《地下水动力学》实验指导书

杨国勇　王长申　董贵明 刘 博 编

资源与地球科学学院水文与水资源系

中国矿业大学

2022 年 11 月

目 录

预备知识：Windows 下的 Python 科学计算环境搭建

Python 是一种面向对象、解释型计算机程序设计语言，其语法简洁而清晰，易于上手。尤其是具有丰富和强大的类库，常被昵称为胶水语言，能够把用其他语言制作的各种模块轻松地联结在一起。

Python 非常适合做科学计算，相应的科学计算软件包齐全，如 NumPy，SciPy，Matplotlib 等。

要使用 Python 进行各种开发和科学计算，需要安装对应的软件包。软件包单独安装起来非常费事，尤其是 SciPy。如果有一款集成的安装环境，将各种包打包发行，对终端用户非常便利。

Anaconda 就是这样一个 Python 环境，集成了常用的科学计算和数据科学软件包。对于新用户，强烈建议安装 Anaconda。也有其他的 Python 科学计算发行版，如 Python(x, y)、WinPython、Enthought Canopy 等，建议用户具有一定经验后再尝试。经验丰富的用户也可自行搭建满足自己需求的 Python 程序开发环境。

按《地下水动力学》课程的实验要求，本指导书将简要介绍 Python 程序设计中可能用到的相关工具包。

1. Anaconda 个人版

Anaconda 是用于科学计算的 Python 发行版，支持 Linux, Mac, Windows 系统。其中的工具 Conda 可管理包（package）与环境（environment），可以方便地解决多版本并存、切换以及各种第三方包的安装。

Anaconda 个人版（免费）的 下载地址：

https://www.anaconda.com/products/distribution

清华大学开源软件镜像站提供了 Anaconda 镜像：

https://mirrors.tuna.tsinghua.edu.cn/help/anaconda

执行安装程序正常安装即可，不建议在环境变量中写入安装路径，这样会影响已安装的 Python 其他版本的使用。本指导书的 Python 程序，用到了 IPython、NumPy、SciPy、Matplotlib、Pandas、Jupyter、及 JupyterLab 包，这些都是 Anaconda 的默认安装选项。课程用到但 Anaconda 没有提供的包的安装与使用方法，将单独介绍。

Anaconda 的 Conda 用于管理 Python 软件包，而且更关注于数据科学。Conda 可以理解为一个工具，也是一个可执行命令。常用命令格式如下：

- 安装一个包：`conda install package_name`
- 移除一个包：`conda remove package_name`
- 升级包版本：`conda update package_name`

- 查看所有的包：`conda list`

更多的内容看查看在线文档。

2. 搭建自己的 Python 编程环境

用户可以根据需要定制自己的 Python 编程环境。

Python 的 Window 发行版可从 https://www.python.org/downloads/windows/ 下载。可根据自己需要，下载 32-bit、64-bit 版本以及帮助文件。

一般地，由于 SciPy 库支持的 Python 版本号较最新的 Python 版本号要低，建议不要选最新版的 Python。本课程使用的 Python 版本号为 3.9.7。

安装程序提供了一些选项，用户可根据自己需要修改默认这些选项。

- 在指定文件夹中创建 Python 虚拟环境

  用 Win 键 + R 输入 cmd 打开命令提示符窗口，切换到要保存 Python 虚拟环境的位置（子目录）。例如，要将 Python 虚拟环境保存到 D 盘的 Python39 子目录下，在命令提示符窗口行键入如下命令：

  D:（回车）
  python -m venv Python39（回车）

  系统在 D 盘的根目录下建立名称为 Python39 的子目录，该目录及其子目录包含了虚拟环境需要的文件。

- 激活虚拟环境

  运行批处理文件：

  python39\scripts\activate.bat

  退出 Python 虚拟环境后重新进入，需要再次运行 activate.bat。

  若想更方便的进入虚拟环境，建议在桌面建立命令提示符快捷方式，打开快捷方式的属性页，将“目标”改为

  %windir%\system32\cmd.exe /K D:\python39\scripts\activate.bat

  将“起始位置”改为

  d:\python39

- 使用 Pip 工具在虚拟环境中安装所需的程序包

  Pip 是一个 Python 内置的包管理系统。Pip 可以安装、更新或删除任何正式的包。

Pip 通过访问 Python 包索引服务器 PyPI（https://pypi.org/）实现版本控制，PyPI 服务器上存储了近 200，000 个项目以及之前发布的所有此类项目。

因为 PyPI 的主服务器在国外，国内用户使用 Pip 安装 Python 包的时候，经常会出现超时错误，访问起来非常不便，使用国内镜像站点可避免此类错误。

以下是常用的 PyPI 国内镜像：

清华：https://pypi.tuna.tsinghua.edu.cn/simple

阿里云：http://mirrors.aliyun.com/pypi/simple/

中国科技大学 https://pypi.mirrors.ustc.edu.cn/simple/

华中理工大学：http://pypi.hustunique.com/

山东理工大学：http://pypi.sdutlinux.org/

豆瓣：http://pypi.douban.com/simple/

如下为 Pip 常用的命令：

升级 pip ：pip install --upgrade pip 或 python -m pip install --upgrade pip

查看版本：pip --version

安装包： pip install some-package

查询包：pip search some-package

卸载包：pip uninstall some-package

升级包：pip install some-package --upgrade 或 pip install -U some-package

列出已经安装的包：pip list

查看需要升级的包：pip list -o

查看包版本：pip freeze

切换 Pip 镜像：

pip install -i https://pypi.tuna.tsinghua.edu.cn/simple some-package (临时使用)
pip config set global.index-url https://pypi.tuna.tsinghua.edu.cn/simple(设为默认)

本课程用到的包可用如下命令一起安装：

python -m pip install numpy scipy matplotlib pandas jupyter jupyterlab ipympl

或

```
pip install numpy scipy matplotlib pandas jupyter jupyterlab ipympl
```

- 软件包简介：

  IPython：是一个非常流行的 Python 解释器，也是利用 Python 进行科学计算和交互可视化的一个最佳的平台。

  NumPy：使用 Python 进行科学计算的基础软件包，最重要的一个特点是其 N 维数组对象 ndarray，能够快速高效的处理多维数组。

  SciPy：是一个开源的 Python 科学计算库，其中涵盖了科学计算中的各种工具，包括统计、积分、插值、最优化，图像处理等。

  Matplotlib：是一个 Python 的 2D 绘图库，可以绘制线图、散点图、等高线图、条形图、柱状图、3D 图形甚至是图形动画等。

  Pandas：一个强大的分析结构化数据的工具集，它以 NumPy 为基础，可用于数据挖掘和数据分析，同时也提供数据清洗功能。Pandas 中的 Series 是一种类似于一维数组的对象。DataFrame 是 Pandas 中的一个表格型的数据结构，包含有一组有序的列，每列可以是不同的值类型（数值、字符串、布尔型等），DataFrame 既有行索引也有列索引，可以被看作是由 Series 组成的字典。

3. Jupyter Notebook

Jupyter Notebook 是基于网页的用于交互计算的应用，应用于全过程计算：开发、文档编写、运行代码和展示结果。Jupyter Notebook 是以网页的形式打开，可以在网页页面中直接编写和运行代码，代码的运行结果也会直接在代码块下方显示。编程中若需要编写说明文档，可在同一个页面中直接编写，便于作及时的说明和解释。本课程的实验是以 Notebook (`.ipynb`) 形式提供程序文档。

Jupyter Notebook 含两个组件：

- Web 应用程序：基于浏览器的应用，结合了编写说明文档、数学公式、交互计算和其他富媒体形式的工具。

- Notebook 文档：即保存所有交互计算、说明文档、数学公式、图片以及其他富媒体形式的输入和输出的文档。文档为后缀为 .ipynb 的 JSON 格式文件，方便与他人共享。此外，文档还可以导出为 HTML、LaTeX、PDF 等格式。

(1) 安装

Anaconda 发行版自动为你安装了 Jupyter Notebook。也可在终端（Linux 或 macOS 的“终端”，Windows 的 “Anaconda Prompt”）执行命令安装：

```
conda install jupyter notebook
```

(2) 启动 Jupyter Notebook 服务

在终端中输入如下命令启动 Jupyter Notebook 服务：

```
jupyter notebook
```

执行命令后终端中显示一系列 Jupyter Notebook 服务的信息，同时浏览器将会自动加载 Notebook 仪表板，仪表板会列出当前目录的 Notebook、其他类型文件及子目录的列表。

多数情况下，需要在包含 Notebook 文件的上一级目录中启动 Jupyter Notebook，这样可以快捷使用当前目录及子目录下的 Notebook 文件。Jupyter Notebook 通过终端与本地服务器建立链接，一旦关闭，将无法在 Notebook 中进行操作。

(3) 查看帮助

```
jupyter notebook --help
```

或

```
jupyter notebook -h
```

Unofficial Jupyter Notebook Extensions：

(https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/)为 Jupyter Notebook 的扩展工具包，提供许多便捷功能，如果需要请看相关文档。

