Numerical Methods for Finite Temperature Effects in Quantum Field Theory

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Physics and Engineering Physics
University of Saskatchewan
Saskatoon

By
Siyuan Li

Permission to Use

Disclaimer

Abstract

The basic structure of quantum field theory that is used to describe the Standard Model of fundamental interactions of nature is usually formulated for zero temperature. However, the effects of temperature are extremely important for understanding a number of physical processes such as the electroweak phase transition and quark-gluon plasma. The extension of quantum field theory to non-zero temperature is achieved by modifying the propagators in loop integrations represented by Feynman diagrams. The Python-language-based package pySecDec is designed for numerical calculation of dimensionally regulated loop integrals. The research goal for my thesis is to develop a methodology to numerically calculate loop integrations for finite temperature effects in quantum field theory by adapting pySecDec functions and implementing them for such purpose. In this thesis, the methodology is used on one-loop self-energy to achieve numerical calculation results. The pySecDec methodology is validated in comparison to existing analytic results for this topology.

Acknowledgements

This thesis is a brief presentation of my first academic journey. I received tremendous help from some of the kindest person I know.

Professor Steele, one of my supervisors, who have supported and mentored me not only in academia but also in life, have contributed a great amount in this work.

Professor Harnett, who offers me the opportunity to this research field and this institution, without whom the code in this research would never worked. I will keep the "pipeline" burning.

My mother, who have been giving me everything she can and everything I have.

Professor Chizma and professor Cooper, who kindly helped me to get this position as a MSc student and this gruesome life as a young scientist. I will forever remember that I wouldn't be able to start my academia journey without you.

My dear fellow young researcher Bábara, with whom we shared the strangest but awesome connection.

And thank you to Debbie and Marj, my "moms" away from home, who have taken care of everything other than the science that no one understands.

I would also like to express my gratitude towards Jason, Zhi Wei, Robin, Kaden, Thamirys, Alex, Bernard, Bardia, Yueying, Weiwen, my family, Amanda and all those who offered me a hand at my lows. All of you have given me strength and power along the way.

This thesis is powered by all the support and love I have received. And I hope it will bring not only intellectual contribution but also humanity to the world.

Contents

List of Figures

List of Abbreviations

LHS	Left-hand Side
RHS	Right-hand Side
VEV	Vacuum Expectation Value
QCD	Quantum Chromodynamics
QFT	Quantum Field Theory
PI	Path Integral

1 Introduction to Thermal Field theory

Most of the basic monographs and interest in quantum field theory (QFT) have been based on classical mechanics, classical electrodynamics, and quantum mechanics. The thermal background for ordinary QFT is normally considered as zero temperature. There has been growing interest over the years in many physical processes that are affected by temperature. One of the examples would be the thermal equilibrium background for particles in the early universe. This thermal bath provides particles with their temperature-related typical energy [1]. Other examples include the electroweak phase transition [2], the quark-gluon plasma being studied at the large hadron collider (LHC), and gravitational wave signals from strong first-order phase transitions in extensions of the Standard Model (e.g., hidden sector dark matter models) [3, 4].

But before we dig into field theories at finite temperature, we will first start with an introduction to QFT and some of its results that are relevant to this thesis in Section 1.1. Chapter 1 will move on to the concept of thermal field theory and the formulation of the one-loop self-energy topology. With the basic theory and formula introduced, we will perform QFT benchmarking calculations using the numerical program we chose (pySecDec) in Chapter 2. After the Chapter 2 study confirming that the computation program pySecDec is suitable and reliable for our topic, Chapter 3 will show the methodology development for thermal field theory correlation function calculations of the one-loop self-energy topology. Chapter 4 will then show the numerical computational results using our methodology proposed in Chapter 3 with respect to multiple situations and physical variables. Finally, a conclusion will be presented in Chapter 5 for the complete thesis and possible future applications.

1.1 Quantum Field Theory

In the theories of elementary particle interactions, quantum field theory treats subatomic processes as field interactions instead of quantization of particles as in quantum mechanics. Emerging from the Lagrangian, rises the Feynman diagram which provides a diagrammatic method to analyze the mathematical expressions of subatomic particle interactions, also known as the Feynman rules. Along with the Feynman rules, Feynman diagrams provide a pictorial expression for these amplitudes in particle physics. This thesis will be mostly based on the Feynman integral of two-point, one-loop Feynman diagram topologies.

The analytic calculations for loop topologies are quite difficult [5], but the numerical side of loop topology calculation techniques is also developing [6] and providing a promising field of loop integral calculation to the cases that are beyond the limitation of analytic methods. This thesis develops and implements new techniques that use pySecDec, which will be introduced more in detail in Chapter 3, to adapt these numerical approaches to finite temperature. One of the motivations of this numerical algorithm development is to compute finite-temperature correlation functions that occur in quantum chromodynamics (QCD). The QCD sum rules (or Shifman–Vainshtein–Zakharov sum rules) is a method that relates the theory of quarks to the theory of how hadrons interact since the conventional perturbative techniques do not apply to the strong coupling nature of hadrons. The basic concept of QCD sum rules is called duality. If we account for all QCD interactions, that will include all hadron interactions. And with understanding and experience from this research, we hope we can extend the scope of this numerical method to more cases in QCD and contribute to the study of QCD sum rules at finite temperature.

1.1.1 Path Integrals

The basic concepts of Feynman rules can be introduced from many perspectives. The path-integral (PI) formalism is one of the most common ways to show how Feynman integrals are constructed. Following the train of thought from Ref. [7], we can discuss the foundational structure of the loop integral that this thesis is focusing on. In quantum mechanics, the

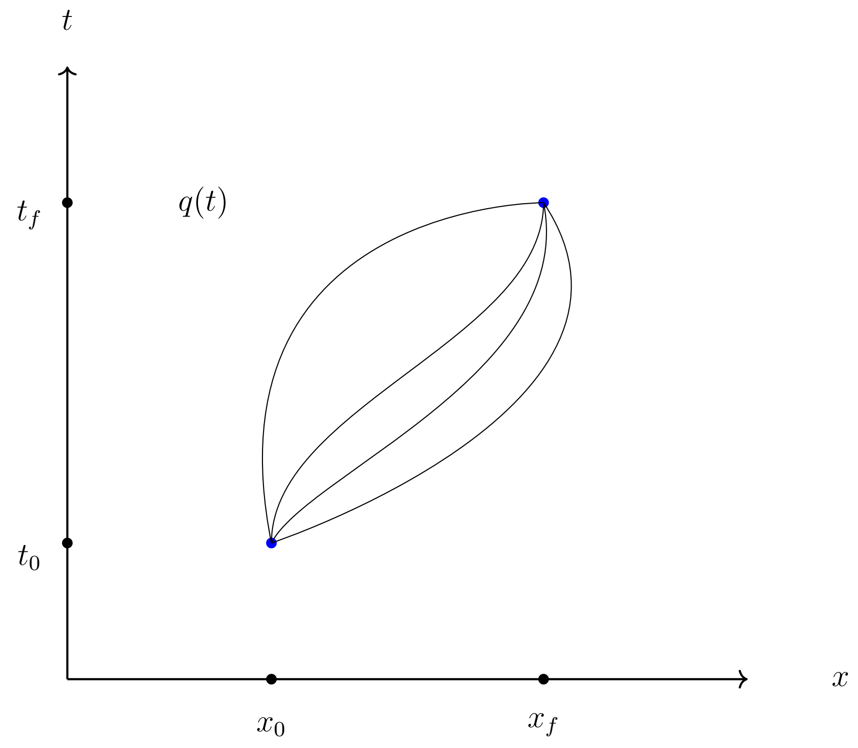


Figure 1.1: The functional integral Eq. (1.1) sums over all paths $q(t)$ connecting (x_0, t_0) to (x_f, t_f).

transition matrix element between initial and final states is represented as

$$\langle x_f; t_f \mid x_0; t_0 \rangle = N \int [dq(t)] \, \exp \left[i \int \mathcal{L}(q, \dot{q}) \, dt \right], \tag{1.1}$$

where $\int [dq(t)]$ is the integral over trajectories $q(t)$ connecting the initial state (x_0, t_0) to the final state (x_f, t_f), N is the normalization factor, $\mathcal{L}(q, \dot{q})$ is the Lagrangian, and the expression in the exponential is the action $\mathcal{S}$. The right hand side (RHS) of Eq. (1.1) is the PI formalism of this transition matrix element. We can see that this fundamental quantity is now written in the form of a functional integral. Equation (1.1) sums up all the possible paths from the initial state to the final state as shown in Fig. 1.1.

Now moving on to field theory, we have a similar representation. Analogous to classical physics, we have classical mechanics and classical field theory. Equation (1.1) transforms into a vacuum-to-vacuum transition amplitude

$$\langle 0 \, |\mathcal{O}| \, 0 \rangle = N \int [d\phi] \, \mathcal{O} \, \exp \left[i \int \mathcal{L}(\phi, \partial_\mu \phi) \, d^4x \right], \tag{1.2}$$

where $\int [d\phi]$ means integration over all field configurations and $\mathcal{O}$ is any combination of field operators. We can therefore write PI formula Eq. (1.2) as a field theory Green's function

$$\langle 0 \,|T\left(\phi(x_1)\phi(x_2)...\phi(x_n)\right)|\, 0\rangle = N\!\int [d\phi]\,\phi(x_1)\phi(x_2)...\phi(x_n)\;\exp\left[i\!\int \mathcal{L}\left(\phi,\partial_\mu\phi\right) d^4x\right], \quad (1.3)$$

where T is the time-ordering symbol[1]. Then, the generating functional $Z[J]$ that stands for the vacuum-to-vacuum transition amplitude with an external source $J(x)$ can be written as

$$Z[J] = N\!\int [d\phi]\;\exp\left[i\int \mathcal{L}(\phi(x),\partial_\mu\phi(x))\;d^4x + i\int J(x)\phi(x)\;d^4x\right]. \quad (1.4)$$

From Eq. (1.4) we can easily see

$$\left.\frac{\partial Z[J]}{i\,\partial J(x)}\right|_{J=0} = N\int [d\phi]\,\phi(x)\;\exp\left[i\int \mathcal{L}(\phi(z),\partial_\mu\phi(z))d^4z\right]. \quad (1.5)$$

The RHS of Eq. (1.5) is the expectation value of $\phi(x)$ which can be seen from Eq. (1.3):

$$\left.\frac{\partial Z[J]}{i\,\partial J(x)}\right|_{J=0} = \langle 0|\,T(\phi(x))\,|0\rangle\,. \quad (1.6)$$

Generalizing, we have

$$\left.\frac{\partial Z[J]}{i\,\partial J(x_1)}\frac{\partial Z[J]}{i\,\partial J(x_2)}...\frac{\partial Z[J]}{i\,\partial J(x_n)}\right|_{J=0} = \langle 0|\,T(\phi(x_1)\phi(x_2)...\phi(x_n))\,|0\rangle. \quad (1.7)$$

With Eq. (1.7) in mind, we will now look at the $\lambda\phi^4$ theory,

$$\mathcal{L}(\phi,\partial_\mu\phi) = \mathcal{L}_0(\phi,\partial_\mu\phi) + \mathcal{L}_{int}(\phi), \quad (1.8)$$

in which

$$\begin{aligned}\mathcal{L}_0(\phi,\partial_\mu\phi) &= \frac{1}{2}(\partial_\mu\phi)(\partial^\mu\phi) - \frac{1}{2}m^2\phi^2 + i\zeta\phi^2, \;\text{ for free field}\\ \mathcal{L}_{int}(\phi) &= -\frac{\lambda}{4!}\phi^4,\end{aligned} \quad (1.9)$$

where $\zeta \to 0^+$. The term $i\zeta\phi^2$ gives the path integral a damping factor to ensure convergence

[1]To avoid confusion with the time-ordering symbol T, $\mathcal{T}$ will be used to represent temperature, as outlined later in the thesis.

of Eq. (1.4) in Minkowski space. This term is related to the Wick rotation which will be discussed later. Between the two parts in Eq. (1.9), we can evaluate the part of $Z[J]$ corresponding to $\mathcal{L}_0$ (free field) exactly

$$Z_0[J] = \exp\left[i \int d^4x\, d^4y\, J(x)\Delta(x-y)J(y)\right], \tag{1.10}$$

$$\Delta(x-y) = \int \frac{d^4k}{(2\pi)^4} \frac{e^{ik\cdot(x-y)}}{k^2 - m^2 + i\zeta}, \tag{1.11}$$

where the Euclidean four-momentum $k = (k_0, \boldsymbol{k})$. The quantity $\Delta(x-y)$ is essentially the inverse of the operator $\partial_\mu^2 - m^2$ for the free field Lagrangian. Now, the interacting part of the Lagrangian can be written as

$$Z[J] = \exp\left[\int d^4x \mathcal{L}_{int}\underbrace{\left(\frac{\partial}{i\partial J(x)}\right)}_{\phi(x)}\right] Z_0[J], \tag{1.12}$$

where the interaction Lagrangian is represented as

$$\mathcal{L}_{int} = -\frac{\lambda}{4!}\phi^4 \to -\frac{\lambda}{4!}\frac{\partial^4}{\partial J^4(x)}. \tag{1.13}$$

Returning back to the derivation of the propagator, we can find the functional differentiation by simply using derivative on the external source function J first.

$$\frac{\partial J(y)}{\partial J(x)} = \delta(x-y). \tag{1.14}$$

Then the free-field Feynman propagator can be derived combining Eqs. (1.10) and (1.14):

$$\begin{aligned} \langle 0| T(\phi(x_1)\phi(x_2)) |0\rangle &= \frac{1}{i}\frac{\partial}{\partial J(x_1)}\frac{1}{i}\frac{\partial}{\partial J(x_2)} Z_0[J]\bigg|_{J=0} \\ &= \Delta(x_1 - x_2). \end{aligned} \tag{1.15}$$

Feynman rules for vertices can easily be obtained from Eqs. (1.10) and (1.12). Path Integral quantization formalism gives an elegant derivation of Feynman rules from the perspective of quantum physics.

1.1.2 Wick Rotation

The essential purpose of the Wick rotation is to adapt a known solution in Euclidean space to a similar problem in Minkowski space. In ordinary QFT, Wick rotations are also used to avoid poles along integration contours. The poles will be located at $p_0 = \pm(E_{\boldsymbol{p}} - i\zeta)$, displaced above and below the real axis [8] as shown in Fig. 1.2.

In the more general situation of QFT, the Wick rotation substitutes the temporal component of the Minkowski space p_0^M with ip_0^E in Euclidean space. That turns the norm square of the four-momentum from

$$ {p_E}^2 = {p_0^E}^2 + \boldsymbol{p}^2 \tag{1.16} $$

into

$$ {p_M}^2 = {p_0^M}^2 - \boldsymbol{p}^2 = -{p_0^E}^2 - \boldsymbol{p}^2 = -{p_E}^2, \tag{1.17} $$

where p_E is the four momentum in Euclidean space and similarly, p_M is the four momentum in Minkowski space.

From Fig. 1.2, we can see that the rotated contour avoids the poles. As a systematic way of working with Feynman integrals, the Wick rotation provided inspiration to the thesis methodology since the pySecDec calculation tool we use works on Minkowski, as opposed to Euclidean integrals. Later in Section 3.3, we will come back to this application of Wick rotations.

1.2 Thermal Field Theory

In this section, we will introduce the finite-temperature QFT following the treatment of Refs. [9] and [1].

In order to develop a finite-temperature QFT, or thermal field theory, it is beneficial to look at a point in a large thermal system. Since there will be energy exchange with its thermal surrounding, the energy of this point fluctuates. This thermal reservoir situation can be described using the canonical ensemble. In equilibrium statistical thermodynamics,

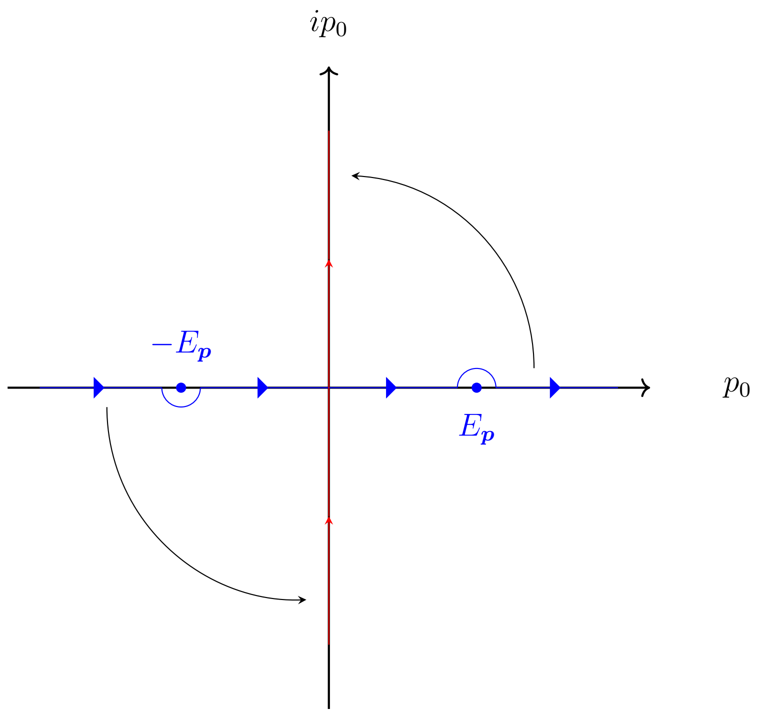


Figure 1.2: The Wick rotation of p_0 contour from Minkowski momentum (real axis) to Euclidean momentum (imaginary axis).

the general density matrix for a canonical ensemble can be defined as

$$\rho(\beta) = e^{-\beta\mathcal{H}}, \tag{1.18}$$

where β represents the inverse of the equilibrium temperature[2] and $\mathcal{H}$ is the Hamiltonian of the particular ensemble (for a canonical ensemble, the dynamical Hamiltonian $H = \mathcal{H}$). Then the partition function of the system can be defined as [9]

$$Z(\beta) = \mathrm{Tr}\, \rho(\beta) = \mathrm{Tr}\, e^{-\beta\mathcal{H}}\,. \tag{1.19}$$

From Eq. (1.19), we can derive the ensemble average (expectation value) of an observable A

$$\langle A \rangle_\beta = Z^{-1}(\beta)\, \mathrm{Tr}\left(\rho(\beta)\, A\right) = \frac{\mathrm{Tr}\left(\, e^{-\beta\mathcal{H}} A\,\right)}{\mathrm{Tr}\, e^{-\beta\mathcal{H}}}, \tag{1.20}$$

[2]We assume that the Boltzmann constant $k_B = 1$; otherwise, $\beta = 1/\left(k_B\mathcal{T}\right)$. In addition, we are using 'nature's units' where $\hbar = c = 1$. Here, temperature has the dimensions of energy.

as well as the average of the correlation function of any two operators,

$$\langle AB\rangle_\beta = Z^{-1}(\beta)\,\mathrm{Tr}\,[\,\rho(\beta)\,AB\,]. \tag{1.21}$$

Denoting the eigenstates of the Hamiltonian as $|n\rangle$, we have

$$\mathcal{H}|n\rangle = E_n|n\rangle. \tag{1.22}$$

In this case, the trace of the product $e^{-\beta\mathcal{H}}A$ can be written as

$$\mathrm{Tr}\,(\,e^{-\beta\mathcal{H}}A\,) = \sum_n e^{-\beta\,E_n}\,\langle n|\,A\,|n\rangle. \tag{1.23}$$

It is easy to see that the zero-temperature limit comes from the ground state of the Hamiltonian. Therefore, the vacuum expectation value (VEV) of the observable A is

$$\lim_{\mathcal{T}\to 0}\mathrm{Tr}\,(\,\rho A\,) = \langle 0|\,A\,|0\rangle, \tag{1.24}$$

where $\mathcal{T}$ is temperature.

Thus, we are viewing Eq. (1.20) as an extension of the zero-temperature theory already known. Equations (1.20) and (1.21) determine expectation values by extending ordinary perturbation theory at the zero-temperature background, and this extended theory is called finite-temperature QFT [1].

1.2.1 Matsubara Formalism

The most direct evaluation method for expectation values such as Eqs. (1.20) and (1.21) for the particular ensembles is using the trace product Eq. (1.23) and standard perturbation theory. However, this straightforward method will be very difficult as the number of states contributing to the sum is large.

The perturbative rules from quantum mechanics for the density operator or even the finite-temperature Feynman rules (which will be discussed later in this section) are expressed in coordinate space, which is usually not very suitable for explicit calculations. The Fourier transform is the mathematical technique used to decompose a set of rules from coordinate space to momentum space. Ideally, all the propagators in the Feynman rules can be Fourier transformed. But the situation here is complicated by thermal effects.

In order to define the contour for thermal time evolution, the density operator is redefined analogous to the approach of Feynman rules at zero temperature:

$$e^{-\beta\mathcal{H}} = e^{-\beta\mathcal{H}_0}\,\mathcal{U}(t_i - i\beta, t_i) = e^{-\beta\mathcal{H}_0}\,T\exp\left[\,i\int_{t_i}^{t_i-i\beta}\int_{\text{all space}}\mathcal{L}_I(\phi_{in}(x))\,dx_0\,d^3x\right], \qquad (1.25)$$

where $\mathcal{H}_0$ is the free-particle Hamiltonian, $\mathcal{U}$ is the time evolution operator, T is the time-ordering symbol, t_i is the time at which the ensemble is prepared in thermal equilibrium, $\phi_{in}(x)$ is the initial field at coordinate x, and $\mathcal{L}_1$ is the interaction term in the Lagrangian. The contour is then $\mathcal{C} = [t_i, +\infty] \cup [+\infty, t_i] \cup [t_i, t_i - i\beta]$ as shown in Fig. 1.3.

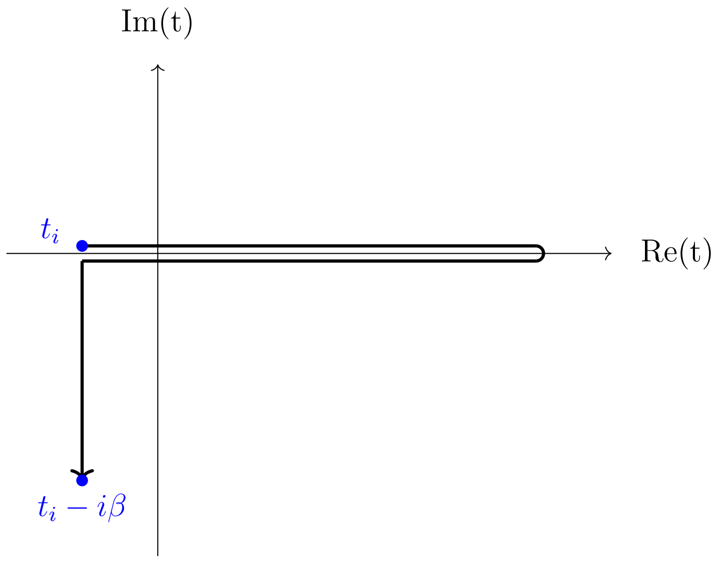


Figure 1.3: The integration contour $\mathcal{C}$ of thermal time for the density operator $e^{-\beta\mathcal{H}}$ in the complex time plane.

It is clear that $\mathcal{C}$ has a nontrivial shape, which makes it difficult to compute Fourier transforms. The solution chosen here is the imaginary time formalism, also called the Matsubara formalism, which provides a diagrammatic approach to sum over discrete imaginary frequencies [1, 10]. Since any physical quantity from a thermal equilibrium system is time-independent, the contour $\mathcal{C}$ can be simplified into the diagram in Fig. 1.4, with variable $\tau \in [\,0, \beta\,)$ and x_0 denoted as $-i\tau$.

