Fonones y espectroscopía Raman en semiconductores bidimensionales.

casimir

March 17, 2021

Contents

Abstract

Los materiales bidimensionales (2D) como el grafeno son de gran interés tanto por sus propiedades físicas exclusivas como por sus aplicaciones potenciales. El estudio de la dinámica de la red cristalina (fonones) de estos materiales es un requisito previo para entender su estabilidad estructural y propiedades térmicas, así como sus propiedades de transporte y ópticas.

Este Trabajo de Fin de Grado consiste en la computación de los modos vibracionales de materiales semiconductores 2D y su correlación con los observables relevantes para la interpretación de los experimentos de dispersión de luz.

La redacción del TFG todavía se encuentra en una versión muy preliminar, de hecho es más un bloc de notas que lo que espero que sea el TFG final.

Para escribir este trabajo estoy usando **org-mode**: un modo de edición del editor *Emacs*, que, entre otras muchas cosas, permite escribir en un mismo fichero tanto el *texto* cómo el *codigo* (en muchos lenguajes, no solo python, y permite también mezclar distintos lenguajes), pudiendo insertar si así se desea la salida de los programas en el mismo documento, y exportat el resultado final a varios formatos (en particular $\LaTeX$)

1 TODO Introduccion

Las vibraciones reticulares están regidas por las fuerzas que experimentan los átomos cuando se desplazan de su posición de equilibrio. La primera hipótesis es que cada átomo tiene una posicion de equilibrio en el cristal, y consideraremos que estos átomos vibran con una amplitud pequeña alrededor de su posición de equilibrio, de manera que el sólido se encuentra en estados que corresponden a lo que macroscópicamente se conoce como *la región de comportamiento elástico lineal*, donde se verifica la ley de Hooke.

Podremos por tanto aproximar la energía potencial de interacción por el termino armónico de su desarrolo en serie de potencias del desplazamiento, y la fuerza resultante es por tanto una función aproximadamente lineal del desplazamiento

1.1 TODO Matriz dinámica

La matriz dinámica es una cantidad central en la dinámica reticular: las frecuencias de los fonones (vibraciones reticulares cuantificadas) se calculan a partir de los valores propios de la matriz dinámica:

$$\sum_{\alpha\prime} D_{\alpha\alpha\prime} \cdot \vec{e}_{\alpha\prime}(\vec{q}) = \omega^2 \vec{e}_{\alpha}(\vec{q}) \tag{1}$$

Por lo tanto, las frecuencias ω como función de vector de ondas $\vec{q}$ del fonón son solución de la ecuación secular:

$$\det \left| \frac{1}{\sqrt{M_\alpha M_{\alpha\prime}}} D^{ij}_{\alpha\alpha\prime}(\vec{q}) - \omega^2(\vec{q}) \right| \tag{2}$$

donde M_α es la masa del átomo α y la matriz dinámica viene definida por:

$$D^{i,j}_{\alpha,\alpha\prime} = \frac{\partial^2 E}{\partial u^{*i}_{\alpha}(\vec{q}) \partial u^{j}_{\alpha\prime}(\vec{q})} \tag{3}$$

donde u^i_α representa el desplazamiento del átomo α en la dirección i.

La segunda derivada de la energía de la ecuación 3 corresponde al cambio en la fuerza que actua sobre el átomo $\alpha\prime$ en la dirección j cuando se desplaza el átomo α en la dirección i

$$D^{ij}_{\alpha\alpha\prime}(\vec{q}) = \frac{\partial}{\partial u^{*\alpha}_i} F^j_{\alpha\prime}(\vec{q}) \tag{4}$$

Puesto que el cálculo de los modos de vibración por primeros principios empieza por establecer la geometria del cristal en equilibrio, vamos a comprobar que con los datos proporcionados que el BN monolayer se trata de un cristal bidimensional de base diatómica, cuya celda unidad viene dada por (datos proporcionados):

$$\vec{a}_1 = a(1,0); \qquad \vec{a}_2 = a\left(-\frac{1}{2}, \frac{\sqrt{3}}{2}\right) \tag{5}$$

