欧拉反正切公式的应用

Abel

1 arctan 1 可以直接编程实现

欧拉的反正切公式是这样的:

$$\mathrm{arctan}(t) = \frac{t}{1+t^2}\sum_{n=0}^{\infty}\frac{(2n)!!}{(2n+1)!!}\left(\frac{t^2}{1+t^2}\right)^n \quad (t \in R)$$

莱布尼兹-格里高利公式是这样的:

$$\mathrm{arctan}(x) = \sum_{n=0}^{\infty}\frac{(-1)^n}{2n+1}x^{2n+1} \quad (|x| \leq 1)$$

欧拉公式与莱布尼兹-格里高利 (J.Gregory) 公式最大的不同是定义域的不同. 欧拉公式在整个实数范围内收敛, 而莱布尼兹-格里高利公式要求自变量的绝对值小于等于 1 才能收敛.

另一个不同是, 如果期望直接计算 arctan 1, 得到 $\frac{\pi}{4}$, 以此来计算 π 值. 莱布尼兹-格里高利公式收敛很慢, 既不适合手动计算, 也不适合编程计算. 而用欧拉公式来计算圆周率, 可以直接用 arctan 1.

将 1 代入欧拉公式，可以得到

$$\frac{\pi}{4} = \frac{1}{2}\sum_{n=0}^{\infty}\frac{(2n)!!}{(2n+1)!!}(\frac{1}{2})^n$$

$$\pi = 2\sum_{n=0}^{\infty}\frac{n!}{(2n+1)!!}$$

可以展开

$$\pi = 2\left(1 + \frac{1}{3} + \frac{1\cdot 2}{3\cdot 5} + \frac{1\cdot 2\cdot 3}{3\cdot 5\cdot 7} + \frac{1\cdot 2\cdot 3\cdot 4}{3\cdot 5\cdot 7\cdot 9}\cdots\right)$$

这种形式就方便编程实现了.

也可写作递归的形式

$$\pi = 2\left(1 + \frac{1}{3}(1 + \frac{2}{5}(1 + \frac{3}{7}(1 + \frac{4}{9}(1 + \cdots))))\right)$$

网络上广为流传的一个简短的 c 语言程序就是用这个公式来计算的.

```c
#include<stdio.h>
int a=10000,b,c=2800,d,e,f[2801],g;
main(){
  for(;b-c;)
    f[b++]=a/5;
  for(;d=0,g=c*2;c-=14,printf(”%.4d”,e+d/a),e=d%a)
    for(b=c;d+=f[b]*a,f[b]=d%--g,d/=g--,--b;d*=b);
}
```

2 两个角的和也可以计算

欧拉最初选用这样两个角来计算圆周率

$$\frac{\pi}{4} = \arctan\frac{1}{7} + \arctan\frac{3}{4}$$

这两个值代入公式，可以得到

$$\pi = \sum_{n=0}^{\infty} \frac{1}{100^{n+1}} (56 \cdot 2^n + 192 \cdot 36^n) \frac{(2n)!!}{(2n+1)!!}$$

这个公式既适合手动计算，也适合计算机编程计算．在欧拉的展开式中，计算一个角或者多个角，区别不大（收敛的速度）．在莱布尼兹的展开式中，除了用 $\arctan 1$ 展开之外，其它两个角或更多角的展开式收敛速度差别并不大，我很好奇，人们自马青公式以后，为什么热衷于发明那么多关于 $\frac{\pi}{4}$ 的和/差角公式．

马青公式是这样的

$$\frac{\pi}{4} = 4\arctan\frac{1}{5} - \arctan\frac{1}{239}$$

因为他开创了用多个角的反正切展开式来计算圆周率的先河，所以记录下来．人们使用多个角来计算圆周率时，大多使用莱布尼兹-格里高利展开．大概是因为那个公式更方便记忆．最主要的原因是，那个时候欧拉还没有出生，人们还不知道关于反正切的这一种展开式．

现如今有了欧拉的展开式，最大的好处就是满足了人们用 $\arctan 1$ 来计算圆周率的愿望．

$$\pi = 2\left(1 + \frac{1}{3}(1 + \frac{2}{5}(1 + \frac{3}{7}(1 + \frac{4}{9}(1 + \cdots))))\right)$$