A scalable algebraic method to infer quadratic invariants of switched systems

Xavier Allamigeon
INRIA and CMAP, École polytechnique, CNRS
xavier.allamigeon@inria.fr

Stéphane Gaubert
INRIA and CMAP, École polytechnique, CNRS
stephane.gaubert@inria.fr

Eric Goubault
LIX, École polytechnique, CNRS
goubault@lix.polytechnique.fr

Sylvie Putot
LIX, École polytechnique, CNRS
putot@lix.polytechnique.fr

Nikolas Stott
INRIA and CMAP, École polytechnique, CNRS
nikolas.stott@inria.fr

ABSTRACT

We present a new numerical abstract domain based on ellipsoids designed for the formal verification of switched linear systems. Unlike the existing approaches, this domain does not rely on a user-given template. We overcome the difficulty that ellipsoids do not have a lattice structure by exhibiting a canonical operator over-approximating the union. This operator is the only one which permits to perform analyses that are invariant with respect to a linear transformation of state variables. Moreover, we show that this operator can be computed efficiently using basic algebraic operations on positive semidefinite matrices. We finally develop a fast non-linear power-type algorithm, which allows one to determine sound quadratic invariants on switched systems in a tractable way, by solving fixed point problems over the space of ellipsoids. We test our approach on several benchmarks, and compare it with the standard techniques based on linear matrix inequalities, showing an important speedup on typical instances.

Categories and Subject Descriptors

D.2.4 [**Software/Program Verification**]: Formal methods; F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Invariants

General Terms

Verification, Algorithms

Keywords

Static analysis, abstract interpretation, invariant generation, stability, hybrid and switched linear systems, matrix information geometry

.

1. INTRODUCTION

Motivation.

Program verification has long relied on affine methods. Analyzers based on abstract interpretation [11] have mostly been using polyhedric or subpolyhedric domains, as in e.g. [21, 27]. Ellipsoidal (or quadratic) invariants have led to more accurate analyses for some classes of programs. They have been used in linear control applications [25] and in program verification of linear recursive filters [14], they are also used locally in the static analyzer Astrée [12]. More general applications of ellipsoids in program validation can be found in [10, 2]. The latter reference develops a template approach, based on the linear template original idea of [36]. In template based methods, the shape of the ellipsoid has to be decided in advance by the user. Still, this is an approach which is adapted to control codes: we may use as (quadratic) templates the (quadratic) Lyapunov functions that the control theorist would have introduced to prove the stability of the underlying algorithms, as put forward in [15, 16].

Recently, some methods have been proposed in program validation for synthesizing invariant ellipsoids for linear systems e.g. [34, 35, 33], using linear matrix inequalities (LMI) techniques of control theory suitably adapted to program analysis.

Contribution.

In this paper, we are motivated by the analysis of more general systems and programs, including the important class of switched linear systems. Our main objective is develop scalable methods adapted to large scale instances. Also, we wish to avoid requiring extra information from the user, as in template based methods.

To this end, we develop a numerical abstract domain based on ellipsoids. The novelty of this domain does not consist in the use of ellipsoids as abstractions, but rather in the fact that we overcome two key difficulties which so far have limited the use of ellipsoids in abstract interpretation. The first issue is that the ordered set of ellipsoids does not constitute a lattice. This implies that there is a priori no canonical choice of the abstraction of the union of two sets, making the analysis less predictable as it relies on the selection of good upper bounds. The second issue is that most recent works using on ellipsoids rely on LMI methods. The latter

are efficient on moderate size examples but they are inherently limited by the complexity of interior point algorithms, which, in the case of matrix inequality problems, do not scale as well as for linear programming or second order cone programming problems.

Our main contributions are the following.

First, we reduce the question of abstracting by an ellipsoid the union of two sets to the selection of a minimal upper bound of two positive semidefinite (PSD) matrices with respect to the Löwner order (recall that the Löwner order is the canonical order on PSD matrices, corresponding to the inclusion of ellipsoids). We show in Theorem 4 that there is a unique selection procedure which has the property of being invariant with respect to linear transformations of the program variables. This invariance property is essential to guarantee that the analysis is robust. We call *invariant join* the minimal upper bound which is selected in this way. We show that the invariant join can be computed with the same cost as performing a Cholesky decomposition, i.e., in $O(n^3)$ arithmetic operations. We also show that the invariant join of two ellipsoids coincides with the minimal volume ellipsoid enclosing these two ellipsoids, so that, surprisingly, two distinct natural approaches lead to the same choice of selection. Then, we show that the invariant join operation can be used as a building block, to construct in a systematic way an abstract functional associated with a program.

Our second main contribution is to show that an an invariant ellipsoid can be computed by a scalable algorithm. This is based on a non-linear generalization of the power algorithm which is classically used to compute the dominant eigenvalue of a matrix. Indeed, we replace the fixed point problem of abstract interpretation by a nonlinear eigenvalue problem. The eigenvalue represents a stability margin and allows us to absorb errors due to finite precision computations. Then, the classical Kleene algorithm of abstract analysis is replaced by the power iteration. We present two variants of this iteration, based on additive and multiplicative perturbations ideas, respectively. We show that the multiplicative iteration does converge by exploiting metric geometry techniques: there is a known metric on the space of positive definite matrices, called Thompson's part metric, for which geodesic triangles satisfy a nonpositive curvature condition. This entails that for suitably chosen perturbation parameters, the multiplicative power iteration is contracting. This leads to a fast algorithm as performing one power iteration is not harder than evaluating the abstract interpretation functional.

We finally illustrate our approach by applying it to examples of switched linear systems. We show that the power iteration leads to important speedups by comparison with LMI based methods, at the price of a limited loss of precision. Indeed, for many typical examples, the ellipsoids computed by both methods yield stability certificates of comparable precision.

Discussion of the results and related work.

An obvious limitation of the present approach is that we only considered *centered* ellipsoids. However, our results apply to the class of linear switched systems, which is already considered to be a challenging and significant one in control applications. Whereas the present ideas may be useful as well for non centered ellipsoids, a systematic study of the non centered cased is left for a further work.

For general hybrid systems, or switched linear systems as in this article, there are several classical approaches from control theory [26], such as piecewise (linear or quadratic) Lyapunov function [22, 1], common (quadratic) Lyapunov functions [31], or multiple Lyapunov functions [8, 4] when the switching may depend on the current state. The question of checking, in an optimal way, the stability of switched linear systems is a well known hard problem, which essentially boils down to computing the joint spectral radius of a set of matrices. The standard approach is to use the theory of Barabanov norms. Computing or approximating the Barabanov norm, for instance by using polyhedral norm approximation [23], or SOS methods for approximating it [30], can be a highly demanding computational task. This paper follows a different path, by achieving much coarser but scalable stability proofs.

The question of selecting minimal upper bound of matrices with respect to the Löwner ordered as appeared in information geometry and mathematical morphology, see specially [9, 3]. Whereas the minimal volume ellipsoid has been considered in this context, its characterization as the invariant join, as well as the fast algorithm to compute it, is new. Also, the idea of studying dynamical systems on the cone positive definite matrices by means of metric geometry techniques was used in [18] in the context of repeated games. Whereas some similarities exist between the fixed point functionals encountered in the present static analysis and the ones encountered in game theory, the former turn out to be less well behaved as they are no longer order preserving. We finally emphasize that the Thompson's part metric used here is a classical notion which has been widely studied [28]. To our knowledge, such metric geometry ideas were not applied previously to invariant synthesis for programs and systems.

Organization of the paper.

