Transporte en nanoestructuras de Bi

Montserrat Navarro Espino

7/31/2023

Table of contents

Prefacio

Este es un sitio creado con Quarto y y publicado a través de GitHub Pages. En él se encuentran conceptos clave para el estudio del transporte electrónico en nanoestructuras de bismuto.

La elaboración de estas notas tiene como propósito brindar acceso a los resultados del trabajo de investigación desempeñado por la autora en el Programa de Maestría en Ciencias Químicas.

1 Introducción

El bismuto es uno de los elementos químicos más pesados y con un mayor tiempo de vida media (2×10^{19} años aproximadamente) [**?**, **?**]. Por ende, posee un enorme acoplamiento espín-orbita (SOC, por sus siglas en inglés) y sus electrones presentan efectos relativistas [**?**, **?**]. El cristal de bismuto es un semimetal en el que se han observado diversas propiedades y fenómenos cuánticos de interés, siendo uno de ellos la topología de su estructura electrónica [**?**].

La estructura electrónica de Bi prístino ha sido considerada por un largo tiempo como topológicamente trivial, lo cual implica que Sin embargo, a través de varios experimentos empleando la espectroscopía de fotoemisión con resolución angular (ARPES, por sus siglas en inglés), se encontró que las bandas de la superficie poseen una topología no trivial [**?**, **?**, **?**], como se observa en la Fig. **??**.

En el año 2018, se realizó un esfuerzo excepcional en este ámbito, donde fue identificada la naturaleza topológica de orden superior de la estructura electrónica de bismuto [**?**]. En dicho trabajo se identificaron estados de arista Figure **??**, o *hinge states*, los cuales están protegidos globalmente por la simetría de reversión temporal (TRS), y discretamente por la simetría rotacional de tercer orden ($\hat{\mathcal{C}}_3$) y la simetría de inversión del cristal de bismuto ($\hat{\mathcal{I}}$).

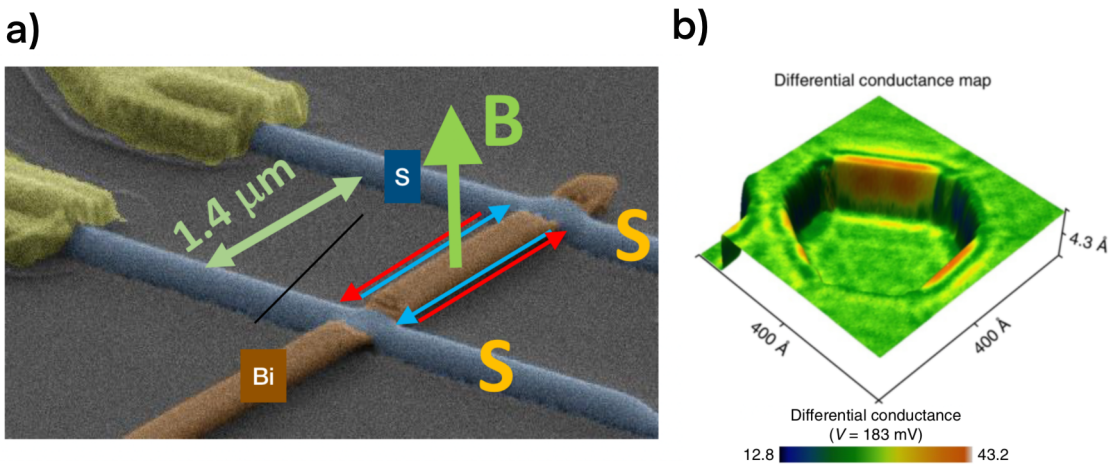

Figure 1.1: Detección experimental de los estados de arista. a) Experimento STM donde se observan estados alternados y fuertemente localizados en el borde de Bi (111). b) Junta de Josephson S/Bi/S donde se detecta corriente fluye a través de canales extremadamente estrechos (unidimensionales) [**?**].

2 Geometría de bismuto

2.1 Estructura cristalina de bismuto

2.1.1 Espacio Real

El bismuto (Bi) es un elemento químico cuyo arreglo cristalino consiste de una celda unitaria romboédrica que contiene dos átomos [**?**], cada uno de ellos con tres primeros vecinos y tres segundos vecinos [**?**], como se señala en la Fig. (**RomboRealRec?**) a).

