Numerical filter code analysis

Frama-C Days 2024

Franck Vedrine [1] Pierre-Yves Piriou [2] Vincent David [3]

[1]CEA - List - LSL

[2]EDF Lab Chatou - PRISME - P11

[3]IRSN

14/06/24

Plan

Plan

Numerical filter

- A **(discrete-time) signal** $x : \mathbb{Z} \to \mathbb{R}$ maps discrete times to real values.
- A **filter** $\mathscr{H} : (\mathbb{Z} \to \mathbb{R}) \to (\mathbb{Z} \to \mathbb{R})$ maps an input signal u into an output signal y (or vector of signals).

$$\xrightarrow{u(k)} \boxed{\mathscr{H}} \xrightarrow{y(k)}$$

- We restrict here to **Linear Time Invariant** (LTI) filters of finite **order** n. Its canonical form is a constant-coefficient difference equation.

LTI filter

$$y(k) = \sum_{i=0}^{n} b_i.u(k-i) - \sum_{i=1}^{n} a_i.y(k-i)$$

where $\{a_i\}_{1 \leq i \leq n}$ and $\{b_i\}_{0 \leq i \leq n}$ are constant real coefficients.

EDF context

- Qualification of critical control systems for nuclear power plant
- We must verify the absence of RTE (*RunTime Error*)
- In particular, the absence of **numerical overflows**
- A control program classically starts with the filtering of sensor signals
- So the proven filter bounds should be as accurate as possible!
- Toy example :

```
y = filter(input) ;
//@ assert y <= 49.9 ;
output = 1.0 / (50.0 - y) ;
```

State of the art

- More than two decades of effort by the abstract interpretation community, that produced a number of tuned **relational numerical abstract domains**, implemented in Astree or Polyspace:
  - Ellipsoids [2, 3, 10]
  - Zonotopes [1]
  - Set of invariants (Boxes, zonotopes, polyhedra...) [7, 9]
- None of them are currently mature in Frama-C
- Current approch:
  1. Pre-compute the invariant once and for all outside Frama-C
  2. Import the invariant as an ACSL annotation
  3. Check the invariant within Frama-C (Eva or Wp)
- By the way: recent breakthrough in the arithmetic community [5]

Bounding filter output

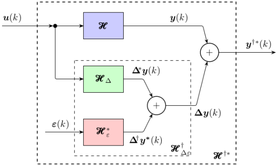
Worst-Case Peak-Gain (WCPG) Theorem

$$\|\mathscr{H}(u)\|_\infty \leq \|\mathbb{h}\|_1 . \|u\|_\infty$$

where $\mathbb{h} = \mathscr{H}(\delta)$ is the **impulse response** of the filter (δ is the impulse signal: $\delta(0) = 1$ and $\forall k \neq 0, \delta(k) = 0$).

- $\|\mathbb{h}\|_1$ is called the **Worst-Case Peak-Gain** (WCPG) and can be correctly computed [12].
- This bound is **optimal** and **reachable** [4].

- To take into account encoding error, we must introduce two additional error filters: $\mathscr{H}_\Delta$ (coefficient quantization) and $\mathscr{H}^*_\varepsilon$ (roundoff error), where ε is the error signal depending on the implementation details (algorithm, machine arithmetic...) [5].



Plan

Warmup

Example of 1st order filter

$$\begin{cases} \forall k & u(k) \in [-10, 10] \\ \forall k < 0 & y(k) = 0 \\ \forall k \geq 0 & y(k) = 0.9y(k-1) + 0.5u(k) \end{cases}$$

$$\begin{aligned} & \mathbb{h}(k) = 0.5 * 0.9^k \\ \Longrightarrow \quad & \|\mathbb{h}\|_1 = 0.5 \sum_{k=0}^{\infty} 0.9^k = 0.5 \frac{1}{1-0.9} = 5 \\ \Longrightarrow \quad & \|y\|_\infty \leq 5\|u\|_\infty = 50 \end{aligned}$$

- 1st order filters can be abstracted in interval domain:

$$0.9 * [-50, 50] + 0.5 * [-10, 10] = [-50, 50]$$

- This is also correct in floating point arithmetic !

