目 录

正演模拟

聚类分析

正演模拟

一、基本理论介绍

地震正演模拟是地下介质结构和参数已知的情况下，利用数值计算的方法来研究地震波在地下介质中的传播规律，从而获得理论地震记录的一种方法。

地震模型正演的基础在于不同的岩层具有不同的速度和密度，两者的乘积为波阻抗。波阻抗的差异会产生反射系数。假设地震波是垂直入射的，则可计算法线入射的反射系数：

$$R = \frac{\rho_1 v_1 - \rho_2 v_2}{\rho_1 v_1 + \rho_2 v_2}$$

其中，R为反射系数，$\rho_1 v_1$，$\rho_2 v_2$为相邻两层的波阻抗。

地震子波是地震模型正演中的一个重要参数，选择合适的地震子波也是决定最终的正演结果与实际地震记录吻合与否的关键。地震子波是一段具有确定的起始时间、能量有限且有一定延续长度的信号，是地震记录中的基本单元。一个地震子波一般有 2 至 3 个相位的延续长度，大约有 90ms 左右。

实际中的地震子波是一个很复杂的问题，因为地震子波与地层岩石性质有关，地层岩石性质本身就是一个复杂体。为了研究方便，仍需要对地震子波进行模拟，目前普遍认为雷克提出的地震子波数学模型具有广泛的代表性，即称雷克子波，公式及样图如下所示。

$$f(t) = (1 - 2(\pi f_m t)^2)e^{-(\pi f_m t)^2}$$

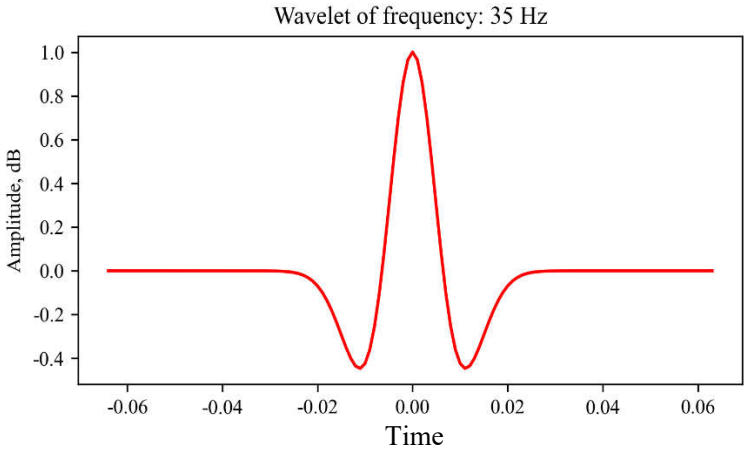


图 1 35Hz 雷克子波

二、算法流程

本文采用一种在地震反演中常见的正演模拟方法，即，首先由层位数据生成地层骨架，再利用井数据插值得到井插值模型，最后转换为波阻抗模型后与地震子波褶积得到模拟地震记录。

Step1.输入地震层位信息；

Step2.插值创建地震骨架；

Step3.向模型导入速度、密度信息，创建波阻抗模型；

Step4.计算得到反射系数序列；

Step5.给定子波，子波与反射系数序列褶积后得到地震模型；

Step6.向模型中加入噪声与衰减。

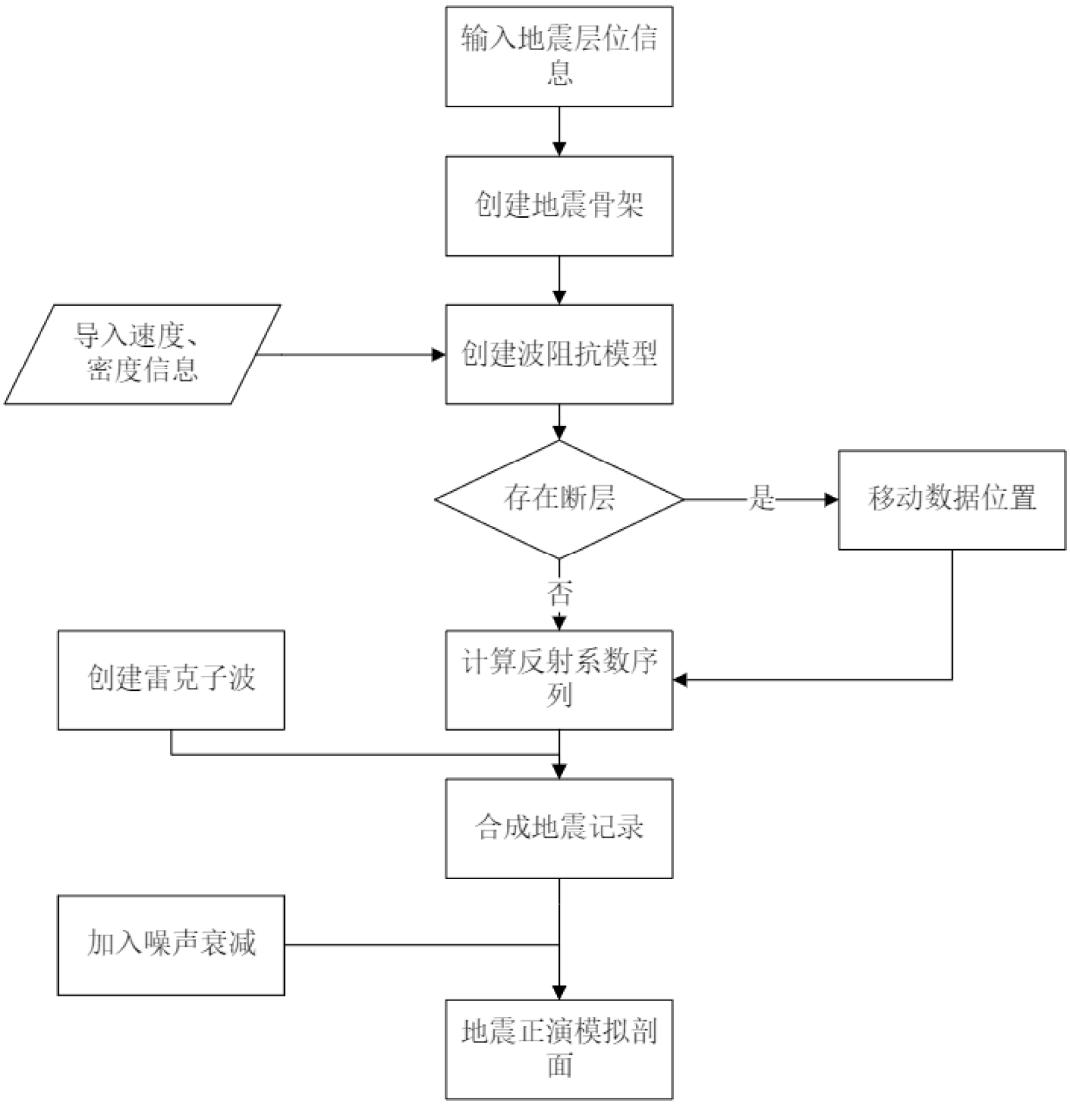
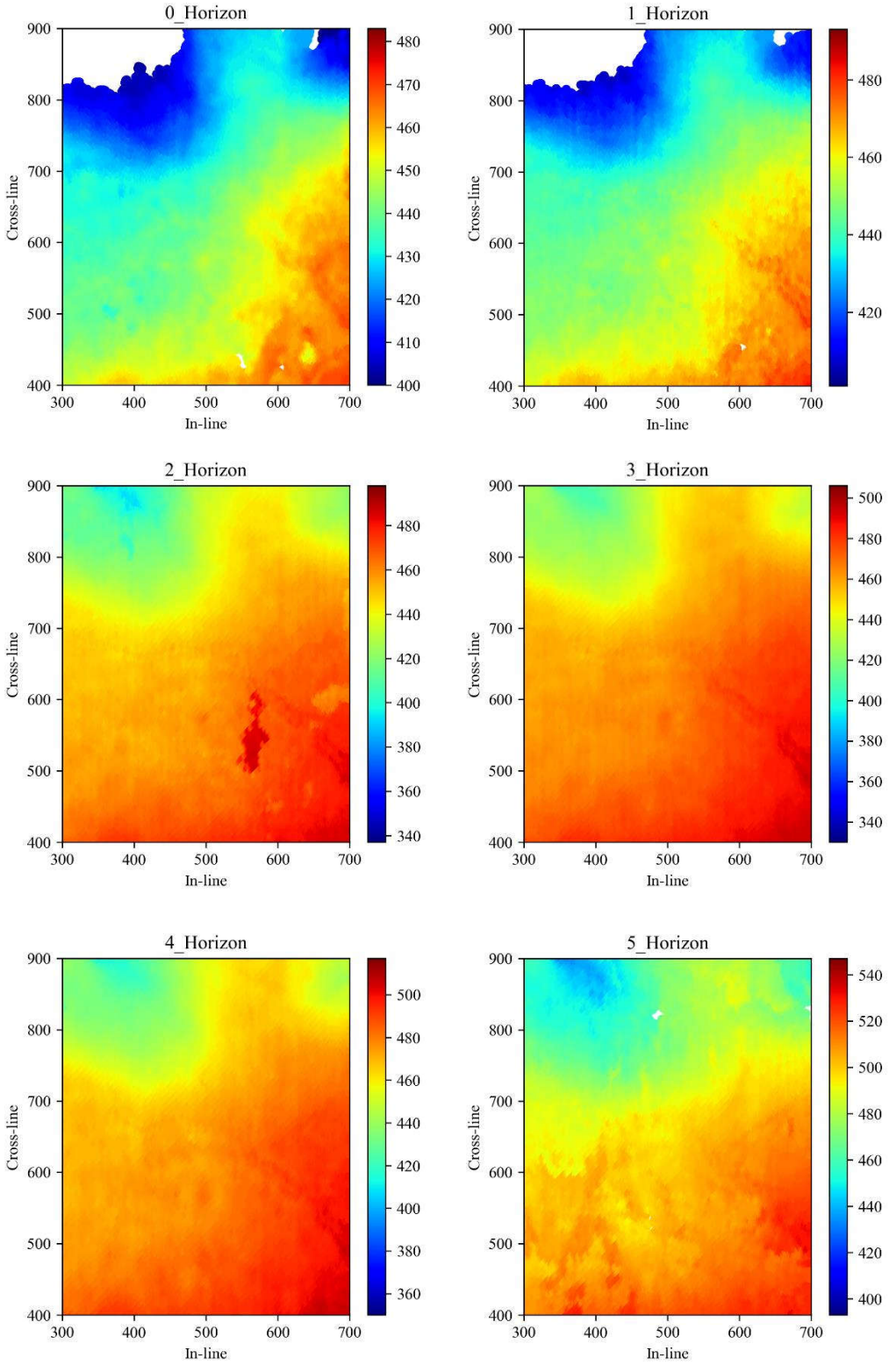