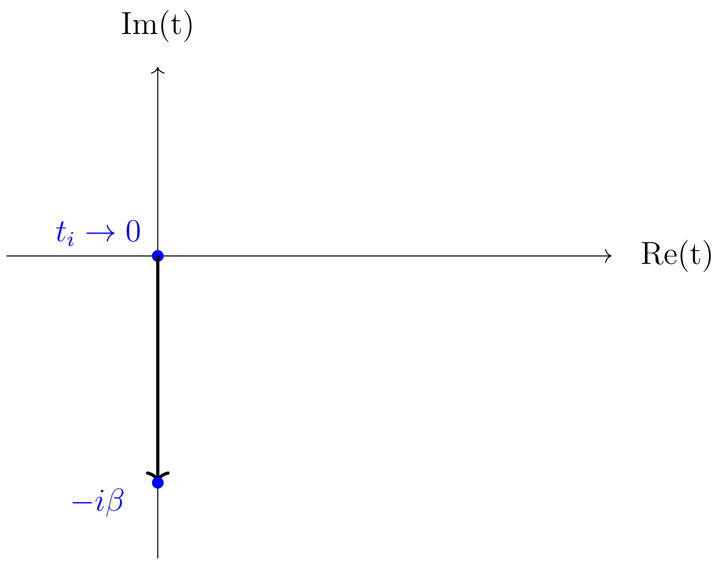


Figure 1.4: Simplified contour $\mathcal{C}$ of thermal time (taking initial thermal time $t_i \to 0$).

Due to the periodicity/anti-periodicity of the Feynman diagram integrand field ϕ [3] the integration can easily be Fourier-transformed into frequency space

$$\mathcal{G}^0(\tau_x, x, \tau_y, y) = \mathcal{T} \sum_{n=-\infty}^{+\infty} \int \frac{d^3\boldsymbol{p}}{(2\pi)^3}\, e^{i\omega_n(\tau_x-\tau_y)} e^{-ip\cdot(x-y)}\, \mathcal{G}^0(\omega_n, \boldsymbol{p}), \tag{1.26}$$

where the Matsubara frequencies are $\omega_n = 2\pi n\mathcal{T}$ for bosons (periodic) and $2\pi(n+\frac{1}{2})\mathcal{T}$ for fermions (anti-periodic) for temperature $\mathcal{T}$. For free bosonic scalar fields,

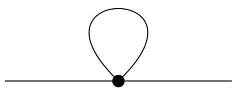
$$\mathcal{G}^0(\omega_n, \boldsymbol{p}) = \frac{1}{\omega_n^2 + \boldsymbol{p}^2 + m^2} \tag{1.27}$$

with ω_n $\mathcal{T}$-dependent. Equation (1.27) gives the free-field propagator of the Feynman rules for perturbative calculations in the Matsubara formalism.

By using the Matsubara formalism, the continuous integration turns into the summation over discrete Matsubara frequencies. The newly derived Feynman rules can be applied to the

[3]This symmetry is known as Kubo-Martin-Schwinger symmetry where all bosonic path-ordered correlators take identical values at the two endpoints of the contour $\mathcal{C}$ [1].

scalar one-loop tadpole topology integral in the $\lambda\phi^4$ theory and it gives



$$
\begin{aligned}
&= \frac{\lambda}{2}\mathcal{T}\sum_n \int \frac{d^3\boldsymbol{p}}{(2\pi)^3}\frac{1}{\omega_n^2+\boldsymbol{p}^{\,2}+m^2} \\
&= \frac{\lambda}{2}\int \frac{d^3\boldsymbol{p}}{(2\pi)^3 2E_{\boldsymbol{P}}}\int_0^\beta d\tau \sum_n \delta(\tau - n\beta)\left[(1+n_B(E_{\boldsymbol{P}}))e^{-E_{\boldsymbol{P}}\tau}+n_B(E_{\boldsymbol{P}})e^{E_{\boldsymbol{P}}\tau}\right] \\
&= \frac{\lambda}{2}\int \frac{d^3\boldsymbol{p}}{(2\pi)^3 2E_{\boldsymbol{P}}}[1+2n_B(E_{\boldsymbol{P}})] \\
&= \lambda\left[\frac{\Lambda^2}{16\pi^2}+\frac{\mathcal{T}^2}{24}+\dots\right],
\end{aligned}
\tag{1.28}
$$

where Λ is an ultraviolet cutoff that restricts the $d^3\boldsymbol{p}$ integration range and $n_B(E_{\boldsymbol{P}}) \equiv 1/\left(e^{\beta E_{\boldsymbol{P}}}-1\right)$ is the Bose-Einstein distribution. The first term in Eq. (1.28) corresponds to the ultraviolet divergence at zero temperature. The second term is ultraviolet finite from the exponential decrease of the Bose-Einstein distribution at large energy.

Thus, in the Matsubara formalism, the temperature dependence is solely in the propagators of the loop integral. The rest of the loop integral (e.g., the vertices) are identical with zero-temperature QFT.

Just as it was introduced in Section 1.1.1, the path integral formalism gives another approach to find thermal field theory Feynman rules other than the Matsubara formalism [9]. If we define $t_1 - t_2 = -i\beta$, then the partition function Eq. (1.4) would look like

$$
Z'[J] = N'\int [d\phi]\ \exp(\mathcal{S}'), \tag{1.29}
$$

where N' is the normalization factor and the action is

$$
\mathcal{S}' = \int_0^\beta dt \int \mathcal{L}' d^3x + \beta\mu N. \tag{1.30}
$$

The contour would look like the one in Fig. 1.4. Both $\mathcal{S}'$ and $\mathcal{L}'$ are in Euclidean space given by the Matsubara formalism. The path integral representation comparing to the original in zero-temperature QFT has a finite interval for the time dimension integration. The Feynman rules are embedded in the path integral and it gives a nontrivial dependence on β [9] providing the temperature effect in the path integral formalism.

1.2.2 Finite-temperature Correlation Functions

The Matsubara formalism gives the means to transform a Feynman integral into its finite-temperature version, as demonstrated above. The propagators are adapted to their finite-temperature version as in Eq. (1.27). We will study the one-loop self-energy for a scalar field in a $\lambda\phi^3$ theory. In this simplest case, we are using spin-zero bosons. The Feynman diagram of the self-energy topology is shown below in Fig. 1.5. In QFT at zero temperature, the Feynman integral for such a diagram can easily be written down by applying the famous Feynman rules [8],

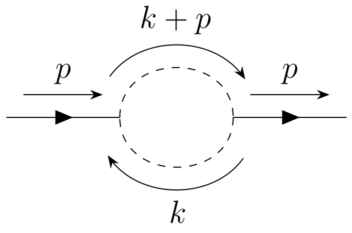


Figure 1.5: The one-loop self-energy Feynman diagram with scalar fields.

$$\Pi_0(\boldsymbol{p}, p_0) = \frac{1}{2}\int \frac{d^4k}{(2\pi)^4}\frac{1}{k^2+{m_1}^2} \times \frac{1}{(k+p)^2+{m_2}^2}, \tag{1.31}$$

where m_1 and m_2 are the masses of the internal fields. A wick rotation has been performed to obtain Eq. (1.31). The subscript of the correlator Π_0 indicates that this is the correlation function for zero temperature.

Now we want to develop the finite-temperature Feynman integral for such topology. The energy integral is now quantized. The temporal k_0 integral needs to be discretized to an infinite summation over integer parameter n due to the implementation of the Matsubara frequencies

$$\int \frac{d^4k}{(2\pi)^4} \to \frac{1}{\beta}\sum_n \int \frac{d^3\boldsymbol{k}}{(2\pi)^3}. \tag{1.32}$$

The correlation function Eq. (1.31) requires adaptation to represent the same one-loop topology with temperature dependence taken into consideration, including the discretization as in Eq. (1.32) and the implementation of the Matsubara frequencies $\omega_n = 2\pi n\mathcal{T}$:

$$\Pi_{\mathcal{T}}(\boldsymbol{p}, p_0) = \frac{1}{2\beta} \sum_{n=-\infty}^{\infty} \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{\frac{4n^2\pi^2}{\beta^2} + \boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{(\frac{2n\pi}{\beta} + p_0)^2 + (\boldsymbol{k} + \boldsymbol{p})^2 + {m_2}^2}, \tag{1.33}$$

where the subscript on the correlator $\Pi_{\mathcal{T}}$ means finite temperature. In the Matsubara formalism, the energy of the external momentum p_0 is assumed to be discrete in finite-temperature correlation function at the values of $\frac{2\pi l}{\beta}$ (l is any integer) [9].

Equation (1.33) will be the core correlation function for the majority of this thesis. The numerical method outlined below is heavily based on the divergences associated with Eq. (1.33).

2 pySecDec Benchmarking for Feynman Diagrams at Zero Temperature

The program package pySecDec is designed to numerically calculate loop integrals in QFT using the sector decomposition approach [6].

One full calculation of a Feynman diagram requires two different Python documents. The "generate" file contains part of the information of one particular diagram such as the topology, the dimensionality and the prefactor. The remaining "integrate" file requires the input of the numerical values of particle masses and kinematics for the calculation. Before developing strategies to apply pySecDec to finite-temperature QFT, multiple zero-temperature loop integrations were done through the Plato cluster at the University of Saskatchewan and compared against analytical results [11, 12] as summarized in Ref. [13].

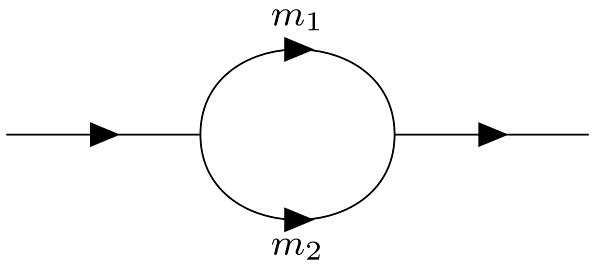


Figure 2.1: One-loop self-energy Feynman diagram (TBI) with scalar fields.

The example used in the pySecDec numerical calculation and reference comparison is the Feynman diagram topology representing the one-loop self-energy integral (TBI) as shown in Fig. 2.1. The TBI integral uses TARCER notation which reduces the general propagator integral expressions to basic integrals in `MATHEMATICA` [14]. The TBI integral, following the TARCER convention, is

$$\mathsf{TBI}\left[d, q^2, \{\{\nu_1, m_1\}, \{\nu_2, m_2\}\}\right] = \frac{1}{\pi^{\frac{d}{2}}} \int \frac{d^d k}{[k^2 - m_1{}^2 + i\zeta]^{\nu_1}[(k-q)^2 - m_2{}^2 + i\zeta]^{\nu_2}}, \quad (2.1)$$

where $\zeta \to 0^+$, m_1 and m_2 are the masses of the two internal lines and the parameters ν_1 and ν_2 are propagator weights, i.e., exponents.

The masses of the two internal lines (m_1 and m_2) in the Feynman diagram above can be specified in pySecDec. To simplify the numerical calculations, we rewrote Eq. (1.31) by multiplying numerators and denominators of both propagators by $\frac{1}{p^2}$

$$\begin{aligned}\Pi_0(\boldsymbol{p}, p^0) &= \frac{\lambda^2}{2} \int \frac{d^4k}{(2\pi)^4} \frac{1/p^2}{\left(\frac{k^2}{p^2} + \frac{{m_1}^2}{p^2}\right)} \times \frac{1/p^2}{\left(\frac{(k+p)^2}{p^2} + \frac{{m_2}^2}{p^2}\right)} \\ &= \frac{\lambda^2}{2(2\pi)^4} \int d^4\left(\frac{k}{p}\right) \frac{1}{\left(\frac{k}{p}\right)^2 + \frac{{m_1}^2}{p^2}} \times \frac{1}{\left(\frac{k}{p}+1\right)^2 + \frac{{m_2}^2}{p^2}}.\end{aligned} \tag{2.2}$$

Since we are using the particle physics convention of $\hbar = c = 1$, both particle masses and momenta have the dimensions of energy. All the expressions in brackets are dimensionless. So technically, what only matters for numerical benchmarking are the ratios $\frac{{m_1}^2}{p^2}$ and $\frac{{m_2}^2}{p^2}$. In Section 2.2, masses are chosen as the mass of a charm quark and a bottom quark for the numerical investigation. Numerical calculations of the loop integral were performed along a grid of points lying just above the positive real axis in the complex q^2-plane where q is the external momentum in Fig. 1.5.

2.1 TBI Massless Integral

Corresponding to the diagram of Fig. 2.1, the analytical result of the loop integral with both propagator masses zero is taken from Ref. [12] and simplified in Ref. [13],

$$\begin{aligned}&\mathsf{TBI}\,[\,4+2\epsilon, q^2, \{\nu_1, 0\}, \{\nu_2, 0\}\,] \\ &\quad= \frac{i}{(4\pi)^2}\left[-\frac{q^2}{4\pi}\right]^{\epsilon} (q^2)^{2-\nu_1-\nu_2}\, \frac{\Gamma\,[2-\nu_1+\epsilon]\,\Gamma\,[2-\nu_2+\epsilon]\,\Gamma\,[\nu_1+\nu_2-2-\epsilon]}{\Gamma\,[\nu_1]\,\Gamma\,[\nu_2]\,\Gamma\,[4-\nu_1-\nu_2+2\epsilon]},\end{aligned} \tag{2.3}$$

where the space-time dimension is $d = 4+2\epsilon$, and we expand about $\epsilon = 0$. Coding Eq. (2.3) in `MATHEMATICA` and setting ν_1 and ν_2 to 1, we have the analytical result of a massless TBI integral. In order to compare with pySecDec imported data results, the analytical formula required an extra factor of $(2\pi)^{4+2\epsilon}(-\frac{i}{\pi^{2+\epsilon}})$. This additional factor is from the convention of pySecDec. A scalar Feynman graph G in d dimensions at one loop with N propagators, where the propagators can have powers of ν_j, has the momentum-space representation of

$$G = \int d^d\kappa \, \frac{1}{\prod_{j=1}^{N} P_j^{\nu_j}(\{k\}, \{q\}, m_j{}^2)}, \tag{2.4}$$

$$d^d\kappa = \frac{\mu^{4-d}}{i\pi^{\frac{d}{2}}} d^d k, \; P_j^{\nu_j} = ((k - q_j)^2 - m_j{}^2 + i\zeta), \tag{2.5}$$

where μ denotes the renormalization scale, q_j are external momenta, and k is the loop momentum [6]. By comparison, the convention in Ref. [11] has the integral expression of

$$G = \lim_{\zeta \to 0^+} \frac{1}{\mu^{d-4}} \int \frac{d^d k}{(2\pi)^d} \frac{1}{[k^2 - m^2 + i\zeta]^n}. \tag{2.6}$$

Equations (2.4) and (2.6) give a coefficient difference of

$$\frac{1}{i\pi^{\frac{d}{2}}} \times \left[\frac{1}{(2\pi)^d}\right]^{-1} = (2\pi)^d \left(-\frac{i}{\pi^{d/2}}\right), \tag{2.7}$$

where $d = 4 + 2\epsilon$.

Importing the pySecDec integration output for the one-loop self-energy Feynman integral, we have the comparison graph shown in Fig. 2.2. The method of comparing the analytic and numerical results is through extracting the coefficients of ϵ^{-1} , ϵ^0, and ϵ^1 from the ϵ expansion of the results for both real and imaginary parts.

According to Fig. 2.2, the pySecDec output of the TBI massless integral matches well with the Ref. [12] analytic formula, providing a reliable benchmark on the implementation of pySecDec, including the necessary prefactors.

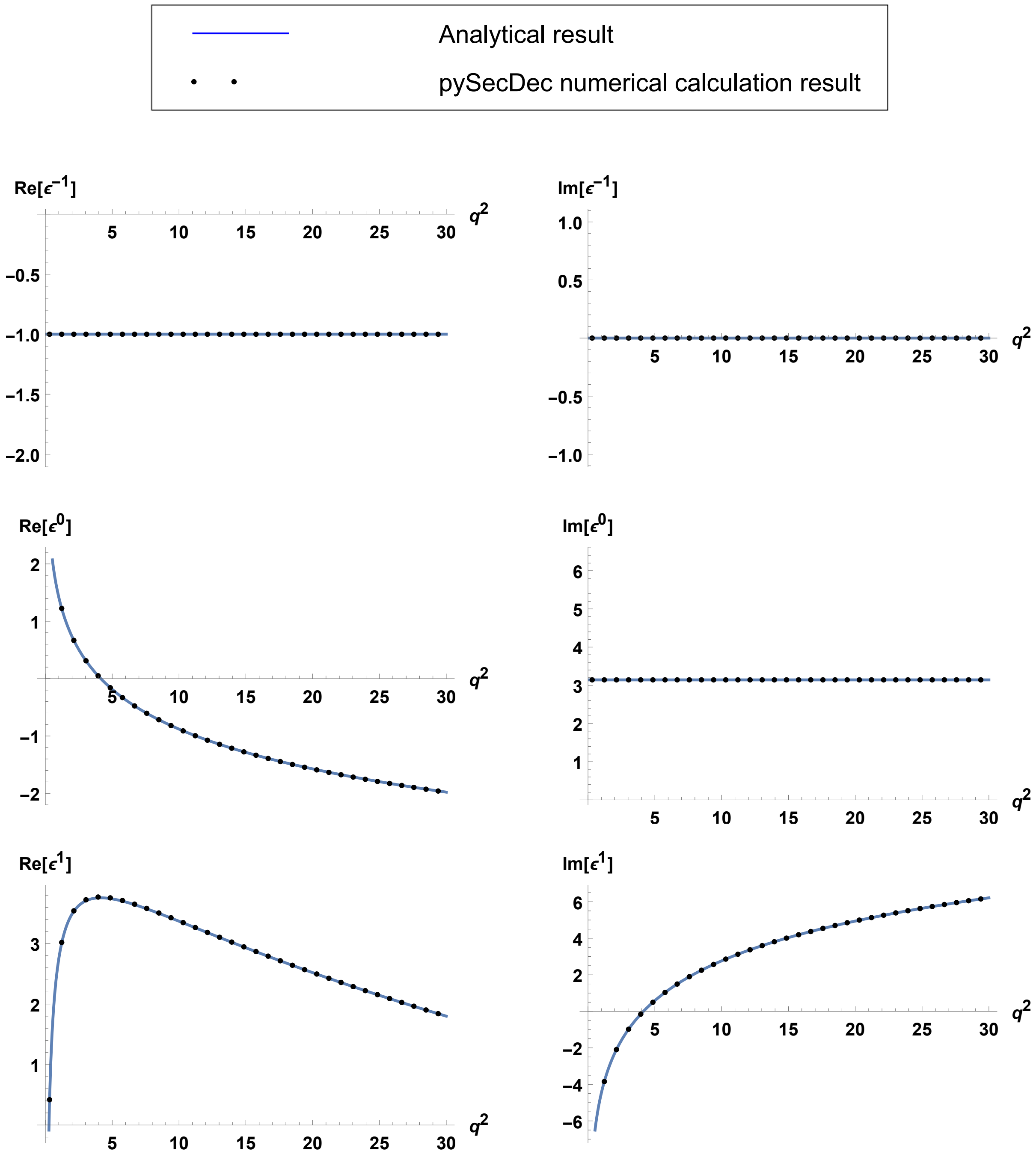


Figure 2.2: Comparison between the ϵ^{-1} , ϵ^{0} and ϵ^{1} coefficients comparison graphs for analytical and numerical results for the one-loop self-energy massless integral Eq. (2.3). The vertical axes labels represent terms of the corresponding coefficients (e.g., the coefficients comparison of the real part of ϵ^{-1} is shown in the top-left graph). The errors between the analytical and numerical results are at the scale of 10^{-4}, while the uncertainties from the numerical results are at the scale of 10^{-2}. The errors are much smaller than the numerical uncertainties.

2.2 TBI Massive Case

The TBI analytical results for massive internal lines come in two types. One of them are integrals with equal masses and the other are integrals with unequal masses, including the case of one massless internal line and one massive line. Starting with the latter case, the loop integral in terms of the Gauss hypergeometric function is [12, 13]

$$\begin{aligned}\mathsf{TBI}\,[\,4+2\epsilon, q^2, \{\nu_1, m\},\{\nu_2, 0\}\,] = {} & \frac{i}{(4\pi)^2}\,[-\frac{q^2}{4\pi}]^{\frac{d}{2}-2}\,(q^2)^{2-\nu_1-\nu_2}\,z^{\nu_1+\nu_2-\frac{d}{2}} \\ & \frac{\Gamma\,[\frac{d}{2}-\nu_1]\,\Gamma\,[\nu_1+\nu_2-\frac{d}{2}]}{\Gamma\,[\nu_1]\,\Gamma\,[\frac{d}{2}]}\,{}_2F_1\,[\nu_1+\nu_2-\frac{d}{2}, \frac{d}{2}-\nu_1; \frac{d}{2}; z],\end{aligned} \tag{2.8}$$

where $z = \frac{1}{1-\frac{m^2}{q^2}}$ and m is the one nonzero propagator mass. Equation (2.8) and the pySecDec numerical values were compared in `MATHEMATICA` as in the previous example. However, since the hypergeometric function has ϵ-dependent parameters, the analytical result (${}_2F_1$ factor) needs to be expanded around ϵ using the `MATHEMATICA` HypExp package [15]. In this pySecDec integration, the charm quark mass ($m = 1.27\,\mathrm{GeV}$)[16] was used to generate numerical data[1]. Similarly, the analytical formula Eq. (2.8) was multiplied by a factor of $(2\pi)^{4+2\epsilon}(-\frac{i}{\pi^{2+\epsilon}})$ from Eq. (2.7). From Fig. 2.3, the pySecDec result agrees with Ref. [13].

Another type of the TBI integral with massive internal lines involves two internal propagators with the same mass. Once again, the charm mass was applied in the following loop integral (Ref. [11, 13]):

$$\begin{aligned}\mathsf{TBI}\,[\,4+2\epsilon, q^2, \{\nu_1, m\},\{\nu_2, m\}\,] = {} & \frac{i}{(4\pi)^2}\left[-\frac{m^2}{4\pi}\right]^{\frac{d}{2}-2}(-m^2)^{2-\nu_1-\nu_2} \\ & \times\frac{\Gamma\,[\nu_1+\nu_2-\frac{d}{2}]}{\Gamma\,[\nu_1+\nu_2]}\,{}_3F_2\left[\begin{matrix}\nu_1 & \nu_2 & \nu_1+\nu_2-\frac{d}{2} \\ \frac{1}{2}\,(\nu_1+\nu_2) & \frac{1}{2}\,(\nu_1+\nu_2+1) & \end{matrix}; \frac{q^2}{4m^2}\right].\end{aligned} \tag{2.9}$$

As shown in Fig. 2.4, the generated graphs show excellent agreement as in the previous two cases. The same calculations for TBI massive integrals were done with bottom quark mass and the results are included in Appendix A. Therefore, pySecDec's outputs show reliable loop integral calculation performance regarding one-loop self-energy integrals. These bench-

[1]The unit of particle mass is eV/c^2.

marks have also allowed determination of the proper conversion factors needed to transform pySecDec output into comparable analytical data.

