Near-Field Radiative Heat Transfer Between Two α-MoO3 Biaxial Crystals

Doke Vinit Prashant, Kaushik Singirikonda, Khade Shourish Shashank, Parth Shrivastava

The near-field Radiative Heat Transfer between to semi infinite $\alpha - MoO_3$ biaxial crystals is studied numerically using the fluctuation dissipation theorem combined with modified 4×4 transfer matrix method.It is found that the maximum heat flux is along the [001] direction, which can be explained by the existence of hyperbolic phonon polaritons (HPPs) inside $\alpha - MoO_3$ and excitation of hyperbolic surface phonon polaritons (HSPhPs) at the vacuum/$\alpha - MoO_3$ interfaces.We also the study the effect of relative rotation between the two $\alpha - MoO_3$ crystals, we notice that the modulation contrast can be as high as two. This is attributed to the misalignment of HSPhPS and HPPs of the emitter and receiver.

Introduction

History

Wein's Law

This was one of the first attempts to describe the spectrum of thermal emissions by a blackbody. Wein derived his law from basic thermodynamics arguments, which assumed a continuous spectrum of energy.

$$I(\nu) = \frac{2h\nu^3}{c^2} e^{-\frac{h\nu}{kT}}$$

Though this law matched the experimental data for small wavelengths, it failed to describe the physics for higher wavelengths.

Rayleigh-Jeans Law

Rayleigh-Jeans law considers the oscillations of atoms as the source of radiation but as the quantization of energy of oscillators was not known, they had assumed it to continuous. This lead to the derivation of the formula

$$I(\lambda) = \frac{2ckT}{\lambda^4}$$

Though this expression gave us accurate results for larger wavelengths, it terribly failed to predict the thermal radiation at smaller wavelengths, it was assumed that the energy goes to infinity as we decrease the wavelength.

Near-Field Radiative Heat Transfer

In 1953, Rytov solved an example problem of radiative heat transfer between two closely-spaced parallel plane surfaces, with one being an arbitrary dissipative medium and the other being a mirror of good electrical conductivity. He suggested that the "energy flow density into the mirror" could increase "without limit" as the spatial separation between the two planes vanishes. This result represents a dramatic deviation from the constant heat flow independent of the separation as predicted by Planck's (Stefan-Boltzmann) law.

In January 1971 Polder and Van Hove presented their widely recognized theory of radiative heat transfer between closely spaced bodies. Specifically, radiative heat transfer between two chromium (Cr) half spaces was studied. Contributions to heat transfer across the vacuum gap from both the propagating and evanescent electromagnetic waves, as well as both the transverse electric (TE, or s-mode) and transverse magnetic (TM, or p-mode) polarizations were consistently considered, with each individual combination (say propagating TE or TM modes) naturally separated from the others. A comparison of the spectrum of radiated power in a small vacuum gap with that in an infinitely large gap clearly demonstrated that the contribution from evanescent TM modes was dominant for small gaps. Compared to the constant heat transfer rate given by Planck's law, several orders of magnitude enhancement in heat transfer between two Cr surfaces across nanometer gaps was predicted at room temperature

Polaritons

Polaritons are hybrid particles(quasiparticles) made up of a photon(EM waves) strongly coupled to an electric dipole or magnetic dipole. Polaritons can be thought of as the new modes(normal modes) that arise from the strong coupling between bare modes of the material and photons.

Polaritons were first introduced in the 1950s by Tolpygo; when he analysed the dispersion relation in ionic crystals, these are now known as phonon polaritons. Fröhlich and Pelzer described the plasmon-polariton in 1955. We are going to limit our discussion to Phonon Polaritons and Surface Plasmon Polaritons(Properties)

Phonon Polariton

Phonon polaritons are observed when EM waves couple to the lattice vibrations modes of ferroelectric crystals. These waves operate at frequencies in the THz range, where these lattice vibrational waves travel at light-like speeds.

The weak van der Waals bonded nature of few materials makes them optically anisotropic. This suggests that the dielectric responses in perpendicular crystallographic directions should have opposite signs. This makes MoO3 a natural Hyperbolic Material, just like hBN.The hyperbolic polaritons produced within bulk $\alpha - MoO_3$ allow electromagnetic modes with large momenta to propagate within the otherwise forbidden Reststrahlen band.

Surface Phonon Polaritons

At particular frequencies(in general infrared frequencies), some lattices have negative permittivity and are related to phonons(these) but exhibit similar properties as that of Surface Plasmon Polaritons; they are called Surface Phonon Polaritons. As $\alpha - MoO_3$ is a Hyperbolic material, the EM waves produce Hyperbolic Surface Phonon Polaritons.

Why?

Why exactly is Near-Field Radiative Heat Transfer (NFRHT) being studied?

In the last two decades, Near-Field Radiative Heat Transfer has been seen to have promising applications in thermophotovoltaics, noncontact refrigeration, thermal transistors, etc. As opposed to far-field radiative heat transfer where only propagating waves contribute to heat flux, in NFRHT, evanescent waves are the dominating component. The near-field radiative heat flux (NFRHF) can exceed the blackbody limit by several orders in magnitude if surface polaritons can be excited.

Why exactly was there a need for this paper?

In the past several years, studies on NFRHT between two hyperbolic metamaterials have revealed that the near-field radiative heat flux can be greatly enhanced via hyperbolic phonon polaritons (HPPs). For most artificially engineered hyperbolic metamaterials, for certain constraints on the tangential wavevector component, the hyperbolic properties do not hold. For natural hyperbolic materials like hexagonal Boron Nitride (hBN) however, these limitations on wavevector are negligible.

Experiments on NFRHT between two hBN crystals have been conducted in the paper titled "Influence of hBN orientation on the near-field radiative heat transfer between graphene/hBN heterostructures." The NFRHF between two heat transfer surfaces can be modulated by introducing a relative rotation between the receiver and emitter. In the case of hBN which is a uniaxial material, modulating the NFRHF by controlling the relative rotation angle is possible only when the optic axis is parallel to the material surface. When it's optic axis is perpendicular to the material surface, the crystal possesses in-plane isotropy and in such a case relative rotation between emitter and receiver does not influence NFRHF. The biaxial MoO_3 crystal considered in this paper possessess anisotropy regardless of whether the optical axis is perpendicular or parallel to the material surface. The purpose of the reviewed paper is to understand whether MoO_3 can outperform hBN in achieving enhanced NFRHF and controlling the NFRHF by using various parameters and orientations of the crystals.

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.spatial.transform import Rotation as R
%matplotlib inline
```

Material Characteristics

$\alpha - MoO_3$ crystals are hyperbolic and biaxial in nature due to which the *NFRHT* depends strongly on the orientation and the material hence shows different optical properties in its 3 crystalline directions. Also relative twist between two surfaces(emitter and receiver) has also proved to be an effective way to modulate NFRHT.

The permitivity tensor of a hyperbolic media has component along one axis of opposite sign compared with other two axes and has ability to support EM feilds with high momenta.

The lattice of $\alpha - MoO_3$ has octahederal unit cells with with nonequivalent $Mo - O$ bondsalong the three principal axes which gives rise to rich phonon modes along different crystalline directions The dielectric tensor can be calculted by using a Lorentz model

$$\epsilon_m = \epsilon_{\infty,m}\left(1 + \frac{\omega_{LO,m}^2 - \omega_{TO,m}^2}{\omega_{TO,m}^2 - \omega^2 - j\omega\Gamma_m}\right),\ m = x, y, z$$

here the principal components of the tensor are calculated

In [2]:
```python
# Universal Constants
h_bar   = 1.055 * 10**(-34)
c       = 2.998* 10**(8)
k_boltz = 1.381*10**(-23)
j       = complex(0,1)