图 2 正演模拟算法流程

三、程序运行结果展示

3.1 层位信息展示

该工区数据范围为 inLine 道号是 301 到 1249，crossLine 道号是 101，746，共含有七个层位，各层位沿层切片如下图所示：



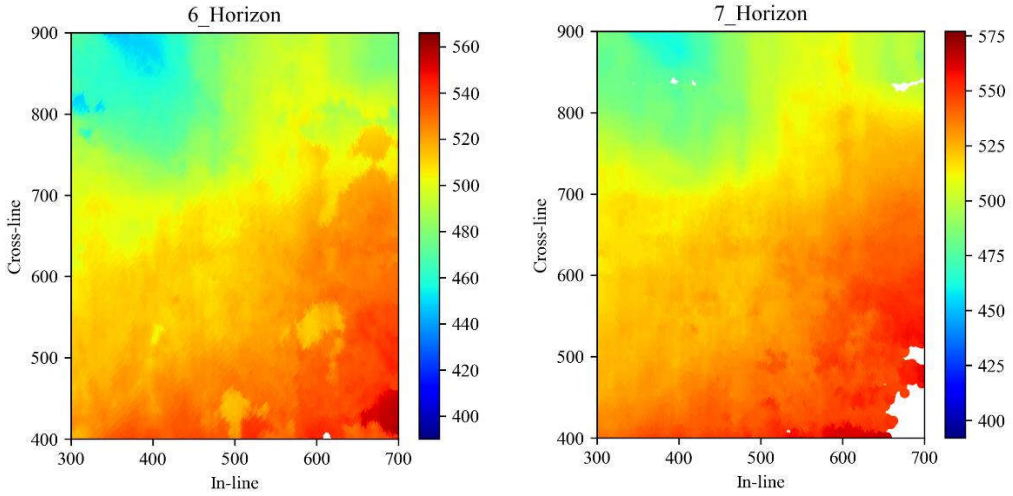


图 3-10 0_Horizon 至 7_Horizon 八个层位的沿层切片，色标为时间深度

将层位合并到一个数据体后分别沿着 inline = 500，crossline = 800 做纵剖面，如图 11、图 12 所示：

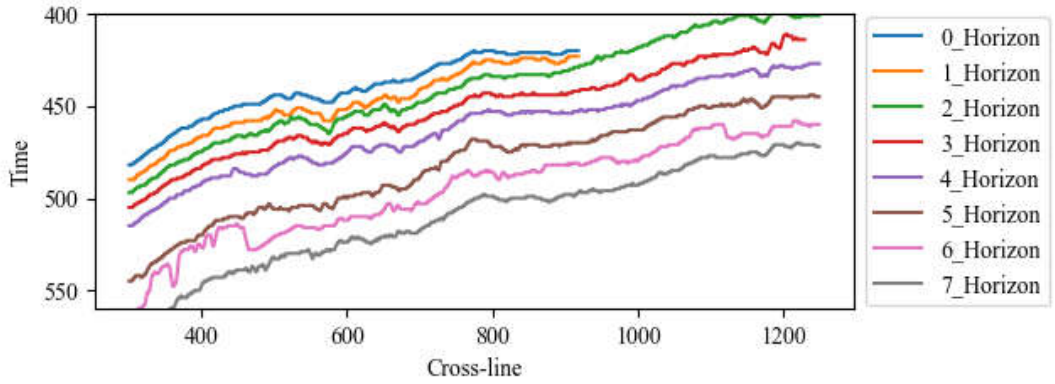


图 11 inline=500 层位剖面

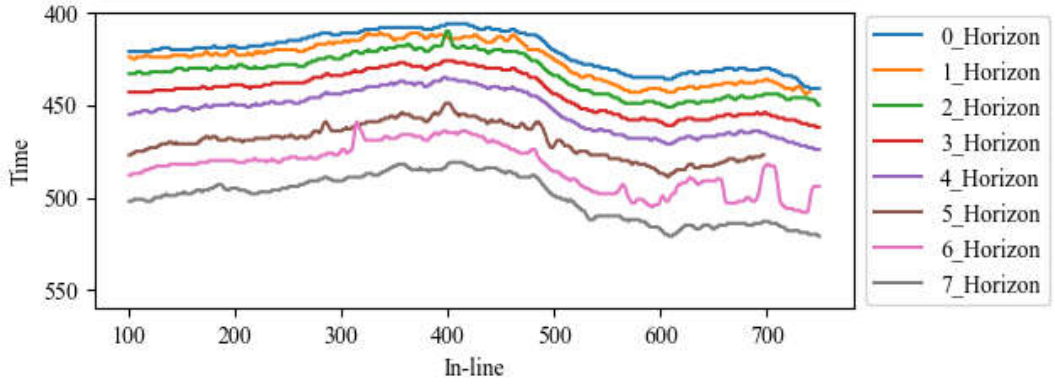


图 12 crossline=800 层位剖面

通过沿层切片以及地层剖面可发现，该工区地层有一定倾向，且地层接触方式复杂，整合接触、角度不整合、地层不整合均有发育，是一个做正演模拟的良好实验区。

3.2 正演范围选取

本次正演程序选用 inLine 范围是 300 到 700，crossLine 范围是 400 到 900，显示二维剖面分别为 inline = 500，crossline = 800，在二维剖面上选取的范围如图 13、图 14 所示：

