A Generic Ellipsoid Abstract Domain for Linear Time Invariant Systems[*]

Pierre Roux
ONERA
Toulouse, FRANCE
pierre.roux@onera.fr

Romain Jobredeaux
Georgia Tech
Atlanta, Georgia, USA
rjobredeaux3@gatech.edu

Pierre-Loïc Garoche
ONERA
Toulouse, FRANCE
pierre-loic.garoche@onera.fr

Éric Féron
Georgia Tech
Atlanta, Georgia, USA
feron@gatech.edu

ABSTRACT

Embedded system control often relies on linear systems, which admit quadratic invariants. The parts of the code that host linear system implementations need dedicated analysis tools, since intervals or linear abstract domains will give imprecise results, if any at all, on these systems. Previous work by FERET proposes a specific abstraction for digital filters that addresses this issue on a specific class of controllers.

This paper aims at generalizing the idea. It works directly on system representation, relying on existing methods from control theory to automatically generate quadratic invariants for linear time invariant systems, whose stability is provable. This class encompasses n-th order digital filters and, in general, controllers embedded in critical systems.

While control theorists only focus on the existence of such invariants, this paper proposes a method to effectively compute tight ones. The method has been implemented and applied to some benchmark systems, giving good results. It also considers floating points issues and validates the soundness of the computed invariants.

Keywords

stable linear systems, ellipsoids, quadratic invariants, Lyapunov functions, semi-definite programming, floating point errors, abstract interpretation.

1. CONTROL-COMMAND BASED CRITICAL SYSTEMS

A wide range of today's real-time embedded systems, especially their most critical parts, rely on a control-command computation core. The control-command of an aircraft, a satellite or a car engine, is processed into a global loop repeated indefinitely during the activity of the controlled device. This loop models the acquisition of new input values via sensors, the update of internal state variables and the generation of new outputs. The acquisition is made either from environmental measurements (like wind speed, acceleration or engine RPM for instance) or from human input via the brakes, the accelerator, the stick or wheel control.

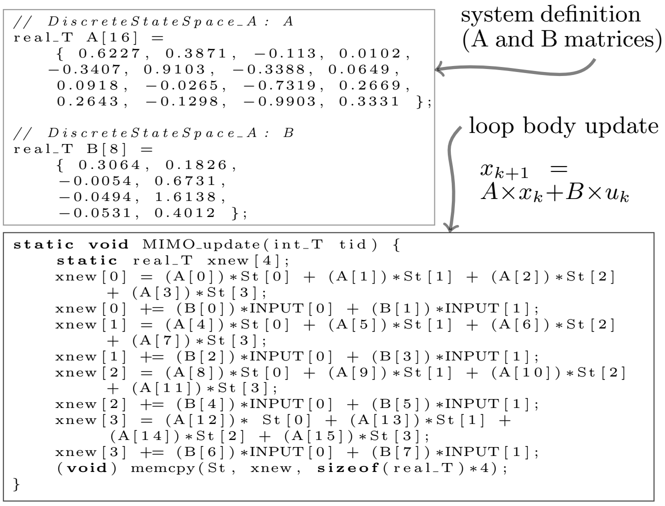
Control theorists are used to model both the environment and the system behavior. Then using their own set of tools, they add the necessary elements to obtain the target controlled system. After discretizing the system, they mathematically prove its stability and its performance by exhibiting a quadratic form, i.e. an ellipsoid, that over-approximates the system behavior with respect to a given input. All these steps are well known by control theory specialists. Reference [15] is a good introduction to these approaches. In this paper, we focus on a class of such systems where control is computed using stable linear systems, i.e. the controller is open-loop stable.

The system control law is then compiled from its description to executable code, like embedded C. This description is usually specified in Matlab Simulink, Scilab Scicos or in a dedicated synchronous language such as Lustre or Scade.

Fig. 1 sketches the loop body of a coupled mass controller, as generated by Matlab. It corresponds to a one step evaluation of the following linear system $x_{k+1} = Ax_k + Bu_k$, where $||u_k||_\infty \leq 1$. Vector x_k represents the state of the system at a given time. Matrix A models the system update according to its previous state, while matrix B expresses the effect of the input values u_k.

Once such controller source code is generated, it is embedded in the controlled device, eg. an aircraft. Critical embedded systems are then a major target for static analysis

[*]This work has been partially supported by the FNRAE Project CAVALE.



```
// DiscreteStateSpace_A : A
real_T A[16] =
    { 0.6227, 0.3871, -0.113, 0.0102,
   -0.3407, 0.9103, -0.3388, 0.0649,
    0.0918, -0.0265, -0.7319, 0.2669,
    0.2643, -0.1298, -0.9903, 0.3331 };

// DiscreteStateSpace_A : B
real_T B[8] =
    { 0.3064, 0.1826,
    -0.0054, 0.6731,
    -0.0494, 1.6138,
    -0.0531, 0.4012 };
```

```
static void MIMO_update(int_T tid) {
    static real_T xnew[4];
    xnew[0] = (A[0])*St[0] + (A[1])*St[1] + (A[2])*St[2]
        + (A[3])*St[3];
    xnew[0] += (B[0])*INPUT[0] + (B[1])*INPUT[1];
    xnew[1] = (A[4])*St[0] + (A[5])*St[1] + (A[6])*St[2]
        + (A[7])*St[3];
    xnew[1] += (B[2])*INPUT[0] + (B[3])*INPUT[1];
    xnew[2] = (A[8])*St[0] + (A[9])*St[1] + (A[10])*St[2]
        + (A[11])*St[3];
    xnew[2] += (B[4])*INPUT[0] + (B[5])*INPUT[1];
    xnew[3] = (A[12])* St[0] + (A[13])*St[1] +
         (A[14])*St[2] + (A[15])*St[3];
    xnew[3] += (B[6])*INPUT[0] + (B[7])*INPUT[1];
    (void) memcpy(St, xnew, sizeof(real_T)*4);
}
```

Figure 1: System update for a coupled mass system controller generated by Matlab.

in order to ensure their good behavior. The success story of Astrée [7] illustrates such needs: it targets the analysis of the control command of the Airbus A380, and was used to formally prove the absence of any runtime error on the 700kloc of the controller source code. It relies on the theory of abstract interpretation [5, 6] to compute a sound overapproximation of all possible values of the program variables in any reachable states. Then it is able to ensure that this over-approximation does not reach any possible bad state like overflows, division by zero, or invalid pointer dereferencing. In [7], the authors enumerate the different abstractions used to compute this over-approximation like intervals or octagons. Among them, we focus here on digital filters abstractions represented by ellipsoid abstract domains.

