# 逸出功的测量-实验报告

**姓名：夏弘宇 学号：2023011004 实验日期：20250305 实验组/台号：L4**

## 【实验目的】

1. 了解热电子发射的规律。

2. 掌握里查孙直线法，测定阴极材料(钨)的电子逸出功的方法。

3. 了解真空电子管的发展历程、电子二极管的结构及实现原理。

## 【实验仪器】

·KEITHLEY2231A 型直流稳压稳流电源

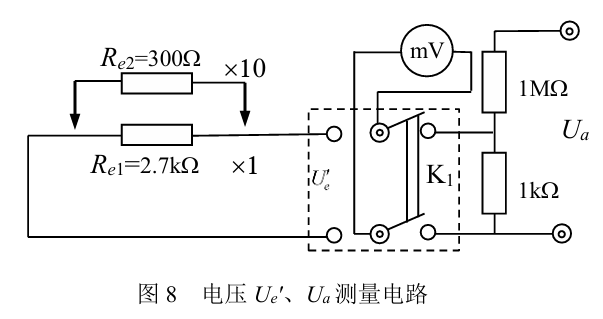
·UTP1003S 型直流稳压稳流电源

·数字电压表（mV）：量程200mV

·实验板：安装有标准二极管，灯丝KH两端已经并联由两个相同电阻R（千欧量级）串联而成的电阻，两个电阻的连接点用C表示。

·导线

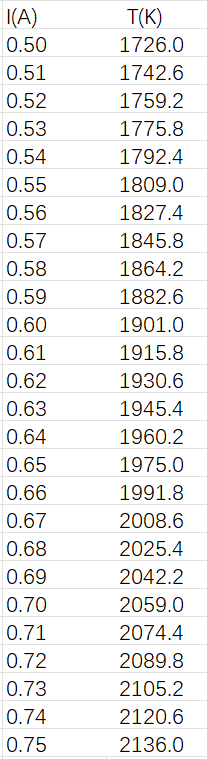
·实验面包板及元件：可以搭建如图8所示的线路。



## 【数据处理】

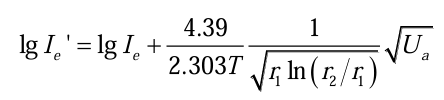
1. 记录的原始数据

其中的温度数值是通过查表+相邻数据线性插值得到的，代码及结果如下

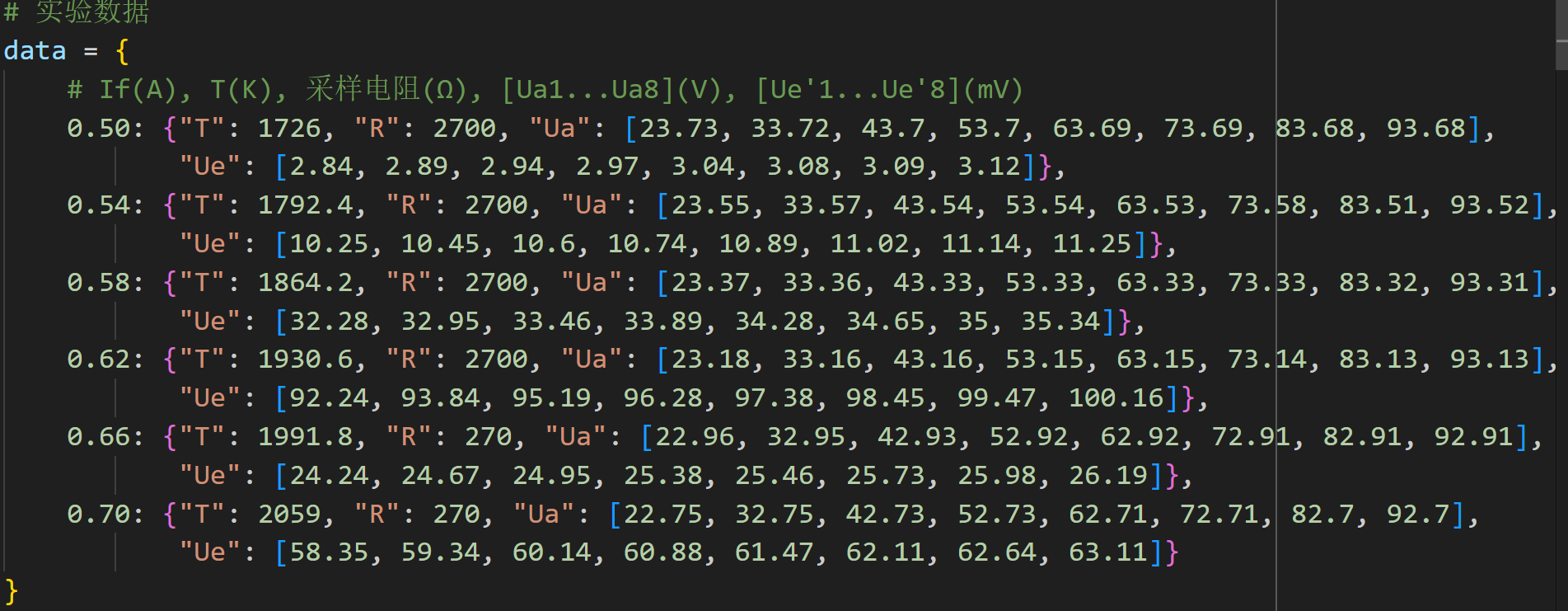


