Berry 相和 Chern 数

2024 年 4 月 12 日

1 Berry 相与 Chern 数

我想把繁琐的涉及到自己进行计算的步骤直接省略，哈密顿量直接用，以及他的特征值和特征向量。但此文完整的展示了 Berry 相和 Chern 数底层的数学逻辑。

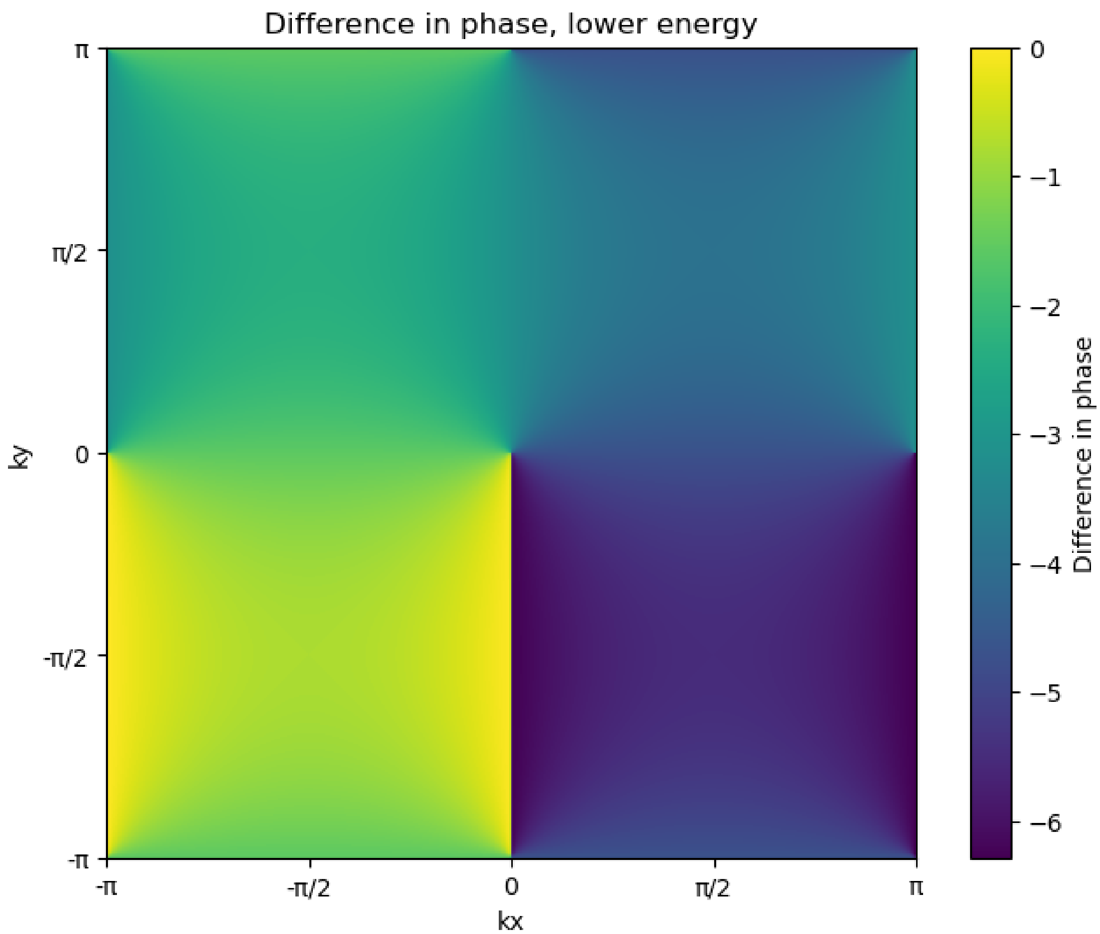
[1]:
```
import plotly.graph_objects as go
import numpy as np

 = 1

labels = ["- ", "- /2", "0", " /2", " "]
ticks = [-np.pi, -np.pi/2, 0, np.pi/2, np.pi]

ks = np.linspace(-np.pi, np.pi, 314)
l = len(ks)

ka = np.zeros((l, l, 2))
for ii in range(l):
    x = ks[ii]
    for jj in range(l):
        ka[ii, jj] = [x, ks[jj]]

def R0(k):
    return 0

def R1(k):
    return -2 * np.sin(k[0])
```

```
def R2(k):
    return -2 * np.sin(k[1])

def R3(k):
      = 1
    return   + 2 * np.cos(k[0]) + 2 * np.cos(k[1])

def R(k):
    return np.sqrt(R1(k)**2 + R2(k)**2 + R3(k)**2)

def  p(k):
    return R0(k) + R(k)

def  m(k):
    return R0(k) - R(k)

 pa = np.zeros((l, l))
 ma = np.zeros((l, l))

for ii in range(l):
    for jj in range(l):
         pa[ii, jj] = p(ka[ii, jj])
         ma[ii, jj] = m(ka[ii, jj])

fig = go.Figure(data=[
    go.Surface(z= pa, x=ks, y=ks),
    go.Surface(z= ma, x=ks, y=ks)
])

fig.update_layout(
    scene=dict(
        xaxis=dict(tickvals=ticks, ticktext=labels),
        yaxis=dict(tickvals=ticks, ticktext=labels),
        zaxis=dict(title="Energy"),
    ),
    title="Band Diagram"
```

```
)

fig.show()
```

