Document Number: P0030R1
Date: 2015-11-06
Project: Programming Language C++
Audience: Library Working Group
Reply-to: Benson Ma <bm@berkeley.edu>

Proposal to Introduce a 3-Argument Overload to `std::hypot`

1 Abstract

This proposal seeks to introduce a 3-argument overload to the **`std::hypot`** function.

2 Introduction and Motivation

The **`std::hypot`** function, adopted for C++11 via [N1836] and later [N2009], has proven to be a useful general math utility function. One major application of this function is the calculation of the distance between 2 points in a 2-dimensional space. However, this is not useful enough, as many scientific and engineering applications (e.g. n-body simulations) model points in 3-dimensional space.

Recall the distance formula for points in a 3-dimensional space.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

Using only the current definition of **`std::hypot`**, the distance between 2 points in a 3-dimensional space can at best be calculated as such:

```
        auto d = std::hypot(x2-x1, std::hypot(y2-y1, z2-z1));
```

This code is far from optimal because it involves 2 function calls to **`std::hypot`** as well as an extra square-root and power operation, and consequently, a user-defined distance function is often used instead. Given that the 3-dimensional analog of **`std::hypot`** can be such a common use case, we propose that **`std::hypot`** be overloaded to support the three-arguments invocation case. The ideal function call should look like this:

```
        auto d = std::hypot(x2-x1, y2-y1, z2-z1);
```

This version can be better optimized, as it involves only one function call, and may allow library implementers to take advantage of any underlying hardware that provides optimized instructions for computing the 3-dimensional hypotenuse.

3 Design & Considerations

This proposal seeks to add the following function declaration overloads to **`std::hypot`**.

```
float       hypot( float x, float y, float z );                    (1)
double      hypot( double x, double y, double z );                 (2)
long double hypot( long double x, long double y, long double y );  (3)
```

We considered proposing a variadic overload of **`std::hypot`**, so that it could be further generalized to compute the Euclidean norm of a vector of arbitrary dimension. However, the general utility of such a feature is not immediately clear; we judge it best (at least for now) to relegate such a feature to domain-specific libraries, like the GNU Scientific Library [GSL].

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`**. The 3-dimensional variant of **`std::hypot`** is as mathematically well-understood as its 2-dimensional counterpart, has been previously implemented in C and C++ (e.g. **`gsl_hypot3()`** in [GSL]), and has proven its utility in practice over a considerable period of time.

5 Proposed Wording Commentary

Two proposed wording options are provided below. In both options, the proposed wording is intended to allow room for extending other currently defined math functions with three-argument overloaded versions in the future if needed. The first proposed wording is an attempt under the suggestion of SG6 (Numerics) to add the function overloads into a new subsection, and may entail larger than hoped for changes to the Standard. The second proposed wording is more conservative and has the advantage of automatically inheriting the guarantees for additional overloads (§26.8.11), but may not be appropriate to place inside §26.8 since all function overloads currently described in §26.8 are argument-type function overloads as opposed to argument-count function overloads.

6 Proposed Wording: Option 1

Place all current content in §26.8 [N4527] except for §26.8.11 into a new subsection:

26.8.1 C Library Overview **[c.math.overview]**

Add new subsection to §26.8 [N4527]:

26.8.2 Three-Argument Overloads **[c.math.3argoverload]**

C++ adds three-argument versions of certain math functions in **`<cmath>`**:

```
double hypot(double x, double y, double z);
float hypot(float x, float y, float z);
long double hypot(long double x, long double y, long double z);
```

Returns: $\sqrt{x^2 + y^2 + z^2}$

Place all current content in §26.8.11 into a new subsection:

26.8.3 Promoted Overloads **[c.math.promotedoverload]**

7 Proposed Wording: Option 2

Add new paragraph(s) between §26.8.9 and §26.8.10 of [N4527]:

C++ adds three-argument versions of certain math functions in **`<cmath>`**:

```
double hypot(double x, double y, double z);
float hypot(float x, float y, float z);
long double hypot(long double x, long double y, long double z);
```

Returns: $\sqrt{x^2 + y^2 + z^2}$

8 Feature Testing

For the purposes of SG10, we recommend the feature-testing macro name **`__cpp_lib_hypot`**.

9 Changes Since Revision 0 of the Proposal

- The proposed wording has been updated per guidance from SG6 (Numerics) and LEWG. Two proposed wording options are provided.
- Added a commentary section in front of the two proposed wording sections.
- Moved Feature Testing into its own section.
- Fixed typo with the [GSL] entry in the References section.

10 Acknowledgements

I am grateful to Walter Brown for the helpful comments on a draft of this proposal.

11 References

[N1836] Matt Austern, Draft Technical Report on C++ Library Extensions
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2005/n1836.pdf
[N2009] Pete Becker, Working Draft, Standard for Programming Language C++
http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2009.pdf
[GSL] Mark Galassi, et al. The GNU Scientific Library.
https://www.gnu.org/software/gsl/manual/
[N4527] Richard Smith, Working Draft, Standard for Programming Language C++
www.open-std.org/jtc1/sc22/wg21/docs/papers/2015/n4527.pdf