Presentación de proyecto

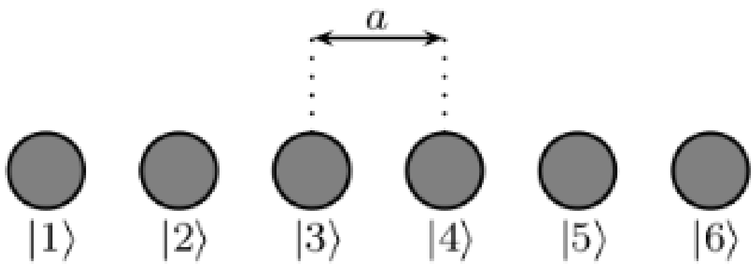
V. MODELO DE TIGHT BINDING EN UNA CADENA MONOATÓMICA CON UN ORBITAL POR ÁTOMO: CÁLCULO DE BANDAS ELECTRÓNICAS.

Barbara Chassoul- B01704

Isaac Prado- B96093

Stephanie Chaves- B92158

Maria Ramirez- B96394

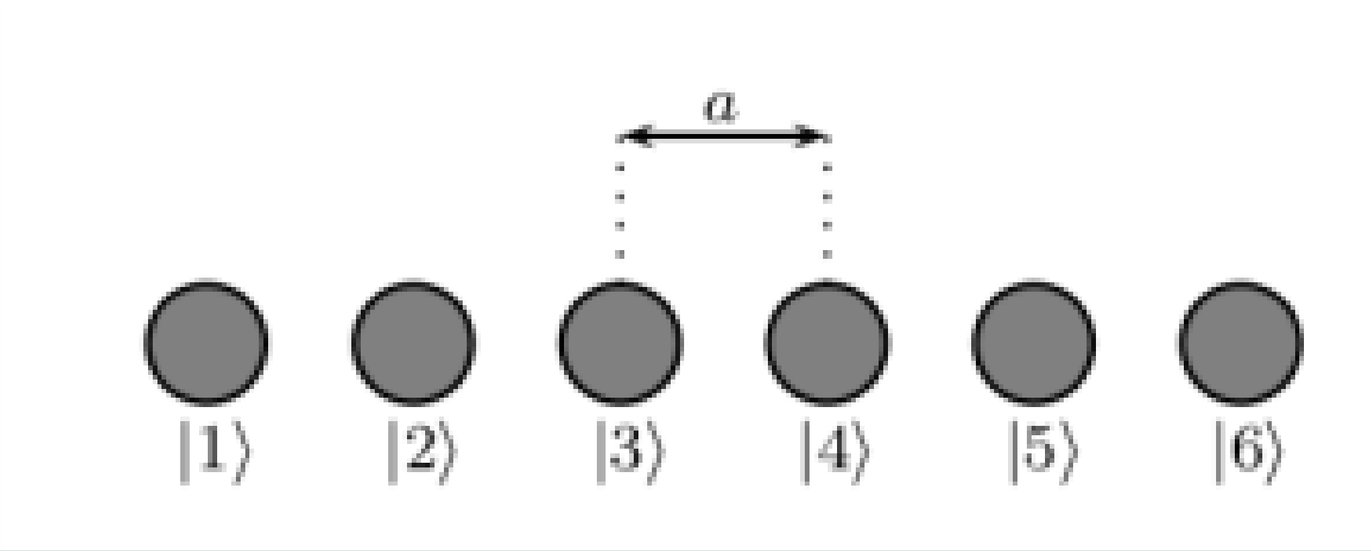


Contenido

Teoría

Modelo de Tight Binding

Sistema de cadena Monoatómica



Orbitales atómicos |n>

Simon, S. The Oxford Solid State Basics, First Edition, Oxford Clarendon Street, [2013]

