Experimental Physics (II)
Notebook

Experiment IV
Crystal And Reciprocal Space

Group 2

洪　瑜 B125090009
黄巧涵 B122030003
洪懌平 B102030019

2025/03/18

1 Preparatory Question

1.1 Define a crystal and provide its definition.

A crystal is a solid material whose atoms, molecules, or ions are arranged in a highly ordered repeating pattern extending in all three spatial dimensions. This regular arrangement forms a crystal lattice, which can be described mathematically by a basis (a set of atoms) and a Bravais lattice (a repeating grid of points in space).

- **basis:** The basis refers to the specific arrangement of atoms or atomic groups that are linked to each lattice point in a crystal structure.
- **Bravais lattice:** Every lattice point is identical, meaning that translating the lattice by any vector does not alter its properties. Moreover, an observer positioned at any given lattice point would experience the same surroundings as if they were at any other lattice point.
- **Lattice (point):** A lattice is a repetitive arrangement of points in a vector space.

1.2 Explain the Fourier transform using a square pulse as an example.

The Fourier Transform (FT) is a mathematical tool that converts a function or signal from the time domain to the frequency domain.

In the time domain, a signal varies with time. The Fourier Transform decomposes it into a combination of sinusoidal components at different frequencies, allowing us to analyze its spectral characteristics.

For a continuous function, the Fourier Transform is defined as:

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-i\omega t} dt \tag{1}$$

This expresses the transformation of its frequency representation.

Simply speaking, the Fourier Transform breaks down a function into its frequency components, helping us understand its spectral properties.

Considering a square pulse defined as:

$$f(t) = \begin{cases} A, & |t| \leq T/2 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Then its Fourier Transform is:

$$F(\omega) = T sinc\left(\frac{\omega T}{2}\right) \tag{3}$$

where the sinc function is defined as:

$$sinc(x) = \frac{sin(x)}{x} \tag{4}$$

This means that the frequency spectrum of a square pulse is a sinc function, which has a main lobe and multiple diminishing side lobes. This phenomenon is similar to the result of single-slit diffraction because the slit aperture can be considered a "square" function in space, and its diffraction pattern corresponds to its Fourier Transform.

Bonus: https://colab.research.google.com/drive/1tj_p0CpQVOKVBAET_oL-Sx_tk9qLaxlg?usp=sharing

Displays a square pulse, where the signal is 1 in the range -T/2 ≤ t ≤ T/2 and 0 elsewhere. The FFT result (blue curve) shows a sinc function, which matches the theoretical result (red dashed line). The width of the main lobe is inversely proportional to T (the pulse width), demonstrating the duality between time and frequency: the wider the time domain signal, the narrower the frequency domain response.

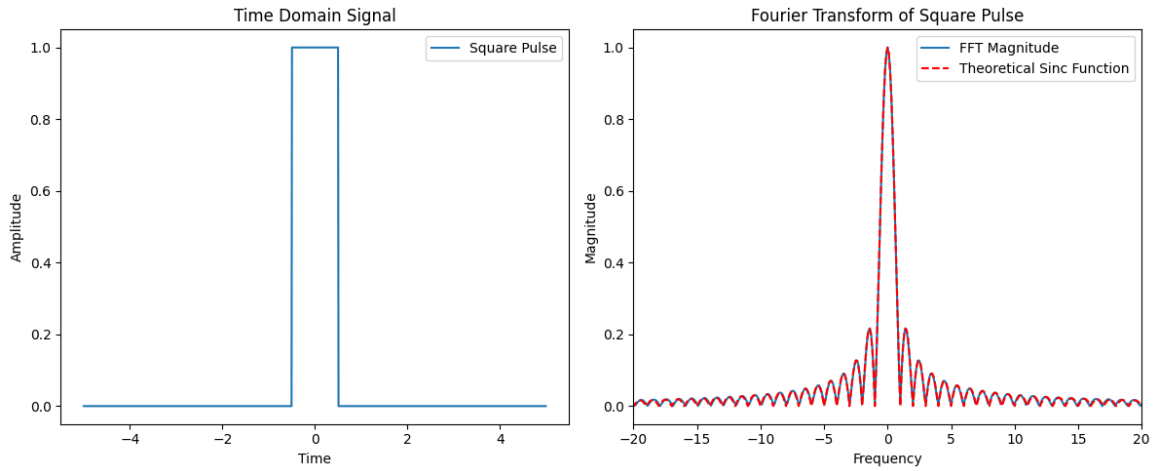

Figure 1: FFT of a square pulse using scipy

1.3 Describe translation symmetry in physics and provide an example.

- **Definition:** Translation symmetry means that a system remains unchanged when shifted in space. This implies the laws of physics are invariant under spatial displacement.
- **Example:** A crystal lattice exhibits translation symmetry, as shifting the entire lattice by one unit cell does not change its appearance.