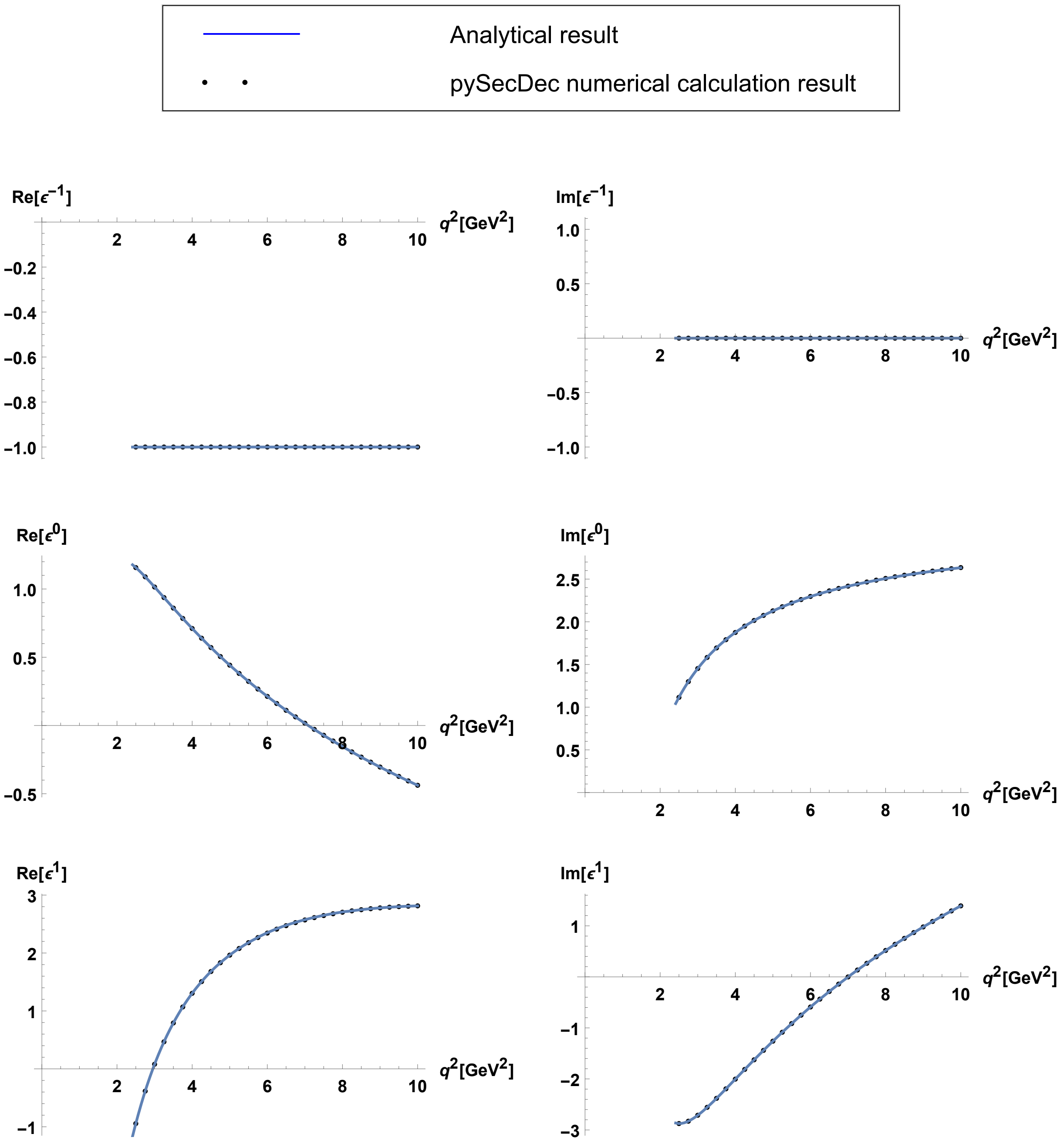


Figure 2.3: Comparisons between the ϵ^{-1}, ϵ^0 and ϵ^1 coefficients of analytical and numerical results for the one-loop self-energy integral Eq. (2.8) ($m_1 = 1.27\,\text{GeV}$, $m_2 = 0$). Similar to Fig. 2.2, the errors are magnitudes smaller than the numerical uncertainties.

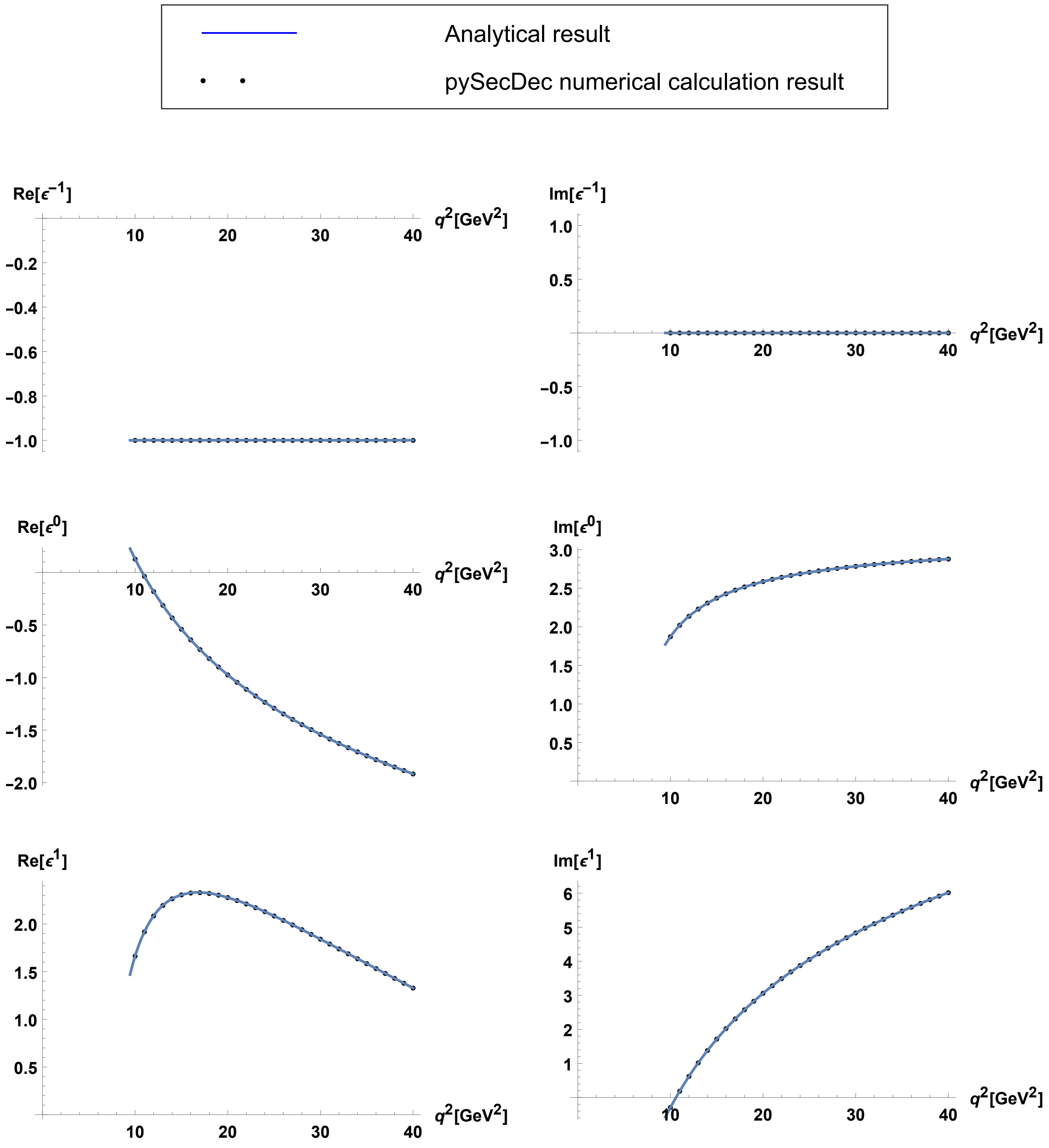
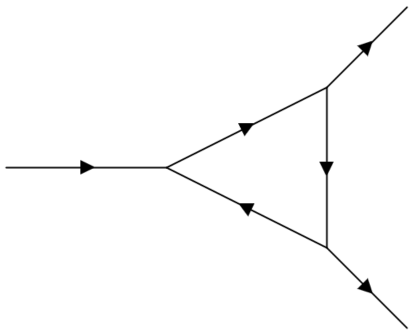


Figure 2.4: Comparison between the ϵ^{-1}, ϵ^{0} and ϵ^{1} coefficients of analytical and numerical results for the one-loop self-energy integral Eq. (2.9) ($m_1 = m_2 = 1.27\,\mathrm{GeV}$). Similar to Fig. 2.2, the errors are magnitudes smaller than the numerical uncertainties.

2.2.1 Cutting Rules

After benchmarking pySecDec against one-loop self-energy Feynman integrals, more trials and tests were performed using the Plato platform including one/two-loop three/four-point functions as in Figs. 2.5 and 2.6.

The processing time for any "generate" file with non-zero masses was significantly longer than any massless Feynman integral. Also two-loop functions requires more time to run integrations compared to one-loop functions. All of the integrations for three-point and four-point functions are calculated at their symmetric points ($p_1^2 = p_2^2 = p_3^2 = p_4^2$) as outlined below.



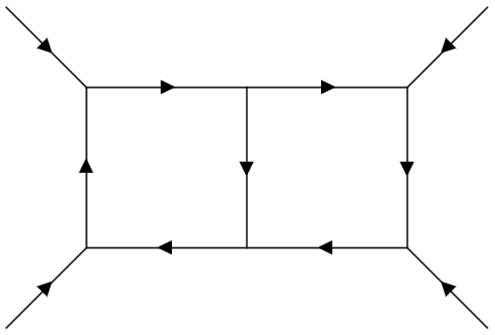


Figure 2.5: Examples of the loop integrals that have been calculated using pySecDec

As the general analytical formula for N-point one-loop integrals from Ref. [11] is difficult to implement due to complicated notations and expressions, the cutting rules [8] were used to benchmark 4-point one-loop pySecDec results. By calculating a four-point one-loop function with identical internal masses (Fig. 2.6) along the above-cut contour, the imaginary coefficient for ϵ^0 is plotted in Fig. 2.7. The coefficient remains zero until a certain threshold value of q^2. The coefficient then takes off and continues increasing as q^2 increases. This behavior was expected as the integral has a branch point on the real axis followed by a branch cut.

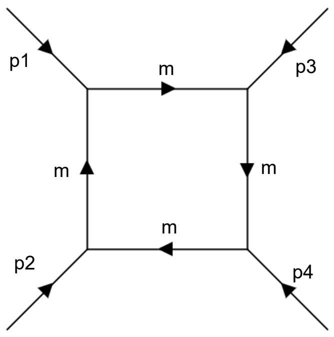


Figure 2.6: Four-point, one-loop topology with equal internal masses

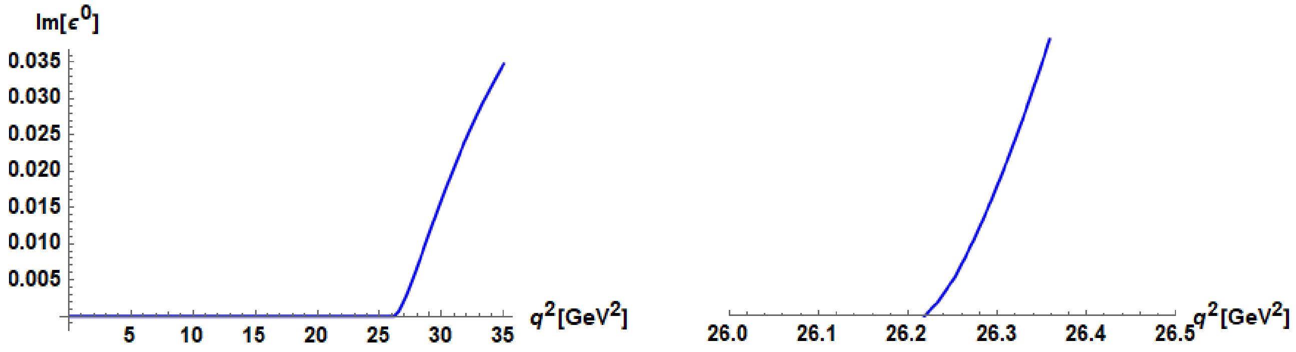
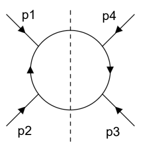
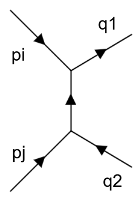


Figure 2.7: The ϵ^0 coefficient for the four-point, one-loop Feynman integral with equal propagator masses ($m = 4.18\,\text{GeV}$) along with an expanded plot on the RHS. The imaginary part remains zero until $q^2 \gtrapprox 26.21\,\text{GeV}^2$.

The cutting rules (Cutkosky rules) [8] are generally used to find the imaginary part of a Feynman diagram. For the same Feynman diagram, the threshold for the integral to have an imaginary part can be retrieved by using the cutting rules,

$2\,\mathrm{Im}\Big($  $\Big) = \int d\Pi \,\Big|$  $\Big|^2 .$ (2.10)

where $\int d\Pi$ represents all possible intermediate-state particles.

Since we are working at the symmetric point, and $p_4 = p_1 + p_2 + p_3$, we can easily derive the expression

$$-\,p_i \cdot p_j = \frac{p_1^2}{3} = \frac{p_2^2}{3} = \frac{p_3^2}{3} = \frac{q^2}{3}, \tag{2.11}$$

where $i, j = 1, 2, 3, i \neq j$ and q^2 is defined as $q^2 = p_1^2 = p_2^2 = p_3^2$. From the right hand side of Eq. (2.10), we have

$$\begin{aligned} p_i + p_j &= q_1 + q_2, \\ \Rightarrow (p_i + p_j)^2 &= (q_1 + q_2)^2, \\ \Rightarrow p_i^2 + p_j^2 + 2p_i \cdot p_j &= q_1^2 + q_2^2 + 2q_1 \cdot q_2. \end{aligned} \tag{2.12}$$

As all the internal lines have the same mass m,

$$2\,q^2 - \frac{2}{3}\,q^2 = 2m^2 + 2\,q_1 \cdot q_2. \tag{2.13}$$

Since $q_1 \cdot q_2 = E_1 E_2 - \boldsymbol{q_1} \cdot \boldsymbol{q_2} > 0$, the relation between q^2 and mass is then

$$q^2 > \frac{3}{2}\,m^2\,. \tag{2.14}$$

The cutting rules allow us to find the kinematical region where the internal lines can be on shell [8]. In this case, the region is $q^2 > \frac{3}{2}m^2$ which agrees with the pySecDec integration result in Fig. 2.7 as

$$\frac{3}{2}\,m^2 = \frac{3}{2} \times (4.18\;\text{GeV})^2 \approx 26.209. \tag{2.15}$$

The cutting rules can be generalized to finite temperature [9] and could also be applied to future finite-temperature N-point integral benchmarking as well as for new results.

3 Methodology

The main quantity of interest of the thesis methodology is Π_s, the difference between the finite-temperature ($\Pi_{\mathcal{T}}$) and the zero-temperature (Π_0) correlation function, also named finite-temperature correction,

$$\Pi_s = \Pi_{\mathcal{T}} - \Pi_0. \tag{3.1}$$

For the one-loop self-energy topology that we are working on in this thesis (Fig. 2.1), the finite-temperature correction can be found using Eqs. (1.33) and (1.31). If both $\Pi_{\mathcal{T}}$ and Π_0 are convergent, we can just simply subtract them to determine the correction term Π_s. In the case of 2-dimensional and $d = 3$ spacetime, both $\Pi_{\mathcal{T}}$ and Π_0 are convergent, and the calculations went as expected without any mathematical problems (those results will be demonstrated and discussed in Chapter 4). When we climbed up to $d = 4$ spacetime (1 temporal dimension and 3 spatial dimensions), we ran into divergences in both $\Pi_{\mathcal{T}}$ and Π_0. In this chapter, we will face the long-term enemies of loop integral calculations, the divergences, and discuss the development of a numerical method for calculating finite-temperature effects of the self-energy topology.

3.1 The Cut-off Method

Most quantum field theories do not hold up to an arbitrarily high energy limit as every physical theory has its own range of validity. Thus, physicists developed regularization methods to deal with these momentum integrations. The concept of a cutoff is not foreign to QFT, rather it is one of the most straightforward and intuitive regulators for ultraviolet (UV) divergences. An example can be found in Ref. [17]. To regularize the divergence for a loop integral in the continuum limit, a lattice cutoff at momentum $k = \frac{1}{a}$ while $a \to 0$ was applied as the upper limit of the integration:

$$\begin{aligned}\Sigma_{inf} &= \int d^4k \frac{1}{(k^2+\mu^2)^2} \quad \text{before applying cut-off} \\ \Rightarrow \Sigma_{cut-off} &= \int_{k_0^2+\boldsymbol{k}^2<\frac{1}{a^2}} dk_0 \, d^3\boldsymbol{k} \, \frac{1}{(k_0^2+\boldsymbol{k}^2+\mu^2)^2} \quad \text{after applying cut-off} \\ &= 2\int_0^{\frac{1}{a}} dk \frac{k^3}{(k^2+\mu^2)^2} \\ &= 2 \ln \frac{1}{a} \quad \text{as } a \to 0.\end{aligned} \tag{3.2}$$

The hard cut-off neglects the domain of integration beyond a certain value (often represented by the symbol Λ) and 'regularizes' the integral in the simplest way. The disadvantage of this naive approach is also very obvious. It breaks Lorentz invariance and most symmetries in the theory [18]. However, since thermal effects already break Lorentz symmetry by providing a preferred reference frame,[1] we can use the cut-off method to regularize divergences associated with the temporal infinite summation in $\Pi_{\mathcal{T}}$ in Eq. (1.33), and with the k_0 integration in Π_0 in Eq. (1.31). In the spatial dimensions, which are still continuous integrations, pySecDec will calculate integrals using dimensional regularization [19] which obeys gauge invariance.

3.2 Large n Behavior

We need to further understand the thermal correlation functions. Looking at the thermal correlation function of the self-energy Feynman integral (Eq. (1.33)), it is easy to find the approximate form when $|n|$ is large compared to the field masses m_1, m_2 and the external momentum $(p_0^E, \boldsymbol{p})$.

$$\begin{aligned}\Pi_{\mathcal{T}} &= \frac{1}{2\beta}\sum_n \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{(\frac{4n^2\pi^2}{\beta^2})+\boldsymbol{k}^2+{m_1}^2} \times \frac{1}{(\frac{2n\pi}{\beta}+p_0^E)^2+(\boldsymbol{k}+\boldsymbol{p})^2+{m_2}^2} \\ &\approx \frac{1}{2\beta}\sum_n \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{(\frac{2n\pi}{\beta})^2+\boldsymbol{k}^2} \times \frac{1}{(\frac{2n\pi}{\beta})^2+\boldsymbol{k}^2}, \quad \text{for } |n| \gg 1 \\ &= \frac{1}{2\beta}\sum_n \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{\left(m'^2+\boldsymbol{k}^2\right)^2}\end{aligned} \tag{3.3}$$

[1]This is particularly evident in the Matsubara formalism as the temporal dimension is discretized.

where $m' = \frac{2n\pi}{\beta}$. For the expression in the last line of Eq. (3.3), there is an identity that can carry the approximation process further [20]

$$\Phi(m,d,B) = \int \frac{\mathrm{d}^d\mathbf{k}}{(2\pi)^d} \frac{1}{\left(\mathbf{k}^2+m^2\right)^B} = \frac{1}{(4\pi)^{\frac{d}{2}}} \frac{\Gamma\left(B-\frac{d}{2}\right)}{\Gamma(B)} \frac{1}{\left(m^2\right)^{B-\frac{d}{2}}}. \tag{3.4}$$

In the case of Eq. (1.33), $d = 3-2\epsilon$. The convention for ϵ is temporarily changed here to align with Ref. [20]. Using Eq. (3.4), we can rewrite Eq. (3.3) as the following expression:

$$\begin{aligned} \Pi_{\mathcal{T}} &\approx \frac{1}{2\beta} \sum_n \underbrace{\frac{1}{(4\pi)^{\frac{3}{2}+\epsilon}} \frac{\Gamma\left(\frac{1}{2}+\epsilon\right)}{1}}_{\text{expand } \epsilon \text{ to } \mathcal{O}(\epsilon)} \frac{1}{\left(\frac{2n\pi}{\beta}\right)^{2(1+2\epsilon)}} \\ &\approx \frac{1}{2\beta} \sum_n \frac{\sqrt{\pi}}{(4\pi)^{\frac{3}{2}}} \frac{\beta}{2\pi\left(n^2\right)^{\frac{1}{2}+\epsilon}} \\ &= \sum_n \frac{1}{32\pi^2} \frac{1}{|n|^{1+2\epsilon}}. \end{aligned} \tag{3.5}$$

So we have all the individual terms of the summation for $\Pi_{\mathcal{T}}$ as

$$\Pi_{\mathcal{T}} \approx \sum_n a_n, \tag{3.6}$$

$$a_n \approx \frac{1}{32|n|\pi^2}, \; |n| \gg 1. \tag{3.7}$$

Equation (3.7) is a very promising relation. It means that the individual terms from finite-temperature correlation functions that are being summed are gradually becoming smaller and less significant in the sense of numerical calculation. Also, because of the $\frac{1}{|n|}$ behavior, the series does not converge. An upper limit $n_{\max}$ provides a suitable regulation method to apply to the divergent $\Pi_{\mathcal{T}}$. In Fig. 3.1, we plot numerical confirmation of Eq. (3.7). As Eq. (3.7) is independent of any momentum or mass values, random values are chosen for the pySecDec numerical calculations. The 'reverse' Wick rotation method needed to calculate $\Pi_{\mathcal{T}}$ will be outlined in Section 3.3.

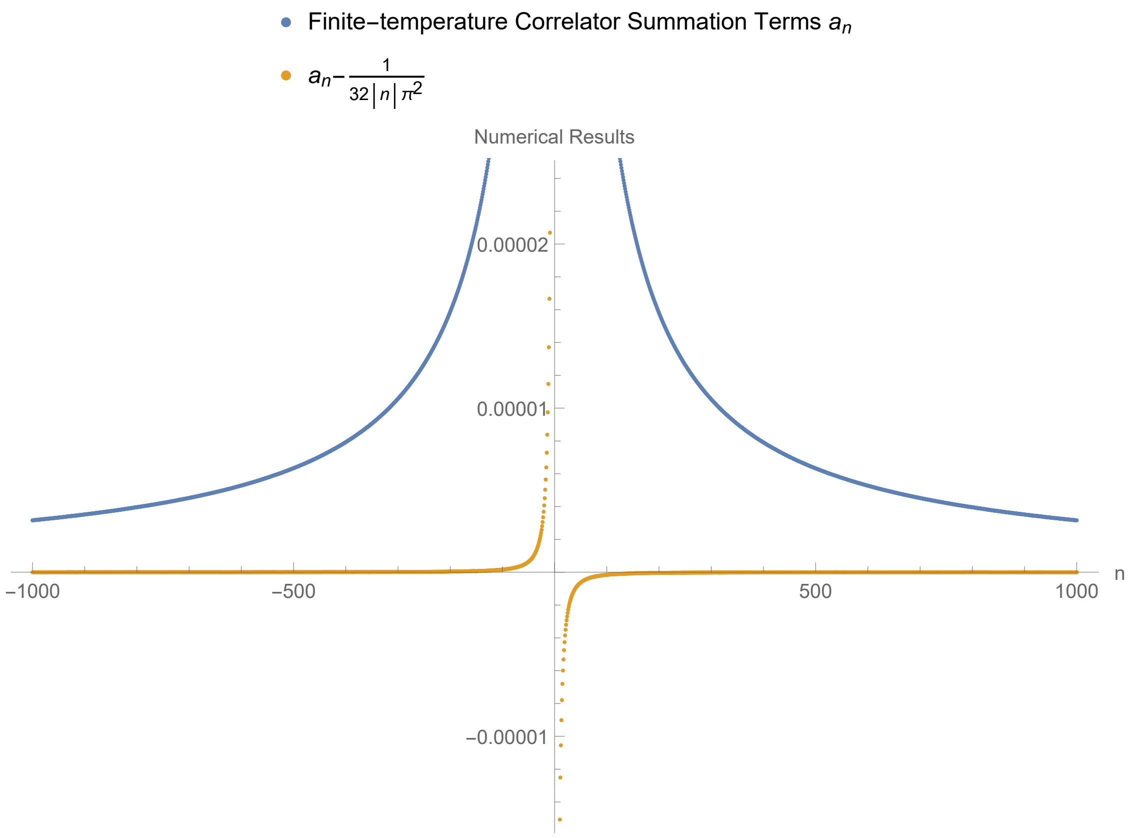


Figure 3.1: The pySecDec-computed numerical finite-temperature correlation function terms a_n (blue) are compared with the approximate value $a_n \approx \frac{1}{32|n|\pi^2}$ by calculating the difference $\Pi_\mathcal{T} - a_n$ as a function of n. The finite-temperature terms were calculated with the parameter values of $m_1 = m_2 = 1.1, p_0^E = \frac{2\pi}{\beta}, \beta = 0.3, \boldsymbol{p}^2 = 5$.

