Closed Loop Analysis of Control Command Software[*]

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

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

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

ABSTRACT

Recent work addressing the stability analysis of controllers at code level has been mainly focused on the controller alone. However, most of the properties of interest of control software lie in how they interact with their environment. We introduce an extension of the analysis framework to reason on the stability of closed loop systems, i.e., controllers along with a model of their physical environment, the plant. The proposed approach focuses on the closed loop stability of discrete linear control systems with saturations, interacting with a discrete linear plant. The analysis is performed in the state space domain using Lyapunov-based quadratic invariants. We specifically address the automatic synthesis of such invariants and the treatment of floating-point imprecision.

1. INTRODUCTION

While control theorists are familiar with the notion of open and closed-loop stability and have developed various means to study it – e.g. Routh-Hurwitz criterion, Root-Locus or Nyquist stability criteria –, its evaluation or formal verification at code or system level remains an open question.

At the computer science level, these control level properties are rarely known and hard to express or evaluate in the latest stages of system development. In other words, these meaningful requirements of the system tend to disappear when defining the software requirements. This absence precludes a precise analysis of the interaction between the real arithmetic equations characterizing the dynamic of the plant and the actual implementation of the controller in a computer, with all its associated limitations: bounded memory, real time issues, floating point computations, etc.

Addressing these questions, i.e., evaluating control level properties at code level, would allow for a clearer understanding of the behavior of the final system and could avoid detecting issues too late in the development process.

In this paper we adopted the usual approach of control theorists while performing static analysis: our input is a linear and discrete model of the controlled system, or plant, and a linear controller as typically defined in Matlab Simulink or Esterel Scade. This controller represents the actual code that will be compiled and embedded in the final product. We focus here on the closed-loop stability of this system: the plant (described with real arithmetic) + the controller (described with floating point arithmetic).

An interesting outcome of this kind of analyses is the possibility to exhaustively evaluate control-level properties, as usually evaluated through simulation on hybrid systems. It also enables the study of these properties over more complex systems that embed the controller and its safety architecture, including redundancy, voters, data consolidation, etc. To the authors' knowledge no existing approach is yet able to perform a formal and exhaustive analysis of these control level properties, such as stability, over a complete system.

Stability as considered in control theory can be expressed in different manners. In the so-called temporal domain, it amounts to guarantee a BIBO property: Bounded Input, Bounded Output, i.e., assuming a bounded command, the system starting in a stable position (typically 0) should remain in a bounded set of states. A common approach to address these issues is to rely on the theory of Lyapunov functions, characterizing both variants and invariants, as sublevel sets of the functions.

We propose to extend previous work focused on static analysis of open-loop stable controllers to closed-loop systems. The main difficulty faced is the addition of saturations between the plant and the controller. Without those saturations, closed loop analysis would just amount to open loop analysis of a compound system of controller and plant.

We perform the analysis by first characterizing an invariant – the Lyapunov function – over the global system as a quadratic template; then we compute bounds on this template, using a method called policy iterations. This allows us to bound the set of reachable states of the system. Our analysis also ensures that floating point computations won't break the stability property computed and the related theorems are formalized within a proof assistant (Coq) to ensure the required confidence.

Related work.

Few analyses address this issue of closed-loop stability in settings comparable to ours.

[*]This work has been partially supported by the following grants: ANR-INSE-2012-CAFEIN, NSF CrAVES (1135955), ARO MURI W911NF-11-1-0046, and NSF SORTIES (1446758).

At control level, this property is historically the earliest considered. Lots of techniques address it through different means, we refer the interested (computer scientist) reader to an introductory lecture on control theory [12]. In control theory, two main approaches exist to analyze systems. Either the temporal domain, mentionned above, or the frequency domain, more commonly used. In the frequency domain, stability is usually analyzed by studying the pole placement of the transfer function, either on the Laplace transform of the signal (negative-real part), or on its Z-transform (within the unit circle). In both cases, the system has to be fully linearized (ie removing saturation around the linearization point) and the analysis assumes a real semantics, without considering floating points computations.

Even in the temporal domains analyses, as computed by control theorists, the effect of floating point computations performed at the controller level and those potentially done during the analysis itself are typically forgotten.

Lyapunov functions rely on a temporal-domain expression of the system. Basically a Lyapunov function expresses a notion of energy that is shown to (strictly) decrease along the evolution of the system. In computer science terms, they act as both a loop invariant – when they are loose – and a variant – when they are strict. In 2010, Féron [6] proposed to annotate the closed-loop system with Lyapunov based Hoare triples in order to express closed-loop stability at code level. Since then, open-loop stability has been verified at code level, either by proving these Lyapunov annotations [9, 26] or by automatically synthesizing them [20, 21].