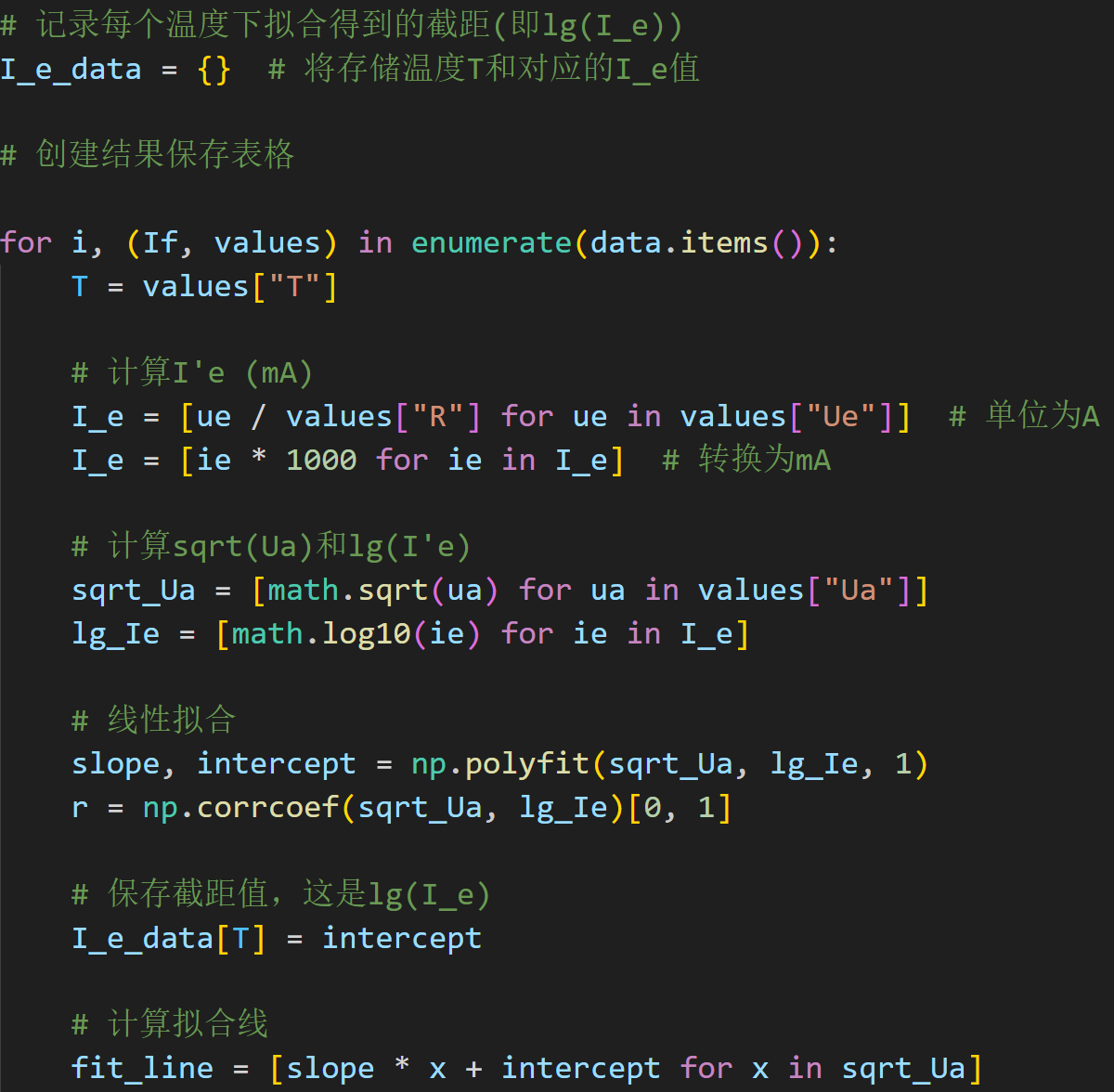


2. 消除肖特基效应影响，计算准确的I\_e值

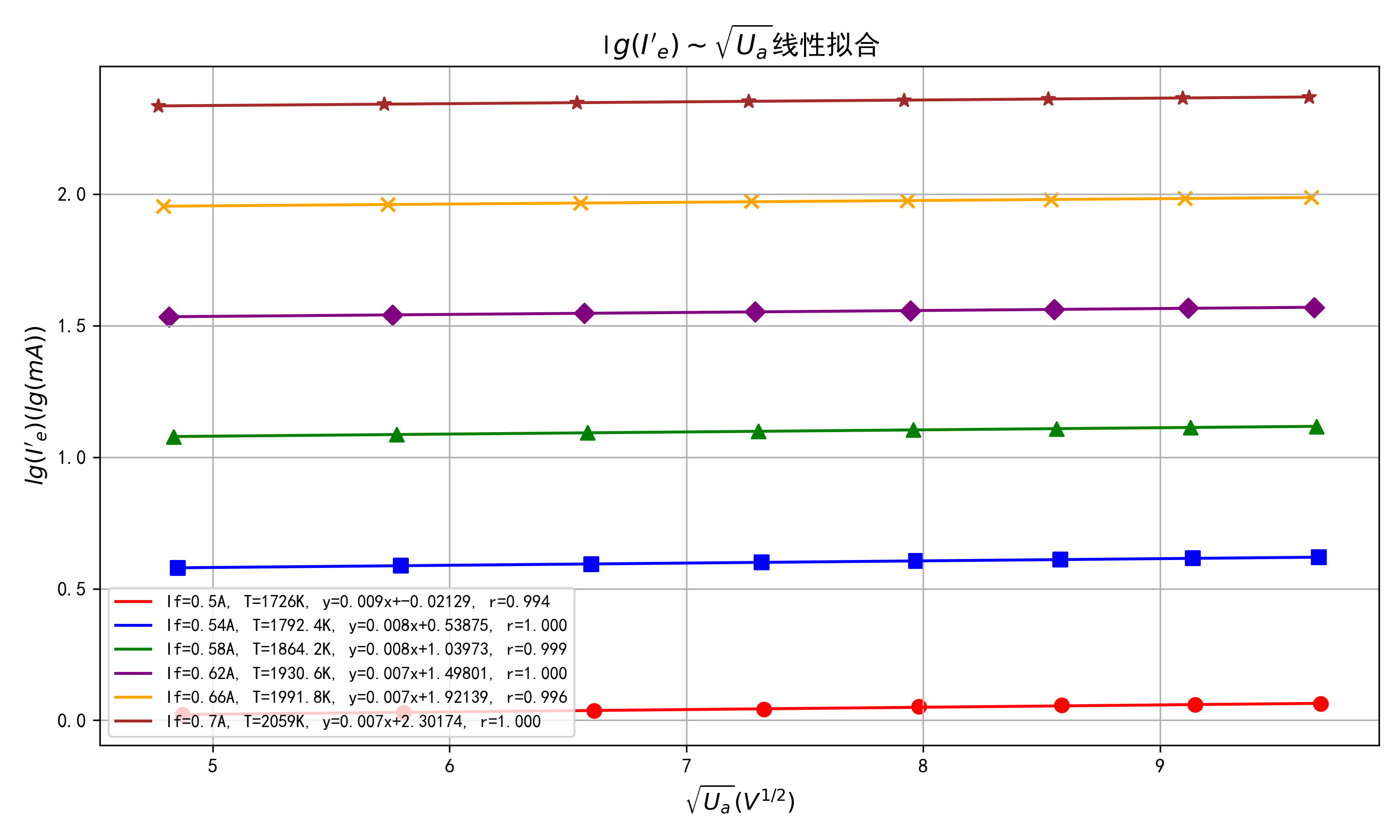


由上述线性关系，每两行数据可以拟合一条直线，得出一个T下的I\_e值。代码如下：





得到如下图像：



输出的信息：零场发射电流I\_e值:

T = 1726K: lg(I\_e) = -0.0213, I\_e = 9.5215e-01 mA

