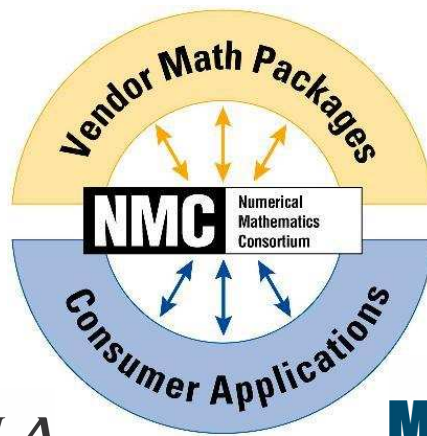


The Numerical Mathematics Consortium



Making Algorithms Portable: An Update from the Numerical Mathematics Consortium August 2006

Sam Shearman - Senior Product Manager
Darren Schmidt - Technical Product Specialist
Analysis, Math and Signal Processing Products
National Instruments



www.nmconsortium.org

Agenda

- Numerical Mathematics Consortium (NMC)
Overview
 - What is the NMC?
 - NMC Progress
- Technical Update
 - What does it take to port an algorithm?
 - Summary of ratified issues and functions



www.nmconsortium.org

Problem Statement

- Scientists, engineers, mathematicians and others develop algorithms in general-purpose numeric math tools
 - Each have different syntax and interpretation
 - Users cannot easily work with algorithms in other tools (portability, specialized tools)
 - Vendors cannot easily work with algorithms from other tool chains (custom links to each tool are required)
- Goal: A common foundation that enables algorithm reusability



www.nmconsortium.org

The NMC exists to address a common problem faced as engineers scientists and mathematicians increasingly use computers in their work. The problem is related to algorithm development.

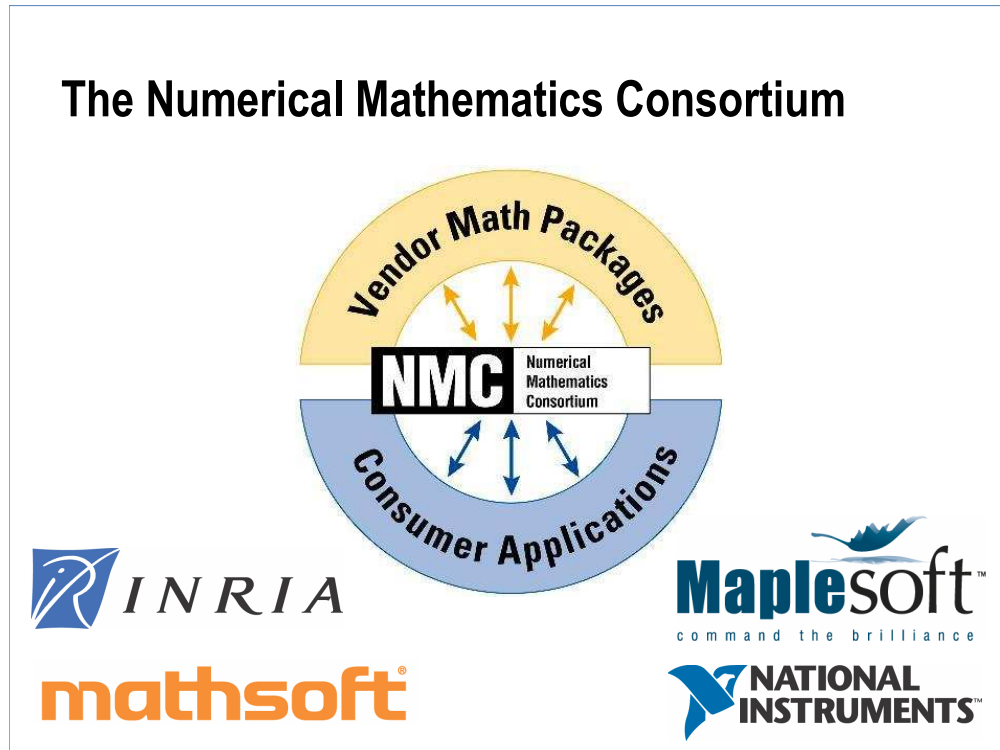
Numerical algorithms form the backbone of many technical pursuits. From advanced research projects to industrial applications, algorithms provide the brains to problem-solving engines. Many engineers, scientists, and researchers spend months, years, even entire careers developing and honing algorithms in efforts to provide breakthrough innovation or create a better world around us.