#Experimental Conditions
d       = 20*10**(-9)
T1      = 300 #Kelvin
T2      = 0   #Kelvin
```

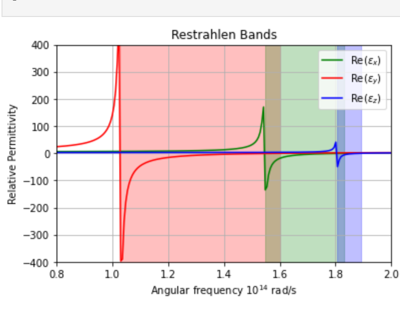
In [3]:
```python
# Material-Specific Constants #(x,y,z)

epsilon_infi = np.array([4,5.2,2.4])
omega_LO     = np.array([1.8322*10**(14),1.6041*10**(14),1.8925*10**(14)])
omega_TO     = np.array([1.5457*10**(14),1.0273*10**(14),1.8058*10**(14)])
Gamma        = np.array([7.5398*10**(11),7.5398*10**(11),3.7699*10**(11)])
```

In [4]:
```python
def evaluate_eps(omega):
  num = omega_LO**2 - omega_TO**2
  den = omega_TO**2 - omega**2 - j*omega*Gamma
  eps = epsilon_infi*(1+num/den)
  return eps
```

Reststrahlen bands

The real parts of the pricipal components are studied and it is seen that the real parts of ϵ_y, ϵ_x and ϵ_z are negative in the three spectral regions from 1.0273×10^{24} to 1.6041×10^{14} rad/s, from 1.5457×10^{14} to 1.8322×10^{14} rad/s, and from 1.8058×10^{14} to 1.8925×10^{14} rad/s, respectively. These areas are called the Reststrahlen bands in which the hyperbolicity can be exhibited by the EM wave dispersion.

In [5]:
```python
#Plotting Real Part of Permittivity against Omega

omega_val = np.linspace(0.8,2,200)
eps = []
for omega in omega_val:
  eps.append(evaluate_eps(omega*10**14).real)
y_vals = np.array(eps).reshape(200,3)
##
plt.plot(omega_val, y_vals[:,0], 'g')
plt.plot(omega_val, y_vals[:,1], 'r')
plt.plot(omega_val, y_vals[:,2], 'b')
##
plt.grid()
plt.ylim(-400,400)
plt.xlim(0.8,2)
plt.legend(["Re($\epsilon_{x}$)","Re($\epsilon_{y}$)","Re($\epsilon_{z}$)"])
plt.title("Restrahlen Bands")
plt.xlabel("Angular frequency $10^{14}$ rad/s")
plt.ylabel("Relative Permittivity")
plt.fill_between(omega_val, -400, 400, where = (y_vals[:,1]<0), color = 'r',alpha=0.25
plt.fill_between(omega_val, -400, 400, where = (y_vals[:,0]<0), color = 'g',alpha=0.25
plt.fill_between(omega_val, -400, 400, where = (y_vals[:,2]<0), color = 'b',alpha=0.25
# plt.figure( figsize = (20,10),dpi=1000)
plt.show()
```



Setup

Here we study NFRHT between two slabs of $\alpha - MoO_3$ i.e. emitter and receiver seperated by a vacuum gap of distance 'd'and both are considered to be semi-infinite(therefore no role of transmitive coefficients). We take 3 configurations in order to study properties in three different pricipal crytalline directions, they are $[010]$, $[100]$ and $[001]$ for instance when $[010]$ is perpendicular to the surfaces, the heat flux is said to be along $[010]$ direction. Also we analyze the properties when emiiter is rotated by an angle γ with respect to the receiver along the pricipal axis, in that case the relative permitivity tensor of emiiter is given by -

$$\begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \epsilon \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Reflection Matrix

$$R_{1,2} = \begin{pmatrix} r_{ss}^{1,2} & r_{sp}^{1,2} \\ r_{ps}^{1,2} & r_{pp}^{1,2} \end{pmatrix}$$

This is the Fresnel's Reflection matrix, for incidence of s- and p- polarized plane waves from vacuum to medium 1 or 2 respectively. These can be obtained using the 4 × 4 matrix method as described below.

In [6]:
```python
#####
def Get_k(beta,omega):
  k_0 = omega/c
  kout = np.sqrt((k_0**(2)- beta**(2))*(-1*j**2)) #numpy sqrt
  return kout

####
def Diagonal_Dielectric_Tensor(omega):
  eps_x, eps_y, eps_z = evaluate_eps(omega)
  matrix = np.array([
      [eps_x,0,0],
      [0,eps_y,0],
      [0,0,eps_z]
  ]
  )
  return matrix
#####
def RotationMatrix(axis, angle, degree = False):
  r = R.from_euler(axis, angle)
  r = np.array(r.as_matrix())
  return r
#####
def PermittivityTensor(omega, axis ='z' , angle = 0):
  eps_diag = Diagonal_Dielectric_Tensor(omega)
  R = RotationMatrix(axis, angle)
#   R_inv = np.linalg.inv(R)
  return R.dot(eps_diag).dot(R.T)
#####
def R_matrix2(matrix,omega,beta):
    Rekt_matrix1, _ = reflect(matrix,omega,beta)
    Rekt_matrix2, _ = reflect4(matrix,omega,beta)
    Rekt_final = np.array([[Rekt_matrix2[1],Rekt_matrix2[0]],
                           [Rekt_matrix1[1],Rekt_matrix1[0]]])
    return Rekt_final

def R_matrix(matrix,omega,beta):
    Rekt_matrix1, _ = reflect(matrix,omega,beta)
    Rekt_matrix2 = reflect3(matrix,omega,beta)
    Rekt_final = np.array([[Rekt_matrix2[1],Rekt_matrix2[0]],
                           [Rekt_matrix1[1],Rekt_matrix1[0]]])
    return Rekt_final
#####
def DMatrix(omega,beta,R1,R2,d1=d):
    k = Get_k(beta,omega)
    exp = np.exp(2*j*k*d1)
    D_inv = np.eye(2) - (R1.dot(R2))*exp
    D = np.linalg.inv(D_inv)
    return D
#####
def Xi_z(omega, beta, phi1, phi2,d1=d):
    k= Get_k(beta,omega)
    k_0 = omega/c
    p_tensor1 = PermittivityTensor(omega,'z',phi1)
    p_tensor2 = PermittivityTensor(omega,'z',phi2)
    matrix1 = A_Matrix(omega, beta, p_tensor1)
    # matrix1 = A_Matrix(omega, beta, np.eye(3))
    matrix2 = A_Matrix(omega, beta, p_tensor2)
    R1 = R_matrix2(matrix1,omega,beta)
    R2 = R_matrix2(matrix2,omega,beta)
    R1_star = np.conj(R1.T)
    R2_star = np.conj(R2.T)
    D       = DMatrix(omega, beta, R1, R2,d1)
    D_star  = np.conj(D.T)
    I = np.eye(2)
    if beta < k_0:
        Xi = (I - R2_star.dot(R2)).dot(D).dot(I - R1_star.dot(R1)).dot(D_star)
        return np.trace(Xi)
    else:
        exp = np.exp(-2*abs(k)*d1)
        Xi = (R2_star - R2).dot(D).dot(R1 - R1_star).dot(D_star)
        return  np.trace(Xi)*exp

