用 Stirling 逼近近似计算阶乘的探讨与应用

[江苏省连云港市赣榆高级中学] 仲晨

myheimu@yahoo.com.cn

【关键词】：Stirling 逼近，阶乘，极限论，微积分，数学实验，计算机算法

“阶乘”（factorial）在信息学竞赛中具有重要角色，更广泛的说，“阶乘”在数学领域也是占有重要地位。在许多人刚刚学习计算机语言的时候，大多会被要求写一个算阶乘的程序，而在学习高精度算法的时候，也会写一个计算较大数字阶乘的程序。不过，在实际的运用之中，可能遇到更大数字的阶乘计算和不同要求的阶乘结果，例如：TOJ（同济大学 ACM 网络题库，http://acm.tongji.edu.cn/problem.php）的 1016 题——“求 N！左边第二位的数字”，这就需要一定的精度思考了。

可是我们通常对于较大数字阶乘的要求是求结果位数或前几位数字，这怎么办呢？

在刘汝佳的《算法艺术与信息学竞赛》一书中，（Page241）介绍了 Stirling 公式：

$$n! \sim \sqrt{2n\pi}\left(\frac{n}{e}\right)^n$$

其中的～符号是指“同阶”或“相当”，即两者随 n 增加的大致速度相同，在 n 较大时，两者极其相近。

这是一个极限的概念（现行教材高二下学期数学内容），属于微分学内容，准确写法为：

$$\lim_{n\to+\infty}\frac{n!}{\sqrt{2n\pi}(n/e)^n}=1$$

但遗憾的是在《算法艺术与信息学竞赛》书中只提供了这个算式，并无他物！

本人近日看到一本数学科普读物——《好玩的数学——不可思议的 e》（陈任政著，科学出版社），其中 5.12 节简介了 Stirling 逼近近似算阶乘，本人感到好奇，于是对这种算法的具体步骤进行了分析，并研究了它的精确度，故为本文。在 2005 年 8 月 7 日完工之日，笔者上网搜索了一下，找到了一些关于 Stirling 逼近的文章，偶然地在臺灣亞洲聚合公司蔡永裕《談 Stirling 公式的改良》（刊自台湾《數學傳播》20 卷 4 期，民国 85 年 12 月）一文中找到同感，蔡先生的做法于笔者方向不同，作出来的结果比笔者的算法精确一个数量级左右，惭愧，于是，笔者又再次研究，寻找更好算法，写于本文后部。

在 1730 年，棣莫弗（棣，音 Dì）（法国数学家，Abraham De Moiver，1667～1754）发表的《分析杂论》中首先对 n!地一个无穷级数展开式给出了近似公式：

$$n!\approx\sqrt{2n\pi}(n/e)^n$$

但是，现在我们叫这个式子为“Stirling 逼近”，中文叫做“斯特林逼近”，这是为什么呢？

因为棣莫弗的朋友苏格兰数学家斯特林(James Stirling,1696~1770)在同年的《微分法或无穷级数的简述》中也给出了等价的级数。

事实上，棣莫弗首先得到的式子是 $n!\sim C\cdot n^{\left(n+\frac{1}{2}\right)}\cdot e^{-n}$，但是，他没有把 C 求出来。而斯特林则利用棣莫弗的发现做了探讨，求出了 $C=\sqrt{2\pi}$。

这些式子的来源是一个无穷级数展开式：

$$\ln(n!)=\left(n+\frac{1}{2}\right)\ln n-n+\ln\sqrt{2\pi}+\frac{B_2}{1\cdot2}\cdot\frac{1}{n}+\frac{B_4}{3\cdot4}\cdot\frac{1}{n^3}+\cdots+\frac{B_{2k}}{(2k-1)\cdot2k}\cdot\frac{1}{n^{2k-1}}+\cdots$$ 其中 B_2=1/6，B_4=-1/30，B_6=1/42 … B_{2k}是雅格布·伯努力数。（具体内容请参见后文介绍）