Teoría

Combinación lineal de orbitales atómicos

$$|k\rangle = \frac{1}{\sqrt{N}} \sum_{n=1}^{N} e^{inka} |n\rangle$$

Elementos de la matriz hamiltoniana

$$\langle n|H|n\rangle = E_0 = E_i - U$$

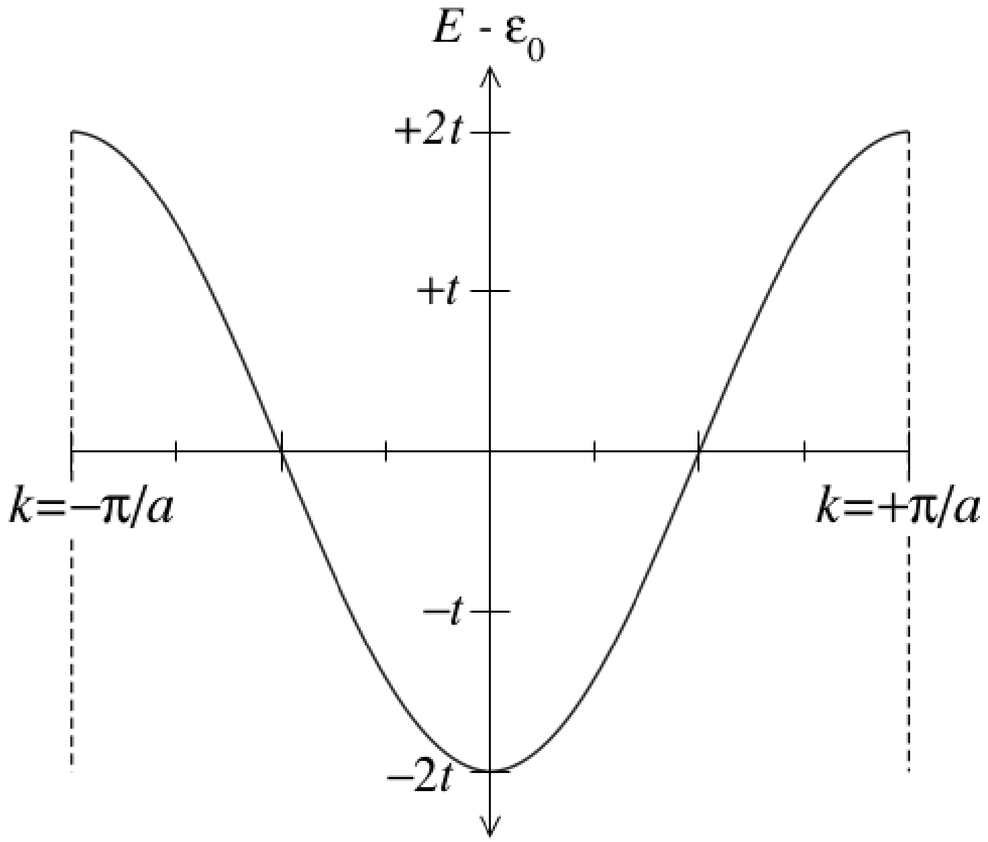
$$\langle n \pm 1|H|n\rangle = -t$$

Diagonalización de la matriz hamiltoniana

E(k)= Eo-2 t cos(ka)

Teoría

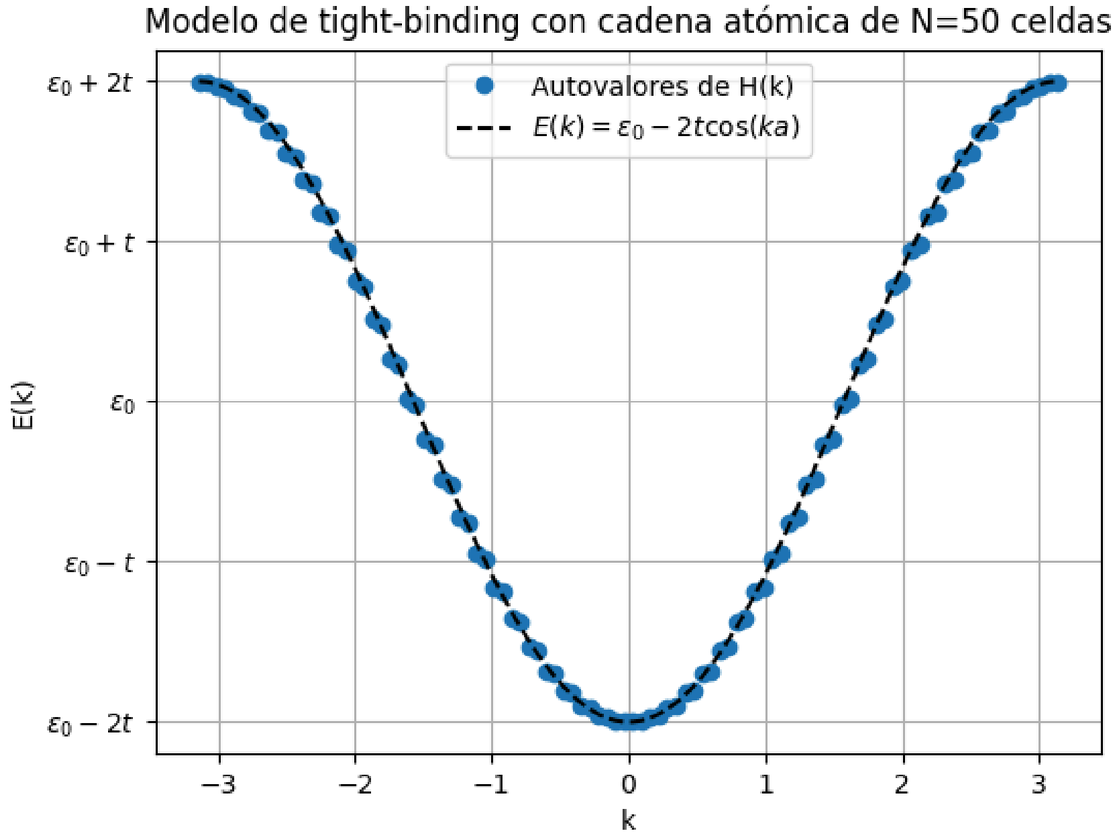
Comportamiento esperado a partir de la teoría conocida



