姓名：陈墨霏

学号：1120233329

班级：07112304

专业：计算机科学与技术

指导教师：孙新

关于编程实现数值积分的

实验报告

（第六章上机实验报告）

目录

[1. 实验目标 1](#_Toc183898232)

[2. 实验内容 1](#_Toc183898233)

[2.1 实现数值积分核心算法 1](#_Toc183898234)

[2.2 展示算法效果 1](#_Toc183898235)

[3. 实现过程 1](#_Toc183898236)

[3.1 实现函数表达式的操作与存储 1](#_Toc183898237)

[3.1.1 实现基础：抽象语法树（AST） 2](#_Toc183898238)

[3.1.2 表达式的输入和存储 2](#_Toc183898239)

[3.1.3 表达式求值（evaluate 方法） 3](#_Toc183898240)

[3.1.4 表达式求导（derivative 方法） 3](#_Toc183898241)

[3.2 实现数值积分核心算法 4](#_Toc183898242)

[3.2.1 牛顿——柯特斯求积公式 4](#_Toc183898243)

[3.2.2 复化求积公式 5](#_Toc183898244)

[3.2.3 龙贝格法 5](#_Toc183898245)

[3.2.4 高斯求积公式 6](#_Toc183898246)

[3.2.5 重积分的求积公式 8](#_Toc183898247)

[4. 输入、输出测试 8](#_Toc183898248)

[4.1 牛顿——柯特斯求积公式 8](#_Toc183898249)

[4.2 复化求积公式 9](#_Toc183898250)

[4.3 龙贝格法 10](#_Toc183898251)

[4.4 高斯求积公式 10](#_Toc183898252)

[4.5 重积分的求积公式 11](#_Toc183898253)

[5. 实验分析、结论、改进策略和心得体会 12](#_Toc183898254)

[5.1 实验分析和实验结论 12](#_Toc183898255)

[5.2 改进策略及方向 12](#_Toc183898256)

[5.2.1 使用SymPy库 12](#_Toc183898257)

[5.2.2 错误处理 13](#_Toc183898258)

[5.2.3 普适的数值积分公式 13](#_Toc183898259)

# 实验目标

使用Python 语言实现数值积分的牛顿——柯特斯求积公式，复化求积公式，龙贝格法，高斯求积公式以及重积分的求积公式，并展示算法效果。

# 实验内容

## 实现数值积分核心算法

结合课本所学，利用Python语言实现数值积分的牛顿——柯特斯求积公式，复化求积公式，龙贝格法，高斯求积公式以及重积分的求积公式。这些公式之间存在一定的依赖关系，但为了逻辑清晰，还是按照不同的方法大类分装到了不同文件当中。

## 展示算法效果

利用一些文字，引导用户输入被积函数和上下限，并选择所需的算法进行计算，并输出计算结果和必要的中间过程。

# 实现过程

## 实现函数表达式的操作与存储

**代码在 ./libs/MathExpr.py 中。**

为了实现函数表达式的操作与存储，我结合数据结构课程知识，并借助AI工具，实现了一个MathExpression类。

MathExpression 类支持一系列常见的数学运算符和函数，包括加法、减法、乘法、除法、幂运算、三角函数（如 sin、cos、tan）、对数函数（如 log）等。该类能够处理幂运算中负指数的情况，以实现分式运算，也能处理带分数的根式运算。此外，MathExpression 类支持常数 e 和 π（输入时应为“pi”） 的表达，提供了将数学表达式转化为字符串形式的功能，便于可视化输出。该类还具备表达式求值和数值微分求导的功能，通过符号计算，能够对数学模型进行数值分析与处理。

特别的，MathExpression类采用了\_\_call\_\_方法和一系列相关方法，使得其能够直接在代码中通过f(x, y)这样的形式来表示任意元数的函数值。

### 实现基础：抽象语法树（AST）

抽象语法树（AST）是编程语言或数学表达式的抽象层次化表示，它以树形结构展现源代码或表达式的核心逻辑。每个节点表示一个语言元素，如操作符、变量、常数等，而边则表示元素之间的关系。与源代码中的具体语法不同，AST 只关注结构和语法规则，去除了括号、分号等无关细节，因此它简化了表达式并清晰地呈现了计算顺序。AST 是编译器和解释器中的重要数据结构，广泛用于语法分析、优化和代码生成等操作。