这里介绍一下，还没上高中的同学还没有学到，“乘方”的逆运算有两种：开方和对数。

对于一个幂：$a^n=b$，其中 a 成为底数，n 成为指数，b 成为幂。已知 a 和 n 求 b，就是乘方运算；已知 b 和 n，求 a，就是开方运算；而已知 a 和 b 求 n，就是对数运算，写做：$\log_a b=n$，这里 n 就称为以 a 为底 b 的对数（logarithm）。

当底数为 10 的时候，叫做**常用底数**，简写做 $\lg b=n$；当底数为 e 的时候，叫做**自然对数**，简写做 $\ln b=n$。

至于 e 的含义：e 是重要性仅次于 π 的数，是极限论的主要内容，具体的说，即：

$$\lim_{n\to+\infty}\left(1+\frac{1}{n}\right)^n=e$$

意思是当 n 趋向于正的无限大的时候，$\left(1+\frac{1}{n}\right)^n$ 趋向于 e。

e 是无理数，即无限不循环小数，也是超越数，即不能满足某个整数系数代数方程的数（不能满足某个整数系数代数方程的数叫做代数数）。

目前 e 只算到了几千位。

$$e=2.71828182845904523536028747135266249775724709 3...$$

特别说明的是，在 Pascal 语言中，exp(n)函数就是 **e 的 n 次方**。

另外，有个著名的公式被成为“整个数学中最卓越的公式”：

$$e^{i\pi}+1=0$$

其中的 i 为虚数的单位，$i=\sqrt{-1}$。来自算术的 0、1，来自代数的 i，来自几何的 π，来

自分析学的e，奇妙的组成了一个公式！

这是**欧拉**（瑞士数学家，Leonhard Euler，1707~1783）发现的！所以称作“欧拉公式”。

不过，真正的欧拉公式是：

$$\begin{cases} e^{ix}=(\cos x+i\sin x) \\ \cos x=\dfrac{e^{ix}+e^{-ix}}{2} \\ \sin x=\dfrac{e^{ix}-e^{-ix}}{2i} \end{cases}$$

那个“最卓越的公式”只是欧拉公式的一个推倒公式。

言归正传，由公式

$$\ln(n!)=\left(n+\frac{1}{2}\right)\ln n-n+\ln\sqrt{2\pi}+\frac{B_2}{1\cdot 2}\cdot\frac{1}{n}+\frac{B_4}{3\cdot 4}\cdot\frac{1}{n^3}+\cdots+\frac{B_{2k}}{(2k-1)\cdot 2k}\cdot\frac{1}{n^{2k-1}}+\cdots$$

两边同时取e的幂，得

$$e^{\ln(n!)}=e^{\left((n+1/2)\ln n-n+\ln\sqrt{2\pi}+\frac{B_2}{1\cdot 2}\cdot\frac{1}{n}+\frac{B_4}{3\cdot 4}\cdot\frac{1}{n^3}+\cdots+\frac{B_{2k}}{(2k-1)\cdot 2k}\cdot\frac{1}{n^{2k-1}}+\cdots\right)}$$ 即：

$$n!=\frac{n^{(n+1/2)}}{e^n}\cdot\sqrt{2\pi}\cdot e^{\left(\frac{B_2}{1\cdot 2}\cdot\frac{1}{n}+\frac{B_4}{3\cdot 4}\cdot\frac{1}{n^3}+\cdots+\frac{B_{2k}}{(2k-1)\cdot 2k}\cdot\frac{1}{n^{2k-1}}+\cdots\right)}$$

再经过近似等处理（这些处理比较麻烦，而且牵扯到微积分内容）,我们得到了**Stirling公式**：

$$n!\backsimeq\sqrt{2n\pi}(n/e)^n$$

注：在本文后部有Stirling公式的推倒过程。

当然，我们可以得到它得更具体形式：

$$n!=\sqrt{2n\pi}\left(\frac{n}{e}\right)^n e^{\frac{\theta}{12n}}(0<\theta<1)$$

其中的θ是不定的，在（0,1）区间之内。

讲到这里，我们有了公式，可以开始计算了。

但是，我们计算什么呢？难道我们要计算N！的值吗？

虽然这个式子比阶乘原始计算式简单，但是实际计算的时候仍然得到的将是上百上千位的数字，而且这个式子毕竟是近似公式，无法真正得到确切得数！

难道我们辛苦的到的式子无用了吗？

答案当然是否定的！我们可以求N！的位数！

求位数的方法是**取常用对数（以10为底的对数）**。那，这是为什么呢？看看下表：

n	$\lg n$	n位数
1	0.000000	1
10	1.000000	2
100	2.000000	3
1000	3.000000	4
10000	4.000000	5
100000	5.000000	6
1000000	6.000000	7
10000000	7.000000	8
100000000	8.000000	9