Simon, S. The Oxford Solid State Basics, First Edition, Oxford Clarendon Street, [2013]

Resultado

Gráfica obtenida para N = 50 celdas



Código

```python
def calcular_estructura_de_bandas(N, a, t, eps, valores_k):
    autovalores = []

    for k in valores_k:
        H = np.zeros((N, N), dtype=complex)

        # Término de energía en el sitio
        for i in range(N):
            H[i, i] = eps

        # Términos de hopping con condiciones de contorno periódicas
        for i in range(N):
            H[i, (i+1) % N] = -t * np.exp(1j * k * a)
            H[(i+1) % N, i] = -t * np.exp(-1j * k * a)

        evals = np.linalg.eigvalsh(H)
        autovalores.append(evals)

    return autovalores
```

$$H(k) = \begin{pmatrix} \epsilon_0 & -te^{ika} & 0 & \cdots & 0 & -te^{-ika} \\ -te^{-ika} & \epsilon_0 & -te^{ika} & \cdots & 0 & 0 \\ 0 & -te^{-ika} & \epsilon_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \epsilon_0 & -te^{ika} \\ -te^{ika} & 0 & 0 & \cdots & -te^{-ika} & \epsilon_0 \end{pmatrix}$$

```python
def seleccionar_autovalores_aleatorios(autovalores):
    indice_aleatorio = random.randint(0, len(autovalores) - 1)
    return autovalores[indice_aleatorio]
```

```python
def aplicar_condiciones_de_contorno_periodicas(autovalores):
    return np.concatenate((np.flip(autovalores, 0), autovalores), axis=0)
```

```python
def calcular_banda_analitica(valores_k, t, eps, a):
    return eps - 2 * t * np.cos(valores_k * a)
```

```python
def graficar_estructura_de_bandas(valores_k, banda_numerica, t, eps, a, N):
    banda_analitica = calcular_banda_analitica(valores_k, t, eps, a)

    fig, axs = plt.subplots()
    axs.plot(valores_k, banda_numerica, 'o', label=f'Autovalores de H(k)')
    axs.plot(valores_k, banda_analitica, label=r'$E(k) = \epsilon_0 - 2t\cos(ka)$', linestyle='--', color='black')
    axs.set_yticks([eps-2.*t, eps-t, eps, eps+t, eps+2.*t])
    axs.set_yticklabels(['$\epsilon_0-2t$', '$\epsilon_0-t$', '$\epsilon_0$', '$\epsilon_0+t$', '$\epsilon_0+2t$'])
    axs.set_xlabel('k')
    axs.set_ylabel('E(k)')
    axs.set_title(f'Modelo de tight-binding con cadena atómica de N={N} celdas')
    axs.legend()
    axs.grid(True)

    fig.savefig('tight_binding.png')
```

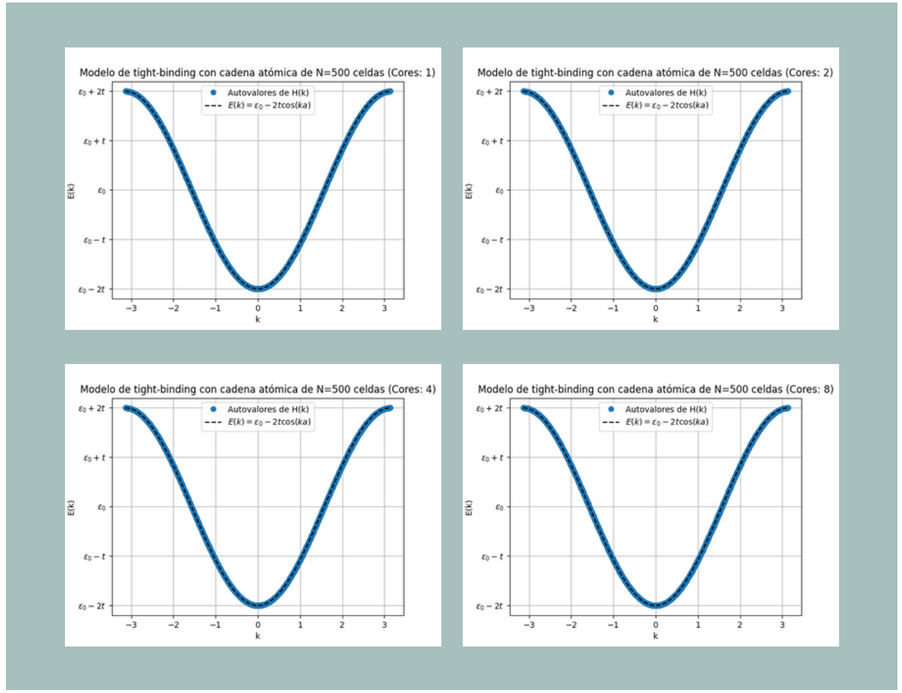
```python
# Parámetros
N = 8      # Número de orbitales atómicos
a = 1       # Espaciado de la red
t = 0.5     # Parámetro de hopping
eps = 0.0   # Energía en el sitio
max_iterations = 2 * N
valores_k = np.linspace(-np.pi/a, np.pi/a, max_iterations)  # Array de valores k

# Calcular la estructura de bandas
autovalores = calcular_estructura_de_bandas(N, a, t, eps, valores_k)

# Seleccionar un conjunto arbitrario de autovalores para graficar
autovalores_seleccionados = seleccionar_autovalores_aleatorios(autovalores)

# Aplicar las condiciones de contorno periódicas
autovalores_con_ccp = aplicar_condiciones_de_contorno_periodicas(autovalores_seleccionados)

# Graficar la estructura de bandas
graficar_estructura_de_bandas(valores_k, autovalores_con_ccp, t, eps, a, N)
```

