python 作业 level3 实验报告

舒文炫

2021 年 7 月 16 日

目录

Chapter 1

实验介绍

1.1 实验内容

扩展 python 的 num 类型，使得其支持四元数的运算，以及四元数与欧拉角的相互转化

1.2 实验环境

python 版本 3.7

Chapter 2

实验实现

```python
import numpy as np
import math

class Quaternion:
    '''
    in init part, the parameter a,b,c,d shows the quaternion a+bi+cj+dk saved in the way of quanum vector.
    self.angle is the Eular angle of the quaternion.
    '''
    def __init__(self, a, b, c, d):
        self.quanum = np.array([a, b, c, d])
        self.angle=np.array([0,0,0])
    '''
    In this part, printq print the quaternion in the style: a+bi+cj+dk. Printa print the quaternion in the style:(alpha,bet
    '''
    def printq(self):
        symbol = [" ", "i ", "j ", "k"]
        q = self.quanum.copy()
        result = ""
        for i in range(3):
            if q[i]>1e-4 or q[i]<-1e-4:
                result = result + str(round(q[i], 4)) + symbol[i]
            if q[i+1] > 1e-4:
                result=result+"+"
        if(q[3]>1e-4 or q[3]<-1e-4):
            result = result + str(round(q[3], 4)) + symbol[3]
        if result == "":
            return "0"
        else:
            return result
```

这一部分是进行四元数类的初始化，我设置了两个属性对应四元数本身的一个四维向量和对应其欧拉角的三维向量，这个欧拉角向量全部初始化为 0，只要调用了四元数转欧拉角才会对其赋值，并保存。后面是进行四元数的标准化输出，输出形式 $a+bi+cj+dk$. 这里考虑的精度问题，对比较小的值，就直接视为 0.

```python
    def printa(self):
        a=self.angle.copy()
        result="("
        for i in range(3):
                result = result + str(round(a[i], 4)) + ","
        result=result[0:-1]
        result=result+")"
        if result =="()":
            return "0"
        else:
            return result


    '''
    the addition of quaternion
    '''
    def add(self, x):
        q = self.quanum.copy()
        q=q+x.quanum
        return Quaternion(q[0], q[1], q[2], q[3])
    '''
    the subtraction of quaternion
    '''
    def sub(self, x):
        q = self.quanum.copy()
        q=q-x.quanum
        return Quaternion(q[0], q[1], q[2], q[3])
    '''
    the multiplication of quaternion
    '''
```

这一部分有对欧拉角的标准化输出，输出形式 (a, b, c), 后面是四元数的加减，直接对各元素加减即可。

```
    def mul(self, x):
        q = self.quanum.copy()
        p = x.quanum.copy()
        a = q[0]*p[0] - q[1]*p[1] - q[2]*p[2] - q[3]*p[3]
        b = q[0]*p[1] + q[1]*p[0] + q[2]*p[3] - q[3]*p[2]
        c = q[0]*p[2] - q[1]*p[3] + q[2]*p[0] + q[3]*p[1]
        d = q[0]*p[3] + q[1]*p[2] - q[2]*p[1] + q[3]*p[0]
        return Quaternion(a, b, c, d)
    '''
    the square of mod of quaternion
    '''


    def modpow(self):
        q = self.quanum.copy()
        return np.sum(q*q)
    '''
    the inverse of quaternion
    '''
    def inverse(self):
        q = self.quanum.copy()
        mod = self.modpow()
        q=q/mod
        return Quaternion(q[0], -q[1], -q[2], -q[3])
    '''
    the division of quaternion
    '''
    def divide(self, x):
        result = self.mul(x.inverse())
        return result
```

这一部分就是四元数的乘除，这一块乘法就直接按公式算就好，除法的实现稍微复杂，我拆成了三块，先求四元数的模，通过模求四元数的逆，通过逆进行四元数乘法获得除的结果。

```
    def qua2eular(self):
        q=self.quanum.copy()
        mod=math.pow(self.modpow(),0.5)
        q=q/mod
        alpha=math.atan2(2*(q[0]*q[1]+q[2]*q[3]),1-2*(q[1]**2+q[2]**2))
        beta=math.asin(2*(q[0]*q[2]-q[1]*q[3]))
        gamma=math.atan2(2*(q[0]*q[3]+q[1]*q[2]),1-2*(q[2]**2+q[3]**2))
        alpha=alpha/math.pi*180
        beta=beta/math.pi*180
        gamma=gamma/math.pi*180
        angle=np.array([alpha,beta,gamma])
        self.angle=angle
    '''
    transform eular angle to quaternion
    '''
def eular2qua(alpha,beta,gamma):
    q=np.array([0.0 for x in range(4)])
    q[0]=math.cos(alpha/2)*math.cos(beta/2)*math.cos(gamma/2)+math.sin(alpha/2)*math.sin(beta/2)*math.sin(gamma/2)
    q[1]=math.sin(alpha/2)*math.cos(beta/2)*math.cos(gamma/2)-math.cos(alpha/2)*math.sin(beta/2)*math.sin(gamma/2)
    q[2]=math.cos(alpha/2)*math.sin(beta/2)*math.cos(gamma/2)+math.sin(alpha/2)*math.cos(beta/2)*math.sin(gamma/2)
    q[3]=math.cos(alpha/2)*math.cos(beta/2)*math.sin(gamma/2)-math.sin(alpha/2)*math.sin(beta/2)*math.cos(gamma/2)
    return Quaternion(q[0], q[1], q[2], q[3])
```

