第三次上机作业

18029100040

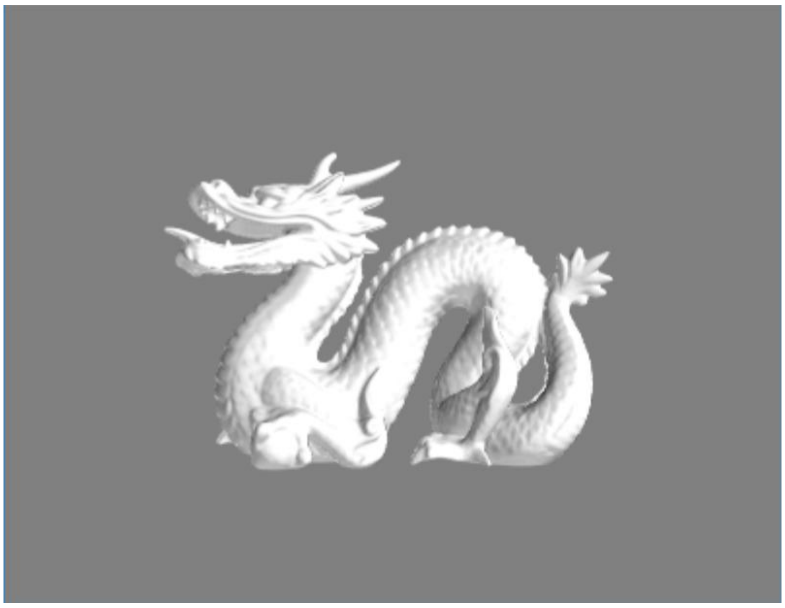
吴程锴

一、作业 19：Dragon 绘制：

1.1 代码

```python
from mayavi import mlab
from os.path import join
import tarfile

# 读取 tar 压缩文件
dragon_tar_file = tarfile.open('dragon.tar.gz')
try:
    os.mkdir('dragon_data')
except:
    pass
dragon_tar_file.extractall('dragon_data')
dragon_tar_file.close()
dragon_ply_file = join('dragon_data', 'dragon_recon', 'dragon_vrip.ply')

# 渲染 dragon ply 文件
mlab.pipeline.surface(mlab.pipeline.open(dragon_ply_file))
mlab.show()

# 删除解压的文件夹
import shutil

shutil.rmtree('dragon_data')
```

