文章编号: 1671-1114(2014)04-0025-03

同符号数相加"大数吃小数"的界限: 理论分析

曹 靖[1,3], 李建平[1,2]

(1. 中国科学院 a. 大气物理研究所, b. 研究生院, 北京 100029; 2. 北京师范大学 全球变化与地球系统科学研究院, 北京 100875; 3. 天津理工大学 理学院, 天津 300384)

摘 要: 研究计算机内部二进制浮点数 IEEE754 存储规则及相加过程, 给出数值计算中两同号规范化数相加发生"大数吃小数"现象的严格理论界限, 为实际数值计算中避免此类现象提供理论依据, 并利用所得理论对数值试验中的现象及结论进行解释.

关键词: "大数吃小数"; 数值计算; IEEE754 标准; 规范化二进制浮点数

中图分类号: O246 **文献标志码**: A

On bound of a large number annihilating a small number in addition operation of two numbers with same sign: theoretical analysis

CAO Jing[1,3], LI Jianping[1,2]

(1a. Institute of Atmospheric Physics, 1b. Graduate University, Chinese Academy of Sciences, Beijing 100029, China;
2. Institute of Global Change and Earth System Science, Beijing Normal University, Beijing 100875, China;
3. College of Science, Tianjin University of Technology, Tianjin 300384, China)

Abstract: By analyzing about storage pattern and addition operation of format binary floating-points based on IEEE754 standard, this paper found strict theoretical bounds for avoiding the phenomenon that a large number annihilating a small number in numerical computations when both of these two numbers are positive or negative. The bounds can be applied in practical numerical computations for any machine precision and any programming language. Moreover, the conclusions in this article give theoretical explanations for the phenomenons in numerical experiments.

Keywords: a large number annihilating a small number; numerical computations; IEEE754 standard; format binary floating-points

"大数吃小数"是一种常见的影响数值计算精度的现象.文献[1]通过对同符号机器数相加的情况进行大量数值试验, 给出了避免"大数吃小数"现象发生的近似界限.但目前尚未见关于"大数吃小数"现象严格理论界限的报道.众所周知, "大数吃小数"是由于计算机的字长有限, 两阶码不同的二进制机器数对位相加所引起的[2]. 同时, 文献[1]中的数值试验结果也表明, "大数吃小数"的界限应与机器二进制数的有效数字有关. 因此, 本研究通过对计算机二进制浮点数相加运算过程进行分析, 对于两同号数相加的情况, 给出"大数吃小数"现象的严格理论界限, 从而为实际数值计算中避免此类现象发生提供理论依据.

1 同符号数相加发生"大数吃小数"的严格理论界限

本节对于两同号机器数 a 与 b 的相加运算, 寻找机器数 c_a, 若 $|b| \leqslant |c_a|$, 则 b 会被 a "吃掉", 而若 $|b| > |c_a|$, 则不会被"吃掉". c_a 即为可被机器数 a "吃掉"的最大"小数", 可作为 a 与 b 相加运算发生"大数吃小数"的严格界限.

1.1 机器单精度的情况

首先, 考虑两符号为正的浮点数相加的情形. 由 IEEE754 标准中机器单精度下规范化二进制浮点机器

收稿日期: 2013-12-15
基金项目: 国家自然科学基金资助项目(41375110); 中国科学院大气物理研究所大气科学和地球流体力学数值模拟国家重点实验室(LASG)2012年度开放课题
第一作者: 曹 靖(1981—), 女, 讲师, 主要从事计算数学方面的研究.
通信作者: 李建平(1969—), 男, 研究员, 主要从事气候动力学、数值模拟与计算等方面的研究.

数的存储格式[3-5]，可将相加运算中的“大数”a 表示为 $s_a\times 2^{e_a}$，其中：$s_a=(1.a_1a_2\cdots a_{23})_2$ 为具有 24 位有效数字的二进制数，$e_a\in\mathbf{Z}$. 在计算机中，$a_1a_2\cdots a_{23}$ 与 e_a 分别作为 a 的尾码与阶码进行存储. 同时，将能被 a“吃掉”的任一正规范化浮点数设为 $c=s_c\times 2^{e_c}=(1.c_1c_2\cdots c_{23})_2\times 2^{e_c}$，并将其中的最大值记为 c_a，为可被 a“吃掉”的最大“小数”.

