Static Analysis of the Accuracy and Robustness of Finite Precision Implementations

Eric Goubault and Sylvie Putot

CEA, LIST

FLUCTUAT

FLUCTUAT is a static analyzer by abstract interpretation, of C and ADA programs, that infers and checks numerical properties such as:

- Tight enclosure of values of program variables, both in the idealized real number and finite precision semantics
- Rounding error and uncertainty propagation
- Functional proof: method error, implementation error on remarkable identities or expressions

It can also be used for worst-case generation and sensitivity analysis.

ZONOTOPIC ABSTRACT DOMAINS

We developed a version adapted to static analysis of the affine forms introduced by Comba et Stolfi in 1993. These forms describe the sets of values taken by affine combinations of finitely many independent symbolic variables called “noise symbols” (the ε_i) taking their values in the interval [-1, 1]. The concretisations of such forms as subsets of $\mathbb{R}^n$ yield a classical class of polyhedra with a central symmetry called zonotopes.

$$\hat{x} = 20 - 4\varepsilon_1 + 2\varepsilon_3 + 3\varepsilon_4$$
$$\hat{y} = 10 - 2\varepsilon_1 + \varepsilon_2 - \varepsilon_4$$

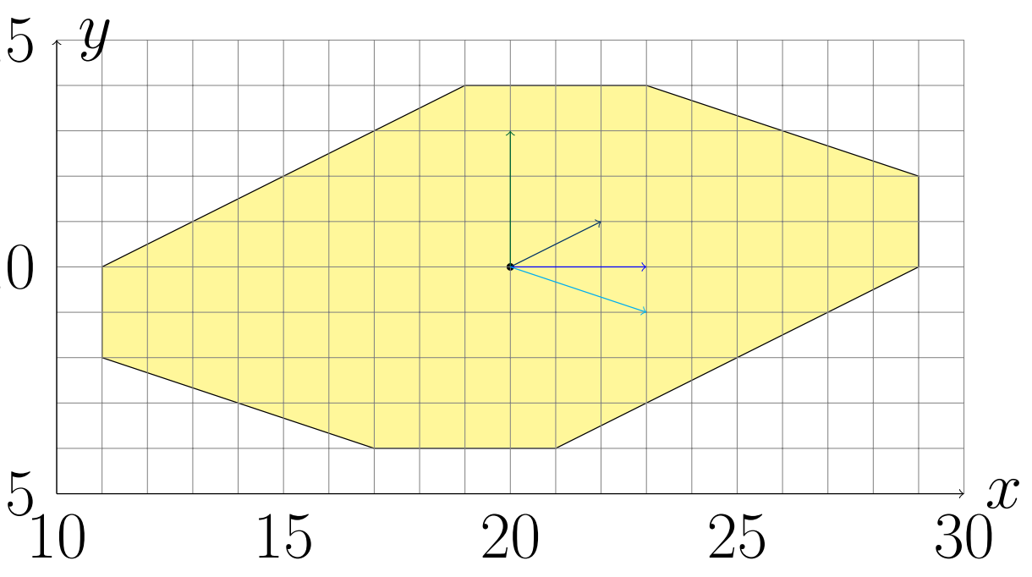


FIGURE 1: Zonotopic concretization in $\mathbb{R}^n$

However, the explicit parametrisation by affine forms shows a strong analogy with the Taylor methods used in guaranteed numerical computations or hybrid systems analysis. Abstract transfert functions are time and space efficient, and provide functional abstractions (i.e. abstraction of the input-output relationship), as in e.g. the interpretation of the expression $y = x^2 - x$ where $x \in [0, 10]$. Variable x is abstracted by $5 + 5\varepsilon_1$ and function $x \rightarrow y = x^2 - x$ is abstracted as:

$$\begin{aligned} y &= 32.5 + 50\varepsilon_1 + 12.5\eta_1 \\ &= -17.5 + 10x + 12.5\eta_1 \end{aligned}$$

The red line below is the linearization of the function (blue dashed lines) and the green zonotope is the concretization of $(\hat{x}, \hat{y})$, its width is given by the green term, in η_1:

