matsubara Documentation

Release 0.1

Shahnawaz Ahmed, Neill Lambert

Aug 28, 2019

Contents

A QuTiP plugin to study the spin-boson model in the Ultra-Strong Coupling limit at zero temperature. We provide the code to reproduce the results in [LACN19].

CHAPTER 1

Introduction

We provide the code to reproduce the results in [LACN19] using QuTiP.

The code is only for the zero temperature case where the correlation function can be expressed using four exponents - two non Matsubara contributions and a bi-exponential fitting of the infinite Matsubara sum. We will be extending it to a finite temperature case soon.

A special *matsubara.heom.HeomUB* class is provided to implement the Hierarchical Equations of Motion (HEOM) method adapted for the underdamped Brownian motion spectral density. We also implement the Reaction Coordinate (RC) and pseudomode method to tackle the same problem and show how a flat residual bath assumption (which is usually employed for the RC) gives the same results as a HEOM evolution without the Matsubara terms.

The results predict the emission of virtual photons from the ground state which is unphysical. Adding the Matsubara terms with a biexponential fitting of the infinite sum recovers back the detailed balance in the system and gives the correct ground state. In the RC method, this amounts to assuming an Ohmic residual bath. Similarly, we can use three modes (one non Matsubara) and two Matsubara modes to recover the correct result using the pseudomodes although this requires the evolution of a non-Hermitian Hamiltonian.

We also provide some comparision with the analytical solution of a pure dephasing model to quantify the error in the steady-state populations.

Lastly, we show how to calculate the virtual photon populations (bath occupations) from all the methods. While for the RC and pseudomode it is a straight-forward expectation calculation using the bath operators, in the HEOM formulation it is not so direct. In the HEOM method, we recover the bath occupation from the Auxiliary Density Operators (ADOs) of the evolution.

The spin-boson Hamiltonian

$$H = \frac{\omega_q}{2}\sigma_z + \frac{\Delta}{2}\sigma_x + \sum_k \omega_k b_k^\dagger b_k + \sigma_z \tilde{X}$$

where

$$\tilde{X} = \sum_k \frac{g_k}{\sqrt{2\omega_k}} \left(b_k + b_k^\dagger\right)$$

We focus on the the underdamped Brownian motion spectral density defined as

$$J(\omega) = \frac{\gamma\lambda^2\omega}{(\omega^2 - \omega_0^2)^2 + \gamma^2\omega^2}$$

CHAPTER 2

Installation

QuTiP is required to run the code. Also the standard python scientific computing packakges (numpy, scipy, cython, matplotlib) are necessary.

Download or clone the code from https://github.com/pyquantum/matsubara and install using the following command from your terminal from the matsubara folder:

```
python setup.py develop
```

This performs an "in-place" installation which means that everything gets installed from the local folder you downloaded and you can make changes to the code which will be immediately reflected system-wide, e.g., if you insert a print statement in some part of the code and then run any example, you can see it immediately. We hope this will allow users to change things and develop the code further. Please open a pull request if you want to add some features or find a bug.

Numpy, Scipy and QuTiP are required. Install them with conda/pip if you do not already have them using:

```
pip install cython numpy scipy
pip install qutip
```

Matplotlib is required for plotting and visualizations.

CHAPTER 3

Matsubara correlation

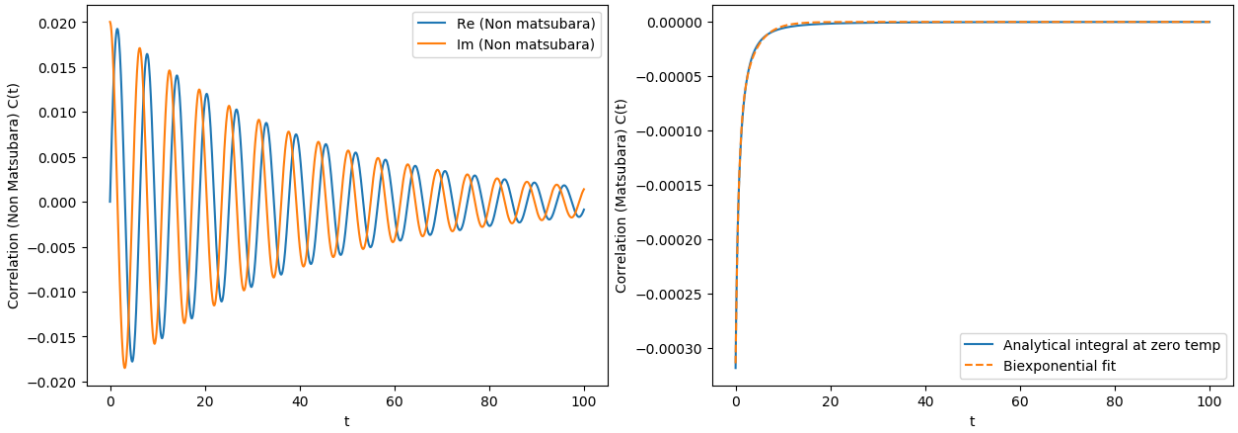
In this example, we take the underdamped Brownian motion spectral density for a bath and calculate the Matsubara and non-Matsubara expoentials to express the correlation function as a sum of four exponents.

```
import numpy as np
from matsubara.correlation import (nonmatsubara_exponents,
                                   matsubara_zero_analytical,
                                   biexp_fit, sum_of_exponentials)
import matplotlib.pyplot as plt


coup_strength, bath_broad, bath_freq = 0.2, 0.05, 1.
tlist = np.linspace(0, 100, 1000)

# Zero temperature case beta = 1/kT
beta = np.inf
ck1, vk1 = nonmatsubara_exponents(coup_strength, bath_broad, bath_freq, beta)
corr_nonmats = sum_of_exponentials(1j*ck1, vk1, tlist)

# Analytical zero temperature calculation of the Matsubara correlation
mats_data_zero = matsubara_zero_analytical(coup_strength, bath_broad,

↪bath_freq, tlist)

# Fitting a biexponential function
ck20, vk20 = biexp_fit(tlist, mats_data_zero)

print("Non matsubara coefficients: ", ck1)
print("Non matsubara frequencies:", vk1)
print("Fitted matsubara coefficients: ", ck20)
print("Fitted matsubara frequencies:", vk20)
# Plotting the fit and non Matsubara parts
corr_fit = sum_of_exponentials(ck20, vk20, tlist)

fig, ax = plt.subplots(1, 2, figsize=(15, 5))
```

(continues on next page)

