# 初等函数可验证赋值及误差分析

刘 剑[1] 唐 敏[1,2] 曾 霞[1] 曾振柄[1]
(华东师范大学上海高可信计算重点实验室 上海 200062)[1]
(桂林电子科技大学数学与计算机学院 桂林 541004)[2]

**摘 要** 研究了GNU标准下初等函数的赋值原理及算法实现。基于IEEE 754-2008浮点标准,利用误差分析基本结论,对GNU下C语言标准数学函数库中的初等函数赋值程序进行理论误差分析。利用Boost库中提供的区间类,将以浮点数作为基本数据类型的程序重写成以区间作为基本类型的程序,使用区间算术对初等函数进行可验证赋值,从而得到一个包含真实值的区间包络,并由此给出GNU下初等函数的数值误差界。

**关键词** 初等函数,误差分析,区间算术,可验证赋值
**中图法分类号** TP30 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.4.003

## Validated Evaluation and Error Analysis of Elementary Functions

LIU Jian[1] TANG Min[1,2] ZENG Xia[1] ZENG Zhen-bing[1]
(Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai 200062, China)[1]
(School of Mathematics & Computing Science, Guilin University of Electrical Technology, Guilin 541004, China)[2]

**Abstract** This paper discussed evaluation principle and implementation of elementary functions according to the standard of GNU. Based on the IEEE 754-2008 standard, theoretical error on elementary functions program in C language standard library was analyzed by error analysis fundamental theory. Firstly, the floating-point C program was transferred to the corresponding interval computing one using interval class provided by Boost library. Secondly, validated evaluation was used on these elementary functions by interval arithmetic. Finally, the interval, including real numeric, was obtained. Based on the interval, numerical error bounds on these elementary functions were presented.

**Keywords** Elementary functions, Error analysis, Interval arithmetic, Validated evaluation

## 1 引言

在科学研究和工程应用中,通常需要对大量的数据进行存储和计算。然而,实数在计算机中只能以二进制浮点数的形式进行存储,从而会产生舍入误差。除此之外,用一个基本表达式逼近一个相当复杂的算术表达式时会产生截断误差;在连续计算的过程中会导致误差累积。因此,如何刻画数值算法的精确性和稳定性始终是数值领域的研究热点。目前为止,对数值计算的研究主要集中在有限步算术运算的有穷序列的数值算法领域[1-4],如矩阵计算、方程组求解等,而涉及初等函数问题的数值算法非常少。

众所周知,初等函数(如三角函数、指数函数、对数函数等)经常应用于很多科学问题的数学模型中,在数值计算领域更是使用普遍。在很多性能要求严格的科研领域,其主要是用C语言编写的,尤其在单片机和嵌入式系统开发中。所以对C语言数学函数库中提供的初等函数赋值运算进行误差分析是值得研究的课题。

一般来说,传统的数值法使用单一的浮点数作为赋值运算结果的近似值,由于数值计算缺乏绝对的稳定性,在某些情形下误差非常大。相比之下,区间算术(即可验证方法)能获得包含问题真实解的区间,是一种误差可控的可信算法。关于数值计算结果的可验证赋值是近年来数值计算领域研究的热点[5]。目前关于初等函数的误差分析及可验证赋值的主要工作有GMP(The GNU Multiple Precision Arithmetic Library)和MPFR(The Multiple Precision Floating-Point Reliable Library)中针对多精度的特殊函数和初等函数[6-8],以及Antwerp大学的研究者和NIST(National Institute of Standards and Technology)合作正在开发的可验证的、多精度的、独立于基数的特殊(和初等)函数库[9]。

初等函数是一类特殊的函数,它们本身并不能通过有限次数的加减乘除运算求得准确结果,通常需要用无穷级数或者特殊的有理函数来表示,并取前 $N$ 项和作为近似值,由此将会产生截断误差和舍入误差。本文以GNU下初等函数的实现为研究对象,利用误差分析基本结论,对C语言标准数学函数库中的初等函数赋值程序进行算法研究及误差分析,给出理论误差界。此外,借助成熟的区间算术软件包,重写浮

到稿日期:2014-06-12 返修日期:2014-08-24 本文受国家自然科学基金(91118007),上海市教育委员会创新基金(14ZZ046)资助。
刘 剑(1989-),男,硕士生,主要研究方向为符号计算、形式化方法,E-mail:ljian-1989@163.com;唐 敏(1980-),女,博士生,主要研究方向为符号计算、形式化方法;曾 霞(1987-),女,博士生,主要研究方向为符号计算、形式化方法;曾振柄(1963-),男,博士,教授,主要研究方向为符号计算、形式化方法、数学机械化、人工智能软件设计、生物信息处理技术。

点计算程序，为初等函数提供可靠的可验证赋值，即得到一个包含真实值的区间包络，由此给出 GNU 下初等函数的数值误差界。

本文第 2 节介绍浮点计算误差分析基本结论和区间算术基本概念；第 3 节针对 GNU 下的初等函数进行理论误差分析，得到理论误差界；第 4 节给出利用区间算术进行可验证赋值及误差分析的结果；第 5 节的数值实验给出 4 个初等函数的误差界变化趋势。