Código [en paralelo]

```python
import matplotlib.pyplot as plt
from multiprocessing import Pool, cpu_count
import argparse
import time
```

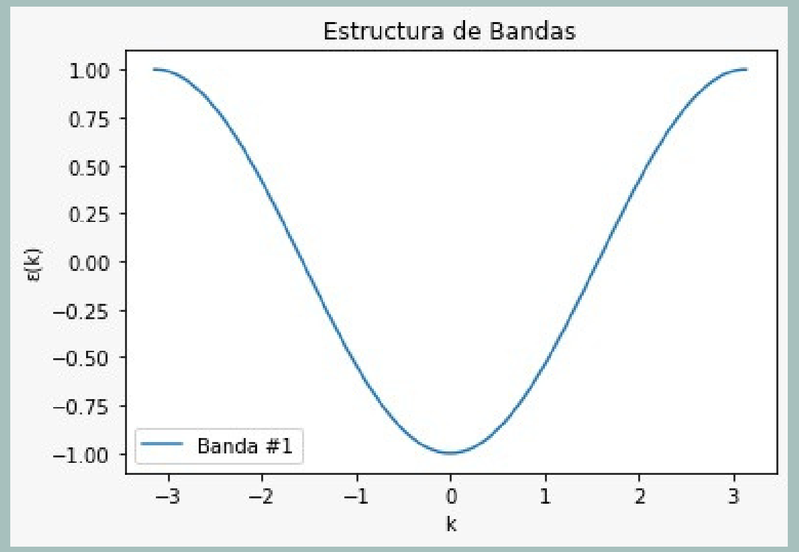
```python
def calcular_estructura_de_bandas(N, a, t, eps, valores_k, num_cores):
    with Pool(num_cores) as p:
        autovalores = p.map(calcular_estructura_de_bandas_para_k, [(k, N, a, t, eps) for k in valores_k])

    return autovalores
```

