Prof. W. Kahan's Comments on SORN Arithmetic

What follows pertains to John L. Gustafson's proposal

"A Radical Approach to Computation with Real Numbers"

presented in two documents:

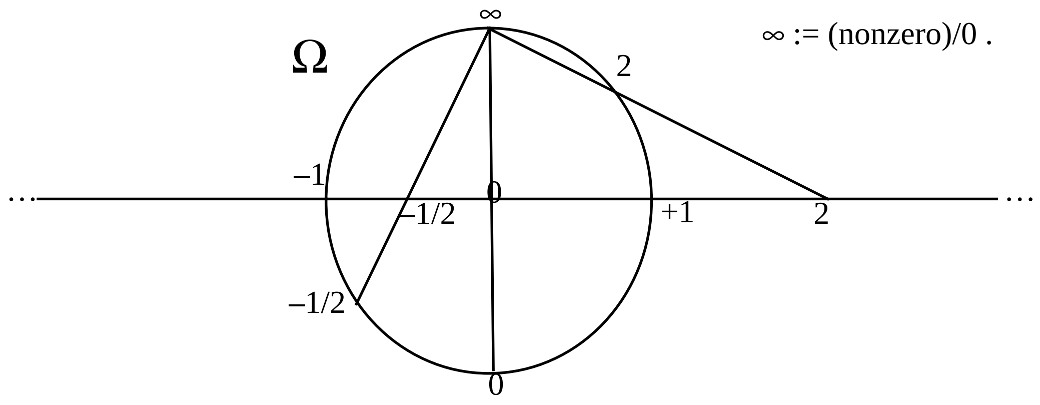
Unums2.0slides.pptx	48 numbered pages	"Updated April 23 2016"
RadicalApproach.pdf	16 unnumbered pages	undated

The .pdf file is "published with open access at SuperFri.org" but I could not find it there. A version later than the one found on his web page on 18 May 2016 may exist; I know not where.

What are SORNs ?

These are **S**ets **O**f **R**eal **N**umbers upon which Gustafson's proposal would perform arithmetic indirectly by table-lookups to achieve higher speed than if performed by floating-point arithmetic. Before describing how that would work, let's describe his SORNs as briefly as possible.

They are subsets drawn from all Extended Real Numbers, which are the real numbers augmented by one point at ∞ whose sign is ignored. SORNs are best visualized as point-sets on the circle Ω obtained by *Stereographic Projection* from the real axis:



$\infty := (\text{nonzero})/0$.

A motivation for doing this is that, regarded as a map from the circle Ω to itself, each rational function of just one real variable becomes infinitely differentiable despite any poles it may have. Plotted on a torus, the function has an infinitely smooth graph. However, although those rational functions constitute a *Field*, the Extended Reals do not. These cannot satisfy all of a Field's cancellation laws because of *Exceptions* at $\infty \pm \infty$, ∞/∞ and $\infty \cdot 0$, besides the usual $0/0$, that would generate NaNs in IEEE 754 floating-point. ("NaN" means "**N**ot **a** **N**umber".)

Gustafson disparages NaNs, so he assigns the set of all Extended Reals — call it Ω — as the arithmetic result of those failures (.pptx p. 9); then he can boast that his SORN arithmetic has …

"No exceptions (subnormals, NaNs, 'negative zero', …)" [p. 3 of .pptx, p. 2 of .pdf]

But his assignment of a value Ω to those Field exceptions deprives SORN arithmetic of what I have called *Algebraic Integrity*, to be explained in a moment. That lack deprives programmers of confidence that the numerical evaluation of formulas derived via ordinary rational algebra will produce predictable results or else signal malfunctions detectable and correctable by the program.

What is *Algebraic Integrity* ?

Absent roundoff, if several different expressions for the same rational function produce different values when evaluated numerically in IEEE floating-point, at most two different values can be produced, and either the two values are $\pm\infty$ or else at least one is a NaN, which is easy to detect.