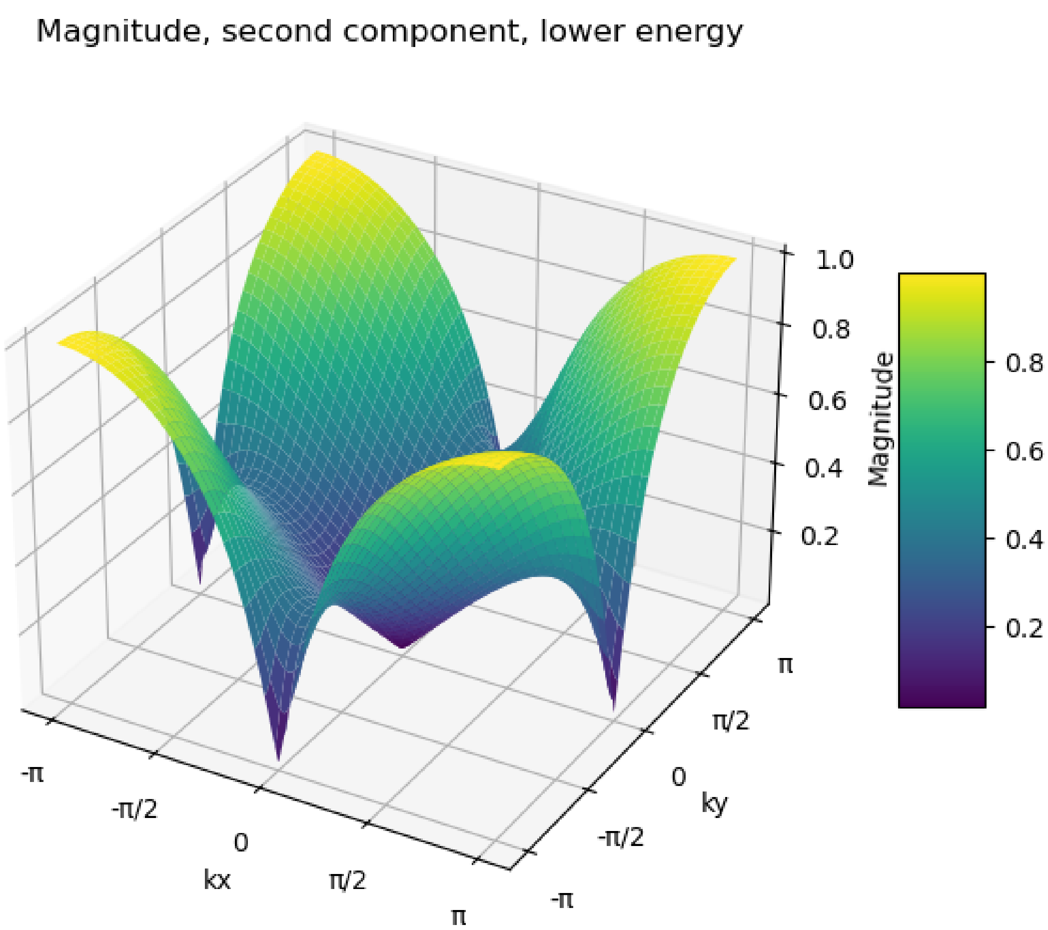
[2]:
```
import matplotlib.pyplot as plt
def up1(k):
    denom = 2 * R(k) * (R(k) + R3(k))
    front = 1 / np.sqrt(denom) if denom != 0 else 0  # Add condition to avoid␣
 ↪division by zero
    return front * (R(k) + R3(k))

def up2(k):
    denom = 2 * R(k) * (R(k) + R3(k))
    front = 1 / np.sqrt(denom) if denom != 0 else 0  # Add condition to avoid␣
 ↪division by zero
    return front * (R1(k) + 1j * R2(k))

up = [up1, up2]

# Define um1 and um2 functions
def um1(k):
    denom = 2 * R(k) * (R(k) - R3(k))
    front = 1 / np.sqrt(denom) if denom != 0 else 0  # Add condition to avoid␣
 ↪division by zero
    return front * (-R(k) + R3(k))

def um2(k):
    denom = 2 * R(k) * (R(k) - R3(k))
    front = 1 / np.sqrt(denom) if denom != 0 else 0  # Add condition to avoid␣
 ↪division by zero
    return front * (R1(k) + 1j * R2(k))

um = [um1, um2]

# Define ka, l, and other necessary variables here
```

```
upa = np.zeros((2, l, l), dtype=np.complex128)
uma = np.zeros((2, l, l), dtype=np.complex128)

for ii in range(l):
    for jj in range(l):
        upa[0, ii, jj] = up[0](ka[ii, jj])
        upa[1, ii, jj] = up[1](ka[ii, jj])
        uma[0, ii, jj] = um[0](ka[ii, jj])
        uma[1, ii, jj] = um[1](ka[ii, jj])



plt.figure(figsize=(8, 6))
plt.imshow(np.angle(uma[1,:,:]) - np.angle(uma[0,:,:]), extent=[-np.pi, np.pi,␣
 ↪-np.pi, np.pi], cmap='viridis')
plt.colorbar(label='Difference in phase')
plt.xlabel('kx')
plt.ylabel('ky')
plt.title('Difference in phase, lower energy')
plt.xticks(ticks, labels)
plt.yticks(ticks, labels)
plt.show()
```