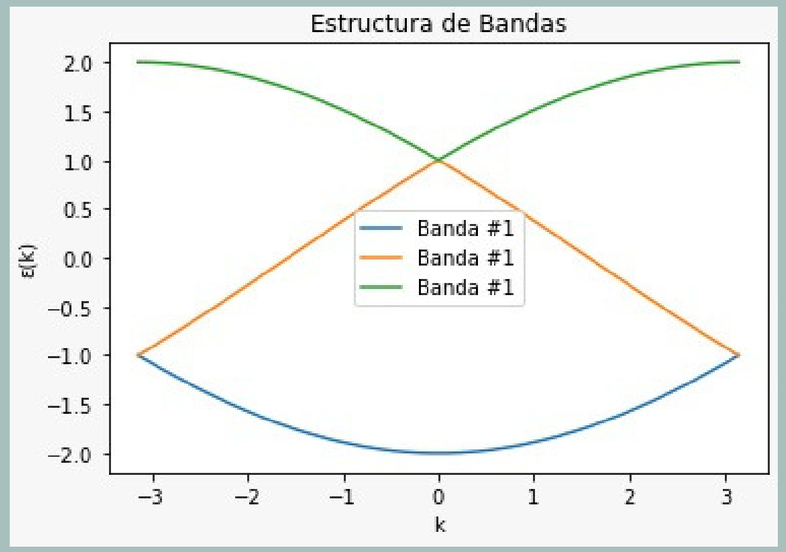
```python
def main():
    # Definir los argumentos de línea de comandos
    parser = argparse.ArgumentParser(description='Modelo de tight-binding en 1D')
    parser.add_argument('--cores', type=int, default=cpu_count(), help='Número de núcleos a utilizar')

    args = parser.parse_args()
    num_cores = args.cores

    # Parámetros
    N = 500     # Número de orbitales atómicos
    a = 1       # Espaciado de la red
    t = 0.5     # Parámetro de hopping
    eps = 0.0   # Energía en el sitio
    max_iterations = 2 * N
    valores_k = np.linspace(-np.pi/a, np.pi/a, max_iterations)  # Array de valores k

    # Medir el tiempo de ejecución
    start_time = time.time()

    # Calcular la estructura de bandas
    autovalores = calcular_estructura_de_bandas(N, a, t, eps, valores_k, num_cores)

    # Seleccionar un conjunto arbitrario de autovalores para graficar
    autovalores_seleccionados = seleccionar_autovalores_aleatorios(autovalores)

    # Aplicar las condiciones de contorno periódicas
    autovalores_con_ccp = aplicar_condiciones_de_contorno_periodicas(autovalores_seleccionados)

    # Graficar la estructura de bandas
    graficar_estructura_de_bandas(valores_k, autovalores_con_ccp, t, eps, a, N, num_cores)

    # Calcular y mostrar el tiempo de ejecución
    end_time = time.time()
    execution_time = end_time - start_time
    print(f'Tiempo de ejecución: {execution_time} s')

if __name__ == '__main__':
    main()
```

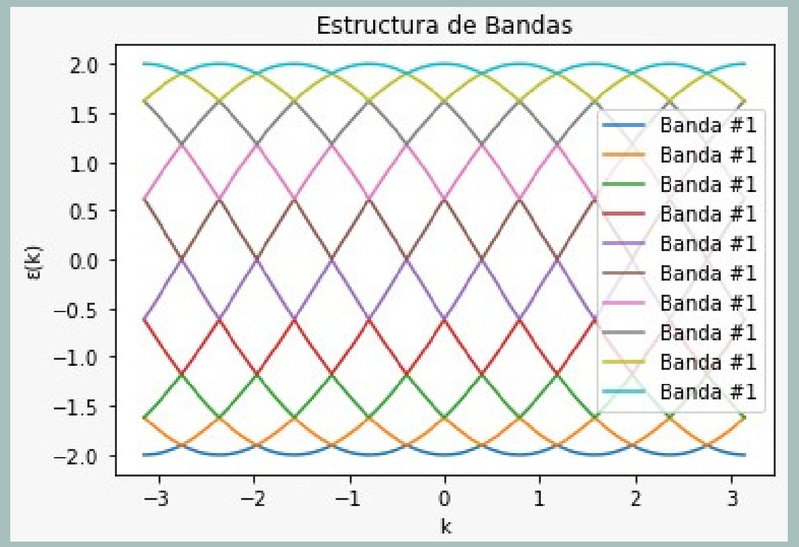


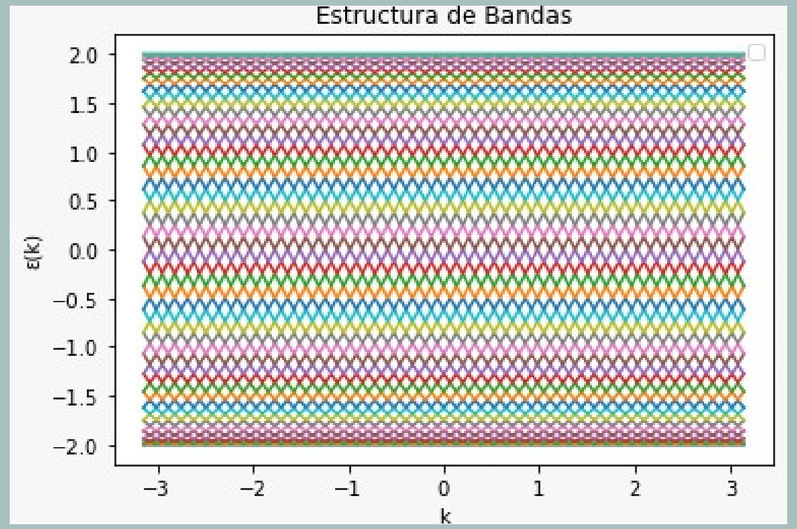

```
● Barbaras-iMac:tight_binding main $ python3 tight_binding_parallel.py --cores 1
  Tiempo de ejecución: 23.329487800598145 s
● Barbaras-iMac:tight_binding main $ python3 tight_binding_parallel.py --cores 2
  Tiempo de ejecución: 17.519796133041382 s
● Barbaras-iMac:tight_binding main $ python3 tight_binding_parallel.py --cores 4
  Tiempo de ejecución: 18.517837047576904 s
● Barbaras-iMac:tight_binding main $ python3 tight_binding_parallel.py --cores 8
  Tiempo de ejecución: 15.589223861694336 s
○ Barbaras-iMac:tight_binding main $
```