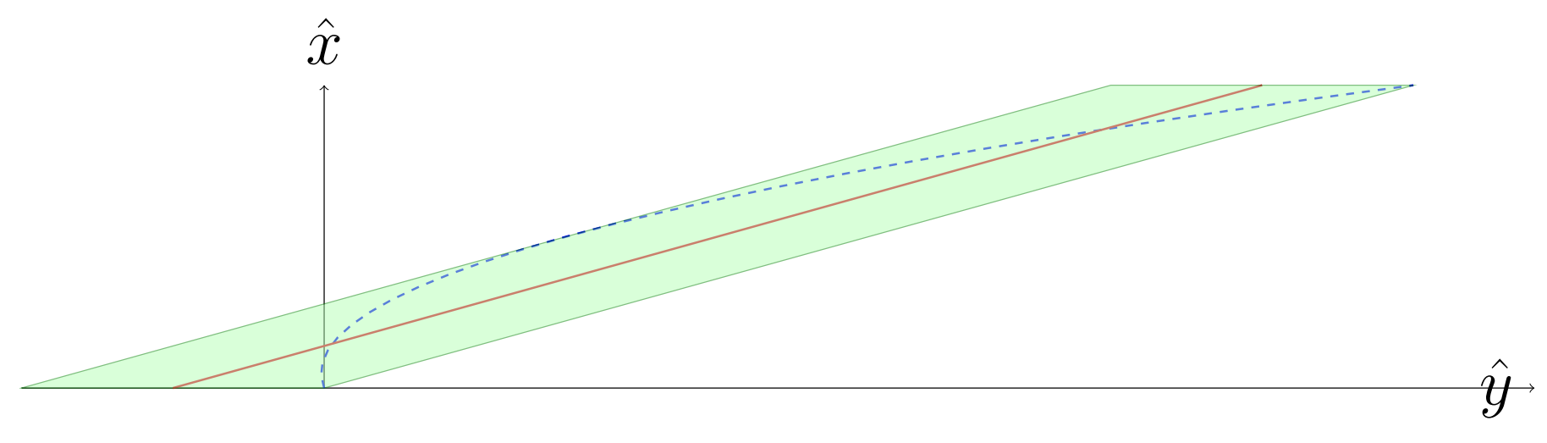


FIGURE 2: Functional interpretation

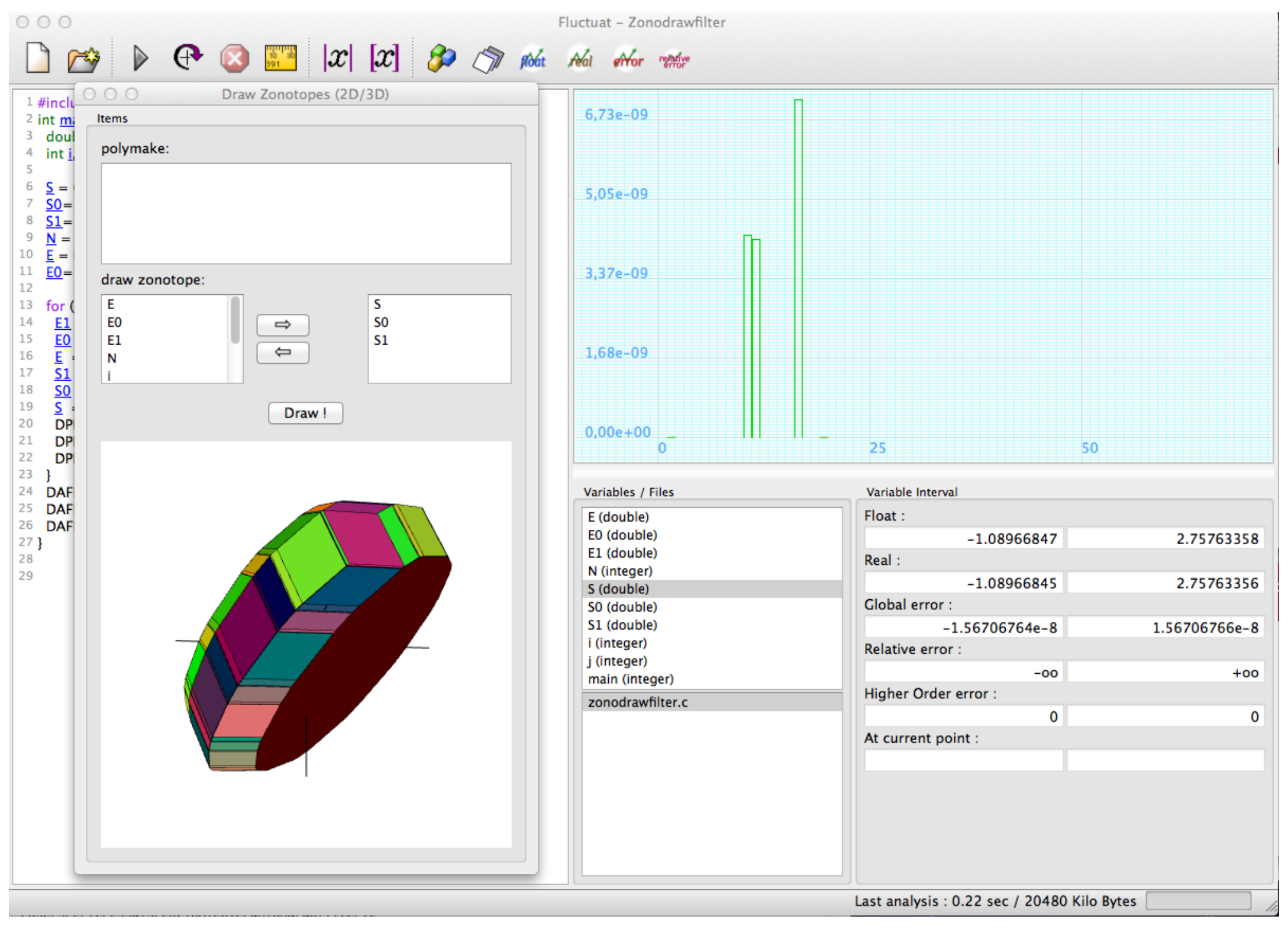
We also designed efficient abstract join and meet operations, useful for interpreting tests, and for computing least fixed points, ie invariants.