In Section 2, we specify the class of programs and switched systems addressed in this paper. In Section 3, we present the domain of ellipsoids and give an abstract semantics to general programs, exploiting the equivalence between the poset of the ellipsoid and the poset of positive semidefinite matrices equipped with the Löwner order. In Section 4, we establish the main properties of the invariant join of two matrices. In Section 5, we describe the power iteration and establish convergence properties. Benchmarks on programs implementing switched linear systems or taken from the literature are presented in Section 6. Some technical proofs are omitted owing to the lack of space.

2. PROGRAMS AND SYSTEMS OF INTEREST

The programs we are considering in the sequel consist of a sequence of possibly nested while loops of the form:

```
while (rand_bool) {
  switch (rand_bool) {
    case 0:
      I0
    case 1:
      I1
    ...
    case k:
      Ik
```

```
  }
}
```

meaning that any `I0`, `I1` to `Ik` can be of a similar form, or an instruction. Instructions can be either, variable initialization (or declaration) or (parallel linear) assignments. Variables have a local scope, in the same way as in C or Java for instance. This will be semantically encoded by a variable deletion operator, in Section 3.2.3 (but has no specific syntax, as in C or Java). The expression `rand_bool` stands for a boolean with random value.

A variable declaration corresponds to the introduction of a new variable in the program. In addition, this variable is initialized according to two possible modes: (i) either with the constant 0, (ii) or with an arbitrary value within a symmetric interval $[-R, R]$. The latter possibility allows in particular to handle the declaration of a variable associated with a sensor measuring a physical value.

The programs may contain general assignments of the form `(x_1,...,x_n) <- P(x_1,...,x_n)`, where P is an $(n \times n)$-matrix. This operation corresponds to n parallel assignments of the variables $\mathtt{x}_i$ to linear terms $\sum_j P_{ij}\mathtt{x}_j$ respectively (note that the assignment $\mathtt{x}_i$ `<-` $\mathtt{x}_i$ encodes the fact that the variable $\mathtt{x}_i$ remains unchanged).

Programs expressed in this grammar encompass many classical situations we may find in embedded systems. For instance, linear control programs, with (uninterpreted) modes, that is a controller which might have several control modes. They can also represent (discrete) simulation schemes to compute the state of a switched linear system, modelling some complex physical phenomenon. Let us recall that a discrete-time switched linear system can be modeled as a finite collection of equations

$$x_{k+1} = A_i x_k,\ k \in \mathbb{N},\ i \in \mathcal{I} = \{1, ..., N\}$$

where $x \in \mathbb{R}^n$ is the state, $\{A_i\}_{i\in\mathcal{I}}$ are transition matrices and $\mathcal{I}$ enumerates the different modes. Our method will also allow for conservative proofs of stability of such systems, by constructing invariant ellipsoids which do not exploit the knowledge of the switching mechanism.

For example, we may consider a system switching between two damped harmonic oscillators:

$$\begin{cases} \ddot{x} + \omega_0 \dot{x} + \omega_0^2 x &= 0 \\ \ddot{x} + \omega_1 \dot{x} + \omega_1^2 x &= 0 \end{cases}. \tag{1}$$

Using $\omega_0 = 1$ and $\omega_1 = 0.8$, we can simulate this system with the program in Figure 1.

```
(* h, ω0 and ω1 are numerical constants      *
 * h = 0.01, ω0 = 1, ω1 = 0.8                *)
declare x = [-1,1];
declare v = [-1,1];
while (rand_bool) {
  switch (rand_bool) {
    case 0:
      (x, v) <- (x+h*v, -(hω0²)*x+(1-hω0)*v);
    case 1:
      (x, v) <- (x+h*v, -(hω1²)*x+(1-hω1)*v);
  }
}
```

Figure 1: Implementation of system (1) using an explicit Euler integration scheme

3. THE DOMAIN OF ELLIPSOIDS

3.1 Poset of ellipsoids and Löwner order

We begin by introducing some usual notation in linear algebra. We denote by I_n the identity matrix of size $n \times n$, or simply I when the dimension is clear from the context. The transpose of a matrix M is denoted by M^T. In particular, if $x = (x_i) \in \mathbb{R}^n$ is a column vector, the notation x^T simply stands for the associated row vector with entries x_i.

Recall that a matrix $M = (M_{ij}) \in \mathbb{R}^{n\times n}$ is said to be *symmetric* when $M = M^T$, i.e. $M_{ij} = M_{ji}$ for all i, j. We will denote by $\mathcal{S}_n$ the set of symmetric matrices of size $n \times n$. A matrix $U \in \mathbb{R}^{n\times n}$ is *orthogonal* if $UU^T = U^TU = I$. Every symmetric matrix A can be diagonalized by an orthogonal change of basis. This means that we can write A under the form U^TDU, where D is a diagonal matrix, and U is orthogonal. The group of invertible matrices of $\mathbb{R}^{n\times n}$ is denoted by $GL_n(\mathbb{R})$.

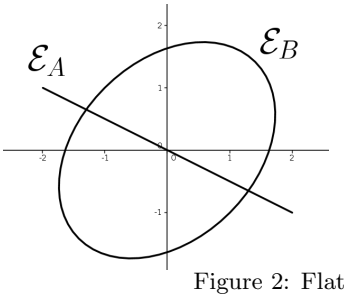
A symmetric matrix $A = (A_{ij}) \in \mathcal{S}_n$ is said to be *positive semidefinite* (shortened *PSD*) if for all $x \in \mathbb{R}^n$, we have $x^TAx = \sum_{ij} A_{ij}x_ix_j \geqslant 0$. This is equivalent to any of the two following properties: all the eigenvalues of A are non-negative; A can be written as $A = MM^T$ for some $M \in \mathbb{R}^{m\times n}$ where $m \leqslant n$ is the rank of M. We write $A \succcurlyeq 0$ to mean that A is PSD, and we denote by $\mathcal{S}_n^+$ the set of PSD matrices. The relation $\succcurlyeq$ extends to any pair of matrices $A, B \in \mathcal{S}_n$ by writing $A \succcurlyeq B$ when $A - B \succcurlyeq 0$. This provides a partial ordering over $\mathcal{S}_n$, referred to as the *Löwner order*. When $x^TAx > 0$ holds for all non-zero $x \in \mathbb{R}^n$, the matrix A is said to be *positive definite*. A matrix is positive definite if and only if it is PSD and invertible. In this case, its inverse is also positive definite. We shall write $A \succ 0$ to mean that A is positive definite.

We now review some properties on PSD matrices which will be useful in the sequel.

First, PSD matrices have square roots, just as non-negative numbers. More precisely, if A is a PSD matrix, there exists a unique PSD matrix $A^{1/2}$ such that $(A^{1/2})^2 = A$. Writing A under the form U^TDU as above, then $A^{1/2}$ is given by the matrix $U^T\sqrt{D}U$, where $\sqrt{D}$ stands for the diagonal matrix with diagonal entries $\sqrt{D_{ii}}$. More generally, for all positive scalars s, we define A^s to be the matrix U^TD^sU, where D^s is the diagonal matrix obtained by raising to the power s every diagonal entry of D. We shall refer to A^s as the *s-th power* of A, as it coincides with the usual s-th power for integer values of s.