T = 1792.4K: lg(I\_e) = 0.5387, I\_e = 3.4574e+00 mA

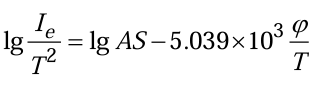
T = 1864.2K: lg(I\_e) = 1.0397, I\_e = 1.0958e+01 mA

T = 1930.6K: lg(I\_e) = 1.4980, I\_e = 3.1478e+01 mA

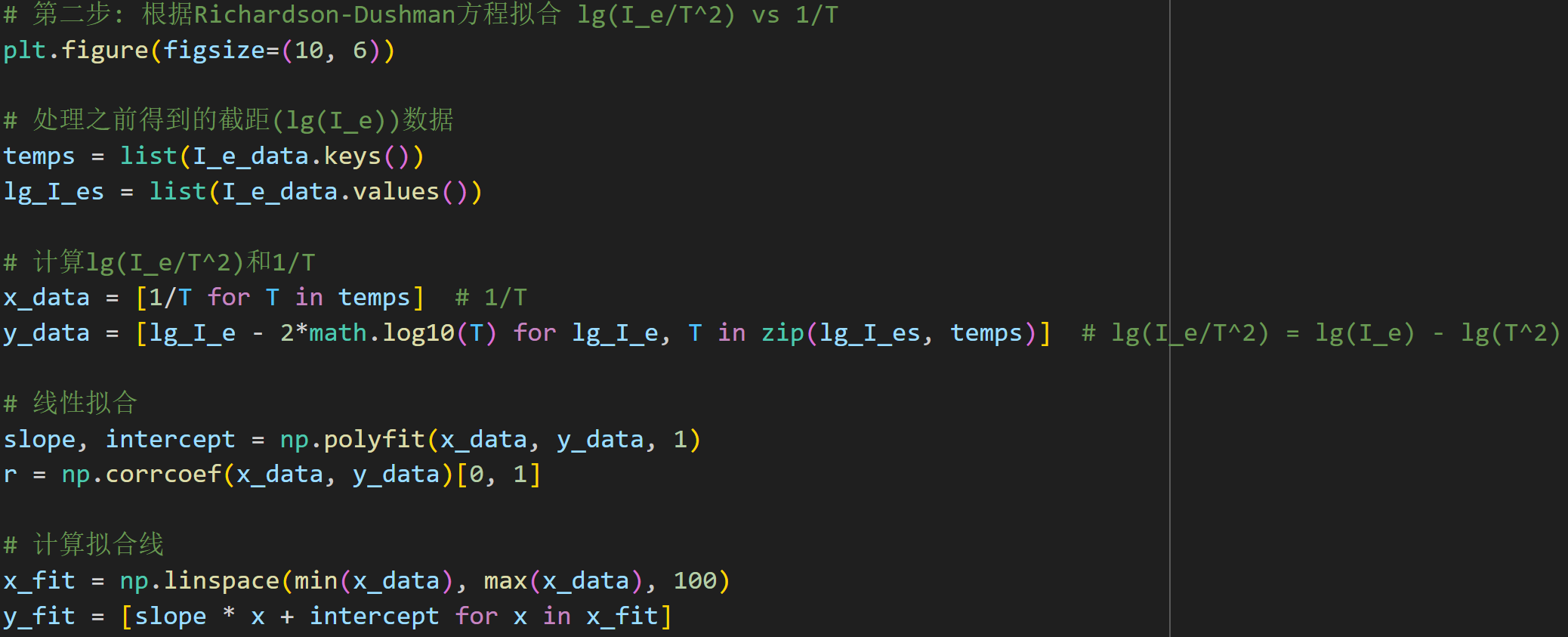
T = 1991.8K: lg(I\_e) = 1.9214, I\_e = 8.3442e+01 mA