1.2 结果

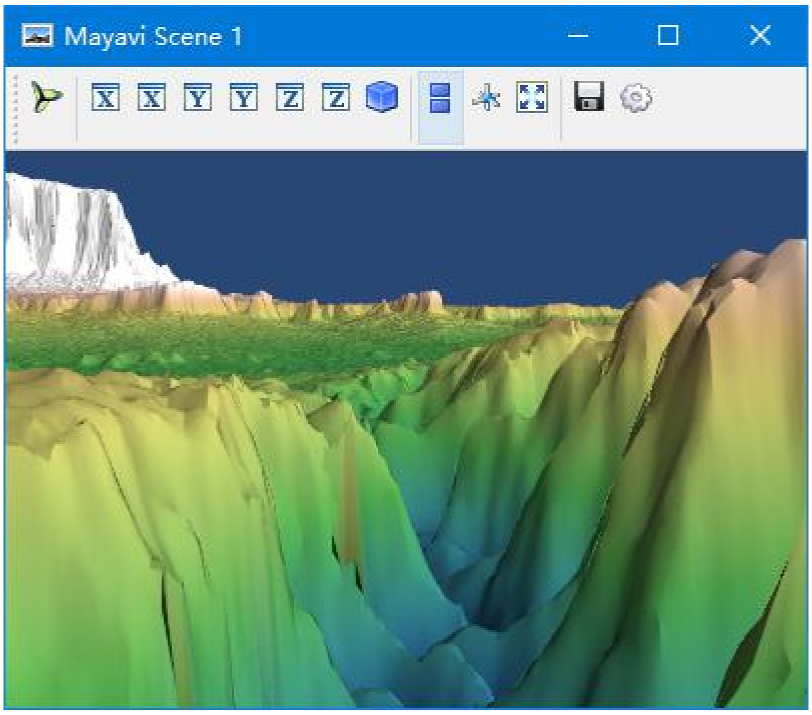


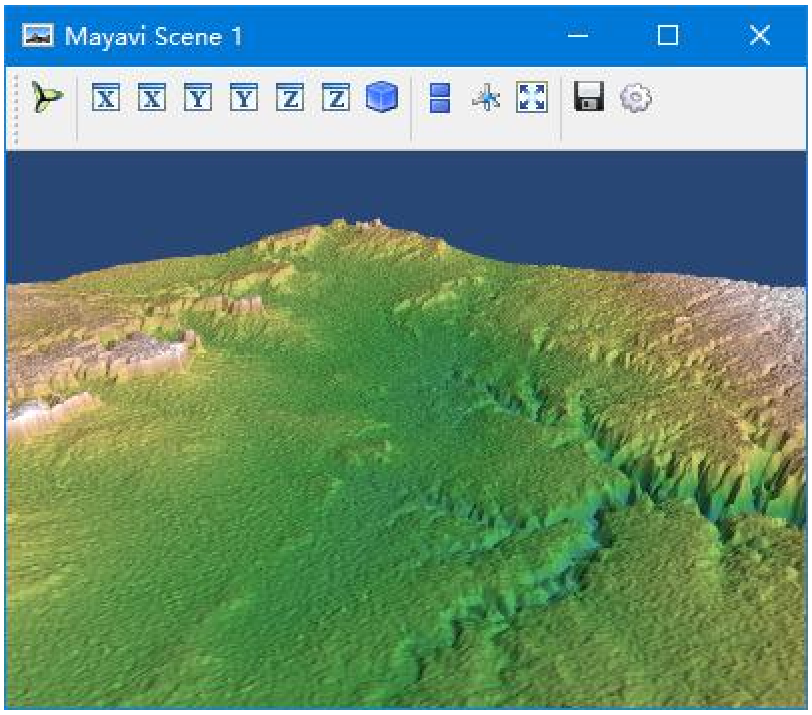
二、作业 20：地形可视化

2.1 代码

```python
import zipfile
import numpy as np
from mayavi import mlab

# 读取压缩文件
hgt = zipfile.ZipFile('5.1.6 N36W113.hgt.zip').read('N36W113.hgt')
data = np.fromstring(hgt, '>i2')
data.shape = (3601, 3601)
data = data.astype(np.float32)
data = data[:1000, 900:1900]
data[data == -32768] = data[data > 0].min()

# 渲染地形 hgt 的数据 data
mlab.figure(size=(400, 320), bgcolor=(0.16, 0.28, 0.46))
mlab.surf(data, colormap='gist_earth', warp_scale=0.2,
          vmin=1200, vmax=1610)

# 清空内存
del data
# 创建交互式的可视化窗口
mlab.view(-5.9, 83, 570, [5.3, 20, 238])
mlab.show()
```