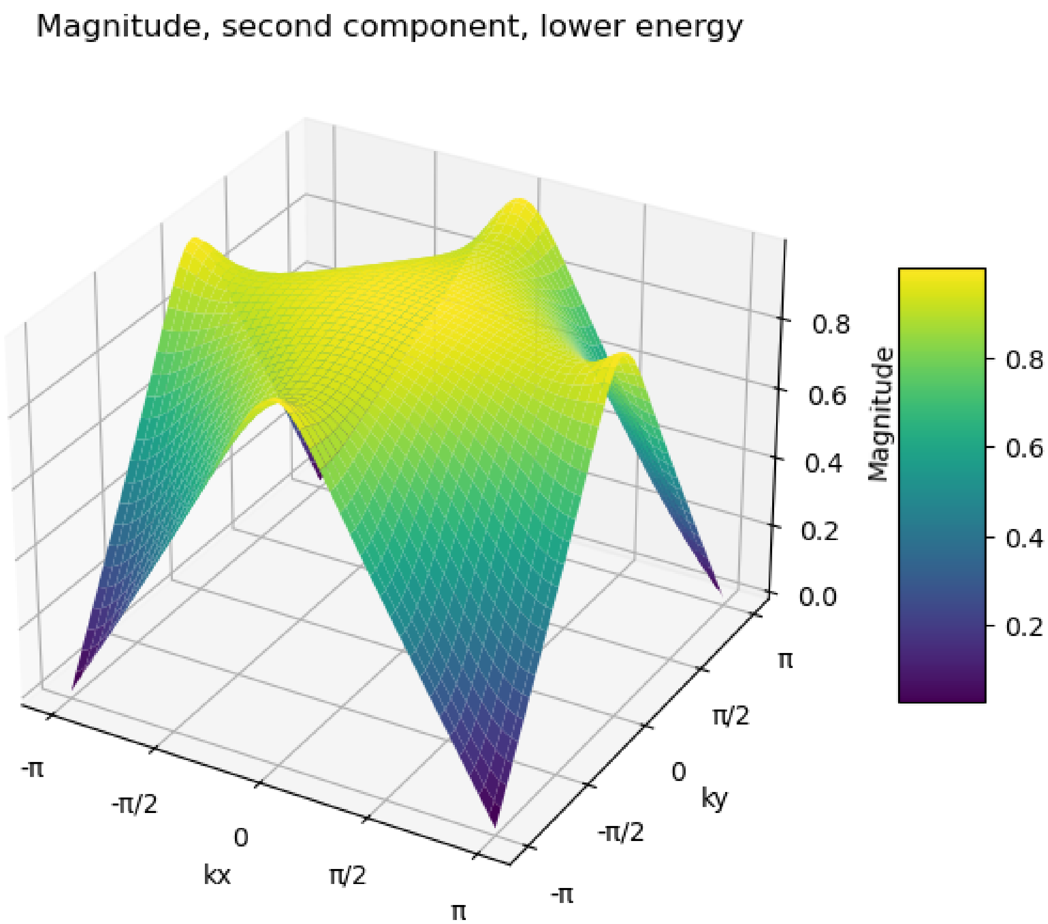
[3]:
```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 创建一个三维图形
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# 创建网格
X, Y = np.meshgrid(ks, ks)

# 绘制 uma[2,:,:] 的表面
surf = ax.plot_surface(X, Y, np.abs(uma[0,:,:]), cmap='viridis')
```

```python
# 设置坐标轴标签和标题
ax.set_xlabel('kx')
ax.set_ylabel('ky')
ax.set_zlabel('Magnitude')
ax.set_title('Magnitude, second component, lower energy')

# 设置刻度
ax.set_xticks(ticks)
ax.set_xticklabels(labels)
ax.set_yticks(ticks)
ax.set_yticklabels(labels)

# 添加 colorbar
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```



[4]:
```python
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# 创建一个三维图形
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# 创建网格
X, Y = np.meshgrid(ks, ks)