SORN arithmetic has no such integrity. SORN arithmetic's evaluations of different expressions for the same function, though they may have different removable singularities, can produce any number of different values, none of them recognizably exceptional. Here is an example:

$$u(t) := 2t/(1+t)\ ; \quad v(t) := 2 - (2/t)/(1+1/t)\ ;$$

$$x(t) := (1+u(t)^2)/(2+u(t)^2)\ ; \quad y(t) := (1+v(t)^2)/(2+v(t)^2)\ ; \quad z(t) := 1 - 1/(2+v(t)^2)\ ;$$

An automated algebra system would confirm that $u = v$ and $x = y = z$ as rational functions of t disregarding removable singularities at $t = 0$ and $t = -1$. However, SORN arithmetic yields

$$x(0) = 1/2\ , \quad y(0) = \{0 < y \le \infty\}\ , \quad z(0) = \{1/2 \le z \le 1\}\ .$$

A program cannot sense something amiss with $y(0)$ or $z(0)$ since, ideally, SORN arithmetic has no rounding errors according to p. 3 of .pptx, p. 2 of .pdf. Instead it may "lose information" by producing undeserved intervals instead of single points. This has happened to $y(0)$ and $z(0)$.

Like interval arithmetic, SORN arithmetic is *Inclusion-Monotonic*; this means that if SORNs X and Y overlap, then SORN evaluations of the same expression at X and at Y must produce SORNs that overlap. Consequently, SORN evaluations of different expressions for the same *rational* function cannot be arbitrarily different; their intersection must be non-empty.

Why would a programmer put different expressions for the same function into a program? It is a common practice to enhance the reliability and/or accuracy of numerical software relying upon that function. Over different parts of the function's domain, different expressions may be less sensitive to roundoff or "loss of information"; or different expressions may have different costs of evaluation. On the boundaries of subdomains different expressions would be expected to agree within roundoff. If one expression malfunctions, the program can try another,

but only if it detects the malfunction.

An arithmetic system that hides malfunctions must produce quite misleading results occasionally. We shall see that happen to SORN arithmetic later below.

• • • • • • • • • •

In several respects SORN intervals containing ∞ resemble the *Exterior Intervals* I advocated in 1968 during a Summer Course on Numerical Analysis (#6818) at the Univ. of Michigan at Ann Arbor. These were intervals like $X = [6,\ 8]/[-1,\ 2] = [3,\ -6] = \{x = \infty \text{ or } x \ge 3 \text{ or } x \le -6\}$. They were introduced to help evaluate continued fractions, and were expected to exist rarely and briefly. However, they complicated the implementation of interval arithmetic, perhaps more than they were worth at the time. We shall see them complicate SORN arithmetic too, worsened by its lack of Algebraic Integrity. Gustafson could have avoided that lack by a small change in his definitions, but then he could not have boasted that SORN arithmetic had "No exceptions".

How does SORN arithmetic work?

It uses table-look-ups to act upon *pointers* to SORN values, not upon the values themselves, in the hope of performing every arithmetic operation in a cycle or two on the CPU chip.

Let us consider pointers that are binary integers N bits wide. N is expected to be small, likely 16 or less. Each pointer has two integer values, a 2's complement value j and an unsigned value J. Context will determine which interpretation is used. It won't be confusing:

$$j = \text{if } J < 2^{N-1} \text{ then } J \text{ else } J - 2^N \,; \qquad J = \text{if } j \ge 0 \text{ then } j \text{ else } 2^N + j \,;$$
$$J = +2^{N-1} \text{ just when } j = -2^{N-1} \,.$$

Each even index j points to a single Extended Real value $x(j)$ ordered monotonically so that $x(j+2) > x(j)$ except that $x(\pm 2^{N-1}) = \infty$. SORN's values $x(j)$ inherit the sign-symmetry of their indices: $x(-j) = -x(j)$ except that $x(\pm 2^{N-1}) = \infty$; consequently $x(0) = 0.0$. SORN's values $x(j)$ occur in reciprocal pairs: $x(j) = 1/x(\text{sign}(j)\cdot 2^{N-1} - j)$ except $0.0 = 1/x(\pm 2^{N-1}) = 1/\infty$. Therefore $x(\pm 2^{N-2}) = \pm 1.0$ respectively. Otherwise you may distribute SORN values $x(j)$ as you please.

