CMP project 2

Karmukilan Somasundaram

February 2026

1 Free Electron Band Structure Plot

Our goal is to plot the free electron Band Structure for an almost free electron by exploiting the Bloch Theorem in 1D. From the figure we can also correlate the band index.For a fixed k value

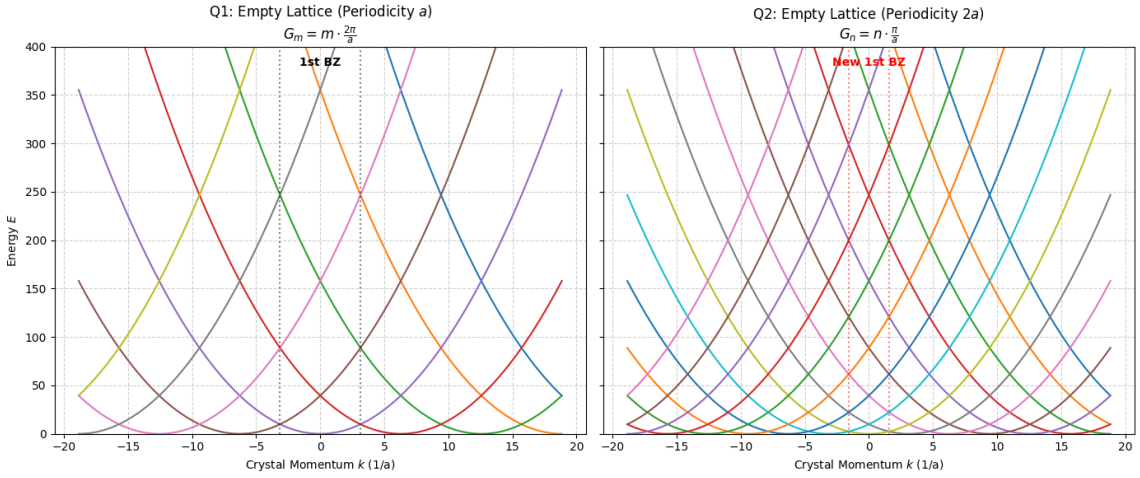

Figure 1: Free electron band structure for a and 2a

there are two cases

1. Case1:k is also a reciprocal lattice vector. In this case the ground state is clearly at 0 and the next nearest parabolas intersect the line x=k at the same point and the pattern follows for all subsequent parabolas
2. Case2:k is not a reciprocal lattice vector and lies in midpoint of two lattice vectors: Then the ground state is not zero clearly but the intersection pattern is similar to that of case1
3. Case3:k is not a reciprocal lattice vector and neither does it lie in the intersection of two lattice vectors: Then the nearest reciprocal lattice vector is unique which implies that will become the first ground state and the next parabola intersection should come from the opposite side and this pattern continues

