三角函数的优化问题

Wang Feng
wanng.fenng@gmail.com

June 11, 2010

在一个数值拟合程序中，在检查程序热点的时候发现正弦函数和余弦函数占用了大约90%以上的运行时间。 而在实际数值计算拟合过程中，并不需要非常精确的三角函数，只需要一个高效并且接近准确值的函数即可。 而且实际工作中所使用的采样卡只有 18 位，采样量化精度为10^{-6}；同时在计算机拟合运算时发现，只要三角函数的绝对误差不大于 10^{-6}，数值计算精度就不受影响。

由于三角函数本身的特殊性质，只要知道了在正弦函数或者余弦函数在区间 $[0,\pi]$ 上的值，其余区间的函数值均可通过反转平移变换得到， 因此以下讨论中只关注正弦函数在区间 $[0,\pi]$ 上的值。

0.1 三角函数的多项式逼近

假如将正弦函数 $\sin x$ 在零点处展开为多项式，可以得到

$$\sin x = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 - \frac{1}{5040}x^7 + \frac{1}{362880}x^9 - \frac{1}{39916800}x^{11} + O\left(x^{13}\right) \tag{1}$$

将余弦函数在零点处展开，可以得到

$$\cos x = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 - \frac{1}{720}x^6 + \frac{1}{40320}x^8 - \frac{1}{3628800}x^{10} + \frac{1}{479001600}x^{12} + O\left(x^{13}\right) \tag{2}$$

直接截取以上的 Taylor 级数的前面若干项对三角函数逼近是一种显而易见的方法，但是却不是最优的方法； 经过尝试之后发现，如果要求精度控制在 10^{-6}，那么如果使用正弦函数逼近则需要前 5 项，余弦函数需要前 6 项。

一种很自然的想法是，如果对上边的多项式进行参数拟合，可能会得到更好的多项式系数，使用这些系数对三角函数进行多项式逼近， 可能只需要很少的项数就可达到更好的精度。

参数拟和的具体操作过程如下：

1. 在区间 $[0,\frac{\pi}{2}]$ 等间距采样 1000000 点，得到一个数列 $x[n]$
2. 计算这个数列的三角函数值，得到另外两个数列 $y_{sin}[n]$ 和 $y_{cos}[n]$
3. 构造两个多项式 $f(n)=\sum_{m=0}^{M} a[i]x[n]^m$ 对 $(x[n], y[n])$ 进行拟合，其中正弦函数只包含奇数次方项， 余弦函数只包含偶数次方项，多项式的项数 M 从少到多逐渐递增

如果确切地知道将要计算的三角函数 $\sin x$ 中的 x 的概率分布，那么上述第一个步骤可以稍加修改， 增加概率信息，也就是对于每一个采样点增加一个权值分量，这个权值对应概率密度函数 $p(x)$。

对正弦函数进行拟合后得到的多项式逼近（5 到 11 阶）为：

$$\begin{aligned}\sin x =&0.99977140848759160 63\,x - 0.16582704394468567033\,x^3 + \\ &0.0075742480210386417538\,x^5\end{aligned} \tag{3}$$

$$\begin{aligned}\sin x =&0.99999748720576542294\,x - 0.16665168104891719958\,x^3 + \\ &0.0083095169821075058614\,x^5 - 0.00018447221849405965569\,x^7\end{aligned} \tag{4}$$

$$\begin{aligned}\sin x =&0.9999999827874411773\,x - 0.16666651524066689039\,x^3 + \\ &0.0083329640842244867 56\,x^5 - 0.00019804759284944954147\,x^7 + \\ &2.5981182589280610796\times10^{-6}\,x^9\end{aligned} \tag{5}$$

$$\begin{aligned}\sin x =&0.99999999999664868078\,x - 0.16666666635797344753\,x^3 + \\ &0.0083333315580773111714\,x^5 - 0.00019840928425862410303\,x^7 + \\ &2.7528502432894751691\times10^{-6}\,x^9 - 2.3944042171351925706\times10^{-8}\,x^{11}\end{aligned} \tag{6}$$