1.4 Identify the shapes (triangle, square, rectangle, parallelogram, rhombus, pentagon, hexagon, and octagon) that can completely cover a space through translation operations without gaps or overlaps.

Shapes that can tessellate the plane through translation:

- **Triangle**
- **Square**
- **Rectangle**
- **Parallelogram**
- **Hexagon**

The pentagon and octagon generally cannot tile the plane alone. However, some special types of pentagons can tessellate. A rhombus is a special case of a parallelogram, so it also tessellates.

1.5 What are diffraction and interference? Please look up the diffraction pattern of a single-slit experiment. Does this diffraction pattern appear similar to the result you obtained from question 2?

- **Diffraction:** The bending and spreading of waves when they encounter an obstacle or pass through a narrow opening.
- **Interference:** The overlapping of two or more waves, leads to constructive or destructive patterns.
- **Single-Slit Diffraction Pattern:** The intensity distribution follows a $sinc^2$ function:

$$I(\theta) = I_0 \left(\frac{\sin(\pi a \sin\theta/\lambda)}{\pi a \sin\theta/\lambda} \right)^2 = I_0 sinc^2(\pi a \sin\theta/\lambda) \tag{5}$$

This pattern closely resembles the Fourier transform of a square pulse, which is also a sinc function. This is because the single-slit aperture acts as a spatial filter, and its diffraction pattern is its Fourier transform.

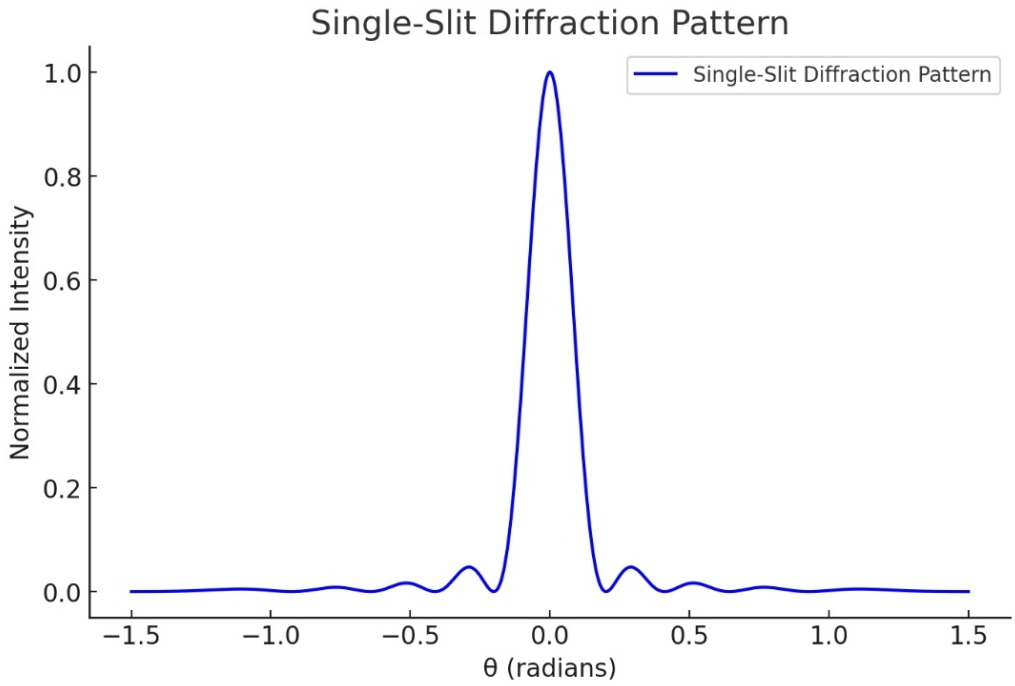

Figure 2: Single-slit diffraction pattern

2 Experimental steps, preliminary results, and analysis

2.1 Graphene Crystal Structure

1. **Describe the type of lattice present in graphene.**
   Graphene has a honeycomb lattice structure, which can be described as a hexagonal (2D) Bravais lattice with a two-atom basis. The underlying lattice is a triangular Bravais lattice, but the honeycomb structure emerges due to the presence of two inequivalent carbon atoms (A and B) in each unit cell.

2. **Identify the basis atoms in the unit cell of graphene.**
   Graphene's unit cell consists of two carbon atoms, labeled as A and B.