4. JupyterLab

JupyterLab 是基于 Web 用户界面的 Jupyter 项目。JupyterLab 使用户能够以灵活、集成和可扩展的方式处理文档和行为。JupyterLab 由同一台服务器提供服务，使用与经典 Jupyter Notebook 相同的笔记本文档格式。Anaconda 发行版默认安装 JupyterLab。

(1) 安装

使用 `conda` 安装 JupyterLab:

```
conda install -c conda-forge jupyterlab
```

使用 `pip` 安装 JupyterLab:

```
pip install jupyterlab
```

如果使用（`pip install --user`）进行安装，则必须将用户级 `bin` 目录添加到 `PATH` 环境变量。

(2) 启动 JupyterLab

```
jupyter lab
```

或

```
jupyter-lab
```

Jupyter Lab 在浏览器中自动打开。由于 Jupyter Lab 是对 Jupyter Notebook 服务的扩展，因此可通过 Jupyter 启动 JupyterLab。

5. IPyWidgets

Widgets（小部件）是事件驱动的 Python 对象，通常作为控件（如滑块，文本框等）使用。使用 Widgets 可为 Notebook 构建交互式的 GUI，如配线法就需要该功能。Widgets 还可以在 Python 和 JavaScript 之间同步 stateful 与 stateless 信息。IPyWidgets 是可为 Jupyter Notebook 提供交互功能的 Widgets。若安装了 Jupyter，则 IPyWidgets 也被默认安装。

(1) 安装

使用 `Pip` 安装 IPyWidgets：

```
pip install ipywidgets
jupyter nbextension enable --py widgetsnbextension
```

使用 virtual env 并在激活的虚拟环境中工作时，可能需要 `--sys-prefix` 选项才能启用扩展并保持环境隔离：

```
jupyter nbextension enable --py widgetsnbextension --sys-prefix
```

使用 Conda 安装 IPyWidgets：

```
conda install -c conda-forge ipywidgets
```

(2) 安装 JupyterLab Extension

JupyterLab 扩展需要先安装 Node.js，命令为

```
conda install -c conda-forge nodejs
```

安装 labextension：

```
jupyter labextension install @jupyter-widgets/jupyterlab-manager
```

此命令默认为安装最新版本的 IPyWidgets JupyterLab 扩展。如果在 JupyterLab 运行时安装，则需要刷新页面或重新启动 JupyterLab 才能使更改生效。关于 Jupyter Notebook、JupyterLab、IPyWidgets 的使用，帮助菜单中有相关链接。

6. ipympl

默认情况下，Matplotlib 在 Jupyter 环境即便开启了交互绘图模式，但交互功能有限。ipympl 包（https://github.com/matplotlib/ipympl）是为 Jupyter 环境开发提供交互式后端，增强了 Jupyter 环境下的交互式绘图功能。

交互绘图图形提供如下功能：

- 显示图形标题；

- 有工具栏，可以使用交互工具；
- 图形大小可调整；
- 其他单元格的绘图语句会持续作用于交互绘图图形。

ipympl 的安装方法如下：

```
conda install -c conda-forge ipympl
```

或

```
pip install ipympl
```

要使用 ipympl 后端，只需使用 Jupyter 魔法命令：

```
%matplotlib widget
```

mpl_interactions（https://github.com/ianhi/mpl-interactions）是另一个在 Jupyter 中实现交互式绘图功能的工具，用户可根据喜好自己安装尝试。

实验一、单井稳定流求参

1. 影响半径经验公式

吉哈尔特（承压水）：

$$R = 10 s_w \sqrt{K}$$

库萨金（潜水）：

$$R = 2 s_w \sqrt{K H_0}$$

式中，K — 渗透系数，m/d；s — 设计降深，m；H_0 — 自底板算起的含水层静止水位（厚度），m。

只有单孔抽水数据时可构造迭代公式计算 K 及 R：

承压井：

$$\begin{cases} K &= \dfrac{Q}{2\pi M s_w} \ln \dfrac{R}{r_w} \\ R &= 10 s_w \sqrt{K} \end{cases}$$

潜水井：

$$\begin{cases} K &= \dfrac{Q}{\pi (2H_0 - s_w) s_w} \ln \dfrac{R}{r_w} \\ R &= 2 s_w \sqrt{K H_0} \end{cases}$$

2. 程序设计

用 Excel 表计算

因为迭代是收敛的，用 Excel 表可以快速得出结果：

- 先假设初始参数 K_0, R_0 为一对大于 0 的值，在 Excel 中设两列分别代表 K 和 R；
- 第 1 行 K 值先输入大于零的值，R 值输入自定义公式，其中 K 引用同行的 K 值单元格；
- 第 2 行的 K 值输入自定义公式，其中 R 引用第 1 行的 R 值单元格；
- 选定含公式的单元格向下填充，可以看到经过有限次运算，结果趋于稳定。

Python 程序

推荐使用 NumPy 库。NumPy 是一个开放源代码的 Python 库，在科学和工程学的每个领域中都有应用，为 Python 的通用标准库。 NumPy API 已在 Pandas，SciPy，Matplotlib，scikit-learn，scikit-image 等软件包中广泛使用。NumPy 库包含多维数组和矩阵数据结构，其中的

ndarray 为齐次 n 维数组对象。NumPy 可以对数组和矩阵进行高效计算，并且提供了庞大的高级数学函数库。

NumPy 进行数组运算非常有效，其中的 numpy.vectorize 将自定义的函数向量化，可以接受向量参数，并以向量返回结果，处理向量非常方便；numpy.array 将 Python 列表转化为 NumPy 数组；numpy.ones、numpy.zeros 用于构造元素为 1 或 0 的数组。常用的数学函数在 NumPy 中都有对应的形式，如 numpy.sin 、numpy.abs 、numpy.exp 等 。

NumPy 库使用前需要导入：

```
import numpy as np
# 设置浮点数显示方式
np.set_printoptions(precision=4)
```

以承压水影响半径的经验公式为例，自定义一个子程序 empirical，通过循环迭代，当两次计算结果误差的绝对值小于误差限时终止计算。

```
def empirical(rw, M, Q, sw):
    K0 = 10
    R0 = 10*sw*np.sqrt(K0)
    while True:
        R = 10*sw*np.sqrt(K0)
        K = 0.5*Q*np.log(R/rw)/M/sw/np.pi
        if np.abs(R - R0) < 1.0e-8 and np.abs(K - K0) < 1.0e-6:
            break
        else:
            K0 = K
            R0 = R
    return K, R
```

计算一次抽水试验数据：

```
rw = 0.4    # 井半径
M = 16.5    # 含水层厚度
Q = 320.54  # 抽水量
s = 1.16    # 降深值

K, R = empirical(rw, M, Q, s)

print('  K = {:.4f} m^2/d'.format(K))
print('  R = {:.4f} m'.format(R))
```

输出：

```
    K = 12.3221 m^2/d
    R = 40.7192 m
```

计算多次抽水试验数据，向量化的函数非常方便：

```
vempirical= np.vectorize(empirical) # 将 empirical 函数向量化
rw = np.ones(3)*0.4  # 井半径向量
M = np.ones(3)*16.5  # 含水层厚度向量
Q = np.array([320.54, 421.63, 536.54])  # 抽水量向量
s = np.array([1.16, 1.60, 1.90])        # 降深值向量

K, R = vempirical(rw, M, Q, s)

print('  K = {} m^2/d'.format(K))
print('  R = {} m'.format(R))
```

输出：

```
    K = [12.3221 12.5963 14.1221] m^2/d
    R = [40.7192 56.786  71.4009] m
```

练习

某潜水含水层完整井稳定流抽水试验，已知潜水面最初水位高度 43.6m，抽水孔半径 0.15m，降深 2.8m，稳定抽水量 2380m^3/d。求潜水含水层渗透系数 K 及影响半径 R（精度要求 1e-4，参考答案 22m/d）。

思考题

上述迭代求参方法的适用条件包括哪些？

实验二、最小二乘法确定 $Q \sim s$ 曲线类型

1. 最小二乘原理

根据观测数据 $(x_i, y_i), i = 1,2,\cdots,n$，求拟合直线 $y = \alpha + \beta x$，使 $E(\alpha,\beta) = \sum_{i=1}^{n}(y_i - \alpha - \beta x_i)^2$ 取最小值。令 $\frac{\partial E}{\partial \alpha} = 0, \frac{\partial E}{\partial \beta} = 0$，

$$\begin{cases} \sum_{i=1}^{n}(\alpha + \beta x_i - y_i) & = 0 \\ \sum_{i=1}^{n}(\alpha + \beta x_i - y_i)\, x_i & = 0 \end{cases}$$

写成矩阵形式

$$AX = b$$

式中

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, X = \begin{bmatrix} \alpha \\ \beta \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

