The Forthcoming IEEE1788 Standard for Interval Arithmetic

John Pryce

School of Mathematics, Cardiff University
smajdp1@cardiff.ac.uk

16th GAMM-IMACS symposium on
Scientific Computing, Computer Arithmetic and Validated Numerics
21–26 September 2014
Würzburg, Germany

Outline

What IA is and does

- Interval Arithmetic (IA) implements "validated" (="verified") numerics—it can enclose solution components x of a problem in an interval, i.e. between lower and upper bounds $x \in \mathbf{x} = [\underline{x}, \overline{x}] = \{ t \in \mathbb{R} \mid \underline{x} \le t \le \overline{x} \}$, even in finite-precision arithmetic.
- E.g. it makes Brouwer's fixed point theorem:

  If $K \subset \mathbb{R}^n$ is compact convex, and function f is everywhere defined & continuous on K, and $f(K) \subseteq K$, then f has a fixpoint in K

  constructive in the sense that sufficient conditions for "everywhere defined & continuous" can be found while computing f.
- IA's history: back to Archimedes (?) but mostly 20th century: Sunaga (Japan), Rall (USA), *et al.*
  Modern theory R. Moore (1966), e.g. validated ODE solver.
- Current significant validated software exists for: global optimisation; large sparse linear systems; particle beam design for LHC, ...

The basic idea

- Interval operations take all combinations of points in the inputs, i.e.
  $$\mathbf{x} \bullet \mathbf{y} = \{\, x \bullet y \mid x \in \mathbf{x} \text{ and } y \in \mathbf{y} \,\}, \quad \text{where } \bullet \text{ is one of } \{+ \;\; - \;\; \times \; \div\}$$
  For $\div$ don't allow $0 \in \mathbf{y}$ for now. In finite precision round outward.
- Fundamental Theorem of Interval Arithmetic
  If function $f(x_1, \ldots, x_n)$, defined by an expression, is evaluated with interval operations on interval inputs to get $\mathbf{y} = \mathbf{f}(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ then
  $$\mathbf{y} \supseteq \text{range of } f \text{ over box } \mathbf{x}_1 \times \cdots \times \mathbf{x}_n \text{ in } \mathbb{R}^n.$$
- E.g. $f(x_1, x_2) = x_1 + \dfrac{x_2}{x_1}$; 2-digit decimal arith; $\mathbf{x}_1 = [3, 4]$, $\mathbf{x}_2 = [3, 5]$:

$$\mathbf{x}_1 + \frac{\mathbf{x}_2}{\mathbf{x}_1} = [3,4] + \frac{[3,5]}{[3,4]} = [3,4] + \left[\frac{3}{4}, \frac{5}{3}\right] \xrightarrow{\text{round}} [3,4] + [.75, 1.7]$$
$$= [3.75, 5.7] \xrightarrow{\text{round}} [3.7, 5.7] = \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{y}.$$

$\mathbf{y}$ *does* contain the range of f over $[3,4] \times [3,5]$, which is $[4, 5\frac{1}{4}]$.

Outline

Why do intervals need new algorithms?

Example: Newton's method for solving a 1-D nonlinear system.

Why a specific iteration for the interval case? Usual formula:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Direct interval transposition:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{f}(\mathbf{x}_k)}{\mathbf{f}'(\mathbf{x}_k)}$$

($\mathbf{f}, \mathbf{f}'$ = interval versions of f, f', see last slide.)

Width of the resulting interval:

$$w(\mathbf{x}_{k+1}) = w(\mathbf{x}_k) + w\left(\frac{\mathbf{f}(\mathbf{x}_k)}{\mathbf{f}'(\mathbf{x}_k)}\right) > w(\mathbf{x}_k)$$

Divergence!

Back to basic theory

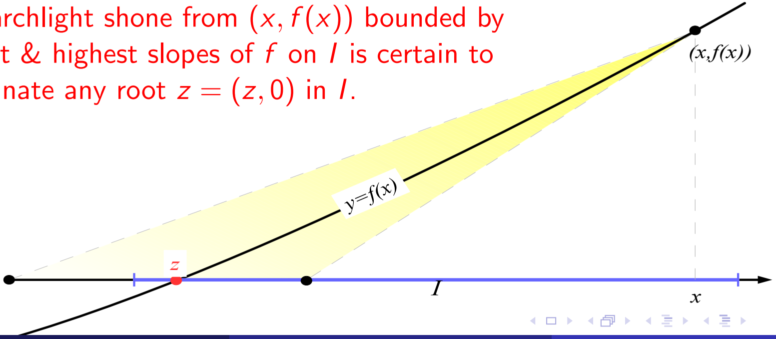
Let f be C^1 function on interval I.
By Mean Value Theorem MVT, $\forall$ root $z \in I$, $\forall x \in I$, $\exists \xi \in I$ s.t.