It is this huge investment in intellectual property that the Numerical Mathematics Consortium (NMC) seeks to preserve by standardizing on a core set of mathematical functions applicable to numerical algorithms. With so much value placed in numerical algorithms, it is critical to allow the work of these technical professionals to be maintained across platforms, tools, and environments, without spending a significant amount of valuable time in "porting" or re-developing them.

Scientists and engineers have a plethora of tools at their disposal to help streamline the process of creating, optimizing, visualizing, translating, or targeting key algorithms needed for their work. Whether analyzing how different materials dissipate heat, or controlling an automobile's engine performance with code running on an ECU, much of this work can leverage a basic set of numerical functions and libraries. However, each of the current numerical tools and platforms have created their own set, even for the most common and core functions, which makes it very difficult for users to move their algorithms from one tool or platform to another. With rapidly converging technologies in the products we use today, while combining electronics, mechanical, physical, and multi-physics elements, designers must be able to freely use tools that specialize in these different domains during the process of bringing their product to market. Without standard function definitions, engineers are trapped in their current tool or platform with no hope of leveraging the best aspects of each tool without long and arduous rewriting of their algorithms.

Differences in functional behavior makes things difficult.

The Numerical Mathematics Consortium



The Numerical Mathematics Consortium is a group of vendors, professors, and users committed to eliminating this unnecessary reworking of algorithms. By standardizing the semantics of functions, inputs and outputs will have the same meaning in any compliant application. While the syntax may change among compliant applications, a developer well versed in the syntax of multiple products will no longer have to concern himself with the details of standard parameter types and behavior.

Membership levels include full members, advisory, and associate membership.

NMC Events and Activities

- Events

- Joint Mathematics Meetings Exposition
(San Antonio, Jan 2006)

- Activities

- Group Forum Establishment
- Weekly Conference Calls
- Working Group Meeting
(Boston, July 2006)



www.nmconsortium.org

The NMC has been busy over the last year. Founding members MathSoft and National Instruments attended the recent Joint Mathematics Meeting in January 2006. We staffed a booth in the exposition area, establishing dialogs and gauging interest among the 4500 attendees of the meeting.

The group also engaged in activities to facilitate technical progress. First, we established a group forum to facilitate discussion. We used the Yahoo groups service and have created a group called numath. The group has proven a good venue for online discussion of technical topics, to record drafts, for group announcement and more. The group resembles a newsgroup in that we can post topics and respond to existing topics. A file area and calendar have also proven useful in posting drafts and scheduling.

Founding members have also initiated a weekly conference call to discuss issues that deserve a more interactive forum.

Finally, in July 2006, we met in Boston, MA for a face-to-face meeting. MathSoft graciously provided a venue for the meeting which included attendees from all four founding members. Members from Maplesoft, MathSoft, and National Instruments met on site, while member INRIA phoned in through a conference call. The July working group meeting was very productive and resulted in significant progress toward definition and agreement on technical topics and ratification of functions.

NMC Technical Progress

- Technical Progress
 - Technical Issues Resolved
 - Functions Ratified
 - Draft Standard Imminent
 - New Technical Issues Identified
- By exploring what it takes to port an algorithm, we can
 - Give you a practical progress report
 - Leave you with a practical understanding of the resolved issues



www.nmconsortium.org

The NMC has made technical progress that includes agreement on technical issues and ratification of functions.

For a meaningful review of this progress, we will now work through what it takes to port an example algorithm.