(continued from previous page)

```
ax[0].plot(tlist, np.real(corr_nonmats), label="Re (Non matsubara)")
ax[0].plot(tlist, np.imag(corr_nonmats), label="Im (Non matsubara)")
ax[0].set_xlabel("t")
ax[0].set_ylabel("Correlation (Non Matsubara) C(t)")
ax[0].legend()

ax[1].plot(tlist, mats_data_zero, label="Analytical integral at zero temp")
ax[1].plot(tlist, corr_fit, "--", label="Biexponential fit")
ax[1].set_xlabel("t")
ax[1].set_ylabel("Correlation (Matsubara) C(t)")
ax[1].legend()
plt.show()
```



CHAPTER 4

Positivity constrained fitting

An example on how to impose a constraint on the fit of the correlation functions so that positivity in the final pseudo-mode model is preserved.

```
import numpy as np
from matsubara.correlation import spectrum_matsubara, spectrum_non_matsubara, spectrum
from matsubara.correlation import (sum_of_exponentials, biexp_fit,  biexp_fit_
↪constrained,
                                   bath_correlation, underdamped_brownian,
                                   nonmatsubara_exponents, matsubara_exponents,
                                   matsubara_zero_analytical, coth)
import matplotlib.pyplot as plt

bath_freq = 1.
bath_width = 1.0
coup_strength = 0.01

# Zero temperature
beta = np.inf
w = np.linspace(-3, 3, 1000)
tlist = np.linspace(0, 20, 1000)

#Fit without additional constraint:
mats_data_zero = matsubara_zero_analytical(coup_strength, bath_width,
                                           bath_freq, tlist)
ck2, vk2 = biexp_fit(tlist, mats_data_zero)


# For a given exponential we got the following contribution
# to the power spectrum

def spectrum_matsubara_approx(w, ck, vk):
    """
    Calculates the approximate Matsubara correlation spectrum
    from ck and vk.
```

(continues on next page)

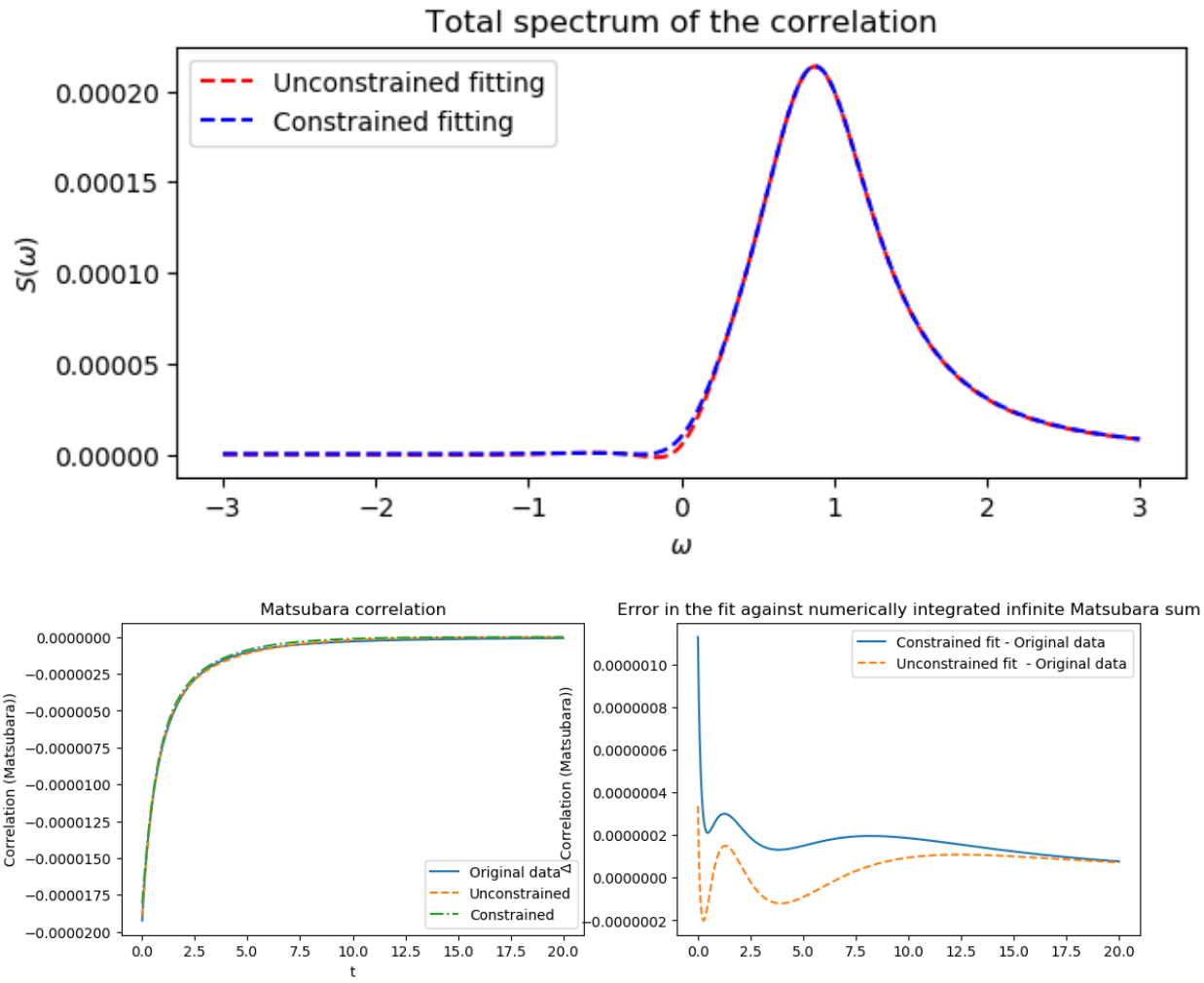
(continued from previous page)

```
    Parameters
    ==========

    w: np.ndarray
        A 1D numpy array of frequencies.

    ck: float
        The coefficient of the exponential function.

    vk: float
        The frequency of the exponential function.
    """
    return ck*2*(vk)/(w**2 + vk**2)


def spectrum_non_matsubara_approx(w, coup_strength, bath_broad, bath_freq):
    """
    Calculates the approximate non Matsubara correlation spectrum
    from the bath parameters.

    Parameters
    ==========
    w: np.ndarray
        A 1D numpy array of frequencies.

    coup_strength: float
        The coupling strength parameter.

    bath_broad: float
        A parameter characterizing the FWHM of the spectral density, i.e.,
        the bath broadening.

    bath_freq: float
        The bath frequency.
    """
    lam = coup_strength
    gamma = bath_broad
    w0 = bath_freq

    gam = gamma/2.
    om = np.sqrt(w0**2-gam**2)
    return (lam**2/(2*om))*2*(gam)/((w-om)**2+gam**2)

sm1 = spectrum_matsubara_approx(w,ck2[0],-vk2[0])
sm2 = spectrum_matsubara_approx(w,ck2[1],-vk2[1])
snm = spectrum_non_matsubara_approx(w,coup_strength,bath_width,bath_freq)

total_spectrum_unconstrained = (sm1 + sm2 + snm)

#fix the fit with  a cost function to remove negative parts of spectrum
#for a given frequency range
ck2c, vk2c = biexp_fit_constrained(tlist, mats_data_zero, w, coup_strength,
                                   bath_width, bath_freq, weight=1.)

#check spectrum again
sm1 = spectrum_matsubara_approx(w,ck2c[0],-vk2c[0])
```

