Exercise 1. ***Chirality of an isotropic Weyl node***

An isotropic Weyl node is described locally by the Hamiltonian

$$\mathcal{H}(\mathbf{k}) = \sum_{i\in\{x,y,z\}} k_i \sigma_i, \tag{1}$$

where σ_i are the Pauli matrices. Depending on its chirality, the Weyl node is either a source or sink of Berry curvature. Consequently, the chirality can be calculated by evaluating the Chern number on a sphere surrounding the Weyl node.

The following skeleton code (`ex1/weyl_skeleton.py`) contains a function which represents the Hamiltonian from eq. (1).

(a) Create a `z2pack.hm.System` instance which represents this system.

(b) Run a calculation for a surface enclosing the Weyl point - which is located at $\mathbf{k} = (0, 0, 0)$ - by a sphere of radius 0.1.

Hint: Try finding the documentation for the `z2pack.shape` submodule in the Z2Pack reference.

(c) What is the Chern number associated with this surface?

(d) Create a plot which demonstrates the result of your calculation.

Listing 1: Skeleton code containing the function `Hamilton(k)`

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np

# defining pauli matrices
identity = np.identity(2, dtype=complex)
pauli_x = np.array([[0, 1], [1, 0]], dtype=complex)
pauli_y = np.array([[0, -1j], [1j, 0]], dtype=complex)
pauli_z = np.array([[1, 0], [0, -1]], dtype=complex)

# defining the Weyl Hamiltonian
def Hamilton(k):
    kx, ky, kz = k
    return kx * pauli_x + ky * pauli_y + kz * pauli_z
```

Exercise 2. $\mathbb{Z}_2$ ***invariant in a tight-binding model***

We consider a two-dimensional system which consists of two inter-penetrating square lattices (see fig. 1). Each lattice point hosts two orbitals, which have on-site energy $+1$ for one sub-lattice, and -1 for the other. We consider two kinds of hopping:

- Nearest-neighbour hopping with strength t_1, between the first / second orbital in one sub-lattice and the same orbital in the other sub-lattice. Depending on the direction, the hoppings acquire a phase factor.
- Next-nearest neighbour hopping with strength $\pm t_2$. The hopping is positive for the sub-lattice with positive on-site energy, and negative for the other one.

In this exercise, the code for creating such a tight-binding model with `tbmodels` is given. We study the $\mathbb{Z}_2$ invariant for the system (in the $k_z = 0$ plane).

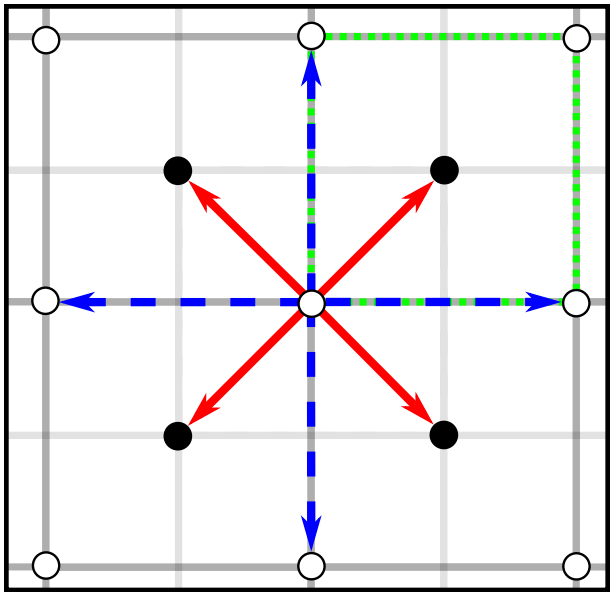

Figure 1: Sketch of a model with two square sublattices. The arrows indicate nearest-neighbour (red) and next-nearest-neighbour (blue) interaction.

(a) Implement the `run` function such that it prints the $\mathbb{Z}_2$ invariant of the $k_z = 0$ plane for given parameters t_1, t_2. Evaluate this function for $(t_1, t_2) = (0.1, 0.2)$ and $(0.2, 0.3)$.

Hint: Remember that Wannier charge centers should be calculated only on half the Brillouin zone to calculate the $\mathbb{Z}_2$ invariant.

(b) Expand the function such that it also creates a plot showing the Wannier charge centers and their largest gaps.

Listing 2: Skeleton code, containing a function which returns the tight-binding model for given hopping strengths t_1, t_2.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```
import itertools

import tbmodels

def get_model(t1, t2):
    model = tbmodels.Model(
        on_site=(1, 1, -1, -1),
        pos=[
            [0., 0., 0.],
            [0., 0., 0.],
            [0.5, 0.5, 0.],
            [0.5, 0.5, 0.]
        ],
        occ=2
    )

    for p, R in zip([1, 1j, -1j, -1], itertools.product(
            [0, -1], [0, -1], [0]
        )):
        model.add_hop(
            overlap=p * t1,
            orbital_1=0,
            orbital_2=2,
            R=R
        )
        model.add_hop(
            overlap=p.conjugate() * t1,
            orbital_1=1,
            orbital_2=3,
            R=R
        )

    for R in ((r[0], r[1], 0) for r in itertools.permutations(
        [0, 1]
    )):
        model.add_hop(t2, 0, 0, R)
        model.add_hop(t2, 1, 1, R)
        model.add_hop(-t2, 2, 2, R)
        model.add_hop(-t2, 3, 3, R)
    return model

def run(t1, t2):
    # Task a: print the Z2 invariant for the kz=0 plane

    # Task b: create a plot showing WCC and the largest gap

if __name__ == '__main__':
    # Call the run function here
```