2.2 结果





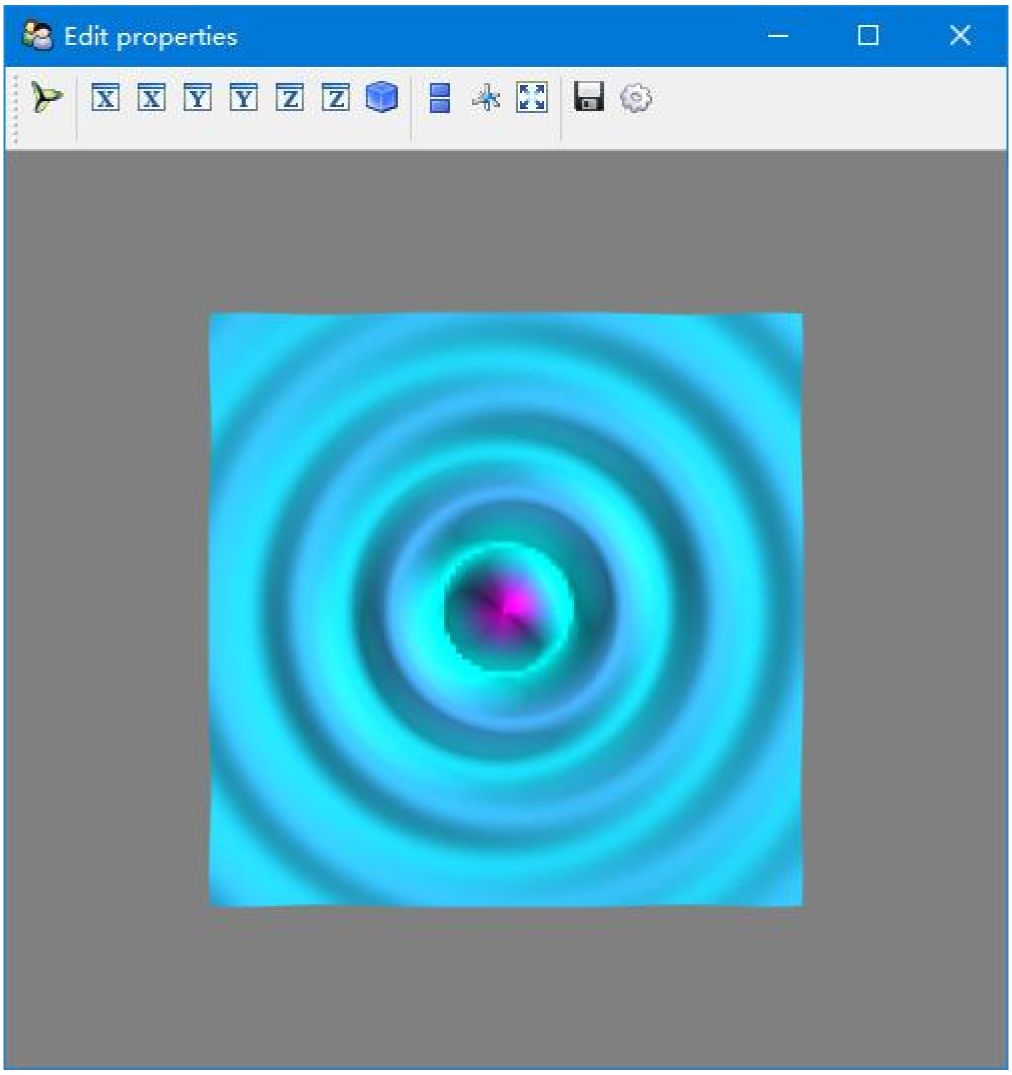
三、作业 21：建立简单的 mayavi 窗口

3.1 代码

```
from numpy import sqrt, sin, mgrid
from traits.api import HasTraits, Instance
```

```python
from traitsui.api import View, Item
from tvtk.pyface.scene_editor import SceneEditor
from mayavi.tools.mlab_scene_model import MlabSceneModel
from mayavi.core.ui.mayavi_scene import MayaviScene

class ActorViewer(HasTraits):
    # 场景模型
    scene = Instance(MlabSceneModel, ())
    # 建立视图
    view = View(Item(name='scene',
                     editor=SceneEditor(scene_class=MayaviScene),
                     show_label=False,
                     resizable=True,
                     height=500,
                     width=500),
                 resizable=True)

    def __init__(self, **traits):
        HasTraits.__init__(self, **traits)
        self.generate_data()

    def generate_data(self):
        # 建立数据
        X, Y = mgrid[-2:2:100j, -2:2:100j]
        R = 10*sqrt(X**2 + Y**2)
        Z = sin(R)/R
        # 绘制数据
        self.scene.mlab.surf(X, Y, Z, colormap='cool')

a = ActorViewer()
a.configure_traits()
```

