energy parts. We define Σ to be the sum of all proper self-energy parts. One can then convince themselves that every possible Feynman diagram, G, will be obtained in the series

$$G = G_0 + G_0\Sigma G_0 + G_0\Sigma G_0\Sigma G_0 + \dots \ . \tag{2.162}$$

That is, all possible Feynman diagrams can be generated by combinations of proper self-energy parts connected by non-interacting Green's functions. This is known as Dyson's equation. Factoring this expression we find

$$\begin{aligned} G &= G_0 + G_0\Sigma(G_0 + G_0\Sigma G_0 + \dots) \\ &= G_0 + G_0\Sigma G. \end{aligned} \tag{2.163}$$

Note that there is an implied integration on all of these terms, and so the factoring process is much more complicated than it is written here. The factoring process is

similar to the vacuum polarization diagrams earlier. But now we may need to swap integration variables in order separate integrals to factor. This has been proved that is this possible [8]. Eq. (2.163) has a solution

$$G(k, i\omega_n) = \frac{G_0}{1 - G_0\Sigma} = \frac{1}{G_0^{-1} - \Sigma} = \frac{1}{i\omega_n - \epsilon_k + \mu - \Sigma}. \tag{2.164}$$

So we can express G by evaluating Σ. But of course, this is no better than before since there is still an infinite set of diagrams that need to be evaluated. In the Hubbard model, with onsite interaction U, we find

$$\Sigma(k, i\omega_n) = \sum_{\ell=0}^{\infty} a_\ell \, U^\ell, \tag{2.165}$$

where ℓ denotes the order of proper self-energy part, a_ℓ denotes all Feynman diagrams of order ℓ and the weight U^ℓ is the interaction described in Sec. 2.4.

Note that G has an expression that is the same as G_0 but with a shifted energy by Σ. This is why it is called the self-energy, Σ is essentially a shift in energy to account for all interactions.

At this point, we take approximate values for Σ. Typically, one only needs a few orders of diagrams for a good approximation [14]. Using this as motivation, in this thesis, we will focus on calculating self-energy diagrams.

2.6 Diagrammatic Techniques

Now having introduced Feynman diagrams at all temperatures and shown the powerful Matsubara formalism, let us quickly recap how we will evaluate Feynman diagrams.

As seen in the terms of Eq. (2.77), in the evaluation of Feynman diagrams we

must integrate over all the internal degrees of freedom. As described in Sec. 2.3.2 the physically relevant retarded Green's functions are most easily obtained by using the Matsubara formalism. Because of this, we will use this formalism to evaluate Feynman diagrams and, therefore, we will need a method to evaluate the summation over all Matsubara frequencies.

As mentioned before, in this thesis, we will work with the Matsubara formalism with the two dimensional Hubbard model. To solidify the comments made in Sec. 2.4 regarding Feynman diagrams in the two dimensional Hubbard model, we provide a short dictionary of the relevant components of a Feynman diagram to translate between the pictorial and mathematical representations in Table 2.1.

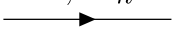
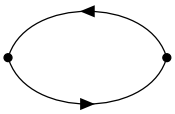
Diagram Component	Mathematical Representation	Name
	$\mathcal{G}^{(0)}(\mathbf{k}, i\omega_n) = \dfrac{1}{i\omega_n - \epsilon_{\mathbf{k}} + \mu}$	Non-interacting Green's function
	U	Bosonic interaction
	(-1)	Fermionic loop
$(\mathbf{k}, i\omega_n)$	$\displaystyle\sum_{\mathbf{k}} \equiv \int \frac{d^2\mathbf{k}}{(2\pi)^2},\ \frac{1}{\beta}\sum_{n=-\infty}^{\infty}$	Independent label

Table 2.1: Dictionary of the relevant diagrammatic components in the two dimensional Hubbard model using the Matsubara formalism. This table is similar to dictionaries found in Mattuck [1].

As an example of how we evaluate the frequency summations in the Matsubara formalism, we will evaluate the labeled second order self-energy diagram shown in Fig. 2.7. This will serve two purposes: we will see how this summation is evaluated and it will provide an analytical expression to test our code against later on. First, we

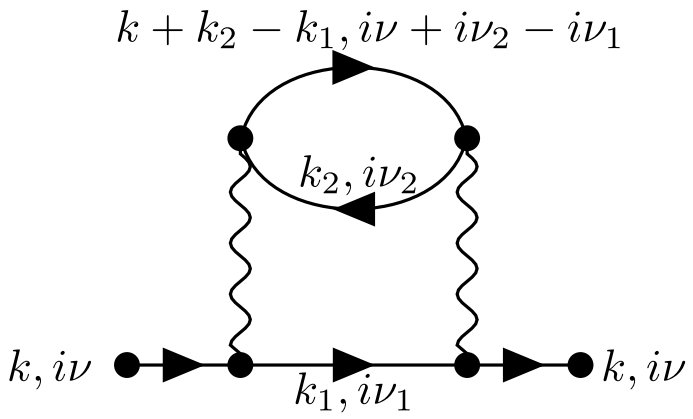


Figure 2.7: Second order self-energy Feynman diagram. Labeling is specified to match the indices in Eq. 2.166.

assign momentum and frequency conserving labels to all fermionic lines in the diagram as labeled in Fig. 2.7. It is important to note that this labeling is not unique, there are multiple ways to label the Green's functions in the diagram. However, after all the integrals are completed, the value determined will be the same. Once labeled, we can form the expression for the self-energy diagram using the Dictionary in Table 2.1.

$$\Sigma^{(2)}(k, i\nu) = \frac{-U^2}{\beta^2} \sum_{\{k_i\}} \sum_{\{i\nu_n\}} \frac{1}{i\nu_1 - \epsilon_{k_1}} \frac{1}{i\nu_2 - \epsilon_{k_2}} \frac{1}{i\nu + i\nu_2 - i\nu_1 - \epsilon_{k+k_2-k_1}}. \tag{2.166}$$

As an illustration, we will walk through the process of building this expression. Here we have two independent internal labels, $(k_1, i\nu_1)$ and $(k_2, i\nu_2)$ leading to the double summation written in the short hand notation

$$\frac{1}{\beta^2} \sum_{\{k_i\}} \sum_{\{i\nu_n\}}. \tag{2.167}$$

Next, we write the product of the non-interacting Green's functions for each fermionic

line with their respective momenta and frequency

$$\frac{1}{i\nu_1 - \epsilon_{k_1}} \frac{1}{i\nu_2 - \epsilon_{k_2}} \frac{1}{i\nu + i\nu_2 - i\nu_1 - \epsilon_{k+k_2-k_1}}. \tag{2.168}$$

Lastly, we denote the two bosonic lines with each giving a factor of U and the fermionic loop giving a factor of -1 to arrive at Eq. (2.166).

Next, we will evaluate the frequency summations in this expression. First, we will look at

$$\frac{1}{\beta} \sum_{i\nu_1} \frac{1}{i\nu_1 - \epsilon_{k_1}} \frac{1}{i\nu_2 - \epsilon_{k_2}} \frac{1}{i\nu + i\nu_2 - i\nu_1 - \epsilon_{k+k_2-k_1}} \equiv \frac{1}{\beta} \sum_{i\nu_1} H(i\nu_1). \tag{2.169}$$

A key observation is that this summation over the fermionic Matsubara frequencies is the sum of H evaluated at the poles of of the Fermi-Dirac distribution $f(\omega)$ (Eq. (2.98)). Recall that the poles of $f(\omega)$ are when

$$e^{\beta\omega} = -1 \implies \omega = \frac{(2n+1)\pi i}{\beta} = i\omega_n. \tag{2.170}$$

These have corresponding residues

$$\mathrm{Res}(f; i\omega_n) = \lim_{\omega \to i\omega_n} (\omega - i\omega_n) \frac{1}{e^{\beta\omega} + 1} \overset{\text{H}}{=} \lim_{\omega \to i\omega_n} \frac{1}{\beta e^{\beta\omega}} = \frac{-1}{\beta}. \tag{2.171}$$

Since $H(\omega)$ is analytic on the imaginary ω axis, we can say that the evaluation of $H(\omega)$ at the poles is the residue of $H(\omega)$ at the poles. Further, since it is analytic, we may write

$$\frac{1}{\beta} H(i\nu_1 = i\omega_n) = \frac{1}{\beta} \mathrm{Res}(H; i\omega_n) = -\mathrm{Res}(H; i\omega_n) \mathrm{Res}(f; i\omega_n) = -\mathrm{Res}(H(\omega) f(\omega); i\omega_n). \tag{2.172}$$

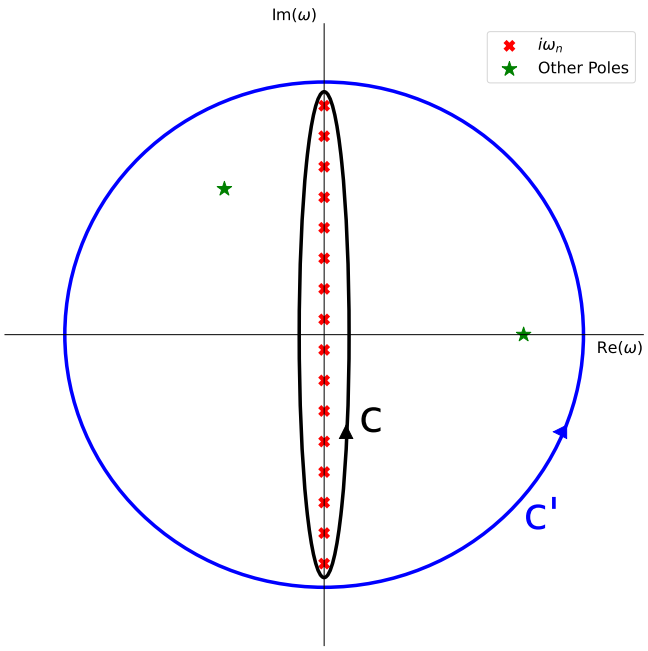


Figure 2.8: Complex plane of ω including poles of the frequency integrand and proposed contours C in black and C' in blue. Also shown are the fermionic Matsubara poles in red crosses and the remaining poles from the product of Propagators in green stars. Note that this figure is truncated. The Matsubara poles are in one to one correspondence with $\mathbb{Z}$ and, therefore, the contours are also infinite in length.

