天然气水合物资源量评价

摘要

天然气水合物，即可燃冰，广泛分布于深海或陆域永久冻土中。由于其燃烧后仅生成少量的二氧化碳和水，且储量巨大，因此被国际公认为石油等传统能源的理想替代品。随着《2024 年能源工作指导意见》[1]的出台，推动经济社会向清洁化、低碳化方向发展的呼声日益高涨，加大清洁低碳能源消费替代力度成为当务之急。在这一背景下，天然气水合物作为一种潜在的新型清洁能源备受瞩目。对其资源进行准确评估显得尤为重要。

针对问题一，对附件数据进行探索性数据分析（EDA）和数据预处理，根据资源量与储层参数的线性关系建立天然气水合物资源量评价指标。并结合井位坐标和 Kriging 空间插值确定天然气水合物资源分布范围。

针对问题二，为确定有效厚度、地层孔隙度和饱和度的概率分布以及在勘探区域内的变化规律，采用 Q-Q 图和 Shapiro-Wilk、Anderson-Darling 对储层参数分布进行检验，并结合问题一中的探索性数据分析确定在勘探区域内的变化规律。

针对问题三，利用在问题一中的 Kriging 空间插值估计每个井位的面积进而估计各个井位的天然气水合物资源量，结合问题二中三个参数的分布规律，对资源量与储层参数的函数取对数运算，使用核密度估计获取天然气水合物资的概率分布，利用蒙特卡洛模拟估计该勘探区域的天然气水合物资源量。

针对问题四，对本区域储量有个更精细勘查结果，拟在本区域再增加 5 口井，先在勘探区域以一定的距离为间距进行模拟井位的插入，运用 Kriging 插值计算模拟点的天然气水合物资源量，结合多目标优 NSGA-II 算法迭代出最优的 5 个勘探井口。

以上是关于天然气水合物资源量评价问题的分析和解决方案。

关键字： Kriging；统计检验；蒙特卡洛；Shoelace；NSGA-II；

目录

摘要.....1
目录.....2
一、问题重述.....3
1.1 问题背景.....3
1.2 问题提出.....3
二、模型假设.....3
三、符号说明.....4
四、问题一的模型建立与求解.....4
4.1 问题分析.....4
4.2 数据预处理.....5
4.3 参数计算.....6
4.4 问题求解.....8
五、问题二的模型建立与求解.....10
5.1 问题分析.....10
5.2 方法介绍.....10
5.2.1Q-Q 图.....10
5.2.2 Shapiro-Wilk 检验.....10
5.2.3 Anderson-Darling 检验.....11
5.3 问题求解.....11
5.3.1 有效厚度概率分布及变化规律.....11
5.3.2 孔隙度概率分布及变化规律.....13
5.3.3 饱和度概率分布及变化规律.....14
六、问题三的模型建立与求解.....15
6.1 问题分析.....15
6.2 概率分布.....15
6.3 蒙特卡洛模拟.....17
七、问题四的模型建立与求解.....18
7.1 问题分析.....18
7.2NSGA-II 算法.....18
7.3 问题求解.....19
八、模型的评价与推广.....20
8.1 模型的优点.....20
8.2 模型的缺点.....20
8.3 模型的推广.....21
九、参考文献.....21
附录.....21

一、 问题重述

1.1 问题背景

天然气水合物分布于深海或陆域永久冻土中，其燃烧后仅生成少量的二氧化碳和水，污染远小于煤、石油等，且储量巨大，因此被国际公认为石油等的接替能源。作为一种高效的清洁后备能源，受到发达国家和能源缺乏国家高度重视，但要实现产业化仍存在水合物的资源勘探、空间分布定位、资源量的评价、产量的经济评估和对气候变化影响评价等一系列技术问题，且天然气水合物的资源评估更是受到了水合物饱和度、分布深度、分布面积和孔隙度的影响。因此进行准确和可靠的资源量评估，对天然气水合物后续开发利用至关重要。在已知某海域 14 个位置钻孔深度、对应深度的测量的孔隙度和天然气水合物饱和度信息的情况下解决以下四个问题。

1.2 问题提出

问题一：根据附件勘探井位信息确定天然气水合物资源分布范围。

问题二：确定研究区域内天然气水合物资源参数有效厚度、地层孔隙度和饱和度的概率分布及其在勘探区域内的变化规律。

问题三：请给出天然气水合物资的概率分布，以及估计天然气水合物资源量。

问题四：为了对本区域储量有个更精细勘查结果，拟在本区域再增加 5 口井，问如何安排井位？

二、 模型假设

假设 1：勘探区域的井位在同一个水平面；

假设 2：天然气水合物资源分布均匀；

假设 3：实验中数据预处理步骤正确；

假设 4：测量数据异常值是由观测误差造成的；

假设 5：有效面积服从正态分布；

三、 符号说明

符号	意义
D_i	深度测量数据
ϕ_i	样本点的孔隙度
S_i	样本点的饱和度
n	样本数
K	未知点的估计值
M	已知点的观测值
x_i	单个样本
$\overline{x}$	样本均值
μ	总体均值
σ	总体方差

四、 问题一的模型建立与求解

4.1 问题分析

根据附件勘探井位信息确定天然气水合物资源分布范围，题目中已给出计算资源量与储层参数的线性关系，但测量数据为深度点的数据，因此无法确定有效面积，这里利用有效厚度、孔隙度以及水合物饱和度建立评价指标，对资源量进行估计，并结合井位坐标和 Kriging 空间插值即可确定天然气水合物资源分布范围。

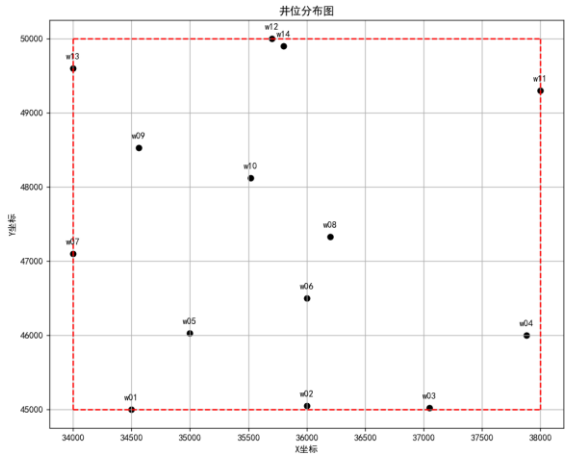

图 1 井位分布图

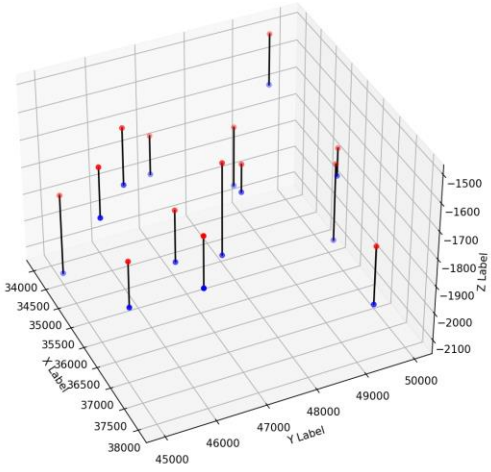

图 2 井位测量深度三维图

根据井位分布以及深度可视化，可以观察到天然气水合物大致分布在一个矩形区域内。在这个区域中，天然气水合物含量会受到有效厚度、孔隙度等因素的影响而有所不同。

4.2 数据预处理

经探索性数据分析发现，钻井测量数据中存在异常值“-9999”，并出现在每个数据中的两端，在对“孔隙度”进行可视化时发现孔隙度符合多项式回归分布，而对于“含水合物饱和度”的分布暂未发现相关规律，相关可视化如下：

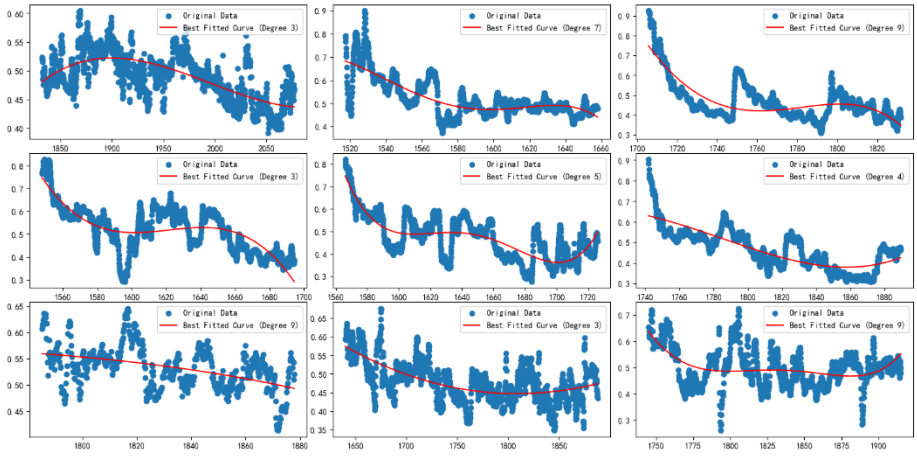

图 3 孔隙度数据可视化图

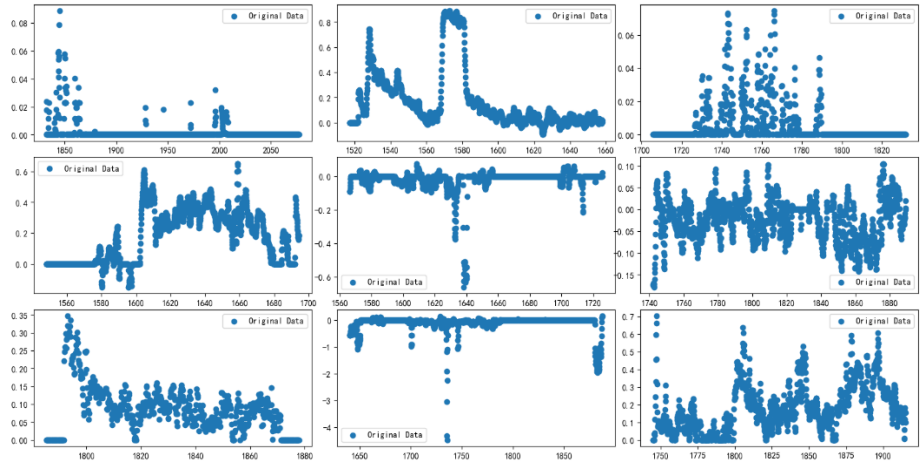

图 4 含水合物饱和度数据可视化图

由于在各个测量数据之间并未发现“-9999”异常值，故在这里采用的策略是将“-9999”视为观测误差，即可能是由于仪器不准确、操作失误或环境因素等引起的，对数据中含有“-9999”的行进行删除。

