1.Introduction

The study of the correlation between the particle-like systems in a many-body framework can predict a wide range of novel phenomena in condensed matter physics, especially in the aspects of phase transition. This perspective is crucial for the recent advances in quantum technologies such as quantum simulators implemented through optical lattice, superconducting circuits, and quantum materials (e.g., quantum dots).

The Phase transition between metal and insulator is defined as a rapid transition of conductivity. In the metallic phase, inside the material structure, there are free electrons exist and make available to electric current flows throughout the material, in the insulating phase the electrons are bonded tightly with atoms which hinders the flow of electric current.Several theories have been developed to describe this phase transition phenomenon. In the Mott transition, the change of conductivity is explained through the Coulomb interactions between the electrons. While the density of the electrons inside the material becomes saturated, the interaction effect leads to repulsion between the particles which impedes current flow. Alternatively, In Andersen's transition model, the transition between the two phases is explained from more microscopic points of view by using the concept of disorder and impurities.

1.1 Summary

In our study, we investigate the phase transition of a resistively shunted Josephson junction within the framework of the quantum impurity model. We establish a quantum mechanical framework based on the state vector of the macroscopic Josephson junction model. As starting in the very first step, we use impurity solving method (Non-crossing approximation, One-crossing approximation, TOA) in thermally stable state with Matsubara formalism , then evaluate the evolution of correlation of energy change ratio.

By employing the framework of strong correlation method, we expect a concrete description of the phase transition in a resistively shunted Josephson junction which is an ongoing debate. The precise nature of this transition is to be addressed through the analysis of the quantum state of electrons.

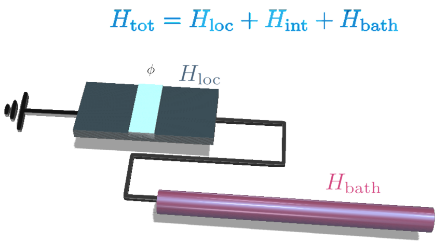

Figure 1: Brief Image of total circuit.("LATER FIX")

1.2 Previous work

A resistively shunted Josephson junction(RSJJ) is a circuit consisting of a single Josephson junction simultaneously connected to an RC resonator structure which acts as a circuit resistance. Several studies have reported that the Josephson junction in this structure reveals a unique phase transition scheme that depends on the coupling strength between the Junction and the circuit resistance. This phenomenon allows RSJJ to play a physical role as a simulator for the quantum phase transition, especially for the Schmid quantum dissipative transition. Although RSJJ can be a well-defined physical structure for simulating the phase transition, it is limited to zero-temperature conditions, which is hard to achieve in an experimental approach.

1.3 Our approach

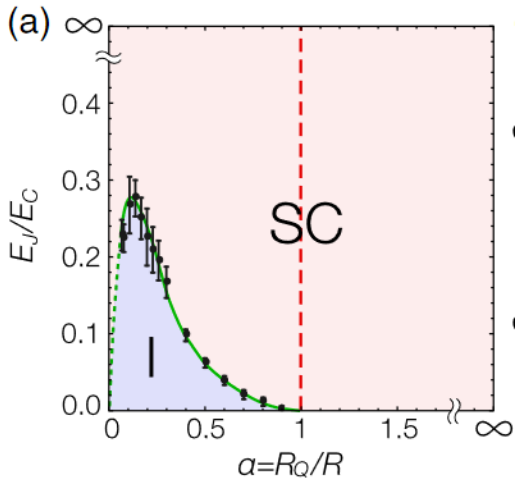

Figure 2: Zero temperature phase diagram with reentrant phase transition

In the latest study, a novel phase transition scheme called the 'reentrant phase transition' was predicted due to the potential independence of the dispersion relation of the RC resonator. Based on previous research, we'll investigate this structure in a finite-temperature scheme. We construct the RSJJ Hamiltonian as a pseudo particle impurity model, which is described by the action of system S separated into two parts: for local system S_{loc} and for interaction S_{int},

$$S = S_{loc} + S_{int}$$

using Matsubara Green's function.

In our model, S_{loc} will be correspond to the dynamics of Josephson junction, and S_{int} will represents the interaction of Josephson junction and the RC resonator.

2. Diagrammatic method

In this chapter, We will briefly introduce the theoretical framework of the method of Green's function, and pseudo-particle solvers, such as Non-crossing approximation. The detailed mathmatical derivation will treated on appendix.

2.1 Field operator and Green's function

In quantum many-body theory, the physical system is depicted over the frame of the fermionic(bosonic) creation and annihilation operator, and $\hat{a}(\hat{a}^\dagger)$ as an annihilation(creation) operator, it satisfies the following commutation relations:

$$[\hat{a_i}, \hat{a_j}] = \{\hat{a_i}, \hat{a_j}\} = 0$$
$$[\hat{a_i}^\dagger, \hat{a_j}^\dagger] = \{\hat{a_i}^\dagger, \hat{a_j}^\dagger\} = 0$$
$$[\hat{a_i}, \hat{a_j}^\dagger] = \{\hat{a_i}, \hat{a_j}^\dagger\} = \delta_{ij} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$

Where the square bracket informs the bosonic case, and its RHS indicates the fermionic case. In the following discussion, we will drop out the lower indices of the field operator, assuming it can indicate any arbitrary quantum states.

If we consider the Hamiltonian system can be split into two parts: for the local system and for interaction, we can write the total Hamiltonian H:

$$H = H_0 + H_{int}$$

Here, H_0 is the local(Free) Hamiltonian, H_{int} is the Hamiltonian for interaction (external potential). To express the dynamics of a given operator, we depict the time dependency of the operator in the interaction picture :

$$\hat{a}(t) = e^{iH_0 t}\hat{a}e^{-iH_0 t}$$
$$\hat{a}^\dagger(t) = e^{-iH_0 t}\hat{a}^\dagger e^{iH_0 t}$$

With the above time dependency, we can define the Green's function as follows :

$$G_0(t, t') = \frac{i}{\hbar}\langle \mathcal{T}\hat{a}^\dagger(t)\hat{a}(t')\rangle = \begin{cases} \frac{i}{\hbar}\langle \hat{a}^\dagger(t)\hat{a}(t')\rangle & (t > t') \\ \pm\frac{i}{\hbar}\langle \hat{a}(t')\hat{a}^\dagger(t)\rangle & (t < t') \end{cases}$$

Here, the symbol $\mathcal{T}$ is the time ordering operator. One of the features of Green's function is that it satisfies the

given equation of motion :

$$[i\partial_t - H]G(t,t') = \delta(t-t')$$
$$[i\partial_t - H_0]G_0(t,t') = [i\partial_t - H + H']G_0(t,t') = \delta(t-t')$$

2.2 Diagrammatic expansion of Green's function

In the statistical framework, it is easy to calculate Green's function in the imaginary time τ due to the wick rotation of the time axis, which mapped time-dependent system progression into thermal equilibrium problems with temperature dependency.