Putting this into Eq. (2.169), we see that Matsubara summation is the sum of residues for the product of $f(\omega)H(\omega)$ evaluated at the poles of f. Of course, the sum of the residues is the result of the residue theorem to evaluate the contour integral which encloses the fermionic Matsubara frequencies on the imaginary axis

$$\frac{1}{\beta}\sum_{i\nu_1} H(i\nu_1) = -2\pi i \sum_{n=-\infty}^{\infty} \text{Res}\left(\frac{H(\omega)f(\omega)}{2\pi i}; i\omega_n\right) = -\frac{1}{2\pi i}\oint_C d\omega\, f(\omega)H(\omega). \quad (2.173)$$

Here C is a contour which only encloses the imaginary axis of the complex ω plane seen in Fig. 2.8. Also shown in Fig. 2.8, we consider another contour C' which encloses

both the imaginary ω axis and the other poles from the product of Green's functions. We denote the set of extra poles $\{\omega_p\}$

$$\{\omega_p\} = \text{Poles}(H) = \{\epsilon_{k_1}, i\nu + i\nu_2 - \epsilon_{k+k_2-k_1}\}. \tag{2.174}$$

Using the new contour we may write

$$-\frac{1}{2\pi i}\oint_{C'} d\omega\, H(\omega)f(\omega) = -\frac{1}{2\pi i}\oint_{C} d\omega\, H(\omega)f(\omega) - \sum_{\omega'\in\{\omega_p\}} \text{Res}(H(\omega)f(\omega);\omega'). \tag{2.175}$$

Since the contour C' encloses all of the fermionic Matsubara frequencies, it must be infinite in length. To deal with this, we further define C' to be a circular contour of radius R and then take $R \to \infty$. Now we are in a position to use Jordan's lemma from complex analysis where we take $\omega = Re^{i\theta}$ so

$$\oint_{C'} d\omega\, H(\omega)f(\omega) = \lim_{R\to\infty}\int d(Re^{i\theta})H(Re^{i\theta})f(Re^{i\theta}). \tag{2.176}$$

Noting that the non-interacting Green's functions have the form

$$\frac{1}{\omega - \epsilon_{k_1}} \sim \frac{1}{Re^{i\theta}}, \tag{2.177}$$

we find

$$\oint_{C'} d\omega\, f(\omega)H(\omega) \sim \lim_{R\to\infty}\int d(Re^{i\theta})\frac{1}{Re^{i\theta}}\frac{-1}{Re^{i\theta}} \sim \lim_{R\to\infty}\frac{1}{R} = 0, \tag{2.178}$$

which implies that Eq. (2.175) becomes

$$\frac{1}{\beta}\sum_{i\nu_1} H(i\nu_1) = \frac{-1}{2\pi i}\oint_{C} d\omega\, H(\omega)f(\omega) = \sum_{\omega'\in\{\omega_p\}} \text{Res}(H(\omega)f(\omega);\omega'). \tag{2.179}$$

This means that we can evaluate the Matsubara sums as the residues of the product of Green's functions and Fermi function at the poles of the Green's functions. So going back to our toy example in Eq. (2.169), we see that there are 2 simple poles (Eq. (2.172)) with residues

$$\begin{aligned} R_1 &= \lim_{\omega\to\epsilon_{k_1}} (\omega-\epsilon_{k_1}) \frac{f(\omega)}{(\omega-\epsilon_{k_1})(i\nu_2-\epsilon_{k_2})(i\nu+i\nu_2-\omega-\epsilon_{k+k_2-k_1})} \\ &= \frac{f(\epsilon_{k_1})}{(i\nu_2-\epsilon_{k_2})(i\nu+i\nu_2-\epsilon_{k_1}-\epsilon_{k+k_2-k_1})}, \end{aligned} \tag{2.180}$$

$$\begin{aligned} R_2 &= \lim_{\omega\to i\nu+i\nu_2-\epsilon_{k+k_2-k_1}} \frac{(\omega-i\nu+i\nu_2-\epsilon_{k+k_2-k_1})f(\omega)}{(\omega-\epsilon_{k_1})(i\nu_2-\epsilon_{k_2})(i\nu+i\nu_2-\omega-\epsilon_{k+k_2-k_1})} \\ &= \frac{-f(i\nu+i\nu_2-\epsilon_{k+k_2-k_1})}{(i\nu+i\nu_2-\epsilon_{k+k_2-k_1}-\epsilon_{k_1})(i\nu_2-\epsilon_{k_2})}. \end{aligned} \tag{2.181}$$

Next, we note that terms like $f(i\nu+i\nu_2-\epsilon_{k+k_2-k_1})$ may be simplified as

$$f(i\nu+i\nu_2-\epsilon_{k+k_2-k_1}) = f(-\epsilon_{k+k_2-k_1}) \tag{2.182}$$

by noting that the sum of fermionic frequencies is equivalent to an even bosonic frequency whose exponential is 1. Putting the residues together we find

$$\frac{1}{\beta}\sum_{i\nu_1} \frac{1}{i\nu_1-\epsilon_{k_1}} \frac{1}{i\nu_2-\epsilon_{k_2}} \frac{1}{i\nu+i\nu_2-i\nu_1-\epsilon_{k+k_2-k_1}} = \frac{f(\epsilon_{k_1})-f(-\epsilon_{k+k_2-k_1})}{(i\nu+i\nu_2-\epsilon_{k+k_2-k_1}-\epsilon_{k_1})(i\nu_2-\epsilon_{k_2})}. \tag{2.183}$$

Proceeding with the next summation over $i\nu_2$,

$$\frac{1}{\beta}\sum_{i\nu_2} \frac{f(\epsilon_{k_1})-f(-\epsilon_{k+k_2-k_1})}{(i\nu+i\nu_2-\epsilon_{k+k_2-k_1}-\epsilon_{k_1})(i\nu_2-\epsilon_{k_2})}, \tag{2.184}$$

we follow the same process noting the two simple poles are $i\nu_2 \in \{\epsilon_{k_2}, \epsilon_{k+k_2-k_1}+\epsilon_{k_1}-i\nu\}$

$$R_1 = \frac{f(\epsilon_{k_2})(f(\epsilon_{k_1})-f(-\epsilon_{k+k_2-k_1}))}{i\nu+\epsilon_{k_2}-\epsilon_{k+k_2-k_1}-\epsilon_{k_1}}, \tag{2.185}$$

$$R_2 = \frac{f(\epsilon_{k_1} + \epsilon_{k+k_2-k_1} - i\nu)(f(\epsilon_{k_1}) - f(-\epsilon_{k+k_2-k_1}))}{(\epsilon_{k+k_2-k_1} + \epsilon_{k_1} - i\nu - \epsilon_{k_2})}. \tag{2.186}$$

Similarly to the trick in Eq. (2.182), we can simplify by

$$f(\epsilon_{k_1} + \epsilon_{k+k_2-k_1} - i\nu) = -n_B(\epsilon_{k_1} + \epsilon_{k+k_2-k_1}) \equiv -n(\epsilon_{k_1} + \epsilon_{k+k_2-k_1}). \tag{2.187}$$

So we get an analytic solution to the summation over the Matsubara frequencies

$$\begin{aligned} \frac{1}{\beta^2} \sum_{\{i\nu_n\}} \frac{1}{i\nu_1 - \epsilon_{k_1}} \frac{1}{i\nu_2 - \epsilon_{k_2}} \frac{1}{i\nu + i\nu_2 - i\nu_1 - \epsilon_{k+k_2-k_1}} \\ = \frac{(f(\epsilon_{k_1}) + n(\epsilon_{k_1} + \epsilon_{k+k_2-k_1})(f(\epsilon_{k_1}) - f(-\epsilon_{k+k_2-k_1}))}{i\nu + \epsilon_{k_2} - \epsilon_{k+k_2-k_1} - \epsilon_{k_1}}. \end{aligned} \tag{2.188}$$

Meaning the expression for the second order self-energy diagram depicted in Fig. 2.7 becomes

$$\Sigma^{(2)}(k, i\nu) = \sum_{\{k_i\}} \frac{-U^2(f(\epsilon_{k_1}) + n(\epsilon_{k_1} + \epsilon_{k+k_2-k_1})(f(\epsilon_{k_1}) - f(-\epsilon_{k+k_2-k_1}))}{i\nu + \epsilon_{k_2} - \epsilon_{k+k_2-k_1} - \epsilon_{k_1}}. \tag{2.189}$$

All that remains is to complete the integration over the internal momenta. This is typically done by a Monte Carlo method since in higher order diagrams, the dimensionality of the integral becomes quite high.

This process allows us to analytically evaluate the summation over the fermionic Matsubara frequencies and returns an expression in which we can perform the analytical continuation $i\nu \to \omega + i\Gamma$ to acquire the retarded Green's function on the real frequency axis for physical applications.

We will see in the next section how this method can be automated for a general topology of a Feynman Diagram in a process known as Algorithmic Matsubara Integration.