We can see from Fig. 3.1 that a_n is in good agreement with the approximate value of $\frac{1}{32|n|\pi^2}$ when n is large ($|n| \gtrsim 100$). Also, Fig. 3.1 demonstrates that the pre-factor (see Eq. (2.7)) that was implemented in pySecDec is correct. The relation $a_n \sim \frac{1}{|n|}$ proves that $\Pi_\mathcal{T}$ is divergent as the series $\sum_n \frac{1}{|n|}$ diverges. To regulate this divergence, the $\Pi_\mathcal{T}$ series for the self-energy topology will be cut off at a large value of $n_{\max} = A$. We will chose A through numerical experiments in Chapter 4, where A will be related to the k_0 cut-off within Π_0. Thus,

$$\Pi_\mathcal{T}(\boldsymbol{p}, p_0^E) = \frac{1}{2\beta} \sum_{n=-A}^{A} \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{\omega_n^2 + \boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{(\omega_n + p_0^E)^2 + (\boldsymbol{k}+\boldsymbol{p})^2 + {m_2}^2}, \tag{3.8}$$

where $A \to \infty$.

3.3 The 'Reverse' Wick Rotation

The next step is to put the cut-off discussion aside and look at the self-energy finite-temperature correlation function Eq. (1.33) itself. We still need to find a method to numerically calculate it using pySecDec despite the integral not being in the form of a Feynman integral. We regulate the temporal dimension divergence in $\Pi_\mathcal{T}$ with the cut-off method, and we use the pySecDec dimensional-regularization regulator for the spatial integrations:

$$\Pi_\mathcal{T}(\boldsymbol{p}, p_0^E) = \frac{1}{2\beta}\sum_{n=-\infty}^{\infty}\int \frac{d^3\boldsymbol{k}}{(2\pi)^3}\frac{1}{\omega_n^2+\boldsymbol{k}^2+{m_1}^2}\times\frac{1}{(\omega_n+p_0^E)^2+(\boldsymbol{k}+\boldsymbol{p})^2+{m_2}^2}, \tag{3.9}$$

where $\omega_n = \frac{2n\pi}{\beta}$ is the Matsubara frequency and the superscripts E are stating that this function is in Euclidean space. As discussed in Section 3.2, the summation over n diverges even with $\Pi_\mathcal{T}$ calculated using dimensional regularization. Hence the n range is chosen to be $-A \leq n \leq A$ corresponding to a cut-off. As for the spatial integrations in Eq. (3.9), we can still utilize pySecDec to numerically evaluate them. The spatial integral in Eq. (3.9) is expressed as

$$I(\boldsymbol{p}\,;\Lambda_1,\Lambda_2) = \int \frac{d^D\boldsymbol{k}}{(2\pi)^D}\frac{1}{\boldsymbol{k}^2+{\Lambda_1}^2}\times\frac{1}{(\boldsymbol{k}+\boldsymbol{p})^2+{\Lambda_2}^2}, \tag{3.10}$$

where D is defined as the number of spatial dimensions. Since d is generally used to represent the number of space-time dimensions of a Feynman integral, we have $d = D + 1$. With Eq. (3.10), Eq. (3.9) can be written as

$$\Pi_\mathcal{T}(\boldsymbol{p}, p_0^E) = \frac{1}{2\beta}\sum_{n=-\infty}^{\infty} I(\boldsymbol{p}\,; {m_1}^2+\omega_n^2, m_2^2+(p_0^E+\omega_n)^2). \tag{3.11}$$

The spatial integral in Eq. (3.11) is a partial Feynman integral as it includes the spatial dimensions but not the temporal dimension. But it cannot be input directly into pySecDec, because pySecDec calculates for space-time dimensions in Minkowski space, not pure spatial integrals. In order to evaluate the function I using pySecDec, we need to perform manipulations on the momentum to give I a form suitable for pySecDec. We define momentum components as

$$p_1^m = ip_1,\ k_1^m = ik_1,\ dk_1^m = i\,dk_1, \tag{3.12}$$

where p_1 and k_1 are the first components of the spatial momentum vectors $\boldsymbol{p}$ and $\boldsymbol{k}$ respectively from the integral I.[2] The superscript m in Eq. (3.12) represents Minkowski. Now, Eq. (3.10) becomes

$$\begin{aligned} I(\boldsymbol{p}\,;\Lambda_1,\Lambda_2) &= \int \frac{d^D\boldsymbol{k}}{(2\pi)^D} \frac{1}{\boldsymbol{k}^2+{\Lambda_1}^2} \times \frac{1}{(\boldsymbol{k}+\boldsymbol{p})^2+{\Lambda_2}^2} \\ &= -i \int \frac{dk_1^m\, dk_2\, dk_3 \ldots dk_D}{(2\pi)^D} \\ &\quad \times \frac{1}{-(k_1^m)^2+k_2^2+k_3^2+\ldots+k_D^2+\Lambda_1^2} \\ &\quad \times \frac{1}{-(k_1^m+p_1^m)^2+\ldots+(k_D+p_D)^2+\Lambda_1^2} \\ &= -i \int \frac{d^Dk^m}{(2\pi)^D} \frac{1}{k^m\cdot k^m-\Lambda_1^2} \times \frac{1}{(k^m+p^m)\cdot(k^m+p^m)-\Lambda_2^2}, \end{aligned} \tag{3.13}$$

where

$$k^m\cdot k^m = (k_1^m)^2-k_2^2-k_3^2-\ldots-k_D^2, \tag{3.14}$$

$$(k^m+p^m)\cdot(k^m+p^m) = (k_1^m+p_1^m)^2-\ldots-(k_D+p_D)^2, \tag{3.15}$$

and D represents the number of integrated spatial dimensions. Equation (3.13) now is in the form of a Minkowski Feynman integral and can be numerically computed on pySecDec, where Λ_1 and Λ_2 implicitly include an $i\zeta$ term as in Eq. (1.31). So we have

$$(k_1^m)^2 = (ik_1)^2 = -k_1^2 \text{ and } (p_1^m)^2 = (ip_1)^2 = -p_1^2. \tag{3.16}$$

Note that a standard Wick rotation of Eq. (3.13) $k_1^m \to ik_1$ (see Ref. [8]) justifies the definitions in Eq. (3.12). From Eq. (3.16), we also have the relations

$$k^m\cdot k^m = -k_1^2-k_2^2-k_3^2-\ldots-k_D^2 = -\boldsymbol{k}^2, \tag{3.17}$$

$$(k^m+p^m)\cdot(k^m+p^m) = -(k_1+p_1)^2-\ldots-(k_D+p_D)^2 = -(\boldsymbol{k}+\boldsymbol{p})^2. \tag{3.18}$$

[2]The choice of sign in Eq. (3.12) will be justified below.

Similarly,

$$p^m \cdot p^m = -p_1^2 - p_2^2 - p_3^2 - \ldots - p_D^2 = -\boldsymbol{p}^2. \tag{3.19}$$

Equations (3.17), (3.18) and (3.19) can be referred to when inputting momentum values of this Minkowski-space-form integral to pySecDec. Thus, we have an expression for the finite-temperature correlation function for the purpose of pySecDec numerical computation.

$$\begin{aligned}\Pi_{\mathcal{T}}(\boldsymbol{p}, p_0^E) &= \frac{1}{2\beta} \sum_{n=-\infty}^{\infty} \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{\omega_n^2 + \boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{(\omega_n + p_0^E)^2 + (\boldsymbol{k}+\boldsymbol{p})^2 + {m_2}^2} \\ &= -i\, \frac{1}{2\beta} \sum_{n=-\infty}^{\infty} \int \frac{d^3k^m}{(2\pi)^3} \frac{1}{(k^m)^2 - \Lambda_1^2} \times \frac{1}{(k^m + p^m)^2 - \Lambda_2^2},\end{aligned} \tag{3.20}$$

where $\boldsymbol{p} = (p_1, p_2, p_3)$, $p_1^m = ip_1$, $k_1^m = ik_1$, $\Lambda_1^2 = {m_1}^2 + \omega_n^2 + i\zeta$ and $\Lambda_2^2 = m_2^2 + (p_0^E + \omega_n)^2 + i\zeta$. With the integrals sorted in Eq. (3.11), we will turn to the summation over n in the next section.

3.4 The Subtraction

The divergent sum in the finite-temperature correlation function $\Pi_{\mathcal{T}}$ denies us the possibility of subtractions between fully computed finite-temperature and zero-temperature correlators. Yet, from the previous section, a solution with a cut-off regulator and a 'reverse' Wick rotated integral was proposed for the finite-temperature case. Now we will work with the zero-temperature correlator and see if we can develop a methodology to find the finite-temperature corrections of the one-loop self-energy topology.

Taking the $\beta \to \infty$ ($\mathcal{T} \to 0$) limit of Eq. (1.33) via

$$\lim_{\beta\to\infty} \frac{1}{\beta} \sum_n \to \frac{1}{2\pi} \int dk_0^E, \tag{3.21}$$

we find

$$\Pi_0(\boldsymbol{p}, p_0^E) = \frac{1}{2} \int \frac{d^dk^E}{(2\pi)^d} \frac{1}{(k^E)^2 + {m_1}^2} \times \frac{1}{(k^E + p^E)^2 + {m_2}^2}. \tag{3.22}$$

Analogous to the finite-temperature correlation function, Eq. (3.10) can be used to re-express

Eq. (1.31) as well:

$$\begin{aligned}\Pi_0(\boldsymbol{p}, p_0^E) &= \frac{1}{2}\int \frac{dk_0^E}{2\pi}\int \frac{d^D\boldsymbol{k}}{(2\pi)^D}\frac{1}{\boldsymbol{k}^2 + {m_1}^2 + (k_0^E)^2} \times \frac{1}{(\boldsymbol{k}+\boldsymbol{p})^2 + {m_2}^2 + (k_0^E + p_0^E)^2} \\ &= \frac{1}{2}\int \frac{dk_0^E}{2\pi}\, I\left(\boldsymbol{p}\,;\, {m_1}^2 + (k_0^E)^2, {m_2}^2 + (k_0^E + p_0^E)^2\right).\end{aligned} \tag{3.23}$$

The integral I in Eq. (3.23) can be computed in pySecDec using a 'reverse' Wick rotation as in the finite-temperature case previously discussed. The difficulty in subtraction is due to the temporal dimension integration dk_0. With the discrete temporal dimension summation in Eq. (3.11) and continuous temporal integral in Eq. (3.23), we need to fit one of them to the other's structure. Breaking down one integration region into numerous equally divided subintervals seems to be the most obvious approach. The idea is to divide the zero-temperature temporal momentum integral into numerous smaller-region momentum integrals:

$$\frac{1}{4\pi}\int dk_0^E\; I_0(k_0^E) \to \frac{1}{4\pi}\sum_{n=-A}^{A-1}\int_{2\pi n/\beta}^{2\pi(n+1)/\beta} dk_0^E\; I_0(k_0^E), \tag{3.24}$$

$$\text{where}\;\; I_0(k_0^E) = I\left(\boldsymbol{p}\,;\, {m_1}^2 + (k_0^E)^2, {m_2}^2 + (k_0^E + p_0^E)^2\right). \tag{3.25}$$

The integrations from $2\pi n/\beta$ to $2\pi(n+1)/\beta$ in k_0^E can be computed using Simpson's rule in Python's SimPy package.

The finite-temperature summation looks like

$$\frac{1}{2\beta}\sum_{n=-A}^{+A} I_{\mathcal{T}}(\omega_n), \tag{3.26}$$

$$\text{where}\;\; I_{\mathcal{T}}(\omega_n) = I\left(\boldsymbol{p}\,; {m_1}^2 + \omega_n^2, m_2^2 + (p_0^E + \omega_n)^2\right). \tag{3.27}$$

We can see that both Eqs. (3.24) and (3.26) have been broken down into pieces that correspond to different values/ranges of k_0. The next step is to match them up along with the integrals I for both correlators of zero temperature and finite temperature. Subtractions will be performed for each pair of $I_0(k_0^E)$ and $I_{\mathcal{T}}(\omega_n)$. To put them in the bigger picture with

Eqs. (3.11) and (3.23),

$$\begin{aligned}\Pi_s &= \Pi_{\mathcal{T}} - \Pi_0 \\ &= \frac{1}{2\beta}\sum_{n=-\infty}^{\infty} I_{\mathcal{T}}(\omega_n) - \frac{1}{2}\int \frac{dk_0^E}{2\pi}\, I_0(k_0^E) \\ &\approx \frac{1}{2}\left(\sum_{n=-A}^{A}\frac{I_{\mathcal{T}}(\omega_n)}{\beta} - \sum_{n=-A}^{A-1}\int_{2\pi n/\beta}^{2\pi(n+1)/\beta}\frac{dk_0^E}{2\pi}\, I_0(k_0^E)\right).\end{aligned} \tag{3.28}$$

Equation (3.28) should give us convergent numerical calculation results for the finite-temperature corrections of the one-loop self-energy topology, because the divergence in the series should cancel the divergence of the k_0 integration. We can perform a test calculation to see if the divergent thermal correlator and divergent zero-temperature correlator (both in $d = 4$ spacetime) now have a convergent difference. The results are shown in Fig. 3.2.

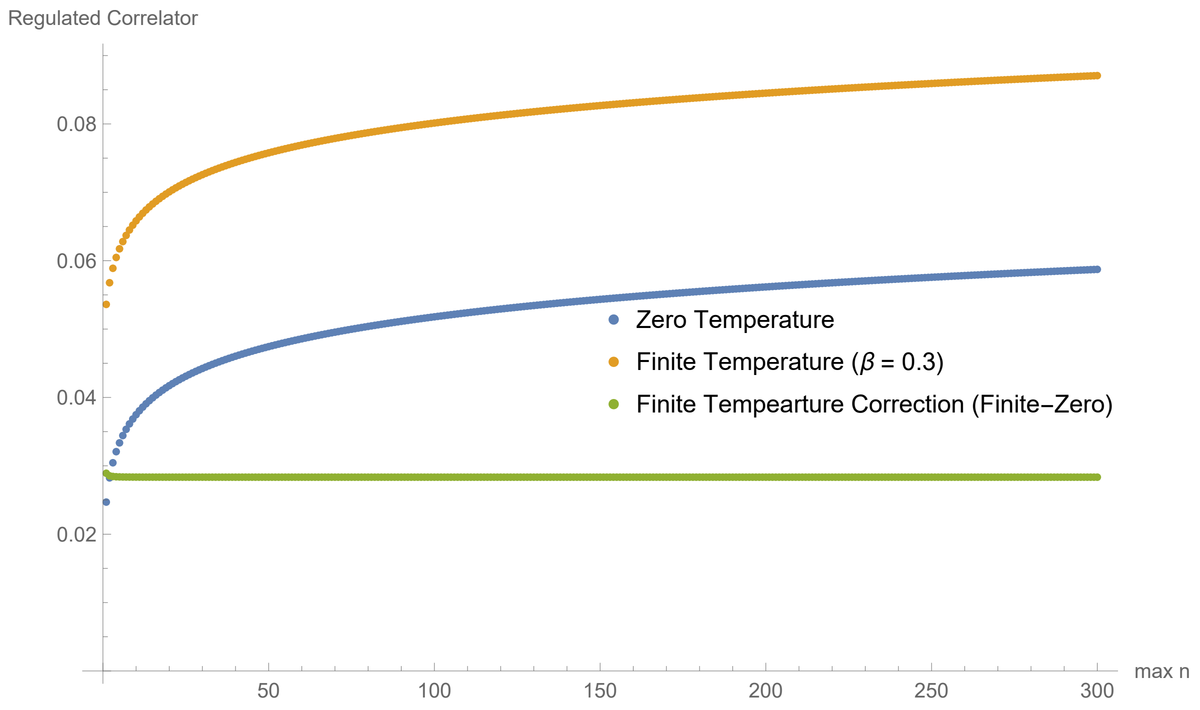


Figure 3.2: The numerical calculation results from pySecDec of zero-temperature correlation function Π_0 (blue), finite-temperature correlation function $\Pi_{\mathcal{T}}$ (yellow) and finite-temperature correction Π_s (green). The parameter values are $m_1 = 1.1, m_2 = 2, p_0^E = \frac{2\pi}{\beta}, \beta = 0.3, \boldsymbol{p}^2 = 1$ and maximum $|n|$ up to 300.

It is clear from Fig. 3.2 that the zero-temperature and finite-temperature correlators with expected divergences produced a convergent subtraction correction Π_s using the method of

Eq. (3.28). To look into this convergence from a mathematical perspective, define C_n

$$C_n = A_n - A_{n+1}, \tag{3.29}$$

where A_n represents individual terms from the summation in Eq. (3.28),

$$A_n = \frac{I_{\mathcal{T}}(\omega_n)}{\beta} - \int_{2\pi n/\beta}^{2\pi(n+1)/\beta} \frac{dk_0^E}{2\pi}\, I_0(k_0^E). \tag{3.30}$$

Since the value of C_n drops dramatically as n increases, we decided to use a log-log plot to find a relation between C_n and n of the form

$$C_n \approx \frac{a}{n^\gamma}, \tag{3.31}$$

where γ is defined positive for easier analysis which will be demonstrated later in this section and a is a constant. We will investigate the convergence properties of Eq. (3.28) by plotting $\ln(C_n)$ versus $\ln(n)$ using the same set of data as in Fig. 3.2. The results are shown in Fig. 3.3

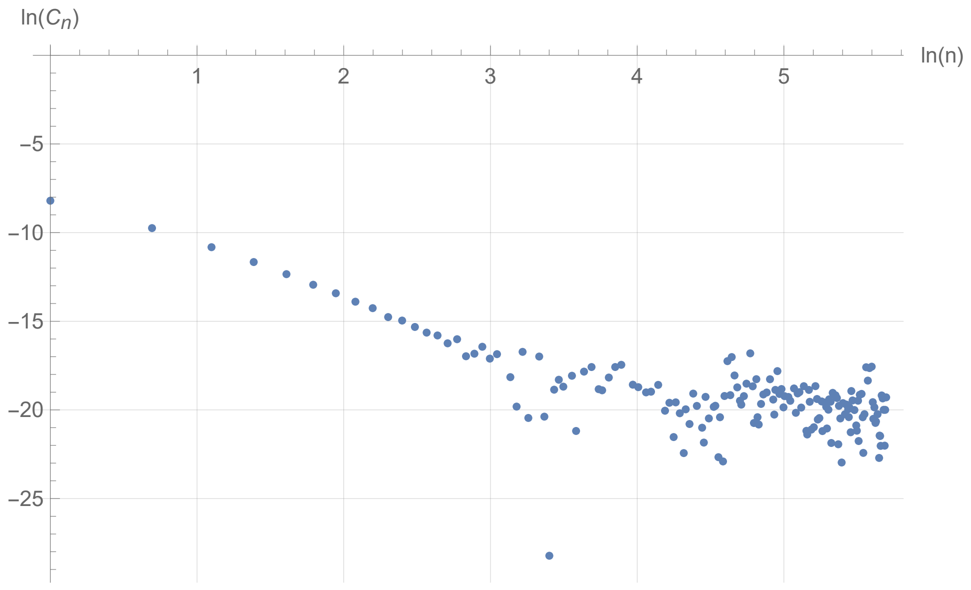


Figure 3.3: The plot between $\ln(C_n)$ and $\ln(n)$ shows a linear relation with a slope $-\gamma \approx -3.57$ corresponding to $C_n \approx \frac{a}{n^\gamma}$. The data in the figure was generated with the same parameters as in Fig. 3.2.

Figure 3.3 shows a clear linear relation before the numerical noise starts to dominate the results as n becomes larger. The relative numerical uncertainties from pySecDec are at the scale of 10^{-4} while most of the data in the C_n terms are at an even smaller scale, around 10^{-9}. From the assumed power law behavior Eq. (3.31),

$$C_n \approx \frac{a}{n^{\gamma}} \Rightarrow \ln(C_n) \approx -\gamma \ln(n) + \ln(a), \tag{3.32}$$

where $\ln(a)$ is defined as the intercept in Fig. 3.3. Thus we find

$$C_n = A_n - A_{n+1} \approx \frac{a}{n^{\gamma}}, \tag{3.33}$$

with $\gamma \approx 3.57 > 1$. Figure 3.3 and Eq. (3.31) demonstrate that the difference between successive terms in the summation in Eq. (3.28) is rapidly decreasing and approximately following the relation of Eq. (3.31) corresponding to a convergent series. The cut-off method developed here was used to successfully calculate the regulated finite-temperature correlation function for the one-loop self-energy topology. The Python code that applied our methodology, Eq. (3.28), to perform the test shown in Figs. 3.2 and 3.3 can be found in Appendix B, along with the `MATHEMATICA` notebook for the data analysis.

Additionally, from this test, we can see that the cut-off at $|n|_{\max} = 300$ is more than enough to get a numerically accurate result since the convergence of the finite-temperature correction term from Fig. 3.2 stabilized quickly after around $|n|_{\max} = 6$. We will choose $|n|_{\max} = 100$, which occurs before the numerical scatter in Fig. 3.3, as our cut-off threshold for later numerical calculations presented in Chapter 4.

4 Numerical Calculation of the Self-Energy Topology at Finite Temperature

Now that we have introduced thermal field theory and the cut-off methodology Eq. (3.28) to numerically calculate the correlation function in Eq. (1.33) of the self-energy topology as in Fig. 1.5, we can investigate the computations and analyze the results from the developed methods. We will start with lower dimensions where the correlation functions are convergent. That allows us to directly calculate them with pySecDec and benchmark the results using the developed methodology from Chapter 3 that uses Eq. (3.28) for the same cases (using the same parameters). Then we will move on to $d = 4$ spacetime in which both zero-temperature and finite-temperature correlators diverge and we need to retrieve the numerical results for the finite-temperature correction from our method.

4.1 Calculations in $d = 2$ and $d = 3$ Spacetime

At lower spacetime dimensions such as $d = 2$ and $d = 3$ respectively, we have the correlation functions in the form of

$$\Pi_{\mathcal{T}}(p_1, p_0) = \frac{1}{2\beta} \sum_{n=-\infty}^{\infty} \int \frac{dk_1}{2\pi} \frac{1}{\frac{4n^2\pi^2}{\beta^2} + k_1^2 + {m_1}^2} \times \frac{1}{(\frac{2n\pi}{\beta} + p_0)^2 + (k_1 + p_1)^2 + {m_2}^2} \tag{4.1}$$

$$\Pi_{\mathcal{T}}(\boldsymbol{p}, p_0) = \frac{1}{2\beta} \sum_{n=-\infty}^{\infty} \int \frac{d^2\boldsymbol{k}}{(2\pi)^2} \frac{1}{\frac{4n^2\pi^2}{\beta^2} + \boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{(\frac{2n\pi}{\beta} + p_0)^2 + (\boldsymbol{k} + \boldsymbol{p})^2 + {m_2}^2}. \tag{4.2}$$

These cases can be computed directly using pySecDec as both of them converge after the cut-off regularization on the temporal dimension is taken to $A \to \infty$ in the summation. Also, the zero-temperature correlator can be computed from Eq. (1.31) but with lower dimensional integrals. Since we have both Π_0 and $\Pi_{\mathcal{T}}$ from pySecDec computations, the subtraction can be handled easily. In the benchmark analysis presented below, two main aspects will be studied. The first compares the direct evaluation of Π_0 using pySecDec with the cutoff at $2\pi A/\beta$. The second aspect explored is the numerical convergence of the truncated series of $\Pi_\mathcal{T}$, again parameterized by A. We will now examine Eqs. (4.1) and (4.2) as the finite-temperature piece of the finite-temperature correction Π_s.

