西安交通大學

XI'AN JIAOTONG UNIVERSITY

本科实验报告

系统传输函数零极点分析

课程名称：	数字信号处理
姓名：	饭饭
学院：	生命学院
专业：	生物工程
学号：	**2182333333**
指导老师：	**Hao**

2021 年 5 月 29 日

西安交通大學实验报告

专业：生物工程
姓名：饭饭
学号：2182333333
日期：2021 年 5 月 29 日
地点：寝室

课程名称：数字信号处理　指导老师：Hao　成绩：59
实验名称：系统传输函数零极点分析　实验类型：设计实验　同组学生姓名：Bob

一、 实验目的和要求

系统差分方程和传输函数是线性系统的重要概念，通过分析系统差分方程和传输函数的特性，编程查看系统零极点分布，加深对线性系统的了解。

二、 实验内容和步骤

1. 实验内容

给出如下差分方程：

$$y(n) - (0.5 + a) \times y(n-1) + 0.5ay(n-2) = x(n)$$

(1) 求解系统传输函数表达式。

(2) 当 a 取 $0.8, 0.9, 1.0, 1.1$ 时，画出零极点分布图。

(3) 根据 **??** 中 a 的取值，分别画出幅频响应函数图像。

2. 实验步骤

(1) 编写程序，求解零极点

(2) 画出图形。

(3) 观察结果。

三、 主要仪器设备

计算机，Matlab 软件

四、 操作方法和实验步骤

1. 传输函数

对差分方程进行处理，求出传输函数表达式。

2. 零极点分布图

在此基础上，使用 Matlab 中的 `zplane` 函数进一步画出在不同 a 取值情况下的零极点分布图。

3. 幅频响应

之后使用 `freqz` 函数画出不同 a 取值情况下的频率响应图像。

五、 实验数据记录和处理

1. 传输函数

根据差分方程，传输函数如下：

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^2}{z^2 - (0.5+a)z + 0.5a}$$

2. 零极点分布图

$a = 0.8, 0.9, 1.1$ 时，系统的零极点分布图及程序如下：

(1) 图像如图 **??** 所示。

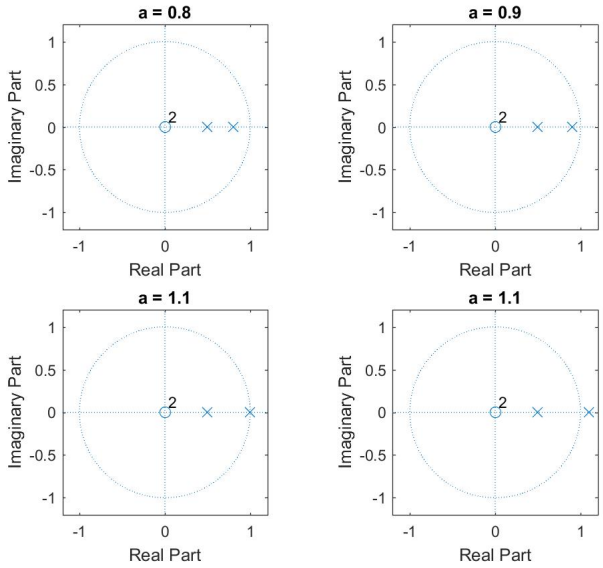


图 1: 系统的零极点分布图

(2) 代码

```
clc;clear;

B = [1 0 0];

```

```
subplot(2,2,1);
a = 0.8;
A = [1 -(0.5+a) 0.5*a];
zplane(B, A);
axis([-1.2 1.2 -1.2 1.2]);
title('a = 0.8');

subplot(2,2,2);
a = 0.9;
A = [1 -(0.5+a) 0.5*a];
zplane(B, A);
axis([-1.2 1.2 -1.2 1.2]);
title('a = 0.9');

subplot(2,2,3);
a = 1.0;
A = [1 -(0.5+a) 0.5*a];
zplane(B, A);
axis([-1.2 1.2 -1.2 1.2]);
title('a = 1.1');

subplot(2,2,4);
a = 1.1;
A = [1 -(0.5+a) 0.5*a];
zplane(B, A);
axis([-1.2 1.2 -1.2 1.2]);
title('a = 1.1');
```