此外，在附件数据的“含水合物饱和度”中存在负值，水合物饱和度是指单位体积大矿物质中水合物所占的体积分数，根据相关信息查询，含水合物饱和度可以为负值，因此不对其进行处理。

4.3 参数计算

如问题分析中提出的，各个井位的有效面积未知，而产气量因子已经给出，故这里通过建立天然气水合物资源评价指标来估计天然气水合物资源的分布。

评价指标计算：这里资源量定义q，与有效厚度、孔隙度和水合物饱和度有关，公式如下：

$$q = Z \times \phi \times S \tag{1}$$

有效厚度计算：在地质学中的有效厚度与多种因数有关，如实际厚度、孔隙度、含水饱和度、岩石密度、地层构造等有关，在估计天然气水合物资源分布范围时，这里简化相关计算，水合物饱和度非零时被计为有效厚度，通过差分进行累加求和，其中设置阈值去除有效厚度间差距过大的数据（有效厚度并不是连续的），公式如下：

$$Z = \sum_{i=1}^{n} (D_{i+1} - D_i), where(D_{i+1} - D_i < 0.2) \tag{2}$$

其中Z表示有效厚度，D_i表示井位的深度测量数据。

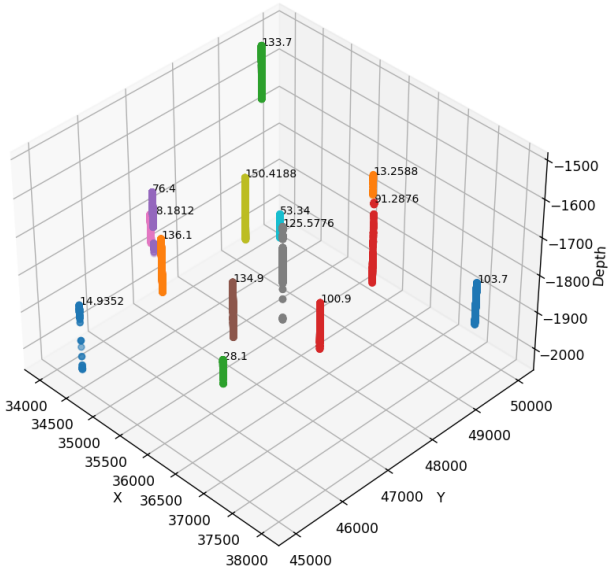

图 5 有效厚度可视化

孔隙度与水合物饱和度计算：孔隙度和水和物饱和度均采用均值代表整体，公式如下：

$$\phi = \frac{\sum_{i=1}^{n} \phi_i}{n} \tag{3}$$

其中，ϕ_i是第i个样本点的孔隙度，n是样本点的总数。

$$S = \frac{\sum_{i=1}^{n} S_i}{n} \tag{4}$$

其中，S_i是第i个样本点的水合物饱和度，n是样本点的总数。

Kriging 算法: Kriging 算法[2]是一种空间插值方法，用于根据已知点的观测值推断未知点的值。在地质勘探中，Kriging 算法可以用于估计地下储层的性质分布，例如孔隙度、含水合物饱和度等，帮助绘制出天然气水合物资源的空间分布图。

其公式为：

$$K(u) = \sum_{i=1}^{n} \lambda_i M(u_i) + \varepsilon(u) \tag{5}$$

$K(u)$其中是未知点u的估计值，$\mathrm{Z(u_i)}$是已知点$\mathrm{u_i}$的观测值，λ_i是用于估计的权重。

问题一模型如下：

$$\begin{cases} q = Z \times \varphi \times S \\ Z = \sum_{i=1}^{n}(D_{i+1} - D_i), where(D_{i+1} - D_i < 0.2) \\ \varphi = \dfrac{\sum_{i=1}^{n}\varphi_i}{n} \\ S = \dfrac{\sum_{i=1}^{n} S_i}{n} \\ K(u) = \sum_{i=1}^{n}\lambda_i Z(u_i) + \varepsilon(u) \end{cases}$$

4.4 问题求解

将模型应用到处理后的数据，结果如下：

井位	有效厚度	孔隙度均值	含水合物饱和度均值	评价指标

W01	14.9352	0.498	0.0178	0.1326
W02	136.1	0.5205	0.178	12.6075
W03	28.1	0.4571	0.0183	0.2354
W04	100.9	0.504	0.02543	12.9308
W05	76.4	0.4897	0.0424	1.5846
W06	134.9	0.4659	0.0266	1.6726
W07	78.1812	0.5331	0.1021	4.2546
W08	125.5776	0.4849	0.2032	12.3753
W09	150.4188	0.5011	0.1972	14.8674
W10	53.34	0.5273	0.1354	3.8098
W11	103.7	0.4278	0.0165	0.7324
W12	13.2588	0.5367	0.0075	0.0535
W13	133.7	0.5427	0.0236	1.7139
W14	91.2876	0.4873	0.02	0.8902

表 1 评价指标计算

根据评价指标，结合地质学中的常用的 Kriging 算法进行空间差值，并结合各个井位的坐标绘制天然气水合物资源分布热力图如图所示：

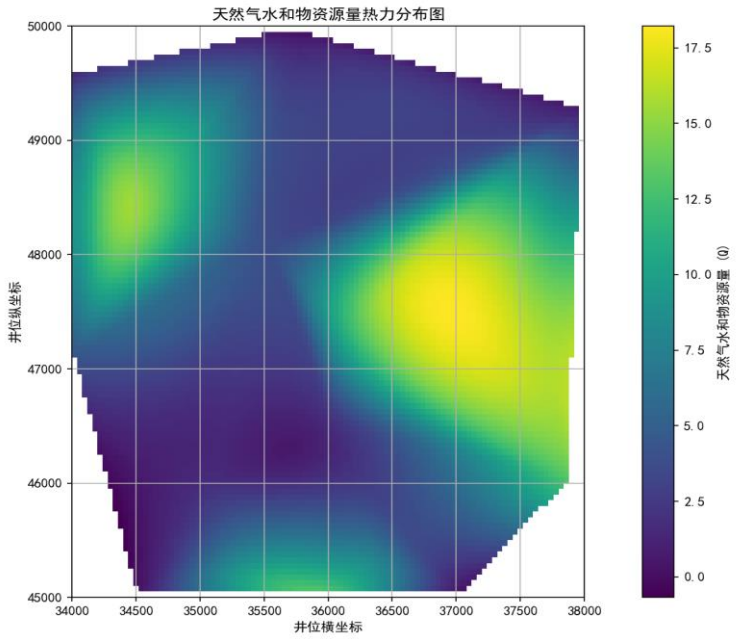

图 6 天然气水合物资源分布热力图

五、 问题二的模型建立与求解

5.1 问题分析

确定研究区域内天然气水合物资源参数有效厚度、地层孔隙度和饱和度的概率分布及其在勘探区域内的变化规律，利用 Q-Q 图以及正态分布进行假设检验，结合问题一中的探索性数据分析确定其变化规律。

5.2 方法介绍

5.2.1Q-Q 图

Q-Q 图（Quantile-Quantile Plot），也称为分位数-分位数图，是一种图形工具，用于比较两个概率分布。它通常用于检验一组数据是否遵循某个特定分布（例如正态分布）。Q-Q 图通过将数据的分位数与某个理论分布的分位数进行比较，从而直观地展示数据分布与理论分布之间的相似性或差异。

5.2.2 Shapiro-Wilk 检验

Shapiro-Wilk： Shapiro-Wilk 检验[3]是一种统计方法，用于检验一个样本是否

来自正态分布。该检验基于排序的检验方法，计算样本的 W 统计量，并与理论上的 W 值进行比较，以判断样本是否来自正态分布。多用于统计分析之前，验证数据是否符合正态分布的假设。

利用 Shapiro-Wilk 进行正太分布假设检验，公式如下：

$$W = \frac{\sum_{i=1}^{n}(a_i x_i)^2}{\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{6}$$

其中，x_i表示样本值按升序排列后的第 i 个值，$\overline{x}$是样本均值，a_i是 Shapiro-Wilk 检验中使用的系数。

5.2.3 Anderson-Darling 检验

Anderson-Darling：Anderson-Darling 检验[4]是一种统计检验方法，用于确定数据集是否服从正态分布。它通过计算数据集的经验累积分布函数与理论上的正态分布累积分布函数之间的差异来评估数据的正态性。多用于数据分布检查以确认数据是否符合正态分布。

利用 Anderson-Darling 进行正太分布假设检验，公式如下：

$$A^2 = -n - \frac{1}{n}\sum_{i=1}^{n}\left[(2i-1)\log\left(F\left(X_{(i)}\right)\right) + (2(n-i)+1)\log\left(1 - F\left(X_{(i)}\right)\right)\right] \tag{7}$$

其中，A^2是 Anderson-Darling 统计量，n是样本大小$X_{(i)}$是按升序排列的第i个观测值，$F(X_{(i)})$是累计分布函数。

5.3 问题求解

5.3.1 有效厚度概率分布及变化规律

经检验，有效厚度数据经 Shapiro-Wilk 检验符合正态分布，有效厚度 QQ 图如下：

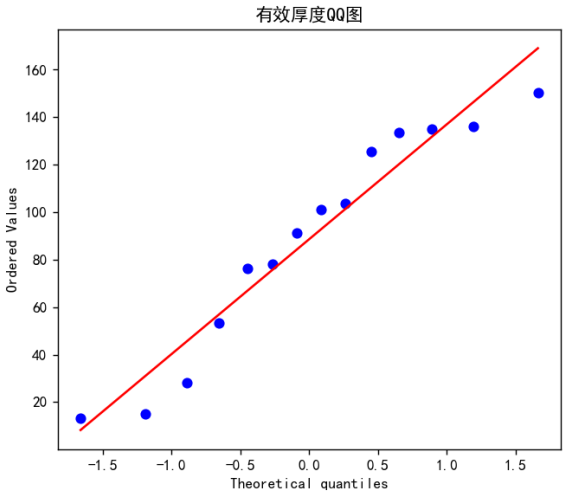

图 7 有效厚度 QQ 图

则有效厚度的概率分布公式如下：

$$f(Z)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{8}$$

将各个井位的深度最高点与深度最低点绘制一个面，三维可视化如下图所示：

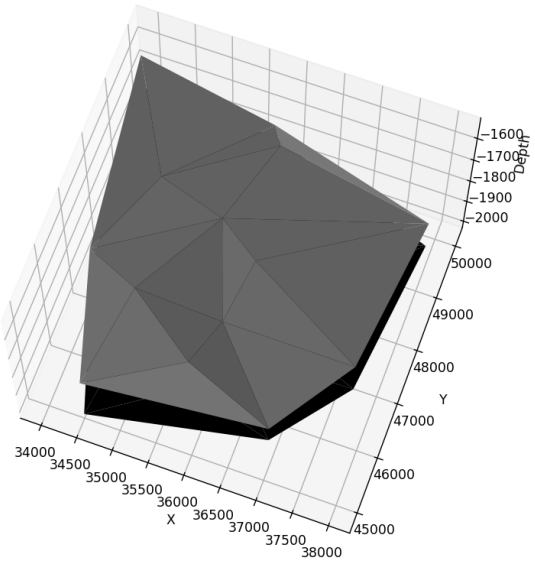

图 8 井位深度三维可视化图

有效厚度在勘探区域内的分布呈现出“三个顶峰”的规律，即天然气水合物资源可能集中在这三个区域中。

5.3.2 孔隙度概率分布及变化规律

相较于 Shapiro-Wilk 检验，Anderson-Darling 检验在检验大量样本是有一定的优势，利用 Anderson-Darling 检验对孔隙度进行假设检验，并绘制 Q-Q 图和直方图直观的了解孔隙度的分布。