4.1.1 Benchmarking in the Lower Dimensions

Starting with a lower dimension, we will now focus on the $d = 2 = 1+1$ $(D = 1)$ case where the finite-temperature correction is

$$\begin{aligned}\Pi_s(p_1, p_0) &= \Pi_\mathcal{T} - \Pi_0 \\ &= \frac{1}{2\beta}\sum_{n=-A}^{A}\int \frac{dk_1}{2\pi}\frac{1}{\frac{4n^2\pi^2}{\beta^2} + k_1^2 + {m_1}^2} \times \frac{1}{(\frac{2n\pi}{\beta} + p_0)^2 + (k_1 + p_1)^2 + {m_2}^2} \\ &\qquad - \frac{1}{2}\int \frac{d^2k}{(2\pi)^2}\frac{1}{k^2 + {m_1}^2} \times \frac{1}{(k+p)^2 + {m_2}^2},\end{aligned} \tag{4.3}$$

where $p_\mu = (p_0, p_1)$. The benchmark calculation results for our methodology in the $d = 2$ case are shown in Fig. 4.1.

Figure 4.1 shows that the two methods for evaluating Π_0 (the cut-off and direct evaluation) are in close agreement for $A > 6$ (the data points overlap almost completely). Furthermore, the finite-temperature correction stabilizes for $A > 6$, indicating rapid numerical convergence of the truncated series for $\Pi_\mathcal{T}$. The differences between Π_s given by the two methods are at the scale of 10^{-7} at larger maximum n. Meanwhile, the numerical uncertainty from the pySecDec calculations gave an error scale at 10^{-4} and therefore confirm the reliability of the developed methodology Eq. (3.28) in $d = 2$ spacetime. Similarly, we will now perform an analogous calculation in $d = 3$ spacetime (one temporal dimension and two spatial dimensions, i.e., $d = 1+2$ $(D = 2)$).

In Fig. 4.2, we have good agreement between the two methods in $d = 3$ as well. Similar to the $d = 2$ case above, the difference between the two sets of data is at the scale of 10^{-4} with numerical uncertainty at 10^{-3}, which allows us to consider the difference between the two methods to be negligible. Thus, from the convergent zero- and finite-temperature

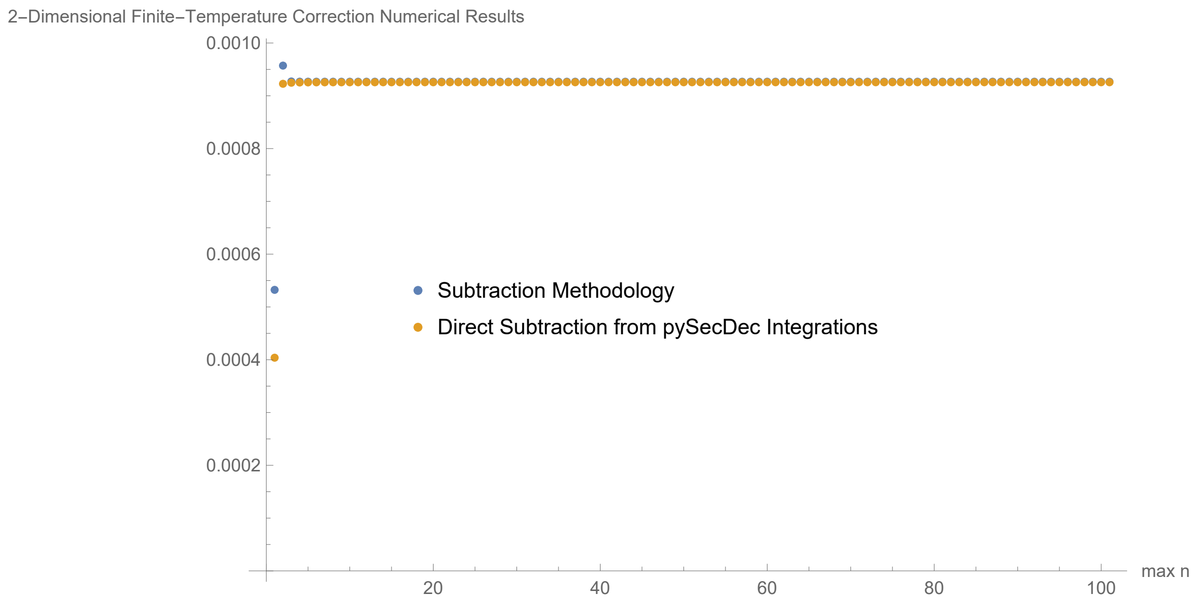


Figure 4.1: Comparing the numerical calculation results from the pySecDec-implemented methodology Eq. (3.28) and direct subtractions from complete pySecDec results of both zero and finite-temperature correlation functions at $d = 2$ spacetime (with the parameter values of $m_1 = 1.1, m_2 = 1.2, p_0^E = 2\pi/\beta, \beta = 0.1, p_1 = 1.3$ and maximum $|n|$, i.e., the parameter A up to 100. The Python code including pySecDec files that were coded to perform the comparison calculations in this figure is included in Appendix C, along with the `MATHEMATICA` notebook for the plot analysis.

correlation functions at lower dimensions, we have successfully benchmarked the methodology that breaks down the temporal dimension (see Eq. (3.28)) that is necessary for the higher-dimensional divergent calculations in $d = 4$ spacetime.

4.1.2 Relationship with Respect to External Momenta

Before diving into applying the methodology in higher dimensions, we examine some interesting relationships between finite-temperature corrections and input parameters at lower dimensions since the calculations are easier without divergences. One of the physical quantities that can be altered in the calculation is the external momentum $p_\mu = (p_0, \boldsymbol{p})$. Since in our methodology, we can vary the p_0 and $\boldsymbol{p}$ variables separately, this gives us a chance to look into their effects on Π_s.

The variable p contains two components: the temporal component p_0^E and the spatial component $\boldsymbol{p}$. We can account for both in one plot and look for trends or relations with the

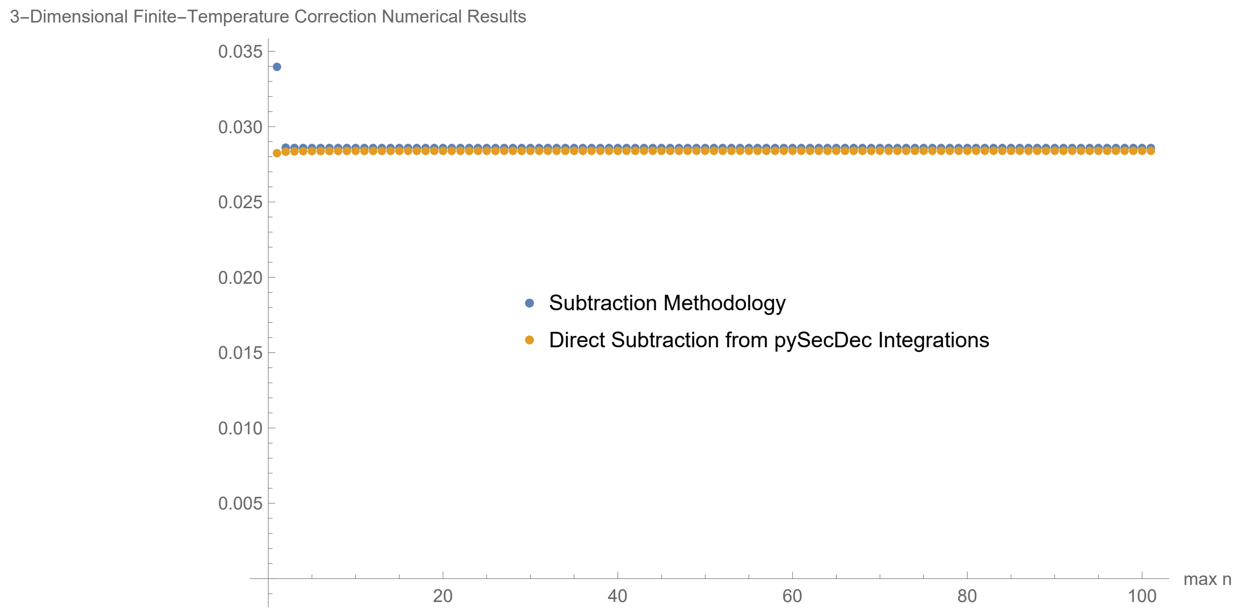


Figure 4.2: Comparing the numerical calculation results from pySecDec-implemented methodology and direct subtractions from complete pySecDec results of both zero and finite-temperature correlation functions at $d = 3$ spacetime (with the parameter values of $m_1 = 1.1, m_2 = 1.2, p_0^E = 6, \beta = 0.1, \boldsymbol{p} = (1.3, 1.4)$ and maximum $|n|$ up to 100 (i.e., the parameter A)).

finite-temperature correction function Π_s in Eq. (3.1) for the one-loop self-energy topology. The temperature in the calculation will be held at $\beta = 0.3$.

Figure 4.3 shows that the finite-temperature correction Π_s peaks at $p = 0$, and decreases when either p_0^E or $\boldsymbol{p}$ increases. That means that the thermal effects from the temperature are largest when the external momentum of the one-loop self-energy topology goes to zero. A similar pattern can be found in Π_s at $d = 3$ as well (see Fig. 4.4).

Looking at Eq. (4.1) and Eq. (4.2), we can understand the numerical pattern that is shown in Figs. 4.3 and 4.4. The spatial momentum components are always squared in the calculation; therefore, the results are symmetric with respect to the variable $\boldsymbol{p}$. The sign of the temporal component of external momentum p_0^E contributes due to the expression $(\frac{2n\pi}{\beta} + p_0)^2$ in the denominator of $\Pi_{\mathcal{T}}$. However, we have noted in Section 3.4 that the cutoff at $|n|_{\max} = A = 100$ should be large enough for our study. Therefore, the asymmetry from this expression is relatively small.

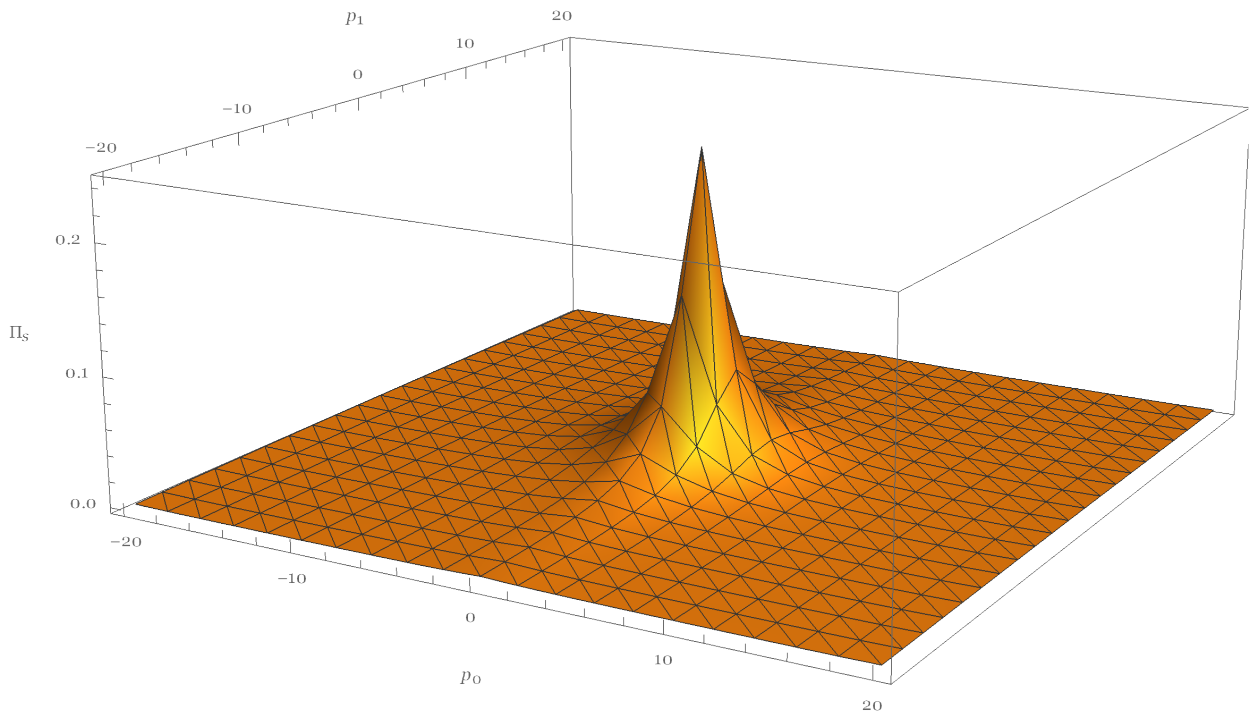


Figure 4.3: The effects of the external momentum on the $d = 2$ finite-temperature corrections Eq. (3.1) of the one-loop self-energy topology (with the parameter values $m_1 = 1.1, m_2 = 1.2, \beta = 0.3, p_0^E$ and $\boldsymbol{p}$ varying in the range $[-20, 20]$ and maximum $|n|$ at 100 (i.e., the parameter A)). The python code including pySecDec program that was coded to perform the comparison calculations in this figure is included in Appendix D, along with the `MATHEMATICA` notebook for the plot analysis.

4.1.3 Relationship with Respect to Temperature

Studying the temperature effects in thermal field theory is one of the key objects of this thesis. In this subsection, we will explore temperature effects in lower dimensions to see if there is a mathematical relationship between temperature $\mathcal{T}$ and the temperature correction correlator Π_s for the self-energy integral. With constant values for the external momentum, the thermal parameter β has been varied over a range of values. The $d = 2$ plot in Fig. 4.5 (left) of Π_s as a function of β shows an interesting trend indicating that there may be a reciprocal function relation. Therefore, an inverse plot Fig. 4.5 (right) was made to look into the possibility that $\Pi_s \sim \frac{1}{\beta}$.

The physical quantity $1/\beta$ has a clear linear relation with Π_s for $\frac{1}{\beta} \gg 1$. Recall that, by definition, the inverse of β is the temperature $\mathcal{T}$, so one immediate result is that when

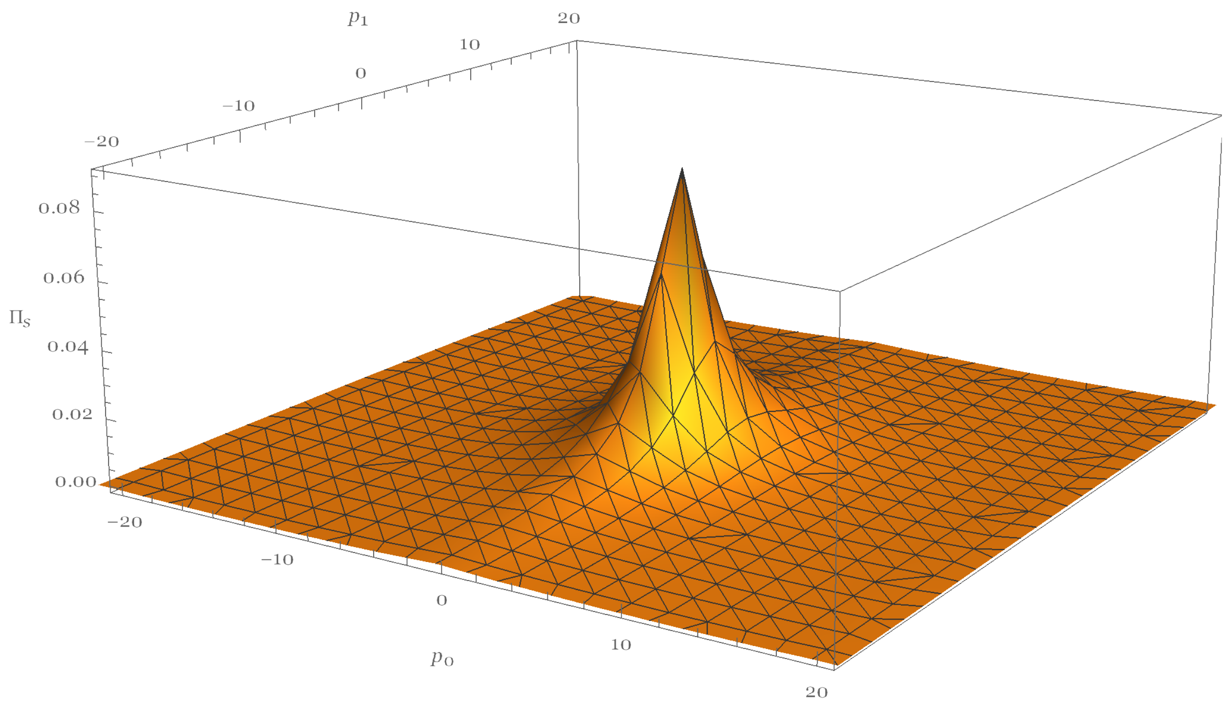


Figure 4.4: The effects of the external momentum on the $d = 3$ finite-temperature correction Π_s (Eq. (3.1)) of the one-loop self-energy topology (with same parameter values as in Fig. 4.3).

temperature goes to zero, the finite-temperature correction goes to zero as well. The small $\mathcal{T}$ behavior is examined in more detail in Section 4.1.4. The difference between $\Pi_\mathcal{T}$ and Π_0 disappears as expected. To analyze the linear trend indicated in Fig. 4.5 (right), we will investigate the behavior at large $\frac{n}{\beta}$ (i.e., large n behavior similar to Section 3.2 and large $\mathcal{T}$). Similar to Eq. (3.5), in $d = 2$ dimension we have the approximate relation

$$\begin{aligned} \Pi_\mathcal{T} &\approx \frac{1}{2\beta} \sum_n \frac{1}{(4\pi)^{\frac{1}{2}}} \frac{\Gamma\left(\frac{3}{2}\right)}{\left(\frac{4n^2\pi^2}{\beta^2}\right)^{3/2}} \\ &\approx \frac{\beta^2}{2} \sum_n \frac{1}{32\pi^3 |n|^3}; \qquad (n \neq 0). \end{aligned} \tag{4.4}$$

From Eq. (4.4), the $n \neq 0$ terms give the relation of $\Pi_\mathcal{T} \sim \beta^2$, or $\Pi_\mathcal{T} \sim \frac{1}{\mathcal{T}^2}$. For $n = 0$, we can easily see that $\Pi_\mathcal{T} \sim \frac{1}{\beta}$ from Eq. (4.1), or $\Pi_\mathcal{T} \sim \mathcal{T}$ $(n = 0)$. Then,

$$\Pi_s = \Pi_\mathcal{T} - \Pi_0 = \Pi_\mathcal{T}|_{n\neq 0} + \Pi_\mathcal{T}|_{n=0} - \Pi_0. \tag{4.5}$$

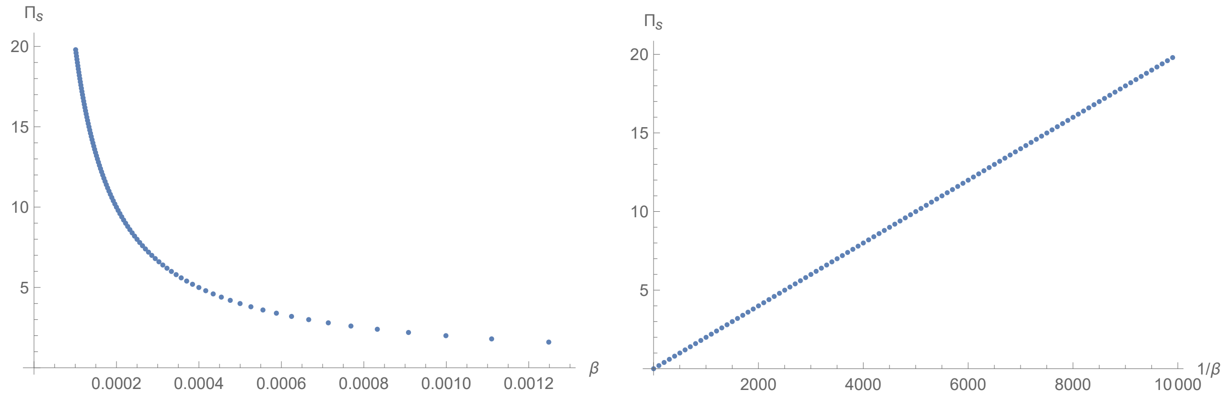


Figure 4.5: The finite-temperature correction Π_s plotted with respect to β (left) and $1/\beta$ (right) respectively for $d = 2$ spacetime. The slope on the right plot is approximately 0.00199939. The parameters used in the calculation are $m_1 = 1.1, m_2 = 1.2, p = (7, 8, 0, 0)$. The python code including pySecDec program that was coded to perform the comparison calculations in Fig. 4.5 is included in Appendix E, along with the `MATHEMATICA` notebook for the plot analysis.

Since Π_0 does not have any $\mathcal{T}$-dependence, it will act as a constant, so that $\Pi_s \sim \Pi_{\mathcal{T}}$ at large temperature ($\mathcal{T} \gg 1$). $\mathcal{T}$ dominates $\frac{1}{\mathcal{T}^2}$. Therefore,

$$\Pi_S \sim \mathcal{T} \text{ at large } \mathcal{T} \tag{4.6}$$

which agrees with the general linear shape in the numerical plot in Fig. 4.5 (right). The slope itself can be calculated using the first term of Eq. (4.3) for the $n = 0$ term. We find

$$\begin{aligned} \text{slope}_{2d} &= \frac{1}{2} \int \frac{dk_1}{2\pi} \frac{1}{k_1^2 + {m_1}^2} \times \frac{1}{p_0^2 + (k_1 + p_1)^2 + {m_2}^2} \\ &\approx 0.00199942 \text{ (calculated by } \texttt{MATHEMATICA}\text{)}. \end{aligned} \tag{4.7}$$

The slope calculated in Eq. (4.7) has good agreement with the numerical result from Fig. 4.5 (right) at 0.00199783.

For $d = 3$ spacetime, we have analogous results. At large n in Eq. (4.2),

$$\begin{aligned} \Pi_{\mathcal{T}} &\approx \frac{1}{2\beta} \sum_n \frac{1}{4\pi} \frac{\Gamma(1)}{\Gamma(2)} \frac{1}{\left(\frac{2n\pi}{\beta}\right)^2} \\ &\approx \frac{\beta}{32\pi^2} \sum_n \frac{1}{|n|^2}; \quad (n \neq 0). \end{aligned} \tag{4.8}$$

Combining Eqs. (4.8), (4.2), and Π_0 is a constant, we find for $d = 3$

$$\Pi_s \sim \frac{1}{\mathcal{T}} \quad (n \neq 0) \tag{4.9}$$

$$\Pi_s \sim \mathcal{T} \quad (n = 0). \tag{4.10}$$

Equation (4.10) will be dominant at large $\mathcal{T}$ and give the numerical result shown in Fig. 4.6 (right). Referring again to the first term of Eq. (4.2), the slope in $d = 3$ spacetime is

$$\begin{aligned}\text{slope}_{3d} &= \frac{1}{2}\int \frac{d^2\boldsymbol{k}}{(2\pi)^2}\frac{1}{\boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{p_0^2 + (\boldsymbol{k}+\boldsymbol{p})^2 + {m_2}^2} \\ &\approx 0.00130947 \text{ (calculated by pySecDec)},\end{aligned} \tag{4.11}$$

which agrees with that of the plot shown in Fig. 4.6 (right).