In [8], Feret proposes an analysis dedicated to stable linear filters in control command programs. These short pieces of code correspond to the kind of systems illustrated in Fig. 1, restricted to only one input argument and its past history, i.e. the matrix B is a column vector, and specific types of matrices A, i.e. companion matrices.

Most of the abstract domains available in actual tools only represent linear properties, leading on our target systems at best to rather costly analysis [11] or at worst to no result at all. For example, *no interval invariant exists* for the example of Fig. 1. Thus analyzing it with intervals will give the $(-\infty; +\infty)$ over-approximation, whereas quadratic invariants will bound all parameters in $[-5; 5]$.

In this paper, we propose to generalize the approach of [8] by considering any inherently stable linear system. In particular, we

- characterize quadratic forms, invariants of the linear system analyzed, with techniques inspired by the control theory community;
- propose an open implementation of the analysis that handles floating point rounding errors;
- validate the result using a sound external solver.

Unlike in [8], current work is supposed to take place during the development process and matrices A and B are assumed to be given.

The paper is structured as follow: Section 2 introduces the reader to the notion of stability based on Lyapunov invariants. Section 3 presents our global approach and the steps of our algorithm. Sections 4, 5 and 6 detail the main steps while Section 7 covers floating point issues and soundness. Finally concrete results and related work are presented in Sections 8 and 9.

2. INTRODUCTION TO LYAPUNOV STABILITY THEORY

One common way to establish stability of a discrete, time-invariant closed (i.e. with no inputs) system described in state space form, (i.e $x_{k+1} = f(x_k)$) is to use what is called a Lyapunov function. It is a function $V : \mathbb{R}^n \to \mathbb{R}$ which must satisfy the following properties

$$V(0) = 0 \land \forall x \in \mathbb{R}^n \backslash \{0\}, V(x) > 0 \land \lim_{||x|| \to \infty} V(x) = \infty \tag{1}$$

$$\forall x \in \mathbb{R}^n, V(f(x)) - V(x) \leq 0. \tag{2}$$

It is shown for example in [13] that exhibiting such a function proves the so-called Lyapunov stability of the system, meaning that its state variables will remain bounded through time. Equation (2) expresses the fact that the function $k \mapsto V(x_k)$ decreases, which, combined with (1), shows that the state variables remain in the bounded sublevel-set $\{x \in \mathbb{R}^n | V(x) \leq V(x_0)\}$ at all instants $k \in \mathbb{N}$.

In the case of Linear Time Invariant systems (of the form $x_{k+1} = Ax_k$, with $A \in \mathbb{R}^{n \times n}$), one can always look for V as a quadratic form in the state variables of the system: $V(x) = x^T Px$ with $P \in \mathbb{R}^{n \times n}$ a symmetric matrix such that

$$P \succ 0 \tag{3}$$

$$A^T P A - P \preceq 0 \tag{4}$$

where "$P \succ 0$" means that the matrix P is positive definite, i.e. for all non-zero vector $x, x^T Px > 0$.

Now, to account for the presence of an external input to the system (which is usually the case with controllers: they use data collected from sensors to generate their output), the model is usually extended into the form

$$x_{k+1} = Ax_k + Bu_k, ||u_k||_\infty \leq 1. \tag{5}$$

To study this equation as precisely as possible, another model, expressing the behavior of the controlled system (the plant), is usually introduced. The two systems taken together form a closed system with no inputs which can be analyzed by looking for a P matrix matching the criteria mentioned before. Such an analysis is refered to as 'closed loop stability analysis'. Here we seek not to model the plant, instead we only require for $||u||_\infty$ to remain bounded[1]. Then, through

[1] While we could consider different bounds for each compo-

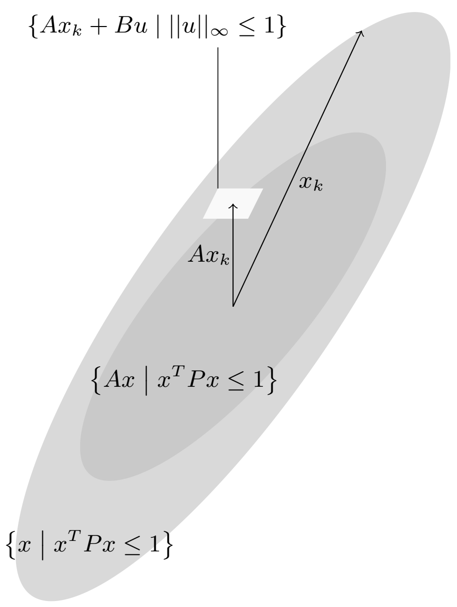

$\{Ax_k + Bu \mid ||u||_\infty \leq 1\}$
x_k
Ax_k
$\{Ax \mid x^T Px \leq 1\}$
$\{x \mid x^T Px \leq 1\}$

Figure 2: Illustration of the stability concepts: if x_k is in the light gray ellipse, then, after a time step, Ax_k is in the dark gray ellipse, which is exactly what is expressed by Equation (4). The white box represents the potential values of x_{k+1} after adding the effect of the bounded input u_k.We see here the necessity that the dark gray ellipse be strictly included in the light gray one, which is the stronger condition expressed by Equation (6).

a slight reinforcement of Equation (4) into

$$A^T P A - P \prec 0 \qquad (6)$$

we can still guarantee that the state variables of (5) will remain in a sublevel set $\left\{x \in \mathbb{R}^n \mid x^T Px \leq \lambda\right\}$ (for some $\lambda > 0$), which is an ellipsoid in this case. This approach only enables us to study control laws that are inherently stable, i.e stable when taken separately from the plant they control. Nevertheless a wide range of controllers remain that can be analyzed, and this encompasses in particular all those handled by Astrée. In addition, inherent stability is required in a context of critical applications.