2.7 Algorithmic Matsubara Integration

The process in Sec. 2.6 used to evaluate the summations over Matsubara frequencies can be generalized for the topology of any diagram as shown in [3]. This process is known as Algorithmic Matsubara Integration (AMI). Without making assumptions about the topology of a Feynman diagram, it will have an expression like

$$\frac{U^{n_v}}{\beta^n} \sum_{\{k_n\}} \sum_{\{\nu_n\}} \prod_{j=1}^{N} G^j(\epsilon^j, X^j) = U^{n_v} \sum_{\{k_n\}} I^{(n)}, \tag{2.190}$$

$$I^{(n)} = \frac{1}{\beta^n} \sum_{\{\nu_n\}} G^j(\epsilon^j, X^j), \tag{2.191}$$

where n_v is the number of vertices or order of the diagram, n is the number of summations over Matsubara frequencies $\{v_n\}$ and internal momenta $\{k_n\}$ and N is the number of internal non-interacting Green's functions $G(\epsilon, X)$. We use the notation for the j^{th} Green's function

$$G^j(\epsilon^j, X^j) = \frac{1}{X^j - \epsilon^j}, \tag{2.192}$$

where X^j is the frequency and $\epsilon^j = \epsilon^j(k_j)$ is the free particle dispersion. Enforcing conservation of momenta and frequency at all N vertices, allows us to express X^j and k_j as linear combinations of the internal $\{v_n, k_n\}$ and external $\{v_\gamma, k_\gamma\}$ frequencies and momenta.

$$k_j = \sum_{\ell=1}^{m} \alpha_\ell^j k_\ell, \tag{2.193}$$

$$X^j = \sum_{\ell=1}^{m} i\alpha_\ell^j v_\ell, \tag{2.194}$$

where $\gamma = m - n$ is the number of unconstrained external frequencies. The coefficients α_ℓ^j are only allowed to take on the values $-1, 0, 1$ to show each label's presence in the Green's function. We then take the representation of each Green's function to be an array of these coefficients in the linear combinations

$$G^j(X^j) \to [\epsilon^j, \vec{\alpha}^j], \tag{2.195}$$

where ϵ^j is used to keep track of unique labeling of the Green's function's dispersion and $\vec{\alpha}^j = (\alpha_1^j, \ldots \alpha_m^j)$. We can then automate the residue process as in Sec. 2.6 for this representation of the Green's functions. For each Matsubara summation, the poles are found and the residues are evaluated. The result of this process is an integrand that remains to be integrated over all the possible spatial degrees of freedom just like Eq. (2.189).

Returning to the second order self-energy from Fig. 2.7, with a slight notation change, we define the label 3 to be the label of the top propagator

$$\begin{aligned} k_3 &\equiv k + k_2 - k_1 \\ i\nu_3 &\equiv i\nu + i\nu_2 - i\nu_1. \end{aligned} \tag{2.196}$$

As described in this section, we may translate each Green's function from this topology into ϵ and α arrays

$$\begin{aligned} &\alpha_1 = \{1, 0, 0\},\ \epsilon_1 = \{1, 0, 0\} \Longleftrightarrow k_1 \\ &\alpha_2 = \{0, 1, 0\},\ \epsilon_2 = \{0, 1, 0\} \Longleftrightarrow k_2 \\ &\alpha_1 = \{-1, 1, 1\},\ \epsilon_3 = \{0, 0, 1\} \Longleftrightarrow k_3 = k + k_2 - k_1. \end{aligned} \tag{2.197}$$

Note that we use the convention that the last element is the external label. So

then the diagram may be represented as the collection of these $\vec{\epsilon} = \{\epsilon_1, \epsilon_2, \epsilon_3\}$ and $\vec{\alpha} = \{\alpha_1, \alpha_2, \alpha_3\}$, meaning we establish the isomorphism

$$\Sigma^{(2)}(k, i\nu) \cong \{\vec{\epsilon}, \vec{\alpha}\}. \tag{2.198}$$

2.7.1 The `libami` library

The algorithmic Matsubara integration (AMI) process has been implemented into a C++ library called `libami` [4]. Since we will be writing bindings for this library, the basic elements of the code are defined here and code snippets are shown to see how it is used in practice.

As described in the previous section, the topology of any Feynman diagram may be translated into an array of integers called $\vec{\alpha}$ and $\vec{\epsilon}$ for each Green's function. These are defined as `std::vector<int>` objects under the `AmiBase` class called `AmiBase::alpha_t` and `AmiBase::epsilon_t`. Then, each Green's function is represented by a `AmiBase::g_struct` object, which is a vector of a `AmiBase::alpha_t` and `AmiBase::epsilon_t`. Finally, a vector of `AmiBase::g_struct` denoted as a `AmiBase::g_prod` is the complete Feynman diagram. We may define the second order self-energy diagram as follows

```
AmiBase::alpha_t alpha_1={1,0,0};
AmiBase::alpha_t alpha_2={0,1,0};
AmiBase::alpha_t alpha_3={-1,1,1};

AmiBase::epsilon_t epsilon_1={1,0,0};
AmiBase::epsilon_t epsilon_2={0,1,0};
AmiBase::epsilon_t epsilon_3={0,0,1};
```

```
AmiBase::g_struct g1(epsilon_1,alpha_1);
AmiBase::g_struct g2(epsilon_2,alpha_2);
AmiBase::g_struct g3(epsilon_3,alpha_3);

AmiBase::g_prod_t R0={g1,g2,g3};
```

Once the topology of the diagram is inserted, as in the original AMI paper [3] we define 3 arrays called Sign, Pole and Residue array. As the names might suggest, these are used in the residue process described in Sec. 2.6. These structures are also under the `AmiBase` class, `AmiBase::S_t`, `AmiBase::P_t`, `AmiBase::R_t`. From here we insert the parameters of the problem, such as β, the dispersion of the propagators in the integrand. The classes `AmiBase::energy_t` and `AmiBase::frequency_t` are used to put all the external parameters into an object called `AmiBase::ami_vars`.

```
AmiBase ami;

AmiBase::S_t S_array;
AmiBase::P_t P_array;
AmiBase::R_t R_array;

double E_REG=0; // Numerical regulator for small energies.
int N_INT=2;  // Number of integrations

AmiBase::energy_t energy={-4,0.1,-1}; // values of momentum's dispersion
AmiBase::frequency_t frequency;
for(int i=0;i<2;i++){ frequency.push_back(std::complex<double>(0,0));} //
    internal placeholder
frequency.push_back(std::complex<double>(0,M_PI/5)); // external frequency
```

```
double BETA=5.0;
AmiBase::ami_vars external(energy, frequency,BETA);
```

Finally, we construct the integrand with `ami.construct()` and evaluate the integrand at it's current external parameters with `ami.evaluate()`

```
ami.construct(test_amiparms, R0, R_array, P_array, S_array);

std::complex<double> calc_result=ami.evaluate(test_amiparms,R_array,
    P_array, S_array, avars); // Evaluate integrand for parameters in
    'avars'
```

We provided this example here to develop the terminology used when we discuss the Python bindings for this library as well as to contrast with the Python analog to this code snippet.

2.8 Renormalized Perturbation Theory

In our recent work [6], we proposed a new method for a faster evaluation of Feynman diagrams on the real frequency axis. As described in Sec. 2.3.2, this is when we perform an analytic continuation $i\omega_n \to \omega + i0^+$ to return from the Matsubara formalism to the real frequency retarded Green's function. This works well with AMI since we can symbolically replace $i\omega_n$ with $\omega + i\Gamma$ ($\Gamma \to 0^+$) because we analytically evaluate the Matsubara sums.

What has yet to be discussed is the issue when evaluating the remaining spatial integrals after this analytic continuation. Recall, Γ is only introduced as a numerical regulator when converting from Matsubara formalism to the retarded Green's function

(Eq. (2.156)). If we take Γ too small, we lose the numerical regulation provided by Γ and the peaks from the AMI integrand become very sharp leading to large uncertainty in a Monte Carlo method evaluating the remaining integrals. But, the physical properties of the retarded Green's functions are approximated unless we take $\Gamma \to 0^+$.

In [6], we show if a large Γ is used, features of plots are lost which may be physically relevant to a specific application. However, while physical correctness is the number one priority, we cannot take Γ to be too small since the integral will be intractable to evaluate. We propose a new renormalized perturbation scheme to help avoid this issue. We will introduce this scheme as it will be used to see `pyami` in action in the Results section of this thesis.

We stick to the 2D tight binding Hubbard model as described in Sec. 2.4 with Hamiltonian as in Eq. (2.157). But we introduce a single particle term

$$\delta = z \sum_{i\sigma} \hat{n}_{i\sigma}, \tag{2.199}$$

where z is an arbitrary complex constant for now. Note that z is essentially a global chemical potential shift. We then write the Hamiltonian as

$$\begin{aligned} H &= H_0 + H_\nu + \delta - \delta \\ &= (H_0 - \delta) + (H_\nu + \delta) \\ &= H_0' + H_\nu'. \end{aligned} \tag{2.200}$$

Since we have made no change to the Hamiltonian, we are free to expand around the known solution of H_0'. As mentioned, z plays the role of a chemical potential shift

and so we can write the non-interacting Green's function as

$$G_0^{-1}(k, i\omega_n) = i\omega_n - \epsilon_k + \mu + z. \tag{2.201}$$

Then we look at H'_ν and see that there is now an extra term

$$H'_\nu = U \sum_i n_{i\uparrow} n_{i\downarrow} + \delta. \tag{2.202}$$

Recalling the discussion about the S-matrix expansion, the interaction H'_ν will give rise to the Feynman diagrams that must be summed. So when we have powers of H'_ν in contrast to powers of H_ν in the expansion, we can think of this as a binomial theorem kind of perturbation where we will end up with extra cross terms of all the possible places to insert the extra δ in the terms of the original expansion of H_ν. Also, recall the discussion about the electron-phonon interaction, where we found that the number operator, δ, (Eq. (2.73)) has a diagrammatic component that resembles a tadpole seen in Fig. (2.1c). Although this was for a different interaction, it has the same effect in the 2D Hubbard model. Putting these two concepts together, any diagram that we originally had from H_ν will now have an infinite set of diagrams consisting of all the possible ways to place tadpoles on the propagators in the original diagram. We use the terminology of self-energy insertions instead of tadpoles in this scheme. In diagrams, the self-energy insertions are denoted by a circle with a cross in it as seen in Fig. 2.9. It is also worth noting that these insertions will have no extra effect on the mathematical expressions for the diagrams built using Table 2.1 except that each insertion will break the propagator that it sits on into two identical propagators in the expression. In terms of the AMI process, this will cause higher order poles in the residue theorem process.

