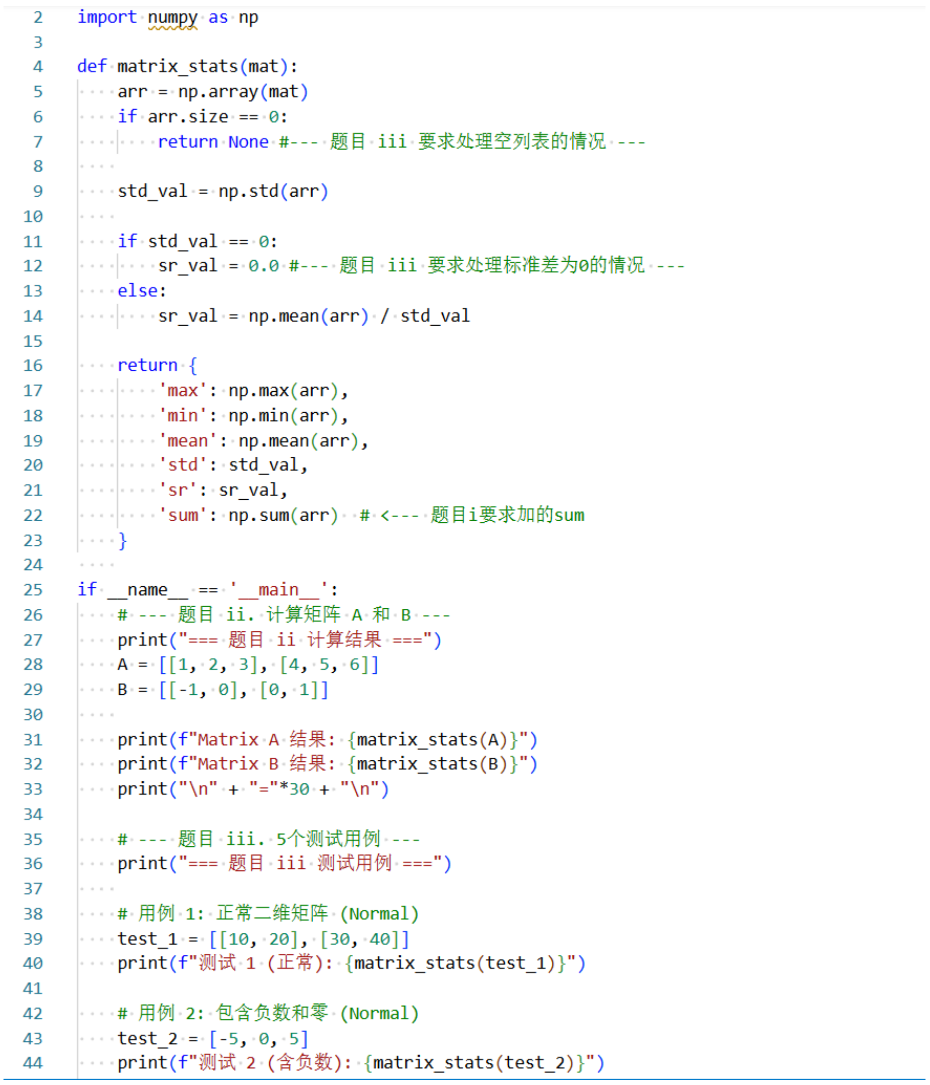
计算机应用课程 final exam 第二部分

杨雅婷　物理学系

一、 Python 代码测试

代码部分



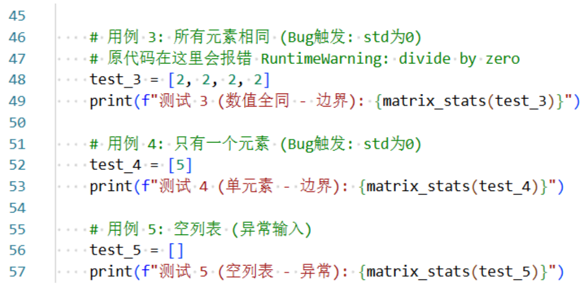
```python
import numpy as np

def matrix_stats(mat):
    arr = np.array(mat)
    if arr.size == 0:
        return None #--- 题目 iii 要求处理空列表的情况 ---
    
    std_val = np.std(arr)
    
    if std_val == 0:
        sr_val = 0.0 #--- 题目 iii 要求处理标准差为0的情况 ---
    else:
        sr_val = np.mean(arr) / std_val

    return {
        'max': np.max(arr),
        'min': np.min(arr),
        'mean': np.mean(arr),
        'std': std_val,
        'sr': sr_val,
        'sum': np.sum(arr)  # <--- 题目i要求加的sum
    }
    
if __name__ == '__main__':
    # --- 题目 ii. 计算矩阵 A 和 B ---
    print("=== 题目 ii 计算结果 ===")
    A = [[1, 2, 3], [4, 5, 6]]
    B = [[-1, 0], [0, 1]]
    
    print(f"Matrix A 结果: {matrix_stats(A)}")
    print(f"Matrix B 结果: {matrix_stats(B)}")
    print("\n" + "="*30 + "\n")

    # --- 题目 iii. 5个测试用例 ---
    print("=== 题目 iii 测试用例 ===")
    
    # 用例 1: 正常二维矩阵 (Normal)
    test_1 = [[10, 20], [30, 40]]
    print(f"测试 1 (正常): {matrix_stats(test_1)}")

    # 用例 2: 包含负数和零 (Normal)
    test_2 = [-5, 0, 5]
    print(f"测试 2 (含负数): {matrix_stats(test_2)}")
```



```python

    # 用例 3: 所有元素相同 (Bug触发: std为0)
    # 原代码在这里会报错 RuntimeWarning: divide by zero
    test_3 = [2, 2, 2, 2]