对余弦函数进行拟合之后得到的多项式逼近（4 到 10 阶）为：

$$\cos x = 0.9995795170399418561 - 0.49639233850557584748\,x^2 + 0.037209332130166132557\,x^4 \tag{7}$$

$$\cos x = 0.99999528272039384102 - 0.49993092003443151405\,x^2 + 0.041511736836071483348\,x^4 - 0.0012787139913533120562\,x^6 \tag{8}$$

$$\cos x = 0.99999996727291751153 - 0.49999926901995966899\,x^2 + 0.041664091445612914943\,x^4 - 0.0013857422438193591537\,x^6 + 2.3237659272108738645\times 10^{-5}\,x^8 \tag{9}$$

$$\cos x = 0.99999999985553855719 - 0.49999999531049355017\,x^2 + 0.041666642521451954795\,x^4 - 0.0013888439838802731086\,x^6 + 2.4764124863984253\times 10^{-5}\,x^8 - 2.6120941216892450088\times 10^{-7}\,x^{10} \tag{10}$$

多项式拟合的结果与 Taylor 逼近近似的误差统计如下表：

阶次	误差					
	多项式拟合误差			Taylor 截断误差		
	最大误差	最小误差	误差估计	最大误差	最小误差	误差估计
4	0.0005	-0.001	0.0003	1.1e-16	-0.019	0.0055
5	6.6e-05	-0.00016	4.1e-05	5.2e-15	-0.0045	0.0011
6	1.7e-05	-6.9e-06	4.1e-06	0.00089	-4.4e-16	0.00021
7	1.5e-06	-6.4e-07	3.6e-07	0.00015	-2.3e-13	3.6e-05
8	5.3e-08	-1.3e-07	2.8e-08	7.7e-14	-2.4e-05	5.4e-06
9	4.0e-09	-9.9e-09	2.0e-09	2.1e-14	-3.5e-06	7.3e-07
10	7.0e-10	-2.8e-10	1.3e-10	4.6e-07	-1.5e-15	9.3e-08
11	7.9e-11	-3.1e-11	1.2e-11	5.6e-08	-1.5e-12	1.0e-08

其中误差估计方法采用无偏估计：

$$\delta_{err} = \sqrt{\frac{\sum_{i=0}^{N-1}[y_i - f(x_i)]^2}{N-1}} \tag{11}$$

拟合的结果经过分析后发现，正弦函数只需要 4 项，余弦函数只需要 5 项，即可达到 10^{-6} 的精度，相较之 Taylor 展开，拟合算法减少了一次乘法运算的时间。 图1到图8使用相同阶次的泰勒截断近似和多项式拟合误差对比。

0.2 再次优化

熟悉 C/C++ 语言的程序员都知道乘法是一种非常昂贵的操作，能够避免时候尽量避免； 考虑一个普通的四阶多项式，如式(9)的变形，对它进行求值的时候，最少需要多少次乘法运算呢？

由于多项式都是具有这样的形式:

$$f_1(x) = a_0 + a_1 x \tag{12a}$$
$$f_2(x) = a_0 + a_1 x + a_2 x^2 \tag{12b}$$
$$f_3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{12c}$$
$$f_4(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \tag{12d}$$
$$f_5(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 \tag{12e}$$
$$f_6(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + a_6 x^6 \tag{12f}$$
$$\ldots \tag{12g}$$

显然对于式 (12b)，最少要一次乘法运算，对于式 (12c) 和式 (12d) 最少需要两次和三次乘法运算， 没有优化的余地。那么对于式 (12e)，是否必须四次乘法运算才能得到多项式的数值呢？

