Cálculo de la estructura electrónica de bandas

Ricardo López & Donaldo Garrido

Tecnológico de Monterrey, Monterrey, MX

E-mail: A01066515@itesm.mx, A01275416@itesm.mx

Junio, 2021

Resumen. Este documento es una reproducción de los resultados presentados por Pavelich y Masiglio, en su artículo llamado *Calculation of 2D electronic band structure using matrix mechanics* [1]. En este trabajo se comenzó por implementar el modelo de "Kronig-Penny 2Dçon el propósito de obtener las estructuras de banda para una estructura con una red cristalina hexagonal, como el grafeno. Posteriormente, se utilizó el modelo de "moffin tin ",un modelo más realista basdo en fuerzas centrales,para comparar las estructuras de banda entre ambos modelos. Las estructuras de banda fueron generadas con un código desarrollado en Python, el cuál es mostrado en este mismo trabajo.

1. Introduction

Una capa de grafeno puro, atomos de carbono unidos etre si con una red hexagonal, es el "sistema físico de dos dimensiones por excelencia". Dichas capas pueden existir en distintas formas. Por ejemplo, varias capas apiladas de grafeno es lo que compone al grafito, los nanotubos de carbono son çapas de grafeno envueltasz los fullerenos, que son son pequeñas areas de grafenos en un estructura aproximadamente esférica. Hasta 2004, se creía que una capa extendida de grafeno no era estable en contra de los efectos termales y otras fluctuaciones; incluso si eran estables, sería imposible aislarlos para estudiar sus propiedades. Sin embargo, en ese mismo a;o Andr'e Geim y sus colegas en la Universidad de Manchester en Reino Unido demostraron que ambas creencias eran falsas. Se discutía desde hace más de 60 años atrásque la estructura de la banda electrónica del grafeno, sería particularmente interesante [2]

Este trabajo reproduce los resultados de Pavelich y Maesiglio mostrados en su artículo *Calculation of 2D electronic band structure using matrix mechanics* [1], por lo que cada una de las figuras presentadas fueron generadas con un código implentado en Python (el código se encuentra en la sección de apéndices). En dicho artículo, se generan estructuras de banda para un arreglo cuadrado "modelo 2D de Kronin-Penny ", para un potencial de pozos circulares "muffin-tinz pozos Gaussianos. Sin embargo, en el presente trabajo solo reproducimos los resultados para los dos primeros modelos mencionados, con esto se busca resaltar algunas de las propiedades electrónicas del grafeno (estructura con una red hexagonal).

2. Formalismo

Como base, se explora el formalismo matemático para estudiar la situación en una sola dimensión, para después proceder a un breve desarrollo del formalismo en dos dimensiones.

2.1. Unidimensional

Comenzamos con un potencial unidimensional $V(x)$ que satisface condiciones de frontera periódicas, con una periodicidad de ancho a. A continuación, introducimos la condición de Bloch:

$$\psi(x+a) = \psi(x) \tag{1}$$

.

Es conocido que las soluciones bases para esto, son ondas planas ortonormales:

$$\psi_n^0 = \sqrt{\frac{1}{a}} e^{i\frac{2\pi n}{a}x} \tag{2}$$

donde $n \in \mathrm{Z}$, y con energías (eigenvalores):

$$E_n^0 = 4\left(\frac{n^2\pi^2\hbar^2}{2m_0a^2}\right), \tag{3}$$

Y de donde definimos la unidad de energía como $E_{ISW} = \dfrac{\pi^2\hbar^2}{2m_0a^2}$.