On the static analysis side, few existing analyses are able to express the simple property of stability. Most of the existing abstract domains, used to compute an over-approximation of reachable states, rely on linear approximations. Some classes of non linear domains have been introduced specifically to analyze control software, e.g., second order linear filters [5]. Another static analysis approach named policy iteration, see [8] for a global survey, allows to manipulate quadratic properties using semi-definite programming (SDP) numerical solvers [25]. However, an appropriate quadratic template must be provided, and is usually not computed in a tool but rather as a set of scripts in Matlab. In [20], we presented our approach to perform those policy iterations in an automatic manner using the template synthesis of [21]; our analysis is implemented and can be applied on any linear controller. However it was only applicable on globally stable systems while the method proposed in this paper is more suited to deal with saturated systems.

Finally a last line of work has to be mentioned: the vast set of work focusing on hybrid systems [14, 16, 17, 18, 23]. It is difficult to summarize in a few words those analyses. We could however say that usually (1) they address systems of a somewhat different nature with a central continuous behavior described by differential equations and few discrete events (for instance a bouncing ball or an overflowing water tank) whereas controllers perform discrete transitions on a periodical basis, and (2) they focus on bounded time properties rather than invariant generation. These two points can be major obstacles to the adaptation of very interesting techniques to our setting.

For instance, although bounded time analyses do not provide invariants, they enable the use of techniques directly analyzing the continuous plant, such as guaranteed integration [2]. This avoids discretizing the plant, as done in this paper, which can introduce additional conservatism in the analysis.

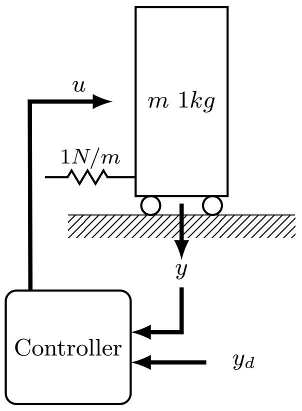


Figure 1: Motivating example: a spring-mass damper

Outline.

The paper is structured as follows. Section 2 presents the running example, inspired from [6]. Section 3 gives the global approach of our analysis. It performs a static analysis of the closed-loop system by computing an over-approximation of all reachable states. It relies on a policy iteration algorithm parametrized by an appropriate quadratic template. In this first setting, we assume the quadratic template given as well as real arithmetic computation.

Then Section 4 offers ways to automatically compute a quadratic template for closed-loop stable linear systems with saturations.

Finally, the last sections address technical issues when analyzing automatically a closed-loop system and how to solve them: removing linear redundancy from the closed-loop system in Sections 5 and taking floating point computations into account in Section 6.

2. MOTIVATING EXAMPLE

We reuse the running example of [6, 7] and achieve an automatic closed-loop stability analysis of this system. This dynamical system is composed of a single mass and a single spring. The control is performed by a lead-lag controller obtained through classical control recipes where the input is defined as the saturation in the interval $[-1, 1]$ of $y - y_d$ with y the measure of the mass position and $|y_d| \leq 0.5$ a bounded command.

Both controller and plant have been discretized at an execution rate of 100Hz.

The plant is described by a linear system over the state variables $p = [x_{p1}\ x_{p2}]^T \in \mathbb{R}^2$, characterized by the matrices $A_P \in \mathbb{R}^{2\times 2}$, $B_P \in \mathbb{R}^{1\times 2}$ and $C_P \in \mathbb{R}^{2\times 1}$ where u denotes the actuator command of the plant and y the projection of the plant state p over the y sensor:

$$\begin{array}{rcl} p_{k+1} & = & A_P p_k + B_P u_k \\ y_{k+1} & = & C_P p_{k+1} \end{array} \quad (1)$$

with

$$A_P := \begin{bmatrix} 1 & 0.01 \\ -0.01 & 1 \end{bmatrix} \quad B_P := \begin{bmatrix} 0.00005 \\ 0.01 \end{bmatrix} \quad C_P := \begin{bmatrix} 1 & 0 \end{bmatrix}$$

The controller without saturation is similarly described by a linear system over the state variables $c = [x_{c1}\ x_{c2}]^T \in \mathbb{R}^2$, controlled by both the feedback from the plant sensors $y \in \mathbb{R}^{d_y}$ and the user command $y_d \in \mathbb{R}$, and parametrized by the four real matrices $A_C \in \mathbb{R}^{2\times 2}$, $B_C \in \mathbb{R}^{1\times 2}$, $C_C \in \mathbb{R}^{2\times 1}$ and $D_C \in \mathbb{R}$:

$$\begin{array}{rcl} c_{k+1} & = & A_C c_k + B_C(y_k - y_{d,k}) \\ u_{k+1} & = & C_C c_{k+1} + D_C(y_{k+1} - y_{d,k+1}) \end{array} \tag{2}$$

with

$$A_C := \begin{bmatrix} 0.4990 & -0.05 \\ 0.01 & 1 \end{bmatrix} \quad B_C := \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$C_C := \begin{bmatrix} 564.48 & 0 \end{bmatrix} \quad D_C := -1280$$

These numerical values have been obtained by control theorists applying any of their classical control recipes.