INVARIANT SYNTHESIS

The domain is expressive enough to synthetise invariants for many stable recurrent sequences such as the linear recursive filter of order 2

$$S_{n+2} = 0.7E_{n+2} - 1.3E_{n+1} + 1.1E_n + 1.4S_{n+1} - 0.7S_n,$$

for E_n inputs in [-1,1]. Here, Fluctuat is able to determine a zonotopic invariant set of the S_n iterates, as shown on the graphic interface below:



```
#inclu
int ma
 doul
 int i,

 S = 
 S0=
 S1=
 N = 
 E = 
 E0=

 for (
  E1
  E0
  E 
  S1
  S0
  S 
  DP
  DP
  DP
 }
 DAF
 DAF
 DAF
}
```

draw zonotope: E, E0, E1, N, i ⇨ ⇦ S, S0, S1

Variables / Files: E (double), E0 (double), E1 (double), N (integer), S (double), S0 (double), S1 (double), i (integer), j (integer), main (integer), zonodrawfilter.c

Variable Interval
Float : -1.08966847 2.75763358
Real : -1.08966845 2.75763356
Global error : -1.56706764e-8 1.56706766e-8
Relative error : -oo +oo
Higher Order error : 0 0
At current point :

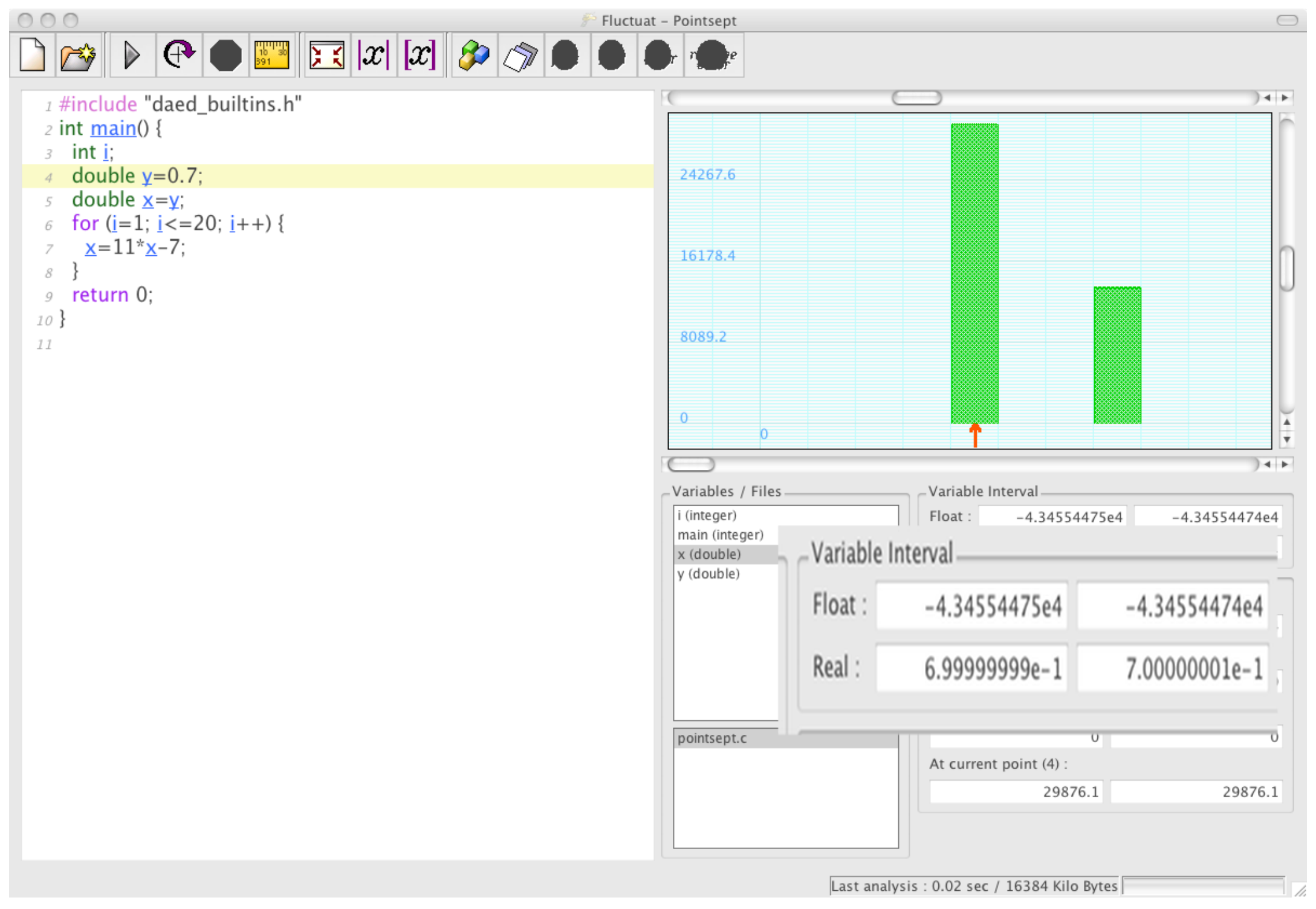
Last analysis : 0.22 sec / 20480 Kilo Bytes