Demo 1

```c
double out1;
const double ki[] = { 0.5 };
const double ko[] = { 1.0, -0.9 };
int main() {
  double in=0, out=0;
  //@ widen_hints out, -50.0, 50.0;
  while (Frama_C_interval(0, 1)) {
    out1 = out;
    in = Frama_C_double_interval(-10.0, +10.0);
    out = -ko[1]*out1 + ki[0]*in;
  }
}
```

The command

```
$ frama-c -eva filter1.c
```

produces the following output:

```
[eva] ====== VALUES COMPUTED ======
[eva:final-states] Values at end of function main:
  y ∈ [-50. .. 50.]
```

Plan

Increasing the complexity

Example of 2nd order filter with real eigenvalues

$$\begin{cases} \forall k & u(k) \in [-10, 10] \\ \forall k < 0 & y(k) = 0 \\ \forall k \geq 0 & y(k) = y(k-1) - 0.2 * y(k-2) + 0.1 * u(k) \end{cases}$$

The characteristic polynom is $p(\lambda) = \lambda^2 + \lambda - 0.2$. We can check that its discriminant is $\Delta = 0.2$ and so its eigenvalues are real:

$$\begin{cases} \lambda_1 \approx 0.27639320225 \\ \lambda_2 \approx 0.72360679775 \end{cases}$$

- WCPG theorem application[1]: $\|y\|_\infty \leq 0.5 * \|u\|_\infty = 5$
- Interval domain does not help anymore !

$$[-5, 5] - 0.2 * [-5, 5] + 0.1 * [-10, 10] = [-8.5, 8.5] \not\subseteq [-5, 5]$$

[1] WCPG can be safely computed using https://github.com/fixif/WCPG

Wp comes to rescue

Linear decomposition theorem

A filter of order n having only real eigenvalues can be decomposed into a linear combination of n filters of order 1.

$$y = \frac{\lambda_1.e_1 + \lambda_2.e_2}{\lambda_1 - \lambda_2}$$

with

$$\begin{cases} e_1(k) & = & \lambda_1.e_1(k-1) + 0.1 * u(k) \\ e_2(k) & = & \lambda_2.e_2(k-1) - 0.1 * u(k) \end{cases}$$

- We can implement this implementation as ghost code and prove it with Wp and external solvers.
- Eva finish the proof to bound e_1, e_2 and then y.
- Limitation: the Wp proof rely on the real model !
- Perspective: use an external mechanized filter theory (like in [4]) to bound the floating-point error and import the proof in Wp .

Demo 2

The command

```
$ frama-c -wp -wp-prover altergo,z3 -wp-model real
filter2-fc.c -then -eva filter2-fc.c
```

produces the following output:

```
[wp] Proved goals:    17 / 18
  Qed:               13   (4ms-25ms-52ms)
  Alt-Ergo 2.3.3:     2   (164ms-176ms) (127) (interrupted: 1)
  Z3 4.8.6:           2   (50ms-410ms) (1076863) (interrupted: 1)

[eva] ====== VALUES COMPUTED ======
[eva:final-states] Values at end of function main:
  e1 ∈ [-1.38196601125 .. 1.38196601125]
  e2 ∈ [-3.61803398875 .. 3.61803398875]
  y  ∈ [-6.7082039325 .. 6.7082039325]
```

Plan

Serious business

Example of 2nd order filter with complex eigenvalues

$$\begin{cases} \forall k & u(k) \in [-10, 10] \\ \forall k < 0 & y(k) = 0 \\ \forall k \geq 0 & y(k) = 1.5 * y(k-1) - 0.75 * y(k-2) + 0.5 * u(k) \end{cases}$$

- We can check that $\Delta = -0.75 < 0$
- WCPG theorem application: $\|y\|_\infty \leq 4.91892\|u\|_\infty = 49.1892$
- For this kind of filter, we have to generate relational invariants:
  - ellipsoid
  - set of boxes
  - zonotope
  - fractal zonotope

