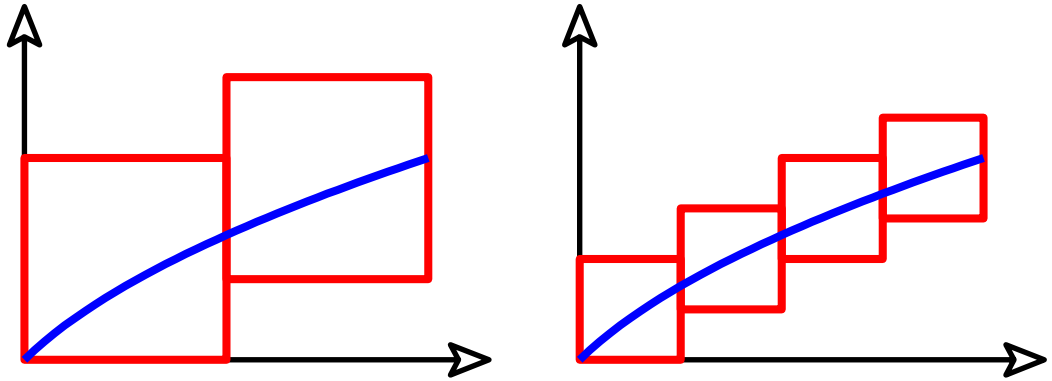
Towards an Interval Subroutine Library

George F. Corliss
Electrical & Computer Engineering
Marquette University
George.Corliss@Marquette.edu
with R. Baker Kearfott, Ned Nedialkov,
John Pryce, Spencer Smith

Outline:

- Interval subroutine library
- Mission
- Product
- Organization
- Partners

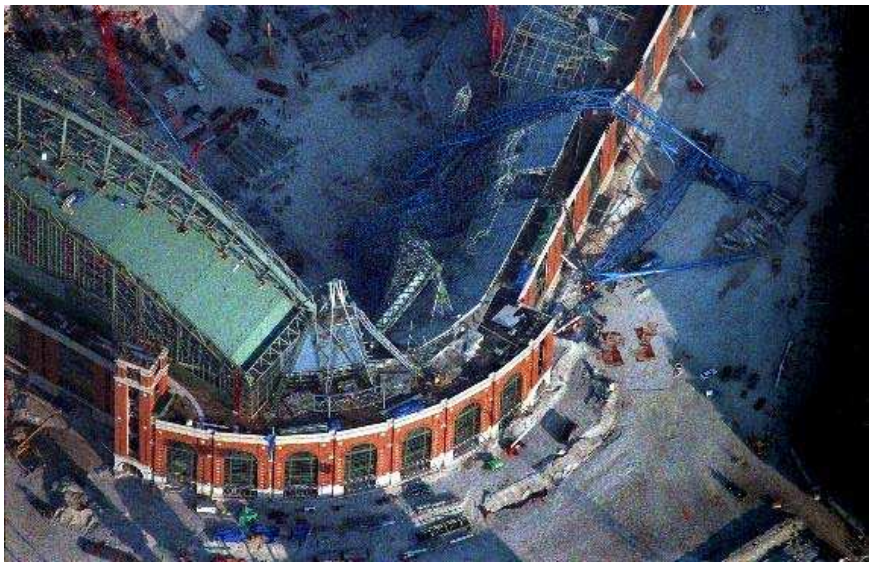


Presented at NSF Workshop on Reliable Engineering Computing
Georgia Tech Savannah, 22 – 14 February 2006
Slides: http://www.eng.mu.edu/corlissg/06Savannah

Interval Subroutine Library?

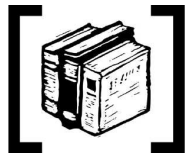
Suppose you believe advantages of interval methods include

- Guarantee "Thou shalt not lie"
- Enclose modeling, truncation, & roundoff errors
- Handle uncertain parameters
- Validate existence and uniqueness of solutions
- Highly reliable scientific and engineering computation



Now what?