3. 频率响应

$a = 0.8, 0.9, 1.0, 1.1$ 时，系统的频率响应函数图形及程序如下：

(1) 图像如图 **??** 所示。

(2) 代码

```
clc;clear;

B = [1 0 0];

figure;
a = 0.8;
A = [1 -(0.5+a) 0.5*a];
freqz(B, A);
title('a = 0.8');

figure;
a = 0.9;
A = [1 -(0.5+a) 0.5*a];
```

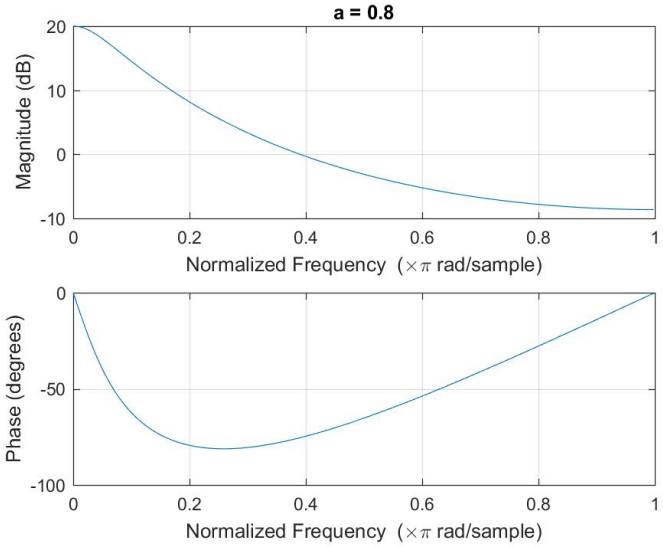


图 2: 系统的频率响应函数图形

```
freqz(B, A);
title('a = 0.9');

figure;
a = 1.0;
A = [1 -(0.5+a) 0.5*a];
freqz(B, A);
title('a = 1.0');

figure;
a = 1.1;
A = [1 -(0.5+a) 0.5*a];
freqz(B, A);
title('a = 1.1');
```

六、 实验结果与分析

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras sit amet pharetra orci. Pellentesque ex est, ultricies non viverra sed, iaculis sed est. Suspendisse ex felis, aliquam et nulla in, vehicula gravida velit. In porttitor volutpat tincidunt. Aenean eleifend non augue sit amet ultrices. Proin rutrum odio vitae est luctus, eu facilisis leo dapibus. Aliquam commodo efficitur erat, sit amet pulvinar nulla ultrices ac. Vestibulum ullamcorper malesuada odio. Suspendisse id sollicitudin arcu. Nam fringilla commodo neque quis posuere. Sed lobortis justo nisl, ut molestie sapien semper ac. Proin in massa vel neque tempus porta id accumsan magna. Aliquam pharetra lacus ac arcu accumsan varius. Praesent condimentum mi vitae purus elementum imperdiet.

Etiam in bibendum arcu. Etiam porta metus ligula. Cras tempor, leo vel faucibus consequat, diam justo malesuada ex, et pretium risus est sit amet lectus. In sit amet efficitur quam. Maecenas

quis molestie odio, et consectetur augue. Mauris erat erat, tristique a nunc in, molestie molestie lectus. Nullam commodo ornare urna, vel tincidunt libero aliquam ut. Ut hendrerit nunc id sapien tristique mollis. Aenean maximus, neque molestie dignissim cursus, augue neque dictum purus, vitae varius diam ex vitae est. Nulla bibendum facilisis semper. Nulla placerat ultricies lorem, quis consequat metus pulvinar eu. Morbi sed massa arcu. Nulla sagittis felis a suscipit elementum.

