Python 程序设计

第五次作业：简易客户端应用

2021211306 班 | 杜抒泽 2021211110

2023 年 11 月 17 日

目录

1 作业题目

基于 #3 作业、#4 作业获取的 No_Smoothing、Lowess 数据项，在同一个图上分别绘制出折线图（No_Smoothing）和平滑线图（Lowess）。绘制结果对照参考图片（test.png）。

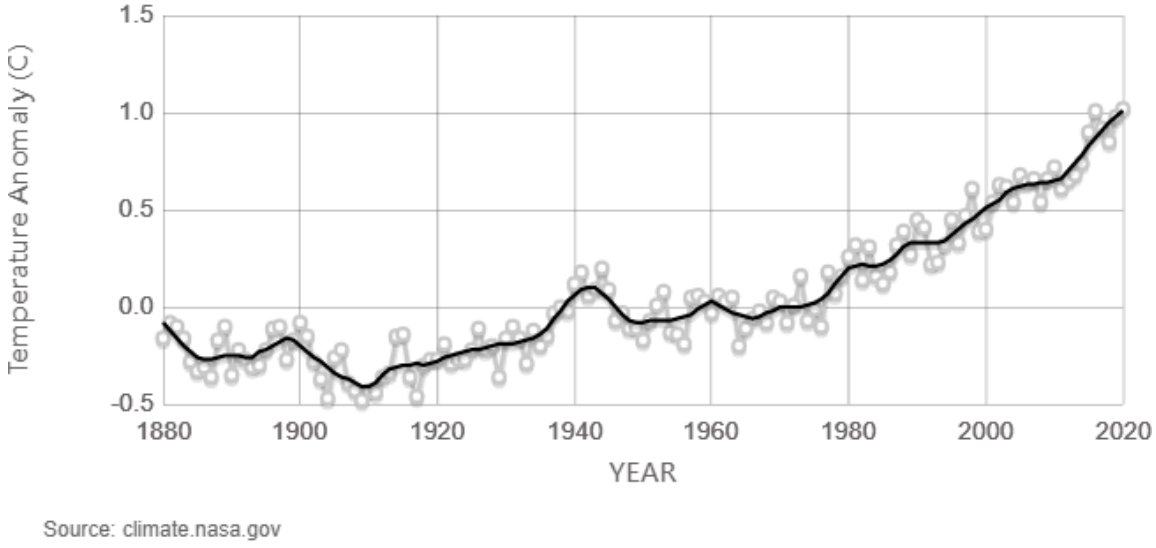

图 1　参考图片 test.png

2 作业内容

哎呀，这个就很尴尬了。我上一次作业预判了这一次作业的内容，用 `matplotlib` 绘制 Lowess 的需求已经完成了。这次把上次的代码直接拉过来改了改，加了个保存图片到文件的 GUI 接口，姑且也算是工作量够了吧。