其中，$a_{11} = n, a_{12} = a_{21} = \sum_{i=1}^{n} x_i$，$a_{22} = \sum_{i=1}^{n} x_i^2$，$b_1 = \sum_{i=1}^{n} y_i$，$b_2 = \sum_{i=1}^{n} x_i\, y_i$

解得

$$\beta = \frac{a_{11}b_2 - a_{12}b_1}{a_{11}a_{22} - a_{12}^2}, \alpha = \frac{b_1 - a_{12}\beta}{a_{11}}$$

一般形式

考虑超定方程组（超定指未知数小于方程个数）

$$\sum_{j=1}^{n} x_{ij}\,\beta_j = y_i, \quad i = 1,2,3,\cdots,m$$

其中，m 代表样本数，n代表参数维度，写成矩阵形式

$$X\beta = Y$$

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}, \quad \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_n \end{bmatrix}, \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

为得到 β 的最佳估计 $\hat{\beta}$，将问题转化为如下的最值问题：

$$\min E(\beta) = \min \|X\beta - Y\|$$

通过微分求最值，得

$$X^TX\beta = X^TY$$

若 X^TX 为非奇异矩阵，则 β 有唯一解。

$$\hat{\beta} = (X^TX)^{-1}X^TY$$

可以看出，求解最小二乘问题的关键是构造方程组。numpy 库中的 numpy.linalg.solve 可用于求解形如 $AX = b$ 的方程组，其中的 numpy.linalg.lstsq 就是最小二乘法。scipy 库的 scipy.linalg.lstsq 也是最小二乘法。

例

一次模拟实验中，输入 t（自变量）为 (0.00, 0.30, 0.60, 0.90, 1.08, 1.20, 1.30, 1.48, 1.60, 1.70, 1.78, 1.85, 1.90, 1.95, 2.00)，观测到的输出 y（因变量）为(1.78, 1.91, 2.01, 2.12, 2.20, 2.22, 2.25, 2.32, 2.38, 2.41, 2.43, 2.47, 2.49, 2.48, 2.51)，根据实验分析，y 与 t 成线性关系，试确定关系表达式。

以下为 Python 程序，其中 numpy.array.T 将矩阵转置，matplotlib 库的 matplotlib.pyplot 模块用于绘制图形。

```
%matplotlib widget

# 导入 numpy 与 matplotlib 库
import numpy as np
import matplotlib.pyplot as plt

np.set_printoptions(precision=4)
# 控制小数的显示精度

plt.rcParams['font.sans-serif']=['KaiTi']
# 指定用来显示中文的字体，fc-list 命令可查看可用字体
plt.rcParams['axes.unicode_minus']=False

# 将原始数据转化为 numpy 的 array 数组
t = np.array([0.00, 0.30, 0.60, 0.90, 1.08, 1.20, 1.30, 1.48, 1.60, 1.70, 1.78, 1.8
5, 1.90, 1.95, 2.00])
y = np.array([1.78, 1.91, 2.01, 2.12, 2.20, 2.22, 2.25, 2.32, 2.38, 2.41, 2.43, 2.4
7, 2.49, 2.48, 2.51])

# 形成方程组
X = np.array([np.ones(len(t)), t]).T
A = np.dot(X.T, X)
b = np.dot(X.T, y)
```

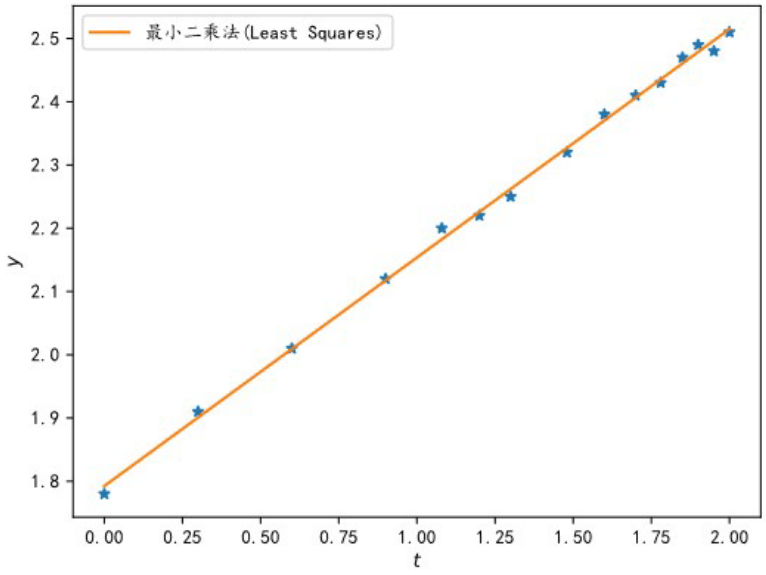
按公式计算：

```
beta = np.zeros(2)

# 求解方程组
beta[1] = (A[0,0]*b[1]-A[0,1]*b[0])/(A[0,0]*A[1,1]-A[0,1]*A[0,1])
beta[0] = (b[0]-A[0,1]*beta[1])/A[0,0]
```

显示计算结果并绘制图形

```
# 显示计算结果与误差
print("beta =", beta, "residuals =", "{:.4f}".format(sum((beta[0]+beta[1]*t-y)**2)))

# 绘制图形
plt.plot(t, y, "*")
plt.plot(t, beta[0]+beta[1]*t, "-", label="最小二乘法(Least Squares)")
plt.xlabel("$t$")
plt.ylabel("$y$")
plt.legend()
plt.show()
```

```
beta = [1.7924 0.3612] residuals = 0.0014
```



可以将图形保存为文件

```
# 保持图形，可用格式：png, pdf, svg 等
plt.savefig("fig1-1.svg", bbox_inches ="tight")
```

也可调用 np.linalg.solve 求解：

```
# 求解方程组，beta 返回解
beta = np.linalg.solve(A, b)
```

2. 确定 $Q \sim s$ 曲线类型

程序设计思路

将常用的 $Q \sim s$ 类型变换形式：

- 直线型： $Q = bs_w$ (不变，要求通过原点）$(b > 0)$
- 抛物线型： $\frac{s_w}{Q} = a + bQ$，$(a > 0)$
- 幂函数型： $\lg Q = a + b\lg s_w$，（取 $s_w = 1$ 可知对 a 符号没有要求）
- 对数型： $Q = a + b\lg s_w$ (不变)，（取 $s_w = 1$ 可知 $a > 0$ ）

根据最小二乘法原理构造方程组 $X\beta = Y$，进一步形成方程组 $A\beta = b$，调用 `numpy.linalg.solve` 求解，其中 $A = X^TX, b = X^TY$。

如何判断那种类型是最适合的？

比较笨拙的方法是，按统一的误差标准计算误差，如 $\sum(\hat{Q}_i - Q_i)^2$ 或 $\sum(\hat{s}_i - s_i)^2$，选择误差最小的类型为最适合的类型。

准备数据

```
Q = np.array([320.54, 421.63, 536.54])
s = np.array([1.16, 1.60, 1.90])
```

构造方程组系数矩阵

- 直线型
  只有一个系数，直接解出 `beta = np.dot(s, Q.T)/np.dot(s, s.T))`
- 抛物线型
  ```
  X = np.vstack([np.ones(len(Q)), Q]).T
  A = np.dot(X.T, X)
  b = np.dot(X.T, s/Q)
  ```
- 幂函数型
  ```
  X = np.vstack([np.ones(len(Q)), np.log10(s)]).T
  A = np.dot(X.T, X) b = np.dot(X.T, np.log10(Q))
  ```
- 对数型
  ```
  X = np.vstack([np.ones(len(Q)), np.log10(s)]).T
  A = np.dot(X.T, X) b = np.dot(X.T, Q)
  ```