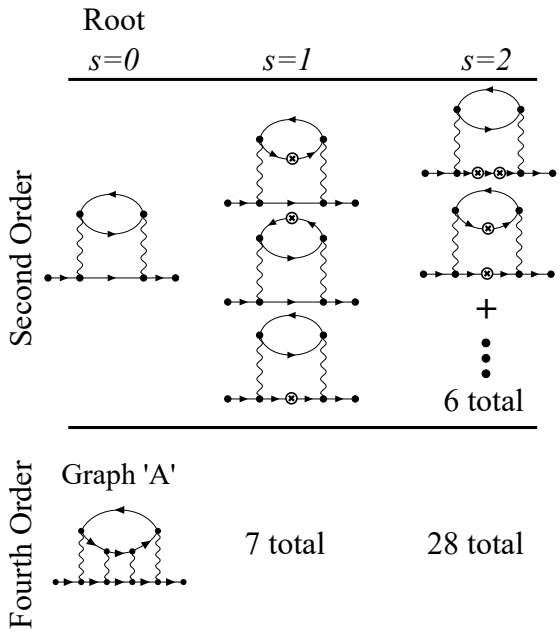


Figure 2.9: Counter term self-energy diagrams with $s = 0, 1, 2$ insertions at second order and fourth order which arise in the renormalized perturbation theory scheme. Figure taken from [6].

As an example of using this scheme, we compute the self-energy diagrams that one would need for Dyson's equation Eq. (2.165). But now we must consider the whole expansion including counter term diagrams with self-energy insertions as seen in Fig. 2.9

$$\Sigma_k(i\omega_n) = \sum_{\ell=0}^{\infty} \sum_{s=0}^{\infty} a_{\ell,s}(z) U^{\ell}(z)^s. \tag{2.203}$$

Here ℓ is the order of the self-energy diagram, s is the number of self-energy insertions on the diagram and $a_{l,s}$ is the sum of all diagrams of order ℓ, containing s insertions. We denote the truncation to order m diagrams with c insertions as

$$\Sigma_k(i\omega_n) = \sum_{\ell=0}^{m} \sum_{s=0}^{c} a_{\ell,s}(z) U^{\ell}(z)^s. \tag{2.204}$$

This is the notation used in Fig. 2.9. Now we return to the value of z. By choosing a

purely imaginary value

$$z = i\alpha. \tag{2.205}$$

We see that the analytically continued-perturbed Green's functions become

$$G_0^{-1}(k,\omega) = \omega - \epsilon_k + \mu + i(\alpha + \Gamma), \tag{2.206}$$

in contrast to the original analytically-continued Green's functions

$$G_0^{-1}(k,\omega) = \omega - \epsilon_k + \mu + i\Gamma. \tag{2.207}$$

That is, Γ is replaced with $\Gamma + \alpha$. Meaning that α will act as a new numerical regulator in the remaining spatial integrals whose effect to the self-energy may be systematically removed by summing enough counter term diagrams. Seen in Fig. 2.10 is a schematic showing the role of the numerical regulator as the term Γ in Eq. (2.207) and Eq. (2.206). Originally (Fig. 2.10 : left figure), Γ plays the role of the widths of sharp peaks in the AMI integrand and goes to zero in the physical limit ($\Gamma \to 0^+$). But after this scheme (Fig. 2.10 : right figure), the widths of the peaks go as $\alpha + \Gamma \to \alpha^+$ and do not vanish.

This is the main reason this scheme works, by introducing this new regulator α, we may take the physically correct $\Gamma \to 0^+$ limit and still be able to evaluate the spatial integrals since they have peaks of non-zero width. Of course, the consequence of the new regulator is that we now have to sum an infinite number of extra diagrams. But as seen in the expansion of the self-energy, Eq. (2.204), each diagram with s insertions is weighted by a factor of z^s. Meaning, for a small in magnitude z while still being significantly larger than Γ, the series will be well approximated by a low order truncation. Here we see the balancing act on the magnitude of z, for a precisely

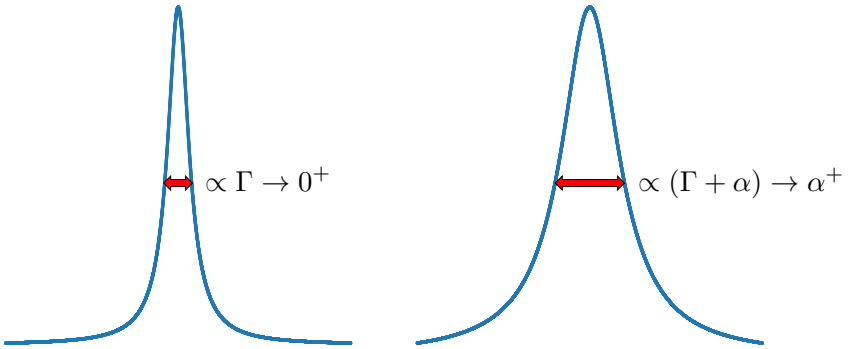


Figure 2.10: Schematic representation of the effect on the sharp peaks of the AMI integrand as a result of the introduced numerical regulator α in the renormalized perturbation theory scheme. Left curve is a original Lorentzian function with width proportional to Γ. Right curve is the renormalized perturbation theory Lorentzian with width proportional to $(\Gamma + \alpha)$. The limit as $\Gamma \to 0^+$ is also depicted to show the resulting peaks of non-vanishing width in the renormalized perturbation theory scheme.

chosen α, this method has a potentially massive payoff: a limited number of extra diagrams and each having easier-to-evaluate broadened peaks.

This paper continues to show that indeed the correct results are acquired by this process and we are able to work with values of Γ that would have taken significantly more time to evaluate if we never used this scheme.

Chapter 3

Methods

3.1 Writing `pyami`

Initially there were three candidates to write the binding code to implement `libami` into Python: Cython, Pybind11 and SWIG. However, it seemed that Pybind11 was most focused on C++ to Python bindings where others were more focused on bindings for all types of scripting languages and not only Python. Pybind11 [15] also came with the added bonus that it was built for the C++11 version of C++, which happened to be the language in which `libami` is written.

3.1.1 Pybind11

Pybind11 [15] is a lightweight header-only library that exposes the C++ types and `libami` class objects into Python without alterations to the C++ source code. This made things especially nice to convert into Python as the `pyami` code can follow exactly the same structure as the pre-existing C++ code.

The implementation of `pyami` is essentially all the outermost parts of `libami` that were introduced in Sec. 2.7.1. This is why Pybind11 is so useful, all the C++ code under the hood is left untouched and we only bind the code that is part of a application programming interface (API) of the library.

This included typecasting Python classes to play the equivalent role as C++'s `vector<T>` objects to store the topology of the Feynman diagrams as described in [16] and making an equivalent Python class to play the role of `AmiBase` where all the calculations take place as described in [4].

All of the binding code for the `pyami` is provided in Sec.(A.1). It is rather tedious to explain so we will point those interested to the documentation in [15]. The notable changes to using `pyami` compared to `libami` are pointed out in the next section.

3.1.2 Using `pyami`

We will walk through the `pyami` analog to the code snippet in Sec. 2.7.1, again looking at the toy second order self-energy diagram in Fig. 2.7 and we will comment on the differences. We first define the topology using α and ϵ.

```
import pyami


alpha1 = pyami.VectorInt([1, 0, 0])
alpha2 = pyami.VectorInt([0, 1, 0])
alpha3 = pyami.VectorInt([-1, 1, 1])


epsilon1 = pyami.VectorInt([1, 0, 0])
epsilon2 = pyami.VectorInt([0, 1, 0])
epsilon3 = pyami.VectorInt([0, 0, 1])
```

```
g1 = pyami.AmiBase.g_struct(epsilon1, alpha1)
g2 = pyami.AmiBase.g_struct(epsilon2, alpha2)
g3 = pyami.AmiBase.g_struct(epsilon3, alpha3)

R0 = pyami.g_prod_t([g1, g2, g3])
```

One difference here is that pybind11 only permits one mapping to Python lists which contain the same type. So although there are several C++ instances of classes that only contain `std::vector<int>`, like `AmiBase::alpha_t` and `AmiBase::epsilon_t`, they are mapped to one Python class `pyami.VectorInt()`. This was the case for any C++ `AmiBase` classes that apart from a different name, had the same underlying structure, e.g. `std::vector<std::complex>` are all mapped to the Python class `pyami.VectorComplex()` which is used to store both the energy and external frequency.