$$\frac{it}{\hbar} = \tau = \beta = \frac{1}{k_B T}$$

We will use $\hbar = 1$ in overall discussion. In this new frame, Green's function can be rewritten :

$$G_0(\tau,\tau') = \langle \mathcal{T}\hat{a}^\dagger(\tau)\hat{a}(\tau)\rangle \begin{cases} \langle \hat{a}^\dagger(\tau)\hat{a}(\tau')\rangle & (\tau > \tau') \\ \pm\langle \hat{a}(\tau')\hat{a}^\dagger(\tau)\rangle & (\tau < \tau') \end{cases}$$

There are particular relationship between real-time Green's function and imaginary time(temperature dependence) Green's function, which is :

$$G(t,t')|_{t=0} = \pm e^{\beta H} G(t,t')|_{t=i\beta}$$

These temperature dependence Green's function is often called in another term, Matsubara Green's function. Single Green's function corresponding with one specific imaginary time interval can be represented in a diagrammatic way, which is the very basic formulation of the diagrammatic method. We can draw a line from τ' to τ,

$\hat{a}(\tau')$ ⟶ $\hat{a}^\dagger(\tau)$ $\qquad\qquad$ $\hat{a}(\tau')$ ⟵ $\hat{a}^\dagger(\tau)$

According to statistical mechanics, the expectation value of the physical observable is characterized by the partition function that satisfies the given restricted energy condition in phase space. Based on these concepts, on the grand canonical ensemble, the statistical structure of counting Green's function becomes :

$$G_0(\tau,\tau') = \frac{\mathrm{Tr}[e^{-\beta H_0}\hat{a}^\dagger(\tau)\hat{a}(\tau')]}{\mathrm{Tr}[e^{-\beta H_0}]} = \frac{\mathrm{Tr}[e^{-\beta H_0}e^{iH_0\tau}\hat{a}^\dagger e^{-H_0(\tau-\tau')}\hat{a}e^{-iH_0\tau'}]}{\mathrm{Tr}[e^{-\beta H_0}]}$$

This is the very basic form of Green's function in imaginary time corresponding to the diagram structure. Similar case with the creation and annihilation operator, the time dependency of external potential term Hamiltonian :

$$H'(\tau) = e^{H_0\tau} H e^{-H_0\tau}$$

If we consider the effect of interaction, we can adopt full Green's function includes the interaction operator $U(\tau)$,

$$G(\tau,\tau') = \frac{\mathrm{Tr}[e^{-\beta H_0}U(\beta)\hat{a}^\dagger(\tau)\hat{a}(\tau')]}{\mathrm{Tr}[e^{-\beta H_0}U(\beta)]} = \frac{\mathrm{Tr}[e^{-\beta H}U(\tau)\hat{a}^\dagger U(\tau'-\tau)\hat{a}U(\tau')]}{\mathrm{Tr}[e^{-\beta H}]}$$

where interaction operator is :

$$U(\tau) = \mathcal{T}e^{-\int_0^\tau d\tau' H_{int}(\tau')}$$

In the imaginary time interval $[\tau, 0]$, full Green's function in the expectation form can be represented ,

$$G(\tau,0) = \frac{\mathrm{Tr}[e^{-\beta H_0}U(\beta)\hat{a}^\dagger(\tau)\hat{a}]}{\mathrm{Tr}[e^{-\beta H_0}U(\beta)]} = \frac{\langle U(\beta)\hat{a}^\dagger(\tau)\hat{a}\rangle_0}{\langle U(\beta)\rangle_0}$$

The under index 0 of angled bracket indicates its expectation value was calculated in H_{loc} frame. If the interaction hamiltonian is in the form of multiplication of pair of annihilation-creation operator $\hat{a}^\dagger$ and $\hat{a}$, we can adjust Wick's theorem to expand the each of the expectation value in denominator and numerator into a multiplication of Green's functions in each time interval. For example, if

$$H_{int} = V_{ijkl}\hat{a}_i^\dagger\hat{a}_j^\dagger\hat{a}_k\hat{a}_l$$

Here the term V_{ijkl} is the scalar value (or function) that represents the interaction effect. then

$$\begin{aligned}\langle U(\beta)\rangle_0 &\sim 1 - \frac{1}{2}V_{ijkl}\int_0^\beta d\tau_1\langle\mathcal{T}\hat{a}_i^\dagger(\tau_1)\hat{a}_j^\dagger(\tau_1)\hat{a}_k(\tau_1)\hat{a}_l(\tau_1)\rangle + ... \\ &= 1 - \frac{1}{2}\int_0^\beta \langle\mathcal{T}\hat{a}_i^\dagger(\tau_1)\hat{a}_j^\dagger(\tau_1)\rangle\langle\mathcal{T}\hat{a}_k(\tau_1)\hat{a}_l(\tau_1)\rangle + ...\end{aligned}$$

This kind of form of expansion represents the expectation value of the interaction operator in the multiplication of diagrams.

2.3 Diagrammatic Hybridization expansion in strong coupling case

In this section, we will introduce the hybridization method based on the former discussion. We follow the notation of []. In the case of the impurity model with a bosonic bath, the action of the full system can be written into :

$$S = \int_0^\beta d\tau H_{loc}(\tau) + \int_0^\beta d\tau d\tau' P(\tau)\mathcal{W}(\tau - \tau')P(\tau')$$

Here, $H_{loc}$is the Hamiltonian of the local system, indicating the impurity that we want to focus on. The terms P and $\mathcal{W}$ correspond with the Hermitian operator and the interaction term can be assumed as $\mathcal{W}$ in the previous section. As we treat the interaction Hamiltonian in the previous section, the term $\mathcal{W}$ includes the scalar factor to represent the effect of the bosonic bath. With similar procedure with the diagram expansion, We can calculate the expectation value of observable using the partition function Z,

$$Z = \text{tr}\left[T_\tau e^{-S}\right]$$

And in the given frame, the expectation value of observable O can be measured as

$$\langle T_\tau O(\tau)\rangle = \frac{\text{tr}[T_\tau e^{-S} O(\tau)]}{\text{tr}[T_\tau e^{-S}]}$$

Which has a similiar form of the full Green's function. Similiar with it's diagrammatic expansion, the partition function can be expanded to

$$Z = \sum_{n=0}^{\infty} Z^{(n)}$$

With it's full expression,

$$\begin{aligned} Z = &\sum_{n=0}^{\infty} \frac{(-1)^n}{n!} \sum_{\pi \in S_{2n}} \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \cdots \int_0^{\tau_{2n-1}} d\tau_{2n} \text{tr}\left[U(\beta - \tau_1)PU(\tau_1 - \tau_2)P \cdots PU(\tau_{2n})\right] \\ &\times \mathcal{W}(\tau_{\pi(1)} - \tau_{\pi(2)}) \cdots \mathcal{W}(\tau_{\pi(2n-1)} - \tau_{\pi(2n)}) \end{aligned}$$