Each odd index j points to the open interval $X(j)$ between adjacent single values $x(j \pm 1)$; $X(j) = \{x(j-1) < X < x(j+1)\}$ except $X(2^{N-1} - 1) = \{x(2^{N-1} - 2) < X < +\infty\}$. Equivalently $X(J)$ is the open interval between $x(J-1)$ and $x(J+1)$ except $X(2^N-1) = \{x(2^N-2) < X < 0\}$. The symmetries possessed by single values $x(j)$ are inherited by the intervals $X(j)$. The exceptional cases occur in different places for unsigned than for signed indices; this will matter later.

Thus the entire circle Ω of Extended Reals is covered once by a system that combines single values $x(\text{even } j)$ and intervals $X(\text{odd } j)$. Rational arithmetic operations on the system need two tables, one for add/subtract, one for multiply/divide, with 2^N rows and 2^N columns and 2^{2N} entries, each an arithmetical result. Unfortunately, except perhaps for some very small N, …

the system is not yet closed under rational arithmetic operations.

The arithmetical result of combining an $x(j)$ or $X(j)$ with an $x(k)$ or $X(k)$ need not be any of these, but can overlap more than one of them. Gustafson describes such an overlap as "a loss of information" on .pptx p. 10, .pdf p. 5. To close the system each arithmetical result in each arithmetic operation's table must be an interval on Ω. This interval can be represented by a pair «I, ΔI» of unsigned integers pointing to an interval that runs counter-clockwise on Ω beginning at $x(I)$ or $X(I)$ and ending at $x(I+\Delta I)$ or $X(I+\Delta I)$. I think this is what Gustafson intended by "run-length encoding" mentioned on .pptx pp. 36 and 47, described at the top of .pdf p. 15 but a little incorrectly. The 2^N pairs «I, 2^N–1» are redundant; only one, say «2^N–1, 2^N–1», is needed for the whole circle Ω. Assign another arbitrarily, say «0, 2^N–1», to the empty set Ø. Ignore unsigned integer overflow if it happens to I+ΔI. (This is what unsigned indices are for.)

As a SORN computation proceeds, intervals will combine with intervals to yield usually wider intervals each represented by a pair «I, ΔI». Addition/subtraction requires two fetches from its table, multiplication/division as many as four, followed by some further logic. SORN arithmetic acting upon pairs «I, ΔI» is no less complicated than interval arithmetic with exterior intervals.

Originally Gustafson came up with a simpler kind of SORN arithmetic suitable only for smaller N, say $N < 9$, which implements very low precision exterior interval arithmetic and more. Let W be a word 2^N bits wide representing an arbitrary subset of elements $x(J)$ and $X(J)$ on the circle Ω: Bit #J of W is 1 if and only if $x(J)$ (for even J) or $X(J)$ (for odd J) belongs to the SORN represented by W. There are 2^{2^N} such SORNs. A rational operation upon two SORNs, represented by two such words, is selected from a table with 2^N rows, 2^N columns, and 2^{2N} entries, each a word 2^N bits wide, but wired together to put out their logical OR as one word.

Thus can each rational operation upon two arbitrary subsets of the points and intervals on Ω be performed by a CPU's chip in one cycle if N is small enough for the tables to fit on the chip.

SORN Arithmetic's Cost vs. Benefit

Two ways to perform SORN arithmetic have been described. The faster way, acting upon words W each 2^N bits wide, requires on-chip tables occupying areas proportional to 2^{3N}. This has to compete with other demands, like the cache, for that area. Besides, computations nowadays tend to wait longer for memory management than for arithmetic. Why build an arithmetic unit so fast that it spends most of its time waiting to be fed? I doubt that this way will ever be built.

The slower way to perform SORN arithmetic, acting upon pairs «I, ΔI» each $2N$ bits wide, requires on-chip tables occupying areas proportional to $N \cdot 2^{2N}$ plus some complicated logic that will take at least two or three cycles. This runs about as fast as ordinary floating-point. SORNs beat interval arithmetic with exterior intervals programmed in software on hardware of roughly the same precision, somewhat less than N sig.bits, occupying an area on chip proportional to N^2. Consequently SORN computation can compete successfully against floating-point only if its precision N is not too big, and then only if the value added to computation by interval arithmetic offsets the higher cost of the greater area SORN hardware occupies on the CPU chip.

Low precisions N have sufficed for much of the world's data. Engineers used slide rules (< 10 sig.bits) for most calculations for many decades before electronic computers and aircraft existed. How much value does interval arithmetic, with or without SORNs, add to computation at low precision without it? Gustafson advocates SORN/interval arithmetic as insurance against being misled when computed results are corrupted by roundoff. History is not entirely on his side.