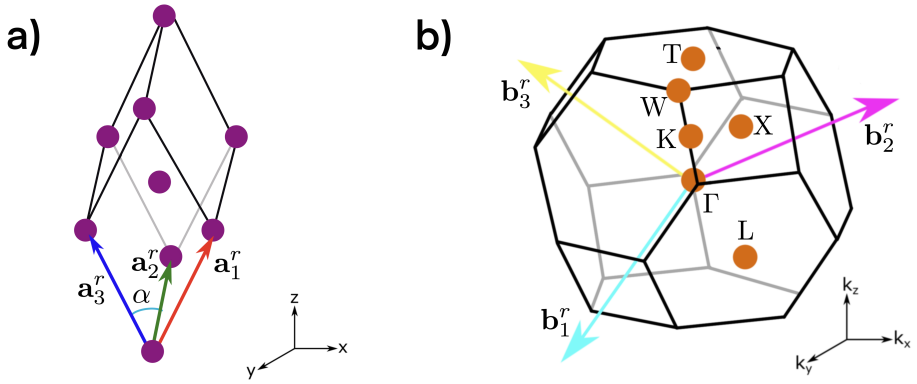

Figure 2.1: a) Celda romboédrica de bismuto en el espacio real, con los vectores de red $\mathbf{a}_n^r$ $(n = 1, 2, 3)$. b) Primera zona de Brillouin de la estructura romboédrica del Bi, con los vectores $\mathbf{b}_n^r$ y los puntos de alta simetría (naranja).

La estructura romboédrica de bismuto se esquematiza en la Fig. Figure **??** (a), donde los vectores de la red en el espacio real son:

$$\mathbf{a}_1^r = \left(-\frac{1}{2}a, -\frac{\sqrt{3}}{6}a, \frac{1}{3}c\right) \qquad \mathbf{a}_2^r = \left(\frac{1}{2}a, -\frac{\sqrt{3}}{6}a, \frac{1}{3}c\right) \qquad \mathbf{a}_3^r = \left(0, -\frac{\sqrt{3}}{3}a, \frac{1}{3}c\right)$$

siendo $a = 4.5332$, $c = 11.7967$ y $\alpha = 57°19'$ [**?**].

El grupo espacial de la estructura cristalina es $R\bar{3}m$ y su grupo puntual es el D_{3d}. Por lo tanto, las operaciones de simetría espacial que caracterizan este arreglo cristalino son [**?**]:

- la identidad ($\hat{E}$),
- la inversión ($\hat{I}$),
- las rotaciones de 120° ($\hat{C}_3$) respecto el eje z y 180° ($\hat{C}_2$) respecto el eje y y
- los planos de reflexión $\mathcal{M}_a$, $\mathcal{M}_b$ y $\mathcal{M}_c$, perpendiculares al eje de rotación $\hat{C}_2$.

2.1.2 Espacio recíproco

La primera zona de Brillouin (1ZB) para la celda romboédrica tiene la forma de una octaedro truncado, el cual se esquematiza en la Fig. Figure **??** b). Los vectores de la red recíproca son:

$$\mathbf{b}_1^r = \left(-1, -\frac{\sqrt{3}}{3}, b\right) g \qquad \mathbf{b}_2^r = \left(1, -\frac{\sqrt{3}}{3}, b\right) g \qquad \mathbf{b}_3^r = \left(0, -2\frac{\sqrt{3}}{3}, b\right) g$$

donde $b = a/c$ y $g = 1.3861\ ^{-1}$. Las coordenadas relativas de algunos puntos de alta simetría en esta 1ZB son:

$$\Gamma = (0, 0, 0)$$
$$\mathrm{K} = \left[0, \left(\frac{3}{4} - \frac{1}{2}h\right), \left(\frac{1}{2}h + \frac{1}{4}\right)\right]$$
$$\mathrm{X} = \left(0, \frac{1}{2}, \frac{1}{2}\right)$$
$$\mathrm{W} = \left(h, 1-h, \frac{1}{2}\right)$$
$$\mathrm{T} = \left(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}\right)$$
$$\mathrm{L} = \left(0, \frac{1}{2}, 0\right)$$
$$\Lambda = (0, 0, 0)$$

donde $h = 0.2303$ en el caso de bismuto [**?**].

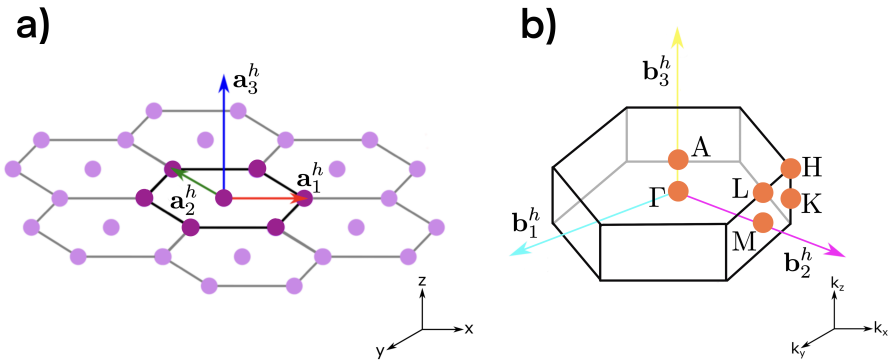

Figure 2.2: a) Estructura de la celda unitaria para el Hamiltoniano topológico de bismuto en el espacio real y b) recíproco. Los vectores de la red real están etiquetados por $\mathbf{a}_n^h$ y los de la red recíproca por $\mathbf{b}_n^h$ $(n = 1, 2, 3)$.