## 2 预备知识

无论要计算的表达式有多么复杂，利用计算机进行赋值时，最终都要通过基本算术运算即加、减、乘、除来实现。对于矩阵计算、多项式赋值、传统的方程组求解等，可由有限步的算术运算得到，在这种情况下会出现舍入误差的积累；而对于基本初等函数（如正弦函数、指数函数等），以及特殊函数（如误差函数等），不能由有限步算术运算得到，对它们的赋值首先要选择合适的数学模型，比如级数或连分数形式，进行适当的截断后，再利用有限步的基本算术运算求出舍去余项的部分和。在这种情况下，不仅产生了舍入误差，还会产生截断误差。

### 2.1 误差分析基本结论

#### 2.1.1 基本算术运算舍入误差

在 IEEE 754-2008 标准下，浮点数通过 4 个参数表示：基 $\beta$、精度 $p$、指数域 $[L,U]$[10]。一个实数 $x$ 采用就近舍入模式存储到计算机中，记为 $fl(x)$，或简记为 $\tilde{x}$。

在基 $\beta$、精度 $p$ 下，两个浮点数 $x$ 和 $y$ 进行基本算术运算（加减乘除）的结果用 $fl(x+y)$、$fl(x-y)$、$fl(x\times y)$、$fl(x/y)$ 来表示，或统一用 $fl(x\circ y)$ 来表示，其中 $\circ\in\{+,-,\times,/\}$，运算结果的相对误差界是

$$u(p):=\frac{1}{2}\beta^{-p+1} \tag{1}$$

也就是说，

$$fl(x\circ y)=(x\circ y)(1+\delta)$$

其中，$\circ\in\{+,-,\times,/\}$，$|\delta|\leqslant u(p)$。

一个表达式或运算序列由若干个基本算术运算构成，对其进行误差分析的基本目标如 2.1.2 节所述，而进行误差分析需要的重要理论成果将在 2.1.3 节中给出。

#### 2.1.2 误差分析基本目标

设表达式 $Y$ 由基本算术运算及其混合运算构成，在基 $\beta$、精度 $p$ 下，计算得到的浮点数值为 $\tilde{Y}$，满足

$$\tilde{Y}=Y(1+\theta_{n(Y)})\quad |\theta_{n(Y)}|\leqslant\frac{n(Y)u(p)}{(1-n(Y)u(p))}=:\gamma(n(Y),p) \tag{2}$$

其中，$n(Y)$ 是与 $Y$ 的表达式有关的一个整数。

在对一个特定的 $Y$ 进行误差分析时，希望得到一个较紧的相对误差界 $|\theta_{n(Y)}|$，一旦确定了 $n(Y)$，即可通过计算式(2)得到一个理论误差界。

#### 2.1.3 误差分析基本结论

对于一个运算序列而言，计算相对误差，必须追踪所有的误差项 $(1+\delta)$。以下给出误差分析基本结论，利用这些结论，可以得到一个运算序列的误差界。也就是说，可以从理论上得到 $Y$ 的相对误差 $|\theta_{n(Y)}|$。

**定理 1** 如果对于所有的 $\delta_i,\rho_i$ 有 $|\delta_i|\leqslant u(p),\rho_i=\pm1$，并且 $nu(p)<1$，则

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i}=1+\theta_n \tag{3}$$

其中，$|\theta_n|\leqslant\frac{nu(p)}{1-nu(p)}=:\gamma(n,p)$。

利用数学归纳法可对定理 1 进行证明。

**定理 2** 如果对于所有的 $\delta_i,\rho_i$ 有 $p_i=+1$，$|\delta_i|\leqslant u(p)$ 并且 $nu(p)<1$，则能推导出一个更紧的界：

$$\prod_{i=1}^{n}(1+\delta_i)=1+\theta_n \tag{4}$$

其中，$|\theta_n|\leqslant\frac{nu(p)}{1-nu(p)/2}$。

利用一个特殊的不等式及级数展开可对定理 2 进行证明。

**定理 3** 如果是以误差因子为 $(1+\theta_i)$ 的数学量进行乘除或加减运算，则有如下结论：

$$(1+\theta_k)(1+\theta_l)=1+\theta_{k+l} \tag{5}$$

$$\frac{1+\theta_k}{1+\theta_l}=\begin{cases}1+\theta_{k+l}, & l\leqslant k\\ 1+\theta_{k+2l}, & l>k\end{cases} \tag{6}$$

当 $x,y\geqslant0$ 时，

$$x(1+\theta_k)+y(1+\theta_l)=(x+y)(1+\theta_{\max(k,l)}) \tag{7}$$

$$x(1+\theta_k)-y(1+\theta_l)=(x-y)(1+\theta_{j\max(k,l)}) \tag{8}$$

其中，$j\geqslant\frac{|x|+|y|}{|x-y|}$，$j\in N$ 并且 $x,y\geqslant0$。

定理 3 的证明要利用不等式的缩放。定理 1 至定理 3 的详细证明及应用可参考文献[11]。

误差分析基本结论既能用于理论误差的确定，又能用于对误差的控制。具体来说，对于给定的一个表达式，在基 $\beta$、精度 $p$ 下，利用式(1)－式(8)分析得出理论误差界，当然这个误差界可能比实际的误差要大；如果要控制给定的一个表达式的相对误差，那么除了上述结论，还需要子表达式误差分配的相关结论[9]。