$$f(x) = f(x) - f(z) = (x - z)f'(\xi) \tag{1}$$

so provided $f'(\xi) \neq 0$, see later,

$$z = x - \frac{f(x)}{f'(\xi)}. \tag{2}$$

A searchlight shone from $(x, f(x))$ bounded by lowest & highest slopes of f on I is certain to illuminate any root $z = (z, 0)$ in I.



Computable version

When computing $x - f(x)/f'(\xi)$

- x is "point". Arbitrary in I ($\forall$), typically midpoint.
- $f(x)$ must be "interval", as f is code, liable to roundoff.
- $f'(\xi)$ must be "interval", as (a) f' is code, (b) $\xi \in I$ is uncertain ($\exists$).

So ($\forall$) if any root $z \in I$ then also

$$z \in \left(x - \frac{[f(x)]}{[f'(\xi)]} \right)$$

[...] meaning "some interval containing"

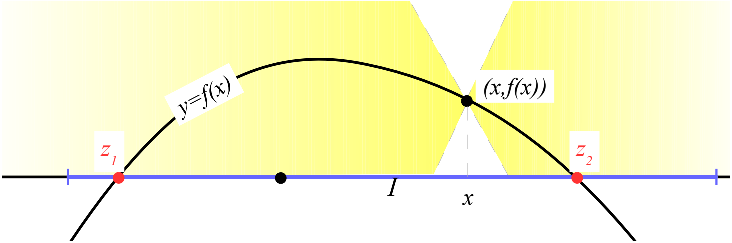
or in more current notation, renaming I as $\mathbf{x}$

$$z \in \left(x - \frac{\mathbf{f}([x])}{\mathbf{f}'(\mathbf{x})} \right) = \left(\text{point} - \frac{\text{interval function of point}}{\text{interval function of interval}} \right)$$

where $[x]$ is 1-point interval $\{x\}$ and $\mathbf{f}, \mathbf{f}'$ are interval versions of f, f'.

More general picture

- Actually searchlight shines in both directions, crucial when range of slopes includes $+$ and $-$ values:



- . . . provided one interprets $\div$ as reverse multiplication

$$c/b = (\text{any solution of } bx = c), \qquad \text{P1788's } \texttt{mulRev(b,c)}.$$

So $0/0$ means "whole real line" instead of "undefined".

- Now we enclose all roots even when many exist! Note searchlight can split I into 2 pieces.

Interval Newton iteration

(Hansen–Greenberg 1983; Kearfott & many others since)

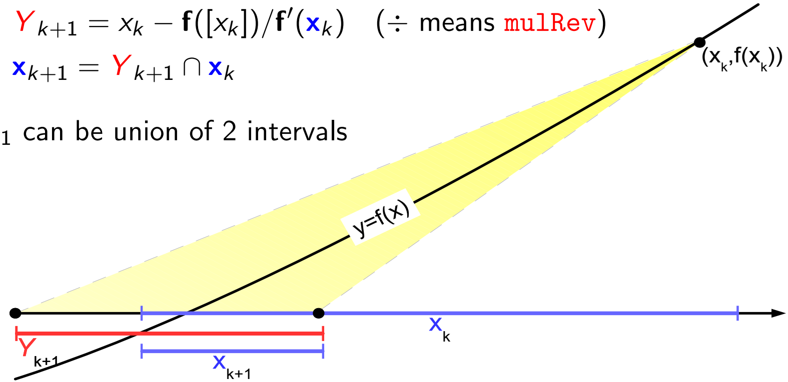
Set $\mathbf{x}_0$ = initial interval I
For $k = 0, 1, 2, \ldots$

x_k = some chosen point in $\mathbf{x}_k$

$Y_{k+1} = x_k - \mathbf{f}([x_k])/\mathbf{f}'(\mathbf{x}_k)$ ($\div$ means `mulRev`)

$\mathbf{x}_{k+1} = Y_{k+1} \cap \mathbf{x}_k$

Y_{k+1} can be union of 2 intervals



Comments

On the algorithm

This method guarantees to enclose all roots, but "2-way searchlight" case splits $\mathbf{x}_k$ in two, producing a possible tree of computations.