Here, $\pi(m)$ refers that there are permutations and S_{2n} is a set of available permutations for time ordering. The main Idea of the expansion is, that we only consider the certain "partial set" of full permutation, which corresponds with the non-crossing or crossing connected diagrams. Define the new $\bar{\mathcal{W}}$ function,

$$\bar{\mathcal{W}} = (\mathcal{W}(\tau) + \mathcal{W}(-\tau))$$

And consider the permutation which only restricted on the case of ,

$$\pi(2j-1) < \pi(2j)$$

We can consider the normalization of the partition function using the chemical potential λ in a grand canonical ensemble. With the normalization and the above two rules, we can construct the new formula about the expansion of the partition function, which can be rewritten as :

$$\mathcal{G}(\tau) = \sum_{n=0}^{\infty} \mathcal{G}^{(n)}(\tau) = e^{\tau\lambda} Z$$

which is :

$$\begin{aligned}
\mathcal{G}^{(0)}(\tau) &= \mathcal{G}_0(\tau) \\
\mathcal{G}^{(1)}(\tau) &= \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \Big[\mathcal{G}_i(\beta-\tau) P \mathcal{G}_i(\tau_1-\tau_2) P \mathcal{G}_i(\tau_2)\Big] \bar{\mathcal{W}}(\tau_1-\tau_2) \\
\mathcal{G}^{(2)}(\tau) &= \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \int_0^{\tau_2} d\tau_3 \int_0^{\tau_3} d\tau_4 \Big[\mathcal{G}_i(\beta-\tau) P \mathcal{G}_i(\tau_1-\tau_2) P \mathcal{G}_i(\tau_2-\tau_3) P \mathcal{G}_i(\tau_3-\tau_4) P \mathcal{G}_i(\tau_4)\Big] \\
&\quad \times \Big(\bar{\mathcal{W}}(\tau_1-\tau_2)\bar{\mathcal{W}}(\tau_3-\tau_4) + \bar{\mathcal{W}}(\tau_1-\tau_4)\bar{\mathcal{W}}(\tau_2-\tau_3) + \bar{\mathcal{W}}(\tau_1-\tau_3)\bar{\mathcal{W}}(\tau_2-\tau_4)\Big)
\end{aligned}$$

The term $\mathcal{G}_i(\tau)$ is a matrix correspond with $U(\tau)$. During the calculation and the diagrams earned from the previous procedure, We can collect the irreducible diagrams that cannot be separated into a number of single propagators. These diagrams are defined self-energy, described as follows :

$$\Sigma(\tau) = \sum_{n=1}^{\infty} \Sigma^{(n)}(\tau)$$

and,

$$\begin{aligned}
\Sigma^{(1)}(\tau) &= \Big[P\mathcal{G}_i(\tau)P\Big]\bar{\mathcal{W}}(\tau) \\
\Sigma^{(2)}(\tau) &= \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \Big[P\mathcal{G}_i(\tau-\tau_1)P\mathcal{G}_i(\tau_1-\tau_2)P\mathcal{G}_i(\tau_2)P\Big](\bar{\mathcal{W}(\tau)}\bar{\mathcal{W}}(\tau_1-\tau_2) + \bar{\mathcal{W}}(\tau-\tau_2)\bar{\mathcal{W}}(\tau_1))
\end{aligned}$$

Adopting the self-energy in normalized Partition function, we can gain the equation :

$$\begin{aligned}
\mathcal{G} =& \mathcal{G}_0(\tau) + \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \mathcal{G}_i(\beta-\tau_1)\Sigma(\tau_1-\tau_2)\mathcal{G}^{(2)}(\tau_2) \\
&+ \int_0^\beta d\tau_1 \int_0^{\tau_1} d\tau_2 \int_0^{\tau_2} d\tau_3 \int_0^{\tau_3} d\tau_4 \mathcal{G}_i(\beta-\tau_1)\Sigma(\tau_1-\tau_2)\mathcal{G}_i(\tau_2-\tau_3)\Sigma(\tau_3-\tau_4)\mathcal{G}_i + ...
\end{aligned}$$

This form of Dyson equation can be represented into diagram structure. If we define full $\mathcal{G}$ as :

$$\mathcal{G} = \langle \psi_a(\tau)\psi_b^\dagger(\tau')\rangle = \text{Tr}[e^{\beta H}\psi_a(\tau)\psi_b^\dagger(\tau')]$$

Then the diagram of the Dyson equation turns out to Figure.3. While representing the self-energy in extended diagrammatic formation, we can collect the diagrams with some featured topological structures. For instance, if we consider only the first-order self-energy expansion, the result coincides with the well-known Non-crossing approximation method. This form of equation satisfies the integral-differential form, agrees with the equation of

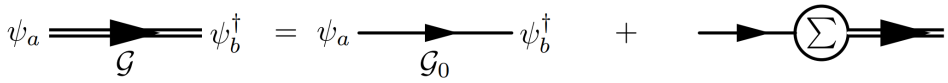

Figure 3: Diagram for Dyson equation with self energy

motion of in the quantum mechanical aspect.

$$[-\partial_\tau - H_{loc} + \lambda]\mathcal{G}(\tau) - \int_0^\tau d\tau_1 \Sigma(\tau - \tau_1)\mathcal{G}(\tau_1) = 0 \quad , \quad (\tau > 0)$$

In our study, the function which corresponds with $\mathcal{W}$ is written as $V_k(\tau)$,

$$V_k(\tau) = e^{-\omega_k \tau}\theta(-\tau)n_B(\omega_k\beta) + e^{\omega_k\tau}\theta(\tau)(1 + n_B(\omega_k\beta))$$

Consider the form of $\bar{\mathcal{W}}$,

$$V_k(\tau) = e^{-\omega_k\tau}\frac{\cosh\frac{\omega_k\beta}{2}}{\sinh\frac{\omega_k\beta}{2}}$$

Evaluation of correlation function

We can get the connected diagram from the above procedures, and the connected correlation function can be calculated as:

$$c = \frac{\delta}{\delta a(\beta)}\frac{\delta}{\delta a(\tau)}\ln(Z[a(\tau)])\Big|_{a=0}$$

3. Simulation model

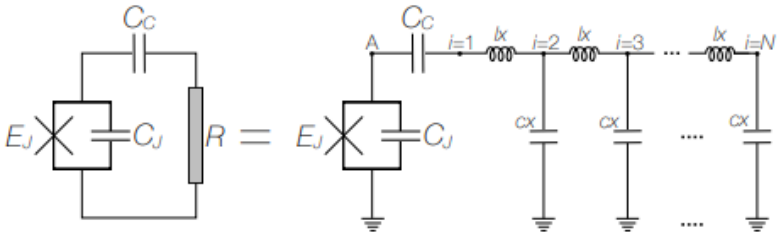

Figure 4: Picture of resistivity shunted Josephson junction connected with exterior transmission line.