(continues on next page)

(continued from previous page)

```
sm2 = spectrum_matsubara_approx(w,ck2c[1],-vk2c[1])
snm = spectrum_non_matsubara_approx(w,coup_strength,bath_width,bath_freq)
total_spectrum_constrained = (sm1 + sm2 + snm)

fig, ax1 = plt.subplots(figsize=(12, 7))
ax1.plot(w, total_spectrum_unconstrained, "--", color = "red", label="Unconstrained␣
↪fitting")
ax1.plot(w, total_spectrum_constrained, "--", color = "b", label="Constrained fitting
↪")
ax1.set_xlabel(r"$\omega$")
ax1.set_ylabel(r"$S(\omega)$")
ax1.legend()
ax1.set_title("Total spectrum of the correlation")
plt.show()

reconstructed_mats = sum_of_exponentials(ck2, vk2, tlist)
reconstructed_mats_constrained = sum_of_exponentials(ck2c, vk2c, tlist)

#compare original and new fit in terms of correlations
fig, [ax1, ax2] = plt.subplots(1, 2, figsize=(20, 7))
ax1.plot(tlist, mats_data_zero , label="Original data")
ax1.plot(tlist, reconstructed_mats,linestyle='--', label="Unconstrained")
ax1.plot(tlist, reconstructed_mats_constrained,linestyle='-.', label="Constrained")
ax1.set_ylabel("Correlation (Matsubara))")
ax1.set_xlabel("t")
ax1.legend()

ax2.plot(tlist, reconstructed_mats_constrained - mats_data_zero , linestyle='-',label=
↪"Constrained fit - Original data")
ax2.plot(tlist, reconstructed_mats - mats_data_zero , linestyle='--', label=
↪"Unconstrained fit  - Original data")
ax2.set_ylabel(r"$\Delta$ Correlation (Matsubara))")
ax1.set_xlabel("t")
ax2.legend()

ax1.set_title("Matsubara correlation")
ax2.set_title("Error in the fit from numerical intergration of infinite Matsubara␣
↪terms")
plt.show()
```


Total spectrum of the correlation
Unconstrained fitting
Constrained fitting
S(ω)
ω
Matsubara correlation
Original data
Unconstrained
Constrained
Correlation (Matsubara))
t
Error in the fit against numerically integrated infinite Matsubara sum
Constrained fit - Original data
Unconstrained fit - Original data
Δ Correlation (Matsubara))

CHAPTER 5

Hierarchical Eq. of Motion (HEOM)

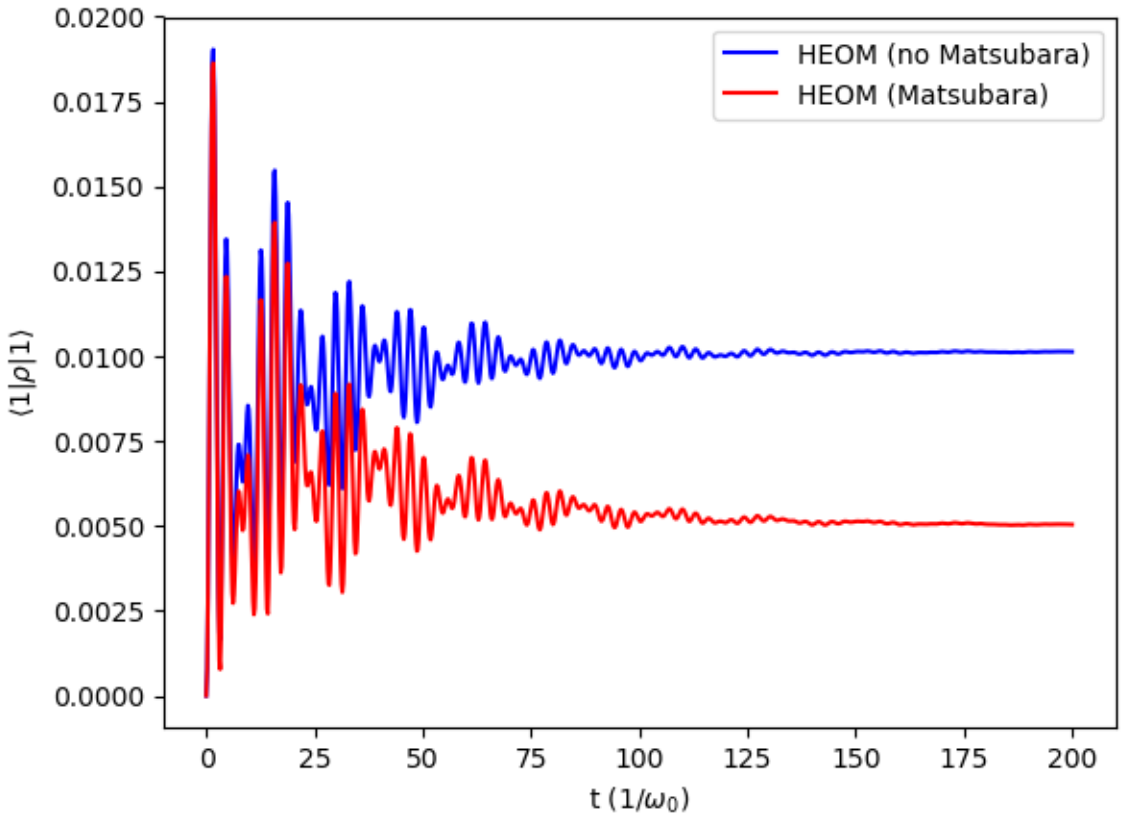
We can calculate the dynamics of the spin-boson model at zero temperature with the Hierarchical Equations of Motion (HEOM) method in the following example.

```
import numpy as np
from matsubara.correlation import (sum_of_exponentials, biexp_fit,
                                   bath_correlation, underdamped_brownian,
                                   nonmatsubara_exponents, matsubara_exponents,
                                   matsubara_zero_analytical, coth)
from qutip.operators import sigmaz, sigmax
from qutip import basis, expect
from qutip.solver import Options, Result, Stats

from matsubara.heom import HeomUB

import matplotlib.pyplot as plt


Q = sigmax()
wq = 1.
delta = 0.
coup_strength, bath_broad, bath_freq = 0.2, 0.05, 1.
tlist = np.linspace(0, 200, 1000)
Nc = 9
Hsys = 0.5 * wq * sigmaz() + 0.5 * delta * sigmax()
initial_ket = basis(2, 1)
rho0 = initial_ket*initial_ket.dag()
omega = np.sqrt(bath_freq**2 - (bath_broad/2.)**2)
options = Options(nsteps=1500, store_states=True, atol=1e-12, rtol=1e-12)

# zero temperature case, renormalized coupling strength
beta = np.inf
lam_coeff = coup_strength**2/(2*(omega))

ck1, vk1 = nonmatsubara_exponents(coup_strength, bath_broad, bath_freq, beta)
```

