Function Interval Arithmetic

Jan Duracz[1], Amin Farjudian[2], Michal Konečný[3], and Walid Taha[4]

[1] jan@duracz.net,
http://duracz.net/jan

[2] School of Computer Science, University of Nottingham Ningbo, China,
Amin.Farjudian@nottingham.edu.cn,
www.cs.nott.ac.uk/~avf

[3] School of Engineering and Applied Science, Aston University, Birmingham, UK,
M.Konecny@aston.ac.uk,
www-users.aston.ac.uk/~konecnym

[4] Halmstadt University, Sweden & Rice University, Houston, Texas, USA,
Walid.Taha@hh.se,
http://bit.ly/WT-EMG

Abstract. We propose an arithmetic of *function intervals* as a basis for convenient rigorous numerical computation. Function intervals can be used as mathematical objects in their own right or as *enclosures* of functions over the reals. We present two areas of application of function interval arithmetic and associated software that implements the arithmetic: (1) Validated ordinary differential equation solving using the AERN library and within the Acumen hybrid system modeling tool. (2) Numerical theorem proving using the PolyPaver prover.

Keywords: Validated Numeric Computation, ODEs, Theorem Proving

1 Background

In validated numerical computation, all values are computed together with rigorous upper bounds on their errors or uncertainty. Applications of this approach range from pure mathematics to the development of safety-critical systems.

Using more conventional methods, one can achieve such level of reliability by a combination of: (1) Approximate numerical computation based on floating-point numbers; (2) a specification of error bounds and a formal proof that the implementation of the algorithm stays within the bounds. While the conventional approach is appropriate in many applications, a formally proved numerical analysis can easily become too complex. Validated numerical computation often offers a viable alternative in such cases.

Keeping the derived error bounds relatively small is essential. We are interested in methods that support reducing the bounds arbitrarily close to their theoretical limits.

Interval computation. It is impossible to represent all real numbers or functions in finite space. Computation over these objects is realizable only through computation over their valid finite representations. For the purposes of validated computation, the most widely used approach is to represent a real number by an interval with floating-point endpoints that enclose the real number, bounding it from below and from above.

A driving force behind the development of interval computation has been a number of notable applications to rigorous differential equation solving, global optimization, and theorem proving. In the past couple of decades these application areas have also been combined in the construction of rigorous algorithms for reachability analysis of dynamical systems. A famous example of such advances is Tucker's proof that the Lorenz attractor is a strange attractor [Tuc02]. Interval computation has gained more recognition since the formation of the IEEE Interval Standard Working Group P1788 in 2008.

Interval computations use and produce enclosures of numerical quantities and functions. In their simplest form, such enclosures are just real intervals, most often used to bound single real values. Cartesian products of real intervals are interval vectors, or boxes. A box can be used to approximate a single real vector, a geometric entity, or a part of a function graph. Our focus is the approximation of functions, using enclosures that are more refined than boxes.

Function intervals are intervals whose endpoints are functions and can serve as enclosures of functions in the same way that real intervals can serve as enclosures of numbers. More formally, a function interval is a pair

$$[f(x_1, \ldots, x_n), g(x_1, \ldots, x_n)]$$

where f and g belong to the set $D \to \mathbb{R}$ of real-valued functions on a box domain $D \subset \mathbb{R}^n$. A function interval $[f, g]$ is typically used to enclose a single continuous function h in $D \to \mathbb{R}$—*i. e.*, $f \leq h \leq g$—and its *accuracy* increases as its width —*i. e.*, $||g - f||_\infty$—decreases.

Established special cases of function intervals include Berz and Makino's Taylor Models (TMs) [MB02] and affine forms [CS93]. Arithmetics of TMs and affine forms have been used with considerable success to bound rounding errors in floating-point computations [DK14a] and to approximate the solutions of differential equations in beam physics simulations [MB09]. Our implementation of function intervals is more general than TMs and affine forms.

