Proposal to Introduce Two and Three Argument hypot Functions to <complex>

Document Number: PnnnnR0
Date: 2017-03-06
Project: Programming Language C++
Audience: Library Working Group
Reply-to: Edward Smith-Rowland <3dw4rd@verizon.net>

0.1 Abstract

This proposal seeks to introduce two-argument and three-argument hypoteneuse functions to std::complex to match those functions for floating point types.

0.2 Introduction and Motivation

In complex mathematics and applications one often requires the distance between two points in the complex plane for segments of contour integrals or the evaluation of complex functions. In many fields of engineering and science one requires the magnitude of three-dimensional complex vector fields which often vary over many orders of magnitude across a region of space. These distances and magnitudes are best computed in ways which prevent avoidable overflow and underflow. Without these functions, programmers

might be encouraged to take the square root of the sum of the squares and hope for the best.

In addition, it is important to provide a uniform standard math library across all mathematical types in order to support the implementation of type-generic mathematical, statistical, and engineering libraries. To the greatest posible extent, complex and real floating point types should share a suite of standalone functions even when some of them are no-ops such as std::real() and std::imag() for foating point types.

Recall the distance formulas in two and three dimensions:

$$\mathsf{d}_2 = \sqrt{|\mathsf{x}_2 - \mathsf{x}_1|^2 + |\mathsf{y}_2 - \mathsf{y}_1|^2}$$

$$\mathsf{d}_3 = \sqrt{|\mathsf{x}_2 - \mathsf{x}_1|^2 + |\mathsf{y}_2 - \mathsf{y}_1|^2 + |\mathsf{z}_2 - \mathsf{z}_1|^2}$$

where the absolute value for complex numbers is used:

$$|\mathsf{w}| = \sqrt{Re[\mathsf{w}]^2 + Im[\mathsf{w}]^2}$$

Note that $|\mathsf{w}|$ is supported by std::abs(w)

0.3 Design & Considerations

This proposal requires that the value types of all complex arguments be subjected to the usual floating-point promotion rules. This is the return type for the hypot functions.

One could further require that if any one variable is complex, any real variables are promoted to complex and the promotion proceed as above. I'm not proposing that now but it would be easy to add if it is desired.

This proposal does not add functions prefixed by 'c' for complex or suffixed by type 'l' or 'f'. The latter is in keeping with the real counterparts and this proposal seeks to widen genericity.

0.4 Impact on the Standard

This proposal is a minimal library extension and does not require any changes to the core language, nor will this affect code that depends on the current definition of std::hypot for floating-point types. The complex variants of std::hypot is as mathematically well-understood as their real counterparts

0.5 Proposed Wording

Under 26.5.1 Header <complex> synopsis [complex.syn] ... // 26.5.7, values: ... template<class T> T abs(const complex<T>&); template<class T> T arg(const complex<T>&); template<class T> T norm(const complex<T>&); + template<class T> T hypot(const complex<T>&, const complex<T>&); + template<class T> T hypot(const complex<T>&, const complex<T>&, const complex<T>&);

Under 26.5.7 complex value operations [complex.value.ops]

Insert

template<class T> T hypot(const complex<T>& x, const complex<T>& y); /6 Returns: The two-dimensional hypoteneuse $\sqrt{x^2 + y^2}$.

template<class T> T hypot(const complex<T>& x, const complex<T>& y, const complex<T>& z); /7 Returns: The three-dimensional hypoteneuse $\sqrt{x^2 + y^2 + z^2}$.

0.6 Feature Testing

For the purposes of SG10, we recommend the feature-testing macro name ___cpp_lib_hypot added for real three-argument hypot be retained for this feature with a new date indicating extension of both std::hypot overloads to std::complex types.

0.7 Changes Since Revision 0 of the Proposal

This is the first draft of this proposal.

0.8 Acknowledgements

I am grateful to Walter Brown for encouraging this proposal.

0.9 References

[N1836] Matt Austern, Draft Technical Report on C++ Library Extensions http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1836.pdf

[P0030R1] Benson Ma, Proposal to Introduce a 3-Argument Overload to std::hypot http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/p003