T = 2059K: lg(I\_e) = 2.3017, I\_e = 2.0033e+02 mA

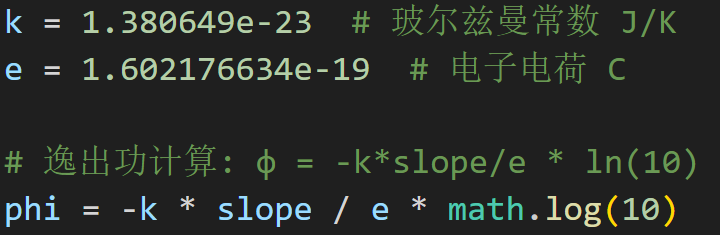
3. 利用里查孙直线法计算逸出功的值



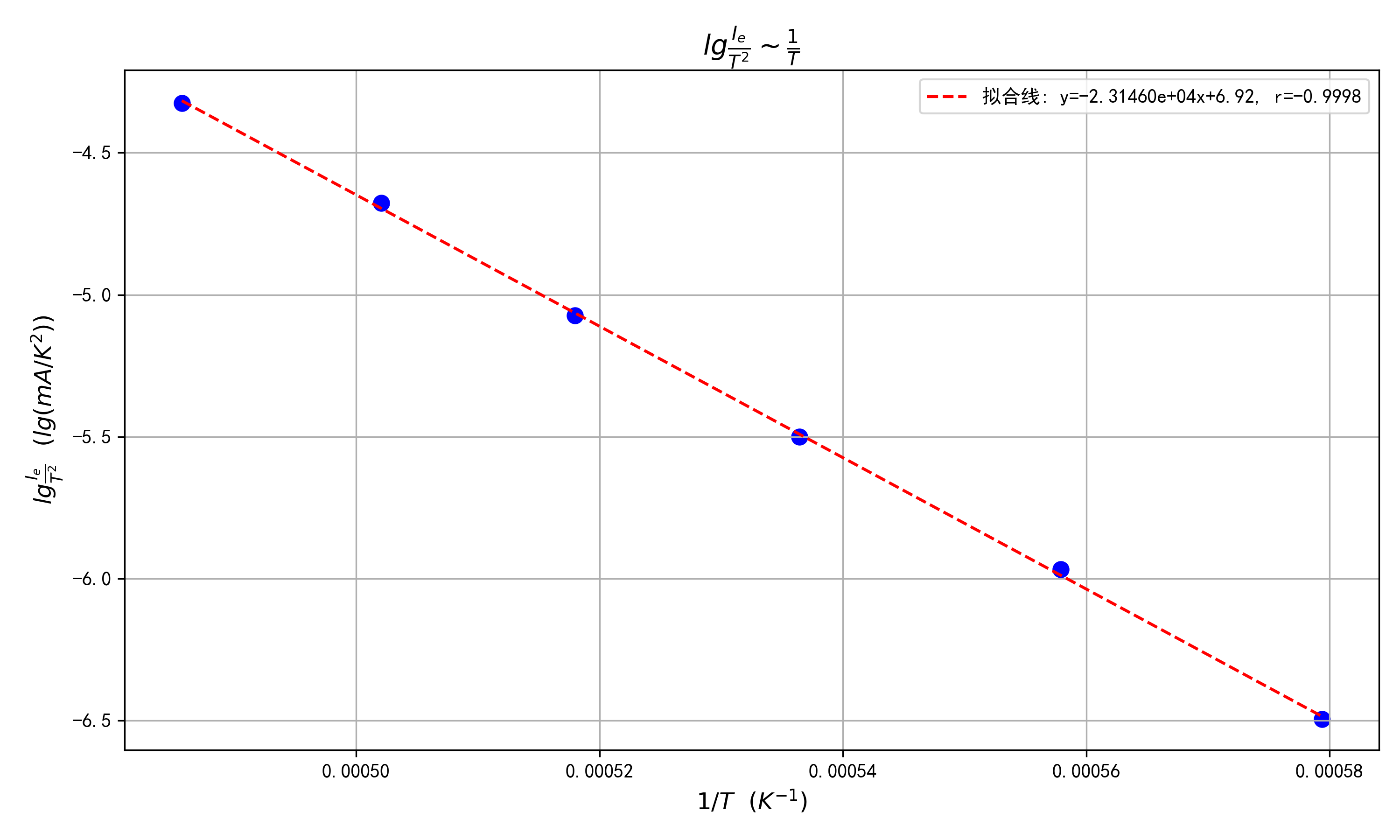
依据上述线性关系，以及第二步中得出的6组数据，可以拟合一条直线，此时我们关注直线的斜率。代码如下：



