Chapter_6.2

June 6, 2024

[ ]:
```
import matplotlib.pyplot as plt
import numpy as np
import scipy as sci
import qutip as qt
from qutip import (Qobj, about, basis, coherent, coherent_dm, create, destroy,
                   expect, fock, fock_dm, mesolve, qeye, sigmax, sigmay,
                   sigmaz, tensor, thermal_dm)
import lattpy as lp
from scipy.optimize import curve_fit
from scipy.linalg import expm
```

[ ]:
```
plt.rcParams['text.usetex'] = True
plt.rcParams['font.family'] = 'serif'
plt.rcParams['text.latex.preamble'] = r'\usepackage{amsmath}'

N = 50
d_value = 1
gamma0 = 1
lambda0 = 2 * np.pi
distances = np.arange(0.00001 * np.pi, 4 * np.pi, 0.01 * np.pi)
k0 = 1

atoms1 = np.zeros((N, 3), dtype=np.complex128)

decay_rates_dictionary = {}
min_decay_rates = []

"""changed the atom position to a circle"""

for index, distance in enumerate(distances):
    radius = distance * N / (2 * np.pi)
    for i in range(N):
        angle = 2 * np.pi * i / N
        atoms1[i, 1] = radius * np.cos(angle)
        atoms1[i, 2] = radius * np.sin(angle)

    r1 = [atoms1[i] for i in range(N)]
    r_T1 = [i.reshape(-1, 1) for i in r1]
```

```
    d = np.array([d_value, 0, 0])
    d_abs = np.linalg.norm(d)
    d = d.reshape(-1, 1)
    d_T = d.reshape(1, -1)

    def G0(rij, k0):
        r0 = np.linalg.norm(rij)
        rij_T = rij.reshape(-1, 1)
        return ((3 * np.pi * k0 * np.exp(1j * k0 * r0)) / (4 * np.pi * (k0 *␣
 ↪r0) ** 3)) * \
               ((k0 ** 2 * r0 ** 2 + 1j * k0 * r0 - 1) * np.identity(3) +
                ((-k0 ** 2 * r0 ** 2 - 3j * k0 * r0 + 3) * (np.dot(rij, rij_T) /
 ↪ r0 ** 2)))

    def H_eff(r1, d, N):
        Matrix = np.zeros((N, N), dtype=np.complex128)
        k0 = 1
        for i in range(N):
            for j in range(N):
                if i == j:
                    Matrix[i, j] += -1j * gamma0 / 2
                else:
                    if i < N and j < N:
                        rij = r1[i] - r1[j]
                    Matrix[i, j] += -np.dot(d_T, np.dot(G0(rij, k0), d))
        return Matrix

    H_eff_matrix = H_eff(r1, d, N)
    eigenvalues = np.imag(np.linalg.eigvals(H_eff_matrix))
    H_eff_imag_eigenvalues_sorted = np.sort(eigenvalues)

    decay_rates_dictionary[distance] = (2 * H_eff_imag_eigenvalues_sorted /␣
 ↪(-gamma0))
    min_decay_rates.append(np.min(np.abs(decay_rates_dictionary[distance])))

fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 8))

for distance, decay_rates in decay_rates_dictionary.items():
    lattice_constant = distance / lambda0
    jitter = np.random.normal(0, 0.01, size=len(decay_rates))
    ax1.scatter(lattice_constant + jitter, decay_rates, s=1, color='blue')

ax1.set_title(r'\textbf{Decay Rates vs. Lattice Constant}', fontsize=16)
ax1.set_xlabel(r'$d / \lambda_{0}$', fontsize=16)
ax1.set_ylabel(r'$\mathbf{\Gamma}_{\xi} / \mathbf{\Gamma}_{0}$', fontsize=14)
ax1.set_ylim([0, 3])
```

```
ax1.set_xlim([0, 2])
ax1.axhline(y=1, color='black', linestyle='--')
ax1.axvline(x=0.5, color='black', linestyle='--')
ax1.set_xticks(np.arange(0, 2, 0.2))
ax1.set_yticks(np.arange(0, 3, 1))

ax2.scatter(distances / lambda0, min_decay_rates, color='blue', s=5)
ax2.set_title(r'\textbf{Min Decay Rates vs. Lattice Constant (0 to 0.5)}',␣
 ↪fontsize=16)
ax2.set_xlabel(r'$d / \lambda_{0}$', fontsize=16)
ax2.set_ylabel(r'$\mathbf{\Gamma}_{1} / \mathbf{\Gamma}_{0}$', fontsize=14)
ax2.set_xlim([0, 0.5])
ax2.set_yscale('log')
ax2.set_yticks(np.logspace(-2, -26, num=9))
ax2.text(0.012, 10**-1, r'$\mathbf{b)}$', fontsize=20, fontweight='bold',␣
 ↪verticalalignment='top')
ax2.axhline(y=10**(-16), color='black', linestyle='--')

plt.tight_layout()
plt.show()
```