# 绘制 uma[2,:,:] 的表面
surf = ax.plot_surface(X, Y, np.abs(uma[1,:,:]), cmap='viridis')
```

```python
# 设置坐标轴标签和标题
ax.set_xlabel('kx')
ax.set_ylabel('ky')
ax.set_zlabel('Magnitude')
ax.set_title('Magnitude, second component, lower energy')

# 设置刻度
ax.set_xticks(ticks)
ax.set_xticklabels(labels)
ax.set_yticks(ticks)
ax.set_yticklabels(labels)

# 添加 colorbar
fig.colorbar(surf, shrink=0.5, aspect=5)

plt.show()
```




```
[5]: import plotly.graph_objects as go
     import numpy as np

     # 创建网格
     X, Y = np.meshgrid(ks, ks)

     # 绘制 uma[2,:,:] 的表面
     fig = go.Figure(data=[go.Surface(x=X, y=Y, z=np.abs(uma[1,:,:]),␣
      ↪colorscale='viridis')])

     # 设置坐标轴标签和标题
     fig.update_layout(scene=dict(xaxis_title='kx', yaxis_title='ky',␣
      ↪zaxis_title='Magnitude'),
```

```python
                  title='Magnitude, second component, lower energy')

# 设置刻度
fig.update_layout(scene=dict(xaxis=dict(tickvals=ticks, ticktext=labels),
                             yaxis=dict(tickvals=ticks, ticktext=labels)))

# 添加 colorbar
fig.update_layout(coloraxis_colorbar=dict(title='Magnitude', len=0.5))

fig.show()
```

[6]:

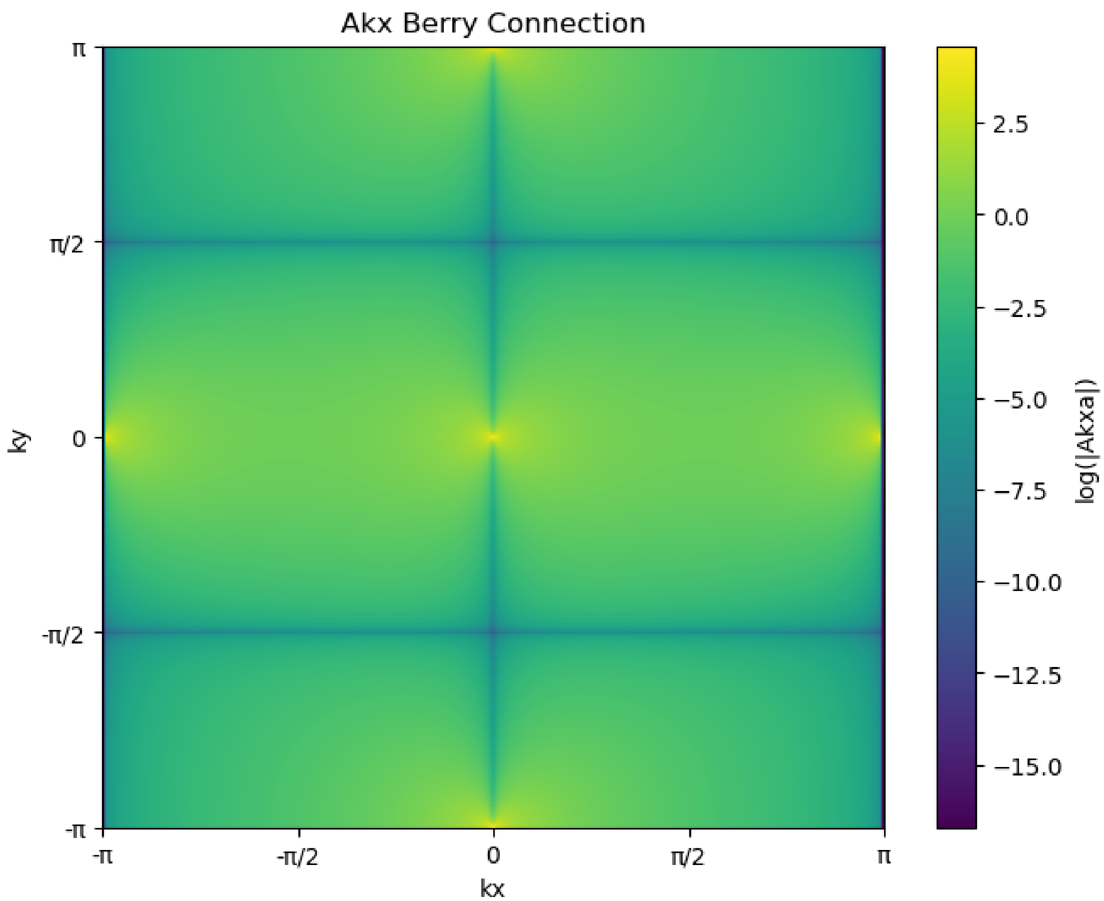
```python
import plotly.graph_objects as go
import numpy as np