例如，$x,y,z,w$ 是不能精确表示成浮点数的实数，在基 $\beta$、精度 $p$ 下，计算表达式 $Y=(x+y)z^2/(1.25w)$，则

$$\tilde{Y}=\frac{(x(1+\delta_1)+y(1+\delta_2))z(1+\delta_3)z(1+\delta_3)}{(1.25\times w(1+\delta_4))(1+\delta_5)}=Y(1+\theta_5)$$

其中，$|\delta_i|\leqslant u(p),i=1,\cdots,5$，$|\theta_5|\leqslant\gamma(5,p)=\frac{5u(p)}{1-5u(p)}$。

利用误差分析基本结论，可推导出 $Y$ 的理论相对误差界为 $\theta_5$，在 $\beta=2$、$p=53$ 下，计算表达式 $Y$ 产生的相对误差小于等于 $5.56\times10^{-16}$。

### 2.2 区间算术

区间算术是集合上的运算，将初始给定的数据及其运算的所有可能性包含在计算的结果里，利用它可对任意一个表达式进行可验证赋值（即给出区间包络），再利用这个区间包络，可给出一个通过数值计算得到的误差界。对于区间及其基本算术运算的定义如 2.2.1 节及 2.2.2 节所述。

#### 2.2.1 区间基本概念

区间是一个集合，通常用一对实数表示，即

$$[a,b]=\{x\in R:a\leqslant x\leqslant b\}$$

习惯上，用大写字母表示区间和其端点。即

$$X=[\underline{X},\overline{X}]$$

其中，$\underline{X}$ 是区间 $X$ 的下界，$\overline{X}$ 是区间 $X$ 的上界。

区间 $X$ 的宽度定义为 $w(X)=\overline{X}-\underline{X}$，区间中点定义为 $mid(X)=(\overline{X}+\underline{X})/2$。

2.2.2 区间算术基本运算

区间的加减乘除运算是集合运算，结果是所有属于这两个区间的元素做相应运算时构成的集合[12]。即

区间加法：$X+Y=[\underline{X}+\underline{Y},\overline{X}+\overline{Y}]$。

区间减法：$X-Y=[\underline{X}-\overline{Y},\overline{X}-\underline{Y}]$。

区间乘法：$X\cdot Y=[\min(S),\max(S)]$，其中 $S=\{\underline{X}\underline{Y},\underline{X}\overline{Y},\overline{X}\underline{Y},\overline{X}\overline{Y}\}$。

区间除法：$X/Y=X\cdot(1/Y)$，其中 $1/Y=[1/\overline{Y},1/\underline{Y}]$。

## 3 初等函数理论误差分析

本节对 GNU 下的初等函数（包括正弦函数、余弦函数、指数函数、对数函数）的实现过程进行分析（源代码来源于文献[13]），从理论上得到一个相对误差界 $\Delta$。根据这个误差界，可对初等函数 $f(x)$ 进行可验证赋值，得到：

$$[\frac{fl(f(x))}{1+\Delta},\frac{fl(f(x))}{1-\Delta}],f(x)>0$$

或

$$[\frac{fl(f(x))}{1-\Delta},\frac{fl(f(x))}{1+\Delta}],f(x)<0 \tag{9}$$

保证式(9)是包含真实值的区间包络。

### 3.1 三角函数理论误差分析

GNU 下，正弦函数 $\sin(x)$ 和余项函数 $\cos(x)$ 的实现是通过级数展开式来近似求值。不失一般性，函数 $f(z)$ 的幂级数展开式为：

$$f(z)=\sum_{i=0}^{\infty}a_iz^i \tag{10}$$

采用式(10)次数小于等于 $N$ 的前 $N$ 项部分和 $S_N(z)$ 近似计算 $f(z)$ 时，会产生截断误差 $R_n(z)=|f(z)-T_N(z)|$，如果 $f(z)$ 是收敛的幂级数，则

$$R_n(z)=\sum_{i=N+1}^{\infty}a_iz^i\to 0,N\to\infty$$

$R_n(z)$ 的上界可由序列 $\{a_iz^i\}$ 进行分析，如果级数项正负交错，满足 $-a_iz/a_{i-1}$ 是正的并且递减，那么对任意奇数 $N$，

$$\sum_{i=N+1}^{\infty}a_iz^i\leqslant a_{N+1}z^{N+1}$$

此外，如果 $|q(z)|$ 是 $|f(z)|$ 的一个紧下界，那么截断产生的相对误差小于等于 $|\frac{f(z)-T_N(z)}{q(z)}|=|\frac{o(a_Nz^N)}{q(z)}|$。