(continues on next page)

(continued from previous page)

```
# Ignore Matsubara
hsolver2 = HeomUB(Hsys, Q, lam_coeff, ck1, -vk1, ncut=Nc)
output2 = hsolver2.solve(rho0, tlist, options)
heom_result_no_matsubara = (np.real(expect(output2.states, sigmaz())) + 1)/2


# Add zero temperature Matsubara coefficients
mats_data_zero = matsubara_zero_analytical(coup_strength, bath_broad, bath_freq,
tlist)
ck20, vk20 = biexp_fit(tlist, mats_data_zero)
hsolver = HeomUB(Hsys, Q, lam_coeff, np.concatenate([ck1, ck20]),
                     np.concatenate([-vk1, -vk20]), ncut=Nc)
output = hsolver.solve(rho0, tlist, options)
heom_result_with_matsubara = (np.real(expect(output.states, sigmaz())) + 1)/2

plt.plot(tlist, heom_result_no_matsubara, color="b", label= r"HEOM (no Matsubara)")
plt.plot(tlist, heom_result_with_matsubara, color="r", label=r"HEOM (Matsubara)")
plt.xlabel("t ($1/\omega_0$)")
plt.ylabel(r"$\langle 1 | \rho | 1 \rangle$")
plt.legend()
plt.show()
```



CHAPTER 6

Pseudomodes

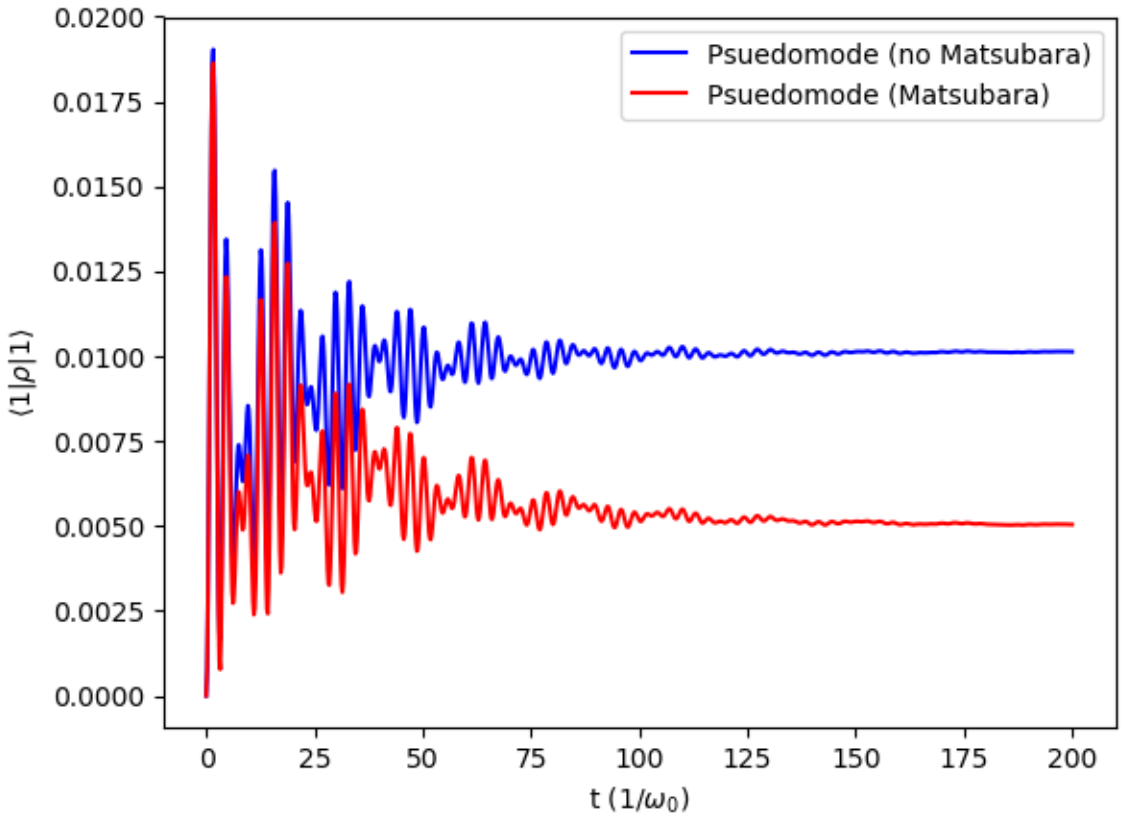
In this example we show how a pseudomode approach can be used to calculate the correct evolution using the Matsubara modes.

```
import numpy as np
from matsubara.correlation import (sum_of_exponentials, biexp_fit,
                                   bath_correlation, underdamped_brownian,
                                   nonmatsubara_exponents, matsubara_exponents,
                                   matsubara_zero_analytical, coth)
from qutip.operators import sigmaz, sigmax
from qutip import basis, expect, tensor, qeye, destroy, mesolve
from qutip.solver import Options, Result, Stats

from matsubara.heom import HeomUB

import matplotlib.pyplot as plt

Q = sigmax()
wq = 1.
delta = 0.
beta = np.inf
coup_strength, bath_broad, bath_freq = 0.2, 0.05, 1.
tlist = np.linspace(0, 200, 1000)
Ncav = 4

mats_data_zero = matsubara_zero_analytical(coup_strength, bath_broad,
↪bath_freq, tlist)
# Fitting a biexponential function
ck20, vk20 = biexp_fit(tlist, mats_data_zero)

omega = np.sqrt(bath_freq**2 - (bath_broad/2.)**2)
lam_renorm = coup_strength**2/(2*omega)

lam2 = np.sqrt(lam_renorm)
```

(continues on next page)