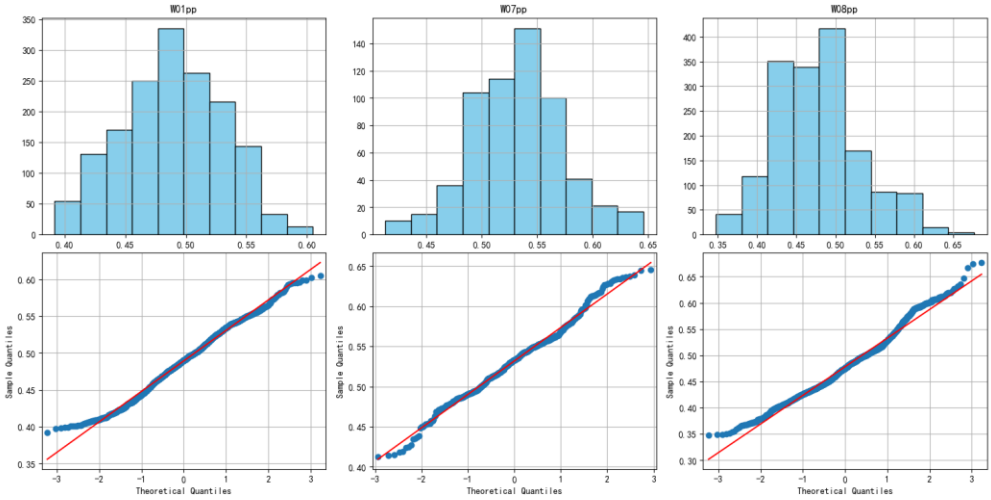

图 9 孔隙度分布和 Q-Q 图

孔隙度的分布公式如下：

$$f(\varphi)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(\varphi-\mu)^2}{2\sigma^2}} \tag{9}$$

在问题一中已经拟采用多项式回归对孔隙度的分布进行拟合，故孔隙度在勘探区域内的变化规律呈现出一定的周期性波动以及随深度逐渐减小的规律。

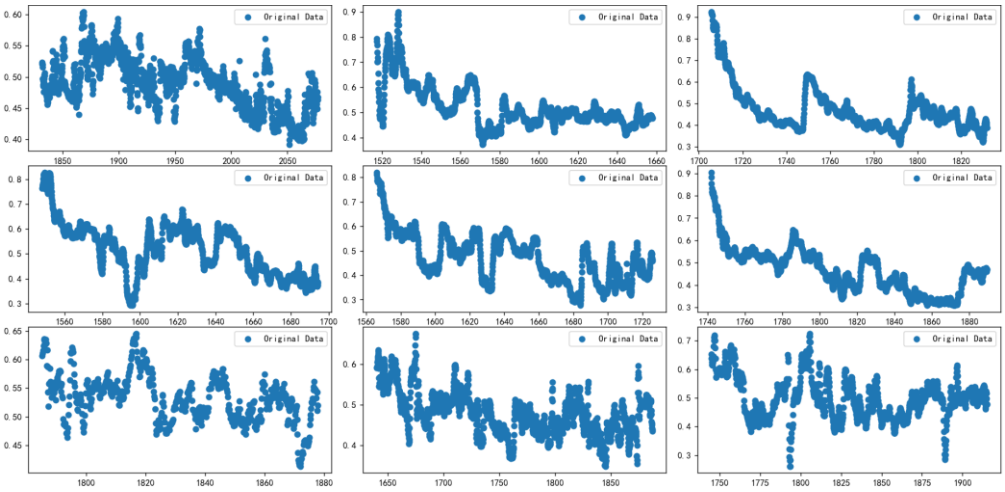

图 10 孔隙度随深度变化

5.3.3 饱和度概率分布及变化规律

同理，利用 Anderson-Darling 检验对饱和度进行假设检验，并绘制 Q-Q 图和直方图直观的了解饱和度的分布。

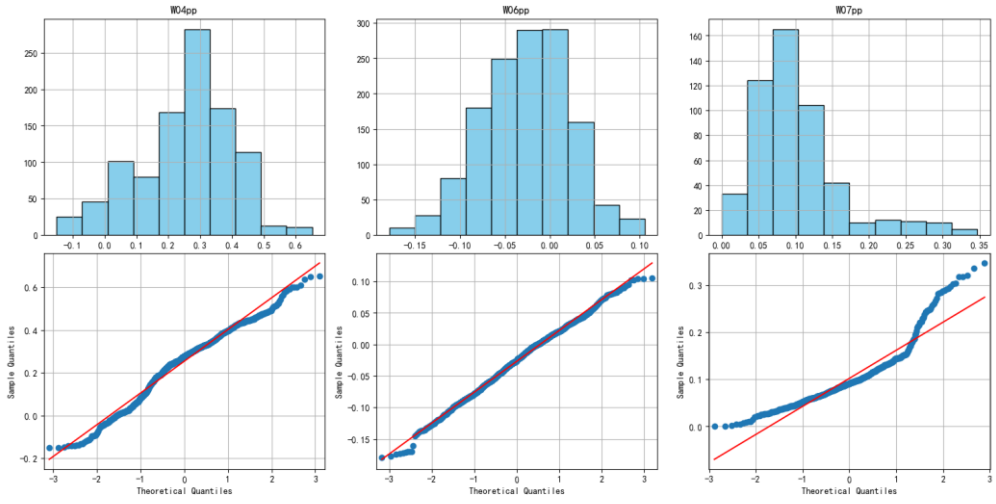

图 11 饱和度分布和 Q-Q 图

饱和度的分布公式如下：

$$f(S) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(S-\mu)^2}{2\sigma^2}} \tag{9}$$

饱和度在勘探区域内随深度变化如下，这里猜测饱和度的变化规律与地质有关，呈现除一个分类的变化规律，即不同类别表现出不同的规律。

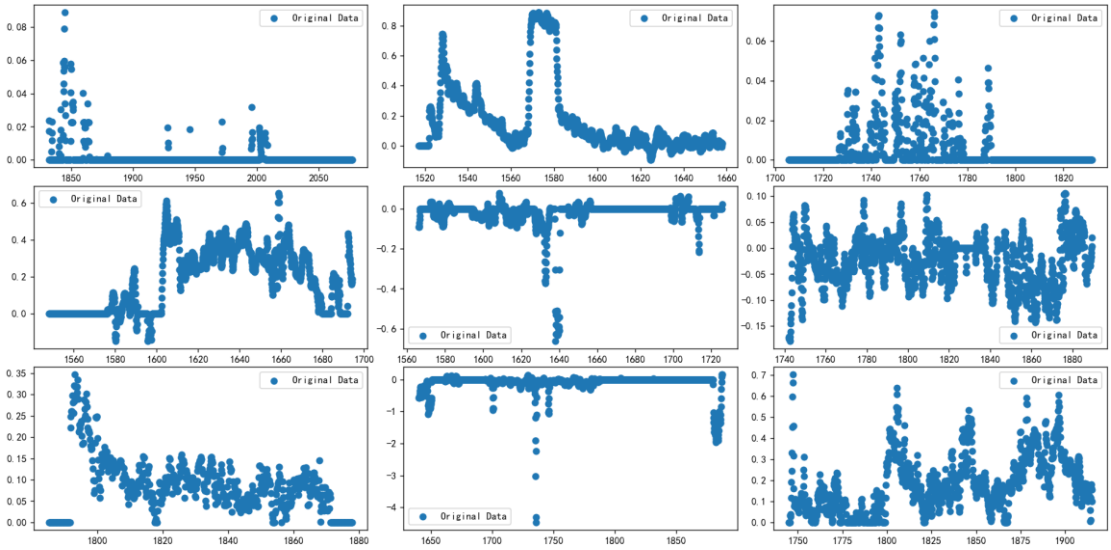

图 12 饱和度随深度变化

六、 问题三的模型建立与求解

6.1 问题分析

确定天然气水合物资的概率分布，可以先通过 Kriging 空间插值估计每个井位的面积进而计算天然气水合物资源量，结合问题二中三个参数的分布规律，对资源量与储层参数的函数进行对数运算，进而使用核密度估计获取天然气水合物资的概率分布，最后利用蒙特卡洛模拟估计天然气水合物资源量。

6.2 概率分布

根据 Kriging 空间插值的面积计算各个井位的天然气水合物资源量，对面积进行可视化如下图所示：

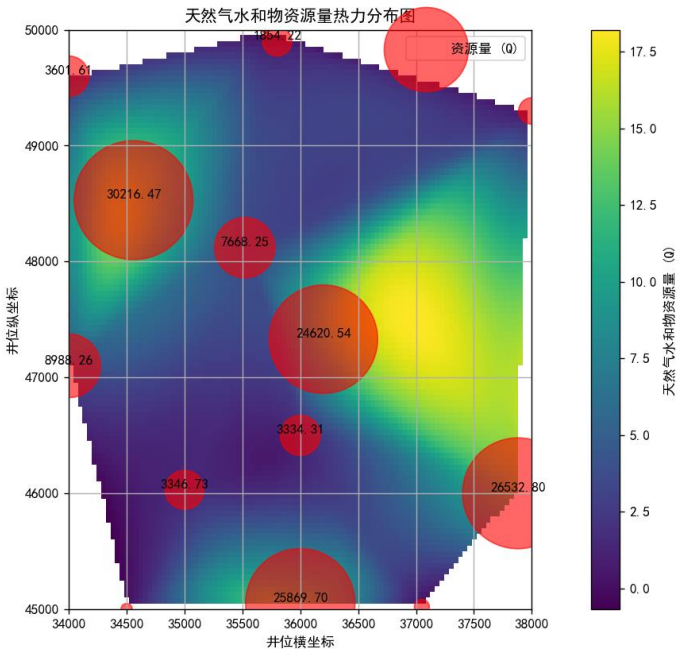

图 13 天然气水合物资源量热力分布图

在问题二中已检验有效厚度、孔隙度、水合物饱和度服从正态分布，而天然气水合物资源服从线性关系，函数公式如下：

$$Q = A \times Z \times \varphi \times S \times E \tag{10}$$

对该函数进行对数化简，公式如下:

$$\ln(Q) = \ln(A) + \ln(Z) + \ln(\varphi) + \ln(S) + E \tag{11}$$

其中，问题二中以确定有效厚度，孔隙度和饱和度服从正太分布，这里假设有效面积服从正态分布，则资源类 Q 服从对数正太分布，公式如下：

$$f(Q|\mu,\sigma) = \frac{1}{Q\sigma\sqrt{2\pi}} e^{-\frac{(\ln(Q)-\mu)^2}{2\sigma^2}} \tag{12}$$