The resulting closed-loop system is defined by considering Equations (1) and (2) at once. It can be expressed over the state space $x := [c\ p]^T$ as

$$x_{k+1} = Ax_k + By_{d,k} \tag{3}$$

with

$$A := \begin{bmatrix} A_C & B_c C_P \\ B_P C_C & A_P + B_P D_C C_P \end{bmatrix} = \begin{bmatrix} 0.499 & -0.05 & 1 & 0 \\ 0.01 & 1 & 0 & 0 \\ 0.028224 & 0 & 0.936 & 0.01 \\ 5.6448 & 0 & -12.81 & 1 \end{bmatrix}$$

$$B := \begin{bmatrix} -B_C \\ -B_P D_C \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0.064 \\ 12.8 \end{bmatrix}$$

Or, with the saturation over $(y - y_d)$:

$$x_{k+1} = Ax_k + B\,\mathrm{SAT}(Cx_k - y_{d,k}) \tag{4}$$

where

$$A := \begin{bmatrix} A_C & 0 \\ B_P C_C & A_P \end{bmatrix} = \begin{bmatrix} 0.499 & -0.05 & 0 & 0 \\ 0.01 & 1 & 0 & 0 \\ 0.028224 & 0 & 1 & 0.01 \\ 5.6448 & 0 & -0.01 & 1 \end{bmatrix}$$

$$B := \begin{bmatrix} B_C \\ B_P D_C \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -0.064 \\ -12.8 \end{bmatrix} \qquad C := \begin{bmatrix} 0 \\ C_P \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}^T$$

and SAT is defined as

$$\mathrm{SAT}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 \le x \le 1 \\ 1 & \text{if } x > 1 \end{cases}$$

The closed-loop stability will be expressed as a BIBO property: given a bound on the input y_d, find a bound on the vector x.

3. COMPUTING QUADRATIC INVARIANTS FROM GIVEN TEMPLATES

```
xc1 = xc2 = xp1 = xp2 = 0;
while (1) {
  yd = acquire_input();
  assert(yd >= -0.5 && yd <= 0.5);
  oxc1 = xc1; oxc2 = xc2; oxp1 = xp1; oxp2 = xp2;
  xc1 = 0.499 * oxc1 - 0.05 * oxc2 + (oxp1 - yd);
  xc2 = 0.01 * oxc1 + oxc2;
  xp1 = 0.028224 * oxc1 + oxp1 + 0.01 * oxp2
        - 0.064 * (oxp1 - yd);
  xp2 = 5.6448 * oxc1 - 0.01 * oxp1 + oxp2
        - 12.8 * (oxp1 - yd);
  wait_next_clock_tick();
}
```

Figure 2: Analyzed code for the closed loop system.

3.1 Without Saturation

An ideal system is first considered in this section. Thus, an already discretized model of the plant is considered, the controller is purely linear, without any saturation, and its computations are assumed to be performed in the real field $\mathbb{R}$. Moreover, we assume a quadratic Lyapunov function for the closed loop system is already known. Most of these limitations will be alleviated, one by one, in further sections of the paper.

Figure 2 displays the analyzed code for the closed loop system described in the previous section. From such a code, our analyzer extracts the control flow graph of Figure 3.

REMARK 1. *This corresponds to the system presented in Equation (3) with the input y_d bounded by 0.5 ($|y_{d,k}| \le 0.5$ for all k).*

From this control flow graph and a set of expressions t_i on program variables, called *templates*, policy iterations techniques [8] can compute, for each graph vertex, bounds b_i such that $\bigwedge_i t_i \le b_i$ is an invariant. This is basically done by reducing the problem to subproblems, called policies[1], which can be solved thanks to some numerical solver. For instance, linear programming could be used with linear templates. Focusing on quadratic templates, SDP [25] is used here.