We set the Resistivity shunted Josephson junction(RSJJ) circuit as an impurity model. The circuit is composed of a single Josephson junction connected to the transmission line, which acts as the resistance of the total circuit system. Each component of the circuit is mapped onto each composition of the total Hamiltonian. The given circuit Hamiltonian form is represented as follows :

$$H_{sys} = (E_c\hat{N}^2 - E_J\cos\phi)\otimes I - \hat{N}\otimes\sum_{0<k\leq K}\hbar g_k(\hat{b}_k^\dagger + \hat{b}_k) + I\otimes\sum_{0<k\leq K}\hbar\omega_k\hat{b_k}^\dagger\hat{b_k}$$

Specific description of each parts of system Hamiltonian composed by :

$$\begin{aligned} H_{loc} &= E_C\hat{N}^2 - E_J\cos\phi \\ H_{bath} &= \sum_{0<k\leq K}\hbar\omega_k\hat{b}_k^\dagger\hat{b}_k \\ H_{int} &= -\hat{N}\otimes\sum_{0<k\leq K}\hbar g_k(\hat{b}_k^\dagger + \hat{b}_k) \end{aligned}$$

Interaction effect of local(impurity) system between bath is described through g_k and the parameter $\alpha = \frac{R_Q}{R}$. We use α and $\gamma = \frac{E_J}{E_C}$ as parameters to control the criticality of the Josephson junction. We've set E_C and $\hbar$ as energy units, 1 as the value of each parameter during the calculation process. Detailed description of basic symbols is in Table. 1.

3.1 Span of H_{loc} in Josephson effect basis

To adjust the expansion method to the RSJJ Hamiltonian model, We represent the local system Hamiltonian in matrix formulation using the Josephson wave function as its basis. We adopt the macroscopic point of view to observe the superconducting effect expected in our junction system. The free-particle basis in exponential form is written as :

$$|\psi_{JJ}\rangle = \sum_{-\infty}^{\infty} c_m e^{im\phi}$$

Here, c_m and m indicates the normalization factor and wave number which analogous to energy excitations.

Symbols	Formula	Physical quantities
ϕ		Phase of Josephson junction
$\hat{N}$	$-i\frac{\partial}{\partial\phi}$	Charge operator
ω_k	$vk = \frac{vm\pi}{L}$	Bath frequency
$\hat{b_k}^\dagger, \hat{b_k}$		Bosonic creation, annihilation operator
R_Q	$\frac{h}{4e^2}$	Quantum resistance of junction
R		Resistance of bath
E_C	$\frac{(2e)^2}{2C_J}$	Josephson junction charging energy
E_J		Josephson coupling energy
g_k	$\sqrt{\frac{2\pi v}{\alpha L}\frac{\omega_k}{1+(\frac{\nu\omega_k}{W})^2}}$	Coupling with bath in mode k
ν	$\frac{\pi}{\alpha\epsilon_C}$	
ϵ_C	$\frac{E_C}{\hbar W}$	

Table 1: Table 1.Basic symbols used for describing Hamiltonian.

In trigonometric form, it turns out :

$$|\psi_{JJ}\rangle = \sum_{m=0}^{\infty} a_m \cos m\phi + b_m \sin m\phi$$

Where a_m and b_m are the normalization factor corresponds to even and odd trigonal function. separating in odd and even function,

$$\text{even part of the basis} : \sum_{m=0}^{\infty} a_m \cos m\phi$$
$$\text{odd part of the basis} : \sum_{n=1}^{\infty} b_n \sin n\phi$$

In a Hamiltonian of a system with arbitrary dimensions, the eigenvector written in the form of an even function in form of series expansion:

$$|\text{even}_k\rangle = \frac{a_0^{(k)}}{\sqrt{2\pi}} + \sum_{n=1} \frac{a^{(k)}}{\sqrt{\pi}} \cos n\phi$$

Here, (k) represents the k-th split state, that is, the energy state of the even function basis split by external perturbation in the k-th excited state. In the same way, the eigenvector written in the form of an odd function is as follows:

$$|\text{odd}_k\rangle = \sum_{m=1} \frac{b_m^{(k)}}{\sqrt{\pi}} \sin m\phi$$

And its matrix form is :

$$|\text{even}_k\rangle = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_m \end{pmatrix} \quad , \quad |\text{odd}_k\rangle = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{pmatrix}$$

We can span the target Hamiltonian in given basis.Considering the differential form of local system Hamiltonian with using above result,

$$H_{loc} = -\frac{\partial^2}{\partial\phi^2} - \gamma\cos\phi$$

$$H_{loc}\,|\psi_{JJ}\rangle = H_{loc}(|\text{even}_k\rangle + |\text{odd}_k\rangle)$$

$$\Big(-\frac{\partial^2}{\partial\phi^2} - \gamma\cos\phi\Big)\Big(\sum_{m=0}^{\infty} a_m\cos m\phi + \sum_{n=1}^{\infty} b_n\sin n\phi\Big)$$
$$= -\sum_{m=0}^{\infty}(ma_m\cos m\phi + \gamma\cos\phi\cos m\phi) - \sum_{n=1}^{\infty}(nb_n\sin n\phi + \gamma\cos\phi\sin n\phi)$$

In matrix representation,

$$H_{\text{loc even}} = \begin{pmatrix} \alpha_0 & -\frac{\gamma}{\sqrt{2}} & 0 & \cdots & \\ -\frac{\gamma}{\sqrt{2}} & \alpha_1 & -\frac{\gamma}{2} & 0 & \cdots \\ & & & & \\ 0 & -\frac{\gamma}{2} & \alpha_2 & & \\ \vdots & & & \ddots & \end{pmatrix}$$

$$H_{\text{loc odd}} = \begin{pmatrix} \beta_1 & -\frac{\gamma}{2} & 0 & \cdots & \\ -\frac{\gamma}{2} & \beta_2 & -\frac{\gamma}{2} & 0 & \cdots \\ & & & & \\ 0 & -\frac{\gamma}{2} & \beta_3 & & \\ \vdots & & & \ddots & \end{pmatrix}$$

To study how system changes depends on coupling parameters and finite temperature conditions, we measure the total RSJJ Hamiltonian under local system basis engaged from diagonalization of the $H_{\text{loc even}}$ and $H_{\text{loc odd}}$.

3.2 Matrix form of $\hat{N}$ in local basis

The charge operator $\hat{N}$ pursues polarization operator in our expansion model, endows coupling effect between local system and bath system. Exploiting local basis from previous procedure, We can write $\hat{N}$ in aspect of local system view.

