

散件时代

1 第一版

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # 解决 Matplotlib 中文显示问题
5 import matplotlib
6 matplotlib.rcParams['font.sans-serif'] = ['SimHei'] # 例如使用黑体 (SimHei)
7 matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
8
9 # 读取 Excel 文件
10 file_path = "./datas_learn/B1.xlsx" # 请确保该文件在当前目录
11 xls = pd.ExcelFile(file_path)
12
13 # 选择第一个 sheet
14 df = xls.parse(sheet_name=0)
15
16 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
17 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
18
19 # 重新命名列名
20 df_filtered.columns = ["SMU-1 电压 (V)", "SMU-1 电流 (A)"]
21
22 # 转换数据为数值类型 (防止字符串干扰)
23 df_filtered = df_filtered.astype(float)
24
25 # 提取自变量 (电压) 和因变量 (电流)
26 x = df_filtered["SMU-1 电压 (V)"]
27 y = df_filtered["SMU-1 电流 (A)"]
28
29 print("df_filtered: ", df_filtered.head()) # 打印前几行查看数据
30
31 # 绘制曲线
32 plt.figure(figsize=(8, 6))
33 plt.plot(x, y, marker='o', linestyle='-', color='b', label="Voltage-Current Curve")
34 plt.xlabel("SMU-1 电压 (V)")
35 plt.ylabel("SMU-1 电流 (A)")
36 plt.title("电压-电流曲线")
37 plt.legend()
38 plt.grid(True)
39
40 # 显示图像
41 plt.show()
42
```

2 第二版

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # 解决 Matplotlib 中文显示问题
5 import matplotlib
```

```

6 matplotlib.rcParams['font.sans-serif'] = ['SimHei'] # 例如使用黑体 (SimHei)
7 matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
8
9 # 读取 Excel 文件
10 file_path = "./datas_learn/B1.xlsx" # 请确保该文件在当前目录
11 xls = pd.ExcelFile(file_path)
12
13 # 选择第一个 sheet
14 df = xls.parse(sheet_name=0)
15
16 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
17 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
18
19 # 重新命名列名
20 df_filtered.columns = ["SMU-1 电压 (V)", "SMU-1 电流 (A)"]
21
22 # 转换数据为数值类型 (防止字符串干扰)
23 df_filtered = df_filtered.astype(float)
24
25 # 提取自变量 (电压) 和因变量 (电流)
26 x = df_filtered["SMU-1 电压 (V)"]
27 y = df_filtered["SMU-1 电流 (A)"]
28
29 print("df_filtered: ", df_filtered.head()) # 打印前几行查看数据
30
31 # 绘制曲线
32 plt.figure(figsize=(8, 6))
33 plt.plot(x, y, marker='o', linestyle='-', color='b', label="Voltage-Current Curve")
34 plt.xlabel("SMU-1 电压 (V)")
35 plt.ylabel("SMU-1 电流 (A)")
36 plt.title("电压-电流曲线")
37 plt.legend()
38 plt.grid(True)
39
40 # 显示图像
41 plt.show()
42

```

3 基于第二版修改列名

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3
4 # 解决 Matplotlib 中文显示问题
5 import matplotlib
6 matplotlib.rcParams['font.sans-serif'] = ['SimHei'] # 例如使用黑体 (SimHei)
7 matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
8
9 # 读取 Excel 文件
10 file_path = "./datas_learn/B1.xlsx" # 请确保该文件在当前目录
11 xls = pd.ExcelFile(file_path)
12
13 # 选择第一个 sheet
14 df = xls.parse(sheet_name=0)
15
16 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)

```

```

17 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
18
19 # 重新命名列名
20 df_filtered.columns = ["电压 (V)", "电流 (A)"]
21
22 # 转换数据为数值类型 (防止字符串干扰)
23 df_filtered = df_filtered.astype(float)
24
25 # 提取自变量 (电压) 和因变量 (电流)
26 x = df_filtered["电压 (V)"]
27 y = df_filtered["电流 (A)"]
28
29 print("df_filtered: ", df_filtered.head()) # 打印前几行查看数据
30
31 # 绘制曲线
32 plt.figure(figsize=(8, 6))
33 plt.plot(x, y, marker='o', linestyle='-', color='b', label="V - A Curve")
34 plt.xlabel("电压 (V)")
35 plt.ylabel("电流 (A)")
36 plt.title("电压-电流曲线")
37 plt.legend()
38 plt.grid(True)
39
40 # 显示图像
41 plt.show()
42

```

4 基于第二版+写出拟合函数表达式

```

1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import numpy as np
4
5 # 解决 Matplotlib 中文显示问题
6 import matplotlib
7 matplotlib.rcParams['font.sans-serif'] = ['SimHei'] # 例如使用黑体 (SimHei)
8 matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
9
10 # 读取 Excel 文件
11 file_path = "./datas_learn/B2-3-21-30.xlsx" # 请确保该文件在当前目录
12 xls = pd.ExcelFile(file_path)
13
14 # 选择第一个 sheet
15 df = xls.parse(sheet_name=0)
16
17 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
18 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
19
20 # 重新命名列名
21 df_filtered.columns = ["电压 (V)", "电流 (A)"]
22
23 # 转换数据为数值类型 (防止字符串干扰)
24 df_filtered = df_filtered.astype(float)
25
26 # 提取自变量 (电压) 和因变量 (电流)
27 x = df_filtered["电压 (V)"]

```

```

28 y = df_filtered["电流 (A)"]
29
30 # 多项式拟合函数
31 def polynomial_fit(x, y, degree=3):
32     p = np.polyfit(x, y, degree) # 拟合多项式
33     poly = np.poly1d(p)
34     y_fit = poly(x)
35     return p, y_fit
36
37 # 调用多项式拟合
38 degree = 10 # 可以调整为你需要的拟合次数
39 p, y_fit = polynomial_fit(x, y, degree)
40
41 # 打印拟合的多项式系数
42 print(f"拟合多项式的系数（从高次到低次）：{p}")
43
44 # 准备拟合函数的字符串表达式
45 equation_str = " + ".join([f"{coef:.2e}x^{degree-i}" for i, coef in enumerate(p)])
46
47 # 绘制数据和拟合曲线
48 plt.figure(figsize=(8, 6))
49 plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据")
50 plt.plot(x, y_fit, linestyle='--', color='r', label=f"{degree}次多项式拟合")
51
52 # 在图形中添加拟合函数的表达式
53 plt.text(0.1, 0.1, f"拟合函数: y = {equation_str}", transform=plt.gca().transAxes,
54         fontsize=12, verticalalignment='top')
55
56 # 图表设置
57 plt.xlabel("电压 (V)")
58 plt.ylabel("电流 (A)")
59 plt.title("电压-电流曲线及拟合")
60 plt.legend()
61 plt.grid(True)
62
63 # 显示图像
64 plt.show()

```

5 增加了决定系数 R^2

为了衡量拟合效果，常用的指标是 **决定系数 (R^2)**，它能够衡量拟合曲线与实际数据的接近程度。 R^2 值越接近 1，说明拟合效果越好。

R^2 的计算公式为：

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$$

其中：

- y_i 是原始数据点的实际值，
- \hat{y}_i 是拟合曲线上的预测值，
- \bar{y} 是实际数据的均值。

```

1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  # 解决 Matplotlib 中文显示问题
6  import matplotlib
7  matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 使用微软雅黑 (或其他支持上标的
   字体)
8  matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
9
10 # 读取 Excel 文件
11 # file_path = "./datas_learn/B1.xlsx" # 请确保该文件在当前目录
12 # file_path = "./datas_learn/B2-3-21-30.xlsx" # 请确保该文件在当前目录
13 file_path = "./datas_learn/T2.xlsx" # 请确保该文件在当前目录
14 xls = pd.ExcelFile(file_path)
15
16 # 选择第一个 sheet
17 df = xls.parse(sheet_name=0)
18
19 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
20 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
21
22 # 重新命名列名
23 df_filtered.columns = ["电压 (V)", "电流 (A)"]
24
25 # 转换数据为数值类型 (防止字符串干扰)
26 df_filtered = df_filtered.astype(float)
27
28 # 提取自变量 (电压) 和因变量 (电流)
29 x = df_filtered["电压 (V)"]
30 y = df_filtered["电流 (A)"]
31
32 # 多项式拟合函数
33 def polynomial_fit(x, y, degree=3):
34     p = np.polyfit(x, y, degree) # 拟合多项式
35     poly = np.poly1d(p)
36     y_fit = poly(x)
37     return p, y_fit
38
39 # 调用多项式拟合
40 degree = 10 # 可以调整为你需要的拟合次数
41 p, y_fit = polynomial_fit(x, y, degree)
42
43 # 打印拟合的多项式系数
44 print(f"拟合多项式的系数 (从高次到低次) : {p}")
45
46 # 准备拟合函数的字符串表达式
47 equation_str = " + ".join([f"{coef:.2e}x^{degree-i}" for i, coef in enumerate(p)])
48
49 # 计算 R² (决定系数)
50 ss_residual = np.sum((y - y_fit)**2) # 残差平方和
51 ss_total = np.sum((y - np.mean(y))**2) # 总平方和
52 r_squared = 1 - (ss_residual / ss_total)
53
54 # 打印 R² 值
55 print(f"决定系数: R² = {r_squared:.4f}")

```