Given the templates $t_1 := x^T Px$, $t_2 := x_{c1}^2$, $t_3 := x_{c2}^2$, $t_4 := x_{p1}^2$ and $t_5 := x_{p2}^2$ where x is the vector $[x_{c1}\ x_{c2}\ x_{p1}\ x_{p2}]^T$ and (rounded to four digits)

$$P := \begin{bmatrix} 1.7776 & 1.3967 & -0.6730 & 0.1399 \\ 1.3967 & 1.1163 & -0.4877 & 0.1099 \\ -0.6730 & -0.4877 & 0.3496 & -0.0529 \\ 0.1399 & 0.1099 & -0.0529 & 0.0111 \end{bmatrix},$$

policy iterations compute the invariant

$$t_1 \le 0.2302 \wedge t_2 \le 51.0162 \wedge t_3 \le 15.4720 \\ \wedge t_4 \le 10.1973 \wedge t_5 \le 1767.75$$

which implies

$$|x_{c1}| \le 7.1426 \wedge |x_{c2}| \le 3.9334 \wedge |x_{p1}| \le 3.1933 \\ \wedge |x_{p2}| \le 42.0446.$$

Our static analyzer took 1.28s to produce this result on an Intel Core2 @ 1.2GHz.

[1]The word "strategies" is also used in the literature, with equivalent meaning.

$$\begin{tikzcd}[column sep=huge]
\text{st} \arrow[r, "{\mathit{true}\ ,\ \begin{array}{l} x_{c1} := 0 \\ x_{c2} := 0 \\ x_{p1} := 0 \\ x_{p2} := 0 \end{array}}"] & 1 \arrow[loop right, "{-0.5 \le y_d \le 0.5\ ,\ \begin{array}{l} x_{c1} := 0.499\, x_{c1} - 0.05\, x_{c2} + x_{p1} - y_d \\ x_{c2} := 0.01\, x_{c1} + x_{c2} \\ x_{p1} := 0.028224\, x_{c1} + x_{p1} + 0.01\, x_{p2} - 0.064\, (x_{p1} - y_d) \\ x_{p2} := 5.6448\, x_{c1} - 0.01\, x_{p1} + x_{p2} - 12.8\, (x_{p1} - y_d) \end{array}}"]
\end{tikzcd}$$

Figure 3: Control flow graph for code of Figure 2.

REMARK 2. *The actual maximal reachable values for x_{c1}, x_{c2}, x_{p1} and x_{p2} are* 2.0234*,* 0.0850*,* 0.7796 *and* 23.1525 *respectively. The bounds above are then rather conservative.*

3.2 With Saturation

Actual controllers usually contain saturations to bound the values read from sensors or sent to actuators, in order to ensure that these values remain in the operating ranges of those devices. With such a saturation on its input, the control flow graph of our running example changes to the one shown in Figure 4.

REMARK 3. *This corresponds to the system presented in Equation (4) with the input y_d bounded by 0.5 ($|y_{d,k}| \le 0.5$ for all k).*

Given the templates $t_1 := x^T P x$, $t_2 := x_{c1}^2$, $t_3 := x_{c2}^2$, $t_4 := x_{p1}^2$ and $t_5 := x_{p2}^2$ where x is the vector $[x_{c1}\ x_{c2}\ x_{p1}\ x_{p2}]^T$ and (rounded to four digits)

$$P := \begin{bmatrix} 0.2445 & 0.3298 & -0.0995 & 0.0197 \\ 0.3298 & 1.0000 & -0.0672 & 0.0264 \\ -0.0995 & -0.0672 & 0.0890 & -0.0075 \\ 0.0197 & 0.0264 & -0.0075 & 0.0016 \end{bmatrix},$$

policy iterations compute the invariant

$$t_1 \le 0.1754 \wedge t_2 \le 6.1265 \wedge t_3 \le 0.3505 \\ \wedge t_4 \le 4.1586 \wedge t_5 \le 1705.1748$$

which implies

$$|x_{c1}| \le 2.4752 \wedge |x_{c2}| \le 0.5921 \wedge |x_{p1}| \le 2.0393 \wedge \\ |x_{p2}| \le 41.2938.$$

Our static analyzer took 1.39s to produce this result on an Intel Core2 @ 1.2GHz.

4. COMPUTING SUITABLE QUADRATIC TEMPLATES

In the previous section, a quadratic template P was required to perform the analysis. In order to get a fully automatic analysis method, this section addresses the computation of such templates.

4.1 Without Saturation

Given a system with $x_0 = 0$ and $x_{k+1} = Ax_k + By_k$ with a bounded input y ($\|y_k\|_\infty \le 1$ for all k), control theorist have known for long that this system is stable (i.e., x remains bounded) if and only if the Lyapunov equation[2] [3, 13]

$$A^T P A - P \prec 0$$

admits a positive definite matrix P (i.e. for all $x \neq 0$, $x^T P x > 0$) as a solution.

[2]In which $M \prec 0$ means that M is negative definite (i.e., $-M$ is positive definite).