FIGURE 3: Invariant set for tuples (S_n, S_{n+1}, S_{n+2})

ACCURACY OF FINITE PRECISION IMPLEMENTATIONS

Fluctuat considers both real number and finite precision arithmetics, bounds the errors due to the finite precision implementation and traces the source of errors. When a program variable is selected, a graph is displayed, representing the bound for the contribution of each program line on the error on this variable at the end of the program.

For instance below, we see that variable x is very inaccurately computed in floating-point numbers, and that the main source of error comes from the fact that 0.7 is not exactly representable, propagated through computations (cancellation here).



```
#include "daed_builtins.h"
int main() {
  int i;
  double y=0.7;
  double x=y;
  for (i=1; i<=20; i++) {
    x=11*x-7;
  }
  return 0;
}
```

Variables / Files: i (integer), main (integer), x (double), y (double), pointsept.c

Variable Interval
Float : -4.34554475e4 -4.34554474e4
Real : 6.99999999e-1 7.00000001e-1
At current point (4) : 29876.1 29876.1

Last analysis : 0.02 sec / 16384 Kilo Bytes

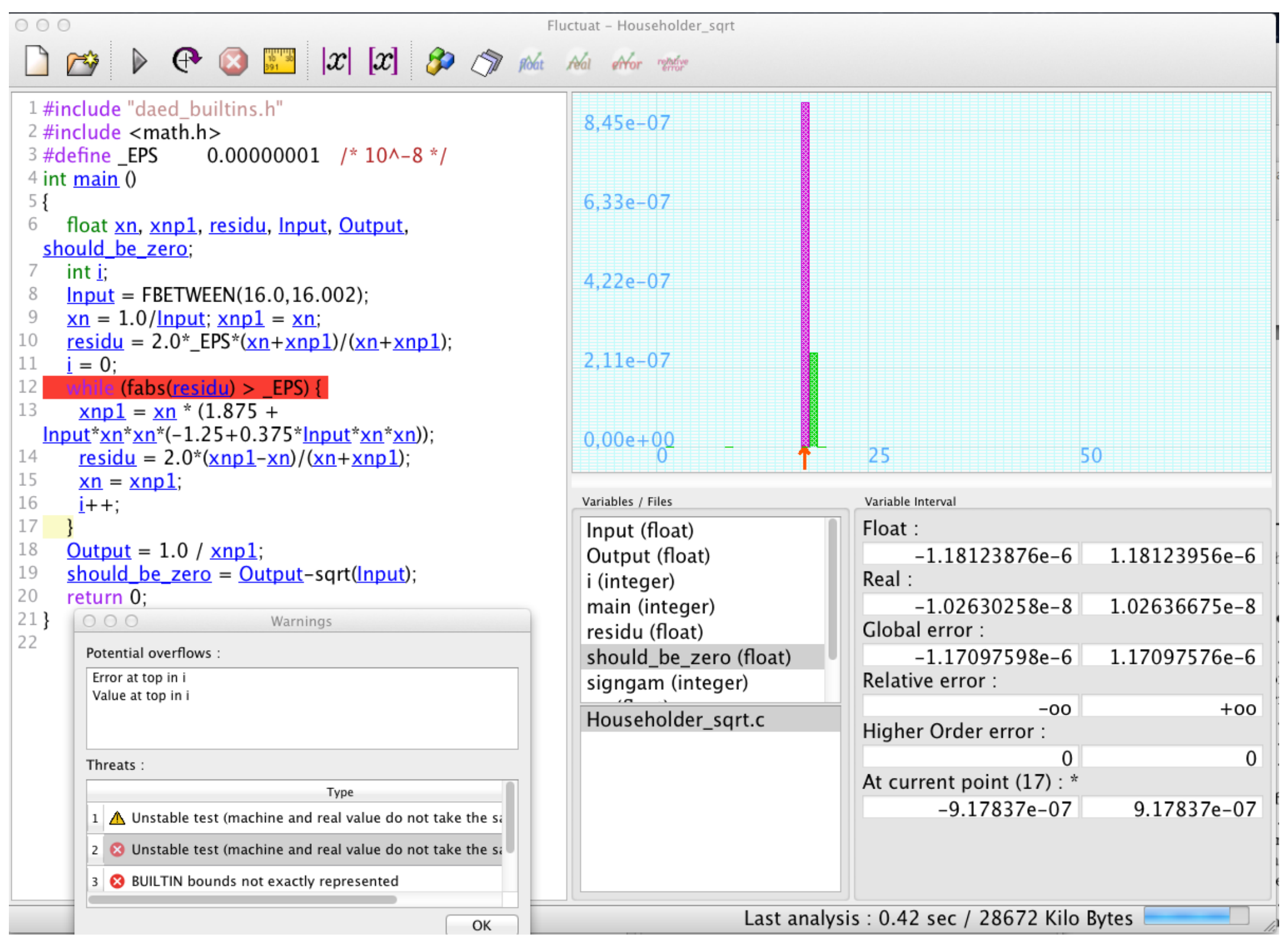
FIGURE 4: Rounding error amplified by cancellation

Uncertainties can also be attached to inputs and parameters, with a special language of assertions, and propagated through computations.

DISCONTINUITY AND ROBUSTNESS ANALYSIS

A recurrent difficulty when we want to assess the influence of finite precision, is the possibility for a test to be unstable: when, for a given input, the finite precision control flow can differ from the control flow that would be taken by the same execution in real numbers. Not taking this possibility into account may be unsound if the difference of paths leads to a discontinuity in the computation. But taking it into account without special care soon leads to large over-approximations.