```

56
57 # 绘制数据和拟合曲线
58 plt.figure(figsize=(8, 6))
59 plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据")
60 plt.plot(x, y_fit, linestyle='--', color='r', label=f"{degree}次多项式拟合")
61
62 # 在图形中添加拟合函数的表达式
63 plt.text(0, 0.04, f"拟合函数: y = {equation_str}", transform=plt.gca().transAxes,
64         fontsize=12, verticalalignment='top')
65
66 # 在图形中添加  $R^2$  值, 使用  $R^2$  替代  $R^2$ 
67 plt.text(0.8, 0.5, f" $R^2 = {r_squared:.4f}$ ", transform=plt.gca().transAxes, fontsize=12,
68         verticalalignment='top')
69
70 # 图表设置
71 plt.xlabel("电压 (V)")
72 plt.ylabel("电流 (A)")
73 plt.title("电压-电流曲线及拟合")
74 plt.legend()
75 plt.grid(True)
76
77 # 显示图像
78 plt.show()
79

```

函数时代

1 单文件画图+多项式拟合二合一函数 - analyze_data

功能:

1. 自定义参数

(a) 文件路径

(b) 拟合次数

(c) 拟合曲线的线型, 粗细和颜色

```

1  import pandas as pd
2  import matplotlib
3  import matplotlib.pyplot as plt
4  import numpy as np
5
6  # 设置中文字体, 解决字体显示问题
7  matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 例如使用微软雅黑 (SimHei)
8  matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
9
10 def analyze_data(file_path, degree=3, show_equation=True, show_r_squared=True,
11                 line_style='-', line_color='b',
12                 line_width=1.5):
13     """
14     读取数据, 进行多项式拟合并绘制结果, 同时可以控制图表的细节和拟合的显示内容。
15
16     参数:
17     - file_path: Excel 文件的路径。
18
19

```

```

17 - degree: 多项式拟合的次数, 默认3。
18 - show_equation: 是否在图中显示拟合函数表达式, 默认显示。
19 - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
20 - line_style: 曲线的线型, 默认为 '-' (实线)。
21 - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
22 - line_width: 曲线的线宽, 默认为 1.5。
23 """
24
25 # 读取 Excel 文件
26 xls = pd.ExcelFile(file_path)
27 df = xls.parse(sheet_name=0)
28
29 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
30 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
31
32 # 重新命名列名
33 df_filtered.columns = ["电压 (V)", "电流 (A)"]
34
35 # 转换数据为数值类型 (防止字符串干扰)
36 df_filtered = df_filtered.astype(float)
37
38 # 提取自变量 (电压) 和因变量 (电流)
39 x = df_filtered["电压 (V)"]
40 y = df_filtered["电流 (A)"]
41
42 # 多项式拟合函数
43 def polynomial_fit(x, y, degree=3):
44     p = np.polyfit(x, y, degree) # 拟合多项式
45     poly = np.poly1d(p)
46     y_fit = poly(x)
47     return p, y_fit
48
49 # 调用多项式拟合
50 p, y_fit = polynomial_fit(x, y, degree)
51
52 # 打印拟合的多项式系数
53 print(f"拟合多项式的系数 (从高次到低次): {p}")
54
55 # 准备拟合函数的字符串表达式
56 equation_str = " + ".join([f"{coef:.2e}x^{degree - i}" for i, coef in enumerate(p)])
57
58 # 计算  $R^2$  (决定系数)
59 ss_residual = np.sum((y - y_fit) ** 2) # 残差平方和
60 ss_total = np.sum((y - np.mean(y)) ** 2) # 总平方和
61 r_squared = 1 - (ss_residual / ss_total)
62
63 # 打印  $R^2$  值
64 print(f"决定系数:  $R^2$  = {r_squared:.12f}")
65
66 # 绘制数据和拟合曲线
67 plt.figure(figsize=(8, 6))
68 plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据")
69 plt.plot(x, y_fit, linestyle=line_style, color=line_color, label=f"{degree}次多项式拟
合", linewidth=line_width)
70
71 # 在图形中添加拟合函数的表达式

```

```

72     if show_equation:
73         plt.text(0, 0.1, f"拟合函数: y = {equation_str}", transform=plt.gca().transAxes,
74                 fontsize=12,
75                 verticalalignment='top')
76         # 在图形中添加 R² 值, 使用 R^2 替代 R²
77         if show_r_squared:
78             plt.text(0.8, 0.5, f"R^2 = {r_squared:.4f}", transform=plt.gca().transAxes,
79                     fontsize=12,
80                     verticalalignment='top')
81         # 图表设置
82         plt.xlabel("电压 (V)")
83         plt.ylabel("电流 (A)")
84         plt.title("电压-电流曲线及拟合")
85         plt.legend()
86         plt.grid(True)
87
88         # 显示图像
89         plt.show()
90
91
92 # 示例: 使用该函数进行分析
93 file_path = "./datas_learn/B1.xlsx" # 替换为你自己的文件路径
94 analyze_data(file_path, degree=5, show_equation=True, show_r_squared=True, line_style='--',
95             line_color='r',
96             line_width=1)

```

2 DeepSeek修改

必须吐槽两句, gpt-4o打死改不出Latex指数显示, ds一次成功...

增加功能:

1. 原始数据点和连接线可以自定义
2. 修改了表达式的输出方式, 更加好看了

```

1  import pandas as pd
2  import matplotlib
3  import matplotlib.pyplot as plt
4  import numpy as np
5
6  # 设置中文字体, 解决字体显示问题
7  matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 例如使用微软雅黑 (SimHei)
8  matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
9
10 def analyze_data(file_path, degree=3, show_equation=True, show_r_squared=True,
11                 line_style='-', line_color='b',
12                 line_width=1.5):
13     """
14     读取数据, 进行多项式拟合并绘制结果, 同时可以控制图表的细节和拟合的显示内容。
15
16     参数:
17     - file_path: Excel 文件的路径。

```



```

17 - degree: 多项式拟合的次数, 默认3。
18 - show_equation: 是否在图中显示拟合函数表达式, 默认显示。
19 - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
20 - line_style: 曲线的线型, 默认为 '-' (实线)。
21 - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
22 - line_width: 曲线的线宽, 默认为 1.5。
23 """
24
25 # 读取 Excel 文件
26 xls = pd.ExcelFile(file_path)
27 df = xls.parse(sheet_name=0)
28
29 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
30 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
31
32 # 重新命名列名
33 df_filtered.columns = ["电压 (V)", "电流 (A)"]
34
35 # 转换数据为数值类型 (防止字符串干扰)
36 df_filtered = df_filtered.astype(float)
37
38 # 提取自变量 (电压) 和因变量 (电流)
39 x = df_filtered["电压 (V)"]
40 y = df_filtered["电流 (A)"]
41
42 # 多项式拟合函数
43 def polynomial_fit(x, y, degree=3):
44     p = np.polyfit(x, y, degree) # 拟合多项式
45     poly = np.poly1d(p)
46     y_fit = poly(x)
47     return p, y_fit
48
49 # 调用多项式拟合
50 p, y_fit = polynomial_fit(x, y, degree)
51
52 # 打印拟合的多项式系数
53 print(f"拟合多项式的系数 (从高次到低次) : {p}")
54
55 # 准备拟合函数的字符串表达式 (优化科学计数法显示)
56 equation_terms = []
57 for i, coef in enumerate(p):
58     power = degree - i
59     # 格式化系数为科学计数法
60     coef_str = f"{coef:.2e}".replace("e", "\\cdot10^{") + "}"
61     # 处理负号和小数点
62     if coef < 0:
63         coef_str = f"({coef_str})"
64     # 添加 x 的幂次
65     if power == 0:
66         term = coef_str
67     else:
68         term = f"{coef_str}x^{power}"
69     equation_terms.append(term)
70
71 # 拼接拟合函数表达式
72 equation_str = " + ".join(equation_terms)

```