def Xi(omega, beta, phi1, phi2):
    k= Get_k(beta,omega)
    k_0 = omega/c
    p_tensor1 = PermittivityTensor(omega,'z',phi1)
    p_tensor2 = PermittivityTensor(omega,'z',phi2)
    matrix1 = A_Matrix(omega, beta, p_tensor1)
    # matrix1 = A_Matrix(omega, beta, np.eye(3))
    matrix2 = A_Matrix(omega, beta, p_tensor2)
    R1 = R_matrix(matrix1,omega,beta)
    R2 = R_matrix(matrix2,omega,beta)
    R1_star = np.conj(R1.T)
    R2_star = np.conj(R2.T)
    D       = DMatrix(omega, beta, R1, R2)
    D_star  = np.conj(D.T)
    I = np.eye(2)
    if beta < k_0:
        Xi = (I - R2_star.dot(R2)).dot(D).dot(I - R1_star.dot(R1)).dot(D_star)
        return np.trace(Xi)
    else:
        exp = np.exp(-2*abs(k)*d)
        Xi = (R2_star - R2).dot(D).dot(R1 - R1_star).dot(D_star)
        return  np.trace(Xi)*exp
#####
```

4 × 4 Transfer Matrix Method

Here we are going to derive the Fresnel coefficients by taking the z axis as the principal axis, this can be generalized later on as per need. The permittivity tensor is modified accordingly as given in the above equation, for our rotated system. We take a TM wave in order to solve for the coefficients, this can be done using a TM wave as well. The EM fields can be written as:

$$H = U(z)\,exp\,(j\omega t - j\beta x)\ where\, U = (U_x, U_y, U_z)$$

And

$$E = j(\mu_0/\epsilon_0)^{0.5} S(z)\,exp\,(j\omega t - j\beta x)\,,\ where\, S = (S_x, S_y, S_z)$$

Where β is the wave vector component along the x axis. The wave vector along the z-axis is $k_z = \sqrt{k_0^2 - \beta^2}$ in vacuum. Now using the maxwell's equation we get

$$\frac{d}{dz}\begin{pmatrix} S_x \\ S_y \\ U_x \\ U_y \end{pmatrix} = k_0\mathbf{A}\begin{pmatrix} S_x \\ S_y \\ U_x \\ U_y \end{pmatrix}$$

Where, the coefficient matrix is

$$\mathbf{A} = \begin{bmatrix} jK_x\varepsilon_{zx}/\varepsilon_{zz} & jK_x\varepsilon_{zy}/\varepsilon_{zz} & 0 & K_x^2/\varepsilon_{zz} - 1 \\ 0 & 0 & 1 & 0 \\ \varepsilon_{yz}\varepsilon_{zx}/\varepsilon_{zz} - \varepsilon_{yx} & \varepsilon_{yz}\varepsilon_{zy}/\varepsilon_{zz} + K_x^2 - \varepsilon_{yy} & 0 & -jK_x\varepsilon_{yz}/\varepsilon_{zz} \\ \varepsilon_{xx} - \varepsilon_{xz}\varepsilon_{zx}/\varepsilon_{zz} & \varepsilon_{xy} - \varepsilon_{xz}\varepsilon_{zy}/\varepsilon_{zz} & 0 & jK_x\varepsilon_{xz}/\varepsilon_{zz} \end{bmatrix}$$

We calculate the reflection and transmission coefficients by matching the tangential electric and magnetic component at the top of the surface.

$$\begin{pmatrix} -jk_z/k_0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} jk_z/k_0 & 0 \\ 0 & -j \\ 0 & k_z/k_0 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} r_{pp} \\ r_{ps} \end{pmatrix} = \begin{bmatrix} \mathbf{W}_1 & \mathbf{W}_2\mathbf{X} \end{bmatrix}\begin{bmatrix} \mathbf{C}^+ \\ \mathbf{C}^- \end{bmatrix}$$

Where W = [W1 W2] is the eigen vector matrix of A . X is a diagonal matrix with the diagonal elements as exp($-k_0 q_m$d);m=3,4 and Y is a diagonal matrix with the diagonal elements as exp($-k_0 q_m$d);m=1,2 .

Solving the same way for TE waves gives the equation given below,

$$\begin{pmatrix} 0 \\ 1 \\ -jk_z/k_0 \\ 0 \end{pmatrix} + \begin{pmatrix} -k_z/k_0 & 0 \\ 0 & 1 \\ 0 & jk_z/k_0 \\ j & 0 \end{pmatrix}\begin{pmatrix} r_{sp} \\ r_{ss} \end{pmatrix} = [\mathbf{W}_1 \quad \mathbf{W}_2\mathbf{X}]\begin{bmatrix} \mathbf{C}^+ \\ \mathbf{C}^- \end{bmatrix}$$

To preempt the numerical instability associated with the inversion of the matrix, we propose to adopt the enhanced transmittance matrix approach, for a general L layer system.

$$\begin{bmatrix} \mathbf{C}_{(L)}^+ \\ \mathbf{C}_{(L)}^- \end{bmatrix} = \begin{bmatrix} \mathbf{W}_{(L)|}\mathbf{Y}_{(L)} & \mathbf{W}_{(L)2} \end{bmatrix}^{-1}\begin{pmatrix} \mathbf{f}_{L+1} \\ \mathbf{g}_{L+1} \end{pmatrix}\mathbf{t}$$

$$\mathbf{t} = \begin{pmatrix} t_{pp} \\ t_{ps} \end{pmatrix}, \mathbf{f}_{L+1} = \begin{pmatrix} -jk_z/k_0 & 0 \\ 0 & -j \end{pmatrix}, \mathbf{g}_{L+1} = \begin{pmatrix} 0 & -k_z/k_0 \\ 1 & 0 \end{pmatrix}$$

We can now introduce parameters aL and bL to help us solve further.

$$\begin{pmatrix} \mathbf{a}_L \\ \mathbf{b}_L \end{pmatrix} = \begin{bmatrix} \mathbf{W}_{(L)1} & \mathbf{W}_{(L)2} \end{bmatrix}^{-1}\begin{pmatrix} \mathbf{f}_{L+1} \\ \mathbf{g}_{L+1} \end{pmatrix}$$

The iterative expression for fL and gL can be written as

$$\begin{pmatrix} \mathbf{f}_L \\ \mathbf{g}_L \end{pmatrix} = \mathbf{W}_{(L)1} + \mathbf{W}_{(L)2}\mathbf{X}_{(L)}\mathbf{b}_L\mathbf{a}_L^{-1}\mathbf{Y}_{(L)}$$

Using the iterative expression to solve for f1 and g1 , we now use these values to determine the final Fresnel coefficients using the equation

$$\begin{pmatrix} -jk_z/k_0 \\ 0 \\ 0 \\ 1 \end{pmatrix} + \begin{pmatrix} jk_z/k_0 & 0 \\ 0 & -j \\ 0 & k_z/k_0 \\ 1 & 0 \end{pmatrix}\mathbf{r} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{g}_1 \end{pmatrix}\mathbf{t}_1, \mathbf{r} = \begin{pmatrix} r_{pp} \\ r_{ps} \end{pmatrix}$$

We now repeat the same steps for TE waves in order to determine the values of r_{ss} & r_{sp}. Thus we finally have the values of required fresnel coefficients in order to proceed with our plotting.

In [7]:
```python
# transfer matrix
def A_Matrix(omega, beta, p_tensor):
    k_0 = omega/c
    K_x = beta/k_0
    A00 = j*K_x*p_tensor[2,0]/p_tensor[2,2]
    A01 = j*K_x*p_tensor[2,1]/p_tensor[2,2]
    A02 = 0
    A03 = (K_x**2)/p_tensor[2,2] - 1
    A10 = 0
    A11 = 0
    A12 = 1
    A13 = 0
    A20 = p_tensor[1,2]*p_tensor[2,0]/p_tensor[2,2] - p_tensor[1,0]
    A21 = p_tensor[1,2]*p_tensor[2,1]/p_tensor[2,2]-p_tensor[1,1]+K_x**2
    A22 = 0
    A23 = -j*K_x*p_tensor[1,2]/p_tensor[2,2]
    A30 = p_tensor[0,0] - p_tensor[0,2]*p_tensor[2,0]/p_tensor[2,2]
    A31 = p_tensor[0,1] - p_tensor[0,2]*p_tensor[2,1]/p_tensor[2,2]
    A32 = 0
    A33 = j*K_x*p_tensor[0,2]/p_tensor[2,2]
    A   = [[A00,A01,A02,A03],
           [A10,A11,A12,A13],
           [A20,A21,A22,A23],
           [A30,A31,A32,A33]]
    A   = np.array(A)
    return A
```

In [8]:
```python
# eigen values and eigen vectors of the transfer matrix
def eigen(matrix):
    eigenvalues, eigenvectors = np.linalg.eig(matrix)
    index_vals = [0,0,0,0]
    i,j = 0,2
    for k in range(4):
        if eigenvalues[k].real < 0:
            index_vals[i] = k
            i+=1
        else:
            index_vals[j] = k
            j+=1
    neweig = np.array([])
    newvec = []
    for index in index_vals:
        neweig = np.append(neweig,eigenvalues[index])
        newvec.append(eigenvectors[index])
    return neweig, np.array(newvec)
# equation 35 in reference 33
def albl(matrix, omega, beta):
    k_0 = omega/c
    k_z = np.sqrt((k_0**2 - beta**2)*(-j**2))
    flgl_ = np.array([[-j*k_z/k_0,0],
                      [0,-j],
                      [0,-k_z/k_0],
                      [1,0]])
    q_vec, W_matrix = eigen(matrix)
    # W1 = W_matrix[:2,:]
    # W2 = W_matrix[2:,:]
    W_inv = np.linalg.inv(W_matrix.T)
    albl = W_inv.dot(flgl_)
    al = albl[:2,:]
    bl = albl[2:,:]
    return al,bl
#equation 38 in reference 33
def flgl(matrix,omega, beta):
    k_0 = omega/c
    q_vals, W_matrix = eigen(matrix)
    W1 = W_matrix[:2,:].T
    W2 = W_matrix[2:,:].T
    X  = np.array([[np.exp(-k_0*q_vals[2]*d),0],
                   [0,np.exp(-k_0*q_vals[3]*d)]])
    Y  = np.array([[np.exp(k_0*q_vals[0]*d),0],
                   [0,np.exp(k_0*q_vals[1]*d)]])
    al_,bl_ = albl(matrix, omega, beta)
    if (np.linalg.det(al_)==0):
        al_ = np.identity(2)
    al_inv = np.linalg.inv(al_)
    flgl_ = W1 + W2.dot(X).dot(bl_).dot(al_inv).dot(Y)
    return flgl_
#equation 39, ref 33
def reflect(matrix,omega,beta):
    k_0 = omega/c
    k_z = np.sqrt((k_0**2 - beta**2)*(-j**2))
    flgl_ = flgl(matrix,omega, beta)
    r_coeff_neg = np.array([[-j*k_z/k_0,0],
                            [0,j],
                            [0,-k_z/k_0],
                            [-1,0]])
    coeff_matrix = np.concatenate((r_coeff_neg,flgl_),axis=1)
    if (np.linalg.det(coeff_matrix)==0):
        coeff_matrix = np.identity(4)
#   print(coeff_matrix)
    b   = np.array([-j*k_z/k_0, 0, 0, 1])
    x   = np.linalg.solve(coeff_matrix, b)
    r_ppps = x[:2]
    t_1 = x[2:]
    return r_ppps, t_1