Ellipsoïd

- **Idea:** Find an invariant of the form [3, 10]:

$$y(k)^2 + a * y(k-1)^2 + b * y(k) * y(k-1) + c * y(k) + d * y(k-1) \leq e$$

  - Invariant found for our toy example:

$$y(k)^2 + 0.75 * y(k-1)^2 - 1.5 * y(k) * y(k-1) \leq 1239.81$$

- **Advantages:** short, readable and verifiable by hand invariant
- **Drawbacks:**
  - Invariant verification with Wp is hard [8]
  - rely on an external polynomial solver
  - don't work if the norm of the complex eigenvalues are too close to 1
  - worst accuracy $\longrightarrow |y(k)| \leq 81.3162$ (recall: $\|y\|_\infty = 49.1892$)

Set of boxes

- **Idea:** state partitioning abstraction to compute an invariant of kind **disjunction of intervals** [9]
  - Divide and conquer (split the input and state intervals in many chunks)
  - The trick comes from constraint programming research field
- **Advantages:**
  - fully automatic: no need for a priori knowledge
  - verifiable with Eva within interval domain $\longrightarrow$ natively valid in floating-point arithmetic
- **Drawbacks:**
  - bad performance $\longrightarrow$ generation : 4s , verification : 62s
  - better but still bad accuracy $\longrightarrow |y(k)| \leq 65$ (recall: $\|y\|_\infty = 49.1892$)

Zonotope

- **Idea:** Find an invariant in an affine form [1]:

$$\bigwedge_{k=0}^{n} y(k) \in \alpha_{k,0} + \sum_{i=1}^{N} \alpha_{k,i} * \varepsilon_i$$

where $\forall i, \varepsilon_i \in [-1, 1]$ are shared **noise symbols**

  - at each iteration, instead of applying classic widening, we introduce a symbolic perturbation over the affine equations, until a post-fixpoint is reached.

- **Advantages:**
  - good accuracy $\longrightarrow |y(k)| \leq 50.32$ (recall: $\|y\|_\infty = 49.1892$)
  - relatively basic reasoning to prove the invariant (linear simplification + interval arithmetic)

- **Drawbacks:**
  - rely on external complex algorithms and heuristics (Fourier-Motzkin, Simplex)
  - don't work if the norm of the complex eigenvalues are too close to 1

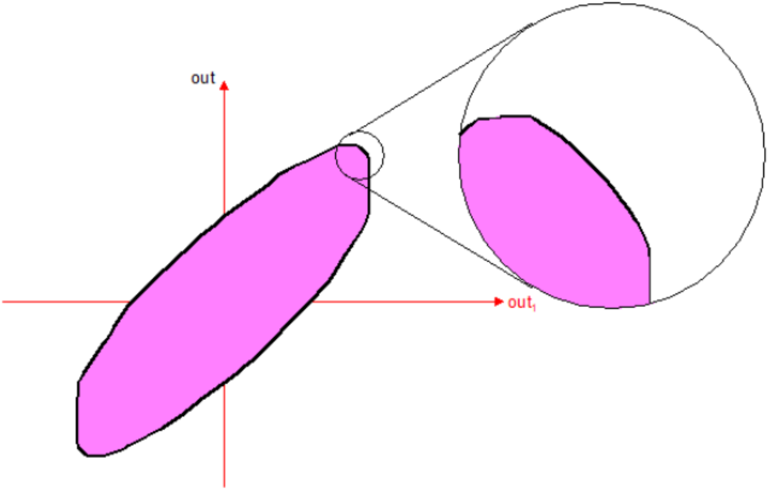
Fractal zonotope

- **Idea:** expliciting the raw relation between y and u, through the impulse response $\mathbb{h}$:

$$y(k) = (\mathbb{h} * u)(k) = \sum_{i=0}^{k} \mathbb{h}(i).u(k-i)$$

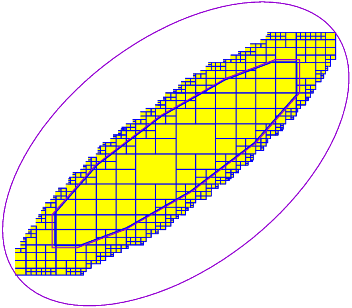
  - Remark: the impulse response $\mathbb{h}$ is an inductive sequence
  - the associated invariant is a fractal zonotope (a kind of "zonotope with infinite vertices").
- **Advantages:**
  - find the optimal bound
  - work in the general case
  - the invariant is automatically computed, without hints
- **Drawbacks:**
  - cannot be automatically verified within Frama-C yet (soon within the Numerors plugin [6])

Fractal zonotope (cont.)