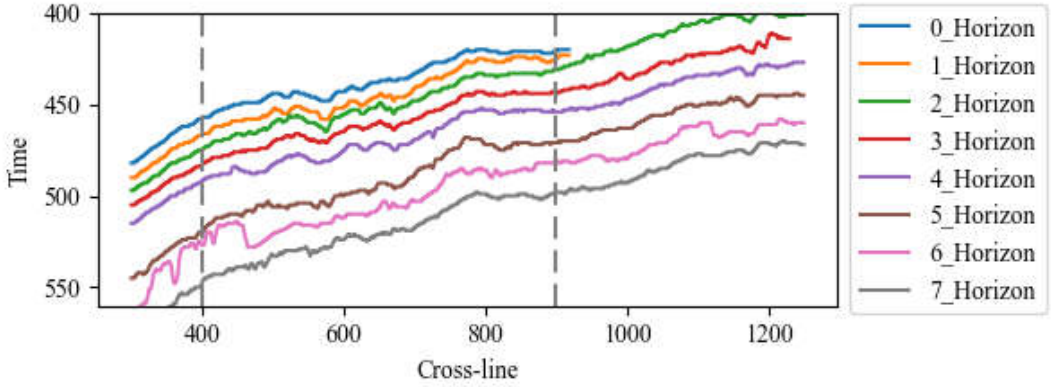


图 13 inline=500 层位选取剖面

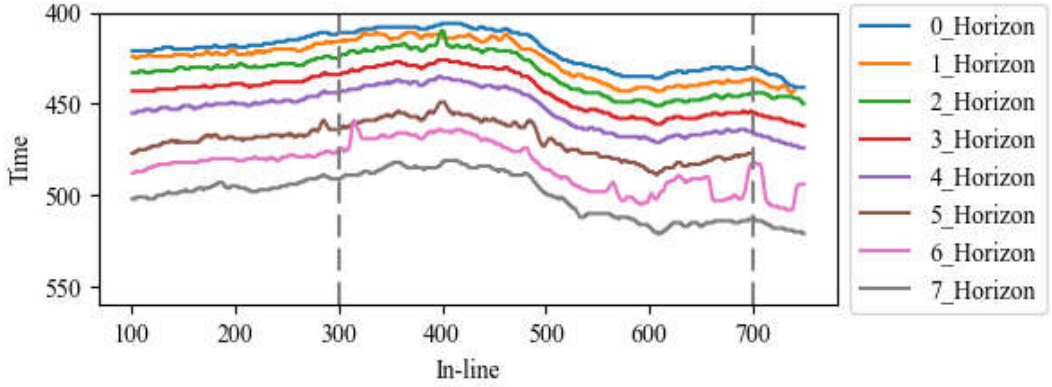


图 14 crossline=800 层位选取剖面

3.3 正演模拟

3.3.1 创建骨架

对选取剖面内的层位之间进行插值，创建地震骨架，如图 15 所示。

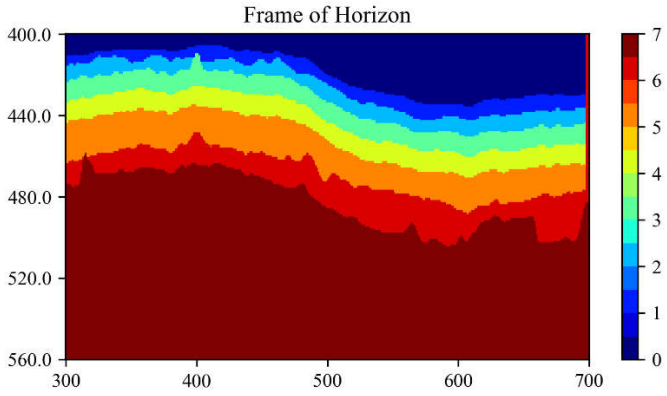


图 15 crossline=800 地震骨架，色标为骨架顺序

3.3.2 创建波阻抗模型

该工区岩性参数如下：

0 层是干净的砂岩；

1 地层由钙质粘土岩、粉砂岩和泥灰岩组成；

2 层由砂岩组成；

3 层是页岩；

4 层是干净的页岩；

5 层由钙质粘土岩、粉砂岩和泥灰岩组成；

6 层由钙质粘土岩、粉砂岩和泥灰岩组成；

7 层是页岩；

因此我们假设 0 层为页岩，其属性为 $\rho = 2650\ \mathrm{kg/m3}, V_p = 2300\ \mathrm{m/s}$

1 层为 $\rho = 2750\ \mathrm{kg/m3}, V_p = 2530\ \mathrm{m/s}$

2 层为 $\rho = 2675\ \mathrm{kg/m3}, V_p = 2500\ \mathrm{m/s}$

3 层为 $\rho = 2680\ \mathrm{kg/m3}, V_p = 2590\ \mathrm{m/s}$

4 层为 $\rho = 2600\ \mathrm{kg/m3}, V_p = 2340\ \mathrm{m/s}$

5 层为 $\rho = 2450\ \mathrm{kg/m3}, V_p = 2530\ \mathrm{m/s}$

6 层为 $\rho = 2420\ \mathrm{kg/m3}, V_p = 2680\ \mathrm{m/s}$

7 层为 $\rho = 2375\ \mathrm{kg/m3}, V_p = 2500\ \mathrm{m/s}$

将以上密度与纵波速度的乘积到入骨架，得到纵波阻抗模型，如图 16 所示

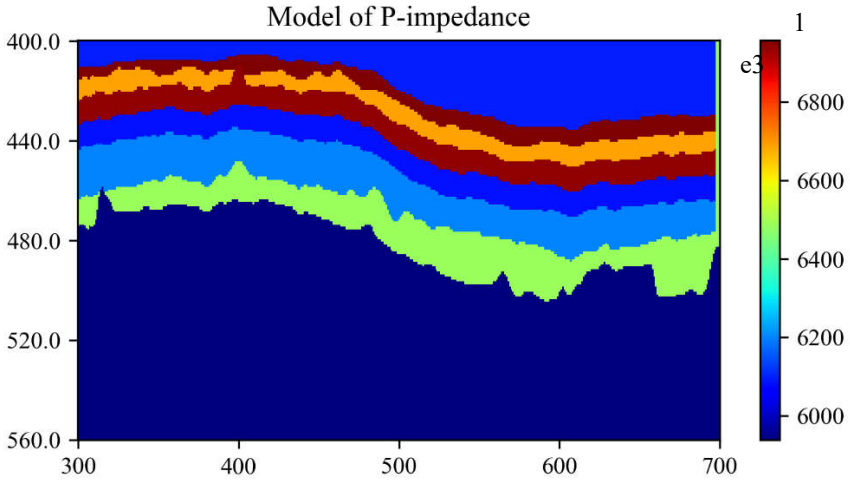


图 16 crossline=800 地震波阻抗模型，色标为波阻抗值

3.3.3 创建波阻抗模型

由阻抗模型我们已知地层分界面上下的波阻抗信息，利用反射系数计算公式，即可得到反射系数序列，反射系数序列如图 17 所示：

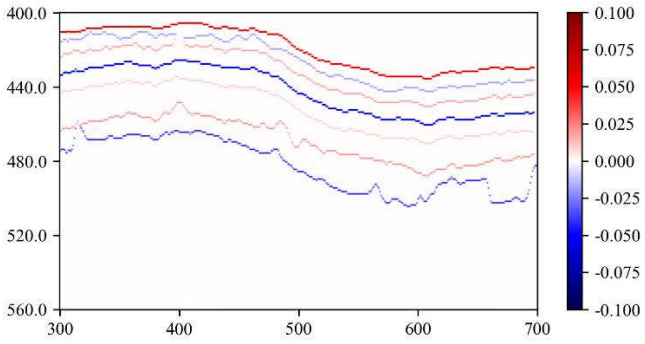


图 17 crossline=800 反射系数序列

3.3.4 合成地震记录

创建主频为 25Hz 的雷克子波，如图 18 所示，并将该子波与反射系数序列褶积，褶积后得到合成地震记录，如图 19 所示

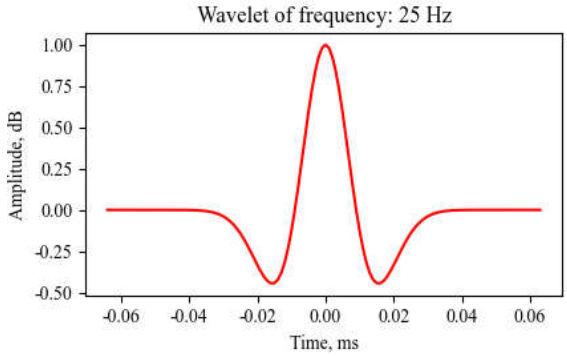


图 18 25Hz 主频的雷克子波

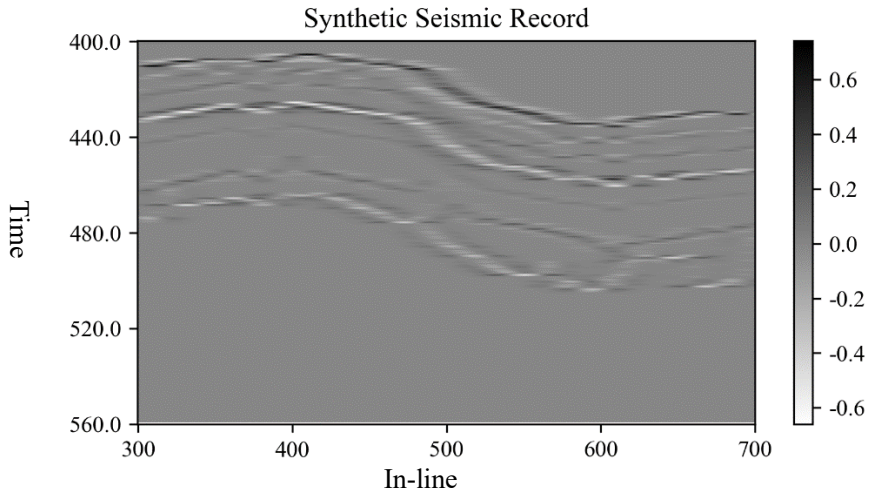


图 19 crossline=800 合成地震记录剖面

3.3.5 加入噪声、衰减

加入随机噪声、衰减得到最终正演剖面，如图 20、21 所示。随机噪声由 python 的科学计算库 Numpy 库所带 random 函数提供，衰减由简单的指数衰减函数提供。