2.1.3 Estructura cristalina para modelo topológico

Al proyectar en el plano (111), el bulto del cristal forma una red hexagonal con dos átomos por celda unitaria.

En el modelo topológico de bismuto [**?**] que reproduce los estados conductores de borde en una nanoestructura (ver Sección **??**), se considera una estructura como la que se esquematiza en la Fig. Figure **??**.

Este sistema consiste de una red hexagonal simple que preserva las simetrías espaciales de la red de bismuto. Los vectores de esta red hexagonal en el espacio real son:

$$\mathbf{a}_1^h = (a, 0, 0) \qquad \mathbf{a}_2^h = \left(-\frac{1}{2}a, \frac{\sqrt{3}}{2}a, 0\right) \qquad \mathbf{a}_3^h = (0, 0, c)$$

La 1ZB en el espacio recíproco también consiste de una celda hexagonal. Además, las coordenadas relativas de algunos de los puntos de alta simetría en ella son:

$$\Gamma = (0, 0, 0)$$
$$\mathrm{M} = \left(\frac{1}{2}, 0, 0\right)$$
$$\mathrm{K} = \left(\frac{1}{3}, \frac{1}{3}, 0\right)$$
$$\mathrm{A} = \left(0, 0, \frac{1}{2}\right)$$

$$\mathrm{L} = \left(\frac{1}{2}, 0, \frac{1}{2}\right)$$
$$\mathrm{H} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{2}\right).$$

3 Hamiltoniano de amarre fuerte de bismuto

En el año 2018, Schindler y colaboradores calcularon la estructura electrónica a primeros principios del bulto de bismuto y obtuvieron una brecha energética directa entre la banda de conducción y la de valencia en cada punto del espacio . Debido a la presencia de esta brecha, pudieron analizar la estructura electrónica del bismuto en el esquema de la Química Cuántica Topológica [**?**] (la cual se basa en el paradigma de la Representación de Bandas [**?**]), y confirmaron el carácter de aislante topológico de orden superior (HOTI) de este material. Finalmente, propusieron un Hamiltoniano de amarre fuerte topológicamente equivalente a un modelo realista de bismuto que facilita la identificación de los estados de borde pues no contienen los estados del bulto semimetálico. Además, al únicamente considerar 8 orbitales por celda unitaria, las simulaciones de sistemas tridimensionales grandes son computacionalmente realizables.

3.1 Expresión analítica

El modelo del Hamiltoniano de 8 bandas de amarre fuerte del bismuto propuesto por Schindler y colaboradores [**?**] es:

$$H_{\mathrm{TB}}^{Schin}() = \begin{pmatrix} H_{TB,I}() + \epsilon I & \delta M_{TB}() \\ \delta M_{TB}()^{\dagger} & H_{TB,II}() - \epsilon I \end{pmatrix}$$

donde los términos de $H_{TB,I}()$, $H_{TB,II}()$ y $M_{TB}()$ tienen la forma:

$$\begin{aligned} H_{TB,I}() = \Gamma_1\{m_I(1+\cos\cdot^h) - t_I[\cos\cdot^h + \cos\cdot^h + \cos\cdot(^h{+}^h)]\} \\ + \lambda_I[\Gamma_2 \sin\cdot^h + \Gamma_{2,1}^{I,I}\sin\cdot^h + \Gamma_{2,2}^{I,I}\sin\cdot(^h{+}^h) + \Gamma_3\sin\cdot^h] \end{aligned}$$

$$\begin{aligned} H_{TB,II}() = \Gamma_1\{m_{II}(1+\cos\cdot^h) - t_{II}[\cos\cdot^h + \cos\cdot^h + \cos\cdot(^h{+}^h)]\} \\ + \lambda_{II}[\Gamma_2 \sin\cdot^h + \Gamma_{2,1}^{II,II}\sin\cdot^h + \Gamma_{2,2}^{II,II}\sin\cdot(^h{+}^h) + \Gamma_3\sin\cdot^h] \end{aligned}$$

$$
\begin{aligned}
M_{TB}() = \Gamma_2[\sin\cdot^h + \sin\cdot(2^h{+}^h)] + \Gamma_{2,1}^{I,II}[\sin\cdot^h + \sin\cdot(^h{-}^h)] \\
- \Gamma_{2,2}^{I,II}[\sin\cdot(^h{+}^h) + \sin\cdot(^h{+}2^h)] - i\Gamma_5[\cos\cdot^h + \cos\cdot(2^h{+}^h)] \\
- i\Gamma_{5,1}^{I,II}[\cos\cdot^h + \cos\cdot(^h{-}^h)] - i\Gamma_{5,2}^{I,II}[\cos\cdot(^h{+}^h) + \cos\cdot(^h{+}2^h)]
\end{aligned}
$$