Moreover, given $A, B \in \mathcal{S}_n^+$, the combination $\lambda A + \mu B$ also lies in $\mathcal{S}_n^+$ for all $\lambda, \mu \in \mathbb{R}_+$. In other words, the set $\mathcal{S}_+^n$ of PSD matrices forms a *convex cone*. It is also a *pointed* cone, meaning that there is no $A \in \mathcal{S}_+^n$ such that $A \succcurlyeq 0$ and $-A \succcurlyeq 0$, unless $A = 0$. Any convex and pointed cone C induces a partial ordering defined by "$x \geqslant y$" if $x - y \in C$. The most famous cone is certainly the orthant $\mathbb{R}_+^n$ of $\mathbb{R}^n$; the associated partial order gives rise to systems of linear inequalities and to linear programming. Other cones play a major role in optimization, such as the Lorentz cone used in quadratic programming, and the PSD cone $\mathcal{S}_n^+$ in semidefinite programming.

We point out that the cone $\mathcal{S}_n^+$ does not yield a lattice structure on $\mathcal{S}_n$. Indeed, $\mathcal{S}_n^+$ is not a polyhedral cone: it has infinitely many extreme rays, generated by the matrices of rank 1. In contrast, cones providing a lattice structure



$$A = \begin{pmatrix} 4 & -2 \\ -2 & 1 \end{pmatrix}$$
$$B = \begin{pmatrix} 3 & 1 \\ 1 & 3 \end{pmatrix}$$

Figure 2: Flat ellipsoids

are characterized by the following result of Krein and Rutman [24]: a finite dimensional cone yields a lattice order if and only if it is simplicial, i.e. it is generated by finitely many linearly independent extreme rays.

We now introduce ellipsoids and relate them with PSD matrices. Recall that, in this paper, we restrict our attention to *centered* ellipsoids.

Formally, a *(centered) ellipsoid* is defined as the image of the unit ball $\mathcal{B}(0,1) := \{x \in \mathbb{R}^n \mid x^T x \leqslant 1\}$ under a linear map $x \mapsto Mx$, where $M \in \mathbb{R}^{n \times n}$. When the matrix M is invertible, this ellipsoid is given by the set

$$\{x \in \mathbb{R}^n \mid x^T A^{-1} x \leqslant 1\}, \tag{2}$$

where A is the matrix MM^T. Note in particular that A is positive definite. However, it will be convenient for our purposes to allow M to be singular. Equivalently the matrix $A = MM^T$ will be allowed to be only PSD (not necessarily definite). In this case, we can show that the former ellipsoid is characterized as the following set

$$\mathcal{E}_A := \{x \in \mathbb{R}^n \mid xx^T \preccurlyeq A\}. \tag{3}$$

Equivalently, $\mathcal{E}_A$ is the set of vectors $x \in \mathbb{R}^n$ satisfying $(y^T x)^2 \leqslant y^T A y$ for all $y \in \mathbb{R}^n$.

The benefit of relaxing the condition on the invertibility of A is that it allows to consider ellipsoids which are not necessarily full-dimensional, while this is not possible with the former representation (2) in which A is positive definite. Indeed, the ellipsoid $\mathcal{E}_A$ is contained in the range $\{Ay \mid y \in \mathbb{R}^n\}$ of the matrix A. Thus, when A is not invertible, the ellipsoids $\mathcal{E}_A$ are "flat", see Figure 2 for an example. Handling non-full dimensional ellipsoids will be useful in the setting of formal verification by abstract interpretation, since we also have to handle destructive updates, i.e. non invertible assignments such as an assignment of a variable to 0.

We are now ready to present (centered) ellipsoids in the framework of abstract interpretation. In essence, we use PSD matrices to over-approximate bounded subsets of $\mathbb{R}^n$ by ellipsoids. Following the terminology of abstract interpretation, our concrete domain is defined as $\wp^{\text{bounded}}(\mathbb{R}^n)$, the lattice of bounded subsets of $\mathbb{R}^n$ equipped with the subset partial order $\subseteq$. The abstract domain is the cone $\mathcal{S}_n^+$ of PSD matrices. The concretization operator γ, which maps an abstract element to a concrete one, is defined as the function $\gamma : \mathcal{S}_n^+ \to \wp^{\text{bounded}}(\mathbb{R}^n)$ which associates a PSD matrix A to the corresponding ellipsoid $\mathcal{E}_A$. In other words, every PSD matrix $A \in \mathcal{S}_n^+$ "abstracts" the subsets contained in $\mathcal{E}_A$.

We first check that the concretization operator is monotone:

Lemma 1. *The map γ is order-preserving.*

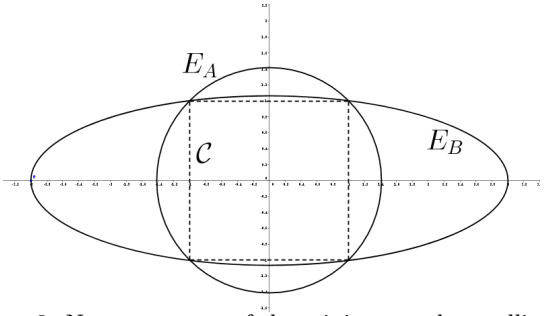


Figure 3: Non-monotony of the minimum volume ellipsoid

Proof. Let $0 \preccurlyeq A \preccurlyeq B$ and let $x \in \mathcal{E}_A$. By transitivity of $\preccurlyeq$, we get $xx^T \preccurlyeq A \preccurlyeq B$, thus $x \in \mathcal{E}_B$. □

Like several other abstract domains (convex polyhedra [13], zonotopes [19], etc.), the domain of ellipsoids cannot be equipped with an abstraction operator α. Indeed, this operator would be supposed to map any bounded subset S to the "smallest" PSD matrix A such that $S \subset \mathcal{E}_A$.[1] Such a matrix A does not exist in general since the cone $\mathcal{S}_n^+$ of PSD matrices does not constitute a lattice in the Löwner order.

An important result on ellipsoids, by John, states that given a bounded full-dimensional subset S of $\mathbb{R}^n$, there is a unique ellipsoid of minimal volume containing S, see [5, p. 19]. Note that this can be extended to any bounded subset S (not necessarily full-dimensional), by solving the minimization problem in a subspace of $\mathbb{R}^n$, and then mapping back to the initial space, which ultimately yields a flat ellipsoid. As illustrated in Figure 3, the function which maps S to the minimum volume ellipsoid containing S is not order preserving. Indeed, the minimum volume ellipsoid containing the square $\mathcal{C}$ is the disk $\mathcal{E}_A$, and the minimum volume ellipsoid containing an ellipsoid E_B is itself. However, we have here $\mathcal{C} \subseteq \mathcal{E}_B$ but $\mathcal{E}_A \not\subseteq \mathcal{E}_B$.

3.2 Abstract primitives

We present here our abstractions for the following operations: declaration of a variable, linear assignment, deletion of a variable, switch/case statement and the while loop. More precisely, each of these operations can be modeled by a function $[\![\texttt{op}]\!]$ mapping the concrete states to the new ones arising after the operation. Then our goal is to define a corresponding operator $[\![\texttt{op}]\!]^\sharp$ dealing with abstract states and which is *sound*, i.e. such that $[\![\texttt{op}]\!](\gamma(A)) \subset \gamma([\![\texttt{op}]\!]^\sharp(A))$ for all PSD matrix A. The latter condition ensures that the abstract primitive propagates over-approximations of the concrete states in a correct way. When the inclusion turns to be an equality, the abstract operator is said to be *exact*, which means that it does not introduce an additional approximation of the concrete states.

