The CoStLy C++ Class Library

Markus Neher
Universität Karlsruhe
Institut für Angewandte und Numerische Mathematik
76128 Karlsruhe, Germany

April 21, 2008

1 Introduction

CoStLy (Complex Standard Functions Library) has been developed as a C++ class library for the validated computation of function values and of ranges of the complex standard functions in

$$S_F = \{\text{exp, ln, arg, sqr, sqrt, power, pow, root, cos, sin, tan, cot, cosh, sinh, tanh, coth, acos, asin, atan, acot, acosh, asinh, atanh, acoth}\},$$

where power(z, n) is the power function for integer exponents, pow(z, p) is the power function for real or complex exponents, and root(z, n) denotes the nth root function.

An interval library for the real standard functions in S_F is required by CoStLy, for evaluating the compositions of real functions that make up the real and imaginary parts of the complex functions in S_F. CoStLy has been programmed such that either C-XSC [3, 4] or filib++ [6, 7] can be used for this purpose. Today, CoStLy it is also integrated in the C-XSC library.

CoStLy is distributed under the terms of the GNU General Public License. The software is currently available at the following sites:

`http://www.xsc.de`
`http://iamlasun8.mathematik.uni-karlsruhe.de/~ae16/CoStLy.html`

2 Notation and Preliminary Remarks

2.1 Rectangular Complex Interval Arithmetic

The set of compact real intervals is defined by

$$\mathbb{IR} = \{\, \boldsymbol{x} = [\underline{x}, \overline{x}] \mid \underline{x}, \overline{x} \in \mathbb{R},\ \underline{x} \leq \overline{x} \,\}.$$

Dagstuhl Seminar Proceedings 08021
Numerical Validation in Current Hardware Architectures
http://drops.dagstuhl.de/opus/volltexte/2008/1449

A real number x is identified with a point interval $\boldsymbol{x} = [x, x]$. Throughout this paper, intervals are denoted by boldface.

A rectangular complex interval $\boldsymbol{z}$ is defined by a pair of two real intervals $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\boldsymbol{z} = \boldsymbol{x} + \imath\boldsymbol{y}, \quad \boldsymbol{z} = \{z = x + \imath y \mid x \in \boldsymbol{x},\ y \in \boldsymbol{y}\}.$$

The set of all complex rectangular intervals is denoted by $\mathbb{IC}$. For a bounded subset M of $\mathbb{C}$, the *interval hull* $\Box M$ of M is the smallest rectangular interval that contains M. We have

$$\Box M = [\inf_{z\in M} \operatorname{Re} z, \sup_{z\in M} \operatorname{Re} z] + \imath[\inf_{z\in M} \operatorname{Im} z, \sup_{z\in M} \operatorname{Im} z].$$

2.2 Inclusion Functions

For $D \subseteq \mathbb{C}$, the range $\{f(z) : z \in D\}$ of a function $f : D \longrightarrow \mathbb{C}$ is denoted by $\operatorname{Rg}(f, D)$. An *inclusion function* F of a given function $f : \mathbb{C} \to \mathbb{C}$ is an interval function $F : \mathbb{IC} \to \mathbb{IC}$ that encloses the range of f on all intervals $\boldsymbol{z} \subseteq D$:

$$F(\boldsymbol{z}) \supseteq \operatorname{Rg}(f, \boldsymbol{z}) \quad \text{for all} \quad \boldsymbol{z} \subseteq D.$$

If $F(\boldsymbol{z}) = \Box\operatorname{Rg}(f, \boldsymbol{z})$ holds for all $\boldsymbol{z} \subseteq D$, then F is called *optimal*.

The real and imaginary parts of any function in S_F can be expressed as compositions of real standard functions. Optimal inclusion functions are obtained by determining the extremal values of these compositions [1, 2, 5, 8].