在数学表达式中，AST 将运算符作为内部节点，操作数（如常数和变量）作为叶子节点。例如，表达式 3 \* (x + 2) 的 AST 根节点是 \*，左子节点是常数 3，右子节点是加法操作符 +，它的子节点分别是变量 x 和常数 2。通过构建树形结构，AST 清晰展示了运算顺序，便于后续的求值、优化和转换。

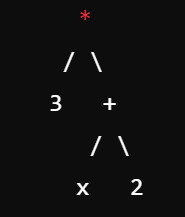
### 表达式的输入和存储

首先是词法分析（Tokenization）。\_tokenize 方法对输入的字符串进行词法分析，将表达式字符串分割成有意义的“标记”（tokens）。这些标记包括数字、运算符、常数（如 e 和 π）、函数名（如 sin、cos）以及括号等。这个过程通过正则表达式将原始字符串拆解为一个个标记。

接着是语法分析（Parsing）。\_build\_ast 方法通过递归的方式解析这些标记，并构建 AST。它从简单的“因子”开始解析，然后逐步构建更复杂的结构。例如，首先会解析数字和变量，再递归解析运算符、函数以及括号中的表达式。AST 的每个节点对应一个表达式的元素，如操作符、函数或操作数。这个过程涉及到优先级和括号的处理，确保解析顺序正确。

解析完成后是树的构建。构建的 AST 被存储在 MathExpression 对象的 ast 属性中。AST 中的每个节点都是 Node 类的实例，包含了节点类型（如 operator、number、function 等）、节点的值（如 +、3、sin 等）以及子节点。通过这种方式，整个表达式的层次结构和运算顺序得以准确表示。

最终，AST 树形结构可以用于进一步的运算、优化或转换。例如，表达式 3 \* (x + 2) 的 AST 会是：



1. AST示例

### 表达式求值（evaluate 方法）

MathExpression 类通过 evaluate 方法实现表达式求值。该方法首先定义了一个递归函数 \_eval，用于遍历抽象语法树（AST）。根据节点的类型，\_eval 逐层计算节点的值。如果节点类型是数字，则直接返回其数值；如果是常数（如 e 或 π），则返回对应的数学常数；若是变量，则通过 variables 字典获取其值。对于函数节点（如 sin、cos 等），递归地求值其子节点并调用相应的数学函数。对于运算符节点（如 +、-、\*、/ 和 ^），则分别执行相应的数学运算（加法、减法、乘法、除法和幂运算）。最终，evaluate 方法返回整个表达式的计算结果。

### 表达式求导（derivative 方法）

虽然课程中没有讲到数值微分的知识，但出于求积公式方法误差（实际上可以计算出具体数值的是方法误差限，在不引起混淆的前提下以后简称方法误差）求解的需要，我借助AI工具实现了这一部分的算法。经检验，这部分算法在低阶导数（导数阶数 < 4时）求解时比较准确，高阶导数具有较大的误差。相应的，一些求积公式的方法误差计算结果，也出现了较大的误差（特别是高斯求积公式的方法误差）。

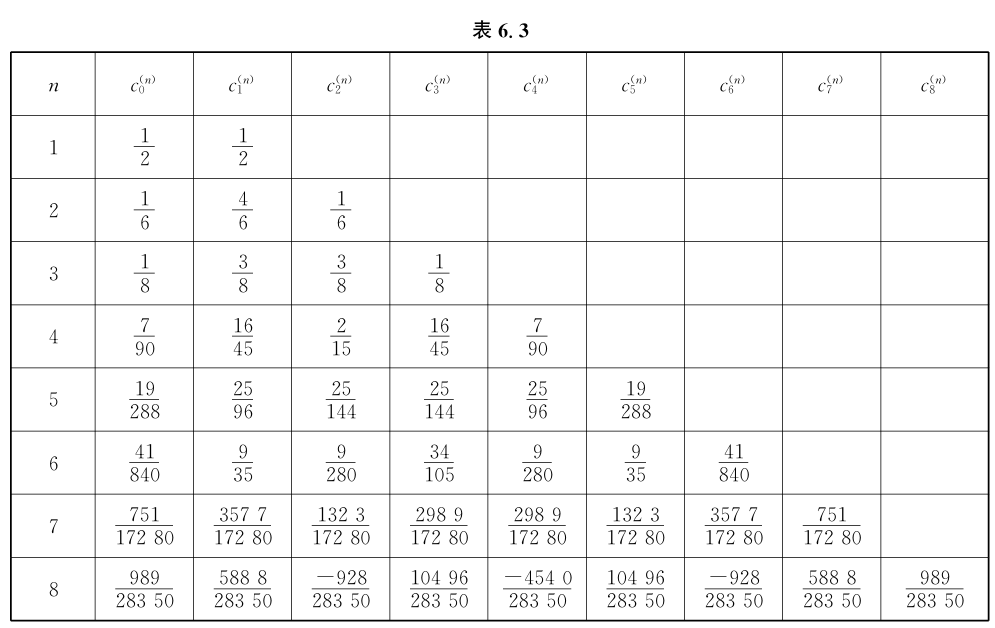
MathExpression 类通过数值微分的中心差分法实现求导功能。derivative 方法接受求导变量 var、阶数 order 和微小增量 h 作为参数，使用中心差分公式计算导数。对于一阶导数，计算表达式在 x + h 和 x - h 处的差值，并除以 2h。对于高阶导数，递归调用数值微分函数，逐步计算更高阶的导数。最终，derivative 返回一个步长为h的表格函数，该函数可以用于计算指定点处的导数值。

