实验 3. 类与继承

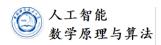
一、实验目标

本次实验旨在深化同学们对面向对象编程中"类"与"继承"核心概念的理解与掌握。通过设计和实现一个数值积分的框架,我们希望同学们能够亲手实践类的定义、方法的重写以及继承关系的应用,从而更直观地体会这些抽象概念在实际编程中的作用。

实验通过构建一个通用的 `Integrator` 基类及其三个子类 (`Trapezoidal`、 Simpson` 和 `GaussLegendre`), 让同学们探索如何利用继承机制复用代码、如何通过方法重载实现不同积分算法的差异化逻辑, 并在此过程中加深对面向对象设计原则 (如封装、抽象和多态)的认识。

二、实验任务

2. 定积分的数值计算

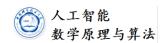


函数 f(x) 在区间 [a,b] 上的定积分可用区间内选取的 n+1 个点 x_i (i=0,1,...,n) (称为积分节点) 上的函数值的加权和近似计算:

$$\int_{a}^{b} f(x) dx \approx \sum_{i=0}^{n} w_{i} f(x_{i})$$

其中 w_i 是函数值 $f(x_i)$ 的权值,称为积分系数。不同的数值计算公式的区别体现在积分节点和积分系数上。

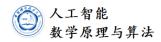
2. 定积分的数值计算



公式名称	积分节点的坐标和积分系数
复合梯形公式	$x_i = a + ih$ for $i = 0,, n$, $h = \frac{b-a}{n}$,
	$w_0 = w_n = \frac{h}{2}, w_i = h \text{ for } i = 1,, n-1$
复合辛普森公式	$x_i = a + ih$ for $i = 0,, n$, $h = \frac{b-a}{n}$,
n必须是偶数	$ w_0 = w_n = \frac{h}{3}, w_i = \frac{2h}{3} \text{for} i = 2, 4,, n-2, $
若输入的 n 是奇数,则执行 $n = n + 1$	$w_i = \frac{4h}{3}$ for $i = 1, 3,, n - 1$
复合高斯-勒让德公式	$x_i = a + \frac{i+1}{2}h - \frac{\sqrt{3}}{6}h$ for $i = 0, 2,, n-1$,
n必须是奇数	$x_i = a + \frac{i}{2}h + \frac{\sqrt{3}}{6}h$ for $i = 1, 3,, n$,
若输入的 n 是偶数,则执行 n = n+1	$h = \frac{2(b-\tilde{\delta})}{n+1}, w_i = \frac{h}{2}, \text{for} i = 0, 1,, n$

表: 定积分的几种数值计算公式

2. 定积分的数值计算



在程序 5.9 中实现 Integrator 类的 integrate 方法和它的三个子类,分别对应表中的三种公式。在每个子类中只需覆盖父类的 compute_points 方法计算并返回两个列表,它们分别存储了所有积分节点的坐标和积分系数。test() 函数用函数 $f(x) = (x \cos x + \sin x)e^{x \sin x}$ 和它的解析形式的积分函数 $F(x) = e^{x \sin x}$ 测试这三个公式的精确度。

本次实验的任务是完成程序 5.9。为了方便大家理解题意,助教完成了Trapezoidal的部分作为参考并添加了一些注释。

import math

class Integrator:

def __init__(self, a, b, n):

self.a, self.b, self.n = a, b, n

```
self.points, self.weights = self.compute_points()
   def compute_points(self):
      raise NotImplementedError(self.__class__.__name__)
   def integrate(self, f): # 使用积分点 (self.point) 和权重 (self.weights)
计算积分,返回积分值
      # 待完成
class Trapezoidal(Integrator):
   def compute_points(self): # 示例代码
      h = float(self.b - self.a) / self.n
      p = [self.a + i*h for i in range(self.n+1)]
      w = [h] + [2*h]*(self.n - 1) + [h]
      return p, w # 返回两个列表,分别表示积分点(self.point)和权重(self.
weights)
class Simpson(Integrator):
   def compute_points(self):
      # 待完成
class GaussLegendre(Integrator):
```

```
def compute_points(self):
      # 待完成
def test():
   def f(x): return (x * math.cos(x) + math.sin(x)) * \
                  math.exp(x * math.sin(x))
   # "\"表示一行太长时可以分成多行,实际上是一行,不要在行末加空格
   def F(x): return math.exp(x * math.sin(x))
   a = 2; b = 3; n = 200
   I_exact = F(b) - F(a)
   tol = 1E-3
   methods = [Trapezoidal, Simpson, GaussLegendre] # 三个类
   for method in methods:
      integrator = method(a, b, n) # 注意此处 integrator 是一个类的实例,与 In
tegrator 类不同,与 Integrate 类中的 integrate 函数仅同名,也是不同的东西。
      I = integrator.integrate(f) # 调用从 Integrate 类继承的 integrate 函数
      rel_err = abs((I_exact - I) / I_exact)
      print('%s: %g' % (method.__name__, rel_err))
      if rel_err > tol:
```

```
print('Error in %s' % method.__name__)

if __name__ == '__main__':
    test()
```