# 创建网格
X, Y = np.meshgrid(ks, ks)

# 绘制 uma[2,:,:] 的表面
fig = go.Figure(data=[go.Surface(x=X, y=Y, z=np.abs(uma[0,:,:]),␣
 ↪colorscale='viridis')])

# 设置坐标轴标签和标题
fig.update_layout(scene=dict(xaxis_title='kx', yaxis_title='ky',␣
 ↪zaxis_title='Magnitude'),
                  title='Magnitude, second component, lower energy')

# 设置刻度
fig.update_layout(scene=dict(xaxis=dict(tickvals=ticks, ticktext=labels),
                             yaxis=dict(tickvals=ticks, ticktext=labels)))

# 添加 colorbar
fig.update_layout(coloraxis_colorbar=dict(title='Magnitude', len=0.5))

fig.show()
```

[7]:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import approx_fprime

# Define um1 and um2 functions in Python if not defined already

# Define functions dum1 and dum2
def dum1(kt):
    # Compute gradient of um1 with respect to kt using approx_fprime from scipy.
 ↪optimize
    return approx_fprime(kt, um1, epsilon=1e-6)

def dum2(k):
    # Compute gradients of real and imaginary parts of um2 with respect to k␣
 ↪using approx_fprime
    Rdum2 = approx_fprime(k, lambda t: np.real(um2(t)), epsilon=1e-6)
    Idum2 = approx_fprime(k, lambda t: np.imag(um2(t)), epsilon=1e-6)
    # Combine real and imaginary parts into complex gradient
    return Rdum2 + 1j * Idum2

# Define functions Amkx and Amky
def Amkx(k):
    # Compute the gradient of um1 and um2 at point k
    grad_um1 = np.conj(um1(k)) * dum1(k)
    grad_um2 = np.conj(um2(k)) * dum2(k)

    # Calculate Amkx using the gradients
    return grad_um1[0] + grad_um2[0]

def Amky(k):
    # Compute the gradient of um1 and um2 at point k
    grad_um1 = np.conj(um1(k)) * dum1(k)
    grad_um2 = np.conj(um2(k)) * dum2(k)

    # Calculate Amky using the gradients
    return grad_um1[1] + grad_um2[1]
```

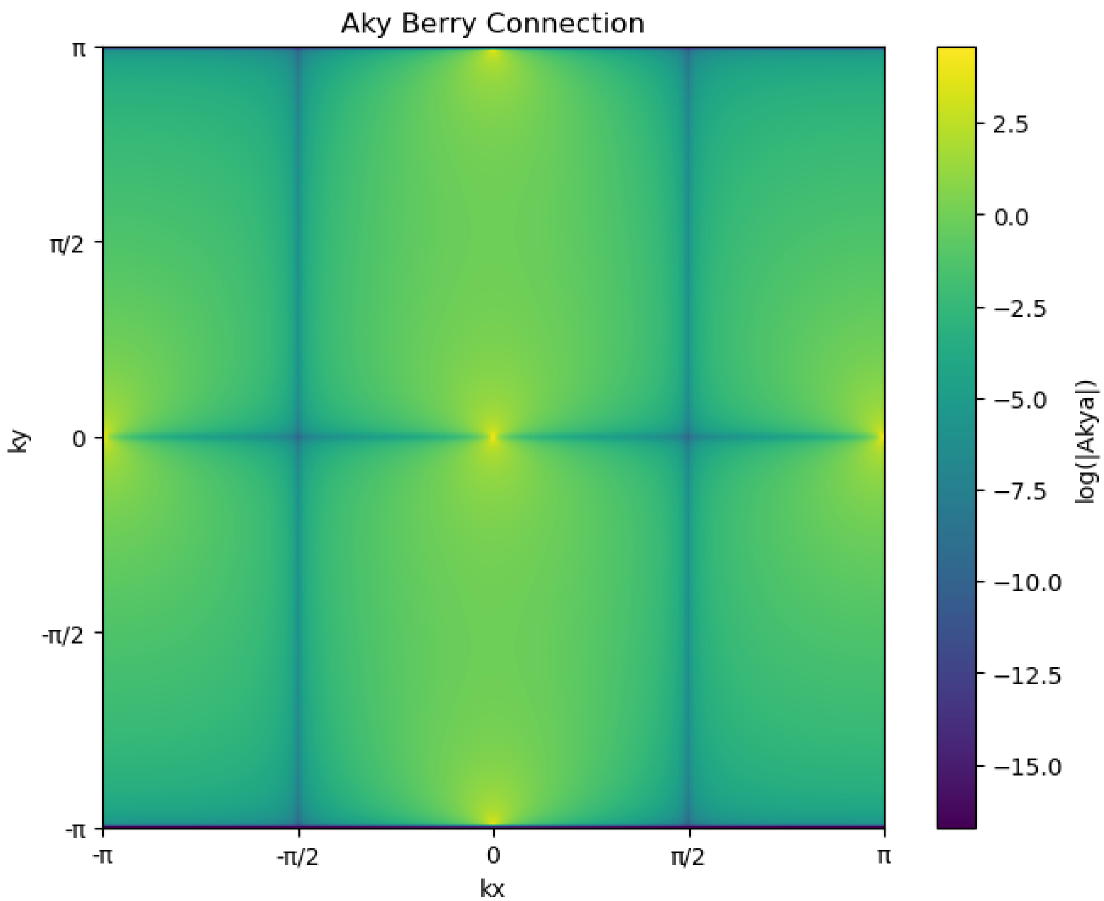
```python
# Define arrays Akxa and Akya
Akxa = np.zeros((l, l), dtype=np.complex128)
Akya = np.zeros((l, l), dtype=np.complex128)

# Fill arrays Akxa and Akya
for ii in range(l):
    for jj in range(l):
        Akxa[ii, jj] = Amkx(ka[ii, jj])
        Akya[ii, jj] = Amky(ka[ii, jj])
```