Long experience and some error-analyses support a rough rule-of-thumb that renders roundoff extremely unlikely to causes embarrassment if *all* intermediate floating-point computations are performed carrying a little more than twice the precision trusted in data and desired in results. This advice has survived the test of time in statistics, optimization, root-finding, geometry, structural analysis and differential equations, among other things. Of course exceptions exist; their rarity has given rise to a wry joke among numerical analysts:

> Nobody unlucky enough to have been betrayed by that rule-of-
> thumb need concern us; he has already been run over by a truck.

SORN/interval arithmetic insures against that bad luck but invites a different kind of betrayal.

Excessively Wide Intervals

Gustafson knows that interval arithmetic can produce intervals that are too wide, sometimes by orders of magnitude. He knows about two of the many ways that can happen: One is called "The Wrapping Effect"; another is called "The Dependency Problem". He asserts that his SORN arithmetic is free from such shortcomings. He is mistaken.

The wrapping effect arises when a repetitive computation like a vector iteration $\mathbf{x}_{n+1} := \boldsymbol{f}(\mathbf{x}_n)$ is programmed in interval arithmetic as $\mathbf{X}_{n+1} := \mathbf{F}(\mathbf{X}_n)$ by transliterating a formula for $\boldsymbol{f}$ directly into an interval arithmetic program $\mathbf{F}$. The effect occurs because coffin $\mathbf{F}(\mathbf{X}) \supseteq \boldsymbol{f}(\mathbf{X})$, which is not often a coffin. Here a *coffin*, represented by a vector of d intervals in a d-dimensional space, is a rectangular parallelepiped with $d \cdot 2^{d-1}$ edges each parallel to a coordinate axis. Even if the image $\boldsymbol{f}(\mathbf{X})$ of a coffin $\mathbf{X}$ is merely a slightly tilted box, $\mathbf{F}(\mathbf{X})$ must be bigger than $\boldsymbol{f}(\mathbf{X})$ by some factor $\Lambda > 1$, and then $\mathbf{X}_n$ must be too big by a factor Λ^n. Exponential growth.

Gustafson asserts on .pptx pp. 42-3 that SORN computation's "Uncertainty grows *linearly* in general" and displays about 30 steps of an orbit calculation drawn from p. 306 of his book *THE END OF ERROR — Unum Computing* (2015, CRC press). There his word for a coffin is "ubox". The book's elaborate computation resembles the numerical solution of the differential equation for a two-body orbit but actually, by taking account of its conservation of energy and angular momentum, the computation amounts to an iterative solution of a trigonometric equation. See www.eecs.berkeley.edu/~wkahan/Math128/KeplerOrbits.pdf for details. Uncertainty for that orbit should grow linearly. But Λ exceeds 1 by a little for each short step $\mathbf{F}$ of his method to calculate orbits for three or more bodies. I don't think he has done that for more than a few dozen steps amounting to a tiny fraction of an orbit. Had he carried his calculation out for one or two complete and stable orbits he would have seen linear turn into excessive exponential growth.

Since SORN arithmetic produces only coffins, it cannot avoid the wrapping effect. To attenuate this effect the coffin $\mathbf{X}_0$ must be subdivided into smaller coffins at a cost in parallel computations that grows exponentially with the dimension d. Attenuation is affordable only if d is small.

The dependency problem arises when an expression's subexpressions are correlated but interval arithmetic disregards their correlations while evaluating the expression numerically. Two simple examples are Z := X – Y and Q := X/Y when an earlier assignment Y := X is disregarded. Here are three program extracts that expose the mistake in Gustafson's assertions, on .pptx pp. 44-6 and .pdf p. 12, that SORN arithmetic has no dependency problem:

Y := ... independent of X ;	Y := … accidentally matches X ;	X := … ;
… no change to X nor Y ;	… no change to X nor Y ;	… no X nor Y ;
Z := X – Y ;	Z := X – Y ;	Z := X – X ;

What interval operations will the computer execute for the programs' last subtractions? Normally the width of an interval difference is the sum of the operands' widths. Gustafson expects the computer to treat the last program's subtraction differently from the others, obtaining for Z an interval narrower than X. Gustafson repeats the operation by iterating X := X – X to generate a dwindling sequence that "Converges to the smallest open interval containing zero." If a computer is smart enough to perform a different operation for "X – X" than for other subtractions, why not save time by simply "optimizing" X – X to zero as many an optimizing compiler would?