好了！我们知道了n的位数等于$\lg n+1$，对吗？

不对！$\lg n$不一定是整数，比如$\lg 5=0.69897\cdots$，所以，我们得到的公式是：

$$n\text{的位数}=\lfloor \lg n\rfloor+1$$

其中$\lfloor n\rfloor$是取整符号，指不大于n的最大整数，在Pascal语言中为trunc(n)函数。

如果我们用 Stirling 逼近算出 n！再算它的位数就太不值得了，这里我们就要运用**对数的一些特性**来简化公式。

我们两边取常用对数：

$$\lg(n!) \approx \lg\left[\sqrt{2n\pi}\cdot(n/e)^n\right] = \lg\sqrt{2n\pi} + \lg(n/e)^n$$

进而化简为:

$$\lg(n!) = \frac{1}{2}\lg(2n\pi) + n\lg(n/e)$$

现在我们可以来求值了！

以求 1000！的位数为例，

$$\begin{aligned}\lg(1000!) &\approx \frac{1}{2}\lg(2\times 1000\times\pi) + 1000\times\lg\left(\frac{1000}{e}\right)\\ &\approx 0.5\times\lg 6283.1853 + 1000\times\lg 367.87944\\ &\approx 2567.604\end{aligned}$$

所以 1000！的位数为 2568 位！

我们可以看到，这里的计算复杂度很低，都是 O(n) 级别的！对于其计算机编程运用来说，计算过程中数字不大不会溢出，取对数后的精确度也可以保证。

这样我们就解决了计算阶乘位数问题！而且可以绝对放心，这个公式在位数方面的准确率是 99.9999999999999％以上。(而且这个仅有的可能性也只是从正态性来推测的)

用 Pascal 语言来书写：

```
Trunc(0.5 * Ln(2 * n * 3.1415926) / Ln(10) + n * Ln(n / Exp(1)) / Ln(10)) + 1
```

建议使用分步计算，这样能避免计算过程中的溢出现象。

这样，笔者使用批处理计算了几个值：

n	*n*!位数
1	1
10	7
100	158
1000	2568
10000	35660
100000	456574
1000000	5565709
10000000	65657060
100000000	756570557
1000000000	8565705523

由于 **Longint** 范围的限制,无法计算再大的数值，但是可以通过使用 **int64** 或者 **extended** 来解决。

不过，我们有点不甘心，只计算位数难免用途不广，其实，我们可以用来计算 n！的前几位！

什么原理呢？

例如计算 1000！时，我们得到的数为 2567.60460802722309714418713610939 45……

而我们从下表中可以看出点东西来:

n	lg *n*
5	0.698970004336019……
50	1.698970004336019……
500	2.698970004336019……
5000	3.698970004336019……

我们可以得知：一个数增大 10 倍，它的常用对数整数部分将增加 1，而小数部分不变。

我们得到的更重要的结果是：一个数的常用对数的小数部分所对应的幂的各位数字（不考虑小数点）与原数各位数字相同。

所以，对于 1000！来说：2567.60460802722309714418713610939 45…… 的小数部分为 0.60460802722309714418713610939 45……，

它的10为底的幂为4.023537257719700539383631576563 5……，也就是说1000！这个2568位数的前几位为402353725771970053938363157656 35……。

