

目录	1
----	---

## 目录

<b>1 辛普森法则的证明</b>	<b>2</b>
<b>2 找零点的Python和C++实现</b>	<b>3</b>
2.1 Python实现方法 . . . . .	3
2.2 C++二分法实现 . . . . .	4

李子龙

201818000807036

2019 年 3 月 24 日

## 1 辛普森法则的证明

在积分时不再使用梯形近似，而是使用二次函数来近似函数的值。由于二次函数需要三个点来确定，所以很自然地，用近邻的三点来确定在此区间内的二次函数的具体形式。直接利用拉格朗日插值法求解 $2i, 2i+1, 2i+2$ 三个点所确定的二次函数。

$$\begin{aligned} f(x) = & \frac{(x-x_{2i+1})(x-x_{2i+2})}{(x_{2i}-x_{2i+1})(x_{2i}-x_{2i+2})}f_{2i} + \frac{(x-x_{2i})(x-x_{2i+2})}{(x_{2i+1}-x_{2i})(x_{2i+1}-x_{2i+2})}f_{2i+1} \\ & + \frac{(x-x_{2i})(x-x_{2i+1})}{(x_{2i+2}-x_{2i})(x_{2i+2}-x_{2i+1})}f_{2i+2} + O(h^3) \end{aligned} \quad (1)$$

在区间 $[x_{2i}, x_{2i+2}]$ 中，对上式进行积分，假定 $x_{i+1}-x_i = h$ , 对 $\forall i \in 0, 1, \dots, N-1$ 。  
令 $f_1(x) = \frac{(x-x_{2i+1})(x-x_{2i+2})}{(x_{2i}-x_{2i+1})(x_{2i}-x_{2i+2})}f_{2i}$ ,  $f_2(x) = \frac{(x-x_{2i})(x-x_{2i+2})}{(x_{2i+1}-x_{2i})(x_{2i+1}-x_{2i+2})}f_{2i+1}$ ,  $f_3(x) = \frac{(x-x_{2i})(x-x_{2i+1})}{(x_{2i+2}-x_{2i})(x_{2i+2}-x_{2i+1})}f_{2i+2}$ 。

$$\begin{aligned} \int_{x_{2i}}^{x_{2i+2}} f(x)dx &= \int_{-h}^h f_1(x+x_{2i+1})dx + \int_0^{2h} f_2(x+x_{2i}) + f_3(x+x_{2i})dx \\ &= \frac{1}{3}hf_{2i} + \frac{4}{3}hf_{2i+1} + \frac{1}{3}hf_{2i+2} + O(h^4) \end{aligned} \quad (2)$$

从而：

$$\int_{x_0}^{x_N} f(x)dx = \frac{h}{3} \sum_{j=0}^{N/2-1} (f_{2j} + 4f_{2j+1} + f_{2j+2}) + O(h^4) \quad (3)$$

辛普森法则证毕。□

## 2 找零点的Python和C++实现

### 2.1 Python实现方法

用Python实现的目的是，希望能够不依赖于所给函数的形式，对所有初等函数都能自动给出其导数，从而实现牛顿法。这一想法主要是通过sympy库实现的，它具有非常强大的符号计算功能。所以，在程序的开头，导入了sympy库。

```
1 import sympy
2 from sympy import log, exp, cos, sin
```

为了使得程序更加简洁，所以定义了一个找零点的对象。

```
1 class Find_Zero_Point(object):
```

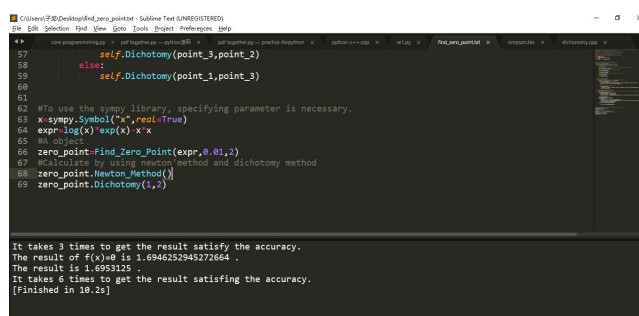
这个对象包含所需精度、迭代起始点以及函数表达形式为参数。

```
1 def __init__(self, expr, accuracy, begin_point):
```

这个对象包含了两个方法，分别是牛顿法和二分法求零点,但是二分法需要额外的参数，需要给定初始的两个点。

```
1 def Newton_Method(self):
2 def Dichotomy(self, point_1, point_2):
```

由于Python是解释型语言，所以没有编译过程，在这里直接给出运行的结果（第一个结果是牛顿法得到的，第二个是二分法得到的）。



```
57 self.Dichotomy(point_3, point_2)
58 else:
59 self.Dichotomy(point_3, point_3)
60
61
62 #to use the sympy library, specifying parameter is necessary.
63 x=sympy.Symbol("x", real=True)
64 expr=log(x)*exp(x)*x*x
65 #a object
66 zero_point=Find_Zero_Point(expr,0.01,2)
67 #Calculate by using Newton method and dichotomy method
68 zero_point.Newton_Method()
69 zero_point.Dichotomy(1,2)

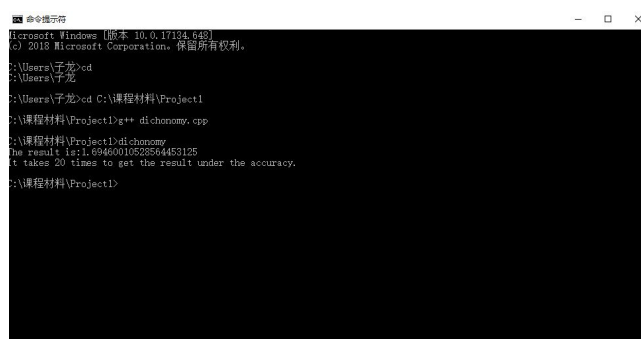
It takes 3 times to get the result satisfy the accuracy.
The result of f(x)=0 is 1.6946252945272664 .
The result is 1.6953125
It takes 6 times to get the result satisfying the accuracy.
[Finished in 10.2s]
```

图 1: 输出结果

## 2.2 C++二分法实现

利用的算法和Python定义的对象中二分法相同，这里不再赘述。需要提及的是，二分法的主函数当中我使用了函数指针，所以理论上可以自己定义函数，让二分法的函数进行零点的求解。

```
1 void dichotomy(double accuracy, double point_1,
2 double point_2, double (*func)(double)){
```



```
Microsoft Windows [版本 10.0.17134.648]
(c) 2018 Microsoft Corporation. 保留所有权利。

C:\Users\子龙>cd
C:\Users\子龙>cd C:\课程材料\Project1
C:\课程材料\Project1>g++ dichotomy.cpp
C:\课程材料\Project1>dichotomy
The result is:1.69460010523594453125
It takes 20 times to get the result under the accuracy.
C:\课程材料\Project1>
```

图 2: 输出结果