For instance, we consider a Householder scheme to approximate the square root function. Fluctuat shows a discontinuity error (in purple) on the stopping criterion of the loop, signalling that it may not converge in the same number of iterations in finite precision and in real numbers.



```
#include "daed_builtins.h"
#include <math.h>
#define _EPS      0.00000001  /* 10^-8 */
int main ()
{
  float xn, xnp1, residu, Input, Output,
should_be_zero;
  int i;
  Input = FBETWEEN(16.0,16.002);
  xn = 1.0/Input; xnp1 = xn;
  residu = 2.0*_EPS*(xn+xnp1)/(xn+xnp1);
  i = 0;
  while (fabs(residu) > _EPS) {
    xnp1 = xn * (1.875 +
Input*xn*xn*(-1.25+0.375*Input*xn*xn));
    residu = 2.0*(xnp1-xn)/(xn+xnp1);
    xn = xnp1;
    i++;
  }
  Output = 1.0 / xnp1;
  should_be_zero = Output-sqrt(Input);
  return 0;
}
```

Warnings
Potential overflows :
Error at top in i
Value at top in i
Threats :

	Type
1	⚠ Unstable test (machine and real value do not take the s
2	⊗ Unstable test (machine and real value do not take the s
3	⊗ BUILTIN bounds not exactly represented

Variables / Files: Input (float), Output (float), i (integer), main (integer), residu (float), should_be_zero (float), signgam (integer), Householder_sqrt.c

Variable Interval
Float : -1.18123876e-6 1.18123956e-6
Real : -1.02630258e-8 1.02636675e-8
Global error : -1.17097598e-6 1.17097576e-6
Relative error : -oo +oo
Higher Order error : 0 0
At current point (17) : * -9.17837e-07 9.17837e-07