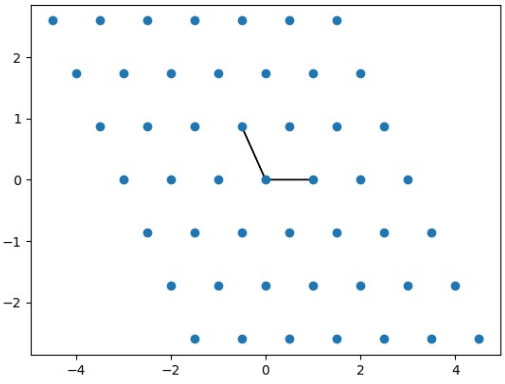
```
import numpy as np
from  numpy  import array, sqrt, sort, vdot, pi, arccos
from numpy.linalg import norm
import pandas as pd
import matplotlib
from matplotlib import pyplot as plt

a=1
a_1=np.array([a,0])
a_2=np.array([-a/2,sqrt(3)*a/2])
```

Podemos comprobar que efectivamente se trata de una celdilla hexagonal, pues los dos vectores base forman un angulo de $2\pi/3\,\mathrm{rad}$

Numeraremos las celdillas unidad con un índice vectorial $\vec{l} = (l_1, l_2)$, notemos que aunque es habitual representar la celdilla con un indice entero n es más sencillo y facilita los cálculos (así como determinar a que celdilla no referimos) el uso de un índice vectorial. Las posiciones de los nudos son $\vec{R}_{\vec{l}} = l_1\vec{a}_1 + l_2\vec{a}_2$.

Visualizamos una región de la red hexagonal, con los correspondientes nudos (que no átomos), así como la correspondiente celda unidad,

```
def R_l(l_1,l_2):
 return l_1*a_1+l_2*a_2

reddenudos=array([R_l(l_1,l_2) for l_1 in range(-3
  for l_2 in range(-3,4)])

x = reddenudos[:,0]
y = reddenudos[:,1]
plt.plot(x,y,"o")
ax = plt.axes()
ax.arrow(R_l(0,0)[0],R_l(0,0)[1],R_l(1,0)[0],R_l(1
ax.arrow(R_l(0,0)[0],R_l(0,0)[1],R_l(0,1)[0],R_l(0
plt.savefig("Graficas/Reddenudos.jpg")
plt.close()
```



Para calcular los modos de vibración por primeros principios debemos determinar primero las posiciones atómicas de equilibrio en la celda unidad **nota: proporcionadas como datos**

Los átomos estan situados en:

$$\begin{aligned} \vec{R}_B &= \frac{1}{3}\vec{a_1} + 2\vec{a_2} \\ \vec{R}_N &= \frac{2}{3}\vec{a_1} + \frac{1}{3}\vec{a_2} \end{aligned} \tag{6}$$

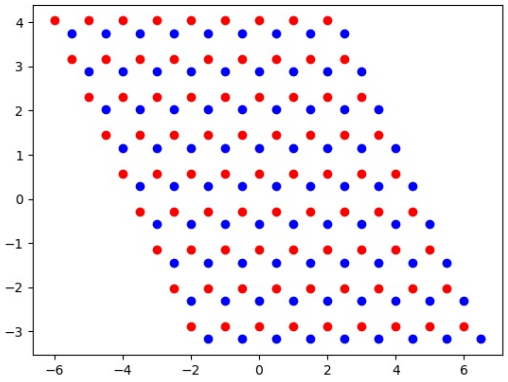
```
R_B=1/3*a_1+2/3*a_2
R_N=2/3*a_1+1/3*a_2
```

Las posiciones de equilibrio de los átomos de la base respecto de su nudo son $\vec{R}^0_\alpha$, con $\alpha = 1, 2$, puesto que la base tiene 2 átomos, el 1 hará referencia a los átomos de B y 2 a los de átomos de N (notemos que aunque los átomos fuesen idénticos tendriamos que especificar a que átomo de la base nos referimos, puesto que no ocupan posiciones equivalentes).

Las posiciones de equilibrio de los átomos: $\vec{R}_{\alpha,\vec{l}} = \vec{R}_{\vec{l}} + \vec{R}^0_\alpha$ así como los desplazamientos atómicos: $\vec{u}_{\alpha,\vec{l}}$ quedarán por tanto identificados por medio de dos índices.