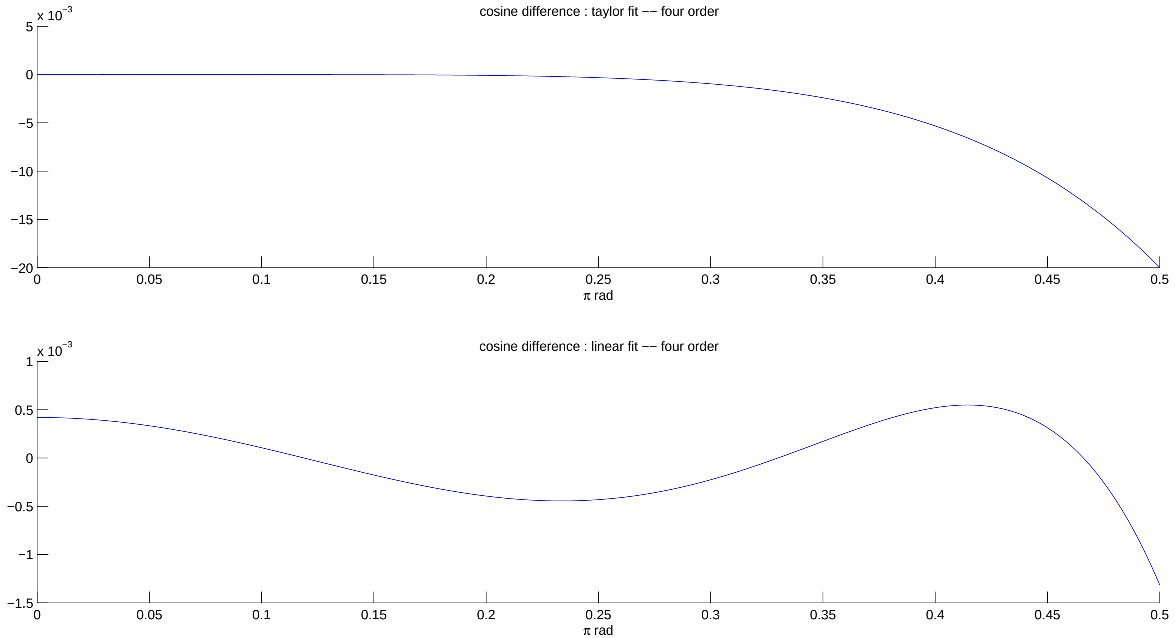

Figure 1: 4 阶多项式（下）拟合余弦函数与 4 阶泰勒截断（上）近似余弦函数之间的误差比较

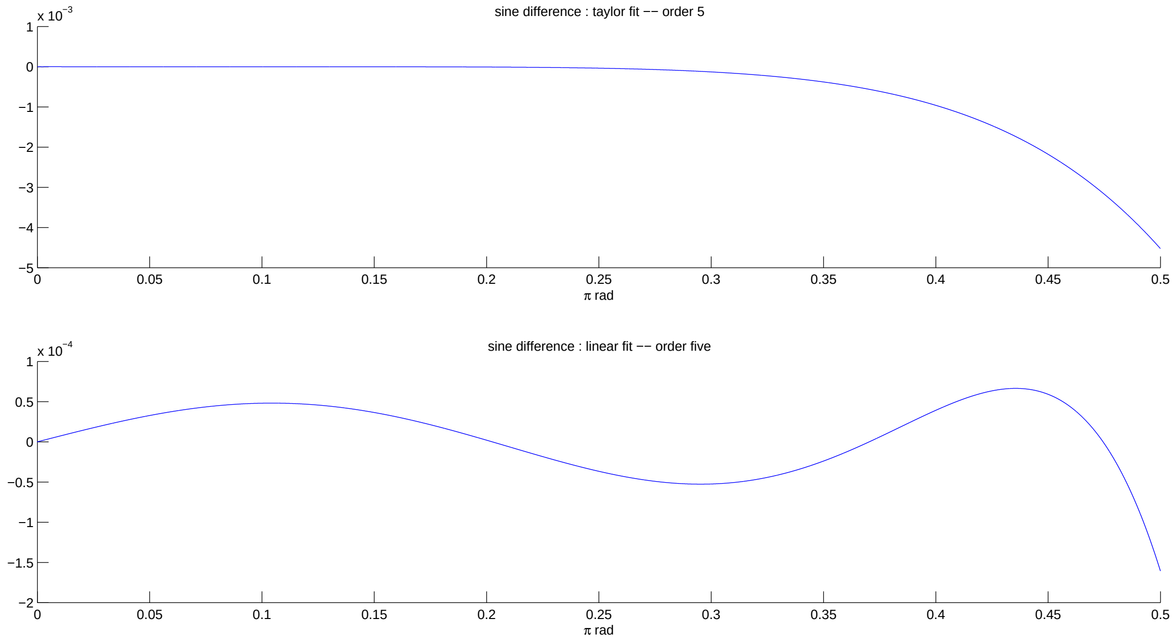

Figure 2: 5 阶多项式（下）拟合正弦函数与 5 阶泰勒截断（上）近似正弦函数之间的误差比较

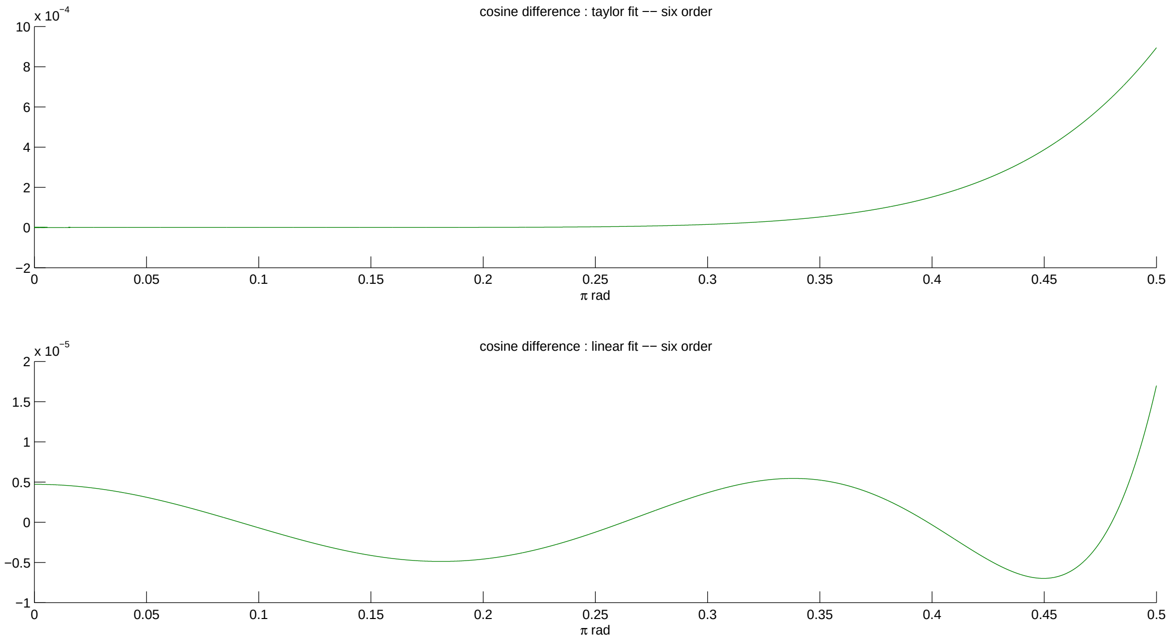

Figure 3: 6 阶多项式（下）拟合余弦函数与 6 阶泰勒截断（上）近似余弦函数之间的误差比较

