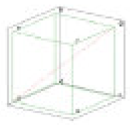


ch10 sec10.2 泰勒级数

Table of Contents

在前面我们已经见过了余弦函数在$x=0$ 处的泰勒多项式:

$$cosx \approx P_0(x)=1$$

$$cosx \approx P_2(x)=1-\frac{x^2}{2!}$$

$$cosx \approx P_4(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}$$

$$cosx \approx P_6(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}$$

$$cosx \approx P_8(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}$$

在$x=0$ 处, 阶数越高, 对函数的近似就越好, 不断的添加后续的项, 如果添加项目没有限制

$$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-....$$

这样的无限项和的形式就是泰勒级数

如果泰勒级数是部分项的和, 就是泰勒多项式.

$sinx$ 和e^x 都有类似余弦函数的定义:

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+....$$

$$sinx=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-...$$

$$cosx=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-...$$

泰勒级数的通项给出了级数中每一项的表达式. 例如, $\frac{x^n}{n!}$ 是e^x的泰勒展开式的通项. $(-1)^k x^{2k}/(2k)!$ 是 余弦函数泰勒展开式的通项. n 和k 表示索引

```
md"""
在前面我们已经见过了余弦函数在$x=0$ 处的泰勒多项式:

$cosx \approx P_0(x)=1$
$cosx \approx P_2(x)=1-\frac{x^2}{2!}$
$cosx \approx P_4(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}$
$cosx \approx P_6(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}$
$cosx \approx P_8(x)=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}$

在$x=0$ 处, 阶数越高, 对函数的近似就越好, 不断的添加后续的项, 如果添加项目没有限制

$1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-....$

这样的无限项和的形式就是泰勒级数

如果泰勒级数是部分项的和, 就是泰勒多项式.
```

```
  $sinx$ 和$e^x$ 都有类似余弦函数的定义:

$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\frac{x^4}{4!}+....$

$sinx=x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-...$
$cosx=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+\frac{x^8}{8!}-...$

泰勒级数的通项给出了级数中每一项的表达式. 例如, $\frac{x^n}{n!}$ 是$e^x$的泰勒展开式的通项.
  $(-1)^kx^{2k}/(2k)!$ 是 余弦函数泰勒展开式的通项. $n$ 和$k$ 表示索引

"""
```

泰勒级数的一般形式

如果一个函数在$x=0$ 处可导, 那么它就有泰勒级数表示. 但是要注意, 函数的泰勒级数展开式并不一定在任何x 处都能收敛. 对于可以可以收敛到$f(x)$的x,有一下表达式:

Definition

$f(x)$ 在$x=0$ 处的泰勒级数为:

$$f(x)=f(0)+f'(0)+\frac{f''(0)}{2!}x^2+\frac{f'''(0)}{3!}x^3+...+\frac{f^(n)(0)}{n!}x^n+...$$

此外, 在$x=a$ 点, 如果函数的所有阶导数都存在, 而且级数收敛于$f(x)$, 可以有如下表达式:

Definition

$f(x)$ 在$x=a$ 处的泰勒级数为:

$f(x)=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+...+\frac{f^{(n)}(a)}{n!}(x-a)^n$

对于一些函数, 即便在某点$x=a$ 处的泰勒级数是收敛的, 但是很可能收敛不到$f(x)$. 这没关系, 因为一些常见函数的泰勒级数可以在任何点收敛到$f(x)$,我们可以把这些常用的基础函数作为构建模块来构建"复杂函数"

```
md"""

## 泰勒级数的一般形式


如果一个函数在$x=0$ 处可导, 那么它就有泰勒级数表示. 但是要注意, 函数的泰勒级数展开式并不一定在任何$x$ 处都能收敛. 对于可以可以收敛到$f(x)$的$x$,有一下表达式:

!!! definition

	$f(x)$ 在$x=0$ 处的泰勒级数为:

    $f(x)=f(0)+f'(0)+\frac{f''(0)}{2!}x^2+\frac{f'''(0)}{3!}x^3+...+\frac{f^(n)(0)}{n!}x^n+...$

此外, 在$x=a$ 点, 如果函数的所有阶导数都存在, 而且级数收敛于$f(x)$, 可以有如下表达式:

!!! definition
	$f(x)$ 在$x=a$ 处的泰勒级数为:

	$f(x)=f(a)+f'(a)(x-a)+\frac{f''(a)}{2!}(x-a)^2+\frac{f'''(a)}{3!}(x-a)^3+...+\frac{f^{(n)}(a)}{n!}(x-a)^n$


对于一些函数, 即便在某点$x=a$ 处的泰勒级数是收敛的, 但是很可能收敛不到$f(x)$. 这没关系, 因为一些常见函数的泰勒级数可以在任何点收敛到$f(x)$,我们可以把这些常用的基础函数作为构建模块来构建"复杂函数"


"""
```