3.2 结果

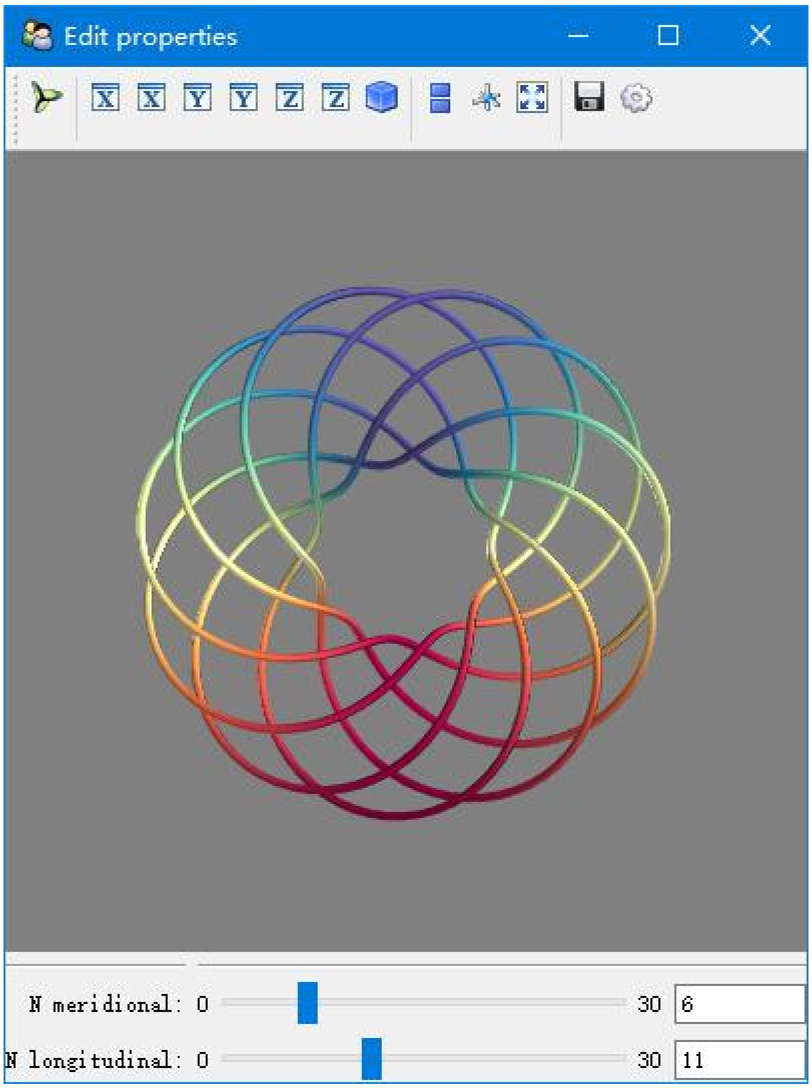


四、作业 22：基于交互控制的 Mayavi 窗口

4.1 代码

```
from traits.api import HasTraits, Range, Instance, on_trait_change
from traitsui.api import View, Item, Group
from mayavi.core.api import PipelineBase
from mayavi.core.ui.api import MayaviScene, SceneEditor, MlabSceneModel

from numpy import arange, pi, cos, sin
dphi = pi/300.
phi = arange(0.0, 2*pi + 0.5*dphi, dphi, 'd')
def curve(n_mer, n_long):
    mu = phi*n_mer
    x = cos(mu) * (1 + cos(n_long * mu/n_mer)*0.5)
    y = sin(mu) * (1 + cos(n_long * mu/n_mer)*0.5)
    z = 0.5 * sin(n_long*mu/n_mer)
    t = sin(mu)
    return x, y, z, t

class MyModel(HasTraits):
    n_meridional    = Range(0, 30, 6)
    n_longitudinal  = Range(0, 30, 11)
    # 场景模型实例
    scene = Instance(MlabSceneModel, ())
    # 管线实例
    plot = Instance(PipelineBase)
    #当场景被激活，或者参数发生改变，更新图形
    @on_trait_change('n_meridional,n_longitudinal,scene.activated')
    def update_plot(self):
        x, y, z, t = curve(self.n_meridional, self.n_longitudinal)
        if self.plot is None:#如果 plot 未绘制则生成 plot3d
            self.plot = self.scene.mlab.plot3d(x, y, z, t,
                        tube_radius=0.025, colormap='Spectral')
        else:#如果数据有变化，将数据更新即重新赋值
            self.plot.mlab_source.set(x=x, y=y, z=z, scalars=t)

    # 建立视图布局
    view = View(Item('scene', editor=SceneEditor(scene_class=MayaviScene),
                     height=250, width=300, show_label=False),
                Group('_', 'n_meridional', 'n_longitudinal'),
                resizable=True)

model = MyModel()
model.configure_traits()
```