下面，利用 IEEE754 规则下二进制浮点数的对位加法过程的 8 个步骤[6]，给出计算机运算中 a 与 c 的相加过程，并分析 c 及 c_a 的取值. 因 a 与 c 均为正数，其和 $a+c$ 也为正数，所以运算过程的第 8 步计算符号的步骤可以省去.

1) 比较：因 c 被 a“吃掉”，所以必有 $e_a>e_c$，阶码不发生交换.

2) a 与 c 符号相同，不进行此步骤.

3) 对位：将 s_c 存入一个 24 位寄存器，并将 s_c 向右移动 $e=e_a-e_c$ 位，左面移出的位补 0，s_c 移位后在寄存器中的存储格式如表 1 所示.

表 1 s_c 移位后在寄存器中的存储格式

Tab. 1 Storage format in register while s_c shifted

位数	1 至 e	e+1	e+2 至 e+24
移位后的 s_c	0	1	$c_1\cdots c_{23}$

4) 相加：将 s_a 与移位后的 s_c 进行对位加法. 由于 $a+c=a$，因此对位相加过程中 a 与 c 的有效数字 s_a 与 s_c 之间不能发生加法运算，对位相加结果更不能发生进位，否则不论后面的步骤如何舍入，a 在加 c 以后必然改变其值. 也就是说，s_c 必须向右完全移出 s_a 的有效数字范围，即 $e\geqslant 24$. 表 2 给出 e=24 时 s_a 与 s_c 对位相加过程及相加结果 S，S 的有效数字位为 1~24. 另外，此步骤中需计算的参数 g、r、s 的值可分为以下几种情况：

当 $e=24$ 时，$g=1$，$r=c_1$，$s=c_2\vee c_3\vee\cdots\vee c_{23}$；

当 $e=25$ 时，$g=0$，$r=1$，$s=c_1\vee c_3\vee\cdots\vee c_{23}$；

当 $e=26$ 时，$g=0$，$r=0$，$s=1\vee c_1\vee c_3\vee\cdots\vee c_{23}=1$；

当 $e>26$ 时，$g=0$，$r=0$，$s=0\vee 1\vee c_1\vee c_3\vee\cdots\vee c_{23}=1$.

表 2 $e=24$ 时 s_a 与 s_c 对位相加表

Tab. 2 Adding s_a to s_c while $e=24$

位数	1	2 至 24	25	26 至 48
s_a	1	$a_1\cdots a_{23}$		
移位后的 s_c	0	0	1	$c_1\cdots c_{23}$
相加结果 S	1	$a_1\cdots a_{23}$	1	$c_1\cdots c_{23}$

5) 正规化：由于不涉及进位，因此无需对 S 进行移位操作.

6) 调整 r 和 s：第 4 步中不存在移位，r 和 s 根据 e 取值的不同情况进行调整：

当 $e=24$ 时，$r=g=1$，$s=r\vee s=c_1\vee c_2\vee c_3\vee\cdots\vee c_{23}$；

当 $e=25$ 时，$r=g=0$，$s=r\vee s=1\vee c_1\vee c_3\vee\cdots\vee c_{23}=1$；

当 $e>25$ 时，$r=g=0$，$s=r\vee s=1$.

7) 舍入：因为 $a+c=a$，所以 S 的舍入结果须为 s_a，即 S 的最低位保持不变(即舍)，因此需要 $(r\wedge p_0)\vee(r\wedge s)=0$，其中：$p_0$ 为 S 有效数字部分的最低位，它应为 a 的尾码最后一位 a_{23}. 此时 c 及 c_a 的取值可根据 p_0 的值分为 2 种情况讨论.

当 $p_0=a_{23}=0$ 时. $r\wedge p_0=0$，若要 $(r\wedge p_0)\vee(r\wedge s)=0$，应有 $r\wedge s=0$. 当 $e=24$ 时，$r=1$，因此，只有 $s=0$，才能保证 $r\wedge s=0$，而此时 $c=(1.00\cdots0)_2\times 2^{e_a-24}$ 为可被 a“吃掉”的机器数；当 $e>24$ 时，$r=0$，$s=1$，则 $r\wedge s=0$ 必成立，此时无论 c 取何值，均会被 a“吃掉”. 由此可得，此时可被 a“吃掉”的最大“小数”$c_a=(1.00\cdots0)_2\times 2^{e_a-24}$.