2 Functionality

The AERN library[5] formally defines *functional interval arithmetic* as an *algebraic structure* over function intervals and provides an implementation based on polynomials. The algebraic structure includes the following operations:

– *constructors*: constant functions $0, \pi, [0, 1]$, projections, such as $\lambda(x, y).x$

[5] AERN is freely available from `https://github.com/michalkonecny/aern`

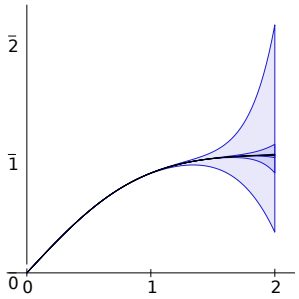

Fig. 1. Enclosures of $\mathrm{erf}(x)$ computed by AERN with various effort settings

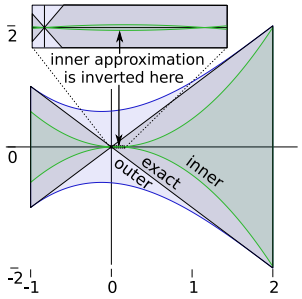

Fig. 2. The product $[-1,1]\cdot x$, its outer and inner approximations by AERN

- *pointwise operations*: the field operations $+, -, *, /$, common elementary functions such as e^x, $\sqrt{x}$, minimum, maximum and absolute value
- *analytic operations*: integration, such as $\int_0^x f(\xi, y)\, d\xi$
- *domain-changing operations*: evaluation (*e. g.*, $f(1)$, $f([0,1])$), composition (*e. g.*, $f(g(1,x),x)$), adding a variable (*e. g.*, $f(x) \mapsto f(x,y)$), restricting the domain of a variable (*e. g.*, $f(x,y)|_{x\in[0,1]}$)

Most of these operations can be computed only approximately, producing enclosures of the exact results. Each approximate operation has an optional parameter that gives the user control over the trade-off between computation effort and accuracy. For example, consider the task of approximating the error function

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

over the domain $x \in [0,2]$. In our implementation of this function using AERN[6], the following parameters are available to control the approximation effort:

- The precision of polynomial coefficients and constants such as π
- An upper bound on the polynomial degree
- The Taylor degree for approximating e^x

Fig. 1 shows the enclosures of the error function computed by AERN for several settings of the above effort parameters. The only parameter that changes is the upper bound on the polynomial degree and it ranges from 5 to 25.

Defaults are available for these effort parameters. There is also an iRRAM-style [Mül01] adaptive mode, in which the user specifies the desired accuracy of the result and the library adaptively increases effort parameters until the accuracy is reached.

AERN provides facilities to approximate not only real functions but also real interval functions, such as the product

$$[-1,1]\cdot x = [\min(-x,x), \max(-x,x)]\,. \tag{1}$$

[6] The code is available at `https://github.com/michalkonecny/aern/blob/master/aern-poly-plot-gtk/demos/erf.hs`

When approximating such intervals, it is often not sufficient to provide an enclosure. For example, to prove $0 \in [-\varepsilon, \varepsilon] + [-1, 1] \cdot x$, it is useful to compute an *inner* approximation of the right-hand-side interval function. If the inner approximation contains 0, so will the exact function. Fig. 2 shows an outer and an inner approximation of the function (1) computed within AERN[7]. Near $x = 0$ the plotted inner enclosure is slightly inverted. Adding $[-\varepsilon, \varepsilon]$ turns the enclosure into a consistent function interval containing 0.

3 Applications

We demonstrate the practical utility of function interval arithmetic via three concrete applications in computational mathematics.

Solving ODE IVPs. The arithmetic was used to compute enclosures for solutions of ordinary differential equation initial value problems (ODE IVPs) by means of a direct implementation of the interval Picard operator of Edalat and Pattinson [EP07]. Moreover, function interval arithmetic provides a conceptually simple way of extending Edalat and Pattinson's work to the case of uncertain initial conditions. Fig. 3 shows a parametric plot of an enclosure produced by AERN for the following Lorenz IVP with an uncertain initial value:

$$\begin{cases} y_1' = 10(y_2 - y_1) \qquad y_2' = y_1(28 - y_3) - y_2 \qquad y_3' = y_1 y_2 - 8y_3/3 \\ y(0) \in (15 \pm 0.01, 15 \pm 0.01, 36 \pm 0.01) \end{cases} \tag{2}$$

Note that the rectangular initial value uncertainty results in a non-rectangular intermediate value uncertainty as time progresses. The short elongated shapes visible in Fig. 3 are enclosures of the uncertainty sets for sample time points.