   - **Atom A:** Located at $(0, 0)$
   - **Atom B:** Located at $\left(\frac{a}{2}, \frac{\sqrt{3}a}{6}\right)$, where a is the nearest-neighbor bond length

3. **How many carbon atoms constitute the basis?**
   The basis of graphene consists of two carbon atoms per unit cell.

4. **Express the lattice vectors.**
   Graphene's **primitive lattice vectors** (a_1 and a_2) in Cartesian coordinates are:

$$\mathbf{a_1} = \left(\frac{3a}{2}, \frac{\sqrt{3}a}{2}\right) \tag{6}$$

$$\mathbf{a_2} = \left(\frac{3a}{2}, -\frac{\sqrt{3}a}{2}\right) \tag{7}$$

   where a is the **nearest-neighbor bond length** (approximately 1.42Å), and the **lattice constant** a_0 (distance between two adjacent unit cells) is given by:

$$a_0 = \sqrt{3}a \approx 2.46\mathring{A} \tag{8}$$

5. **Use these vectors to generate graphene lattice via Python.**

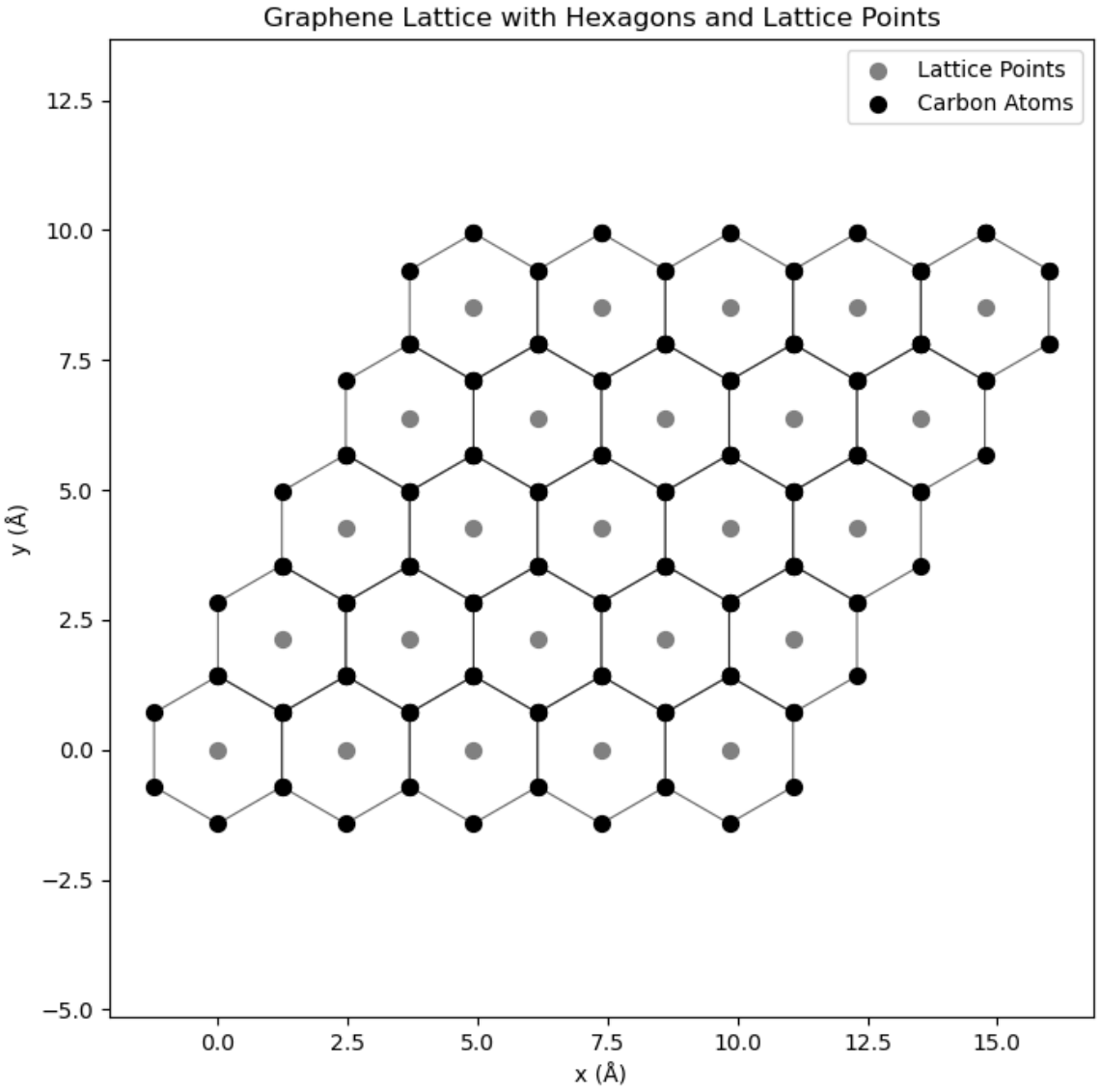

Figure 3: Lattice points of graphene

Preliminary analysis:
Theoretically, the lattice shape of graphene is **hexagon**, and with this exercise of the Python simulation, we, fortunately, reproduce the hexagon shapes with the basis atoms and the lattice vectors mentioned above.

Bonus:

1. **Use MATLAB/Python to create a twisted bilayer graphene Moiré pattern (in real space).**

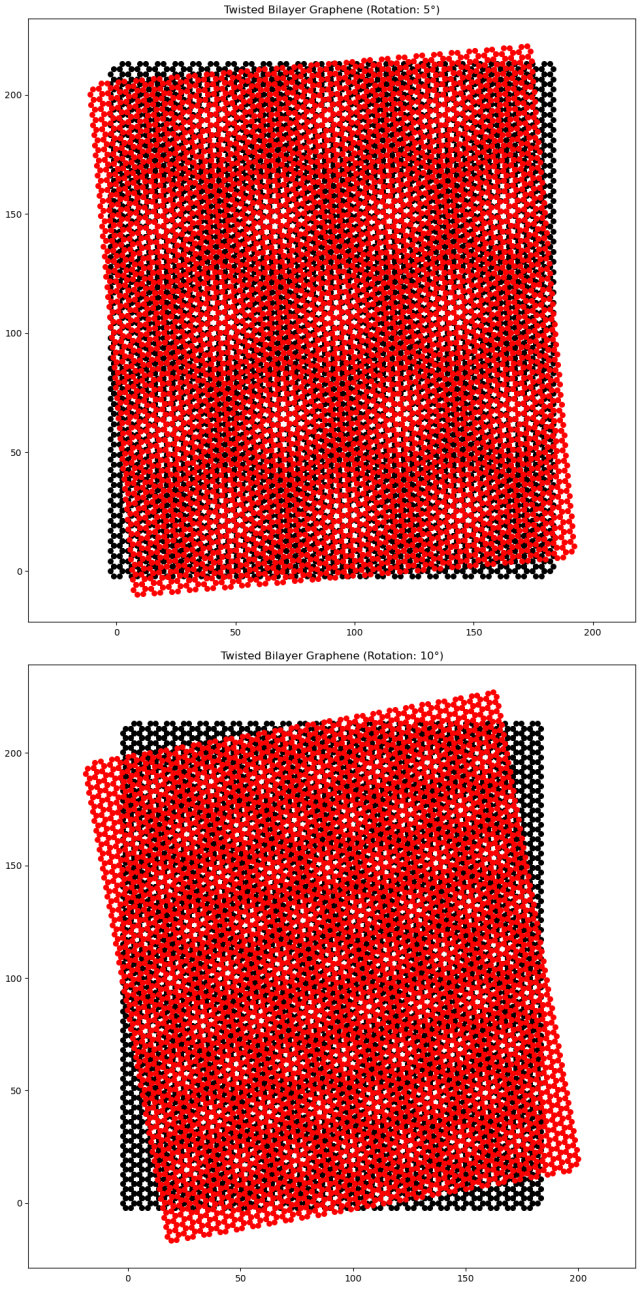

Figure 4: A twisted bilayer graphene Moiré pattern (top:5°; bottom:10°)