当 $p_0=a_{23}=1$ 时. 当 $e=24$ 时，$r=1$，则 $(r\wedge p_0)\vee(r\wedge s)=1$，发生入位，$S$ 的最低位发生变化，此时不符合 c 取值的要求；当 $e>24$ 时，$r=0$，$s=1$，则必有 $(r\wedge p_0)\vee(r\wedge s)=0$，则无论 c 取何值，均被 a“吃掉”，显然，此时 $c_a=(1.11\cdots1)_2\times 2^{e_a-25}=2^{e_a-24}-2^{e_a-48}$.

由以上相加过程可得能被“大数”a“吃掉”的最大“小数”c_a 的取值：当 $a_{23}=0$ 时，$c_a=2^{e_a-24}$，当 $a_{23}=1$ 时，$c_a=2^{e_a-24}-2^{e_a-48}$. 这说明 c_a 与 a 尾码的最后一位 a_{23} 的值以及二进制阶码 e_a 有关.

另外，当 a、c 同为负规范化机器数时，其相加过程与同为正数相比，只是第 8 步将正号变为负号，其余步骤均相同. 因此，只需将 a、c 同为正数时得到的 c_a 值取相反数即可. 综合正、负 2 种情况，可得机器单精度下同号规范化机器数相加运算“大数吃小数”的严格界限：

定理 1 设 a 与 b 为机器单精度下同号二进制规范化浮点数，则当 $|b|\leqslant c_a$ 时，机器运算结果为 $a+b=a$，否则 $a+b\neq a$，其中：

$$c_a=\begin{cases}2^{e_a-24} & a_{\text{low}}=0\\ 2^{e_a-24}-2^{e_a-48} & a_{\text{low}}=1\end{cases}\tag{1}$$

$e_a=\text{floor}(\log_2|a|)$，floor 表示向负方向取整，$a_{\text{low}}$ 为机器数 a 尾码的最低位.

1.2 任意机器精度的情况

将上节中两同号机器数对位相加过程由机器单精度推广至具有任意 n 位二进制有效数字的情况，可得“大数吃小数”的严格界限：

定理 2 设 a 与 b 为具有 n 位二进制有效数字的同号规范化浮点数，则当 $|b|\leqslant c_{a,n}$ 时，机器运算结果为 $a+b=a$，否则 $a+b\neq a$，其中：

$$c_{a,n}=\begin{cases}2^{e_a-n} & a_{\text{low}}=0\\ 2^{e_a-n}-2^{e_a-2n} & a_{\text{low}}=1\end{cases}\tag{2}$$

$e_a=\mathrm{floor}(\log_2|a|)$，$a_{\mathrm{low}}$为机器数$a$尾码的最低位.

2　数值试验现象的理论解释

本节由定理 1 和定理 2 对文献[1]中的数值试验现象进行理论解释.

首先，由定理 1 和定理 2 可见，对于具有 n 位二进制有效数字的规范化机器数 a，只要其阶码 e_a 固定，则根据其尾码最低位 a_{low}，$c_{a,n}$ 的取值只有 2^{e_a-n} 与 $2^{e_a-n}-2^{e_a-2n}$ 这 2 种选择.实际上，若 e_a 固定，在 $|s_a|$ 的取值由 $(1.00\cdots0)_2$ 增大至 $(1.11\cdots1)_2$ 的过程中，$|s_a|$ 的最低位不断在‘0’和‘1’之间交替变化，则在此过程中 $c_{a,n}$ 的取值也在 2^{e_a-n} 和 $2^{e_a-n}-2^{e_a-2n}$ 之间交替变化. 并且，2^{e_a-n} 与 $2^{e_a-n}-2^{e_a-2n}$ 是相邻的机器数，它们的差距非常小. 也就是说，当 $|a|\in[2^{e_a},2^{e_a+1})$ 时，能够被 a “吃掉”的最大“小数”是十分接近的. 另外，当 a 的阶码发生变化时，如由 e_a 变为 e_a+1 时，$c_{a,n}$ 的取值将随之翻倍. 这就解释了文献[1] 中“大数”A 和最大“小数” C_A 的关系.