时间	双蒸水-1	双蒸水-2	双蒸水-3	待测样本-1	待测样本-2	待测样本-3	双蒸水	待测样本
20s	0.1628	0.1666	0.1617	1.3338	1.3826	1.3744	0.1637	1.3636
80s	0.1630	0.1669	0.1619	1.3626	1.4130	1.4054	0.1639	1.3936
ΔA							0.0002	0.0300

表 1: 吸光度数据

May 3, 2021

使用 **sklearn** 完成以下实验项目，实验项目所用数据集从思源学堂的实验中下载

[39]:
```
from sklearn.datasets import load_wine
from sklearn.preprocessing import MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import fowlkes_mallows_score
from sklearn.metrics import silhouette_score
from sklearn.metrics import calinski_harabasz_score
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.linear_model import LinearRegression
from sklearn.metrics import explained_variance_score,␣
↪mean_absolute_error,mean_squared_error,median_absolute_error,r2_score
from sklearn.ensemble import GradientBoostingRegressor

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

scaler = MinMaxScaler()
svm = SVC()
clf = LinearRegression()
gbr = GradientBoostingRegressor()
```

实验项目 1

1）使用 sklearn 读取数据集 wine

[40]:
```
from sklearn.datasets import load_wine
wine = load_wine()
```

2）拆分数据集 wine 的数据和标签（class）

[41]:
```
wine_data = wine['data']
wine_target = wine['target']
print(wine_data, wine_target)
```

```
[[1.423e+01 1.710e+00 2.430e+00 … 1.040e+00 3.920e+00 1.065e+03]
 [1.320e+01 1.780e+00 2.140e+00 … 1.050e+00 3.400e+00 1.050e+03]
```

```
 [1.316e+01 2.360e+00 2.670e+00 … 1.030e+00 3.170e+00 1.185e+03]
 …
 [1.327e+01 4.280e+00 2.260e+00 … 5.900e-01 1.560e+00 8.350e+02]
 [1.317e+01 2.590e+00 2.370e+00 … 6.000e-01 1.620e+00 8.400e+02]
 [1.413e+01 4.100e+00 2.740e+00 … 6.100e-01 1.600e+00 5.600e+02]] [0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

3）对数据集 wine 进行标准化

```
[42]: wine_data = scaler.fit_transform(wine_data)
      wine_data
```

```
[42]: array([[0.84210526, 0.1916996 , 0.57219251, …, 0.45528455, 0.97069597,
         0.56134094],
        [0.57105263, 0.2055336 , 0.4171123 , …, 0.46341463, 0.78021978,
         0.55064194],
        [0.56052632, 0.3201581 , 0.70053476, …, 0.44715447, 0.6959707 ,
         0.64693295],
        …,
        [0.58947368, 0.69960474, 0.48128342, …, 0.08943089, 0.10622711,
         0.39728959],
        [0.56315789, 0.36561265, 0.54010695, …, 0.09756098, 0.12820513,
         0.40085592],
        [0.81578947, 0.66403162, 0.73796791, …, 0.10569106, 0.12087912,
         0.20114123]])
```

4）对数据集 wine 进行 PCA 降维

```
[43]: wine_pca = PCA(n_components=10).fit_transform(wine_data)
      print('before PCA:', wine_data.shape)
      print('after PCA:', wine_pca.shape)
```

```
before PCA: (178, 13)
after PCA: (178, 10)
```

5）构建聚类数目为 3 的 K-Means 模型

```
[44]: wine_kmeans = KMeans(n_clusters = 3).fit(wine_data)
      wine_kmeans
```