二项式级数展开

现在来求函数 $f(x)=(1+x)^p$ 在$x=0$ 处的泰勒级数,p是常数,可以为负值

求各阶导数，并带入$x=0$:

$$f(x)=(1+x)^p, 所以有 : f(0)=1$$

$$f'(x)=p(1+x)^{p-1}, 所以有 : f'(0)=p$$

$$f''(x)=p(p-1)(1+x)^{p-2}, 所以有 : f''(0)=p(p-1)$$

$$f'''(x)=p(p-1)(p-2)(1+x)^{p-3}, 所以有 : f'''(0)=p(p-1)(p-2)$$

由此可得在$x=0$ 处的三阶泰勒多项式为:

$$(1+x)^p \approx P_3(x)=1+px+\frac{p(p-1)}{2!}+\frac{p(p-1)(p-2)}{3!}x^3$$

二项式 $f(x)=(1+x)^p$ 在$x=0$ 处的泰勒级数为:

$$(1+x)^p=1+px+\frac{p(p-1)}{2!}+\frac{p(p-1)(p-2)}{3!}x^3+... \ 当 -1<x<1$$

```
md"""
## 二项式级数展开

现在来求函数 $f(x)=(1+x)^p$ 在$x=0$ 处的泰勒级数,$p$是常数,可以为负值

求各阶导数, 并带入$x=0$:

$f(x)=(1+x)^p, 所以有 : f(0)=1$
$f'(x)=p(1+x)^{p-1}, 所以有 : f'(0)=p$
$f''(x)=p(p-1)(1+x)^{p-2}, 所以有 : f''(0)=p(p-1)$
$f'''(x)=p(p-1)(p-2)(1+x)^{p-3}, 所以有 : f'''(0)=p(p-1)(p-2)$

由此可得在$x=0$ 处的三阶泰勒多项式为:

$(1+x)^p \approx P_3(x)=1+px+\frac{p(p-1)}{2!}+\frac{p(p-1)(p-2)}{3!}x^3$


二项式 $f(x)=(1+x)^p$ 在$x=0$ 处的泰勒级数为:

$(1+x)^p=1+px+\frac{p(p-1)}{2!}+\frac{p(p-1)(p-2)}{3!}x^3+... \ 当 -1<x<1$
"""
```

Example

example 1

求表达式 $\frac{1}{1+x}$ 在 $x=0$ 处的泰勒级数

因为 $\frac{1}{1+x}$ 可以改写为:$(1+x)^{-1}$,所以 $p=-1$. 因此可以按照二项式进项展开

$$\frac{1}{1+x}=(1+x)^{-1}=1+(-1)x+\frac{(-1)(-2)}{2!}x^2+\frac{(-1)(-2)(-3)}{3!}x^3+...+$$