This equation can be numerically solved thanks to a SDP solver [3, 25]. However, in practice, it has many solutions, some dramatically worse than others (i.e., leading to much less precise invariants). Fortunately enough, simple heuristics [21] allow to easily compute good templates P (i.e., with which policy iterations can compute small invariants). The matrix P used in Section 3.1 was computed this way. Thus, the analysis becomes completely automatic, since matrices A and B, needed to compute P, can just be extracted from the control flow graph of Figure 3. Our static analyzer took 0.76s to produce this template on an Intel Core2 @ 1.2GHz, hence a fully automatic computation in a total of 2.19s.

4.2 With Saturation

The previous method does not readily apply for a system with saturation such as the one of Section 3.2.

A first idea could be to try to generate, as previously described, a quadratic template P for each edge of the control flow graph of Figure 4. This approach sometimes proves successful but fails on our running example. Indeed, only one of the edges of the graph on Figure 4 leads to a template P (for other edges, the Lyapunov equation has no solution) and this template does not allow policy iterations to compute a worthwhile invariant on the whole program.

Using common Lyapunov functions constitutes a second idea. That is, looking for a solution to the conjunction of Lyapunov equations for each edge. Again, this fails since Lyapunov equations have no solution for some of the edges. This is due to the fact that the closed-loop system is not globally stable. Indeed, intuitively, when its input is saturated, the controller is not able to stabilize any arbitrary state of the plant.

The following two Sections 4.2.1 and 4.2.2 offer two alternative ways to generate a template $x^T P\, x$ such that $x^T P\, x \le r$ is an invariant of the closed loop system with saturation for some r. Both methods manage to produce such a template but more investigations are needed to determine their relative advantages and drawbacks.

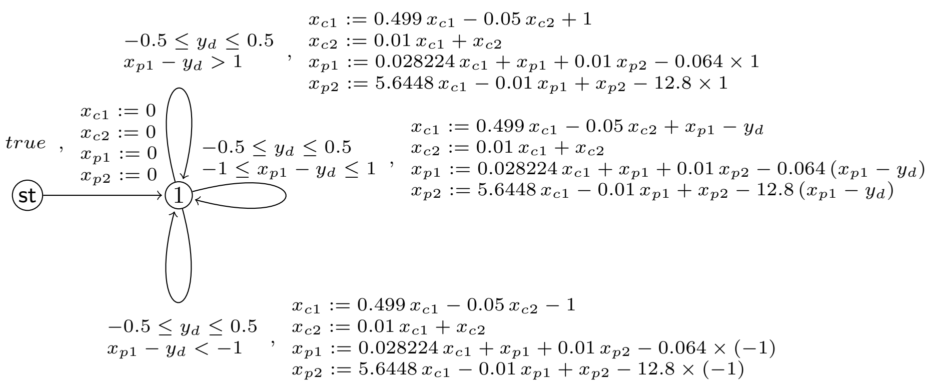
4.2.1 Linearizing the Saturation

One solution in this case, strongly inspired from [6], provides a heuristic that can be used on systems with saturations, such as the one described in equation (4). Indeed, let P be a candidate matrix describing an invariant ellipsoid for the system. We try to characterize P as closely as possible while keeping the solving process tractable:

Assuming $x_k^T P x_k \le 1$, a bound on $|Cx_k|$ is given by $\gamma := \sqrt{CP^{-1}C^T}$. Since $|y_{d,k}| \le 0.5$, the constant $\tilde{\gamma} := \gamma + 0.5$ is an upper bound on $|Cx_k - y_{d,k}|$. Letting $y_{c,k} := \mathrm{SAT}(Cx_k - y_{d,k})$, we have the following sector bound:

$$\left(y_{c,k} - \frac{1}{\tilde{\gamma}}(Cx_k - y_{d,k})\right)(y_{c,k} - (Cx_k - y_{d,k})) \le 0. \quad (5)$$

