Internship Proposal
Acceleration of Interval Contractors

Supervisors: Christophe Jermann, Anastasia Volkova
Université de Nantes, LS2N
christophe.jermann@univ-nantes.fr, anastasia.volkova@univ-nantes.fr

Keywords: interval arithmetic, certified optimization, HC4 contractor, acceleration, elementary functions

1 Context of the Internship

Interval arithmetic is a way to handle uncertainty or imprecision in numerical data that is bound to happen when using computers to calculate with real numbers. Interval contractors are numerical algorithms that aim at refining the interval evaluation of real equations or inequalities given intervals for the involved variables. They are employed in interval-based branch&prune or branch&bound algorithms that tackle continuous, nonlinear, satisfaction or optimization problems. They are generally implemented in a fixed-point-type algorithm which repeats as long as a sufficient contraction is obtained. These reiterations involve intensive interval operations and can be costly in terms of computation time. Moreover, they are applied at each node of the search-tree followed by the solving process, sometimes without even producing significant contractions, especially in initial phases when the domains are still large.

This internship is interested in the use of an interval arithmetic parameterized by the precision for the sake of acceleration of such contractors. Indeed, for complicated interval operations, such as trigonometric functions, it is possible to use less accurate but faster approximations. We will study the impact of the chosen precision/accuracy on the result obtained by the contractor and its performance, considering classic contractors implemented in IBEX[1] for a start.

1.1 Interval Contractors

Interval Contractors generally enforce some *local consistency* property on the interval domains of the variables with respect to the constraints (equations and inequalities) of the problem. They implement refutation mechanisms allowing to reduce the bounds of the domains of variables by eliminating solution-less parts.

HC4 [1] is prototypical of such contractors. It is a fixed-point propagation algorithm that establishes *hull consistency* using a tree representation of the expression of the constraints: Each node is a mathematical operator, whose evaluation and projections are implemented using corresponding interval operators; the leaves are the variables, while the root[2] is the relation imposed by the constraint. HC4 alternates *forward* and *backward* phases as long as the domain

[1] http://www.ibex-lib.org

[2] In practice, there can be many roots since all constraints can be encoded in a single DAG, hence sharing common sub-expressions

of some variable is sufficiently reduced, with respect to some prescribed relative or absolute precision. In forward phases the nodes are evaluated from leaves to root, while in backward phases the nodes are projected from root to leaves. Each phase hence induces many interval operations for each node of the expression tree.

1.2 Automatic code generation for mathematical functions

Typical implementations of trigonometric functions in interval arithmetic rely on their floating-point counterparts. Reliable interval arithmetic libraries, such as GAOL[3] which provides the back-end for the IBEX tool, use correctly-rounded double-precision mathematical functions, e.g., using the CRlibm library. These implementations verify that the results of calls to elementary functions, for example $\sin(x)$, are as if they were computed exactly and then rounded only once to the double precision. This guarantee is the strongest one could hope for (and for double precision, the relative error is $\approx 10^{-16}$) but unfortunately it results in relatively slow implementations. However, in interval arithmetic applications the bounds can be computed with extremely high errors, for example around 10^{-3} in early stages of HC4 contraction, and there is no need to be extremely precise with the elementary function calls. Hence, some space for a trade-off becomes available: Instead of using double-precision elementary functions, we can tailor the function implementations to our lower accuracy needs.

Usually, transcendental functions are implemented using a polynomial approximation: The higher the required accuracy, the higher the polynomial degree and the slower the implementations becomes [2]. Hence, it is possible to explore the trade-off between the accuracy and the execution time via the polynomial degree (and some other parameters). The state-of-the-art tool performing an automatic design space exploration and providing a C/C++ code generation is Metalibm[4] [3]. This tool, given an expression of a mathematical function, the domain it is defined on and the required accuracy, generates a custom implementation that is accurate just enough to respect the given error bound. Using the custom implementations of elementary functions has been shown to provide a significant speedup in execution time [4].

2 Objectives of the internship

With this internship we will study the impact of the accuracy of interval bounds on the execution time for the HC4 contractor and develop a heuristic guiding the adaptation of the precision of interval computations to speed up the overall performance.

After an initiation to the domain of the study, the intern will work on one or several aspects of this topic depending on her/his aspirations.

One aspect of the study is the parametrization an existing interval arithmetic library with the required accuracy. Here, the idea is to generate the elementary functions for a set of accuracies (e.g. 8-, 16-, 24-, 32-bits) using the Metalibm tool, and adapt them for the interval operators. This step will provide a necessary testing framework and familiarize the intern with the finite-precision arithmetic and error analysis [5, 6].

A second aspect of the study is to enhance the HC4 algorithm with the error parameters for each interval operation (both forward and backward) and theoretically model the error propagation through the algorithm. This theoretical part will be guided by the simulations using the extended interval operators. This work could be incorporated into the IBEX tool.

[3] https://github.com/goualard-f/GAOL
[4] http://www.metalibm.org/lutetia.html

A third aspect of the study is to develop several heuristics that will guide the precision change for the interval computations within the contraction iterations. Indeed, the first iterations perform large contractions, while tightening the bounds towards the end. We will start off by using a contraction ratio to guide the precision increase, and tune it based on the theoretical error propagation model developed earlier.

Recommended skills

- interval arithmetic, mathematical modeling
- programming experience in C/C++
- notions of computer arithmetic is a plus

Duration and location

∼2 months, Université de Nantes, France

References

[1] F. Benhamou, F. Goualard, L. Granvilliers, and J.-F. Puget. Revising hull and box consistency. In *International conference on logic programming*, pages 230–244, 1999.

[2] J.M. Muller. *Elementary Functions: Algorithms and Implementation*. Birkhäuser Boston, 2016.

[3] Nicolas Brunie, Florent de Dinechin, Olga Kupriianova, and Christoph Lauter. Code generators for mathematical functions. In *22d IEEE Symposium on Computer Arithmetic*, Lyon, France, June 2015. Best paper award.

[4] Eva Darulova and Anastasia Volkova. Sound approximation of programs with elementary functions, 2018.

[5] Jean-Michel Muller et al. *Handbook of Floating-Point Arithmetic*. Birkhäuser Basel, 2nd edition, 2018.

[6] N J Higham. *Accuracy and stability of numerical algorithms (2 ed.)*. SIAM, 2002.