Podemos introducir ahora el potencial de interez, y tratar de solucionar el problema con una estrategia matricial:

$$\sum_{m=1}^{\infty} H_{mn}c_m = Ec_n, \tag{4}$$

donde tenemos el Hamiltoniano base

$$H_0 = -\frac{\hbar^2}{2m_0}\frac{\mathrm{d}^2}{\mathrm{d}x^2}. \tag{5}$$

Con lo que expandiendo el Hamiltoniano H_{mn}, obtenemos:

$$\begin{aligned} H_{mn} &= \langle\psi_n^0|(H_0 + V(x))|\psi_m^0\rangle & (6)\\ &= \delta_{mn}E_n^0 + H_{nm}^V & (7)\end{aligned}$$

Donde podemos usar cualquiera de las dos operaciones siguientes:

$$\begin{aligned} H_{nm}^V &= \langle\psi_n^0|V(x)|\psi_m^0\rangle & (8)\\ &= \frac{1}{a}\int_0^a e^{-i\frac{2\pi n}{a}x}V(x)e^{i\frac{2\pi m}{a}x}\mathrm{d}x. & (9)\end{aligned}$$

Es importante notar que vamos a calcular los elementos matriciales y las adimensionales (i.e. $h_{nm} = H_{nm}/E_{ISW}$, E_n^0/E_{ISW}). Ahora, para pasar a la formulación

de un potencial periódico, hay que tomar en cuenta nuevamente el teorema de Bloch, que se escribe:

$$\psi(x+a) = e^{iKa}\psi(x), \tag{10}$$

con la condición $Ka \in [-\pi, \pi]$. La introducción del teorema de Bloch lleva a que tengamos la condición de frontera $ka = 2\pi n + Ka$, por lo que la energía se modifica a

$$E_n^0 = E_{ISW}\left(2n + \frac{Ka}{\pi}\right)^2. \tag{11}$$

2.2. Bidimensional

Comenzamos por introducir una celda rectangular única de medidas a_x y a_y, con las condiciones periódicas:

$$\psi(x+a_x, y) = \psi(x, y) \tag{12}$$
$$\psi(x, y+a_y) = \psi(x, y). \tag{13}$$

Realizando un procedimiento de separación de variables, obtenemos los estados base de con la siguiente forma:

$$\psi^0_{n_x n_y}(x, y) = \frac{1}{\sqrt{a_x a_y}} e^{i\frac{2\pi n_x}{a_x}x + i\frac{2\pi n_y}{a_y}y} \text{donde} n_x, n_y \in \text{Z}. \tag{14}$$

Las energías propias del sistema son:

$$E^0_{n_x n_y} = a\left[n_x^2 + n_y^2\left(\frac{a_x^2}{a_y^2}\right)\right] E_{ISW} \tag{15}$$
$$= E_{n_x}^{\ 0} + E_{n_y}^{\ 0}. \tag{16}$$

Donde definimos ahora $E_{ISW} = \dfrac{\pi^2\hbar^2}{2m_0 a_x^2}$ como la energía unidad.
Dado esto, los elementos de la matriz hamiltoniana pueden ser calculados de la forma:

$$H_{n_x n_y, m_k m_y} = \left\langle \psi^0_{n_x n_y} \left| (H_0 + V(x,y)) \right| \psi^0_{m_x m_y} \right\rangle \tag{17}$$
$$= \delta_{n_x m_x}\delta_{n_y m_y} E^0_{n_x n_y} + H^V_{n_x n_y, m_x m_y} \tag{18}$$

y explícitamente, tenemos:

$$H^V_{n_x n_y, m_x m_y} = \left\langle \psi^0_{n_x n_y} \left| (V(x,y)) \right| \psi^0_{m_x m_y} \right\rangle. \tag{19}$$

Ahora, para pasar a el potencial periódico que deseamos, imponemos la condición de Bloch y obetenemos la periodicidad:

$$\psi(x+a_x, y) = e^{iK_x a_x}\psi(x, y) \tag{20}$$
$$\psi(x, y+a_y) = e^{iK_y a_y}\psi(x, y). \tag{21}$$