程序源代码嵌入下方的 code block 中。

```
#! /usr/bin/env python3
# -*- coding: utf-8 -*-


from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure
from os import getenv
from tkinter.filedialog import askopenfilename, asksaveasfilename
from typing import List, Optional

import matplotlib.pyplot as plt
import numpy as np
import sys
import tkinter as tk
```

```python
def my_lowess(
    data_x: np.ndarray, data_y: np.ndarray, window_size: int
) -> np.ndarray:
    def w(x: float) -> float:
        return (1 - np.abs(x) ** 3) ** 3 if np.abs(x) <= 1 else 0

    def weighted_linear_regression(
        x: np.ndarray, y: np.ndarray, w: np.ndarray
    ) -> np.ndarray:
        w_sum = np.sum(w)
        x_mean = np.sum(x * w) / w_sum if w_sum != 0 else 0
        y_mean = np.sum(y * w) / w_sum if w_sum != 0 else 0
        x_diff: np.ndarray = x - x_mean
        y_diff: np.ndarray = y - y_mean
        b = np.sum(w * (x_diff * y_diff)) / np.sum(w * x_diff ** 2)
        a = y_mean - b * x_mean
        return a + b * x

    result: np.ndarray = np.zeros(len(data_y))
    for i in range(len(data_y)):
        start: int = i - window_size
        end: int = i + window_size + 1
        if start < 0:
            end -= start
            start = 0
        if end > len(data_y):
            start -= end - len(data_y)
            end = len(data_y)
        start = max(start, 0)
        end = min(end, len(data_y))

        window: np.ndarray = np.arange(start, end)
        x_local: np.ndarray = data_x[window]
        y_local: np.ndarray = data_y[window]
        result[i] = weighted_linear_regression(
            x_local - i,
            y_local,
            np.array([w(x) for x in (x_local - i) / max(abs(x_local - i))]),
        )[np.where(x_local == i)[0][0]]
    return result


def lowess(data_y: np.ndarray, windows_size: int) -> np.ndarray:
```

```
    if getenv('SM_LOWESS'):
        import statsmodels.api as sm
        return sm.nonparametric.lowess(
            data_y,
            np.arange(len(data_y)),
            frac=(2 * windows_size) / len(data_y),
        ).T[1]

    return my_lowess(np.arange(len(data_y)), data_y, windows_size)


class WeatherDataEntry:
    def __init__(
        self, year: int, temperature: float, temperature_smoothed: float
    ) -> None:
        self.year: int = year
        self.temperature: float = temperature
        self.temperature_smoothed: float = temperature_smoothed

    @classmethod
    def try_from(
        cls, year: str, temperature: str, temperature_smoothed: str
    ) -> 'WeatherDataEntry':
        try:
            return cls(
                int(year), float(temperature), float(temperature_smoothed)
            )
        except ValueError as e:
            raise Exception(f'Invalid weather data entry {e}')

    def __str__(self) -> str:
        return f'{self.year} {self.temperature} {self.temperature_smoothed}'


class WeatherData:
    def __init__(self, data: Optional[List[WeatherDataEntry]] = None) -> None:
        self.data: List[WeatherDataEntry] = data if data else []

    def append(self, entry: WeatherDataEntry) -> None:
        self.data.append(entry)

    def __str__(self) -> str:
        return 'year temperature temperature_smoothed\n' + \
            '\n'.join(str(entry) for entry in self.data)
```

```
    def get_year(self) -> List[int]:
        return [entry.year for entry in self.data]

    def get_temperature(self) -> List[float]:
        return [entry.temperature for entry in self.data]

    def get_temperature_smoothed(self) -> List[float]:
        return [entry.temperature_smoothed for entry in self.data]


class App(tk.Tk):
    def __init__(self) -> None:
        super().__init__()
        self.title('Simple Weather Data Analysis')
        self.create_widgets()

    def create_widgets(self) -> None:
        self.read_from_file_button = tk.Button(
            self, text='从文件读取', command=self.read_from_file
        )
        self.read_from_file_button.grid(row=0, column=0, pady=5, sticky='E')

        self.status_label = tk.Label(self, text='未加载数据')
        self.status_label.grid(row=0, column=1, pady=5, sticky='W')

        self.save_to_file_button = tk.Button(
            self, text='保存到文件', command=self.save_to_file
        )
        self.save_to_file_button.grid(row=0, column=2, pady=5, sticky='E')

        self.save_status_label = tk.Label(self, text='', width=30, anchor='w')
        self.save_status_label.grid(row=0, column=3, pady=5, sticky='W')
        self.empty_save_status_label_handle = None

        self.plot = FigureCanvasTkAgg(
            Figure(figsize=(13, 8), dpi=120), master=self)
        self.plot.get_tk_widget().grid(
            row=1, column=0, columnspan=4, padx=5, pady=5, sticky='NSEW'
        )

    @staticmethod
    def validate_input(lines: List[str]) -> None:
        if len(lines) < 5:
```

```
            raise Exception('数据条数过少')
        for line in lines:
            if len(line.split()) != 3:
                raise Exception('文件格式错误')
        years = [int(line.split()[0]) for line in lines]
        for i in range(len(years) - 1):
            if years[i + 1] - years[i] != 1:
                raise Exception('年份不连续递增')

    def read_from_file(self) -> None:
        file_path: str = askopenfilename(
            title='选择文件', filetypes=[('文本文件', '*.txt')]
        )
        if not file_path:
            return
        with open(file_path, 'r') as f:
            lines = f.readlines()[1:]
            self.file_name = file_path.split('/')[-1]

            try:
                self.validate_input(lines)
            except Exception as e:
                self.status_label.config(
                    text='{} 加载失败：{}'.format(self.file_name, e)
                )
                return

            self.status_label.config(text='已加载 {}'.format(self.file_name))
            self.create_plot(WeatherData(
                [WeatherDataEntry.try_from(*line.split()) for line in lines]
            ))

    def save_to_file(self) -> None:
        if self.status_label['text'] == '未加载数据':
            self.save_status_label.config(text='请先加载数据')
            if self.empty_save_status_label_handle:
                self.after_cancel(self.empty_save_status_label_handle)
            self.empty_save_status_label_handle = self.after(
                5000, self.empty_save_status_label
            )
            return
        file_path: str = asksaveasfilename(
            title='保存图像', filetypes=[('PNG Image', '*.png')],
```

```
            defaultextension='.png',
            initialfile=f'{".".join(self.file_name.split(".")[:-1])}.png',
        )
        if not file_path:
            return
        file_name = file_path.split('/')[-1]
        self.plot.figure.savefig(file_path, format='png')
        print(f'Saved to {file_name}')
        self.save_status_label.config(text=f'已保存到 {file_name}')
        if self.empty_save_status_label_handle:
            self.after_cancel(self.empty_save_status_label_handle)
        self.empty_save_status_label_handle = self.after(
            5000, self.empty_save_status_label
        )

    def empty_save_status_label(self) -> None:
        self.save_status_label.config(text='')

    def create_plot(self, data: WeatherData):
        years: List[int] = data.get_year()
        temperatures: List[float] = data.get_temperature()
        std_lowess: np.ndarray = np.array(data.get_temperature_smoothed())
        my_lowess: np.ndarray = lowess(np.array(temperatures), 5)
        diff: np.ndarray = std_lowess - my_lowess

        self.plot.figure.clear()

        ax = self.plot.figure.add_subplot()
        ax.plot(
            years, temperatures, marker=".", markersize=2, linewidth=0.5,
            label='原始数据', color='lightgrey'
        )
        ax.plot(
            years, std_lowess, marker="o", markersize=2, label='样例 Lowess(5)'
        )
        ax.plot(
            years, my_lowess, marker="o", markersize=2,
            label='{} Lowess(5)'.format(
                'statsmodels' if getenv('SM_LOWESS') else '手动实现'
            )
        )
        ax.plot(
            years, diff, marker="o", markersize=1, linestyle='', label='差值'
```

```
        )

        width = max(years) - min(years)
        step = 20 if width > 100 else 10 if width > 50 else 5
        ax.set_xticks(range(
            ((min(years) - 1) // step + 1) * step,
            max(years) // step * step + 1,
            step,
        ))
        ax.set_xlabel('年份')
        ax.set_ylabel('温度')
        ax.set_title('年份-温度图')
        ax.grid(True)
        ax.legend()

        self.plot.figure.tight_layout()
        self.plot.draw()


def main() -> None:
    if getenv('SM_LOWESS'):
        sys.stderr.write('Using statsmodels `lowess`\n')
    else:
        sys.stderr.write('Using manually implemented `lowess`\n')

    plt.rcParams['font.sans-serif'] = ['HYZhengYuan']  # 在其他环境中可能需要修改字
体
    plt.rcParams['axes.unicode_minus'] = False
    app = App()
    app.mainloop()


if __name__ == '__main__':
    main()
```

