ch06 sec6.4 微积分第二基本定理

Table of Contents

使用定积分构造反导函数

下面看看如何用定积分来构造原函数

假设有函数 $f(x) = e^{-{}^2}$. 它揭示了函数$F(x)$的变化情况, 我们的目的就是通过积分的方式找到这个函数.

根据积分基本定理有:

$$F(b) - F(a) = \int_a^b e^{-t^2} dt$$

设定 $a = 0$,b 替换为x,得到:

$$F(x) - F(0) = \int_0^x e^{-t^2} dt$$

如果反导函数满足 $F(0) = 0$ 则

$$F(x) = \int_0^x e^{-t^2} dt$$

由此就得到了一个反导函数的另一种表达式. 对于定义域的的x,可以通过黎曼和的方法求出近似值.

通过这种构造方式得到反导函数$F(x)$ 求导数必然会得到 $f(x)$.

在一般的微积分思维逻辑中,要先给出一个满足连续性的函数,然后求导, 研究变化情况.在实际中绝大多数的情况下我们只知道到变化是什么, 原本的函数是什么并不清楚, 因为复杂的变化用我们之前学过的一些基础函数单独表示不了.

如果我们通过可以获得描述变化的函数, 然后对其进行积分就可以构造出一个新的函数, 这个函数的表达式是不是和原来的函数一样无关紧要, 重要的是他们的特性是一样的. 什么样的特性一样?

1. 定义域一样
2. 值域一样
3. 变化的性质一样(导函数一样)

这种思路对于理解复杂系统的建模至关重要. 我再举两个例子.

⑴. 黑箱模型

看看饮料,比如可口可乐， 如果你在很多地方买过可乐， 你会发现标注的生产地其实有好几个． 如果喝起来味道一样， 我们就感觉不到灌装厂的区别．　对于可乐我们关注的的是好喝不好喝,而不是具体在那里生产的．

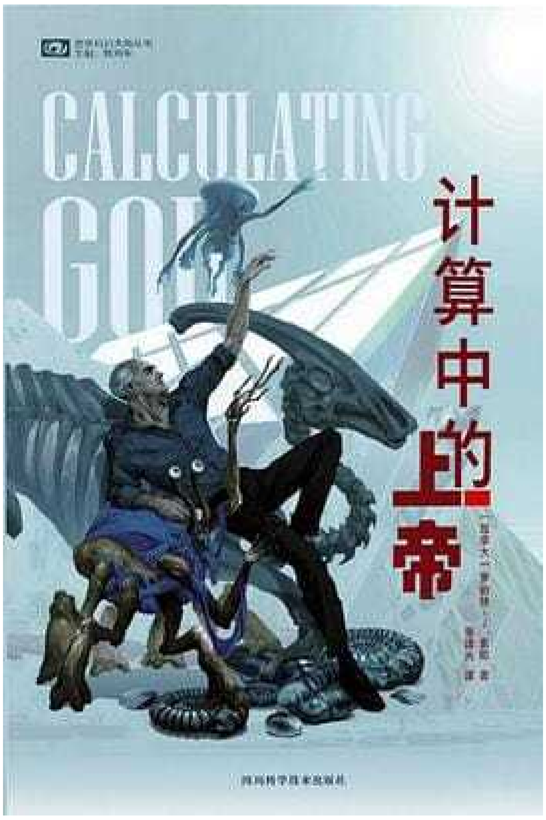
(2). "鸭子类型"

> 这是编程技术里的一种方法. 具体说就是如果叫起来和鸭子一样"嘎嘎!"的都是鸭子, 不管他是不是鸭子.
> 常规思维, 我们要想听到"嘎嘎"叫.必须先有鸭子,然后才能有"嘎嘎".
>
> 但是如果我们只想听"嘎嘎",那么只要它能"嘎嘎"就行, 到底是不是鸭子都没关系.
>
> 鸭子类型关注的只是行为.

和以上讲到的实例一样, 在实际研究复杂系统的时候, 我们只关注系统的行为, 根据行为进行建模,得到一个"函数",至于到底生成这个行为的真实函数是什么, 我们不需要知道,可能也无法知道.

创世论也是如此, 古代人总是认为世界的诞生必然会有一个有形函数(上帝)存在, 世界的变化都是这个函数在起作用. 如果这世界真的是上帝编写的一个函数. 那么这个世界的值域, 作用域, 映射规则都是已知的. 这个函数的导数(变化) 行为都是可以预测的. 我们从哪里来来, 最终命运是什么都写在这个函数里了.

以上这个话题可以看看这本书 :计算中的上帝



但是如果有我们有一个系统能够重现这种变化, 到底有没有上帝就不是重点了. 这就是积分第二定理的威力所在.

Theorem

6.2 用于构造反导函数定理

如果f 是区间上的连续函数, a 点是区间内的任意一点, 那么可以定义一个函数F,作为f 的反导函数

$$F(x)=\int_a^x f(t)dt$$

构建定理和基础定理的联系

简单说 基础定理可以由构造定理得出

$$F(x)=\int_a^x \frac{sint}{t}dt$$

传入一个x, 利用积分计算返回值. 这个函数定义为 $Si(x)$

计算方法是:

1. 遍历数组(length=n)一组值,取第一个值变量x
2. 传入值x, 作为定积分的上界
3. 使用 黎曼和公式 计算$a \to x$ 的定积分值
4. 重复 1-3 直到数组元素x_n
5. 返回计算结果

Example

example1 用数据 [0,1,2,3] 计算$Si(x)$

由于积分在0处无定义, 用 $a=0.0001$ 代替0, $x_1=0.00011$.

结果如下, 黎曼和公式使用之前定义的, 值返回左和与右侧和的平均值

	x	Six
1	0.00011	0.0001
2	1.0	0.9461
3	2.0	1.6054
4	3.0	1.84865

```
let
    f(t)=sin(t)/t
    a= 0.00001
    n=250

    bspan=[0.00011,1,2,3]

    function curryf(a,n,f)   #科里化函数传递一组参数,返回一个新函数等待调用

        return  function (b)
                    res=getRiemannSum(a,b,n,f)
                    return (res["leftsums"]+res["rightsums"])/2
                end
    end

    getVal=curryf(a,n,f)    #生成函数等待传递参数 b

    res=[getVal(b) for b in bspan]

    @show df=DataFrame(;x=bspan,Six=res)

end
```

```
df = DataFrame(; x = bspan, Six = res) = 4×2 DataFrame
 Row │ x        Six
     │ Float64  Float64
─────┼──────────────────
   1 │ 0.00011  0.0001
   2 │ 1.0      0.9461
   3 │ 2.0      1.6054
   4 │ 3.0      1.84865
```

```
getRiemannSum (generic function with 1 method)
```