These stability proofs have the very nice side effect that they provide a quadratic invariant on the state variables, which can be used at the code level to find bounds on the program variables. Furthermore, there are many P matrices that fulfill the equations described above. This gives some flexibility as to the choice of such a matrix: by adding relevant constraints on P, one can obtain increasingly better bounds.

3. OVERALL METHOD

3.1 Separate Shape and Ratio

We keep the same overall representation as Feret [8, 9], representing an ellipsoid by a pair (P, λ) where $P \in \mathbb{R}^{n\times n}$ is a symmetric positive definite matrix giving the *shape* of the ellipsoid and $\lambda \in \mathbb{R}$ a scalar giving its *ratio*. The represented ellipsoid is then the set of all $x \in \mathbb{R}^n$ such that $x^T Px \leq \lambda$, i.e. the concretization function γ is given by $\gamma : (P, \lambda) \mapsto \left\{x \in \mathbb{R}^n \mid x^T Px \leq \lambda\right\}$. To avoid having multiple representations for the same ellipsoid[2] we can normalize P for instance by requiring its largest coefficient to be 1. The underlying lattice also remains the same. In particular the join of two abstract values (P, λ) and (P', λ') is $(P, \max(\lambda, \lambda'))$ if $P = P'$ and $\top$ otherwise.

This seemingly strange choice at first sight allows us to decompose the computation in two successive steps

1. first determine the shape of the ellipsoid by choosing a well suited matrix P;
2. then find the smallest possible ratio λ such that $x \in \gamma(P, \lambda)$ is an invariant.

Various methods for both steps are detailed and compared in Sections 4 and 5.

3.2 Instrumentation: Use of Semidefinite Programming

To perform the aforementioned computations we rely heavily on semidefinite programming [4, 10]. These tools allow us to compute in polynomial time a solution to a linear matrix inequality (LMI) while minimizing a linear objective function. A LMI is an inequality of the form

$$A_0 + \sum_{i=1}^{k} y_i A_i \succeq 0$$

where the A_i are known matrices, the y_i are the unknowns and "$P \succeq 0$" means that the matrix P is positive semidefinite, i.e. $x^T Px \geq 0$ for all vector x. Indeed we can easily have unknown matrices since a matrix $A \in \mathbb{R}^{n\times n}$ can be expressed as $\sum_{i=1,j=1}^{n,n} A_{i,j} E^{i,j}$, where $E^{i,j}$ is the matrix with zeros everywhere except a one at line i and column j. Likewise multiple LMIs can be grouped into one since $A \succeq 0 \wedge B \succeq 0$ is equivalent to $\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \succeq 0$.

We will also have to deal with some implications which will be achieved by transforming them into a LMI thanks to the following theorem.

Theorem 1 (S-Procedure). *For any $P, P' \in \mathbb{R}^{n\times n}$, $a, a' \in \mathbb{R}^n$ and $b, b' \in \mathbb{R}$, following conditions are equivalent*

1. $\forall x \in \mathbb{R}^n$,
   $x^T P\, x + 2a^T x + b \geq 0 \quad \Rightarrow \quad x^T P'\, x + 2a'^T x + b' \geq 0$
2. $\exists \tau \in \mathbb{R}, \quad \tau > 0 \quad \wedge \quad \begin{pmatrix} P' & a' \\ a'^T & b' \end{pmatrix} - \tau \begin{pmatrix} P & a \\ a^T & b \end{pmatrix} \succeq 0$

Proof. Soundness ($2 \Rightarrow 1$) is obvious. A proof of completeness ($1 \Rightarrow 2$) can be found in [15]. □

nent of the input u, we will only deal with $||u||_\infty \leq 1$ for simplicity of the exposition.

[2] For instance $(P, 2\lambda)$ and $\left(\frac{P}{2}, \lambda\right)$ represent exactly the same ellipsoid.

4. SHAPE OF THE ELLIPSOID

As was presented in Section 2, any positive definite matrix P satisfying the Lyapunov equation

$$A^T P A - P \prec 0 \tag{7}$$

will yield a proof of stability and provide *some* bound on the variables. However, additional constraints on P can be introduced that make it possible to obtain better results than others.

While in Control Theory the existence of such ellipsoids is sufficient to prove stability of the system, we are here interested in characterizing it concretely. Investigating heuristically multiple possible shapes allows us to find one which is more adequate, i.e. more precise, with respect to the analyzed system.

The following subsections describe three different types of additional constraints on P and their respective advantages.

4.1 Minimizing Condition Number

Graphically, the condition number of a positive definite matrix expresses a notion similar to that addressed by excentricity for ellipses in dimension 2. It measures how 'close' to a circle (or its higher dimension equivalent) the resulting ellipsoid will be. Multiples of the identity matrix, which all represent a circle, have a condition number of 1. Thus one idea of constraint we can impose on P is to have its condition number as close to 1 as possible. One reason is that 'flat' ellipsoids can yield a very bad bound on one of the variables. This is done [3] by minimizing a new variable, r, in the following matrix inequality

$$I \preceq P \preceq rI.$$

This constraint, along with the others (Lyapunov equation, positive definiteness, ...), can be expressed as an LMI, which is solved using the semi definite programming techniques mentioned in Section 2.