(continued from previous page)

```
# Construct the pseudomode operators with one extra underdamped pseudomode
sx = tensor(sigmax(), qeye(Ncav))
sm = tensor(destroy(2).dag(), qeye(Ncav))
sz = tensor(sigmaz(), qeye(Ncav))
a = tensor(qeye(2), destroy (Ncav))

Hsys = 0.5*wq*sz + 0.5*delta*sx + omega*a.dag()*a + lam2*sx*(a + a.dag())
initial_ket = basis(2, 1)
psi0=tensor(initial_ket, basis(Ncav, 0))

options = Options(nsteps=1500, store_states=True, atol=1e-13, rtol=1e-13)
c_ops = [np.sqrt(bath_broad)*a]
e_ops = [sz, ]
pseudomode_no_mats = mesolve(Hsys, psi0, tlist, c_ops, e_ops, options=options)
output = (pseudomode_no_mats.expect[0] + 1)/2

# Construct the pseudomode operators with three extra pseudomodes
# One of the added modes is the underdamped pseudomode and the two extra are
# the matsubara modes.
sx = tensor(sigmax(), qeye(Ncav), qeye(Ncav), qeye(Ncav))
sm = tensor(destroy(2).dag(), qeye(Ncav), qeye(Ncav), qeye(Ncav))
sz = tensor(sigmaz(), qeye(Ncav), qeye(Ncav), qeye(Ncav))
a = tensor(qeye(2), destroy(Ncav), qeye(Ncav), qeye(Ncav))

b = tensor(qeye(2), qeye(Ncav), destroy(Ncav), qeye(Ncav))
c = tensor(qeye(2), qeye(Ncav), qeye(Ncav), destroy(Ncav))

lam3 =1.0j*np.sqrt(-ck20[0])
lam4 =1.0j*np.sqrt(-ck20[1])

Hsys = 0.5*wq*sz + 0.5*delta*sx + omega*a.dag()*a + lam2*sx*(a + a.dag())
Hsys = Hsys + lam3*sx*(b+b.dag())
Hsys = Hsys + lam4*sx*(c + c.dag())

psi0 = tensor(initial_ket, basis(Ncav,0), basis(Ncav,0), basis(Ncav,0))
c_ops = [np.sqrt(bath_broad)*a, np.sqrt(-2*vk20[0])*b, np.sqrt(-2*vk20[1])*c]
e_ops = [sz, ]
pseudomode_with_mats = mesolve(Hsys, psi0, tlist, c_ops, e_ops, options=options)
output2 = (pseudomode_with_mats.expect[0] + 1)/2

plt.plot(tlist, output, color="b", label="Psuedomode (no Matsubara)")
plt.plot(tlist, output2, color="r", label="Psuedomode (Matsubara)")
plt.xlabel("t ($1/\omega_0$)")
plt.ylabel(r"$\langle 1 | \rho | 1 \rangle$")
plt.legend()
plt.savefig("pseudo.png", bbox_inches="tight")
plt.show()
```


Psuedomode (no Matsubara)
Psuedomode (Matsubara)
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
⟨1|ρ|1⟩
0
25
50
75
100
125
150
175
200
t (1/ω0)

CHAPTER 7

Reaction Coordinate (RC)

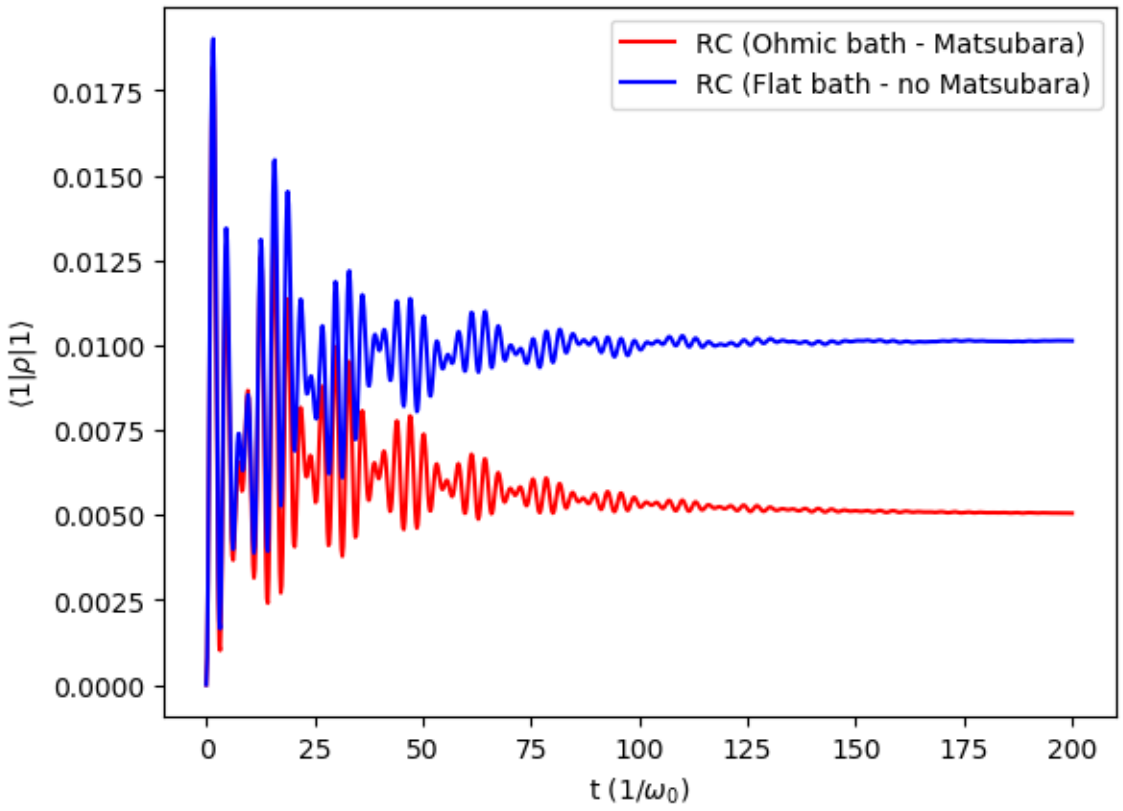
In this example, we calculate dynamics of the spin-boson model using a Reaction Coordinate (RC) approach corresponsing to two type of spectral densitities for the RC - flat bath (non Matsubara) and a Ohmic bath (Matsubara).

```
import numpy as np
from matsubara.correlation import (sum_of_exponentials, biexp_fit,
                                   bath_correlation, underdamped_brownian,
                                   nonmatsubara_exponents, matsubara_exponents,
                                   matsubara_zero_analytical, coth)
from qutip.operators import sigmaz, sigmax
from qutip import basis, expect, tensor, qeye, destroy, mesolve
from qutip.solver import Options, Result, Stats

from matsubara.heom import HeomUB

import matplotlib.pyplot as plt

Q = sigmax()
wq = 1.
delta = 0.
beta = np.inf
coup_strength, bath_broad, bath_freq = 0.2, 0.05, 1.
tlist = np.linspace(0, 200, 1000)
Ncav = 5

omega = np.sqrt(bath_freq**2 - (bath_broad/2.)**2)

# Try with omega = bath_freq
wrc = bath_freq
lam_renorm = coup_strength**2/(2*wrc)
lam2 = np.sqrt(lam_renorm)

# Construct the RC operators with a flat bath assumption for RC
sx = tensor(sigmax(), qeye(Ncav))
sm = tensor(destroy(2).dag(), qeye(Ncav))
```