Well！我们可以求出一个阶乘的前几位数了！

但是，不要高兴太早！这里的精度无法保证！

下面让我们来看看精确度方面：

特别说明的是，这里说的精确度是指n!与Stirling逼近公式的精确度，而不是位数计算的精确度，因为位数计算基本上是精确的！

n	逼近值	n！准确值	相对误差
10	3598695.619	3628800	0.008365359
20	2.42279E+18	2.4329E+18	0.004175011
30	2.64517E+32	2.65253E+32	0.002781536
40	8.14E+47	8.16E+47	0.002085461
50	3.04E+64	3.04E+64	0.001668034
60	8.31E+81	8.32E+81	0.001389841
70	1.20E+100	1.20E+100	0.001191177
80	7.15E+118	7.16E+118	0.001042204
90	1.48E+138	1.49E+138	0.000926351
100	9.32E+157	9.33E+157	0.000833678
110	1.59E+178	1.59E+178	0.000757861
120	6.68E+198	6.69E+198	0.000694684
130	6.46E+219	6.47E+219	0.000641230
140	1.35E+241	1.35E+241	0.000595414
150	5.71E+262	5.71E+262	0.000555709
160	4.71E+284	4.71E+284	0.000520968
170	7.25E+306	7.26E+306	0.000490316

可以看到，本身它的相对误差就很小，并且这个误差是在逐渐减小。

当n＝1000的时候，相对误差只有0.000083336802873339866575 87……！

这个误差可以保证我们能够取得阶乘值的前3-4位准确值，但是，还是有点遗憾，感觉不是太好，我们最起码需要7-8位的精确度，可是，我们能做到吗？

可以！

还记得吗？我们曾得到了一个更精确的算式：

$$n!=\sqrt{2n\pi}\left(\frac{n}{e}\right)^n e^{\frac{\theta}{12n}}\ (0<\theta<1)$$

（其中的θ是不定的，在（0，1）区间之内。）

从这个式子里我们也可以看出准确值和逼近值之比就是

$$e^{\frac{\theta}{12n}}\ (0<\theta<1)$$

随着 n 的增大，显然这个值是随 n 逐渐缩小的！并且是缩得很小，这也符合我们上面表格所体现的内容。

可是，这里的θ是不固定的，要是固定的，我们就可以求出准确值。但是，有没有想看看θ的变化趋势呢？我们用上面算出的相对误差反算θ。

（计算公式为 ln（相对误差＋1）×12×n＝θ）

n	θ
10	0.999667612
20	0.999916726
30	0.999962975
40	0.999979170
50	0.999986668
60	0.999990741
70	0.999993198
80	0.999994792
90	0.999995885
100	0.999996667
110	0.999997245
120	0.999997685
130	0.999998028
140	0.999998299
150	0.999998519
160	0.999998698
170	0.999998847
180	0.999998971
190	0.999999077
200	0.999999167

我们惊喜地发现θ竟然十分接近 1，而且在逐渐地逼近 1，这是个令人兴奋地消息！

实际上，即使是求 1 的阶乘，θ 也会达到 0.9727376027，这是一个本身就是一个很“精确”的数字了！当 n＝1000 时，θ 将 0.99999996665875876427498746773752，与 1 的差别只有 0.00000003334124123572501253226 3（约等于 3.33412×10^{-8}）！

如果可以精确到这样，我们就太高兴了！

我们把 θ 用 1 带入，然后求值，这次得到的结果出奇的好！

下表是经过 θ=1 带入修正的各项指标：

n	修正后逼近值	n！准确值	二者比例
10	3.62881005142693E+06	3.62880000000000E+06	0.999997230103867
20	2.43290285233215E+18	2.43290200817664E+18	0.999999653025394
30	2.65252887092925E+32	2.65252859812191E+32	0.999999897151981
40	8.15915318654567E+47	8.15915283247897E+47	0.999999956604970
50	3.04140938775049E+64	3.04140932017133E+64	0.999999977780315
60	8.32098721974147E+81	8.32098711274139E+81	0.999999987140939
70	1.19785717669724E+100	1.19785716699698E+100	0.999999991901990
80	7.15694574345356E+118	7.15694570462638E+118	0.999999994574895
90	1.48571597014272E+138	1.48571596448176E+138	0.999999996189743
100	9.33262157031762E+157	9.33262154439441E+157	0.999999997222301
110	1.58824554483731E+178	1.58824554152274E+178	0.999999997913062
120	6.68950292420235E+198	6.68950291344912E+198	0.999999998392522
130	6.46685549739670E+219	6.46685548922047E+219	0.999999998735671
140	1.34620124893450E+241	1.34620124757175E+241	0.999999998987707
150	5.71338396114816E+262	5.71338395644585E+262	0.999999999176966
160	4.71472363918940E+284	4.71472363599206E+284	0.999999999321839
170	7.25741561941125E+306	7.25741561530799E+306	0.999999999434611
180	2.00896062594820E+00	2.00896062499134E+00	0.999999999523704
190	9.68032267917558E+00	9.68032267525524E+00	0.999999999595020