Next, we define the S, P, R arrays and external parameters. Then construct the AMI integrand and evaluate for the given parameters.

```
S_array = pyami.S_t()
P_array = pyami.P_t()
R_array = pyami.R_t()

E_REG = 0 # numberical regulator for small energies. If inf/nan results
    try E_REG=1e-8
N_INT = 2 # number of matsubara sums to perform
test_amiparms = pyami.AmiBase.ami_parms(N_INT, E_REG)
```

```
energy = pyami.VectorComplex([-4, 0.1, -1])
frequency = pyami.VectorComplex()
for i in range(2):
    frequency.append(0+0j)

frequency.append(0+math.pi*1j)
beta = 1.0
external = pyami.AmiBase.ami_vars(energy, frequency, beta)

ami.construct(test_amiparms, R0, R_array, P_array, S_array)
calc_result = ami.evaluate(test_amiparms, R_array, P_array, S_array, avars)
```

Other than the multiple `std::vector<T>` types being put under the same Python class, this code snippet is identical to the `libami` example in Sec. 2.7.1.

From here, we can then put a Monte Carlo integration code around this integrand process to evaluate the complete expressions of Feynman diagrams like Eq. (2.189). Of course, `pyami` has the added bonus of being in Python and, therefore, has much more options to perform the remaining integrals compared to `libami`. For the example calculations later on, we use the Monte Carlo importance sampling library, VEGAS [17], to see the benefit of having AMI available in Python.

To see how these libraries are integrated together, a Python class was written to hold all of the `pyami` objects and make the process of updating the internal parameters as a function of momenta easier in Sec. B.1. Then, to evaluate the diagram as a function of external parameters, we show an example integration function using VEGAS in Sec. B.2.

3.2 Testing

As alluded to earlier, the expression for the second order self-energy diagram in Fig. 2.7 was analytically derived both as an example of the residue method in AMI and to provide a check to ensure the values returned by `pyami` indeed agreed with the expression worked out by hand. Further testing between `libami` and `pyami` was also performed for higher order diagrams in the scaling tests as another verification process.

3.3 Scaling

`libami` [4] is currently a 5-year-old project which has had several updates and improvements along the way. There are two main methods that the `libami` library has used over the years to evaluate the analytic integrand.

One method, called SPR uses 3 arrays called S, P, R that are described in [3] to represent the unique signs, poles and residues in the analytical integral. This is the method that we have seen in the code snippets in the previous sections. The other method, called `terms`, stores the expression in a more intuitive way similar to how it would be written out by hand.

Since in a general diagram, there may be repeated poles, the SPR method ends up always being faster to evaluate since it only stores the unique poles whereas the `terms` method will evaluate the same pole multiple times [4]. However, if we were curious about the specific terms in the numerator of the analytic integrand, we would not be able to use the SPR method. As a result of the highly abstract method of storing the information, the numerator is in a factorized form so that each term cannot be looked at individually. These two methods also differ in overhead costs to evaluate - we see

a trade-off in efficiency with the order of the diagram being evaluated. The `terms` method being the most simple, is the most efficient for lower order diagrams. Whereas SPR takes more work to initially setup, but benefits with higher order diagrams where `terms` struggles.

There are two versions of each method giving a total of four methods to evaluate the integrands which we will call SPR, Optimized SPR, `terms` and Optimized `terms`. These are the product of small improvements during the 5 years of `libami`'s development. All four methods have been written into the binding codes in Sec. A.1.

This leads to the Optimized SPR (OPT. SPR) method being the method of choice in `libami` for a general diagram. Of course, the optimizations made to C++ library lead to faster evaluations, however, it was not clear if the OPT. SPR method would still be the best method Python. This is due to how Python and C++ differ in their data management, namely, how each moves lists or arrays. While this may sound like something that one could look up online, due to our uncertainty in how pybind11 writes the C++ code into Python, the fastest and most certain way to find the optimized method in `pyami` would be to race all four of the methods for various order diagrams.

There was also the question of whether this binding code would create a bottleneck when changing the parameters in the integrand, as one would do in a Monte Carlo routine.

These two questions were answered by completing scaling tests. These tests included performing Monte Carlo simulations in both `libami` and `pyami` codes and comparing the average time to evaluate the integrand. To make these tests as fair as possible and narrow down on potential bottlenecks, 50,000 random numbers were generated to an exterior file and the numbers were loaded into RAM prior to timing

the loop to complete each Monte Carlo routine. This essentially found the average time to change the values being integrated over and to evaluate the integrand for all four methods methods initially developed for `libami`.

3.4 Renormalized Perturbation Theory

As described in Sec. 2.8, we will look at diagrams with self-energy insertions and propagators with added numerical regulators as chemical potential shifts. The goal of this result is to show the benefit of having AMI available in Python. To do this, we will use the VEGAS importance sampling library in conjunction with our renormalized perturbation theory approach. The idea is that with the newly regulated integrand seen in Fig. 2.10, it will be even easier to evaluate than a plain Monte Carlo method.

3.4.1 Modifications to Diagram's Topologies

Recall that the self-energy insertion diagrams are treated with the same rules as in Table 2.1, with the exception that the propagators which contain insertions are broken into two and, therefore, appear twice in the product. Hence, we will need to modify the topologies that we input into our `pyami` codes.

As an example, consider the second order self-energy diagram with one insertion in Fig. 3.1. Using Table 2.1, we are able to form an expression for this diagram, noting that the top propagator is repeated

$$\frac{-U^2}{\beta^2}\sum_{\{k_i\}}\sum_{\{i\nu_n\}}\frac{1}{i\nu_1-\epsilon_{k_1}}\frac{1}{i\nu_2-\epsilon_{k_2}}\left(\frac{1}{i\nu_3-\epsilon_{k_3}}\right)^2. \tag{3.1}$$

So when we go to use this with `pyami` we need to create an extra propagator that

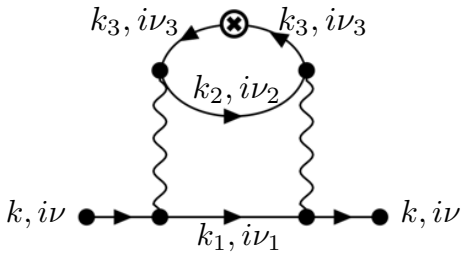


Figure 3.1: Labeled second order self-energy diagram from the renormalized perturbation theory scheme with one insertion on the top propagator labeled $(k, i\nu_3)$

is identical to the propagator with the insertion.

```
import pyami


alpha1 = pyami.VectorInt([1, 0, 0])

alpha2 = pyami.VectorInt([0, 1, 0])

alpha3 = pyami.VectorInt([-1, 1, 1])

alpha4 = pyami.VectorInt([-1, 1, 1])


epsilon1 = pyami.VectorInt([1, 0, 0])

epsilon2 = pyami.VectorInt([0, 1, 0])

epsilon3 = pyami.VectorInt([0, 0, 1])

epsilon4 = pyami.VectorInt([0, 0, 1])


g1 = pyami.AmiBase.g_struct(epsilon1, alpha1)

g2 = pyami.AmiBase.g_struct(epsilon2, alpha2)

g3 = pyami.AmiBase.g_struct(epsilon3, alpha3)

g4 = pyami.AmiBase.g_struct(epsilon4, alpha4)


R0 = pyami.g_prod_t([g1, g2, g3, g4])
```

Along with having to sum the new diagrams, the other detail when using this renormalized perturbation scheme is the chemical potential shift, z. As seen in Sec. (B.1), there is a complex class variable called `mu` so that the propagators are modified according to Eq. (2.201) (note we take $\mu = 0$ for this work, so we can use `mu` as z). Then, we can use the integrate function in Sec. (B.2) as usual.

3.4.2 Example Calculations

Rather than reproduce figures from [6], we would like to display the utility of now having AMI available in Python. To do this, we will perform a calculation which would arise while performing this scheme and use the VEGAS importance sampling library.

The reason this specific library was chosen for this scheme is to take advantage of the new numerical regulator $z = i\alpha$. As shown in Fig. 2.10, introducing this regulator will broaden the sharp peaks in the AMI integrand to make it easier to sample. But if we used VEGAS's importance sampling routine, it would be able to find these newly broadened peaks faster than the flat Monte Carlo methods used in the original work [6]. In principle, this should lead to a even faster evaluation, meaning less Monte Carlo samples will be required to get a result within a desired uncertainty.

To test this hypothesis, we will evaluate only one diagram that arises in the Renormalized Perturbation self-energy series in Eq. (2.204). We will use a fourth order self-energy diagram, which is labeled "Graph A" shown in Fig. 2.9. Although this is not one of the new counter-term diagrams, it will still have a numerical broadening by the chemical potential shift of $z = i\alpha$ and will act as a proof-of-concept calculation to show that this will be the case for all of the diagrams in the series Eq. (2.204).

In this example, we use the values $\Gamma = 0.0002$, $z = 0.2i$ to evaluate the fourth order self-energy diagram with the external momenta $k = (\pi, 0)$ and real frequency $\omega = 0.3$. In order to see the benefit of using VEGAS as an external library, we will evaluate the diagram both with VEGAS and a uniform or flat Monte Carlo sampling method as a function of Monte Carlo samples to gauge the convergence to the correct answer.

The other metric that may be of importance is the CPU time per Monte Carlo step with these methods. For example, if VEGAS was able to get to the desired uncertainty with 10% of the number of Monte Carlo samples that the flat distribution needed, but it took 100 times longer per step, this would not provide a time improvement. Since VEGAS employs a importance sampling algorithm, we expect it to take a longer time to decide where to sample the parameter space. Therefore, we will also keep track of the CPU time to achieve the number of Monte Carlo samples to see if this is an issue.

By doing this calculation, we hope to see a reduced uncertainty in the VEGAS Monte Carlo method compared to the flat Monte Carlo method for similar computational times. This will showcase the utility of now having AMI in Python. Not only will this show the benefit of using VEGAS, but we can think of VEGAS as a placeholder for one of the many math Python libraries, which could lead to a faster evaluation of the remaining spatial integrals. Ultimately, this calculation will display the potential of the `pyami` library.