```

73
74 # 计算  $R^2$  (决定系数)
75 ss_residual = np.sum((y - y_fit) ** 2) # 残差平方和
76 ss_total = np.sum((y - np.mean(y)) ** 2) # 总平方和
77 r_squared = 1 - (ss_residual / ss_total)
78
79 # 打印  $R^2$  值
80 print(f"决定系数:  $R^2 = \{r\_squared:.12f\}")$ 
81
82 # 绘制数据和拟合曲线
83 plt.figure(figsize=(8, 6))
84
85 # 设置原始数据点的样式和连接线的样式
86 plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据", markersize=3,
87          markerfacecolor='white', markeredgewidth=1, linewidth=2) # 数据点为白色, 线
条粗细为2
88
89 # 绘制拟合曲线
90 plt.plot(x, y_fit, linestyle=line_style, color=line_color, label=f"{degree}次多项式拟
合", linewidth=line_width)
91
92 # 在图形中添加拟合函数的表达式 (使用 LaTeX 样式)
93 if show_equation:
94     plt.text(0.0, 0.06, f"拟合函数:  $y = \{equation\_str\}$ ",
95             transform=plt.gca().transAxes, fontsize=12,
96             verticalalignment='top', ha='left', bbox=dict(facecolor='white',
97                 alpha=0.8))
98
99 # 在图形中添加  $R^2$  值, 使用 LaTeX 样式
100 if show_r_squared:
101     plt.text(0.8, 0.5, f" $R^2 = \{r\_squared:.4f\}$ ", transform=plt.gca().transAxes,
102             fontsize=12,
103             verticalalignment='top', ha='left')
104
105 # 图表设置
106 plt.xlabel("电压 (V)")
107 plt.ylabel("电流 (A)")
108 plt.title("电压-电流曲线及拟合")
109 plt.legend()
110 plt.grid(True)
111
112 # 显示图像
113 plt.show()
114
115 # 示例: 使用该函数进行分析
116 file_path = "./datas_learn/B2-3-21-30.xlsx" # 替换为你自己的文件路径
117 analyze_data(file_path, degree=3, show_equation=True, show_r_squared=True, line_style='--', line_color='r',
118             line_width=1)

```

3 迭代拟合函数 - analyze_data_with_outlier_removal

```
1 import os
2 import pandas as pd
3 import matplotlib
4 import matplotlib.pyplot as plt
5 import numpy as np
6 from sklearn.ensemble import IsolationForest
7
8 # 设置中文字体, 解决字体显示问题
9 matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 例如使用微软雅黑 (SimHei)
10 matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
11
12
13 # # 对应使用例 1
14 # def analyze_data(file_path, degree=3, show_equation=True, show_r_squared=True,
15 #                  line_style='-', line_color='b',
16 #                  line_width=1.5):
17 #     """
18 #     读取数据, 进行多项式拟合并绘制结果, 同时可以控制图表的细节和拟合的显示内容。
19 #
20 #     参数:
21 #     - file_path: Excel 文件的路径。
22 #     - degree: 多项式拟合的次数, 默认3。
23 #     - show_equation: 是否在图中显示拟合函数表达式, 默认显示。
24 #     - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
25 #     - line_style: 曲线的线型, 默认为 '-' (实线)。
26 #     - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
27 #     - line_width: 曲线的线宽, 默认为 1.5。
28 #     """
29 #
30 #     # 读取 Excel 文件
31 #     xls = pd.ExcelFile(file_path)
32 #     df = xls.parse(sheet_name=0)
33 #
34 #     # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
35 #     df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
36 #
37 #     # 重新命名列名
38 #     df_filtered.columns = ["电压 (V)", "电流 (A)"]
39 #
40 #     # 转换数据为数值类型 (防止字符串干扰)
41 #     df_filtered = df_filtered.astype(float)
42 #
43 #     # 提取自变量 (电压) 和因变量 (电流)
44 #     x = df_filtered["电压 (V)"]
45 #     y = df_filtered["电流 (A)"]
46 #
47 #     # 多项式拟合函数
48 #     def polynomial_fit(x, y, degree=3):
49 #         p = np.polyfit(x, y, degree) # 拟合多项式
50 #         poly = np.poly1d(p)
51 #         y_fit = poly(x)
52 #         return p, y_fit
53 #
54 #     # 调用多项式拟合
```

```

54 # p, y_fit = polynomial_fit(x, y, degree)
55 #
56 # # 打印拟合的多项式系数
57 # print(f"拟合多项式的系数（从高次到低次）：{p}")
58 #
59 # # 准备拟合函数的字符串表达式（优化科学计数法显示）
60 # equation_terms = []
61 # for i, coef in enumerate(p):
62 #     power = degree - i
63 #     # 将python中 "e数字" 的科学计数法格式更改为latex形式
64 #     coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{") + "}"
65 #     # 处理负号和小数点
66 #     if coef < 0:
67 #         coef_str = f"({coef_str})"
68 #     # 添加 x 的幂次
69 #     if power == 0:
70 #         term = coef_str
71 #     else:
72 #         term = f"{coef_str}x^{{{power}}}"
73 #     equation_terms.append(term) # 使用 {} 包裹幂次 防止出现x^10的情况出现 (LaTeX规范: x^{10})
74 #
75 #
76 # # 拼接拟合函数表达式
77 # equation_str = " + ".join(equation_terms)
78 #
79 # # 计算 R^2 (决定系数)
80 # ss_residual = np.sum((y - y_fit) ** 2) # 残差平方和
81 # ss_total = np.sum((y - np.mean(y)) ** 2) # 总平方和
82 # r_squared = 1 - (ss_residual / ss_total)
83 #
84 # # 打印 R^2 值
85 # print(f"决定系数: R^2 = {r_squared:.12f}") # 12位保证在拟合效果较好时能看出差距
86 #
87 # # 绘制数据和拟合曲线
88 # plt.figure(figsize=(8, 6))
89 #
90 # # 设置原始数据点的样式和连接线的样式
91 # plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据", markersize=3,
92 #          markerfacecolor='white', markeredgewidth=1, linewidth=2) # 数据点为白色,
# 线条粗细为2
93 #
94 # # 绘制拟合曲线
95 # plt.plot(x, y_fit, linestyle=line_style, color=line_color, label=f"{degree}次多项式
# 拟合", linewidth=line_width)
96 #     # 不想传参就改这里即可
97 #
98 # # 当 show_equation == True, 在图形中添加拟合函数的表达式 (使用 LaTeX 样式)
99 # if show_equation:
100 #     plt.text(0.0, 0.06, f"拟合函数: $y = {equation_str}$",
# transform=plt.gca().transAxes, fontsize=12,
101 #             verticalalignment='top', ha='left', bbox=dict(facecolor='white',
# alpha=0.8))
102 #
103 # # 当 show_r_squared == True, 在图形中添加 R^2 值, 使用 LaTeX 样式
104 # if show_r_squared:

```

```

105 #         plt.text(0.8, 0.5, f"$R^2 = {r_squared:.4f}$", transform=plt.gca().transAxes,
      fontsize=12,
106 #             verticalalignment='top', ha='left')
107 #
108 # # 图表设置
109 #     plt.xlabel("电压 (V)")
110 #     plt.ylabel("电流 (A)")
111 #     plt.title("电压-电流曲线及拟合")
112 #     plt.legend()
113 #     plt.grid(True)
114 #
115 # # 显示图像
116 #     plt.show()
117
118
119 # 对应使用例 2
120 def plot_multiple_files(file_paths, colors=None, labels=None, line_styles=None,
      line_widths=None):
121     """
122     绘制多个文件的数据到同一张图中，方便对比。
123
124     参数：
125     - file_paths: 文件路径列表，例如 ["/file1.xlsx", "/file2.xlsx"]。
126     - colors: 每个文件的曲线颜色列表，例如 ['b', 'r', 'g']。
127     - labels: 每个文件的图例标签列表，例如 ["文件1", "文件2", "文件3"]。
128     - line_styles: 每个文件的线型列表，例如 ['-', '--', ':']。
129     - line_widths: 每个文件的线宽列表，例如 [1.5, 1.5, 1.5]。
130     """
131     # 设置默认值
132     if colors is None:
133         colors = ['b', 'r', 'g', 'c', 'm', 'y', 'k'] # 默认颜色列表
134     if labels is None:
135         labels = [f"文件 {i+1}" for i in range(len(file_paths))] # 默认标签
136     if line_styles is None:
137         line_styles = ['-'] * len(file_paths) # 默认线型
138     if line_widths is None:
139         line_widths = [1.5] * len(file_paths) # 默认线宽
140
141     # 创建图表
142     plt.figure(figsize=(10, 6))
143
144     # 遍历文件路径列表
145     for i, file_path in enumerate(file_paths):
146         # 读取 Excel 文件
147         xls = pd.ExcelFile(file_path)
148         df = xls.parse(sheet_name=0)
149
150         # 提取 C35 到 D536 的数据（假设列名在第 35 行开始）
151         df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
152
153         # 重新命名列名
154         df_filtered.columns = ["电压 (V)", "电流 (A)"]
155
156         # 转换数据为数值类型（防止字符串干扰）
157         df_filtered = df_filtered.astype(float)
158

```