(continues on next page)

(continued from previous page)

```
sz = tensor(sigmaz(), qeye(Ncav))
a = tensor(qeye(2), destroy (Ncav))
options = Options(nsteps=1500, store_states=True, atol=1e-13, rtol=1e-13)        #␣
↪Options for the solver.

Hsys = 0.5*wq*sz + 0.5*delta*sx + wrc*a.dag()*a + lam2*sx*(a + a.dag())
initial_ket = basis(2, 1)
psi0 = tensor(initial_ket, basis(Ncav,0))

#coup_strength of SD is 1/2 coup_strength used in ME
c_ops = [np.sqrt(bath_broad)*a]
e_ops = [sz, sm.dag(), a, a.dag(), a.dag()*a, a**2, a.dag()**2]
rc_flat_bath = mesolve(Hsys, psi0, tlist, c_ops, e_ops, options=options)
output = (rc_flat_bath.expect[0] + 1)/2.


# RC with a Ohmic spectral density. `c_ops` are calculated using
c_ops = []
wrc = bath_freq
groundval, gstate = Hsys.eigenstates()

bath_op_renorm = (a + a.dag())/np.sqrt(2*wrc)

for j in range(2*Ncav):
    for k in range(j, 2*Ncav):
        e_diff = groundval[k] - groundval[j]
        matrix_element = bath_op_renorm.matrix_element(gstate[j], gstate[k])
        rate = 2.*e_diff*bath_broad*(matrix_element)**2

        if np.real(rate) > 0. :
            c_ops.append(np.sqrt(rate) * gstate[j]* gstate[k].dag())

e_ops = [sz,]
rc_ohmic_bath = mesolve(Hsys, psi0, tlist, c_ops, e_ops,options=options)
output2 = (rc_ohmic_bath.expect[0] + 1)/2

plt.plot(tlist, output2, color="b", label="RC (Ohmic bath - Matsubara)")
plt.plot(tlist, output, color="r", label="RC (Flat bath - no Matsubara)")
plt.xlabel("t ($1/\omega_0$)")
plt.ylabel(r"$\langle 1 | \rho | 1 \rangle$")
plt.legend()
plt.show()
```


RC (Ohmic bath - Matsubara)
RC (Flat bath - no Matsubara)
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
⟨1|ρ|1⟩
0
25
50
75
100
125
150
175
200
t (1/ω0)

CHAPTER 8

Pure dephasing

The analytical evolution of the density matrix in the pure dephasing case is given by

$$\rho = \begin{bmatrix} \rho_{00}(0) & \rho_{01}(0)e^{-F(t)} \\ \rho_{10}(0)e^{-F(t)} & \rho_{11}(0) \end{bmatrix}$$

where F(t) is an integral which can be computed analytically in the zero temperature limit given that the correlation can be expressed as a sum of exponents. This analytical formula uses the Matsubara and non-Matsubara fitting exponents.

The exact evolution can be computed directly by a double integration of the correlation function from the underdamped Brownian motion spectral density. We can now compare both the results to see the error in the dynamics due to the fitting procedure.

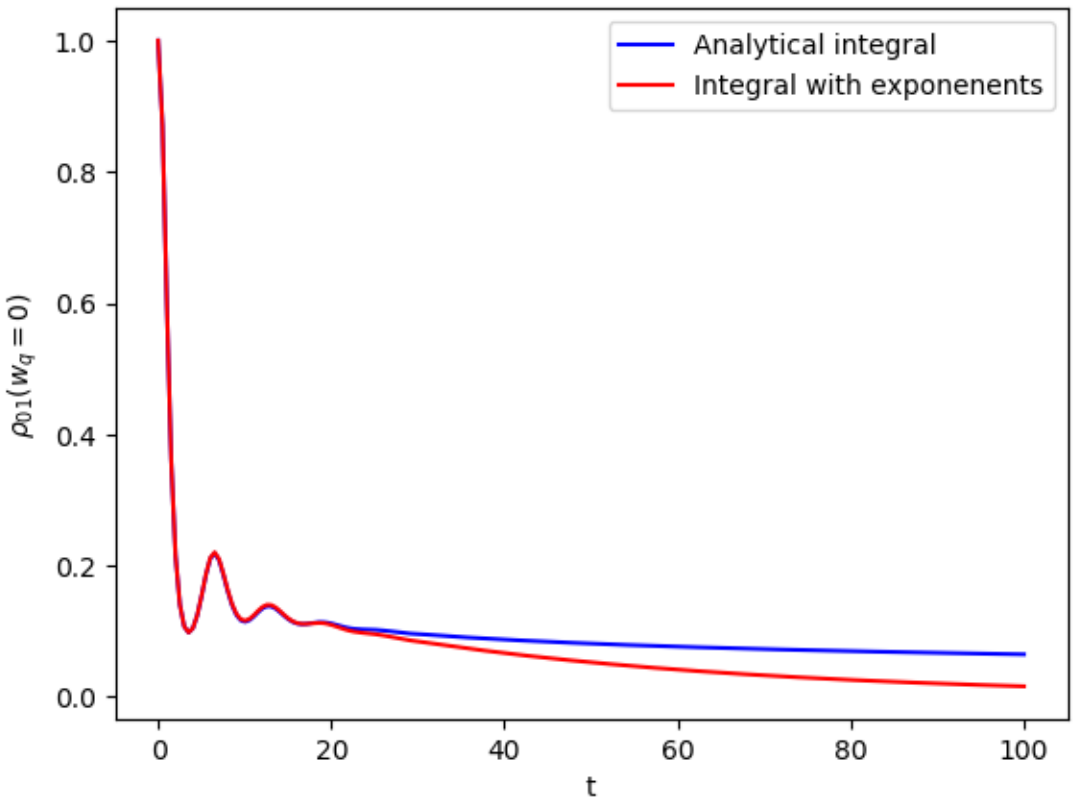
We switch to a frame rotating with the qubit frequency and can thus set it to zero to see the dynamics due to the interaction only and calculate the evolution in this example.

```
import numpy as np
from numpy.testing import (run_module_suite, assert_,
                           assert_array_almost_equal, assert_raises)

from matsubara.pure_dephasing import (pure_dephasing_integrand,
                                      pure_dephasing_evolution,
                                      pure_dephasing_evolution_analytical)
from matsubara.correlation import (biexp_fit, nonmatsubara_exponents, matsubara_
↪exponents,
                                   matsubara_zero_analytical)

import matplotlib.pyplot as plt


coup_strength, bath_broad, bath_freq = .8, .4, 1.

tlist = np.linspace(0, 100, 200)
```