Figure 5 illustrates the reason for this inequality. With the



$-0.5 \le y_d \le 0.5$
$x_{p1} - y_d > 1$
$x_{c1} := 0.499\, x_{c1} - 0.05\, x_{c2} + 1$
$x_{c2} := 0.01\, x_{c1} + x_{c2}$
$x_{p1} := 0.028224\, x_{c1} + x_{p1} + 0.01\, x_{p2} - 0.064 \times 1$
$x_{p2} := 5.6448\, x_{c1} - 0.01\, x_{p1} + x_{p2} - 12.8 \times 1$
$true$
$x_{c1} := 0$
$x_{c2} := 0$
$x_{p1} := 0$
$x_{p2} := 0$
st
1
$-0.5 \le y_d \le 0.5$
$-1 \le x_{p1} - y_d \le 1$
$x_{c1} := 0.499\, x_{c1} - 0.05\, x_{c2} + x_{p1} - y_d$
$x_{c2} := 0.01\, x_{c1} + x_{c2}$
$x_{p1} := 0.028224\, x_{c1} + x_{p1} + 0.01\, x_{p2} - 0.064\,(x_{p1} - y_d)$
$x_{p2} := 5.6448\, x_{c1} - 0.01\, x_{p1} + x_{p2} - 12.8\,(x_{p1} - y_d)$
$-0.5 \le y_d \le 0.5$
$x_{p1} - y_d < -1$
$x_{c1} := 0.499\, x_{c1} - 0.05\, x_{c2} - 1$
$x_{c2} := 0.01\, x_{c1} + x_{c2}$
$x_{p1} := 0.028224\, x_{c1} + x_{p1} + 0.01\, x_{p2} - 0.064 \times (-1)$
$x_{p2} := 5.6448\, x_{c1} - 0.01\, x_{p1} + x_{p2} - 12.8 \times (-1)$

Figure 4: Control flow graph for the system with a saturation.

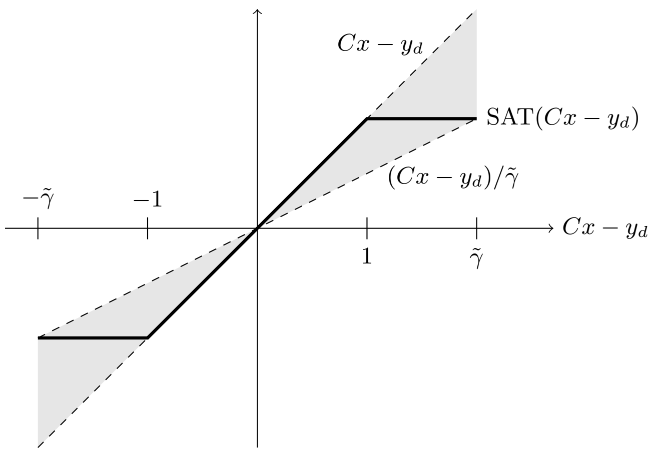


Figure 5: Illustration of the sector bound relationship. The equality $y_c = \mathrm{SAT}(Cx - y_d)$ (thick line) is abstracted by the inequalities $(Cx - y_d)/\tilde{\gamma} \le y_c \le Cx - y_d$ (grey area).

added bound $\tilde{\gamma}$ on $|Cx_k - y_{d,k}|$, we see that $y_{c,k}$ necessarily lies between $Cx_k - y_{d,k}$ and $\frac{1}{\tilde{\gamma}}(Cx_k - y_{d,k})$. Then $y_{c,k} - \frac{1}{\tilde{\gamma}}(Cx_k - y_{d,k})$ and $y_{c,k} - (Cx_k - y_{d,k})$ must be of opposite signs, hence the inequality.

We thus look for a matrix P such that

$$\sqrt{CP^{-1}C^{\mathrm{T}}} \le \gamma \tag{6}$$

and

$$\left(x_k^{\mathrm{T}} P x_k \le 1 \wedge y_{d,k}^2 \le 0.5^2 \wedge (5)\right) \implies x_{k+1}^{\mathrm{T}} P x_{k+1} \le 1. \tag{7}$$

Defining an extended state vector $\epsilon_k := [x_k\ y_{c,k}\ y_{d,k}\ 1]^T$ and the matrices

$$\mathcal{U} := \begin{bmatrix} A^{\mathrm{T}}PA & A^{\mathrm{T}}PB & 0_{4\times 1} & 0_{4\times 1} \\ B^{\mathrm{T}}PA & B^{\mathrm{T}}PB & 0 & 0 \\ 0_{1\times 4} & 0 & 0 & 0 \\ 0_{1\times 4} & 0 & 0 & -1 \end{bmatrix}$$

$$\mathcal{V} := \begin{bmatrix} P & 0_{4\times 1} & 0_{4\times 1} & 0_{4\times 1} \\ 0_{1\times 4} & 0 & 0 & 0 \\ 0_{1\times 4} & 0 & 0 & 0 \\ 0_{1\times 4} & 0 & 0 & -1 \end{bmatrix},$$

$$\mathcal{W} := \begin{bmatrix} \frac{2}{\tilde{\gamma}}C^{\mathrm{T}}C & -\left(1+\frac{1}{\tilde{\gamma}}\right)C^{\mathrm{T}} & -\frac{2}{\tilde{\gamma}}C^{\mathrm{T}} & 0_{4\times 1} \\ -\left(1+\frac{1}{\tilde{\gamma}}\right)C & 2 & 1+\frac{1}{\tilde{\gamma}} & 0 \\ -\frac{2}{\tilde{\gamma}}C & 1+\frac{1}{\tilde{\gamma}} & \frac{2}{\tilde{\gamma}} & 0 \\ 0_{1\times 4} & 0 & 0 & 0 \end{bmatrix},$$