对数转换后的天然气水合物资源的概率密度估计如下图所示：

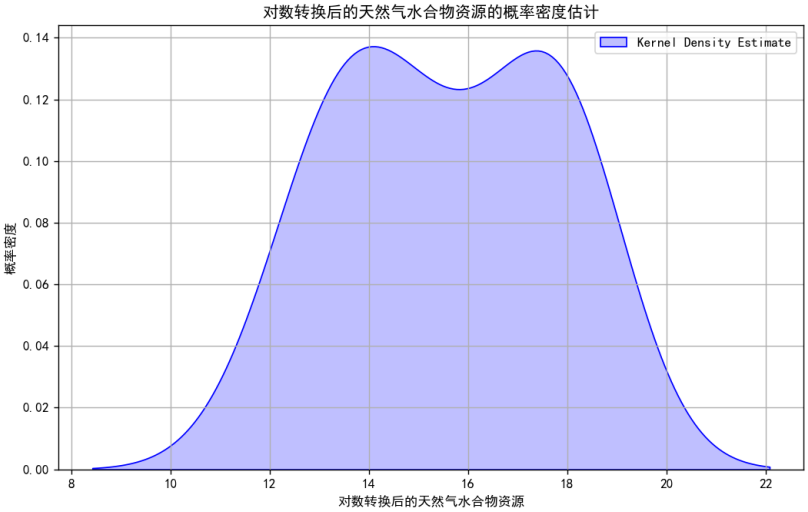

图 14 天然气水合物资源的概率密度估计图

由于样本量（井位）较少的问题，该图大致符合对数正太分布。

6.3 蒙特卡洛模拟

蒙特卡洛模拟是一种基于随机抽样或统计模拟的计算方法，通过重复生成随机样本，计算参数估计量和统计量，进而研究其分布特征。多用于解决难以建立精确数学模型或模型过于复杂的问题。

这里对于有效厚度、孔隙度以及饱和度进行模拟，利用 Shoelace 对勘探区域面积进行计算，设置模拟次数为 10000000 次，进行结果绘制如下图所示：

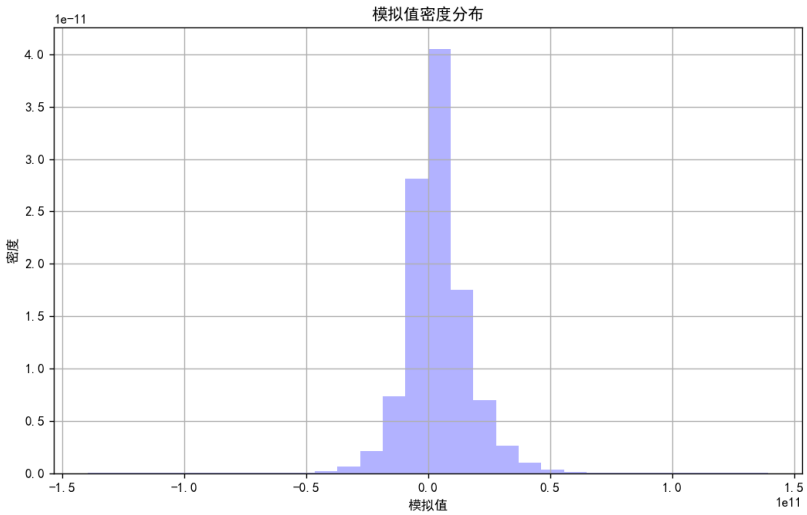

图 15 模拟值密度分布

计算该区域的水合物资源量为 3942056435.7473 立方米。

七、 问题四的模型建立与求解

7.1 问题分析

对本区域储量有个更精细勘查结果，拟在本区域再增加 5 口井，结合前三问的对天然气水合物资源分布可知，可以在评价指标高的地区附近进行拟开井，根据克里金相关插值参数[5]进行求解天然气水合物资源，结合多目标优化 NSGA-II 算法[6]筛选出新增的 5 口井的位置信息。

7.2NSGA-II 算法

NSGA-II（Non-dominated Sorting Genetic Algorithm II）[7]，非支配排序遗传算法 II）是一种多目标优化算法，用于解决具有多个冲突目标的优化问题。它是基于遗传算法的进化算法，旨在寻找一组解决方案，这些解决方案在多个目标下都是非支配的，即不存在其他解决方案在所有目标上都优于它们。

以下是 NSGA-II 算法的主要步骤：

1.初始化种群： 随机生成初始的解决方案集合，即种群。

2.评估适应度： 对种群中的每个个体，计算其在各个目标函数下的适应度值。

3.非支配排序： 将种群中的个体进行非支配排序，将其划分为不同的前沿层级。在这个过程中，每个个体的被支配数量和被支配集合都会被计算出来。

4.计算拥挤度： 对于每个前沿层级中的个体，计算其在目标空间中的拥挤度，用于保持多样性。

5.选择新的种群： 根据非支配排序和拥挤度，从种群中选择下一代个体。这通常通过将前沿层级中的个体按照非支配排序和拥挤度进行排序，然后依次选择最优的个体来完成。

6.遗传操作： 对选定的个体进行交叉和变异操作，生成新的解决方案。

7.重复进化： 重复执行步骤 2 至步骤 6，直到达到预定的停止条件，如达到最大迭代次数或收敛到一定程度。

NSGA-II 算法通过不断地选择优秀的个体、维持种群的多样性和平衡性，以及促进新的解决方案的生成，来搜索出一组在多个目标下都具有良好性能的解决方案集合。这使得 NSGA-II 在解决多目标优化问题时表现出色，被广泛应用于工程设计、资源分配、调度等领域。

7.3 问题求解

首先计算 14 个井位间每两个点的坐标，发现除 12，14 号井位以外均相隔 1000 米以上，故这里假设分布的点以每 500 米为间距进行插入点。

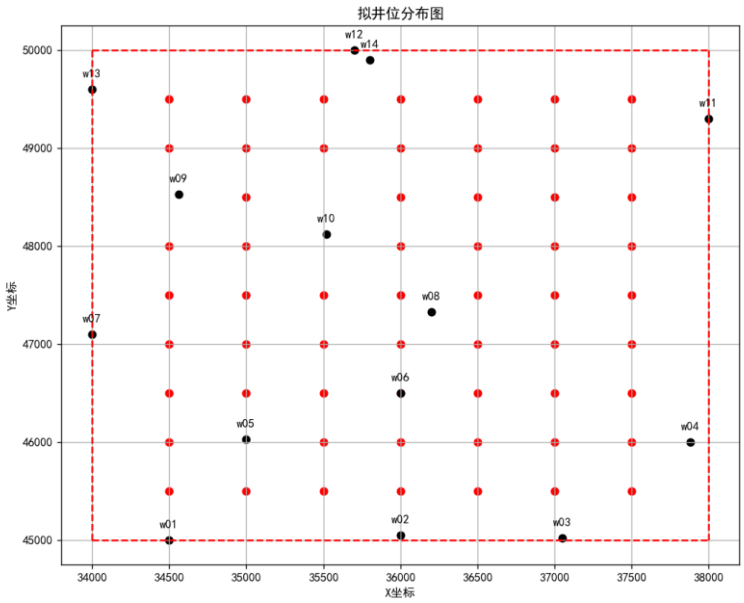

图 16 拟井位分布图

利用第一问中的 Kriging 对拟井位的天然气水合物资源类进行估算，结合 NSGA-II 算法参数调优和迭代，最终选取的 5 个井位如下：

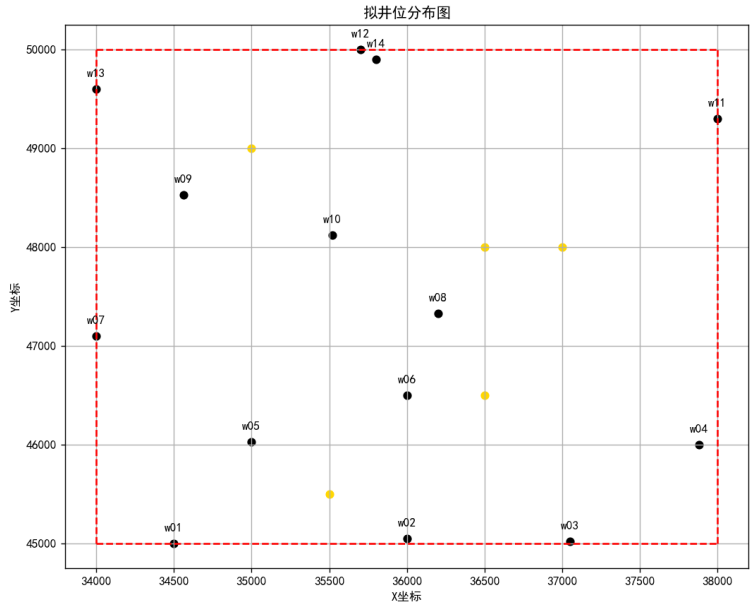

图 17 新井位分布图

井位	X 坐标	Y 坐标
W15	35500	45000
W16	35500	49000
W17	36500	46500
W18	36500	48000
W19	36000	48000