3 代码说明

程序的大多数部分都已在之前的实验报告中介绍过，因此本次的实验只侧重于 GUI 的修改与绘图组件。

3.1 程序需求分析

从第三次作业实现的客户端中获取数据，调用第四次作业实现的算法进行数据处理，再使用 `matplotlib` 库将处理后的数据与原始数据在同一张图上绘制出来，用以分析数据的平滑效果。

具体地，实现以下功能：

- 采取 GUI 形式，提供用户界面。
- 提供第四次作业中的所有功能，包括 Lowess 算法。
- 用折线图和散点图图形化地展示原始数据、提供的 lowess(5) 结果和计算得到的 lowess(5) 结果及其差值。

3.2 程序基本介绍与运行方式

这段代码延续第四次作业的简易的天气数据处理 GUI 客户端（暂定名为 Simple Weather Data Analysis），将对属于采用 lowess 算法进行平滑处理的结果图形化地展示，并与提供的原始数据进行比较。

程序风格方面，由于作业要求，代码实现为了单文件，没有以文件的形式分离各个模块。与前三次作业一样，这次作业仍然严格遵守 Python 的 PEP8 编码规范。

开发环境与用到的第三方库如下表：

软件环境或第三方库名	**版本**
操作系统 / macOS	13.0 22A380 arm64
Python	3.10.6
`numpy`	1.26.2
`matplotlib`	3.8.2
`statsmodels`	0.14.0