计算步骤如下，直接推导了原始式子，在计算过程中更为精确



拟合结果图示如下



输出的计算结果如下：

斜率 = -2.3146e+04

截距 = 6.9250

相关系数 r = -0.9998

逸出功 φ = 4.5927 eV

得出结论，所测逸出功为4.5927eV。

查阅资料知钨电子逸出功为4.54eV

相对误差

## 【实验总结】

本实验的巧妙之处在于通过取对数、作图拟合等方法，消除了肖特基效应的影响，并绕开了A、S等物理量的测量，难点在于理解实验原理后电路的搭建。总体说来，以下几点是本次实验的收获与总结：

本次实验总计要拟合7组数据，用excel半自动地处理工作量自然很大，因此我在本次实验中数据处理的部分利用Python既有的线性拟合库以及便捷的向量处理简化了上述过程，并使得数据在中间过程几乎不会丢失精度，极大提高了数据处理的效率。

作为电学实验，实验数据的准确度也比较高，在拟合过程中，可以看到|r|值非常接近于1，拟合误差很小，最终的结果也可以看出相对误差仅1%。

实验过程中，一开始我发现测得的U’e仅有不到3mV，身边同学基本是20mV左右及以上，就怀疑实验仪器有问题。但经过分析，以及实验后续结果测试后，发现这事实上是在预期范围内的。经思考，这应该是因为不同的灯丝情况不同，自然会有个体差异。所以不应该直接从数值上与周边同学进行对比，而是在原理上验证自身数据的合理性。

最后感谢助教老师在实验过程中对我的悉心指导，受益匪浅！

## 【原始数据记录】



## 【完整代码】

1. 插值.py

from scipy import interpolate

# 原始数据

A\_original = [0.500, 0.550, 0.600, 0.650, 0.700, 0.750]

T\_original = [1726, 1809, 1901, 1975, 2059, 2136]

# 创建线性插值函数

f = interpolate.interp1d(A\_original, T\_original, kind='linear')

# 打印表头

print("A\t\tT(K)")