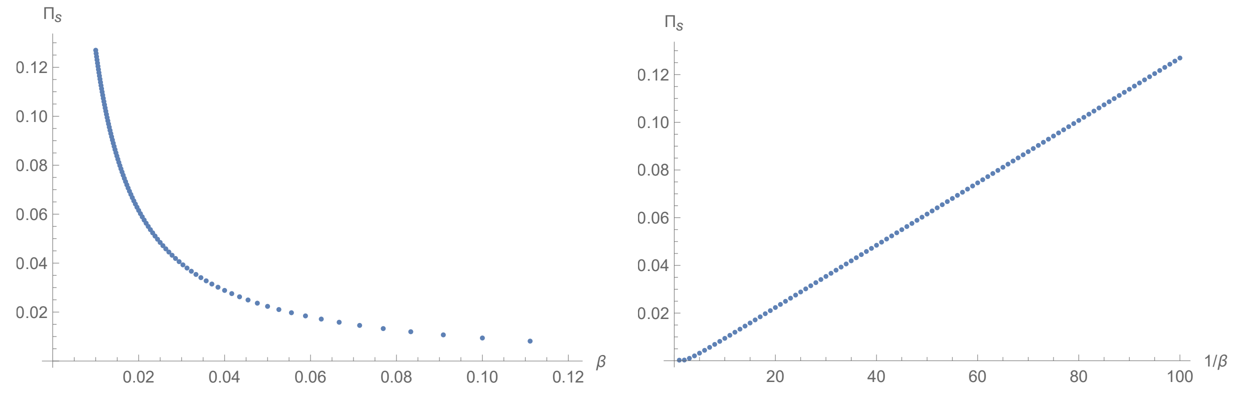


Figure 4.6: The plot of the finite-temperature correction Π_s as a function of β (left) and $1/\beta$ (right) at $d = 3$ spacetime. The slope on the right plot is approximately 0.00130617. The parameters used in the calculation are $m_1 = 1.1, m_2 = 1.2, p_\mu = (7, 8, 9, 0)$.

4.1.4 Small Temperature Behavior in 2-dimensional Spacetime

The primary goal of this research is to develop a proof of concept for a numerical method to calculate Feynman integrals in thermal field theory. We have been choosing largely arbitrary values for the external momentum in spacetimes with different numbers of spatial dimensions. However, due to the nature of the imaginary time formalism (Matsubara formalism), p_0^E is assumed to be discrete at finite temperature,

$$p_0^E = \frac{2\pi l}{\beta},\ l = 0, \pm 1, \pm 2, \ldots. \tag{4.12}$$

Whether we set the external momentum p_0^E or β as constant and alter the other, we can change either β or l to treat p_0^E and β separately. So far, from Figs. 4.1 and 4.2 as well as the slope confirmation in Section 4.1.3, we have great agreement in our data indicating that the numerical methodology proposed in Chapter 3 works for arbitrary external four-momentum.

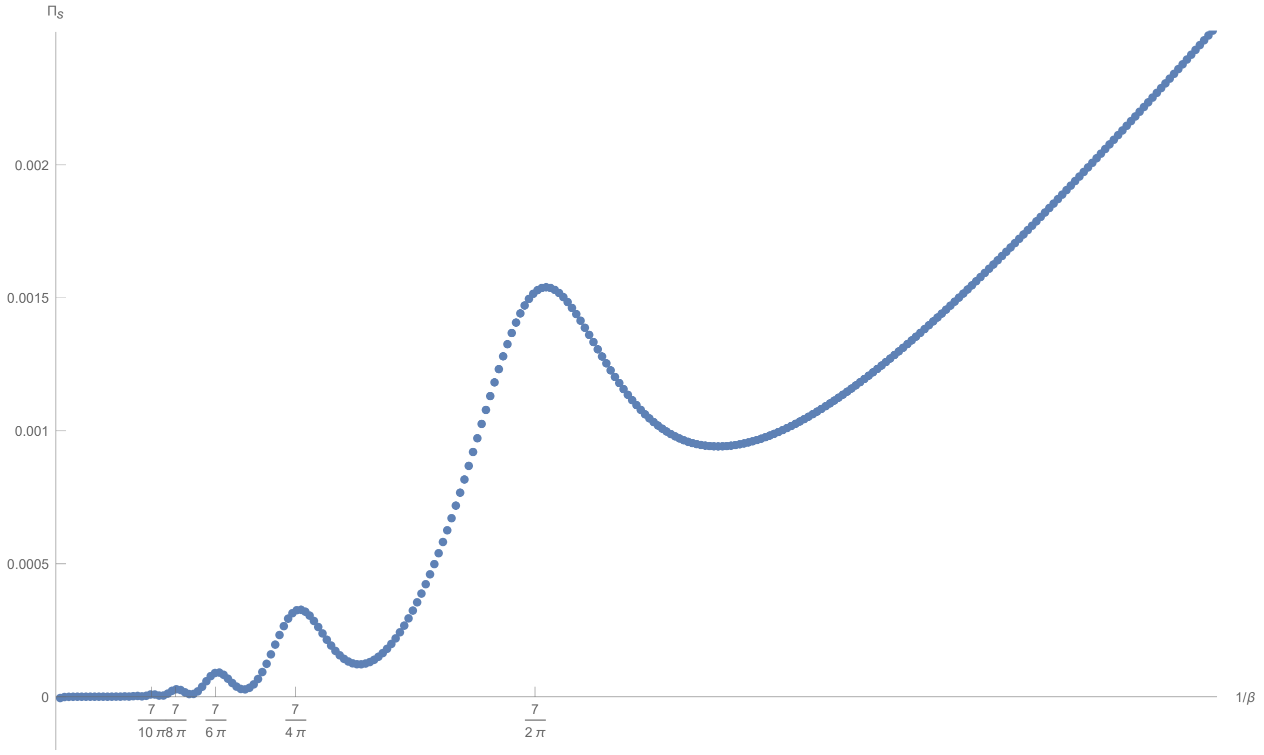


Figure 4.7: A plot of the finite-temperature correction Π_s as a function of $1/\beta$ at fixed p_0^E in $d = 2$ spacetime. The parameters used in the calculation are $m_1 = 1.1, m_2 = 1.2, p_0^E = 7, p_1 = 8, |n|_{\max} = A = 1000$.

While working with the numerical calculation presented in Fig. 4.5, a certain oscillatory behavior was found at small temperature ($\mathcal{T}$) as shown in Fig. 4.7, which exhibits a pattern of peaks. Since

$$p_0^E = 7 = \frac{2\pi l}{\beta}, \tag{4.13}$$

the temperature parameter would actually be restricted to the discrete values of

$$\frac{1}{\beta} = \mathcal{T} = \frac{7}{2\pi l}, \tag{4.14}$$

where l is integer. The maxima are showing up at exactly those discrete $\mathcal{T}$ values (where $l \in \{1, 2, 3, 4, 5\}$) even though the condition of Eq. (4.12) was not purposely imposed during the computation. The denominator factor from Eq. (4.1)

$$\frac{1}{(\frac{2n\pi}{\beta} + p_0)^2 + (k_1 + p_1)^2 + {m_2}^2} \tag{4.15}$$

gives the origin of this intriguing resonance effect in $\Pi_{\mathcal{T}}$. In order to look into this phenomenon from a different perspective, we set $\beta = \frac{2\pi}{7}$ for the next trial calculation shown in Fig. 4.8. Therefore, temperature $\mathcal{T} = \frac{1}{\beta} = \frac{7}{2\pi}$ and

$$p_0^E = \frac{2\pi l}{\beta} = 7l. \tag{4.16}$$

This should give us resonant peaks at multiples of 7 in temporal momentum values in the plot of Π_s as a function of $\frac{1}{\beta}$. To view the resonant phenomenon, p_0^E will range through intermediate values in between each $7l$.

From Fig. 4.8, we can see that the resonance peaks shows up at multiples of 7 as we predicted. We can also see that, if we choose the p_0^E values strictly following the restriction in Eq. (4.16), the relation should present in a smooth line (orange line in Fig. 4.8) as Π_s is a deceasing function of p_0^E. This is a reflection of the discretization in temporal dimension from the Matsubara formalism.

4.2 Calculations in $d = 4$ Spacetime

Mostly in physics, when talking about spacetime, we are referring to a model with three dimensions of space and one dimension of time. The frequently used concept of four-momentum also includes these dimensions. The difficulty in working in $d = 4$ spacetime when calculating correlation functions for the one-loop self-energy topology is that both zero- and finite-

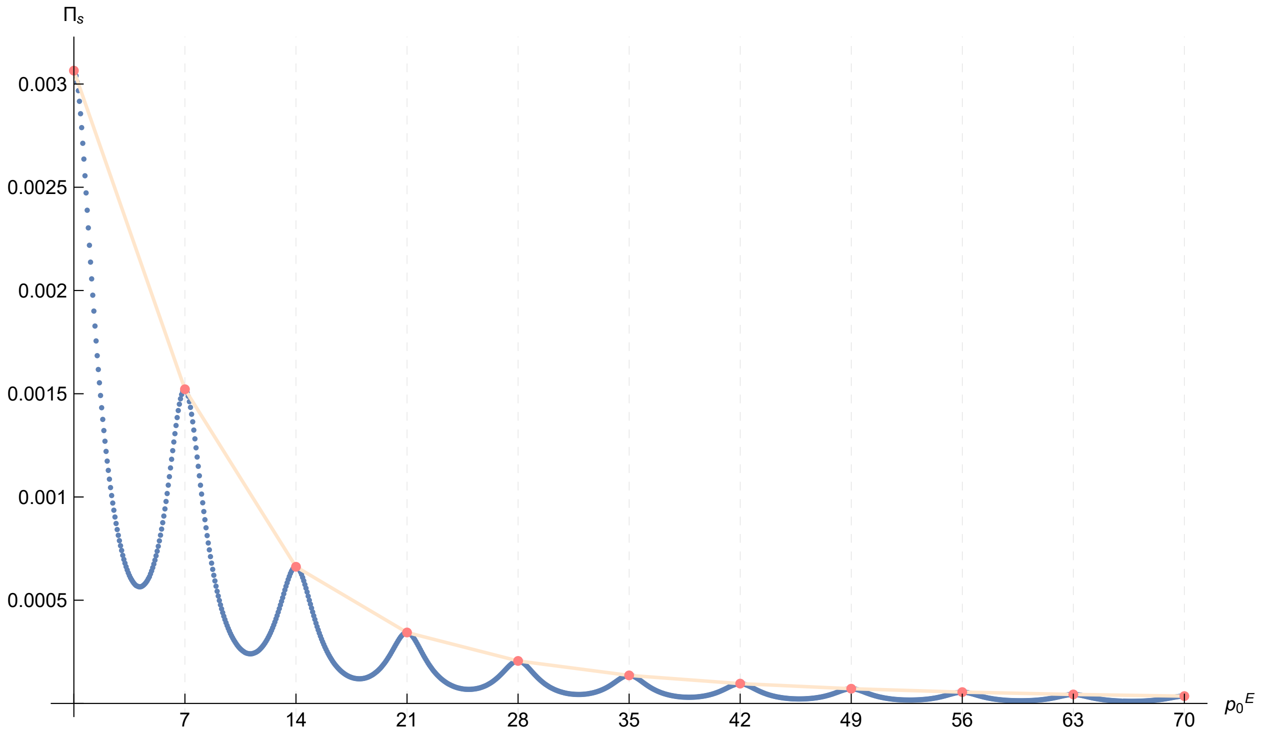


Figure 4.8: The plot of the finite-temperature correction Π_s with respect to $1/\beta$ in $d = 2$ spacetime. The parameters used in the calculation are $m_1 = 1.1, m_2 = 1.2, \beta = \frac{2\pi}{7}, p_0^E = 7l, p_1 = 8, |n|_{\text{max}} = A = 100$ and l are integers. The discrete restriction on p_0^E (red dots) gives a relation with Π_s (orange line) without the oscillation-like behavior.

temperature correlators diverge. Now, with the Chapter 3 development and Section 4.1 benchmarking of the methodology which allows us to work with the divergent numerical calculations with the chosen regulators (cut-off) and programming methodology (reverse Wick rotation), we can finally show some computation results in the $d = 4$ spacetime. Consider the finite-temperature correction for the self-energy topology:

$$\Pi_s \approx \frac{1}{2}\left(\sum_{n=-A}^{A} \frac{I_{\mathcal{T}}(\omega_n)}{\beta} - \sum_{n=-A}^{A-1} \int_{2\pi n/\beta}^{2\pi(n+1)/\beta} \frac{dk_0^E}{2\pi}\, I_0(k_0^E)\right), \tag{4.17}$$

$$\text{where} \quad I_{\mathcal{T}}(\omega_n) = \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{\omega_n^2 + \boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{(\omega_n + p_0^E)^2 + (\boldsymbol{k} + \boldsymbol{p})^2 + {m_2}^2}, \tag{4.18}$$

$$\text{and} \quad I_0(k_0^E) = \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{\boldsymbol{k}^2 + {m_1}^2 + (k_0^E)^2} \times \frac{1}{(\boldsymbol{k} + \boldsymbol{p})^2 + {m_2}^2 + (k_0^E + p_0^E)^2}. \tag{4.19}$$

Just like in the lower spacetime dimensions $d = 2$ and $d = 3$, the external momenta and temperature parameter are the variables we are interested in. Using the constant value $\beta = 0.3$, Fig. 4.9 shows the plot of Π_s as a function of p in $d = 4$ spacetime.

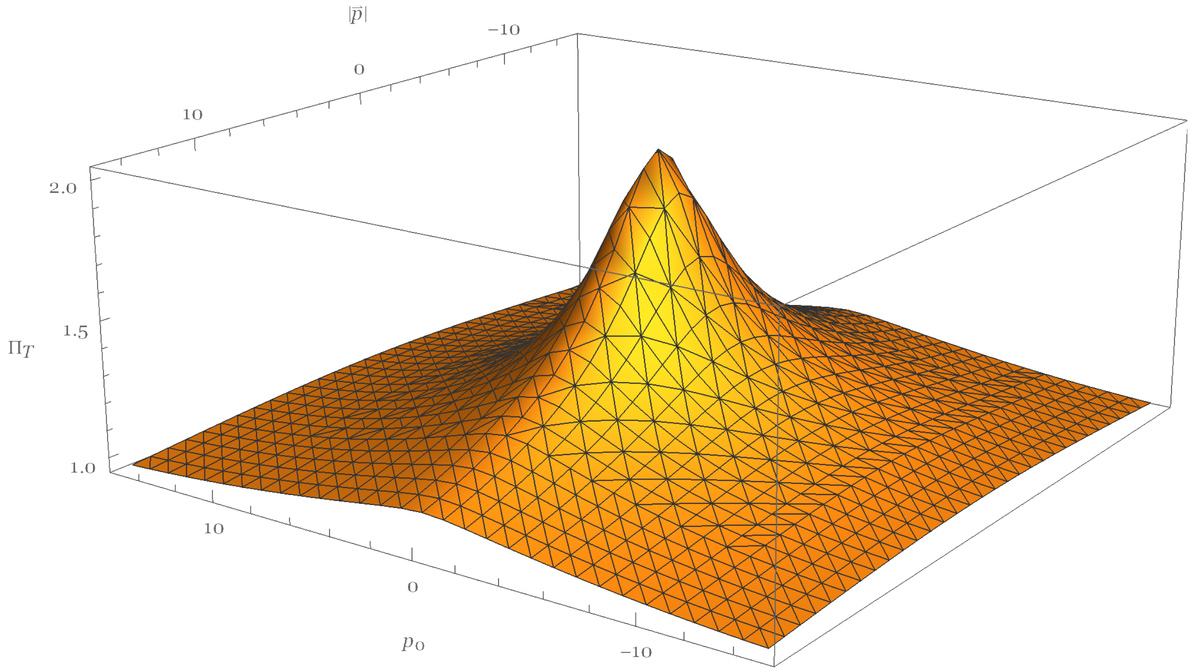


Figure 4.9: The effects of the external momentum p_0^E and $|\boldsymbol{p}|$ on the $d = 4$ finite-temperature corrections Eq. (3.1) of the one-loop self-energy topology (with the parameter values of $m_1 = 1.5$, $m_2 = 1.2$, $\beta = 0.3$, p_0^E and where $\pm|\boldsymbol{p}|$ varies in the range $[-10, 10]$ and maximum $|n|$ up to 1000 (i.e., the parameter A)).

As expected, the external momentum gives the maximum peak at $p = 0$, similar to the lower dimensional results in Figs. 4.3 and 4.4. This is a very good sign, indicating that the methodology was successfully applied to the $d = 4$ spacetime case of the one-loop self-energy topology. The complete numerical method provides convergent Π_s correlator results and shows the expected pattern with respect to the external momentum as shown in Fig. 4.9.

Next, we will hold the external momentum and field masses constant so we can look into the relation between the the temperature β and finite-temperature correction Π_s on the correlation function.

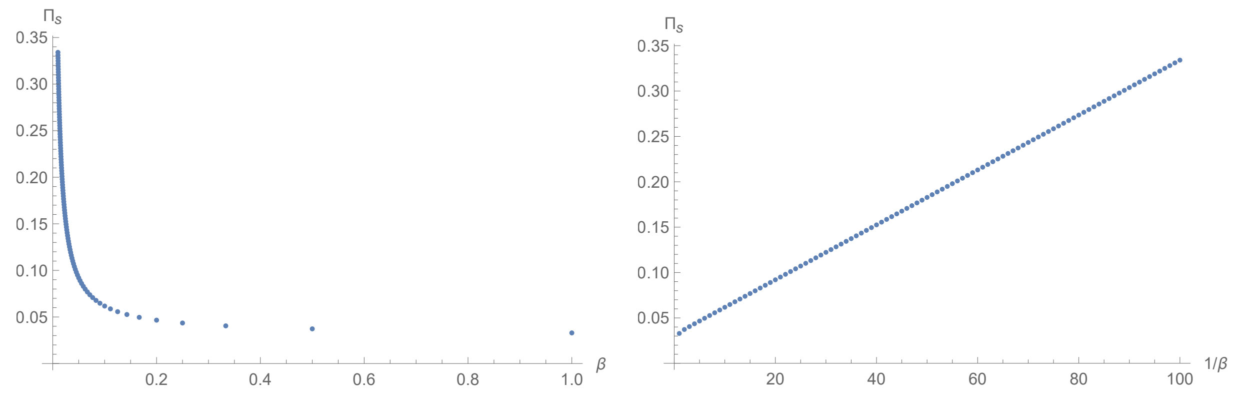


Figure 4.10: Above we have plotted the finite-temperature correction Π_s with respect to β (left) and $1/\beta$ (right) respectively for $d = 4$ spacetime. The slope in the right plot is approximately 0.00302586. The intercept of the right plot is approximately 0.0314978. The parameters used are $m_1 = 1.1$, $m_2 = 1.2$, $p = (7, 8, 9, 6)$. Truncation at maximum $|n|$ is up to 100 (i.e., the parameter $A = 100$)

Referring to Chapter 3, Eq. (3.5) gives an approximation of the large $\mathcal{T}$ behavior of $\Pi_{\mathcal{T}}$,

$$\begin{aligned}\Pi_{\mathcal{T}} &\approx \frac{1}{2\beta}\sum_n \frac{1}{(4\pi)^{\frac{3}{2}}}\frac{\Gamma\left(\frac{1}{2}\right)}{1}\frac{1}{\left(\frac{2n\pi}{\beta}\right)^{2\times\frac{1}{2}}} \\ &= \frac{1}{32\pi^2}\sum_n \frac{1}{|n|} \quad (n \neq 0).\end{aligned} \tag{4.20}$$

Equation (4.20) suggests that $\Pi_{\mathcal{T}}$ $(n \neq 0)$ is independent of $\mathcal{T}$. For the $n = 0$ term, similar to the $d = 2$ and $d = 3$ cases in Section 4.1.3, we have the relation $\Pi_{\mathcal{T}} \sim \mathcal{T}$ as well as the linear slope between $\Pi_{\mathcal{T}}$ and $\mathcal{T}$ from Eq. (1.33). Because Π_0 is considered to be constant since the cutoff brought by $2\pi A/\beta$ should be negligible, we again have a linear relation and slope between Π_s and $\mathcal{T}$ of

$$\begin{aligned}\text{slope}_{4d} &= \frac{1}{2}\int \frac{d^3\boldsymbol{k}}{(2\pi)^3}\frac{1}{\boldsymbol{k}^2 + {m_1}^2} \times \frac{1}{p_0^2 + (\boldsymbol{k}+\boldsymbol{p})^2 + {m_2}^2} \\ &\approx 0.00302636 \text{ (calculated by pySecDec).}\end{aligned} \tag{4.21}$$

Figure 4.10 shows excellent agreement in the linear relation prediction and the slope value of Eq. (4.21). We can also see that the $n \neq 0$ terms give the Π_s-axes intercept a shift upwards to the value 0.0314723 (the intercept was calculated from Fig. 4.10 (right) by `MATHEMATICA`).

From Eq. (4.20), we have calculated the intercept from the $n \neq 0$ terms to be

$$\sum_{n=-100}^{100} \frac{1}{32\pi^2|n|} = 0.0328495. \tag{4.22}$$

The intercept from the linear fit of Fig. 4.10 (right) also agrees with prediction from Eq. (4.20) to the second decimal. A small difference between the plot intercept and calculated intercept is expected because the actual intercept would also rely on the value of Π_0 which may have a small cutoff effect. Considering the linear relation only applies at larger temperature, as $1/\beta$ becomes smaller, Π_s deviates away from the linear trend and approaches zero as $1/\beta$ goes to zero (see Fig 4.11).

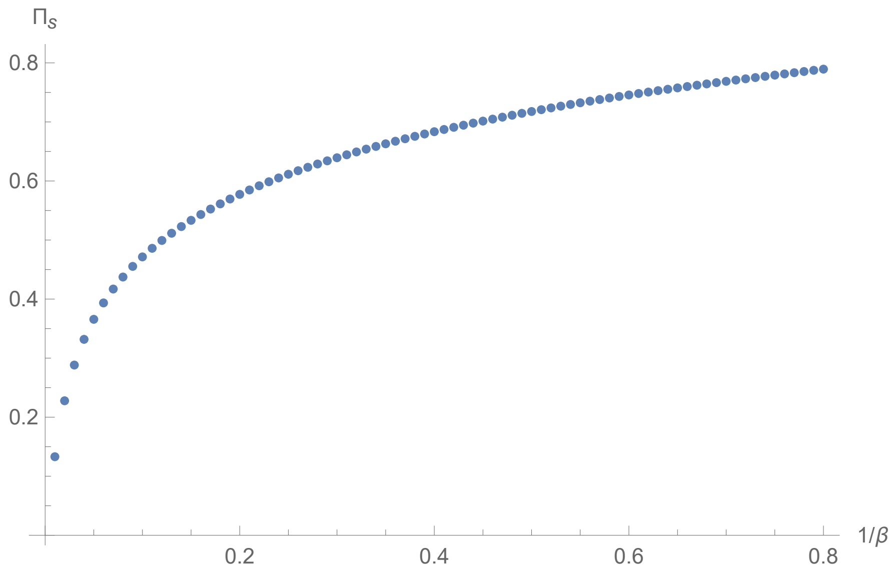


Figure 4.11: The plot shows the Π_s behavior at small temperature. The finite-temperature correction Π_s goes to zero as temperature goes to zero. The parameters used are the same as in Fig. 4.10.

5 Conclusion

There now exist powerful computational tools including pySecDec [6] to perform numerical calculations for QFT Feynman diagram loop integrals. We were curious to see if we could apply these computational tools to finite-temperature QFT. By the end of this thesis, we have conquered the first step towards this ambitious goal.