Gustafson makes the same mistake when he starts $X := [1/2, 2]$ and iterates $X := X/X$ on .pptx p. 46 to obtain convergence to $(5/8, 8/5)$, not just 1.0. Why does he make this mistake again?

Perhaps the algorithms he has in mind for SORN arithmetic are incorrect for SORNs that are independent intervals. Has he programmed and compared them yet with interval arithmetic?

Closely related to the dependency problem is a third way for SORN/interval arithmetic to yield vastly pessimistic over-estimates of a computed result's uncertainty. It occurs when a numerically precarious algorithm has been chosen to compute the desired result. For example, the result may satisfy a system of equations whose coefficients depend upon a few parameters each uncertain independently within small tolerances like, say, 0.1%. Consequently the coefficients will inherit uncertainties, perhaps as little as 0.1%, but not independently; the coefficients' uncertainties will be correlated. Resist the temptation to carry only three or four sig.dec. while computing the coefficients and solving the equations. The solution may be far more sensitive to uncorrelated 0.1% perturbations in the coefficients than to 0.1% perturbations in the parameters. It happens to calculations of deflections under load of cantilevered elastic structures, especially wings and shells supported only at their periphery, like the roof of a stadium unsupported by pillars that would block the views of some spectators. It happens to computer-simulated crash tests of cars and aircraft that derive much of their strength from their outer shells. The rule-of-thumb cited above offers some protection against roundoff. SORN/interval arithmetic carrying only three or four sig.dec. would produce intervals far wider than the uncertainties inherited from parameters by the correct results, but a naive user could misinterpret those wide intervals as if they were uncertainties inherent in the results, and then react inappropriately.

A simple didactic example of a geometrical problem whose solution is far less sensitive to small perturbations of the data than to small perturbations in the linear equations the solution satisfies is a tetrahedron's incenter posted on p. 26 of www.cs.berkeley.edu/~wkahan/MathH110/Cross.pdf .

SORN Equation-Solving Without Algebraic Integrity

An important application of interval arithmetic locates *all* solutions $\mathbf{z}$ of an equation $\text{æ}(\mathbf{z}) = \mathbf{o}$ given a program Æ to compute a vector-valued function $\text{æ}(\mathbf{x})$ in interval arithmetic at interval vectors $\mathbf{X}$ (coffins) obtaining coffins $\text{Æ}(\mathbf{X}) \supseteq \text{æ}(X)$. The program Æ has to be good enough that $\text{width}(\text{Æ}(\mathbf{X})) \to 0$ as $\text{width}(\mathbf{X}) \to 0$ except for rounding errors, so $\text{width}(\text{Æ}(\mathbf{z}))$ must be relatively tiny wherever $\text{æ}(\mathbf{z}) = \mathbf{o}$. Coffin $\mathbf{X}$ cannot contain a solution $\mathbf{z}$ if $\text{Æ}(\mathbf{X})$ is an interval that excludes $\mathbf{o}$, in which case $\mathbf{X}$ is discarded. Otherwise $\mathbf{X}$ is partitioned into smaller coffins $\underline{\overline{\mathbf{X}}}$ each of which is tested and either discarded or kept for further subdivision.

We hope that this process will ultimately converge to a collection of relatively tiny coffins $\mathbf{Z}$ at none of which does interval $\text{Æ}(\mathbf{Z})$ exclude $\mathbf{o}$, and at some of which $\text{Æ}(\mathbf{Z})$ includes $\mathbf{o}$. Each $\mathbf{Z}$ that includes $\mathbf{o}$, if any, has located a solution $\mathbf{z} \in \mathbf{Z}$. Any other $\mathbf{Z}$ encloses a singularity around which $\text{Æ}(\mathbf{Z})$ is NaN. Gustafson calls this process "C-Solution" in his book.