Summary

	ellipsoïd	set of boxes	zonotope	fractal zonotope
accuracy[2]	**worst**	**bad**	**good**	**optimal**
standalone	✘	✔	✘	✔
robustness	✘	∼	∼	✔
by-hand proof	✔	✘	∼	✔
automatic proof	∼(Wp)	✔(Eva)	∼	∼(soon)
float semantic	∼	✔	∼	∼(soon)



[2] Every method is accurate enough to prove that the filter itself does not overflow

Plan

Take away

- Control program starts by filter ⟶ need for **accurate analysis**!
- Arithmeticians are tackling the challenge to design **correct-by-construction filters** (WCPG theorem + sound error analysis) [5, 4, 12] ⟶ reached for fixed-point implementations
- Eva can prove **1st order filter optimal bounds**
- For higher order filter, arbitrary accurate bounds can be automatically obtained within Frama-C using **subdivision techniques**...but it's costly!
- **work-in-progress:** implementing the **fractal zonotope** in Frama-C to infer automatically and efficiently optimal bounds.
- more details on the Frama-C book's chapter 12 and the companion document [11]

Keep going

- Merge effort from arithmetic and abstract interpretation communities.
- Filter detection: manage the diversity of implementations
- Next challenge: **mutable filter**

```
int main() {
  double y, y1, y2, u=0;
  while (Frama_C_interval(0, 1)) {
    u = Frama_C_double_interval(IN_MIN, IN_MAX);
    fast_convergence = Frama_C_interval(0, 1);
    y2 = y1;
    y1 = y;
    if (fast_convergence) {
      y = a11 * y1 + b11 * u ;
    }
    else {
      y = a21 * y1 + a22 * y2 + b21 * u ;
    }
  }
}
```

Annex

Optimal zonotope construction for the toy example

In our toy example, we have:

$$\begin{cases} & \mathbb{h}(0) = 0.5 \\ & \mathbb{h}(1) = 0.75 \\ \forall k & \mathbb{h}(k) = 1.5 * \mathbb{h}(k-1) - 0.75 * \mathbb{h}(k-2) \end{cases}$$

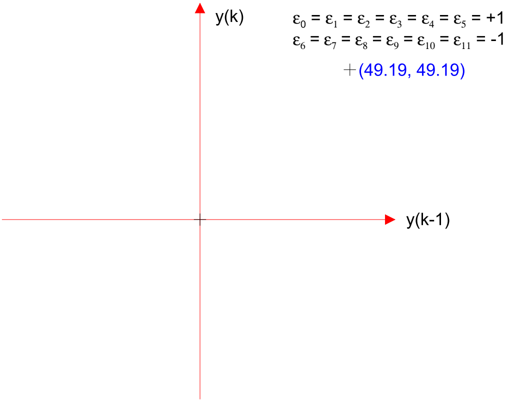
We can check that $\mathbb{h}$ is pseudo-periodic, so the optimal invariant is a classic 12-faces zonotope, described by the following formula (see [11] for details):

$$\begin{cases} y(k) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right)\varepsilon_k \\ y(k-1) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right)\varepsilon_k \end{cases}$$

where $\forall k \in [0, 11], \varepsilon_k \in [-1, 1]$.

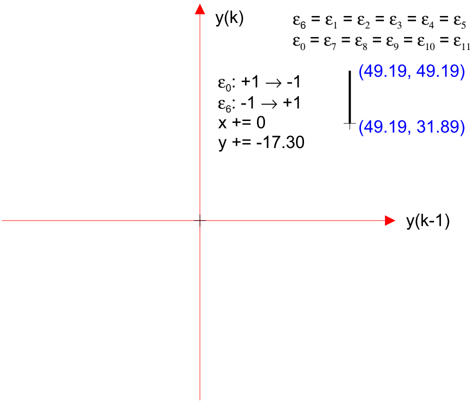
Graphical construction of the zonotope

$$
\begin{cases}
y(k) & \in \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k} \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right) \varepsilon_k \\
y(k-1) & \in \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right) \varepsilon_k
\end{cases}
$$