给定 $f(z)$、$N$、$z$、$\beta$、$p$，可计算由截断误差引入的相对误差。另一方面，如果想把截断误差控制在 $\Delta$ 内，可令

$$|\frac{a_Nz^N}{q(z)}|\leqslant\Delta$$

由此计算满足条件的 $N$，以达到精度要求。

下面分析计算部分和时产生的舍入误差，$S_N(z)$ 的标准计算方法是嵌套模式的 Horner 算法[14,15]：

$S_N(z)=a_0+z(a_1+z(a_2+z(\cdots+za_N))\cdots)$

采用下面的递归算法来实现：

$S_{N,N}(z)=a_Nz$

$S_{N,i}(z)=z(a_i+T_{i+1}),i=N-1,N-2,\cdots,1$

$S_N(z)=a_0+S_{N,1}$

令 $fl(z)$、$fl(a_i)$ 分别是 $z$、$a_i$ 的机器表示，满足

$fl(z)=z(1+\delta_z)$

$fl(a_i)=a_i(1+\delta_{a_i})$

由上述的递归算法和第 2 节的舍入误差基本结论，得到

$\tilde{S}_N(z)=S_N(z)\cdot(1+\theta_{n(S_N(z))})$

下面的式子给出了误差 $\theta_{n(T_N(z))}$ 的累积过程：

$$fl(S_{N,N}(z))=fl(a_N\times z)$$
$$=(a_N(1+\delta_{a_N})\times z(1+\delta_z))(1+\delta_m)$$
$$fl(S_{N,i}(z))=fl(z\times(a_i+T_{i+1}))$$
$$=(z(1+\delta_2)\times(a_i(1+\delta_{a_i})+fl(T_{i+1}))(1+\delta_a))(1+\delta_m)$$
$$fl(S_N(z))=fl(a_0+S_{N,1})$$
$$=(a_0(1+\delta_{a_0})+fl(S_{N,1}))(1+\delta_a))$$

其中，$|\delta_m|$、$|\delta_a|$、$|\delta_z|\leqslant u(p)$。

基于上述分析，针对正弦函数和余弦函数在 GNU 下的具体实现，详细分析得到它们的理论误差界。

3.1.1 正弦函数理论误差分析

GNU 下正弦函数 $\sin(x)$ 的赋值程序实现采用了泰勒级数展开式，在计算部分和时采用了 Horner 算法。$\sin(x)$ 的级数表达式为：

$$\sin(x)=\sum_{i=0}^{\infty}\frac{(-1)^i}{(2i+1)!}x^{2i+1}$$

因正弦函数是周期函数，程序首先把 $x$ 映射到区间 $[-\frac{\pi}{4},\frac{\pi}{4}]$ 上，然后做变量替换，用 $z=x^2$ 替换 $x$，则

$$\sin(z)=\sum_{i=0}^{\infty}a_iz^i=x-\frac{x}{3!}z+\frac{x}{5!}z^2-\frac{x}{7!}z^3+\cdots$$

其中，$a_i=\frac{(-1)^i}{(2i+1)!}x$。

$\sin(x)$ 赋值程序在 $i=6$ 时进行截断，则

$$|\sum_{i=7}^{\infty}a_iz^i|\leqslant|a_7z^7|=|\frac{(-1)^7}{15!}x\cdot z^7|=|\frac{1}{15!}x^{15}|$$

注意到 $\sin(z)$ 的充分小的下界是 $q(z)=x-\frac{x}{3!}z$，所以截断产生的相对误差界为：

$$\epsilon_T=|\frac{1}{15!}x^{15}/(x-\frac{x^3}{3!})|$$

计算部分和 $S_7(z)$ 的舍入误差为

$\tilde{S}_7(z)=S_7(z)\cdot(1+\theta_{n(S_7(z))})$

其中，$|\theta_{n(S_7(z))}|\leqslant\frac{16u(p)}{1-16u(p)}$。

也就是说，截断误差界与自变量 $x$ 的取值有关，而舍入误差界 $|\theta_{n(S_7(z))}|$ 与基 $\beta$ 和精度 $p$ 有关。若 $\epsilon_T\leqslant\phi_1\Delta$，$|\theta_{n(S_7(z))}|\leqslant\phi_2\Delta$，且 $\phi_1+\phi_2\leqslant1$，则计算正弦函数产生的误差小于等于 $\Delta$。

**定理 4** 给定基 $\beta=2$，精度 $p=53$，GNU 标准下正弦函数赋值程序理论误差界为 $3.1\times10^{-14}$，其中截断误差取其上界：

$$\max(|\frac{1}{15!}x^{15}/(x-\frac{x^3}{3!})|)\leqslant2.9\times10^{-14},x\in[-\frac{\pi}{4},\frac{\pi}{4}]$$

例如，取 $x=0.75$，那么 $\epsilon_T\leqslant1.5\times10^{-14}$，$|\theta_{n(S_7(z))}|\leqslant1.78\times10^{-15}$，所以，计算 $\sin(0.75)$ 的相对误差小于等于 $1.68\times10^{-14}$。

3.1.2 余弦函数理论误差分析

正弦函数 $\cos(x)$ 的级数表达式为

$$\cos(x)=\sum_{i=0}^{\infty}\frac{(-1)^i}{(2i)!}x^{2i}$$

做变量替换，用 $z=x^2$ 替换 $x$，则

$$\cos(z)=\sum_{i=0}^{\infty}a_i z^i=1-\frac{1}{2!}z+\frac{1}{4!}z^2-\frac{1}{6!}z^3+\cdots$$

其中，$a_i=\frac{(-1)^i}{(2i)!}$。

$\cos(x)$在 $i=7$ 时进行截断，则

$$\sum_{i=8}^{\infty}a_i z^i\leqslant a_8 z^8=\frac{(-1)^8}{16!}z^8=\frac{1}{16!}z^8$$