2-level case

Now, we can span the charge operator in eigenvectors of $H_{\text{loc even}}$ and $H_{\text{loc odd}}$. In 2-level case, we can use total lowest three states as basis vector,

$$
\begin{aligned}
|\text{gs}\rangle &= \frac{a_0^{(0)}}{\sqrt{2\pi}} + \sum_{n=1} \frac{a_n^{(0)}}{\sqrt{\pi}} \cos n\phi = \frac{a_0}{\sqrt{2\pi}} + \frac{a_1}{\sqrt{\pi}} \cos\phi + \frac{a_2}{\sqrt{\pi}} \cos 2\phi + \cdots \\
|\text{1st}\rangle &= \sum_{m=1} \frac{b_m^{(1)}}{\sqrt{\pi}} \sin m\phi = \frac{b_1}{\sqrt{\pi}} \sin\phi + \frac{b_2}{\sqrt{\pi}} \sin 2\phi + \cdots \\
|\text{2nd}\rangle &= \frac{a_0^{(1)}}{\sqrt{2\pi}} + \sum_{n=1} \frac{a_n^{(1)}}{\sqrt{\pi}} \cos n\phi = \frac{a_0'}{\sqrt{2\pi}} + \frac{a_1'}{\sqrt{\pi}} \cos\phi + \frac{a_2'}{\sqrt{\pi}} \cos 2\phi + \cdots
\end{aligned}
$$

Here we note the upper indices with round brace to indicate the energy excitation of local system. Then matrix form of $\hat{N}$ becomes :

$$
\hat{N} = \begin{pmatrix}
\langle \text{gs}| - i\frac{\partial}{\partial\phi} |\text{gs}\rangle & \langle \text{gs}| - i\frac{\partial}{\partial\phi} |\text{1st}\rangle & \langle \text{gs}| - i\frac{\partial}{\partial\phi} |\text{2nd}\rangle \\
\langle \text{1st}| - i\frac{\partial}{\partial\phi} |\text{gs}\rangle & \langle \text{1st}| - i\frac{\partial}{\partial\phi} |\text{1st}\rangle & \langle \text{1st}| - i\frac{\partial}{\partial\phi} |\text{2nd}\rangle \\
\langle \text{2nd}| - i\frac{\partial}{\partial\phi} |\text{gs}\rangle & \langle \text{2nd}| - i\frac{\partial}{\partial\phi} |\text{1st}\rangle & \langle \text{2nd}| - i\frac{\partial}{\partial\phi} |\text{2nd}\rangle
\end{pmatrix}
$$

$$
= i \begin{pmatrix}
0 & \sum_{n=1} n a_n^{(0)} b_n^{(1)} & 0 \\
-\sum_{n=1} n a_n^{(0)} b_n^{(1)} & 0 & -\sum_{n=1} n a_n^{(1)} b_n^{(1)} \\
0 & \sum_{n=1} n a_n^{(1)} b_n^{(1)} & 0
\end{pmatrix}
$$

Multilevel case

For a 2-level system (without considering energy level splitting; the energy level needs to be split by external perturbation to become a 3-level Hamiltonian) or higher, the process of expressing the order parameter in matrix form for the Hamiltonian H_{loc} of the system under consideration involves the following steps. Now based on the

above discussion, the extended form of charge operator in higher dimension is :

$$\hat{N} = \begin{pmatrix} \langle \text{gs}| - i\frac{\partial}{\partial\phi}|\text{gs}\rangle & \langle \text{gs}| - i\frac{\partial}{\partial\phi}|\text{1st}\rangle & \cdots \\ \langle \text{1st}| - i\frac{\partial}{\partial\phi}|\text{gs}\rangle & \ddots & \vdots \\ \vdots & \cdots & \langle \text{nth}| - i\frac{\partial}{\partial\phi}|\text{nth}\rangle \end{pmatrix}$$

$$= i \begin{pmatrix} 0 & \sum_{n=1} n a_n^{(0)} b_n^{(1)} & \cdots \\ -\sum_{n=1} n a_n^{(0)} b_n^{(1)} & \ddots & \\ & & -\sum_{n=1} n a_n^{(n-1)} b_n^{(n)} \\ \vdots & \sum_{n=1} n a_n^{(n-1)} b_n^{(n)} & \end{pmatrix}$$

In this case, the maximum dimension of $\hat{N}$ is $2n + 1$.

3.3 Transformation of the Order Parameter into Matrix Form

The order parameter represents a set of values arranged in an ordered manner according to a specific rule for the system under consideration. In the investigation of phase transitions, we determine the current state of the system by observing changes in the order parameter. In this simulation, the cos function, which represents the phase difference of the Josephson current flowing on both sides of the Josephson junction, was set as the order parameter. A larger value of cos can be interpreted as the phase difference of the currents flowing on both sides of the junction approaching 0. This indicates a state where current flows smoothly without the influence of resistance, suggesting that the entire junction is in a conductive state.

2-level case

Since we aim to understand the system from the perspective of which represents the Josephson junction with three energy levels, we intend to expand order parameter $\cos\phi$ in simliar procedure with calculating $\hat{N}$ matrix. Adopting the form of fourier basis in the case of 2-level in the representing $\hat{N}$ matrix, cos can be rewritten as a matrix operator, and

$$\hat{\cos}\,\phi = \begin{pmatrix} \langle \text{gs}|\cos\phi|\text{gs}\rangle & \langle \text{gs}|\cos\phi|\text{1st}\rangle & \langle \text{gs}|\cos\phi|\text{2nd}\rangle \\ \langle \text{1st}|\cos\phi|\text{gs}\rangle & \langle \text{1st}|\cos\phi|\text{1st}\rangle & \langle \text{1st}|\cos\phi|\text{2nd}\rangle \\ \langle \text{2nd}|\cos\phi|\text{gs}\rangle & \langle \text{2nd}|\cos\phi|\text{1st}\rangle & \langle \text{2nd}|\cos\phi|\text{2nd}\rangle \end{pmatrix}$$

When obtaining eigenvectors for the 2-level truncated 3*3 H_{loc} matrix, the matrix form of $\cos\phi$ is expressed as follows.

$$\hat{\cos}\,\phi = \begin{pmatrix} \frac{2}{\sqrt{2}} a_0 a_1 + a_1 a_2 & 0 & \frac{1}{\sqrt{2}}(a_1 a_0' + a_0 a_1') + \frac{1}{2}(a_1 a_2' + a_2 a_1') \\ 0 & b_1 b_2 & 0 \\ \frac{1}{\sqrt{2}}(a_1 a_0' + a_0 a_1') + \frac{1}{2}(a_1 a_2' + a_2 a_1') & 0 & \frac{2}{\sqrt{2}} a_0' a_1' + a_1' a_2' \end{pmatrix}$$

Multilevel case

For a 2-level system (without considering energy level splitting; the energy level needs to be split by external perturbation to become a 3-level Hamiltonian) or higher, the process of expressing the order parameter in matrix form for the Hamiltonian H_{loc} of the system under consideration involves the following steps. First, the calculation of the even function form of the order parameter $\cos\phi$ is as follows:

$$\cos\phi\,|\text{even}_k\rangle = \frac{a_0^{(k)}}{\sqrt{2\pi}}\cos\phi + \sum_{n=1}\frac{a^{(k)}}{\sqrt{\pi}}\cos\phi\cos n\phi$$

And Final innerproduct result is:

$$\langle\text{even}_l|\cos\phi\,|\text{even}_k\rangle = \int_0^{2\pi}\left(\frac{a_0^{(m)}}{\sqrt{2\pi}} + \sum_{m=1}\frac{a^{(m)}}{\sqrt{\pi}}\cos m\phi\right)\left(\frac{a_0^{(n)}}{\sqrt{2\pi}}\cos\phi + \sum_{n=1}\frac{a^{(n)}}{\sqrt{\pi}}\cos\phi\cos n\phi\right)$$