Con esto, al igual que en el caso unidimensional, tenemos afecciones sólo en la parte cinética del hamiltoniano:

$$E_{n_x}^{\ 0} = E_{ISW}\left(2n_x + \frac{K_x a_x}{\pi}\right)^2 \tag{22}$$

$$E_{n_y}^{\ 0} = E_{ISW}\left(2n_y + \frac{K_y a_y}{\pi}\right)^2 \left(\frac{a_x^2}{a_y^2}\right) \tag{23}$$

$$\tag{24}$$

3. Modelo Kronig-Penney en 2D

Trabajaremos ahora con el model de Kronig-Penney que consta de una celda unitaria y cuadrada de lado a, se tiene un cascarón con una profundidad V_0 ($v_0 = V_0/E_{ISW}$) de valor negativo. El pozo se extiende sobre q_1 y q_2 de forma que $0 \leq q_1 \leq q_2 \leq a$. Esto es:

$$V(x,y) = \begin{cases} \mathrm{V}_0 & \text{para } \mathrm{q}_1 < x < q_2 \mathrm{y} q_1 < y < q_2 \\ 0 & \text{todo lo demás} \end{cases} \tag{25}$$

De forma que no tengamos dimensiones de unidad, normalizamos todo como hasta el momento haciendo el cambio de variable $p_1 = q_1/a, p_2 = q_2/a$. Los elementos de la diagonal en este caso se calculan con:

$$H^V_{n_x n_y, m_x m_y} = \frac{V_0}{a^2}\int_{q_1}^{q_2}\int_{q_1}^{q_2} e^{i2\pi(m_x - n_x)x/a} e^{i2\pi(m_y - n_y)y/a} dxdy, \tag{26}$$

o lo podemos escribir en forma adimensional:

$$h^V_{n_x n_y, m_x m_y} = v_0 \int_{p_1}^{p_2}\int_{p_1}^{p_2} e^{i2\pi(m_x - n_x)x} e^{i2\pi(m_y - n_y)y} dxdy, \tag{27}$$

Resolviendo las integrales anteriores, obtenemos:

$$h^V_{n_n n_y, m_x m_y} = v_0 I\left(n_x, m_x\right) I\left(n_y, m_y\right), \tag{28}$$

donde,

$$I\left(n_j, m_j\right) = \left(p_2 - p_1\right)\delta_{n_j m_j} + i\frac{e^{i2\pi(m_j - n_j)p_1} - e^{i2\pi(m_j - n_j)p_2}}{2\pi\left(m_j - n_j\right)}\left(1 - \delta_{n_j m_j}\right). \tag{29}$$

donde $j = x\mathrm{y}$. Es importante notar que los los elementos no cinéticos de la diagonal principal son $v_0(p_2 - p_1)^2$.