求系数

`beta = np.linalg.solve(A, b)` 解方程，`beta` 返回拟合系数，其中 `a = beta[0]`，`b = beta[1]`。

计算降深误差平方和

- Q 误差

  直线型：$\sum_{i=1}^{n}(bs_i - Q_i)^2$

  抛物线型：$\sum_{i=1}^{n}\left(\frac{-a+\sqrt{a^2+4bs_i}}{2b} - Q_i\right)^2$

  幂函数型：$\sum_{i=1}^{n}\left(10^a s_i^b - Q_i\right)^2$

  对数型：$\sum_{i=1}^{n}(a + b\lg s_i - Q_i)^2$

- s 误差

  直线型：$\sum_{i=1}^{n}\left(\frac{Q_i}{b} - s_i\right)^2$

  抛物线型：$\sum_{i=1}^{n}(aQ_i + bQ_i^2 - s_i)^2$

  幂函数型：$\sum_{i=1}^{n}\left(Q_i^{1/b}10^{-a/b} - s_i\right)^2$

  对数型：$\sum_{i=1}^{n}\left(10^{Q_i/b}10^{-a/b} - s_i\right)^2$

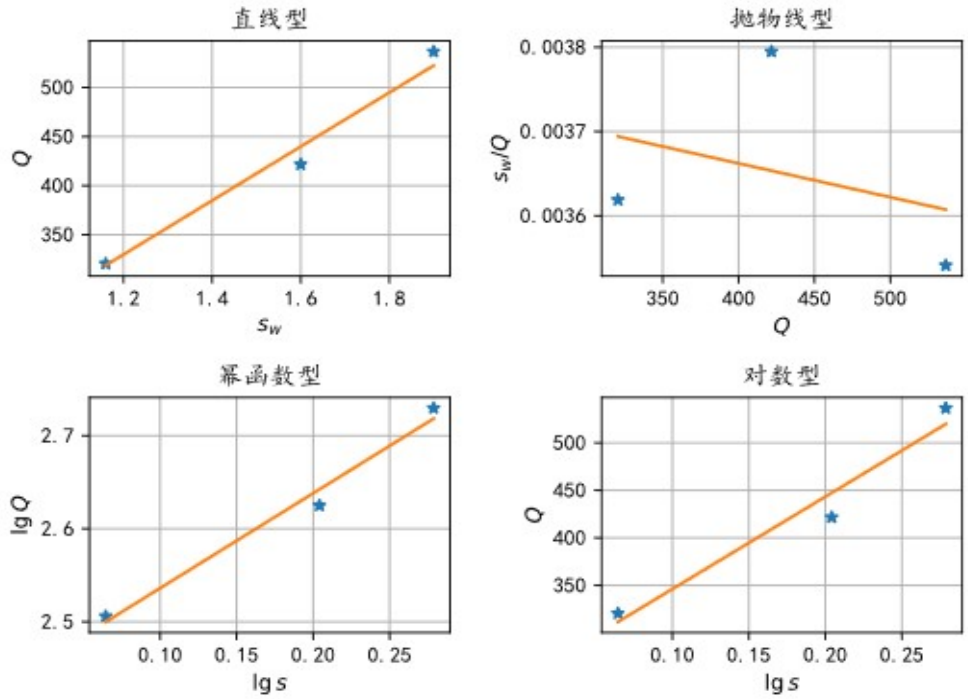
Python 程序

```
%matplotlib widget

# 导入 numpy 与 matplotlib 库
import numpy as np
import matplotlib.pyplot as plt

# 控制小数的显示精度
np.set_printoptions(precision=4)

plt.rcParams['font.sans-serif']=['KaiTi']
# 指定用来显示中文的字体，fc-list 命令可查看可用字体
plt.rcParams['axes.unicode_minus']=False

# (Q_i,s_i) 数据
Q = np.array([320.54, 421.63, 536.54])
s = np.array([1.16, 1.60, 1.90])

# 直线型，直接求解
beta1 = s.dot(Q.T)/s.dot(s.T)
# 抛物线型
X = np.vstack([np.ones(len(Q)), Q]).T  # 形成系数
beta2 = np.linalg.solve(np.dot(X.T, X), np.dot(X.T, s/Q))
```

```
# 幂函数型
X = np.vstack([np.ones(len(Q)), np.log10(s)]).T  # 形成系数
beta3 = np.linalg.solve(np.dot(X.T, X), np.dot(X.T, np.log10(Q)))
# 对数型
X = np.vstack([np.ones(len(Q)), np.log10(s)]).T  # 形成系数
beta4 = np.linalg.solve(np.dot(X.T, X), np.dot(X.T, Q))

# Plot the data
fig, axs = plt.subplots(2, 2)
# 直线型
axs[0,0].plot(s, Q, '*', label="ob")
x = np.linspace(np.min(s),np.max(s), 20)
axs[0,0].plot(x, beta1*x)
axs[0,0].set_title("直线型")
axs[0,0].set(xlabel=r'$s_w$', ylabel=r'$Q$')

# 抛物线型
axs[0,1].plot(Q, s/Q, '*')
x = np.linspace(np.min(Q),np.max(Q), 20)
axs[0,1].plot(x, beta2[0]+beta2[1]*x)
axs[0,1].set_title("抛物线型")
axs[0,1].set(xlabel=r'$Q$', ylabel=r'$s_w/Q$')

# 幂函数型
axs[1,0].plot(np.log10(s), np.log10(Q), '*')
x = np.linspace(np.min(np.log10(s)),np.max(np.log10(s)), 20)
axs[1,0].plot(x, beta3[0]+beta3[1]*x)
axs[1,0].set_title("幂函数型")
axs[1,0].set(xlabel=r'$\lg s$', ylabel=r'$\lg Q$')

# 对数型
axs[1,1].plot(np.log10(s), Q, '*')
x = np.linspace(np.min(np.log10(s)),np.max(np.log10(s)), 20)
axs[1,1].plot(x, beta4[0]+beta4[1]*x)
axs[1,1].set_title("对数型")
axs[1,1].set(xlabel=r'$\lg s$', ylabel=r'$Q$')

for ax in axs.flat:
    ax.grid()

plt.tight_layout()
plt.show()

print('s 误差平方和：')
rss = sum((s - Q/beta1)**2)
print('  直线型:','beta = {:.4f},'.format(beta1),' RSS of s = {:.4e}'.format(rss))
rss = sum((s-beta2[0]*Q - beta2[1]*Q**2)**2)
print('  抛物型:','beta = {},'.format(beta2),' RSS of s = {:.4e}'.format(rss))
```

```
rss = sum((np.float_power(10, (np.log10(Q) - beta3[0])/beta3[1]) - s)**2)
print('幂函数型:','beta = {},'.format(beta3),' RSS of s = {:.4e}'.format(rss))
rss = sum((np.float_power(10, (Q - beta4[0])/beta4[1]) - s)**2)
print('  对数型:','beta = {},'.format(beta4),' RSS of s = {:.4e}'.format(rss))
print()
```

输出：




```
s 误差平方和：
  直线型: beta = 274.8763,  RSS of s = 7.1050e-03
  抛物型: beta = [ 3.8228e-03 -4.0151e-07],  RSS of s = 5.3899e-03
幂函数型: beta = [2.4341 1.0195],  RSS of s = 6.4173e-03
  对数型: beta = [248.9729 971.6074],  RSS of s = 1.5437e-02
```

计算 Q 的误差平方和

```
print('Q 误差平方和：')
rss = sum((Q - s*beta1)**2)
print('  直线型:','beta = {:.4f},'.format(beta1),' RSS of Q = {:.4e}'.format(rss))
rss = sum(((-beta2[0]+np.sqrt(beta2[0]*beta2[0]+4*beta2[1]*s))/2/beta2[1]-Q)**2)
print('  抛物型:','beta = {},'.format(beta2),' RSS of s = {:.4e}'.format(rss))
rss = sum((np.power(10,beta3[0])*np.power(s,beta3[1])-Q)**2)
print('幂函数型:','beta = {},'.format(beta3),' RSS of s = {:.4e}'.format(rss))
ss = sum((beta4[0]+np.log10(s)*beta4[1]-Q)**2)
print('  对数型:','beta = {},'.format(beta4),' RSS of s = {:.4e}'.format(rss))
```

思考题

(1) 通过比较，抛物线型的 s 或 Q 的误差平方和最小。问该井的 $Q \sim s$ 曲线确定为抛物型是否合适？为什么？

(2) 对比抛物线型的 $Q \sim s$ 与 C. E. Jacob 井损表达式可以看出，它们的形式是一样的。若按上述示例数据计算井损系数 C，发现 C 是负值，试分析原因。

(3) 若已知含水层某钻孔的 $Q \sim s$ 类型，问该类型及数据能否用于预测同含水层其它钻孔的降深？

实验三、配线法求参

1. 配线法原理

地下水动力学非稳定流的计算，难点在于井函数的计算，本指导书附录介绍了 Theis、Hantush-Jacob 井函数的计算方法，复杂的井函数要求较高的数学基础。配线法需要提前计算井函数，并绘出标准曲线。以 Theis 配线法为例。

Theis 公式及 u 的表达式取对数

$$\lg s+\lg\frac{4\pi T}{Q}=\lg W(u),\quad \lg\frac{t}{r^2}+\lg\frac{4T}{S}=\lg\frac{1}{u}$$

在标准曲线坐标系中移动 $s\sim\frac{t}{r^2}$ 散点图，记坐标平移量为

$$\Delta\lg(s)=\lg\frac{4\pi T}{Q},\quad \Delta\lg\frac{t}{r^2}=\lg\frac{4T}{S}$$

根据平移量按如下公式计算 (T,S)：

$$T=\frac{Q}{4\pi}10^{\Delta\lg(s)},\quad S=4T10^{-\Delta\lg\frac{t}{r^2}}$$

2. 用 Excel 表配线

地动-配线法上机用表.xlsm 中有已计算好的井函数表，并绘制好标准曲线。

程序设计思路

利用 VBA 编程，用滚动条滚动控制散点图的平移量，并将数值返给储存 $\Delta\lg(s),\Delta\lg\frac{t}{r^2}$ 的单元格，在标准曲线上以 $\left(\frac{t}{r^2}\times10^{\Delta\lg(t/r^2)},s\times10^{\Delta\lg(s)}\right)$ 绘图，实现滚动条、图形的联动。假定单元格 K27 为水平平移量 $\Delta\lg\frac{t}{r^2}$），单元格 K28 为垂直平移量 $\Delta\lg(s)$，这两个单元格初始值可设为 0。