By utilizing the imaginary time formalism (Matsubara formalism) [9, 1, 10], we have reached an expression (Eq. (1.33)) for the one-loop self-energy topology (Fig. 2.1). We chose the Matsubara formalism as the approach because its representation of the correlation function Eq. (1.33) is in a form in which we can employ the powerful computation functionality of pySecDec. One key technique we developed is called the 'reverse' Wick rotation (see Section 3.3) in order to apply pySecDec to our expression of finite-temperature correlation function. We 'reverse' Euclideanized the input momentum so that pySecDec would only integrate on the spatial dimensions of the Feynman integrals. To benchmark, as well as to study the use of pySecDec on the chosen topology (Fig. 2.1), several calculations were performed for zero-temperature correlation functions for the designated topology (Fig. 2.1). In Chapter 2, we used pySecDec to numerically compute the QFT Feynman integrals of TBI topology (Eq. (1.31)) under certain masses' values. Meanwhile, we used `MATHEMATICA` to compute the same integrations by directly coding them in. The results from both programs match perfectly and allow us to proceed with pySecDec as a reliable tool for numerical calculation. After confirming pySecDec is suitable for the form of zero-temperature correlation function Eq. (1.31), we applied the program to the finite-temperature correlation function Eq. (1.33) in thermal field theory.

A major challenge first came in the form of divergences. Even though pySecDec has dimensional regularization embedded in the program, the discretized temporal summation over n in $\Pi_{\mathcal{T}}$ revealed a new divergence. As the summation index n goes from $-\infty$ to $+\infty$, the correlator $\Pi_{\mathcal{T}}$ diverges and requires a regulator. The method chosen for $\Pi_{\mathcal{T}}$ is the cut-off method. We found the appropriate cut-off ($A = 100$) for the chosen parameters by observing the convergence behavior for different maximum cut-offs in Fig. 3.2.

The other major challenge we encountered was also related to the divergence of $\Pi_{\mathcal{T}}$. If $\Pi_{\mathcal{T}}$ diverges and the zero-temperature correlator Π_0 (which can be regulated by pySecDec directly) also diverges, then the physical variable $\Pi_s = \Pi_{\mathcal{T}} - \Pi_0$ converges. Our solution is to break down the temporal integration over k_0 in Π_0 so we can find corresponding terms between $\Pi_{\mathcal{T}}$ and Π_0 as shown in Eq. (3.28). Equation (3.28) also applies a cutoff at $2\pi A/\beta$ on the zero-temperature correlator Π_0. Since in lower dimension spacetimes such as $d = 2$ and $d = 3$, both finite-temperature correlator $\Pi_{\mathcal{T}}$ and zero-temperature correlator Π_0 are convergent, we were able to test the Cut-off methodology in these spacetimes. Comparison calculations (Figs. 4.1 and 4.2) were performed in lower dimensions to confirm that the Π_0 cut-off along with the A truncation on $\Pi_{\mathcal{T}}$ are negligible within numerical error at large cut-off A.

With a reliable methodology developed, we successfully calculated an example of Π_s in Section 4.2 in $d = 4$ spacetime where the finite-temperature $\Pi_{\mathcal{T}}$ and zero-temperature Π_0 for the one-loop self-energy topology separately diverge but yield a convergent finite-temperature correction Π_s.

5.1 Future Directions in Field Theory

This thesis is a proof of concept for adapting QFT numerical loop calculation methods to thermal field theory. It is also a starting point for including more complicated topologies in this new numerical calculation methodology.

We have been focusing on the one-loop self-energy topology for scalar field in this thesis. Another possible starting topology would be the self-energy tadpole topology. Equation (1.28) contains an expanded version of such correlator topology in thermal field theory. The reason that we did not choose this topology for the proof of concept of our methodology is that the tadpole topology diverges in $d = 2$ spacetime and higher spacetime dimensions give even worse divergences. Therefore, we would not have a benchmark reference for our methodology. Now that we have a reliable numerical method for the one-loop self-energy topology, we are able to apply it to the tadpole topology to find the known $\Pi_s(\mathcal{T}) \sim \mathcal{T}^2$ relationship from Eq. (1.28) (Ref. [1]):

$$\Pi_{\mathcal{T}} = \lambda \left[\frac{\Lambda^2}{16\pi^2} + \frac{\mathcal{T}^2}{24} + \dots \right]. \tag{5.1}$$

The expression for Π_s for the tadpole topology for numerical calculation is

$$\begin{aligned}\Pi_s^{\text{tadpole}} &= \Pi_{\mathcal{T}}^{\text{tadpole}} - \Pi_0^{\text{tadpole}} \\ &= \frac{1}{2\beta} \sum_{n=-A}^{A} \int \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{(\frac{2n\pi}{\beta})^2 + \boldsymbol{k}^2 + m^2} - \frac{1}{2} \sum_{n=-A}^{A} \int_{2\pi n/\beta}^{2\pi(n+1)/\beta} \frac{dk_0^E}{2\pi} \frac{d^3\boldsymbol{k}}{(2\pi)^3} \frac{1}{k^2 + m^2}.\end{aligned} \tag{5.2}$$

A log-log plot can provide a direct visual confirmation for the relation $\Pi_s(\mathcal{T}) \sim \mathcal{T}^2$,

$$\ln(\Pi_s) = 2\ln(\mathcal{T}) + \text{constant}. \tag{5.3}$$

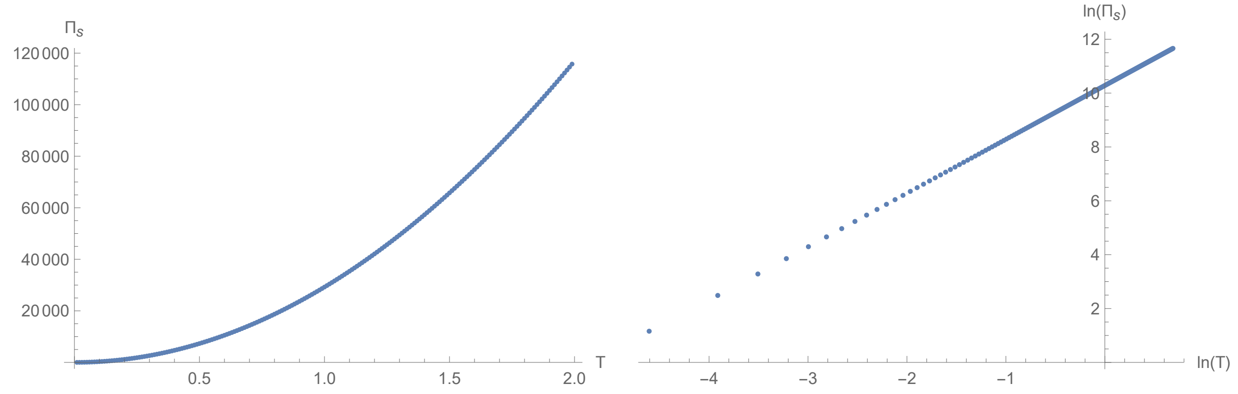


Figure 5.1: On the LHS is a plot of the finite-temperature correction Π_s with respect to $\mathcal{T} = 1/\beta$ at $d = 4$ spacetime for the self-energy topology. On the RHS is the log-log plot for the Π_s vs $\mathcal{T}$ relation. The slope on the right plot is approximately 1.99946. The mass parameter used in the calculation is $m = 1.1$. The maximum $|n|$ is up to 100 (i.e., the parameter $A = 100$).

Figure 5.1 shows excellent agreement with the $\Pi_s \sim \mathcal{T}^2$ relation because the log-log plot provides a slope close to 2.

Both the TBI (Fig. 2.1) and the tadpole (Eq. (1.28)) topologies have one-loop structure in their Feynman diagrams. They are the simplest cases for numerical calculations which allowed us to focus only on the basic single loop calculations. Now that we have a methodology built for the one-loop self-energy topology, in theory, we can compute higher-loop Feynman diagrams with scalar fields. Each loop will bring in a discretized summation for its temporal dimension. For example, a two-point, two-loop topology will have a Feynman integral with two scalar loops. Each loop has its own summation over an index. That means a pair of summations over n_1 and n_2, and would require cut-offs in n_1, n_2 and their corresponding Π_0 integrations.

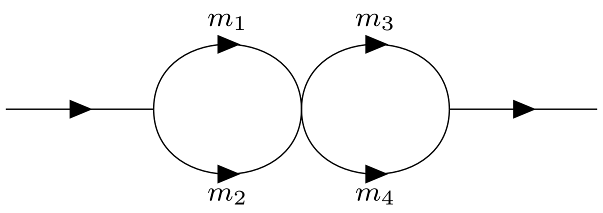


Figure 5.2: Two-Point, two-loop Feynman diagram with scalar fields.

Using the same basic ideas developed in this thesis, we can compute thermal field theory correlators that have more than one loop. Our methodology can be generalized to the Feynman rules for any topology with loop structure in thermal field theory.

The cut-off methodology from Chapter 3 is a robust way to calculate the finite-temperature correction for the one-loop self-energy correlation function. However, when we were studying the large n behavior in the summation in Section 3.2, within the dimensional regularization $d = 4 - 2\epsilon$ we reached an expression for.................................... $\Pi_{\mathcal{T}}$ as in Eq. (3.5). If we use ζ function to transform the expression [20]

$$\sum_{n=1}^{\infty} \frac{1}{n^{1+2\epsilon}} \equiv \zeta(1+2\epsilon), \tag{5.4}$$

which gives the ϵ expansion

$$\zeta(1+2\epsilon) = \frac{1}{2\epsilon} + \gamma_E + \vartheta(\epsilon). \tag{5.5}$$

Now combining with Eq. (3.5), $\Pi_{\mathcal{T}}$ now has an expression with $\frac{1}{\epsilon}$ as

$$\begin{aligned} \Pi_{\mathcal{T}} &\approx \sum_n \frac{1}{32\pi^2 |n|^{1+2\epsilon}} \\ &\approx \frac{2}{32\pi^2}\left(\frac{1}{2\epsilon} + \gamma_E\right), \end{aligned} \tag{5.6}$$

where the factor of 2 comes from the negative and positive values of n. Equation (5.6) does not include the $n = 0$ term. Then we have the finite-temperature correction in terms of ϵ at large temperature $\mathcal{T}$ limit (equivalent to large n limit)

$$\lim_{\mathcal{T}\to\infty} = \Pi_{\mathcal{T}}(n=0) + \frac{2}{32\pi^2}\left(\frac{1}{2\epsilon} + \gamma_E\right) - \Pi_0. \tag{5.7}$$

It can easily be verified that Π_0 has a $\frac{1}{32\pi^2\epsilon}$ divergence, which leaves Eq. (5.7) finite as $\epsilon \to 0$. Thus with Eq. (5.7), we can calculate $\Pi_{\mathcal{T}}$ and Π_0 purely in dimensional regularization without cutoff. We can possibly generalize this method to more topologies. This is another promising approach for thermal field theory finite-temperature correction numerical calculations that could be developed in future work.

Bibliography

[1] F. Gelis, *Quantum Field Theory: From Basics to Modern Topics* (Cambridge University Press, 2019).

[2] M. Gogberashvili, Advances in High Energy Physics **2018**, 1–5 (2018).

[3] J. Ellis, M. Lewicki, and J. M. No, Journal of Cosmology and Astroparticle Physics **2020**, 050–050 (2020).

[4] W. Huang, F. Sannino, and Z. Wang, Physical Review D **102** (2020).

[5] V. A. Smirnov, *Analytic tools for Feynman integrals* (Springer, 2013).

[6] S. Borowka, G. Heinrich, S. Jahn, S. Jones, M. Kerner, J. Schlenk, and T. Zirke, Computer Physics Communications **222**, 313–326 (2018).

[7] T.-P. Cheng and L.-F. Li, *Gauge theory of elementary particle physics* (Oxford university press, 1994).

[8] M. E. Peskin, *An introduction to quantum field theory* (CRC press, 2018).

[9] A. Das, *Finite temperature field theory* (World scientific, 1997).

[10] T. Matsubara, Progress of theoretical physics **14**, 351 (1955).

[11] A. I. Davydychev, J. Math. Phys. **32**, 1052 (1991).

[12] P. Pascual and R. Tarrach, *QCD: Renormalization for the Practitioner*, Lecture notes in physics (Springer-Verlag, 1984).

[13] R. Kleiv, Qcd sum rule studies of heavy quarkonium-like states, 2014, arXiv: 1407.2292.

[14] R. Mertig and R. Scharf, Comput. Phys. Commun. **111**, 265 (1998), arXiv: hep-ph/9801383.

[15] T. Huber and D. Maitre, Comput. Phys. Commun. **175**, 122 (2006), arXiv: hep-ph/0507094.

[16] K. Nakamura *et al.*, Journal of Physics G Nuclear and Particle Physics **86** (2010).

[17] J. C. Collins and J. C. Collins, *Renormalization: an introduction to renormalization, the renormalization group and the operator-product expansion* (Cambridge university press, 1985).

[18] M. D. Schwartz, *Quantum field theory and the standard model* (Cambridge University Press, 2014).

[19] G. Leibbrandt, Reviews of Modern Physics **47**, 849 (1975).

[20] M. Laine and A. Vuorinen, *Basics of Thermal Field Theory* (Springer International Publishing, 2016).

Appendix A

TBI Comparison with massive TBI Integrals

Appendix A includes the graphs for `MATHEMATICA` analytic calculation results comparing with pySecDec numerical calculation results. The mass values for the internal lines are defined 4.18 GeV to align with the bottom quark mass. By exploring a different mass input, we were ensuring that pySecDec is accurate across a wide range of physical scale.

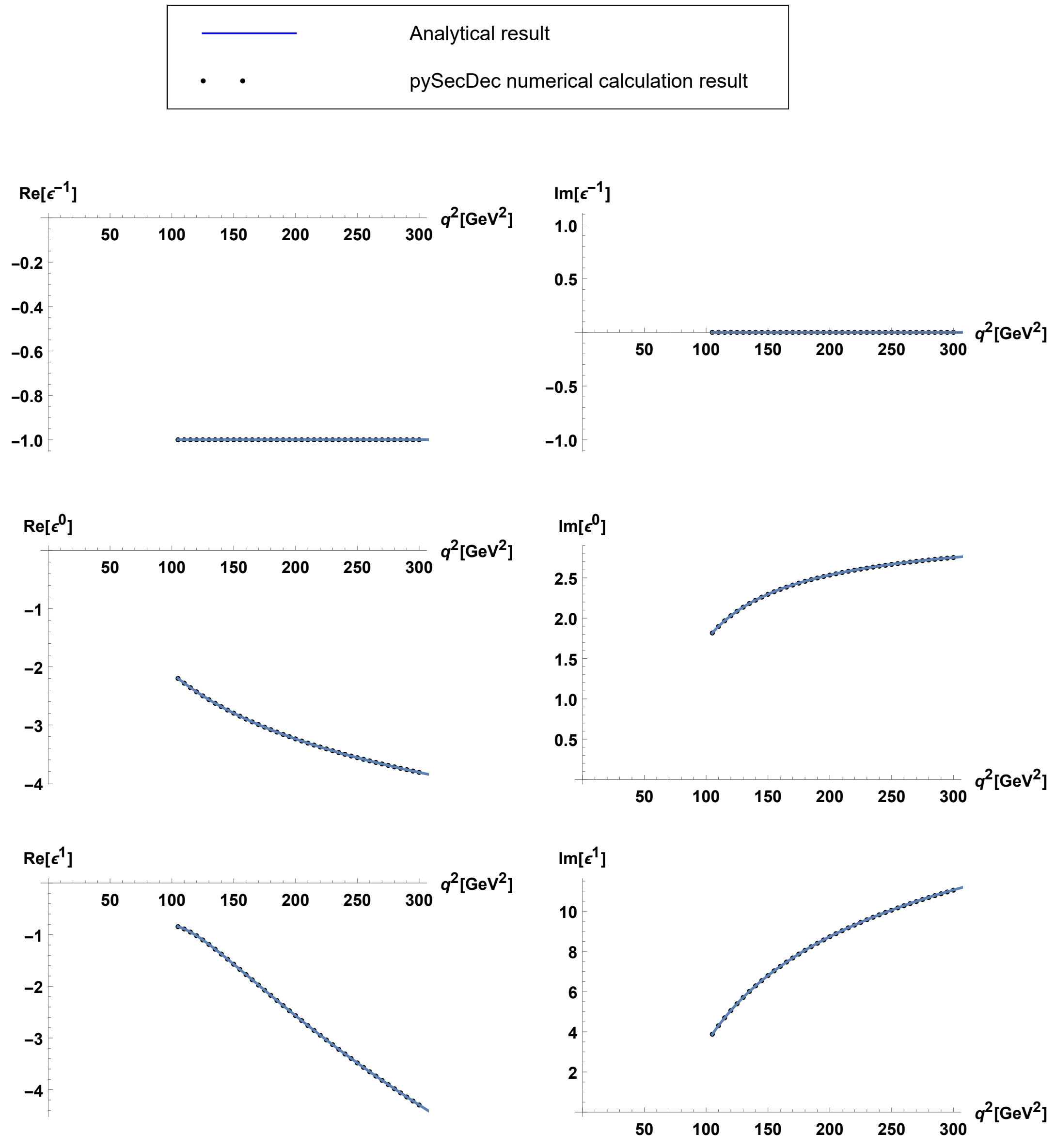


Figure A.1: ϵ^{-1} , ϵ^{0} and ϵ^{1} coefficients comparison graphs for analytical and numerical results for the one-loop self-energy integral ($m_1 = m_2 = 4.18\,\text{GeV}$).

Appendix B

Integration Code and Results for Testing Convergence in Π_s

Appendix B includes the python code involving pySecDec program to test if the methodology proposed in Eq. (3.28) gives a convergent finite-temperature correction Π_s. The calculation results are shown in Fig. 3.2. The MATHEMATICA code that generates Fig. 3.2 and the log-log plot Fig. 3.3 are also attached after the python code.

```
from __future__ import print_function
from pySecDec.integral_interface import IntegralLibrary
import sympy as sp
import numpy as np
import scipy.integrate as si

# load c++ library
zero_temp_cal = IntegralLibrary('/home/siyuan/Documents/
   ↪ self_energy_3D_corrected/self_energy_3D_corrected_pylink.so')
finite_temp_cal = IntegralLibrary('/home/siyuan/Documents/
   ↪ self_energy_3D_corrected/self_energy_3D_corrected_pylink.so')

# choose integrator
       # pySecDec integration method
zero_temp_cal.use_Vegas(flags=0) # ``flags=2``: verbose --> see Cuba manual
finite_temp_cal.use_Vegas(flags=0)

# paremates

n_range = np.linspace (-300,300, 601) # Memory error, start small
m1, m2 = 1.1,2



beta = 0.1

# Momentum input
p_vec = np.array([1,0,0])
pp = np.sum(np.square(p_vec))
#l = 1
#p0 = 2*l*np.pi/beta
p0 = 7
```

```
# Function Definitions
def m1Sq(n):
    return m1**2 + (2*np.pi*n/beta)**2
def m2Sq(n):
    return m2**2 + (2*np.pi*n/beta + p0)**2
def zero_psd(p_0, k_0E):
    raw_zero = zero_temp_cal(complex_parameters = [-pp], real_parameters=[(
       ↪ m1**2+k_0E**2), (m2**2+(k_0E+p_0)**2)])
    str_raw_zero = raw_zero[2]
    # convert complex numbers from c++ to sympy notation for zero temp 0+1D
       ↪  integral before integrate it over the n interval
    str_raw_zero = str_raw_zero.replace(',','+I*')
    # convert to sympy expressions
    zero_psd_result = sp.sympify(str_raw_zero.replace('+/-','*value+error*')
       ↪ )
    zero_psd_result_err = sp.sympify(str_raw_zero.replace('+/-','*value+
       ↪ error*'))
    return zero_psd_result.coeff('eps',0).coeff('value')

# Set variables
sub_sum = 0

# Start summation over index n
for n in n_range:
    #psd calculation of 0+nD integral
    raw_finite = finite_temp_cal(complex_parameters = [-pp], real_parameters
       ↪ =[m1Sq(n), m2Sq(n)])


    #integrate over n and (n+1) interval
    k_0E_range = np.linspace(2*(n)*np.pi/beta, 2*(n+1)*np.pi/beta, 7) #x
        #p_0 over grid for manual integral

    y = [zero_psd(p0, k_0E) for k_0E in k_0E_range]

    int_zero = si.simps(y, k_0E_range)

    #convert finite temp to proper format before the subtraction
    str_raw_finite = raw_finite[2]
    str_raw_finite = str_raw_finite.replace(',','+I*')
    finite_psd_result = sp.sympify(str_raw_finite.replace('+/-','*value+
       ↪ error*'))
    finite_psd = finite_psd_result.coeff('eps',0).coeff('value')
```

```
# the subtraction
sub_raw = finite_psd/(2*beta) - int_zero/(4*np.pi)

#print the subtration results conrresponding to each individual n
print('{ "%s" , "%s" , "%s" ,"%s" , "%s","%s" , "%s"},' % (n, sp.re(
    ↪ sub_raw), sp.im(sub_raw), sp.re(finite_psd/(2*beta)), sp.im(
    ↪ finite_psd/(2*beta)), sp.re(int_zero/(4*np.pi)), sp.im(int_zero
    ↪ /(4*np.pi))))
```

Integration Result Analysis for Testing Convergence in Π_s

Input variables :

n = -300~300

m_1 = 1.1

m_2 = 2

p_0 = 7

p_1 = 1

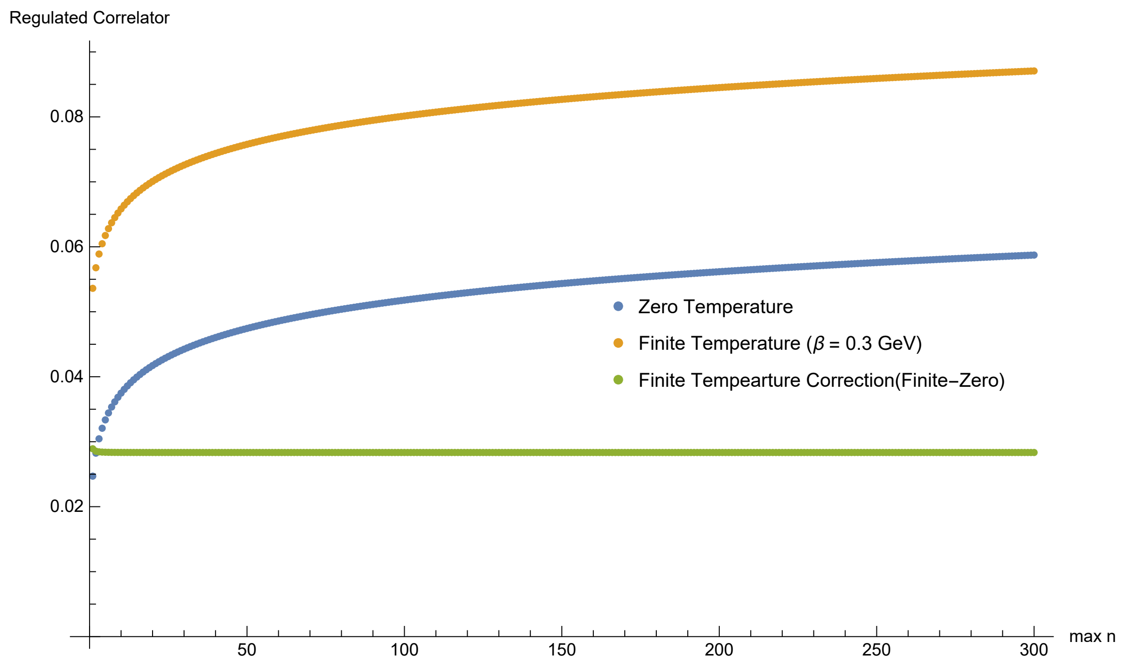
```
In[4003]:= files = {"1+3D_data_corrected_v8.m", "1+3D_data_corrected_mit_err_v9.m"};
           data = Get[#, Path → NotebookDirectory[]] & /@ files;
```