3.2.1 Variable declaration

In both cases, starting from a PSD matrix $A \in \mathcal{S}_n^+$, the abstraction of the introduction of a new variable corresponds to an augmentation of A in block form:

$$A \mapsto \left[\begin{array}{c|c} \alpha_1 A & \\ \hline & \alpha_2 \end{array}\right] \in \mathcal{S}_{n+1}^+ \tag{4}$$

for some well-chosen $\alpha_1, \alpha_2 \geqslant 0$. If the variable is set to 0, we take $\alpha_1 = 1$ and $\alpha_2 = 0$. In this case, the ellipsoid is

[1] In this case, (α, γ) forms a *Galois connection*.

unchanged, but is now flat with respect to the new variable. When the variable is initialized with the interval $[-R, R]$, we set

$$\alpha_1 := \begin{cases} 1 + 1/\operatorname{rk} A & \text{if } A \neq 0\,, \\ 1 & \text{otherwise,} \end{cases} \qquad \alpha_2 := (1 + \operatorname{rk} A)R\,, \quad (5)$$

where $\operatorname{rk} A$ stands for the rank of A. It can be proved that with this choice, the extended matrix given in (4) corresponds to the minimum volume ellipsoid containing the truncated cylinder $\mathcal{E}_A \times [-R, R]$. By construction, we deduce that our abstract primitive is sound.

Let us give an example from the first two variable declarations in Figure 1. Initially, since no variable is declared, the abstract state A is the (0×0)-matrix. After the declaration of `x`, this matrix is augmented to a (1×1)-matrix A', whose only entry is equal to $1 + \operatorname{rk} A = 1$. At this point, the abstraction is exact since this precisely corresponds to the set of concrete states, formed by the interval $[-1, 1]$. Next, for the declaration of `v`, Eq. (5) provides the parameters $\alpha_1 = \alpha_2 = 2$ (as $\operatorname{rk} A' = 1$). Hence, the new abstract element is the PSD matrix $A'' = \left(\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right)$. This encodes the disk of radius $\sqrt{2}$, which tightly over-approximates the set of concrete states, given by the square $[-1, 1] \times [-1, 1]$.

3.2.2 Linear assignment

We consider here linear assignment operations applied to vectors, i.e. of the form $(\mathtt{x}_1, \ldots, \mathtt{x}_n)$ `<-` $P(\mathtt{x}_1, \ldots, \mathtt{x}_n)$, where P is an $(n \times n)$-matrix. As an example, the first parallel assignment in Figure 1 corresponds to the matrix $P = \left(\begin{smallmatrix} 1 & h \\ -h\omega_0 & 1 - h\omega_0^2 \end{smallmatrix}\right)$. We define the corresponding abstract operator as the function $A \mapsto PAP^T$. The following lemma shows that this operator is sound and even exact:

LEMMA 2. *We have* $\{Px \mid x \in \mathcal{E}_A\} = \mathcal{E}_{PAP^T}$.

PROOF. Let M such that $A = MM^T$. We know that $\mathcal{E}_A$ and $\mathcal{E}_{PAP^T}$ respectively correspond to the image of the unit ball $\mathcal{B}(0, 1)$ under the linear maps $x \mapsto Mx$ and $x \mapsto PMx$. The expected result follows straightforwardly. □

3.2.3 Variable deletion

The deletion of a variable $\mathtt{x}_i$ consists in the projection of the state on the other remaining variables. This is also a linear transformation and its matrix P is the identity whose i-th line has been deleted. For instance, deleting `w` from the state $[\mathtt{x}, \mathtt{v}, \mathtt{w}]^T$ would yield the matrix $P = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right)$. As for the assignment operation, the abstract operator for deleting a variable is the function mapping $A \in \mathcal{S}_n^+$ to $PAP^T \in \mathcal{S}_{n-1}^+$. Again, this operator can be shown to be exact, using the same argument as in the proof of Lemma 2.

3.2.4 Switch statement

As described in Section 2, we are interested in switched systems (or programs) with an arbitrary switching rule. This means that our abstraction of a switch/case statement that tests the value of an expression does not guard the state space with respect to the outcome of this expression. Instead, we choose a coarser abstraction by considering that the expression has a random value. In the standard framework, this abstract operator would return the supremum of the abstract elements resulting from each branch of the *switch/case* statement. But as ellipsoids do not form a lattice, this supremum may not exist in general. Nevertheless, we would like to settle for a *minimal upper bound* of the results from each branch. We develop in Section 4 the selection of such a minimal upper bound of two positive semidefinite matrices, denoted by $\sqcup$. Note that this operator only takes two arguments and is not associative (as is the general case for non-lattice abstract domains [17]). If there are more than two branches $A_1, \ldots, A_n$, we simply join the branches sequentially by computing $\big((A_1 \sqcup A_2) \sqcup A_3\big) \cdots \sqcup A_n$.

3.2.5 While loop

Consider a while loop of the form `while (e < r) { s }` where `e` and `r` are general arithmetic expressions, and `s` is a sequence of instructions corresponding to the body of the loop. Like for the switch statement, our abstraction ignores the boolean expression `e < r` and replaces it with a random exit of the loop. We can do this without loss of generality under the assumption that the loop condition is eventually verified. If we were in the usual lattice based framework, the loop invariant could be determined as the least post-fixpoint of the function $X \mapsto A \sqcup s(X)$, where A corresponds to the abstract state at the entry of the loop, and s to the abstract operator of the body of the loop. However, in our setting, the join operator is not defined, and the least fixed point of the latter function may not exist. Instead, we define an abstract primitive which, given the abstract element A, returns an abstract element X satisfying $A \preccurlyeq X$ and $s(X) \preccurlyeq X$.

To this end, we first look for a positive definite matrix T satisfying $s(T) \preccurlyeq T$ (i.e. a post-fixpoint of the abstract operator s) using the power-like algorithm described in Section 5. The matrix T is then scaled in order to "contain" the abstract element A, i.e. we return the abstract element $X := \mu_T T$, where $\mu_T = \inf\{\mu \in \mathbb{R}_+ \mid \mu T \succcurlyeq A\}$. This approach yields a sound invariant, because essentially every operator s presented here is positively homogeneous: for all nonnegative λ and $Y \in \mathcal{S}_n^+$, we have $s(\lambda Y) = \lambda s(Y)$. (The only one that is not positively homogeneous is the bounded variable declaration, however, we assume this type of declaration do not appear in loops.) Consequently, we deduce that the abstract element X that we have computed also satisfies $s(X) \preccurlyeq X$.

As a consequence, the matrix T somehow serves as a template which is here computed in an automatic way. A similar scaling technique appeared in [34], in which the template is computed using semidefinite programming.

We point out that our method is open for a possible refinement, by scaling individually the semi-axes of the ellipsoid $\mathcal{E}_T$, instead of applying the same scaling factor μ.

4. SELECTION OF MINIMAL UPPER BOUNDS IN THE LÖWNER ORDER

A binary operation defined on a poset will be called a *selection of a minimal upper bound* or *quasi-join* if it sends every pair of elements of this poset to a minimal upper bound of this pair. The question of the selection of a minimal upper bound of positive definite matrices, with respect to the Löwner order, has arisen in several fields. In particular, the following selection has been considered in mathematical morphology [9]:

$$A \sqcup_m B = \frac{A + B}{2} + \frac{1}{2}\big((A - B)(A - B)^T\big)^{1/2}\,.$$

This is the analogue of the usual expression of the maximum of two scalars: $\max(a, b) = \frac{a+b}{2} + \frac{1}{2}\,|a - b|$. Based on this,

```
switch (rand_bool)          switch (rand_bool)
  case 0: x <- Px;            case 0: y <- UPU^{-1}y;
  case 1: x <- Qx;            case 1: y <- UQU^{-1}y;
```

Figure 4: Linear change of variables in a switch statement

we introduce an operation $(A, B) \mapsto A \sqcup B$ on the set of PSD matrices. For simplicity, we assume that one of the two matrices, say A, is invertible. Then,

$$A \sqcup B := X\Big(I_n \sqcup_m X^{-1}B(X^{-1})^T\Big)X^T\,, \tag{6}$$

where X denotes a square matrix (not necessarily PSD) such that $A = XX^T$. It can be shown that $A \sqcup B$ does not depend on the choice of X.

