目录

[模块一、沿b轴方向晶格常数热力图绘制 2](#_Toc178674198)

[模块二、双取代结构热力图 3](#_Toc178674199)

[模块三、gap与导电性质热力图 4](#_Toc178674200)

[模块四、D带中心热力图 6](#_Toc178674201)

[模块五、能带和dos图绘制 7](#_Toc178674202)

# 模块一、沿b轴方向晶格常数热力图绘制

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

def plot\_b\_heatmap(file\_path, sheet\_name, output\_path):

"""

读取给定Excel文件路径和工作表名称的数据，并绘制d带中心的热力图，然后保存到指定路径

并返回保存热力图的文件路径

:param file\_path: Excel文件的路径

:param sheet\_name: 工作表名称

:param output\_path: 保存热力图的文件路径

:return: 保存热力图的文件路径

"""

# 使用pandas读取指定工作表

df = pd.read\_excel(file\_path, sheet\_name=sheet\_name, index\_col=0)

# 创建热力图，设置数值的字体大小和保留小数位数

plt.figure(figsize=(12, 10))

sns.heatmap(df, annot=True, fmt=".2f", cmap="YlGnBu", linewidths=.5,

xticklabels=df.columns, yticklabels=df.index,

annot\_kws={'size': 8}) # 设置热力图中数值的字体大小

# 添加标题

plt.title('Heatmap of d-band Center', fontsize=16, fontweight='bold')

# 设置x轴和y轴标签的旋转角度和字体大小

plt.xticks(rotation=0, fontsize=12)

plt.yticks(rotation=0, fontsize=12)

# 保存热力图到指定路径

plt.savefig(output\_path, bbox\_inches='tight', dpi=300)

# 显示热力图

plt.close() # 关闭图像，避免在内存中占用资源

# 返回保存热力图的文件路径

return output\_path

# 主程序，调用函数并传入用户输入的文件路径、工作表名称和输出图像路径

if \_\_name\_\_ == "\_\_main\_\_":

# 让用户输入Excel文件路径和工作表名称

file\_path = r' 1.xlsx'

sheet\_name = '1'

output\_path = r'E:\桌面\heatmap.png' # 指定保存热力图的路径

# 调用函数并获取保存的图像路径

saved\_image\_path = plot\_b \_heatmap(file\_path, sheet\_name, output\_path)

print(f"Heatmap saved to: {saved\_image\_path}")

# 模块二、双取代结构热力图

import numpy as np

import pandas as pd

import seaborn as sns

from matplotlib import pyplot as plt

def plot\_heatmap\_from\_excel(file\_path, sheet\_name):

"""

从指定的Excel文件和工作表中读取数据并绘制热力图

:param file\_path: Excel文件的路径

:param sheet\_name: Excel文件中的工作表名称

"""

try:

# 从Excel文件读取数据

data = pd.read\_excel(file\_path, sheet\_name=sheet\_name)

# 设置索引

index = ["Sc", "Ti", "V", "Cr", "Mn", "Fe", "Co", "Ni", "Cu", "Zn",

"Y", "Zr", "Nb", "Mo", "Tc", "Ru", "Rh", "Pd", "Ag", "Cd",

"Hf", "Ta", "W", "Re", "Os", "Ir", "Pt", "Au", "Hg"]

data.index = index

# 绘制热力图

sns.set(font\_scale=0.8)

plt.figure(figsize=(10, 8))

plot = sns.heatmap(data, cmap="YlGnBu", vmax=2, vmin=-2, annot=True, fmt=".1f")

# 添加标题和标签

plt.xlabel("Metal Replacement Element", size=15)

plt.ylabel("Metal Replacement Element", size=15)

plt.title("Ef Calculation of Bimetallic Atomic Substitution in DDA-Cu Bimetallic Systems", size=20)

# 设置轴标签的旋转角度

plt.yticks(rotation=0)

plt.xticks(rotation=0)

# 显示图像

plt.show()

except Exception as e:

print(f"读取Excel文件或绘图时出错：{e}")

if \_\_name\_\_ == "\_\_main\_\_":

# 指定Excel文件路径和工作表

file\_path = "形成能PLUS.xlsx"

sheet\_name = "Sheet2"

# 调用绘图函数

plot\_heatmap\_from\_excel(file\_path, sheet\_name)

# 模块三、gap与导电性质热力图

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

from matplotlib.patches import Patch, Rectangle

from matplotlib.colors import ListedColormap

def plot\_bandgap\_heatmap(file\_path):

"""

读取给定Excel文件路径中的数据并绘制Bandgap的热力图，同时进行特殊标记

:param file\_path: Excel文件的路径

"""