Example Algorithm

```
t := {-3, -2.9, ..., 2.9, 3}
u := GaussianWindow(Size(t))
v := {0, ..., 0, 1, 1, ..., 1, 1}
w := Convolve(u, v)
y := ErrorFct(t)
if w ~ y then ...
```



www.nmconsortium.org

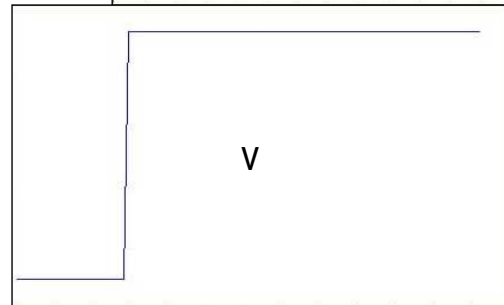
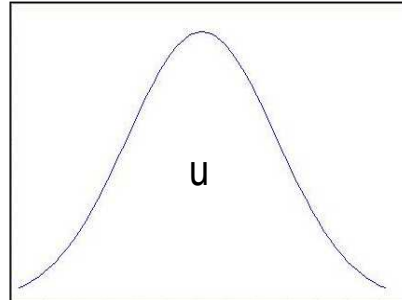
Porting an example algorithm

Here's an example of a numeric algorithm that generates scaled versions of the cumulative distribution function (CDF) for a normal distribution. The first version using convolution, a Gaussian window, and the a step function (given here in discrete vector form) computes the CDF by a filtering approach. The latter version uses the Error function (common notation 'erf') which encapsulates a scaled version of the CDF using a more direct mathematics approach.

While this CDF is grounded in statistics, forms of it are used in other application areas such as edge measurements in both signal and image domains. Here, we simply present a numeric algorithm that could exist in part in an existing application.

Example Algorithm

```
t := {-3, -2.9, ..., 2.9, 3}
u := GaussianWindow(Size(t))
v := {0, ..., 0, 1, 1, ..., 1, 1}
w := Convolve(u, v)
y := ErrorFct(t)
if w ~ y then ...
```

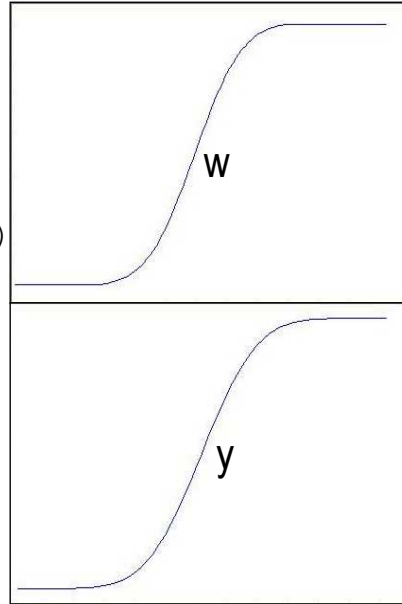


www.nmconsortium.org

This shows the graphic representation of the Gaussian and step functions.

Example Algorithm

```
t := {-3, -2.9, ..., 2.9, 3}
u := GaussianWindow(Size(t))
v := {0, ..., 0, 1, 1, ..., 1, 1}
w := Convolve(u, v)
y := ErrorFct(t)
if w ~ y then ...
```



www.nmconsortium.org

This shows the close relationship between each version of the CDF.

Algorithm Portability

- Functions
- Data Management
- Decision-making



www.nmconsortium.org

To properly handle algorithm portability, three main areas require standardization: functions, data management, and decision-making. Here's a breakdown of these areas:

- Functions: Domains, Behavior (Semantics)
- Data Management: Generation, Manipulation (Indexing, Reorganizing, etc)
- Decision-making: Conditionals, Looping (Programming constructs)

Initial focus is on defining functions. This allows us to:

- 1) Concentrate on the mathematics and build around it.
- 2) Limit the data manipulation to types of data required by functions.
- 3) Avoid the language-centric areas of standardization (such as looping) until the mathematical foundation is set.

To this day, the most universally used form of mathematics has been in the form of mathematical libraries (BLAS, LAPACK). This is due to the fact the users are solving the same mathematical problems. However, they generate these solutions in different ways based on the particular languages or development environments.

Algorithm Portability: Functions

- Definition
- Transformation(s)
- Compliance



www.nmconsortium.org

When focusing on functions, obviously a definition is required. Once you have function definitions, there has to be some declaration of compliance to make the standard applicable to (numerical) algorithm development. To facilitate 'porting' some infrastructure is required to map from one environment to another.

As stated, we started w/ definitions. In this process, we made decisions that dramatically affect these other areas.

What is in a Function Definition?