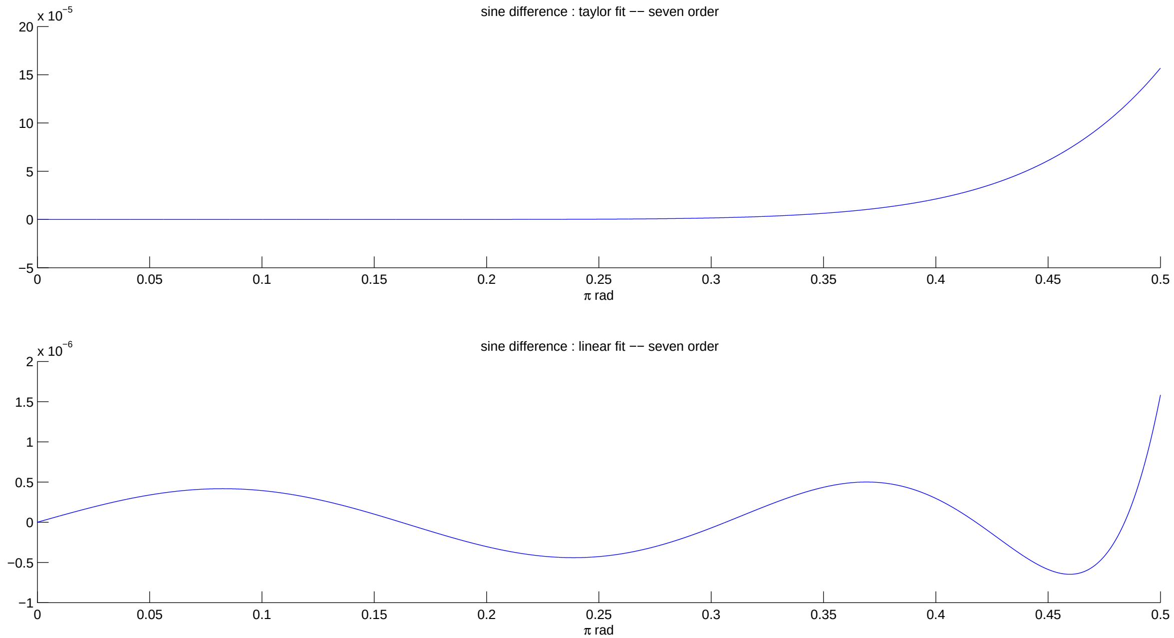

Figure 4: 7 阶多项式（下）拟合正弦函数与 7 阶泰勒截断（上）近似正弦函数之间的误差比较

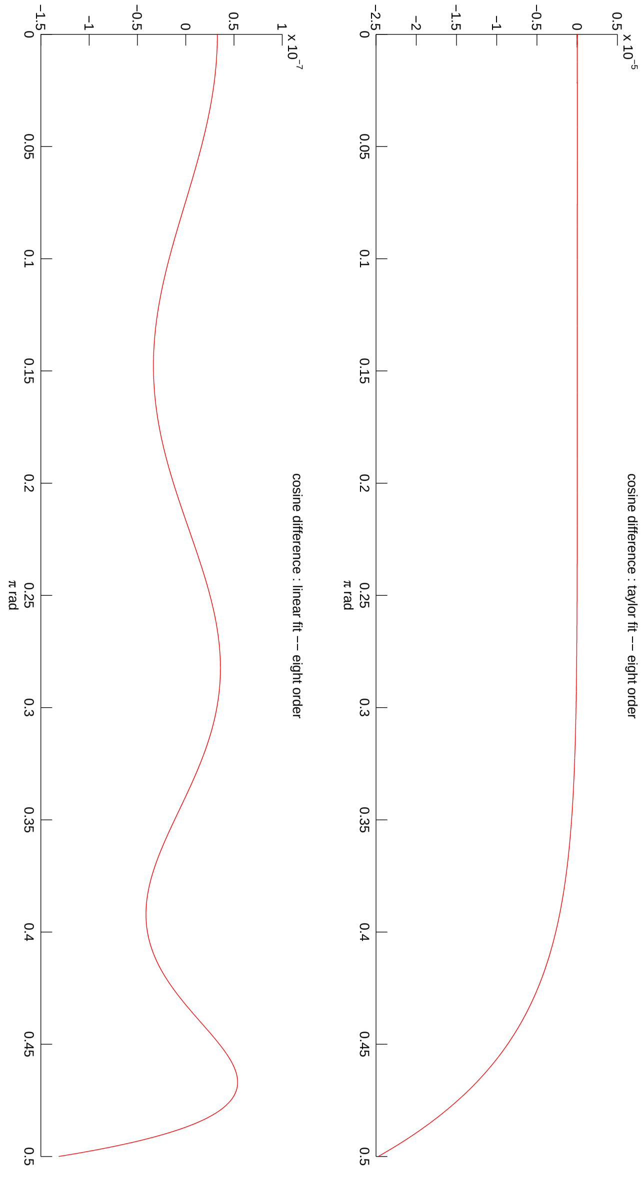

Figure 5: 8 阶多项式（下）拟合余弦函数与 8 阶泰勒截断（上）近似余弦函数之间的误差比较

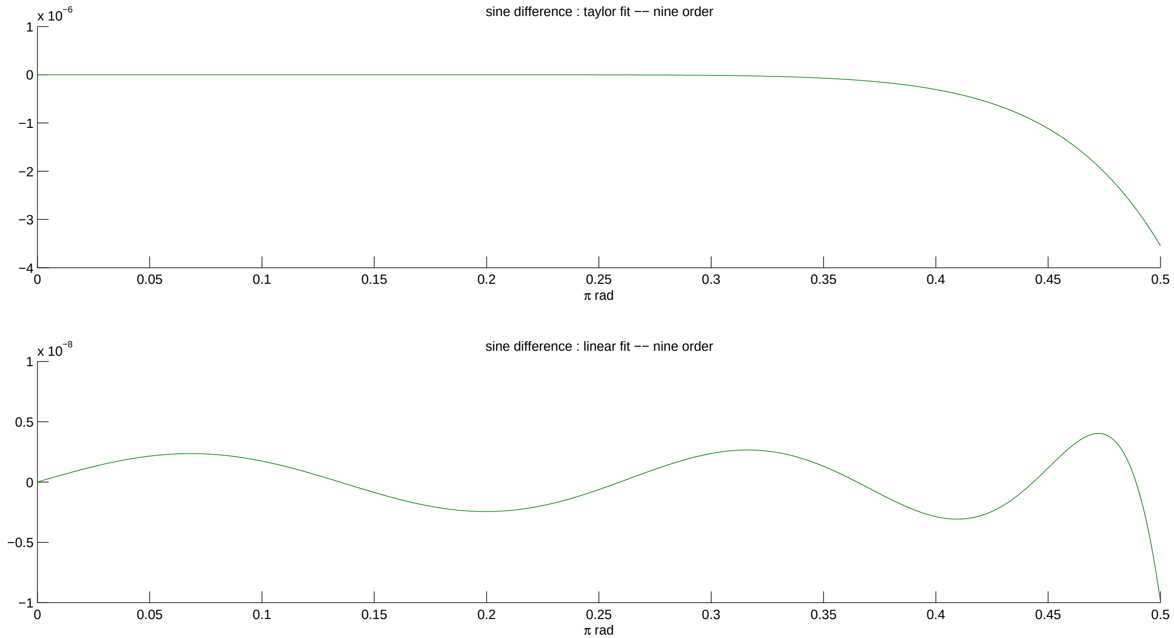

Figure 6: 9 阶多项式（下）拟合正弦函数与 9 阶泰勒截断（上）近似正弦函数之间的误差比较

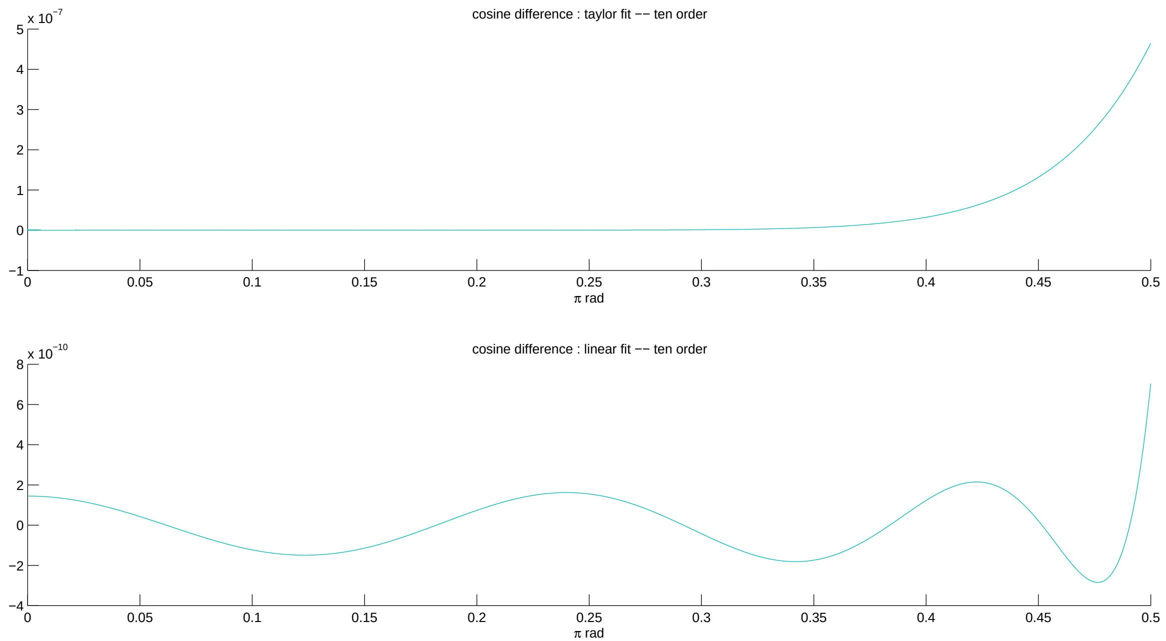