# 读取Excel文件

df = pd.read\_excel(file\_path, sheet\_name=0) # 读取第一个sheet数据

sheet3 = pd.read\_excel(file\_path, sheet\_name=1, usecols="A:AC", nrows=26, header=None) # 读取第三个sheet的A1到AC26范围内数据，并包含第一行

# 定义刻度标签

labels = ['Sc', 'Ti', 'V', 'Cr', 'Mn', 'Fe', 'Co', 'Ni', 'Cu', 'Zn',

'Y', 'Zr', 'Nb', 'Mo', 'Tc', 'Ru', 'Rh', 'Pd', 'Ag', 'Cd',

'Hf', 'Ta', 'W', 'Re', 'Os', 'Ir', 'Pt', 'Au', 'Hg']

# 确保数据的第一行及以后的数据作为数据部分

df = df.iloc[0:, 1:]

# 将数据类型转换为数值型，强制转换所有数据

df = df.apply(pd.to\_numeric, errors='coerce') # 将无法转换的数据设为 NaN

# 定义颜色映射，使用渐深颜色和特殊的红色标记

colors = sns.color\_palette("Blues", as\_cmap=True) # 使用渐深蓝色

cmap = colors(np.linspace(0, 1, 256))

cmap[0] = np.array([0.5, 0.8, 0, 1]) # 将 0 值设为红色

cmap = ListedColormap(cmap)

# 统计不同颜色区块的数量

color\_counts = {'Metal': 0, 'Semi-metal': 0, 'Semi-conductor': 0}

for value in df.values.flatten():

if pd.notna(value): # 忽略 NaN 值

if value == 0:

color\_counts['Metal'] += 1

elif value < 0.1:

color\_counts['Semi-metal'] += 1

elif value >= 0.1:

color\_counts['Semi-conductor'] += 1

# 找出含有“half”的位置

half\_mask = sheet3.astype(str).applymap(lambda x: 'half' in x)

# 绘制热力图

plt.figure(figsize=(12, 10)) # 调整图表宽度以容纳左侧的图例

ax = sns.heatmap(df, annot=True, fmt=".2f", cmap=cmap, cbar=True, linewidths=0.5, linecolor='black', annot\_kws={"size": 8})

# 设置坐标轴标签

plt.xticks(ticks=np.arange(len(labels)) + 0.5, labels=labels, rotation=0, ha='right')

plt.yticks(ticks=np.arange(len(labels)) + 0.5, labels=labels, rotation=0)

# 添加标题和坐标轴标签

plt.title('Bandgap Analysis')

plt.xlabel('Elements')

# 添加图例，设置背景为白色和调整字体大小

handles = [

Patch(color='red', label=f'Metal (0.0): {color\_counts["Metal"]}'),

Patch(color=cmap(25), label=f'Semi-metal (<0.1): {color\_counts["Semi-metal"]}'),

Patch(color=cmap(200), label=f'Semi-conductor (>=0.1): {color\_counts["Semi-conductor"]}')

]

plt.legend(handles=handles, loc='upper left', bbox\_to\_anchor=(1, 1), fontsize='large', frameon=True, facecolor='white', framealpha=1)

# 在带有“half”的位置添加标记（例如圆圈）

for (i, j), val in np.ndenumerate(half\_mask):

if val:

plt.gca().add\_patch(plt.Circle((j + 0.5, i + 0.5), 0.2, color='black', fill=False, lw=2))

# 在数值为0.0的位置添加红色标记

for (i, j), val in np.ndenumerate(df.values):

if val == 0.0:

plt.gca().add\_patch(Rectangle((j, i), 1, 1, fill=False, edgecolor='green', lw=0.1))

# 显示图表

plt.show()

# 主程序，调用函数并传入用户输入的文件路径

if \_\_name\_\_ == "\_\_main\_\_":

file\_path = 'E:\\桌面\\bandgap.xlsx' # 这里输入文件路径

plot\_bandgap\_heatmap(file\_path)

# 模块四、D带中心热力图

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

def create\_heatmap(xlsx\_filename, sheet\_name, output\_filename=None):

"""

创建热力图的主函数。

参数:

xlsx\_filename: str, Excel文件的路径。

sheet\_name: str, 工作表的名称。

output\_filename: str, 可选，输出图片的文件名。

"""

try:

# 使用pandas读取指定工作表

df = pd.read\_excel(xlsx\_filename, sheet\_name=sheet\_name, index\_col=0)

# 创建热力图，设置数值的字体大小和保留小数位数

plt.figure(figsize=(12, 10))

heatmap = sns.heatmap(df, annot=True, fmt=".2f", cmap="YlGnBu", linewidths=.5,

xticklabels=df.columns, yticklabels=df.index,

annot\_kws={'size': 8}) # 设置热力图中数值的字体大小

# 添加标题

plt.title('Heatmap of d-band Center', fontsize=16, fontweight='bold')