可以看到，这里的差别已经很小了！

当 n＝1000，二者比例达到了 0.999999999997221，这足以保证 10 位的准确度！

到这里，我们就找到了一个精确度高并且速度快的算法来计算阶乘的前几位！

我们求阶乘前几位的最后公式为

$$10^{frac(0.5\cdot\lg(2n\pi)+n\cdot\lg n-n\cdot\lg e)}\times e^{\frac{1}{12n}}$$

frac(n)为取小数部分

顺便说一下，以上的数据都是用 Free Pascal 计算的，使用的是 Extended 格式，由此看来，Extended 的精确度也是很高的！

用 Pascal 语言来书写：

```
10**frac(0.5*Ln(2*n*3.1415926) / Ln(10) + n * Ln(n / Exp(1)) /
Ln(10))*exp(1/12/n)
```

有个技巧是：在 **FP** 中，可以使用 a**b 来计算 a^b，可以不必再用 exp(y*ln(x))。

笔者希望读者仔细思考，如何书写精度最高，速度最快（尽管速度已经是毫秒级了！）？还请注意数据溢出地情况。

还有，为了输出前几位，需要下点功夫处理一下输出问题。笔者在这里就简略了。

别急，我们的“征途”还没完，我们要继续精确化！

下面是来自 http://mathworld.wolfram.com/StirlingsSeries.html 的资料，介绍 Stirling's Series，即“斯特林级数”，也就是 Stirling 逼近的原始式。比较抽象，读者浏览一下即可。

笔者英语不佳，勉强翻译了一下，便于大家阅读。

Stirling's Series

斯特林级数

The asymptotic series for the gamma function is given by

这个Γ函数的渐进级数如下（译者注：Γ为γ的大写，希腊字母，读做“伽马”）

$$\Gamma(x) \sim e^{-x} x^{x+1/2} \sqrt{2\pi}\left(1+\frac{1}{12x}+\frac{1}{288x^2}-\frac{139}{51840x^3}+\frac{571}{2488320x^4}+\cdots\right)$$

The coefficient a_n of x^{-x} can given explicitly by

x^{-x} 的系数 a_n 可以明确地给出

$$a_n = \sum_{k=1}^{2n}\left[(-1)^k \cdot \frac{d_3(2n+2k,k)}{2^{n+k}\cdot(n+k)!}\right]$$

where $d_3(n,k)$ is the number of permutations of *n* with *k* permutation cycles all of which are ≥3(Comtet 1974, p. 267). Another formula for the a_n s is given by the recurrence relation

上面的 $d_3(n,k)$ 是一个以*k*为循环的*n*的变量，它是始终≥3的(Comtet 1974, p. 267)。另外的一个计算 a_n 的关系如下

$$b_n = \frac{1}{n+1}\left(b_{n-1} - \sum_{k=1}^{n-1}(kb_k b_{n+1-k})\right)$$

with $b_0 = b_1 = 1$, then

当 $b_0 = b_1 = 1$，则

$$a_n = (2n+1)!!b_{2n+1}$$

where $x!!$ is the double factorial (Borwein and Corless 1999).

这里的 $x!!$ 的意思是双阶乘(Borwein and Corless 1999).

