```
            # Heisenberg interaction + tunneling for unequal nearest neighbors
            else:
                h[b,b]-=0.25*V
                bp = findIndex(table, bitFlip(t, j, k, length))
                #bp = findIndex(table, bitFlip(t, j, k, length))-1
                h[b,bp]=1.0-delta*(-1)**j
    return h

def calc_LE(vmi,Umi,vmf,Umf,t): # return rate starting from intial state
    """compute time evolution from ground state of MPHi"""
    Uindex=vmi.tolist().index(min(vmi))
    Ui=Umi[Uindex]
    LE = np.zeros(len(t))
    evol = np.exp(-1j*t[:,np.newaxis]*vmf[np.newaxis])
    vv=np.dot(Umf.transpose().conj(),Ui)
    """ |<Psi0|U*exp(-iEt)*U|Psi0>|"""
    LE=np.abs(np.inner(evol*vv[np.newaxis,:],vv[np.newaxis,:]))
    return LE

# Run the program for both cases
t=np.arange(args.tint,args.tmax+args.dt/2,args.dt)

# calculate many-particle Hamiltonian
particlenumber=args.L/2
Psi=manyPsi(particlenumber,args.L)
table = findMagnetizationStates(args.L,particlenumber)
# condition for fixing interaction strength V
if args.Vfix == 1:
    MPHi = construct_MPH(args.V,args.delta,args.L,table)
elif args.Vfix == 0:
    MPHi = construct_MPH(0,args.delta,args.L,table)
MPHf = construct_MPH(args.V,-args.delta,args.L,table)

Store2=0
for samp in range(int(args.sample)):
    MPDW = construct_MPDW(Psi,args.L,args.W)
    MPHiW = MPHi + MPDW
    MPHfW = MPHf + MPDW
    vmi,Umi = np.linalg.eig(MPHiW)
    Umi=Umi.transpose()
    vmf,Umf = np.linalg.eig(MPHfW)
    Store2 += calc_LE(vmi,Umi,vmf,Umf,t)

LE2=np.squeeze(Store2/args.sample)
RR2=-2*np.log(LE2)/args.L

for item in RR2:
    print(item)
```