# 设置x轴和y轴标签的旋转角度和字体大小

plt.xticks(rotation=0, fontsize=12)

plt.yticks(rotation=0, fontsize=12)

# 如果指定了输出文件名，则保存图片

if output\_filename:

plt.savefig(output\_filename)

# 显示热力图

plt.show()

except Exception as e:

print(f"An error occurred: {e}")

# 主程序，调用函数并传入用户输入的文件路径

if \_\_name\_\_ == "\_\_main\_\_":

# 使用示例

xlsx\_filename = r'E:\桌面\collected\_summaries\D带中心整理.xlsx'

sheet\_name = 'Bigger'

create\_heatmap(xlsx\_filename, sheet\_name)

# 模块五、能带和dos图绘制

import numpy as np

import matplotlib.pyplot as plt

import os

import pandas as pd

import re

from matplotlib.gridspec import GridSpec

def replace\_doscar\_data(current\_folder, reference\_folder):

current\_doscar\_path = os.path.join(current\_folder, 'DOSCAR')

reference\_doscar\_path = os.path.join(reference\_folder, 'DOSCAR')

with open(reference\_doscar\_path, 'r') as reference\_file:

for \_ in range(5):

next(reference\_file)

sixth\_line = next(reference\_file).strip().split()

if len(sixth\_line) >= 4:

data\_to\_replace = sixth\_line[3]

else:

print(f"参考文件 {reference\_doscar\_path} 中第六行第四列数据不完整。")

return

if os.path.exists(current\_doscar\_path):

with open(current\_doscar\_path, 'r') as current\_file:

lines = current\_file.readlines()

if len(lines) >= 6:

current\_data = lines[5].strip().split()

if len(current\_data) >= 4:

current\_data[3] = data\_to\_replace

lines[5] = ' '.join(current\_data) + '\n'

else:

print(f"当前文件夹 {current\_doscar\_path} 中 DOSCAR 第六行数据不完整。")

return

else:

print(f"当前文件夹 {current\_doscar\_path} 中 DOSCAR 文件行数不足。")

return

with open(current\_doscar\_path, 'w') as current\_file:

current\_file.writelines(lines)

else:

print(f"找不到路径 {current\_doscar\_path} 下的 DOSCAR 文件。")

def run\_vaspkit():

os.system("vaspkit 211")

os.system('echo "211" | vaspkit')

def load\_pdos\_data(file\_path, spin):

if not os.path.exists(file\_path):

raise FileNotFoundError(f"File {file\_path} not found.")

name = pd.read\_csv(file\_path, sep='\s+', header=None, nrows=1).values.tolist()[0]

df = pd.read\_csv(file\_path, sep='\s+', header=None, names=name)[1:]

df = df.apply(pd.to\_numeric, errors='coerce')

df['#Energy'] = df['#Energy'].astype(float)

cond = df[(df['#Energy'] > -2) & (df['#Energy'] < 2)]

return cond['#Energy'], cond['tot']

def plot\_band\_structure(band\_data\_file, dos\_data\_folder):

data = np.loadtxt(band\_data\_file, comments='#')

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

gs = GridSpec(1, 2, width\_ratios=[1, 1], wspace=0)

ax1 = fig.add\_subplot(gs[0])

ax2 = fig.add\_subplot(gs[1])

k = data[:, 0]

num\_bands = int((len(data[0]) - 1) / 2)

k\_min = np.min(k)

k\_max = np.max(k)

ignore\_indices = np.where((k == k\_min) | (k == k\_max))[0]

colors = ['red', 'blue']

for i in range(num\_bands):

energy\_up = data[:, 2 \* i + 1]

energy\_down = data[:, 2 \* i + 2]

energy\_up[ignore\_indices] = np.nan

energy\_down[ignore\_indices] = np.nan

ax1.plot(k, energy\_up, linestyle='-', color=colors[0], linewidth=5, label='Spin-Up' if i == 0 else "")

ax1.plot(k, energy\_down, linestyle='-', color=colors[1], linewidth=1.5, label='Spin-Down' if i == 0 else "")

sorted\_data = np.unique(k)

second\_smallest = sorted\_data[2]

second\_largest = sorted\_data[-2]

ax1.set\_xlabel('Wave Vector')

ax1.set\_ylabel('Energy (eV)')

ax1.set\_title('Band Structure')

ax1.grid(True)

ax1.set\_ylim(-2, 2)

ax1.set\_xlim(second\_smallest, second\_largest)

ax1.legend(fontsize=15)

ax1.text(0, -2.3, 'Γ', fontsize=12, ha='center')

ax1.text(np.max(k), -2.3, 'Y', fontsize=12, ha='center')

ax1.set\_xticks([])

folder\_path = os.getcwd()

element\_combo = folder\_path.split("/")[-1]

file\_name = folder\_path.split("/")[-2] + "/" + folder\_path.split("/")[-1]