```

159         # 提取自变量 (电压) 和因变量 (电流)
160         x = df_filtered["电压 (V)"]
161         y = df_filtered["电流 (A)"]
162
163         # 绘制数据
164         plt.plot(x, y, linestyle=line_styles[i], color=colors[i], label=labels[i],
165 linewidth=line_widths[i])
166
167         # 图表设置
168         plt.xlabel("电压 (V)")
169         plt.ylabel("电流 (A)")
170         plt.title("电压-电流曲线对比")
171         plt.legend()
172         plt.grid(True)
173
174         # 显示图像
175         plt.show()
176
177     # # 对应使用例 3
178     # def analyze_data_with_outlier_removal(file_path, degree=3, show_equation=True,
179 show_r_squared=True, line_style='-',
180 #                                     line_color='b', line_width=1.5,
181 remove_outliers=True, threshold=3):
182     #     """
183     #     读取数据, 进行多项式拟合并绘制结果, 支持去除异常值。
184     #
185     #     参数:
186     #     - file_path: Excel 文件的路径。
187     #     - degree: 多项式拟合的次数, 默认3。
188     #     - show_equation: 是否在图中显示拟合函数表达式, 默认显示。
189     #     - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
190     #     - line_style: 曲线的线型, 默认为 '-' (实线)。
191     #     - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
192     #     - line_width: 曲线的线宽, 默认为 1.5。
193     #     - remove_outliers: 是否去除异常值, 默认 True。
194     #     - threshold: 异常值判断的阈值 (基于残差的标准差倍数), 默认 3。
195     #     """
196     #     # 读取 Excel 文件
197     #     xls = pd.ExcelFile(file_path)
198     #     df = xls.parse(sheet_name=0)
199     #
200     #     # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
201     #     df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
202     #
203     #     # 重新命名列名
204     #     df_filtered.columns = ["电压 (V)", "电流 (A)"]
205     #
206     #     # 转换数据为数值类型 (防止字符串干扰)
207     #     df_filtered = df_filtered.astype(float)
208     #
209     #     # 提取自变量 (电压) 和因变量 (电流)
210     #     x = df_filtered["电压 (V)"]
211     #     y = df_filtered["电流 (A)"]
212     #
213     #     # 第一次拟合 (用于检测异常值)
214     #     p, y_fit = np.polyfit(x, y, degree), np.poly1d(np.polyfit(x, y, degree))(x)

```

```

212 # residuals = y - y_fit # 计算残差
213 # residual_std = np.std(residuals) # 残差的标准差
214 #
215 # # 去除异常值
216 # if remove_outliers:
217 #     # 判断异常值: 残差的绝对值大于 threshold * 残差的标准差
218 #     mask = np.abs(residuals) <= threshold * residual_std
219 #     x_cleaned = x[mask]
220 #     y_cleaned = y[mask]
221 # else:
222 #     x_cleaned = x
223 #     y_cleaned = y
224 #
225 # # 第二次拟合 (使用去除异常值后的数据)
226 # p_cleaned, y_fit_cleaned = np.polyfit(x_cleaned, y_cleaned, degree),
np.poly1d(np.polyfit(x_cleaned, y_cleaned, degree))(x_cleaned)
227 #
228 # # 计算清除异常值前后的  $R^2$ 
229 # ss_residual = np.sum((y - y_fit) ** 2) # 清除前的残差平方和
230 # ss_total = np.sum((y - np.mean(y)) ** 2) # 清除前的总平方和
231 # r_squared = 1 - (ss_residual / ss_total)
232 #
233 # ss_residual_cleaned = np.sum((y_cleaned - y_fit_cleaned) ** 2) # 清除后的残差平方和
234 # ss_total_cleaned = np.sum((y_cleaned - np.mean(y_cleaned)) ** 2) # 清除后的总平方和
235 # r_squared_cleaned = 1 - (ss_residual_cleaned / ss_total_cleaned)
236 #
237 # # 打印清除异常值前后的  $R^2$ 
238 # print(f"清除异常值前的决定系数:  $R^2 = \{r\_squared:.12f\}$ ")
239 # print(f"清除异常值后的决定系数:  $R^2 = \{r\_squared\_cleaned:.12f\}$ ")
240 #
241 # # 准备拟合函数的字符串表达式 (优化科学计数法显示)
242 # def format_equation(p):
243 #     equation_terms = []
244 #     for i, coef in enumerate(p):
245 #         power = degree - i
246 #         coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{" + ") + ")
247 #         if coef < 0:
248 #             coef_str = f"({coef_str})"
249 #         if power == 0:
250 #             term = coef_str
251 #         else:
252 #             term = f"{coef_str}x^{{{power}}}"
253 #         equation_terms.append(term)
254 #     return " + ".join(equation_terms)
255 #
256 # equation_str = format_equation(p)
257 # equation_str_cleaned = format_equation(p_cleaned)
258 #
259 # # 绘制数据和拟合曲线
260 # plt.figure(figsize=(10, 6))
261 #
262 # # 绘制原始数据点
263 # plt.plot(x, y, marker='o', linestyle='', color='b', label="原始数据", markersize=3,
264 #          markerfacecolor='white', markeredgewidth=1)
265 #
266 # # 绘制清除异常值后的数据点

```

```

267 #         if remove_outliers:
268 #             plt.plot(x_cleaned, y_cleaned, marker='o', linestyle='', color='g', label="清除
异常值后的数据", markersize=3,
269 #                     markerfacecolor='green', markeredgewidth=1)
270 #
271 #         # 绘制清除前的拟合曲线
272 #         plt.plot(x, y_fit, linestyle='--', color='r', label=f"清除前 {degree}次多项式拟合",
linewidth=line_width)
273 #
274 #         # 绘制清除后的拟合曲线
275 #         plt.plot(x_cleaned, y_fit_cleaned, linestyle='-', color='m', label=f"清除后 {degree}
次多项式拟合", linewidth=line_width)
276 #
277 #         # 添加拟合函数表达式
278 #         if show_equation:
279 #             plt.text(0.05, 0.95, f"清除前拟合函数: $y = {equation_str}$",
transform=plt.gca().transAxes, fontsize=10,
280 #                     verticalalignment='top', ha='left', bbox=dict(facecolor='white',
alpha=0.8))
281 #             plt.text(0.05, 0.85, f"清除后拟合函数: $y = {equation_str_cleaned}$",
transform=plt.gca().transAxes, fontsize=10,
282 #                     verticalalignment='top', ha='left', bbox=dict(facecolor='white',
alpha=0.8))
283 #
284 #         # 添加 R2 值
285 #         if show_r_squared:
286 #             plt.text(0.8, 0.5, f"清除前 $R^2 = {r_squared:.4f}$",
transform=plt.gca().transAxes, fontsize=12,
287 #                     verticalalignment='top', ha='left')
288 #             plt.text(0.8, 0.4, f"清除后 $R^2 = {r_squared_cleaned:.4f}$",
transform=plt.gca().transAxes, fontsize=12,
289 #                     verticalalignment='top', ha='left')
290 #
291 #         # 图表设置
292 #         plt.xlabel("电压 (V)")
293 #         plt.ylabel("电流 (A)")
294 #         plt.title("电压-电流曲线及拟合 (清除异常值)")
295 #         plt.legend()
296 #         plt.grid(True)
297 #
298 #         # 显示图像
299 #         plt.show()
300
301
302 # # 对应使用例 4
303 # def analyze_data_with_outlier_removal(file_path, degree=3, show_equation=True,
show_r_squared=True, line_style='-',
304 #                                     line_color='b', line_width=1.5,
remove_outliers=True, threshold=3):
305 #     """
306 #     读取数据, 进行多项式拟合并绘制结果, 支持去除异常值。
307 #
308 #     参数:
309 #     - file_path: Excel 文件的路径。
310 #     - degree: 多项式拟合的次数, 默认3。
311 #     - show_equation: 是否在图中显示拟合函数表达式, 默认显示。

```



```

312 # - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
313 # - line_style: 曲线的线型, 默认为 '-' (实线)。
314 # - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
315 # - line_width: 曲线的线宽, 默认为 1.5。
316 # - remove_outliers: 是否去除异常值, 默认 True。
317 # - threshold: 异常值判断的阈值 (基于残差的标准差倍数), 默认 3。
318 #
319 # 读取 Excel 文件
320 # xls = pd.ExcelFile(file_path)
321 # df = xls.parse(sheet_name=0)
322 #
323 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
324 # df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
325 #
326 # 重新命名列名
327 # df_filtered.columns = ["电压 (V)", "电流 (A)"]
328 #
329 # 转换数据为数值类型 (防止字符串干扰)
330 # df_filtered = df_filtered.astype(float)
331 #
332 # 提取自变量 (电压) 和因变量 (电流)
333 # x = df_filtered["电压 (V)"]
334 # y = df_filtered["电流 (A)"]
335 #
336 # 第一次拟合 (用于检测异常值)
337 # p, y_fit = np.polyfit(x, y, degree), np.poly1d(np.polyfit(x, y, degree))(x)
338 # residuals = y - y_fit # 计算残差
339 # residual_std = np.std(residuals) # 残差的标准差
340 #
341 # 去除异常值
342 # if remove_outliers:
343 #     # 方法 1: 基于残差的绝对值
344 #     mask_residual = np.abs(residuals) <= threshold * residual_std
345 #
346 #     # 方法 2: 基于局部离群点检测 (Isolation Forest)
347 #     clf = IsolationForest(contamination=0.05) # 假设 5% 的数据是异常值
348 #     mask_isolation = clf.fit_predict(np.column_stack((x, y))) == 1
349 #
350 #     # 结合两种方法
351 #     mask = mask_residual & mask_isolation
352 #
353 #     x_cleaned = x[mask]
354 #     y_cleaned = y[mask]
355 # else:
356 #     x_cleaned = x
357 #     y_cleaned = y
358 #
359 # 第二次拟合 (使用去除异常值后的数据)
360 # p_cleaned, y_fit_cleaned = np.polyfit(x_cleaned, y_cleaned, degree),
np.poly1d(np.polyfit(x_cleaned, y_cleaned, degree))(x_cleaned)
361 #
362 # 计算清除异常值前后的  $R^2$ 
363 # ss_residual = np.sum((y - y_fit) ** 2) # 清除前的残差平方和
364 # ss_total = np.sum((y - np.mean(y)) ** 2) # 清除前的总平方和
365 # r_squared = 1 - (ss_residual / ss_total)
366 #

```