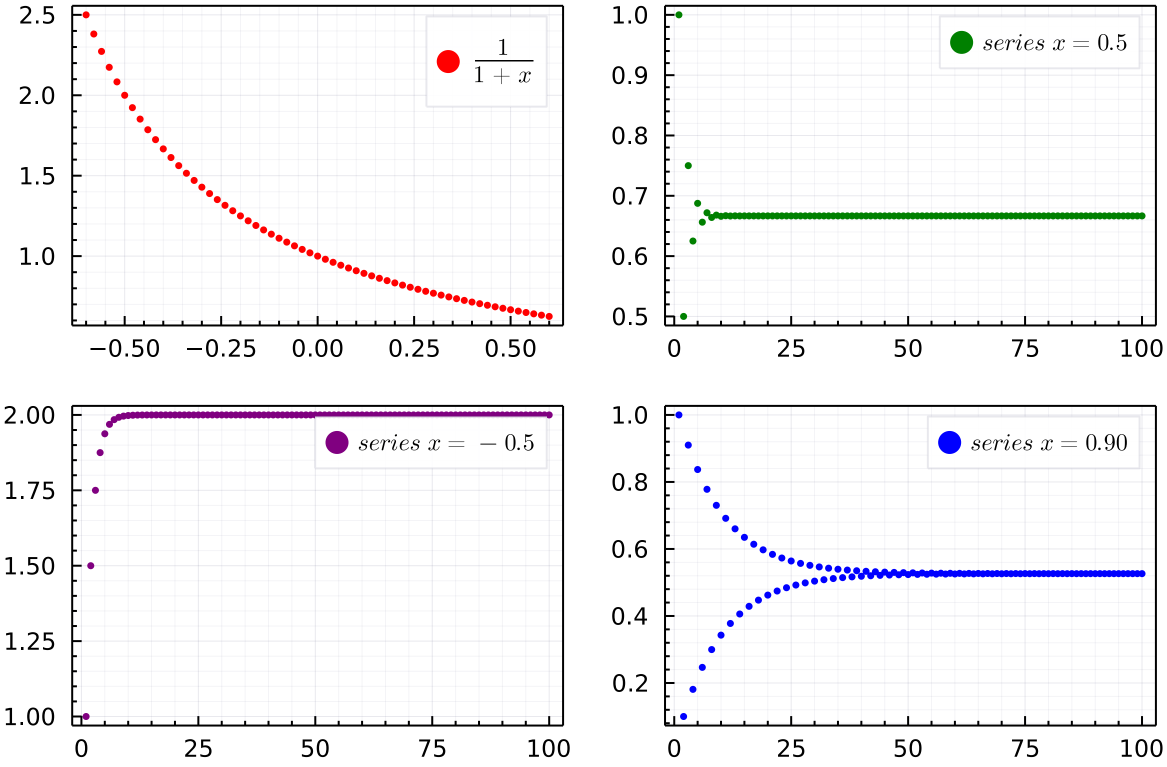
化简得到:

$$\frac{1}{1+x}=1-x+x^2-x^3+...\ ,-1<x<1$$

这也是几何级数, 通项是: $g_{n}(x)=-x*g_{n-1}(x)$

下图是原始函数和级数在三个点的近似收敛情况: 真实函数调用值打印在代码下方

```
md"""

!!! example

    example 1

    求表达式$\frac{1}{1+x}$ 在$x=0$ 处的泰勒级数


因为$\frac{1}{1+x}$ 可以改写为:$(1+x)^{-1}$,所以$p=-1$. 因此可以按照二项式进项展开

$\frac{1}{1+x}=(1+x)^{-1}=1+(-1)x+\frac{(-1)(-2)}{2!}x^2+\frac{(-1)(-2)(-3)}{3!}x^3+...+$

化简得到:

$\frac{1}{1+x}=1-x+x^2-x^3+...  \ , -1<x<1$

这也是几何级数, 通项是:$g_{n}(x)=-x*g_{n-1}(x)$

下图是原始函数和级数在三个点的近似收敛情况: 真实函数调用值打印在代码下方


"""

```




```
let
    n=100
    nspan=1:n
    xspan=-0.6:0.02:0.6

    function g(x)
        return  function f(n)
            return n==0 ? 1 : -x*f(n-1)
        end
    end

    originalfunc(x)=1/(1+x)
    @show originalfunc(0.5)
    @show originalfunc(-0.5)
    @show originalfunc(1)
    fn1=g(0.5)
    fn2=g(-0.5)
    fn3=g(0.90)
    mapreucefunc(fn,n)=mapreduce(x->fn(x),+, 0:n)
    res1=[mapreucefunc(fn1,n) for n in 0:n]
    res2=[mapreucefunc(fn2,n) for n in 0:n]
    res3=[mapreucefunc(fn3,n) for n in 0:n]

    p1=scatter(nspan, res1,label=L"series \ x=0.5",ms=2, color=:green, frame=:box)
    p2=scatter(nspan, res2,label=L"series \ x=-0.5",ms=2, color=:purple, frame=:box)
    p3=scatter(nspan, res3,label=L"series \ x=0.90",ms=2, color=:blue, frame=:box)
    p4=scatter(originalfunc,xspan, label=L"\frac{1}{1+x}",ms=2, color=:red,
frame=:box)

    plot!(p4,p1,p2,p3)

end
```

```
originalfunc(0.5) = 0.6666666666666666
originalfunc(-0.5) = 2.0
originalfunc(1) = 0.5
```