3.2 Implementación en PythTB

```
from pylab import *
from pythtb import *

a1 = [   1,        0,0]
a2 = [-1/2,sqrt(3)/2,0]
a3 = [   0,        0,1]

lat = [a1,a2,a3]
orb = [[0,0,1/2],
       [0,0,1/2],
       [0,0,1/2],
       [0,0,1/2],
       [0,0,1/2],
       [0,0,1/2],
       [0,0,1/2],
       [0,0,1/2]]

tI  = 1; tII = 1
mI  = 2; mII = 2
 = 0.1
I = 0.3; II = 1
II = 1
 = 0.3

Bismuto = tb_model(3,3,lat,orb)
```

```
# [0,0]
Bismuto.set_hop( -tI/2,0,0,[ 1,0,0] ) # cell = [ 1,0,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop( -tI/2,0,0,[ 0,1,0] ) # cell = [ 0,1,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop( -tI/2,0,0,[ 1,1,0] ) # cell = [ 1,1,0] -> e^{ik·[(cell)·(a1,a2,a3)]}

Bismuto.set_hop(  mI/2,0,0,[ 0,0,1] ) # cell = [ 0,0,1] -> e^{ik·[(cell)·(a1,a2,a3)]}
```

```
# [0,2]

Bismuto.set_hop( -I/2,0,2,[0,0, 1] ) # cell = [ 0,0,1] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop(  I/2,0,2,[0,0,-1] ) # cell = [ 0,0,-1] -> e^{ik·[(cell)·(a1,a2,a3)]}
```

```
# [0,3]
Bismuto.set_hop(  I/(2J),0,3,[ 1,0,0] ) # cell = [ 1,0,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop( -I/(2J),0,3,[-1,0,0] ) # cell = [-1,0,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop(  I/(2J)*exp(2J*pi/3),0,3,[ 0, 1,0] ) # cell = [ 0,1,0] -> e^{ik·[(cell)·(
Bismuto.set_hop( -I/(2J)*exp(2J*pi/3),0,3,[ 0,-1,0] ) # cell = [ 0,-1,0] -> e^{ik·[(cell)·
Bismuto.set_hop(  I/(2J)*exp(1J*pi/3),0,3,[ 1, 1,0] ) # cell = [ 1, 1,0] -> e^{ik·[(cell)·
Bismuto.set_hop( -I/(2J)*exp(1J*pi/3),0,3,[-1,-1,0] ) # cell = [-1,-1,0] -> e^{ik·[(cell)·
```

```
# [0,5]
Bismuto.set_hop(  1J*/2*exp(1J*pi/3),0,5,[ 0, 1,0] ) # cell = [ 0,1,0] -> e^{ik·[(cell)·(a
Bismuto.set_hop(  1J*/2*exp(1J*pi/3),0,5,[ 0,-1,0] ) # cell = [0,-1,0] -> e^{ik·[(cell)·(a
Bismuto.set_hop(  1J*/2*exp(1J*pi/3),0,5,[-1, 1,0] ) # cell = [-1,1,0] -> e^{ik·[(cell)·(a
Bismuto.set_hop(  1J*/2*exp(1J*pi/3),0,5,[ 1,-1,0] ) # cell = [1,-1,0] -> e^{ik·[(cell)·(a

Bismuto.set_hop( -1J*/2,0,5,[ 1, 0,0] ) # cell = [1,0,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop( -1J*/2,0,5,[-1, 0,0] ) # cell = [-1,0,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop( -1J*/2,0,5,[ 2, 1,0] ) # cell = [2,1,0] -> e^{ik·[(cell)·(a1,a2,a3)]}
Bismuto.set_hop( -1J*/2,0,5,[-2,-1,0]) # cell = [-2,-1,0] -> e^{ik·[(cell)·(a1,a2,a3)]}

Bismuto.set_hop( -1J*/2*exp(2J*pi/3),0,5,[ 1, 1,0] ) # cell = [1,0,0] -> e^{ik·[(cell)·(a1
Bismuto.set_hop( -1J*/2*exp(2J*pi/3),0,5,[-1,-1,0] ) # cell = [-1,0,0] -> e^{ik·[(cell)·(a