这一部分是四元数与欧拉角的转化，这都是按照公式来的，四元数转欧拉角会更新四元数类的欧拉角属性，可以使用 printa 方法将其以标准的格式打印出来。欧拉角转四元数是一个单独的函数，不在四元数类，不过会返回一个四元数类。

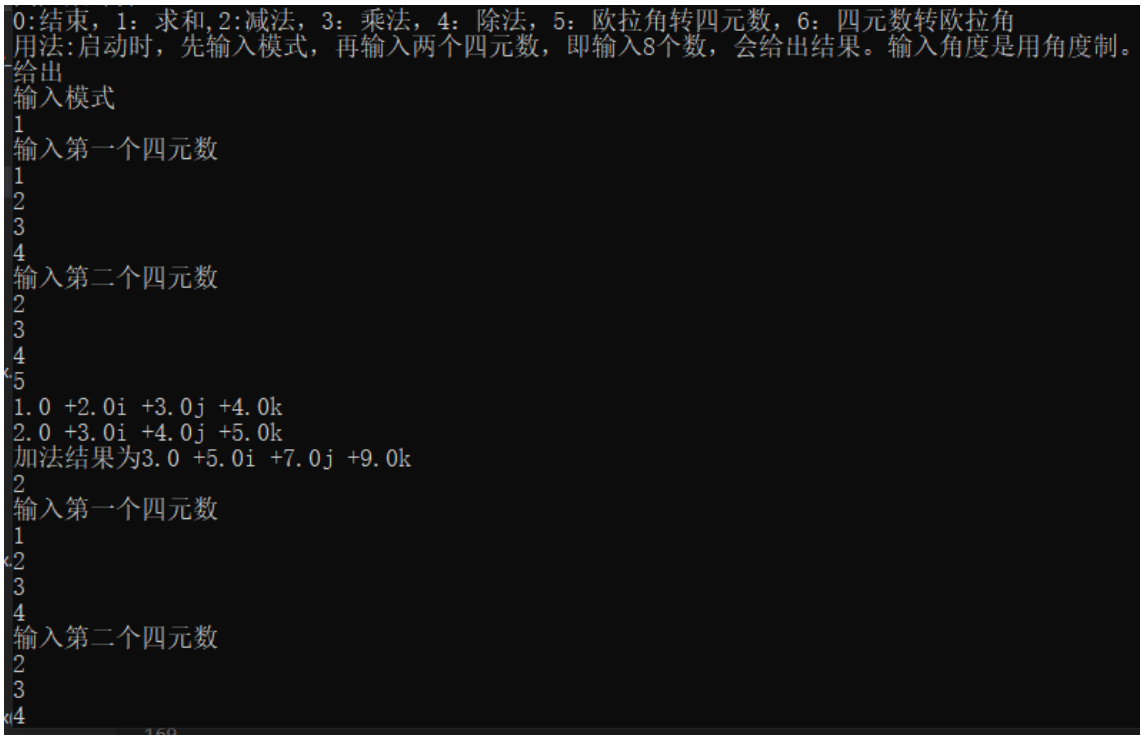
```
print("四元数计算")
print("0:结束，1：求和,2:减法，3：乘法，4：除法，5：欧拉角转四元数，6：四元数转欧拉角")
print("用法:启动时，先输入模式，再输入两个四元数，即输入8个数，会给出结果。输入角度是用角度制。四元数转欧拉角是以(z,x,z)轴顺序给出")
print("输入模式")
mode=int(input())
while(mode!=0):
    if(mode in [1,2,3,4,5]):
        a=np.array([0.0 for x in range(4)])
        b=np.array([0.0 for x in range(4)])
        print("输入第一个四元数")
        for i in range(4):
            a[i]=float(input())
        print("输入第二个四元数")
        for i in range(4):
            b[i]=float(input())

        a=Quaternion(a[0], a[1], a[2], a[3])
        b=Quaternion(b[0], b[1], b[2], b[3])
        print(a.printq())
        print(b.printq())


        if(mode==1):
            print("加法结果为"+a.add(b).printq())
        if(mode==2):
            print("减法结果为"+a.sub(b).printq())
        if(mode==3):
            print("乘法结果为"+a.mul(b).printq())
        if(mode==4):
            print("除法结果为"+a.divide(b).printq())
        if(mode==5):
            a.qua2eular()
            b.qua2eular()
```