在安装 Python 后，可以使用以下命令安装依赖与运行本程序（`4th.py`）：

```
$ python3 -m pip install numpy matplotlib statsmodels
$ python3 5th.py
```

3.3 程序组件介绍

数据类与算法已分别在第二次与第四次作业中介绍，这里不再赘述。

3.3.1 GUI 界面与应用

程序的 GUI 界面托管于 tkinter 库，由继承自 `tkinter.Tk` 的 `App` 类实现。绘图功能托管于 matplotlib 库，由其提供的 tkinter 适配器 `FigureCanvasTkAgg` 实现。

GUI 布局十分简单：

从文件读取按钮的点击事件会调用读取文件的 API，运行算法，更新绘图区域，并更新数据加载状态。保存到文件按钮的点击时间会调用保存文件的 API，将生成的图片保存到文件，更新保存状态（保存状态维持 5s 后消失）。绘图区域会显示原始数据、提供的 lowess(5) 结果和计算得到的 lowess(5) 结果及其差值。

3.3.2 绘图组件

绘图组件托管于 matplotlib 库，由其提供的 `FigureCanvasTkAgg` 实现。绘图组件的绘图逻辑在 `app.create_plot` 函数中实现。其代码如下，包含获取数据，绘制数据，绘制标题、标签、图例、网格、调整布局等步骤，具体逻辑在注释中介绍：

```
def create_plot(self, data: WeatherData):
    years = ... # 年份作为图像的 x 轴

    # 以下数据均是关于年份的函数
    temperatures = ... # 原始数据
    std_lowess = ... # 提供的样例 lowess(5) 结果
    my_lowess = ... # 计算得到的 lowess(5) 结果
    diff = ... # 两个 lowess(5) 结果的差值

    # 清空原有的图像
    self.plot.figure.clear()

    # 绘制原始数据、提供的 lowess(5) 结果和计算得到的 lowess(5) 结果及其差值
    ax = self.plot.figure.add_subplot()
    ax.plot(
        years, temperatures, marker=".", markersize=2, linewidth=0.5,
        label='原始数据', color='lightgrey'
```

```
    )
    ax.plot(
        years, std_lowess, marker="o", markersize=2, label='样例 Lowess(5)'
    )
    ax.plot(
        years, my_lowess, marker="o", markersize=2,
        label='{} Lowess(5)'.format(
            'statsmodels' if getenv('SM_LOWESS') else '手动实现'
        )
    )
    ax.plot(
        years, diff, marker="o", markersize=1, linestyle='', label='差值'
    )

    # 设置图像的 x 轴范围和刻度
    width = max(years) - min(years)
    step = 20 if width > 100 else 10 if width > 50 else 5
    ax.set_xticks(range(
        ((min(years) - 1) // step + 1) * step,
        max(years) // step * step + 1,
        step,
    ))

    ax.set_xlabel('年份') # 设置 x 轴标签
    ax.set_ylabel('温度') # 设置 y 轴标签
    ax.set_title('年份-温度图') # 设置图像标题
    ax.grid(True) # 显示网格
    ax.legend() # 显示图例

    self.plot.figure.tight_layout() # 调整图像布局更紧凑
    self.plot.draw() # 绘制图像
```