$$\mathcal{Y} := \begin{bmatrix} 0_{4\times 4} & 0_{4\times 1} & 0_{4\times 1} & 0_{4\times 1} \\ 0_{1\times 4} & 0 & 0 & 0 \\ 0_{1\times 4} & 0 & 1 & 0 \\ 0_{1\times 4} & 0 & 0 & -0.5^2 \end{bmatrix},$$

we can rewrite equation (7) as

$$\left(\epsilon_k^{\mathrm{T}}\mathcal{V}\epsilon_k \le 0 \wedge \epsilon_k^{\mathrm{T}}\mathcal{Y}\epsilon_k \le 0 \wedge \epsilon_k^{\mathrm{T}}\mathcal{W}\epsilon_k \le 0\right) \implies \epsilon_k^{\mathrm{T}}\mathcal{U}\epsilon_k \le 0.$$

Equation (7) can then be relaxed by S-procedure: it will hold if there exists positive coefficients λ, μ, and ν, such that

$$\mathcal{U} - \lambda\mathcal{V} - \mu\mathcal{W} - \nu\mathcal{Y} \preceq 0. \tag{8}$$

Equation (6) can be rewritten using Schur complement:

$$\begin{bmatrix} \gamma^2 & C \\ C^{\mathrm{T}} & P \end{bmatrix} \preceq 0. \tag{9}$$

Note that for fixed λ and γ, equations (8) and (9) form a Linear Matrix Inequality (LMI) in P, μ and ν, which means it can be solved by an SDP solver. $\tilde{\gamma} = \gamma + 0.5$ is expected to be larger than 1 (otherwise the saturation would never be activated), moreover since the saturation should somewhat "bound" this value, we can expect it not to span over multiple orders of magnitude. We also know that $\lambda \in (0, 1)$ thanks to the bottom right coefficient of the LMI (8) (since $\nu > 0$). One possible strategy is then to iterate on potential values of λ and γ, and solving the corresponding LMI at each iteration. If a solution exists, it will provide the invariant $x^T P x \le 1$ for the system with saturation.

For our running example, we generated a suitable template in 279s on an Intel Core2 @ 2.4GHz. Values for λ are chosen by exploring $(0, 1)$ with numbers of the form $\frac{k}{2^i}$ for increasing values of $i \ge 1$, and $k < 2^i$. For each choice of λ, the LMI is solved with values of $\tilde{\gamma}$ ranging from 1 to 5 by increments of .1. The solution is found for $\lambda = \frac{63}{64}$ and $\tilde{\gamma} = 3.1$, which amounts to 2605 calls to the LMI solver.

4.2.2 *First Abstracting the Disturbance*

In the previous section, the method used was mainly based on an abstraction of the saturation. This section exposes an alternative method in which the disturbance y_d, rather than the saturation, is abstracted.

Let us first neglect the disturbance y_d and look for a Lyapunov function for the following system:

$$x_{k+1} = \begin{cases} Ax_k - B & \text{if } Cx_k \leq -0.5 \\ (A + BC)x_k & \text{if } -1.5 \leq Cx_k \leq 1.5 \\ Ax_k + B & \text{if } Cx_k \geq 0.5 \end{cases} \qquad (10)$$

where A, B and C are the matrices given in (4).

REMARK 4. *y_d is abstracted in the sense that the term $(A + BC)x - By_d$ of (4) is replaced by $(A + BC)x$ in (10). Similarly, guards such as $Cx - y_d \leq -1$ are replaced by $Cx \leq -0.5$ (since $|y_d| \leq 0.5$).*

REMARK 5. *In case $0.5 \leq \pm Cx_k \leq 1.5$, the system non deterministically takes one of the two available transitions, the transition taken by the actual system (4) being determined by the value of the abstracted variable y_d.*

A quadratic Lyapunov function $x \mapsto x^T Px$ for this system must then satisfy $x_{k+1}^T Px_{k+1} \leq x_k^T Px_k$ for all $x_k \in \mathbb{R}^4$ and all possible transitions from x_k to x_{k+1}. Hence for all $x \in \mathbb{R}^4$

$$\begin{cases} Cx \leq -0.5 \Rightarrow (Ax - B)^T P(Ax - B) \leq x^T Px \\ -1.5 \leq Cx \leq 1.5 \Rightarrow ((A + BC)x)^T P((A + BC)x) \leq x^T Px \\ Cx \geq 0.5 \Rightarrow (Ax + B)^T P(Ax + B) \leq x^T Px. \end{cases}$$