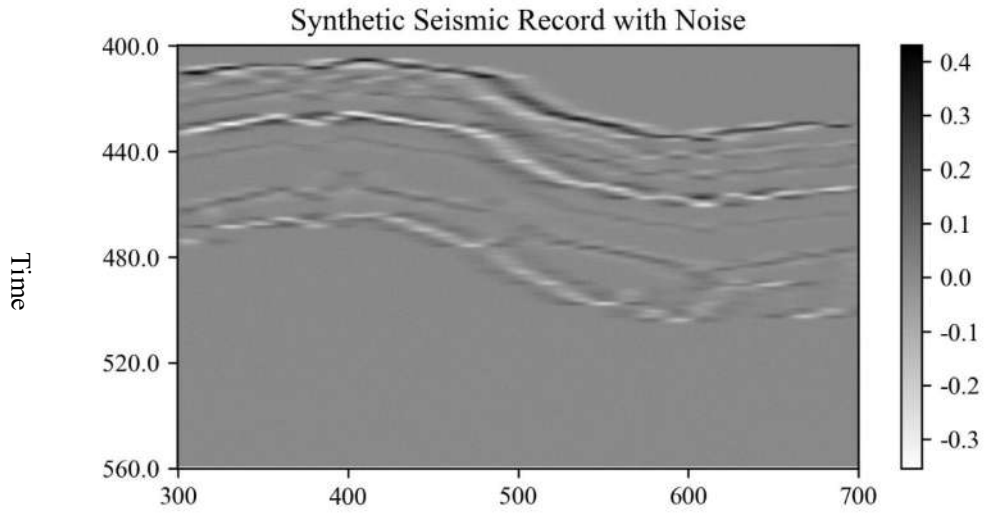


图 20 crossline=800 地震正演模拟剖面，灰度显示

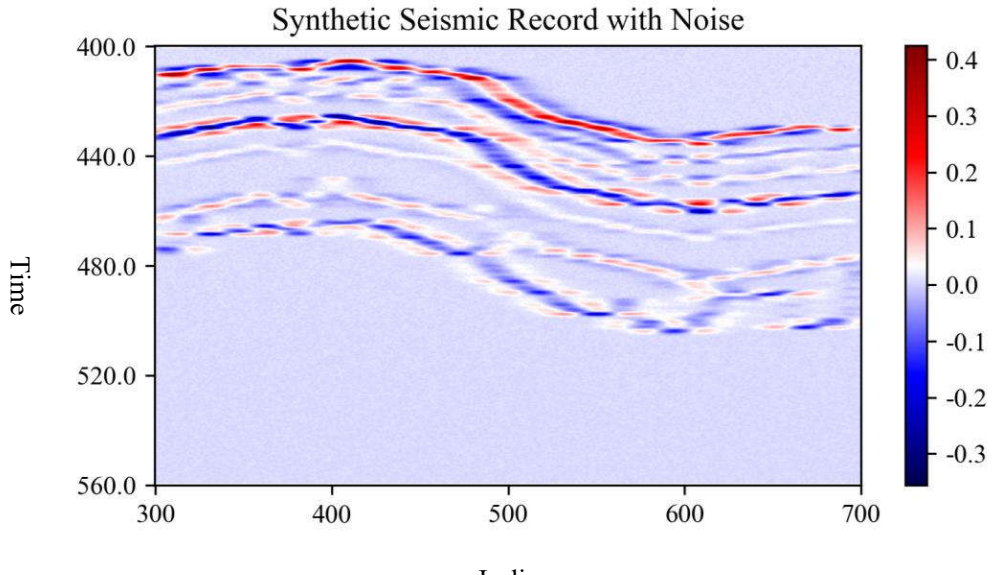


图 21 crossline=800 地震正演模拟剖面，变密度显示

3.3.6 inline=500 正演模拟

重复上述流程，对该工区所选取的数据范围，选取 inline=500 的测线进行正演模拟,结果如图 23 所示。

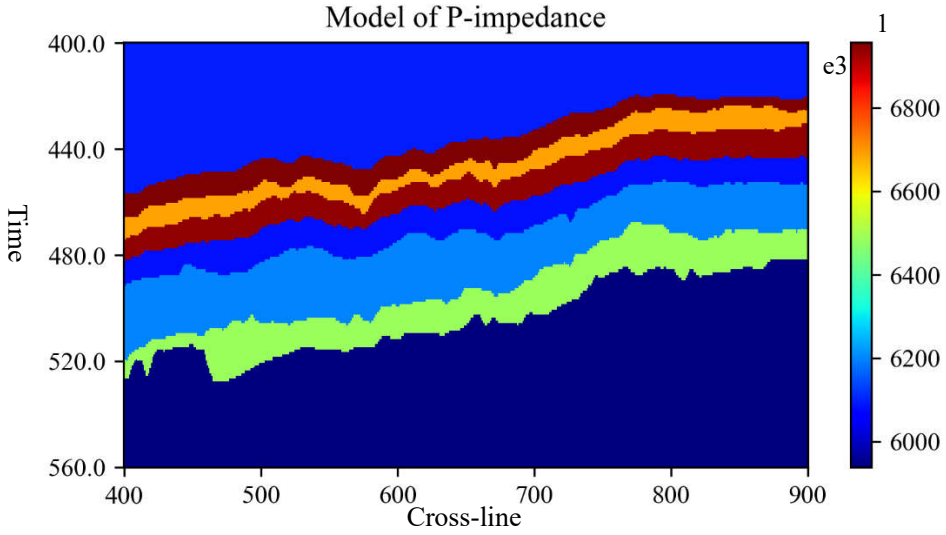


图 22 inline=500 波阻抗剖面，色标为波阻抗值

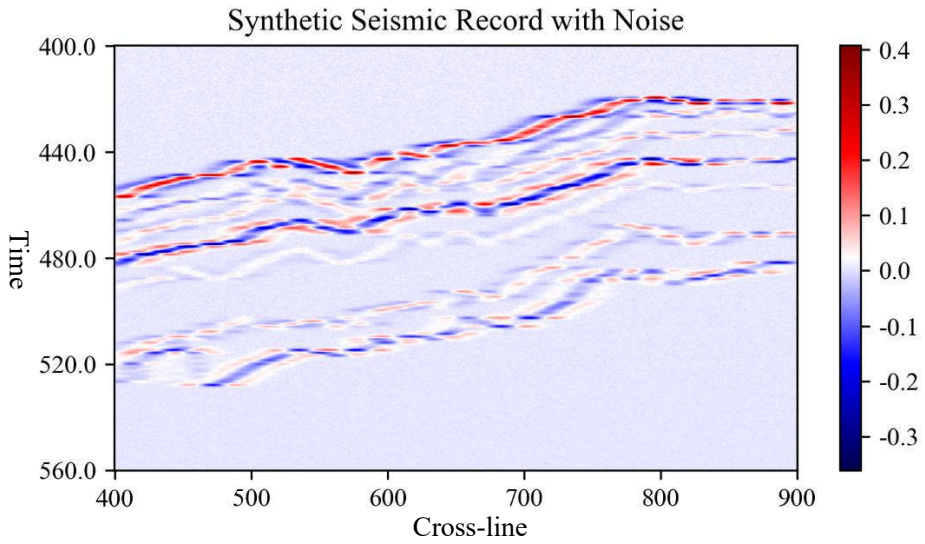


图 23 inline=500 地震正演模拟剖面，变密度显示

四、程序来源及改进

4.1 来源

本程序所含数据及部分代码来源于 GitHub 上的项目 Seismic_Forward_Modeling（https://github.com/crimeacs/Seismic_Forward_Modeling），将原有的 inpy 格式文件重写为三个文件，分别为 InputHorizon.py、MakeModel.py、MakeSeismic.py，重新选取正演模拟范围。

4.2 改进

对程序中部分进行改进，如：

1. 正确刻画纵、横坐标显示范围，新增代码：

```
    plt.xticks(np.arange(0, n_cell1+1, 100), np.arange(range_lift,range_right+1, 100))
    plt.yticks(np.linspace(0, n_cell1, 5), np.linspace(400, 560,5))
```

2. 增加合适图件标题及文字显示，新增代码:

```
    # 将 matplotlib 字体设置为 Times New Roman
    plt.rcParams['font.sans-serif'] = ['Times New Roman']
    plt.rcParams['axes.unicode_minus'] = False
```

3. 新增图片保存选项，

```
plt.savefig('Picture/Model.png', dpi=300)
```

4. 正演过程增加地层衰减；

```
    noisy = noisy * np.exp(-0.001 * np.arange(noisy.shape[0])[:, None])
```

4.3 原有 Bug 修改

1.原有代码选择其他范围或者 Inline 测线剖面后无法正常进行插值，出现报错：

```
    A value (401.0) in x_new is above the interpolation range's maximum value (400.0).
```

发现是由于原始程序所选取道的范围（500-700），该范围内所有地层均发育，可以正常插值。

但一旦选取道号范围改变，选取范围内存在地层消失时，不存在的地层位置数据变为 nan，使用 interp1d 进行插值时，自变量因变量范围不一致，默认（bounds_error=True）插值范围须在插值对象的范围内，当尝试插入范围不在插值对象范围内时，便会引发 ValueError 报错(需要外插)。针对此 bug 做出如下修改：

原始：

```
    f = interp1d(x, layer, kind='cubic')
```

改为：

```
    f = interp1d(x, layer, kind='cubic', bounds_error=False)
```

聚类算法

一、聚类算法介绍

对于"监督学习"(supervised learning)，其训练样本是带有标记信息的，并且监督学习的目的是：对带有标记的数据集进行模型学习，从而便于对新的样本进行分类。

而在“无监督学习”(unsupervised learning)中，训练样本的标记信息是未知的，目标是通过对无标记训练样本的学习来揭示数据的内在性质及规律，为进一步的数据分析提供基础。对于无监督学习，应用最广的便是"聚类"(clustering)。

聚类算法试图将数据集中的样本划分为若干个通常是不相交的子集，每个子集称为一个“簇”(cluster)，通过这样的划分，每个簇可能对应于一些潜在的概念或类别。

简单来说：聚类算法的样本集都是没有标签的，那我们就需要根据样本的特征，给样本数据集进行聚类。并且无标签的算法也叫做无监督学习。

二、K-means 算法

kmeans 算法，又称为 k 均值算法。K-means 算法中的 k 表示的是聚类为 k 个簇，means 代表取每一个聚类中数据值的均值作为该簇的中心，或者称为质心，即用每一个的类的质心对该簇进行描述。

2.1 算法思想和流程介绍

2.1.1 算法思想

以空间中 K 个点为中心进行聚类（即先从样本集中随机选取 k 个样本作为簇中心），对最靠近他们的对象归类（所有样本与这 k 个“簇中心”的距离，对于每一个样本，将其划分到与其距离最近的“簇中心”所在的簇中）。通过迭代的方法，逐次更新各聚类中心的值，直至得到最好的聚类结果。

2.1.2 算法流程：

先从没有标签的元素集合 A 中随机取 K 个元素，作为 K 个子集各自的质心；

分别计算剩下的元素到 K 个子集质心的距离，根据距离将元素分别划分到

最近的子集；

根据聚类结果，重新计算质心(计算方法为子集中所有元素各个维度的算术平均数)

将集合 A 中全部元素按照新的质心然后再重新聚类；

重复第 4 步，直到聚类结果不再发生变化。

2.2 K-means 算法的代码实现

使用随机数，并对随机数展开聚类

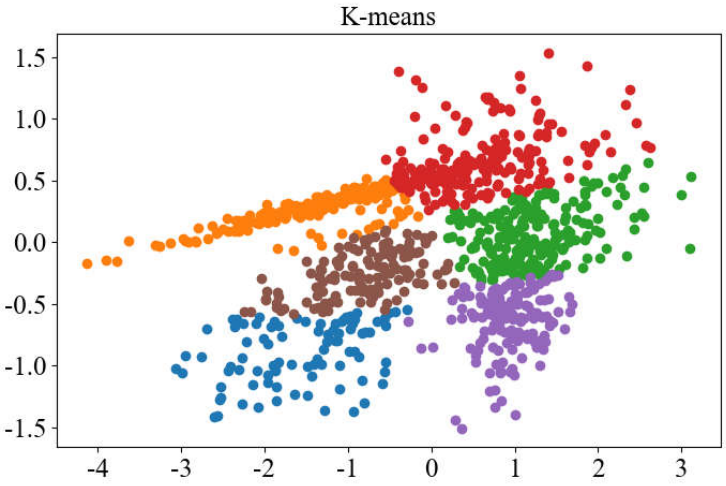


图 24 随机数据做 k 均值聚类

基于前文正演所得到剖面数据，将不同 CDP 道上的数据点与其所在时间深度重新构造成二维数据，根据时间点的中心进行聚类分析

注：本节所用数据仅对聚类算法进行探究分析，并无具体的地球物理含义；

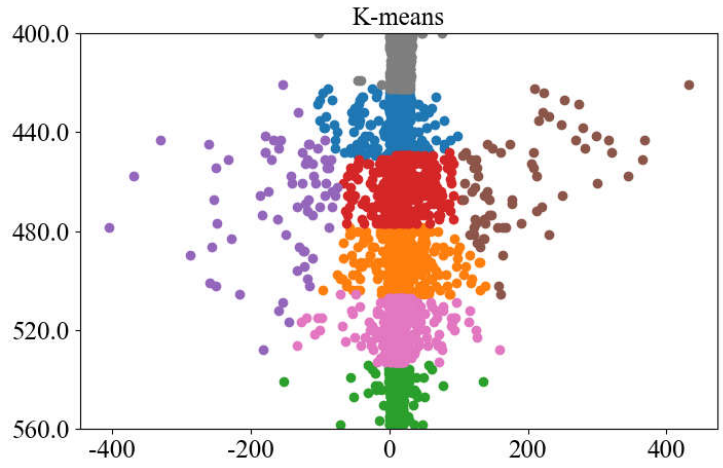


图 25 正演数据做 k 均值聚类

三、层次聚类

层次的聚类方法（Hierarchical Clustering），从字面上理解，其是层次化的聚类，最终得出来的是树形结构。专业一点来说，层次聚类通过 计算不同类别数据点间的相似度 来创建一棵有层次的嵌套聚类树。

层次聚类的好处是不需要指定具体类别数目的，其得到的是一颗树，聚类完成之后，可在任意层次横切一刀，得到指定数目的簇。

3.1 算法思想和流程介绍

3.1.1 算法思想

自下而上的 凝聚方法（agglomerative：先将所有样本的每个点都看成一个簇，然后找出距离最小的两个簇进行合并，不断重复到预期簇或者其他终止条件），

凝聚方法的代表算法：AGNES，Agglomerative Nesting；

自顶向下的 分裂方法（divisive：先将所有样本当作一整个簇，然后找出簇中距离最远的两个簇进行分裂，不断重复到预期簇或者其他终止条件）。

分裂方法的代表算法：DIANA，Divisive Analysis。

3.1.2 基本流程

AGNES 算法步骤：

（1） 初始化，每个样本当做一个簇

（2） 计算任意两簇距离，找出 距离最近的两个簇，合并这两簇

（3） 重复步骤 2……

直到，最远两簇距离超过阈值，或者簇的个数达到指定值，终止算法

DIANA 算法步骤：

（1） 初始化，所有样本集中归为一个簇

（2） 在同一个簇中，计算任意两个样本之间的距离，找到 距离最远 的两个样本点 a,b，将 a,b 作为两个簇的中心;