- Behavior (Semantics)
- Domain
 - Inputs
 - Outputs
- Example(s)
- References
- Related functions



www.nmconsortium.org

What's in a function definition? Well, the approach NMC has taken is to establish the behavior (semantics) of the function. This is different from specifying the syntax of the function and allows the definition to be applied to any number of algorithm development environments. In fact, these environments do not have to be part of the traditional floating point arenas.

What I mean by this is this approach is just as valid for definition math functionality in symbolic packages, arbitrary precision engines, as well as more vertically-oriented areas of numeric computation. For example, DSPs and FPGAs don't necessarily have the floating point support found on everyday desktop computers. Hence, numeric algorithms based on fixed-point mathematics don't always conform to the standards established by IEEE 754. However, the need for these math functions is just as important for these types of targets.

In addition to the behavior, the proper types of input and output data are required. This is fundamental to the definition because it adds clarity to the behavior. Defining a function on a data type for which it was not intended creates confusion for those who understand the basis for the function as well as the use of that function in practice. It also leads to an inconsistency in the function library making learning and retention more difficult.

Of course, examples are an additional way to show what the function is doing and the references provided give valuable links to educational material for better understanding of the function.

Function Definition: Convolution

```
t := {-3, -2.9, ..., 2.9, 3}
u := GaussianWindow(Size(t))
v := {0, ..., 0, 1, 1, ..., 1, 1}
w := Convolve(u, v)
y := ErrorFct(t)
if w ~ y then ...
```



www.nmconsortium.org

Let's examine our algorithm and look at a function the NMC has ratified recently. Convolution is a relatively basic operation in signal processing.

Let's consider the semantic definition of convolution.

However, it seems at first glance that getting to a proper semantic is trivial. Unfortunately, that's not the case.

Questions:

- 1) How many of you are familiar with the concept of convolution?
- 2) How many have written algorithms using it?

There are several possibilities....

Semantic Possibility #1

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t - \tau)d\tau$$

- Integral form
- How to implement?
- What are f and g ?



www.nmconsortium.org

Here's a classic definition of convolution using integration using functions 'f' and 'g'. How many people know how to implement this as an integral?

Expectation – not many. If there are hands, interact by asking some questions about their approach. Typically, integrals are implemented as summations of varying granularity – this leads to a better form!

While this form is perfectly valid mathematically, it doesn't help in the implementation of the function. Also, the inputs are significantly more generic than is typical – they are continuous functions defined over the entire real domain

Semantic Possibility #2

Convolution & Polynomials

$$p_c(x) = \sum_{i=0}^{N_c} c_i x^i$$

- Related operations

- Polynomial multiplication!

$$p_a(x)p_b(x) = \sum_{i=0}^{N_a * N_b} \left(\sum_{k=0}^i a_k b_{i-k} \right) x^i$$



www.nmconsortium.org

What about convolution using certain data types?

Questions:

- 1) Who knows what basic operation on polynomials is equivalent to convolution?

Expectation – some one may get this. Either way, make sure the relationship is valid for polynomials of finite order.

But, we have a problem defining it this way. In the world of polynomials, it's just multiplication – the name convolution would be redundant, so what's the difference?

Which Semantic?

Conclusions

- Implementation concerns

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)d\tau$$

- Valid domain data types

$$p_a(x)p_b(x) = \sum_{i=0}^{N_a * N_b} \left(\sum_{k=0}^i a_k b_{i-k} \right) x^i$$



www.nmconsortium.org

Conclusion #1

While many semantics are possible, choosing a semantic should help the end-user with an implementation if at all possible.

Conclusion #2

While the math is potentially the same across different domains, insure that the function is being defined on the data types for which it is intended.

The Semantic of Choice

$$(u * v)_i = \sum_k u_k v_{i-k}$$

- Implementation
 - Summation rather than an integral



www.nmconsortium.org

The preferred semantic is relatively straightforward and lends itself nicely to practical implementation. The ‘removal’ of the integral helps quite a bit here.

This semantic which shows the relationship between the resulting elements of the convolution and the signals used to generate it. The particular form given here is intended to support convolution on any range (duration) the signal may represent. However, the definition as it appears in the specification will have to deal with the boundary conditions to cover the behavior completely.