4.2 Preserving the Shape

Another approach [25] is to minimize $r \in (0,1)$ in the following inequality

$$A^T P A - rP \preceq 0.$$

Intuitively, this corresponds to finding the shape of ellipsoid that gets 'preserved' the best when the update $x_{k+1} = Ax_k$ is applied. This is the choice implicitly made in [8] for a particular case of 2×2 matrices A. With this technique however, the presence of a quadratic term rP in the equation prevents the use of usual LMI solving tools 'as is'. To overcome this we chose an approach where we try a value for r and refine it by dichotomy. Only a few steps are required to obtain a good approximation of the optimal value.

4.3 All in One

The two previous methods were based only on A, completely abstracting B away, which could lead to rather coarse abstractions. We try here to take both A and B into account by finding the smallest possible P such that

$$\begin{aligned} \forall x, \forall u, ||u||_\infty \leq 1 \Rightarrow x^T P\, x \leq 1 \Rightarrow \\ (Ax + Bu)^T\, P\, (Ax + Bu) \leq 1 \end{aligned}$$

which, using the S procedure, amounts to the existence of $\tau_i > 0$ such that

$$\begin{pmatrix} -A^T P A & -A^T P B e_i \\ -e_i^T B^T P A & 1 - e_i^T B^T P B e_i \end{pmatrix} - \tau_i \begin{pmatrix} -P & 0 \\ 0 & 1 \end{pmatrix} \succeq 0$$

for all the vertices e_i of the hypercube of dimension p, the number of inputs. The rationale behind this formula is explained in Section 5.2. This is not an LMI since τ and P are both variables but a reasonably good solution can be found by trying various values of τ between some $\tau_{min} \in (0,1)$, which can be found by dichotomy, and 1.

4.4 Comparison and Combination

There is no proof that one method always performs better than the others, and, for each method, there exists examples where it performs better than the other two, see Section 8. It appears, however, that the third method, albeit a little more costly, yields the best bounds in general. In fact the cost is also debatable since, despite being costlier, it does not require the search for the ratio, a necessary step for the first two methods described in Sections 4.1 and 4.2.

In any case, the methods are not exclusive of each other and can be combined: the resulting (sound) value will be the intersection of the projection of each obtained ellipsoids. Having multiple, not-always-comparable values will only yield more precise results.

5. FINDING A STABLE RATIO

Now that we have chosen a matrix P, we need to find a ratio λ such that $x^T P\, x \leq \lambda$ is an invariant for the whole system $x_{k+1} = Ax_k + Bu_k$ with a bounded input u that satisfies $||u||_\infty \leq 1$. The existence of such a λ is guaranteed by the choice of P as a solution of the Lyapunov inequality (7). Those λ are exactly the fixpoints of the function mapping λ_k to the maximum of λ_k and the least λ_{k+1} such that

$$\begin{aligned} \forall x_k \in \mathbb{R}^n, u_k \in \mathbb{R}^p, ||u_k||_\infty \leq 1 \Rightarrow x_k^T P\, x_k \leq \lambda_k \Rightarrow \\ x_{k+1}^T P\, x_{k+1} \leq \lambda_{k+1} \end{aligned} \tag{8}$$

where $x_{k+1} = Ax_k + Bu_k$. We are of course interested in the least fixpoint.

5.1 Initial Ratio λ_0

Since the system starts in state x_0, we initialize λ_0 as $x_0^T P x_0$. If instead of a simple point the initial conditions are only known to lie in a polyhedron, we just have to take the maximum of $x^T P x$ among all vertices x of the polyhedron.

5.2 One Iteration

Given some λ_k, we want to compute the least λ_{k+1} satisfying Equation (8). By a convexity argument[3], it is enough to have the following for every vertex $e_i, i \in [\![1, 2^p]\!]$ of the hypercube[4] $\{u_k \mid ||u_k||_\infty \leq 1\}$ of dimension p

$$\begin{aligned} &\forall x_k \in \mathbb{R}^n, x_k^T P\, x_k \leq \lambda_k \Rightarrow \\ &(Ax_k + Be_i)^T\, P\, (Ax_k + Be_i) \leq \lambda_{k+1}. \end{aligned}$$

[3] See Figure 2 for a graphical illustration of this.

[4] A major drawback of the approach is that the number of vertices is exponential in the number of inputs p. We could design a cheaper abstraction but it would be coarser, in addition the number of inputs p often remains reasonable.

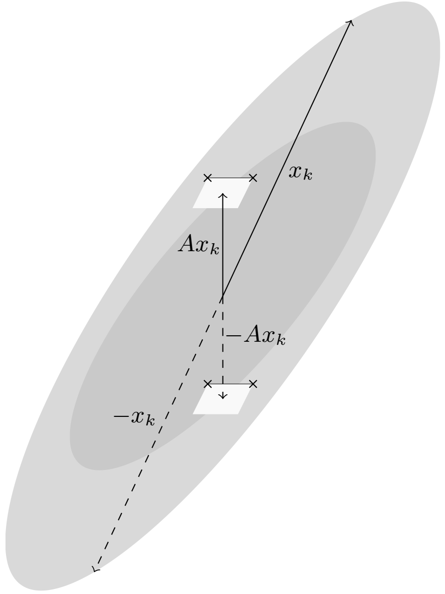


Figure 3: We can forget half of the vertices of the white box as they will be taken into account on the opposite side.

Using the S-procedure[5] we get the equivalent formulation

$$\begin{array}{l}\forall i \in [\![1, 2^p]\!], \exists \tau_i, \tau_i \geq 0 \wedge \\ \left(\begin{array}{cc} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & \lambda_{k+1} - e_i^T B^T PBe_i \end{array}\right) \\ \qquad -\tau_i \left(\begin{array}{cc} -P & 0 \\ 0 & \lambda_k \end{array}\right) \succeq 0 \end{array}$$

which is an LMI in λ_{k+1} and the τ_i which is solved by minimizing λ_{k+1}.

We can notice that, by a symmetry argument, we can forget about half of the e_i as depicted on Figure 3.

5.3 Iterating to Fixpoint and Widening

Now we can compute Kleene iterates but it will be slow to converge to a fixpoint. To accelerate, we can use a widening with thresholds, which allows us to find a value for λ up to some factor q of the least one by using a sequence of powers of q as thresholds.