Features, assuming f is C^1 on initial interval:

- $\mathbf{x}_{k+1} = \emptyset$ guarantees $\nexists$ root in $\mathbf{x}_k$.
- If $0 \notin \mathbf{f}'(\mathbf{x}_k)$, $\exists$ at most one root in $\mathbf{x}_k$ (which must be in $\mathbf{x}_{k+1}$).
- Less obvious, if Y_{k+1} is $\neq \emptyset$, bounded, $\subseteq \mathbf{x}_k$, $\exists$ just one root in $\mathbf{x}_k$.

Comments

On the algorithm

This method guarantees to enclose all roots, but "2-way searchlight" case splits $\mathbf{x}_k$ in two, producing a possible tree of computations.

Features, assuming f is C^1 on initial interval:

- $\mathbf{x}_{k+1} = \emptyset$ guarantees $\nexists$ root in $\mathbf{x}_k$.
- If $0 \notin \mathbf{f}'(\mathbf{x}_k)$, $\exists$ at most one root in $\mathbf{x}_k$ (which must be in $\mathbf{x}_{k+1}$).
- Less obvious, if Y_{k+1} is $\neq \emptyset$, bounded, $\subseteq \mathbf{x}_k$, $\exists$ just one root in $\mathbf{x}_k$.

What does it show about the interval mindset?

- This analysis wasn't rocket science, just a careful look at the $\forall$, $\exists$ in a use of the MVT.
- But in general, seeing how mathematics converts to interval algorithms takes time and practice.

Outline

The need for a standard

- Dozens of excellent interval software packages have been written, with not quite compatible math foundations:
  - Support unbounded intervals and the empty set? Moore IA didn't.
  - Is an interval a set of numbers? Kaucher IA has intervals like $[4, 3]$.
  - How to handle $\sqrt{[-2, 2]}$, or $\mathbf{x}/\mathbf{y}$ when $0 \in \mathbf{y}$?

  ...as well as different software interfaces.
- Currently one can't write algorithms that are portable at a mathematical level, let alone portable software.
- At Dagstuhl, Germany (Jan '08) a project was started, which became IEEE Working Group P1788 "A standard for interval arithmetic".
- Officers: chair, vice-chair, technical editor (me), co-editors, web master, secretary/archivist, voting tabulator. $\sim$ 45 voting members.
- We have (May '14) voted to approve a final document, and (Aug '14) initiated IEEE "sponsor ballot" stage.

Outline

Definition of an interval

- In the current standard
  - An interval **x** is a set of numbers.
  - $\pm\infty$ not allowed as members of **x**, so intervals are subsets of $\mathbb{R}$.
  - Open/half-open intervals not allowed, but unbounded intervals are.
  - Empty set is an interval.

  So interval means topologically closed and connected subset of $\mathbb{R}$.
- There is a framework—so called *flavors*—to support alternative mathematical foundations, such as Kaucher IA in which an interval is an ordered pair $(\underline{x}, \overline{x})$ with $\underline{x}, \overline{x} \in \mathbb{R}$:

$$(\underline{x}, \overline{x}) \text{ “means” } \begin{cases} \text{set } [\underline{x}, \overline{x}] \subset \mathbb{R} & \text{if } \underline{x} \le \overline{x} \text{ (“proper” interval)} \\ \text{something weird} & \text{if } \underline{x} > \overline{x} \text{ (“improper” interval).} \end{cases}$$

The Levels structure

Distinguish 4 specification levels (as in floating point standard IEEE754):

Level 1. Mathematical theory of intervals & their operations.

Level 2. Finite precision intervals—datums—& operations, independently of their representation.

Level 3. Representation of datums by objects, e.g. in terms of floating point numbers.

Level 4. Encoding of Level 3 objects as bit-strings.

Inter-level maps: a key decision

Maps between levels are crucial—especially L1 $\longleftrightarrow$L2. We decided:

- Each datum *is* a mathematical interval, i.e.

$$\text{L2 datums} \overset{\text{identity map}}{\longrightarrow} \text{L1 intervals} \qquad (*)$$

- Datums are organised into finite sets $\mathbb{T}$ called interval types.
- A L1 interval $\mathbf{x}$ maps to an interval of type $\mathbb{T}$ (a $\mathbb{T}$-interval) by the $\mathbb{T}$-hull operation = smallest (in $\supseteq$ sense) $\mathbb{T}$-interval that contains $\mathbf{x}$.

$$\text{L1 intervals} \overset{\mathbb{T}\text{-hull}}{\longrightarrow} \text{L2 datums of type } \mathbb{T} \qquad (**)$$

- To do an operation $\mathbf{x} \bullet \mathbf{y}$ at L2 on $\mathbb{T}$-intervals:
  map $\mathbf{x}, \mathbf{y}$ to L1 by (*); do operation at L1; map back to L2 by (**).