PROPOSITION 3. *The matrix $A \sqcup B$ is a minimal upper bound of A and B.*

PROOF. This is a consequence of the following two properties: (i) $I_n \sqcup_m X^{-1}BX^{-1^T}$ is a minimal upper bound of I_n and $X^{-1}B(X^{-1})^T$; (ii) The Löwner ordering is compatible w.r.t. any invertible linear transformation $L \in GL_n(\mathbb{R})$, meaning that

$$\forall X, Y \in \mathcal{S}_n^+\,, \quad X \preccurlyeq Y \iff LXL^T \preccurlyeq LYL^T\,. \quad \square \tag{7}$$

An essential property of the selection $(A, B) \mapsto A \sqcup B$, which we shall establish below, is that it is *invariant under invertible linear transformations*, meaning that

$$\forall U \in GL_n(\mathbb{R}), \big(UAU^T\big) \sqcup \big(UBU^T\big) = U\big(A \sqcup B\big)U^T\,. \tag{8}$$

This entails that the precision of the operator is not affected by a linear change of variables in the program. Let us illustrate the latter property on the programs given in Figure 4. The right-hand side program is obtained from the left-hand side one by applying the linear change of variables $U\mathtt{x} \to \mathtt{y}$. We denote by X_0 the initial abstract state in the analysis of the left-hand side program, i.e. before the execution of the switch statement. Accordingly, we assume that the abstract state of the right-hand side program is given by $Y_0 = UX_0U^T$. Following the definition of the abstract primitives in Section 3.2, the analysis of the two programs respectively provides the following final invariants:

$$\begin{aligned} X_f &= (PX_0P^T) \sqcup (QX_0Q^T) \\ Y_f &= (UPU^{-1}Y_0(U^{-1})^TP^TU^T) \sqcup (UQU^{-1}Y_0(U^{-1})^TQ^TU^T) \\ &= (UPX_0P^TU^T) \sqcup (UQX_0Q^TU^T)\,. \end{aligned}$$

Then it can be verified using (8) that the final invariant of the second program corresponds to a rewriting of the invariant of the first program, i.e. $Y_f = UX_fU^T$.

The following theorem justifies the term *invariant join* for the operator $\sqcup$.

THEOREM 4. *The binary operator $\sqcup$ defined in* (6) *is the only selection of a minimal upper bound, with respect to the Löwner order, that is invariant by invertible linear transformations.*

The proof of this result will appear elsewhere.

As a consequence of Theorem 4, we can show that $A \sqcup B$ corresponds to the minimum volume ellipsoid containing $\mathcal{E}_A$ and $\mathcal{E}_B$. Usually, this ellipsoid is computed as the inverse of the optimal solution C^* of the following semidefinite program:

$$\begin{array}{rl} \text{minimize} & -\log\det C \\ \text{subject to} & C \preccurlyeq A^{-1},\, C \preccurlyeq B^{-1}\,. \\ & C \succ 0 \end{array} \tag{9}$$

The fact that $(C^*)^{-1}$ coincides with $A \sqcup B$ results from the fact that this selection is invariant by linear transformation. The minimum volume ellipsoid enclosing $\mathcal{E}_A \cup \mathcal{E}_B$ is usually determined by solving an LMI problem. As a surprising by-product of Theorem 4, we obtain a new way to compute this minimum volume ellipsoid by using basic algebraic operations.

We point out in (6), it is convenient to use the Cholesky decomposition of A into XX^T where X is a lower triangular matrix, so that the inverse X^{-1} in (6) can be easily computed in $O(n^2)$ arithmetic operations by back-substitution. We conclude this section with the characterization of the time complexity of the invariant join operator.

PROPOSITION 5. *The invariant joint $A \sqcup B$ can be computed in $O(n^3)$ arithmetic operations.*

Theorem 4 shows that selecting the minimal volume ellipsoid yields a canonical abstraction of the union of two ellipsoids. However, since the invariant join $\sqcup$ is not a monotone map, using this selection locally, in an analysis, may not necessarily lead to the tightest global invariant.

5. A SCALABLE NONLINEAR FIXPOINT ALGORITHM

5.1 Additive and multiplicative power iterations

In this section, we present two scalable algorithms which will allow us to find an ellipsoid invariant. As explained in Section 3.2.5, if s denotes the abstract operator of the body of the loop, this boils down to finding a non-zero positive semidefinite matrix X such that $s(X) \preccurlyeq X$. To this end, we shall consider an auxiliary nonlinear spectral problem, which consists in finding a non-zero matrix $X \in \mathcal{S}_n^+$ and a scalar $\lambda > 0$ such that

$$s(X) = \lambda X\ . \tag{10}$$

If we find a matrix X for which $\lambda \leqslant 1$, then the original problem $s(X) \preccurlyeq X$ is solved. An interest of introducing the extra degree of freedom λ is to allow for finite precision computations. If $s(X) = \lambda X$ holds for $\lambda < 1$ and X positive definite, then, the relation $s(X) \preccurlyeq X$ remains valid under a small perturbation of X.

A simple idea to solve (10) is to choose an order preserving linear form $\psi : \mathcal{S}_n^+ \to \mathbb{R}_+$, and to define the following fixed point scheme

$$X_{k+1} = \frac{s(X_k)}{\psi(s(X_k))} \tag{11}$$

initialized with a positive definite X_0. A convenient choice of ψ is the trace functional. The latter has the property that it does not vanish on $\mathcal{S}_n^+$ except at the zero matrix. So, a division by zero will not occur in (11), unless $s(X_k)$ vanishes at some iteration, which will not be the case for the abstract operators considered here. By construction, $\psi(X_{k+1}) = 1$

holds for all k. Moreover, the set of positive semidefinite matrices X of trace one is bounded. Therefore, an additional advantage of the trace functional is that the sequence X_k remains bounded. If X_k converges to a matrix X, we get $X = \frac{s(X)}{\psi(s(X))}$ and so, $s(X) = \lambda X$ with $\lambda = \psi(s(X))$, which solves problem (10).

The algorithm (11) is a non-linear analogue of the power algorithm which is familiar in matrix theory [20]. The latter allows one to compute an eigenvector associated to a dominant eigenvalue (eigenvalue of maximal modulus) of a real matrix M by computing the sequence

$$x_{k+1} = \frac{Mx_k}{\|Mx_k\|_2} \tag{12}$$

where x_0 is a non-zero vector. This is similar to (11), except that we replaced the Euclidean norm $\|\cdot\|_2$ by the linear functional ψ. The well known advantage of the power algorithm is its scalability. To implement it, the matrix M need not be explicitly stored, it suffices to have an oracle which takes x as input and return Mx, hence, it is adapted to instances of large dimension (e.g., the "pagerank" algorithm is a variant of the power iteration). The classical power iteration is known to converge for generic values of the initial vector x_0, provided that the matrix M has a unique eigenvalue of maximal modulus. This is the case in particular when the matrix M has positive entries. It is straightforward to find examples in which the power iteration (12) does not converge if the latter positivity condition is relaxed.