5.4 An alternative to Classical Widening

When looking for a good postfixpoint we are indeed looking for a small λ satisfying the following equation

$$\begin{array}{l}\forall i \in [\![1, 2^p]\!], \tau_i > 0 \wedge \\ \left(\begin{array}{cc} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & \lambda - e_i^T B^T PBe_i \end{array}\right) \\ \qquad -\tau_i \left(\begin{array}{cc} -P & 0 \\ 0 & \lambda \end{array}\right) \succeq 0. \end{array} \tag{9}$$

This is not an LMI because of the τ_i but if we used the method described in Section 4.2 for choosing the shape of P, we have obtained[6] a parameter $r \in (0, 1)$ such that $\tau_i \in [r, 1]$. Computing the smallest λ satisfying the following LMI then directly gives a postfixpoint

$$\begin{array}{l}\forall i \in [\![1, 2^p]\!], \left(\begin{array}{cc} -A^T PA & -A^T PBe_i \\ -e_i^T B^T PA & \lambda - e_i^T B^T PBe_i \end{array}\right) \\ \qquad -\frac{r+1}{2} \left(\begin{array}{cc} -P & 0 \\ 0 & \lambda \end{array}\right) \succeq 0. \end{array}$$

5.5 Refining a Postfixpoint by Dichotomy

Once we have found a postfixpoint λ_{max} using widening with thresholds, we can refine it through decreasing iterations with narrowing but this usually does not lead quickly to anything close to the least fixpoint. However, an interesting property of this least fixpoint λ_{min} is that $\lambda_{k+1} \leq \lambda_k$ exactly when $\lambda_k \geq \lambda_{min}$, then enabling to efficiently and tightly overapproximate it by a dichotomy testing satisfiability of the LMI (9) for values of λ between zero[7] and λ_{max}.

6. BACK TO INTERVALS

While quadratic forms precisely over-approximate the set of reachable states of linear systems subject to bounded inputs, they are hardly usable as such in conjunction with other abstractions. We can solve LMIs to project the obtained ellipsoid and get bounds on the variables, $x_i \in [-a, a]$ with a the least value such that

$$\left(\begin{array}{cc} 0 & -\frac{e_i}{2} \\ -\frac{e_i^T}{2} & a \end{array}\right) - \tau \left(\begin{array}{cc} -P & 0 \\ 0 & \lambda \end{array}\right) \succeq 0$$

where e_i is the ith vector of the canonical base.

This is not limited to intervals, the same thing can be done for octagons [16] or more generally linear [23] or even quadratic [1, 10] templates.

7. FLOATING POINT ISSUES

Two fundamentally different issues with floating point numbers must be considered

the analyzed system contains floating point computations with rounding errors making it behave differently from the way it would if the same computations were done with real numbers, this is discussed in Section 7.1;

the implementation of the abstract domain is also carried out with floating point computations for the sake of efficiency, this usually works well in practice but can give erroneous results, hence the need for some a posteriori validation, see Section 7.2 for further details.

7.1 Taking Rounding Errors Into Account

The sum of two floating point values is, in all generality, not representable as a floating point value and must consequently be rounded. The accumulation of rounding errors can potentially lead to far different results from the ones expected with real numbers, thus floating point computations must be taken into account in our analysis [19].

The rounding errors can be of two different types :

[5] See Theorem 1.

[6] Otherwise we can still recompute such a parameter r.

[7] Or, better, the last prefixpoint encountered during the widening iterations.

- for normalized numbers represented with a fixed number of bits, we get a relative error: $\text{round}(a+b) \in [(1-\epsilon)(a+b), (1+\epsilon)(a+b)]$;
- for denormalized numbers (i.e ones very close to 0), we get an absolute error: $\text{round}(a+b) \in [a+b-\omega, a+b+\omega]$.

A common and easy solution to take both possible errors into account is to sum them which in practice leads only to a very slight overapproximation: $\text{round}(a+b) \in [(1-\epsilon)(a+b)-\omega, (1+\epsilon)(a+b)+\omega]$. Although only addition is illustrated here, the method works exactly the same way for any other floating point operation. The actual values of ϵ and ω depend on the characteristics of the considered floating point system. For instance we will take $\epsilon = 2^{-23}$ and $\omega = 2^{-149}$ for single precision[8].

Combining these elementary errors we get a simple postprocessing for each iteration of Section 5.2 to soundly overapproximate rounding errors.

Definition 1. fl(e) represents floating point evaluation of expression e with any rounding mode and any order of evaluation[9] [22].

LEMMA 1. *Assuming $\epsilon \leq 2^{-23}$, $\omega \leq 2^{-149}$ and $n \leq 2^{11} = 2048$, we have*[10]

$$\text{fl}\left(\sum_{i=1}^{n} a_i x_i\right) \leq (1+(n+1)\epsilon)\left(\sum_{i=1}^{n} a_i x_i\right) + n(n+1)\omega.$$

PROOF. By induction on n. □

LEMMA 2. *For any $a \in \mathbb{R}$, $x, y \in \mathbb{R}^n$, $P \in \mathbb{R}^{n\times n}$ a symmetric positive definite matrix and $\lambda, \mu \in \mathbb{R}$, if we have $||y||_2 \leq \mu$ and $x^T P\, x \leq \lambda$ then*

$$(ax+y)^T P\,(ax+y) \leq a^2\lambda + 2ab\sqrt{\lambda} + b^2$$

with $b = \sqrt{r}$ with r the least scalar such that there exists $\tau \in \mathbb{R}$ satisfying

$$\tau \geq 0 \wedge \begin{pmatrix} -P & 0 \\ 0 & r \end{pmatrix} - \tau \begin{pmatrix} -I & 0 \\ 0 & \mu \end{pmatrix} \succeq 0.$$

PROOF. Using Theorem 1 we have $y^T P\, y \leq r$ hence the result by expansion and Cauchy-Schwartz inequality. □

THEOREM 2. *For any $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$, $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $P \in \mathbb{R}^{n\times n}$ a symmetric positive definite matrix and $\lambda \in \mathbb{R}$ if $n+p \leq 2048$ then*

$$\begin{array}{l}(Ax+Bu)^T P\,(Ax+Bu) \leq \lambda \Rightarrow \\ (\text{fl}(Ax+Bu))^T P\,\text{fl}(Ax+Bu) \leq a^2\lambda + 2ab\sqrt{\lambda} + b^2\end{array}$$

with $a = 1+(n+p+1)\epsilon$ and b defined as in Lemma 2 for $\mu = \sqrt{n}(n+p)(n+p+1)\omega$.