3.1. Límite del electrón aproximadamente libre

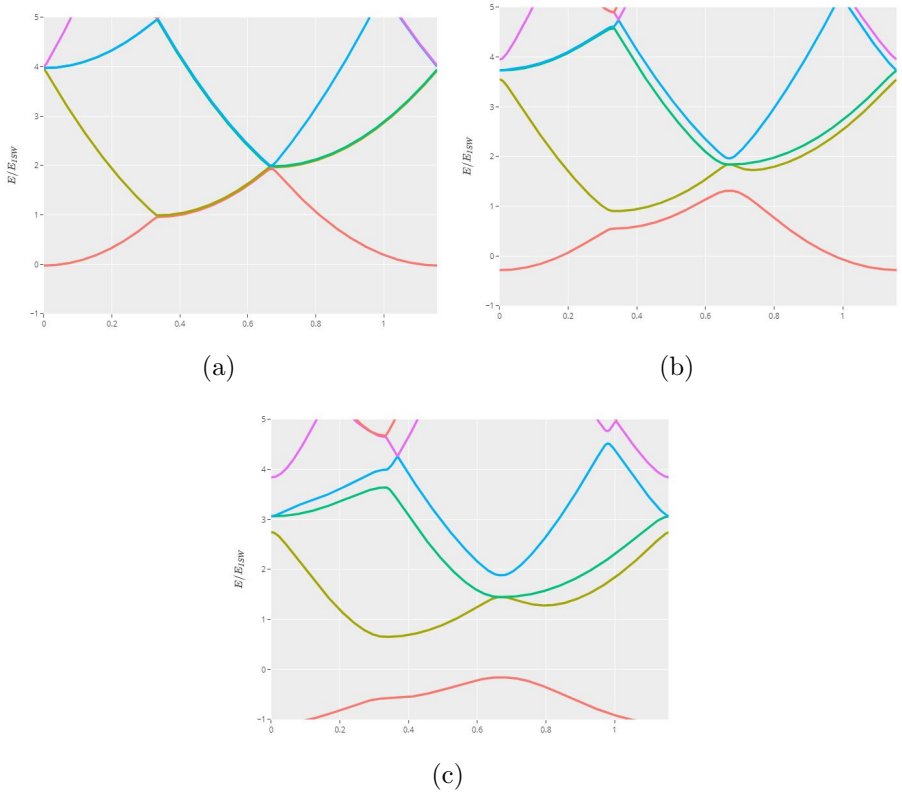
Este enfoque genera la llamada aproximación de red vacía con bandas parabólicas de electron libre, esto sucede cuanto $v_0 = 0$. Además, este método provee un esquema de codificación para las bandas de energía en términos de la base de estados elegida. Cuando v_0 tiene valore pequeños, se tiene casi un modelo de electrón libre, del cual una de sus principales características son las degeneracias donde los gaps de banda surgen, y las afiladas cuspides para $v_0 = 0$ se convierten en parabolas suaves, como se puede ver en las primeras las figura 1 (a). Además la fig.1(c) se muestra la estructura de banda para un pozo con una profundidad más grande , donde el gap de energía entre la banda baja y las otras bandas existen para todos los vectores de onda.

3.2. Límite del modelo de atadura fuerte

Por otro lado, se pueden tener pozos de potencial con una gran profundidad, en este caso se debe considerar el modelo de atadura fuerte. Para sistemas bidimensionales como una simetría cuadrada y con interacciones de primeros vecinos, las bandas de energía tienen la siguiente forma muy bien conocida

$$\mathbf{E}(\mathbf{K_x}, \mathbf{K_y}) = -2t\left[\cos(\mathbf{K_x a}) + \cos(\mathbf{K_y a})\right] \tag{30}$$

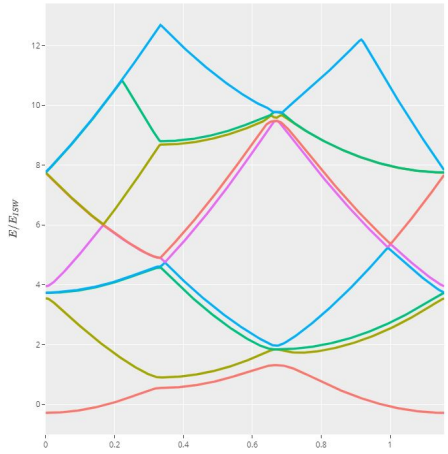
donde t es la llamada integral de salto y representa una probabilidad de amplitud para que un electrón tenga un tunelamiento al pozo vecino.



(a) (b)

(c)

Figura 1: Modelo de kroning-Penney con distintas profundidades v del pozo (a) v = -0.1 (b) v = -1 (c) v = -3



(a)

Figura 2: Modelo de kroning-Penney con una profundidad del pozo v = -1 (Mostrando un rango mayor en el eje de la energía)

4. Potencial de molde para muffins

En está sección se presentan las estructuras de banda resultantes de utilizar el modelo de muffin, el cual es un potencial más realista. Pues los potenciales realistas surgen a través de fuerzas centrales. Como se mencionó anteriormente, el modelo de "Kroning Penney en 2D.[es] una serie de pozos de potencial cuadrados repetidos en una red cuadrada. Por el otro lado, el modelo de "potencial muffin-tin.[es] unos con pozos circulares. La matriz para el modelo queda definida de la siguente manera, considerando $a_x = a_y = a$

$$H^V_{n_x n_y, m_x m_y} = \frac{V_0}{a^2} \int_{\frac{a}{2}-r}^{\frac{a}{2}+r} dx \int_{\frac{a}{2}-\sqrt{r^2-\left(x-\frac{a}{2}\right)^2}}^{\frac{a}{2}+\sqrt{r^2-\left(x-\frac{a}{2}\right)^2}} e^{i2\pi(m_x-n_x)x/a} e^{i2\pi(m_y-n_y)y/a} dy \tag{31}$$

Los limites de integracion vienen de la ecuación de un circulo con radio r centrado en $(x, y) = (a/2, a/2)$.