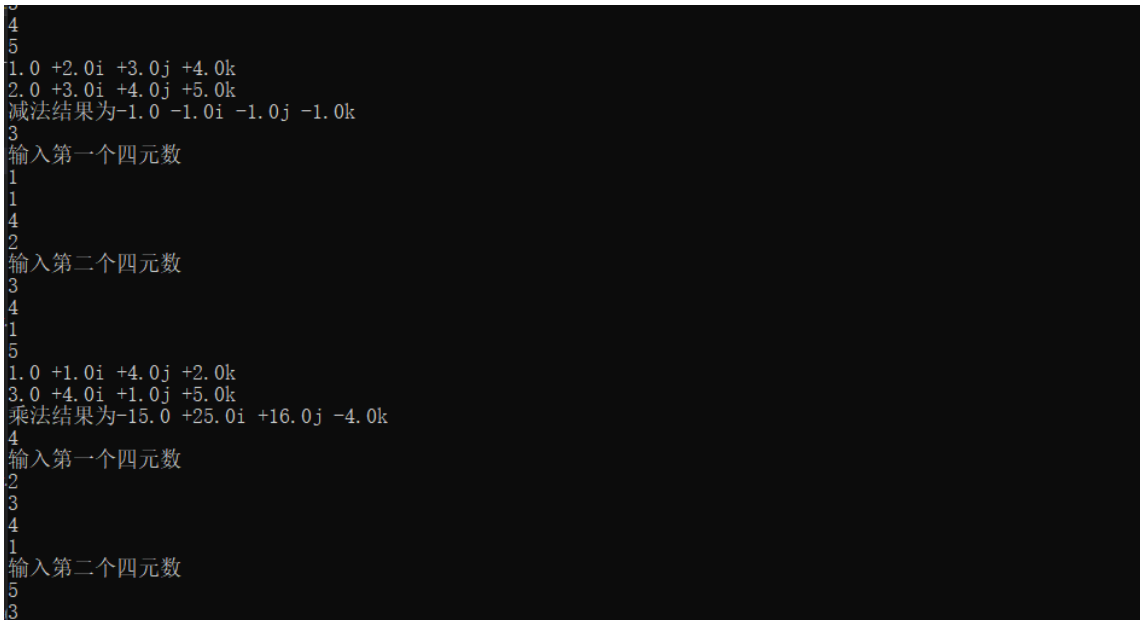
```
            a.qua2eular()
            b.qua2eular()
            print("第一个四元数对应欧拉角为"+a.printa())
            print("第一个四元数对应欧拉角为"+b.printa())
    if(mode==6):
        c=np.array([0.0 for x in range(3)])
        print("输入欧拉角")
        for i in range(3):
            c[i]=float(input())/180.0*math.pi
        print("对应四元数为："+eular2qua(c[0],c[1],c[2]).printq())

    mode=int(input())
```

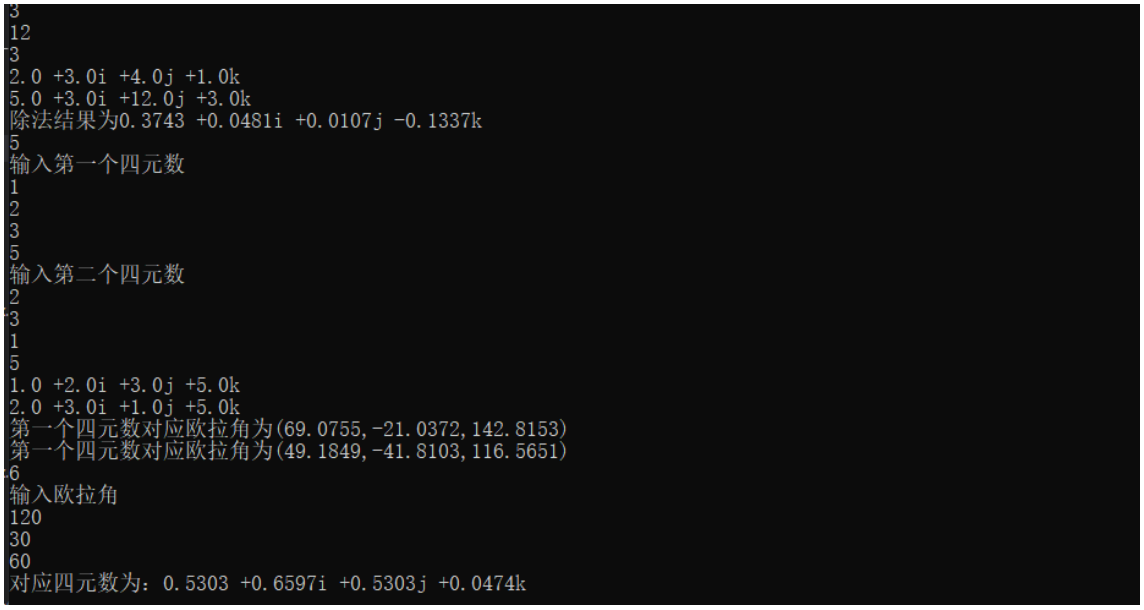
这部分是我写的四元数计算器模块，为了检验我的四元数运算是否正确，基本操作就是输入对应的功能数字，再输入四元数进行计算，会给出结果。

Chapter 3

实验结果







这三张图片是我的输出截图，可以看到这些计算都实现了，助教有兴趣也可以自行尝试更多的例子。