The calculation for the odd function form is as follows:

$$|\text{odd}\rangle = \sum_{n=1}\frac{b_n^{(k)}}{\sqrt{\pi}}\sin n\phi \quad , \quad \cos\phi\,|\text{odd}\rangle = \sum_{n=1}\frac{b_n^{(k)}}{\sqrt{\pi}}\cos\phi\sin n\phi$$

Similar with the case of even state eigenvector, innerproduct result is:

$$\langle\text{odd}_l|\cos\phi\,|\text{odd}_k\rangle = \sum_{n,m=1}\int_0^{2\pi}d\phi\left(\frac{b_n^{(k)}b_m^{(l)}}{\pi}\cos\phi\sin n\phi\sin m\phi\right)$$

$$= \begin{cases} \text{if k} \not\leq \text{l}: & \sum_{n=1}^{N}\left(\frac{\hat{b}_n^{(k)}\hat{b}_{n-1}^{(l)}}{2} + \frac{\hat{b}_n^{(k)}\hat{b}_{n+1}^{(l)}}{2}\right) \\ \text{if k} \not\geq \text{l}: & \sum_{n=1}^{N}\left(\frac{\hat{b}_{n-1}^{(k)}\hat{b}_n^{(l)}}{2} + \frac{\hat{b}_{n+1}^{(k)}\hat{b}_n^{(l)}}{2}\right) \end{cases}$$

Therefore, when the entire $\cos\phi$ is expressed in matrix form, it takes the following form:

$$\hat{\cos}\,\phi = \begin{pmatrix} \ddots & \vdots & & \\ \langle\text{even}_k|\cos\phi\,|\text{even}_k\rangle & 0 & \langle\text{even}_k|\cos\phi\,|\text{even}_{k+1}\rangle & \cdots \\ 0 & \langle\text{odd}_k|\cos\phi\,|\text{odd}_k\rangle & 0 & \langle\text{odd}_k|\cos\phi\,|\text{odd}_{k+1}\rangle \\ \langle\text{even}_{k+1}|\cos\phi\,|\text{even}_k\rangle & 0 & \langle\text{even}_{k+1}|\cos\phi\,|\text{even}_{k+1}\rangle & \\ & \vdots & & \ddots \end{pmatrix}$$

4. Numerical Solution of the Integro-differential Equation

The integro-differential equation central to the analysis of our model was solved numerically using C++ code. This section outlines the key computational techniques employed within the code.

4.1 Structure of C++ code

The structure of the C++ code designed to solve the integro-differential equation is outlined below. The C++ code for solving the integro-differential equation is structured as follows: the functions performing the calculations are declared beforehand within a header file (.hpp) using a class, and then defined within a source file (.cpp). Appropriate numerical methods were adopted for each computational step.

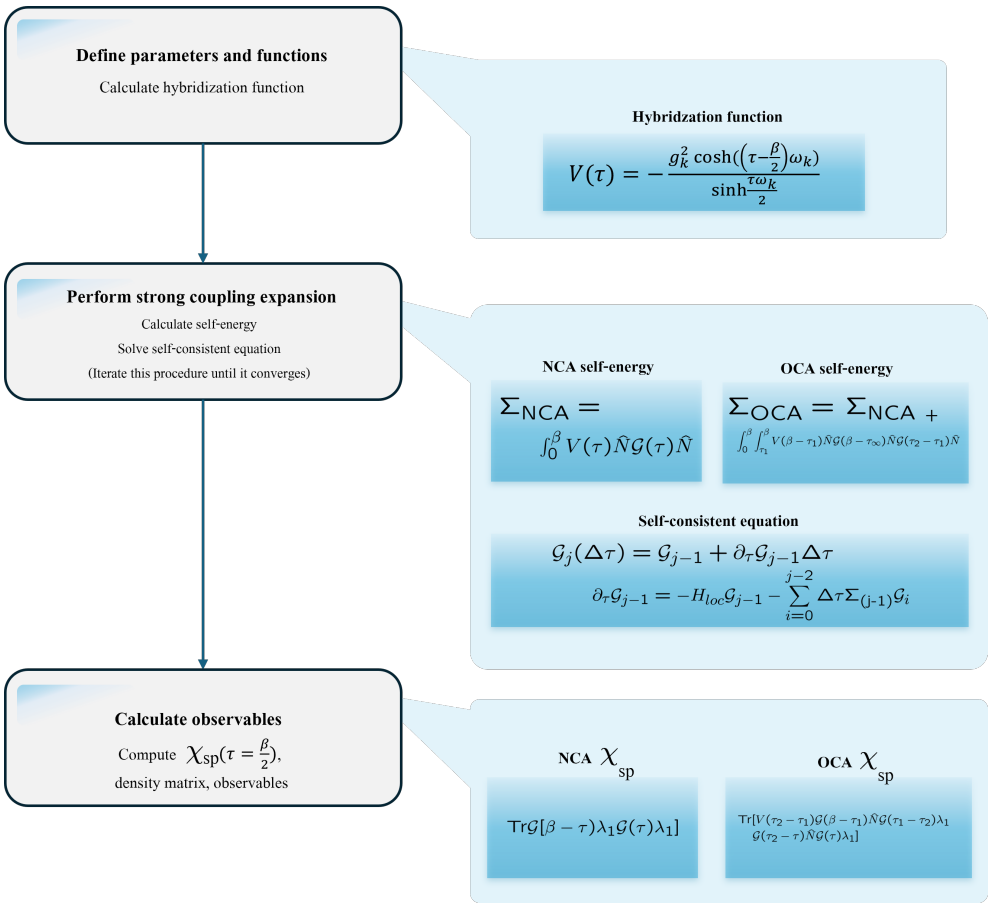

Figure 5: The flowchart of Program structure

4.2 Hybridzation function - Simpson's Rule

The hybridization function, calculated as a scalar rather than a matrix, was evaluated by considering the form of the retarded Matsubara Green's function. To compute the value of g_k, a parameter that determines the coupling between the RC circuit and the Josephson junction, Simpson's rule of integration was employed. The formula for

calculating g_k for the application of this integration method is given below:

$$g_k = \sqrt{\frac{2W}{\alpha}\left(\frac{W}{1+(\nu x)^2}\right)} \quad , \quad (x = \frac{k}{W})$$

The integration method using Simpson's method follows:

$$\int f(x)dx = \frac{b-a}{3n}\left[f(x_0) + \sum_{i=odd} 4f(x_i) + 2\sum_{i=\text{even}} f(x_i) + f(x_0) + f(x_f)\right]$$

Here, we calculate the initial value using the case of $V(\tau, k = 0)$. The part where the Simpson's rule is implemented in the code is as follows. After generating a vector array suitable for the length of the k-index to calculate V(), the integration is performed only for the k-value for g_k and ω_k in the V() expression. Using the resulting expression, the value for each interval is stored in an array of length .