```
[44]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
        n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
        random_state=None, tol=0.0001, verbose=0)
```

6）使用 FMI 评价聚类模型

[45]:
```
score = fowlkes_mallows_score(wine_target, wine_kmeans.labels_)
score
```

[45]: 0.8914327267284605

7）确定最佳聚类数目（2~10 类）

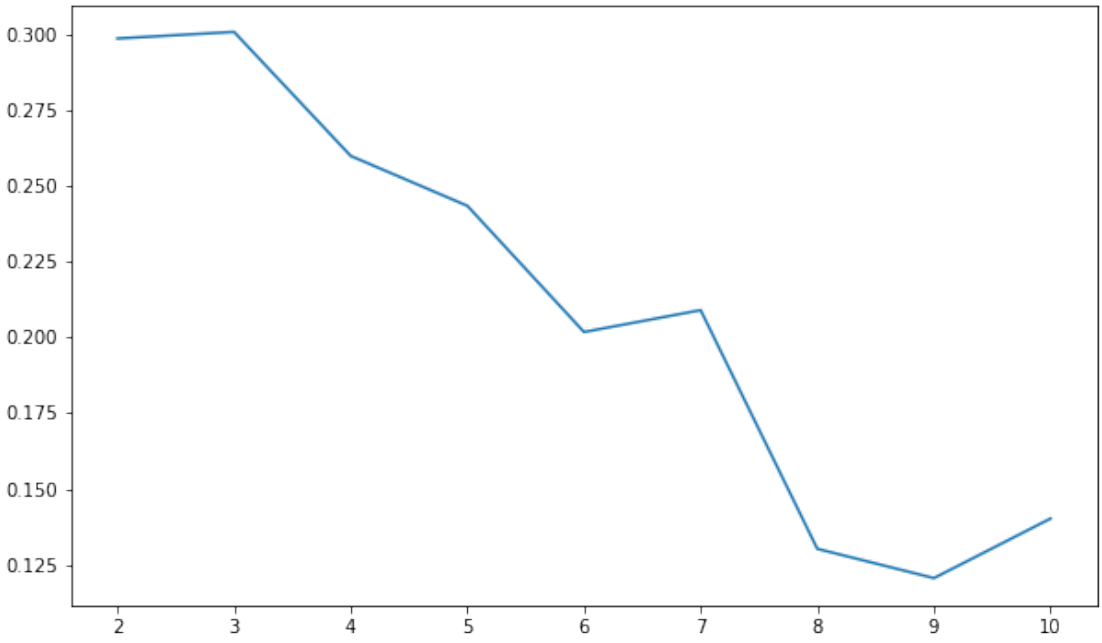
[46]:
```
for i in range(2,11):
    kmeans = KMeans(n_clusters = i).fit(wine_data)
    score = fowlkes_mallows_score(wine_target,kmeans.labels_)
    print('{}评价分值为: {}'.format(i, score))
```

2 评价分值为: 0.6428941723110392
3 评价分值为: 0.9026207781786737
4 评价分值为: 0.814628876858432
5 评价分值为: 0.7594489413791359
6 评价分值为: 0.7144734035621569
7 评价分值为: 0.7436519237861382
8 评价分值为: 0.5825510050786507
9 评价分值为: 0.6790142214965376
10 评价分值为: 0.5194805742858095

因此最佳数目为 3

8）使用轮廓系数评价聚类模型

[47]:
```
silhouettteScore = []
for i in range(2,11):
    kmeans = KMeans(n_clusters = i).fit(wine_data)
    score = silhouette_score(wine_data,kmeans.labels_)
    silhouettteScore.append(score)
plt.figure(figsize=(10,6))
plt.plot(range(2,11),silhouettteScore,linewidth=1.5, linestyle="-")
plt.show()
```



9）使用 Calinski-Harabasz 指数评价聚类模型

[48]:
```
for i in range(2,11):
    kmeans = KMeans(n_clusters = i).fit(wine_data)
    score = calinski_harabasz_score(wine_data,kmeans.labels_)
    print('聚{}类 calinski_harabaz 指数为：{}'.format(i, score))
```

```
聚 2 类 calinski_harabaz 指数为：84.7085044440733
聚 3 类 calinski_harabaz 指数为：83.37374750844354
聚 4 类 calinski_harabaz 指数为：65.6072858687198
聚 5 类 calinski_harabaz 指数为：54.86980191194707
聚 6 类 calinski_harabaz 指数为：47.47436301767781
聚 7 类 calinski_harabaz 指数为：43.327230206209855
聚 8 类 calinski_harabaz 指数为：39.89936971821639
聚 9 类 calinski_harabaz 指数为：37.74944751126178
聚 10 类 calinski_harabaz 指数为：34.68847492267841
```

因此最佳数目为 3

实验项目 2

1）使用 sklearn 读取数据集 wine

2）拆分数据集 wine 的数据和标签（class）

3）将数据集 wine 划分为训练集和测试集

[49]:
```
wine_data_train,wine_data_test,wine_target_train,wine_target_test=train_test_split(wine_data,w
↪2)
```