```

367 # ss_residual_cleaned = np.sum((y_cleaned - y_fit_cleaned) ** 2) # 清除后的残差平方和
368 # ss_total_cleaned = np.sum((y_cleaned - np.mean(y_cleaned)) ** 2) # 清除后的总平方和
369 # r_squared_cleaned = 1 - (ss_residual_cleaned / ss_total_cleaned)
370 #
371 # # 打印清除异常值前后的 R²
372 # print(f"清除异常值前的决定系数: R² = {r_squared:.12f}")
373 # print(f"清除异常值后的决定系数: R² = {r_squared_cleaned:.12f}")
374 #
375 # # 准备拟合函数的字符串表达式 (优化科学计数法显示)
376 # def format_equation(p):
377 #     equation_terms = []
378 #     for i, coef in enumerate(p):
379 #         power = degree - i
380 #         coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{") + "}"
381 #         if coef < 0:
382 #             coef_str = f"({coef_str})"
383 #         if power == 0:
384 #             term = coef_str
385 #         else:
386 #             term = f"{coef_str}x^{{{power}}}"
387 #         equation_terms.append(term)
388 #     return " + ".join(equation_terms)
389 #
390 # equation_str = format_equation(p)
391 # equation_str_cleaned = format_equation(p_cleaned)
392 #
393 # # 绘制数据和拟合曲线
394 # plt.figure(figsize=(10, 6))
395 #
396 # # 绘制原始数据点
397 # plt.plot(x, y, marker='o', linestyle='', color='b', label="原始数据", markersize=3,
398 #          markerfacecolor='white', markeredgewidth=1)
399 #
400 # # 绘制清除异常值后的数据点
401 # if remove_outliers:
402 #     plt.plot(x_cleaned, y_cleaned, marker='o', linestyle='', color='g', label="清除
403 # 异常值后的数据", markersize=3,
404 #             markerfacecolor='green', markeredgewidth=1)
405 #
406 # # 绘制清除前的拟合曲线
407 # plt.plot(x, y_fit, linestyle='--', color='r', label=f"清除前 {degree}次多项式拟合",
408 #          linewidth=line_width)
409 #
410 # # 绘制清除后的拟合曲线
411 # plt.plot(x_cleaned, y_fit_cleaned, linestyle='-', color='m', label=f"清除后 {degree}
412 # 次多项式拟合", linewidth=line_width)
413 #
414 # # 添加拟合函数表达式
415 # if show_equation:
416 #     plt.text(0.05, 0.95, f"清除前拟合函数: $y = {equation_str}$",
417 #              transform=plt.gca().transAxes, fontsize=10,
418 #              verticalalignment='top', ha='left', bbox=dict(facecolor='white',
419 #                  alpha=0.8))
420 #     plt.text(0.05, 0.85, f"清除后拟合函数: $y = {equation_str_cleaned}$",
421 #              transform=plt.gca().transAxes, fontsize=10,

```

```

416 #             verticalalignment='top', ha='left', bbox=dict(facecolor='white',
alpha=0.8))
417 #
418 #     # 添加  $R^2$  值
419 #     if show_r_squared:
420 #         plt.text(0.8, 0.5, f"清除前  $R^2 = \{r\_squared:.4f\}$ ",
transform=plt.gca().transAxes, fontsize=12,
421 #             verticalalignment='top', ha='left')
422 #         plt.text(0.8, 0.4, f"清除后  $R^2 = \{r\_squared\_cleaned:.4f\}$ ",
transform=plt.gca().transAxes, fontsize=12,
423 #             verticalalignment='top', ha='left')
424 #
425 #     # 图表设置
426 #     plt.xlabel("电压 (V)")
427 #     plt.ylabel("电流 (A)")
428 #     plt.title("电压-电流曲线及拟合 (清除异常值)")
429 #     plt.legend()
430 #     plt.grid(True)
431 #
432 #     # 显示图像
433 #     plt.show()
434
435 # 对应使用例 5
436 def analyze_data_with_outlier_removal(file_path, degree=3, show_equation=True,
show_r_squared=True, line_style='-',
437                                     line_color='b', line_width=1.5,
remove_outliers=True, target_r_squared=0.9,
438                                     min_threshold=0.3, initial_threshold=3):
439     """
440     读取数据，进行多项式拟合并绘制结果，支持迭代去除异常值。
441
442     参数：
443     - file_path: Excel 文件的路径。
444     - degree: 多项式拟合的次数，默认3。
445     - show_equation: 是否在图中显示拟合函数表达式，默认显示。
446     - show_r_squared: 是否显示决定系数 $R^2$ ，默认显示。
447     - line_style: 曲线的线型，默认为 '-'（实线）。
448     - line_color: 曲线的颜色，默认为 'b'（蓝色）。
449     - line_width: 曲线的线宽，默认为 1.5。
450     - remove_outliers: 是否去除异常值，默认 True。
451     - target_r_squared: 目标决定系数  $R^2$ ，默认 0.9。
452     - min_threshold: threshold 的最小值，默认 0.3。
453     - initial_threshold: threshold 的初始值，默认 3。
454     """
455     # 读取 Excel 文件
456     xls = pd.ExcelFile(file_path)
457     df = xls.parse(sheet_name=0)
458
459     # 提取 C35 到 D536 的数据（假设列名在第 35 行开始）
460     df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
461
462     # 重新命名列名
463     df_filtered.columns = ["电压 (V)", "电流 (A)"]
464
465     # 转换数据为数值类型（防止字符串干扰）
466     df_filtered = df_filtered.astype(float)

```

```

467
468 # 提取自变量 (电压) 和因变量 (电流)
469 x = df_filtered["电压 (V)"]
470 y = df_filtered["电流 (A)"]
471
472 # 定义异常值清除和拟合的函数
473 def fit_and_remove_outliers(x, y, degree, threshold):
474     # 第一次拟合 (用于检测异常值)
475     p, y_fit = np.polyfit(x, y, degree), np.poly1d(np.polyfit(x, y, degree))(x)
476     residuals = y - y_fit # 计算残差
477     residual_std = np.std(residuals) # 残差的标准差
478
479     # 方法 1: 基于残差的绝对值
480     mask_residual = np.abs(residuals) <= threshold * residual_std
481
482     # 方法 2: 基于局部离群点检测 (Isolation Forest)
483     clf = IsolationForest(contamination=0.05) # 假设 5% 的数据是异常值
484     mask_isolation = clf.fit_predict(np.column_stack((x, y))) == 1
485
486     # 结合两种方法
487     mask = mask_residual & mask_isolation
488
489     x_cleaned = x[mask]
490     y_cleaned = y[mask]
491
492     # 第二次拟合 (使用去除异常值后的数据)
493     p_cleaned, y_fit_cleaned = np.polyfit(x_cleaned, y_cleaned, degree),
np.poly1d(np.polyfit(x_cleaned, y_cleaned, degree))(x_cleaned)
494
495     # 计算 R²
496     ss_residual_cleaned = np.sum((y_cleaned - y_fit_cleaned) ** 2) # 清除后的残差平方
和
497     ss_total_cleaned = np.sum((y_cleaned - np.mean(y_cleaned)) ** 2) # 清除后的总平方
和
498     r_squared_cleaned = 1 - (ss_residual_cleaned / ss_total_cleaned)
499
500     return x_cleaned, y_cleaned, p_cleaned, y_fit_cleaned, r_squared_cleaned
501
502 # 初始化变量
503 threshold = initial_threshold
504 x_cleaned, y_cleaned = x, y
505 r_squared_cleaned = 0
506
507 # 迭代清除异常值
508 if remove_outliers:
509     while threshold >= min_threshold:
510         x_cleaned, y_cleaned, p_cleaned, y_fit_cleaned, r_squared_cleaned =
fit_and_remove_outliers(x_cleaned, y_cleaned, degree, threshold)
511         print(f"当前 threshold: {threshold:.2f}, R²: {r_squared_cleaned:.4f}")
512
513         # 如果达到目标 R², 停止迭代
514         if r_squared_cleaned >= target_r_squared:
515             print(f"达到目标 R²: {target_r_squared}")
516             break
517
518         # 降低 threshold

```