Pasamor ahora a representar la red de átomos:

```
#Posiciones de equilibrio de los átomos

def R_alpha_l(alpha,l_1,l_2):
  if alpha == 1:
    return l_1*a_1+l_2*a_2+R_B

  elif alpha == 2:
    return l_1*a_1+l_2*a_2+R_N

  else:
    print("Error, alpha solo puede ser 1 o 2 "

AtomosB=array([R_alpha_l(1,l_1,l_2) for l_1 in
               for l_2 in range(-4,5)])

AtomosN=array([R_alpha_l(2,l_1,l_2) for l_1 in
               for l_2 in range(-4,5)])

xB = AtomosB[:,0]
yB = AtomosB[:,1]
plt.plot(xB,yB,"o",color="red")

xN = AtomosN[:,0]
yN = AtomosN[:,1]
plt.plot(xN,yN,"o",color="blue")

plt.savefig("Graficas/Reddeatomos.jpg")
plt.close()
```



Las dimensiones del cristal son $L_1 = N_1 a_1$ y $L_2 = N_2 a_2$, donde N_i $(i = 1, 2)$ es el número de celdillas en la dirección $\vec{a}_i$. El cristal tiene tiene $N = N_1 N_2$ celdillas unidad primitivas y $2N$ átomos.

La idea básica es que si la base tiene r átomos entonces debemos plantear y resolver las ecuaciones de movimiento de los r átomos de la base de la celdilla $\vec{0}$, por lo tanto en el caso que estamos estudiando debemos resolver 2 ecuaciones vectoriales de movimiento: una para el átomo de B y la otra para el de N.

Falta reescribir completamente y ampliar mucho la introducción teórica

Pasamos a comprobar que a partir de las ecuaciones de movimiento, y buscando soluciones de la forma:

$$\vec{u}_{\alpha,\vec{l}} = \vec{A}_\alpha e^{i(\vec{q}\cdot\vec{R}_{\vec{l}} - \omega t)}$$

donde $\vec{A}_\alpha$ es el *vector de polarización*, que determina la amplitud y dirección de vibración de los átomos de tipo α podemos plantear el problema como la determinación de los valores propios de la matriz dinámica.

Notemos que aunque el cristal sea bi-dimensional, los átomos de este pueden vibrar en las 3 direcciones espaciales.

Esta ecuación tiene $3N_\nu = 6$ soluciones que describen las 6 ramas de la relación de dispersión, es decir, las 6 frecuencias características de los 6 modos normales de vibración de vector de onda $\vec{q}$. Se cumple que el número total de modos normales de vibración coincide con el triple del número total de átomos, es decir, *el número total de modos normales de vibración coincide con el de grados de libertad de movimiento de los átomos*

Explicar que podemos tratar por un lado las vibraciones en el plano del cristal y por otro las vibraciones perpendiculares a este, ya que se trata de vibraciones completamente desacopladas.