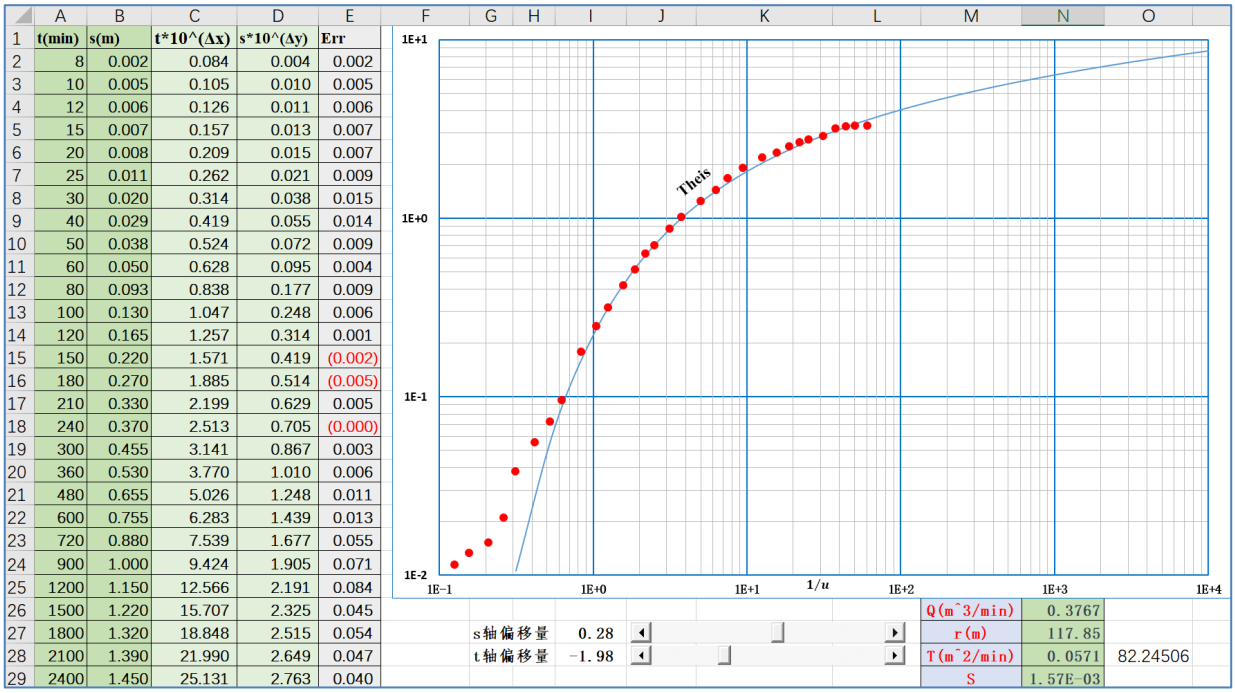
- 控制水平平移量的滚动条属性为
  `name = "HS", Min = -500, Max = 500, Value = 0,`
  `SmallChange = 1, LargeChange = 10`
- 控制垂直平移量的滚动条属性为
  `name = "VS", Min = -300, Max = 300, Value = 0,`
  `SmallChange = 1, LargeChange = 10`
- 滚动条添加事件：

```
Private Sub HS_Change()
    Range("K27").Value = HS.Value / 100
End Sub
```

```
Private Sub VS_Change()
    Range("K28").Value = VS.Value / 100
End Sub
```

以上代码在对数刻度上左右最大移动 5 个周期，上下最大移动 3 个周期，移动最小值为一个周期的 1%。将 K27、K28 的值代入公式可计算参数 T、S。

显示结果：



	A	B	C	D	E
1	t(min)	s(m)	t*10^(Δx)	s*10^(Δy)	Err
2	8	0.002	0.084	0.004	0.002
3	10	0.005	0.105	0.010	0.005
4	12	0.006	0.126	0.011	0.006
5	15	0.007	0.157	0.013	0.007
6	20	0.008	0.209	0.015	0.007
7	25	0.011	0.262	0.021	0.009
8	30	0.020	0.314	0.038	0.015
9	40	0.029	0.419	0.055	0.014
10	50	0.038	0.524	0.072	0.009
11	60	0.050	0.628	0.095	0.004
12	80	0.093	0.838	0.177	0.009
13	100	0.130	1.047	0.248	0.006
14	120	0.165	1.257	0.314	0.001
15	150	0.220	1.571	0.419	(0.002)
16	180	0.270	1.885	0.514	(0.005)
17	210	0.330	2.199	0.629	0.005
18	240	0.370	2.513	0.705	(0.000)
19	300	0.455	3.141	0.867	0.003
20	360	0.530	3.770	1.010	0.006
21	480	0.655	5.026	1.248	0.011
22	600	0.755	6.283	1.439	0.013
23	720	0.880	7.539	1.677	0.055
24	900	1.000	9.424	1.905	0.071
25	1200	1.150	12.566	2.191	0.084
26	1500	1.220	15.707	2.325	0.045
27	1800	1.320	18.848	2.515	0.054
28	2100	1.390	21.990	2.649	0.047
29	2400	1.450	25.131	2.763	0.040

3. 配线法的 Python 程序设计

Theis 井函数可以用 scipy.special.exp1 计算。SciPy 是基于 NumPy 的数学算法程序包。借助 SciPy，交互式 Python 数据处理环境具有与 MATLAB，IDL，Octave，R-Lab 和 SciLab 等类似的功能。

程序中用到了 numpy, math , matplotlib, ipywidgets 等程序库，相关介绍见联机文档。程序设计流程为“库导入 ⟹ 数据准备 -> 绘图准备 ⟹ widgets.interact 互动”。

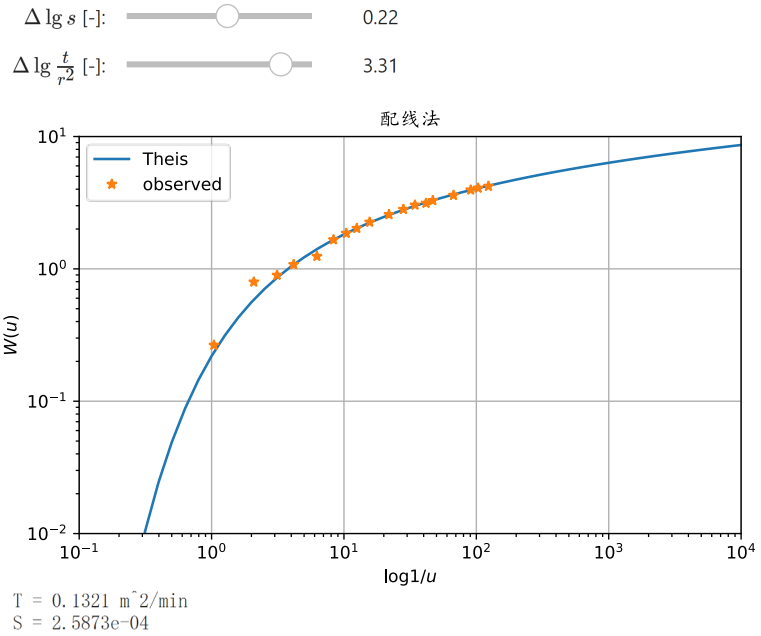
Python 程序

```
%matplotlib inline

# 导入库
import numpy as np
from scipy.special import exp1
import math as math
import matplotlib.pyplot as plt
```

```
import ipywidgets as widgets

# 控制小数的显示精度
np.set_printoptions(precision=4)

plt.rcParams['font.family']=['SimHei']
# 指定用来显示中文的字体，fc-list 命令可查看可用字体
plt.rcParams['axes.unicode_minus']=False

# 准备数据，根据配线法的种类，需要专门准备数据
Q = 60/60    # m^3/min
r2 = 140

t2 = np.array([10, 20, 30, 40, 60, 80, 100, 120, 150, 210,
               270, 330, 400, 450, 645, 870, 990, 1185])
s2 = np.array([0.16, 0.48, 0.54, 0.65, 0.75,
               1.00, 1.12, 1.22, 1.36, 1.55,
               1.70, 1.83, 1.89, 1.98, 2.17,
               2.38, 2.46, 2.54])
tr2 = t2/r2**2
s = s2

# 设置标准曲线的界限
ymin = 1.0e-2
ymax = 10.0
xmin = 1.0e-1
xmax = 1.0e4

# 坐标加密可以绘出光滑的曲线
x = np.linspace(-1, 4, 51)
x = np.float_power(10, x) # 1/u

# 初始参数，相当于位移 = 0
T = Q/4/np.pi
S = 4*T

# Plot the data
def Theis_fit(dlogs, dlogtr2):
    global T, S# 设置全局变量，值可以穿透程序
    # 图形设置
    plt.style.use('default')
    fig, ax = plt.subplots()
    ax.set_xlim(xmin, xmax)
    ax.set_ylim(ymin, ymax)
    ax.set_xscale("log")
    ax.set_yscale("log")
    ax.set_aspect(1)
    plt.xlabel('$\log 1/u$')
```

```
    plt.ylabel('$W(u)$')
    ax.grid(True)
    # 计算参数，配线法不同，公式也不一样
    T = Q/4/np.pi*np.float_power(10, dlogs)
    S = 4*T*np.float_power(10, -dlogtr2)
    # 绘制标准曲线
    ax.plot(x, exp1(1/x), label="Theis")
    # 绘制平移的散点图形
    ax.plot(tr2*np.float_power(10, dlogtr2), s*np.float_power(10, dlogs), '*',
            label="observed")
    plt.legend()
    # 指定图标题，显示中文
    ax.set_title("配线法", fontproperties="KaiTi", fontsize=12)
    plt.tight_layout()
    plt.show()
    # 输出参数
    print('  T = {:.4f} m^2/min'.format(T))
    print('  S = {:.4e}'.format(S))

# ipywidgets 小部件控制参数实现互动。ipywidgets 有缓冲功能，
# 同一个 Notebook 复制的代码得不到所期望的结果
widgets.interact(
    Theis_fit,
    dlogs = widgets.FloatSlider(
        value=0, min=-3.0, max=+3.0, step=0.01,
        description=r'$\Delta\lg s$ [-]:',
        continuous_update=False,
        readout_format='.2f',
        disabled=False),
    dlogtr2 = widgets.FloatSlider(
        value=0, min=-5.0, max=+5.0, step=0.01,
        description=r'$\Delta\lg\frac{t}{r^2}$ [-]:',
        continuous_update=False,
        readout_format='.2f',
        disabled=False),
    );
```