```

519         threshold -= 0.1
520     else:
521         print(f"未达到目标 R², 当前 R²: {r_squared_cleaned:.4f}, threshold 已降至最小值
{min_threshold}, 该方法不可行。")
522
523     # 如果没有清除异常值, 直接拟合
524     else:
525         p_cleaned, y_fit_cleaned = np.polyfit(x_cleaned, y_cleaned, degree),
np.polyval(np.polyfit(x_cleaned, y_cleaned, degree))(x_cleaned)
526         ss_residual_cleaned = np.sum((y_cleaned - y_fit_cleaned) ** 2) # 清除后的残差平方
和
527         ss_total_cleaned = np.sum((y_cleaned - np.mean(y_cleaned)) ** 2) # 清除后的总平方
和
528         r_squared_cleaned = 1 - (ss_residual_cleaned / ss_total_cleaned)
529
530     # 准备拟合函数的字符串表达式 (优化科学计数法显示)
531     def format_equation(p):
532         equation_terms = []
533         for i, coef in enumerate(p):
534             power = degree - i
535             coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{") + "}"
536             if coef < 0:
537                 coef_str = f"({coef_str})"
538             if power == 0:
539                 term = coef_str
540             else:
541                 term = f"{coef_str}x^{{{power}}}"
542             equation_terms.append(term)
543         return " + ".join(equation_terms)
544
545     equation_str_cleaned = format_equation(p_cleaned)
546
547     # 绘制数据和拟合曲线
548     plt.figure(figsize=(10, 6))
549
550     # 绘制原始数据点
551     plt.plot(x, y, marker='o', linestyle='', color='b', label="原始数据", markersize=3,
552             markerfacecolor='white', markeredgewidth=1)
553
554     # 绘制清除异常值后的数据点
555     if remove_outliers:
556         plt.plot(x_cleaned, y_cleaned, marker='o', linestyle='', color='g', label="清除异
常值后的数据", markersize=3,
557                 markerfacecolor='green', markeredgewidth=1)
558
559     # 绘制清除后的拟合曲线
560     plt.plot(x_cleaned, y_fit_cleaned, linestyle='--', color='m', label=f"清除后 {degree}次
多项式拟合", linewidth=line_width)
561
562     # 添加拟合函数表达式
563     if show_equation:
564         plt.text(0.05, 0.95, f"清除后拟合函数: $y = {equation_str_cleaned}$",
transform=plt.gca().transAxes, fontsize=10,
565                 verticalalignment='top', ha='left', bbox=dict(facecolor='white',
alpha=0.8))
566

```

```

567     # 添加 R2 值
568     if show_r_squared:
569         plt.text(0.8, 0.5, f"清除后 R2 = {r_squared_cleaned:.4f}$",
transform=plt.gca().transAxes, fontsize=12,
570                 verticalalignment='top', ha='left')
571
572     # 图表设置
573     plt.xlabel("电压 (V)")
574     plt.ylabel("电流 (A)")
575     plt.title("电压-电流曲线及拟合 (清除异常值)")
576     plt.legend()
577     plt.grid(True)
578
579     # 显示图像
580     plt.show()
581
582
583     ## 使用例 1: 使用 analyze_data 函数进行单文件作图和曲线拟合
584     ## file_path = "./datas_learn/B00.xlsx" # 替换为你自己的文件路径(一定要是.xlsx文件!)
585     ## file_path = "./datas_learn/test/B1.xlsx" # 替换为你自己的文件路径(一定要是.xlsx文件!)
586     ##
587     ## 分析file_path对应路径的文件, 用3次多项式进行拟合, 拟合多项式表达式和决定系数均展示, 拟合
    曲线的线型为虚线, 红色, 宽度为1
588     # analyze_data(file_path, degree=3, show_equation=False, show_r_squared=True,
    line_style='--', line_color='r',
589     #                 line_width=1)
590
591
592
593     ## 使用例 2: 使用 plot_multiple_files 进行多文件对比
594     # file_paths = [
595     #     "./datas_learn/compare1/B0.xlsx",
596     #     "./datas_learn/compare1/B0.5.xlsx",
597     #     # "./datas_learn/compare1/B1.xlsx",
598     #
599     #     # "./datas_learn/B0.xlsx",
600     #     # "./datas_learn/B0.5.xlsx",
601     #     # "./datas_learn/B1.xlsx",
602     # ]
603     #
604     # colors = ['b', 'r', 'g'] # 每个文件的曲线颜色
605     # labels = ["B0", "B0.5", "B1"] # 每个文件的图例标签
606     # line_styles = ['-', '--', ':'] # 每个文件的线型
607     # line_widths = [1.5, 1.5, 1.5] # 每个文件的线宽
608     #
609     # colors = ['b', 'r'] # 每个文件的曲线颜色
610     # labels = ["B0", "B0.5"] # 每个文件的图例标签
611     # line_styles = ['-', '--'] # 每个文件的线型
612     # line_widths = [1.5, 1.5] # 每个文件的线宽
613     #
614     # plot_multiple_files(file_paths, colors=colors, labels=labels, line_styles=line_styles,
    line_widths=line_widths)
615
616
617     ## 使用例 3: 使用 analyze_data_with_outlier_removal 进行分析

```

```

618 # file_path = "./datas_learn/test2/T1-5-12-21(slightly boom,900V one time).xlsx" # 替换为
    你自己的文件路径
619 # analyze_data_with_outlier_removal(file_path, degree=3, show_equation=True,
    show_r_squared=True,
620 #                                     line_style='--', line_color='r', line_width=1,
    remove_outliers=True, threshold=1)
621
622 ## 使用例 4: 使用 analyze_data_with_outlier_removal 进行分析
623 # file_path = "./datas_learn/B00.xlsx" # 替换为你自己的文件路径
624 # analyze_data_with_outlier_removal(file_path, degree=3, show_equation=True,
    show_r_squared=True,
625 #                                     line_style='--', line_color='r', line_width=1,
    remove_outliers=True, threshold=0.5)
626
627 # 使用例 5: 使用 analyze_data_with_outlier_removal 进行分析
628 file_path = "./datas_learn/B00.xlsx" # 替换为你自己的文件路径
629 analyze_data_with_outlier_removal(file_path, degree=3, show_equation=False,
    show_r_squared=True,
630 #                                     line_style='--', line_color='r', line_width=1,
    remove_outliers=True,
631 #                                     target_r_squared=0.9, min_threshold=1,
    initial_threshold=3)
632

```

4 增加多文件画图函数 - plot_multiple_files

```

1  import pandas as pd
2  import matplotlib
3  import matplotlib.pyplot as plt
4  import numpy as np
5
6  # 设置中文字体, 解决字体显示问题
7  matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 例如使用微软雅黑 (SimHei)
8  matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
9
10 def analyze_data(file_path, degree=3, show_equation=True, show_r_squared=True,
    line_style='-', line_color='b',
11                  line_width=1.5):
12     """
13     读取数据, 进行多项式拟合并绘制结果, 同时可以控制图表的细节和拟合的显示内容。
14
15     参数:
16     - file_path: Excel 文件的路径。
17     - degree: 多项式拟合的次数, 默认3。
18     - show_equation: 是否在图中显示拟合函数表达式, 默认显示。
19     - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
20     - line_style: 曲线的线型, 默认为 '-' (实线)。
21     - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
22     - line_width: 曲线的线宽, 默认为 1.5。
23     """
24
25     # 读取 Excel 文件
26     xls = pd.ExcelFile(file_path)
27     df = xls.parse(sheet_name=0)

```



```

28
29 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
30 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
31
32 # 重新命名列名
33 df_filtered.columns = ["电压 (V)", "电流 (A)"]
34
35 # 转换数据为数值类型 (防止字符串干扰)
36 df_filtered = df_filtered.astype(float)
37
38 # 提取自变量 (电压) 和因变量 (电流)
39 x = df_filtered["电压 (V)"]
40 y = df_filtered["电流 (A)"]
41
42 # 多项式拟合函数
43 def polynomial_fit(x, y, degree=3):
44     p = np.polyfit(x, y, degree) # 拟合多项式
45     poly = np.poly1d(p)
46     y_fit = poly(x)
47     return p, y_fit
48
49 # 调用多项式拟合
50 p, y_fit = polynomial_fit(x, y, degree)
51
52 # 打印拟合的多项式系数
53 print(f"拟合多项式的系数 (从高次到低次) : {p}")
54
55 # 准备拟合函数的字符串表达式 (优化科学计数法显示)
56 equation_terms = []
57 for i, coef in enumerate(p):
58     power = degree - i
59     # 将python中 "e数字" 的科学计数法格式更改为latex形式
60     coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{" + ")")
61     # 处理负号和小数点
62     if coef < 0:
63         coef_str = f"({coef_str})"
64     # 添加 x 的幂次
65     if power == 0:
66         term = coef_str
67     else:
68         term = f"{coef_str}x^{{{power}}}"
69     equation_terms.append(term) # 使用 {} 包裹幂次 防止出现x^10的情况出现 (LaTeX规范:x^{10})
70
71
72 # 拼接拟合函数表达式
73 equation_str = " + ".join(equation_terms)
74
75 # 计算 R2 (决定系数)
76 ss_residual = np.sum((y - y_fit) ** 2) # 残差平方和
77 ss_total = np.sum((y - np.mean(y)) ** 2) # 总平方和
78 r_squared = 1 - (ss_residual / ss_total)
79
80 # 打印 R2 值
81 print(f"决定系数: R^2 = {r_squared:.12f}") # 12位保证在拟合效果较好时能看出差距
82

```