3.4 程序运行结果截图与分析

以下给出程序在两份数据下的运行指引和效果截图。

打开程序后主界面如下：

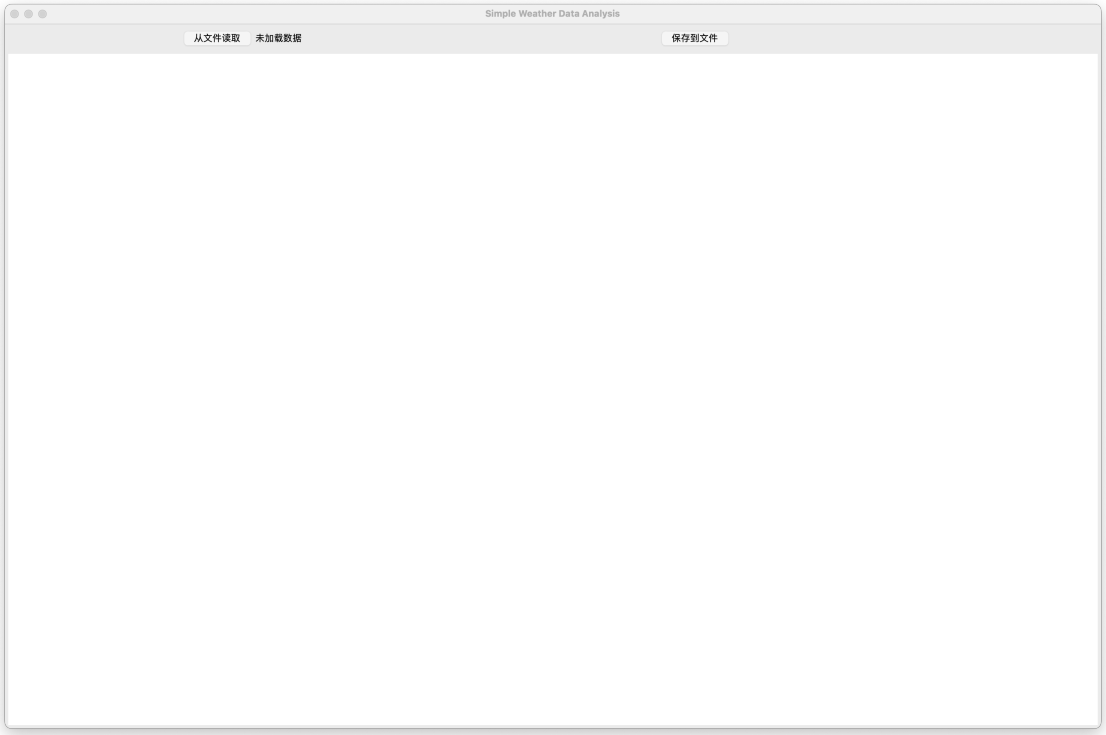

图 2　主界面（相当空荡）

加载 `graph.txt` 中全部数据（1880 年到 2022 年），程序显示如下。其中浅灰色折线图是原始数据，蓝色折线图是提供的样例 lowess(5) 结果，橙色折线图是计算得到的 lowess(5) 结果，绿色散点图是两个 lowess(5) 结果的差值。