# equation 36, ref 33
def CpCm(matrix,omega,beta):
    k_0 = omega/c
    q_vals, W_matrix = eigen(matrix)
    Y  = np.array([[np.exp(k_0*q_vals[0]*d),0],
                   [0,np.exp(k_0*q_vals[1]*d)]])
    al_,bl_ = albl(matrix, omega, beta)
    if (np.linalg.det(al_)==0):
        al_ = np.identity(2)
    al_inv = np.linalg.inv(al_)
    b_la_inv_Y = bl_.dot(al_inv).dot(Y)
    t_l_coeff = np.concatenate((np.eye(2),b_la_inv_Y))
    r_pp,t_1 = reflect(matrix,omega,beta)
    CpCm_out = t_l_coeff.dot(t_1)
    return CpCm_out

# equation 27, ref 33
def reflect2(matrix,omega,beta):
    k_0 = omega/c
    k_z = np.sqrt((k_0**2 - beta**2)*(-j**2))
    q_vals, W_matrix = eigen(matrix)
    X  = np.array([[np.exp(-k_0*q_vals[2]*d),0],
                   [0,np.exp(-k_0*q_vals[3]*d)]])
    W1 = W_matrix[:2,:].T
    W2 = W_matrix[2:,:].T
    W2X = W2.dot(X)
    W1W2X = np.concatenate((W1,W2X),axis =1)
    CpCm_ = CpCm(matrix,omega,beta)
    W1W2XCpCm = W1W2X.dot(CpCm_)
    const = np.array([0, 1, -j*k_z/k_0,0])
    Sub =  W1W2XCpCm - const
    coeff_rspss = np.array([[-k_z/k_0,0],
                            [0,1],
                            [0,j*k_z/k_0],
                            [j,0]])
    # reminder
    rspss1 = np.linalg.solve(coeff_rspss[:2,:],Sub[:2])
    rspss2 = np.linalg.solve(coeff_rspss[2:,:],Sub[2:])

    return  rspss1


######
def reflect3(matrix,omega,beta):
    k_0 = omega/c
    k_z = np.sqrt((k_0**2 - beta**2)*(-j**2))
    #q_vals, W_matrix = eigen(matrix)
    #X  = np.array([[np.exp(-k_0*q_vals[2]*d),0],
                   #[0,np.exp(-k_0*q_vals[3]*d)]])
    #W1 = W_matrix[:2,:].T
    #W2 = W_matrix[2:,:].T
    #W2X = W2.dot(X)
    #W1W2X = np.concatenate((W1,W2X),axis =1)
    #CpCm_ = CpCm(matrix,omega,beta)
    #W1W2XCpCm = W1W2X.dot(CpCm_)
    rppps, _ = reflect(matrix,omega,beta)
    const = np.array([0, 1, -j*k_z/k_0,0])
    #Sub =  W1W2XCpCm - const
    coeff_rspss = np.array([[-k_z/k_0,0],
                            [0,1],
                            [0,j*k_z/k_0],
                            [j,0]])
    const2= np.array([-j*k_z/k_0, 0, 0, 1])
    coeff_rppps = np.array([[j*k_z/k_0,0],
                            [0,-j],
                            [0,k_z/k_0],
                            [1,0]])
    rhs= coeff_rspss.dot(rppps) + const2
    Sub= rhs - const

    # reminder
    rspss1 = np.linalg.solve(coeff_rspss[:2,:],Sub[:2])
    rspss2 = np.linalg.solve(coeff_rspss[2:,:],Sub[2:])

    return  rspss2

def albl2(matrix, omega, beta):
    k_0 = omega/c
    k_z = np.sqrt((k_0**2 - beta**2)*(-j**2))
    flgl_ = np.array([[k_z/k_0,0],
                      [0,1],
                      [0,-j*k_z/k_0],
                      [j,0]])
    q_vec, W_matrix = eigen(matrix)
    # W1 = W_matrix[:2,:]
    # W2 = W_matrix[2:,:]
    W_inv = np.linalg.inv(W_matrix.T)
    albl = W_inv.dot(flgl_)
    al = albl[:2,:]
    bl = albl[2:,:]
    return al,bl

#equation 38 in reference 33
def flgl2(matrix,omega, beta):
    k_0 = omega/c
    q_vals, W_matrix = eigen(matrix)
    W1 = W_matrix[:2,:].T
    W2 = W_matrix[2:,:].T
    X  = np.array([[np.exp(-k_0*q_vals[2]*d),0],
                   [0,np.exp(-k_0*q_vals[3]*d)]])
    Y  = np.array([[np.exp(k_0*q_vals[0]*d),0],
                   [0,np.exp(k_0*q_vals[1]*d)]])
    al_,bl_ = albl2(matrix, omega, beta)
    if (np.linalg.det(al_)==0):
        al_ = np.identity(2)
    al_inv = np.linalg.inv(al_)
    flgl_ = W1 + W2.dot(X).dot(bl_).dot(al_inv).dot(Y)
    return flgl_