```

83     # 绘制数据和拟合曲线
84     plt.figure(figsize=(8, 6))
85
86     # 设置原始数据点的样式和连接线的样式
87     plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据", markersize=3,
88             markerfacecolor='white', markeredgewidth=1, linewidth=2) # 数据点为白色, 线
条粗细为2
89
90     # 绘制拟合曲线
91     plt.plot(x, y_fit, linestyle=line_style, color=line_color, label=f"{degree}次多项式拟
合", linewidth=line_width)
92     # 不想传参就改这里即可
93
94     # 当 show_equation == True, 在图形中添加拟合函数的表达式 (使用 LaTeX 样式)
95     if show_equation:
96         plt.text(0.0, 0.06, f"拟合函数: $y = {equation_str}$",
transform=plt.gca().transAxes, fontsize=12,
97         verticalalignment='top', ha='left', bbox=dict(facecolor='white',
alpha=0.8))
98
99     # 当 show_r_squared == True, 在图形中添加 R2 值, 使用 LaTeX 样式
100    if show_r_squared:
101        plt.text(0.8, 0.5, f"$R^2 = {r_squared:.4f}$", transform=plt.gca().transAxes,
fontsize=12,
102        verticalalignment='top', ha='left')
103
104    # 图表设置
105    plt.xlabel("电压 (V)")
106    plt.ylabel("电流 (A)")
107    plt.title("电压-电流曲线及拟合")
108    plt.legend()
109    plt.grid(True)
110
111    # 显示图像
112    plt.show()
113
114
115    def plot_multiple_files(file_paths, colors=None, labels=None, line_styles=None,
line_widths=None):
116        """
117        绘制多个文件的数据到同一张图中, 方便对比。
118
119        参数:
120        - file_paths: 文件路径列表, 例如 ["/file1.xlsx", "/file2.xlsx"]。
121        - colors: 每个文件的曲线颜色列表, 例如 ['b', 'r', 'g']。
122        - labels: 每个文件的图例标签列表, 例如 ["文件1", "文件2", "文件3"]。
123        - line_styles: 每个文件的线型列表, 例如 ['-', '--', ':']。
124        - line_widths: 每个文件的线宽列表, 例如 [1.5, 1.5, 1.5]。
125        """
126        # 设置默认值
127        if colors is None:
128            colors = ['b', 'r', 'g', 'c', 'm', 'y', 'k'] # 默认颜色列表
129        if labels is None:
130            labels = [f"文件 {i+1}" for i in range(len(file_paths))] # 默认标签
131        if line_styles is None:
132            line_styles = ['-'] * len(file_paths) # 默认线型

```

```

133     if line_widths is None:
134         line_widths = [1.5] * len(file_paths) # 默认线宽
135
136     # 创建图表
137     plt.figure(figsize=(10, 6))
138
139     # 遍历文件路径列表
140     for i, file_path in enumerate(file_paths):
141         # 读取 Excel 文件
142         xls = pd.ExcelFile(file_path)
143         df = xls.parse(sheet_name=0)
144
145         # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
146         df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
147
148         # 重新命名列名
149         df_filtered.columns = ["电压 (V)", "电流 (A)"]
150
151         # 转换数据为数值类型 (防止字符串干扰)
152         df_filtered = df_filtered.astype(float)
153
154         # 提取自变量 (电压) 和因变量 (电流)
155         x = df_filtered["电压 (V)"]
156         y = df_filtered["电流 (A)"]
157
158         # 绘制数据
159         plt.plot(x, y, linestyle=line_styles[i], color=colors[i], label=labels[i],
160 linewidth=line_widths[i])
161
162     # 图表设置
163     plt.xlabel("电压 (V)")
164     plt.ylabel("电流 (A)")
165     plt.title("电压-电流曲线对比")
166     plt.legend()
167     plt.grid(True)
168
169     # 显示图像
170     plt.show()
171
172 # 使用例 1: 使用 analyze_data 函数进行单文件作图和曲线拟合
173 # file_path = "./datas_learn/B00.xlsx" # 替换为你自己的文件路径(一定要是.xlsx文件!)
174 file_path = "./datas_learn/compare1/B1.xlsx" # 替换为你自己的文件路径(一定要是.xlsx文件!)
175
176 # 使用例 1: 分析file_path对应路径的文件, 用3次多项式进行拟合, 拟合多项式表达式和决定系数均展
177 # 示, 拟合曲线的线型为虚线, 红色, 宽度为1
178 analyze_data(file_path, degree=3, show_equation=False, show_r_squared=True, line_style='--', line_color='r',
179 line_width=1)
180
181 # 使用例 2: 使用 plot_multiple_files 进行多文件对比
182 file_paths = [
183     "./datas_learn/compare1/B0.xlsx",
184     "./datas_learn/compare1/B0.5.xlsx",
185     # "./datas_learn/compare1/B1.xlsx",

```

```

186     # "./datas_learn/B0.xlsx",
187     # "./datas_learn/B0.5.xlsx",
188     # "./datas_learn/B1.xlsx",
189 ]
190
191 # colors = ['b', 'r', 'g'] # 每个文件的曲线颜色
192 # labels = ["B0", "B0.5", "B1"] # 每个文件的图例标签
193 # line_styles = ['-', '--', ':'] # 每个文件的线型
194 # line_widths = [1.5, 1.5, 1.5] # 每个文件的线宽
195
196 colors = ['b', 'r'] # 每个文件的曲线颜色
197 labels = ["B0", "B0.5"] # 每个文件的图例标签
198 line_styles = ['-', '--'] # 每个文件的线型
199 line_widths = [1.5, 1.5] # 每个文件的线宽
200
201 plot_multiple_files(file_paths, colors=colors, labels=labels, line_styles=line_styles,
202                     line_widths=line_widths)

```

批量处理时代

1 传文件夹，批量画所有表格的图.(batch_analyze_data.py)

```

1  import os
2  import pandas as pd
3  import numpy as np
4  import matplotlib
5  import matplotlib.pyplot as plt
6
7
8  # 设置中文字体，解决字体显示问题
9  matplotlib.rcParams['font.sans-serif'] = ['Microsoft YaHei'] # 例如使用微软雅黑 (SimHei)
10 matplotlib.rcParams['axes.unicode_minus'] = False # 解决负号显示问题
11
12 def analyze_data(file_path, degree=3, show_equation=True, show_r_squared=True,
13                 line_style='--', line_color='r',
14                 line_width=1.5, connect_points=True):
15     """
16     读取数据，进行多项式拟合并绘制结果，同时可以控制图表的细节和拟合的显示内容。
17
18     参数：
19     - file_path: Excel 文件的路径。
20     - degree: 多项式拟合的次数，默认3。
21     - show_equation: 是否在图中显示拟合函数表达式，默认显示。
22     - show_r_squared: 是否显示决定系数 $R^2$ ，默认显示。
23     - line_style: 曲线的线型，默认为 '--' (虚线)。
24     - line_color: 曲线的颜色，默认为 'r' (红色)。
25     - line_width: 曲线的线宽，默认为 1.5。
26     - connect_points: 是否连接原始数据点，默认连接。
27     """
28
29     # 读取 Excel 文件
30     xls = pd.ExcelFile(file_path)
31     df = xls.parse(sheet_name=0)

```

```

32 # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
33 df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
34
35 # 重新命名列名
36 df_filtered.columns = ["电压 (V)", "电流 (A)"]
37
38 # 转换数据为数值类型 (防止字符串干扰)
39 df_filtered = df_filtered.astype(float)
40
41 # 提取自变量 (电压) 和因变量 (电流)
42 x = df_filtered["电压 (V)"]
43 y = df_filtered["电流 (A)"]
44
45 # 多项式拟合函数
46 def polynomial_fit(x, y, degree=3):
47     p = np.polyfit(x, y, degree) # 拟合多项式
48     poly = np.poly1d(p)
49     y_fit = poly(x)
50     return p, y_fit
51
52 # 调用多项式拟合
53 p, y_fit = polynomial_fit(x, y, degree)
54
55 # 打印拟合的多项式系数
56 print(f"拟合多项式的系数 (从高次到低次) : {p}")
57
58 # 准备拟合函数的字符串表达式 (优化科学计数法显示)
59 equation_terms = []
60 for i, coef in enumerate(p):
61     power = degree - i
62     # 将python中 "e数字" 的科学计数法格式更改为latex形式
63     coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{") + "}"
64     # 处理负号和小数点
65     if coef < 0:
66         coef_str = f"({coef_str})"
67     # 添加 x 的幂次
68     if power == 0:
69         term = coef_str
70     else:
71         term = f"{coef_str}x^{{{power}}}"
72     equation_terms.append(term) # 使用 {} 包裹幂次 防止出现x^10的情况出现 (LaTeX规范: x^{10})
73
74 # 拼接拟合函数表达式
75 equation_str = " + ".join(equation_terms)
76
77 # 计算 R² (决定系数)
78 ss_residual = np.sum((y - y_fit) ** 2) # 残差平方和
79 ss_total = np.sum((y - np.mean(y)) ** 2) # 总平方和
80 r_squared = 1 - (ss_residual / ss_total)
81
82 # 打印 R² 值
83 print(f"决定系数: R^2 = {r_squared:.12f}") # 12位保证在拟合效果较好时能看出差距
84
85 # 绘制数据和拟合曲线
86 plt.figure(figsize=(8, 6))

```