注意到 $\cos(z)$的充分小的下界是 $q(z)=1-\frac{1}{2!}z$，所以截断误差

$$\varepsilon_T\leqslant\left|\frac{1}{16!}z^8/(1-\frac{1}{2!}z)\right|=\left|\frac{1}{16!}x^{16}/(1-\frac{1}{2!}x^2)\right|$$

计算部分和 $S_7(z)$的舍入误差为

$$\tilde{S}_N(z)=S_N(z)\cdot(1+\theta_{n(S_7(z))})$$

其中，$|\theta_{n(S_7(z))}|\leqslant\frac{16u(p)}{1-16u(p)}$。

**定理 5** 给定基 $\beta=2$，精度 $p=53$，GNU 标准下余弦函数赋值程序理论误差界为 $3.3\times10^{-15}$，其中截断误差取其上界：

$$\max\left(\left|\frac{1}{16!}x^{16}/(1-\frac{1}{2!}x^2)\right|\right)\leqslant1.45\times10^{-15}$$

$$x\in[-\frac{\pi}{4},\frac{\pi}{4}]$$

例如，取 $x=0.7$，那么 $\varepsilon_T\leqslant2.11\times10^{-16}$，$|\theta_{n(S_7(z))}|\leqslant1.78\times10^{-15}$，计算 $\cos(0.7)$的相对误差小于等于 $2.0\times10^{-15}$。

### 3.2 指数函数误差分析

GNU 下指数函数的赋值，采用的策略不是级数表示法而是特殊的有理函数来逼近，指数函数 $\exp(x)$的实现分为 3 步：

首先令 $r=x-k\ln2$，$k\in z$，满足

$$|r|\leqslant0.5\ln2 \tag{11}$$

然后，用特殊有理函数求 $\exp(r)$的近似值，记：

$$R=r(\exp(r)+1)/(\exp(r)-1)=2+r^2/6-r^4/360+\cdots$$

使用 Remes 算法产生次数为 5 的多项式求出 $R$，此多项式的误差小于等于 $2^{-59}$，然后计算 $\exp(r)$：

$$\exp(r)=1+\frac{2r}{R-r}$$

最后，计算 $\exp(x)=2^k\exp(r)$。

在对指数函数赋值的计算过程中，误差来源于 $r$、$\exp(r)$和 $\exp(x)$的计算。

由 $k\in z$ 及式(11)的要求，计算得到整数 $k$，$\tilde{r}=fl(x-k\ln2)=(x-k\ln2)(1+\theta_{j\max(n(x),n(k\ln2))})$，其中，$j\geqslant\frac{|x|+|k\ln2|}{|x-k\ln2|}$，$j\in N$，随着 $x$ 的增大，$\exp(x)$的误差界也增大。

$$fl(\exp(r))=fl(1+\frac{2r}{R-r})=(1+\frac{2r}{R-r})(1+\theta_{3n(r)+2})$$

其中，$n(r)$是计算 $r$ 时产生的误差。

$$fl(\exp(x))=2^k\exp(r)(1+\theta_{3n(r)+3})$$

**定理 6** 给定基 $\beta=2$，精度 $p=53$，GNU 标准下指数函数赋值程序理论误差界为 $1.21\times10^{-16}$。

例如，$x=-5$ 时，由式(11)可知 $k=-6$，在基 $\beta=2$、精度 $p=56$ 下，计算 $\exp(5)$的误差界是 $4.99\times10^{-16}$。

### 3.3 对数函数误差分析

对数函数 $\log(x)$的赋值原理与指数函数 $\exp(x)$算法类似，对于给定的参数 $x$，首先通过迭代求出 $k$ 和 $f$，满足

$$x=2^k\times(1+f),\frac{\sqrt{2}}{2}<1+f<\sqrt{2}$$

令 $s=f/(2+f)$，则 $1+f=(1+s)/(1-s)$，接下来求 $\log(1+f)$。

$$\begin{aligned}\log(1+f)&=\log(1+s)-\log(1-s)\\&=2s+\frac{2}{3}s^3+\frac{2}{5}s^5+\cdots\\&=2s+s\times R\end{aligned}$$

其中，$R$ 为 Remes 算法在区间$[0,0.01716]$上构造的一个 14 阶的多项式，误差上界为 $2^{-58}$。

为减小舍入误差，对 $2s$ 和 $\log(1+f)$进行了等价变换，ln2 被分成高位和低位两部分，最后求出 $\log(x)$。

$$\ln2=\ln2_H+\ln2_L$$

$$2s=f-s\times f=f-\frac{f^2}{2}+s\times\frac{f^2}{2}$$

$$\log(1+f)=f-(\frac{f^2}{2}-s\times(\frac{f^2}{2}+R))$$

$$\begin{aligned}\log(x)&=k\times\ln2+\log(1+f)\\&=k\times\ln2_H+(f-(\frac{f^2}{2}-(s\times(\frac{f^2}{2}+R)+k\times\ln2_L)))\end{aligned}$$