表 2 新井位位置信息表

八、 模型的评价与推广

8.1 模型的优点

Kriging 模型的优点：

空间插值能力强：Kriging 模型能够有效地进行空间插值，对于缺失的空间数据有很好的估计能力。

考虑空间相关性：Kriging 考虑了空间上各点之间的相关性，能够更准确地估计未知位置的数值。

提供不确定性估计：Kriging 不仅可以给出预测值，还能够提供预测的不确定性范围，这对于风险评估和决策制定非常有帮助。

蒙特卡洛模拟的优点：

适用范围广：蒙特卡洛模拟在金融、工程、物理等领域都有广泛的应用，能够处理各种类型的随机变量问题。

能够处理复杂问题：对于复杂的多维、多参数系统，蒙特卡洛模拟能够提供一种灵活的、数值稳健的求解方法。

提供结果的置信区间：通过多次模拟得到的结果，可以给出较为准确的置信区间，帮助决策者更好地理解问题的不确定性。

8.2 模型的缺点

在估计天然气水合物资源，蒙特卡洛方法未能考虑地质空间的资源分布，因此，估计值可能会与实际值有较大偏差。

8.3 模型的推广

本文采用的方法与模型在一定程度上可以估计天然气水合物的资源分布，但在具体的应用场景中，要结合其它参数综合考虑。

九、 参考文献

[1] 国家能源局 国家能源局关于印发《2024 年能源工作指导意见》的通知 https://www.gov.cn/zhengce/zhengceku/202403/content_6941170.html 2024 年 03 月 18 日

[2] 邓显石.一种自适应选取参考点的 Kriging 插值方法研究[J].科学技术创新,2020(29):20-23.2020

[3] 林军,蔡国军,刘松玉,等.基于孔压静力触探的岩土参数正态性检验方法对比[J].吉林大学学报(地球科学版), 51(05):1408-1415.2021

[4] 刘金东,王天铖,陈永红,等.基于 Anderson-Darling 正态性检验的改进方法及应用[J].数学的实践与认识, 53(04):184-193. 2023

[5] 黎嵘繁,钟婷,吴劲,等.基于时空注意力克里金的边坡形变数据插值方法[J].计算机科学, 49(08):33-39. 2022

[6] 高培超,王昊煜,宋长青,等.多目标优化 NSGA 系列算法与地理决策：原理、现状与展望[J].地球信息科学学报, 25(01):25-39. 2023

[7] Peichao Gao;Haoyu Wang;Samuel A. Cushman;Changxiu Cheng;Changqing Song;Sijing Ye. Sustainable land-use optimization using NSGA-II: theoretical and experimental comparisons of improved algorithms. Landscape Ecology. Volume , Issue prepublish.1-16.2020

附录

问题一代码

```
import csv

def annex1_csv(file_path):

    with open(file_path, 'r', encoding='gbk') as file:
```

```
        # 读取文件内容
        file_content = file.read()
    # 去除标题行并分割数据
    lines = file_content.strip().split('\n')[1:]  # 去除标题行
    data = [line.split('\t') for line in lines]  # 分割每行数据
    # 将数据转换为字典列表
    wells_data = []
    for i in range(len(data)):
        if data[i][1] == "":
            a = 2
            b = 3
        else:
            a = 1
            b = 2
        well = {'深度': data[i][0], '孔隙度': data[i][a], '含水合物饱和度': data[i][b]}
```

```
        wells_data.append(well)
    # 写入 CSV 文件
    csv_file_path = f'{file_path[:3]}.csv'
    with open(file=csv_file_path, mode='w',
newline='', encoding='utf-8') as csvfile:
        # 创建 CSV writer 对象
        fieldnames = ['深度', '孔隙度', '含水合物饱和度
']
        writer = csv.DictWriter(csvfile,
fieldnames=fieldnames)
        # 写入表头
        writer.writeheader()
        # 写入数据
        for well in wells_data:
            writer.writerow(well)
            print(well)


 def annex2_csv(file_path):
```

```
    # 使用 with open 语句打开并读取文本文件
    with open(file_path, 'r', encoding='gbk') as file:
        # 读取文件内容
        file_content = file.read()


    # 去除标题行并分割数据
    lines = file_content.strip().split('\n')[1:]  # 去除标题行
    data = [line.split('\t') for line in lines]  # 分割每行数据


    # 将数据转换为字典列表
    wells_data = [{'井名': data[i][0], 'X': int(data[i][1]), 'Y': int(data[i][2])}
                  for i in range(len(data))]


    # 写入 CSV 文件
    csv_file_path = '井位信息.csv'
```

```
    with open(csv_file_path, 'w', newline='',
encoding='gbk') as csvfile:
        # 创建 CSV writer 对象
        fieldnames = ['井名', 'X', 'Y']
        writer = csv.DictWriter(csvfile,
fieldnames=fieldnames)
        # 写入表头
        writer.writeheader()


        # 写入数据
        for well in wells_data:
            writer.writerow(well)
            print(well)


 if __name__ == '__main__':


    num_list = [
        "01", "02", "03", "04", "05", "06", "07",
```

```
        "08", "09", "10", "11", "12", "13", "14"
    ]
    try:
        for num in num_list:
            annex1_csv(f"W{num}_Por_So.txt")

        annex2_csv("附件 2：井位信息.txt")
    except Exception as e:
        print(f"Error: {e}")
```

```
import pandas as pd

num_list = ["01", "02", "03", "04", "05", "06", "07",
            "08", "09", "10", "11", "12", "13", "14"]

for num in num_list:
    file_path = f'W{num}.csv'
```

```
    # 读取 CSV 文件
    data = pd.read_csv(file_path, encoding="utf-8")

    # 过滤掉孔隙度和含水合物饱和度为 -9999 的行
    filtered_data = data[(data['孔隙度'] != -9999) &
(data['含水合物饱和度'] != -9999)]

    # 将过滤后的数据写入新的 CSV 文件
    # 假设新文件的名称为 'filtered_W01.csv'
    output_file_path = f'W{num}p.csv'
    filtered_data.to_csv(output_file_path,
index=False, encoding="utf-8")

    print(f"过滤后的数据已写入 '{output_file_path}'")

 for num in num_list:
    file_path = f'W{num}p.csv'
```

```
    # 读取 CSV 文件
    data = pd.read_csv(file_path, encoding="utf-8")

    # 过滤掉孔隙度和含水合物饱和度为 -9999 的行
    filtered_data = data[(data['孔隙度'] != 0) &
(data['含水合物饱和度'] != 0)]

    # 将过滤后的数据写入新的 CSV 文件
    # 假设新文件的名称为 'filtered_W01.csv'
    output_file_path = f'W{num}pp.csv'
    filtered_data.to_csv(output_file_path,
index=False, encoding="utf-8")

    print(f"过滤后的数据已写入 '{output_file_path}'")

 import pandas as pd
 import matplotlib.pyplot as plt
```

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


# 读取 CSV 文件
file_path = '井位信息.csv'
data = pd.read_csv(file_path, encoding="gbk")


# 绘制散点图
plt.figure(figsize=(10, 8))  # 设置图形大小
plt.scatter(data['X'], data['Y'], color='black')  # 绘制散点图
plt.title('井位分布图')  # 设置图形标题
plt.xlabel('X 坐标')  # 设置 X 轴标签
plt.ylabel('Y 坐标')  # 设置 Y 轴标签
plt.grid(True)  # 显示网格


# 添加井名作为标签
```

```
 for i, txt in enumerate(data['井名']):
    plt.annotate(txt, (data['X'][i], data['Y'][i]),
textcoords="offset points", xytext=(0, 10),
ha='center')


 min_x = data['X'].min()
 max_x = data['X'].max()
 min_y = data['Y'].min()
 max_y = data['Y'].max()


 # 绘制四个边缘的虚线
 plt.plot([min_x, min_x], [min_y, max_y], 'r--') # 左
边界虚线
 plt.plot([max_x, max_x], [min_y, max_y], 'r--') # 右
边界虚线
 plt.plot([min_x, max_x], [min_y, min_y], 'r--') # 下
边界虚线
 plt.plot([min_x, max_x], [max_y, max_y], 'r--') # 上
边界虚线
```

```
# 显示图形
plt.show()
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import griddata


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


# 定义井位位置和资源量
points = [
    (34500, 45000),
    (36000, 45050),
    (37050, 45020),
    (37880, 46000),
    (35000, 46030),
```

```
    (36000, 46500),
    (34000, 47100),
    (36200, 47330),
    (34560, 48530),
    (35520, 48120),
    (38000, 49300),
    (35700, 50000),
    (34000, 49600),
    (35800, 49900)
 ]

 Q_list = [0.1326, 12.6075, 0.2354, 12.9308, 1.5846, 1.6726, 4.2546, 12.3753, 14.8674, 3.8098, 0.7324, 0.0535, 1.7139, 0.8902]

 # 创建网格
 xmin, xmax = min(points, key=lambda x: x[0])[0], max(points, key=lambda x: x[0])[0]
```

```
 ymin, ymax = min(points, key=lambda x: x[1])[1], max(points, key=lambda x: x[1])[1]
 xi = np.linspace(xmin, xmax, 100)
 yi = np.linspace(ymin, ymax, 100)
 xi, yi = np.meshgrid(xi, yi)


 # 插值数据
 zi = griddata(points, Q_list, (xi, yi), method='cubic')


 # 绘制图表
 plt.figure(figsize=(10, 8))
 plt.imshow(zi, extent=[xmin, xmax, ymin, ymax], origin='lower', cmap='viridis')
 plt.colorbar(label='天然气水和物资源量 (Q)')
 plt.title('天然气水和物资源量热力分布图')
 plt.xlabel('井位横坐标')
 plt.ylabel('井位纵坐标')
```

```
plt.grid(True)

plt.show()

import matplotlib.pyplot as plt

import pandas as pd

import warnings



warnings.filterwarnings("ignore")



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

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

plt.rcParams['axes.unicode_minus'] = False



# 设置子图的尺寸

plt.rcParams['figure.figsize'] = (18, 18)



fig, axs = plt.subplots(3, 3, figsize=(18, 18))
```

```
# 读取 CSV 文件
for i in range(1, 10):
    data = pd.read_csv("W0{}p.csv".format(i),
encoding="utf-8")

    X = data[['深度']].values
    y = data['含水合物饱和度'].values

    plt.subplot(3, 3, i)
    plt.scatter(X, y, label='Original Data')
    plt.legend()

# 调整子图之间的间距
plt.subplots_adjust(wspace=0.5, hspace=0.5)
plt.tight_layout()
plt.show()
import matplotlib.pyplot as plt
import pandas as pd
```

```
import warnings


warnings.filterwarnings("ignore")


plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


# 设置子图的尺寸
plt.rcParams['figure.figsize'] = (18, 18)


fig, axs = plt.subplots(3, 3, figsize=(18, 18))