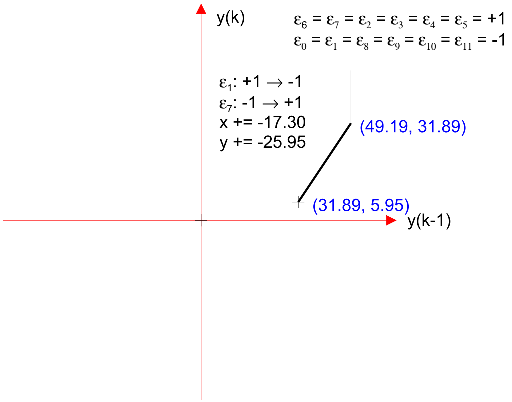
Graphical construction of the zonotope

$$
\begin{cases}
y(k) & \in \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right) \varepsilon_k \\
y(k-1) & \in \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right) \varepsilon_k
\end{cases}
$$



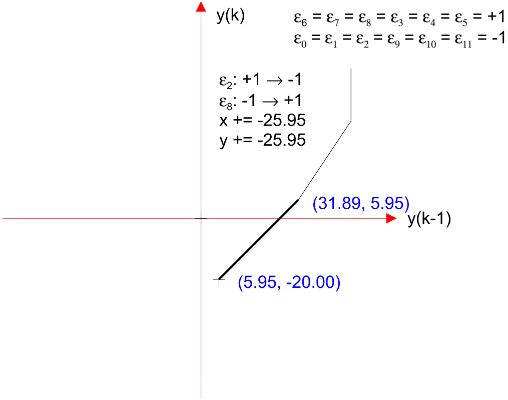
Graphical construction of the zonotope

$$
\left\{
\begin{array}{rcl}
y(k) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right)\varepsilon_k \\
y(k-1) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right)\varepsilon_k
\end{array}
\right.
$$



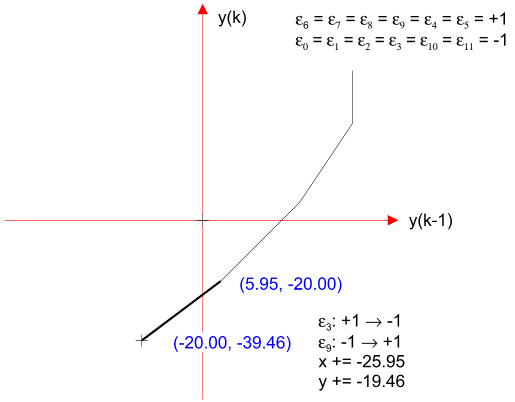
Graphical construction of the zonotope

$$\begin{cases} y(k) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right) \varepsilon_k \\ y(k-1) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right) \varepsilon_k \end{cases}$$



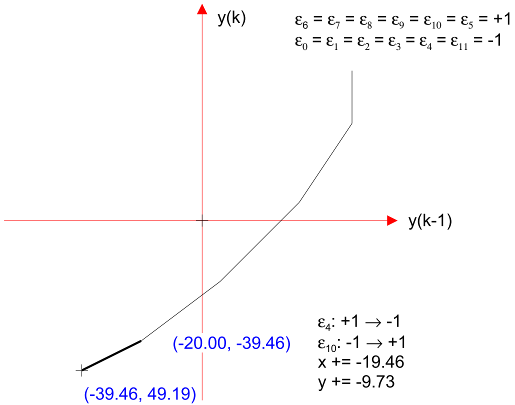
Graphical construction of the zonotope

$$\begin{cases} y(k) & \in \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right)\varepsilon_k \\ y(k-1) & \in \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right)\varepsilon_k \end{cases}$$



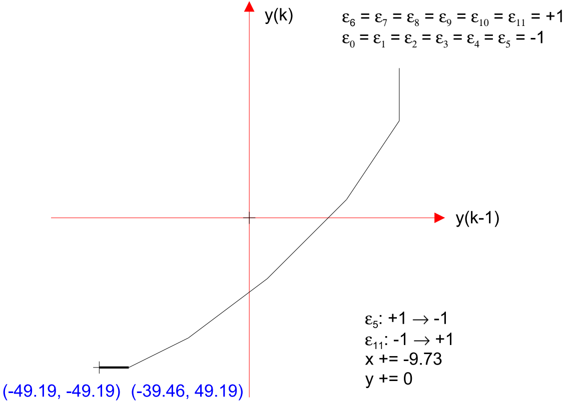
Graphical construction of the zonotope

$$\begin{cases} y(k) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right) \varepsilon_k \\ y(k-1) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right) \varepsilon_k \end{cases}$$