A Summation-based Semantic

$$(u * v)_i = \sum_k u_k v_{i-k}$$

- Domain data type is valid
 - It is an operation on a vector space.
 - But *not* a linear algebra operation!

$$\left(\sum_{k=0}^i a_k b_{i-k} \right)$$

$$p_a(x)p_b(x) = \sum_{i=0}^{N_a+N_b} \left(\sum_{k=0}^i a_k b_{i-k} \right) x^i$$



www.nmconsortium.org

If we look at the different semantics we've looked at, this 'reduction' of inputs to vectors makes sense because:

- 1) The operation is already called by another name when using polynomials, and
- 2) To see that convolution is an operation on a vector space, consider the semantic based on polynomials. Given a set of polynomials with coefficients in a field (F), then that set is a vector space over that field (F[x]). Multiplication is an operation on that (infinite) space.
- 3) While vectors are valid inputs here, they do not have orientation which is common for vectors used in a Linear Algebra sense.

In industry, math environments often have vectors with orientation to properly model linear algebra functionality. It is not common for a particular math environment to implement both types of vectors (those in vector spaces, and those in linear algebra with orientation – as matrices). That's because it would be very confusing for the user!

So, what about vector orientation?

Vector Orientation

- Not in the current semantic
- It's a system-level problem
- In this case:
 - Convolution has two inputs
 - Multiple orientations
 - Which is right?

$$(u * v)_i = \sum_k u_k v_{i-k}$$

```
...
w := Convolve(u, v)
...
```



www.nmconsortium.org

While vector orientation is not really part of convolution, we realized that, at the system level, this issue has to be dealt with to insure that the function definitions do not alienate these packages.

Notice that the semantic only uses single indexing. If orientation were part of the formula, two indices would be required per variable. This is the first of the higher-level technical topics we discovered while doing function definitions. It transcends functions or function groupings.

Handling it at the function level would:

- 1) Increase confusion (Why is this here when the math doesn't dictate it?)
- 2) Possibly lead to inconsistencies in dealing with orientation. (We've seen evidence of this in industry.)
- 3) Make it difficult for readers to understand the overall approach.

When looking back at the algorithm example, the convolution function call involves two vector inputs. What happens when one is a row vector and the other a column? That's the ambiguous case that needs to be handled consistently.

Resolution: Vector Orientation

“In general, the standard does not assert orientation on vectors in contexts outside linear algebra.

For those contexts, orientation, if present in the inputs to a function, is preserved whenever possible. When ambiguity arises, ‘column’ is the preferred orientation.”



www.nmconsortium.org

Simply put, outside linear algebra we don't enforce orientation – we don't even acknowledge it because it's not part of the mathematics. It is important to note that there are plenty of cases where vectors are used outside linear algebra. Convolution is only one of many examples.

NMC Function Definition: Convolution

Convolution
Class Vector Analysis
Semantic Computes the convolution of two vectors \mathbf{u} and \mathbf{v} , of sizes M and N respectively, defined by $(\mathbf{u} * \mathbf{v})_j = \sum_{k=i_0}^{i_0+M-1} u_k v_{j-k},$ for $j = i_0, i_0 + 1, \dots, i_0 + M + N - 2$ where $v_i = \begin{cases} v_i & i_0 \leq i < i_0 + N \\ 0 & \text{otherwise} \end{cases}, \text{ and}$ i_0 is the index of the first vector element.



www.nmconsortium.org

This slide shows a snapshot of the documentation associated with the convolution functional definition.

NMC Function Definition: Convolution (cont.)

Inputs

u : a real or complex vector
 v : a real or complex vector

Outputs

$u * v$: a real or complex vector

Example usages

Reference

Related functions

[Deconvolution](#), Fourier Transform



www.nmconsortium.org

Function Definition for Error

```
t := {-3, -2.9, ..., 2.9, 3}
u := GaussianWindow(Size(t))
v := {0, ..., 0, 1, 1, ..., 1, 1}
w := Convolve(u, v)
y := ErrorFct(t)
if w ~ y then ...
```



www.nmconsortium.org

The error function was also ratified by the NMC members. It's fundamental usage is in statistics but is also useful (in altered forms) to model filters as w/ our algorithm example.