Exercise 3. ***Haldane model***

The Haldane model has a honeycomb lattice with two sublattices (see fig. 2) which have on-site energy $+M$ and $-M$, respectively. The model contains nearest-neighbour (inter-sublattice) hoppings with a hopping strength t_1 and next-nearest-neighbour hoppings with strength t_2. The hopping terms acquire a phase due to a microscopic magnetic field, whose strength is parametrized by a variable ϕ. In this exercise, the Hamiltonian for the Haldane model is given, and we investigate the Chern number for different values of the parameters M, t_1, t_2, ϕ.

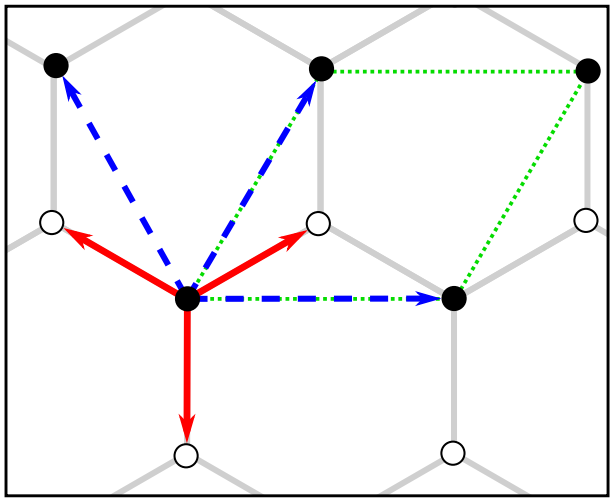

Figure 2: A honeycomb lattice with two sublattices. The arrows indicate nearest-neighbour (red) and next-nearest-neighbour (blue) hoppings.

(a) Implement the `get_chern` function, which should return the Chern number of the Haldane model for a given set of parameters M, t_1, t_2, ϕ. Using this function, evaluate the Chern number for $M = 0.5, t_1 = 1, t_2 = 1/3$, and $\phi = \pm\pi/2$.