Therefore, in order to guarantee that the non-linear iteration (11) converges, we need to find an analogue of the classical positivity condition. Geometrically speaking, the latter means that the map $x \mapsto Mx$ sends the cone $\mathbb{R}^n_+$ to its interior. By analogy, it is natural to require that the abstract operator s sends the cone of PSD matrices $\mathcal{S}_n^+$ to its interior, i.e., to require that $s(X)$ is positive definite as soon as X is a non-zero PSD matrix. We can always make sure that this assumption is satisfied by introducing a damping parameter $\epsilon > 0$ and replacing the operator s by

$$x \mapsto s(x) + \epsilon I \psi(x) \ .$$

This leads to the damped non-linear power iteration

$$X_{k+1} = \frac{s(X_k) + \epsilon\psi(X_k) I}{\psi\left[s(X_k) + \epsilon\psi(X_k) I\right]} \ . \tag{13}$$

We shall refer to (13) as the *non-linear additive power iteration* in the sequel, for the ϵ-perturbation acts in an additive way on s.

The choice of ϵ will be a trade off between making the perturbation small, which requires to choose a small ϵ, and ensuring a fast convergence, which is the case when ϵ is large. For the present experimental purposes, we will see that taking $\epsilon \in [10^{-2}, 10^{-1}]$ leads to satisfactory results. The interest of the non-linear additive power iteration is its simplicity of implementation. However, in the present setting, its convergence study is not-immediate. Indeed, most studies concerning non-linear power type algorithms over cones [28] require the map s to be order preserving, and this assumption is not satisfied here. However, the simple modification of the perturbation idea that we next present leads to a variant of the power algorithm which will be easier to analyze theoretically, and which experimentally gives comparable results. This variant uses a multiplicative perturbation instead of an additive one:

$$X_{k+1} = \frac{s(X_k)^{1-\epsilon}}{\psi\left[s(X_k)^{1-\epsilon}\right]} \ . \tag{14}$$

Recall that for all PSD matrices Y and for all $s > 0$, Y^s denotes the s-th power of Y defined in Section 3.1. For brevity, we write $s(X_k)^{1-\epsilon}$ for $(s(X_k))^{1-\epsilon}$. We refer to (14) as the *non-linear multiplicative power iteration.*

5.2 Convergence analysis of the multiplicative power iteration

The reason for considering the multiplicative power iteration is that, when Y is positive definite and $0 < s < 1$, the map $Y \mapsto Y^s$ is a contraction with respect to a classical metric on the cone, called *Thompson's (part) metric* [28].

Let us recall the definition of this metric. Given two positive definite matrices X and Y, the Thompson's distance $d_T(X, Y)$ is defined by

$$d_T(X, Y) = \log \min\{\alpha > 0 \mid \alpha^{-1} X \preccurlyeq Y \preccurlyeq \alpha X\} \, .$$

It can be easily computed as

$$d_T(X, Y) = \max(\log \lambda_{\max}(X^{-1}Y), \log \lambda_{\max}(Y^{-1}X)) \ ,$$

where $\lambda_{\max}$ denotes the largest eigenvalue of a matrix, see e.g. [18].

It is known (*ibid.*) that a geodesic for Thompson's metric linking I and a positive matrix X is given by the curve sending $t \in [0, 1]$ to $t \mapsto X^t$. By geodesic, we mean that the equality holds in the triangular inequality $d_T(I, X^t) \leqslant d_T(I, X^s) + d_T(X^s, X^t)$ for all $0 < s < t$. The Thompson's part metric has the remarkable property of being invariant with respect to the action of the linear group, that is,

$$d_T(PXP^\top, PYP^\top) = d_T(X, Y)$$

for all $P \in GL_n(\mathbb{R})$. This is similar to the invariance property (7) which we required for the selection operator $\sqcup$, and this is the reason for using this metric.

The geodesics between I and two positive matrices X and Y have the following property, which is known in metric geometry as *nonpositive curvature in the sense of Busemann*,

$$d_T(X^t, Y^t) \leqslant t \, d_T(X, Y) \ . \tag{15}$$

This can be deduced either from [6] or from classical log-majorization inequalities for matrix eigenvalues [38], see [18] for details. This inequality means that the triangles are thin, it is illustrated Figure 5. We warn the reader that non positive curvature in the sense of Busemann is a milder condition than other nonpositive curvature conditions more commonly used like being CAT(0), see [29] for background.

The property given in (15) is the core of the multiplicative perturbation method. Indeed, in the case where the body of the loop contains at least two case branches, the abstract loop operator s involves the *invariant join*. In addition of not preserving the Löwner order, the latter is also possibly expansive: it is easy to find positive definite matrices X_i and Y_i $(1 \leqslant i \leqslant 2)$ for which $d_T(X_1 \sqcup X_2, Y_1 \sqcup Y_2) > d_T(X_1, Y_1) \vee d_T(X_2, Y_2)$. Let C_n denote the Lipschitz constant of the invariant join operator, i.e., the infimum of the positive numbers C such that

$$d_T(X_1 \sqcup X_2, Y_1 \sqcup Y_2) \leqslant C\Big[d_T(X_1, Y_1) \vee d_T(X_2, Y_2)\Big]$$

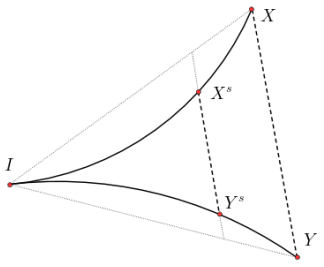


Figure 5: The nonpositive curvature property of Thompson's metric on the space of positive definite matrices.

holds for all positive definite matrices X_1, X_2, Y_1, Y_2. We show in a companion work that $C_n \leqslant 1 + 2(\log n)/\pi$.

The next result shows that the power algorithm does converge for a large enough ϵ. This is mostly of theoretical interest. In the present experiments, we use a much smaller value of ϵ.

THEOREM 6. *Let N denote the number of invariant join operations involved in the abstract loop operator s. If $\epsilon > 1 - 1/(2C_n^N)$, then the sequence X_k produced by the multiplicative power algorithm satisfies*

$$\frac{d_T(X_{k+1}, X_\infty)}{d_T(X_k, X_\infty)} \leqslant 2C_n^N(1-\epsilon) < 1\,, \text{ where } X_\infty = \lim_{k\to\infty} X_k\ .$$

This follows by combining the nonpositive curvature property (15) with standard metric estimates. The details of the proof will be given elsewhere.

Remark 1. The limit X_∞ can be approximated with an accuracy η in $p^* := \lceil \log(\eta/d_T(X_0, X_\infty))/\log(2C_n^N(1-\epsilon))\rceil$ iterations, leading to $O(n^3 N p^*)$ arithmetic operations.

6. BENCHMARKS

We now experiment the methods that we have introduced, and we compare them with alternative techniques based on LMI. The experiments are implemented in MATLAB, running on one core of an 2.2GHz Intel Core i7 with 8Gb RAM.

We compare in Figure 6a the execution time of the two possible implementations of the invariant join given in Section 4: (i) the original algebraic definition (6), depicted in green, (ii) the implementation based on the LMI formulation (9), plotted in red. The comparison is made on random matrices of dimension up to 25. We first observe that at any dimension, the algebraic definition provided a speed-up by a factor of order 10^3–10^4. Moreover, we note that asymptotically, the time to solve the LMI increases as the time to solve the algebraic equation *squared* (beware that the time is given in logarithmic scale).

We next show in Figure 6b the average time to find an invariant using LMIs (in red), the additive nonlinear power algorithm (in blue) and the multiplicative power algorithm (in green). These results were obtained on randomly generated programs of the form:

```
while (rand_bool) {
  switch (rand_bool) {
    case 0:
      x <- S0x;
    case 1:
      x <- S1x;
  }
}
```

where S_0 and S_1 are invertible matrices. For the benchmarks, the power algorithms are always initialized at I_n and the LMI approach for finding an invariant is done by testing the feasibility of the following LMI:

$$\left\{\begin{array}{lll} X & \succcurlyeq & S_0 X S_0^T \\ X & \succcurlyeq & S_1 X S_1^T \\ X & \succ & 0 \end{array}\right. . \tag{16}$$

Such a feasible element X is an invariant for the programs described above. We observe that the power type algorithms bring a significant speed-up over the LMI technique.