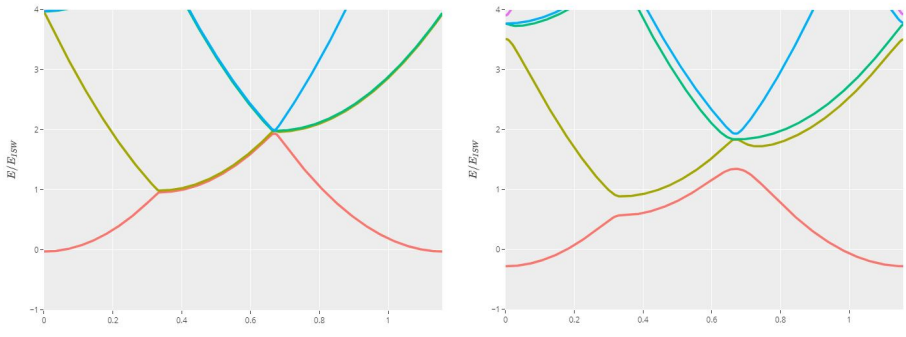
$$\left(x-\frac{a}{2}\right)^2 + \left(y-\frac{a}{2}\right)^2 = r^2. \tag{32}$$

Como antes, la profundidad del pozo es V_0 y tipicamente tendrá un valor despreciable. Haciendo las siguientes sustituciones $x/a \to x, y/a \to y$,y definiendo $\bar{r} \equiv r/a$, we get

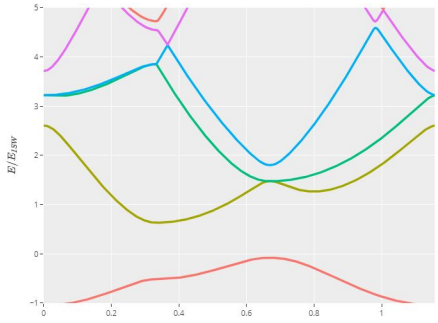
$$h^V_{n_x n_y, m_x m_y} = v_0 \int_{\frac{1}{2}-\bar{r}}^{\frac{1}{2}+\bar{r}} \mathrm{d}x \int_{\frac{1}{2}-\sqrt{r^2-\left(x-\frac{1}{2}\right)^2}}^{\frac{1}{2}+\sqrt{\bar{r}^2-\left(x-\frac{1}{2}\right)^2}} \mathrm{d}y e^{i2\pi(m_x-n_x)x} e^{i2\pi(m_y-n_y)y} \mathrm{d} \tag{33}$$

La integral anterior fue resuleta numéricamente para poder armar la matriz, dicho proceso está mostrado en el código de Python presentado en los apéndices. Una vez armada la integlar, los procesos númericos eran exactamente iguales a los del modelo

anterior, de está manera se generaron las estructuras de banda para el modelo de moffin mostradas en la fig.3-4

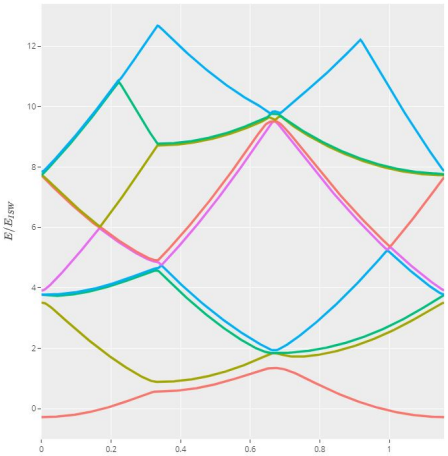


(a) (b)