[8]:
```python
# Plot heatmap
plt.figure(figsize=(8, 6))
extent = [np.min(ks), np.max(ks), np.min(ks), np.max(ks)]
plt.imshow(np.log(np.abs(Akxa)), extent=extent, origin='lower')
plt.colorbar(label='log(|Akxa|)')
plt.xlabel('kx')
plt.ylabel('ky')
plt.title('Akx Berry Connection')
plt.xticks(ticks, labels)
plt.yticks(ticks, labels)
plt.show()
```



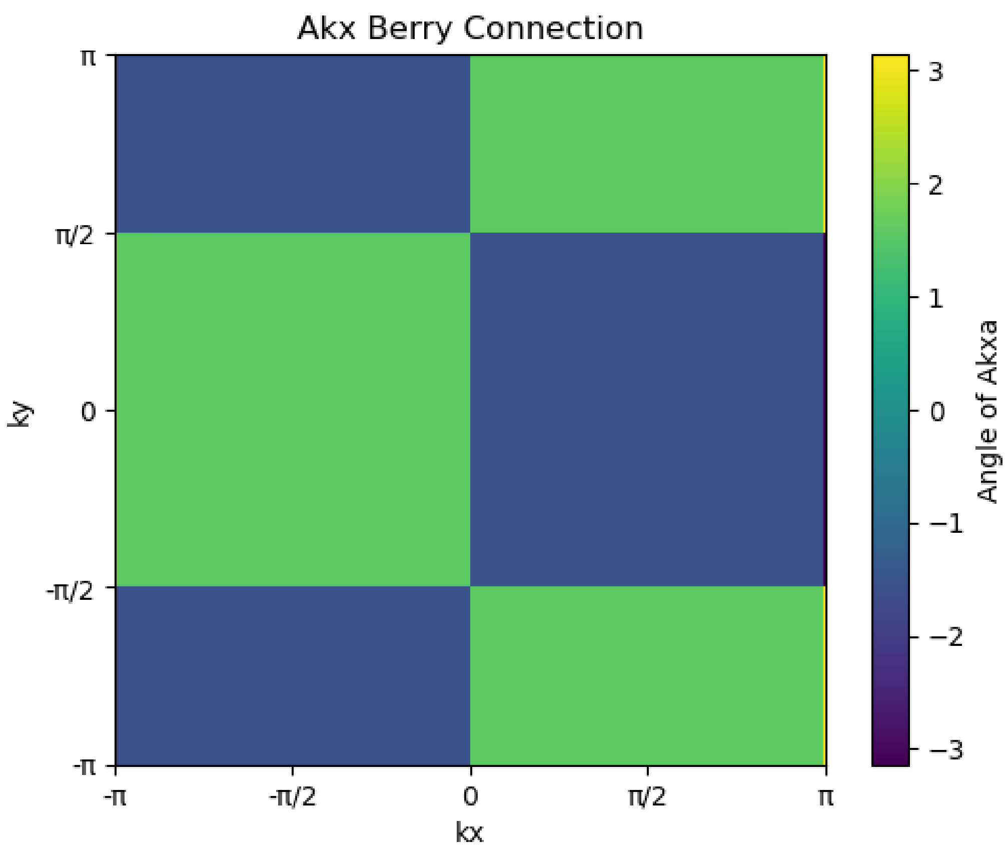
[9]:

```
# Plot heatmap for Akya
plt.figure(figsize=(8, 6))
extent = [np.min(ks), np.max(ks), np.min(ks), np.max(ks)]
plt.imshow(np.log(np.abs(Akya)), extent=extent, origin='lower')
plt.colorbar(label='log(|Akya|)')
plt.xlabel('kx')
plt.ylabel('ky')
plt.title('Aky Berry Connection')
plt.xticks(ticks, labels)
plt.yticks(ticks, labels)
plt.show()
```