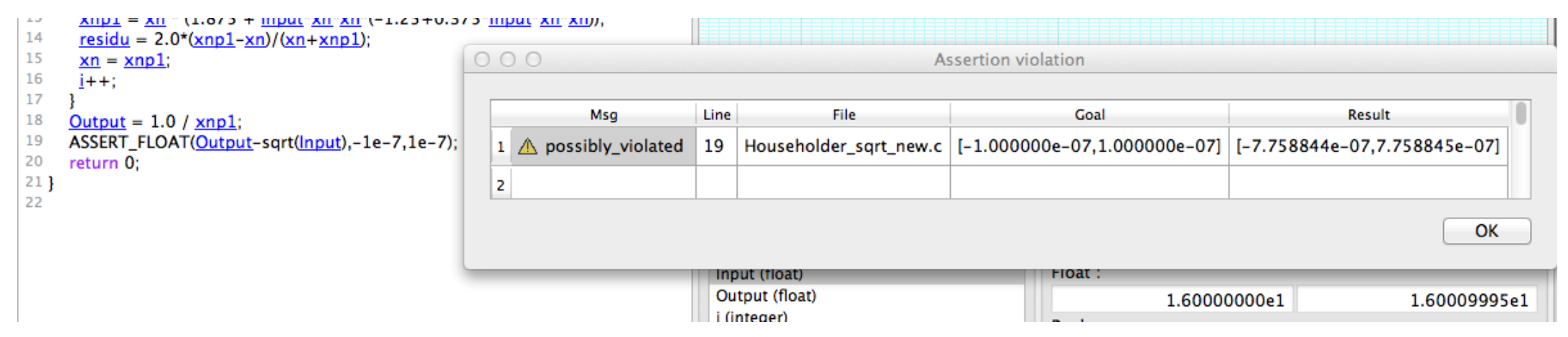
Last analysis : 0.42 sec / 28672 Kilo Bytes

FIGURE 5: Discontinuity error due to the stopping criterion

Our analysis computes unstable tests conditions as the intersection of the constraints on noise symbols induced by the differing path conditions, and deduces tight bounds on the difference between the computation in both branches. We automatically characterize conditional blocks that perform a continuous treatment of inputs, and are thus robust, and those that do not, circumventing the path-space explosion problem that a naive approach would run into.

FUNCTIONAL PROOF

We can also prove functional properties, both in the real number semantics (method error) and in the finite precision semantics (implementation error), for instance here on the Householder scheme for square root:



```
    residu = 2.0*(xnp1-xn)/(xn+xnp1);
    xn = xnp1;
    i++;
  }
  Output = 1.0 / xnp1;
  ASSERT_FLOAT(Output-sqrt(Input),-1e-7,1e-7);
  return 0;
}
```

Assertion violation

	Msg	Line	File	Goal	Result
1	⚠ possibly_violated	19	Householder_sqrt_new.c	[-1.000000e-07,1.000000e-07]	[-7.758844e-07,7.758845e-07]
2					

OK

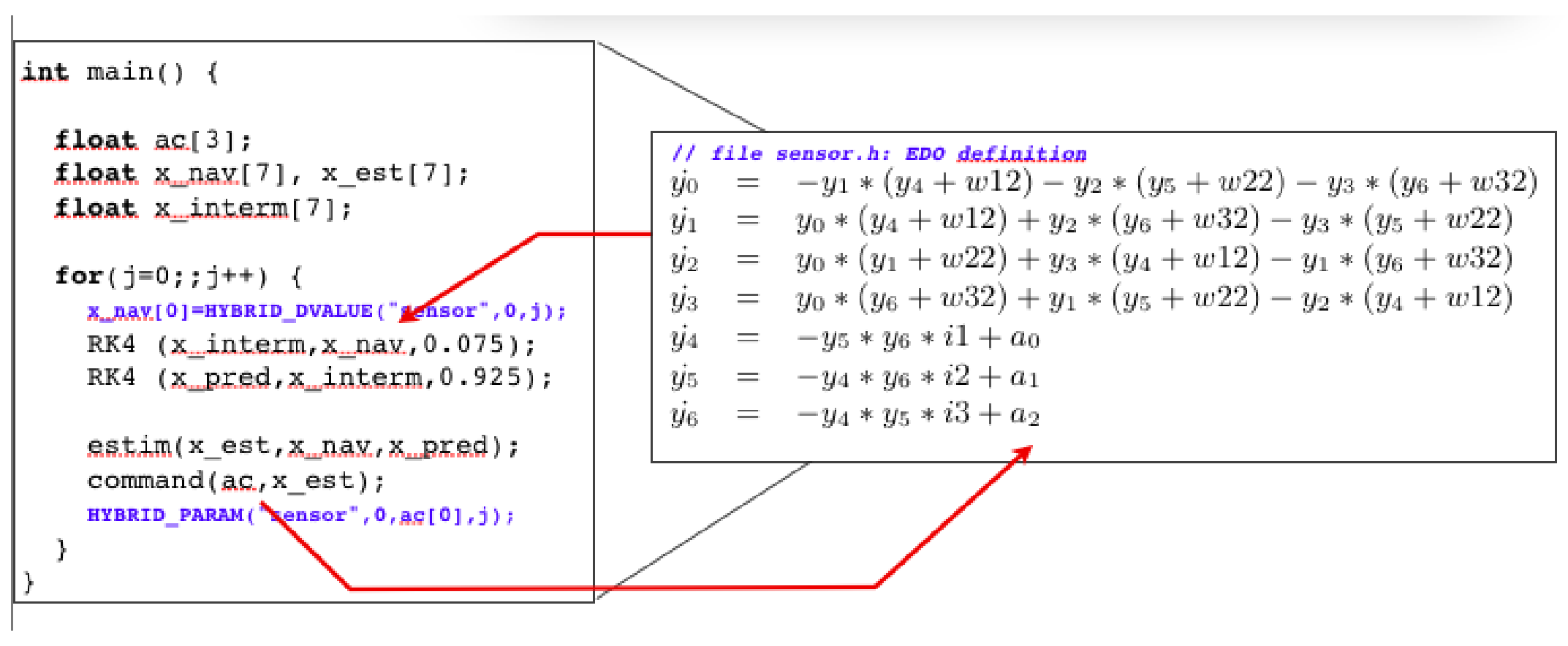
Input (float)
Output (float)
i (integer)

