```python
import numpy as np
import matplotlib.pyplot as plt
import random
import mcint


def diff_r_rprim(r, r_p, theta, theta_p, phi, phi_p):
    '''
    Returns the difference between two arbitrary vectors r and r',
    given their position.
    '''
    term1 = r**2 + r_p**2
    term2  = 2*r*r_p*(np.sin(theta)*np.sin(theta_p)*np.cos(theta-
phi)+np.cos(theta)*np.cos(theta_p))
    diff_sqrt = np.sqrt(term1-term2)
    return diff_sqrt


def k_dot_r(k,r,theta):
    '''
    Returns the value of the dot product between k and r,
    with the coordinates aligned with the z || z' axes.
    '''
    return k*r*np.cos(theta)


""" def kp_dot_rp(alpha, beta, kp, rp, thetap, phip):
    '''
    alpha and beta are the azimuthal and the polar angles for
    k' respectively. The integral should be independent of
    beta, due to spherical symmetry.
    '''
    return kp*rp*(np.sin(alpha)*np.cos(beta-phip)+np.cos(alpha)*np.cos(thetap))


def integrand(x, alpha, beta, k, kp):
    # Dimensions to integrate over
    r = x[0]
    phi = x[1]
    theta = x[2]
    rp = x[3]
    phip = x[4]
    thetap = x[5]

    factor1 = np.exp(-1j*kp_dot_rp(alpha, beta, kp, rp, thetap, phip))
    factor2 = np.exp(1j*k_dot_r(k, r, theta))
    diffrrp = diff_r_rprim(r, rp, theta, thetap, phi, phip)
    factor3 = np.exp(1j*k*diffrrp)/diffrrp
    return factor1*factor2*factor3 """

# Start from scratch
def volume(R):
    return 2*np.pi**2 * R


def differens(r, r_p, theta, theta_p, phi, phi_p):
    '''
    Returns the difference between two arbitrary vectors r and r',
    given their position.
    '''
    return np.sqrt(r**2 + r_p**2 - 2*r*r_p*
(np.sin(theta)*np.sin(theta_p)*np.cos(phi-phi_p) + np.cos(theta)*np.cos(theta_p)))
```

```
def dotprod1(k, r_p, alpha, theta_p, beta, phi_p):
    '''
    alpha and beta are the azimuthal and the polar angles for
    k' respectively. The integral should be independent of
    beta, due to spherical symmetry.
    '''
    return k*r_p*(np.sin(alpha)*np.sin(theta_p)*np.cos(beta-phi_p) +
np.cos(alpha)*np.cos(theta_p))


def dotprod2(k,r,theta):
    '''
    Returns the value of the dot product between k and r,
    with the coordinates aligned with the z || z' axes.
    '''
    return k*r*np.cos(theta)


def integrand(k, r, r_p, theta, theta_p, alpha, beta, phi, phi_p):
    ''' The integrand that we wish to approximate. '''
    exp1 = np.exp(-1j*dotprod1(k,r_p,alpha,theta_p,beta,phi_p))
    exp2 = np.exp(1j*dotprod2(k,r,theta))
    exp3 = np.exp(1j*differens(r, r_p, theta, theta_p, phi, phi_p))
    fac = r**2*r_p**2*np.sin(theta)*np.sin(theta_p)/differens(r, r_p, theta,
theta_p, phi, phi_p)
    return exp1*exp2*exp3*fac*volume(R)**2


def calc_q(alpha,k):
    return 2*k*np.sin(alpha/2)


def calc_f1(R, k, alpha):
    q = calc_q(alpha, k)
    return (np.sin(q*R)-R*q*np.cos(q*R))


# constants
N = int(1e7)
eV = 1.6e-19  # J
c = 3e8  # m/s
hbar = 1.054e-34  # ev s
R = 1e-10  # 1 ångström
#v0 = (1e-38)*eV  # simon test
v0 = 1*eV  # 1 eV
E = 10*eV  # 1 ev
m = 938e6*eV/c**2  # 938 MeV/c^2
k = np.sqrt(2*m*E)/hbar
kp = k  # Elastic

# R = 1
# hbar = 1
# E = 100
# V0 = 1
# m = 1
# k = 70
# kp = 1
# v0 = 1



r     = R*np.random.rand(N)
r_p   = R*np.random.rand(N)
theta = np.pi*np.random.rand(N)
```

```python
theta_p = np.pi*np.random.rand(N)
phi   = 2*np.pi*np.random.rand(N)
phi_p   = 2*np.pi*np.random.rand(N)

alpha = np.linspace(0,0.6,100)
beta = 0

f_1 = np.zeros(len(alpha))
f_2 = np.zeros(len(alpha))

for j,alph in enumerate(alpha):
    print(f'{j+1} of {len(alpha)}')
    I =  integrand(k, r, r_p, theta, theta_p, alph, beta, phi, phi_p)
    s = (I/N).sum()
    f_1[j] = -4*m*v0/(hbar**2 * calc_q(alph, k)**3) * calc_f1(R, k, alph)
    f_2[j] = s


# Insert prefactor for f_2 and plot
f_2 = (m*v0/hbar**2)**2 * f_2
sigma_contrib2 = np.abs(f_2)**2
sigma_contrib1 = np.abs(f_1)**2
sigma_contrib_tot = np.abs(f_1 + f_2)**2  # Total differential cross section
fig, ax = plt.subplots(3,1)
fig.suptitle("Problem 8: Comparison of first and second order")
ax[0].plot(alpha, sigma_contrib1, 'b--')
ax[0].set_xlabel("alpha, rad")
ax[0].set_ylabel("|f1|^2, m^2")
ax[0].grid()

ax[1].plot(alpha, sigma_contrib2, 'r--')
ax[1].set_xlabel("alpha, rad")
ax[1].set_ylabel("|f2|^2, m^2")
ax[1].grid()

ax[2].plot(alpha, sigma_contrib1, 'b--')
ax[2].plot(alpha, sigma_contrib2, 'r--')
ax[2].plot(alpha, sigma_contrib_tot, 'g-')
ax[2].set_xlabel("alpha, rad")
ax[2].set_ylabel("|f1 + f2|^2, m^2")
ax[2].grid()
plt.tight_layout()
plt.savefig("problem8.png")
plt.show()

```