# python 作业 level2 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

用 munpy,math 库独立实现积分计算 $\int_0^1 \int_{-x}^{x} \sqrt{x+y^2} dxdy$，与 scipy.integrate 比较计算效率

## 1.2 实验环境

python 版本 3.7

# Chapter 2

# 实验实现

```python
#my method

def integrate1(start,end,i):
    x=np.linspace(start,end,10000)
    y=np.sqrt(i+x**2)
    dx=x[1]-x[0]
    s=np.sum(y*dx)
    return s
'''
the function above is used to calculate the integral of sqrt(i+x**2)
From numpy, we can generate an array, which is used to divide the integral domain equally into small intervals. I set the 1
y is the array of the value of sqrt(i+x**2) at the endpoint of the intervals
dx is the length of the small interval.
we add y*dx to get the reault.
This is the traditional way of Riemman integral.
'''
x=np.linspace(0,1,10000)      #the same division
res=0

for i in x:
    res=res+integrate1(-i,i,i)# for each i, we get the value of integral(sqrt(i+x^2)) from -i to i
res=res/10000                 #the Riemman sum
print("the result is {}\n".format(res))
```

这一部分是我用 numpy 实现积分运算，思想就是将积分拆成小段的黎曼和，只要这个拆的足够细，就能很好的逼近积分值。使用 linespace 方法拆分 (0,1) 区间 10000 段，我写了一个 intergrate 函数，这是进行一重积分，二重积分只需要再嵌套一层即可。

```python
#scipy method

from scipy import integrate
def f(y,x):                   #the integrand, the order of the variable is the order of integral. In this situation, we
    return math.sqrt(x+y**2)
def boundx():                 #the integrating interval of x
    return [0,1]
def boundy(x):                #the integrating interval of y
    return [-x,x]
v,err=integrate.nquad(f,[boundy,boundx]) #v is the value, err is the error.
print("the result is {}\n".format(v))
```

这一部分是用 scipy.integrate 进行计算以便比较结果。

```
d=input("the times you want:")
#my method
start=time.perf_counter()#start time
for k in range(int(d)):#we execute the coda block for d times
    x=np.linspace(0,1,10000)
    res=0
    for i in x:  (function) integrate1: (start, end, i) -> Any
        res=res+integrate1(-i,i,i)
    res=res/10000

end=time.perf_counter()#end time
print("the time comsumed through my method is {}\n".format(end-start))#the total time used

#scipy method
start1=time.perf_counter()
for k in range(int(d)):
    v,err=integrate.nquad(f,[boundy,boundx])
end1=time.perf_counter()
print("the time comsumed through scipy method is {}\n".format(end1-start1))
```

这一部分需要输入一个数，表示计算的次数，我们用程序运行时间表示这个计算的效率，会输出两个结果，一个是我的方法运行时间，一个是 yongscipy 方法的时间。

# Chapter 3

# 实验结果

```
(base) G:\绸带\python>python 2.py
the result is 0.8865989289905856

the result is 0.886468247630174

the times you want:3
the time comsumed through my method is 2.3931330000000006

the time comsumed through scipy method is 0.00622369999999961
```

这里是程序的输出，上面的 result 是我的方法算的积分结果，下面的 result 是用 scipy 算的结果，然后我输入计算次数为 3，可以看到我的方法算三次要 2.39 秒。然而 scipy 方法用了 0.006 秒。我实现的效率是比较低的，不过这里貌似可以用 numba 来加速计算。要求里面并没有提到使用 numba，其实可以考虑。