下面取更多的x,看看级数的收敛情况

终端显示的是实际函数的调用情况

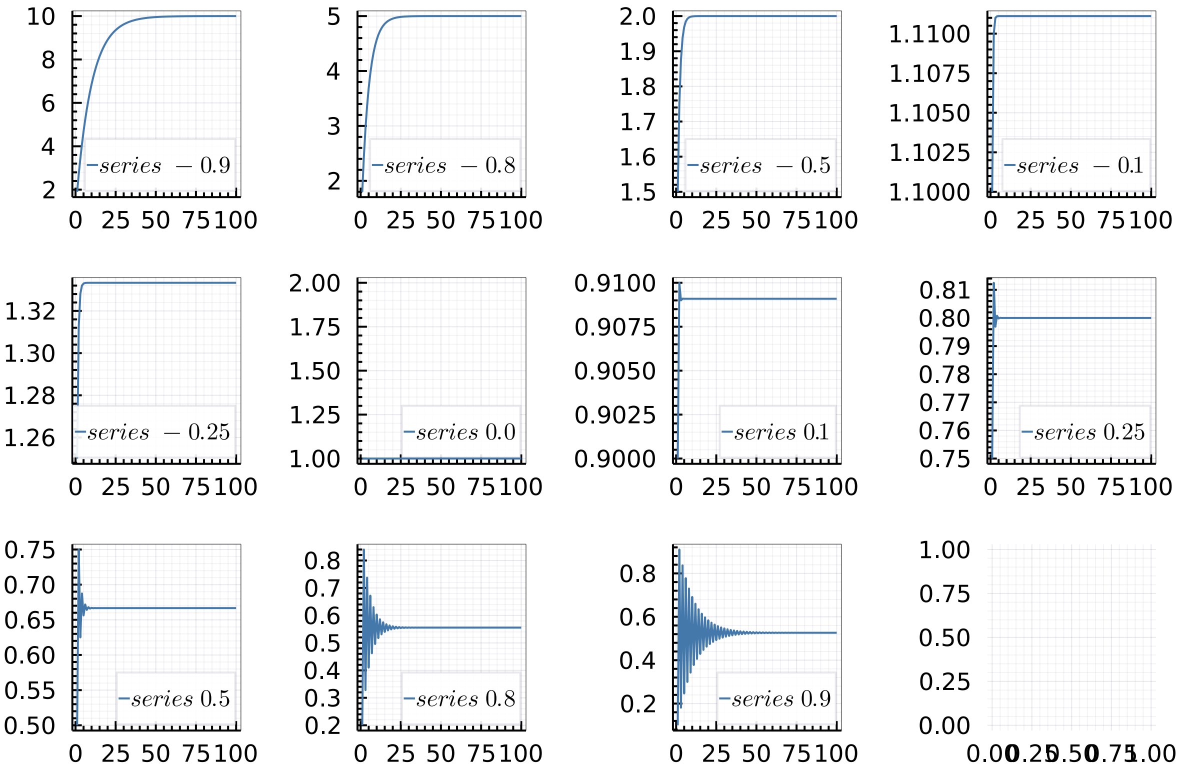
图列出的是用二项式级数在各个不同x 取值处的收敛近似值,在$-1<x<1$区间的收敛情况

```
md"""
下面取更多的$x$,看看级数的收敛情况

终端显示的是实际函数的调用情况

图列出的是用二项式级数在各个不同$x$ 取值处的收敛近似值,在$-1<x<1$区间的收敛情况\
"""
```

```
["at -0.9 val is 10.000000000000002", "at -0.8 val is 5.0000000000000001", "at -0.5 val i
```

```
let
    xcollection=[-0.900,-0.80,-0.5,-0.1,-0.25,0,0.1,0.25,0.5,0.80,0.900]
    originalfunc(x)=1/(1+x)
     showatshell(originalfunc,xcollection)  #在本页面最底部有函数定义
end
```

```
str = "at -0.9 val is 10.000000000000002"
str = "at -0.8 val is 5.0000000000000001"
str = "at -0.5 val is 2.0"
str = "at -0.1 val is 1.1111111111111112"
str = "at -0.25 val is 1.3333333333333333"
str = "at 0.0 val is 1.0"
str = "at 0.1 val is 0.9090909090909091"
str = "at 0.25 val is 0.8"
str = "at 0.5 val is 0.6666666666666666"
str = "at 0.8 val is 0.5555555555555556"
str = "at 0.9 val is 0.5263157894736842"
```




```julia
let
    #最后一幅图没有意义，为了绘图用的小技巧
    n=100
    nspan=1:n

    plotarr=[]
    xcollection=[-0.900,-0.80,-0.5,-0.1,-0.25,0,0.1,0.25,0.5,0.80,0.900]
    originalfunc(x)=1/(1+x)
    function g(x)
        return  function f(n)
             return n==0 ? 1 : -x*f(n-1)
        end
    end
    function mapreucefunc(fn)
      return function  (n)
             return mapreduce(fn,+, 0:n)
      end
    end


    for x in xcollection
        fn=g(x)
        mapfunc=mapreucefunc(fn)
        p=plot(mapfunc,nspan,label=L"series\
        %$(x)",lw=1,frame=:semi,legend=:bottomright)
        push!(plotarr, p)
    end


    p1=plot([0],[0],label=false)
    plot!(plotarr...,p1)

end
```