(c)

Figura 3: Modelo de muffin con distintas profundidades v del pozo (a) v = -0.1 (b) v = -1 (c) v = -3



(a)

Figura 4: Modelo de muffin con una profundidad del pozo v = -1 (Mostanto un rango mayor en el eje de energía)

5. Conclusiones

Como se vio anteriormente, logramos reproducir los resultados mostrados por Pavelich y Masiglio presentados en su artículo *Calculation of 2D electronic band structure using matrix mechanics*. Dichos resultados parecen muy similares a simple vista, pues tienen variaciones muy pequeñas. Entonces, se mostró que el modelo de "Kroning-Penney 2D"se aproxima bastante bien a modelos más realistas como el modelo de "moffin". Ambos modelos fueron similares a la hora de la implementación numérica, a excepción de definir la matriz $h^V_{n_x n_y, m_x m_y}$. Ya que para el modelo de moffin fue necesario resolver una integral doble numéricamente de forma iterativa para llenar la matriz con todos sus elementos. Este trabajo se puede extender considerando los potenciales Gaussianos, una reproducción de resultados que podría llevarse a cabo en futuros proyectos.

6. Referencias

[1] R. Pavelich and F. Marsiglio, "Calculation of 2d electronic band structure using matrix mechanics," *American Journal of Physics*, vol. 84, 02 2016.

[2] M. H. R. Jamie H. Warner, Franziska Schäffel and A. Bachmatiuk, "Graphene fundamentals and emergent applications," *Elsevier Inc.*, 2013.

Apéndice A: Código para las gráficas de las bandas de energía

```
# -*- coding: utf-8 -*-
"""Bands.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1NTGRseePmGlswtKTczHR2VPiXVqTeYUs
"""

# Importación de Librerías
import numpy as np
import pandas as pd
from scipy import io, integrate, linalg, signal
from scipy.sparse.linalg import eigs
import math
import matplotlib
import plotly.graph_objects as go
import plotly.express as px
import plotly.io as pio

#pio.templates.default = "ggplot2"

# Definición de la función para potenciales cuadrados
# Modelo de Kronig-Penney
def Periodic_Potential(N,V_0,p_1,p_2):
  #Generación primaria de los enteros
  jj = np.arange(N+1)+1
  # Re-escritura de los enteros
  # Método descrito en el paper
  a = (1+(2*jj-1)*(-1)**jj)/4

  # Generación de combinaciones con meshgrid
  p, q = np.meshgrid(a,a)

  n = np.concatenate((q.reshape(len(a)**2, 1),p.reshape(len(a)**2, 1))
    ,axis = 1)

  nx = n[:,0]
  ny = n[:,1]

  # Incremento de codificar energía
  nSqr = np.zeros(((N+1)*(N+1),1))
  n = np.concatenate((nSqr,n),axis = 1)

  n[:,0] = n[:,1]*n[:,1] + n[:,2]*n[:,2];

  index = np.argsort(n[:,0], kind="mergesort")

  n = n[index,:]
  nx = n[:,1]
  ny = n[:,2]
```

```
  n2 = n[:,0]
  # Definición de la diagonal del Hamiltoniano
  p = p_2 - p_1
  h = np.diag(4*n2+V_0*p*p)

  # Precálculo de los elementos fuera de la diagonal
  No_diag = np.zeros((2*N+1))
  for ii in range(2*N+1):
    if ii == N:
      No_diag[ii] = np.nan
    else:
      No_diag[ii] = 1/(2*np.pi*1j*(ii-N))*(-np.exp(1j*2*np.pi*(ii-N)*
   p_1) + np.exp(1j*2*np.pi*(ii-N)*p_2))
  # Cálculo de los elementos fuera de la diagonal
  II = 0
  JJ = 0
  for kk in range((N+1)*(N+1)):
    for ll in range(kk+1,(N+1)**2):
      if nx[kk] == nx[ll]:
        II = p;
      else:
        II = No_diag[int(nx[kk]-nx[ll])+N+1-1]

      if ny[kk] == ny[ll]:
        JJ = p;
      else:
        JJ = No_diag[int(ny[kk]-ny[ll])+N+1-1]
      h[kk,ll] = h[kk,ll] + V_0*II*JJ
  h = h + np.transpose(np.triu(h,k=1))

  return h

def muffinPotential(N,V_0,p_1,p_2):
  #Generación primaria de los enteros
  jj = np.arange(N+1)+1
  # Re-escritura de los enteros
  # Método descrito en el paper
  a = (1+(2*jj-1)*(-1)**jj)/4

  # Generación de combinaciones con meshgrid
  p, q = np.meshgrid(a,a)

  n = np.concatenate((q.reshape(len(a)**2, 1),p.reshape(len(a)**2, 1))
    ,axis = 1)

  nx = n[:,0]
  ny = n[:,1]

  # Incremento de codificar energía
  nSqr = np.zeros(((N+1)*(N+1),1))
  n = np.concatenate((nSqr,n),axis = 1)

  n[:,0] = n[:,1]*n[:,1] + n[:,2]*n[:,2];