Errores

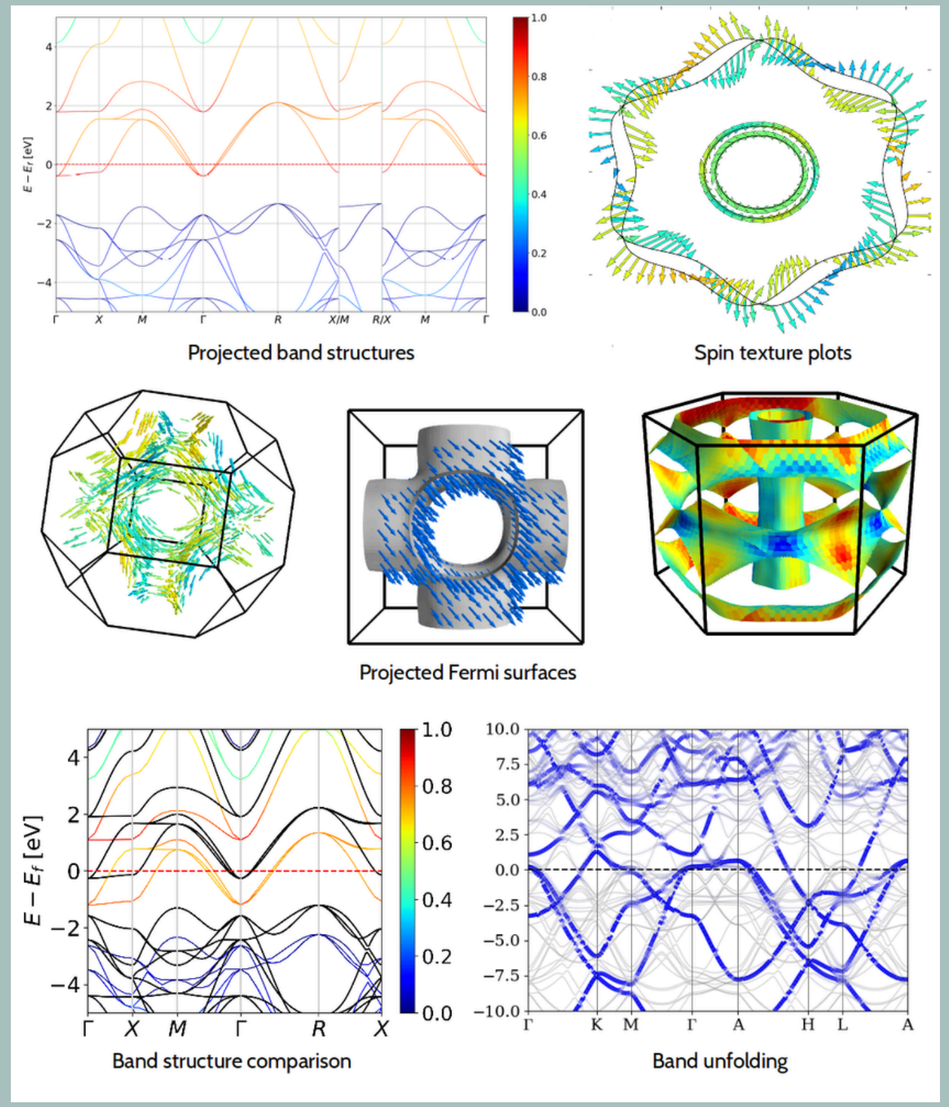

Estructura de Bandas
ε(k)
k
Banda #1

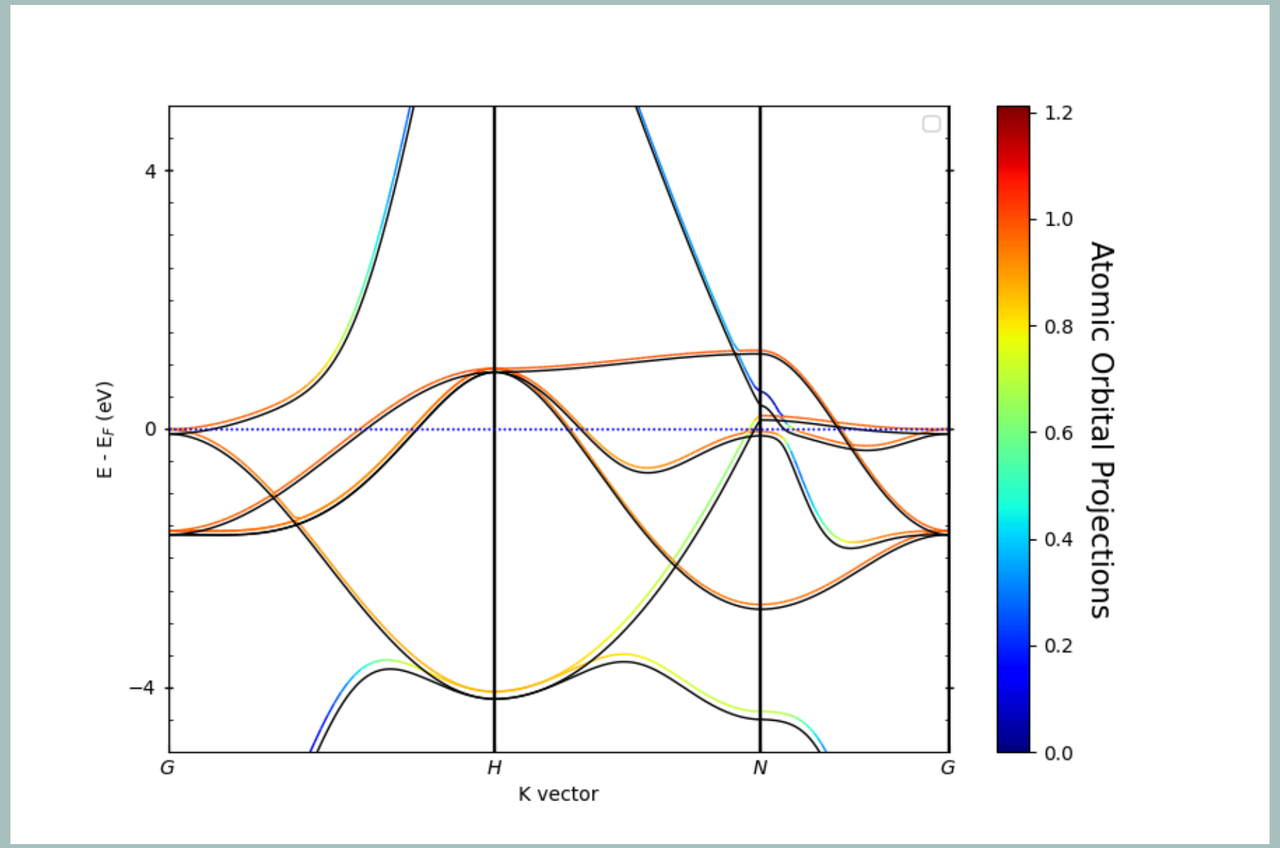

Estructura de Bandas
ε(k)
k
Banda #1
Banda #1
Banda #1


Estructura de Bandas
ε(k)
k
Banda #1
Banda #1
Banda #1
Banda #1
Banda #1
Banda #1
Banda #1
Banda #1
Banda #1
Banda #1