# 读取 CSV 文件
for i in range(1, 10):
    data = pd.read_csv("W0{}p.csv".format(i),
encoding="utf-8")
```

```
    X = data[['深度']].values
    y = data['孔隙度'].values


    plt.subplot(3, 3, i)
    plt.scatter(X, y, label='Original Data')
    plt.legend()


# 调整子图之间的间距
plt.subplots_adjust(wspace=0.5, hspace=0.5)
plt.tight_layout()
plt.show()
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# 读取深度数据并计算有效厚度
depth_data = {}
```

```
depth_sum_list = []
num_list = [
    "01", "02", "03", "04", "05", "06", "07",
    "08", "09", "10", "11", "12", "13", "14"
]

for num in num_list:
    file_path = f'W{num}pp.csv'
    data = pd.read_csv(file_path, encoding="utf-8")
    depth_differences = data['深度'].diff()
    depth_sum  =  depth_differences[depth_differences
<= 0.2].sum()
    depth_sum_list.append(round(depth_sum, 4))
    depth_data[num] = -data['深度'].values

# 创建数据列表
coordinate = [
    ["w01", 34500, 45000],
```

```
    ["w02", 36000, 45050],
    ["w03", 37050, 45020],
    ["w04", 37880, 46000],
    ["w05", 35000, 46030],
    ["w06", 36000, 46500],
    ["w07", 34000, 47100],
    ["w08", 36200, 47330],
    ["w09", 34560, 48530],
    ["w10", 35520, 48120],
    ["w11", 38000, 49300],
    ["w12", 35700, 50000],
    ["w13", 34000, 49600],
    ["w14", 35800, 49900]
]


# 提取横纵坐标和深度数据
x_coords = [point[1] for point in coordinate]
```

```
    y_coords = [point[2] for point in coordinate]



    # 绘制三维散点图并标注有效厚度值
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')



    for i, (x, y, z) in enumerate(zip(x_coords, y_coords,
depth_data.values())):
        ax.scatter(x, y, z, label=f'W{num_list[i]}')
        ax.text(x,  y,  z.max(),  f"{depth_sum_list[i]}",
fontsize=8)



    # 设置坐标轴标签
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Depth')



    # 显示图表
```

```
plt.show()

from mpl_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt



data = [

    ["w01", 34500, 45000, (-1814.9316, -2107.8444)],

    ["w02", 36000, 45050, (-1515, -1693.6)],

    ["w03", 37050, 45020, (-1700, -1865.3)],

    ["w04", 37880, 46000, (-1545, -1727.8)],

    ["w05", 35000, 46030, (-1556, -1760.7)],

    ["w06", 36000, 46500, (-1741, -1931)],

    ["w07", 34000, 47100, (-1764.9444, -1908.3528)],

    ["w08", 36200, 47330, (-1586.484, -1921.1544)],

    ["w09", 34560, 48530, (-1729.8924, -1947.672)],

    ["w10", 35520, 48120, (-1719.9864, -1822.5516)],

    ["w11", 38000, 49300, (-1738, -1950)],

    ["w12", 35700, 50000, (-1729.8924, -1832.1528)],
```

```
    ["w13", 34000, 49600, (-1510.0, -1694.5)],
    ["w14", 35800, 49900, (-1769.9736, -2056.1808)]
]


# 创建一个新的图形和三维子图
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')


# 绘制三维散点图
x, y, z_min, z_max = [], [], [], []
for d in data:
    x.append(d[1])
    y.append(d[2])
    z_min.append(d[3][0])
    z_max.append(d[3][1])


ax.scatter(x, y, z_max, color='b')  # 绘制最高点
```

```
 ax.scatter(x, y, z_min, color='r')  # 绘制最低点


 # 连接最高点和最低点
 for i in range(len(x)):
     ax.plot([x[i], x[i]], [y[i], y[i]], [z_min[i],
z_max[i]], color='k')


 # 设置坐标轴标签
 ax.set_xlabel('X Label')
 ax.set_ylabel('Y Label')
 ax.set_zlabel('Z Label')


 # 显示图形
 plt.show()
 import pandas as pd
 import numpy as np


 num_list = [
```

```
    "01", "02", "03", "04", "05", "06", "07",
    "08", "09", "10", "11", "12", "13", "14"
]


for num in num_list:


    porosity_values1 = []
    porosity_values2 = []
    file_path = f'W{num}pp.csv'


    data = pd.read_csv(file_path, encoding="utf-8")
    porosity_values1.extend(data["孔隙度"].tolist())
    porosity_values2.extend(data["含水合物饱和度"].tolist())


    # 计算孔隙度的平均值
    average_porosity1 = np.mean(porosity_values1)
```

```
    # 计算饱和度的平均值
    average_porosity2 = np.mean(porosity_values2)


    depth_differences = data['深度'].diff()
    depth_differences_list                                  =
depth_differences.tolist()


    depth_sum = 0
    for depth in depth_differences_list[1:]:
        if depth > 0.2:
            continue
        else:
            depth_sum += depth


    print(
        f'W{num}'
        f'有效厚度: {round(depth_sum, 4):<10}'
```

```
        f'孔隙度均值：{round(average_porosity1, 4):<10}'
        f'含水合物饱和度均值：{round(abs(average_porosity2), 4):<10}'
        f'评价指标：{round(depth_sum * average_porosity1 * abs(average_porosity2), 4):<10}'
    )
 import matplotlib.pyplot as plt
 import pandas as pd
 import statsmodels.api as sm


 plt.rcParams['font.family'] = ['sans-serif']
 plt.rcParams['font.sans-serif'] = ['SimHei']
 plt.rcParams['axes.unicode_minus'] = False


 num_list = ["01", "07", "08"]


 fig, axes = plt.subplots(2, 3, figsize=(15, 10))
```

```
 for i, num in enumerate(num_list):
     data = pd.read_csv(f"W{num}p.csv", encoding="utf-
8")
     porosity = data["孔隙度"]

     row = i // 3
     col = i % 3

     axes[0,   col].hist(porosity,   color='skyblue',
edgecolor='black')
     axes[0, col].set_title(f'W{num}pp')
     axes[0, col].grid()

     sm.qqplot(porosity, line='s', ax=axes[1, col])  #
在指定的子图上绘制 QQ 图
     axes[1, col].grid()
```

```
# 调整子图之间的间距

plt.subplots_adjust(wspace=0.5, hspace=0.5)

plt.tight_layout()

plt.show()

import matplotlib.pyplot as plt

import pandas as pd

import warnings



warnings.filterwarnings("ignore")



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

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

plt.rcParams['axes.unicode_minus'] = False



# 设置子图的尺寸

plt.rcParams['figure.figsize'] = (18, 18)
```

```
 fig, axs = plt.subplots(3, 3, figsize=(18, 18))



 # 读取 CSV 文件

 for i in range(1, 10):

     data     =     pd.read_csv("W0{}p.csv".format(i),
encoding="utf-8")



     X = data[['深度']].values

     y = data['孔隙度'].values



     plt.subplot(3, 3, i)

     plt.scatter(X, y, label='Original Data')

     plt.legend()



 # 调整子图之间的间距

 plt.subplots_adjust(wspace=0.5, hspace=0.5)

 plt.tight_layout()

 plt.show()
```

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# 读取深度数据并计算有效厚度
depth_data = {}
depth_sum_list = []
num_list = [
    "01", "02", "03", "04", "05", "06", "07",
    "08", "09", "10", "11", "12", "13", "14"
]


for num in num_list:
    file_path = f'W{num}pp.csv'
    data = pd.read_csv(file_path, encoding="utf-8")
    depth_differences = data['深度'].diff()
    depth_sum  =  depth_differences[depth_differences
<= 0.2].sum()
```

```
    depth_sum_list.append(round(depth_sum, 4))
    depth_data[num] = -data['深度'].values


# 创建数据列表
coordinate = [
    ["w01", 34500, 45000],
    ["w02", 36000, 45050],
    ["w03", 37050, 45020],
    ["w04", 37880, 46000],
    ["w05", 35000, 46030],
    ["w06", 36000, 46500],
    ["w07", 34000, 47100],
    ["w08", 36200, 47330],
    ["w09", 34560, 48530],
    ["w10", 35520, 48120],
    ["w11", 38000, 49300],
    ["w12", 35700, 50000],
```

```
    ["w13", 34000, 49600],
    ["w14", 35800, 49900]
]

# 提取横纵坐标和深度数据
x_coords = [point[1] for point in coordinate]
y_coords = [point[2] for point in coordinate]

# 绘制三维散点图并标注有效厚度值
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

for i, (x, y, z) in enumerate(zip(x_coords, y_coords,
depth_data.values())):
    ax.scatter(x, y, z, label=f'W{num_list[i]}')
    ax.text(x, y, z.max(), f"{depth_sum_list[i]}",
fontsize=8)
```

```
# 设置坐标轴标签

ax.set_xlabel('X')

ax.set_ylabel('Y')

ax.set_zlabel('Depth')



# 显示图表

plt.show()



import matplotlib.pyplot as plt

from mpl_toolkits.mplot3d import Axes3D

import numpy as np



# 定义坐标和厚度数据

coordinate = [

    ["w01", 34500, 45000, -1832.0004, -2008.3272],

    ["w02", 36000, 45050, -1521.8, -1657.9],

    ["w03", 37050, 45020, -1726.4, -1789.7],
```

```
    ["w04", 37880, 46000, -1576.2, -1694.3],
    ["w05", 35000, 46030, -1566.1, -1725.8],
    ["w06", 36000, 46500, -1742.0, -1889.4],
    ["w07", 34000, 47100, -1791.7668, -1871.1672],
    ["w08", 36200, 47330, -1640.1288, -1887.474],
    ["w09", 34560, 48530, -1745.742, -1915.0584],
    ["w10", 35520, 48120, -1727.454, -1790.8524],
    ["w11", 38000, 49300, -1766.0, -1877.9],
    ["w12", 35700, 50000, -1741.0176, -1796.796],
    ["w13", 34000, 49600, -1518.1, -1658.8],
    ["w14", 35800, 49900, -1799.9964, -2021.8908]
]


fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')


# 提取坐标数据
```

```
 x = np.array([point[1] for point in coordinate])

 y = np.array([point[2] for point in coordinate])

 z_bottom = np.array([point[4] for point in
coordinate])

 z_top = np.array([point[3] for point in coordinate])



 # 绘制底部和顶部平面

 ax.plot_trisurf(x, y, z_bottom, color='black')

 ax.plot_trisurf(x, y, z_top, color='grey')



 # 设置坐标轴标签

 ax.set_xlabel('X')

 ax.set_ylabel('Y')

 ax.set_zlabel('Depth')



 plt.show()

 import numpy as np

 import matplotlib.pyplot as plt
```

```
from scipy import stats



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

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

plt.rcParams['axes.unicode_minus'] = False



# 将有效厚度数据放入 NumPy 数组

thickness_data = np.array([14.9352, 136.1, 28.1,
100.9, 76.4, 134.9, 78.1812, 125.5776, 150.4188, 53.34,
103.7, 13.2588, 133.7, 91.2876])



# 绘制 Q-Q 图

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

stats.probplot(thickness_data, dist="norm", plot=plt)

plt.title('有效厚度 QQ 图')

plt.show()



# Shapiro-Wilk 检验
```

```
statistic, p_value = stats.shapiro(thickness_data)

print("Shapiro-Wilk Test:")

print("Statistic:", statistic)

print("P-value:", p_value)

if p_value > 0.05:

    print("数据符合正态分布")

else:

    print("数据不符合正态分布")

import matplotlib.pyplot as plt

import pandas as pd

import statsmodels.api as sm



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

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

plt.rcParams['axes.unicode_minus'] = False



num_list = ["04", "06", "07"]
```

```
 fig, axes = plt.subplots(2, 3, figsize=(15, 10))


 for i, num in enumerate(num_list):
     data = pd.read_csv(f"W{num}pp.csv",
encoding="utf-8")
     porosity = data["含水合物饱和度"]


     row = i // 3
     col = i % 3


     axes[0, col].hist(porosity, color='skyblue',
edgecolor='black')
     axes[0, col].set_title(f'W{num}pp')
     axes[0, col].grid()


     sm.qqplot(porosity, line='s', ax=axes[1, col])  #
在指定的子图上绘制 QQ 图
```

```
    axes[1, col].grid()


# 调整子图之间的间距
plt.subplots_adjust(wspace=0.5, hspace=0.5)
plt.tight_layout()
plt.show()


import matplotlib.pyplot as plt
import pandas as pd
import warnings


warnings.filterwarnings("ignore")


plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False
```

```
# 设置子图的尺寸
plt.rcParams['figure.figsize'] = (18, 18)


fig, axs = plt.subplots(3, 3, figsize=(18, 18))


# 读取 CSV 文件
for i in range(1, 10):
    data = pd.read_csv("W0{}p.csv".format(i),
encoding="utf-8")


    X = data[['深度']].values
    y = data['含水合物饱和度'].values


    plt.subplot(3, 3, i)
    plt.scatter(X, y, label='Original Data')
    plt.legend()


# 调整子图之间的间距
```

```
plt.subplots_adjust(wspace=0.5, hspace=0.5)

plt.tight_layout()

plt.show()
```