```

87
88     # 设置原始数据点的样式和连接线的样式
89     if connect_points:
90         plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据",
91 markersize=3,
92                 markerfacecolor='white', markeredgewidth=1, linewidth=2) # 数据点为白
93     else:
94         plt.plot(x, y, marker='o', linestyle='', color='b', label="原始数据",
95 markersize=3,
96                 markerfacecolor='white', markeredgewidth=1) # 数据点为白色, 不连接
97
98     # 绘制拟合曲线
99     plt.plot(x, y_fit, linestyle=line_style, color=line_color, label=f"{degree}次多项式拟
100 合", linewidth=line_width)
101
102     # 当 show_equation == True, 在图形中添加拟合函数的表达式 (使用 LaTeX 样式)
103     if show_equation:
104         plt.text(0.0, 0.06, f"拟合函数:  $y = \{equation\_str\}$ ",
105 transform=plt.gca().transAxes, fontsize=12,
106                 verticalalignment='top', ha='left', bbox=dict(facecolor='white',
107 alpha=0.8))
108
109     # 当 show_r_squared == True, 在图形中添加  $R^2$  值, 使用 LaTeX 样式
110     if show_r_squared:
111         plt.text(0.8, 0.5, f" $R^2 = \{r\_squared:.4f\}$ ", transform=plt.gca().transAxes,
112 fontsize=12,
113                 verticalalignment='top', ha='left')
114
115     # 图表设置
116     plt.xlabel("电压 (V)")
117     plt.ylabel("电流 (A)")
118     plt.title("电压-电流曲线及拟合")
119     plt.legend()
120     plt.grid(True)
121
122     # 显示图像
123     plt.show()
124
125 def analyze_data_no_display(file_path, degree=3, show_equation=True, show_r_squared=True,
126 line_style='-', line_color='b',
127                             line_width=1.5, connect_points=True):
128     """
129     与 analyze_data 功能相同, 但不显示图像。
130     """
131     # 读取 Excel 文件
132     xls = pd.ExcelFile(file_path)
133     df = xls.parse(sheet_name=0)
134
135     # 提取 C35 到 D536 的数据 (假设列名在第 35 行开始)
136     df_filtered = df.iloc[34:536, [2, 3]].dropna() # 选择 C 和 D 列 (0-based 索引)
137
138     # 重新命名列名
139     df_filtered.columns = ["电压 (V)", "电流 (A)"]
140
141     # 转换数据为数值类型 (防止字符串干扰)

```

```

135 df_filtered = df_filtered.astype(float)
136
137 # 提取自变量 (电压) 和因变量 (电流)
138 x = df_filtered["电压 (V)"]
139 y = df_filtered["电流 (A)"]
140
141 # 多项式拟合函数
142 def polynomial_fit(x, y, degree=3):
143     p = np.polyfit(x, y, degree) # 拟合多项式
144     poly = np.poly1d(p)
145     y_fit = poly(x)
146     return p, y_fit
147
148 # 调用多项式拟合
149 p, y_fit = polynomial_fit(x, y, degree)
150
151 # 准备拟合函数的字符串表达式 (优化科学计数法显示)
152 equation_terms = []
153 for i, coef in enumerate(p):
154     power = degree - i
155     # 将python中 "e数字" 的科学计数法格式更改为latex形式
156     coef_str = f"{coef:.2e}".replace("e", "\\cdot 10^{" + ")")
157     # 处理负号和小数点
158     if coef < 0:
159         coef_str = f"({coef_str})"
160     # 添加 x 的幂次
161     if power == 0:
162         term = coef_str
163     else:
164         term = f"{coef_str}x^{{{power}}}"
165     equation_terms.append(term) # 使用 {} 包裹幂次 防止出现x^10的情况出现 (LaTeX规
范: x^{-10})
166
167 # 拼接拟合函数表达式
168 equation_str = " + ".join(equation_terms)
169
170 # 计算 R^2 (决定系数)
171 ss_residual = np.sum((y - y_fit) ** 2) # 残差平方和
172 ss_total = np.sum((y - np.mean(y)) ** 2) # 总平方和
173 r_squared = 1 - (ss_residual / ss_total)
174
175 # 绘制数据和拟合曲线
176 plt.figure(figsize=(8, 6))
177
178 # 设置原始数据点的样式和连接线的样式
179 if connect_points:
180     plt.plot(x, y, marker='o', linestyle='-', color='b', label="原始数据",
markersize=3,
181             markerfacecolor='white', markeredgewidth=1, linewidth=2) # 数据点为白
色, 线条粗细为2
182 else:
183     plt.plot(x, y, marker='o', linestyle='', color='b', label="原始数据",
markersize=3,
184             markerfacecolor='white', markeredgewidth=1) # 数据点为白色, 不连接
185
186 # 绘制拟合曲线

```

```

187     plt.plot(x, y_fit, linestyle=line_style, color=line_color, label=f"{degree}次多项式拟
    合", linewidth=line_width)
188
189     # 当 show_equation == True, 在图形中添加拟合函数的表达式 (使用 LaTeX 样式)
190     if show_equation:
191         plt.text(0.0, 0.06, f"拟合函数:  $y = \{equation\_str\}$ ",
transform=plt.gca().transAxes, fontsize=12,
192                 verticalalignment='top', ha='left', bbox=dict(facecolor='white',
alpha=0.8))
193
194     # 当 show_r_squared == True, 在图形中添加  $R^2$  值, 使用 LaTeX 样式
195     if show_r_squared:
196         plt.text(0.8, 0.5, f" $R^2 = \{r\_squared:.4f\}$ ", transform=plt.gca().transAxes,
fontsize=12,
197                 verticalalignment='top', ha='left')
198
199     # 图表设置
200     plt.xlabel("电压 (V)")
201     plt.ylabel("电流 (A)")
202     plt.title("电压-电流曲线及拟合")
203     plt.legend()
204     plt.grid(True)
205
206     # 返回图像对象
207     return plt.gcf()
208
209 def batch_analyze_data(folder_path, output_folder, degree=3, show_equation=True,
show_r_squared=True, line_style='-', line_color='b',
210                        line_width=1.5, connect_points=False):
211     """
212     批量处理文件夹中的 Excel 文件, 进行多项式拟合并保存图像。
213
214     参数:
215     - folder_path: 包含 Excel 文件的文件夹路径。
216     - output_folder: 保存图像的文件夹路径。
217     - degree: 多项式拟合的次数, 默认3。
218     - show_equation: 是否在图中显示拟合函数表达式, 默认显示。
219     - show_r_squared: 是否显示决定系数 $R^2$ , 默认显示。
220     - line_style: 曲线的线型, 默认为 '-' (实线)。
221     - line_color: 曲线的颜色, 默认为 'b' (蓝色)。
222     - line_width: 曲线的线宽, 默认为 1.5。
223     - connect_points: 是否连接原始数据点, 默认不连接。
224     """
225     # 创建输出文件夹
226     if not os.path.exists(output_folder):
227         os.makedirs(output_folder)
228
229     # 遍历文件夹中的所有 Excel 文件
230     for file_name in os.listdir(folder_path):
231         if file_name.endswith('.xlsx') or file_name.endswith('.xls'):
232             file_path = os.path.join(folder_path, file_name)
233             # 调用 analyze_data_no_display 函数生成图像
234             fig = analyze_data_no_display(file_path, degree, show_equation,
show_r_squared, line_style, line_color,
235                                         line_width, connect_points)
236             # 保存图像

```

```
237         output_file_path = os.path.join(output_folder, f"{os.path.splitext(file_name)
[0]}.png")
238         fig.savefig(output_file_path, dpi=240)
239         plt.close(fig)
240         print(f"已保存图像: {output_file_path}")
241         print('分析完成.')
242
243     # 示例调用
244     # batch_analyze_data('./datas', './datas/datas_img')
245
246     batch_analyze_data('./datas_learn/old_34', './datas_learn/old_34/old34_img',
line_style='--', line_color='r', connect_points=False)
247
248
```