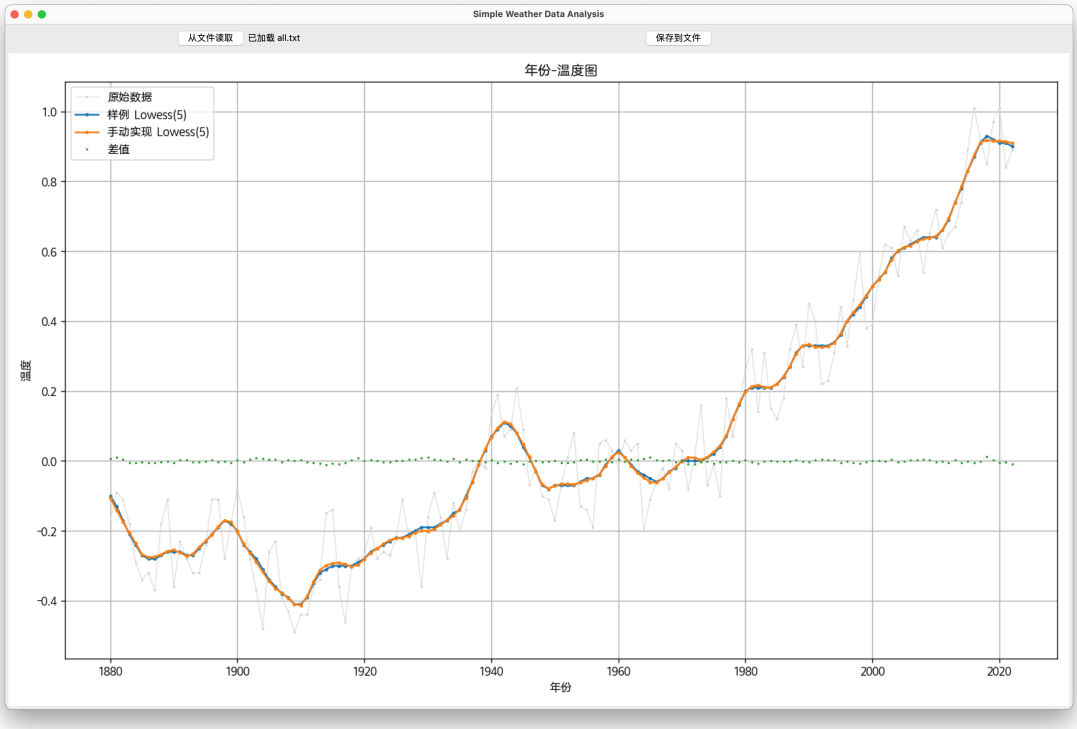

图 3　1880~2022 年数据绘图

点击保存到文件按钮，调用系统 API 弹窗如下。文件名缺省为与打开的文件同名，用户可以选择保存的位置。

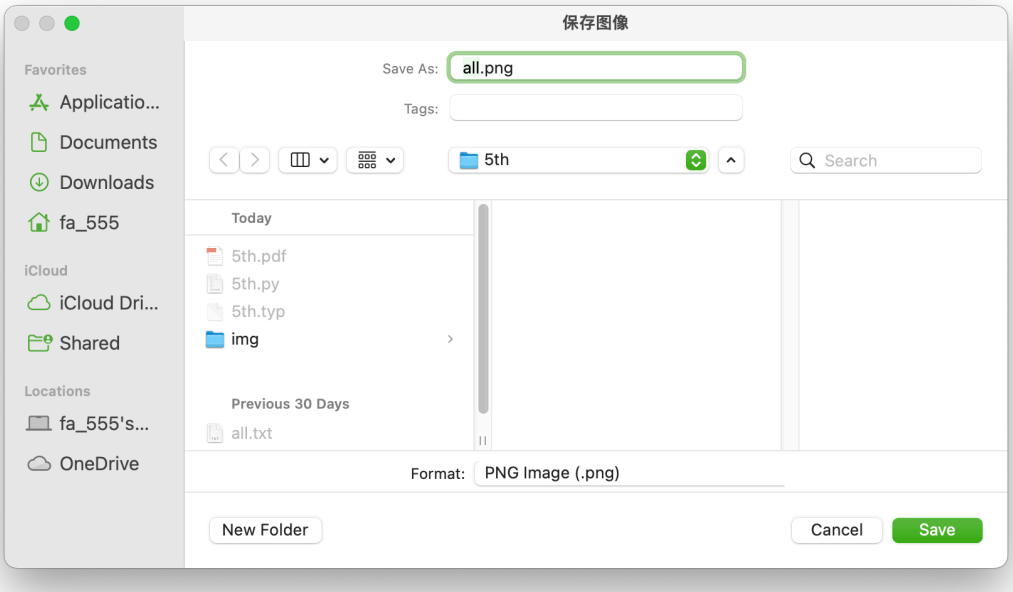

图 4　保存到文件

之后，程序会在保存状态栏显示保存成功的消息，5s 后消失。以下是保存的图像：

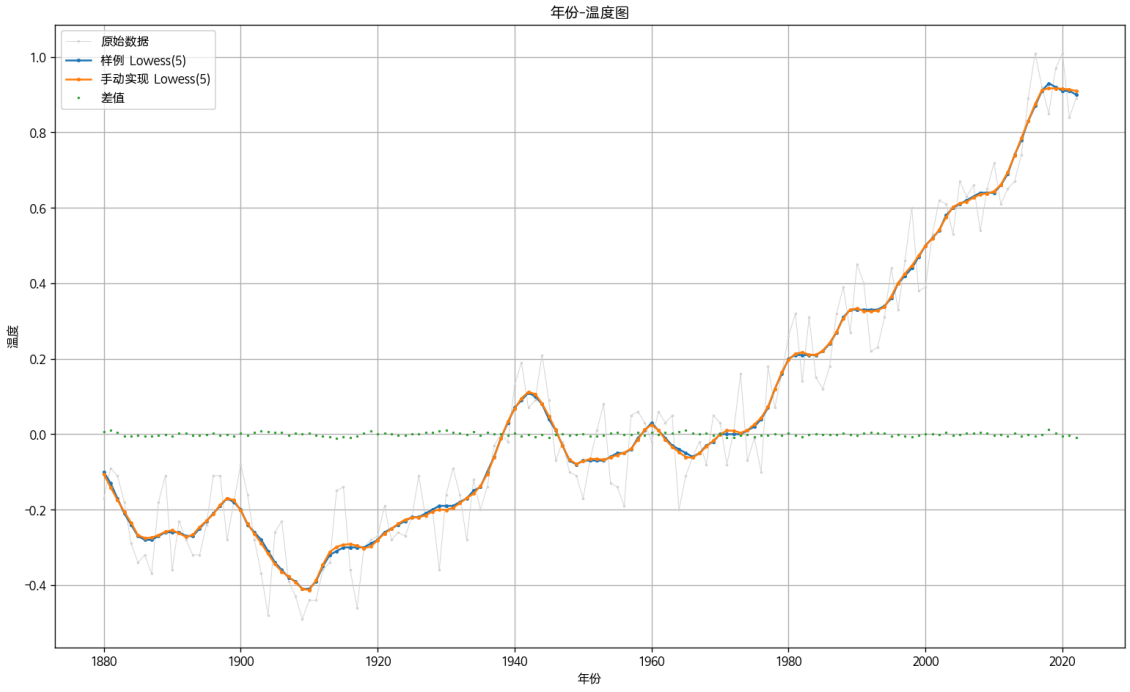

图 5　1880~2022 年数据绘图