问题三代码

```
import numpy as np

import matplotlib.pyplot as plt

from scipy.interpolate import griddata



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

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

plt.rcParams['axes.unicode_minus'] = False



# 定义井位位置和资源量

points = [

    (34500, 45000),

    (36000, 45050),

    (37050, 45020),
```

```
    (37880, 46000),
    (35000, 46030),
    (36000, 46500),
    (34000, 47100),
    (36200, 47330),
    (34560, 48530),
    (35520, 48120),
    (38000, 49300),
    (35700, 50000),
    (34000, 49600),
    (35800, 49900)
 ]

 Q_list = [0.1326, 12.6075, 0.2354, 12.9308, 1.5846, 1.6726, 4.2546, 12.3753, 14.8674, 3.8098, 0.7324, 0.0535, 1.7139, 0.8902]

 # 创建网格
```

```
 xmin, xmax = min(points, key=lambda x: x[0])[0], max(points, key=lambda x: x[0])[0]
 ymin, ymax = min(points, key=lambda x: x[1])[1], max(points, key=lambda x: x[1])[1]
 xi = np.linspace(xmin, xmax, 100)
 yi = np.linspace(ymin, ymax, 100)
 xi, yi = np.meshgrid(xi, yi)


 # 插值数据
 zi = griddata(points, Q_list, (xi, yi), method='cubic')


 # 绘制图表
 plt.figure(figsize=(10, 8))


 # 绘制资源量分布的热力图
 plt.imshow(zi, extent=[xmin, xmax, ymin, ymax], origin='lower', cmap='viridis')
```

```
 plt.colorbar(label='天然气水和物资源量 (Q)')
 plt.title('天然气水和物资源量热力分布图')
 plt.xlabel('井位横坐标')
 plt.ylabel('井位纵坐标')


 # 绘制点的资源量分布
 size_scale = 500  # 调整散点的大小比例
 plt.scatter([p[0] for p in points], [p[1] for p in
points], s=[q * size_scale for q in Q_list], c='red',
alpha=0.6, label='资源量 (Q)')
 plt.legend()


 plt.grid(True)


 # 输出每个点周围的面积
 for i, point in enumerate(points):
     x, y = point
```

```
    x_idx = np.abs(xi[0] - x).argmin()  # 找到最接近点的 x 坐标索引
    y_idx = np.abs(yi[:, 0] - y).argmin()  # 找到最接近点的 y 坐标索引


    # 计算面积
    dx = xi[0, 1] - xi[0, 0]
    dy = yi[1, 0] - yi[0, 0]
    area = round(zi[y_idx, x_idx] * dx * dy, 4)  # 使用插值后的值计算面积
    print(area)
    if not np.isnan(area):  # 添加条件判断，只有当面积不为 NaN 时才进行输出
        plt.text(point[0], point[1], f"{area:.2f}", ha='center', va='bottom')


plt.show()
import pandas as pd
```

```
import matplotlib.pyplot as plt



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

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



# 读取 CSV 文件

file_path = '井位信息.csv'

data = pd.read_csv(file_path, encoding="gbk")



# 绘制散点图

plt.figure(figsize=(10, 8))

plt.scatter(data['X'], data['Y'], color='black')

plt.title('井位分布图')

plt.xlabel('X 坐标')

plt.ylabel('Y 坐标')

plt.grid(True)
```

```
 # 添加井名作为标签
 for i, txt in enumerate(data['井名']):
     plt.annotate(txt, (data['X'][i], data['Y'][i]),
textcoords="offset points", xytext=(0, 10),
ha='center')


 # 定义坐标点
 points = {
     'w01': (34500, 45000),
     'w03': (37050, 45020),
     'w04': (37880, 46000),
     'w11': (38000, 49300),
     'w14': (35800, 49900),
     'w12': (35700, 50000),
     'w13': (34000, 49600),
     'w07': (34000, 47100),
 }
```

```
# 提取点的坐标并添加第一个点以闭合多边形
x_coords = [points[key][0] for key in points]
y_coords = [points[key][1] for key in points]
x_coords.append(x_coords[0])
y_coords.append(y_coords[0])

# 计算多边形的面积
area = abs(0.5 * sum((x_coords[i] * y_coords[i+1] - x_coords[i+1] * y_coords[i]) for i in range(len(points))))

print(f"多边形的面积为: {area} 平方米")

# 绘制多边形的虚线
plt.plot(x_coords, y_coords, 'r--', lw=2)

# 显示图形
plt.show()
```

```
from scipy.stats import shapiro, norm
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np


plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


data = {
    "W02": [136.1, 25869.6965, 0.5205, 0.178, 155],
    "W04": [100.9, 26532.7973, 0.504, 0.2543, 155],
    "W05": [76.4, 3346.7264, 0.4897, 0.0424, 155],
    "W06": [134.9, 3334.3096, 0.4659, 0.0266, 155],
    "W07": [78.1812, 8988.2581, 0.5331, 0.1021, 155],
    "W08":  [125.5766,  24620.5407,  0.4849,  0.2032,
155],
```

```
    "W09": [150.4188, 30216.4688, 0.5011, 0.1972,
155],
    "W10": [53.34, 7668.2471, 0.5273, 0.1354, 155],
    "W13": [133.7, 3601.6148, 0.5427, 0.0236, 155],
    "W14": [91.2876, 1854.2212, 0.4873, 0.02, 155],
 }


 # 计算水资源的乘积
 water_resources = [np.prod(values) for values in
data.values()]


 # 对数据进行对数转换
 log_water_resources = np.log(water_resources)


 # 进行 Shapiro-Wilk 检验以判断数据是否为正态分布
 stat, p = shapiro(log_water_resources)
 print(f"Shapiro-Wilk test statistic: {stat}, p-value:
{p}")
```

```
# 如果 p 值大于 0.05，我们不能拒绝数据来自正态分布的假设
if p > 0.05:
    print("数据可能遵循对数正态分布。")
else:
    print("数据可能不遵循对数正态分布。")

# 绘制对数转换后的数据及其正态分布曲线的线图

# 计算正态分布的均值和标准差
mu, std = norm.fit(log_water_resources)

plt.figure(figsize=(10, 6))
sns.kdeplot(log_water_resources,          fill=True,
color='b', label='Kernel Density Estimate')

# 标记
plt.title("对数转换后的天然气水合物资源的概率密度估计")
```

```
plt.xlabel("对数转换后的天然气水合物资源")

plt.ylabel("概率密度")

plt.legend()

plt.grid(True)

plt.show()

import matplotlib.pyplot as plt

import numpy as np



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

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

plt.rcParams['axes.unicode_minus'] = False



# 已知常数 A 和 E

A = 17483200

E = 155



# 从 14 个样本中得到的 Z，ϕ，S 的样本均值和标准差
```

```
mu_Z = 88.628514

std_Z = 46.635309

mu_phi = 0.498293

std_phi = 0.088779

mu_S = 0.032961

std_S = 0.087437



# 模拟参数

n_simulations = 10000000  # 模拟次数



# 进行蒙特卡洛模拟

Z_sim = np.random.normal(mu_Z, std_Z, n_simulations)

phi_sim = np.random.normal(mu_phi, std_phi,
n_simulations)

S_sim = np.random.normal(mu_S, std_S, n_simulations)



# 计算 q 的模拟值

q_sim = A * E * (Z_sim * phi_sim * S_sim)
```

```
# 分析模拟结果
mean_q = round(np.mean(q_sim), 4)


# 打印结果
print(f"天然气水合物资源估计为: {mean_q}")


plt.figure(figsize=(10, 6))
plt.title('模拟值密度分布')
plt.xlabel('模拟值')
plt.ylabel('密度')
plt.hist(q_sim, bins=30, density=True, alpha=0.3,
color='blue', label='Simulated data')
plt.grid()
plt.show()
```