Chapter 4

Results

4.1 Scaling Tests

Scaling tests were performed with all 4 methods: SPR, Optimized SPR (OPT. SPR), `terms`, Optimized `terms` (OPT. `terms`) from the `libami` library via `pyami` to see what method was the best and to see how these methods scaled with more complex diagrams. These scaling tests were performed with the `pyami` Python code and `libami` C++ code and the results are posted in Tables 4.1 and 4.2 respectively.

Order	SPR	Opt. SPR	`terms`	Opt. `terms`
2	9.14262	8.51610	6.77916	6.40774
4	203.336	142.855	272.305	228.400
6	7887.82	4044.12	14195.2	11856.8

Table 4.1: `pyami` results for the average time in microseconds to change external parameters and evaluate the integrand of 2^{nd}, 4^{th} and 6^{th} order self-energy integrands for 50000 pre-generated random numbers using different methods of storing the analytic expression of a Feynman diagram after algorithmic Matsubara integration.

From Tables 4.1 and 4.2, we can see how the different versions of storing the analytic expression change the time to evaluate the integrand for increasingly complex

Order	SPR	Opt.SPR	Terms	Opt.Terms
2	7.74534	6.56688	5.5832	4.89694
4	203.791	135.860	269.803	230.332
6	7863.04	3862.21	14203.7	11831.3

Table 4.2: `libami` (C++) results for the average time in microseconds to change external parameters and evaluate the integrand of 2^{nd}, 4^{th} and 6^{th} order self-energy integrands for 50000 pre-generated random numbers using different methods of storing the analytic expression of a Feynman diagram after algorithmic Matsubara integration.

diagrams. More specifically, how the optimized SPR method is best for higher order diagrams while the optimized terms method is the winner for low order as explained in [4]. Now comparing `libami` and `pyami` library's average times we see a nearly perfect scaling. Almost all the average evaluations are just slightly slower in Python and do not appear to be affected by the order of the diagram. Of course, there are exceptions, namely, the 4^{th} order SPR and optimized Terms and 6^{th} order Terms methods in `pyami` actually had a faster average evaluation than `libami`. This also points to a nearly perfect scaling but we do expect Python to be slower in general, although particular cases may vary. Our best guess as to why this is happening is that Python is using an optimization so that when evaluating a function whose inputs only slightly change in each call, a shortcut is made where the state is saved so that it jumps to the saved state each call after and, therefore, saves time. In contrast to C++, such an optimization probably does not exist. This would also explain why these methods are so close in general and not only the case for when Python is faster. Although this result is unexpected, this is ideal for the future of this library and further applications. We conclude from Table 4.1 that just like in `libami`, we can continue to use the optimized SPR method in `pyami` given its ability to evaluate high order diagrams efficiently.

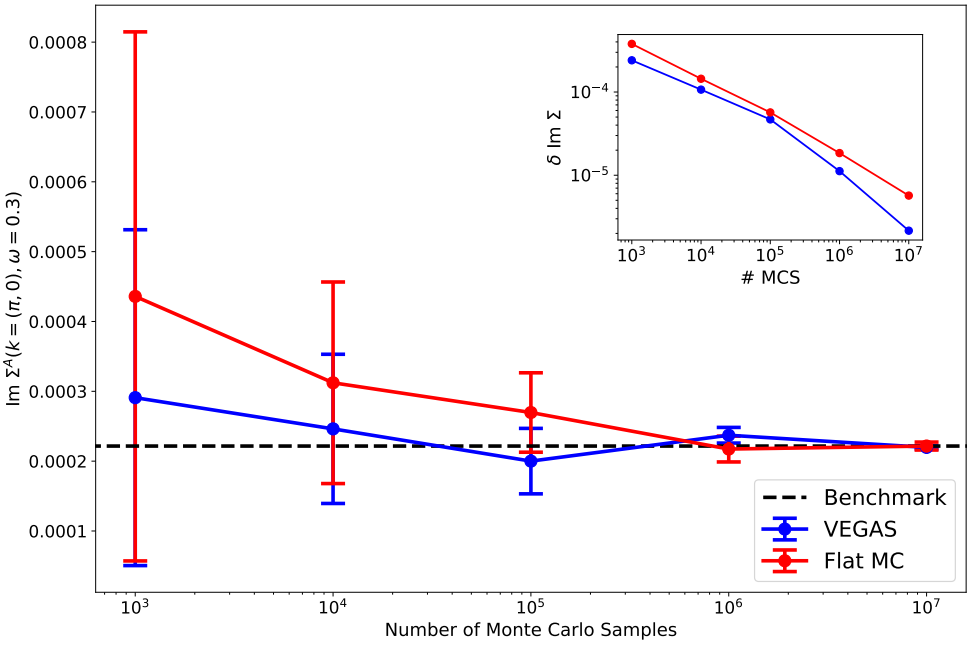


Figure 4.1: Imaginary part of the fourth order self-energy diagram using a renormalized perturbation scheme with $z = 0.2i$ and $\Gamma = 0.0002$ as a function of the number of Monte Carlo samples using various Monte Carlo methods. The magnitude of the uncertainty is plotted in the inset. Here VEGAS is plotted in blue and a flat Monte Carlo sampling method is in blue. The parameters of the model are $U = 1$, $\beta = 5$, $k = (\pi, 0)$ and $\omega = 0.3$. The benchmark value was obtained by the Flat Monte Carlo sampling ran with 10^9 Monte Carlo samples.

4.2 Renormalized Perturbation Theory Calculations

We evaluated a fourth order self-energy diagram with no self-energy insertions that would arise in the renormalized perturbation scheme proposed in [6] by using the VEGAS importance sampling library. The value of the diagram was recorded as a function of Monte Carlo samples to compare its convergence to flat distribution Monte Carlo sampling. The results are in Fig. 4.1. Here we see that the two external methods to evaluate the remaining integrals indeed converge to the correct benchmark value. The most interesting part of this figure is the magnitude of uncertainty in the Monte

Carlo methods plotted on a log scale in the inset. For a flat Monte Carlo method, the uncertainty, $\delta\,\mathrm{Im}\Sigma$, is supposed to scale as $\delta\,\mathrm{Im}\Sigma \sim \frac{1}{\sqrt{N}}$, which on a log scale as in the inset of Fig. 4.1, will correspond to a line of slope -0.5. So when we look at the efficiency of the methods, ideally we will have a slope from linear regression, $m < -0.5$. The flat Monte Carlo method has a slope of $m_{\text{Flat MC}} = -0.454$. This is close, but worse than the expected value. Comparing this to VEGAS's linear regression's slope which is $m_{\text{VEGAS}} = -0.507$, we see that this is a slight improvement over the expected value. But looking closer at the log inset of line of Fig. 4.1, we see that the blue VEGAS line has a kink in it and becomes significantly steeper after 10^5 Monte Carlo samples. Doing a linear regression for the latter half of VEGAS's uncertainty, we find a slope of $m_{\text{VEGAS},\, N\geq 10^5} = -0.668$. This is where we see the benefit of the VEGAS importance sampling library. Once it is ran long enough, (10^5 samples in this case) the algorithm will know which parts of the parameter space give the most variance and it will sample these regions more frequently to lower the uncertainty in the result [17]. In this renormalized perturbation theory case, recalling Fig. 2.10, we have an integrand containing peaks with exaggerated widths that otherwise would be vanishingly small (Here, $\Gamma = 0.0002 << 0.2 = \alpha$). So VEGAS' importance sampling has an even higher likelihood of finding these peaks to acquire the correct result if we were to sum the series in Eq. (2.204).

Lastly, we address the comment made in Sec. 3.4.2 about the time per Monte Carlo sample for the two libraries. Of course, this is an important metric to get a better idea of the time required rather than just number of samples. In Table 4.3, we show the time to complete each data point in Fig. 4.1 as well as the difference and relative difference in the times for the two integration methods. Of course, since the VEGAS importance sampling library has extra components to its algorithm, where it decides what regions of the parameter space are most efficient to sample rather

# MCS	t_{VEGAS}	$t_{\text{Flat MC}}$	Δt	% Diff
10^3	0.96549	0.942336	0.023154	2.45709
10^4	9.53672	9.05231	0.484414	5.35128
10^5	98.2184	97.8152	0.403202	0.412208
10^6	964.366	972.804	-8.43762	-0.867351
10^7	9895.75	9809.99	85.7516	0.874125

Table 4.3: Time to evaluate integration of a fourth order self-energy diagram with a increasing number of Monte Carlo samples using the VEGAS importance sampling and a Flat Monte Carlo sampling. Also shown is the time difference $\Delta t = t_{\text{VEGAS}} - t_{\text{Flat MC}}$ and relative difference of the times $\Delta t / t_{\text{Flat MC}}$.

than always taking random numbers, we expect the VEGAS integration to be slower. From Table 4.3, we see that this is mostly the case with the exception being when 10^6 Monte Carlo samples are used. Here the VEGAS algorithm is faster, although this is unexpected, by looking at the relative differences, we see that for all number of Monte Carlo samples, $N \geq 10^5$, the relevant difference is very close to zero and in this case, VEGAS just happened to be slightly faster. This sharp drop in relative difference can be once again explained by VEGAS' importance sampling being most effective when it is ran long enough so that it knows where it has to sample in the parameter space to be most efficient. Once N exceeds this threshold, VEGAS does not have to sample neighbourhoods and essentially performs a flat Monte Carlo sampling on the regions that it has already deemed as important, therefore, having nearly no difference in the time required to evaluate the integrals.

Returning to the the comment made in Sec. 3.4.2, for the VEGAS library, we see the time per Monte Carlo step are nearly equal to the flat Monte Carlo sampling. So the decrease in numerical uncertainty per number of Monte Carlo samples described in Fig. 4.1 also indicates an improved time to evaluate the integrals to a desired level of uncertainty.