2.3 The CoStLy Library Functions

The design of inclusion functions for the functions in S_F has been guided by the paradigm that range bounds must be valid in any circumstance. For a single-valued complex function f, its inclusion function $F : \mathbb{IC} \to \mathbb{IC}$, and some given rectangular complex interval $\boldsymbol{z}$, validity means only that $F(\boldsymbol{z})$ must contain the set $\{f(z)|z \in \boldsymbol{z}\}$. The functions exp, sqr, power, cos, sin, tan, cot, cosh, sinh, tanh, and coth are single-valued and analytic on their respective domain. CoStLy contains optimal inclusion functions for these functions (where optimal refers to the accuracy of the implemented algorithms, if performed in exact arithmetic).

The meaning of a valid enclosure is less obvious for a multi-valued function. For example, the definition of $\sqrt{-1}$ depends very much on the context of the computation. Possible values include $+\imath$, $-\imath$, $\{+\imath, -\imath\}$, $\imath \cdot [-1, 1]$, or the empty set. Three types of inclusion functions have been implemented in CoStLy to accommodate varied demands.

Each multi-valued function f in S_F has analytic branches on appropriate subsets of the complex plane. The CoStLy library contains an inclusion function F_p for the single-valued principal branch of f. Usually, F_p is defined

on a subset of $\mathbb{IC}$. If $\boldsymbol{z}$ is not in the domain of definition of f, the computation is aborted throwing an exception and issuing a warning message.

Inclusion functions that are defined for all $\boldsymbol{z} \in \mathbb{IC}$, for which at least one branch of f is bounded on $\boldsymbol{z}$, are denoted by F_c. Depending on the location of $\boldsymbol{z}$ in the complex plane, $F_c(\boldsymbol{z})$ returns function values belonging to different branches of f. As an immediate consequence, there are regions in $\mathbb{C}$ where inclusion isotonicity is lost.

Where applicable, an inclusion function F_a enclosing all function values of f has also been implemented. For example, it is available for roots. For some multi-valued complex functions, however, the set of all function values is unbounded in general. In this case, no such inclusion function is available.

We refer to [8] for a detailed description of all inclusion functions contained in CoStLy. The implementation is also extensively commented in the CoStLy source code.

3 Numerical Examples

If performed in exact arithmetic, the inclusion functions of type F_p compute optimal range bounds. For the sake of accuracy, a major effort has been made in the implementation of the algorithms in floating point arithmetic to eliminate all intermediate expressions subject to numerical overflow, underflow, or cancellation. The CoStLy library has been extensively tested for arguments with absolute values ranging from 1.0E-300 to 1.0E+300. For most arguments, the computed bounds for function values are highly accurate. In many test cases, the observed precision of the result was about 50 correct bits (out of the 53 bits available in IEEE 754 floating point arithmetic) for point arguments.

Example 1: Function Values for Point Intervals

In Table 1, we list results for point intervals for selected library functions. The following arguments were chosen:

$$z_1 = 1.0^{-300} + \imath\ 1.0^{-300}, \quad z_2 = 1.0^{300} + \imath\ 1.0.$$

Here and in the following, a short notation for numbers is used. $a.b^c$ means $a.b \times 10^c$.

In the execution of CoStLy, the arguments were entered as real constants, so that their IEEE 754 best approximations were actually used in the computation. In the second and the fourth column of Table 1, the approximate principal value of each function at the respective argument is given. In columns three and five, a pair of positive numbers denotes the number of correct bits for the real/imaginary part of the computed function

f	$f(z_1)$	Correct bits	$f(z_2)$	Correct bits
sqrt	$1.1^{-150} + \imath\ 4.6^{-151}$	51.0 / 50.9	$1.0^{150} + \imath\ 5.0^{-151}$	51.7 / 51.7
ln	$-6.9^{2} + \imath\ 7.9^{-1}$	49.9 / 48.7	$6.9^{2} + \imath\ 1.0^{-300}$	49.9 / 52.4
asin	$1.0^{-300} + \imath\ 1.0^{-300}$	51.0 / 50.2	$1.6 + \imath\ 6.9^{2}$	52.6 / 48.6
acos	$1.6 - \imath\ 1.0^{-300}$	52.6 / 50.2	$1.0^{-300} - \imath\ 6.9^{2}$	49.9 / 48.6
atan	$1.0^{-300} + \imath\ 1.0^{-300}$	51.4 / 49.5	$1.6 + \imath\ 0.0$	52.6 / %
acot	$1.6 - \imath\ 1.0^{-300}$	52.6 / 49.5	$1.0^{-300} + \imath\ 0.0$	50.8 / %

Table 1: Function values for selected inclusion functions for point intervals.

value bounds. % is used if the IEEE 754 best approximation of the function value is zero. In this case, the relative accuracy of the function value enclosure is undefined.