Miller Park crane collapse July 14, 1999. Milwaukee Journal / Sentinal, http://www.jsonline.com/news/metro/jul99/mpgallery71499.asp



Production Quality Interval Packages

Library
Packages
Prob-solving
ISL project
Product
Organization
Partners
Conclusions

- INTLAB - INTerval LABoratory from TU Hamburg-Harburg, http://www.ti3.tu-harburg.de/english/index.html
- PROFIL/BIAS for Matlab from TU Hamburg-Harburg, http://www.ti3.tu-harburg.de/Software/PROFILEnglisch.html
- Fortran 95 and C++ from Sun Microsystems, http://wwws.sun.com/software/sundev/suncc/index.html
- C-XSC, Pascal-XSC packages from TU Karlsruhe, http://www.uni-karlsruhe.de/~iam/html/language/xsc-sprachen.html
- FI_LIB and FILIB++, Hofschuster, Krämer, et al., Bergische Universität Wuppertal, http://www.math.uni-wuppertal.de/wrswt/software/filib.html
- BOOST, www.boost.org. Look at [Libraries], then [Math and numerics], then [Intervals], or http://www.boost.org/libs/numeric/interval/doc/interval.htm
- Maple has range arithmetic, http://www.maplesoft.com
- Mathematica has RealInterval arithmetic, http://www.wri.com
- COSY from Michigan State University, http://cosy.pa.msu.edu

See also *Languages for Interval Analysis*, http://www.cs.utep.edu/interval-comp/intlang.html

Problem-Solving Routines

Software for chapters in your numerical analysis text?

- Linear systems
- Nonlinear root finding
- Bounding ranges of functions
- Global optimization
- Quadrature
- Ordinary differential equations
- Partial differential equations
- Statistics
- ...

Research quality codes

No "Here. Use this CD"

Interval Subroutine Library

Mission: We propose the development of a full-featured, production quality library of validating routines for use by the wide community of applications developers.

Steering Committee includes:

George F. Corliss	Marquette University
John D. Pryce	Cranfield University, RMCS Shrivenham
R. Baker Kearfott	University of Louisiana at Lafayette
Spencer Smith	McMaster University
Ned Nedialkov	McMaster University and Lawrence Livermore National Laboratory

Success? Four mutually incompatible criteria:

Quality (30%): Requirements of customers include "Thou shalt not lie," easy of use at many levels, API that fits needs, etc.
Timeliness (30%): The sooner the better
Coverage (30%): The more the better
Performance (30%): Tightness and speed

Issues to Be Considered Include

1. Mission – What is our goal?
2. Product – What will we deliver?
3. Organization and process - How will we do that?
4. Partners – How can other people help?

Current status

- URL: http://homepage.ntlworld.com/j.d.pryce/isloct05/
- Pryce EPSRC grant
- NAG & Sun letters of support
- October 2005 kick-off workshop in Shrivenham
- Commonality/Variability analysis toward BIAS
- C++ interval standard proposed by Brönnimann, Melquiond, & Pion
- Workshops in February & May/June 2006

C++ Interval Standard

Herve Brönnimann, Guillaume Melquiond, and Sylvain Pion,
A Proposal to Add Interval Arithmetic to the C++ Standard Library.
(2005). Doc. no. N1843-05-0103.

Why standardize interval arithmetic in C++?

- Functionality is needed in many areas
- Many existing implementations
- All with different design choices
- Basic version is not hard to implement with standard components
- No need for auxiliary libraries
- Opportunity for more optimized implementations

Prototype implementation and example programs at
http://www-sop.inria.fr/geometrica/team/Sylvain.Pion/cxx

C++ Interval Standard - Advantages

Provides a level 0 basic interval arithmetic:

- Does not contain high-level algorithms
- Provides only basic operations on intervals defined by native floating-point numbers
- Standardized interface means that interval arithmetic is available to anybody with a C++ compiler
- Interval arithmetic is pushed to users, instead of waiting for them to hear about it
- Incentive for designers of compilers (and processors) to improve generated code for interval operations

C++ Interval Standard - Need

To promote adoption of proposed standard

- Comment on the proposal
- Convince the committee (composed of companies developing compilers and STL implementations) there are customers
- Testimony / use case / numbers demonstrating these companies would be **wise** to invest in intervals now

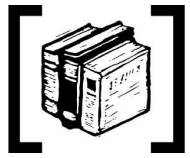
Contact

- Guillaume Melquiond, guillaume.melquiond@ens-lyon.fr
- Herve Bronnimann, hbr@poly.edu
- Sylvain Pion, Sylvain.Pion@sophia.inria.fr
- Standardization bodies to voice their support to this proposal

Library – What is our goal?

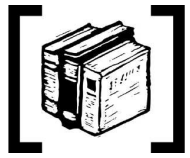
Mission: We propose the development of a full-featured, production quality library of validating routines for use by the wide community of applications developers.

1. Problem? No CD
2. Target market?
   Applications developers?
3. Models? NAG, IMSL, Open source,
   BLAS, LAPACK, ...?
4. Scope? Numerical Recipes,
   early NAG or IMSL?
5. Name? Interval Subroutine Library?