PROOF. By successive applications of Lemmas 1 and 2. □

Thus, provided that the number of variables of the system plus its number of inputs is less than 2048, which is a reasonable assumption, we just have to compute a and b once as defined in Theorem 2 then apply to each step of Section 5.2 the postprocessing $\lambda \mapsto a^2\lambda + 2ab\sqrt{\lambda} + b^2$ to take into account computation with floats, whatever the rounding mode and the order of evaluation.

Such use of abstract domains in the real field to soundly analyze floating point computations is not new [17] and some techniques even allow to finely track rounding errors and their origin in the analyzed program [12].

7.2 Checking Soundness of the Result

Because the LMI solver is implemented with floating point computations, we have no guarantee on the results it provides[11]. Hence the need to check them. This amounts to checking that a given matrix is actually positive definite.

This is done by carefully bounding the rounding error on a floating point Cholesky decomposition [22][12]. Proof of positive definiteness of an $n \times n$ matrix can then be achieved in time $O(n^3)$ which in practice induces only a very small overhead to the whole analysis.

8. EXPERIMENTAL RESULTS

All the elements presented in this paper have been implemented as an autonomous linear system analysis engine. The tool is composed of three parts:

- The core mathematical computations are done with Scilab [24], mainly with the LMI solver [20] from an OCaml front-end. This part is a set of functions that implement the algorithms presented in Sections 4, 5, 6 and 7.1, as well as projections of ellipsoids over intervals. Computation in Scilab are done using double precision floats.
- The front-end is an OCaml code using rational numbers (Num library). It loads the A and B matrices and interacts with Scilab to compute the different sequence of calls to Scilab functions.
- A last part, also in OCaml, interfaces the obtained quadratic form with a particular C implementation of a Cholesky decomposition [22] to ensure its stability as explained in Section 7.2.

The code is released under a GPLv2 license and is available at `http://cavale.enseeiht.fr/`.

Experiments were conducted on a set of stable linear systems. These systems were extracted from [9], [1] or from

[8] Type `float` in C.

[9] Order of evaluation matters since floating point addition is not associative.

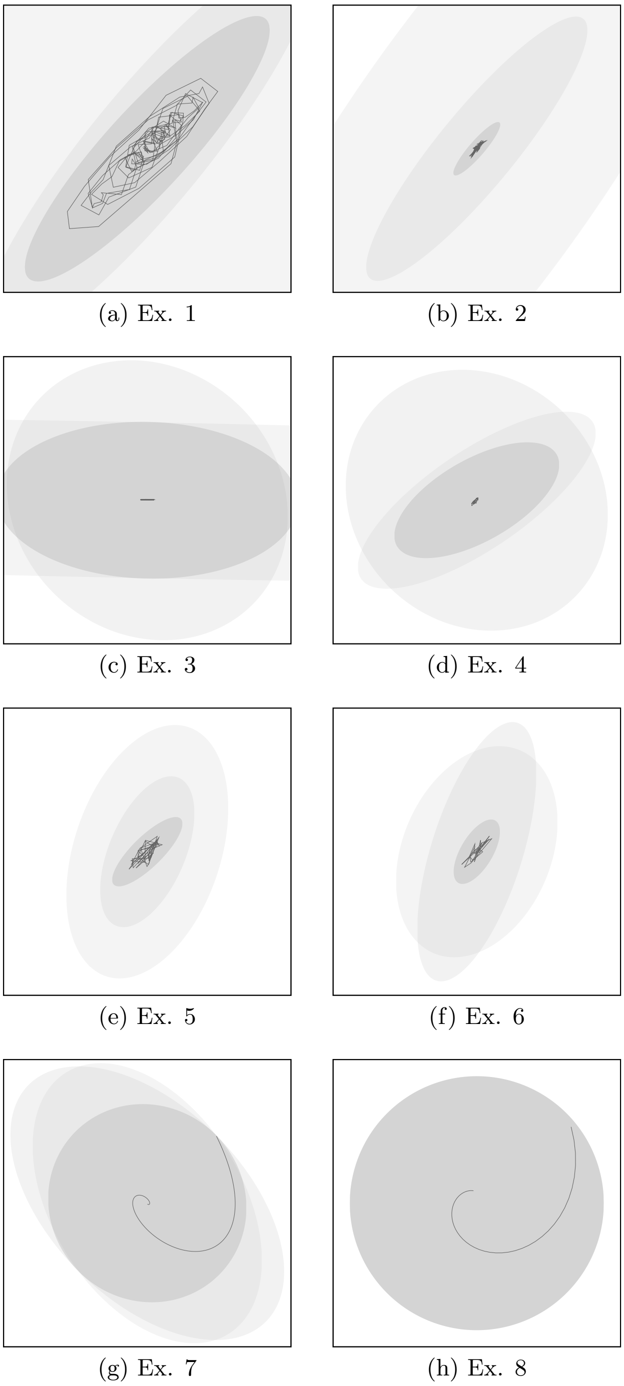
[10] Those are the values for single precision but it would work just the same for any other precision with adequate values .

[11] There also exists guaranteed SDP solvers now [14] over and underapproximating the primal and the dual problem to guarantee an error bound on the result. However we only need to check the final result. Thanks to Éric GOUBAULT for pointing that to us.

[12] Thanks to Timothy WANG for pointing this to us.