（译者注：把阶乘的定义引申，定义N!! = N*(N-2)*(N-4)*...，若N为偶数，则乘至2，反之，则乘至1，而0!! = 0。我们称之为双阶乘(Double Factorial)）

The series for $x!$ is obtained by adding an additional factor of x,

$x!$ 级数是由x+1的Γ函数获得，

$$x!=\Gamma(x+1)$$
$$=e^{-x}x^{x+1/2}\sqrt{2\pi}\left(1+\frac{1}{12x}+\frac{1}{288x^2}-\frac{139}{51840x^3}-\frac{571}{2488320x^4}+\cdots\right)$$

The expansion of $\ln\Gamma(x)$ is what is usually called Stirling's series. It is given by the simple analytic expression

$\ln\Gamma(x)$ 的展开式通常被叫做斯特林级数。它简单地分析表示为,

$$\ln\Gamma(x)=\frac{1}{2}\ln(2\pi)+\left(x+\frac{1}{2}\right)\ln x-x+\sum_{n=1}^{\infty}\frac{B_{2n}}{2n\cdot(2n-1)\cdot x^{2n-1}}$$
$$=\frac{1}{2}\ln(2\pi)+\left(x+\frac{1}{2}\right)\ln x-x+\frac{1}{12x}-\frac{1}{360x^3}+\frac{1}{1260x^5}-\cdots$$

where B_n is a Bernoulli number. Interestingly, while the numerators in this expansion are the same as those of for the first several hundred terms, they differ at n=574, 1185, 1240, 1269, 1376, 1906, 1910, ... , with the corresponding ratios being 37, 103, 37, 59, 131, 37, 67, ...

$B_{2n}/(2n)$

这里地 B_n 是伯努力数。有趣的是，当n每增加数百后，$B_{2n}/(2n)$ 会出现重复，比如当 n=574, 1185, 1240, 1269, 1376, 1906 , 1910 , ... 时，对应的 是37, 103, 37, 59, 131, 37, 67, ...

$B_{2n}/(2n)$

对于以上内容，单就本文所讨论的主题来说，没有太大用途，许多人在用 Stirling 公式计算大数阶乘的时候（注意，他们是直接计算阶乘近似值，而笔者是通过常用对数反算近似值），常常不使用“Stirling 逼近”而直接使用“Stirling 级数展开式”，这样主要是因为他们注意到了“Stirling 逼近”简单式的误差过大，转而使用 10－100 项的“Stirling 级数展开式”。在本文中，笔者使用“Stirling 逼近”的“精确式”，采用修正的方法求近似值，已经取得了不亚于使用 100 项的“Stirling 级数展开式”的精确度，并且避免了阶乘数值的溢出。

笔者在上文也说过，笔者看了台湾蔡永裕先生的《談 Stirling 公式的改良》一文，感觉非

常有水平，笔者有时很有疑问，台湾地区的计算机学、数学等科学的发展水平还是比较高的！经常性的笔者搜索数学、计算机学的内容都会搜到许多台湾网站的内容，可能还有一个原因就是台湾地区的计算机普及率较高，资料上网率较高。

这里两个网址：

臺灣“中央研究院”數學研究所（数学研究所）http://www.math.sinica.edu.tw/

《數學傳播》杂志（《数学传播》）http://www.math.sinica.edu.tw/media/default.jsp

读者能看得顺 繁体字 更好，如果看不大习惯，就用一些软件来“转换”一下。

再次言归正传，蔡永裕先生的原理与笔者基本相同，都是对 Stirling 逼近公式进行修正，蔡先生文中联想到公式：

$$e=1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\frac{1}{4!}+\frac{1}{5!}+\frac{1}{6!}+\frac{1}{7!}+\cdots+\frac{1}{n!}+\cdots$$

于是设 e 的渐近相等值 E，将 Stirling 公式变为（笔者注：$\doteqdot$ 即≈）

$$n!\doteqdot\left(\frac{n}{E}\right)^n\sqrt{2n\pi}$$

反算得 $\ln E$ 渐近于 1，于是设 $E=e^{\left(1-\frac{1}{F_n}\right)}$

其中 F_n 为修正参数，则导出 $F_n=\frac{1}{1-\ln E}$

然后反算得数据表格。继而欲求 F_n 的表达式，经过试验，选择了 $F_n=12n^2+0.4$

得到了 Stirling 公式的修正版：$n!=\frac{n^n\cdot\sqrt{2n\pi}}{e^{\left[n\cdot\left(1-\frac{1}{12n^2+0.4}\right)\right]}}$