For brevity, we show only results for one argument with a very small absolute value and one argument with a very large absolute value. For arguments with absolute values in between, similar accuracy of the computed function values was observed in many test cases.

Example 2: Range Bounds for Interval Arguments

Range bounds for

$$\boldsymbol{z}_3 = [0.1, 2.1] + \imath[0.1, 2.1], \quad \boldsymbol{z}_4 = [1.0^{20}, 9.0^{20}] + \imath\ [1.0^{20}, 9.0^{20}].$$

are shown in Table 2 for selected inclusion functions. For clarity, the range bounds are displayed rounded. As in Example 1, the observed accuracy of the computed bounds was about 50 bits with respect to the optimal bounds in exact arithmetic.

Example 3: Root Functions

For complex roots of integral orders, an inclusion function of type F_a is included in CoStLy. It is implemented as a set function root_all($\boldsymbol{z}, n$), which computes a list of n intervals that contain all n-th roots of $\boldsymbol{z}$:

$$\text{root\_all}(\boldsymbol{z}, n) \supseteq \{w : w^n = z, z \in \boldsymbol{z}\}.$$

For some interval $\boldsymbol{z} \subset \mathbb{C} - \{0\}$ with sufficiently small width, the optimal interval enclosure for all n-th roots consists of n distinct intervals. While these could be computed (in exact arithmetic), such a procedure would be computationally expensive, especially for large values of n. We use

f	$F(\boldsymbol{z}_3)$	$F(\boldsymbol{z}_4)$
sqrt	$[3.4^{-1}, 1.6] + \imath\,[3.4^{-2}, 1.1]$	$[1.0^{10}, 3.3^{10}] + \imath\,[1.6^{9}, 2.1^{10}]$
ln	$[-2.0,\ 1.1] + \imath\,[\ 4.7^{-2},\ 1.6]$	$[4.6^{1}, 4.9^{1}] + \imath\,[1.1^{-1}, 1.5]$
tan	$[-5.0,\ 5.0] + \imath\,[\ 1.0^{-1},\ 1.1^{1}]$	$[-1.2^{-308}, 1.2^{-308}] + \imath\,[9.9^{-1}, 1.1]$
cot	$[-5.8^{-1},\ 5.0] + \imath\,[-5.1,-1.0^{-1}]$	$[-1.2^{-308}, 1.2^{-308}] + \imath\,[-1.1, -9.9^{-1}]$
asin	$[\ 4.2^{-2},\ 1.6] + \imath\,[\ 1.0^{-1},\ 1.8]$	$[1.1^{-1}, 1.5] + \imath\,[4.7^{1}, 5.0^{1}]$
acos	$[\ 5.4^{-2},\ 1.6] + \imath\,[-1.8,-1.0^{-1}]$	$[1.1^{-1}, 1.5] + \imath\,[-5.0^{1}, -4.7^{1}]$
atan	$[\ 1.0^{-1},\ 1.6] + \imath\,[\ 1.8^{-2},\ 1.6]$	$[1.5, 1.6] + \imath\,[1.2^{-22}, 5.1^{-21}]$

Table 2: Range bounds for interval arguments.

the following strategy instead: $\boldsymbol{z}$ is enclosed in the polar interval $(\boldsymbol{r}, \boldsymbol{\phi}) = ([\underline{r}, \overline{r}], [\underline{\varphi}, \overline{\varphi}]) = (\mathrm{abs}(\boldsymbol{z}), \arg \boldsymbol{z})$. All n-th roots of $\boldsymbol{z}$ are then contained in the union $\cup_{k=0}^{n-1}(\boldsymbol{r}_k, \boldsymbol{\phi}_k)$, where

$$\boldsymbol{r}_k = [\underline{r}^{(1/n)}, \overline{r}^{(1/n)}], \quad \boldsymbol{\phi}_k = (\boldsymbol{\phi} + 2k\pi)/n, \quad k = 0, 1, \ldots, n-1.$$

