Complex Number TR1 Algorithms

John Maddock

Table of Contents

Complex Number Inverse Trigonometric Functions 1
Implementation and Accuracy 1
asin 1
acos 2
atan 2
asinh 2
acosh 3
atanh 3
History 3

Complex Number Inverse Trigonometric Functions

The following complex number algorithms are the inverses of trigonometric functions currently present in the C++ standard. Equivalents to these functions are part of the C99 standard, and are part of the Technical Report on C++ Library Extensions.

Implementation and Accuracy

Although there are deceptively simple formulae available for all of these functions, a naive implementation that used these formulae would fail catastrophically for some input values. The Boost versions of these functions have been implemented using the methodology described in "Implementing the Complex Arcsine and Arccosine Functions Using Exception Handling" by T. E. Hull Thomas F. Fairgrieve and Ping Tak Peter Tang, ACM Transactions on Mathematical Software, Vol. 23, No. 3, September 1997. This means that the functions are well defined over the entire complex number range, and produce accurate values even at the extremes of that range, where as a naive formula would cause overflow or underflow to occur during the calculation, even though the result is actually a representable value. The maximum theoretical relative error for all of these functions is less than 9.5E for every machine-representable point in the complex plane. Please refer to comments in the header files themselves and to the above mentioned paper for more information on the implementation methodology.

asin

Header:

```
#include <boost/math/complex/asin.hpp>
```

Synopsis:

```
template<class T>
std::complex<T> asin(const std::complex<T>& z);
```

Effects: returns the inverse sine of the complex number z.

Formula: $\sin^{-1}(z) = -i\log(iz + \sqrt{1 - z^2})$

acos

Header:

```
#include <boost/math/complex/acos.hpp>
```

Synopsis:

```
template<class T>
std::complex<T> acos(const std::complex<T>& z);
```

Effects: returns the inverse cosine of the complex number z.

Formula: $\cos^{-1}(z) = \frac{\pi}{2} + i\log(iz + \sqrt{1 - z^2})$

atan

Header:

```
#include <boost/math/complex/atan.hpp>
```

Synopsis:

```
template<class T>
std::complex<T> atan(const std::complex<T>& z);
```

Effects: returns the inverse tangent of the complex number z.

Formula: $\tan^{-1}(z) = \frac{i}{2}(\log(1 - iz) - \log(iz + 1)) = -i\tanh^{-1}(iz)$

asinh

Header:

```
#include <boost/math/complex/asinh.hpp>
```

Synopsis:

```
template<class T>
std::complex<T> asinh(const std::complex<T>& z);
```

Effects: returns the inverse hyperbolic sine of the complex number z.

Formula: $\sinh^{-1}(z) = \log(z + \sqrt{z^2 + 1}) = i\sin^{-1}(-iz)$

acosh

Header:

```
#include <boost/math/complex/acosh.hpp>
```

Synopsis:

```
template<class T>
std::complex<T> acosh(const std::complex<T>& z);
```

Effects: returns the inverse hyperbolic cosine of the complex number z.

Formula: $\cosh^{-1}(z) = \log(z + \sqrt{z-1}\sqrt{z+1}) = \pm i\cos^{-1}(z)$

atanh

Header:

```
#include <boost/math/complex/atanh.hpp>
```

Synopsis:

```
template<class T>
std::complex<T> atanh(const std::complex<T>& z);
```

Effects: returns the inverse hyperbolic tangent of the complex number z.

Formula: $\tanh^{-1}(z) = \frac{1}{2}(\log(1+z) - \log(1-z))$

History

- 2005/12/17: Added support for platforms with no meaningful numeric_limits<>::infinity().
- 2005/12/01: Initial version, added as part of the TR1 library.