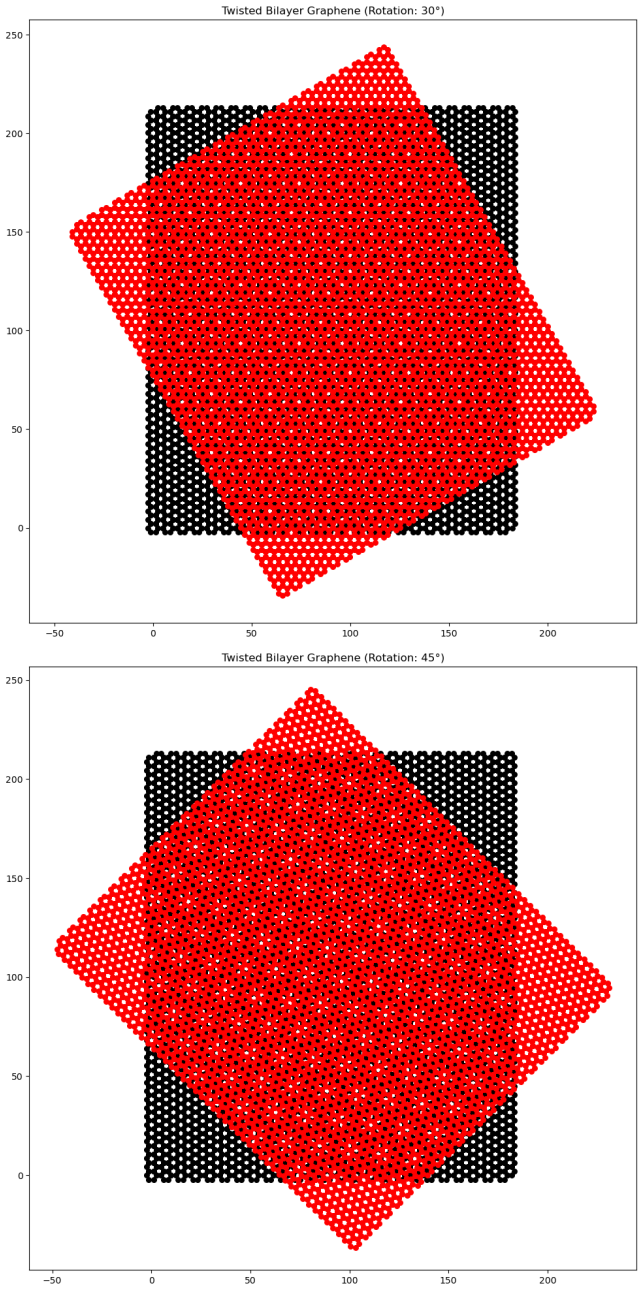

Figure 5: A twisted bilayer graphene Moiré pattern (top:30°; bottom:45°)

Preliminary analysis:
We modified the codes, which produce the lattice shape of graphene, from step 5, making them able to rotate with a given angle, the Moiré patterns are shown eventually.

The first layer of graphene is arranged in a standard hexagonal honeycomb lattice, forming a regular crystal structure. The second layer of graphene is very close to the first layer but is slightly rotated by a small angle, creating a visual "interference pattern" between the two lattices. Due to the subtle misalignment of the two layers, certain regions have a higher point density, while others are sparser, forming a large-scale periodic pattern known as the Moiré pattern. The period of this pattern depends on the rotation angle. When the angle is sufficiently small, the Moiré pattern's period becomes large, resulting in a clear interference pattern.

Further analysis of the relationship between the periodicity of the Moiré pattern and its

twist angles will be added to the report.

Possible problems and error handling:
During the process which producing these images, we encountered some error messages or something unexpected. We then modified the source code each line at a time to debug. Eventually, we got the expected images.

2. **Find the relationship between the periodicity of the Moiré pattern and its twist angles.**
When the lattice constant of the two layers of graphene (a_0, approximately 2.46 Å) and their twist angle (θ , assuming θ is very small) are given, the Moiré pattern periodicity L can be approximated as:

$$L \approx \frac{a_0}{2sin(\theta/2)} \tag{9}$$

where L = Moiré pattern periodicity, a_0 is graphene lattice constant (approximately 2.46 Å), and θ is twist angle (in radians)

As the twist angle θ decreases, the Moiré pattern periodicity L increases.

2.2 Reciprocal Space

1. **Please draw the reciprocal lattice vector G in Fig.6(b) and define the length of G.**

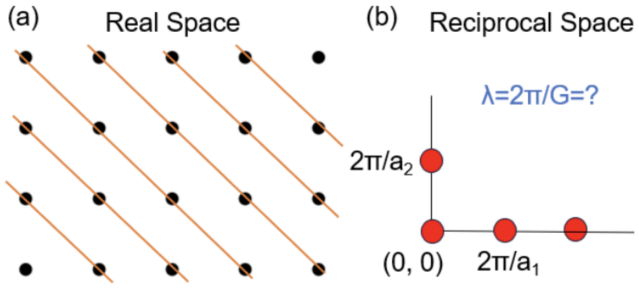

Figure 6: Demonstration of reciprocal space (credit: Experiment Guideline)