DOS\_file = os.path.join("/home/xuff/zy/dos/01DRAWING", file\_name)

replace\_doscar\_data(folder\_path, DOS\_file)

num = re.findall('[A-Z][^A-Z]\*', element\_combo)

M1, M2 = num[0], num[1]

elements = [M1, M2, 'C', 'N', 'O', 'H']

y\_data\_up = []

y\_data\_dw = []

for element in elements:

x, y\_up = load\_pdos\_data(f'PDOS\_{element}\_UP.dat', os.path.join(folder\_path, 'PDOS\_{element}\_UP.dat'))

\_, y\_dw = load\_pdos\_data(f'PDOS\_{element}\_DW.dat', os.path.join(folder\_path, 'PDOS\_{element}\_DW.dat'))

y\_data\_up.append(y\_up)

y\_data\_dw.append(y\_dw)

x, y\_1\_up = load\_pdos\_data(f'PDOS\_{M1}\_UP.dat', os.path.join(folder\_path, 'PDOS\_{M1}\_UP.dat'))

\_, y\_1\_dw = load\_pdos\_data(f'PDOS\_{M1}\_DW.dat', os.path.join(folder\_path, 'PDOS\_{M1}\_DW.dat'))

x, y\_2\_up = load\_pdos\_data(f'PDOS\_{M2}\_UP.dat', os.path.join(folder\_path, 'PDOS\_{M2}\_UP.dat'))

\_, y\_2\_dw = load\_pdos\_data(f'PDOS\_{M2}\_DW.dat', os.path.join(folder\_path, 'PDOS\_{M2}\_DW.dat'))

y\_data\_up = [pd.to\_numeric(y, errors='coerce').fillna(0) for y in y\_data\_up]

y\_data\_dw = [pd.to\_numeric(y, errors='coerce').fillna(0) for y in y\_data\_dw]

y\_1\_up = pd.to\_numeric(y\_1\_up, errors='coerce').fillna(0)

y\_1\_dw = pd.to\_numeric(y\_1\_dw, errors='coerce').fillna(0)

y\_2\_up = pd.to\_numeric(y\_2\_up, errors='coerce').fillna(0)

y\_2\_dw = pd.to\_numeric(y\_2\_dw, errors='coerce').fillna(0)

total\_y\_up = sum(y\_data\_up)

total\_y\_dw = sum(y\_data\_dw)

ax2.fill\_betweenx(x, total\_y\_up, where=(total\_y\_up >= 0), color='grey', label='Total', alpha=0.3)

ax2.fill\_betweenx(x, total\_y\_dw, where=(total\_y\_dw <= 0), color='grey', alpha=0.3)

ax2.plot(y\_1\_up, x, color='deeppink', label=f'{M1}', linewidth=1)

ax2.plot(y\_1\_dw, x, color='deeppink', linewidth=1)

if M1 == M2:

ax2.plot(y\_2\_up, x, color='blue', linewidth=1)

else:

ax2.plot(y\_2\_up, x, color='blue', label=f'{M2}', linewidth=1)

ax2.plot(y\_2\_dw, x, color='blue', linewidth=1)

ax2.set\_xlabel('DOS')

ax2.set\_title('Density of States (DOS)')

ax2.grid(True)

ax2.set\_xlim(-20, 20)

ax2.set\_ylim(-2, 2)

ax2.legend(fontsize=15)

ax2.yaxis.set\_tick\_params(labelleft=False)

ax2.yaxis.set\_ticks\_position('none')

plt.setp(ax1.get\_xticklabels(), visible=False)

plt.subplots\_adjust(wspace=0)

plt.tight\_layout()

plt.show()

plt.savefig(f"{element\_combo}.png")

def main(current\_folder, reference\_folder, band\_data\_file, dos\_data\_folder):

run\_vaspkit()

plot\_band\_structure(band\_data\_file, dos\_data\_folder)

if \_\_name\_\_ == "\_\_main\_\_":

current\_folder = os.getcwd()

reference\_folder='/home/xuff/zy/dos/01DRAWING/05Mn/16MnRu'

# 设置路径

#reference\_folder = "/path/to/reference/folder" # 替换为参考文件夹路径

band\_data\_file = current\_folder+"/BAND.dat" # 替换为BAND.dat文件的路径

dos\_data\_folder = current\_folder+"/DOSCAR" # 替换为DOS数据文件夹的路径

# 调用主函数

main(current\_folder, reference\_folder, band\_data\_file, dos\_data\_folder)