```
print(wine_data_train.shape, wine_target_train.shape)
print(wine_data_test.shape, wine_target_test.shape)
```

```
(142, 13) (142,)
(36, 13) (36,)
```

4）使用离差标准化标准化数据集

[50]:
```
wine_scaler = scaler.fit(wine_data_train)
wine_data_train = wine_scaler.transform(wine_data_train)
wine_data_test = wine_scaler.transform(wine_data_test)
```

5）构建 SVM 模型

[51]:
```
wine_svm = svm.fit(wine_data_train, wine_target_train)
wine_svm
```

[51]:
```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

6）给出评价分类模型性能的分类报告

[52]:
```
pred = wine_svm.predict(wine_data_test)
print(pred)
print(wine_target_test)
pred_true = np.sum(pred == wine_target_test)
print('预测准确率为:', pred_true/wine_target_test.shape[0])
```

```
[0 1 2 1 1 1 0 0 2 2 1 2 1 0 2 1 0 1 2 1 1 2 2 0 2 1 0 2 0 2 2 1 0 1 0 1]
[0 1 2 1 1 1 0 0 2 2 1 2 1 0 2 1 0 1 2 1 1 2 2 0 2 2 0 2 0 2 2 1 0 1 0 1]
预测准确率为: 0.9722222222222222
```

实验项目 3

1）使用 sklearn 读取数据集 winequality

[53]:
```
wineq = pd.read_csv('./work/winequality.csv', delimiter=';')
wineq.head()
```

[53]:

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56
1	7.8	0.88	0.00	2.6	0.098					
2	7.8	0.76	0.04	2.3	0.092					
3	11.2	0.28	0.56	1.9	0.075					
4	7.4	0.70	0.00	1.9	0.076					

```
1                 25.0                 67.0   0.9968  3.20       0.68
2                 15.0                 54.0   0.9970  3.26       0.65
3                 17.0                 60.0   0.9980  3.16       0.58
4                 11.0                 34.0   0.9978  3.51       0.56

   alcohol  quality
0      9.4        5
1      9.8        5
2      9.8        5
3      9.8        6
4      9.4        5
```

2）拆分数据集 winequality 的数据和标签（quality）

[54]:
```
wineq_data = wineq.iloc[:, :-1]
wineq_data
```

[54]:
```
      fixed acidity  volatile acidity  citric acid  residual sugar  chlorides  \
0               7.4             0.700         0.00             1.9      0.076
1               7.8             0.880         0.00             2.6      0.098
2               7.8             0.760         0.04             2.3      0.092
3              11.2             0.280         0.56             1.9      0.075
4               7.4             0.700         0.00             1.9      0.076
…               …                 …            …                …        …
1594            6.2             0.600         0.08             2.0      0.090
1595            5.9             0.550         0.10             2.2      0.062
1596            6.3             0.510         0.13             2.3      0.076
1597            5.9             0.645         0.12             2.0      0.075
1598            6.0             0.310         0.47             3.6      0.067

      free sulfur dioxide  total sulfur dioxide  density    pH  sulphates  \
0                    11.0                  34.0  0.99780  3.51       0.56
1                    25.0                  67.0  0.99680  3.20       0.68
2                    15.0                  54.0  0.99700  3.26       0.65
3                    17.0                  60.0  0.99800  3.16       0.58
4                    11.0                  34.0  0.99780  3.51       0.56
…                     …                     …       …   …          …
1594                 32.0                  44.0  0.99490  3.45       0.58
1595                 39.0                  51.0  0.99512  3.52       0.76
1596                 29.0                  40.0  0.99574  3.42       0.75
1597                 32.0                  44.0  0.99547  3.57       0.71
1598                 18.0                  42.0  0.99549  3.39       0.66

      alcohol
0         9.4
1         9.8
2         9.8
```

```
3         9.8
4         9.4
…         …
1594     10.5
1595     11.2
1596     11.0
1597     10.2
1598     11.0

[1599 rows x 11 columns]
```

[55]:
```
wineq_target = wineq.iloc[:, -1]
wineq_target
```

[55]:
```
0       5
1       5
2       5
3       6
4       5
       ..
1594    5
1595    6
1596    6
1597    5
1598    6
Name: quality, Length: 1599, dtype: int64
```