Float : 1.60000000e1 1.60009995e1

HYBRIDFLUCTUAT: extension to hybrid systems analysis

With a connection to a guaranteed ODE solver, we can also analyze hybrid systems such as control systems. The discrete control software is analyzed together with a physical continuous environment, described by switched ODE systems. This helps for instance proving the stability of a closed loop system, that otherwise, in open loop, would be unstable.

The solutions of the ODE systems are abstracted by a call to a guaranteed integration tool, that computes an over-approximation of the actual solution of the ODE. The interaction between the program and the environment are modelled by assertions in the program. The program can change parameters of modes of the ODE system at time t (HYBRID_PARAM, modelling actuators). Conversely, the program can request the value of a continuous variable at time t (HYBRID_DVALUE, modelling sensors).



```
int main() {

  float ac[3];
  float x_nav[7], x_est[7];
  float x_interm[7];

  for(j=0;;j++) {
    x_nav[0]=HYBRID_DVALUE("sensor",0,j);
    RK4 (x_interm,x_nav,0.075);
    RK4 (x_pred,x_interm,0.925);

    estim(x_est,x_nav,x_pred);
    command(ac,x_est);
    HYBRID_PARAM("sensor",0,ac[0],j);
  }
}
```

// file sensor.h: EDO definition

$$\begin{aligned}
\dot{y_0} &= -y_1 * (y_4 + w12) - y_2 * (y_5 + w22) - y_3 * (y_6 + w32) \\
\dot{y_1} &= y_0 * (y_4 + w12) + y_2 * (y_6 + w32) - y_3 * (y_5 + w22) \\
\dot{y_2} &= y_0 * (y_1 + w22) + y_3 * (y_4 + w12) - y_1 * (y_6 + w32) \\
\dot{y_3} &= y_0 * (y_6 + w32) + y_1 * (y_5 + w22) - y_2 * (y_4 + w12) \\
\dot{y_4} &= -y_5 * y_6 * i1 + a_0 \\
\dot{y_5} &= -y_4 * y_6 * i2 + a_1 \\
\dot{y_6} &= -y_4 * y_5 * i3 + a_2
\end{aligned}$$

FIGURE 6: Schematic model of the control software of the ATV

We could thus demonstrate convergence towards the safe escape state, in a model of the ATV (automated transfer vehicle for the international space station) escape mechanism.

EXTENSIONS

We developped some extensions of the zonotopic abstract domains, still relying on the affine forms parameterization $x = \sum_i x_i \varepsilon_i$.

- *Inner-approximation* : with interval or zonotopic coefficients x_i, and interpretation in Kaucher arithmetic, we get generalized affine sets for inner-approximation. We can use them to determinine sets of values of the outputs, that are sure to be reached for some inputs in the specified ranges.
- *Probabilistic affine forms* : with noise symbols ε_i coding sets of probability distributions (P-boxes, Dempster-Shafer structures on intervals), we get probabilistic affine forms. They model both non-deterministic and probabilistic uncertainties, and deliver guaranteed sets of distribution of values of program variables. For instance, on the linear filter example, we can prove that reachable values for the output are deterministically in [-1.09,2.76], but outside [-0.25,1.75] only with very low probability.