Bismuto.set_hop( -1J*/2*exp(2J*pi/3),0,5,[ 1, 2,0] ) # cell = [2,1,0] -> e^{ik·[(cell)·(a1
Bismuto.set_hop( -1J*/2*exp(2J*pi/3),0,5,[-1,-2,0]) # cell = [-2,-1,0] -> e^{ik·[(cell)·(a
```

```
# [0,7]
Bismuto.set_hop(  /2J,0,7,[ 1,0,0] )
Bismuto.set_hop( -/2J,0,7,[-1,0,0] )

Bismuto.set_hop( -/2J*exp(1J*pi/3),0,7,[ 0, 1,0] )
Bismuto.set_hop(  /2J*exp(1J*pi/3),0,7,[ 0,-1,0] )
Bismuto.set_hop( -/2J*exp(1J*pi/3),0,7,[-1, 1,0] )
Bismuto.set_hop(  /2J*exp(1J*pi/3),0,7,[ 1,-1,0] )

Bismuto.set_hop(  /2J,0,7,[ 2, 1,0] )
Bismuto.set_hop( -/2J,0,7,[-2,-1,0] )
```

```
Bismuto.set_hop( -/2J*exp(2J*pi/3),0,7,[ 1, 1,0] )
Bismuto.set_hop(  /2J*exp(2J*pi/3),0,7,[-1,-1,0] )
Bismuto.set_hop( -/2J*exp(2J*pi/3),0,7,[ 1, 2,0] )
Bismuto.set_hop(  /2J*exp(2J*pi/3),0,7,[-1,-2,0] )
```

```
# [1,1]
Bismuto.set_hop( -tI/2,1,1,[  1, 0,0] )
#Bismuto.set_hop( -tI/2,1,1,[ -1, 0,0] )
Bismuto.set_hop( -tI/2,1,1,[  0, 1,0] )
#Bismuto.set_hop( -tI/2,1,1,[  0,-1,0] )
Bismuto.set_hop( -tI/2,1,1,[  1, 1,0] )
#Bismuto.set_hop( -tI/2,1,1,[ -1,-1,0] )

Bismuto.set_hop( mI/2,1,1,[ 0,0, 1] )
#Bismuto.set_hop( mI/2,1,1,[ 0,0,-1] )
```

```
# [1,2]
Bismuto.set_hop(  I/(2J),1,2,[ 1,0,0] )
Bismuto.set_hop( -I/(2J),1,2,[-1,0,0] )

Bismuto.set_hop(  I/(2J)*exp(-2J*pi/3),1,2,[0, 1,0] )
Bismuto.set_hop( -I/(2J)*exp(-2J*pi/3),1,2,[0,-1,0] )

Bismuto.set_hop(  I/(2J)*exp(-1J*pi/3),1,2,[ 1, 1,0] )
Bismuto.set_hop( -I/(2J)*exp(-1J*pi/3),1,2,[-1,-1,0] )
```

```
#[1,3]
Bismuto.set_hop( -I/2,1,3,[ 0,0, 1] )
Bismuto.set_hop(  I/2,1,3,[ 0,0,-1] )
```

```
#[1,4]
Bismuto.set_hop(  1J*/2*exp(-1J*pi/3),1,4,[  0, 1,0] )
Bismuto.set_hop(  1J*/2*exp(-1J*pi/3),1,4,[  0,-1,0] )
Bismuto.set_hop(  1J*/2*exp(-1J*pi/3),1,4,[ -1, 1,0] )
Bismuto.set_hop(  1J*/2*exp(-1J*pi/3),1,4,[  1,-1,0] )

Bismuto.set_hop( -1J*/2,1,4,[  1, 0,0] )
Bismuto.set_hop( -1J*/2,1,4,[ -1, 0,0] )
Bismuto.set_hop( -1J*/2,1,4,[  2, 1,0] )
```

```
Bismuto.set_hop( -1J*/2,1,4,[ -2,-1,0] )

Bismuto.set_hop( -1J*/2*exp(-2J*pi/3),1,4,[  1, 1,0] )
Bismuto.set_hop( -1J*/2*exp(-2J*pi/3),1,4,[ -1,-1,0] )
Bismuto.set_hop( -1J*/2*exp(-2J*pi/3),1,4,[  1, 2,0] )
Bismuto.set_hop( -1J*/2*exp(-2J*pi/3),1,4,[ -1,-2,0] )
```

```
#[1,6]
Bismuto.set_hop(   /2J,1,6,[  1,0,0] )
Bismuto.set_hop(  -/2J,1,6,[ -1,0,0] )

Bismuto.set_hop(  -/2J*exp(-1J*pi/3),1,6,[  0, 1,0] )
Bismuto.set_hop(   /2J*exp(-1J*pi/3),1,6,[  0,-1,0] )
Bismuto.set_hop(  -/2J*exp(-1J*pi/3),1,6,[ -1, 1,0] )
Bismuto.set_hop(   /2J*exp(-1J*pi/3),1,6,[  1,-1,0] )

Bismuto.set_hop(   /2J,1,6,[  2, 1,0] )
Bismuto.set_hop(  -/2J,1,6,[ -2,-1,0] )