Finally, $(\boldsymbol{r}_k, \boldsymbol{\phi}_k)$ is enclosed in a rectangular interval $\boldsymbol{w}_k$, such that root_all$(\boldsymbol{z}, n) = \cup_{k=0}^{n-1} \boldsymbol{w}_k$ (Figure 1).

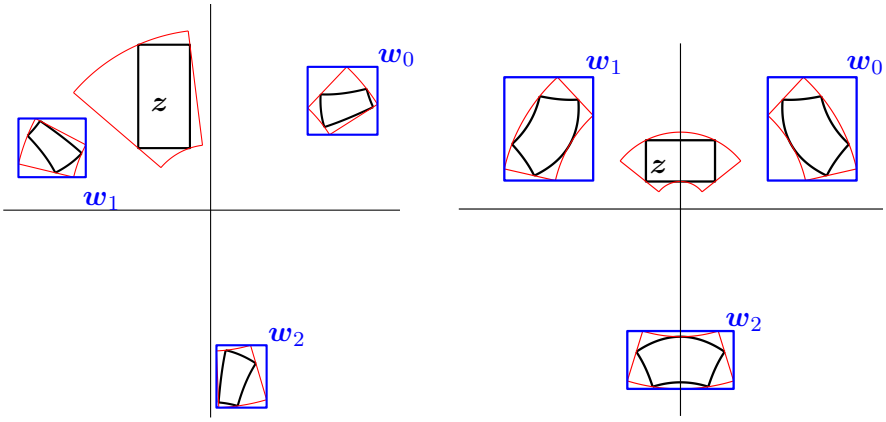

Figure 1: Inclusion sets for root_all$(\boldsymbol{z}, 3)$ for two different intervals $\boldsymbol{z}$.

References

[1] K. Braune. *Hochgenaue Standardfunktionen für reelle und komplexe Punkte und Intervalle in beliebigen Gleitpunktrastern.* PhD thesis, Uni-

versität Karlsruhe, 1987.

[2] K. Braune and W. Krämer. High-accuracy standard functions for real and complex intervals. In E. Kaucher, U. Kulisch, and Ch. Ullrich, editors, *Computerarithmetic: Scientific Computation and Programming Languages*, pages 81–114. Teubner, Stuttgart, 1987.

[3] W. Hofschuster and W. Krämer. C-XSC 2.0: A C++ library for extended scientific computing. In R. Alt, A. Frommer, R. B. Kearfott, and W. Luther, editors, *Numerical Software with Result Verification, Springer Lecture Notes in Computer Science 2991*, pages 15–35. Springer, Berlin, 2004.

[4] R. Klatte, U. Kulisch, Ch. Lawo, M. Rauch, and A. Wiethoff. *C-XSC: A C++ Class Library for Extended Scientific Computing*. Springer, Berlin, 1993.

[5] W. Krämer. *Inverse Standardfunktionen für reelle und komplexe Intervallargumente mit a priori Fehlerabschätzungen für beliebige Datenformate*. PhD thesis, Universität Karlsruhe, 1987.

[6] M. Lerch, G. Tischler, and J. Wolff von Gudenberg. filib++ - Interval library specification and reference manual. Technical Report 279, Universität Würzburg, 2001.

[7] M. Lerch, G. Tischler, J. Wolff von Gudenberg, W. Hofschuster, and W. Krämer. The interval library filib++ 2.0. Design, features and sample programs. Preprint 2001/4, Universität Wuppertal, Wissenschaftliches Rechnen/Softwaretechnologie, 2001.

[8] M. Neher. Complex standard functions and their implementation in the CoStLy library. *ACM TOMS*, 33:20–46, 2007.