显示结果：



练习

承压含水层定流量非稳定流群孔抽水试验，抽水孔定流量为 $60m^3/h$，观 1、观 2、观 3 与观 4 孔距抽水孔分别 $43m$、$140m$、$510m$、$780m$，各观测孔水位降深如下表所示。

累计抽水时间(min)	观 1 降深(m)	观 2 降深(m)	观 3 降深(m)	观 4 降深(m)
10	0.73	0.16	0.04	
20	1.28	0.48		
30	1.53	0.54		
40	1.72	0.65	0.06	
60	1.96	0.75	0.20	
80	2.14	1.00	0.20	0.04
100	2.28	1.12	0.20	
120	2.39	1.22	0.21	0.08
150	2.54	1.36	0.24	0.09
210	2.77	1.55	0.40	0.16
270	2.99	1.70	0.53	0.25
330	3.10	1.83	0.63	0.34
400	3.20	1.89	0.65	0.42
450	3.26	1.98	0.73	0.50
645	3.47	2.17	0.93	0.71
870	3.68	2.38	1.14	0.87
990	3.77	2.46	1.24	0.96
1185	3.85	2.54	1.35	1.06

(1) 按 $s \sim \frac{t}{r^2}$ 配线法求解承压含水层水文地质参数。

思考题：

(1) 如何依据不同抽水试验条件选择合理的 Theis 解配线类型和方法？

(2) 为什么实测与标准曲线重合后，匹配点可以任意选取？任意选取匹配点会影响该方法结果的可靠性吗？

实验四、直线图解法

1. 直线图解法原理

以 $s \sim \lg t$ 为例：

$$s \sim \frac{0.183Q}{T}\lg t + \frac{0.183Q}{T}\lg\frac{2.25T}{r^2S}$$

写成点斜式

$$s \sim i\lg t - i\lg t_0$$

通过调整 t_0 与 i 拟合直线，并用如下公式计算参数：

$$T = \frac{0.183Q}{i}, \quad S = \frac{2.25Tt_0}{r^2}$$

2. Excel 直线图解法

有了配线法的基础，用 Excel 或 Python 实现直线图解法都是比较简单的。

程序设计

- 先以观测数据在数据表中绘制散点图，坐标刻度设为单对数刻度；
- 用两个单元格，如 K27 保存直线斜率i，K28 保存零降深截距 t_0；
- 通过滚动条滚动 i 与 t_0 的变化；
- 计算与 t_j 对应的降深，用点斜式公式 $\hat{s} = i(\lg t - \lg t_0)$，并在散点图中添加以数据 $(t_j, \hat{s}_j)$ 绘制的图形，该图形应为一条直线；
- 根据调整后的 i 与 t_0 按公式计算参数。

3. 直线图解法的 Python 程序

程序设计

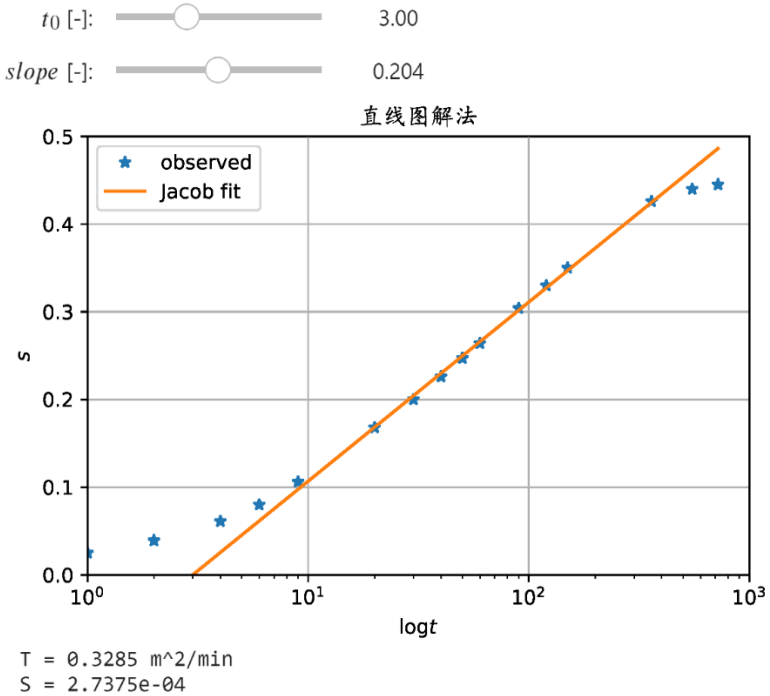
程序设计流程为“库导入 ⟹ 数据准备 -> 绘图准备 ⟹ `widgets.interact` 互动”。

```
%matplotlib inline

import numpy as np
import math as math
import matplotlib.pyplot as plt
import ipywidgets as widgets

# 控制小数的显示精度
np.set_printoptions(precision=4)
```

```
plt.rcParams['font.sans-serif']=['KaiTi']
# 指定用来显示中文的字体，fc-list 命令可查看可用字体
plt.rcParams['axes.unicode_minus']=False

# 准备数据
Q = 528/1440  # m^3/min
r = 90        # m
t = np.array([1, 2, 4, 6, 9, 20, 30, 40, 50, 60,
              90, 120, 150, 360, 550, 720])     # min
s = np.array([2.5, 3.9, 6.1, 8.0, 10.6, 16.8, 20.0, 22.6, 24.7, 26.4,
              30.4, 33.0, 35.0, 42.6, 44.0, 44.5])/100   # m
#绘图界限
ymin = 0.0
ymax = math.ceil(max(s*10))/10
imin = math.floor(math.log10(min(t)))
imax = math.ceil(math.log10(max(t)))
xmin = 10**imin
xmax =  10**imax

# 最小二乘法求初始参数
A = np.vstack([np.ones(len(t)), np.log10(t)]).T  # 形成系数
# 求解方程组
beta = np.linalg.solve(np.dot(A.T, A), np.dot(A.T, s))
i0 = beta[1] # 斜率
t0 = np.float_power(10, -beta[0]/beta[1]) #截距
T = 0.183*Q/i0
S = 2.25*T*t0/r**2

# Plot the data
def line_fit(a, b):  # a - 截距，b - 斜率
    global i0, t0, T, S    # 全局变量
    # 设置图形
    plt.style.use('default')
    fig, ax = plt.subplots(figsize=(6, 4))
    ax.set_xlim(xmin, xmax)
    ax.set_ylim(ymin, ymax)
    ax.set_xscale("log")
    plt.xlabel('$\log t$')
    plt.ylabel('$s$')
    ax.grid(True)
    # 绘观测数据散点图
    ax.plot(t, s, '*', label="observed")
    # 绘直线
    ax.plot(t, b*np.log10(t/a), label="Jacob fit")
    plt.legend()
    # 计算调整后的参数
    T = 0.183*Q/b
```

```
    S = 2.25*T*a/r**2
    ax.set_title("直线图解法", fontproperties='KaiTi',
                 fontsize=12) # 指定显示中文字体
    plt.show()
    # 输出参数
    print('  T = {:.4f} m^2/min'.format(T))
    print('  S = {:.4e}'.format(S))

# 小部件互动
widgets.interact(
    line_fit,
    a = widgets.FloatSlider(
        value=t0, min=1, max=t0*5.0, step=1,
        description='$t_0$ [-]:',
        continuous_update=False,
        readout_format='.2f',
        disabled=False),
    b = widgets.FloatSlider(
        value=i0, min=0.5*i0, max=2*i0, step=0.001,
        description='$slope$ [-]:',
        continuous_update=False,
        readout_format='.3f',
        disabled=False)
    );
```