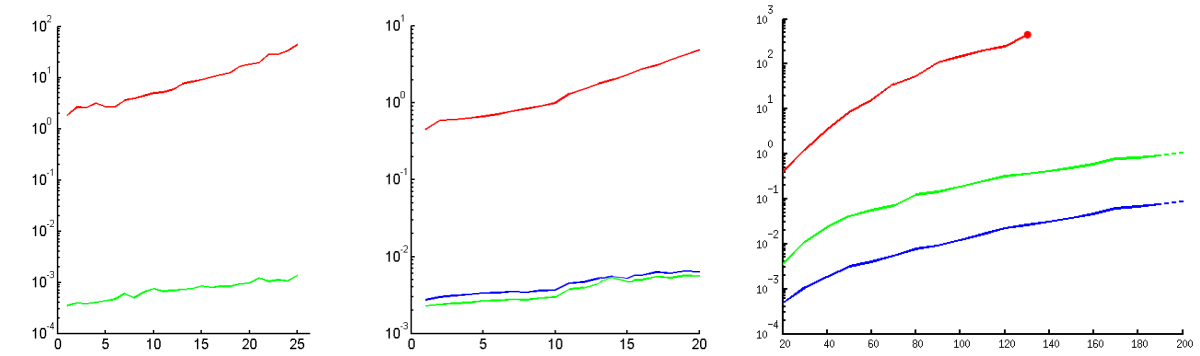
Furthermore, we compare in Figure 6c the execution time of the power algorithms with the resolution of an LMI on a set of high-dimensional linear systems without any switch. The linear systems correspond to parallel simulations of damped oscillators $\ddot{x}_i + c_i\dot{x}_i + k_i x_i = 0$, i.e. given by $S_0 = \left(\begin{smallmatrix} I_n & hI_n \\ -hK & I_n - hC \end{smallmatrix}\right)$, where $h = 0.05$, and $C, K \in \mathbb{R}^{n\times n}$ are diagonal matrices, respectively with positive diagonal elements c_i and k_i. Unlike Figure 6b, the additive power algorithm seems to be faster: here, there is no invariant join computation, hence the cost of the matrix power in the multiplicative algorithm becomes visible. This example highlights another scalability aspect of the power algorithms: while the semidefinite program approach runs out of memory for systems of dimension 140 and beyond, the computation of an invariant through the power-methods is successful and still runs in less than 2s even when there are 200 variables. Note that when there is no switch, the present power algorithm essentially reduces to the classical power algorithm applied to the linear operator $X \mapsto SXS^T$.

In Table 1, we compare our method with an LMI-based approach on a specific set of instances. On top of providing the execution time of the analyses, we also provide the *relative stability margin* of the invariants that we obtain. Given an invariant X, the latter quantity is defined as $\lambda_{\min}\big(X - s(X)\big)/\lambda_{\max}(X)$, where $\lambda_{\min}(M)$ and $\lambda_{\max}(M)$ respectively denote the smallest and largest eigenvalues of the matrix M. This quantity is nonnegative and well defined as an invariant X satisfies $X \succ 0$ and $X \succcurlyeq s(X)$. A large relative margin ensures that the invariant is stable with respect to rounding errors. Except in the last example, the invariants that we obtain using the two approaches are not comparable. However, we give an estimate of the precision of each invariant by using its largest eigenvalue once it has been rescaled to contain the identity matrix, or, in terms of ellipsoids, the unit ball: a size of 1 means that the invariant is very close to the unit ball, while greater sizes mean that the ellipsoid spans far from the unit ball in some directions.

The switched oscillator refers to the example of Figure 1. We also consider another switched linear system, already studied in [37], characterized by the matrices

$$S_0 = \begin{pmatrix} -0.06515 & -0.4744 & 0.3041 \\ -0.4744 & 0.4872 & 0.3732 \\ 0.3041 & 0.3732 & -0.1271 \end{pmatrix}\ ,$$
$$S_1 = \begin{pmatrix} 0.04419 & 0.3155 & -0.04247 \\ 0.1451 & -0.04931 & -0.2805 \\ 0.2833 & -0.01418 & 0.1554 \end{pmatrix}\ .$$

This system allows us to show the importance of the parameter ϵ by its action on the final quality of the invariant. Indeed, if the power algorithms use $\epsilon = 0.1$, then the quality of the invariants is quite bad relative to the one computed by



(a) Invariant join with the LMI (red) and the algebraic definition (green)

(b) Program invariant, with LMI (red), additive power (blue) and multiplicative power (green)

(c) Program invariant in high dimensions, with an LMI (red), additive power (blue) and multiplicative power (green)

Figure 6: Computation times (in s) w.r.t. the dimension of the problem.

Example	ϵ	Time (ms)			Relative stability margin			Invariant size		
		LMI	add	mult	LMI	add	mult	LMI	add	mult
Switched oscillator	0.05	160	**5**	80	4.10^{-3}	4.10^{-4}	$\mathbf{9.10^{-3}}$	**1.52**	1.91	2.48
Switched system	0.1	190	**6**	15	**0.36**	0.07	0.02	**1.56**	23.37	9.78
	0.8	190	**3**	14	**0.36**	**0.36**	**0.36**	1.56	2.19	**1.50**
Symplectic integrator	0.1	100	**1**	3	$\leqslant 5.10^{-3}$	$\leqslant 5.10^{-3}$	$\leqslant 5.10^{-3}$	**1**	**1**	**1**

Table 1: Benchmarks on specific examples

the LMI. In contrast, if they use $\epsilon = 0.8$, then, with even less computation time, the quality of the new invariants similar to the one computed by the LMI.

Finally, we apply the power algorithms to the simulation of the non-damped oscillator $\ddot{x} + c\dot{x} + x = 0$ with $c = 0$. In this case, the energy of the oscillator is preserved. However, the Euler scheme used in the example in Figure 1 is not energy-preserving and even diverges when applied to this system. This is why we use a variant of a symplectic integration scheme $(x_{n+1}, v_{n+1})^T = S(x_n, v_n)^T$, where $S = \begin{pmatrix} 1-\tau^2/2 & \tau^3/4-\tau \\ \tau & 1-\tau^2/2 \end{pmatrix}$ and $\tau = 0.001$. This integration method preserves a quadratic energy function represented by a positive definite matrix Q, i.e. $(x,v)S^TQS(x,v)^T = (x,v)Q(x,v)^T$. This means that there is no stability margin. In spite of that, all three methods return an invariant, scalar multiples of the same matrix $\begin{pmatrix} 1 & 0 \\ 0 & 1-\tau^2/4 \end{pmatrix}$ which is very close to the identity matrix. It is remarkable that both power algorithms successfully compute that invariant, as other algorithms may not even find a bounded invariant [2].

7. CONCLUSION

We developed a static analysis method to synthesize ellipsoidal invariants, avoiding the use of pre-defined templates. We showed by experiments that this method is scalable. For the moment, it is limited to centered ellipsoids. The latter, however, are already very useful invariants, as they have been extensively used in the context of hybrid systems. In particular, our method allows one to find common quadratic Lyapunov functions for switched linear systems, a problem which is receiving attention in the control community.