Figure 7: 10 阶多项式（下）拟合余弦函数与 10 阶泰勒截断（上）近似余弦函数之间的误差比较

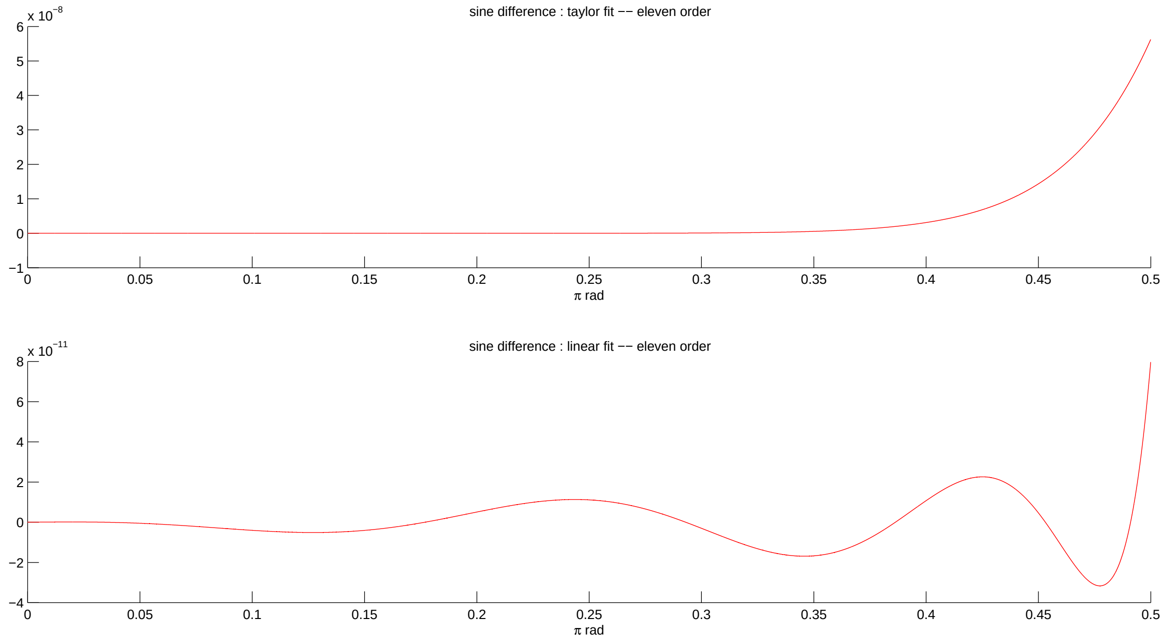

Figure 8: 11 阶多项式（下）拟合正弦函数与 11 阶泰勒截断（上）近似正弦函数之间的误差比较

如果式 (12e) 可以分拆成两个或多个多项式的乘积，并且这两个多项式有多项相同，那么就可以减少一次乘法计算， 考虑到该式有 $a[0]$ 到 $a[4]$ 总共 5 个系数，那么只要这些分拆出来的多项式中有 5 个待定的数就可以了。 一种可能的写法是这样的