Product – What will we deliver?

1. Requirements? Model of intervals
2. Platforms? All?
3. Language? C++, Matlab, Fortran, Java, C#, ...
4. API - External architecture? See Kearfott. Some chapters:
   (a) Basic Interval Arithmetic Subroutines
   (b) Automatic differentiation
   (c) Taylor model arithmetic
   (d) Constraint propagation
   (e) Linear systems
   (f) Nonlinear systems
   (g) Optimization
   (h) Quadrature
   (i) Statistics
   (j) Ordinary differential equations
   (k) Partial differential equations
   (l) ...
5. Documentation
6. ...



Requirements of Validated Calculation

Principle 1. *Interval arithmetic should be founded on standard set theory and real analysis*

Principle 2. *An interval* $[a, b] = \{ x : a \leq x \leq b \}$ *is a particular kind of subset of the number system, chosen because easy to* **represent** *and* **manipulate**

Principle 3 (Thou shalt not lie).
There is but one requirement for interval codes: enclose what you claim to enclose. All else are quality of implementation (QOI) issues

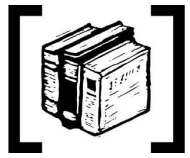
Principle 4. *To compute (interval) enclosures of the range for arbitrary explicit functions, it suffices to implement them for the elementary functions*

Meaning of **implement**?

Neglected design question: what **abstract interval model**?

On ideal machine without roundoff

- What class $\mathcal{I}$ of intervals to represent?
- How define elementary functions, in particular in exceptional cases?



Requirements – Cset model

By Halster and Pryce

Priority model for interval implementations to support:

(a) Number system is $\mathbb{R}^*$ — hence $\pm\infty$ are "first class"

$\mathcal{I}$ is all closed intervals $[a, b]$ with $a, b \in \mathbb{R}^*$, $a \leq b$, together with $\emptyset$

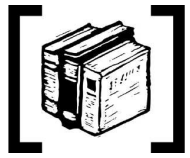
(b) For $X, Y \in \mathcal{I}$ and $\bullet$ a BAO,

$$X \bullet Y \ =_{\text{def}} \ \text{smallest interval in } \mathcal{I} \text{ that contains cset}(\bullet; X, Y),$$

see below. Similarly for other elementary functions

Loose evaluation is used — with a flag

Supported by **Sun**'s compilers and option in **FILIB++**



Cset operations (on arbitrary values)

Cset def. For $z = x \bullet y$, cset value at (x, y) is set of **all possible** limits of z_r where $z_r = x_r \bullet y_r$ is defined in the normal $\mathbb{R}$-sense for all r, and $x_r \to x$, $y_r \to y$

One-point inputs can give many-point result

Cset for set inputs, $X \bullet Y$, is the union of the csets for points $x \in X$, $y \in Y$

For $f(x, y, \ldots)$ of any number of arguments, cset is defined analogously

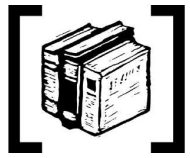
Notation. $\mathbf{cset}(f; x, y, \ldots)$ or $f^*(x, y, \ldots)$ but only when necessary

For basic arithmetic operations, this gives new results

$$x/0 = \{-\infty, +\infty\}\,(x \neq 0), \quad +\infty/+\infty = [0, +\infty],$$
$$(+\infty) + (-\infty) = 0 \times (+\infty) = 0/0 = \mathbb{R}^*$$

Also adds extra values where a function is finite but discontinuous, e.g.

$$\text{sign}^*(0) = \{-1, 0, 1\}$$



Features of Csets

Principle 5. *Cset always $\supseteq$ traditional (exact) range, and equals it when function is continuous at each input point*

Theorem 1 (Fundamental Cset Theorem). *Let each* **elementary** *function be given an extended version that computes an [interval] enclosure of its cset over arbitrary [interval] inputs.*
Then evaluating an **arbitrary** *explicit function $\mathbf{y} = \mathbf{f}(\mathbf{x})$, using these extended elementary functions, yields an [interval] enclosure of the cset of $\mathbf{f}$ over arbitrary [interval] inputs.*

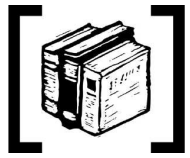
From implementer's viewpoint, Theorem says:

Principle 6. *To compute [interval] enclosures of cset for arbitrary functions it suffices to implement them for elementary functions*

Principle 7. *Let program P run without exceptions on data D using "traditional" interval library L_T and give output O. Then it gives identical output O if L_T is replaced by a cset interval library L_C*