For example take $\text{æ}(\xi) := 3/(\xi+1) - 2/(\xi-1) + 1/(\xi-1)^2$. Interval arithmetic, with or without exterior intervals, produces NaN for $\text{Æ}(X)$ when $X = [0, 4]$ because of $\infty - \infty$; but this X must not be discarded lest it contain solutions z as well as singularities. Subsequent subdivisions converge to tiny intervals around 1, 2 and 3. Evidently $\text{æ}(2) = \text{æ}(3) = 0$ but $\text{æ}(1)$ is NaN.

At every X around 1, SORN arithmetic gets $Æ(X) = æ(1) = \Omega$, which includes 0. SORN arithmetic alone does not reveal that $æ(\xi)$ has a *pole* (goes to ∞), not a *zero*, at $\xi = 1$.

Another more perplexing example is constructed from $R(\xi, \eta) := (\xi-\eta)\cdot(\xi+\eta)/(\xi^2 + \eta^2)$. At $\mathbf{x} = \begin{bmatrix} \xi \\ \eta \end{bmatrix}$ let $\mathbf{æ}(\mathbf{x}) := \begin{bmatrix} R(\xi, \eta) - 9/8 \\ R(\eta, \xi) + 9/8 \end{bmatrix}$, and try to find a C-Solution $\mathbf{z}$ of $\mathbf{æ}(\mathbf{z}) = \mathbf{o}$ using SORN arithmetic to compute $\mathbf{Æ}(\mathbf{X})$ at coffins $\mathbf{X}$ each a rectangle with edges parallel to the coordinate axes. At every $\mathbf{X}$ that encloses $\mathbf{o}$ we find that $\mathbf{Æ}(\mathbf{X}) = \begin{bmatrix} \Omega \\ \Omega \end{bmatrix}$, which encloses $\mathbf{o}$. At every $\mathbf{X}$ that does not enclose $\mathbf{o}$ but has one corner much closer to $\mathbf{o}$ than the others, $\mathbf{Æ}(\mathbf{X})$ is a finite interval that encloses $\mathbf{o}$. C-Solution converges convincingly onto an alleged solution $\mathbf{z} = \mathbf{o}$.

But " $\mathbf{æ}(\mathbf{z}) = \mathbf{o}$ " has no solution $\mathbf{z}$.

It has no solution because $-1 \le R(\xi, \eta) = -R(\eta, \xi) \le +1$ except that $R(0, 0)$ would be NaN in interval arithmetic but not SORN arithmetic. Its lack of Algebraic Integrity has betrayed the C-Solution process. That could be fixed by some small changes to SORN arithmetic.

Another contributor to betrayal by SORNs is not so easy to fix: …

Mathematically Sound?

Gustafson asserts that SORN arithmetic is "mathematically sound, with no rounding errors" on p.3 of .pptx, p. 2 of .pdf. Instead of rounding errors, SORN/interval arithmetic produces overly wide intervals, sometimes vastly too wide. A little too wide suffices to induce mathematically unsound inferences. Let us re-examine the C-Solution process that was betrayed just above. It involved expression $R(\xi, \eta) := (\xi-\eta)\cdot(\xi+\eta)/(\xi^2 + \eta^2) = -R(\eta, \xi) = R(\mu\cdot\xi, \mu\cdot\eta)$ for any $\mu > 0$. Another expression for the same function is $S(\xi, \eta) := 1 - 2/(1 + (\xi/\eta)^2)$. SORN arithmetic, with or without Algebraic Integrity, produces excessively wide intervals for R but not for S :

Ξ	Y	$R(\Xi, Y)$	$S(\Xi, Y)$
$[\mu,\ 2\mu]$	$[\mu,\ 2\mu]$	$[-2,\ 2]$	$[-3/5,\ 3/5]$
$[\mu,\ 10\mu]$	$[\mu,\ 10\mu]$	$[-90,\ 90]$	$[-0.98,\ 0.98]$
$[\mu,\ 10\mu]$	$[-\mu,\ \mu]$	$[0,\ 121]$	$[0,\ 1]$
$[\mu,\ 2\mu]$	$[-\mu,\ \mu]$	$[0,\ 9]$	$[0,\ 1]$
$[\mu,\ 2\mu]$	$[0,\ \mu]$	$[0,\ 6]$	$[0,\ 1]$
$[10\mu,\ 11\mu]$	$[0,\ 0]$	$[0.826,\ 1.21]$	$[1,\ 1]$
$[181\mu,\ 192\mu]$	$[0,\ 0]$	$[0.8887,\ 1.12524]$	$[1,\ 1]$
$(0,\ \mu]$	$[0,\ 0]$	$(0,\ \infty]$	$[1,\ 1]$