Although we only compute one Feynman diagram that would arise in the renormalized perturbation theory scheme, these results act as a proof of concept calculation so that all diagrams in the expansion in Eq. (2.204) would have an improvement by using the VEGAS library. This is due to the exaggerated broadening of the otherwise, vanishing-in-width peaks that are found and sufficiently sampled by the VEGAS integration library resulting in a reduction of uncertainty in the integration.

Chapter 5

Conclusion

5.1 Discussion

In this thesis, we derived how Feynman diagrams arise in Many-body perturbation theory when computing observables of an interacting system. We provided examples of how the parts of the mathematical expressions of diagrams may be resolved and automated in a process known as algorithmic Matsubara integration [3].

We provided an idea of how this process has been implemented into C++ in a library known as `libami` [4]. Although `libami` has been used for years with success [5], it is limited to the external libraries written in C++ to evaluate the remaining spatial integrals that are not handled by AMI. This served as motivation to implement AMI into a more user friendly language with plenty of readily available external packages such as Python.

In this project, we wrote Python bindings by using pybind11 [15] to implement `libami` into a Python library called `pyami`. Exactly like `libami`, `pyami` is a complete Python module so that once provided with a Feynman diagram's topology, by using

AMI, the momentum integrand is formed. But now, the remaining integrals can be handled by the plethora of math Python libraries that are now at one's disposal. In this thesis, we implemented the VEGAS importance sampling library in conjunction with our recent work [6] for an improvement in the time required to evaluate of Feynman diagrams on the real frequency axis.

5.2 Future work

Future work involving the `pyami` library will include examples of pairing `pyami` with other Python libraries to evaluate the remaining spatial integrals. Specifically, since Python is the home of modern machine learning codes, there are several ideas to use machine learning to evaluate the remaining integrals after the AMI process.

One example could be training a neural network to learn the positions of the peaks in Fig. 2.10 as a function of the external parameters used. As an alternative, we could use the renormalized perturbation theory scheme to exaggerate the peaks to better learn the positions and, therefore, have a better chance of predicting the locations that must be sampled more heavily compared to flat areas of the integration space.

Another idea is to learn our AMI integrand with what are called Tensor Trains which have been used in the past for quantum dot calculations [18]. Once the integrand is learned, the multidimensional integrand is decomposed into a product of functions, each dependent on one variable. Then, the multidimensional integral can be written as the product of individual integrals that presumably are easier to evaluate.

These are only a couple of examples, but there are plenty of future applications of this Python library.

Appendix A

pyami Code Excerpts

Most Codes are available to the public on this project's Github page [19]. However, we provide larger relevant codes here.

A.1 Binding codes with pybind11

```
#include "../src/ami_base.hpp"
#include <pybind11/stl.h>
#include <pybind11/stl_bind.h>
#include <pybind11/complex.h>
#include <pybind11/pybind11.h>

PYBIND11_MAKE_OPAQUE(std::vector<int>);
PYBIND11_MAKE_OPAQUE(std::vector<double>);
PYBIND11_MAKE_OPAQUE(std::vector<std::vector<std::vector<AmiBase::pole_struct>>>);
    // for mutability of P_t
```

```
PYBIND11_MAKE_OPAQUE(std::vector<std::vector<std::vector<double>>>); //
    for mutability of S_t
PYBIND11_MAKE_OPAQUE(std::vector<std::vector<std::vector<AmiBase::g_struct>>>);
    // for mutability of R_t
PYBIND11_MAKE_OPAQUE(std::vector<std::complex<double>>);
PYBIND11_MAKE_OPAQUE(std::vector<AmiBase::g_struct>);
PYBIND11_MAKE_OPAQUE(std::vector<std::vector<AmiBase::ref_t>>);
PYBIND11_MAKE_OPAQUE(std::vector<AmiBase::pole_struct>);
PYBIND11_MAKE_OPAQUE(std::vector<AmiBase::term>);

namespace py = pybind11;

void init_pyami_wrapper(py::module &m) {

  py::bind_vector<std::vector<int>>(m, "VectorInt");
  py::bind_vector<std::vector<double>>(m, "VectorDouble");
  py::bind_vector<std::vector<std::vector<std::vector<AmiBase::pole_struct>>>>(m,
      "P_t");
  py::bind_vector<std::vector<std::vector<std::vector<double>>>>(m, "S_t");
  py::bind_vector<std::vector<std::vector<std::vector<AmiBase::g_struct>>>>(m,
      "R_t");
  py::bind_vector<std::vector<std::complex<double>>>(m, "VectorComplex");
  py::bind_vector<std::vector<AmiBase::g_struct>>(m, "g_prod_t");
  py::bind_vector<std::vector<std::vector<AmiBase::ref_t>>>(m, "R_ref_t");
  py::bind_vector<std::vector<AmiBase::pole_struct>>(m, "pole_array_t");
  py::bind_vector<std::vector<AmiBase::term>>(m, "terms");
```

```
py::class_<AmiBase> AmiBase(m, "AmiBase");
AmiBase.def(py::init<>());
AmiBase.def(py::init<AmiBase::ami_parms &>());

py::class_<AmiBase::ami_vars> (AmiBase, "ami_vars")
  .def(py::init<>())
  .def(py::init<AmiBase::energy_t, AmiBase::frequency_t>())
  .def(py::init<AmiBase::energy_t, AmiBase::frequency_t, double>())
  .def(py::init<AmiBase::energy_t, AmiBase::frequency_t, double,
     double>())
  .def_readwrite("energy_", &AmiBase::ami_vars::energy_)
  .def_readwrite("frequency_", &AmiBase::ami_vars::frequency_)
  .def_readwrite("prefactor", &AmiBase::ami_vars::prefactor)
  .def_readwrite("BETA_", &AmiBase::ami_vars::BETA_)
  .def_readwrite("gamma_", &AmiBase::ami_vars::gamma_);

py::class_<AmiBase::ami_parms> (AmiBase, "ami_parms")
  .def(py::init<>())
  .def(py::init<int, double>())
  .def(py::init<int, double, AmiBase::graph_type>())
  .def(py::init<int, double, AmiBase::graph_type, AmiBase::int_type,
     AmiBase::disp_type>())
  .def_readwrite("N_INT_", &AmiBase::ami_parms::N_INT_)
  .def_readwrite("N_EXT_", &AmiBase::ami_parms::N_EXT_)
  .def_readwrite("E_REG_", &AmiBase::ami_parms::E_REG_)
  .def_readwrite("tol_", &AmiBase::ami_parms::tol_)
  .def_readwrite("TYPE_", &AmiBase::ami_parms::TYPE_)
```

```
  .def_readwrite("int_type_", &AmiBase::ami_parms::int_type_)
  .def_readwrite("dispersion_", &AmiBase::ami_parms::dispersion_);

py::class_<AmiBase::g_struct> (AmiBase, "g_struct")
  .def(py::init<AmiBase::epsilon_t, AmiBase::alpha_t,
     AmiBase::stat_type>())
  .def(py::init<AmiBase::epsilon_t, AmiBase::alpha_t>())
  .def(py::init<>())
  .def_readwrite("eps_", &AmiBase::g_struct::eps_)
  .def_readwrite("alpha_", &AmiBase::g_struct::alpha_)
  .def_readwrite("stat_", &AmiBase::g_struct::stat_)
  .def_readwrite("species_", &AmiBase::g_struct::species_)
  .def_readwrite("eff_stat_", &AmiBase::g_struct::eff_stat_)
  .def_readwrite("pp", &AmiBase::g_struct::pp);

py::class_<AmiBase::pole_struct> (AmiBase, "pole_struct")
  .def(py::init<>())
  .def(py::init<AmiBase::epsilon_t, AmiBase::alpha_t>())
  .def_readwrite("eps_", &AmiBase::pole_struct::eps_)
  .def_readwrite("alpha_", &AmiBase::pole_struct::alpha_)
  .def_readwrite("index_", &AmiBase::pole_struct::index_)
  .def_readwrite("multiplicity_", &AmiBase::pole_struct::multiplicity_)
  .def_readwrite("der_", &AmiBase::pole_struct::der_)
  .def_readwrite("which_g_", &AmiBase::pole_struct::which_g_)
  .def_readwrite("x_alpha_", &AmiBase::pole_struct::x_alpha_);
```

```
py::class_<AmiBase::term> (AmiBase, "term")
  .def(py::init<>())
  .def(py::init<double, AmiBase::pole_array_t, AmiBase::g_prod_t>())
  .def_readwrite("sign", &AmiBase::term::sign)
  .def_readwrite("p_list", &AmiBase::term::p_list)
  .def_readwrite("g_list", &AmiBase::term::g_list);

AmiBase.def("construct", py::overload_cast<AmiBase::ami_parms &,
    AmiBase::g_prod_t, AmiBase::R_t &, AmiBase::P_t &, AmiBase::S_t
    &>(&AmiBase::construct), "Construction function for term-by-term
    construction.");

AmiBase.def("evaluate", py::overload_cast<AmiBase::ami_parms &,
    AmiBase::R_t &, AmiBase::P_t &, AmiBase::S_t &, AmiBase::ami_vars
    &>(&AmiBase::evaluate), "This is the primary evaluation which takes
    again ‘ami_parms‘, the outputs from ‘construct‘ as well as the
    ‘ami_vars‘ external values that enter into the expression");


AmiBase.def("factorize_Rn", &AmiBase::factorize_Rn, "Optimize function
    for SPR notation.");
```

```
  AmiBase.def("evaluate", py::overload_cast<AmiBase::ami_parms &,
      AmiBase::R_t &, AmiBase::P_t &, AmiBase::S_t &, AmiBase::ami_vars &,
      AmiBase::g_prod_t &, AmiBase::R_ref_t &, AmiBase::ref_eval_t
      &>(&AmiBase::evaluate), "This is an optimized version of the evaluate
      function. For simplicity if the additional arguments are empty the
      evaluate function is called directly.");

  AmiBase.def("construct", py::overload_cast<int, AmiBase::g_prod_t,
      AmiBase::terms &>(&AmiBase::construct), "Construction function for
      term-by-term construction.");

  AmiBase.def("evaluate", py::overload_cast<AmiBase::ami_parms &,
      AmiBase::terms &, AmiBase::ami_vars &>(&AmiBase::evaluate), "Evaluate
      Terms.");

  AmiBase.def("factorize_terms", &AmiBase::factorize_terms, "Optimize
      factorize function for terms notation.");

  AmiBase.def("evaluate", py::overload_cast<AmiBase::ami_parms &,
      AmiBase::terms &, AmiBase::ami_vars &, AmiBase::g_prod_t &,
      AmiBase::R_ref_t &, AmiBase::ref_eval_t &>(&AmiBase::evaluate),
      "Optimized evaluate function for terms notation.");
}
```

Appendix B

Sample Vegas implementation with pyami

Here we show how the `pyami` class can be wrapped up into a integrand to be paired with an external Python library. We also show an example of a integrate function that takes a file of external parameters.