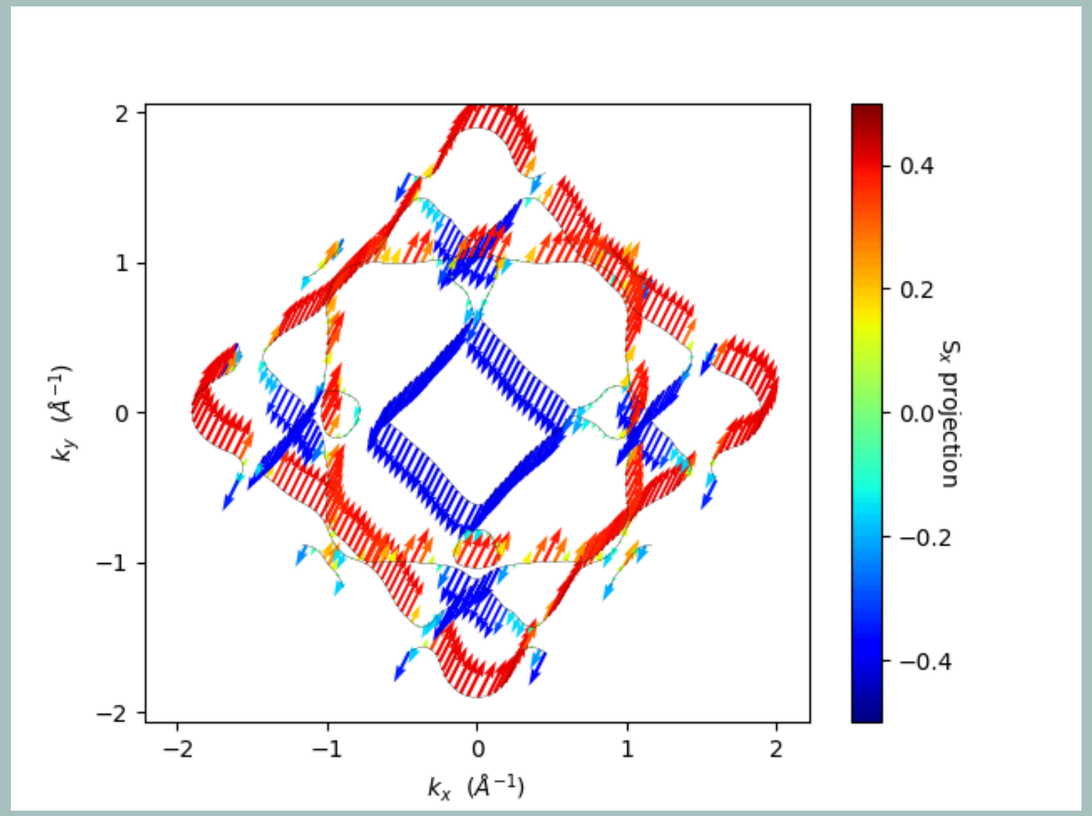

Estructura de Bandas
ε(k)
k

PyProcar

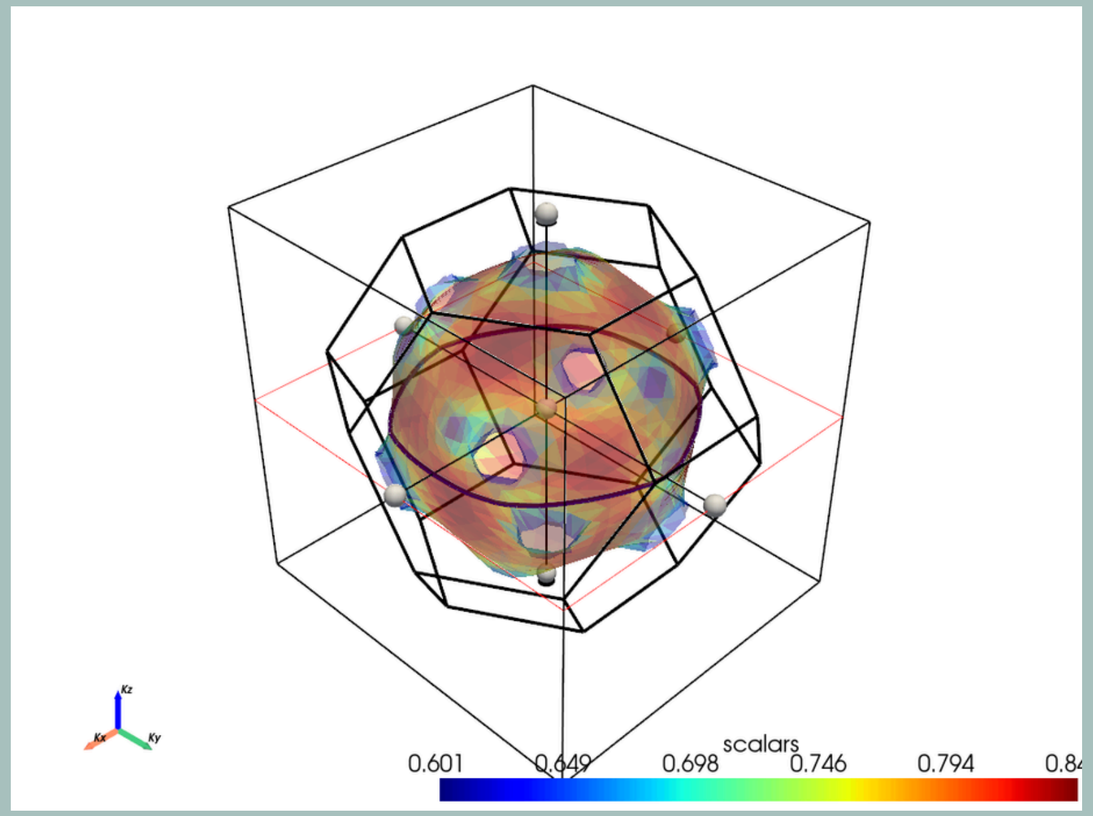

Projected band structures
Spin texture plots
Projected Fermi surfaces
Band structure comparison
Band unfolding


E - E_f (eV)
K vector
Atomic Orbital Projections

PyProcar



Superficie de Fermi en 2-D



Superficie de Fermi en 3-D

Conclusiones

En este proyecto, logramos modelar exitosamente el método de tight-binding en una cadena atómica utilizando Python.

Además, implementamos una versión paralelizada del código utilizando la biblioteca multiprocessing para aprovechar múltiples núcleos del procesador. Los resultados demostraron que la ejecución en paralelo redujo significativamente el tiempo de ejecución del programa.

¡Muchas gracias!