（3） 计算原来簇中剩余样本点距离 a，b 的距离，距离哪个中心近，分配到哪个簇中

（4） 重复步骤 2、3 ……

直到，最远两簇距离不足阈值，或者簇的个数达到指定值，终止算法。

3.2 AgglomerativeClustering 算法的代码实现

使用随机数，并对随机数展开层次聚类

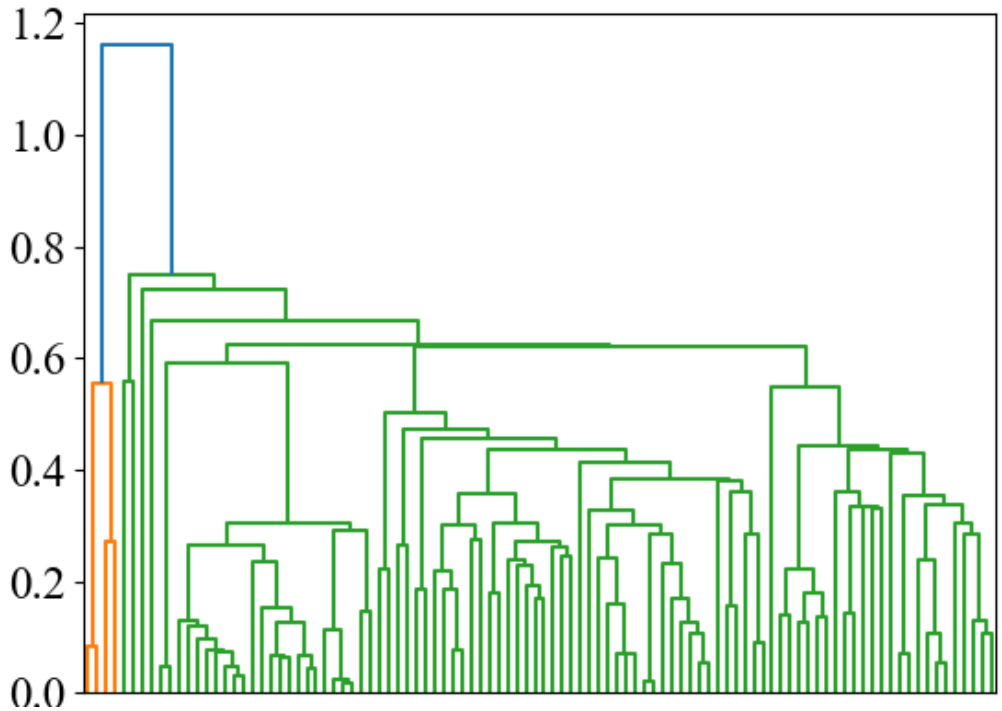


图 26 随机数据层次聚类数示例

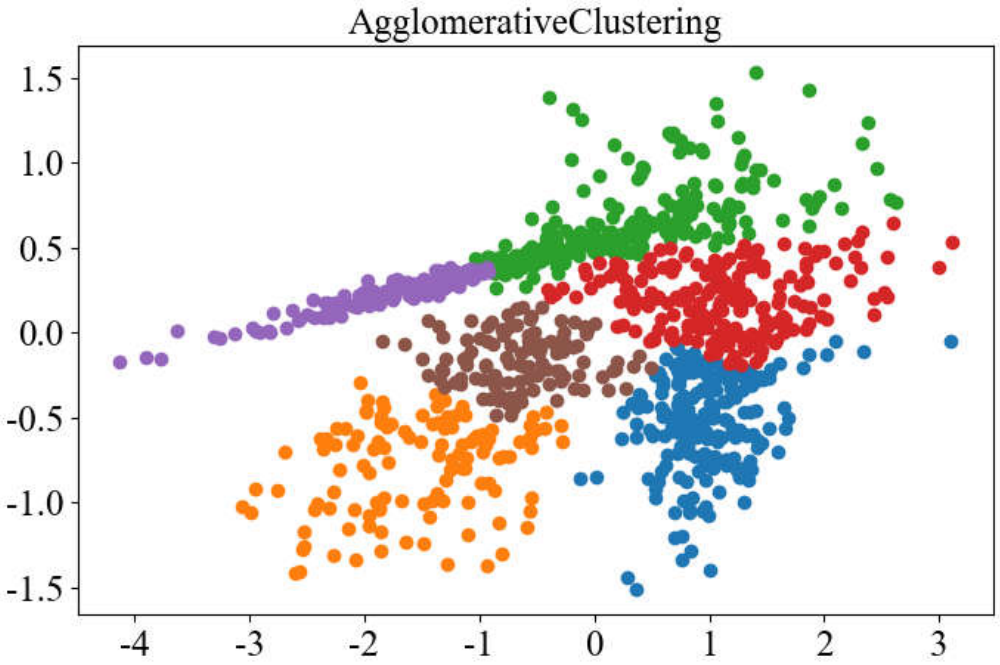


图 27 随机数据层次聚类结果

基于前文正演所得到剖面数据，将不同 CDP 道上的数据点与其所在时间深度重新构造成二维数据，根据时间点的中心进行层次聚类分析。

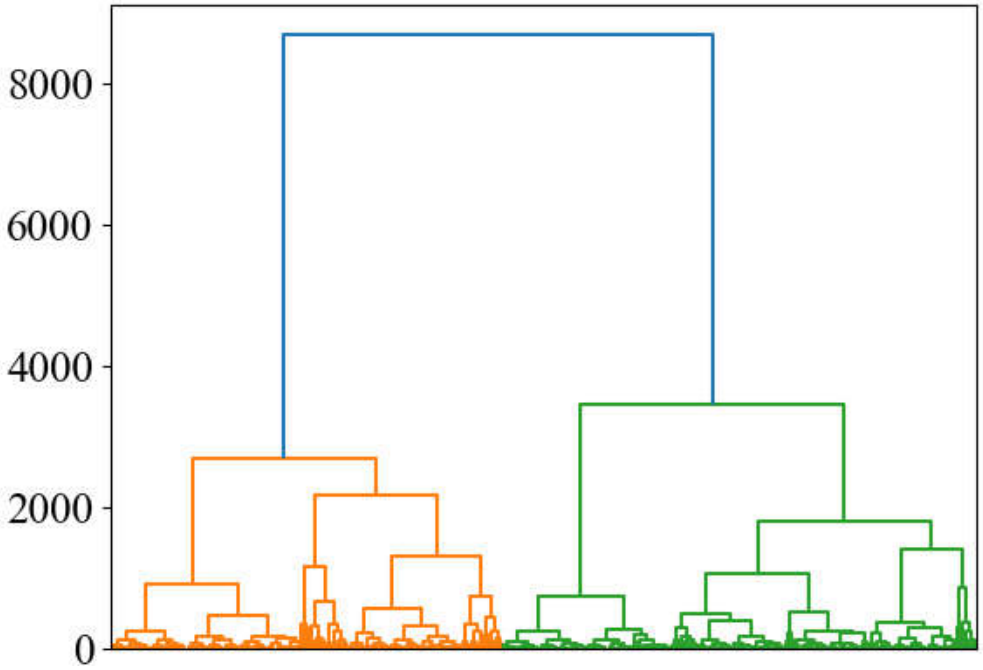


图 28 地震数据聚类数示例

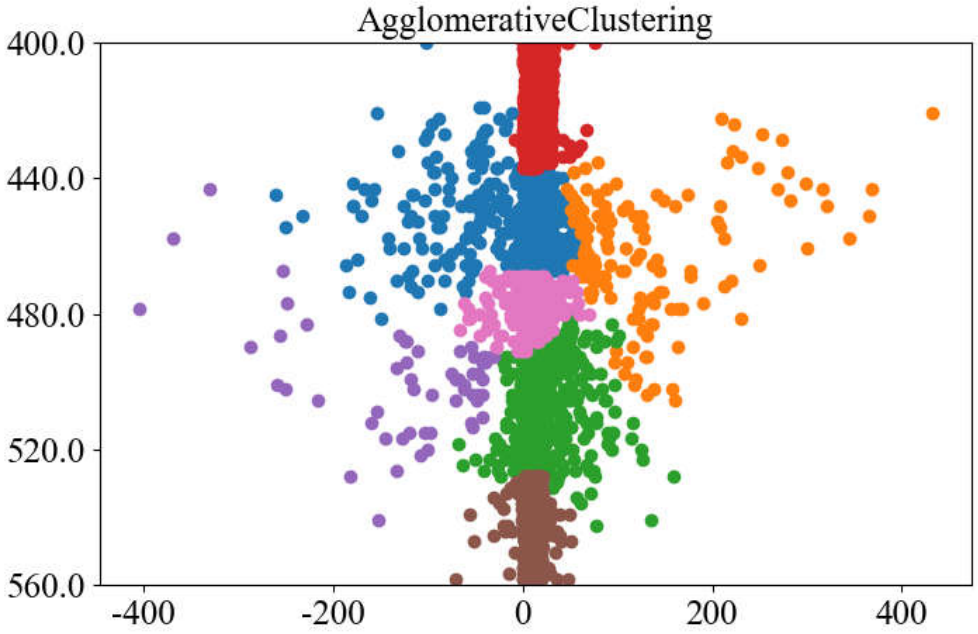


图 29 地震数据聚类数示例

四、密度聚类

密度聚类的代表性算法常见 DBSCAN（Density-Based Spatial Clustering of Applications with Noise，具有噪声的基于密度的聚类方法）为例。全称中透

露出两个重要信息，即 DBSCAN 聚类算法是抗噪的，稍后我们会介绍详细的机制。

密度聚类的思想，其实和人类的思维十分相似，通过判断“是否紧密相连“来确定样本点是否属于一个簇。

4.1 算法思想和流程介绍

4.1.1 算法思想

密度聚类的思想，其实和人类的思维十分相似，通过判断“是否紧密相连“来确定样本点是否属于一个簇。

如果一个点周围环绕很多其他点，并且比他密度大的点离他距离还很远，则认为这个点是一个聚类中心。在 DBSCAN 中，我们将其称之为“核心对象”