1+3D corrected version 8

```
In[4005]:= subresult3Dstrv8  = data[[1]][[All, 2]];
           sub3Dstrrepv8 = StringReplace[subresult3Dstrv8, {"e" → "×10^"}];
           subresult3Dv8 = ToExpression[sub3Dstrrepv8];

In[4008]:= Table[Total[
              subresult3Dv8[[(Length[subresult3Dv8] + 1) / 2 - n ;; (Length[subresult3Dv8] + 1) / 2 + n]]],
             {n, 1, (Length[subresult3Dv8] - 1) / 2}];

In[4009]:= ftreal3Dv8  = data[[1]][[All, 4]]; zeroreal3Dv8 = data[[1]][[All, 6]];

In[4010]:= ftreal3Dv8  = data[[1]][[All, 4]];
           ftstrrepv8 = StringReplace[ftreal3Dv8, {"e" → "×10^"}];
           ft3Dv8 = ToExpression[ftstrrepv8];

In[4013]:= n3Dv8  = ToExpression[ data[[1]][[All, 1]]];

In[4014]:= errv9str  = StringReplace[data[[2]][[All, 8]], {"e" → "×10^"}];

In[4015]:= errv9  = ToExpression[errv9str];

In[4016]:= zero3Dv8 = data[[1]][[All, 6]];
           zerostrrepv8  = StringReplace[zero3Dv8, {"e" → "×10^"}];
           zero3Dv8  = ToExpression[zerostrrepv8];

In[4019]:= subv8 = Table[Total[subresult3Dv8[[(Length[subresult3Dv8] + 1) / 2 - n ;;
                  (Length[subresult3Dv8] + 1) / 2 + n]]], {n, 1, (Length[subresult3Dv8] - 1) / 2}];
           ftv8 = Table[Total[ft3Dv8[[(Length[ft3Dv8] + 1) / 2 - n ;; (Length[ft3Dv8] + 1) / 2 + n]]],
              {n, 1, (Length[ft3Dv8] - 1) / 2}];
           zerov8 = Table[Total[zero3Dv8[[(Length[zero3Dv8] + 1) / 2 - n ;;
                  (Length[zero3Dv8] + 1) / 2 + n]]], {n, 1, (Length[zero3Dv8] - 1) / 2}];

In[4022]:= errsumv9  = Table[Total[errv9[[(Length[errv9] + 1) / 2 - n ;; (Length[errv9] + 1) / 2 + n]]],
              {n, 1, (Length[errv9] - 1) / 2}];
```

In[4023]:=
```
ListPlot[{zerov8, ftv8, subv8},
 PlotLegends → Placed[{"Zero Temperature", "Finite Temperature  (β = 0.3 GeV)",
    "Finite Tempearture Correction(Finite-Zero)" }, {0.75, 0.5}],
 ImageSize → Large, AxesLabel → {"max n", "Regulated Correlator"}]
```

Out[4023]=



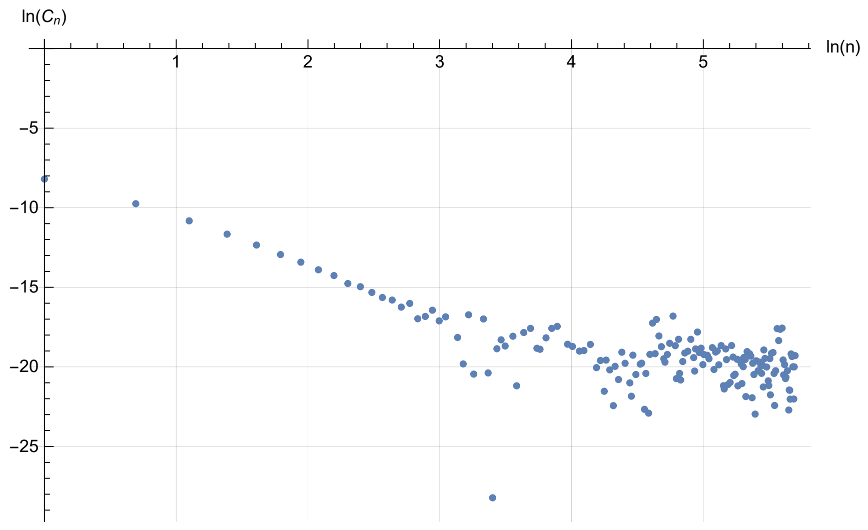
In[4024]:=
```
cauchyv8 = Table[subv8[[n]] - subv8[[n + 1]], {n, 1, Length[subv8] - 1}];
```

In[4025]:=
```
testv8 = Table[cauchyv8[[n]] - cauchyv8[[n + 1]], {n, 1, Length[cauchyv8] - 1}];
```

In[4026]:=
```
loglog = Table[{Log[n], Log[testv8[[n]]]}, {n, 1., 298.}];
```

In[4027]:=
```
ListPlot[loglog, AxesLabel → {"ln(n)", "ln(Cn)"}, GridLines → Automatic]
```

Out[4027]=



Appendix C

Integration Code for Benchmarking Π_0 Cut-off in 2-dimensional Spacetime

Appendix C includes the python code utilizing pySecDec package to compare $\Pi_s = \Pi_{\mathcal{T}} - \Pi_0$ calculated from two slightly different methods. One with truncated $\Pi_{\mathcal{T}}$ and exact Π_0. the other with truncated $\Pi_{\mathcal{T}}$ and Π_0 with cut-off at $2\pi|n|_{max}/\beta$. The code gives the example of the $d = 2$ spacetime calculation code. The $d = 3$ spacetime would be similar but with different .so file and input physical variables, which is presented in the caption of Fig. 4.2. The computation result is plotted in Fig. 4.1.

Following is the python code for $\Pi_s = \Pi_{\mathcal{T}} - \Pi_0$ with Π_0 calculated uses only pySecDec in $d = 2$ spacetime.

```
from __future__ import print_function
from pySecDec.integral_interface import IntegralLibrary
import sympy as sp
import numpy as np
import scipy.integrate as si

# load c++ library
zero_temp_cal = IntegralLibrary('/home/siyuan/Documents/
   ↪ self_energy_2D_corrected/self_energy_2D_corrected_pylink.so')
finite_temp_cal = IntegralLibrary('/home/siyuan/Documents/
   ↪ self_energy_1D_corrected/self_energy_1D_corrected_pylink.so')

# choose integrator
       # pySecDec integration method
zero_temp_cal.use_Vegas(flags=0) # ``flags=2``: verbose --> see Cuba manual
finite_temp_cal.use_Vegas(flags=0)

# parematers
n_range = np.linspace (-100,100, 201)
m1, m2 = 1.1,1.2

l = 1
beta = 0.1

# Momentum input
p_vec = np.array([1.3,0,0])
pp = np.sum(np.square(p_vec))
```

```
#Function Definitions
def m1Sq(n):
    return m1**2 + (2*np.pi*n/beta)**2
def m2Sq(n):
    return m2**2 + (2*np.pi*n/beta + p0)**2
def zero_psd(p_0):
    raw_zero = zero_temp_cal(complex_parameters = [-(p_0)**2-pp],
        ↪ real_parameters=[(m1**2), (m2**2)])
    str_raw_zero = raw_zero[2]
    # convert complex numbers from c++ to sympy notation for zero temp 0+1D
        ↪  integral before integrate it over the n interval
    str_raw_zero = str_raw_zero.replace(',','+I*')
    # convert to sympy expressions
    zero_psd_result = sp.sympify(str_raw_zero.replace('+/-','*value+error*')
        ↪ )
    zero_psd_result_err = sp.sympify(str_raw_zero.replace('+/-','*value+
        ↪ error*'))
    return zero_psd_result.coeff('eps',0).coeff('value')


#Set variables
p0 = 2*np.pi/beta
ft_sum=0

for n in n_range:
    #psd calculation of 0+nD integral
    raw_finite = finite_temp_cal(complex_parameters = [-pp], real_parameters
        ↪ =[m1Sq(n), m2Sq(n)])

    #convert finite temp to proper format before the subtraction
    str_raw_finite = raw_finite[2]
    str_raw_finite = str_raw_finite.replace(',','+I*')
    finite_psd_result = sp.sympify(str_raw_finite.replace('+/-','*value+
        ↪ error*'))
    finite_psd = finite_psd_result.coeff('eps',0).coeff('value')
    finite_psd_err = finite_psd_result.coeff('eps',0).coeff('error')/(2*beta
        ↪ )

    # the subtraction
    ft_sum = ft_sum + finite_psd/(2*beta)
    #print the subtration results conrresponding to each individual n
    print('{ "%s" , "%s" , "%s", "%s"  },' % (n, sp.re(ft_sum), sp.re(
       ↪ finite_psd/(2*beta)),sp.re(finite_psd_err)))

zero_temp = zero_psd(p0)

print('{"%s", "%s"}' %(sp.re(zero_temp)/2,sp.im(zero_temp)/2))
```

Following is the python code for $\Pi_s = \Pi_{\mathcal{T}} - \Pi_0$ with Π_0 calculated uses cut-off at $2\pi|n|_{max}/\beta$ referring to Eq. (3.28).

```
from __future__ import print_function
from pySecDec.integral_interface import IntegralLibrary
import sympy as sp
import numpy as np
import scipy.integrate as si

# load c++ library
zero_temp_cal = IntegralLibrary('/home/siyuan/Documents/
   ↪ self_energy_1D_corrected/self_energy_1D_corrected_pylink.so')
finite_temp_cal = IntegralLibrary('/home/siyuan/Documents/
   ↪ self_energy_1D_corrected/self_energy_1D_corrected_pylink.so')

# choose integrator
# pySecDec integration method
zero_temp_cal.use_Vegas(flags=0) # ``flags=2``: verbose --> see Cuba manual
finite_temp_cal.use_Vegas(flags=0)

# Parameters:
# l = 1
# p0 = 2*l*np.pi/beta
beta = 0.1
p0 = 2*np.pi/beta # Euclidean
p_vec = np.array([1.3,0,0])
pp = np.sum(np.square(p_vec))
m1, m2 = 1.1, 1.2

# n_range = np.linspace(-50, 50, 101) # Memory error, start small


# first mass-squared parameter for finite temp correlator
def m1Sq(n):
    return m1**2 + (2*np.pi*n/beta)**2

# second mass-squared parameter for the finite temp correlator
def m2Sq(n):
    return m2**2 + (2*np.pi*n/beta + p0)**2


def zero_psd(p_0, k_0E):
    """Compute

    Parameters
    ----------
    p_0 : Real
        Euclideanized p0 (external momentum).
    k_0E : Real
        Euclideanized k0 (loop momentum).

    Returns
    -------
    Sympy expression.
        Spatial integral at zero temp from pySecDec libraries.
    """
    # zero temperature spatial integral
    raw_zero = zero_temp_cal(complex_parameters=[-pp],
                             real_parameters=[(m1**2 + k_0E**2),
                                              (m2**2 + (k_0E + p_0)**2)])
    str_raw_zero = raw_zero[2]
    # convert complex numbers to sympy notation
    str_raw_zero = str_raw_zero.replace(',','+I*')
    # convert to sympy expressions
    zero_psd_result = sp.sympify(str_raw_zero.replace('+/-', '*value+error*'
        ↪ ))
    # zero_psd_result_err = sp.sympify(str_raw_zero.replace('+/-', '*value+
        ↪ error*'))
    return zero_psd_result.coeff('eps', 0).coeff('value')


fin_corr = 0 # running sum of finite temp correlator
zero_corr = 0 # running sum of zero temp correlator
nmax = 100 # max series index
nmin = -nmax # min series index
intervals = 51 # intervals used in Simpson's rule
for n in range(nmin, nmax + 1):
    # finite temp spatial integral
    raw_finite = finite_temp_cal(complex_parameters=[-pp],
                                 real_parameters=[m1Sq(n), m2Sq(n)])

    # convert finite temp output
    str_raw_finite = raw_finite[2]
    str_raw_finite = str_raw_finite.replace(',','+I*')
    finite_psd_result = sp.sympify(str_raw_finite.replace('+/-',
                                                           '*value+error*'))
    finite_psd = finite_psd_result.coeff('eps',0).coeff('value')
    # update running sum
    fin_corr += finite_psd/(2*beta)

    # If n is NOT nmax, integrate the zero temp spatial integral over the
    # n to (n + 1) interval
    if n < nmax:
        k_0E_range = np.linspace(2*n*np.pi/beta, 2*(n + 1)*np.pi/beta,
            ↪ intervals)
        y = [zero_psd(p0, k_0E) for k_0E in k_0E_range]
        int_zero = si.simps(y, k_0E_range)
        # update running sum
        #zero_corr += int_zero/(4*np.pi)

    # the subtraction
    sub_raw = finite_psd/(2*beta) - int_zero/(4*np.pi)

    # print the subtration results conrresponding to each individual n

    print('{ "%s" , "%s" , "%s" ,"%s" , "%s","%s" , "%s"},' %
          (n,
           sp.re(sub_raw), sp.im(sub_raw),
           sp.re(finite_psd/(2*beta)), sp.im(finite_psd/(2*beta)),
           sp.re(int_zero/(4*np.pi)), sp.im(int_zero/(4*np.pi)))
          )
```

Appendix D

Integration Code for Calculating the relation between Π_s and external momentum p in 2-dimensional Spacetime

This section will present the calculation python code for finite-temperature correction Π_s with respect to external momentum p in $d = 2$ spacetime. The temperature factor is controlled at a constant of $\beta = 0.3$. The calculation result was plotted in Fig. 4.3. The `MATHEMATICA` file for the mesh plot Fig. 4.3 of the data generated from the python code is also attached at the end of this section.

The redundant part of loading .so files and setting up mass functions for the correlator of the one-loop self-energy topology are as in Appendix C and omitted here.

```
# Variable values
p0_min = -20
p0_max = -p0_min
p1_min = -20
p1_max = -p1_min
nmax = 100 # max series index
nmin = -nmax # min series index
intervals = 50 # intervals used in Simpson's rule

for p0 in range(p0_min,p0_max,2):
        sub_sum = 0
        for p1 in range(p1_min,p1_max,2):
                sub_raw = 0
                for n in range(nmin, nmax + 1):
                    # finite temp spatial integral
                    raw_finite = finite_temp_cal(complex_parameters=[-p1**2],
                                                 real_parameters=[m1Sq(n), m2Sq(n)
                                                    ↪ ])

                    # convert finite temp output
                    str_raw_finite = raw_finite[0]
                    str_raw_finite = str_raw_finite.replace(',','+I*')
                    finite_psd_result = sp.sympify(str_raw_finite.replace('+/-
                        ↪ ',
                                                                           '*value+
                                                              ↪ error
                                                              ↪ *'))
        finite_psd = finite_psd_result.coeff('eps',0).coeff('value
            ↪ ')
        # update running sum
        fin_corr += finite_psd/(2*beta)

        # If n is NOT nmax, integrate the zero temp spatial
            ↪ integral over the
        # n to (n + 1) interval
        if n < nmax:
            k_0E_range = np.linspace(2*n*np.pi/beta, 2*(n + 1)*np.
                ↪ pi/beta, intervals)
            y = [zero_psd(p0, k_0E, p1) for k_0E in k_0E_range]
            int_zero = si.simps(y, k_0E_range)

        # the subtraction
        sub_raw = finite_psd/(2*beta) - int_zero/(4*np.pi)

    sub_sum = sub_sum + sub_raw

    print('{ "%s" , "%s" , "%s"  "%s" },' %
      (p0, p1,
       sp.re(sub_sum), sp.im(sub_sum)))
```

Integration Result Mesh – Plot for Π_s vs External Momentum $p = (p_0, p_1)$ in 2 d Spacetime

```
In[746]:= files = {"pi_vs_p_1+1D_addition.m"};
          data = Get[#, Path → NotebookDirectory[]] & /@ files;
```

For 1 + 1 D

n = -100~ 100

m1 = 1.1

m2 = 1.2

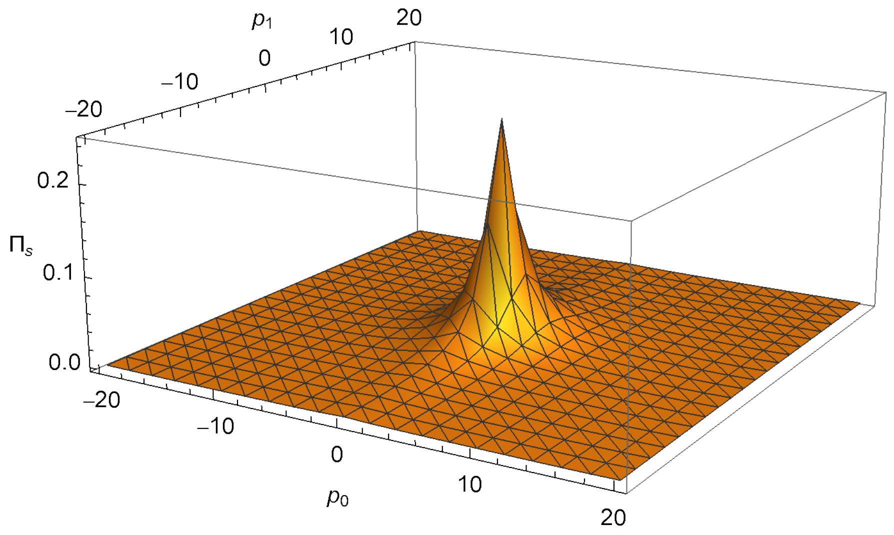
p_0 = -20 ~ 20

p_1 = -20 ~ 20

β = 0.3

```
In[748]:= data2Dv2str = data[[1]][[All, {1, 2, 3}]];
          data2Dv2strrep = StringReplace[data2Dv2str[[All, 3]], {"e" → "×10^"}];

In[750]:= data2Dv2col1 = ToExpression[data2Dv2str[[All, 1]]];
          data2Dv2col2 = ToExpression[data2Dv2str[[All, 2]]];
          data2Dv2col3 = ToExpression[data2Dv2strrep];
          data2Dv2 = Transpose[{data2Dv2col1, data2Dv2col2, data2Dv2col3}];

In[759]:= ListPlot3D[data2Dv2, Mesh → All, AxesLabel → {"p0", "p1", "Πs"},
           PlotRange → Full, LabelStyle → Directive[12]]
```

Out[759]=



Appendix E

Integration Code for Calculating the relation between Π_s and Temperature Factor β in 2-dimensional Spacetime

This section will present the calculation python and MATHEMATICA code for finite-temperature correction Π_s with respect to thermal β and temperature $\mathcal{T} = \frac{1}{\beta}$ in $d = 2$ spacetime. The external momentum is set at constant values of $p = (7, 8, 0, 0)$. The calculation result was plotted in Fig. 4.5. As in Appendix D, the repeated part of code would be omitted.

```
#Set variables
p1 = 8

for T_100 in range(T_min,T_max+1):
        sub_sum = 0
        sub_raw = 0
        ft_sum=0
        T = T_100/100
        beta = 1./T
        #p0 = 2*np.pi/beta
        p0 = 7
        for n in n_range:
            #psd calculation of 0+nD integral
            raw_finite = finite_temp_cal(complex_parameters = [-(p1)**2],
               ↪ real_parameters=[m1Sq(n),m2Sq(n)])

            #convert finite temp to proper format before the subtraction
            str_raw_finite = raw_finite[2]
            str_raw_finite = str_raw_finite.replace(',','+I*')
            finite_psd_result = sp.sympify(str_raw_finite.replace('+/-','*
               ↪ value+error*'))
            finite_psd = finite_psd_result.coeff('eps',0).coeff('value')
            finite_psd_err = finite_psd_result.coeff('eps',0).coeff('error')
               ↪ /(2*beta)

            # update running sum
            fin_corr += finite_psd/(2*beta)
```

```
    # If n is NOT nmax, integrate the zero temp spatial integral
       ↪ over the
    # n to (n + 1) interval
    if n < nmax:
        k_0E_range = np.linspace(2*n*np.pi/beta, 2*(n + 1)*np.pi/beta,
            ↪  intervals)
        y = [zero_psd(p0, k_0E, p1) for k_0E in k_0E_range]
        int_zero = si.simps(y, k_0E_range)

    # the subtraction
    sub_raw = finite_psd/(2*beta) - int_zero/(4*np.pi)

sub_sum = sub_sum + sub_raw

print('{ "%s" , "%s" , "%s"  "%s" },' % (p0, p1, sp.re(sub_sum), sp.
   ↪ im(sub_sum)))
```

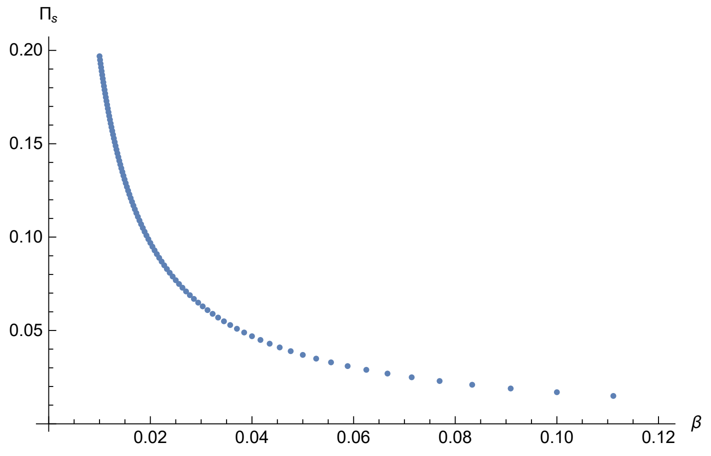
Integration Result Plot for Π_s vs β (and $1/\beta$) in 2 d Spacetime

```
In[5100]:= files = {"1+1D_beta.m"};
            data = Get[#, Path → NotebookDirectory[]] & /@ files;
            data2Dbetav2str = data[[1]][[All, 2]];
```

1 + 1 D

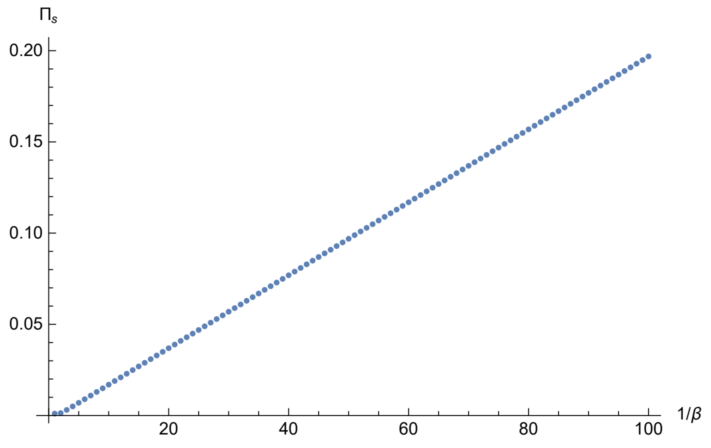
```
In[5103]:= data2Dbetav2strrep = StringReplace[data2Dbetav2str, {"e" → "×10^"}];
            data2Dbetav2 = ToExpression[data2Dbetav2strrep];

In[5105]:= data2Dv2beta1 = ToExpression[data[[1]][[All, 1]]];
            data2Dv2beta = Transpose[{data2Dv2beta1, data2Dbetav2}];
            ListPlot[data2Dv2beta, AxesLabel → {"β", "Πs"}]
```

Out[5107]=



```
In[5108]:= Tv2 = Transpose[{1. / data2Dv2beta1, data2Dbetav2}];

In[5111]:= (Tv2[[70, 2]] - Tv2[[20, 2]]) / (Tv2[[70, 1]] - Tv2[[20, 1]])

Out[5111]= 0.00199939

In[5110]:= ListPlot[Tv2, PlotRange → All, AxesLabel → {"1/β", "Πs"}]
```

Out[5110]=


Π_s
$1/\beta$