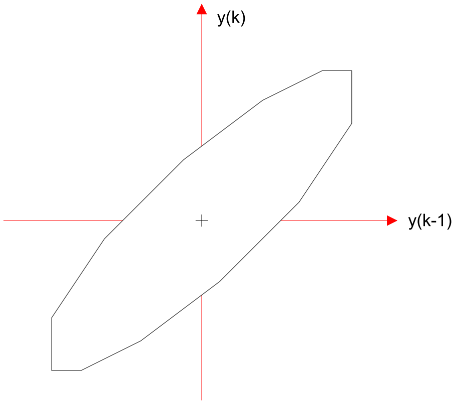
Graphical construction of the zonotope

$$\begin{cases} y(k) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right)\varepsilon_k \\ y(k-1) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right)\varepsilon_k \end{cases}$$



Graphical construction of the zonotope

$$\begin{cases} y(k) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^k \cos\left(\frac{\pi}{3} - k\frac{\pi}{6}\right) \varepsilon_k \\ y(k-1) & \in & \dfrac{\|u\|_\infty}{1-\frac{3^6}{2^{12}}} \displaystyle\sum_{k=0}^{11} \left(\frac{\sqrt{3}}{2}\right)^{k-1} \cos\left(\frac{\pi}{2} - k\frac{\pi}{6}\right) \varepsilon_k \end{cases}$$



Références I

[1] 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 *Formal Methods for Industrial Critical Systems, FMICS*, 2009.

[2] J. Feret.
Static analysis of digital filters.
In *European Symp. on Programming (ESOP)*, 2004.

[3] J. Feret.
Numerical abstract domains for digital filters.
In *European Symp. on Programming (NSAD)*, 2005.

Références II

[4] Diane Gallois-Wong.
Formalisation en Coq des algorithmes de filtre numérique calculés en précision finie. (Coq formalization of digital filter algorithms computed using finite precision arithmetic).
PhD thesis, University of Paris-Saclay, France, 2021.

[5] T. Hilaire.
From filters/controllers to code – contributions to fixed-point arithmetic implementations under accuracy constraint.
Habilitation à Diriger des Recherches (HDR) – Sorbonne Université, 2024.

[6] M. Jacquemin.
Arithmétiques relationnelles pour l'analyse par interprétation abstraite de propriétés de précision numérique. (Relational Arithmetics for Abstract Interpretation Based Analysis of Numerical Accuracy Properties).
PhD thesis, University of Paris-Saclay, France, 2021.

Références III

[7] B. Kabi, E. Goubault, A. Miné, and S. Putot.
Combining zonotope abstraction and constraint programming for synthesizing inductive invariants.
In *Software Verification, VSTTE, and 13th International Workshop, NSV 2020*, 2020.

[8] Elias Khalife, Pierre-Loic Garoche, and Mazen Farhood.
Code-level formal verification of ellipsoidal invariant sets for linear parameter-varying systems.
In *NASA Formal Methods Symposium*, pages 157–173. Springer, 2023.

[9] A. Miné, J. Breck, and T. W. Reps.
An algorithm inspired by constraint solvers to infer inductive invariants in numeric programs.
In *Programming Languages and Systems, ESOP*, 2016.

Références IV

[10] P. Roux, R. Jobredeaux, P.-L. Garoche, and E. Féron.
A generic ellipsoid abstract domain for linear time invariant systems.
In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2012, Beijing, China, April 17-19, 2012*, 2012.

[11] Franck Vedrine, Pierre-Yves Piriou, and Vincent David.
Examples, proofs and algorithms for the verification of loop invariant of linear filters with Frama-C, March 2023.

[12] Anastasia Volkova, Christoph Lauter, and Thibault Hilaire.
Computing the worst-case peak gain of digital filter in interval arithmetic.
In *17th International Symposium on Scientific Computing, Computer Arithmetics and Verified Numerics.*, 2016.