计算 $s=f/(2+f)$时满足 $\tilde{s}=s(1+\theta_{3n(f)+2})$，$\log(x)$的误差界比较复杂，令 $e_1=|s\times(f^2/2+R)+k\times\ln2_L|$，$e_2=\frac{|f^2/2|+e_1}{|f^2/2-e_1|}$，则 $fl(\log(x))=\log(x)(1+\theta_{n(\log(x))})$。其中，$n(\log(x))=\frac{|f|+e_2}{|f-e_2|}(5n(f)+9)+2$。

**定理 7** 给定基 $\beta=2$，精度 $p=53$，GNU 标准下指数函数赋值程序理论误差界为 $7.33\times10^{-15}$。

### 3.4 理论误差分析综述

根据误差分析的结果，总能得到一对数，以它们作为集合的边界，就构成了包含真实值的区间，进而完成了对初等函数的可验证赋值，但是理论误差分析是有技巧和约束条件的，对于某些特殊的情形，有时很难分析出一个较紧的界。对于一般的运算序列，追踪每一步的误差是相当困难的。在这种情况下，区间算术是一个快速获得函数的可验证赋值及进行误差分析的有效方法。

## 4 基于区间算术的可验证赋值及误差分析

### 4.1 基于区间类库 Boost 重写初等函数赋值程序

Boost 库是一个有效而且通用的 C++标准库，其内部实现了区间算术的基本操作，提供了区间的模板类[16]。它是开源的，可以在 www.boost.org 上获取。与其它区间算术软件包的不同之处在于，它使用了策略来规定自变量的行为，包括舍入、检查和比较。运用合适的策略，Boost 区间算术库可以对任意类型的变量进行区间算术操作。

本文基于 Boost 库中区间类提供的基本区间算术运算，对 GNU 下初等函数的赋值程序进行重写，目标是完成函数的可验证赋值和误差界的计算。使用 Boost 区间算术库时，需要包含头文件〈boost/numeric/interval.hpp〉，在实现过程中，将所有的常数系数定义成区间，采用标准的舍入方式及区间算术运算，以保证每一步的计算结果始终包含问题的真实值。

### 4.2 检查策略和舍入策略

程序 1 是就代码重写进行的预定义工作，包括区间变量的检查策略和舍入策略，并重载了输出操作符“≪”。

**程序 1**

```
#ifndef INTERVAL_H
#define INTERVAL_H
#include <boost/numeric/interval.hpp>
namespace myinterval
{   using namespace boost;
    using namespace numeric;
    using namespace interval_lib;
    typedef boost::numeric::interval_lib::rounded_transc_std
            <float> my_rounding;
    typedef save_state<my_rounding> R;
    typedef checking_strict<float> P;
    typedef interval<float,policies<R,P>> I;
    typedef boost::numeric::interval_lib::rounded_transc_std
            <double> my_rounding2;
    typedef save_state<my_rounding2> R2;
    typedef checking_strict<double> P2;
    typedef interval<double,policies<R2,P2>> I2;
}
typedef myinterval::I intv_float;
typedef myinterval::I2 intv_double;
template<class os_t>
os_t& operator≪(os_t &os,const intv_double &a)
{   os≪'['≪a.lower()≪','≪a.upper()≪']';
    return os;
}
template<class os_t1>
os_t1& operator≪(os_t1 &os,const intv_float &a)
{   os≪'['≪a.lower()≪','≪a.upper()≪']';
    return os;
}
#endif
```

基于 Boost 区间算术库，对初等函数实现代码重写后，不仅得到了初等函数的可验证赋值，而且计算了浮点环境下求得的赋值结果的误差界，实验结果详见第 5 节。

## 5 数值实验

数值实验中使用的计算机 CPU 主频为 3.10GHz，内存为 4.00GB，集成开发环境 IDE 为 Code::Blocks。数据的图形显示使用的工具为 MATLAB2011a。

### 5.1 基于区间算术的初等函数可验证赋值

在给定自变量 $x$、基 $\beta$ 和精度 $p$ 的情况下，由第 3 节的理论分析和第 4 节的区间算术技术，可得到一个保证包含真实值的区间，从而进行可验证赋值。

数值实验中，使用 Boost 区间类重写初等函数赋值程序，对于给定的参数值和赋值函数，均给出了包含真实值的区间包络，再与 C 标准下计算的浮点值比较，得到一个误差界。表 1－表 4 中分别给出了正弦函数、余弦函数、指数函数和对数函数在两个不同参数值下的赋值结果。

从表 1－表 4 可以看出，对于同一个函数，不同参数值下得到的浮点值的误差界是不一样的，这不是偶然现象，5.2 节将给出这些函数在不同参数下误差界变化的趋势。

表 1 正弦函数的可验证赋值

| 输入 x | 0.5 | 1.5 |
|---|---|---|
| 浮点值 | 0.47942553860420301 | 0.99749498660405445 |
| 区间赋值 | [0.47942553860420295, 0.47942553860420301] | [0.99749498660405433, 0.99749498660405456] |
| 误差界 | 5.5511151231257827E−17 | 2.2204460492503131E−16 |