2 TODO Desarrollo del trabajo

Puesto que debemos determinar cuales son las posiciones de equilibrio de los átomos más cercanos a los átomos de la celda $\vec{0}$ antes que nada genero un array con los datos que voy a necesitar, ordenando las filas de manera creciente a la distancia a cada uno de los 2 átomos de la celda $l = \vec{0}$ hasta los cuartos vecinos, y guardando el array con la información como un DataFrame de pandas, que facilita mucho la manipulación de los datos.

```
from sympy import *
import pandas as pd

## Parametros de la red, de la celdilla y del cristal
a=Symbol('a', real=True, positive=True)
q_x=Symbol('q_x', real=True);  q_y=Symbol('q_y', real=True)
q=Matrix([q_x,q_y])
a_1=Matrix([a,0]); a_2=Rational(1,2)*Matrix([-a,sqrt(3)*a])
R_B=Rational(1,3)*a_1+Rational(2,3)*a_2; R_N=Rational(2,3)*a_1+Rational(1,3)*a_2

## Masas de los átomos, frecuencia, ...
M_B, M_N, omega=symbols("M_B, M_N, omega") #masa de los átomos de Boro y N.
def masa(alpha):
  if alpha == 1:
    return M_B

  elif alpha == 2:
    return M_N

  else:
    print("Error, alpha sólo puede se 1 o 2")


## Vector R_n (vector de traslación primitivo)
def R_l(l_1,l_2):
  return l_1*a_1+l_2*a_2

## Vector de posición de los átomos del cristal (en equilibrio)
def R_alpha_l(alpha,l_1,l_2):
  if alpha == 1:
    return l_1*a_1+l_2*a_2+R_B

  elif alpha == 2:
    return l_1*a_1+l_2*a_2+R_N

  else:
    print("Error, alpha solo puede ser 1 o 2 ")

## Vector unitario que une uno de los átomos en la celdilla 0 con el átomo considerado
def R_hat(alphaprima,alpha,l_1,l_2):
```

```
  if (R_alpha_l(alpha,l_1,l_2)-R_alpha_l(alphaprima,0,0)).norm()>0:
    return (R_alpha_l(alpha,l_1,l_2)-R_alpha_l(alphaprima,0,0))/(R_alpha_l(alpha,l_1,l_2)
                                                    -R_alpha_l(alphaprima,0,0)).norm()

  else:
    return (R_alpha_l(alpha,l_1,l_2)-R_alpha_l(alphaprima,0,0))

## Finalmente construyo un DataFrame de pandas con la información necesaria para
## identificar a los primeros, segundos, ... vecinos, según su distancia a cada uno
## de los átomos de la celdilla unidad
def propiedades_atomos(l_1, l_2):
  return [(k, m, i, j, exp(I*q.dot(R_l(i,j))), (R_alpha_l(m,i,j)-R_alpha_l(k,0,0)).norm()/
      for k in [1,2] for m in [1,2]  for i in range(-l_1,l_1+1) for j in range(-l_2,l_2+1)

columnas = [r"$\alpha\prime$",r"$\alpha",r"$l_1$", r"$l_2$",'Exponencial','Distancia']

def Atomos(l_1, l_2):
  return pd.DataFrame(propiedades_atomos(l_1,l_2),columns=columnas).sort_values(
       ['Distancia',r"$\alpha\prime$"], ascending=[True, True])
## Mostramos el dataframe como una tabla en formato \LaTeX.
Atomos(2,2).head(38).to_latex(escape=False,float_format="{:0.4f}".format,index=False)
```

$\alpha\prime$	α	l_1	l_2	Exponencial	Distancia
1	1	0	0	1	0
2	2	0	0	1	0
1	2	-1	0	exp(-I*a*q$_x$)	sqrt(3)/3
1	2	0	0	1	sqrt(3)/3
1	2	0	1	exp(I*(-a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	sqrt(3)/3
2	1	0	-1	exp(I*(a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	sqrt(3)/3
2	1	0	0	1	sqrt(3)/3
2	1	1	0	exp(I*a*q$_x$)	sqrt(3)/3
1	1	-1	-1	exp(I*(-a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	1
1	1	-1	0	exp(-I*a*q$_x$)	1
1	1	0	-1	exp(I*(a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	1
1	1	0	1	exp(I*(-a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	1
1	1	1	0	exp(I*a*q$_x$)	1
1	1	1	1	exp(I*(a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	1
2	2	-1	-1	exp(I*(-a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	1
2	2	-1	0	exp(-I*a*q$_x$)	1
2	2	0	-1	exp(I*(a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	1
2	2	0	1	exp(I*(-a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	1
2	2	1	0	exp(I*a*q$_x$)	1
2	2	1	1	exp(I*(a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	1
1	2	-1	-1	exp(I*(-a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	2*sqrt(3)/3
1	2	-1	1	exp(I*(-3*a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	2*sqrt(3)/3
1	2	1	1	exp(I*(a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	2*sqrt(3)/3
2	1	-1	-1	exp(I*(-a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	2*sqrt(3)/3
2	1	1	-1	exp(I*(3*a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	2*sqrt(3)/3
2	1	1	1	exp(I*(a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	2*sqrt(3)/3
1	2	-2	-1	exp(I*(-3*a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	sqrt(21)/3
1	2	-2	0	exp(-2*I*a*q$_x$)	sqrt(21)/3
1	2	0	-1	exp(I*(a*q$_x$/2 − $sqrt(3) * a * q_y$/2))	sqrt(21)/3
1	2	0	2	exp(I*(-a*q$_x$ + $sqrt(3) * a * q_y$))	sqrt(21)/3
1	2	1	0	exp(I*a*q$_x$)	sqrt(21)/3
1	2	1	2	exp(sqrt(3)*I*a*q$_y$)	sqrt(21)/3
2	1	-1	-2	exp(-sqrt(3)*I*a*q$_y$)	sqrt(21)/3
2	1	-1	0	exp(-I*a*q$_x$)	sqrt(21)/3
2	1	0	-2	exp(I*(a*q$_x$ − $sqrt(3) * a * q_y$))	sqrt(21)/3
2	1	0	1	exp(I*(-a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	sqrt(21)/3
2	1	2	0	exp(2*I*a*q$_x$)	sqrt(21)/3
2	1	2	1	exp(I*(3*a*q$_x$/2 + $sqrt(3) * a * q_y$/2))	sqrt(21)/3