```

```
  index = np.argsort(n[:,0], kind="mergesort")

  n = n[index,:]
  nx = n[:,1]
  ny = n[:,2]
  n2 = n[:,0]

  # Definición de la diagonal del Hamiltoniano
  p = p_2 - p_1
  h = np.diag(4*n2+V_0*p*p)

  # Número de puntos para el rango de integración
  pointsInt = 100
  L = 1
  # Definición de los vectores para integrar
  x = np.linspace(0,L,pointsInt)
  y = x.copy()
  # Mesh para zona de integración
  X, Y = np.meshgrid(x,y)
  area = np.zeros((pointsInt,pointsInt))
  # Radio del círculo de la zona de integración
  R = p/2

  # En el círculo de integración, se colocan 1s,
  # fuera del círculo se quedan los 0s
  for ii in range(len(x)):
    for jj in range(len(y)):
      r = np.sqrt((x[ii]-L/2)**2+(y[jj]-L/2)**2)
      if r < R:
        area[ii,jj] = 1

  #fig = go.Figure(data=[
  #  go.Surface(z=area,x = x, y=y),
  #  ])
  #fig.show()
  # Cálculo de los elementos del hamiltoniano
  for kk in range((N+1)*(N+1)):
    for ll in range(kk+1,(N+1)**2):
      # Función a integrar
      F = np.exp(1j*2*np.pi*(nx[kk]-nx[ll])*X)*np.exp(1j*2*np.pi*(ny[
   kk]-ny[ll])*Y)
      # Convertir los valores fuera de la región a 0
      F = np.real(F)*area
      # Integración
      element = np.trapz(np.trapz(F, y, axis=0), x, axis=0)
      h[kk,ll] = element*V_0
  # Relleno de la matriz triangular
  h = h + np.transpose(np.triu(h,k=1))
  return h

# Definición de constantes importantes
Nest = 5 # Npumero de estados
v0 = 0 #Profundidad del pozo
P1 = 0.25 #Inicio del pozo
```

```
156 P2 = 0.75 #Final del pozo
157 repetition = 100 #Número de iteraciones
158
159 E = np.zeros((repetition+1,repetition+1,(Nest+1)**2)) #Inicialización
    Energías
160
161 # Se descomenta la función a usar según el modelo deseado,
162 # Kronig-Penney o Muffin:
163
164 #dimLessHam = Periodic_Potential(Nest,v0,P1,P2)
165 dimLessHam = muffinPotential(Nest,v0,P1,P2)
166
167 jj = np.arange(Nest+1)+1
168
169 # Re-escritura de los enteros
170 # Método descrito en el paper
171 a = (1+(2*jj-1)*(-1)**jj)/4
172
173 # Generación de combinaciones con meshgrid
174 p, q = np.meshgrid(a,a)
175
176 n = np.concatenate((q.reshape(len(a)**2, 1),p.reshape(len(a)**2, 1)),
    axis = 1)
177
178 nx = n[:,0]
179 ny = n[:,1]
180
181 # Incremento de codificar energía
182 nSqr = np.zeros(((Nest+1)*(Nest+1),1))
183 n = np.concatenate((nSqr,n),axis = 1)
184
185 n[:,0] = n[:,1]*n[:,1] + n[:,2]*n[:,2];
186