# 生成从0.50到0.75，步长为0.01的A值，并计算对应的T值

for a\_value in range(50, 76):  # 从50到75，对应0.50到0.75

    a = a\_value / 100.0  # 转换为浮点数

    t = f(a)  # 计算插值

    print(f"{a:.2f}\t\t{t:.1f}")

2. 数据处理与画图.py

import numpy as np

import matplotlib.pyplot as plt

from matplotlib.font\_manager import FontProperties

import math

import sys

# 设置全局字体为黑体

plt.rcParams['font.sans-serif'] = ['SimHei']

# 解决坐标轴负号显示问题

plt.rcParams['axes.unicode\_minus'] = False

class OutputRedirector:

    def \_\_init\_\_(self, filename):

        self.terminal = sys.stdout

        self.file = open(filename, 'w', encoding='utf-8')

    def write(self, message):

        self.terminal.write(message)

        self.file.write(message)

    def flush(self):

        self.terminal.flush()

        self.file.flush()

    def close(self):

        self.file.close()

# 设置输出重定向

sys.stdout = OutputRedirector('实验结果.txt')

# 实验数据

data = {

    # If(A), T(K), 采样电阻(Ω), [Ua1...Ua8](V), [Ue'1...Ue'8](mV)

    0.50: {"T": 1726, "R": 2700, "Ua": [23.73, 33.72, 43.7, 53.7, 63.69, 73.69, 83.68, 93.68],

           "Ue": [2.84, 2.89, 2.94, 2.97, 3.04, 3.08, 3.09, 3.12]},

    0.54: {"T": 1792.4, "R": 2700, "Ua": [23.55, 33.57, 43.54, 53.54, 63.53, 73.58, 83.51, 93.52],

           "Ue": [10.25, 10.45, 10.6, 10.74, 10.89, 11.02, 11.14, 11.25]},

    0.58: {"T": 1864.2, "R": 2700, "Ua": [23.37, 33.36, 43.33, 53.33, 63.33, 73.33, 83.32, 93.31],

           "Ue": [32.28, 32.95, 33.46, 33.89, 34.28, 34.65, 35, 35.34]},

    0.62: {"T": 1930.6, "R": 2700, "Ua": [23.18, 33.16, 43.16, 53.15, 63.15, 73.14, 83.13, 93.13],

           "Ue": [92.24, 93.84, 95.19, 96.28, 97.38, 98.45, 99.47, 100.16]},

    0.66: {"T": 1991.8, "R": 270, "Ua": [22.96, 32.95, 42.93, 52.92, 62.92, 72.91, 82.91, 92.91],

           "Ue": [24.24, 24.67, 24.95, 25.38, 25.46, 25.73, 25.98, 26.19]},

    0.70: {"T": 2059, "R": 270, "Ua": [22.75, 32.75, 42.73, 52.73, 62.71, 72.71, 82.7, 92.7],

           "Ue": [58.35, 59.34, 60.14, 60.88, 61.47, 62.11, 62.64, 63.11]}

}

# 第一步: 创建图1: lg(I'e) ~ sqrt(Ua)

plt.figure(figsize=(10, 6))

colors = ['red', 'blue', 'green', 'purple', 'orange', 'brown']

markers = ['o', 's', '^', 'D', 'x', '\*']

# 记录每个温度下拟合得到的截距(即lg(I\_e))

I\_e\_data = {}  # 将存储温度T和对应的I\_e值

# 创建结果保存表格

for i, (If, values) in enumerate(data.items()):

    T = values["T"]

    # 计算I'e (mA)

    I\_e = [ue / values["R"] for ue in values["Ue"]]  # 单位为A

    I\_e = [ie \* 1000 for ie in I\_e]  # 转换为mA

    # 计算sqrt(Ua)和lg(I'e)

    sqrt\_Ua = [math.sqrt(ua) for ua in values["Ua"]]

    lg\_Ie = [math.log10(ie) for ie in I\_e]

    # 线性拟合

    slope, intercept = np.polyfit(sqrt\_Ua, lg\_Ie, 1)

    r = np.corrcoef(sqrt\_Ua, lg\_Ie)[0, 1]