显示结果：



练习

承压含水层定流量非稳定流群孔抽水试验，抽水孔定流量为 $60m^3/h$，观 1、观 2、 观 3 与观 4 孔距抽水孔分别 $43m$、$140m$、$510m$、$780m$，各观测孔水位降深如下表所示。

累计抽水时间(min)	观 1 降深(m)	观 2 降深(m)	观 3 降深(m)	观 4 降深(m)
10	0.73	0.16	0.04	
20	1.28	0.48		
30	1.53	0.54		
40	1.72	0.65	0.06	
60	1.96	0.75	0.20	
80	2.14	1.00	0.20	0.04
100	2.28	1.12	0.20	
120	2.39	1.22	0.21	0.08
150	2.54	1.36	0.24	0.09
210	2.77	1.55	0.40	0.16
270	2.99	1.70	0.53	0.25
330	3.10	1.83	0.63	0.34
400	3.20	1.89	0.65	0.42
450	3.26	1.98	0.73	0.50
645	3.47	2.17	0.93	0.71
870	3.68	2.38	1.14	0.87
990	3.77	2.46	1.24	0.96
1185	3.85	2.54	1.35	1.06

(1) 根据 Jacob 解应用条件，分析本题列表中适用数据范围；

(2) 绘制 $s(r,t) \sim \lg\frac{t}{r^2}$ 直线并求含水层参数。

思考题：

(1) 对比 Theis 配线、Jacob 直线求参结果的差异并分析原因。

(2) 选择拟合直线数据应该考虑哪些问题？

(3) 试用单个观测孔、多个观测孔求参，并比较结果。

(4) 用最小二乘法按 Jacob 公式求参。

实验五、Theis 公式的非线性拟合

1. 非线性拟合公式

Theis 公式

$$s=\frac{Q}{4\pi T}\int_u^{\infty}\frac{1}{y}e^{-y}dy=\frac{Q}{4\pi T}W(u)$$

取初始参数 T_0，S_0，降深 s 的泰勒级数展开为

$$s(T,S)=s(T_0,S_0)+\left.\frac{\partial s}{\partial S}\right|_{S=S_0}\Delta S+\left.\frac{\partial s}{\partial T}\right|_{T=T_0}\Delta T+O(\Delta T^2+\Delta S^2)$$

式中，$\Delta T=T-T_0,\Delta S=S-S_0$。因为 $\frac{\partial u}{\partial s}=\frac{u}{s},\frac{\partial u}{\partial T}=-\frac{u}{T}$，得

$$\frac{\partial s}{\partial S}=-\frac{Q}{4\pi T}\frac{e^{-u}}{S},\quad \frac{\partial s}{\partial T}=\frac{Q}{4\pi T^2}[-W(u)+e^{-u}]$$

记

$$a=\left.\frac{\partial s}{\partial S}\right|_{S=S_0},\quad b=\left.\frac{\partial s}{\partial T}\right|_{T=T_0},\quad c=s(T_0,S_0)$$

忽略降深 s 泰勒级数展开式的高阶无穷小，有

$$\hat{s}=c+a\cdot\Delta S+b\cdot\Delta T$$

上式以 $(\Delta T,\Delta S)$ 为未知系数，$\hat{s}$ 为 s 的预测值，a,b,c 根据初始参数（T_0,S_0）及观测时间计算。由此式可构造最小二乘问题求参。

2. Theis 公式非线性拟合的 Python 程序

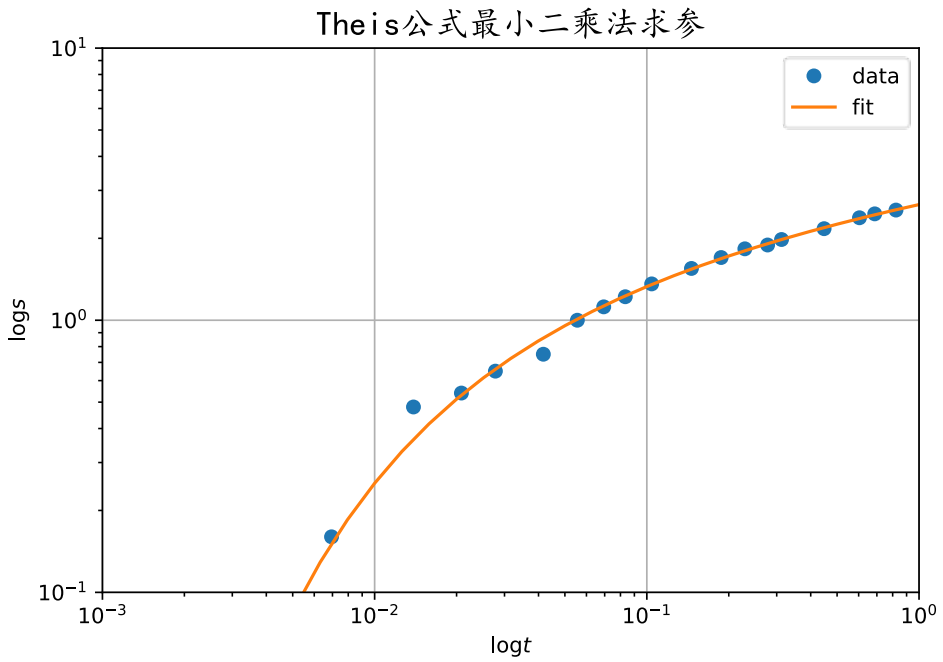
```
%matplotlib inline

import numpy as np
import math as math
import matplotlib.pyplot as plt
import ipywidgets as widgets
from scipy.special import exp1

# 控制小数的显示精度
np.set_printoptions(precision=4)

plt.rcParams['font.sans-serif']=['KaiTi']
# 指定用来显示中文的字体，fc-list 命令可查看可用字体
plt.rcParams['axes.unicode_minus']=False

def calc_u(p, r, t):    # 计算变量 u
```

```
    S, T = p
    return r**2*S/4/T/t

# 定义井函数计算方法，可用 scipy.special.exp1 计算.
# 也可用多项式逼近的方法计算井函数
def theis_drawdown(p, t, Q, r):  # 计算降深
    S, T = p
    u = calc_u(p, r, t)
    s_theis = Q/4/np.pi/T*exp1(u)
    return s_theis

def theis_Dfun(p, t, s, Q, r):  # 计算 ds/dT,ds/S
    """Calculate and return ds/dS and ds/dT for the Theis equation.
    The parameters S, T are packed into the tuple p.
    """
    S, T = p
    u = calc_u(p, r, t)
    dsdS = -Q/4/np.pi/T/S*np.exp(-u)
    dsdT = Q/4/np.pi/T**2*(-exp1(u) + np.exp(-u))
    return np.array((-dsdS, -dsdT)).T   # 返回负梯度

def theis_resid(p, t, s, Q, r): # 计算降深预测误差
    S, T = p
    return s - theis_drawdown(p, t, Q, r)

# 抽水量 Q, 观测孔位置 r
r = 125.0  # m, 15#孔位置
Q = 1440   # m^3/d
#  从文件读数据到数组 t, s 中, 文件和当前脚本在同一目录下不用写具体路径
fname = 'theis_input.txt'
data = np.loadtxt(fname, delimiter=',')
# 2#孔降深 data[0:,1]；15#孔降深 data[0:,2]
t = data[0:, 0]/1440
s = data[0:, 2]
# 取数组长度
n = len(t)
# 初始参数怎么取？随机取两点，用 Jacob 两点公式计算。
np.random.seed()
i1 = np.random.randint(int(n/2))
i2 = np.random.randint(int(n/2), n - 1)
t1 = t[i1]
t2 = t[i2]
s1 = s[i1]
s2 = s[i2]
kk = (s1 - s2)/np.log10(t1/t2)
T0 = 0.183*Q/kk
S0 = 2.25*T0*t1/r**2/np.float_power(10, s1/kk)
```

```
p = S0, T0
S, T = p

# 循环计数器
j = 0
while True:
    c = theis_drawdown(p, t, Q, r)
    a = theis_Dfun(p, t, s, Q, r)[:, 0]
    b = theis_Dfun(p, t, s, Q, r)[:, 1]
    X = np.vstack([a, b]).T  # 形成系数矩阵
    # 调用 np.linalg.solve
    beta = np.linalg.solve(np.dot(X.T, X), np.dot(X.T, c - s))
    DS = beta[0]
    DT = beta[1]

    while True:
        if S + DS < 0:    # 步长太大导致参数为负数，不合理！
            DS = DS/2.0  # 减小步长
        else:
            break
    while True:
        if T + DT < 0:    # 步长太大导致参数为负数，不合理！
            DT = DT/2.0  # 减小步长
        else:
            break
    j = j + 1  # 循环计数器
    if j > 1000:  # 循环次数多，程序可能有错误
        print('  循环超过 1000 次，请先检查程序再运行！')
        break
    if (abs(DT) < 1.0E-6) or (abs(DS) < 1.0e-8):  # 判断计算误差
        break
    # 新参数值
    p = S + DS, T + DT
    S, T = p

# 生成绘图数据
x0 =[i for i in np.arange(-3, 0.1, 0.1)]
x =np.power(10, x0)
y = theis_drawdown(p, x, Q, r)
plt.style.use('default')
fig, ax = plt.subplots()
ax.grid(True)
ax.set_xscale("log")
ax.set_yscale("log")
ax.set_xlim(0.001, 1)
ax.set_ylim(0.1, 10)
ax.set_aspect(1)
ax.plot(t, s, 'o', label='data')
```

```
ax.plot(x, y, label='fit')
plt.legend()
plt.xlabel(r'$\log t$')
plt.ylabel(r'$\log s$')
ax.set_title("Theis 公式最小二乘法求参", fontproperties="KaiTi",
             fontsize=16) # 指定显示中文字体
plt.show()

# 显示最终结果
print('循环次数 = {}'.format(j))
print('T = {:.2f} m^2/d,'.format(T), 'S = {:.4e}'.format(S))
print('RSS = {:.4e}'.format(np.sqrt(np.sum((c - s)**2))))
```