    print(f"测试 3 (数值全同 - 边界): {matrix_stats(test_3)}")

    # 用例 4: 只有一个元素 (Bug触发: std为0)
    test_4 = [5]
    print(f"测试 4 (单元素 - 边界): {matrix_stats(test_4)}")

    # 用例 5: 空列表 (异常输入)
    test_5 = []
    print(f"测试 5 (空列表 - 异常): {matrix_stats(test_5)}")
```

运行输出

```
=== 题目 ii 计算结果 ===
Matrix A 结果:
'max': np.int64(6), 'min': np.int64(1), 'mean': np.float64(3.5),
'std': np.float64(1.707825127659933), 'sr': np.float64(2.04939015319192), 'sum': np.int64(21)
Matrix B 结果:
'max': np.int64(1), 'min': np.int64(-1), 'mean': np.float64(0.0),
'std': np.float64(0.7071067811865476),
'sr': np.float64(0.0), 'sum': np.int64(0)
==============================
=== 题目 iii 测试用例 ===
测试 1 (正常):
'max': np.int64(40), 'min': np.int64(10), 'mean':np.float64(25.0),
'std': np.float64(11.180339887498949),
'sr': np.float64(2.23606797749979), 'sum': np.int64(100)
测试 2 (含负数):
'max': np.int64(5), 'min': np.int64(-5), 'mean': np.float64(0.0),
'std': np.float64(4.08248290463863), 'sr': np.float64(0.0), 'sum': np.int64(0)
测试 3 (数值全同 - 边界):
'max': np.int64(2), 'min': np.int64(2), 'mean': np.float64(2.0),
'std': np.float64(0.0), 'sr': 0.0, 'sum': np.int64(8)
测试 4 (单元素 - 边界):
'max': np.int64(5), 'min': np.int64(5), 'mean': np.float64(5.0),
'std': np.float64(0.0), 'sr': 0.0, 'sum': np.int64(5)
测试 5 (空列表 - 异常): None
```