$$f(x) = [(Ax+B)^2 + C][(Ax+B)^2 + D] + E \tag{13}$$

其中共有 A 到 E 5 个待定系数，而编程计算这个式子的的值只需要 3 次乘法运算。 但是，这个式子中第一个因式 $(Ax+B)^2 + C$ 与第二个因式 $(Ax+B)^2 + D$ 构成类型完全是对称的， C 和 D 轮转不变，因此无法利用式 (13) 展开后的系数与式 (12e) 的系数相等关系求得从 A 到 E 的表达式。

现在考虑对因式 $(Ax+B)^2 + C$ 或者后边的 $(Ax+B)^2 + D$ 添加一个别的项来破坏这种对称性，由于 C 和 D 是常数项， 而 x 的二次项只能添加 $(Ax+B)^2$ 的整数倍项， 这两种类型的项添加到因式中并不能破坏这种对称性，因此可供考虑的添加项只有 x 的整数倍和 Ax 的整数倍这两种； 于是可以立即得到添加项后的形式：

$$f(x) = [(Ax+B)^2 + \mathbf{m}\, x + C][(Ax+B)^2 + \mathbf{n}\, Ax] + E \tag{14}$$

其中 $\mathbf{m}$ 和 $\mathbf{n}$ 为自由指定的整数，只要不同时为 0 即可。 由于在程序编码中需要对这 x 和 Ax 的整数倍的计算手动展开为加法，为了效率起见， 一般可以令其中一个为 0，另外一个为 1。