4.3 Trapezoidal method

The trapezoidal method was employed for the calculation of the self-energy and the propagator. This section provides a brief overview of the implementation for calculating the propagator. The numerical integration formula using the trapezoidal method is as follows:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right]$$

To correspond to the actual numerical integration program code, the formula can be rewritten in the form of an infinite series as follows:

$$\int_a^b f(x)\,dx \approx \Delta x\left[\frac{1}{2}f(x_0) + \sum_{j=1}^{n-1} f(x_j) + \frac{1}{2}f(x_n)\right]$$

And the self-consistent equation we aim to solve is as follows:

$$\mathcal{G}_j = \mathcal{G}_{j-1} + \partial_\tau \mathcal{G}_{j-1}\Delta\tau$$

When implementing the trapezoidal method, it's crucial to follow a specific sequence of calculations:

1. First, update the value of $\partial_\tau \mathcal{G}_i$ using $\mathcal{G}_i$, which corresponds to each $f(x_i)$.
2. Next, calculate the temporary value of $\mathcal{G}'_{i+1}$ using the updated $\partial_\tau \mathcal{G}_i$.
3. Then, calculate $\partial_\tau \mathcal{G}_{i+1}$using the temporary value of $\mathcal{G}'_{i+1}$ calculated in the previous step.
4. Finally, perform the final trapezoidal integration calculation using $\frac{1}{2}(\partial_\tau \mathcal{G}_i + \partial_\tau \mathcal{G}_{i+1})$.

To implement this in the program, we introduced two arrays: P-array to store the final calculation result $\mathcal{G}_i$, and S-array to store the temporary value calculated in step 2. We also devised a method to use the value of S-array in step 3. This process is visually represented in the diagram below, where:

This is implemented in the code as follows:

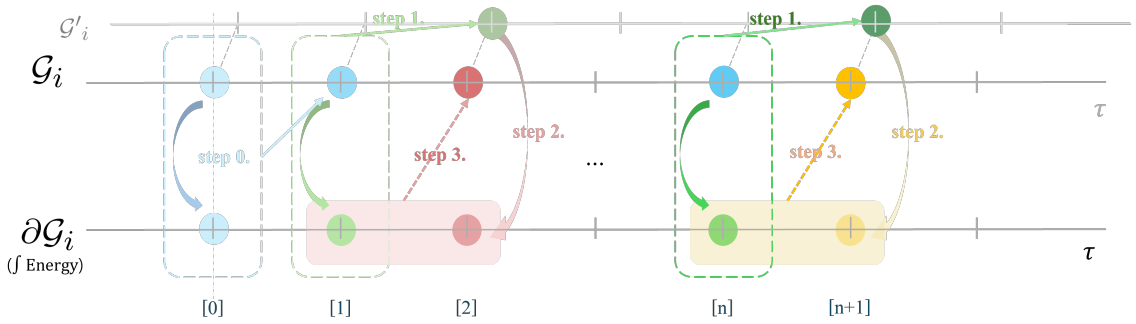

Figure 6: Image depicting the procedure of Trapezoidal process

```cpp
vector<MatrixXd> MD_OC::Propagator(const vector<MatrixXd>& sig, const MatrixXd& loc)
{
    vector<MatrixXd> P_arr(t, MatrixXd::Zero(siz,siz)); // Declaration of an array to store the final calculation result
    vector<MatrixXd> S_arr(t, MatrixXd::Zero(siz,siz)); // Declaration of an array to store temporary results during calculation

    P_arr[0] = MatrixXd::Identity(siz,siz);
    S_arr[0] = MatrixXd::Identity(siz,siz);

    MatrixXd sig_form = MatrixXd::Zero(siz,siz); // A matrix to temporarily store the result of process 1.
    MatrixXd sig_late = MatrixXd::Zero(siz,siz); // A matrix to temporarily store the result of process 3.


    for (int i = 1; i < t; i++)
    {
        P_arr[1] = P_arr[0];
        sig_late = 0.5 * Delta_t * (0.5 * Delta_t * (sig[1] * P_arr[0] + sig[0] * (P_arr[0] + Delta_t * P_arr[0])));
        P_arr[1] = P_arr[0] - 0.5 * Delta_t * loc * (2 * P_arr[0] + Delta_t * P_arr[0]) + sig_late;
        S_arr[1] = P_arr[1];

        if (i > 1)
        {
            sig_form = round_propagator_ite(loc, sig, P_arr, i - 1, 0);
            S_arr[i] = P_arr[i - 1] + Delta_t * sig_form; // Calculation process of 2. is performed in this part.

            sig_late = 0.5 * Delta_t * (round_propagator_ite(loc, sig, P_arr, i - 1, 1) + round_propagator_ite(loc, sig, S_arr, i, 1)); // This part is the result of performing trapezoidal calculation for the integral calculation in the round propagator.
            P_arr[i] = P_arr[i - 1] - 0.5 * Delta_t * loc * (2 * P_arr[i - 1] + Delta_t * sig_form) + sig_late; // This part is the result of performing trapezoidal calculation for
```

```
     the integral calculation in the round propagator.

        }
    }

    return P_arr;
}
```

Listing 1: Trapezoidal calculation code

4.4 Iteration truncation - Relative entropy

The integro-differential equation we target to solve has a recursive structure, which means that a single calculation is not enough to get a complete result. The obtained array of $G(\tau)$ in the previous iteration is to be used as an initial condition for multiple iterative calculations until the result converges sufficiently. To truncate the iteration of calculations in flexibility, the concept of relative entropy was introduced into the code. The formula for calculating relative entropy is as follows:

$$D(p||q) = \sum_{x \in \mathcal{X}} p(x) \ln \frac{p(x)}{q(x)}$$

In the case of $\mathcal{G}(\tau = \beta)$ in the interval $[0, \beta]$, The Green's function is represented by the same formula as the partition function of a thermal equilibrium state in the grand canonical ensemble. In this case, the partition function plays the same role as the density operator, which represents the mixed state of the system being investigated from a quantum mechanical perspective. In quantum mechanics, each diagonal element of the density operator represents the probability of observing the corresponding energy level in the system. This allows us to calculate the probability distribution in the final stage. The code was designed to terminate the calculation of G() when the distance between the probability distributions obtained at each calculation step becomes sufficiently small, indicating that the result has converged.

```
// Function that returns the value of \mathcal{G}(\tau=\beta)
vector<double> MD_OC::temp_itemin(vector<MatrixXd> &arrr, double minpo, int size)
{
    vector<double> dist_return(size,0);
    for (int i = 0 ; i < size; i++){
        dist_return[i] = arrr[t-1](i,i); // Value of \mathcal{G} at \tau = \beta
    }
    return dist_return;
}

// Function that processes recursive calculation.
vector<MatrixXd> MD_OC::Iteration(const int& n)
{
    // ~ Calculation part
    ////////////////////////////////////////////////////////////
    double temp_minpoin;
```

```
    vector<vector<double> > temp_itemi(2,vector<double>(siz,0)); // Arrays to store the
    results of the previous and current iteration. siz represents the dimension of the
    calculated square matrix.
    double RELA_ENTROPY;
    ////////////////////////////////////////////////////////////////

    for (int i = 0; i <= n; i++){
        // ~ Calculation part
            ////////////////////////////////////////////////////////////////////////////////

            temp_minpoin = //beta value;

            ////////////////////////////////////////////////////////////////////////////////
       // ~ Calculation part
           ////////////////////////////////////////////////////////////////////////////////

            temp_itemi[(i-1)%2] = temp_itemin(Prop,temp_minpoin,siz); // temporary store
    previous iteration data
            RELA_ENTROPY = 0;

            ////////////////////////////////////////////////////////////////////////////////
      // ~ Calculation part
            ////////////////////////////////////////////////////////////////////////////////

            temp_itemi[i%2] = temp_itemin(Prop,temp_minpoin,siz); // temporary store present
    iteration data

            // Relative entropy calculation

            for (int j = 0; j < siz; j++){
                RELA_ENTROPY += temp_itemi[i%2][j] * log(temp_itemi[i%2][j]/temp_itemi[(i-1)
    %2][j]);
            }


            if (i > 1){
                cout << "\t""\t" << i << " th Iteration stop value : " << fabs(RELA_ENTROPY)
    << endl;
                if (fabs(RELA_ENTROPY) < 0.00001){
                    break; // If the distance between the probability distributions of the
    previous and current steps is less than 0.00001, the calculation is terminated.
                }
            }
            ////////////////////////////////////////////////////////////////////////////////
        }
    return //(Calculation result);
}
```