其常用对数形式为：

$$\lg(n!)=\lg\left(\sqrt{2n\pi}\right)+(n+1/2)\cdot\lg n-n\left(1-\frac{1}{12n^2+0.4}\right)\cdot\lg e$$

这样一来，精确度提高了很多，当计算 1000！时，蔡先生的算法误差仅有-2.971E-13，而如果使用原始 Stirling 式的误差为-8.33299E-05，笔者之前的算法误差是 1.118E-12，略逊于蔡式算法，于是笔者思考了一下，使用二次修正的方法来实现精确化。

方法如下：

对于公式：$n!=\sqrt{2n\pi}\left(\frac{n}{e}\right)^n e^{\frac{\theta}{12n}}\ (0<\theta<1)$

因为 θ 接近于 1，则令 $\theta=1-\frac{1}{f(n)}$，$f(n)$ 为修正函数。则 $f(n)=\frac{1}{1-\theta}$

为了寻找 $f(n)$ 的式子，从简单的一次函数试起，我们先试一试 $f(n)/n$，发现竟然是等差数列，在除以 n 后，得到的是一个常量！

n	θ	f(n)	f(n)/n	f(n)/n²
50	0.999986668189609	75008.56753	1500.171351	30.00342701
100	0.999996666761655	300008.5492	3000.085492	30.00085492
150	0.999998518536300	675008.1019	4500.054013	30.00036009
200	0.999999166673422	1200009.727	6000.048637	30.00024319
250	0.999999466670049	1875011.893	7500.047571	30.00019028
300	0.999999629630836	2700008.792	9000.029307	30.00009769
350	0.999999727877187	3674811.34	10499.46097	29.99845991
400	0.999999791661586	4799882.935	11999.70734	29.99926834
450	0.999999835405298	6075529.694	13501.1771	30.00261577
500	0.999999866704792	7502145.139	15004.29028	30.00858056
550	0.999999889796665	9074135.523	16498.42822	29.99714222
600	0.999999907419129	10801367.44	18002.27906	30.00379843
650	0.999999921104972	12675069.96	19500.10763	30.00016558
700	0.999999931990204	14703764.09	21005.37728	30.00768183
750	0.999999940767808	16882711.46	22510.28195	30.01370927
800	0.999999947926410	19203592.46	24004.49057	30.00561321
850	0.999999953865914	21675947.14	25501.11429	30.00131093
900	0.999999958850112	24301402.95	27001.55883	30.00173203
950	0.999999963096340	27097582.94	28523.77151	30.02502264
1000	0.999999966659766	29993790.67	29993.79067	29.99379067

显然，它们都与 30 相近；而且，我们完全可以认为，这里的误差是由计算误差引起的！

于是，我们得到$f(n)=30n^2$关系式。

继而，有 $n!\approx\sqrt{2n\pi}\left(\frac{n}{e}\right)^n e^{\left(\frac{1-\frac{1}{30n^2}}{12n}\right)}$

“前几位”公式：

$$10^{frac(0.5\cdot\lg(2n\pi)+n\cdot\lg n-n\cdot\lg e)}\times e^{\left(\frac{1}{12n}-\frac{1}{360n^3}\right)}$$

这样一来

n	二次修正版逼近值 科学计数法系数	n！准确值 科学计数法系数	相对误差
50	3.04140932016361	3.04140932017133	-2.5383029012E-12
100	9.33262154439367	9.33262154439441	-7.9380946261E-14
150	5.71338395644579	5.71338395644585	-1.0547118734E-14
200	7.88657867364788	7.88657867364790	-2.5535129566E-15