A parte de identificar los primeros, segundos, ... vecinos, necesitamos conocer la correspondiente matriz de constantes de fuerza que corresponde a la interacción de cada átomo de la celdilla unidad con su n-esimo vecino.

Vamos a realizar la simplificación que un desplazamiento longitudinal, transversal o perpendicular solo genera una fuerza longitudidal, transversal o perpendicular, de manera que la matriz de constantes de fuerza tiene 12 parámetros no nulos (y que deberemos determinar).

```
PrimerosVecinosBoro= Atomos(1,1)[(Atomos(1,1)['Distancia']<0.9) &\
(Atomos(1,1)['Distancia']>0) & (Atomos(1,1)[r"$\alpha\prime$"]==1)]
PrimerosVecinosNitrogeno= Atomos(1,1)[(Atomos(1,1)['Distancia']<0.9) & \
(Atomos(1,1)['Distancia']>0) & (Atomos(1,1)[r"$\alpha\prime$"]==2)]
```

```
## Constantes de fuerza
phi_BB__xx,phi_BB__yy,phi_BB__zz,phi_NN__xx,phi_NN__yy,phi_NN__zz, \
phi_BN__xx,phi_BN__yy,phi_BN__zz,phi_NB__xx,phi_NB__yy,phi_NB__zz = symbols(\
'phi_BB__xx,phi_BB__yy, phi_BB__zz,phi_NN__xx,phi_NN__yy,phi_NN__zz, \
phi_BN__xx,phi_BN__yy,phi_BN__zz, phi_NB__xx,phi_NB__yy,phi_NB__zz') \

phi=zeros(6)

phi[0,0]=phi_BB__xx
phi[1,1]=phi_BB__yy
phi[2,2]=phi_BB__zz
phi[3,3]=phi_NN__xx
phi[4,4]=phi_NN__yy
phi[5,5]=phi_NN__zz

phi[3,0]=phi_NB__xx
phi[4,1]=phi_NB__yy
phi[5,2]=phi_NB__zz

phi[0,3]=phi_BN__xx
phi[1,4]=phi_BN__yy
phi[2,5]=phi_BN__zz
```

y la matriz de constantes de fuerza tiene la expresión:

$$\phi_1 = \begin{bmatrix} \phi_{BB}^{xx} & 0 & 0 & \phi_{BN}^{xx} & 0 & 0 \\ 0 & \phi_{BB}^{yy} & 0 & 0 & \phi_{BN}^{yy} & 0 \\ 0 & 0 & \phi_{BB}^{zz} & 0 & 0 & \phi_{BN}^{zz} \\ \phi_{NB}^{xx} & 0 & 0 & \phi_{NN}^{xx} & 0 & 0 \\ 0 & \phi_{NB}^{yy} & 0 & 0 & \phi_{NN}^{yy} & 0 \\ 0 & 0 & \phi_{NB}^{zz} & 0 & 0 & \phi_{NN}^{zz} \end{bmatrix} \tag{7}$$