4.1.2 基本流程

（1）先找到一个核心对象，从它出发，确定若干个直接密度可达的对象

（2）再从这若干个对象出发，寻找它们直接密度可达的点，

（3）直至最后没有可添加的对象了，那么一个簇的更新就完成了。

4.2 DBSCAN 算法的代码实现

使用随机数，并对随机数展开密度聚类

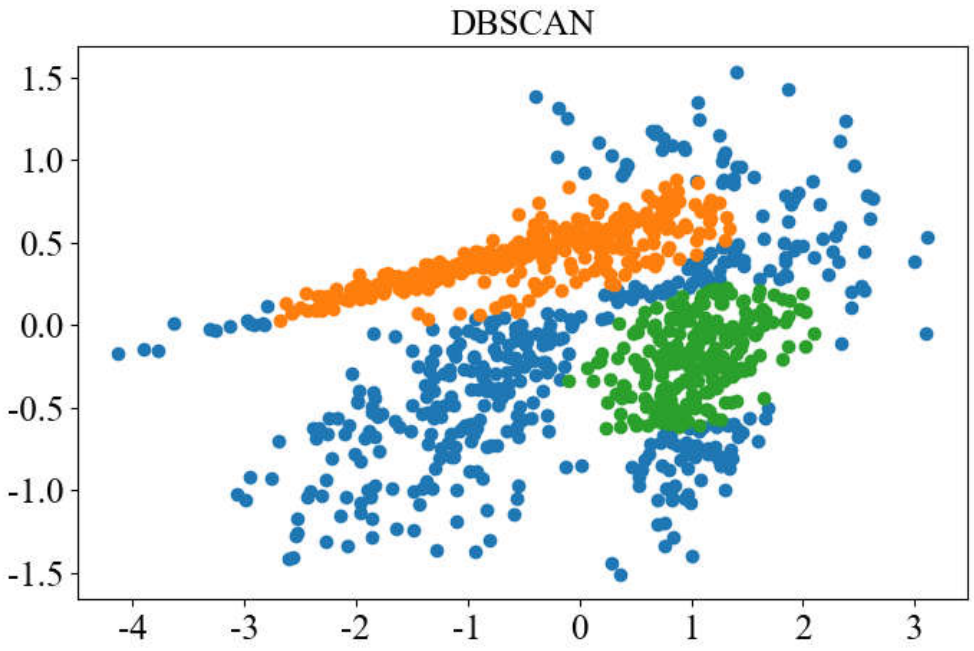


图 30 随机数据密度聚类结果

基于前文正演所得到剖面数据，将不同 CDP 道上的数据点与其所在时间深

度重新构造成二维数据，根据时间点的中心进行密度聚类分析。

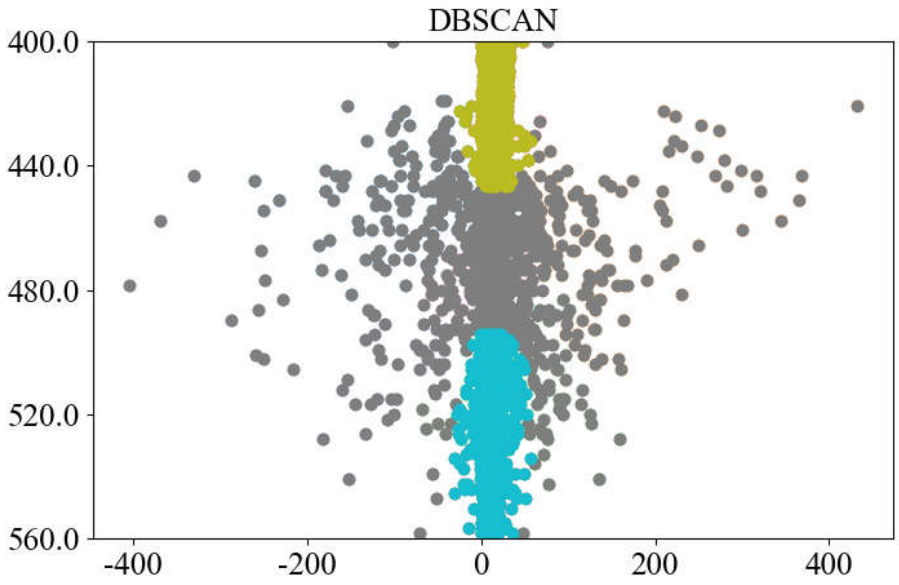


图 31 地震密度聚类结果

五、谱聚类

谱聚类(Spectral Clustering, SC)是一种基于图论的聚类方法——将带权无向图划分为两个或两个以上的最优子图(sub-Graph)，使子图内部尽量相似，而子图间距离尽量距离较远，以达到常见的聚类的目的。