问题四代码

```
import random
from deap import base, creator, tools
```

```
# 定义适应度函数（这里假设资源类越高适应度越高）
def evaluate(individual):
    return sum(individual),  # 返回一个元组形式的结果，因为 NSGA-II 要最小化目标函数，这里只有一个目标，所以返回的是一个包含一个元素的元组


# 创建适应度最小化的 FitnessMin 类
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))

# 创建个体的 Individual 类，继承于 list，并包含适应度属性
creator.create("Individual", list, fitness=creator.FitnessMin)

# 初始化 toolbox
```

```
toolbox = base.Toolbox()


# 注册工具函数
toolbox.register("attr_bool", random.randint, 0, 1)
# 二进制编码，假设每个点是否选择开井
toolbox.register("individual", tools.initRepeat,
creator.Individual, toolbox.attr_bool, n=5)  # 假设有 5
个点，即 5 个可能的开井位置
toolbox.register("population", tools.initRepeat,
list, toolbox.individual)


# 注册遗传操作
toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)  # 交叉操
作
toolbox.register("mutate", tools.mutFlipBit,
indpb=0.05)  # 突变操作
toolbox.register("select", tools.selNSGA2)  # 选择操
作
```

```
# 设置种群大小和迭代次数
POP_SIZE = 100
NGEN = 50

# 创建种群
pop = toolbox.population(n=POP_SIZE)

# 迭代进化
for gen in range(NGEN):
    offspring = toolbox.select(pop, len(pop))
    offspring = [toolbox.clone(ind) for ind in offspring]

    # 对选定的个体进行交叉和变异操作
    for ind1, ind2 in zip(offspring[::2], offspring[1::2]):
        if random.random() < 0.5:
```

```
            toolbox.mate(ind1, ind2)
            del ind1.fitness.values
            del ind2.fitness.values

    for mutant in offspring:
        if random.random() < 0.2:
            toolbox.mutate(mutant)
            del mutant.fitness.values

    # 重新评价被修改的个体
    invalid_ind = [ind for ind in offspring if not
ind.fitness.valid]
    fitnesses     =     toolbox.map(toolbox.evaluate,
invalid_ind)
    for ind, fit in zip(invalid_ind, fitnesses):
        ind.fitness.values = fit

    # 更新种群
```

```
    pop[:] = offspring


# 输出最优解
best_ind = tools.selBest(pop, 1)[0]
print("Best individual is ", best_ind)
print("with fitness ", best_ind.fitness.values)
import pandas as pd
import matplotlib.pyplot as plt


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


# 读取 CSV 文件
file_path = '井位信息.csv'
data = pd.read_csv(file_path, encoding="gbk")


# 绘制散点图
```

```
 plt.figure(figsize=(10, 8))  # 设置图形大小
 plt.scatter(data['X'], data['Y'], color='black')  # 绘制散点图
 plt.title('井位分布图')  # 设置图形标题
 plt.xlabel('X 坐标')  # 设置 X 轴标签
 plt.ylabel('Y 坐标')  # 设置 Y 轴标签
 plt.grid(True)  # 显示网格


 # 添加井名作为标签
 for i, txt in enumerate(data['井名']):
     plt.annotate(txt, (data['X'][i], data['Y'][i]), textcoords="offset points", xytext=(0, 10), ha='center')


 min_x = data['X'].min()
 max_x = data['X'].max()
 min_y = data['Y'].min()
 max_y = data['Y'].max()
```

```
 # 绘制四个边缘的虚线
 plt.plot([min_x, min_x], [min_y, max_y], 'r--')  # 左边界虚线
 plt.plot([max_x, max_x], [min_y, max_y], 'r--')  # 右边界虚线
 plt.plot([min_x, max_x], [min_y, min_y], 'r--')  # 下边界虚线
 plt.plot([min_x, max_x], [max_y, max_y], 'r--')  # 上边界虚线

 # 显示图形
 plt.show()
 import math



 def euclidean_distance(point1, point2):
     x1, y1 = point1[1], point1[2]
```

```
    x2, y2 = point2[1], point2[2]
    distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) **
2)
    return distance



# 定义坐标数据
coordinates = [
    ["w01", 34500, 45000],
    ["w02", 36000, 45050],
    ["w03", 37050, 45020],
    ["w04", 37880, 46000],
    ["w05", 35000, 46030],
    ["w06", 36000, 46500],
    ["w07", 34000, 47100],
    ["w08", 36200, 47330],
    ["w09", 34560, 48530],
    ["w10", 35520, 48120],
```

```
    ["w11", 38000, 49300],
    ["w12", 35700, 50000],
    ["w13", 34000, 49600],
    ["w14", 35800, 49900]
]

# 计算每对点之间的距离
for i in range(len(coordinates)):
    for j in range(i + 1, len(coordinates)):
        point1 = coordinates[i]
        point2 = coordinates[j]
        distance = round(euclidean_distance(point1, point2))
        print(f"距离 {point1[0]} 和 {point2[0]} 的距离为: {distance}")
import numpy as np
import matplotlib.pyplot as plt
from scipy.interpolate import griddata
```

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


# 定义井位位置和资源量
points = [
    (34500, 45000),
    (36000, 45050),
    (37050, 45020),
    (37880, 46000),
    (35000, 46030),
    (36000, 46500),
    (34000, 47100),
    (36200, 47330),
    (34560, 48530),
    (35520, 48120),
    (38000, 49300),
```

```
    (35700, 50000),
    (34000, 49600),
    (35800, 49900)
]

q_list = [0.1326, 12.6075, 0.2354, 12.9308, 1.5846,
          1.6726, 4.2546, 12.3753, 14.8674, 3.8098,
          0.7324, 0.0535, 1.7139, 0.8902]

# 创建网格
xmin, xmax = min(points, key=lambda x: x[0])[0], max(points, key=lambda x: x[0])[0]
ymin, ymax = min(points, key=lambda x: x[1])[1], max(points, key=lambda x: x[1])[1]
xi = np.linspace(xmin, xmax, 100)
yi = np.linspace(ymin, ymax, 100)
xi, yi = np.meshgrid(xi, yi)
```

```
 # 插值数据
 zi = griddata(points, q_list, (xi, yi), method='cubic')

 # 绘制图表
 plt.figure(figsize=(10, 8))
 plt.imshow(zi, extent=[xmin, xmax, ymin, ymax], origin='lower', cmap='viridis')
 plt.colorbar(label='天然气水和物资源量 (Q)')
 plt.title('天然气水和物资源量热力分布图')
 plt.xlabel('井位横坐标')
 plt.ylabel('井位纵坐标')
 plt.grid(True)
 plt.show()
 import pandas as pd
 import matplotlib.pyplot as plt

 plt.rcParams['font.family'] = ['sans-serif']
```

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



# 读取 CSV 文件
file_path = '井位信息.csv'
data = pd.read_csv(file_path, encoding="gbk")



data_x = [
    34500, 34500, 34500, 34500, 34500, 34500, 34500,
34500,
    35000, 35000, 35000, 35000, 35000, 35000, 35000,
35000,
    35500, 35500, 35500, 35500, 35500, 35500, 35500,
    36000, 36000, 36000, 36000, 36000, 36000, 36000,
36000, 36000,
    36500, 36500, 36500, 36500, 36500, 36500, 36500,
36500, 36500,
    37000, 37000, 37000, 37000, 37000, 37000, 37000,
37000, 37000,
```

```
    37500, 37500, 37500, 37500, 37500, 37500, 37500,
37500, 37500,



          ]
 data_y = [
    45500, 46000, 46500, 47000, 47500, 48000, 49000,
49500,
    45500, 46500, 47000, 47500, 48000, 48500, 49000,
49500,
    45500, 46000, 46500, 47000, 47500, 49000, 49500,
    45500, 46000, 46500, 47000, 47500, 48000, 48500,
49000, 49500,
    45500, 46000, 46500, 47000, 47500, 48000, 48500,
49000, 49500,
    45500, 46000, 46500, 47000, 47500, 48000, 48500,
49000, 49500,
    45500, 46000, 46500, 47000, 47500, 48000, 48500,
49000, 49500,
          ]
```

```
 # 绘制散点图
 plt.figure(figsize=(10, 8))  # 设置图形大小
 plt.scatter(data_x, data_y, color='red')  # 绘制散点图
 plt.scatter(data['X'], data['Y'], color='black')  # 绘制散点图
 plt.title('拟井位分布图')  # 设置图形标题
 plt.xlabel('X 坐标')  # 设置 X 轴标签
 plt.ylabel('Y 坐标')  # 设置 Y 轴标签
 plt.grid(True)  # 显示网格

 # 添加井名作为标签
 for i, txt in enumerate(data['井名']):
     plt.annotate(txt, (data['X'][i], data['Y'][i]), textcoords="offset points", xytext=(0, 10), ha='center')

 min_x = data['X'].min()
 max_x = data['X'].max()
```

```
min_y = data['Y'].min()

max_y = data['Y'].max()



# 绘制四个边缘的虚线

plt.plot([min_x, min_x], [min_y, max_y], 'r--')  # 左边界虚线

plt.plot([max_x, max_x], [min_y, max_y], 'r--')  # 右边界虚线

plt.plot([min_x, max_x], [min_y, min_y], 'r--')  # 下边界虚线

plt.plot([min_x, max_x], [max_y, max_y], 'r--')  # 上边界虚线



# 显示图形

plt.show()

import pandas as pd

import matplotlib.pyplot as plt
```

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


# 读取 CSV 文件
file_path = '井位信息.csv'
data = pd.read_csv(file_path, encoding="gbk")


data_x = [
    35500, 36500, 35000, 36500, 37000
          ]
data_y = [
    45500, 46500, 49000, 48000, 48000
          ]
# 绘制散点图
plt.figure(figsize=(10, 8))  # 设置图形大小
plt.scatter(data_x, data_y, color='gold')  # 绘制散点
图
```

```
 plt.scatter(data['X'], data['Y'], color='black')  # 绘制散点图
 plt.title('拟井位分布图')  # 设置图形标题
 plt.xlabel('X 坐标')  # 设置 X 轴标签
 plt.ylabel('Y 坐标')  # 设置 Y 轴标签
 plt.grid(True)  # 显示网格


 # 添加井名作为标签
 for i, txt in enumerate(data['井名']):
     plt.annotate(txt, (data['X'][i], data['Y'][i]), textcoords="offset points", xytext=(0, 10), ha='center')


 min_x = data['X'].min()
 max_x = data['X'].max()
 min_y = data['Y'].min()
 max_y = data['Y'].max()
```

```
 # 绘制四个边缘的虚线
 plt.plot([min_x, min_x], [min_y, max_y], 'r--')  # 左边界虚线
 plt.plot([max_x, max_x], [min_y, max_y], 'r--')  # 右边界虚线
 plt.plot([min_x, max_x], [min_y, min_y], 'r--')  # 下边界虚线
 plt.plot([min_x, max_x], [max_y, max_y], 'r--')  # 上边界虚线


 # 显示图形
 plt.show()
```