Por lo tanto, la matriz dinámica que tenemos es:

```
D1=zeros(6)

D1[0,0]=1/M_B*phi[0,0]
D1[1,1]=1/M_B*phi[1,1]
D1[2,2]=1/M_B*phi[2,2]
D1[3,3]=1/M_N*phi[3,3]
D1[4,4]=1/M_N*phi[4,4]
D1[5,5]=1/M_B*phi[5,5]

D1[3,0]=(1/sqrt(M_N*M_B))*phi[3,0]*sum(PrimerosVecinosBoro['Exponencial'])
D1[4,1]=(1/sqrt(M_N*M_B))*phi[4,1]*sum(PrimerosVecinosBoro['Exponencial'])
D1[5,2]=(1/sqrt(M_N*M_B))*phi[5,2]*sum(PrimerosVecinosBoro['Exponencial'])

D1[0,3]=(1/sqrt(M_B*M_N))*phi[0,3]*sum(PrimerosVecinosNitrogeno['Exponencial'])
D1[1,4]=(1/sqrt(M_B*M_N))*phi[1,4]*sum(PrimerosVecinosNitrogeno['Exponencial'])
D1[2,5]=(1/sqrt(M_B*M_N))*phi[2,5]*sum(PrimerosVecinosNitrogeno['Exponencial'])
```

$$\begin{bmatrix} \frac{\phi_{BB}^{xx}}{M_B} & 0 & 0 & \frac{\phi_{BN}^{xx}\left(e^{i\left(\frac{aq_x}{2}-\frac{\sqrt{3}aq_y}{2}\right)}+e^{iaq_x}+1\right)}{\sqrt{M_BM_N}} & 0 & 0 \\ 0 & \frac{\phi_{BB}^{yy}}{M_B} & 0 & 0 & \frac{\phi_{BN}^{yy}\left(e^{i\left(\frac{aq_x}{2}-\frac{\sqrt{3}aq_y}{2}\right)}+e^{iaq_x}+1\right)}{\sqrt{M_BM_N}} & 0 \\ 0 & 0 & \frac{\phi_{BB}^{zz}}{M_B} & 0 & 0 & \frac{\phi_{BN}^{zz}\left(e^{i\left(\frac{aq_x}{2}-\frac{\sqrt{3}aq_y}{2}\right)}+e^{iaq_x}+1\right)}{\sqrt{M_BM_N}} \\ \frac{\phi_{NB}^{xx}\left(e^{i\left(-\frac{aq_x}{2}+\frac{\sqrt{3}aq_y}{2}\right)}+1+e^{-iaq_x}\right)}{\sqrt{M_BM_N}} & 0 & 0 & \frac{\phi_{NN}^{xx}}{M_N} & 0 & 0 \\ 0 & \frac{\phi_{NB}^{yy}\left(e^{i\left(-\frac{aq_x}{2}+\frac{\sqrt{3}aq_y}{2}\right)}+1+e^{-iaq_x}\right)}{\sqrt{M_BM_N}} & 0 & 0 & \frac{\phi_{NN}^{yy}}{M_N} & 0 \\ 0 & 0 & \frac{\phi_{NB}^{zz}\left(e^{i\left(-\frac{aq_x}{2}+\frac{\sqrt{3}aq_y}{2}\right)}+1+e^{-iaq_x}\right)}{\sqrt{M_BM_N}} & 0 & 0 & \frac{\phi_{NN}^{zz}}{M_B} \end{bmatrix} \tag{8}$$

Como he comenttado, tenemos 12 parámetros (los elementos del tensor constantes de fuerza no nulos) a determinar. Simplifico aún más el problema y asumiento que:

```
phi_BB, phi_NN, phi_BN = symbols('phi_BB, phi_NN, phi_BN')
phi_BB__xx, phi_BB__yy, phi_BB__zz=(phi_BB for i in range(3))
phi_NN__xx, phi_NN__yy, phi_NN__zz=(phi_NN for i in range(3))
phi_NB__xx, phi_NB__yy, phi_NB__zz, phi_BN__xx, phi_BN__yy, phi_BN__zz= \
(phi_BN for i in range(6))
```