## 实现数值积分核心算法

### 牛顿——柯特斯求积公式

**代码在 ./libs/Integral/** **NewtonCotesFormula.py 中。**

这部分我实现了梯形公式、辛卜生公式、3/8公式、柯特斯公式和“通用”牛顿——柯特斯公式（也即根据课本式（6. 26）以及表 6. 3，依据传入的参数nn，来构造的公式）。以下是实现方法。



1. 课本表 6.3 内容

前四个公式，以梯形公式trapezoid\_formula(f, a, b)为例进行说明。

传入的参数f为MathExpression对象，作为函数表达式，a为积分下限，b为积分上限。

求解积分结果I的过程，是依据公式对应的分段数计算出h，之后带入公式进行计算。

求解方法误差限R的过程，首先是求解二阶导数在积分区间上的最大值。通过将积分区间利用NumPy库的linspace方法进行分割取点，得到一系列结点，然后求得f的二阶导数，并代入上述一系列的点，利用max函数和abs函数求得绝对值最大的函数值。之后，将二阶导数在积分区间上的最大值，带入到方法误差的计算公式中，计算求得方法误差。

“通用”牛顿——柯特斯公式的实现流程与上述类似，但相比之下多了一个从Excel表格（./coefficients\_nc.xlsx）中读取数据的，并依据输入的nn值选择所需数据的过程。另外，由于课本没有给出该法的方法误差求解公式，故函数返回值的R部分为一个空字符串。

### 复化求积公式

**代码在 ./libs/Integral/** **CompositeFormula.py 中。**

这部分我实现了复化梯形公式、复化辛卜生公式、复化3/8公式和复化柯特斯公式。四个公式实现方法类似，以复化辛卜生公式composite\_simpson\_formula(f, a, b, M)的实现为例进行说明。

传入的参数中，f，a，b的含义与梯形公式相同。M是积分区间[a, b]的总分段数。

首先要对输入的M进行一个判断，M需要为2的倍数，才能够使用复化辛卜生公式，否则抛出一个ValueError。

求解积分结果I的过程，是依据M来计算出h，之后代入公式进行计算。这里并没有利用到一般的辛卜生公式simpson\_formula(f, a, b)，目的是代码简洁且减少运算量。

求解方法误差限R的过程与梯形公式的过程类似，这里不再赘述。

### 龙贝格法

**代码在 ./libs/Integral/** **RombergMethod.py 中。**

这部分我通过实现逐次加密分点的复化梯形递推公式、梯形序列到辛卜生序列、辛卜生序列到柯特斯序列以及柯特斯序列到龙贝格序列的推导公式，并结合一个收敛判断函数，实现了一套龙贝格法的基本流程。由于课本未给出龙贝格法具体的方法误差求解公式，这部分未实现方法误差的求解。

最终的龙贝格法求解函数romberg\_method(f, a, b, tolerance=1e-6)中，f，a，b的含义与梯形公式相同。tolerance是结果精度，默认值为1e-6。

龙贝格法所有中间结果的来源是梯形序列，而梯形序列则起源于最基本的梯形公式trapezoid\_formula(f, a, b)，其结果为T1，或T20。

通过实现逐次加密分点的复化梯形递推公式calculate\_next\_trapezoid(f, a, b, T\_previous, k)，输入T2k-1，计算求得T2k。

通过实现梯形序列到辛卜生序列calculate\_next\_simpson(T\_current, T\_previous, k)、辛卜生序列到柯特斯序列calculate\_next\_cotes(S\_current, S\_previous, k)以及柯特斯序列到龙贝格序列calculate\_next\_romberg(C\_current, C\_previous, k)的推导公式，由T2k+3、T2k+2，得到S2k+2、S2k+1，进一步得到C2k+1、C2k，进一步得到R2k。