#equation 39, ref 33

def reflect4(matrix,omega,beta):
    k_0 = omega/c
    k_z = np.sqrt((k_0**2 - beta**2)*(-j**2))
    flgl_ = flgl2(matrix,omega, beta)
    r_coeff_neg = np.array([[k_z/k_0,0],
                            [0,-1],
                            [0,-j*k_z/k_0],
                            [-j,0]])
    coeff_matrix = np.concatenate((r_coeff_neg,flgl_),axis=1)
    if (np.linalg.det(coeff_matrix)==0):
        coeff_matrix = np.identity(4)
#   print(coeff_matrix)
    b   = np.array([0, 1, -j*k_z/k_0, 0])
    x   = np.linalg.solve(coeff_matrix, b)
    r_spss = x[:2]
    t_1 = x[2:]
    return r_spss, t_1
```

Heatmaps

Principal axis - z ,ω, $\gamma(0, 90)$

The first 2 plots show the energy transmission coefficient n between the emitter and the receiver varying with the wavevector components kx and ky at x = 1.1310×10^{14} rad/s when the rotation angle γ is equal to 0 deg and 90 deg. In the plots k_x and k_y represent the projection of the β on x and y axis.

The conditions for occurance of HPPs in $\alpha - MoO_3$ is given by

$$\frac{(\epsilon_y k_y^2 + \epsilon_x k_x^2)}{\epsilon_z} < 0$$

Here it is assumed that $k_y >> k_0$ and $k_x >> k_0$, and the imaginary parts of ϵ_x, ϵ_y, and ϵ_z are neglected. Specifically, HPPs can occur in the region

$$-\sqrt{-\frac{\epsilon_x}{\epsilon_y}} < \frac{k_y}{k_x} < \sqrt{-\frac{\epsilon_x}{\epsilon_y}}$$

when ϵ_z > 0 and ϵ_y > 0; ϵ_x < 0, and in the region

$$\frac{k_y}{k_x} > \sqrt{-\frac{\epsilon_x}{\epsilon_y}}\ \&\ \frac{k_y}{k_x} < -\sqrt{-\frac{\epsilon_x}{\epsilon_y}}$$

when ϵ_z > 0 and ϵ_y < 0; ϵ_x > 0. Therefore, the regions in the k_x– k_y plane where HPPs can occur are bounded by the curves defined as

$$\frac{k_y}{k_x} = \pm\sqrt{-\frac{\epsilon_x}{\epsilon_y}}$$

In [9]:
```python
n = 200
def plot_heatmap_Z(omega_no,gamma_no,phno,n):
    omega_vals = [1.131*10**14, 1.4137*10**14, 1.8228*10**14]
    omega = omega_vals[omega_no]
    gamma_vals = [0,np.pi/2]
    gamma = gamma_vals[gamma_no]
    x_axis = np.linspace(-100,100,n)
    y_axis = np.linspace(-100,100,n)
    x,y = np.meshgrid(x_axis, y_axis)
    image = np.zeros((n,n))
    for p in range(len(x)):
        for q in range(len(y)):
            k_0   = omega /c
            kx    = x_axis[p]*k_0
            ky    = y_axis[q]*k_0
            beta = np.sqrt(kx**2 + ky**2)
            phi = [np.arctan2(ky,kx),np.arctan2(ky+kx,ky-kx)]
            image[q,p] = abs(Xi_z(omega, beta,phi[phno]+gamma,phi[phno]))
    #np.save(f"Zaxisomega{omega_no}gamma{gamma_no*90}", image)
    plt.xlabel("Dimensionless Vector $k_x/k_0$")
    plt.ylabel(" Dimensionless Vector $k_y/k_0$")
    plt.title("$Xi(\omega,k_x,k_y)$")
    plt.imshow(image, cmap = 'hot', extent=[-100,100,-100,100])
    plt.colorbar()
```

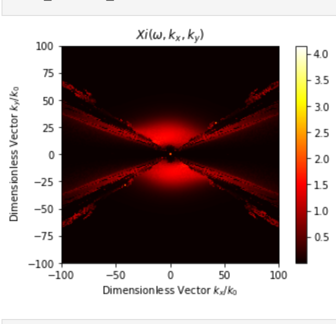
The results for ϵ_z < 0 can be similarly obtained.At x = 1.1310×10^{14} rad/s, the values of the principal relative permittivity components are calculated to be $\epsilon_x = 7.4874 + j0.0268$, $\epsilon_y = □30.0218 + j1.3423$, and $\epsilon_z = 2.7883 + j0.0008$.

Uponn plottign these values, we can clearly see that the bounding curves in the kx–ky plane of the regions for HPPs can be obtained as $k_y = \pm 0.5k_x$ which correspond to the two asymptotes visible.

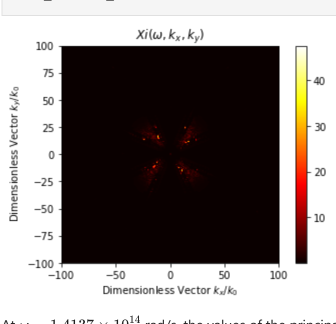
It can be seen that these two lines bound the regions with large heat flux, indicated by the bright color. In addition, because the angle formed by the two bounding curves is equal to 126.9 deg, i.e., greater than 90 deg, the regions for HPPs of the emitter and those of the receiver can partially overlap with each other when $\gamma = 90^o$.

For the γ = 90-degree case, the transmission coefficient is highly dependent on the azimuthal angle(ϕ). The overall heat flux decreases as compared to that of the one when γ is 0. This can be explained directly by the misalignment of HPPs between the emitter and the receiver.

In [10]:
```python
#  fig z.1
plot_heatmap_Z(0,0,0,n)
```

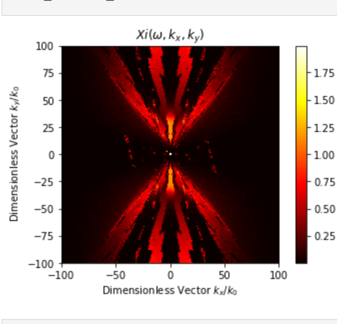


In [11]:
```python
#fig z.2
plot_heatmap_Z(0,1,0,n)
```

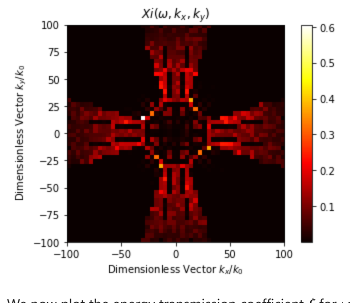


At $\omega = 1.4137 \times 10^{14}$ rad/s, the values of the principal relative permittivity components are $\epsilon_x = 13.9065 + j0.2704$, $\epsilon_y = 3.1663 + j0.0945$ and $\epsilon_z = 3.0097 + j0 : 0026$. Therefore, the bounding curves in the k_x– k_y plane of the regions for HPPs can be obtained as $k_y = \pm 2.1k_x$, which corresponds to the boundaries of the curves observed The bright colour in the regions can be explained in a similar way as we do in the earlier scenario.

In [12]:
```python
#fig z.3
plot_heatmap_Z(1,0,0,n)
```

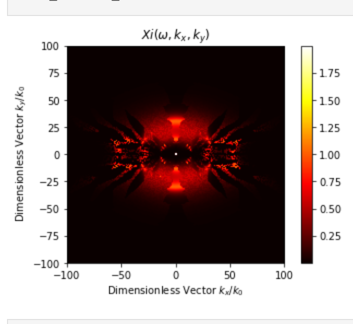


In [13]:
```python
#fig z.4
plot_heatmap_Z(1,1,1,50)
```

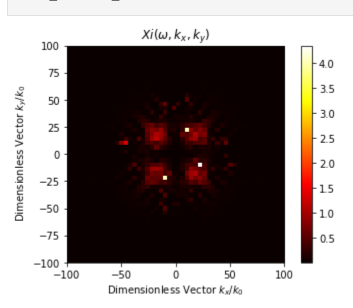


We now plot the energy transmission coefficient ξ for $\omega = 1.8228 \times 10^{14}$ for γ = 0 and 90 degrees, At these values of angular frequency we get the principal relative permittivity components as $\epsilon_x = 0.1467 + j0.0611$, $\epsilon_y = 1.7188 + j0.0211$ and $\epsilon_z = 9:9486 + j1.3785$. In this case we get the bounding curve equation to be $k_y = \pm 0.29 k_x$.

In [14]:
```
#fig z.5
plot_heatmap_Z(2,0,0,n)
```



In [15]:
```
#fig z.6
plot_heatmap_Z(2,1,0,50)
```



These plots are little different as compared to the earlier plots because of the excitation of HPPs. However, there are a few peculiarities here that we should make note of, there is no absolute darkness in the off regions here, that cannot be attributed directly to HPPs, but is explained using resonance of Dyakonov waves. These are surface waves which get excited at interface between an isotropic medium and an anisotropic medium. The necessary condition for existence of Dyakonov waves at an interface is

$$\epsilon_1 > \epsilon_d > \epsilon_2 > \epsilon_3$$

Where ϵ_d is the relative permittivity of isotropic medium (=1 here). One of the conditions for existence is also that the the principal axis with highest relative permittivity should be parallel to the interface. Dyakonov waves have been termed as HSPhPs, as their dispersion exhibits hyperbolic properties. The dispersion curves have asymptotes in the form

$$\frac{k_y}{k_x} = \pm\sqrt{-\frac{\epsilon_x\epsilon_z - \epsilon_d^2}{\epsilon_y\epsilon_z - \epsilon_d^2}}$$

.

Neglecting the imaginary parts, this can be approximated to $k_y = \pm 0.16 k_x$, which can be seen in the plot.

In [16]:
```
def PermittivityTensorx(omega, axis ='x' , angle = 0):
  eps_diag = Diagonal_Dielectric_Tensor(omega)
  R = RotationMatrix(axis, angle)
