import numpy as np  
from scipy.integrate import quad  
from scipy.optimize import curve\_fit  
import matplotlib.pyplot as plt  
import matplotlib as mpl  
  
# ========================  
# 1. Set academic plot style  
# ========================  
mpl.rcParams.update({  
 'font.family': 'Arial',  
 'font.size': 12,  
 'axes.labelsize': 12,  
 'axes.linewidth': 1.5,  
 'xtick.major.width': 1.5,  
 'ytick.major.width': 1.5,  
 'savefig.dpi': 300,  
 'savefig.bbox': 'tight'  
})  
  
# ========================  
# 2. Define Bloch-Grüneisen model  
# ========================  
def bloch\_gruneisen(T, rho0, A, Theta\_D):  
 """  
 Bloch-Grüneisen model function  
 Parameters:  
 T: Temperature (K)  
 rho0: Residual resistivity (Ω·m)  
 A: Material constant  
 Theta\_D: Debye temperature (K)  
 Returns:  
 Resistivity (Ω·m)  
 """  
 def integrand(x):  
 return x\*\*5 / ((np.exp(x) - 1) \* (1 - np.exp(-x)))  
   
 integral = np.array([quad(integrand, 0, Theta\_D/t)[0] if t > 0 else 0 for t in T])  
 return rho0 + A \* (T/Theta\_D)\*\*5 \* integral  
  
# ========================  
# 3. Generate/load experimental data (Example: Copper)  
# ========================  
# Experimental data (Temperature K, Resistivity Ω·m)  
T\_data = np.array([10, 50, 100, 150, 200, 250, 300])  
rho\_data = np.array([1.55, 1.68, 2.02, 2.45, 2.89, 3.30, 3.72]) \* 1e-8  
  
# Add 5% random noise to simulate real data  
np.random.seed(42)  
rho\_data = rho\_data \* (1 + 0.05 \* np.random.randn(len(rho\_data)))  
  
# ========================  
# 4. Fit the model  
# ========================  
# Initial guess [rho0, A, Theta\_D]  
p0 = [1.5e-8, 1e-8, 300]   
# Parameter bounds (avoid unphysical values)  
bounds = ([0, 0, 100], [5e-8, 1e-7, 500])   
  
# Perform fitting  
popt, pcov = curve\_fit(bloch\_gruneisen, T\_data, rho\_data,   
 p0=p0, bounds=bounds, maxfev=10000)  
  
# Calculate fitting errors (1σ standard deviation)  
perr = np.sqrt(np.diag(pcov))  
  
# Generate fitted curve  
T\_fit = np.linspace(10, 300, 100)  
rho\_fit = bloch\_gruneisen(T\_fit, \*popt)  
  
# ========================  
# 5. Plotting  
# ========================  
fig, ax = plt.subplots(figsize=(8, 6))  
  
# Plot experimental data with error bars  
ax.errorbar(T\_data, rho\_data\*1e8, yerr=0.05\*rho\_data\*1e8,   
 fmt='o', color='#d62728', markersize=8, capsize=5,  
 label='Experimental Data (Cu)')  
  
# Plot fitted curve  
ax.plot(T\_fit, rho\_fit\*1e8, 'b-', linewidth=2,  
 label=f'Bloch-Grüneisen Fit\n'  
 f'ρ₀ = {popt[0]\*1e8:.2f} ± {perr[0]\*1e8:.2f} ×10⁻⁸ Ω·m\n'  
 f'Θ\_D = {popt[2]:.0f} ± {perr[2]:.0f} K')  
  
# Axis labels and title  
ax.set\_xlabel('Temperature (K)', fontweight='bold')  
ax.set\_ylabel('Resistivity (×10⁻⁸ Ω·m)', fontweight='bold')  
ax.set\_title('Low-Temperature Resistivity Fit: Bloch-Grüneisen Model',   
 fontsize=14, pad=20, fontweight='bold')  
  
# Add theoretical formula  
ax.text(0.6, 0.2,   
 r'$\rho(T) = \rho\_0 + A\left(\frac{T}{\Theta\_D}\right)^5 \int\_0^{\Theta\_D/T} \frac{x^5}{(e^x-1)(1-e^{-x})}dx$',  
 transform=ax.transAxes, fontsize=10,   
 bbox=dict(facecolor='white', alpha=0.8))  
  