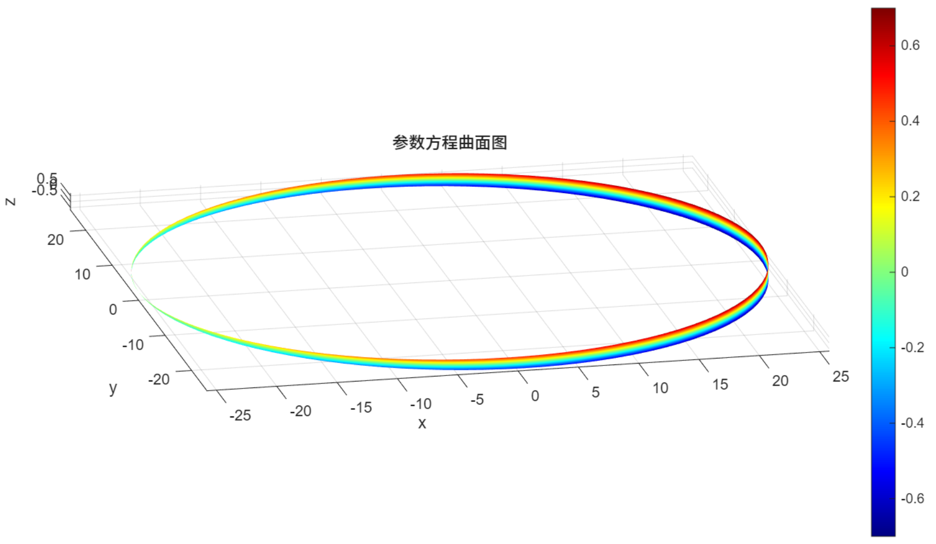
二、 Matlab 作图

代码内容

```
r = 0.7;
R = 3;
```

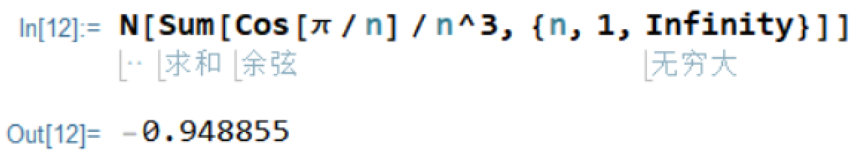
```
[u, v] = meshgrid(linspace(-r, r, 100), linspace(0, 2*pi, 100));
Part_A = R + u*cos(v/2);
x = cos(v) .* (Part_A); y = sin(v) .* (Part_A); z = u.*sin(v/2);
figure;
surf(x, y, z);
shading interp;
axis equal;
colormap jet;
xlabel('x'); ylabel('y'); zlabel('z');
title(' 参数方程曲面图');
```

输出图像



三、 Mathematica 求值

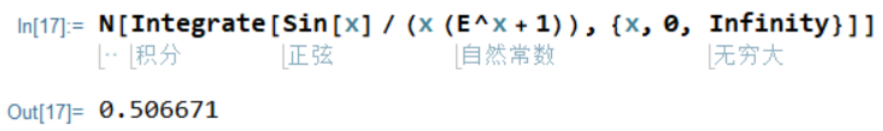
求无穷级数的和



```
In[12]:= N[Sum[Cos[π / n] / n^3, {n, 1, Infinity}]]

Out[12]= -0.948855
```

定积分求值



```
In[17]:= N[Integrate[Sin[x] / (x (E^x + 1)), {x, 0, Infinity}]]

Out[17]= 0.506671
```

四、 LaTeX 书写文本

Linear Least Squares

Linear least squares (LLS) is the least squares approximation of linear functions to data.It is a set of formulations for solving statistical problems involved in linear regression, including variants for ordinary (unweighted), weighted, and generalized (correlated) residuals. Numerical methods for linear least squares include inverting the matrix of the normal equations and orthogonal decomposition methods.

Basic Formulation

Consider the linear equation

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are given and $x \in \mathbb{R}^n$ is variable to be computed. When $m > n$, it isgenerally the case that Eq. (1) has no solution. For example, there is no value of a that satisfies

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} x = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

because the first two rows require that $x = (1, 1)$, but then the third row is not satisfied. Thus, for$m > n$, the goal of solving Eq. (1) exactly is typically replaced by finding the value of x that minimizes some error. There are many ways that the error can be defined, but one of the most common is to define it as $||\mathbf{Ax} - \mathbf{b}||^2$. This produces a minimization problem, called a least squares problem

$$\mathbf{minimize}_{x\in\mathbb{R}^n} ||\mathbf{Ax} - \mathbf{b}||^2$$

The solution to the least squares problem is computed by solving the *normal equation*

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$$

where $\mathbf{A}^T$ denotes the transpose of the matrix $\mathbf{A}$.