2. **Use the relationship ($\mathbf{b_i} \cdot \mathbf{a_i} = 2\pi\delta_{ij}$ where $\delta_{ij} = 1$ if $i = j$ and $\delta_{ij} = 0$ if $i \neq j$) to obtain the reciprocal lattice vectors of graphene and again use python to generate the reciprocal lattice point of graphene. Find the diffraction patterns of graphene from a website and define the reciprocal lattice vectors. Compare these obtained vectors with your results.**
The reciprocal lattice vectors ($\mathbf{b_1}$ and $\mathbf{b_2}$) for graphene can be derived from the real-space primitive lattice vectors using the formula:

$$\mathbf{b_i} \cdot \mathbf{a_j} = 2\pi\delta_{ij} \tag{10}$$

Given that graphene has a triangular Bravais lattice with a honeycomb basis, the real-space primitive lattice vectors are:

$$\mathbf{a_1} = \left(\frac{3a}{2}, \frac{\sqrt{3}a}{2}\right) \tag{11}$$

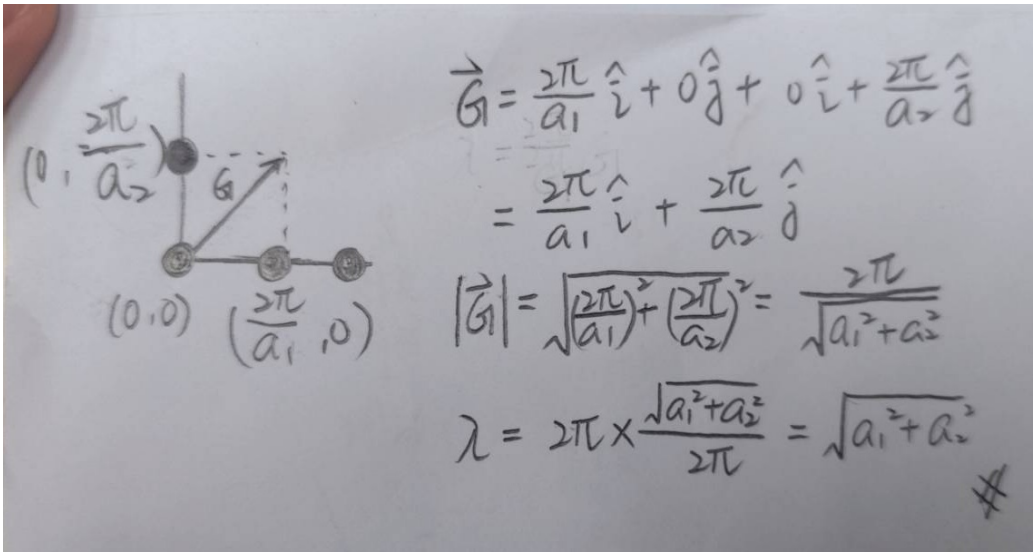

Figure 7: Tle length of G

$$\mathbf{a_2} = \left(\frac{3a}{2}, -\frac{\sqrt{3}a}{2}\right) \tag{12}$$

The corresponding reciprocal lattice vectors are:

$$\mathbf{b_1} = \frac{2\pi}{3a}\left(1, \sqrt{3}\right) \tag{13}$$

$$\mathbf{b_2} = \frac{2\pi}{3a}\left(1, -\sqrt{3}\right) \tag{14}$$

where $a_0 = \sqrt{3}a$ is the graphene lattice constant.

The length of the reciprocal lattice vector G is:

$$G = |\mathbf{b_1}| = |\mathbf{b_2}| = \frac{4\pi}{3a} \tag{15}$$

which is derived from:

$$|\mathbf{b_1}| = \sqrt{\left(\frac{2\pi}{3a}\right)^2 + \left(\frac{2\pi}{3a} \times \sqrt{3}\right)} = \frac{4\pi}{3a} \tag{16}$$

3. **Use the lattice points and the reciprocal lattice vectors you obtained from Python to check if reciprocal lattice points can be mapped out by the set of reciprocal lattice vectors that yield plane waves with the periodicity of a given Bravais lattice.**
   The reciprocal lattice vectors define the wave vectors of allowed plane waves in the crystal. The periodicity of the reciprocal lattice is linked to the Brillouin zone and diffraction patterns.

   - In a Bravais lattice, the reciprocal lattice vectors should yield plane waves that have the same periodicity as the real-space lattice.
   - The plane wave is given by:
     $$\psi(\mathbf{r}) = e^{i\mathbf{G}\cdot\mathbf{r}} \tag{17}$$
     where the length of $\mathbf{G}$ is $\frac{4\pi}{3a}$
   - The reciprocal lattice vectors confirm that plane waves generated by them satisfy the periodicity of the real-space Bravais lattice.