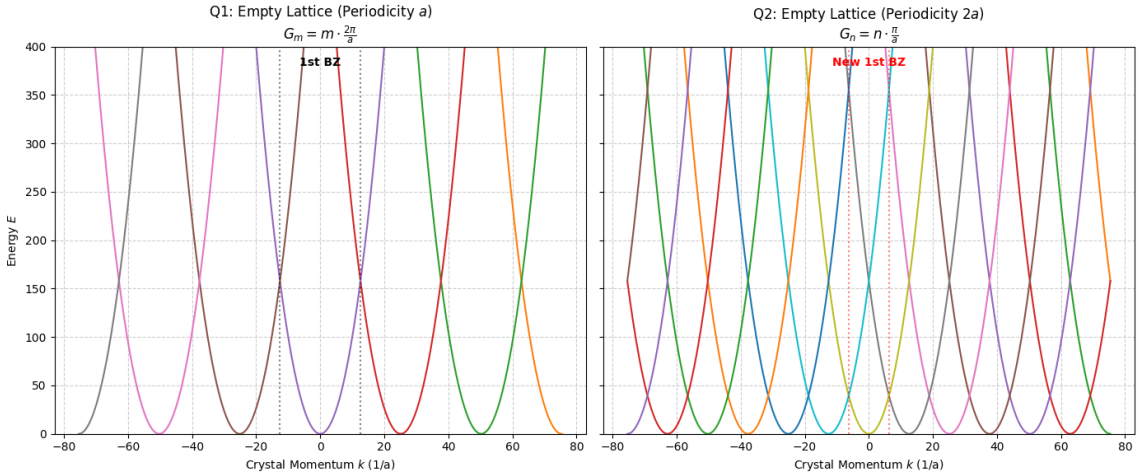

Figure 2: FEBS for smaller 'a' value

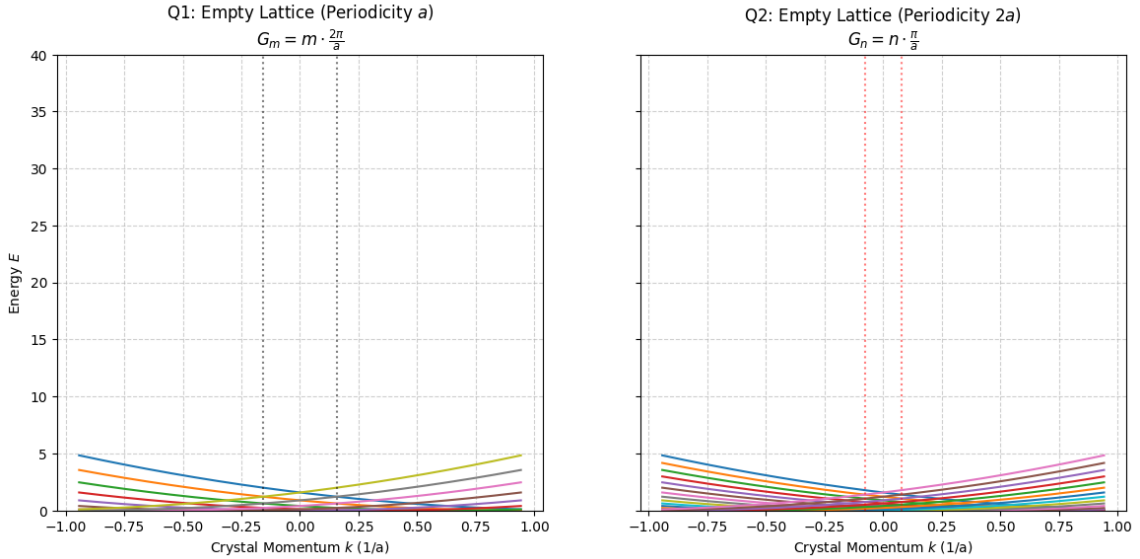

Figure 3: FEBS for larger 'a' value

Observe, as 'a' the primitive cell distance shrinks the energy bands get tightly packed(band-width decreases) and we get a metallic band structure which is in excellent agreement with our intuition.

Also note, we have kept the constants of the Energy eigenvalue equation to be 1 for convenience.

2 Potential ideas

1. Employing the Matrix technique we developed to construct the Electron Band Structure for some simple non-constant potentials and compare it with the free electron band structure

3 Python codes

Listing 1: Free-Electron-Band structure

```
import numpy as np
import matplotlib.pyplot as plt

def plot_empty_lattice():
    # --- Constants & Setup ---
    # We work in normalized units: h_bar^2 / 2m = 1, a = 1
    # This means k is in units of 1/a, and Energy is in arbitrary units.
    a = 200

    # Define the k-range from -6pi/a to +6pi/a as requested
    k_min = -6 * np.pi / a
    k_max =  6 * np.pi / a
    k_points = np.linspace(k_min, k_max, 500)

    # Create a figure with two subplots (Q1 and Q2)
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6), sharey=True)

    # --- Q1: Periodicity "a" ---
    # Reciprocal lattice vectors G = m * (2*pi/a)
    # We loop through several integers m to show multiple bands
    m_values = range(-4, 5)

    for m in m_values:
        G = m * (2 * np.pi / a)

        # Energy E = (hbar^2 / 2m) * (k + G)^2
        # In our units: E = (k + G)^2
        E_k = (k_points + G)**2

        # Plotting
        ax1.plot(k_points, E_k, label=f'm={m}', linewidth=1.5)

    # Formatting Q1 Plot
    ax1.set_title('Q1: Empty Lattice (Periodicity $a$)\n$G_m = m \\cdot \\
        frac{2\\pi}{a}$')
```

```
    ax1.set_xlabel('Crystal Momentum $k$ (1/a)')
    ax1.set_ylabel('Energy $E$')
    ax1.set_ylim(0, 1)  # Limit y-axis to keep plot readable
    ax1.grid(True, linestyle='--', alpha=0.6)

    # Draw vertical lines for the First Brillouin Zone boundaries at +/- pi/a
    ax1.axvline(x=np.pi/a, color='k', linestyle=':', alpha=0.5)
    ax1.axvline(x=-np.pi/a, color='k', linestyle=':', alpha=0.5)
    ax1.text(0, 380, '1st BZ', ha='center', fontsize=10, fontweight='bold')



    # --- Q2: Periodicity "2a" ---
    # Reciprocal lattice vectors G' = n * (2*pi / 2a) = n * (pi/a)
    # This effectively halves the spacing between parabolas
    n_values = range(-8, 9) # Need more indices to cover same energy range

    for n in n_values:
        G_prime = n * (np.pi / a)  # Note the pi/a instead of 2pi/a

        E_k_prime = (k_points + G_prime)**2

        ax2.plot(k_points, E_k_prime, label=f'n={n}', linewidth=1.5)

    # Formatting Q2 Plot
    ax2.set_title('Q2: Empty Lattice (Periodicity $2a$)\n$G_n = n \\cdot \\frac{\\pi}{a}$')
    ax2.set_xlabel('Crystal Momentum $k$ (1/a)')
    ax2.set_ylim(0, 1)
    ax2.grid(True, linestyle='--', alpha=0.6)

    # Draw vertical lines for the NEW First Brillouin Zone boundaries at +/- pi/2a
    ax2.axvline(x=np.pi/(2*a), color='r', linestyle=':', alpha=0.5)
    ax2.axvline(x=-np.pi/(2*a), color='r', linestyle=':', alpha=0.5)
    ax2.text(0, 380, 'New 1st BZ', ha='center', fontsize=10, fontweight='bold', color='r')

    plt.tight_layout()
    plt.show()

if __name__ == "__main__":
    plot_empty_lattice()
```