    # 保存截距值，这是lg(I\_e)

    I\_e\_data[T] = intercept

    # 计算拟合线

    fit\_line = [slope \* x + intercept for x in sqrt\_Ua]

    # 绘制散点和拟合线

    plt.scatter(sqrt\_Ua, lg\_Ie, marker=markers[i], color=colors[i], s=50)

    plt.plot(sqrt\_Ua, fit\_line, color=colors[i],

             label=f'If={If}A, T={T}K, y={slope:.3f}x+{intercept:.5f}, r={r:.3f}')

plt.xlabel('$\sqrt{U\_a} (V^{1/2})$', fontsize=12)

plt.ylabel('$lg(I\'\_e) (lg(mA))$', fontsize=12)

plt.title('l$g(I\'\_e) \sim \sqrt{U\_a}$线性拟合', fontsize=14)

plt.grid(True)

plt.legend(loc='best', fontsize=9)

plt.tight\_layout()

plt.savefig('lg\_Ie\_vs\_sqrt\_Ua\_fitted.png', dpi=300)

# 第二步: 根据Richardson-Dushman方程拟合 lg(I\_e/T^2) vs 1/T

plt.figure(figsize=(10, 6))

# 处理之前得到的截距(lg(I\_e))数据

temps = list(I\_e\_data.keys())

lg\_I\_es = list(I\_e\_data.values())

# 计算lg(I\_e/T^2)和1/T

x\_data = [1/T for T in temps]  # 1/T

y\_data = [lg\_I\_e - 2\*math.log10(T) for lg\_I\_e, T in zip(lg\_I\_es, temps)]  # lg(I\_e/T^2) = lg(I\_e) - lg(T^2)

# 线性拟合

slope, intercept = np.polyfit(x\_data, y\_data, 1)

r = np.corrcoef(x\_data, y\_data)[0, 1]

# 计算拟合线

x\_fit = np.linspace(min(x\_data), max(x\_data), 100)

y\_fit = [slope \* x + intercept for x in x\_fit]

# 绘制散点和拟合线

plt.scatter(x\_data, y\_data, marker='o', s=60, color='blue')

plt.plot(x\_fit, y\_fit, '--', color='red',

         label=f'拟合线: y={slope:.5e}x+{intercept:.2f}, r={r:.4f}')

plt.xlabel('$1/T ~~ (K^{-1})$', fontsize=12)

plt.ylabel('$lg\\frac{I\_e}{T^2} ~~ (lg(mA/K^2))$', fontsize=12)

plt.title('$lg\\frac{I\_e}{T^2} \sim \\frac{1}{T}$', fontsize=14)

plt.grid(True)

plt.legend(loc='best', fontsize=10)

plt.tight\_layout()

plt.savefig('Richardson\_Dushman\_fit.png', dpi=300)

# 计算物理量：逸出功

# 根据Richardson-Dushman方程：ln(I\_e/T^2) = ln(A) - eφ/(kT)

# 斜率 = -eφ/(k\*ln(10))

k = 1.380649e-23  # 玻尔兹曼常数 J/K

e = 1.602176634e-19  # 电子电荷 C

# 逸出功计算: φ = -k\*slope/e \* ln(10)

phi = -k \* slope / e \* math.log(10)

plt.show()

# 打印逸出功结果到一个表格

print("\n========== 实验结果总结 ==========")

print("零场发射电流I\_e值:")

for T, lg\_I\_e in I\_e\_data.items():

    I\_e = 10\*\*lg\_I\_e  # 单位为mA

    print(f"T = {T}K: lg(I\_e) = {lg\_I\_e:.4f}, I\_e = {I\_e:.4e} mA")

print(f"\n斜率 = {slope:.4e}")

print(f"截距 = {intercept:.4f}")

print(f"相关系数 r = {r:.4f}")

print(f"逸出功 φ = {phi:.4f} eV")

sys.stdout.close()

# 恢复标准输出

sys.stdout = sys.\_\_stdout\_\_