在龙贝格法求解函数romberg\_method(f, a, b, tolerance=1e-6)中，用T、S、C、R四个列表存储四个序列。先利用基本的梯形公式计算出T1的值，在此之后，建立一个循环，用k（初值为0）计数，循环内按照T、S、C、R的顺序求解。其中，执行S求解的条件是k >= 1，执行C求解的条件是k >= 2，执行R求解的条件是k >= 3。一次循环结束，k += 1。

为实现收敛的判断，在计算前定义previous\_value和current\_value两个变量。计算出T1后，作为previous\_value的初值，计算出T2后，作为current\_value的初值。之后，每计算出一个新值之前，先令previous\_value = current\_value，再更新current\_value的值，并利用check\_convergence(current\_value, previous\_value, tolerance)函数，计算二者之差的绝对值，并与tolerance比较，符合收敛条件时，则循环结束，输出结果。

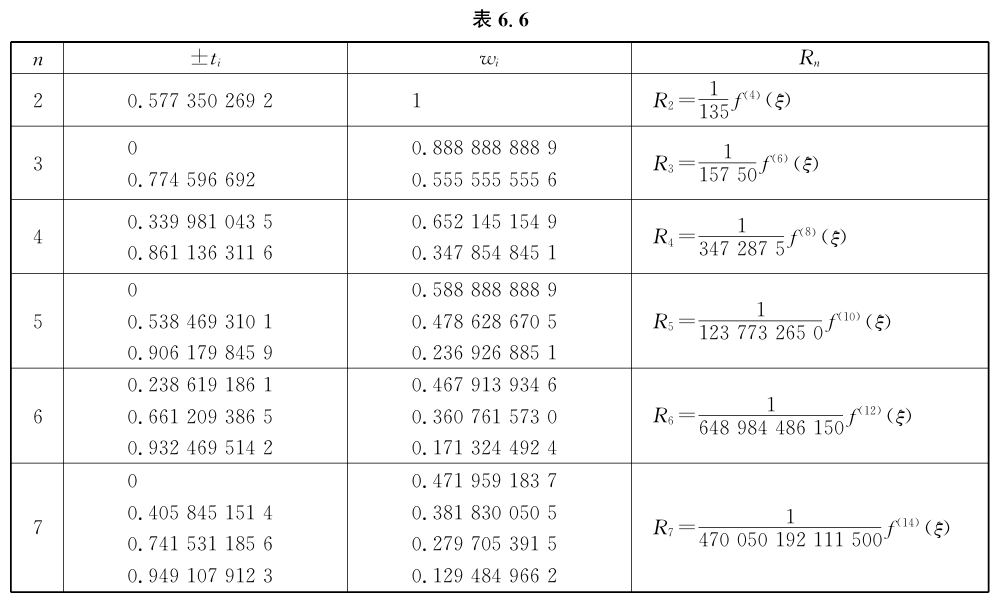
### 高斯求积公式

**代码在 ./libs/Integral/** **GaussianFormula.py 中。**

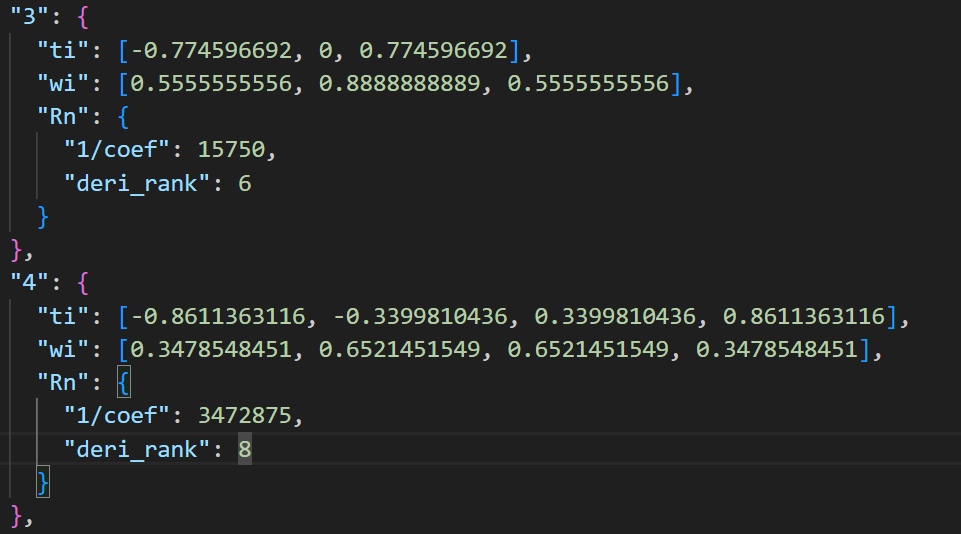
这部分我实现了高斯求积公式gauss\_formula(f, a, b, n)。