表 2 余弦函数的可验证赋值

| 输入 x | 0.5 | 1.5 |
|---|---|---|
| 浮点值 | 0.87758256189037276 | 0.070737201667702906 |
| 区间赋值 | [0.87758256189037265, 0.87758256189037276] | [0.070737201667702893, 0.070737201667702906] |
| 误差界 | 1.1102230246251565E−16 | −1.3877787807814457E−17 |

表 3 指数函数的可验证赋值

| 输入 x | −5 | 5 |
|---|---|---|
| 浮点值 | 0.00673796999085467 | 148.4131591025766 |
| 区间赋值 | [0.00673796999085467, 0.00673796999085479] | [148.4131591025766, 148.41315910257663] |
| 误差界 | 8.6736173798840355E−19 | 2.8421709430404007E−14 |

表 4 对数函数的可验证赋值

| 输入 x | 1500 | 3500 |
|---|---|---|
| 浮点值 | 7.73132203870903014 | 8.1605182474775049 |
| 区间赋值 | [7.73132203870903014, 7.73132203870903023] | [8.1605182474775049, 8.1605182474775066] |
| 误差界 | 8.8817841970012523E−16 | 1.7763568394002505E−15 |

### 5.2 初等函数误差界

利用区间算术对 GNU 下初等函数进行误差测试的过程中，具体的做法是将自变量 $x$ 的取值限定在区间 $[\underline{X},\overline{X}]$ 上，将 $[\underline{X},\overline{X}]$ 划分成 $n$ 个子区间 $[\underline{x_i},\overline{x_i}]$，$i=1,\cdots,n$，然后在每个子区间 $[\underline{x_i},\overline{x_i}]$ 上随机取 100000 个数据值作为自变量，计算浮点环境下初等函数 $f(x)$ 的函数值，以及使用区间算术重写后得到的区间值 $F(x)=[\underline{F(x)},\overline{F(x)}]$，计算误差界：

$$\varepsilon=\max\{|\underline{F(x)}-f(x)|,|\overline{F(x)}-f(x)|\}$$

最后求每个区间段 $[\underline{x_i},\overline{x_i}]$ 的误差界平均值：

$$\bar{\varepsilon}=\left(\sum_{i=1}^{100000}\varepsilon_i\right)/100000$$

以 $[\underline{x_i},\overline{x_i}]$ 的区间中点 $(\underline{x_i}+\overline{x_i})/2$ 为横坐标，$\bar{\varepsilon}$ 的值为纵坐标作图，数值实验结果如图 1 所示（图 1 在 Matlab 下完成[17]，数据使用了 GNU 下测试时使用的参数与误差界平均值）。

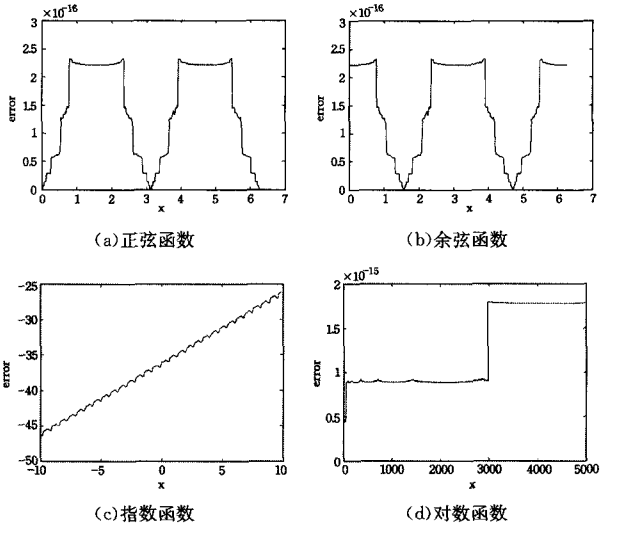


图 1 初等函数误差界变化趋势

具体来说，正弦函数与余弦函数自变量的取值范围为[0，2π]，划分的子区间的个数为 400，即每个子区间为[$i(2\pi/400)$，$(i+1)(2\pi/400)$]，$i=0,\cdots,399$。测试结果分别如图 1(a)和图 1(b)所示。指数函数的测试范围为[−10，10]，分割成 200 段小区间，误差结果如图 1(c)所示，为图示效果，纵坐标为取自然对数之后的值。对数函数的测试范围为[0，5000]，分割成 5000 段小区间，测试结果如图 1(d)所示。

从测试结果可以看出，尽管对于同一个函数，若自变量取值不同，误差界也是不同的。三角函数是一个周期函数，误差的变也呈现出周期变化的趋势。正弦函数的误差界在 0 到 π/4 区间内逐渐增大，在 π/4 到 3π/4 区间达到峰值，而在 3π/4 到 π 区间内误差界又逐渐减小；指数函数的误差界随着自变量的增大而增大；对数函数误差界的变化呈现跳跃阶梯状，在 0 到 1 区间误差界变化平缓，到达临界点 1 时误差界发生跳变，在临界点 3000 处再一次发生跳变。

**结束语** 本文针对 GNU 下初等函数的实现进行误差分析，得到一个理论误差界，基于此可得到一个理论上的区间赋值。由于理论分析比较精妙，需要较多的数学知识，对于更一般的数值程序，要追踪每一步的误差非常困难。相比之下，区间算术能避免理论分析的复杂性，对函数进行可验证赋值，并给出函数的误差界，主要工作在于把浮点程序改写成区间程序。目前我们已经完成了初等函数的可验证赋值及数值误差界的计算，除本文提到的 4 个函数，还包括反三角函数、正切函数、双曲函数等。但是，如果待分析和赋值的函数或数值程序的运算(特别是乘除运算)次数较多，就会导致得到的区间宽度太大而难以接受，在这种情况下，需要结合区间的精化等更为精妙的技术。

