Decay_versus_lattice_constant

May 30, 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
import matplotlib.patches as patches
```

[ ]:
```
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

decay_rates_dictionary = {}

atoms = np.zeros((N, 3))


"""calculate the numerical Setup for different lattice constants and save it in␣
 ↪a dictionary"""
for distance in distances:
    """changable atomposition, calculation valid for different geometries"""
    for i in range(N):
        atoms[i, 2] = i * distance*k0
    r = [atoms[i] for i in range(N)]
    r_T = [i.reshape(-1, 1) for i in r]

    d=np.array([0,d_value,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(r, d, N):
        Matrix = np.zeros((N, N), dtype=complex)
        k0=1
        for i in range(N):
            for j in range(N):
                if i == j:
                    Matrix[i, j] += -1j*gamma0/2
                else:
                    rij = r[i] - r[j]
                    Matrix[i, j] += -np.dot(d_T, np.dot(G0(rij,k0), d))
        return  Matrix

    H_eff_matrix = H_eff(r, d, N)
    eigenvalues = np.imag(np.linalg.eigvals(H_eff_matrix))
    H_eff_imag_eigenvalues_sorted = np.sort(eigenvalues)
    decay_rates_dictionary[distance]=np.abs((2*H_eff_imag_eigenvalues_sorted/
↪-1j*gamma0)) #normalisation


"""Plot the decay rates vs. lattice constant from the dicionary"""

plt.figure()
plt.figure(figsize=(15, 3))
for distance, decay_rates in decay_rates_dictionary.items():
    lattice_constant = distance / lambda0
    jitter = np.random.normal(0, 0.01, size=len(decay_rates))
    plt.scatter(lattice_constant + jitter, decay_rates, s=1, color='blue')

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

plt.title(r'\textbf{Decay rates vs. lattice constant}', fontsize=16)
plt.tick_params(axis='both', which='major', labelsize=13)
plt.xlabel(r'$d / \lambda_{0}$', fontsize=16)
plt.ylabel(r'$\mathbf{\Gamma}_{\xi} / \mathbf{\Gamma}_{0}$', fontsize=14)


plt.axhline(y=1, color='black', linestyle='--')
```

```
plt.axvline(x=0.5, color='black', linestyle='--')
plt.ylim([0, 3])
plt.xlim([0, 2])
plt.yticks(np.arange(0, 3.0001, 1))
plt.xticks(np.arange(min(distances/lambda0), max(distances/lambda0)+0.2, 0.2))

plt.show()
```