# Numerical Set-Up, Chapter 4

May 29, 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
from scipy.linalg import eigvals
from decimal import *
```

[ ]:
```
""" variables """

N=50
d_value=1
gamma0 = 1
lambda0=(2*np.pi)
distance= (lambda0/4)
k0=1
```

[ ]:
```
"""Vectors: Dipolemoment, Atomposition"""

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

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

[ ]:
```
"""Greensfunction G0 and effective Hamiltonian H_eff"""

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)
    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
```