Enclosing Zeno behavior. A restricted version of function interval arithmetic and our ODE solving method are included in the Acumen[8] tool for modeling and rigorous simulation of hybrid dynamical systems. In combination with a novel method for event processing [KTD+13], the Acumen tool can compute a tight enclosure of a trajectory that contains infinitely many events due to so-called Zeno behavior. An example of such enclosure is shown in Fig. 4.

theorem proving. The arithmetic has been used to automatically prove theorems that take the form of inclusions of non-linear interval expressions, such as:

$$1 - e^{x^2}\left(\frac{0.3480242}{1 + 0.47047x} - \frac{0.0958798}{(1 + 0.47047x)^2} + \frac{0.7478556}{(1 + 0.47047x)^3}\right) \in \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \mathrm{d}t \pm 0.00005 \tag{3}$$

[7] The code is available at `https://github.com/michalkonecny/aern/blob/master/aern-poly-plot-gtk/demos/thickprod.hs`

[8] Freely available from `www.acumen-language.org`

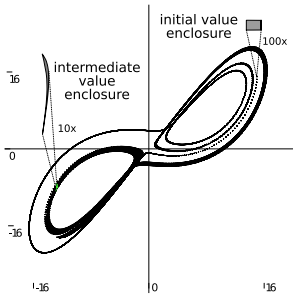

Fig. 3. An AERN enclosure of all solutions of the Lorenz IVP (2)

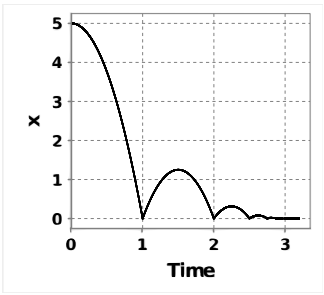

Fig. 4. An Acumen enclosure of a trajectory with Zeno behavior at time 3

A version of AERN has been embedded in our numerical theorem prover PolyPaver[9]. PolyPaver proves such inclusions by constructing an outer approximation of the contained function and an inner approximation of the containing interval function and showing that the inclusion holds for the approximations. (Fig. 2 gives an example of an inner approximation.) PolyPaver has been successfully applied to proving correctness theorems for tight accuracy properties of floating-point programs [DK14b]. For example, we proved that a Riemann integrator produces a value close to the exact integral.

4 Underlying theory

In this section we first give an overview of the main types available in the AERN library, and then explain why these types feature a generalized notion of interval. This is followed by a description of the role of abstract types in specifying and checking the reliability of our implementation. Finally, we highlight some aspects of Domain Theory, which inspires and models the essence of our approach to approximating real numbers, intervals and functions.

Types. The main types provided by the AERN library and their relationships within and outside AERN are outlined in Fig. 5. Specifically, there are two abstract types, one defining an algebraic structure of approximations to the real numbers and intervals and the other one defining a structure of approximations to continuous real functions and function intervals. The former is implemented by a floating-point interval arithmetic and the latter is implemented by a polynomial interval arithmetic.

Generalized intervals. The two algebraic structures are closely linked to the continuous lattice of generalized real intervals. A generalized interval (sometimes called directed or modal interval) [Kau80] is a pair $[c, d]$ with no requirement