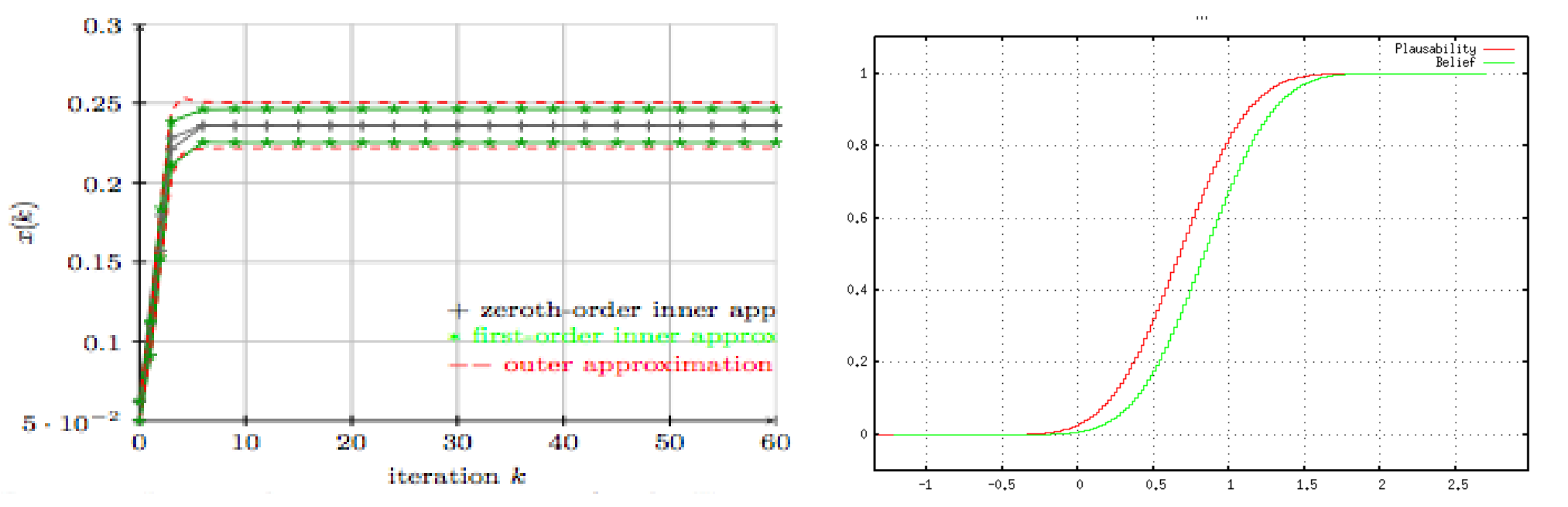


FIGURE 7: Left. Inner and over approximation of the iterates for the Householder scheme. Right. Sets of cumulative distribution functions for the linear filter.

SELECTED PUBLICATIONS

[1] A. Adjé, O. Bouissou, J. Goubault-Larrecq, E. Goubault, and S. Putot. Analyzing probabilistic programs with partially known distributions. In *VSTTE*, 2013.

[2] O. Bouissou, E. Conquet, P. Cousot, R. Cousot, K. Ghorbal, D. Lesens, S. Putot, and M. Turin. Space software validation using abstract interpretation. In *DASIA'09*.

[3] O. Bouissou, E. Goubault, S. Putot, K. Tekkal, and F. Védrine. Hybridfluctuat: A static analyzer of numerical programs within a continuous environment. In *CAV'09*.

[4] D. Delmas, E. Goubault, S. Putot, J. Souyris, K. Tekkal, and F. Védrine. Towards an industrial use of fluctuat on safety-critical avionics software. In *FMICS*, 2009.

[5] E. Goubault and S. Putot. Static analysis of numerical algorithms. In *Proceedings of Static Analysis Symposium, LNCS 4134*, pages 18–34. Springer-Verlag, 2006.

[6] E. Goubault and S. Putot. Under-approximations of computations in real numbers based on generalized affine arithmetic. In *SAS*, pages 137–152, 2007.

[7] E. Goubault and S. Putot. A zonotopic framework for functional abstractions. *CoRR*, abs/0910.1763, available at http://arxiv.org/abs/0910.1763, 2009.

[8] E. Goubault and S. Putot. Static analysis of finite precision computations. In *Proceedings of VMCAI*, volume 6538 of *LNCS*, pages 232–247. Springer, 2011.

Mail: {Eric.Goubault,Sylvie.Putot}@cea.fr