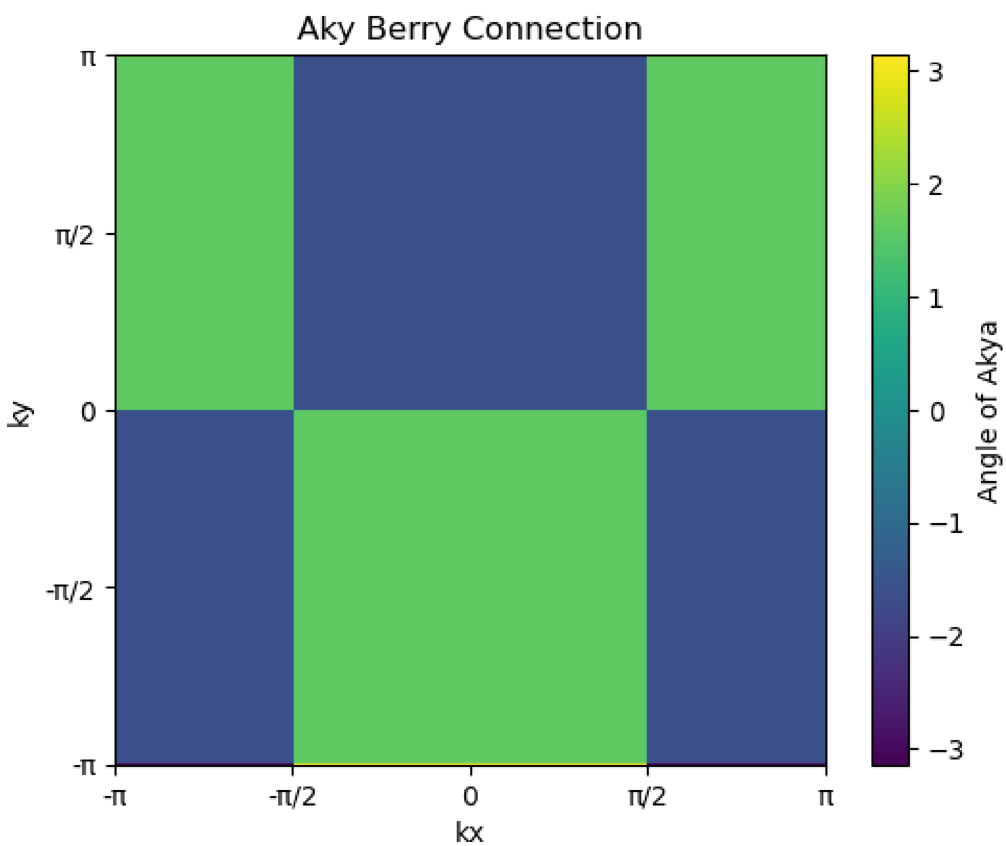
[10]:
```python
import matplotlib.pyplot as plt

# 绘制热图
plt.imshow(np.angle(Akxa), extent=[np.min(ks), np.max(ks), np.min(ks), np.
 ↪max(ks)], cmap='viridis')

# 设置坐标轴标签和标题
plt.xlabel('kx')
plt.ylabel('ky')
plt.title('Akx Berry Connection')

# 设置刻度
plt.xticks(ticks, labels)
```

```
plt.yticks(ticks, labels)

plt.colorbar(label='Angle of Akxa')

plt.show()
```



[11]:
```
import matplotlib.pyplot as plt

# 绘制热图
plt.imshow(np.angle(Akya), extent=[np.min(ks), np.max(ks), np.min(ks), np.
 ↪max(ks)], cmap='viridis')

# 设置坐标轴标签和标题
plt.xlabel('kx')
```

```
plt.ylabel('ky')
plt.title('Aky Berry Connection')

# 设置刻度
plt.xticks(ticks, labels)
plt.yticks(ticks, labels)

plt.colorbar(label='Angle of Akya')

plt.show()
```



[12]:
```
from scipy.optimize import approx_fprime


def DRAmkx(kt):
```

```
    return approx_fprime(kt, lambda t: np.real(Amkx(t)), epsilon=1e-6)

def DImAmkx(kt):
    return approx_fprime(kt, lambda t: np.imag(Amkx(t)), epsilon=1e-6)

def DRAmky(kt):
    return approx_fprime(kt, lambda t: np.real(Amky(t)), epsilon=1e-6)

def DImAmky(kt):
    return approx_fprime(kt, lambda t: np.imag(Amky(t)), epsilon=1e-6)
```