[9] Freely available from `www.github.com/michalkonecny/polypaver`

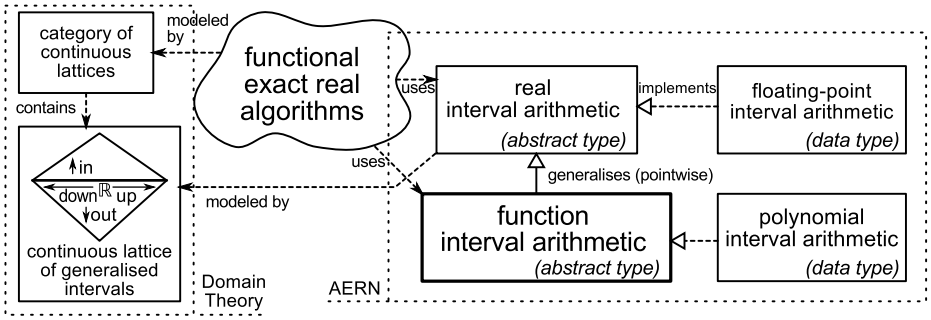

Fig. 5. Main abstract and concrete types in AERN.

that $c \leq d$. If $c > d$, the interval has no canonical set interpretation. The lattice is ordered by the *refinement* relation, denoted $\sqsubseteq$ and defined as follows:

$$[a, b] \sqsubseteq [c, d] \iff (a \leq c \text{ and } d \leq b)$$

If $c \leq d$, then the relation $\sqsubseteq$ can be intuitively interpreted as "$[a, b]$ contains less information than $[c, d]$". The intuition is that the more points an approximating set has, the less it is saying about the location of the approximated object. We consider *inconsistent* intervals, *i. e.*, intervals $[c, d]$ where $c \not\leq d$ because an inner approximation of a normal interval can lead to the bounds of the interval crossing. This happens, for example, near point 0 in Fig. 2 and also when approximating π when proving inclusion (3).

Properties. The abstract types are formalized in the Haskell programming language using its type class feature. A type class is somewhat similar to an interface in object-oriented languages. It facilitates the specification of operations and their signatures. Moreover, AERN also specifies a comprehensive list of algebraic laws that each implementation of the type class should satisfy. For example, one of these laws is the *commutativity of addition*, modified as follows to suit approximate operations:

$$(x + y) \sqsubseteq (y + x)$$

where $+$ and $+$ produce outer and inner approximations of the exact sum, respectively. AERN randomly generates thousands of tests for dozens of such properties using the Haskell QuickCheck library, giving the implementation a very thorough check.

Domain Theory. One of the strongest features of interval-based frameworks is their solid theoretical foundation. In particular, we rely on the rich theory of *continuous lattices* and *continuous partial orders* [GHK+03], which possess useful order-theoretic, algebraic, and topological properties.

The types in Fig 5 implement an algebraic structure on the set of generalized real and function intervals. The operations are all isotonic with respect

to the refinement relation $\sqsubseteq$, which facilitates reasoning about soundness and convergence of the resulting algorithm implementations.

One of the most important properties of continuous lattices is that they form a Cartesian-closed category. This means that the function spaces in the category are also continuous lattices and we can use the rich mathematical theory of such lattices over the function spaces as well.

For example, take the space of continuous generalized interval functions. To approximate elements of this space in software, a countable basis is required. The set $\mathcal{B}_{\text{box}}$ of "box approximations"—*i. e.*, piece-wise constant interval functions that have finitely many "steps" and rational coordinates—is a suitable basis for this space.

Nonetheless, from a practical point of view, the basis $\mathcal{B}_{\text{box}}$ is far from ideal. For example, to enclose a linear function $y(x) = a_0 + a_1 x$ with accuracy $\leq 2^{-n}$, one generally needs $O(2^n)$ boxes, while a more succinct representation could be devised, such as the pair (a_0, a_1). A similar argument can be made using a quadratic function approximated by affine intervals except that one needs $O(\sqrt{2^n})$ intervals to get accuracy $\leq 2^{-n}$. The trade-off between space and accuracy improves with increasing polynomial degree.

The above observation suggests that polynomial approximations provide a more practical basis for enclosing functions than $\mathcal{B}_{\text{box}}$. This view is also supported by the result that, using a specific polynomial approximation, the ODE IVP $y'(x) = f(x), y(0) = 0$ can be solved in polynomial time if f is a polynomial-time computable real function [MM93].