	Method	t_1		λ	$\lambda_\triangledown$	Bounds	t_2	Valid. t_3
Ex. 1 From [9, slides] n=2, 1 input	I	0.07	fp	131072	105341	[140.4; 189.9]	0.48	0.01
			τ	105351			0.40	0.01
	P	0.16	fp	128.0	96.0	[22.2; 26.5]	0.35	0.01
			τ	96.8			0.28	0.01
	U	0.23		$1+\epsilon$		[16.2; 17.6]	0.20	0.01
Ex. 2 From [9, slides] n=4, 1 input	I	0.09	fp	2048	1371	[18.1; 25.2; 24.3; 33.7]	0.48	0.02
			τ	1376			0.40	0.01
	P	0.27	fp	8.0	4.2	[6.3; 7.7; 2.2; 3.4]	0.35	0.01
			τ	6.4			0.27	0.02
	U	0.40		$1+\epsilon$		[1.7; 2.0; 2.2; 2.5]	0.21	0.01
Ex. 3 Discretized lead-lag controller n=2, 1 input	I	0.07	fp	262144	204241	[391.4; 21.6]	0.54	0.01
			τ	$\perp$			$\perp$	$\perp$
	P	0.17	fp	2048	1281	[36.2; 36.1]	0.44	0.02
			τ	1632			0.33	0.01
	U	0.20		$1+\epsilon$		[38.8; 20.3]	0.20	0.01
Ex. 4 Linear quadratic gaussian regulator n=3, 1 input	I	0.09	fp	16.0	10.3	[1.2; 0.9; 0.5]	0.38	0.02
			τ	10.9			0.32	0.02
	P	0.19	fp	1.0	0.7	[0.9; 0.9; 0.9]	0.31	0.02
			τ	1.1			0.26	0.01
	U	0.24		$1+\epsilon$		[0.7; 0.4; 0.3]	0.22	0.02
Ex. 5 Observer based controller for a coupled mass system n=4, 2 inputs	I	0.09	fp	512.0	304.6	[9.8; 8.9; 11.0; 16.8]	0.48	0.03
			τ	323.0			0.43	0.03
	P	0.24	fp	32.0	24.3	[5.7; 5.6; 6.4; 10.1]	0.42	0.03
			τ	28.6			0.33	0.03
	U	0.48		$1+\epsilon$		[5.0; 4.9; 4.8; 4.7]	0.22	0.03
Ex. 6 Butterworth low-pass filter n=5, 1 input	I	0.10	fp	128.0	102.4	[7.5; 8.7; 6.1; 7.0; 6.5]	0.44	0.02
			τ	113.1			0.38	0.03
	P	0.32	fp	8.0	7.1	[3.6; 5.0; 4.7; 8.1; 8.9]	0.37	0.02
			τ	7.7			0.29	0.02
	U	0.78		$1+\epsilon$		[2.3; 1.1; 1.9; 2.0; 2.9]	0.24	0.03
Ex. 7 Dampened oscillator from [1] n=2, no input	I	0.07	fp	353.6	353.6	[1.7; 2.1]	0.22	0.01
			τ	353.6			0.23	0.01
	P	0.15	fp	3.0	3.0	[2.0; 2.0]	0.22	0.01 ($\perp$)
			τ	3.0			0.20	0.01 ($\perp$)
	U	0.27		$1+\epsilon$		[1.5; 1.5]	0.16	0.01
Ex. 8 Harmonic oscillator from [1] n=2, no input	I	0.08	fp	22.9	22.9	[1.5; 1.5]	0.22	0.01
			τ	22.9			0.23	0.01
	P	0.24	fp	2.0	2.0	[1.5; 1.5]	0.24	0.01 ($\perp$)
			τ	2.0			0.20	0.01 ($\perp$)
	U	0.15		$1+\epsilon$		[1.5; 1.5]	0.16	0.01

Table 1: Result of the experiments: quadratic invariants computation. Times are expressed in seconds, t_1 is the time spent to compute the shape of the ellipsoid, t_2 is the time spend to find the appropriate ratio λ and project the resulting invariant on intervals and t_3 is the time needed to validate the stability of the resulting ellipsoid, as explained in Section 7.2. I, P and U are respectively the methods of Sections 4.1, 4.2 and 4.3. $\lambda_\triangledown$ denotes the refined value of λ by dichotomy.



(a) Ex. 1

(b) Ex. 2

(c) Ex. 3

(d) Ex. 4

(e) Ex. 5

(f) Ex. 6

(g) Ex. 7

(h) Ex. 8

Figure 4: Comparison of obtained ellipsoids by methods of Sections 4.1, 4.2 and 4.3 from lighter to darker, plus a random simulation trace ((b), (d), (e) and (f), being of dimension greater than 2, are cuts along planes containing the origin and two vectors of the canonical base, to show how the three different templates compare together).

basic controllers found in the literature. Table 1 illustrates the value computed using the different techniques as well as the time spent at each step. Figure 4 compares some plots of the obtained quadratic forms depending on the approach used to find the ellipsoid.

9. RELATED WORK

Many work in abstract interpretation, and its use to analyze programs, focus on linear patterns to abstract properties. However few work address non linear invariant synthesis.

Feret's work [8, 9] on the one hand is a practical approach to the problem. Its goal is to address the need by Astrée to handle the linear filters present in Airbus' real time software. Previous work [18] by Monniaux also addressed the same class of systems at the same level as in this paper rather than on actual code. As mentioned earlier, this effort addresses a strict subset of the systems we consider. However, it is hard to compare both works in terms of precision on the set of systems they both handle due to the lack of publicly available implementation or figures. Although Feret's work is probably a bit more precise thanks to the way it takes into account a limited number of previous inputs, performing a kind of unrolling, we use better ellipsoids of higher dimension and Figure 4 indicates that the resulting precision is often far from disastrous.