显示结果




```
循环次数 =     5
T = 193.3796 m^2/d
S = 3.1375e-04
RSS = 1.6944e-01
```

注意： 程序设计需要考虑以下问题：

- 最小二乘法得出的 $(T_0 + \Delta T, S_0 + \Delta S)$ 有时为不合理的负值，用二分法缩小步长可保证参数为正值；
- 最小二乘法只是在初值的基础上进行了一步优化。为得到最优参数，需用得出的参数作为新的初值重复计算；
- 初值对最终结果有影响。选取两组观测数据按 Jacob 公式可计算出合理的参数初值。

附录：井函数计算

1. Theis 井函数

$$W(u)=\int_u^{\infty}\frac{\mathrm{e}^{-y}}{y}\mathrm{d}y=\mathrm{E}_1(u)=-\mathrm{E}_i(-u)$$

scipy.special.exp1 可以计算井函数, 也可用多项式逼近。

```
import numpy as np
import scipy.special as sps

u = np.array([10**x for x in range(-4, 1)])
print(sps.exp1(u))
```

输出：

```
[8.6332 6.3315 4.0379 1.8229 0.2194]
```

$W(u)$ 的多项式逼近

- $0<u\le 1$ 时

$$W(u)=-\ln u+a_0+a_1u+a_2u^2+a_3u^3+a_4u^4+a_5u^5$$

式中

$$\begin{array}{llll} a_0= & -0.57721566 & a_3= & 0.05519968 \\ a_1= & 0.99999193 & a_4= & -0.00976004 \\ a_2= & -0.24991055 & a_5= & 0.00107857 \end{array}$$

- $1<u<\infty$ 时

$$W(u)=\frac{b_0+b_1u+b_2u^2+b_3u^3+u^4}{c_0+c_1u+c_2u^2+c_3u^3+u^4}\cdot\frac{e^{-u}}{u}$$

式中

$$\begin{array}{llll} b_0= & 0.2677737343 & c_0= & 3.9584969228 \\ b_1= & 8.6347608925 & c_1= & 21.0996530827 \\ b_2= & 18.0590169730 & c_2= & 25.6329561486 \\ b_3= & 8.5733287401 & c_3= & 9.5733223454 \end{array}$$

Python 程序

```
import numpy as np
np.set_printoptions(precision=4)
# Theis 井函数的多项式逼近
def theis(u):
    a = np.array([-0.57721566, 0.99999193, -0.24991055,
```

```
                  0.05519968, -0.00976004, 0.00107857])
    b = np.array([0.2677737343, 8.6347608925, 18.059016973, 8.5733287401])
    c = np.array([3.9584969228, 21.0996530827, 25.6329561486, 9.5733223454])
    if u <= 1:
        w = -np.log(u) + a[0] + u*(a[1] + u*(a[2] + u*(a[3] + u*(a[4] + u*a[5]))))
    else:
        w1 = b[0] + u*(b[1] + u*(b[2] + u*(b[3] + u)))
        w2 = c[0] + u*(c[1] + u*(c[2] + u*(c[3] + u)))
        w = (w1/w2)*np.exp(-u)/u
    return w
vtheis = np.vectorize(theis) # 向量化函数

u = np.array([10**x for x in range(-4, 1)])
print(vtheis(u))
```

输出：

```
[8.6332 6.3315 4.0379 1.8229 0.2194]
```

2. Hantush-Jacob 井函数

$$W(u,\beta)=\int_u^\infty \frac{1}{y}\exp\left(-y-\frac{\beta^2}{4y}\right)dy,\quad W(u,\beta)=2K_0(\beta)-W\left(\frac{\beta^2}{4u},\beta\right)$$

式中：$u=\frac{r^2S}{4Tt}$，$\beta=\frac{r}{B}$.

级数形式（Hunt，1977）

$$W(u,\beta)=\sum_{n=0}^{\infty}(-\frac{\beta^2}{4u})^n\frac{E_{n+1}(u)}{n!}$$

式中，

$$E_n(u)=\int_1^\infty \frac{e^{-uy}}{y^n}dy=u^{n-1}\int_u^\infty \frac{e^{-y}}{y^n}dy\quad (n=0,1,2,\cdots;\Re u>0)$$

为指数积分，当 $\frac{\beta^2}{4u}<1$ 时级数快速收敛。

`scipy.special.expn` 计算指数积分，`scipy.special.k0` 计算 0 阶第二类修正 Bessel 函数 $K_0(x)$。

Python 程序

```
import numpy as np
import scipy.special as sps
def hantush_jacob(x, y):
    if x < 0:
        print('Negative are not allowed')
        return np.nan
    if x == 0:
```

```
        return 2.0*sps.k0(y)
    r = 1
    t = y**2/(4*x)
    b = 2 * x
    if y <= b: # beta<2u
        W = 0
        n = 0
        term = r*sps.expn(n+1, x)
        while np.abs(term) > 1e-10:
            W = W + term
            n = n + 1
            r = r*(-t)/n
            term = r*sps.expn(n+1, x)
    else:
        W = 2.0*sps.k0(y)
        n = 0
        term = r*sps.expn(n+1, t)
        while np.abs(term) > 1e-10:
            W = W - term
            n = n + 1
            r = r*(-x)/n
            term = r*sps.expn(n+1, t)
    return W
vhantush_jacob = np.vectorize(hantush_jacob) # 向量化函数

y = np.array([0.05 for i in range(5)])
u = np.array([10**x for x in range(-4, 1)])
print(vhantush_jacob(u, y))
```

输出：

```
[6.2282 5.7965 3.9795 1.8184 0.2193]
```

若用 Excel 计算，需要自己编写 VBA 代码，参考公式如下：

指数积分（Abramowitz and Stegun，1964)

$$E_n(u) = \int_1^{\infty} \frac{e^{-uy}}{y^n} dy = u^{n-1} \int_u^{\infty} \frac{e^{-y}}{y^n} dy \quad (n = 0,1,2,\cdots; \Re u > 0)$$

递推关系：

$$E_{n+1}(u) = \frac{1}{n}[e^{-u} - uE_n(u)]$$

$$E_n(u) = \frac{1}{u}[e^{-u} - nE_{n+1}(u)], (n = 0,1,2,\cdots)$$

当 u 为实数时，

- $n = 0$

$$E_0(u) = \frac{e^{-u}}{u}$$

- $n = 1$

$$E_1(u) = W(u)$$

- $n > 1$
  - $u \leq 5$：递推公式向前（递推）计算高阶指数积分；
  - $u > 5$：
    - $n \leq u$ 时用递归公式向后（递归）计算低阶指数积分;
    - $n > u$ 时，记 $n_0 =< u >$（取 u 的整数部分），用递归关系计算小于 n_0 的低阶指数积分，用递推关系计算大于 n_0 的高阶指数积分。

- 对于大的 n, 可用以下的多项式逼近

$$E_n(u) = \frac{e^{-u}}{u+n}\{1 + \frac{n}{(n+u)^2} + \frac{n(n-2u)}{(u+n)^4} + \frac{n(6u^2 - 8nu + n^2)}{(u+n)^6} + R(n,u)\}$$

式中，

$$-0.36\mathrm{n}^{-4} \leq \mathrm{R(n,u)} \leq \left(1 + \frac{1}{\mathrm{u+n-1}}\right)\mathrm{n}^{-4}(\mathrm{u} > 0)$$