传入的参数中，f，a，b的含义与梯形公式相同。n为高斯求积公式的阶数，通过n可以查表得到所需的勒让德多项式的零点，系数和方法误差公式。

相比于表 6. 3，表6. 6所表示的数值的关系不是简单的二维矩阵的关系，而是可以表示成一个树。因此，我将表6. 6的内容存储为了一个json文件（coefficients\_gauss.json）。



1. 课本表6. 6 内容



1. json文件存储结构

存储结构如图4所示：第一层键为n的值字符串化，第二层键为ti、wi、Rn三个字符串，其中ti、wi的值为两个列表，列表元素上下一一对应。Rn下还有一层，分别为1/coef和deri\_rank两个字符串，其值分别为Rn公式系数的分母和Rn公式f(x)的求导阶数。

gauss\_formula(f, a, b, n)函数当中，首先打开coefficients\_gauss.json，并根据n值提取所需的数据。之后，根据公式计算xi的值，进一步根据公式计算I值。

求解方法误差限R的过程与梯形公式的过程类似，不同点就是所求导数阶数为deri\_rank的值。

### 重积分的求积公式

**代码在 ./libs/Integral/** **MutipleIntegral.py 中。**

这部分我实现了二重积分的辛卜生公式double\_composite\_simpson\_formula(f, bounds, m, n)和二重积分的高斯求积公式double\_gauss\_formula(f, bounds, m, n)。

double\_composite\_simpson\_formula(f, bounds, m, n)函数的参数中，f为MathExpression对象，表示一个二元函数表达式，bounds为一个有序偶的列表（实际上就是二元列表的列表），m是x方向的分段数，n是y方向的分段数。

double\_gauss\_formula(f, bounds, m, n)函数的f和bounds与double\_composite\_simpson\_formula(f, bounds, m, n)相同，

求解过程均按照提取上下限——获取网格点值——代入公式计算的步骤进行。只不过，辛卜生公式的网格点是利用numpy.linspace()方法创建的均匀点，而高斯求积公式的网格点是由ti计算得到。