3 Appendix

3.1 Source codes of these exercises

- FFT of a square pulse: https://github.com/hyp0515/exp_phy_ii/blob/main/mar18/squre_pulse.ipynb
- Generating graphene's lattice structure: https://github.com/hyp0515/exp_phy_ii/blob/main/mar18/graphene.ipynb
- Generating a twisted bilayer graphene Moiré pattern: https://github.com/hyp0515/exp_phy_ii/blob/main/mar18/moire_pattern.ipynb

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftfreq, fftshift
```

```python
# 定義參數
T = 1  # 脈衝寬度
N = 1024  # 取樣點數
t_max = 5  # 時間範圍
dt = 2 * t_max / N  # 取樣間隔
t = np.linspace(-t_max, t_max, N, endpoint=False)  # 時間軸

# 定義方形脈衝
square_pulse = np.where(np.abs(t) <= T/2, 1, 0)

# 計算傅立葉變換
freq = fftfreq(N, dt)  # 頻率軸
fft_result = fft(square_pulse)  # FFT
fft_result_shifted = fftshift(fft_result)  # 使頻譜對稱
freq_shifted = fftshift(freq)  # 使頻率對稱

# 理論 sinc 函數
sinc_theory = T * np.sinc(freq_shifted * T)
```

```python
# 繪製時域訊號
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.plot(t, square_pulse, label="Square Pulse")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.title("Time Domain Signal")
#plt.grid()
plt.legend()

# 繪製 FFT 結果
plt.subplot(1, 2, 2)
plt.plot(freq_shifted, np.abs(fft_result_shifted) / np.max(np.abs(fft_result_shifted)), label="FFT Magnitude")
plt.plot(freq_shifted, np.abs(sinc_theory) / np.max(np.abs(sinc_theory)), 'r--', label="Theoretical Sinc Function
plt.xlabel("Frequency")
plt.ylabel("Magnitude")
plt.title("Fourier Transform of Square Pulse")
#plt.grid()
plt.legend()
plt.xlim((-20, 20))

plt.tight_layout()
plt.show()
```

Figure 8: Source code of "FFT of a square pulse"

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

```python
def generate_graphene_lattice(rows, cols, a=2.46):
    """
    生成石墨烯的蜂巢晶格，並標示布拉菲格子的晶格點。
    :param rows: 晶格行數
    :param cols: 晶格列數
    :param a: 晶格常數 (默認 2.46 Å)
    """
    # 碳-碳鍵長 (石墨烯的最近鄰原子間距)
    bond_length = a / np.sqrt(3)  # 約 1.42 Å

    # 定義基底原子的相對位置 (A 原子在晶格點上，B 原子偏移)
    basis = np.array([[0, 0], [bond_length, 0]])

    # 定義布拉菲格子 (簡單六角晶格)
    lattice_vectors = np.array([
        [a, 0],
        [a / 2, a * np.sqrt(3) / 2]
    ])

    # 存儲原子位置和晶格點
    atoms = []
    lattice_points = []
    hexagons = []

    # 生成晶格
    for i in range(rows):
        for j in range(cols):
            # 計算晶格點位置
            lattice_point = i * lattice_vectors[0] + j * lattice_vectors[1]
            lattice_points.append(lattice_point)

            # 添加基底原子
            for b in basis:
                atoms.append(lattice_point + b)

            # 定義六邊形的六個頂點
            hexagon = [
                lattice_point + np.array([0, bond_length]),
                lattice_point + np.array([a / 2, bond_length / 2]),
                lattice_point + np.array([a / 2, -bond_length / 2]),
                lattice_point + np.array([0, -bond_length]),
                lattice_point + np.array([-a / 2, -bond_length / 2]),
                lattice_point + np.array([-a / 2, bond_length / 2]),
                lattice_point + np.array([0, bond_length])  # Close the hexagon
            ]
            hexagons.append(hexagon)
```

```python
    return np.array(atoms), np.array(lattice_points), hexagons


def plot_graphene(rows=5, cols=5):
    """
    繪製石墨烯結構，標示晶格點，並畫出六邊形。
    """
    atoms, lattice_points, hexagons = generate_graphene_lattice(rows, cols)

    plt.figure(figsize=(8, 8))

    # 繪製碳原子
    #plt.scatter(atoms[:, 0], atoms[:, 1], c='black', label='Carbon Atoms', s=50, zorder=2)

    # 繪製晶格點
    plt.scatter(lattice_points[:, 0], lattice_points[:, 1], c='gray', marker='o', label='Lattice Points', s=50, z