现在假设选取的是 $\mathbf{m}$ 为 1, $\mathbf{n}$ 为 0，则有：

$$\begin{aligned} f(x) =& [(Ax+B)^2 + x + C][(Ax+B)^2 + D] + E \\ =& a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \end{aligned} \tag{15}$$

将x 的系数稍加比较，即可得到：

$$\begin{aligned} A =& a_4^{\frac{1}{4}} \\ B =& \frac{a_3 - A^2}{4A^3} \\ C + D =& \frac{a_2 - 6A^2B^2 - 2AB}{A^2} \\ D =& a_1 - 4AB^3 - 2AB(C+D) - B^2 \\ C =& C + D - D \\ E =& a_0 - (B^2 + C)(B^2 + D) \end{aligned} \tag{16}$$

其中式 (16) 要求 $a_4 > 0$，在实际处理中如果出现 $a_4 < 0$ 的情形，只要将 a_0 到 a_4 都取相反数， 分解完成后再取一次各项的相反数即可。

同样，如果 m 为 0，n 为 1，那么可以得到

$$\begin{aligned} A =& a_4^{\frac{1}{4}} \\ B =& \frac{a_3 - A^3}{4A^3} \\ D =& 3B^2 + 8B^3 + \frac{a_1 A + 2a_2 B}{A^2} \\ C =& \frac{a_2}{A^2} - 2B - 6B^2 - D \\ E =& a_0 - B^4 - B^2(C+D) - CD \end{aligned} \tag{17}$$

由此可知，式 (12f) 计算最少需要 4 次乘法，式 (12g) 也是 4 次。 图9展示了使用库函数和使用二次拟合之后的三角函数运行时间的对比。

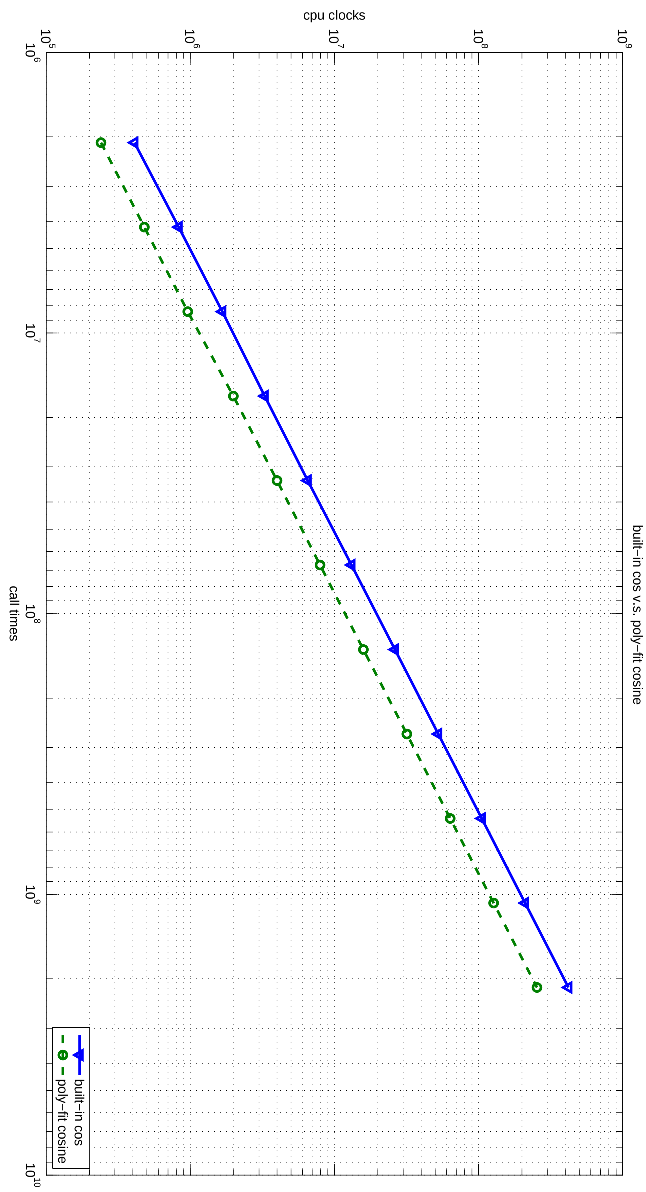

Figure 9: 二次优化之后，8 阶多项式计算余弦函数与库函数中的余弦函数在调用时占用 CPU 时间的对比