Bismuto.set_hop(  -/2J*exp(-2J*pi/3),1,6,[  1, 1,0] )
Bismuto.set_hop(   /2J*exp(-2J*pi/3),1,6,[ -1,-1,0] )
Bismuto.set_hop(  -/2J*exp(-2J*pi/3),1,6,[  1, 2,0] )
Bismuto.set_hop(   /2J*exp(-2J*pi/3),1,6,[ -1,-2,0] )
```

```
#[2,2]
Bismuto.set_hop( tI/2,2,2,[  1,0,0] )
#Bismuto.set_hop( -tI/2,2,2,[ -1,0,0] )
Bismuto.set_hop( tI/2,2,2,[ 0, 1,0] )
#Bismuto.set_hop( -tI/2,2,2,[ 0,-1,0] )
Bismuto.set_hop( tI/2,2,2,[  1, 1,0] )
#Bismuto.set_hop( -tI/2,2,2,[ -1,-1,0] )

Bismuto.set_hop( -mI/2,2,2,[ 0,0, 1] )
#Bismuto.set_hop( -mI/2,2,2,[ 0,0,-1] )
```

```
#[2,5]
Bismuto.set_hop(   /2J,2,5,[  1,0,0] )
Bismuto.set_hop(  -/2J,2,5,[ -1,0,0] )

Bismuto.set_hop(  -/2J*exp(1J*pi/3),2,5,[  0, 1,0] )
```

```
Bismuto.set_hop(   /2J*exp(1J*pi/3),2,5,[  0,-1,0] )
Bismuto.set_hop(  -/2J*exp(1J*pi/3),2,5,[ -1, 1,0] )
Bismuto.set_hop(   /2J*exp(1J*pi/3),2,5,[  1,-1,0] )

Bismuto.set_hop(   /2J,2,5,[  2, 1,0] )
Bismuto.set_hop(  -/2J,2,5,[ -2,-1,0] )

Bismuto.set_hop(  -/2J*exp(2J*pi/3),2,5,[  1, 1,0] )
Bismuto.set_hop(   /2J*exp(2J*pi/3),2,5,[ -1,-1,0] )
Bismuto.set_hop(  -/2J*exp(2J*pi/3),2,5,[  1, 2,0] )
Bismuto.set_hop(   /2J*exp(2J*pi/3),2,5,[ -1,-2,0] )
```

```
#[2,7]

Bismuto.set_hop(  -*1J/2*exp(1J*pi/3) ,2,7,[  0, 1,0] )
Bismuto.set_hop(  -*1J/2*exp(1J*pi/3) ,2,7,[  0,-1,0] )
Bismuto.set_hop(  -*1J/2*exp(1J*pi/3) ,2,7,[ -1, 1,0] )
Bismuto.set_hop(  -*1J/2*exp(1J*pi/3) ,2,7,[  1,-1,0] )

Bismuto.set_hop(   *1J/2, 2,7,[  1, 0,0] )
Bismuto.set_hop(   *1J/2, 2,7,[ -1, 0,0] )
Bismuto.set_hop(   *1J/2, 2,7,[  2, 1,0] )
Bismuto.set_hop(   *1J/2, 2,7,[ -2,-1,0] )

Bismuto.set_hop(   *1J/2*exp(2J*pi/3) ,2,7,[  1, 1,0] )
Bismuto.set_hop(   *1J/2*exp(2J*pi/3) ,2,7,[ -1,-1,0] )
Bismuto.set_hop(   *1J/2*exp(2J*pi/3) ,2,7,[  1, 2,0] )
Bismuto.set_hop(   *1J/2*exp(2J*pi/3) ,2,7,[ -1,-2,0] )
```

```
#[3,4]
Bismuto.set_hop(   /2J,3,4,[  1,0,0] )
Bismuto.set_hop(  -/2J,3,4,[ -1,0,0] )

Bismuto.set_hop(  -/2J*exp(-1J*pi/3),3,4,[  0, 1,0] )
Bismuto.set_hop(   /2J*exp(-1J*pi/3),3,4,[  0,-1,0] )
Bismuto.set_hop(  -/2J*exp(-1J*pi/3),3,4,[ -1, 1,0] )
Bismuto.set_hop(   /2J*exp(-1J*pi/3),3,4,[  1,-1,0] )

Bismuto.set_hop(   /2J,3,4,[  2, 1,0] )
Bismuto.set_hop(  -/2J,3,4,[ -2,-1,0] )
```

```
Bismuto.set_hop(  -/2J*exp(-2J*pi/3),3,4,[  1, 1,0] )
Bismuto.set_hop(   /2J*exp(-2J*pi/3),3,4,[ -1,-1,0] )
Bismuto.set_hop(  -/2J*exp(-2J*pi/3),3,4,[  1, 2,0] )
Bismuto.set_hop(   /2J*exp(-2J*pi/3),3,4,[ -1,-2,0] )
```

```
#[3,6]

Bismuto.set_hop(  -*1J/2*exp(-1J*pi/3) ,3,6,[  0, 1,0] )
Bismuto.set_hop(  -*1J/2*exp(-1J*pi/3) ,3,6,[  0,-1,0] )