## 参 考 文 献


[1] Wang Wen-hua, GUO Zhi-hua, CAO Huai-xin. An upper bound for the adiabatic approximation error[J]. Science China Physics, Mechanics and Astronomy, 2014, 57(2): 218-224

[2] Jeannerod C P, Rump S M. Improved error bounds for inner products in floating-point arithmetic[J]. SIAM Journal on Matrix Analysis and Applications, 2013, 34(2): 338-344

[3] 袁梦. 复杂计算的误差定量分析方法及其应用[D]. 杭州：浙江大学，2006

[4] 周腾飞. Bernstein 基多项式函数的高精度计算及其动态误差分析研究[D]. 长沙：国防科学技术大学，2011

[5] Rump S M. Verification methods: Rigorous results using floating-point arithmetic[J]. Acta Numerica, 2010, 19: 287-449

[6] Muller J M. Elementary functions: Algorithms and Implementation(2nd ed)[M]. Birkhäuser, 2006

[7] Granlund T. The GNU multiple precision arithmetic library [OL]. http://www.swox.se/gmp

[8] Fousse L, Hanrot G, Lefevre V, et al. MPFR: A multiple-precision binary floating-point library with correct rounding[J]. ACM Transactions on Mathematical Software (TOMS), 2007, 33(2): 1-14

[9] Backeljauw F, Becuwe S, Cuyt A, et al. Validated evaluation of special mathematical functions [M] // Intelligent Computer Mathematics. 2008: 206-216

[10] Brisebarre N, De Dinechin F, Jeannerod C P, et al. Handbook of floating-point arithmetic [M]. Springer Science & Business Media, 2009

[11] Higham N J. Accuracy and stability of numerical algorithms(2nd ed)[M]. Siam, 2011

[12] Moore R E, Kearfott R B, Cloud M J. Introduction to interval analysis[M]. Siam, 2009

[13] Stallman R M. Using GCC: The GNU Compiler Collection Reference Manual for GCC 3.3.1[M]. Free Software Foundation, 2003

[14] Sauer T. Numerical Analysis(2nd ed)[M]. 北京：机械工业出版社，2012

[15] Mathews J H, Fink K D. Numerical methods using MATLAB (第四版)[M]. 周璐，陈渝，钱芳，等. 北京：电子工业出版社，2012

[16] Brönnimann H, Melquiond G, Pion S. The design of the Boost interval arithmetic library[J]. Theoretical Computer Science, 2006, 351(1): 111-118

[17] 刘保柱，苏彦华，张宏林. MATLAB 7.0 从入门到精通[M]. 北京：人民邮电出版社，2010


(上接第 18 页)


[160] Mezuman E, Tarlow D, Globerson A, et al. Tighter linear program relaxations for high order graphical models[J]. arXiv preprint arXiv: 1309.6848, 2013

[161] Dworkin L, Kearns M, Xia L. Efficient Inference for Complex Queries on Complex Distributions [C] // Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics. Reykjavik, Iceland: JMLR.org, 2014: 211-219

[162] Scutari M. Bayesian Network Constraint - Based Structure Learning Algorithms: Parallel and Optimised Implementations in the bnlearn R Package[J]. arXiv preprint arXiv: 1406.7648, 2014

[163] Tristan J B, Huang D, Tassarotti J. Augur: a Modeling Language for Data-Parallel Probabilistic Inference[J]. arXiv preprint arXiv: 1312.3613, 2013

[164] Kollar T, Tellex S, Walter M R. Approaching the Symbol Grounding Problem with Probabilistic Graphical Models[J]. AI MAGAZINE, 2011, 32(4): 64-76

[165] 石焕南. 受控理论与解析不等式[M]. 哈尔滨：哈尔滨工业大学出版社，2012

[166] Werner T. High-arity interactions, polyhedral relaxations, and cutting plane algorithm for soft constraint optimisation (MAP-MRF)[C] // IEEE Conference on Computer Vision and Pattern Recognition, 2008(CVPR 2008). IEEE, 2008: 1-8

[167] Aoyama K, Kohsaka F. Fixed point theorem for α-nonexpansive mappings in banach spaces[J]. Nonlinear Analysis, 2011, 74: 4387-4391

[168] Kien B T, Wong M M, Wong N C, et al. Solution existence of variational inequalities with pseudo-monotone operators in the sense of Brbzis[J]. Optimal Theory Application, 2009, 140: 249-263

[169] Genest C, Rémillard B. Test of independence and randomness based on the empirical copula process[J]. Test, 2004, 13(2): 335-369

[170] 张恭庆. 临界点理论及其应用[M]. 上海：上海科学技术出版社，1986

[171] Chen P N, Alajaji F. A Generalized Poor-Verdú Error Bound for Multihypothesis Testing[J]. IEEE Transactions on Information Theory, 2012, 58(1): 311-316