```
'''
PAWANPREET KAUR
2020PHY1092
REFERENCE: https://www.lehman.edu/faculty/anchordoqui/400_5.pdf
'''

import numpy as np
from scipy.constants import hbar,m_e ,e#(Js),kg
import sys
import matplotlib.pyplot as plt
import pandas as pd

def k(E,V):
    if V>E:
        sys.exit("E should be greater than or equal to V")
    else:
        pass
    return np.sqrt(2*m*(E-V))/hbar

def p_mat(k_j,L_j,k_j1):

    p_free=np.zeros((2,2),dtype='complex')
    p_free[0][0]=np.exp(-1j*k_j*L_j)
    p_free[1][1]=np.exp(1j*k_j*L_j)

    p_step=np.zeros((2,2),dtype='complex')
    p_step[0][0]=1+(k_j1/k_j)
    p_step[0][1]=1-(k_j1/k_j)
    p_step[1][0]=1-(k_j1/k_j)
    p_step[1][1]=1+(k_j1/k_j)
    p_step=np.dot(0.5,p_step)

    p_mat=np.matmul(p_free,p_step)

    return p_free,p_step,p_mat

def main(E,V_array,m,L_j):
    V=V_array[-1]
    V0=V_array[0]
    k1=k(E,V0)
    k2=k(E,V)
    p1=p_mat(k1,L_j,k2)
    p_f=p1[0]
    p_s=p1[1]
    p=p1[2]
    A_in_l=p[1][0]/p[0][0]
    A_out_r=1/p[0][0]

    #probabilities
    Trans_prob=abs(A_out_r)**2
    Ref_prob=abs(A_in_l)**2

    #coefficients (Numerical)
    Ref_coef=Ref_prob
    Trans_coef=Trans_prob*(k2/k1)
    sum_of_coeff=Ref_coef+Trans_coef

    #coefficients (THEORETICAL)
    Ref_coef_th=abs((k1-k2)/(k1+k2))**2
    Trans_coef_th=4*k1*k2/abs(k1+k2)**2
    sum_of_coef_th=Ref_coef+Trans_coef

    #current densities
    J_in = e*hbar*k1/m_e
    J_trans=e*hbar*k2*Trans_prob/m_e
    J_ref=-1*e*hbar*k1*Ref_prob/m_e
```

```
    #total probability
    P_T=Trans_prob+Ref_prob

    #print("Trans_prob + Ref_prob = ",P_T)

    if P_T==1:
        msg="Trans_prob + Ref_prob =1 , Hence verified"

    else:
        msg="Trans_prob + Ref_prob !=1 , Not verified"

    #print(msg)
    return
p_f,p_s,p,Trans_prob,Ref_prob,P_T,msg,Ref_coef,Trans_coef,sum_of_coeff,J_in,J_ref,J_trans,Ref_coef_th,Tra


m=m_e #mass of particle
V0=0
V=2*1.6*10**(-19) #J (height of barrier)
L=2e-10 #(m)
E=1.01*V #J

E_array=np.linspace(E,2*V,100) #array of E values
V_array=[V0,V]
p_f,p_s,p,Trans_prob,Ref_prob,P_T,msg,Ref_coef,Trans_coef,sum_of_coeff,J_in,J_ref,J_trans,Ref_coef_th,Tra

print("########################################")
print("V=2 eV")
print("E=2.02 eV")
print("########################################")
print('pf')
print(p_f)
print()
print("########################################")
print('ps')
print(p_s)
print()
print("########################################")
print('p')
print(p)
print()
print("########################################")
print('Trans Coef (Numerical)')
print(Trans_coef)
print()
print("########################################")
print('Ref Ceof (Numerical)')
print(Ref_coef)
print()
print("########################################")
print('Sum of Ceof (Numerical)')
print(sum_of_coeff)
print()

print("########################################")
print('Trans Coef (Theoretical)')
print(Trans_coef_th)
print()
print("########################################")
print('Ref Ceof (Theoretical)')
print(Ref_coef_th)
print()
print("########################################")
print('Sum of Ceof (Theoretical)')
```

```
print(sum_of_coef_th)
print()

print("########################################")
print('Incoming Current Density')
print(J_in)
print()
print("########################################")
print('Reflection Current Density')
print(J_ref)
print()
print("########################################")
print('Transmission Currrent Density')
print(J_trans)
print()

print("########################################")
print('Trans Prob')
print(Trans_prob)
print()
print("########################################")
print('Ref Prob')
print(Ref_prob)
print()
print("########################################")
print('trans_prob + ref_prob')
print(P_T)
print()
print(msg)
print("########################################")
print()

##########
lis=[]   #trans prob
lis2=[]    #ref prob
lis3=[] #total prob

c1=[] #ref coeff num
c2=[] #  trans coef num
c3=[]  #Sum of coeff num

t1=[] #ref coeff th
t2=[] #  trans coef th
t3=[]  #Sum of coeff th


jin=[] #incoming current density
jtr=[] # trans current density
jref=[]  #ref current density

for E in E_array:
    d=main(E,V_array,m,L)
    f=d[3]
    lis.append(f)
    lis2.append(d[4])
    lis3.append(d[5])
    c1.append(d[-9])
    c2.append(d[-8])
    c3.append(d[-7])
    jin.append(d[-6])
    jtr.append(d[-4])
    jref.append(d[-5])
    t1.append(d[-3])
    t2.append(d[-2])
    t3.append(d[-1])
```

```
plt.plot(lis,E_array/(1.6*10**(-19)),label="Transmission Probability")
plt.plot(lis2,E_array/(1.6*10**(-19)),label="Reflection Probability")
plt.plot(lis3,E_array/(1.6*10**(-19)),label="Total Probability")
plt.xlabel(" probability")
plt.ylabel("Energy (in eV)")
plt.legend()
plt.grid()
plt.title("V = 2eV , E0=2.02 eV  (E vs Probability plot)")
plt.show()

######
plt.plot(c2,E_array/(1.6*10**(-19)),label="Transmission Coefficient Numerical")
plt.plot(c1,E_array/(1.6*10**(-19)),label="Reflection Coefficient Numerical")
plt.plot(c3,E_array/(1.6*10**(-19)),label="Transmission Coefficient+Reflection
Coefficient Numerical")

plt.scatter(t2,E_array/(1.6*10**(-19)),label="Transmission Coefficient
Theoretical",marker='o')
plt.scatter(t1,E_array/(1.6*10**(-19)),label="Reflection Coefficient
Theoretical",marker='*')
plt.scatter(t3,E_array/(1.6*10**(-19)),label="Transmission Coefficient+Reflection
Coefficient Theoretical",marker='^')
plt.xlabel(" Coefficient")
plt.ylabel("Energy (in eV)")
plt.legend()
plt.grid()
plt.title("V = 2eV , E0=2.02 eV  (E vs Coefficient plot)")
plt.show()

#######
plt.plot(jtr,E_array/(1.6*10**(-19)),label="Transmission current Density")
plt.plot(jref,E_array/(1.6*10**(-19)),label="Reflection current density")
plt.plot(jin,E_array/(1.6*10**(-19)),label="Incoming current Density")
plt.xlabel(" Current Density")
plt.ylabel("Energy (in eV)")
plt.legend()
plt.grid()
plt.title("V = 2eV , E0=2.02 eV  (E vs Current Density plot)")
plt.show()

energy=E_array/(1.6*10**(-19))
f={"Energy (in eV)":energy,"R (Num)":c1,"T (Num)":c2,"R+T (Num)":c3,"R (Th)":t1,"T
(Th)":t2,"R+T (Th)":t3}
print(pd.DataFrame(f))
```