CoStLy: A Validated Library for Complex Functions

Markus Neher*[1] and **Ingo Eble****[1]

[1] Institut für Angewandte Mathematik, Universität Karlsruhe, 76128 Karlsruhe, Germany

CoStLy is a new C++ library for the validated computation of function values and of ranges of complex standard functions.

1 Introduction

For the validated computation of complex functions, Bühler [1] has implemented algorithms that were presented in [2] as part of a Pascal–XSC interval library. Although Pascal-XSC is still available [4], it is no longer being maintained and it requires outdated compiler versions, which make it difficult to use nowadays.

CoStLy is a new C++ library of complex standard functions for the rigorous computation of function values and of function ranges. To ensure validated results, all truncation and roundoff errors are calculated during the course of the floating-point arithmetic computation and enclosed into the result. CoStLy is based on either one of the well-established interval arithmetic libraries C-XSC or filib++ [4], both of which provide real and complex interval arithmetic, real standard functions, but no complex standard functions.

CoStLy is availabe under the GNU GPL from the CoStLy website [3].

2 Single-valued functions

Complex standard functions fall into two categories: single-valued functions, such as the exponential, hyperbolic or trigonometric functions, and multi-valued functions, like the logarithm or root functions. CoStLy procedures for multi-valued functions are treated in Section 3.

2.1 Separable functions

Some complex functions are separable, which means that their real and imaginary parts can be expressed as products of two real functions of one variable. Separable functions have been implemented in CoStLy via procedures from C-XSC or filib++ for real functions. For example, if $Z = X + \imath Y$ is a rectangular complex interval, then e^Z is implemented as

$$e^Z = e^X \cos Y + \imath e^X \sin Y.$$

Direct interval evaluation of the right hand side yields an optimal rectangular enclosure of the range $\{e^z \mid z \in Z\}$. Similar well-known decompositions are available for $\cos Z$, $\sin Z$, $\cosh Z$, and $\sinh Z$ [2].

2.2 Tangent function

For $\tan Z$, optimal range bounds of

$$\mathrm{Re}\tan z = \frac{\sin(2x)}{\cos(2x)+\cosh(2y)}, \qquad \mathrm{Im}\tan z = \frac{\sinh(2x)}{\cos(2x)+\cosh(2y)}$$

are calculated rigorously [2]. $\cot Z$, $\tanh Z$ and $\coth Z$ are calculated with

$$\cot Z = \tan(\frac{\pi}{2} - Z), \quad \tanh Z = \overline{\imath \tan(\overline{\imath Z})}, \quad \coth Z = \imath \cot(\imath Z).$$

2.3 Single-valued power functions

Single-valued power functions power(Z, n) of integer order have been implemented in the following way: For $n = 2$, power$(Z, 2)$ = sqr$(Z) = X^2 - Y^2 + \imath 2XY$. For $n \geq 3$, power$(Z, n) = e^{n \ln Z}$ (where $\ln Z$ denotes the non-analytic logarithm described in Section 3.2). For $n < 0$, power$(Z, n) = 1/$power$(Z, -n)$ is used.

* Corresponding author: e-mail: markus.neher@math.uni-karlsruhe.de
** e-mail: ingo.eble@math.uni-karlsruhe.de

3 Multi-valued functions

For several multi-valued complex functions, we implemented two different procedures in CoStLy. Firstly, there is a subroutine for computing the principal value of the respective function. The principal value is usually analytic in a subset of the complex plane and undefined otherwise. Secondly, there are procedures for computing interval supersets of all function values of the respective function, where feasible.

3.1 Argument Function

An analytic argument function Arg Z with Arg $z \in (-\pi, \pi)$ is defined on the slit complex plane $\mathbb{C}^- = \mathbb{C} - (-\infty, 0]$. If Z intersects the negative real axes, Arg Z is not defined. As an auxiliary function for the computation of other complex functions, a single-valued, non-analytic argument function $\arg Z$ with $\arg z \in [-\pi, \frac{3\pi}{2}]$ has also been implemented. $\arg Z$ is defined for all rectangular complex intervals Z.

3.2 Logarithms and Roots

The CoStLy library contains the principal value of the logarithm,

$$\text{Ln}\, Z = \ln(|Z|) + \imath \text{Arg}\, Z, \quad Z \subset \mathbb{C}^-,$$

a single-valued, non-analytic logarithm,

$$\ln Z = \ln(|Z|) + \imath \arg Z, \quad Z \subset \mathbb{C} - \{0\},$$

the principal value of the square root,

$$\text{Sqrt}\, Z, \quad \text{Arg}\,(\text{Sqrt}\, z) = \text{Arg}\, z/2, \quad z \in Z \subset \mathbb{C}^-,$$

a list of all square roots,

$$\text{sqrt\_all}\,(Z) = \{W_1, W_2\} \supseteq \{w^2 = z \mid z \in Z \subseteq \mathbb{C}\},$$

and similar functions for other roots of integer order, i.e. procedures Sqrt (Z, n) for computing an analytic n-th root and sqrt_all (Z, n) for computing a list of n intervals containing all n-th roots.

3.3 Power Functions

As for roots, several power functions are implemented in CoStLy. There is an analytic power function,

$$\text{Pow}\,(Z, P) = e^{P \text{Ln}\, Z}, \quad Z \subset \mathbb{C}^-,$$

and a non-analytic power function pow_all (Z, P), which calculates an interval $W =$ pow_all (Z, P) such that

$$W \supseteq \{z^p \mid z \in Z,\ p \in P\}.$$

The implementation of pow_all (Z, P) has been particularly difficult. Depending on Z and P, $\{z^p \mid z \in Z,\ p \in P\}$ can be a set of isolated points, a subset of a circle (or a torus, or a disk), or an unbounded subset of the complex plane.

3.4 Inverse Trigonometric and Inverse Hyperbolic Functions

For inverse trigonometric and inverse hyperbolic functions, only the computation of principal values has been implemented in CoStLy. For the computation of asin Z, acos Z, and acosh Z, Z must not intersect the real axis for $x < -1$ or $x > 1$. Likewise, atan Z, acot Z, asinh Z, and atanh Z are only defined if Z does not intersect the imaginary axis for $y < -1$ or $y > 1$, and the computation of acoth Z requires that Z does not intersect $[-1, 1]$.

Conclusion

We have presented CoStLy, a freely available C++ library for the validated computation of several complex standard functions. Future work will concentrate on improving the computed enclosures.

References

[1] G. Bühler, diploma thesis, Karlsruhe University (1993).
[2] K. Braune and W. Krämer, in: Computerarithmetic: Scientific computation and programming languages, edited by E. Kaucher, U. Kulisch and C. Ullrich (Teubner, Stuttgart, 1987), pp. 81-114.
[3] CoStLy website, `http://www.uni-karlsruhe.de/~Markus.Neher/CoStLy.html [May 2004]`.
[4] XSC website, `http://www.xsc.de [May 2004]`.