Listing 2: Iteration truncation code

4.5 Implementation of T-matrix

The T-matrix was first introduced in the process of representing the OCA self-energy formula in diagrammatic form. Implementing this in a computational setting leads to a reduction in the calculation time of the approximation method. The basic idea is to pre-calculate the product of the propagator and the N matrix, store it in matrix form, and then use the values from the matrix elements as needed during the self-energy calculation.

Correlation function calculation

Inspired by the T-matrix approach, a similar method was adopted for calculating the correlation function,χ_{sp}. In the case of OCA, a total of 8 matrix multiplications are required, and considering the array indices, the calculation process occurs 4 times. This can be simply represented as follows:

$$\chi_{sp} = V_{mn}\mathrm{Tr}(\hat{\mathcal{G}}_{k-m-1} * \hat{N} * \hat{\mathcal{G}}_{m-i} * \hat{\lambda}_1 * \hat{\mathcal{G}}_{i-n} * \hat{N} * \hat{\mathcal{G}}_n * \hat{\lambda}_1)$$

If the part corresponding to $\hat{N} * \hat{\mathcal{G}}_i * \hat{N} * \hat{\mathcal{G}}_j * \hat{\lambda}_1$ is pre-computed and stored in matrix form for later use in the calculation, the eight matrix multiplications are reduced to a single matrix multiplication.

$$\hat{T}_{chi} = \hat{\mathcal{G}}_{i'} * \hat{N} * \hat{\mathcal{G}}_{j'} * \hat{\lambda}_1$$
$$\chi_{sp} = \mathrm{Tr}(\hat{T}_{chi} * \hat{T}'_{chi})$$

χ_{sp} T-matrix $\hat{T}_{chi}$: k=3 case

The code for the general formula to calculate χ_{sp} before introducing the T-matrix is as follows:

```
for (int i=0; i<k; i++)
{
    MatrixXd Stmp = MatrixXd::Zero(3,3);
      for (int n=0; n<=i; n++) for (int m=i; m<k; m++)
      {
         Stmp += V[m-n] * Prop[k-m-1] * N * Prop[m-i] * GELL
                   * Prop[i-n] * N * Prop[n] * GELL;
      }
      OCA_chi_array0[i] = pow(Delta_t,2)*Stmp.trace();
}
```

Listing 3: Full One-crossing Approximation implementation code

In the above equation, for the case of k=3, the number of calculation cases according to n, m, and i is as follows: Here, we can see that the total number of calculation cases can be represented by selecting and arranging two numbers from the three numbers 2, 1, and 0. To achieve this, we can introduce the following matrix: For example, if we want to represent all the possible cases as shown above, using the matrix above and calculating as below, we obtain all possible calculation cases.

$$\hat{G}_{k-m-1} * \hat{N} * \hat{G}_{m-i} * \hat{\lambda}_1 * \hat{G}_{i-n} * \hat{N} * \hat{G}_n * \hat{\lambda}_1$$

(K=3)

$_3P_2$

i=0 → n=0 → m=0	**[2][0][0][0]**
m=1	**[1][1][0][0]**
m=2	**[0][2][0][0]**
i=1 → n=0 → m=1	**[1][0][1][0]**
→ m=2	**[0][1][1][0]**
→ n=1 → m=1	**[1][0][1][0]**
i=2 → n=1 → m=2	**[0][1][0][1]**
→ n=0 → m=2	**[0][0][2][0]**
→ n=1 → m=2	**[0][0][1][1]**
→ n=2 → m=2	**[0][0][0][2]**

Figure 7: Possible calculation cases in k=3

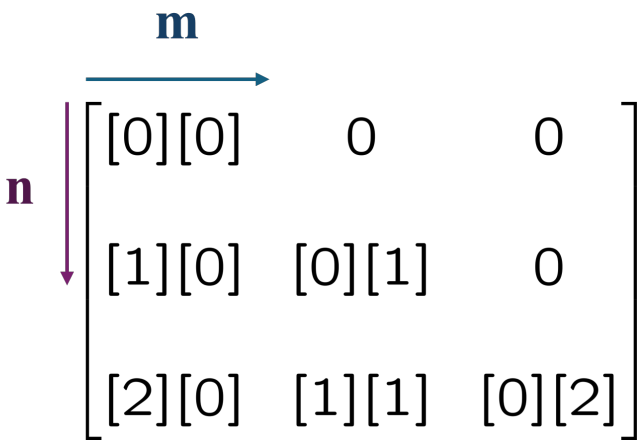

Figure 8: The Structure of correlation-matrix

$\hat{T}_{chi}(i') * \hat{T}_{chi}(j')$

(K=3)

(i', j')	(i', j')	
(2,0)	(0,0)	**[2][0][0][0]**
(2,1)	(0,0)	**[1][1][0][0]**
(2,2)	(0,0)	**[0][2][0][0]**
(1,0)	(1,0)	**[1][0][1][0]**
(1,1)	(1,0)	**[0][1][1][0]**
(1,1)	(1,0)	**[1][0][1][0]**
(1,0)	(1,0)	**[0][1][0][1]**
(0,0)	(2,0)	**[0][0][2][0]**
(0,0)	(2,1)	**[0][0][1][1]**
(0,0)	(2,2)	**[0][0][0][2]**

$$\begin{bmatrix} [2][0] & [1][0] & [0][0] \\ [1][1] & [0][1] & 0 \\ [0][2] & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} [0][0] & 0 & 0 \\ [1][0] & [0][1] & 0 \\ [2][0] & [1][1] & [0][2] \end{bmatrix}$$

Figure 9: Calculation using correlation-matrix