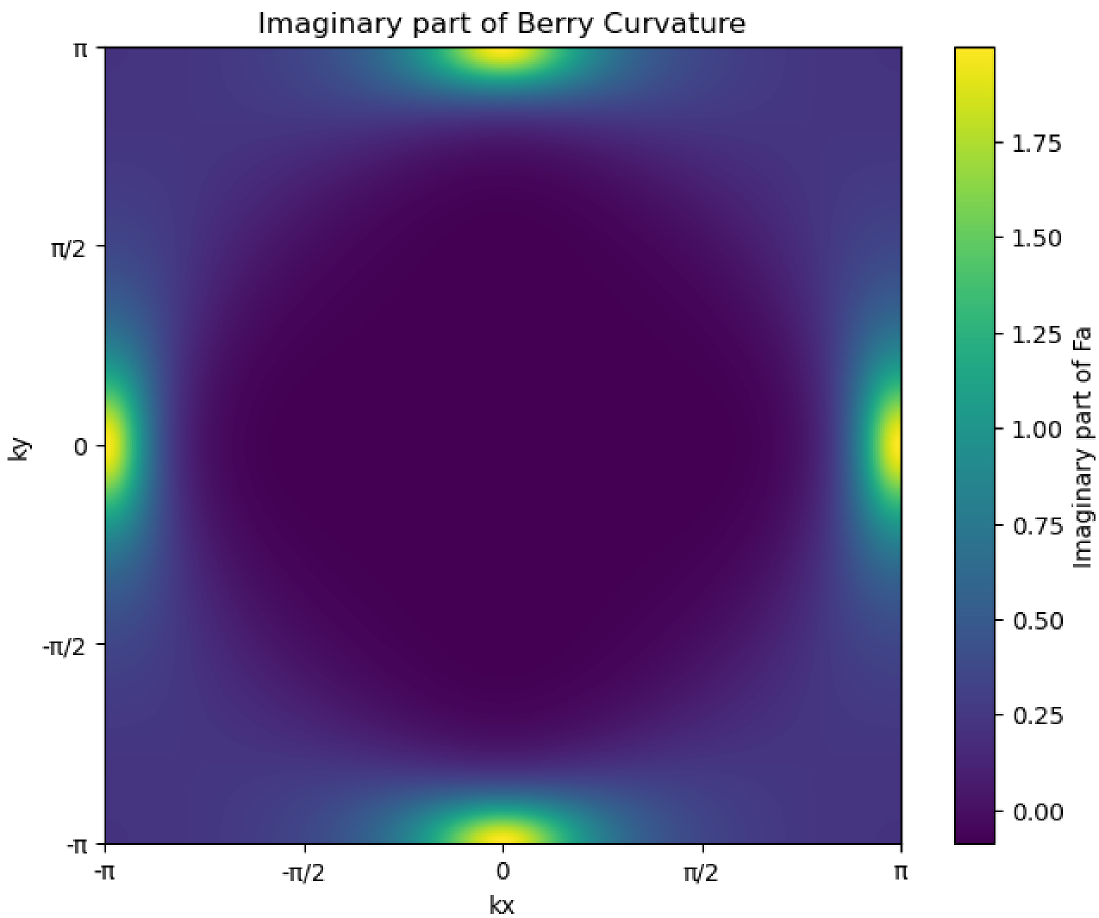
[13]:
```
def F(k):
    real_part = DRAmky(k)[0] + 1j * DImAmky(k)[0] - DRAmkx(k)[1] - 1j *␣
 ↪DImAmkx(k)[1]
    return real_part
```

[14]:
```
Fa = np.zeros((l, l), dtype=np.complex128)
for ii in range(l):
    for jj in range(l):
        Fa[ii, jj] = F(ka[ii, jj])
```

[15]:
```
max_real_part = np.max(np.real(Fa))
print(max_real_part)
```

```
0.9890691490990922
```

[16]:
```
plt.figure(figsize=(8, 6))
extent = [np.min(ks), np.max(ks), np.min(ks), np.max(ks)]
plt.imshow(np.imag(Fa), extent=extent, origin='lower')
plt.colorbar(label='Imaginary part of Fa')
plt.xlabel('kx')
plt.ylabel('ky')
plt.title('Imaginary part of Berry Curvature')
plt.xticks(ticks, labels)
plt.yticks(ticks, labels)
plt.show()
```



[17]:
```python
integral = np.sum(Fa) * (ks[1] - ks[0])**2 / (2 * np.pi * 1j)
print(integral)
```

```
(1.0250189495662116+1.5099338463691475e-12j)
```