Bismuto.set_hop(  -*1J/2*exp(-1J*pi/3) ,3,6,[ -1, 1,0] )
Bismuto.set_hop(  -*1J/2*exp(-1J*pi/3) ,3,6,[  1,-1,0] )

Bismuto.set_hop(   *1J/2, 3,6,[  1, 0,0] )
Bismuto.set_hop(   *1J/2, 3,6,[ -1, 0,0] )
Bismuto.set_hop(   *1J/2, 3,6,[  2, 1,0] )
Bismuto.set_hop(   *1J/2, 3,6,[ -2,-1,0] )

Bismuto.set_hop(   *1J/2*exp(-2J*pi/3) ,3,6,[  1, 1,0] )
Bismuto.set_hop(   *1J/2*exp(-2J*pi/3) ,3,6,[ -1,-1,0] )
Bismuto.set_hop(   *1J/2*exp(-2J*pi/3) ,3,6,[  1, 2,0] )
Bismuto.set_hop(   *1J/2*exp(-2J*pi/3) ,3,6,[ -1,-2,0] )
```

```
#[4,4]
Bismuto.set_hop( -tII/2,4,4,[ 1,0,0] )
Bismuto.set_hop( -tII/2,4,4,[ 0,1,0] )
Bismuto.set_hop( -tII/2,4,4,[ 1,1,0] )

Bismuto.set_hop(  mII/2,4,4,[ 0,0,1] )
```

```
#[4,6]
Bismuto.set_hop( -II/2,4,6,[ 0,0, 1] )
Bismuto.set_hop(  II/2,4,6,[ 0,0,-1] )
```

```
#[4,7]
Bismuto.set_hop(  II/2J,4,7,[  1, 0,0] )
Bismuto.set_hop( -II/2J,4,7,[ -1, 0,0] )
Bismuto.set_hop(  II/2J,4,7,[  0, 1,0] )
Bismuto.set_hop( -II/2J,4,7,[  0,-1,0] )
Bismuto.set_hop( -II/2J,4,7,[  1, 1,0] )
```

```
Bismuto.set_hop(  II/2J,4,7,[ -1,-1,0] )

Bismuto.set_hop( -II/2,4,7,[  1,-1,0] )
Bismuto.set_hop(  II/2,4,7,[ -1, 1,0] )
Bismuto.set_hop(  II/2,4,7,[  2, 1,0] )
Bismuto.set_hop( -II/2,4,7,[ -2,-1,0] )
Bismuto.set_hop( -II/2,4,7,[  1, 2,0] )
Bismuto.set_hop(  II/2,4,7,[ -1,-2,0] )
```

```
#[5,5]
Bismuto.set_hop( -tII/2,5,5,[ 1,0,0] )
Bismuto.set_hop( -tII/2,5,5,[ 0,1,0] )
Bismuto.set_hop( -tII/2,5,5,[ 1,1,0] )

Bismuto.set_hop(  mII/2,5,5,[ 0,0,1] )
```

```
#[5,6]
Bismuto.set_hop(  II/2J,5,6,[  1, 0,0] )
Bismuto.set_hop( -II/2J,5,6,[ -1, 0,0] )
Bismuto.set_hop(  II/2J,5,6,[  0, 1,0] )
Bismuto.set_hop( -II/2J,5,6,[  0,-1,0] )
Bismuto.set_hop( -II/2J,5,6,[  1, 1,0] )
Bismuto.set_hop(  II/2J,5,6,[ -1,-1,0] )

Bismuto.set_hop(  II/2,5,6,[  1,-1,0] )
Bismuto.set_hop( -II/2,5,6,[ -1, 1,0] )
Bismuto.set_hop( -II/2,5,6,[  2, 1,0] )
Bismuto.set_hop(  II/2,5,6,[ -2,-1,0] )
Bismuto.set_hop(  II/2,5,6,[  1, 2,0] )
Bismuto.set_hop( -II/2,5,6,[ -1,-2,0] )
```

```
#[5,7]
Bismuto.set_hop( -II/2,5,7,[ 0,0, 1] )
Bismuto.set_hop(  II/2,5,7,[ 0,0,-1] )
```

```
#[6,6]
Bismuto.set_hop(  tII/2,6,6,[ 1,0,0] )
Bismuto.set_hop(  tII/2,6,6,[ 0,1,0] )
Bismuto.set_hop(  tII/2,6,6,[ 1,1,0] )
```

```
Bismuto.set_hop( -mII/2,6,6,[ 0,0,1] )
```

```
#[7,7]
Bismuto.set_hop(  tII/2,7,7,[ 1,0,0] )
Bismuto.set_hop(  tII/2,7,7,[ 0,1,0] )
Bismuto.set_hop(  tII/2,7,7,[ 1,1,0] )

Bismuto.set_hop( -mII/2,7,7,[ 0,0,1] )


Bismuto.set_onsite( [+mI,+mI,-mI,-mI,-+mII,-+mII,--mII,--mII] )
```