类似地，我们将 `graph.txt` 中 1949 年到 2009 年的数据生成并保存图像，如下图：

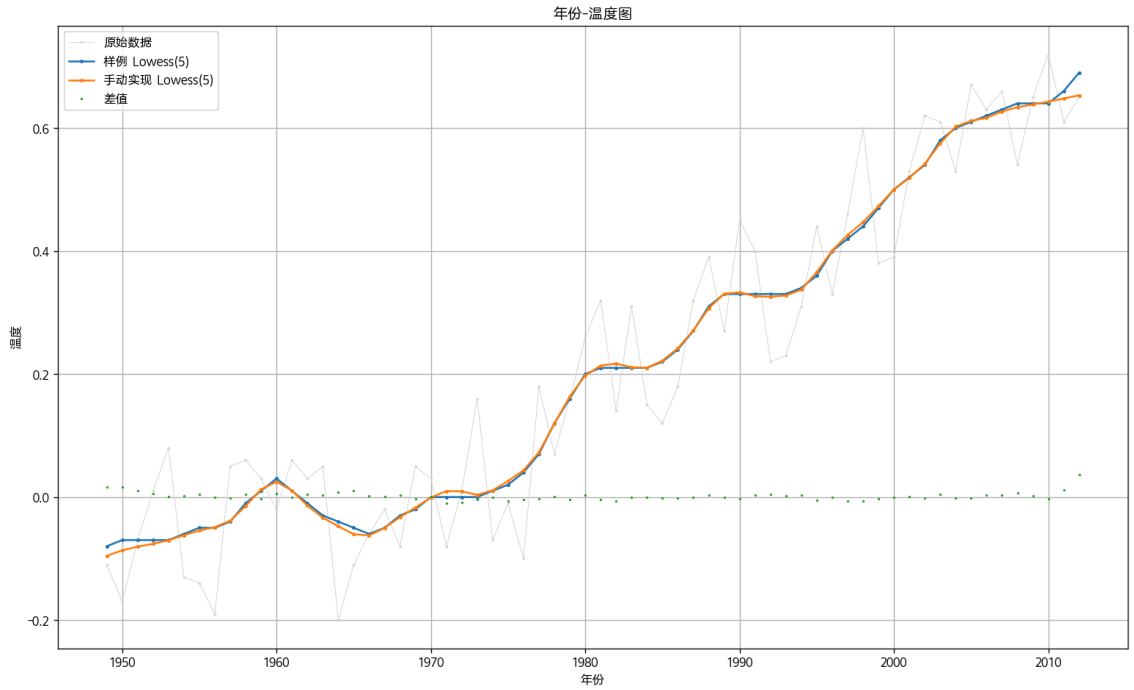

图 6　1949~2009 年数据绘图

可以看到，程序运行正常，绘图结果与预期一致，保存功能也正常可用。

3.5 总结

在这次作业中，我在第三次喝第四次作业的基础上，完善了数据绘图（可视化）功能，并追加了保存图像到文件的功能，大量使用了 `matplotlib` 库的各种 API，实现了一个简易的数据分析 GUI 客户端。

3.6 附录 - 附件清单

```
.
├── 5th.py
├── all.png
└── slice.png
```