B.1 pyami integrand class

This class was written to collect all the `pyami` objects into one so that we could easily modify the internal degrees of freedom to evaluate the integrand via a Python's `__call__` method.

```
import numpy as np
import vegas


class pyami_integrand_Hubbard:
```

```
def __init__(self, ami, parms, R, P, S, avars, unique, rref, eval_list,
    q_ext, part, epsilon, mu, R0, order):
    self.ami = ami
    self.parms = parms
    self.R = R
    self.P = P
    self.S = S
    self.avars = avars
    self.unique = unique
    self.rref = rref
    self.eval_list = eval_list
    self.q_ext = q_ext
    self.part = part
    self.epsilon = epsilon # Python lambda for particle's dispersion as
        a function of momenta
    self.mu = mu # chemical potential shift used in renormalzed PT
        schemes
    self.R0 = R0 # keep topology with integrand to get correct energy
        linear combinations
    self.alphas = np.array([self.R0[i].alpha_ for i in
        range(len(self.R0))])
    self.order = order # order of diagram - needs to be inputed bc
        len(R0) changes with counterterms

def __call__(self, x):
    #update interal dispersions as a function of momenta x
```

```
        self.update_integrand(x)

        # evaluate
        if self.part == 'real':
            return self.ami.evaluate(self.parms, self.R, self.P, self.S,
                self.avars, self.unique, self.rref, self.eval_list).real

        else:
            return self.ami.evaluate(self.parms, self.R, self.P, self.S,
                self.avars, self.unique, self.rref, self.eval_list).imag

    def update_integrand(self, k):
        # use alphas to get correct linear comb of each k for all
            propagators
        k_x = np.append(k[0::2], self.q_ext[0])
        k_y = np.append(k[1::2], self.q_ext[1])
        K_eff = np.vstack((np.matmul(self.alphas, k_x),
            np.matmul(self.alphas, k_y)))

        # extra negative on energies!
        self.avars.energy_ = py.VectorComplex([self.mu -
            self.epsilon(K_eff[:, i]) for i in range(len(self.alphas))])

        return self

    def update_external(self, b_new, qx_new, qy_new, Rew_new, Imw_new):
        self.avars.BETA_ = b_new
```

```
        self.q_ext = [qx_new, qy_new]
        self.avars.frequency_[len(self.avars.frequency_) - 1] = Rew_new +
            1j*Imw_new

    def get_q_ext(self): return self.q_ext

    def get_w_ext(self): return
        self.avars.frequency_[len(self.avars.frequency_) - 1]

    def get_beta(self): return self.avars.BETA_
```

B.2 pyami-VEGAS integration function

Sample integration function with the `pyami` integrand class which uses VEGAS. Here the integrand object is passed in along with a file name containing external parameters to evaluate along with parameters for the VEGAS integration and returns `numpy` array of the results.

```
def vegas_pyami_2dHubbard(integrand, ext_vars, nitn, neval, alpha=0.5,
    beta=0.75, neval_frac=0.75, frac_prime=0.1, adapt=True):

    ans = []

    f = open(ext_vars, 'r')
    lines = f.readlines()

    for i in range(len(lines)):
```

```
# print % complete
if (i!=0 and (10*i)%len(lines) == 0):
    print(f"{i/len(lines) * 100}% complete")

# update external variables that are beta, q_x, q_y, re(w), im(w),
    w_c
l = lines[i]
beta_, q_x, q_y, re_w, im_w, w_c = map(float, l.split(" "))

integrand.update_external(b_new = beta_, qx_new = q_x, qy_new =
    q_y, Rew_new=re_w, Imw_new = im_w)
R = 2 * integrand.order * [[0, 2*np.pi]]

# MC loop
integ = vegas.Integrator(R)
try:
    integ(integrand, nitn=nitn, neval=frac_prime*neval, nproc=1,
        alpha=alpha, beta=beta, neval_frac=neval_frac, adapt=adapt)
        # prime integrand
    result = integ(integrand, nitn=nitn, neval=neval, nproc=1,
        alpha=alpha, beta=beta, neval_frac=neval_frac, adapt=adapt)
    ans.append(result/(2*np.pi)**(2*integrand.order)) # divide by
        (2pi)^dim
except ValueError:
    print(f"VALUE ERROR ENCOUNTERED AT {integrand.get_q_ext()},
        {integrand.get_w_ext()}, {integrand.get_beta()}")
```

```
            ans.append(np.nan+1j*np.nan)

    return np.array(ans)
```

Bibliography

[1] R.D. Mattuck. *A Guide to Feynman Diagrams in the Many-body Problem*. Dover Books on Physics Series. Dover Publications, 1992.

[2] J. P. F. LeBlanc, Andrey E. Antipov, Federico Becca, Ireneusz W. Bulik, Garnet Kin-Lic Chan, Chia-Min Chung, Youjin Deng, Michel Ferrero, Thomas M. Henderson, Carlos A. Jiménez-Hoyos, E. Kozik, Xuan-Wen Liu, Andrew J. Millis, N. V. Prokof'ev, Mingpu Qin, Gustavo E. Scuseria, Hao Shi, B. V. Svistunov, Luca F. Tocchio, I. S. Tupitsyn, Steven R. White, Shiwei Zhang, Bo-Xiao Zheng, Zhenyue Zhu, and Emanuel Gull. Solutions of the two-dimensional hubbard model: Benchmarks and results from a wide range of numerical algorithms. *Phys. Rev. X*, 5:041041, Dec 2015.

[3] Amir Taheridehkordi, S. H. Curnoe, and J. P. F. LeBlanc. Algorithmic matsubara integration for hubbard-like models. *Phys. Rev. B*, 99:035120, Jan 2019.

[4] Hossam Elazab, B.D.E. McNiven, and J.P.F. LeBlanc. Libami: Implementation of algorithmic matsubara integration. *Computer Physics Communications*, 280:108469, 2022.

[5] James P. F. LeBlanc, Kun Chen, Kristjan Haule, Nikolay V. Prokof'ev, and Igor S. Tupitsyn. Dynamic response of an electron gas: Towards the exact exchange-correlation kernel. *Phys. Rev. Lett.*, 129:246401, Dec 2022.

[6] M. D. Burke, Maxence Grandadam, and J. P. F. LeBlanc. Renormalized perturbation theory for fast evaluation of feynman diagrams on the real frequency axis. *Phys. Rev. B*, 107:115151, Mar 2023.

[7] Radi A. Jishi. *Feynman Diagram Techniques in Condensed Matter Physics*. Cambridge University Press, 2013.

[8] G.D. Mahan. *Many-Particle Physics*. Physics of Solids and Liquids. Springer US, 1990.

[9] Murray Gell-Mann and Francis Low. Bound states in quantum field theory. *Phys. Rev.*, 84:350–354, Oct 1951.

[10] Takeo Matsubara. A New Approach to Quantum-Statistical Mechanics. *Progress of Theoretical Physics*, 14(4):351–378, 10 1955.

[11] J. Hubbard. Electron correlations in narrow energy bands. *Proceedings of the Royal Society of London. Series A, Mathematical and physical sciences*, 276(1365):238–257, 1963.

[12] André-Marie S. Tremblay. Two-particle-self-consistent approach for the hubbard model. In *Strongly Correlated Systems*, Springer Series in Solid-State Sciences, pages 409–453. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[13] Mervyn Roy. The tight binding method, May 2015.

[14] Michael V. Sadovskii. *Diagrammatics: lectures on selected problems in condensed matter theory.* World Scientific Publishing Co. Pte. Ltd, 2006.

[15] Wenzel Jakob. Pybind11. https://github.com/pybind/pybind11, 2016.

[16] Amir Taheridehkordi, S. H. Curnoe, and J. P. F. LeBlanc. Algorithmic approach to diagrammatic expansions for real-frequency evaluation of susceptibility functions. *Phys. Rev. B*, 102:045115, Jul 2020.

[17] G P Lepage. VEGAS - an adaptive multi-dimensional integration program. Technical report, Cornell Univ. Lab. Nucl. Stud., Ithaca, NY, 1980.

[18] Yuriel Núñez Fernández, Matthieu Jeannin, Philipp T. Dumitrescu, Thomas Kloss, Jason Kaye, Olivier Parcollet, and Xavier Waintal. Learning feynman diagrams with tensor trains. *Phys. Rev. X*, 12:041018, Nov 2022.

[19] James P.F LeBlanc. libami. https://github.com/jpfleblanc/libami, 2022.