(continues on next page)

(continued from previous page)

```
# Set qubit frequency to 0 to see only the dynamics due to the interaction.
wq = 0
# Zero temperature case
beta = np.inf
ck1, vk1 = nonmatsubara_exponents(coup_strength, bath_broad, bath_freq, beta)

mats_data_zero = matsubara_zero_analytical(coup_strength, bath_broad, bath_freq,␣
↪tlist)
ck20, vk20 = biexp_fit(tlist, mats_data_zero)

ck = np.concatenate([ck1, ck20])
vk = np.concatenate([vk1, vk20])

pd_analytical = pure_dephasing_evolution(tlist, coup_strength, bath_broad,
                                         bath_freq, beta, wq)
pd_numerical_fitting = pure_dephasing_evolution_analytical(tlist, wq, ck, vk)

plt.plot(tlist, pd_analytical, color="b", label="Analytical integral")
plt.plot(tlist, pd_numerical_fitting, color="r", label = "Integral with exponenents")
plt.xlabel("t")
plt.ylabel(r"$\rho_{01}$($w_q = 0$)")
plt.legend()
plt.show()
```



CHAPTER 9

Virtual photon population

In the ultrastrong coupling regime (defined as where the qubit-environment coupling is on the order of the bath frequencies), the combined system-environment "groundstate" contains a finite population of photons (and matter excitations) which in principle cannot be observed. The Matsubara terms are crucial to both calculated the correct properties of this ¨groundstate¨, and make sure that the virtual excitations are trapped in that ground state.

In case of the pseudomode we calculate the expectation value of the non-Matsubara mode bath operators. For the HEOM method, the Auxiliary Density Operators (ADOs) of the evolution contain information about the bath operators [ZLXS12], [SS17] as shown in Eq (18) of [LACN19]. Here we show how to extract them from the full HEOM evolution.

```
import numpy as np
from matsubara.correlation import (nonmatsubara_exponents,
                                   matsubara_zero_analytical,
                                   biexp_fit, sum_of_exponentials)

from qutip.operators import sigmaz, sigmax
from qutip import (basis, expect, tensor, qeye, destroy, mesolve,
                   spre, spost, liouvillian)
from qutip.solver import Options, Result, Stats

from matsubara.heom import HeomUB, get_aux_matrices
import matplotlib.pyplot as plt


# Extract virtual photon population from HEOM
wq = 1.
delta = 0.
coup_strength, bath_broad, bath_freq = 0.2, 0.05, 1.
Q = sigmax()

tlist = np.linspace(0, 200, 1000)
Nc = 9
Hsys = 0.5 * wq * sigmaz() + 0.5 * delta * sigmax()
initial_ket = basis(2, 1)
```

(continues on next page)

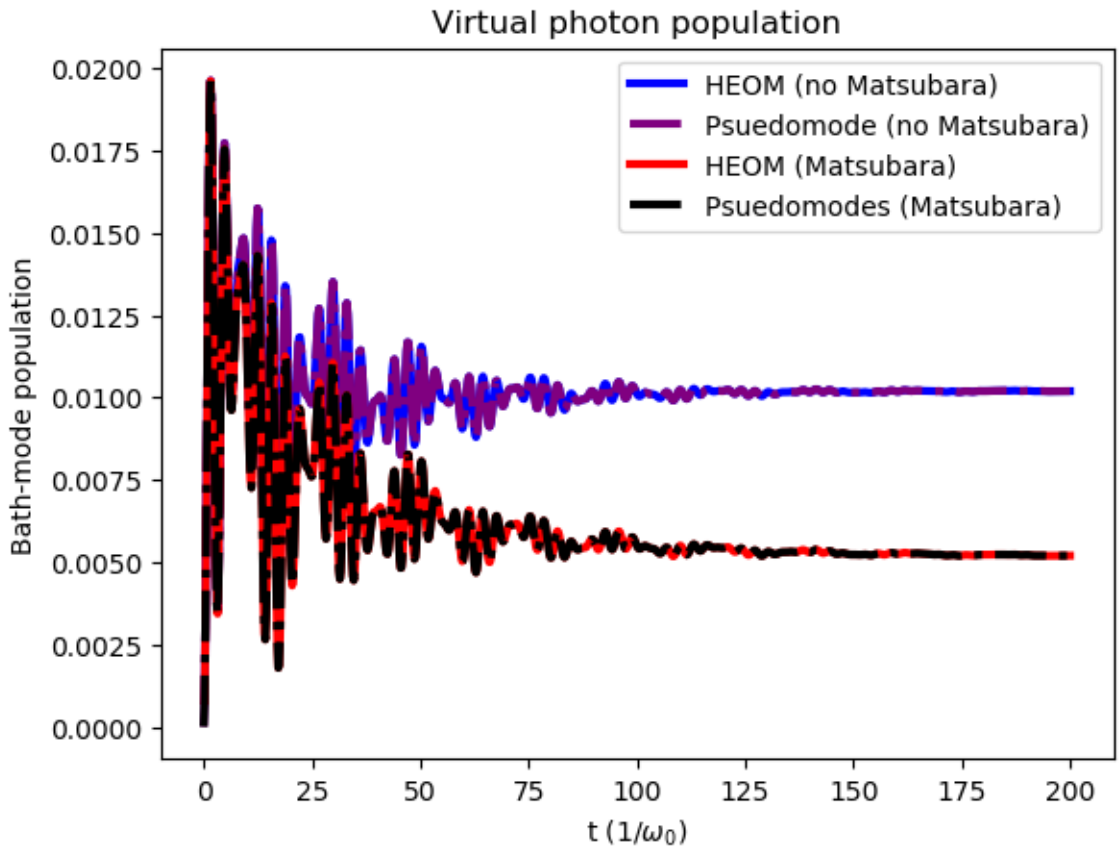
(continued from previous page)

```
rho0 = initial_ket*initial_ket.dag()
omega = np.sqrt(bath_freq**2 - (bath_broad/2.)**2)
options = Options(nsteps=1500, store_states=True, atol=1e-12, rtol=1e-12)

# zero temperature case, renormalized coupling strength
beta = np.inf
lam_renorm = coup_strength**2/(2*(omega))

ck1, vk1 = nonmatsubara_exponents(coup_strength, bath_broad, bath_freq, beta)

# Ignore Matsubara
hsolver_no_matsubara = HeomUB(Hsys, Q, lam_renorm, ck1, -vk1, ncut=Nc)
output_no_matsubara = hsolver_no_matsubara.solve(rho0, tlist, options)

# Add zero temperature Matsubara coefficients
mats_data_zero = matsubara_zero_analytical(coup_strength, bath_broad, bath_freq,
↪tlist)
ck20, vk20 = biexp_fit(tlist, mats_data_zero)
hsolver_matsubara = HeomUB(Hsys, Q, lam_renorm, np.concatenate([ck1, ck20]),
                           np.concatenate([-vk1, -vk20]), ncut=Nc)
output = hsolver_matsubara.solve(rho0, tlist, options)


# Get the auxiliary density matrices from the full Hierarchy ADOs
aux_2_list, indices1 = get_aux_matrices(hsolver_no_matsubara.full_hierarchy, 2, Nc, 2)
aux_2_list_matsubara, indices2 = get_aux_matrices(hsolver_matsubara.full_hierarchy, 2,
↪ Nc, 4)
virtual_photon_heom_no_matsubara = np.array([aux.tr() for aux in aux_2_list[1]])
virtual_photon_heom_matsubara = np.array([aux.tr() for aux in aux_2_list_
↪matsubara[8]])

# ======================================
# Compute the population from pseudomode
# ======================================
print("pseudomode")
lam2 = np.sqrt(lam_renorm)
Ncav = 4

# Construct the pseudomode operators with one extra underdamped pseudomode
sx = tensor(sigmax(), qeye(Ncav))
sm = tensor(destroy(2).dag(), qeye(Ncav))
sz = tensor(sigmaz(), qeye(Ncav))
a = tensor(qeye(2), destroy (Ncav))

Hsys = 0.5*wq*sz + 0.5*delta*sx + omega*a.dag()*a + lam2*sx*(a + a.dag())
initial_ket = basis(2, 1)
psi0=tensor(initial_ket, basis(Ncav, 0))

options = Options(nsteps=1500, store_states=True, atol=1e-13, rtol=1e-13)
c_ops = [np.sqrt(bath_broad)*a]
e_ops = [sz, sm.dag(), a, a.dag(), a.dag()*a, a**2, a.dag()**2]
pseudomode_no_mats = mesolve(Hsys, psi0, tlist, c_ops, e_ops, options=options)
output = (pseudomode_no_mats.expect[0] + 1)/2

# Construct the pseudomode operators with three extra pseudomodes
# One of the added modes is the underdamped pseudomode and the two extra are
```