#   R_inv = np.linalg.inv(R)
  return R.dot(eps_diag).dot(R.T)
```

In [17]:
```
# transfer matrix X
def Ax_Matrix(omega, beta, p_tensor):
    k_0 = omega/c
    K_x = beta/k_0
    A00 = j*K_x*p_tensor[0,1]/p_tensor[0,0]
    A01 = j*K_x*p_tensor[0,2]/p_tensor[0,0]
    A02 = 0
    A03 = (K_x**2)/p_tensor[0,0] - 1
    A10 = 0
    A11 = 0
    A12 = 1
    A13 = 0
    A20 = p_tensor[2,0]*p_tensor[0,1]/p_tensor[0,0] - p_tensor[2,1]
    A21 = p_tensor[2,0]*p_tensor[0,2]/p_tensor[0,0]-p_tensor[2,2]+K_x**2
    A22 = 0
    A23 = -j*K_x*p_tensor[2,0]/p_tensor[0,0]
    A30 = p_tensor[1,1] - p_tensor[1,0]*p_tensor[0,1]/p_tensor[0,0]
    A31 = p_tensor[1,2] - p_tensor[1,0]*p_tensor[0,2]/p_tensor[0,0]
    A32 = 0
    A33 = j*K_x*p_tensor[1,0]/p_tensor[0,0]
    A   = [[A00,A01,A02,A03],
          [A10,A11,A12,A13],
          [A20,A21,A22,A23],
          [A30,A31,A32,A33]]
    A   = np.array(A)
    return A
```

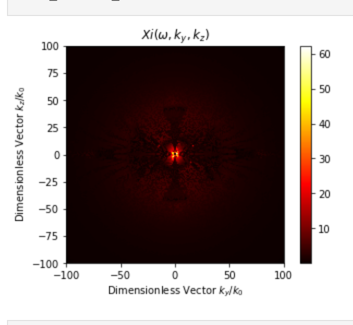
In [18]:
```
def Xi_x(omega, beta, phi1, phi2):
    k= Get_k(beta,omega)
    k_0 = omega/c
    p_tensor1 = PermittivityTensorx(omega,'x',phi1)
    p_tensor2 = PermittivityTensorx(omega,'x',phi2)
    matrix1 = Ax_Matrix(omega, beta, p_tensor1)
    # matrix1 = A_Matrix(omega, beta, np.eye(3))
    matrix2 = Ax_Matrix(omega, beta, p_tensor2)
    R1 = R_matrix(matrix1,omega,beta)
    R2 = R_matrix(matrix2,omega,beta)
    R1_star = np.conj(R1.T)
    R2_star = np.conj(R2.T)
    D       = DMatrix(omega, beta, R1, R2)
    D_star  = np.conj(D.T)
    I = np.eye(2)
    if beta < k_0:
        Xi = (I - R2_star.dot(R2)).dot(D).dot(I - R1_star.dot(R1)).dot(D_star)
        return np.trace(Xi)
    else:
        exp = np.exp(-2*abs(k)*d)
        Xi = (R2_star - R2).dot(D).dot(R1 - R1_star).dot(D_star)
        return  np.trace(Xi)*exp
```

Principal axis - X ,ω, $\gamma(0^o, 90^o)$

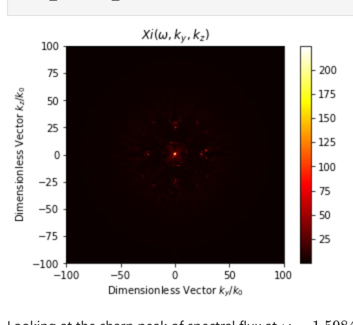
In this case, the enhanced spectral flux in the spectral region is from 1.0327×10^{14} to 1.5457×10^{14} rad/s and the drastic drop of the flux in the spectral region from 1.4137×10^{14} to 1.5268×10^{14} rad/s when $\gamma = \pi/2$. The mechanism is similar to plots for first 4 plots of $[010]$ pricipal axis case.

In [19]:
```
def plot_heatmap_X(omega_no,gamma_no,n):
    omega_vals = [1.5984*10**14, 1.8228*10**14, 1.8171*10**14, 1.8246*10**14]
    omega = omega_vals[omega_no]
    gamma_vals = [0,np.pi/2]
    gamma = gamma_vals[gamma_no]
    x_axis = np.linspace(-100,100,n)
    y_axis = np.linspace(-100,100,n)
    x,y = np.meshgrid(x_axis, y_axis)
    image = np.zeros((n,n))
    for p in range(len(x)):
        for q in range(len(y)):
            k_0  = omega /c
            kx   = x_axis[p]*k_0
            ky   = y_axis[q]*k_0
            beta = np.sqrt(kx**2 + ky**2)
            phi = np.arctan2(ky,kx)
            image[q,p] = abs(Xi_x(omega, beta,phi+gamma,phi))
    # np.save(f"Xaxisomega{omega_no}gamma{gamma_no*90}", image)
    plt.xlabel("Dimensionless Vector $k_y/k_0$")
    plt.ylabel(" Dimensionless Vector $k_z/k_0$")
    plt.title("$Xi(\omega,k_y,k_z)$")
    plt.imshow(image, cmap = 'hot', extent=[-100,100,-100,100])
    plt.colorbar()
```

In [20]:
```
#fig x.1
plot_heatmap_X(0, 0, n)
```



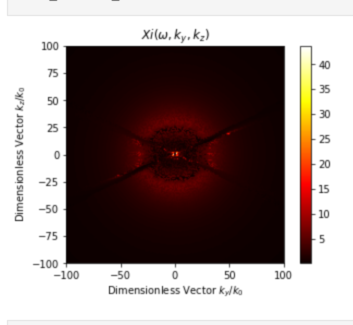
In [21]:
```
#fig x.2
plot_heatmap_X(0, 1, n)
```



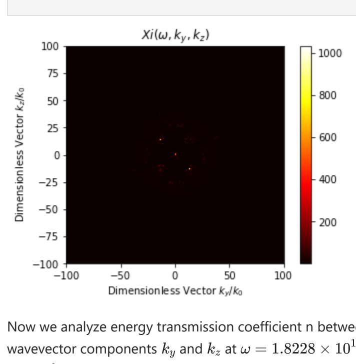
Looking at the sharp peak of spectral flux at $\omega = 1.5984 \times 10^{14}$ rad/s, at which the energy transmission coefficient n varying with the wavevector components k_y and k_z, the values of the principal relative permittivity components are $\epsilon_x = 19.2061 + j1.6854$, $\epsilon_y = -0.0625 + j0.0423$, and $\epsilon_z = 3.4902 + j0.0093$. Thus using the same equation as in $[010]$ pricipal axis case with suitable adjusments we find that the enhanced heat flux represented by the bright color in the regions above and below the origin shown in fig (x.1), which are bounded by the two white dashed lines defined as $k_z = \pm 0.13 k_y$, is due to excitation of HPPs. As for the enhanced heat flux seen in the regions on the left and right sides of the origin, indicated by the brighter color, it is due to excitation of HSPhPs, similar to the case in Fig.(z.5). (The argument is well supported by the dispersion curves as in fig (x.1) by solid lines whose equations are given as $k_z = \pm 0.054 k_y$). By having $\gamma = 90^o$ the corresponding fig indicates the misalignment of the HPPs and HSPhPs between the emitter and the receiver causes the enhanced heat flux to be smaller than the 0^o case.

It should be noted to this end that HSPhPs cannot be excited at $\omega = 1.5984 \times 10^{14}$ rad/s when the heat flux is along the $[010]$ direction, since the principal axis corresponding to ϵ_1 (ϵ_z in this case) is not parallel to the k_x– k_y plane. Therefore, the condition for Dyakonov waves is not satisfied.

In [22]:
```
#fig x.3
plot_heatmap_X(1, 0, n)
```

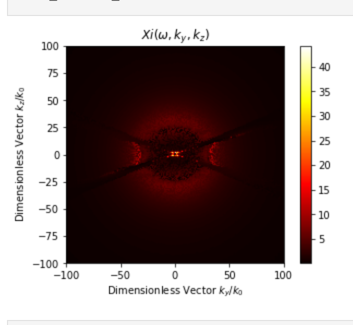


In [23]:
```
#fig x.4
plot_heatmap_X(1, 1, n)
#to be squared
```

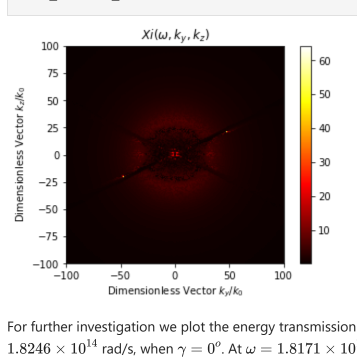


Now we analyze energy transmission coefficient n between the emitter and the receiver varying with the wavevector components k_y and k_z at $\omega = 1.8228 \times 10^{14}$ rad/s when the rotation angle c is equal to 0^o and 90^o , respectively.The enhanced heat flux in the regions on the left and right sides of the origin, indicated by the bright color in first figure above, comes from excitation of HPPs which occur in the regions of $-0.42 < k_z = k_y < 0.42$. Apart from HPPs, excitation of Dyakonov waves is responsible for the greatly enhanced heat flux indicated by the brighter color in the regions above and below the origin. For $\gamma = 90^o$ greatly enhanced heat flux is seen in the four overlapped regions for the surface resonant modes, while the regions for enhanced heat flux due to HPPs can be seen to extend to much larger values of k_y and k_z which may be caused by the interaction between HPPs and HSPhPs

In [24]:
```
#fig x.5
plot_heatmap_X(2, 0, n)
```



In [25]:
```
#fig x.6
plot_heatmap_X(3, 0, n)
```



For further investigation we plot the energy transmission coefficient n for $\omega = 1.8171 \times 10^{14}$ and 1.8246×10^{14} rad/s, when $\gamma = 0^o$. At $\omega = 1.8171 \times 10^{14}$ rad/s, $\epsilon_x = -0.2402 + j0.0636$, $\epsilon_y = 1.6869 + j0.0214$ and $\epsilon_z = -15.8722 + j3.0549$. Excitation of HPPs and HSPhPs can be clearly seen, however, the distribution pattern is closer to the shape of a hyperbola which agrees better with the dispersion curves of HSPhPs. For $\omega = 1.8246 \times 10^{14}$ rad/s, $\epsilon_x = -0.1164 + j0.0602$, $\epsilon_y = 1.7293 + j0.0210$, and $\epsilon_z = -8.7339 + j1.1192$. distribution pattern of n in this case deviates even larger from the shape of a hyperbola and worse agreement is found when it is compared with the dispersion curves of HSPhPs, rhis can be explained by resonant modes being in the form of hybrid modes a transitional state from HSPhPs to ESPhPs (asymptotes given by- equation 12 of paper)