Looks trivial but isn't! Not all IA theories are clear on this.
IMO, this choice defines the whole character of the standard.

Inter-level maps, contd

Then two obvious rules

- L2 $\longleftrightarrow$L3: Each L2 datum is represented by at least one L3 object; each L3 object represents at most one L2 datum.
- L3 $\longleftrightarrow$L4: Each L3 object is encoded by at least one L4 bitstring; each L4 bitstring encodes at most one L3 object.

Outline

Exception handling—a hypothetical scenario

Less than 10 years hence in the Old Bailey . . .

- Crown vs Google concerns Google's driverless car GDC. One of them badly injured a pedestrian who stepped into the road in front of it.
- GDC's emergency stop system is *designed* to act faster than a good human driver (undisputed) but is it badly *implemented* (disputed)?
- The software uses an interval algorithm, built on a 1788-conforming library, which applies Brouwer's fixed point theorem.
- Depending on what software bugs are found (if any), liability might lie with the pedestrian's negligence? GDC's software implementers? the 1788 library implementers? 1788's mathematicians? etc.
- A lot of ££ rides on whether 1788-based code might be wrong, when deciding that a function is defined & continuous on a box.

Exception handling—context

The basic problem is how (at Level 1) to treat operations that aren't everywhere defined [and/or continuous] on the input box, e.g.

(real) square root $\sqrt{[-2,2]}$; $\dfrac{[2,3]}{[-1,1]}$; $\texttt{floor}([2.5,4.5])$?

- We decided the default is "evaluate where defined, ignore where undefined", called non-stop or loose evaluation, e.g.
$$\sqrt{[-2,2]} = \{\, \sqrt{x} \mid x \in [-2,2] \text{ and } x \geq 0 \,\} = [0, \sqrt{2}]$$
with no error reported. (Like IEEE754 floating point.)
OK for, e.g., many global optimisation methods.
- Not OK for applying Brouwer's theorem, which needs to know a function is everywhere defined & continuous on a box.
- Also not OK for some graphics rendering algorithms, which need to know definedness, not bothered about continuity.

Exception handling—decorations

- So one needs a mechanism to track whether a library operation has these desirable properties of definedness and/or continuity.
- This leads to a powerful extension of the Fundamental Theorem of IA based on theorems of set theory & analysis:
  - If for function f given by an expression, each individual library operation is everywhere defined on its inputs, then the same goes for f.
  - Same with defined replaced by defined & continuous.
- We rejected the IEEE754 FP standard's method of *global flags*—obsolete for today's massively parallel platforms.
- Instead provide facility of decorated interval $(\mathbf{y}, dy) =$ interval $\mathbf{y}$ plus tag dy (a decoration)[1] giving information about definedness, continuity, etc.

[1] dy just means "decoration for $\mathbf{y}$", nothing to do with differentials!

Exception handling—decorations contd

- Formally, a decoration d is a label for an assertion (predicate) $p_d(f, \mathbf{x})$ about a function $f : \mathbb{R}^n \to \mathbb{R}$ and a box $\mathbf{x} \subseteq \mathbb{R}^n$, for arbitrary n.
- 5 decorations are defined in increasing order of "goodness" `ill` $<$ `trv` $<$ `def` $<$ `dac` $<$ `com`:
  - `ill` Label for ill-formed intervals, formally "f is nowhere defined".
  - `trv` (trivial) Always true $=$ "no information".
  - `def` f is everywhere defined on $\mathbf{x}$.
  - `dac` As `def`, plus everywhere continuous on $\mathbf{x}$.
  - `com` As `dac`, plus bounded at Level 2 (no overflow while computing it).
- Let $(\mathbf{y}, dy)$ result from evaluating arithmetic expression $f(x_1, \ldots, x_n)$ on *correctly initialised* decorated intervals $(\mathbf{x}_1, dx_1), \ldots, (\mathbf{x}_n, dx_n)$.
- Then, in addition to $\mathbf{y} \supseteq$ range of f over $\mathbf{x} = \mathbf{x}_1 \times \ldots \times \mathbf{x}_n$, the decoration dy makes a true assertion about f over $\mathbf{x}$.
  E.g. if $dy =$ `def` then f was proved to be everywhere defined on $\mathbf{x}$.