其次，定理 1 和定理 2 的结论也可解释文献[1]中 $\lg(c_{a,n}/|a|)$关于 $\lg|a|$的分段线性关系，以及由此得到的“大数吃小数”的近似界限公式(结论 1~结论3). 由上述分析，当 e_a 固定时，$c_{a,n}$ 的取值在 2^{e_a-n} 和 $2^{e_a-n}-2^{e_a-2n}$ 之间变化，这 2 个值十分接近，在数值试验中很难区分，因此文献[1]得到了 $c_{a,n}$ 的一个统一的近似计算公式. 取此 2 值中的任意一个，如 $c_{a,n}=2^{e_a-n}$，便可解释文献[1]的数值试验结果. 容易计算得

$$\lg(c_{a,n}/|a|)=\lg\left(\frac{2^{e_a-n}}{|s_a|\cdot2^{e_a}}\right)=-\lg|s_a|-n\lg2 \quad (3)$$

下面以式(3)为基础，对文献[1]的数值试验结果从以下 3 个方面进行解释：

1) $\lg|a|$分段区间长度的验证：由式(3)可见，$\lg(c_{a,n}/|a|)$的值与 e_a 无关，说明它是一个关于 $\lg|a|$ 的分段周期函数，区间段为$[\lg2^{e_a},\lg2^{e_a+1})(e_a\in\mathbf{Z})$，每个区间段长度均为

$$\lg2^{e_a+1}-\lg2^{e_a}=\lg2\approx0.301$$

2) 分段区间内 $\lg(c_{a,n}/|a|)$函数值变化的验证：由式(3)，在每个区间段$[\lg2^{e_a},\lg2^{e_a+1})$内，当 $|s_a|$的取值由 $(1.00\cdots0)_2$ 增大至 $(1.11\cdots1)_2$ 时，可得 $\lg(c_{a,n}/|a|)$在$[-n\lg2,-(n+1)\lg2)$内单调递减. 如，在机器双精度下，由 $n=53$[5] 可计算出 $\lg(c_{a,n}/|a|)$应在$[-15.955,-16.256)$内单调递减，这解释了文献[1]表 1 中区间顶点纵坐标的试验结果.

3) 分段区间内 $\lg(c_{a,n}/|a|)$与 $\lg|a|$线性关系的验证：因为

$$\lg|a|=\lg(|s_a|\cdot2^{e_a})=\lg|s_a|+e_a\lg2$$

结合式(3)得

$$\lg(c_{a,n}/|a|)=-\lg|a|+e_a\lg2-n\lg2 \quad (4)$$

由上述分析，式(4)中的 $e_a\lg2$ 应为 a 所在区间左顶点横坐标，因此，式(4)可变为

$$\lg(c_{a,n}/|a|)=-(\lg|a|-0.301k)-n\lg2$$
$$k=\mathrm{floor}(\lg|a|/0.301) \quad (5)$$

由式(5)可见，在任一区间$[\lg2^{e_a},\lg2^{e_a+1})$内，$\lg(c_{a,n}/|a|)$为关于 $\lg|a|$斜率为-1 的线性函数，这可解释文献[1]图 1 中 $\lg(C_A/A)$关于 $\lg A$ 的分段线性关系，以及表 1 的拟合斜率. 从而进一步解释了文献[1]的近似计算公式.

3　结论

通过对计算机内部规范化二进制浮点数的存储以及相加过程进行研究，为两同号数相加的“大数吃小数”现象给出了严格的理论界限，结论适用于任意机器精度及计算机语言，可为实际数值计算中避免出现此类现象提供理论依据. 但是，本研究只讨论了两同号数相加的情况，对于异号数相加和非规范化数的情况，由于相加过程非常复杂，尚未得到结论，今后希望将研究推广至这 2 种情况，得到更为完善的理论体系.

参考文献：

[1] 曹靖，李建平. 同符号数相加“大数吃小数”的界限：数值试验[J]. 天津师范大学学报：自然科学版，2014，34(2)：16-18.

[2] 冯有前. 数值分析[M]. 北京：清华大学出版社，2005.

[3] Technical committee on microprocessors and minicomputers. IEEE standard for binary floating-point arithmetic [S]. The Institute of Electrical and Electronics Engineers, Inc, 1985.

[4] MULLER J M, BARRE N B, DINECHIN F D, et al. Handbook of floating-point arithmetic[M]. New York: Springer, 2009.

[5] 朱亚超. 基于 IEEE754 的浮点数存储格式分析研究[J]. 计算机与信息技术，2006，9(9)：50-52.

[6] GOLDBERG D. Computer arithmetic[M]. New York: Elsevier Scicnce, 2003.

（责任编校　马新光）