课本没有提供方法误差的计算公式，故这里没有计算方法误差。

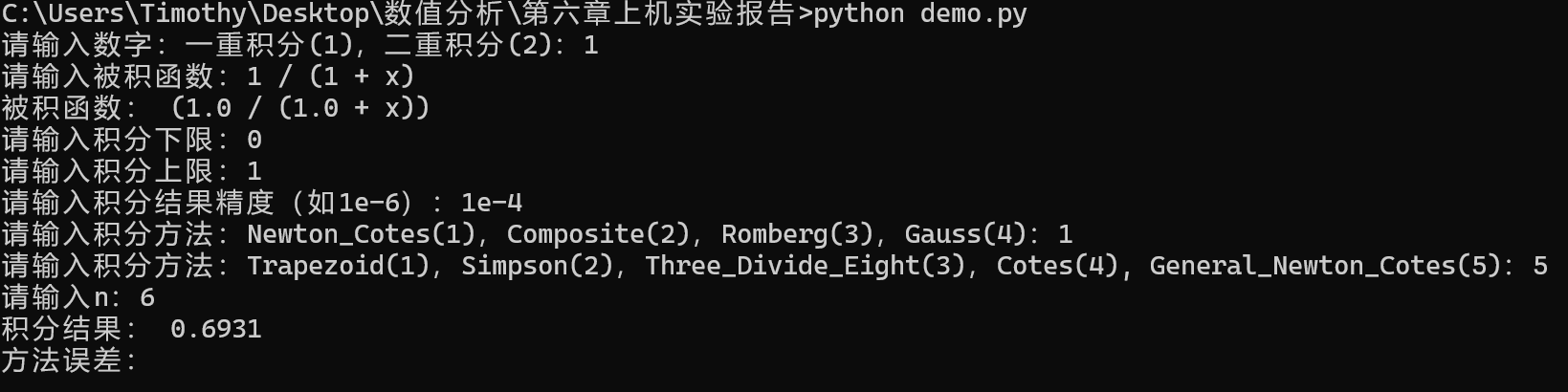
# 输入、输出测试

## 牛顿——柯特斯求积公式

示例采用课本P178 例6. 1。

题目：利用n = 6牛顿——柯特斯公式计算下列定积分值

输入输出结果如图。



1. 输入输出结果

可以看到，积分结果与正确结果完全一致。方法误差课本没有提供求解公式，而本题答案也没有提供方法误差部分，故未输出方法误差。

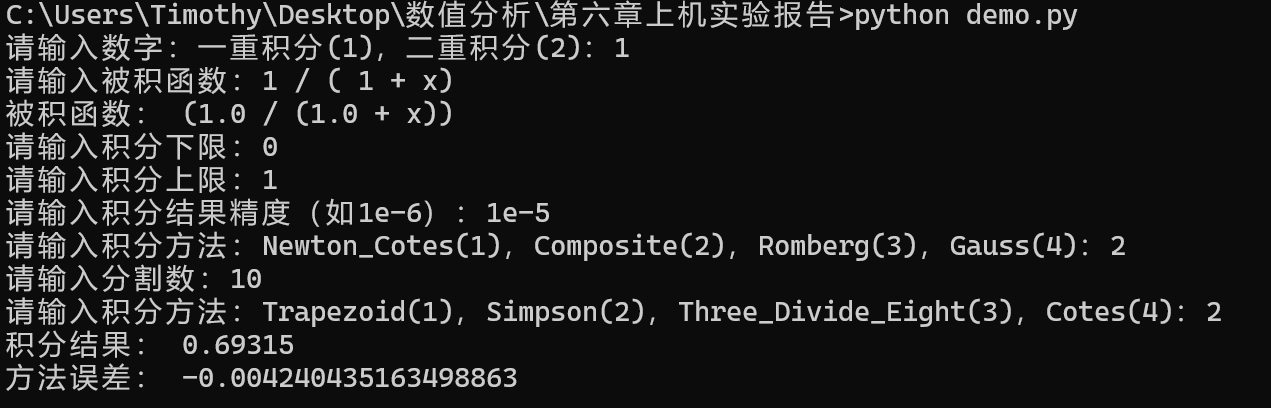
## 复化求积公式

示例采用课本P180 例6. 3。

题目：利用复化辛卜生公式计算积分

取2m = M = 10.