187 index = np.argsort(n[:,0], kind="mergesort")
188
189 n = n[index,:]
190 nx = n[:,1]
191 ny = n[:,2]
192 n2 = n[:,0]
193
194 # Definición de los vectores kx, ky
195 kx = np.arange(0,np.pi+np.pi/repetition,np.pi/repetition)
196 ky = kx.copy()
197
198 # Paametro para indexar la matriz
199 m = np.arange((Nest+1)**2)
200
201 # Cálculo de los elementos de Bloch
202 # Para metrizando de Gamma a X'
203 for ii in range(repetition+1):
204   bloch = (4/np.pi)*(nx[m]*kx[0]+ny[m]*ky[ii])+(kx[0]*kx[0]+ky[ii]*ky[
      ii])/(np.pi**2)
205
206   eigValues, eigVectors = np.linalg.eig(dimLessHam+np.diag(bloch,k =
```

```
    0))
  eig_or = np.sort(eigValues, kind="mergesort")
  E[0,ii,:] = eig_or

# Cálculo de los elementos de Bloch
# Para metrizando de X' a M
for ii in range(repetition+1):
  bloch = (4/np.pi)*(nx[m]*kx[ii]+ny[m]*ky[repetition])+(kx[ii]*kx[ii
    ]+ky[repetition]*ky[repetition])/(np.pi**2)
  eigValues, eigVectors = np.linalg.eig(dimLessHam+np.diag(bloch,k =
    0))
  eig_or = np.sort(eigValues, kind="mergesort")
  E[ii,repetition,:] = eig_or

# Cálculo de los elementos de Bloch
# Para metrizando de M a Gamma
for ii in range(repetition,-1,-1):
  bloch = (4/np.pi)*(nx[m]*kx[ii]+ny[m]*ky[ii])+(kx[ii]*kx[ii]+ky[ii]*
    ky[ii])/(np.pi**2)
  eigValues, eigVectors = np.linalg.eig(dimLessHam+np.diag(bloch,k =
    0))
  eig_or = np.sort(eigValues, kind="mergesort")
  E[ii,ii,:] = eig_or

# Energías para graficar
En2Plot = np.zeros((3*repetition,(Nest+1)**2))
K = np.arange(0,1+1/(3*repetition),1/(3*repetition))

# Se agrega un término de escala (sqrt(2))
K1 = K[0:2*repetition]
K2 = K[2*repetition+1:]
K = np.concatenate((K1,K2+np.sqrt(2)*(1/3)*(K2-(2/3))),axis = 0)

# Llenado de las energías para graficar
for ii in range(repetition):
  En2Plot[ii,:] = E[0,ii,:]
  En2Plot[ii+repetition,:] = E[ii,repetition,:]
  En2Plot[ii+2*repetition,:] = E[repetition-ii,repetition-ii,:] #Dudas

# Grafica de las bandas de energía
fig = px.line(x=K, y=En2Plot[:,0], labels={'x': "", 'y':'$E/E_{ISW}$'
    }) # override keyword names with labels
for i in range(8):
  fig.add_trace(go.Scatter(mode = "lines", x = K, y = En2Plot[:,i+1]))
  fig.update_layout(width = 500, height = 400)
  fig.update_traces(line = dict(width = 3.5))
  fig.update_yaxes(range = [-1,5])
  fig.update_xaxes(range = [0,1.1571])
fig.show()
```

Listing 1: Código en Python: Curvas de bandas de energía.