5.1 算法思想和流程介绍

5.1.1 算法思想

谱聚类是将原问题转化为“图分割“的问题。涉及两个关键的步骤：将原数据，转化为“无向权重图”、对图进行分割。

5.1.2 基本流程：

1 根据输入的相似矩阵的生成方式构建样本的相似矩阵 S ；

2 根据相似矩阵 S 构建邻接矩阵 W，构建度矩阵 D ；

3 计算出拉普拉斯矩阵 L；

4 构建标准化后的拉普拉斯矩阵 $D^{-1/2}LD^{-1/2}$；

5 计算 $D^{-1/2}LD^{-1/2}$ 最小的 k_1 个特征值所各自对应的特征向量 f ；

6 将各自对应的特征向量 f 组成的矩阵按行标准化，最终组成 $n*k_1$ 维的特征矩阵 F ；

7　对 F 中的每一行作为一个 k_1 维的样本，共 n 个样本，用输入的聚类方法进行聚类，聚类维数为 k_2 ；

8　得到簇划分 $C(c_1, c_2, \ldots c_{k2})$ ；

5.2 SpectralClustering 算法及代码实现

使用随机数，并对随机数展开谱聚类

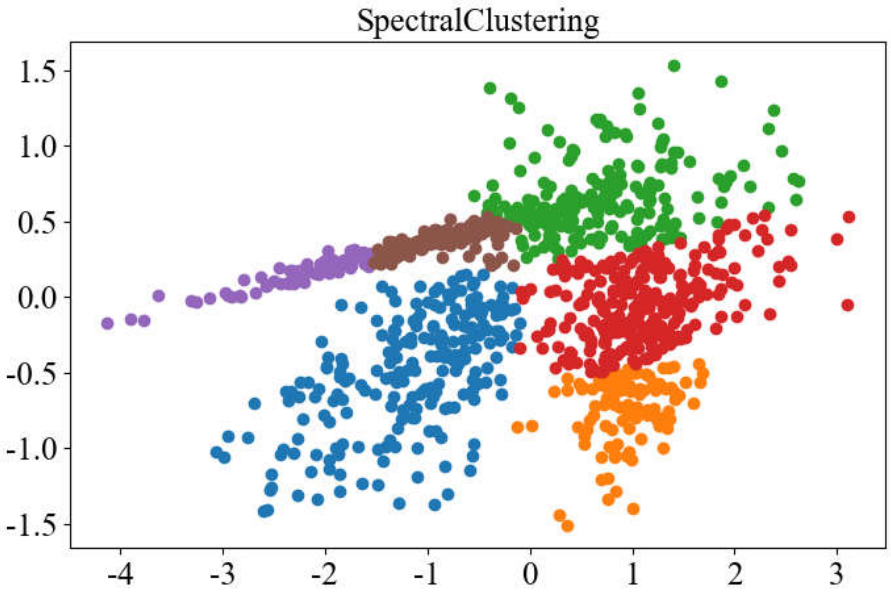


图 32　随机数据谱聚类结果

基于前文正演所得到剖面数据，将不同 CDP 道上的数据点与其所在时间深度重新构造成二维数据，根据时间点的中心进行谱聚类分析。

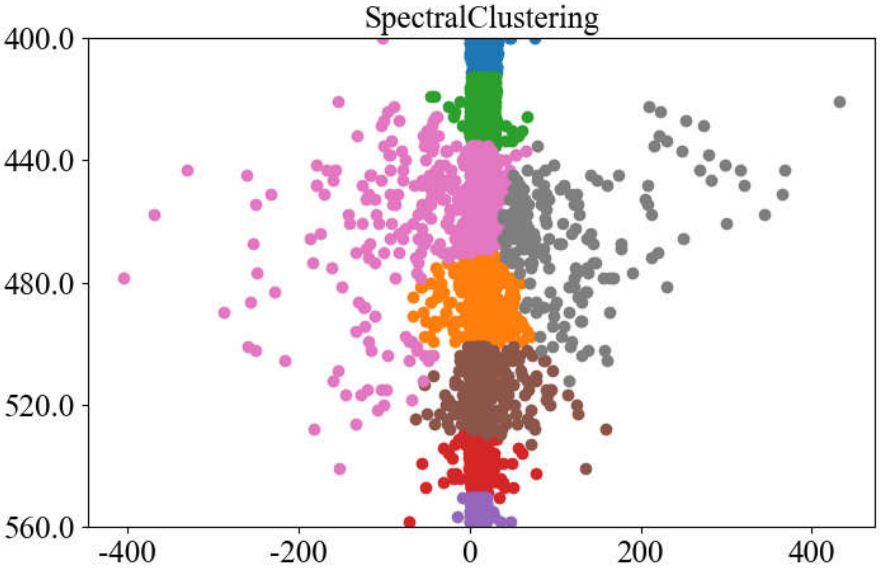


图 33　地震谱聚类结果

附源代码

正演模拟

InputHorizon.py

```
"""
============================
# -*- coding: utf-8 -*-
# Time       : 2023/3/25 13:24
# Author     : Qisx
# FileName: InputHorizon.py
# Software: PyCharm
============================
"""
import os.path
import zipfile
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

# 将 matplotlib 字体设置为 Times New Roman
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.rcParams['axes.unicode_minus'] = False

###############
#
读取地层文件
#
###############
with zipfile.ZipFile('Export.zip', 'r') as zip_ref:
zip_ref.extractall('target_path')
filenames = [f for f in os.listdir('target_path')]
data = pd.DataFrame(columns=['In-line', 'Cross-line', 'z', 'Filename'])
n = 0
for filename in filenames:
df = pd.read_table('target_path/' + filename, header=None, sep=' ', encoding='utf-8',
names=['In-line', 'Cross-line', 'z', 'Filename'])
# header=None：第一列不作为表头
# sep=' '：空格分割
# encoding='utf-8'：utf-8 格式打开
# names=['In-line','Cross-line','t']：定义列名
df['Filename'] = filename
df['In-line'] = pd.to_numeric(df['In-line']).astype(int)
df['Cross-line'] = pd.to_numeric(df['Cross-line']).astype(int)
df['z'] = pd.to_numeric(df['z']).astype(int)

data = pd.concat([data, df], ignore_index=True)
n = n + 1

# 地层文件存入 csv
data.to_csv('Horizon.csv', index=False)
print('**********saved**********')
```

```
        ###############
        #
绘制地层切片
#
        ###############
        xline = 800
        inline = 500
        draw_horizon = 0
        for draw_horizon in range(0, 8):
        group_show_horizon =
data[data['Filename'] == str(draw_horizon) +
'_Horizon']

        plt.figure(figsize=(4, 4))
        plt.scatter(group_show_horizon['In-
line'][::10], group_show_horizon['Cross-
line'][::10],
        c=group_show_horizon['z'][::10], s=20,
cmap='jet')
        plt.ylim(400, 900)
        plt.xlim(300, 700)
        plt.title(str(draw_horizon) + '_Horizon')
        plt.xlabel('In-line')
        plt.ylabel('Cross-line')
        plt.colorbar()
        plt.plot(group_show_horizon['In-
line'][::10],
        np.zeros(shape=(group_show_horizon['C
ross-line'][::10].shape)) + xline, color='black')
        plt.plot(np.zeros(shape=(group_show_h
orizon['Cross-line'][::10].shape)) + inline,
        group_show_horizon['Cross-line'][::10],
color='black')
        plt.show()
        plt.savefig('Picture/line_' +
str(draw_horizon) + '_Horizon.png', dpi=300)

        ###############
        #
绘制地层切片
#
        ###############
        cut_x_line_left = 400
        cut_x_line_right = 900
        cut_in_line_left = 300
        cut_in_line_right = 700

        inline_slice = data.where(data['In-line']
== inline)
        group_inline =
inline_slice.groupby('Filename', sort=True)
        plt.figure(figsize=(10, 8))
        plt.subplot(211)
        plt.ylim(560, 400)
        n = 0
        for name, group in group_inline:
        plt.plot(group['Cross-line'], group['z'],
label=name)
        plt.xlabel('Cross-line')
        plt.ylabel('Time')
        n = n + 1
        plt.legend(loc='center left',
bbox_to_anchor=(1, 0.5))
        # plt.axvline(x=cut_x_line_left, c='grey',
dashes=[6, 3])
        # plt.axvline(x=cut_x_line_right, c='grey',
dashes=[6, 3])

        crossline_slice = data.where(data['Cross-
line'] == xline)
```

```
    group_crossline =
crossline_slice.groupby('Filename', sort=True)
    plt.subplot(212)
    plt.ylim(560, 400)
    n = 0
    for name, group in group_crossline:
    plt.plot(group['In-line'], group['z'],
label=name)
    plt.xlabel('In-line')
    plt.ylabel('Time')
    cture/inline_crossline_nocut.png', dpi=300)
    n = n + 1
    plt.legend(loc='center left',
bbox_to_anchor=(1, 0.5))
    # plt.axvline(x=cut_in_line_left, c='grey',
dashes=[6, 3])
    # plt.axvline(x=cut_in_line_right,
c='grey', dashes=[6, 3])
    plt.show()
    plt.savefig('Pi
```

MakeModel.py

```
"""
============================
# -*- coding: utf-8 -*-
# Time        : 2023/3/25 16:34
# Author    : Qisx
# FileName: MakeModel.py
# Software: PyCharm
============================
"""

import sys
import numpy as np
import pandas as pd
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
import scipy as sp
import scipy.ndimage as snd
import copy

#  将 matplotlib 字体设置为 Times New
Roman
plt.rcParams['font.sans-serif'] = ['Times New
Roman']
plt.rcParams['axes.unicode_minus'] = False


def resizing_array(x, out_max):
    if int(out_max / len(x)) > 1:
        x = x[::int(out_max / len(x))]
    else:
        x = x[::1]
    while len(x) > out_max:
        x = np.delete(x, -1)
    return x


def map_and_resize(x, x_min, x_max,
out_min, out_max):
    mapped = (((x - x_min) * (out_max -
out_min) / (x_max - x_min)) +
out_min).astype(int)
    z = mapped
    return z


if __name__ == '__main__':
    inline = 500
    crossline = 800
    cut_x_line_left = 300
    cut_x_line_right = 700
    cut_in_line_left = 400
    cut_in_line_right = 900

    data = pd.read_csv('Horizon.csv', sep=',',
encoding='utf-8', )
    inline_slice = data.where(data['In-line']
```

```
== inline)
        crossline_slice = data.where(data['Cross-line'] == crossline)

        print('range of inline of inline_slice:', inline_slice['Cross-line'].min(), inline_slice['Cross-line'].max())
        print('range of crossline of crossline_slice:', crossline_slice['In-line'].min(), crossline_slice['In-line'].max())
        ###############
        #
制作地层骨架插值
#
        ###############
        in_slice_crop = inline_slice.where(inline_slice['In-line'] <= cut_x_line_right)
        in_slice_crop = in_slice_crop.where(in_slice_crop['In-line'] >= cut_x_line_left)
        cross_slice_crop = crossline_slice.where(crossline_slice['Cross-line'] <= cut_in_line_right)
        cross_slice_crop = cross_slice_crop.where(cross_slice_crop['Cross-line'] >= cut_in_line_left)

        grouped_in_crop = in_slice_crop.groupby('Filename', sort=True)
        grouped_x_crop = cross_slice_crop.groupby('Filename', sort=True)

        # Choose view
        view = 'Cross-line'

        if view == 'Cross-line':
            n_cell1 = cut_in_line_right - cut_in_line_left
            grouped_crop = grouped_in_crop

range_lift,range_right=cut_in_line_left,cut_in_line_right

        elif view == 'In-line':
            n_cell1 = cut_x_line_right - cut_x_line_left
            grouped_crop = grouped_x_crop

range_lift,range_right=cut_x_line_left,cut_x_line_right

        # Frame of Horizon
        model = np.zeros(shape=(n_cell1, n_cell1), dtype=np.int)
        n = 0
        for name, group in grouped_crop:
            # scale model
            layer = map_and_resize(group['z'].values, 400, 560, 0, n_cell1)
            # Interpolate
            x = np.arange(0, len(layer), 1)
            # print(x, layer)
            print(len(x), len(layer))
            f = interp1d(x, layer, kind='cubic',bounds_error=False,)
            x_new = np.arange(0, n_cell1, 1).astype('int')
            layer_new = f(x_new).astype('int')
            # For the first layer
            if n == 0:
                layer_n = np.zeros_like(x_new)
            # For the last layer
            if n ==    7:
                layer_new = np.zeros_like(x_new) + n_cell1
            # Fill trace by trace
            for i in range(len(x_new)):
                model[layer_n[i]:layer_new[i], x_new[i]] = n
            layer_n = layer_new
            n += 1

        # Convert colormap and show
        plt.figure(figsize=(6,3))
        cmap = plt.get_cmap('jet', 14)
        cmap.set_under('gray')
```

```
    cax = plt.imshow(model,
cmap=cmap,interpolation='nearest',aspect='a
uto')
    plt.title('Frame of Horizon')
    # plt.xticks(np.arange(0, n_cell1+1,
100),np.arange(range_lift,range_right+1,
100))
    plt.yticks(np.linspace(0, n_cell1, 5),
np.linspace(400, 560,5))
    plt.grid(False)
    cbar = plt.colorbar(cax,
ticks=np.arange(0, 13, 1))
    plt.show()
    plt.savefig('Picture/Model.png', dpi=300)

    ###############
    #                    制作引入
密度和纵波速度给模型添加纵波阻抗信息
#
    ###############
    model_dens = np.copy(model)
    model_dens[model_dens == 0] = 2650
    model_dens[model_dens == 1] = 2750
    model_dens[model_dens == 2] = 2675
    model_dens[model_dens == 3] = 2680
    model_dens[model_dens == 4] = 2600
    model_dens[model_dens == 5] = 2450
    model_dens[model_dens == 6] = 2420
    model_dens[model_dens == 7] = 2375

    # Create model filled with Velocity
    model_Vp = np.copy(model)

    fault_position = 100
    with_fault = np.roll(faulted[:,
0:fault_position], fault_throw, axis=0)
    with_fault = np.hstack([with_fault,
faulted[:, fault_position:]])
    faulted_model = with_fault

    plt.figure(figsize=(6,3))
    plt.imshow(faulted_model,
cmap='jet',interpolation='nearest',aspect='aut
o')
    model_Vp[model_Vp == 0] = 2300
    model_Vp[model_Vp == 1] = 2530
    model_Vp[model_Vp == 2] = 2500
    model_Vp[model_Vp == 3] = 2590
    model_Vp[model_Vp == 4] = 2340
    model_Vp[model_Vp == 5] = 2530
    model_Vp[model_Vp == 6] = 2680
    model_Vp[model_Vp == 7] = 2500

    model_p_impedance = model_dens *
model_Vp / 1e3

    plt.figure(figsize=(6,3))
    plt.imshow(model_p_impedance,
cmap='jet',interpolation='nearest',aspect='aut
o')
    plt.title('Model of P-impedance')
    plt.xticks(np.arange(0, n_cell1+1, 100),
np.arange(range_lift,range_right+1, 100))
    plt.yticks(np.linspace(0, n_cell1, 5),
np.linspace(400, 560,5))
    plt.colorbar()
    plt.grid(False)
    plt.show()

plt.savefig('Picture/Model_P_impedance.png'
, dpi=300)
    # Make Faulted Model
    faulted =
copy.deepcopy(model_p_impedance)

    fault_throw = 00

    plt.title('Model of P-impedance with
fault')
    plt.xticks(np.arange(0, n_cell1+1, 100),
np.arange(range_lift,range_right+1, 100))
    plt.yticks(np.linspace(0, n_cell1, 5),
np.linspace(400, 560,5))
    plt.colorbar()
    plt.grid(False)
    plt.show()

plt.savefig('Picture/Model_P_impedance_faul
```

t.png', dpi=300)