输入输出结果如图。



1. 输入输出结果

可以看到，积分结果与正确结果完全一致，但方法误差求解略有不同，不过也保持在了与课本相同的较小程度上。

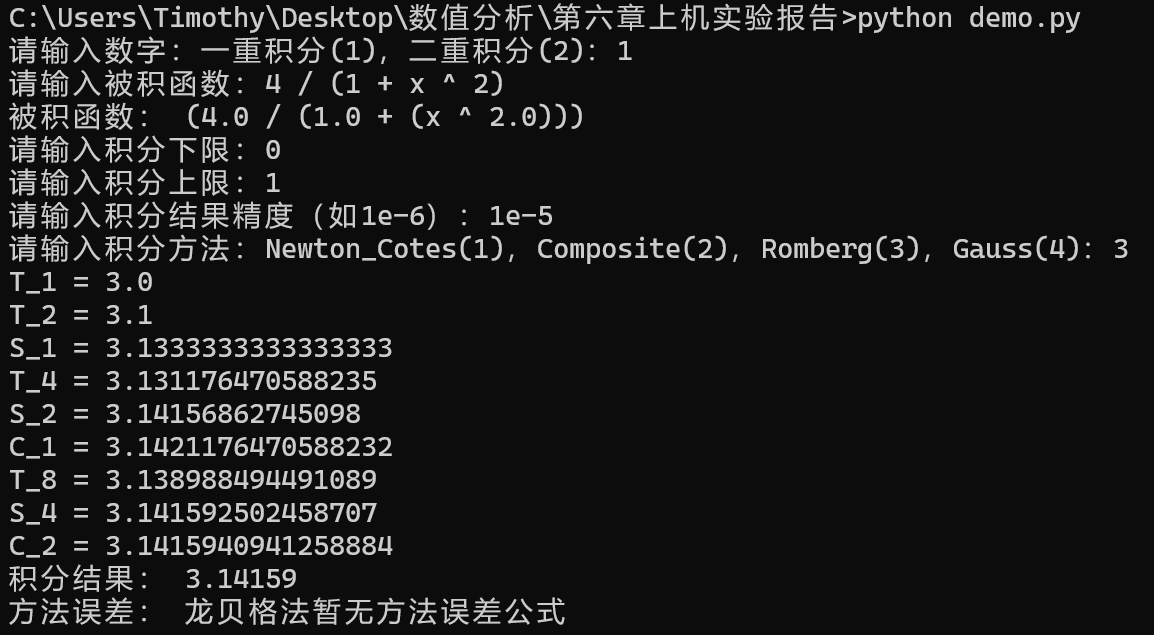
## 龙贝格法

示例采用课本P184 例6. 4。

题目：求

的近似值，其近似值要求稳定至小数后5位。

输入输出结果如图。



1. 输入输出结果

可以看到，积分结果与正确结果完全一致，求解步骤中间结果舍入后也与课本结果完全一致。

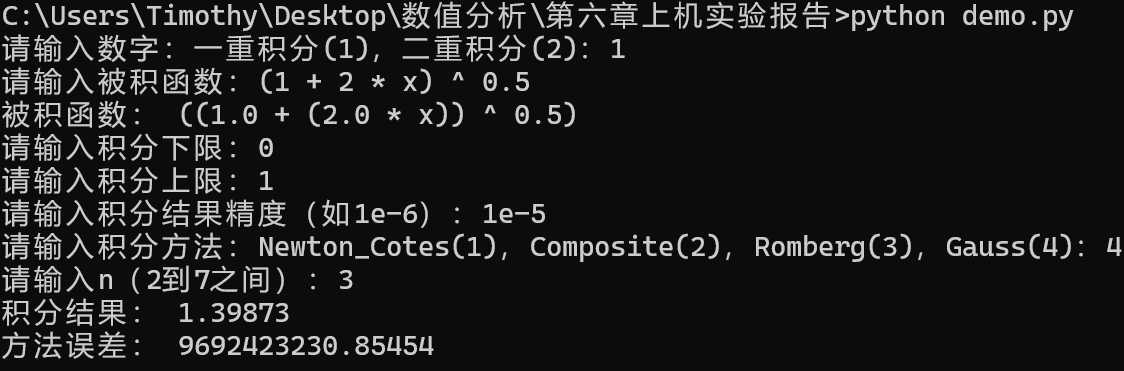
## 高斯求积公式

示例采用课本P196 例6. 10。

题目：利用高斯求积公式（n=3）计算下列积分

的近似值。

输入输出结果如图。



1. 输入输出结果

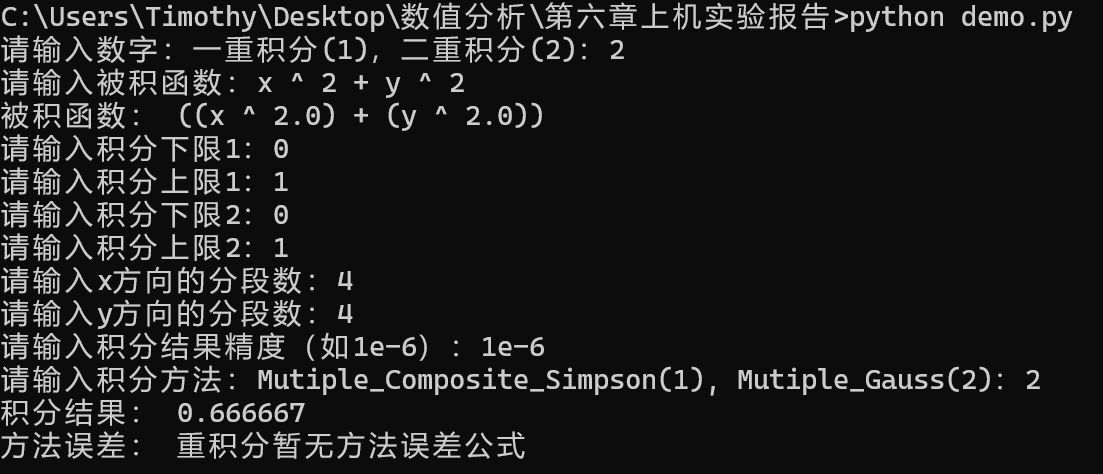
可以看到，积分结果与正确结果基本一致，但方法误差部分出现了较大的偏差，这是求导过程的误差所致。