On the other hand, there are work that target similar properties but are more theory oriented and motivated. One can cite the Lagrangian Relaxation approach applied to program termination analysis as introduced by Cousot in [4] and Roozbehani, Féron and Megretski in [21], or the works of Adjé, Gaubert and Goubault [1] and Gawlitza and Seidl [10] on policy iterations and non linear forms. The latter two aim at replacing a Kleene based fixpoint computation by a symbolic reasoning based on semi-definite programming. They are more inspired by theoretical results leading to the analysis. [1, 2] even cites the existence of Lyapunov based invariant as a prerequisite for the method. *These works are more general than ours*: they address the analysis of non linear systems, even with non convex properties. However *none of them automatically finds the appropriate shape*: templates need to be given, e.g by providing a Lyapunov function, whereas we automatically compute an, in some sense "optimal", template[13]. *They also do not address the floating point issues.*

Our work should be considered as an in-between solution. It takes ideas from control theory results but targets the analysis of specific realistic systems. Furthermore it addresses floating point errors as well as the validity analysis of the obtained invariants.

10. CONCLUSION AND PERSPECTIVES

We have presented a set of analysis allowing us to characterize quadratic invariants, i.e. ellipsoids, for a subset of linear systems: inherently stable linear systems subject to bounded inputs.

[13] It can be interesting to notice that in case of Ex. 7 of Table 1, such an automatically computed template allows to find more precise bounds than in [1, 10] with a manually chosen template.

Most of the critical embedded control command systems rely on such linear systems. But intervals and linear invariants in general will not allow to precisely describe their state space.

This analysis is based on ideas from control theory. They are used to prove the stability of the system by exhibiting a proof of existence of a so-called Lyapunov quadratic form.

This work addresses the explicit computation of such a form by exploring the instantiation of multiple generic templates to find the most appropriate ellipsoids to bound the analyzed system.

Our effort also considers floating point errors and addresses the validity of the computed solution. It has been implemented and applied on several examples. The reduced product between the different templates instantiated gives extremely precise results as illustrated by the experimentations.

The approach of this paper presents the major drawback of being unable to directly analyze actual systems at code level, in particular because such systems are usually equipped with saturations or resets. We believe this can be addressed by using policy iteration methods [1, 10]. The computation of templates does not play any role in soundness of the analysis and may be able to accommodate heuristics extracting potential A and B matrices from the code[14]. Methods used to address floating point issues should also be adaptable to policy iteration.

Acknowledgments.
We deeply thanks Éric Goubault and Jérôme Feret for useful comments on this paper.

11. REFERENCES

[1] A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. In *ESOP*, volume 6012 of *LNCS*. Springer, 2010.

[2] F. Alegre, E. Féron, and S. Pande. Using ellipsoidal domains to analyze control systems software. 2009. http://arxiv.org/abs/0909.1977.

[3] S. Boyd, L. El Ghaoui, E. Féron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, June 1994.

[4] P. Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *VMCAI*, volume 3385 of *Lecture Notes in Computer Science*. Springer, 2005.

[5] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, 1977.

[6] P. Cousot and R. Cousot. Systematic design of program analysis frameworks. In *POPL*, 1979.

[7] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. Combination of abstractions in the Astrée static analyzer. In *ASIAN*, Tokyo, Japan, LNCS 4435, 2006. Springer.

[8] J. Feret. Static analysis of digital filters. In *ESOP*, number 2986 in LNCS. Springer, 2004.

[9] J. Feret. Numerical abstract domains for digital filters. In *International workshop on Numerical and Symbolic Abstract Domains (NSAD)*, 2005.

[10] T. M. Gawlitza and H. Seidl. Computing relaxed abstract semantics w.r.t. quadratic zones precisely. In *SAS*, volume 6337 of *LNCS*. Springer, 2010.

[11] K. Ghorbal, E. Goubault, and S. Putot. The zonotope abstract domain taylor1+. In *CAV*, volume 5643 of *LNCS*. Springer, 2009.

[12] E. Goubault and S. Putot. Static analysis of finite precision computations. In *VMCAI*, volume 6538 of *LNCS*. Springer, 2011.

[13] W. M. Haddad and V. S. Chellaboina. *Nonlinear Dynamical Systems and Control: A Lyapunov-Based Approach*. Princeton University Press, 2008.

[14] C. Jansson, D. Chaykin, and C. Keil. Rigorous error bounds for the optimal value in semidefinite programming. *SIAM J. Numerical Analysis*, 46(1), 2007.

[15] U. T. Jönsson. A lecture on the S-Procedure, 2001.

[16] A. Miné. The octagon abstract domain. In *AST 2001 in WCRE 2001*, IEEE. IEEE CS Press, October 2001.

[17] A. Miné. Relational abstract domains for the detection of floating-point run-time errors. In *ESOP*, volume 2986 of *LNCS*. Springer, 2004.

[18] D. Monniaux. Compositional analysis of floating-point linear numerical filters. In *CAV*, volume 3576 of *LNCS*. Springer, 2005.

[19] D. Monniaux. The pitfalls of verifying floating-point computations. *ACM Trans. Program. Lang. Syst.*, 30(3), 2008.

[20] R. Nikoukhah, F. Delebecque, and L. El Ghaoui. LMITOOL: a Package for LMI Optimization in Scilab User's Guide. Research Report RT-0170, INRIA, Feb. 1995.

[21] M. Roozbehani, E. Féron, and A. Megretski. Modeling, optimization and computation for software verification. In *HSCC*, volume 3414 of *LNCS*. Springer, 2005.

[22] S. M. Rump. Verification of positive definiteness. *BIT Numerical Mathematics*, 46, 2006.

[23] S. Sankaranarayanan, M. Colón, H. B. Sipma, and Z. Manna. Efficient strongly relational polyhedral analysis. In *VMCAI*, volume 3855 of *LNCS*. Springer, 2006.

[24] Scilab Team. Scilab. http://www.scilab.org.

[25] Q. Yang. *Minimum Decay Rate of a Family of Dynamical Systems*. PhD thesis, Stanford, 1992.

[14] The only limitation being to avoid generating too much spurious templates which would lead to a too costly analysis.