看！仅仅 n=200 的时候，误差就小于10^{-14}！

不过，由于毕竟$f(n)=30n^2$是有误差的，所以误差不会一直减少，而会波动，正如上面的求 $f(n)/n^2$ 的表格，它的值是波动的。

这是因为：**我们不可能用固定的多项式来表示出对数型变化！**

但我们把误差降到了最小，当 n = 1000 时，逼近值 4.0238726007709361277668E+2568 与准确值 4.0238726007709377354370E+2568 的误差仅有-3.9953306821471308041868495761274E-16

事实上，在高等数学里，根据 Taylor 展式可以算得：

$$\sqrt{2n\pi}\left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}>n!>\sqrt{2n\pi}\left(\frac{n}{e}\right)^n e^{\left(\frac{1}{12n}-\frac{1}{360n^3}\right)}$$

看似我们属于“碰对了”，其实这就是“数学实验”的魅力！

到此为止，我们可以说获得了成功！！！

在编程方面，注意点很多，主要还是溢出的问题，比如 1/（360*n*n*n）对于较大的可能溢出，而写成 1/360/n/n/n 就可以避免。

```
exp(frac(0.5*Ln(2*n*3.14159265358979323)/Ln(10)+n*Ln(n/Exp(1))/Ln(10))*ln(10))*exp(1/12/n-1/360/n/n)
```

以上内容是笔者看书时偶尔思考到的，并认真得进行了思考和实验，具体地试验比较，这种做法叫做“数学实验”。

《数学实验》是在我国高等学校中新开设的一门课程。现在还处于试点和摸索阶段，有许多不同的想法和作法。课程目的,是使学生掌握数学实验的基本思想和方法，即不把数学看成先验的逻辑体系，而是把它视为一门“实验科学”，从问题出发，借助计算机，通过学生亲自设计和动手，体验解决问题的过程，从实验中去学习、探索和发现数学规律。既然是实验课而不是理论课，最重要的就是要让学生自己动手，自己借助于计算机去“折腾”数学，在“折腾”的过程中去学习，去观察，去探索，去发现，而不是由老师教他们多少内容。既不是由老师教理论，主要的也不是由老师去教计算机技术或教算法。不着意追求内容的系统性、完整性。而着眼于激发学生自己动手和探索的兴趣。

数学实验可以包括两部分主要内容：第一部分是基础部分，围绕高等数学的基本内容，让学生充分利用计算机及软件（比较有名的是 Mathematica）的数值功能和图形功能展示基本概念与结论，去体验如何发现、总结和应用数学规律。另一部分是高级部分，以高等数学为中心向边缘学科发散，可涉及到微分几何，数值方法，数理统计，图论与组合，微分方程，运筹与优化等，也可涉及到现代新兴的学科和方向，如分形、混沌等。这部分的内容可以是新的，但不必强调完整性，教师介绍一点主要的思想，提出问题和任务，让学生尝试通过自己动手和观察实验结果去发现和总结其中的规律。即使总结不出来也没有关系，留待将来再学，有兴趣的可以自己去找参考书寻找答案。

笔者写本文，一来总结自己所想，二来抛砖引玉，希望大家能在数学中寻找灵感，并

优化使之适用于计算机编程，这才是**算法艺术**！

共 12007 字

写于：2005 年 8 月 6 日～8 日

江苏·赣榆

参考文献：

1. 《好玩的数学——不可思议的 e》（陈任政著，科学出版社，2005）；
2. 《談 Stirling 公式的改良》（蔡永裕，臺灣亞洲聚合公司，刊自台湾《數學傳播》20 卷 4 期，民国 85 年 12 月－公元 1996 年 12 月）；pdf 文件下载：http://www.math.sinica.edu.tw/math_media/pdf.php?m_file=ZDIwNC8yMDQwNA==
3. 《談 Stirling 公式》(蔡聰明,載於數學傳播第十七卷第二期,作者當時任教於台大數學系)；

   http://episte.math.ntu.edu.tw/articles/mm/mm_17_2_05/
4. 清华大学在线学习－－《组合数学》之（1.3 节 Stirling 近似公式）；http://www.cs.cityu.edu.hk/~luoyan/mirror/tsinghua/combinemaths/1/1_3.htm
5. Stirling 级数 http://mathworld.wolfram.com/StirlingsSeries.html