4.2 结果



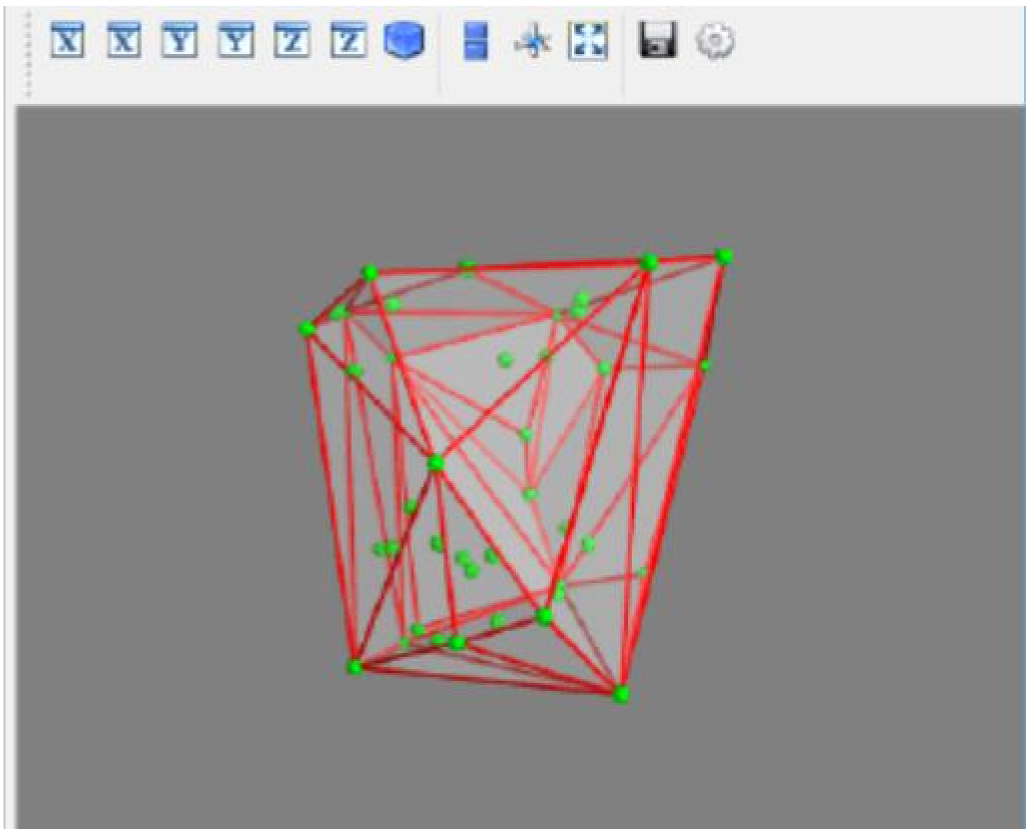
N meridional: 0 30 6
N longitudinal: 0 30 11

五、作业 23：Spatial-ConvexHull 三维凸包可视化

5.1 代码

```python
from tvtk.api import tvtk
import numpy as np
from scipy.spatial import ConvexHull
import matplotlib.pyplot as plt
def convexhull(ch3d): #1 定义凸多面体 tvtk 的 Polydata() 对象
    poly = tvtk.PolyData()
    poly.points = ch3d.points
    poly.polys = ch3d.simplices
    #2 定义凸多面体顶点的小球
    sphere = tvtk.SphereSource(radius = 0.02)
    points3d = tvtk.Glyph3D()
    points3d.set_source_connection(sphere.output_port)
    points3d.set_input_data(poly)
    #3 绘制凸多面体的面，设置半透明度
    m1 = tvtk.PolyDataMapper()
    m1.set_input_data(poly)
    a1 = tvtk.Actor(mapper=m1)
    a1.property.opacity = 0.3
    #4 绘制凸多面体的边，设置为红色
    m2 = tvtk.PolyDataMapper()
    m2.set_input_data(poly)
    a2 = tvtk.Actor(mapper=m2)
    a2.property.representation = 'wireframe'
    a2.property.line_width = 2.0
    a2.property.color = (1.0,0,0)
    #5 绘制凸多面体的顶点，设置为绿色
    m3 = tvtk.PolyDataMapper(input_connection=points3d.output_port)
    a3 = tvtk.Actor(mapper = m3)
    a3.property.color = (0.0,1.0,0.0)
    return [a1,a2,a3]
#4. 定义绘制场景用的函数
def ivtk_scene(actors):
    from tvtk.tools import ivtk
    win = ivtk.IVTKWithCrustAndBrowser() #创建 crust 窗口
    win.open()
    win.scene.add_actor(actors)
    dialog = win.control.centralWidget().widget(0).widget(0) #窗口错误修正
    from pyface.qt import QtCore
    dialog.setWindowFlags(QtCore.Qt.WindowFlags(0x00000000))
    dialog.show()
    return win

np.random.seed(42)
points3d=np.random.rand(40,3)
ch3d=ConvexHull(points3d)
actors=convexhull(ch3d)
win=ivtk_scene(actors)
```