## 重积分的求积公式

示例：求

的近似值。 x方向分段数m = 4， y方向分段数n = 4，精度为1e-6，采用二重积分的高斯求积公式进行演示。

输入输出结果如图。



1. 输入输出结果

经计算，积分I的值为2/3，与数值计算结果相同。

# 实验分析、结论、改进策略和心得体会

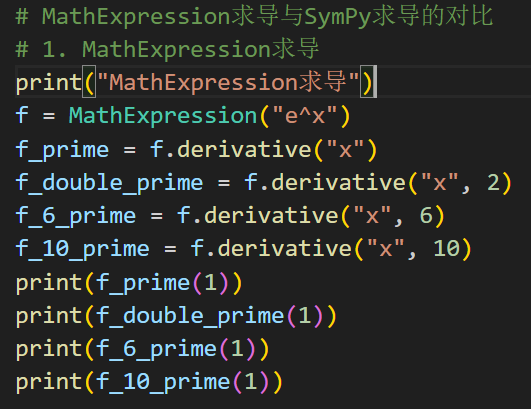
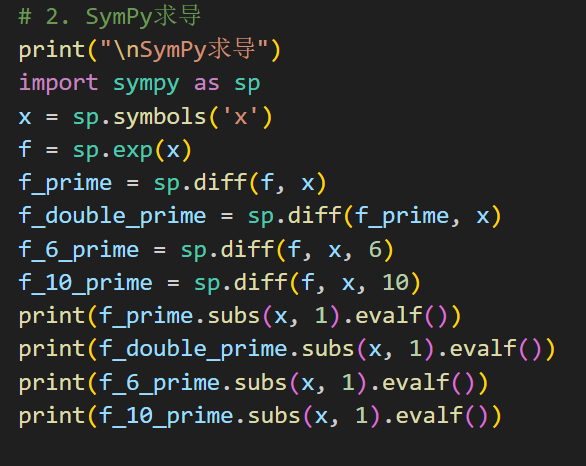
## 实验分析和实验结论

在本实验报告中，我们采用Python语言精心实现了数值积分中的一系列核心算法，包括经典的牛顿-柯特斯求积公式、复化求积公式、精确度更高的龙贝格法、高效的高斯求积公式，以及处理多重积分问题的重积分求积公式。这些算法的实现，涉及到了从基础的梯形规则到复杂的高斯积分技术，不仅展示了数值积分的多样性，也体现了算法在处理不同积分问题时的适用性和效率。

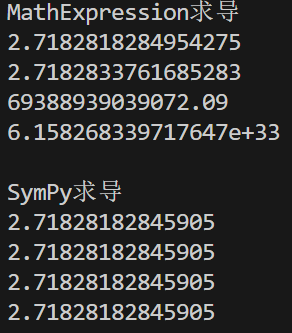
## 改进策略及方向

### 使用SymPy库

直到我完成数值积分的全部公式，着手考虑如何实现方法误差的求解，从而产生对表达式求导的需求时，我才发现了SymPy这样的一个完美贴合我的需求，且能够完美替代掉MathExpression类的库。出于对此前代码的珍惜，我在本实验当中继续使用了已经写好的MathExpression类，并尝试为其选择一个合适的求导方法。不过，自行编写的符号化表达+求导规则非常冗长，且普适性很差，经常性的报错；而数值微分方法在低阶导数时效果良好，但导数阶数稍高，则有巨大的且不可控的误差产生（病态问题）。而SymPy库求导结果非常精确且稳定。

1. 求导代码



1. 求导结果对比

在使用SymPy之后，理论上能够解决高斯求积公式方法误差无法正确求解的问题。同时，能够支持更加丰富的函数表达式。

### 错误处理

本程序未设置诸如表达式输入错误、被积函数不可导、函数不可积一类的错误处理，出现诸如此类错误，则会直接报告Python内置的错误。这样会造成用户体验的不友好，且存在一定的安全风险。应当设置合适的错误捕捉与处理机制，从而实现对用户的引导以及信息的保护。

### 普适的数值积分公式

出于时间原因和自身能力限制，我原本计划实现的任意重积分的复化牛顿——柯特斯公式和任意重积分的高斯求积公式，未能实现。

相较于固定重数的积分，任意重积分需要实现的核心点是n重循环的实现和正确利用。借助搜索引擎和AI工具，我了解到可以利用itertools.product实现n重循环。itertools.product能够通过实现n次的笛卡尔积来创建n^n个n项的列表，从而达到n重循环的效果。除此之外，还有一些其他方法，此处不再赘述。

**附录：**

**程序运行方法**

在命令行中，切换到程序根目录，输入python demo.py，即可运行。

命令行运行方式，需要的库如下（./requirements.txt）：

math

numpy

openpyxl（打开Excel用）

pandas

re