5 Technical contribution

A number of challenges in developing AERN have been related to polynomial approximation of various operations. Perhaps surprisingly, the most complex operation to implement was multiplication. A multiplication of generalized intervals was first introduced by Warmus [War56], but the $\sqsubseteq$-isotonic version that we have adapted was given by Kaucher [Kau80]. The operation is defined in 16 cases, distinguished by the signs of the endpoints of the two interval operands. The challenge is that when the operands are function intervals, the sign of their endpoints may be changing at different places over the domain of their variables. Thus an arbitrary subset of the 16 cases can arise for one pair of functions. This problem is solved in AERN by merging the formulas for the results in all cases that cannot be ruled out using pointwise min and max. Two of Kaucher's 16 original cases also contain min and max in the formulas.

The challenge of implementing pointwise approximate min and max for polynomials has been addressed by a combination of Bernstein approximation and domain translation. The degree of Bernstein approximation used for min and max is one of the effort parameters for any expression that includes multiplication. The enclosures of the product in Fig. 2 have been computed with the use of pointwise min and max.

Almost all polynomial operations have the potential of exceeding the maximum degree and maximum term size limits. To approximate a polynomial by another with lower degree or fewer terms, some terms are carefully eliminated. The version of AERN used in PolyPaver uses the Chebyshev basis to reduce the loss of accuracy due to degree reductions. We plan to port this feature to the main AERN library.

Another challenge was implementing random generation of floating-point intervals and polynomial intervals required for randomized testing of algebraic properties. The generation in AERN produces a distribution of intervals that contains singletons, consistent non-singletons and anti-consistent non-singletons with equal probability. Moreover, special values such as 0 and 1 are generated with a relatively high probability.

References

[CS93] João L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics, October 1993. Presented at SIBGRAPI'93, Recife, PE (Brazil).

[DK14a] Eva Darulova and Viktor Kuncak. Sound compilation of reals. *SIGPLAN Not.*, 49(1):235–248, January 2014.

[DK14b] Jan Duracz and Michal Konečný. Polynomial function intervals for floating-point software verification. *Annals of Mathematics and Artificial Intelligence*, 70:351–398, 2014.

[EP07] Abbas Edalat and Dirk Pattinson. A domain-theoretic account of Picard's theorem. *LMS Journal of Computation and Mathematics*, 10:83–118, 2007.

[GHK+03] G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. W. Mislove, and D. S. Scott. *Continuous Lattices and Domains*, volume 93 of *Encycloedia of Mathematics and its Applications*. Cambridge University Press, 2003.

[Kau80] E. Kaucher. Interval analysis in the extended interval space IR. *Computing Suppl*, 2:33–49, 1980.

[KTD+13] Michal Konečný, Walid Taha, Jan Duracz, Adam Duracz, and Aaron Ames. *Enclosing the behavior of a hybrid system up to and beyond a Zeno point*, pages 120–125. IEEE, 2013.

[MB02] Kyoko Makino and Martin Berz. New applications of Taylor model methods. In *Automatic Differentiation of Algorithms: From Simulation to Optimization*, chapter 43, pages 359–364. Springer, 2002.

[MB09] Kyoko Makino and Martin Berz. Rigorous integration of flows and odes using taylor models. In *Proceedings of the 2009 Conference on Symbolic Numeric Computation*, SNC '09, pages 79–84, New York, NY, USA, 2009. ACM.

[MM93] Norbert Müller and Bernd Moiske. Solving initial value problems in polynomial time. In *Proc. 22 JAIIO - PANEL '93, Part 2*, pages 283–293, 1993.

[Mül01] Norbert Th. Müller. The iRRAM: Exact arithmetic in C++. In *Selected Papers from the 4th International Workshop on Computability and Complexity in Analysis (CCA)*, volume 2064, pages 222–252. Springer-Verlag, 2001. Lecture Notes in Computer Science.

[Tuc02] Warwick Tucker. A rigorous ODE solver and Smale's 14th problem. *Foundations of Computational Mathematics*, pages 53–117, 2002.

[War56] Mieczyslaw Warmus. Calculus of approximations. *Bull. Acad. Polon. Sci. Cl. III*, IV(5):253–259, 1956.