4 Transporte electrónico de bismuto

Los cálculos de transporte electrónico en bismuto a bajas energías se realizaron utilizando el $H_{\mathrm{TB}}^{Schin}()$. Con los parámetros de salto de este Hamiltoniano se construyó la geometría mostrada en la Figura~**??** en la paquetería KWANT [**?**], la cual consiste de una base hexagonal con 18 sitios a lo largo de un lado. Se evaluó el transporte electrónico con los 2 contactos conectados a la nanoestructura de bismuto, utilizando el formalismo de Landauer-B"uttiker revisado en el capítulo **??**. En dicho formalismo, la cantidad clave a calcular es la probabilidad de transmisión, la cual se evaluó utilizando el método de la matriz de dispersión implementado en esta librería.

4.1 Implementación en KWANT

```
from pylab import *
from pythtb import *
import kwant


def make_syst_Schindler(s,height):
    alat  = 1

    a1 = alat*[   1,        0,0]
    a2 = alat*[-1/2,sqrt(3)/2,0]
    a3 = alat*[   0,        0,1]

    bismuth = kwant.lattice.general( [(a1),(a2),(a3)],
                                [(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),
                                 norbs=1 )

    a,b,c,d,e,f,g,h = bismuth.sublattices

    def hexagon_1(pos):
        x, y, z = pos
        return (-s<x<=-(s/2) and -np.sqrt(3)*(s+x-0.6)<y<np.sqrt(3)*(s+x-0.6) and 0 <= z <
    #cuadrado en el centro
```

```
def hexagon_2(pos):
    x, y, z = pos
    return (-s/2<x<=s/2 and -np.sqrt(3)*(s/2)+0.5<y<np.sqrt(3)*(s/2) and 0 <= z <= hei
#triangulo derecho
def hexagon_3(pos):
    x, y, z = pos
    return (s/2<x<=s and np.sqrt(3)*(x-s+0.1)<y<np.sqrt(3)*(s-x) and 0 <= z <= height)
#hexagono completo
def hexagon(pos):
    x,y,z = pos
    return (hexagon_1(pos)+hexagon_2(pos)+hexagon_3(pos))

#triangulo izquierdo
def hexagonL_1(pos):
    x, y, z = pos
    return (-s<x<=-(s/2) and -np.sqrt(3)*(s+x-0.6)<y<np.sqrt(3)*(s+x-0.6))
#cuadrado en el centro
def hexagonL_2(pos):
    x, y, z = pos
    return (-s/2<x<=s/2 and -np.sqrt(3)*(s/2)+0.5<y<np.sqrt(3)*(s/2))
#triangulo derecho
def hexagonL_3(pos):
    x, y, z = pos
    return (s/2<x<=s and np.sqrt(3)*(x-s+0.1)<y<np.sqrt(3)*(s-x))
#hexagono del lead completo
def hexagon_lead(pos):
    x,y,z = pos
    return (hexagonL_1(pos)+hexagonL_2(pos)+hexagonL_3(pos))

def onsite(site):
     = 0.1; mI  = 2; mII = 2
    if site.family   == a:
        return +mI
    elif site.family == b:
        return +mI
    elif site.family == c:
        return -mI
    elif site.family == d:
        return -mI
    elif site.family == e:
        return -+mII
```

```
        elif site.family == f:
            return -+mII
        elif site.family == g:
            return --mII
        elif site.family == h:
            return --mII

    syst = kwant.Builder()
    syst[bismuth.shape(hexagon,(0,0,0))] = onsite


    sym0 = kwant.TranslationalSymmetry(bismuth.vec((0, 0, -1)))
    lead0 = kwant.Builder(sym0)
    lead0[bismuth.shape(hexagon_lead, (0, 0, 0))] = onsite

    sym1 = kwant.TranslationalSymmetry(bismuth.vec((0, 0, 1)))
    lead1 = kwant.Builder(sym1)
    lead1[bismuth.shape(hexagon_lead, (0, 0, 0))] = onsite

    with open('hopps_kwant_Schindler.txt', "rb") as source_file:
        code = compile(source_file.read(), 'hopps_kwant_Schindler.txt', "exec")
        exec(code)

    leads = [lead0, lead1]
    syst.attach_lead(lead0)
    syst.attach_lead(lead1)
    syst = syst.finalized()

    return syst, leads
```

```
def Conductancia(syst,energias,s,height):
  datos = []
  for energia in energias:
    matrizS = kwant.smatrix(syst,energia)
    datos.append(matrizS.transmission(0,1))

  return energias,datos
```

```
s = 3
height = 5
```

```
syst, leads = make_syst_Schindler(s,height)

energias = linspace(-0.5,0.5)

energias,datos = Conductancia(syst,energias,s,height)
```

Referencias