Apart from dealing with the non centered case, our future work comprises the implementation of a guaranteed version of these computations. We have shown convergence of our algorithms in the real numbers domain, and both experiments and theory show robustness under numerical errors, up to some point. Still, we may rely on guaranteed methods as in e.g. [32] to deliver fully guaranteed computations in this ellipsoidal domain.

We replaced here the invariance problem $s(X) \preccurlyeq X$ by an eigenvalue problem $s(X) = \lambda X$, where $\lambda < 1$ represents a stability margin. This may be related to a perturbation technique already introduced in ASTREE to absorb rounding errors (Section 7.1.4 of [7]). It would be interesting to embed both approaches in a common framework. We also believe that the damping idea behind the power type algorithm of Section 5.1 could be useful to handle other domains (for instance combining ellipsoids and linear templates).

8. ACKNOWLEDGMENTS

We thank the reviewers for their detailed and very helpful comments.

The authors were partially supported by the ANR projects CAFEIN, ANR-12-INSE-0007 and MALTHY, ANR-13-INSE-0003, by ICODE and by the academic research chair "Complex Systems Engineering" of École polytechnique - THALES - FX - DGA - DASSAULT AVIATION - DCNS Research - ENSTA ParisTech - Télécom ParisTech - Fondation ParisTech - FDO ENSTA and by the PGMO programme of EDF and FMJH.

9. REFERENCES

[1] A. Adjé and P.-L. Garoche. Automatic synthesis of piecewise linear quadratic invariants for programs. In *Proceedings of VMCAI*, pages 99–116, 2015.

[2] A. Adjé, S. Gaubert, and E. Goubault. Coupling policy iteration with semi-definite relaxation to compute accurate numerical invariants in static analysis. *Logical Methods in Computer Science*, 8(1), 2012.

[3] J. Angulo. Supremum/infimum and nonlinear averaging of positive definite symmetric matrices. In *Matrix Information Geometry*, pages 3–24. Springer, 2013.

[4] A. Bacciotti and L. Mazzi. Stability of dynamical polysystems via families of liapunov functions. *Jour. Nonlin. Analysis*, 2007.

[5] K. Ball. An elementary introduction to modern convex geometry. In *Flavors of geometry*, volume 31 of *Math. Sci. Res. Inst. Publ.*, pages 1–58. Cambridge Univ. Press, 1997.

[6] R. Bhatia. On the exponential metric increasing property. *Linear Algebra and its Applications*, 375:211–220, 2003.

[7] B. Blanchet, P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. A static analyzer for large safety-critical software. In *Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, PLDI '03, pages 196–207, New York, NY, USA, 2003. ACM.

[8] M. S. Branicky. Multiple lyapunov functions and other analysis tools for switched and hybrid systems. In *IEEE TAC*, volume 43, 1998.

[9] B. Burgeth, A. Bruhn, N. Papenberg, M. Welk, and J. Weickert. Mathematical morphology for matrix fields induced by the loewner ordering in higher dimensions. *Signal Processing*, 87:277–290, 2007.

[10] P. Cousot. Proving program invariance and termination by parametric abstraction, lagrangian relaxation and semidefinite programming. In *Proceedings of VMCAI*, volume 3385 of *LNCS*. Springer, 2005.

[11] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of POPL'77*, pages 238–252. ACM, 1977.

[12] P. Cousot, R. Cousot, J. Feret, L. Mauborgne, A. Miné, D. Monniaux, and X. Rival. The astreé analyzer. In *Proceedings of ESOP'05*, pages 21–30, 2005.

[13] P. Cousot and N. Halbwachs. Automatic discovery of linear restraints among variables of a program. In *Proceedings of POPL'78*, pages 84–96. ACM, 1978.

[14] J. Feret. Static analysis of digital filters. In *Proceedings of ESOP'04*, pages 33–48, 2004.

[15] E. Feron and F. Alegre. Control software analysis, part I open-loop properties. *CoRR*, abs/0809.4812, 2008.

[16] E. Feron and F. Alegre. Control software analysis, part II: closed-loop analysis. *CoRR*, abs/0812.1986, 2008.

[17] G. Gange, J. A. Navas, P. Schachte, H. Søndergaard, and P. J. Stuckey. Abstract interpretation over non-lattice abstract domains. In *Static Analysis*, volume 7935 of *LNCS*, pages 6–24. Springer, 2013.

[18] S. Gaubert and G. Vigeral. A maximin characterization of the escape rate of nonexpansive mappings in metrically convex spaces. *Math. Proc. of Cambridge Phil. Soc.*, 152:341–363, 2012.

[19] K. Ghorbal, E. Goubault, and S. Putot. The zonotope abstract domain taylor1+. In *Proceedings of CAV'09*, pages 627–633, 2009.

[20] G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, 2013.

[21] E. Goubault. Static analysis by abstract interpretation of numerical programs and systems, and FLUCTUAT. In *Proceedings of SAS'13*, pages 1–3, 2013.

[22] M. Johansson and A. Rantzer. Computation of piecewise quadratic lyapunov functions for hybrid systems. *IEEE Transactions on Automatic Control*, 43:555–559, 1998.

[23] V. Kozyakin. Iterative building of Barabanov norms and computation of the joint spectral radius for matrix sets. *Discrete Contin. Dyn. Syst., Ser. B*, 14(1):143–158, 2010.

[24] M. G. Krein and M. A. Rutman. Linear operators leaving invariant a cone in a Banach space. *Uspehi Matematičeskih Nauk*, 3:3–95, 1948. AMS Translations Number 26.

[25] A. B. Kurzhanski and I. Vályi. *Ellipsoidal calculus for estimation and control*. Systems & control. IIASA Boston, 1997.

[26] D. Liberzon. *Switching in systems and control*. Springer, 2003.

[27] MathWorks Inc. Polyspace static analyzer, `fr.mathworks.com/products/polyspace/`.

[28] R. D. Nussbaum. Hilbert's projective metric and iterated nonlinear maps. *Mem. Amer. Math. Soc.*, 75(391), 1988.

[29] A. Papadopoulos. *Metric spaces, convexity and nonpositive curvature*. European Mathematical Society, 2005.

[30] P.A. Parrilo and A. Jadbabaie. Approximation of the joint spectral radius using sum of squares. *Linear Algebra and its Applications*, 428(10):2385–2402, 2008.

[31] P. Peleties and R.A. DeCarlo. Asymptotic stability of m-switched systems using lyapunov-like functions. In *Proceedings of ACC*, pages 1679–1684, 1991.

[32] J. Rohn. A handbook of results on interval linear problems, April 2005.

[33] M. Roozbehani, A. Megretski, and E. Feron. Optimization of lyapunov invariants in verification of software systems. *IEEE Trans. Automat. Contr.*, 58(3):696–711, 2013.

[34] P. Roux. *Analyse statique de systèmes de contrôle commande, synthèse d'invariants non linéaires*. PhD thesis, 2013.

[35] P. Roux, R. Jobredeaux, P.-L. Garoche, and E. Feron. A generic ellipsoid abstract domain for linear time invariant systems. In *Proceedings of HSCC*, pages 105–114, 2012.

[36] S. Sankaranarayanan, H. B. Sipma, and Z. Manna. Scalable analysis of linear systems using mathematical programming. In *Proceedings of VMCAI'05*, pages 25–41, 2005.

[37] H. R. Shaker and J. P. How. Stability analysis for class of switched nonlinear systems. In *American Control Conference (ACC)*, pages 2517–2520, Baltimore, MD, July 2010.

[38] X. Zhan. *Matrix inequalities*, volume 1790 of *Lecture Notes in Mathematics*. Springer, 2002.