As $\mu \to 0+$ the rectangles $\mathbf{X} := (\Xi, Y)$ shrink to $\mathbf{o}$ without enclosing it, and every $\mathbf{Æ}(\mathbf{X})$ above includes $\mathbf{o}$. The C-Solution process converges towards and is stopped at tiny rectangles very near $\mathbf{o}$ though it is a singularity, not a solution $\mathbf{z}$ of " $\mathbf{æ}(\mathbf{z}) = \mathbf{o}$ ". SORN arithmetic is not prevented by "no rounding errors" from producing mathematically misleading numerical results.

Example: an Elbow Manipulator

A set of 12 equations has been chosen to illustrate the utility of SORN/interval arithmetic. Six angles $\theta_1, \theta_2, \dots, \theta_5, \theta_6$ determine the posture of a robot arm with three elbow joints and three wrist joints. A posture has been specified by six numerical values given for six polynomials in the 12 variables $c_j := \cos(\theta_j)$ and $s_j := \sin(\theta_j)$, plus six more equations $s_j^2 + c_j^2 = 1$ to constitute 12 equations in the 12 variables. We seek the set $\{\theta_j\}_1^6$ of solutions of these equations. These are exhibited on .pptx p. 12 and .pdf p.7, and solutions of low but adequate accuracy are offered on .pptx p. 24 and .pdf p. 8. Unfortunately the second equations exhibited in .pptx and .pdf differ in a term's sign, so I have not tried to confirm the solutions offered. Still, the equations illustrate well the utility of the C-Solution process using SORN/interval arithmetic because the solution-set occupies a continuum in a 6-dimensional space. *But the equations are not realistic.*

They take no account of constraints that limit the angles to prevent wires from being torn and to protect the robot arm from self-collisions. At the very least, constraints like $|\theta_j| \le 178°$ must be imposed. Such constraints are awkward to impose upon the pairs $\{c_j, s_j\}$ of variables appearing in the 12 equations. Moreover, there are too many equations and variables; the problem really requires only five equations in five unknowns. The equations determine θ_1 immediately; it is either $22.9181°$ or $22.9181° - 180°$. Replacing one by the other replaces θ_j by $\pm180° - \theta_j$ for $j = 2, 3, 4$ and 5, and θ_6 by $-\theta_6$, so only one value of θ_1 need be considered. Replacing $\cos(\theta_j)$ and $\sin(\theta_j)$ by $t_j := \tan(\theta_j/2)$ complicates the five remaining equations, replacing s_j by $2\cdot t_j/(1 + t_j^2)$ and c_j by $2/(1 + t_j^2) - 1$ to turn polynomials into rational expressions. Their extra cost is more than recompensed during the C-solution's partition of coffins into smaller coffins, of which there are far fewer in 5 dimensions than in 12. And constraints like $|\theta_j| \le 178°$ turn simply into $|t_j| \le \tan(89°)$. Solutions fill out a continuum whose dimension matters. What is it?

Loose Ends

SORNs, like Exterior Intervals, complicate the Order Relations $\{<, \le, =, \ne, \ge, >\}$ more than ordinary interval arithmetic does with exclusively finite intervals. Is $-2 < \infty$? Not if $+\infty = -\infty$. What is $\cos(\infty)$? How does the empty set $\emptyset$ compare with 3 ? Distinct intervals X and Y with non-empty overlap $X \cap Y \ne \emptyset$ must make "$X \ne Y$" true even though they differ in just one end-point. And $X \supseteq Y$ cannot imply $X \approx Y$ despite what Gustafson's book says. SORN arithmetic is incomplete without coherent specifications for order relations and the Math library.

Conclusion

SORN arithmetic is a plausible alternative to Exterior Interval arithmetic implemented in software on low-precision floating-point hardware. How much demand exists for that? Hard to say. The temptation to use SORN arithmetic for lengthy computations upon data barely less precise than the arithmetic should be resisted lest misleadingly over-sized uncertainties obscure the results.