for ω going from 1.8171×10^{16} rad/s to 1.8246×10^{14} rad/s the value of the denominator $\epsilon_z\epsilon_x - \epsilon_d^2$ is getting closer and closer to zero

In [26]:
```
def PermittivityTensory(omega, axis ='y' , angle = 0):
  eps_diag = Diagonal_Dielectric_Tensor(omega)
  R = RotationMatrix(axis, angle)
#   R_inv = np.linalg.inv(R)
  return R.dot(eps_diag).dot(R.T)
```

In [27]:
```
# transfer matrix Y
def Ay_Matrix(omega, beta, p_tensor):
    k_0 = omega/c
    K_x = beta/k_0
    A00 = j*K_x*p_tensor[1,2]/p_tensor[1,1]
    A01 = j*K_x*p_tensor[1,0]/p_tensor[1,1]
    A02 = 0
    A03 = (K_x**2)/p_tensor[1,1] - 1
    A10 = 0
    A11 = 0
    A12 = 1
    A13 = 0
    A20 = p_tensor[0,1]*p_tensor[1,2]/p_tensor[1,1] - p_tensor[0,2]
    A21 = p_tensor[0,1]*p_tensor[1,0]/p_tensor[1,1]-p_tensor[0,0]+K_x**2
    A22 = 0
    A23 = -j*K_x*p_tensor[0,1]/p_tensor[1,1]
    A30 = p_tensor[2,2] - p_tensor[2,1]*p_tensor[1,2]/p_tensor[1,1]
    A31 = p_tensor[2,0] - p_tensor[2,1]*p_tensor[1,0]/p_tensor[1,1]
    A32 = 0
    A33 = j*K_x*p_tensor[2,1]/p_tensor[1,1]
    A   = [[A00,A01,A02,A03],
          [A10,A11,A12,A13],
          [A20,A21,A22,A23],
          [A30,A31,A32,A33]]
    A   = np.array(A)
    return A
```

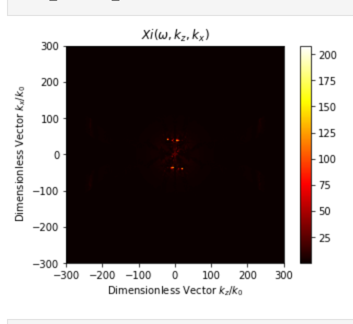
In [28]:
```
def Xi_y(omega, beta, phi1, phi2):
    k= Get_k(beta,omega)
    k_0 = omega/c
    p_tensor1 = PermittivityTensory(omega,'y',phi1)
    p_tensor2 = PermittivityTensory(omega,'y',phi2)
    matrix1 = Ay_Matrix(omega, beta, p_tensor1)
    # matrix1 = A_Matrix(omega, beta, np.eye(3))
    matrix2 = Ay_Matrix(omega, beta, p_tensor2)
    R1 = R_matrix(matrix1,omega,beta)
    R2 = R_matrix(matrix2,omega,beta)
    R1_star = np.conj(R1.T)
    R2_star = np.conj(R2.T)
    D       = DMatrix(omega, beta, R1, R2)
    D_star  = np.conj(D.T)
    I = np.eye(2)
    if beta < k_0:
        Xi = (I - R2_star.dot(R2)).dot(D).dot(I - R1_star.dot(R1)).dot(D_star)
        return np.trace(Xi)
    else:
        exp = np.exp(-2*abs(k)*d)
        Xi = (R2_star - R2).dot(D).dot(R1 - R1_star).dot(D_star)
        return  np.trace(Xi)*exp
```

Principal Axis- Y, ω,$\gamma(0, 90)$

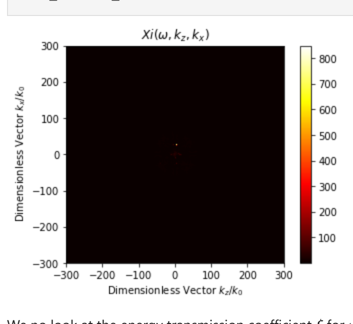
For $\omega = 1.3195 \times 10^{14}$ rad/s when the rotation angle γ is equal to 0 deg and 90 deg. At this angular frequency, the values of the principal relative permittivity components are $\epsilon_x = 9.9719 + j0.0917$, $\epsilon_y = 6.3085 + j0.1670$ and $\epsilon_z = 2.9064 + j0.0017$. The minimal difference between the plots for 0 degree and 90 degrees can be attributed to the negligible interference of HPPs. We get a region bounded by an ellipse.

In [29]:
```
def plot_heatmap_Y(omega_no,gamma_no,n):
    omega_vals = [1.3195*10**14, 1.5984*10**14]
    omega = omega_vals[omega_no]
    gamma_vals = [0,np.pi/2]
    gamma = gamma_vals[gamma_no]
    x_axis = np.linspace(-300,300,n)
    y_axis = np.linspace(-300,300,n)
    x,y = np.meshgrid(x_axis, y_axis)
    image = np.zeros((n,n))
    for p in range(len(x)):
        for q in range(len(y)):
            k_0  = omega /c
            kx   = x_axis[p]*k_0
            ky   = y_axis[q]*k_0
            beta = np.sqrt(kx**2 + ky**2)
            phi = np.arctan2(ky,kx)
            image[q,p] = abs(Xi_x(omega, beta,phi+gamma,phi))
    np.save(f"Yaxisomega{omega_no}gamma{gamma_no*90}", image)
    plt.xlabel("Dimensionless Vector $k_z/k_0$")
    plt.ylabel(" Dimensionless Vector $k_x/k_0$")
    plt.title("$Xi(\omega,k_z,k_x)$")
    plt.imshow(image, cmap = 'hot', extent=[-300,300,-300,300])
    plt.colorbar()
```

In [30]:
```
plot_heatmap_Y(0, 0, n)
```

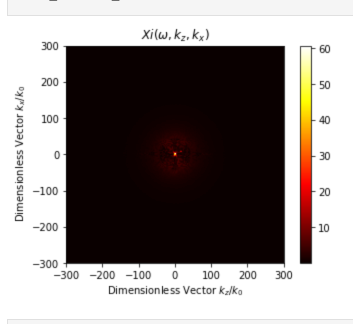


In [31]:
```
plot_heatmap_Y(0, 1, n)
```

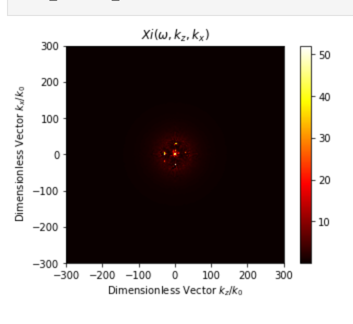


We no look at the energy transmission coefficient ξ for $\omega = 1.5984 \times 10^{14}$. We get similar plots as above, which clearly illustrate the occurrence of HPPs and HSPhPs, in which HSPhPs make dominant contribution to the heat flux enhancement. The results γ = 90 deg clearly illustrate enhanced heat flux due to resonance of HSPhPs and their interactions with HPPs.

In [32]:
```
plot_heatmap_Y(1,0, n)
```



In [33]:
```
plot_heatmap_Y(1, 1, n)
```



Fluctuation-Dissipation Theorem

When light impinges on an object, some fraction of the light is absorbed, making the object hotter. In this way, light absorption turns light energy into heat. The corresponding fluctuation is thermal radiation. Thermal radiation turns heat energy into light energy — the reverse of light absorption. This correlation between Thermal Properties and corresponding Electromagnetic phenomenon is given by the Fluctuation-Dissipation Theorem. The free electric currents J are associated with fluctuating charges in the medium of interest and satisfy the statistical correlation function given by the fluctuation-dissipation theorem as follows :

$$\left\langle J_l(r,\omega)J_m^*(r',\omega)\right\rangle = \frac{4}{\pi}\epsilon_0\epsilon(\omega)\omega\Theta_0(\omega,T)\delta_{lm}\delta(r-r')$$

where,

$$\Theta_0(\omega,T) = \hbar\omega\left(\frac{1}{2} + \frac{1}{exp(\frac{\hbar\omega}{k_BT}) - 1}\right)$$

is the energy of a harmonic oscillator

Thermal Radiation in Parallel-Plane Systems

Based on the Fluctuation-Dissipation Theorem and the reciprocity of the dyadic Green's Functions, the Near Field Radiative Heat Transfer between media 1 and media 2 can be expressed as:

$$Q = \frac{1}{8\pi^3}\int_0^\infty[\Theta(\omega,T_1) - \Theta(\omega,T_2)]d\omega\int_0^{2\pi}\int_0^\infty\xi(\omega,\beta,\phi)\beta d\beta d\phi$$

$$\Theta(\omega,T) = \frac{\hbar\omega}{exp(\frac{\hbar\omega}{k_BT}) - 1}$$

$\xi(\omega,\beta,\phi)$ is called the energy transmission coefficient or the phonon tunneling probability. It is expressed as: \ If $\beta < k_0$:

$$\xi(\omega,\beta,\phi) = Tr[(I - R_2^\dagger R_2)D(I - R_1^\dagger R_1)D^\dagger]$$

If $\beta > k_0$:

$$\xi(\omega,\beta,\phi) = Tr[(R_2^\dagger - R_2)D(R_1 - R_1^\dagger)D^\dagger]e^{-2|k|d}$$

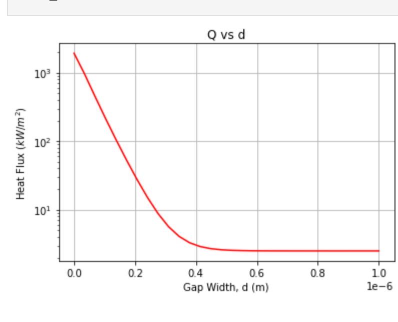
where, $k_0 = \omega/c$ and $k = \sqrt{(k_0^2 - \beta^2)}$ is the perpendicular wavevector component in vacuum.

In [34]:
```
from scipy.integrate import simps, quad