(continues on next page)

(continued from previous page)

```
# the matsubara modes.
sx = tensor(sigmax(), qeye(Ncav), qeye(Ncav), qeye(Ncav))
sm = tensor(destroy(2).dag(), qeye(Ncav), qeye(Ncav), qeye(Ncav))
sz = tensor(sigmaz(), qeye(Ncav), qeye(Ncav), qeye(Ncav))
a = tensor(qeye(2), destroy(Ncav), qeye(Ncav), qeye(Ncav))

b = tensor(qeye(2), qeye(Ncav), destroy(Ncav), qeye(Ncav))
c = tensor(qeye(2), qeye(Ncav), qeye(Ncav), destroy(Ncav))

lam3 =1.0j*np.sqrt(-ck20[0])
lam4 =1.0j*np.sqrt(-ck20[1])

Hsys = 0.5*wq*sz + 0.5*delta*sx + omega*a.dag()*a + lam2*sx*(a + a.dag())
Hsys = Hsys + lam3*sx*(b+b.dag())
Hsys = Hsys + lam4*sx*(c + c.dag())

psi0 = tensor(initial_ket, basis(Ncav,0), basis(Ncav,0), basis(Ncav,0))
c_ops = [np.sqrt(bath_broad)*a, np.sqrt(-2*vk20[0])*b, np.sqrt(-2*vk20[1])*c]
e_ops = e_ops = [sz, sm.dag(), a, a.dag(), a.dag()*a, a**2, a.dag()**2]
L = -1.0j*(spre(Hsys)-spost(Hsys)) + liouvillian(0*Hsys,c_ops)
pseudomode_with_mats = mesolve(L, psi0, tlist, [], e_ops, options=options)

# Plot the bath populations
# Strange bug related to time steps in mesolve.

plt.plot(tlist[1:], np.real(virtual_photon_heom_no_matsubara), "-", color="b",␣
↪linewidth=3, label = r"HEOM (no Matsubara)")
plt.plot(tlist, np.real(pseudomode_no_mats.expect[4]), linestyle="-.",  color="purple
↪", linewidth = 3, label = r"Psuedomode (no Matsubara)")
plt.plot(tlist[1:], np.real(virtual_photon_heom_matsubara), "-", linewidth=3,  color=
↪"r", label = r"HEOM (Matsubara)")
plt.plot(tlist, np.real(pseudomode_with_mats.expect[4]), linestyle="-.", linewidth=3,␣
↪color="black", label="Psuedomodes (Matsubara)")
plt.title("Virtual photon population")
plt.xlabel("t ($1/\omega_0$)")
plt.ylabel("Bath-mode population")
plt.legend()
plt.show()
```


Virtual photon population
HEOM (no Matsubara)
Psuedomode (no Matsubara)
HEOM (Matsubara)
Psuedomodes (Matsubara)
Bath-mode population
0.0200
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
25
50
75
100
125
150
175
200
t (1/ω0)

CHAPTER 10

API Documentation

10.1 matsubara.heom

10.2 matsubara.correlation

CHAPTER 11

References

Bibliography

[LACN19] Neill Lambert, Shahnawaz Ahmed, Mauro Cirio, and Franco Nori. Virtual excitations in the ultra-strongly-coupled spin-boson model: physical results from unphysical modes. *arXiv preprint arXiv:1903.05892*, 2019.

[SS17] Linze Song and Qiang Shi. Hierarchical equations of motion method applied to nonequilibrium heat transport in model molecular junctions: transient heat current and high-order moments of the current operator. *Physical Review B*, 95(6):064308, 2017.

[ZLXS12] Lili Zhu, Hao Liu, Weiwei Xie, and Qiang Shi. Explicit system-bath correlation calculated using the hierarchical equations of motion method. *The Journal of chemical physics*, 137(19):194106, 2012.