Principle 8 (Rearrangement Theorem). *Cset of any rational function $r(x_1, \ldots, x_n)$ is unchanged by rearranging to an algebraically equivalent form*

Lets clever compilers and humans rearrange cset code for tighter enclosures



Rearrangement example

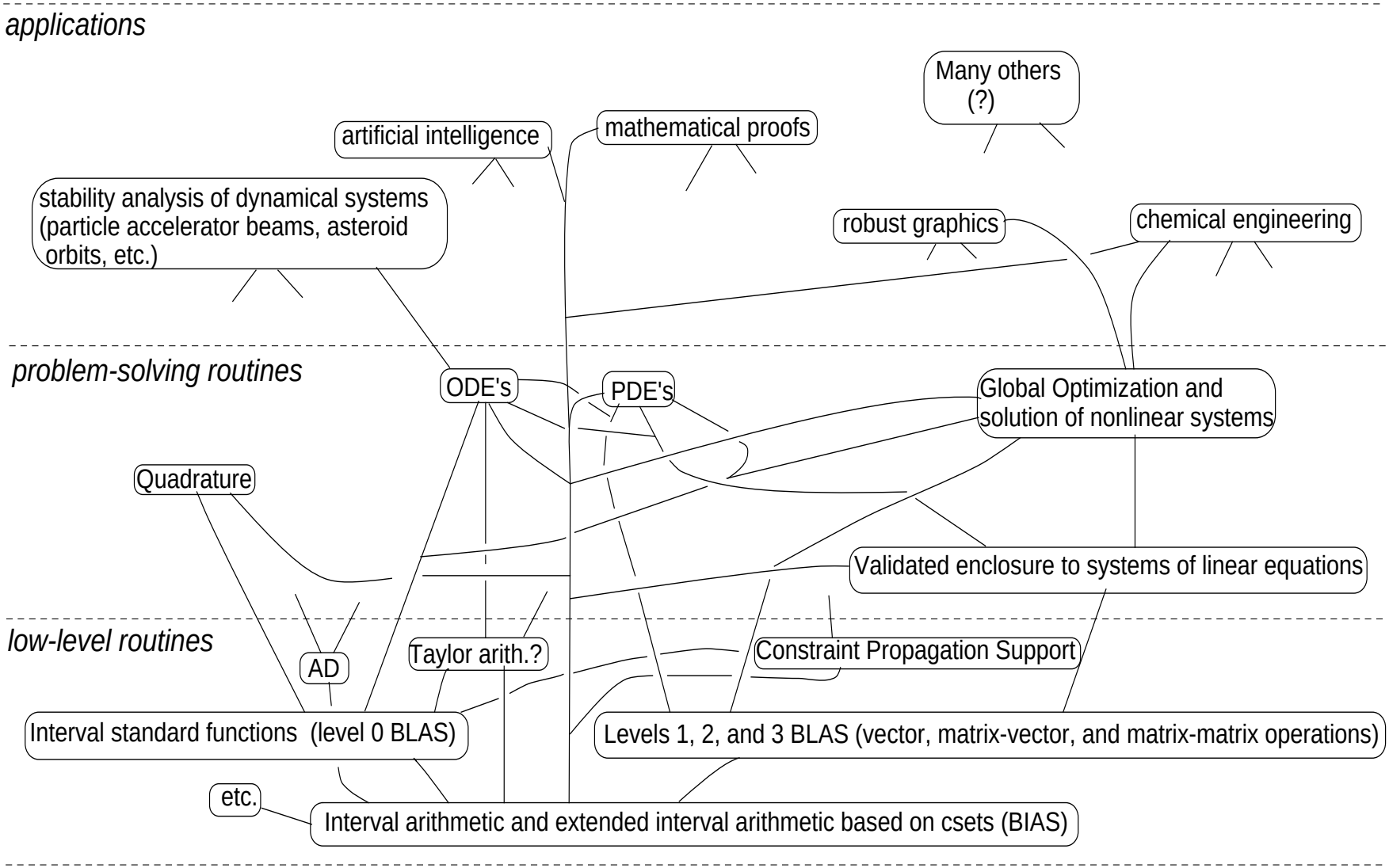
$$f(x,y) = \frac{x^2}{(x^2+y^2)} \text{ has same cset as } g(x,y) = \frac{1}{(1+(y/x)^2)}$$

Use Matlab with "short" display option, with S. Rump's INTLAB and Pryce's trial cset system