big_inf = 10**20
def big_theta(omega,T):
    return h_bar*omega/(np.exp(h_bar*omega/(k_boltz*T))-1)

def nfrhf2(omega,d1,gamma):
    n = 100
    v = 0
    I = 0
    for i in range(0,n):
        v = v+1/n
        x = np.arange(1e-7,1,0.01)
        lst=[]
        for k in x:
            y = abs(Xi_z(omega,1/k - 1,2*np.pi*v+gamma,2*np.pi*v,d1))*(1/k-1)*(2*np.p
            lst.append(y)
        integral = simps(lst,x)
        I = I + integral/n

    integral2 = quad(lambda z: big_theta(omega,T1), 0, big_inf)
    return (1/(8*(np.pi)**3))*I*integral2[0]

omega  = 1.1310*10**14
nfrhf2(omega,6.42857143e-07,0)
```

Out[34]: 2.506368706042888

In [35]:
```
def plot_Qvd(num_pts,  omega  = 1.1310*10**14):
  x = np.linspace(0,1000*10**(-9),num_pts)
  lst=[]
  for k in x:
      #print(k)
      lst.append(nfrhf2(omega,k,gamma=0))
  plt.plot(x,lst,'r')
  plt.yscale('log')
  plt.xlabel('Gap Width, d (m)')
  plt.ylabel('Heat Flux $(kW/m^{2})$')
  plt.grid()
  plt.title("Q vs d")
```

In [36]:
```
plot_Qvd(30)
```

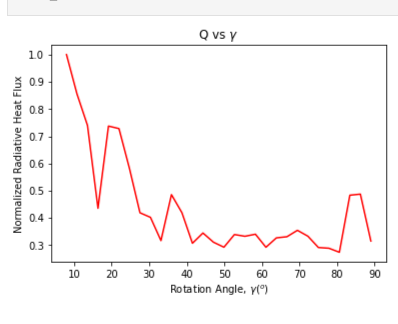


Q vs d

The plot of NFRHF as a function of distance, d between the $\alpha - MoO_3$ crystals aligned along the [010] direction is shown in the figure aboves. The value of NFRHF decreases monotonically with d. Huge heat flux along the crystalline direction [010] of the $\alpha - MoO_3$ crystals is observed when d is in the submicron regime. When d is equal to 20nm, the NFRHF along the [010] direction is found to be much greater than that for a uniaxial hBN crystal.

In [37]:
```
def plot_QvGamma(num_pts,omega  = 1.1310*10**14):
  x = np.linspace(8,89,num_pts)
  lst=[]
  for k in x:
      #print(k)
      lst.append(nfrhf2(omega,d,np.deg2rad(k)))
  lst= np.array(lst)
  m = lst.max()

  plt.plot(x,lst/m,'r')
  plt.xlabel('Rotation Angle, $\gamma (^{o})$')
  plt.ylabel('Normalized Radiative Heat Flux')
  plt.title("Q vs $\gamma$")
```

In [38]:
```
plot_QvGamma(30)
```

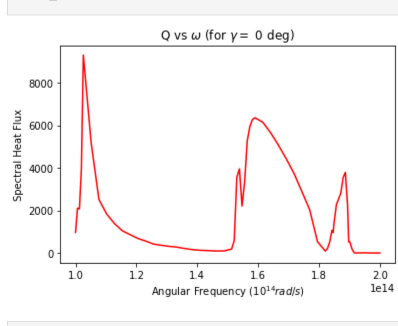


Q vs γ

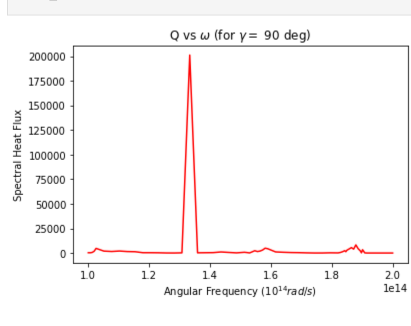
Here, we plot the NFRHF as a function of varying relative rotation angle γ for crystals aligned along [010] direction. As discussed earlier, the NFRHF can be manipulated by controlling the relative rotation angle between the emitter and receiver crystals of $\alpha - MoO_3$. It can be seen that the NFRHF decreases monotonically with increase in γ from 0^o to 90^o

In [39]:
```
def plot_QvOmega(num_pts,gamma_no):
    x = np.linspace(1e14,2e14,num_pts)
    x1 = np.linspace(x[0],x[1],5)
    x2 = np.linspace(x[20],x[23],10)
    x3 = np.linspace(x[32],x[36],15)
    x = np.concatenate((x,x1,x2,x3))
    x = np.sort(x)
    index = 0
    lst=[]
    gamma_val = [0, np.pi/2]
    gamma      = gamma_val[gamma_no]
    for k in x:
        val = nfrhf2(k,d,gamma)
        lst.append(val)
        index=index+1
        #print(index)
        #print(val)

    lst= np.array(lst)
    plt.plot(x,lst,'r')
    plt.xlabel('Angular Frequency ($10^{14} rad/s)$')
    plt.ylabel('Spectral Heat Flux')
    plt.title(f"Q vs $\omega$ (for $\gamma = $ {gamma_no*90} deg)")
```

In [40]:
```
plot_QvOmega(40,0)
```



In [41]:
```
plot_QvOmega(40,1)
```



Spectral Heat Flux vs ω

Here, we plot the heat flux as a function of varying ω, again for crystals aligned along [010] direction. It can be seen that for $\gamma = 0$, the heat flux is enhanced significantly in the three Restrahlen bands as obtained previously. A sharp peak is seen towards the end. When the rotation angle $\gamma = 90$, a smaller heat flux is seen as compared to the $\gamma = 0$ case.

The enhanced heat flux in the Restrahlen bands can be attributed to the excitation of Hyperbolic Phonon Polaritons. The sharp peaks however are said to be observed as a consequence of the Hyperbolic Surface Phonon Polaritons (HSPhPs).

Conclusion

In this executable paper, we have implemented numerically, and investigated Near Field Radiative Heat Transfer between two semi-infinite $\alpha - MoO_3$ biaxial crystals and the effect of crystal orientation on the same. This utilized the combination of Fluctuation-Dissipation Theorem and modified 4×4 Transfer Matrix Method to calculate Near Field Radiative Heat Flux (NFRHF) between biaxial crystals whose temperatures were set at $300K$ and $0K$, at a separation of $d = 20\,nm$. The numerical results show that the NFRHF is higher than that for hBN uniaxial crystals when the surfaces of the emitter and receiver are perpendicular to the $[001]$ crystalline direction. This greatly enhanced NFRHF is attributed to the excitation of Hyperbolic Phonon Polaritons (HPPs) inside $\alpha - MoO_3$ and Hyperbolic Surface Phonon Polaritons (HSPhPs) at the vacuum/$\alpha - MoO_3$ interfaces. It is also observed that NFRHF can be modulated by changing the relative rotation angle between the two biaxial crystals. The results verified in this work hint towards ways to manipulate near-field radiative transfer between anisotropic materials.

Bibliography

- Wu, X.; Fu, C.; Zhang, Z. **"Near-Field Radiative Heat Transfer Between Two α-MoO3 Biaxial Crystals"**. Journal of Heat Transfer 2020, 142, 1–10
- Xiaohu Wu, Ceji Fu, Zhuomin Zhang, **"Influence of hBN orientation on the near-field radiative heat transfer between graphene/hBN heterostructures"**, J. Photon. Energy 9(3), 032702 (2018), doi: 10.1117/1.JPE.9.032702.
- Zheng and et al., **"A mid-infrared biaxial hyperbolic van der Waals crystal"**
- AIP Advances 5, 053503 (2015); https://doi.org/10.1063/1.4919048
- Rousseau, E., Siria, A., Jourdan, G. et al. Radiative heat transfer at the nanoscale. Nature Photon 3, 514–517 (2009). https://doi.org/10.1038/nphoton.2009.144

Contributions

- Theoretical study by everyone
- Restrahlen Bands plots done by Vinit
- Figuring out the methodology and math of 4×4 matrix was done by Kaushik and Parth
- Implementation of ξ function and plotting of heat maps done by Kaushik, Parth and Vinit
- Implementation of Heat Flux function done by Shourish and Vinit
- Plotting of Variation of Heat flux with γ, d, ω done by Shourish