5.2 结果



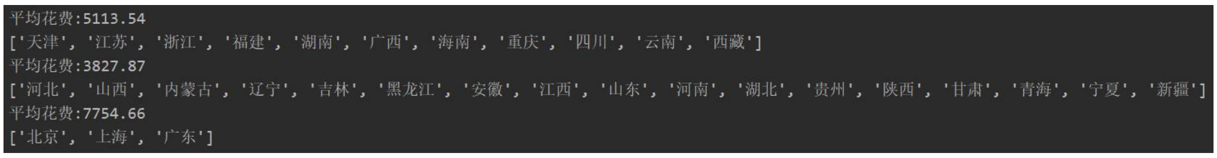
六、作业 24：K-means 聚类

6.1 代码

```
import numpy as np
from sklearn.cluster import KMeans
#定义数据导入方法
def loadData(filePath):
    fr=open(filePath,'r+')  #读写打开一个文本文件
    lines=fr.readlines()    #一次读取整 个文件（类似于.read())
    retData=[]              #城市各项信息
    retCityName=[]          #城市名称
    for line in lines:
        items=line.strip().split(",")
        retCityName.append(items[0])
        retData.append([items[i] for i in range(1,len(items))])
        #print(retCityName)
    return retData,retCityName


#加载数据，创建 K-means 算法实例，并进行训练，获得标签
```

```
if __name__=='__main__':
    filepath='city.txt'
    data,cityName=loadData(filepath)  #利用 loadData 方法读取数据
    km=KMeans(n_clusters=3)              #创建实例
    label=km.fit_predict(data)          #调用 Kmeans() fit_predict()方法进行计算
    expenses=np.sum(km.cluster_centers_,axis=1)
    #print(expenses)
    CityCluster=[[],[],[]]      #将城市 按 label 分成设定的簇，将每个簇的城市输出
    for i in range(len(cityName)):
        CityCluster[label[i]].append(cityName[i])
    for i in range(len(CityCluster)):
        print("Expenses:%.2f" %expenses[i])
        print(CityCluster[i])
```

6.2 结果



```
平均花费:5113.54
['天津', '江苏', '浙江', '福建', '湖南', '广西', '海南', '重庆', '四川', '云南', '西藏']
平均花费:3827.87
['河北', '山西', '内蒙古', '辽宁', '吉林', '黑龙江', '安徽', '江西', '山东', '河南', '湖北', '贵州', '陕西', '甘肃', '青海', '宁夏', '新疆']
平均花费:7754.66
['北京', '上海', '广东']
```