    # 繪製六邊形
    for hexagon in hexagons:
        hexagon = np.array(hexagon)
        plt.plot(hexagon[:, 0], hexagon[:, 1], 'k', lw=0.8, alpha=0.6, zorder=1)
        plt.scatter(hexagon[:, 0], hexagon[:, 1], c='k', marker='o', s=50, zorder=3)
    plt.scatter(hexagon[:, 0], hexagon[:, 1], c='k', marker='o', s=50, zorder=3, label='Carbon Atoms')
    plt.axis('equal')
    plt.legend()
    plt.xlabel('x (Å)')
    plt.ylabel('y (Å)')
    plt.title('Graphene Lattice with Hexagons and Lattice Points')
    plt.show()
```

```python
# 執行繪製
plot_graphene(5, 5)
```

Figure 9: Source code of generating graphene's lattice structure

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

```python
def generate_hexagon(center, a):
    """
    生成以 center 為中心的六邊形頂點座標。
    :param center: 六邊形中心 (x, y)
    :param a: 晶格常數
    :return: 六邊形頂點座標 (6 個點)
    """
    angles = np.linspace(0, 2*np.pi, 7)[:-1]  # 只取 6 點 (不重複)
    hexagon = np.column_stack([center[0] + a * np.cos(angles),
                               center[1] + a * np.sin(angles)])
    return hexagon

def rotate_points(points, angle, center=(0, 0)):
    """
    對一組 2D 點進行旋轉。
    :param points: 點的座標陣列
    :param angle: 旋轉角度 (弧度)
    :param center: 旋轉中心
    :return: 旋轉後的點座標
    """
    theta = np.radians(angle)
    rotation_matrix = np.array([
        [np.cos(theta), -np.sin(theta)],
        [np.sin(theta), np.cos(theta)]
    ])

    # 以 center 為中心旋轉
    return np.dot(points - center, rotation_matrix.T) + center

def generate_graphene_layers(rows, cols, a=2.46, angle=5):
    """
    生成兩層六邊形石墨烯，第二層相對第一層旋轉一定角度。
    :param rows: 六邊形的行數
    :param cols: 六邊形的列數
    :param a: 晶格常數
    :param angle: 第二層旋轉角度 (度)
    :return: 第一層與旋轉後的第二層六邊形頂點座標
    """
    hexagons_layer1 = []
    hexagons_layer2 = []

    dx = 1.5 * a  # x 方向間距
    dy = np.sqrt(3) * a  # y 方向間距
    center_of_rotation = np.array([dx * cols / 2, dy * rows / 2])  # 旋轉中心設在格子的中心
```

```python
    for i in range(rows):
        for j in range(cols):
            x = j * dx
            y = i * dy
            if j % 2 == 1:
                y += dy / 2  # 交錯排列

            # 第一層六邊形
            hexagon = generate_hexagon((x, y), a)
            hexagons_layer1.append(hexagon)

            # 第二層旋轉後的六邊形
            rotated_hexagon = rotate_points(hexagon, angle, center_of_rotation)
            hexagons_layer2.append(rotated_hexagon)

    return hexagons_layer1, hexagons_layer2

def plot_graphene_layers(rows=10, cols=10, angle=5):
    """
    繪製旋轉後的雙層石墨烯六邊形結構 (擴展範圍)。
    """
    hexagons_layer1, hexagons_layer2 = generate_graphene_layers(rows, cols, angle=angle)

    plt.figure(figsize=(12, 12))  # 增加圖的大小

    # 第一層六邊形 (黑色)
    for hexagon in hexagons_layer1:
        plt.plot(hexagon[:, 0], hexagon[:, 1], 'k', lw=0.8)
        plt.scatter(hexagon[:, 0], hexagon[:, 1], c='k', marker='o', s=30, zorder=3)

    # 第二層旋轉後的六邊形 (紅色)
    for hexagon in hexagons_layer2:
        plt.plot(hexagon[:, 0], hexagon[:, 1], 'r', lw=0.8, linestyle='dashed')
        plt.scatter(hexagon[:, 0], hexagon[:, 1], c='r', marker='o', s=30, zorder=3)
    # plt.ylim((0, 175))
    # plt.xlim((0, 175))
    plt.axis('equal')
    plt.title(f'Twisted Bilayer Graphene (Rotation: {angle}°)')
    plt.show()
```

```python
# 繪製更大的雙層石墨烯 (10x10 格，旋轉 5°)
plot_graphene_layers(50, 50, angle=5)
```

Figure 10: Source code of generating a twisted bilayer graphene Moiré pattern