#将 faulted_model 保存为 csv 文件

MakeSeismic.py

```
"""
============================
# -*- coding: utf-8 -*-
# Time        : 2023/3/26 20:09
# Author    : Qisx
# FileName: MakeSeismic.py
# Software: PyCharm
============================
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy as sp

# 将 matplotlib 字体设置为 Times New Roman
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.rcParams['axes.unicode_minus'] = False


def ricker(f, length=0.128, dt=0.001):
    t = np.arange(-length / 2, (length - dt) / 2, dt)
    y = (1.0 - 2.0 * (np.pi ** 2) * (f ** 2) * (t ** 2)) * np.exp(-(np.pi ** 2) * (f ** 2) * (t ** 2))
    return t, y


if __name__ == '__main__':

    inline = 500
    crossline = 800
    cut_x_line_left = 300
    cut_x_line_right = 700
    cut_in_line_left = 400
    np.savetxt('faulted_model.csv', faulted_model, delimiter = ',')

    cut_in_line_right = 900
    view = 'Cross-line'
    if view == 'Cross-line':
        n_cell1 = cut_x_line_right - cut_x_line_left

range_lift,range_right=cut_in_line_left,cut_in_line_right
    elif view == 'In-line':
        n_cell1 = cut_in_line_right - cut_in_line_left

range_lift,range_right=cut_x_line_left,cut_x_line_right

    faulted_model = pd.read_csv('faulted_model.csv', sep=',', encoding='utf-8')
    faulted_model = faulted_model.values

    ###############
    #
制作反射系数序列
#
    ###############
    width = 2

    rc = (faulted_model - np.roll(faulted_model, width, axis=0)) / (
                faulted_model + np.roll(faulted_model, width, axis=0))
    plt.figure(figsize=(6,3))
    plt.imshow(rc, cmap='seismic', vmin=-0.1,
vmax=0.1,interpolation='gaussian',aspect='auto')
    plt.colorbar()
    plt.xticks(np.arange(0, n_cell1+1, 100), np.arange(range_lift,range_right+1, 100))
    plt.yticks(np.linspace(0, n_cell1, 5), np.linspace(400, 560,5))
```

```
        plt.grid(False)
        plt.show()
        plt.savefig('Picture/rc.png', dpi=300)

        ###############
        #
创建子波
#
        ###############
        plt.figure(figsize=(6, 3))
        f = 25
        t, w = ricker(f)
        plt.plot(t, w, 'r')
        plt.xlabel('Time, ms')
        plt.ylabel('Amplitude, dB')
        plt.title('Wavelet of frequency: %i Hz' %
f)
        plt.grid(False)
        plt.show()
        plt.savefig('Picture/Wavelet.png',
dpi=300)

        ###############
        #
制作合成地震记录
#
        ###############
        synth =
np.array([np.apply_along_axis(lambda t:
np.convolve(t, w, mode='same'), axis=0,

arr=r) for r in rc])

        plt.figure(figsize=(6,3))
        plt.imshow(synth,
cmap='gray_r',interpolation='gaussian',aspect
='auto' )
        plt.title('Synthetic Seismic Record')
        plt.colorbar()
        plt.xticks(np.arange(0, n_cell1+1, 100),
np.arange(cut_x_line_left, cut_x_line_right+1,
100))
        plt.yticks(np.linspace(0, n_cell1, 5),
np.linspace(400, 560,5))
        plt.grid(False)
        plt.show()
        plt.savefig('Picture/Synthetic_gery.png',
dpi=300)

        ###############
        #
加入噪声
#
        ###############

        blurred =
sp.ndimage.gaussian_filter(synth, sigma=1.1)
        noisy = blurred + 0.5 * blurred.std() *
np.random.random(blurred.shape)
        # 对 noisy 沿纵向指数衰减
        noisy = noisy * np.exp(-0.001 *
np.arange(noisy.shape[0])[:, None])

        plt.figure(figsize=(6,3))
        plt.imshow(noisy,
cmap='seismic',interpolation='gaussian',aspec
t='auto' )
        plt.title('Synthetic Seismic Record with
Noise')
        plt.colorbar()
        plt.xticks(np.arange(0, n_cell1+1, 100),
np.arange(range_lift,range_right+1, 100))
        plt.yticks(np.linspace(0, n_cell1, 5),
np.linspace(400, 560,5))
        plt.grid(False)
        plt.show()

plt.savefig('Picture/Synthetic_noisy_gery.png',
dpi=300)
```

聚类算法

Clustel4Seismic.py

```
"""
============================
# -*- coding: utf-8 -*-
# Time       : 2023/3/30 20:58
# Author    : Qisx
# FileName: Clustel4Seismic.py
# Software: PyCharm
============================
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans,AgglomerativeClustering, DBSCAN,SpectralClustering

# 将 matplotlib 字体设置为 Times New Roman
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.rcParams['axes.unicode_minus'] = False


# 写一个标准化化函数
def standardize(x):
    return (x - x.mean()) / x.std()


seismic = pd.read_csv('seismic.csv', sep=',', encoding='utf-8')
seismic = seismic.values
# seismic = standardize(seismic)
x = []

for i in range(0, 498, 5):
    for j in range(0, 500, 20):
        x.append([i, seismic[i][j]])
X = np.array(x)

plt.figure(figsize=(8, 5))
model = AgglomerativeClustering(n_clusters=7, linkage='ward')
model.fit(X)
yhat = model.fit_predict(X)
# plt.subplot(2,2,2)
plt.title('AgglomerativeClustering', fontsize=18)

clusters = np.unique(yhat)
for cluster in clusters:
    row_ix = np.where(yhat == cluster)
    plt.scatter(X[row_ix, 1], X[row_ix, 0])
    plt.ylim(500, 0)
    plt.yticks(np.linspace(0, 500, 5), np.linspace(400, 560, 5), fontsize=18)
    plt.xticks(fontsize=18)
plt.show()
plt.figure()
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
Z = linkage(X, 'ward')
dendrogram(Z)
plt.show()

plt.figure(figsize=(8, 5))
model = KMeans(n_clusters=8)
model.fit(X)
yhat = model.fit_predict(X)
# plt.subplot(2,2,3)
plt.title('K-means', fontsize=18)
clusters = np.unique(yhat)
for cluster in clusters:
    row_ix = np.where(yhat == cluster)
    plt.scatter(X[row_ix, 1], X[row_ix, 0])
    plt.ylim(500, 0)
    plt.yticks(np.linspace(0, 500, 5), np.linspace(400, 560, 5), fontsize=18)
    plt.xticks(fontsize=18)
plt.show()
```

```
plt.figure(figsize=(8, 5))
model=DBSCAN(eps=30,min_samples=200)
model.fit(X)
yhat = model.fit_predict(X)
# plt.subplot(2,2,4)
plt.title('DBSCAN', fontsize=18)
clusters = np.unique(yhat)
for cluster in clusters:
    row_ix = np.where(yhat == cluster)
    plt.scatter(X[row_ix, 1], X[row_ix, 0])
    plt.ylim(500, 0)
    plt.yticks(np.linspace(0, 500, 5), np.linspace(400, 560, 5), fontsize=18)
    plt.xticks(fontsize=18)
plt.show()

plt.figure(figsize=(8, 5))
model= SpectralClustering(n_clusters=8, affinity='nearest_neighbors', n_neighbors=10)
model.fit(X)
yhat = model.fit_predict(X)
# plt.subplot(2,2,4)
plt.title('SpectralClustering', fontsize=18)
clusters = np.unique(yhat)
for cluster in clusters:
    row_ix = np.where(yhat == cluster)
    plt.scatter(X[row_ix, 1], X[row_ix, 0])
    plt.ylim(500, 0)
    plt.yticks(np.linspace(0, 500, 5), np.linspace(400, 560, 5), fontsize=18)
    plt.xticks(fontsize=18)
plt.show()
```

Cluast4Data

```
"""
============================
# -*- coding: utf-8 -*-
# Time       : 2023/4/6 15:31
# Author    : Qisx
# FileName: ClusteDatas.py
# Software: PyCharm
============================
"""
# k-means 聚类
from numpy import unique
from numpy import where
from sklearn.datasets import make_classification
from sklearn.cluster import KMeans,AgglomerativeClustering,DBSCAN,SpectralClustering
import matplotlib.pyplot as plt
from scipy.cluster.hierarchy import dendrogram, linkage
# 将 matplotlib 字体设置为 Times New Roman
plt.rcParams['font.sans-serif'] = ['Times New Roman']
plt.rcParams['axes.unicode_minus'] = False


# 定义数据集
X, _ = make_classification(n_samples=1000, n_features=6, n_informative=2, n_redundant=4, n_clusters_per_class=2, random_state=4)
# 定义模型
model = KMeans(n_clusters=6)
# 模型拟合
model.fit(X)
# 为每个示例分配一个集群
yhat = model.predict(X)
# 检索唯一群集
clusters = unique(yhat)
# 为每个群集的样本创建散点图
plt.figure(figsize=(8, 5))
for cluster in clusters:
# 获取此群集的示例的行索引
    row_ix = where(yhat == cluster)
    # 创建这些样本的散布
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
    plt.title('K-means', fontsize=18)
    plt.xticks(fontsize=18)
    plt.yticks(fontsize=18)
plt.show()
```

```
model=AgglomerativeClustering(n_clusters
=6,linkage='ward')
model.fit(X)
yhat=model.fit_predict(X)
plt.figure(figsize=(8, 5))
clusters = unique(yhat)
for cluster in clusters:
	row_ix = where(yhat == cluster)
	plt.scatter(X[row_ix, 0], X[row_ix, 1])
	plt.title('AgglomerativeClustering',
fontsize=18)
	plt.xticks(fontsize=18)
	plt.yticks(fontsize=18)
plt.show()
plt.figure()
plt.xticks(fontsize=18)
plt.yticks(fontsize=18)
Z = linkage(X)
dendrogram(Z)
plt.show()

model=                    DBSCAN(eps=0.9,
min_samples=100, leaf_size=10)
model.fit(X)
yhat=model.fit_predict(X)
plt.figure(figsize=(8, 5))
clusters = unique(yhat)
for cluster in clusters:
	row_ix = where(yhat == cluster)
	plt.scatter(X[row_ix, 0], X[row_ix, 1])
	plt.title('DBSCAN', fontsize=18)
	plt.xticks(fontsize=18)
	plt.yticks(fontsize=18)
plt.show()

model=SpectralClustering(n_clusters=6,affi
nity='nearest_neighbors',n_neighbors=10)
model.fit(X)
yhat=model.fit_predict(X)
plt.figure(figsize=(8, 5))
clusters = unique(yhat)
for cluster in clusters:
	row_ix = where(yhat == cluster)
	plt.scatter(X[row_ix, 0], X[row_ix, 1])
	plt.title('SpectralClustering',
fontsize=18)
	plt.xticks(fontsize=18)
	plt.yticks(fontsize=18)
plt.show()
```