Info

代码中的$xcollection$区间和$nspan$区间可能会让人迷惑, $xcollection$ 区间是真实函数定义域中抽取的一些离散值,$nspan$区间是级数需要的项数, 级数的方法就是针对原函数定义域中的某个点,比如$x=0.5$,用多项相加的方法来近似函数在该点的映射值.

比如我们要用级数近似$x=[0.5,0.2,0.1]$, $3个x$ 的值, 取级数的$n=100项$. 那么实际需要调用通项式次数为:

$$x_{number}*n_{number}=300次$$

看起来比较复杂,其实也是函数, 不过与普通的一元函数相比,级数表示方法特殊一点, 特殊在映射规则是一个多项式. 为什么要用100项多项式的和来近似一个点的取值呢?

这里再重复一个之前讲到的观点: 对于函数我们要关注它的行为, 而不要关注它的形式. 如果两个函数的行为近似一致, 那么任意一个函数都可以用来表现同一个行为.

在实际应用中,那种方法方便就用那种方法. 对于简单的函数比如$f(x)=\frac{1}{1+x}$, 它的表达式和行为比较简单, 很容易理解. 但是更为复杂的函数呢? 或许用级数方法是更容易表示函数的行为.

函数就像一个宠物, 你需要理解它的行为, 知道它的脾气,这样你才能和它融洽相处.

```
md"""

!!! info
    代码中的$xcollection$区间和$nspan$区间可能会让人迷惑， $xcollection$ 区间是真实函数定义域中抽取的一些离散值,$nspan$区间是级数需要的项数， 级数的方法就是针对原函数定义域中的某个点,比如$x=0.5$,用多项相加的方法来近似函数在该点的映射值.

    比如我们要用级数近似$x=[0.5,0.2,0.1]$, $3个x$ 的值， 取级数的$n=100项$. 那么实际需要调用通项式次数为:

    $x_{number}*n_{number}=300次$

    看起来比较复杂,其实也是函数， 不过与普通的一元函数相比,级数表示方法特殊一点， 特殊在映射规则是一个多项式. 为什么要用$100$项多项式的和来近似一个点的取值呢?

    这里再重复一个之前讲到的观点: 对于函数我们要关注它的行为， 而不要关注它的形式. 如果两个函数的行为近似一致， 那么任意一个函数都可以用来表现同一个行为.

    在实际应用中,那种方法方便就用那种方法. 对于简单的函数比如$f(x)=\frac{1}{1+x}$， 它的表达式和行为比较简单， 很容易理解. 但是更为复杂的函数呢? 或许用级数方法是更容易表示函数的行为.

    函数就像一个宠物， 你需要理解它的行为， 知道它的脾气,这样你才能和它融洽相处.
"""
```

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