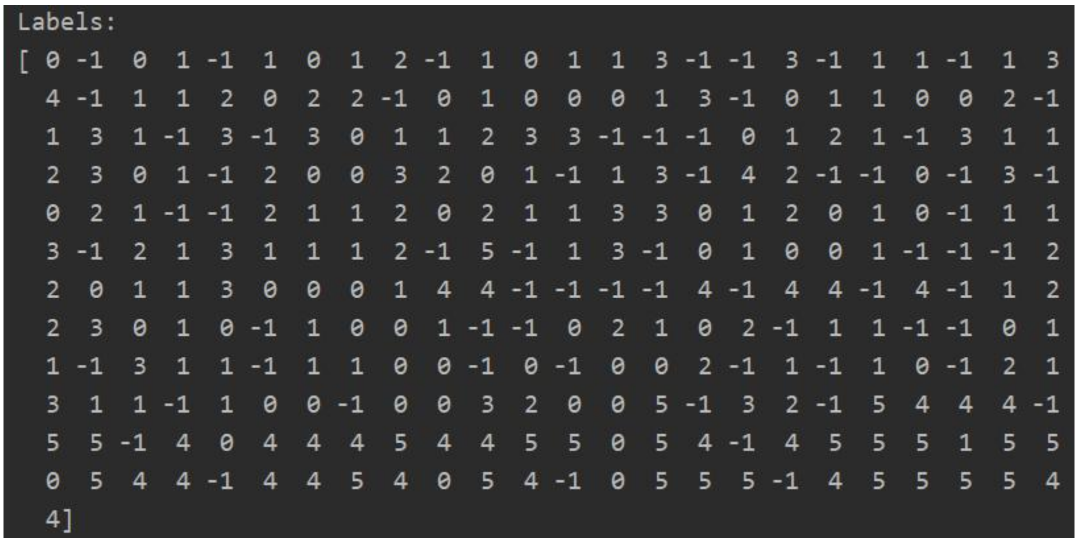
七、作业 25：DBSCAN 聚类

7.1 代码

```
import numpy as np
import sklearn.cluster as skc
from sklearn import metrics
import matplotlib.pyplot as plt

mac2id = dict()
onlinetimes = []
f = open(r'学生月上网时间分布-TestData.txt', encoding='utf-8')
for line in f:
    # 读取每条数据中的 mac 地址，开始上网时间，上网时长
    mac = line.split(',')[2]
    onlinetime = int(line.split(',')[6])
    starttime = int(line.split(',')[4].split(' ')[1].split(':')[0])

    # mac2id 是一个字典：key 是 mac 地址，value 是对应 mac 地址的上网时长以及开始上网时间
    if mac not in mac2id:
        mac2id[mac] = len(onlinetimes)
        onlinetimes.append((starttime, onlinetime))
    else:
```

```python
        onlinetimes[mac2id[mac]] = [(starttime, onlinetime)]
real_X = np.array(onlinetimes).reshape((-1, 2))

# 调用 DBSCAN 方法训练，labels 为每个数据的簇标签
X = real_X[:, 0:1]
db = skc.DBSCAN(eps=0.01, min_samples=20).fit(X)
labels = db.labels_

# 打印数据被记上的标签，计算标签为-1,即噪声数据的比例
print('Labels:')
print(labels)
raito = len(labels[labels[:] == -1]) / len(labels)
print('Noise raito:', format(raito, '.2%'))

# 计算簇的个数并打印，评价聚类效果
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
print("Silhouette Coefficient: %0.3f" % metrics.silhouette_score(X, labels))

# 打印各簇标号以及各簇内数据
for i in range(n_clusters_):
    print('Cluster ', i, ':')
    print(list(X[labels == i].flatten()))

plt.hist(X, 24)
plt.show()
```

7.2 结果



```
Labels:
[ 0 -1  0  1 -1  1  0  1  2 -1  1  0  1  1  3 -1 -1  3 -1  1  1 -1  1  3
  4 -1  1  1  2  0  2  2 -1  0  1  0  0  0  1  3 -1  0  1  1  0  0  2 -1
  1  3  1 -1  3 -1  3  0  1  1  2  3  3 -1 -1 -1  0  1  2  1 -1  3  1  1
  2  3  0  1 -1  2  0  0  3  2  0  1 -1  1  3 -1  4  2 -1 -1  0 -1  3 -1
  0  2  1 -1 -1  2  1  1  2  0  2  1  1  3  3  0  1  2  0  1  0 -1  1  1
  3 -1  2  1  3  1  1  1  2 -1  5 -1  1  3 -1  0  1  0  0  1 -1 -1 -1  2
  2  0  1  1  3  0  0  0  1  4  4 -1 -1 -1 -1  4 -1  4  4 -1  4 -1  1  2
  2  3  0  1  0 -1  1  0  0  1 -1 -1  0  2  1  0  2 -1  1  1 -1 -1  0  1
  1 -1  3  1  1 -1  1  1  0  0 -1  0 -1  0  0  2 -1  1 -1  1  0 -1  2  1
  3  1  1 -1  1  0  0 -1  0  0  3  2  0  0  5 -1  3  2 -1  5  4  4  4 -1
  5  5 -1  4  0  4  4  4  5  4  4  5  5  0  5  4 -1  4  5  5  5  1  5  5
  0  5  4  4 -1  4  4  5  4  0  5  4 -1  0  5  5  5 -1  4  5  5  5  5  4
  4]
```

```
Noise raito: 22.15%
```

```
Estimated number of clusters: 6
Silhouette Coefficient: 0.710
Cluster  0 :
[22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22]
Cluster  1 :
[23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23,
 23, 23]
Cluster  2 :
[20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
 20, 20, 20, 20, 20, 20, 20]
Cluster  3 :
[21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21, 21,
 21, 21, 21, 21]
Cluster  4 :
[8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8]

Cluster  5 :
[7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7]
```