3) 将数据集 winequality 划分为训练集和测试集

[56]:
```
wineq_data_train,wineq_data_test,wineq_target_train,wineq_target_test =␣
 ↪train_test_split(wineq_data,wineq_target,test_size = 0.2)
```

4) 构建线性回归模型

[57]:
```
wineq_clf = clf.fit(wineq_data_train, wineq_target_train)
wineq_clf_pred = wineq_clf.predict(wineq_data_test)
```

5) 计算线性回归模型的平均绝对误差、均方误差、中值绝对误差、可解释方差和 R^2

[58]:
```
print('线性回归模型的平均绝对误差为: ',mean_absolute_error(wineq_target_test,␣
 ↪wineq_clf_pred))
print('线性回归模型的均方误差为: ',mean_squared_error(wineq_target_test,␣
 ↪wineq_clf_pred))
print('线性回归模型的中值绝对误差为: ',median_absolute_error(wineq_target_test,␣
 ↪wineq_clf_pred))
print('线性回归模型的可解释方差值为:
',explained_variance_score(wineq_target_test, wineq_clf_pred))
print('线性回归模型的 R 方值为: ',r2_score(wineq_target_test, wineq_clf_pred))
```

线性回归模型的平均绝对误差为： 0.5285013191278756
线性回归模型的均方误差为： 0.47854963955561286
线性回归模型的中值绝对误差为： 0.4058838482394336
线性回归模型的可解释方差值为： 0.3257707164209458
线性回归模型的 R 方值为： 0.32538329147572576

6) 构建梯度提升回归模型

[59]:
```
wineq_gbr = gbr.fit(wineq_data_train, wineq_target_train)
wineq_gbr_pred = wineq_gbr.predict(wineq_data_test)
```

7）计算梯度提升回归模型的平均绝对误差、均方误差、中值绝对误差、可解释方差和 R^2

[60]:
```
print('梯度提升回归模型的平均绝对误差为：',mean_absolute_error(wineq_target_test,␣
 ↪wineq_gbr_pred))
print('梯度提升回归模型的均方误差为：',mean_squared_error(wineq_target_test,␣
 ↪wineq_gbr_pred))
print('梯度提升回归模型的中值绝对误差为：
',median_absolute_error(wineq_target_test, wineq_gbr_pred))
print('梯度提升回归模型的可解释方差值为：
',explained_variance_score(wineq_target_test, wineq_gbr_pred))
print('梯度提升回归模型的 R 方值为：',r2_score(wineq_target_test,␣
 ↪wineq_gbr_pred))
```

梯度提升回归模型的平均绝对误差为： 0.5042051809687346
梯度提升回归模型的均方误差为： 0.44546257925492155
梯度提升回归模型的中值绝对误差为： 0.38486626423760395
梯度提升回归模型的可解释方差值为： 0.3724670013117275
梯度提升回归模型的 R 方值为： 0.3720264855559139

七、 code

1. Python

```python
    try:
    input_arr = list(map(float, input("Please input the hetght(m) and weight(kg)
        number:").split()))
    if min(input_arr) < 0:
        raise Exception("The numbers should more than 0.")
    if len(input_arr) != 2:
        raise Exception("Should input 2 numbers")
except ValueError:
    print("The input is not number.")
except Exception as e:
    print(e)
else:
    BMI = input_arr[1] / input_arr[0]**2

    print("Your BMI is {0}".format(round(BMI, 1)))

    if BMI <18.5:
        print("Your body is Underweight.")
    elif BMI<23.9:
        print("Your body is Normal.")
    elif BMI<27.9:
        print("Your body is Overweight")
    else:
        print("Your body is Obesity.")
```

2. R code

```r
    # 绘制图形
    rmax <- 3
    # 设置图形分行方式和图形空白边距,并保存原有的方式
    op <- par(mfrow = c(rmax,1), mar = .1+ c(2,2,2,1))
    for(r in 1:rmax){
      tstr <- sprintf("r=%d",r)
      ricker(0.005, r);title(tstr)
    }
    par(op) # 恢复默认图形页面设置
```