Error Function Basics

- Widely used in statistics
- Related to Gaussian (normal) distribution
- Standard semantic

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

What's different here?

```
t := {-3, -2.9, ..., 2.9, 3}
...
y := ErrorFct(t)
...
```



www.nmconsortium.org

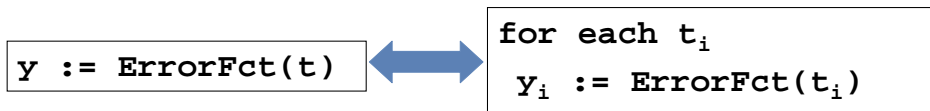
The semantic is standard. You may ask – Why is the use of integral OK here? Answer: This is a definite integral (not infinite).

You can look in Numerical Recipes in C for a sample implementation.

Given this definition, something in our example doesn't jive. The usage of the function is on a vector input, not a scalar value. Similar to the technical topic on vector orientation, here's another implementation-specific detail that can't be ignored. What is being exposed here is a process called vectorization.

Vectorization

- ‘Extension’ of a function
- Allows for ‘collections’ of inputs
- Applies the function iteratively



www.nmconsortium.org

Vectorization is one or more extensions to a function F whereby the set of inputs accepted by F is increased to include collections of the supported data types. When a collection is input to F , F operations on it iteratively as if it were passed the elements of the collection one at a time.

For example, let's take the error function, **erf**, which is defined on scalars. It is possible to define a version of **erf** which accepts a vector. This extension would apply **erf** to every element of the vector. Each element operation is defined by the scalar version of **erf**.

Seems like a slam dunk – why not just include it...

Problems associated with Vectorization

- No mathematical basis
- Multiple arguments → combinatorial explosion
- Creates ambiguities with functional calculus

Example: exponential function (e^x)

$$e^A = \sum_{n=0}^{\infty} \frac{A^n}{n!}, \text{ for } A_{m \times m} \quad \text{vs} \quad e^A = \begin{bmatrix} e^{A_{11}} & \dots & e^{A_{1n}} \\ \dots & \dots & \dots \\ e^{A_{m1}} & \dots & e^{A_{mn}} \end{bmatrix}, \text{ for } A_{m \times n}$$



www.nmconsortium.org

While vectorization can lead to some efficiencies, it also has no basis in mathematics. It is purely an implementation detail grounded in the syntax of the environments language and execution engine.

When vectorization is applied to function of more than one argument, several ways to vectorize are possible. These options explode if each of the arguments can be of different types (e.g. scalar and vector/matrix). This makes it extremely difficult to be consistent when applying this to most if not all functions.

Perhaps the largest problem involves collisions in the functional calculus. This means that function may already have a mathematical definition for an input argument that could be considered a collection of inputs. The exponential function is a great example of this. It is defined for scalars. That definition is also perfectly valid on matrix inputs as long as the matrices are square. However, if vectorization is employed, it is unclear which version of the function is to be executed on matrix inputs.

Resolution: Vectorization

“In some contexts it makes sense for a function F to map onto aggregate structures (e.g. scalar function to the elements of a vector). This results in a separate* definition.

Similarly, if function F is defined on an aggregate structure in a functional calculus sense, then a separate* definition is specified.

Both versions should reduce to F in the case that the argument is a scalar.”



www.nmconsortium.org

The resolution

- 1) When it makes sense to ‘extend’ a function using vectorization, then that extension is defined separate from the base function.
- 2) If the function is also application mathematically to more complicated structures, then that extension is also defined using a separate definition.
- 3) Consistency dictates that either of these extensions should be equivalent to base function when an scalar value is input (e.g. vectorization of one element or mathematical computation of a 1×1 matrix).

Because these function extensions are given in individual definitions, a naming scheme is preferred so that it is easier to tell the base function from which they are derived.

*Separate definitions are crucial for portability. Combining these would force an environment that wishes to be compatible with this standard to implement the combined functionality. We will not force end-users to implement vectorization. Any functions based on vectorization are always optional.

NOTE: This affects the area of transformations which is a key component to portability of functions.

Which Domain for the Error Function?

- Domain (Value) Considerations
 - Is it universally used?
 - Does it take ‘extreme’ expertise?
- Error function is used in statistics
 - Real input values are common

New Technical Issue: Which Domain?



www.nmconsortium.org

Vectorization is not the only higher-level technical topic that the error function exposes.