# Legend and grid  
ax.legend(frameon=False, loc='upper left', fontsize=10)  
ax.grid(linestyle='--', alpha=0.3)  
  
# ========================  
# 6. Save figures  
# ========================  
plt.savefig('Bloch\_Gruneisen\_Fit\_Cu.pdf') # Vector format  
plt.savefig('Bloch\_Gruneisen\_Fit\_Cu.tiff', dpi=600) # Raster format  
plt.show()  
  
# ========================  
# 7. Print fitting results  
# ========================  
print(f"Fitted parameters (95% confidence interval):")  
print(f"ρ₀ = ({popt[0]\*1e8:.3f} ± {1.96\*perr[0]\*1e8:.3f}) ×10⁻⁸ Ω·m")  
print(f"A = ({popt[1]\*1e8:.3f} ± {1.96\*perr[1]\*1e8:.3f}) ×10⁻⁸ Ω·m")  
print(f"Θ\_D = ({popt[2]:.1f} ± {1.96\*perr[2]:.1f}) K")

import numpy as np  
import matplotlib.pyplot as plt  
  
# 数据输入  
metals = ['Pt', 'Cu', 'W', 'Fe', 'Ni']  
alpha = np.array([0.00392, 0.00393, 0.0045, 0.0050, 0.0060]) # 温度系数α (单位：/℃)  
temperatures = np.linspace(0, 80, 100) # 0-80℃的连续温度范围  
  
# 计算电阻变化率 ΔR/R0 = α \* ΔT  
resistivity\_change = []  
for a in alpha:  
 change = a \* (temperatures - 0) \* 100 # 转换为百分比  
 resistivity\_change.append(change)  
  
# 绘图设置  
plt.figure(figsize=(10, 6), dpi=300)  
colors = ['#9467bd', '#1f77b4', '#ff7f0e', '#2ca02c', '#d62728'] # 色盲友好配色  
line\_styles = ['-', '--', '-.', ':', (0, (3, 1, 1, 1))] # 区分线型  
  
# 绘制曲线  
for i, metal in enumerate(metals):  
 plt.plot(temperatures, resistivity\_change[i],   
 label=f'{metal} (α={alpha[i]:.4f}/℃)',  
 color=colors[i],  
 linestyle=line\_styles[i],  
 linewidth=2)  
  
# 图表标注  
plt.title('Resistivity Change Rate vs Temperature', fontsize=14, pad=20)  
plt.xlabel('Temperature (°C)', fontsize=12, labelpad=10)  
plt.ylabel('Resistivity Change Rate (%)', fontsize=12, labelpad=10)  
plt.legend(frameon=False, fontsize=10, loc='upper left')  
plt.grid(True, linestyle='--', alpha=0.3)  
  
# 坐标轴范围调整  
plt.xlim(0, 80)  
plt.ylim(0, 50) # 根据Ni在80℃时的值≈48%设定  
  
# 保存为矢量图  
plt.savefig('Resistivity\_Change\_Rate.pdf', bbox\_inches='tight', transparent=True)  
plt.show()

import matplotlib.pyplot as plt  
  
plt.figure(figsize=(8, 6), dpi=300)  
bars = plt.bar(metals, alpha,   
 color=['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd'],  
 edgecolor='black', linewidth=0.7)  
  
# 添加误差棒（若有）  
plt.errorbar(metals, alpha, yerr=error, fmt='none',   
 ecolor='black', capsize=5, capthick=1)  
  
# 标注数值  
for bar in bars:  
 height = bar.get\_height()  
 plt.text(bar.get\_x() + bar.get\_width()/2., height,  
 f'{height:.1f}', ha='center', va='bottom', fontsize=10)  
  
# 格式设置  
plt.title('Temperature Coefficient (α) of Metals', fontsize=14, pad=20)  
plt.xlabel('Metal Type', fontsize=12, labelpad=10)  
plt.ylabel('α (×10⁻³/℃)', fontsize=12, labelpad=10)  
plt.xticks(rotation=45, ha='right', fontsize=10)  
plt.grid(axis='y', linestyle='--', alpha=0.3)  
plt.tight\_layout()  
plt.savefig('Alpha\_Comparison.png', bbox\_inches='tight', transparent=True)  
plt.show()