Note: The purpose of `**settings` in the signature of `get_chern` is to allow passing keyword arguments to the `z2pack.surface.run` function. To do this, add `**settings` as the last argument: `z2pack.surface.run(..., **settings)`. Keyword arguments passed to `get_chern` will then be forwarded to `z2pack.surface.run`.

(b) Change M to 1.7 and evaluate the two Chern numbers again. Change `min_neighbour_dist` such that all convergence tests pass.

(c) *(optional - for advanced Python users)*
Create a plot which shows the phase of the Haldane model as a function of M and ϕ for $M \in [-2, 2], \phi \in [-\pi, \pi]$, and $t_1 = 1, t_2 = 1/3$.
For certain values of the parameters it is much harder (or even impossible) to reach convergence. Can you find an explanation for this behaviour?

Hint: Create a 2D array containing the values for the Chern number for different values of M and ϕ, and use `matplotlib`'s `imshow` function to plot it.

Listing 3: Skeleton code containing the function for the Hamiltonian in the Haldane model.

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
```

```python
import numpy as np

# defining pauli matrices
identity = np.identity(2, dtype=complex)
pauli_x = np.array([[0, 1], [1, 0]], dtype=complex)
pauli_y = np.array([[0, -1j], [1j, 0]], dtype=complex)
pauli_z = np.array([[1, 0], [0, -1]], dtype=complex)


def Hamilton(k, m, t1, t2, phi):
    kx, ky, _ = k
    k_a = 2 * np.pi / 3. * np.array([
        kx + ky,
        -2. * kx + ky,
        kx - 2. * ky
    ])
    k_b = 2 * np.pi * np.array([-kx + ky, -ky, kx])
    H = 2 * t2 * np.cos(phi) * sum(np.cos(k_b)) * identity
    H += t1 * sum(np.cos(k_a)) * pauli_x
    H += t1 * sum(np.sin(k_a)) * pauli_y
    H += m * pauli_z
    H -= 2 * t2 * np.sin(phi) * sum(np.sin(k_b)) * pauli_z
    return H

def get_chern(m, t1, t2, phi, **settings):
    # This function should return the Chern number for the given
    # parameters m, t1, t2, phi. The **settings should be passed
    # on to the z2pack.surface.run method.

if __name__ == "__main__":
    # Task a)

    # Task b)

    # Task c) (optional - for advanced Python users)
```

Exercise 4. ***Tight-binding model for $MoTe_2$***

The file `ex4/data/wannier90_hr.dat` contains a tight-binding model for $MoTe_2$ that has 56 occupied bands. In this exercise, we study the topological properties of this model.

Note: The result files from this exercise are quite large ($\sim 500\,\mathrm{MB}$ in total). Make sure to delete them later if you have limited disk space.

(a) Calculate the $\mathbb{Z}_2$ invariant for the $k_y = 0$ plane and plot your result. Make sure that the result is saved to a file while the calculation is running.

(b) $MoTe_2$ has a very small band gap in the $k_y = 0$ plane. Do you still trust your result? Try using more than 200 lines in the initial calculation. Adjust the `min_neighbour_dist` and `iterator` keyword arguments as needed.

Note: To speed up the calculation, you can change the code such that the previous result is loaded manually and passed to `z2pack.surface.run` as `init_result`, and disable saving during the run. Make sure to save the result after the calculation.

(c) Calculate the $\mathbb{Z}_2$ invariant for the $k_z = 0$ plane. Try adjusting the `min_neighbour_dist` and `iterator` keyword arguments. What could be the reason that you cannot reach convergence?

(d) *(optional - for advanced Python users)*
Find a point where the direct band gap vanishes, and show that it is a Weyl point.

Hint: You can use the `tbmodels.Model.eigenval` method to calculate the band gap, and `scipy.optimize.minimize` to find the minimum of the gap. Use $\mathbf{k} = (0.9, 0.05, 0)$ as a starting point for the minimization. You might have to try different radii of the sphere to get a non-zero Chern number because the minimum might not be the exact location of the Weyl node.