By domain here, we are not talking about the domain ‘type’ (e.g. scalar or vector) as before. This time it has to do with the range of the input value(s) to support. Because this function is used primarily in statistics, requiring support for the complex domain seems unusually harsh given that it is much harder to implement and not used across several math disciplines. Up pops a new issue: which domain should a function support?

Which Domain?

- Varies from function to function
- Boundaries differ significantly
- Usage & implementation difficulty considered
- Like vectorization, guidance is needed



www.nmconsortium.org

Just as w/ vectorization, it is important to provide guidance in this area. If the standard is to be used broadly, it has to be able to scale to the common use cases that exist in industry.

When the domain of a function is considered, it has to be handled on a case-by-case basis because each function is used differently. In the case of the error function, the entire real domain is important. For functions such as the (complete) gamma function, the positive real values are the most common domain to support and use.

At all times, the universal usage and difficulty of implementation is considered when choosing which domain to support. The unfortunate part is that the math is equally valid even when a domain is hard to implement or not exercised that often. It seems a shame to throw away the definition for those regions.

The Solution is Compliance!



- Two levels
 - Required: must implement
 - Optional: if implemented, must follow definition
- Applies to functions (vectorization)
- Applies to domains (e.g. real vs. complex)
- Verification/validation is treated separately
- ‘Existence’ condition is another issue



www.nmconsortium.org

The solution is establishing what compliance means in the standard in relation to functions and their definitions. The two levels work at both the function and domain levels. The required level means an implementation must exist. The optional level covers additional functionality. If it is implemented, then it must follow the standard’s definition(s).

In the case of vectorization, the functions which are extensions to the base functions are always optional. However, the extensions which are defined in mathematics (i.e. on the functional calculus) are decided on a function basis. It is unlikely that many will be required.

This is great news for the domains because many definitions are perfectly valid over the complex domain. This allows the standard to specify consistent behavior between the real and complex domains but not require an implementation for the latter.

For many, the idea of compliance also includes verification and validation. Because this area is so important, it is handled separately. In essence, consider the compliance condition used here to define what must/may exist in order for a product to adhere to the NMC standard.

Summary of Resolved NMC Technical Topics

- Semantic guidelines
- Vector orientation
- Vectorization
- Domain guidelines
- Compliance levels
- Handling univariate & multivariate functions
- Organizing related functions



www.nmconsortium.org

In the process of ratifying functions, the NMC came to agreement on several technical topics that will make it easier to ratify functions in the future.

This includes:

- Semantic guidelines: implementation help and proper use of domain data types
- Vector orientation: a system-level handling of orientation for non-linear algebra functions
- Vectorization: function extensions that act as a built-in iterators
- Domain guidelines: using common usage and difficult implementation to dictate supported regions
- Compliance levels: gives flexibility to define a consistent math model w/out forcing an implementation

We didn't have a chance to cover the following, but they were ratified:

- Handling univariate & multivariate functions
- Organizing related functions

Summary of Ratified NMC Functions

- Convolution
- Deconvolution
- Polynomial Derivative
- Polynomial Integral
- Error
- Airy (1st Kind)
- Airy (2nd Kind)
- Airy (Derivative of 1st Kind)
- Airy (Derivative of 2nd Kind)
- Gamma



www.nmconsortium.org

These function were ratified by NMC. Both the technical topics and functions will show up in a new revision of the specification draft by Sept 19.

What's Next?

- Upcoming revision of standard
- Continued technical discussions
 - Increased function ratification rate
 - Function examples (MathML / OpenMath)
 - Undefined behavior (IEEE 754)
- Advisory / associate membership involvement
- Expect biannual progress updates



www.nmconsortium.org

With this upcoming revision of the standard we have a handle on some of the technical issues that delay ratification of functions. Expect improved ratification rate. Standard lays out a framework for how to proceed. Also, we've also improved our processes.

This also lays out groundwork for how associate and advisory members can become more involved.

We are looking to other standards for consistency and guidance?

What should happen when people enter wrong values. IEEE 754 includes NaN definition, we are looking to existing standards for guidance and consistency with what's being done.

The Numerical Mathematics Consortium

<http://numath.org>