Exception handling—decorations contd

- This exception handling method is the feature that most distinguishes 1788 from earlier IA systems.
- There's no magic: it relies on systematically exploiting facts such as "composition of everywhere defined functions is everywhere defined". An Annex in the Standard contains a rigorous proof of correctness of the decoration system: a Fundamental Theorem of Decorated Interval Arithmetic.
- Like range enclosures, it's often *not sharp*, e.g. may return `trv` (no info) or `def` (defined) when actually `dac` (defined & continuous) is true.
- Much of the craft of IA is knowing how to "sharpen" such info, e.g. by cutting an input box into smaller boxes handled separately.

Outline

Difficulties the group encountered

Certain topics caused heated debate. Examples:

- Choice of foundational math model of intervals & operations. We split into "set-based" (mostly academic) and "Kaucher" (earn $$ from intervals) factions.
- Flavors: the way of accommodating different foundations.
- The decoration scheme—result of over a year's discussion.
- Correctness proof—to use in hypothetical litigation above?
- Kinds of exception to which decorations are unsuited, e.g. bad interval constructor calls.
- What to say about accuracy? Just leave it as a QoI issue?
- Exact dot-product—should it be part of the 1788 standard?

Outline

1788 project: current state of play

- The current document has
  $\sim$ 60 pp of main text (requirements) of which roughly 50% Level 1, 45% Level 2, 5% Level 3, a tiny bit of Level 4.
  $\sim$ 15 pp of operation tables and other help for implementers
  $\sim$ 18 pp of the *Basic Standard*, a cut down, simpler to implement, version.
- Vote of the group approved it in May 2014.
- We are preparing *Sponsor Ballot* stage of IEEE process, where it is examined by a selected group intended to be
  – representative of academia, software developers, industry, etc.;
  – geographically balanced.
- This should result in changes, hopefully minor ...
- and we hope it will be accepted as an IEEE document in early 2015.

$[\ \rule{3cm}{0.4pt}\ ]$ good

$[\,-\,]$ better

$[\,-\,]_{\mathtt{dac}}$ even better

Decorations example

- Consider fix point problem $g(x) = x$ where
$$g(x) = 2\sqrt{x} - \tfrac{1}{2}.$$
Roots are $x = \frac{3}{2} \pm \sqrt{2} = 0.0858\ldots$ or $2.9142\ldots$
- Use fixed point iteration $\mathbf{x}_{n+1} = g(\mathbf{x}_n)$
- Initial $\mathbf{x}_0 = [2, 3]$ gives
$\mathbf{x}_1 = [2\sqrt{2} - \frac{1}{2}, 2\sqrt{3} - \frac{1}{2}] = [2.3\ldots, 2.9\ldots] \subset \mathbf{x}_0$.
This is genuine and (Brouwer) shows a fixpoint exists in $\mathbf{x}_1$.
- Initial $\mathbf{x}_0 = [-1, \frac{1}{16}]$ gives

$$\mathbf{x}_1 = 2\sqrt{[-1, \tfrac{1}{16}]} - \tfrac{1}{2} = 2{\color{red}[0, \tfrac{1}{4}]} - \tfrac{1}{2} = [0, \tfrac{1}{2}] - \tfrac{1}{2} = [-\tfrac{1}{2}, 0], \text{ again } \subset \mathbf{x}_0 \text{ !}$$

This is spurious, due to 1788 (undecorated) arithmetic discarding the negative part of $\mathbf{x}_0$ without comment.

- Using decorated interval arithmetic—using the rules for propagating decorations through operations, which I skate over—the 2nd example gives

$$\begin{aligned}\mathbf{x}_1 &= [2]_{\mathtt{dac}} \times \sqrt{[-1, \tfrac{1}{16}]_{\mathtt{dac}}} - [\tfrac{1}{2}]_{\mathtt{dac}} \\ &= [2]_{\mathtt{dac}} \times [0, \tfrac{1}{4}]_{\mathtt{trv}} - [\tfrac{1}{2}]_{\mathtt{dac}}, \qquad \mathtt{trv} = \text{“no information”} \\ &= [0, \tfrac{1}{2}]_{\mathtt{trv}} - \tfrac{1}{2}_{\mathtt{dac}} \\ &= [-\tfrac{1}{2}, 0]_{\mathtt{trv}},\end{aligned}$$

while the 1st example produces

$$\mathbf{x}_1 = [2.3\ldots, 2.9\ldots]_{\mathtt{dac}}.$$

- I.e. in 1st case we conclude conditions of Brouwer's Theorem are satisfied, but in 2nd case are unable to do so.