	$X = [-1,2], Y = [0,2]$		$X = [1,3], Y = [0,2]$	
	$f(X,Y)$	$g(X,Y)$	$f(X,Y)$	$g(X,Y)$
INTLAB	[NaN, NaN]	[NaN, NaN]	[0.0769, 9.0000]	**[0.1999, 1.0000]**
Cset	$[-\infty, +\infty]$	**[0, 1]**	[0.0769, 9.0000]	**[0.1999, 1.0000]**
	INTLAB can't handle $\div 0$; exact cset is [0, 1]		no $\div 0$; on both systems g returns correctly rounded exact cset	

Cset results show g gives far better enclosures than f

Tentative Hierarchical Structure



Product – Issues

Some additional issues to settle include

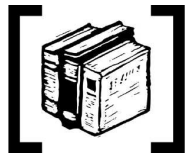
1. Quality assurance and testing
2. Rigorous input/output
3. Parallel
4. Documentation
5. Examples
6. Source vs. executable
7. License?
8. ...

Product – License

1. Protect rights of those whose software we use
2. Protect authors' reputations:
   (a) If you change ISL, you must call it something else?
   (b) Or may not further distribute
   (c) Or note that you changed
   (d) vs. Sven Hammerling on LAPACK vendor improvements
3. Commercialization is good
   (a) NAG, Sun, or others might commercialize
   (b) Free version remains available
   (c) Our rights are unrestricted

Gather examples including

1. Open Source licenses
2. GPL, Modified Berkley, Sun
3. of various interval packages
4. of each university



Organization and Process

How will we do that?

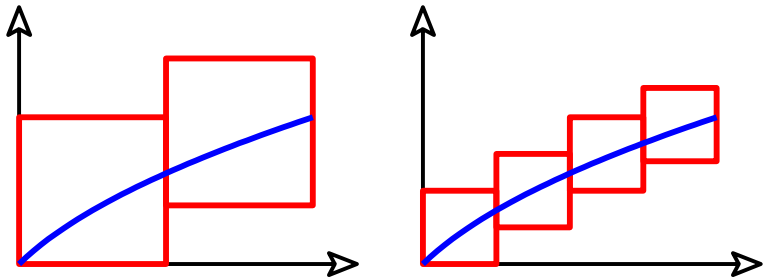
1. Who?
2. Organization?
3. Development tools/host?
4. Internal architecture?
5. Quality assurance?
6. Technology transfer?
7. Publication?
8. Funding?
9. Marketing and recruiting
10. ...

Partners – How can you help?

1. Pilot as applications developers?
2. Standards?
3. Contribute software?
4. Advise on needs?
5. Review and evaluate?
6. Advise on process?
7. ...

Specifically?

1. Experience with interval tools?
2. csets?
3. Possibly vs. certainly operators?
4. Contribute components?
5. Components you need?
6. IBLAS?
7. API advice?
8. QA advice?
9. Documentation advice?
10. ...



As an Applications Developer

How can you help?

As an Applications Developer

How can you help?

Actually, we hope ISL helps you

What would that take?

Perhaps

- Portable basic interval library
-
-

As a Contributing Author

How can you help? Many of your routines might be reused by others

Perhaps

-
-

You might contribute

- Code unit for the library to solve some well-defined problem of scientific computing, e.g., constraint propagation, linear systems, optimization;
- Functionality or performance improvements, corrections, or extensions to an existing unit of the library;
- Test suites;
- Documentation;
- etc.

As a Referee

How can you help?

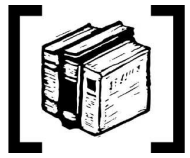
ISL refereeing process may be modeled on the process for refereeing ACM Algorithms

Review materials submitted by contributing authors including source; installation instructions; documentation of the problem, description of algorithms, examples of use, references, etc.; acceptance and other tests.

Assessing the library materials as they affect application developers who use the library, rather than the more academic concerns of a traditional journal referee

You are applications developers
Who better to assess how ISL contributions help you?

Refereeing may lead to publishable comparative testing



Conclusions

1. We envision a comprehensive, universally used library[a]
   Hope to go beyond previous efforts
2. Eventually, problem solving routines from all areas of numerical analysis
3. Promoting standardization, portability, and re-use
4. Better software engineering practive in scientific computing
5. To attain these goals with bounded resources:
   - Cooperation, discussion, consensus, and participation within our professional community
   - Participation from professional software vendors, such as NAG.
   - Support from home institutions and granting agencies (travel grants, release time, etc.)

Quality, comprehensive libraries are not compiled by a single person or small group of people over a short time.

[a] Contrast to offering general languages, such as in the COCONUT project or GAMS, or offering graphical user interfaces such as in various commercial packages