It is worth noting that we can get rid of the first constraint by a symmetry argument. Indeed, the first constraint holds for some x if and only if the third one holds for $-x$. Similarly, we can remove the left part of the implication in the second constraint. Indeed, the right part of the implication holds for some x if and only if it holds for αx and, for α small enough, αx will satisfy the left part of the implication. Thus $x \mapsto x^T Px$ is a Lyapunov equation for (10) if and only if for all $x \in \mathbb{R}^4$

$$\begin{cases} ((A + BC)x)^T P((A + BC)x) \leq x^T Px \\ Cx \geq 0.5 \Rightarrow (Ax + B)^T P(Ax + B) \leq x^T Px. \end{cases} \qquad (11)$$

By defining the vector $x' := [x^T\ 1]^T$, this can be rewritten

$$\begin{cases} x^T(A + BC)^T P(A + BC)x \leq x^T Px \\ [C\ 0]\,x' \geq 0.5 \Rightarrow x'^T[A\ B]^T P\,[A\ B]\,x' \leq x'^T[I_4\ 0]^T P\,[I_4\ 0]\,x'. \end{cases}$$

By a langrangian relaxation, this holds when there exists a $\lambda \geq 0$ such that

$$\begin{cases} P - (A + BC)^T P(A + BC) \succeq 0 \\ [I_4\ 0]^T P\,[I_4\ 0] - [A\ B]^T P\,[A\ B] - \lambda \begin{bmatrix} 0 & C^T \\ C & -1 \end{bmatrix} \succeq 0 \end{cases}$$

where $M \succeq 0$ means that the matrix M is positive semi-definite (i.e., for all x, $x^T Px \geq 0$).

We eventually want the template $x^T P\,x$ to provide an invariant for the original system with the disturbance y_d. For that purpose, we not only want $(A + BC)^T P(A + BC)$ in the first inequality to be less than P but rather the least possible, in order to leave some room to later reintroduce y_d. That is, we look for τ_{min}, the least possible $\tau \in (0, 1)$ satisfying

$$\tau P - (A + BC)^T P(A + BC) \succeq 0$$

for some positive definite matrix P. For any given value of τ, this is a LMI and a SDP solver can be used to decide whether a P satisfying it exists or not. Thus, τ_{min} can be efficiently approximated by a bisection search in the interval $(0, 1)$.

REMARK 6. *τ_{min} is also called minimum* decay rate *[28].*

We are thus looking for a positive definite matrix P satisfying

$$\begin{cases} \tau_{min}P - (A + BC)^T P(A + BC) \succeq 0 \\ [I_4\ 0]^T P\,[I_4\ 0] - [A\ B]^T P\,[A\ B] - \lambda \begin{bmatrix} 0 & C^T \\ C & -1 \end{bmatrix} \succeq 0. \end{cases}$$

This is a LMI and could then be fed to a SDP solver. Unfortunately, it has no solution. Indeed, A has eigenvalues larger than 1 and taking x large enough can break the second constraint in (11) for any value of P.

However, x is saturated when $Cx \geq 1.5$ and it is then reasonable to expect Cx not to go to far beyond this threshold. We thus need to add a constraint $Cx \leq \gamma$ for some $\gamma > 1.5$, in the hope that the generated invariant will eventually satisfy it. This results in the following LMI

$$\begin{cases} \tau_{min}P - (A + BC)^T P(A + BC) \succeq 0 \\ [I_4\ 0]^T P\,[I_4\ 0] - [A\ B]^T P\,[A\ B] - \lambda D \succeq 0 \end{cases} \qquad (12)$$

where $D := [C\ \ -0.5]^T\,[-C\ \gamma] + [-C\ \gamma]^T\,[C\ \ -0.5]$.

Finally, for a solution P of the above LMI, $x^T Px \leq r_{max}$ should be a good candidate invariant for the original system (4), with $r_{max} := \frac{\gamma^2}{CP^{-1}C^T}$ the largest r such that $x^T Px \leq r$ implies $Cx \leq \gamma$.

On our running example, 15 bisection search iterations first enable to compute $\tau_{min} = 0.9804$ (rounded to four digits). Then, the values 2, 3, 4,... are successively tried for γ in (12). The LMI appears to have a solution for $\gamma = 2$ and $\gamma = 3$ but not for $\gamma = 4$. The value of P obtained for the last succeeding value of γ ($\gamma = 3$) is then kept as a template and fed to policy iterations along with $r_{max} = 0.26$. All these computations (bisection search for τ_{min}, tests for γ and computation of r_{max}) took 0.83s on an Intel Core2 @ 1.2GHz.

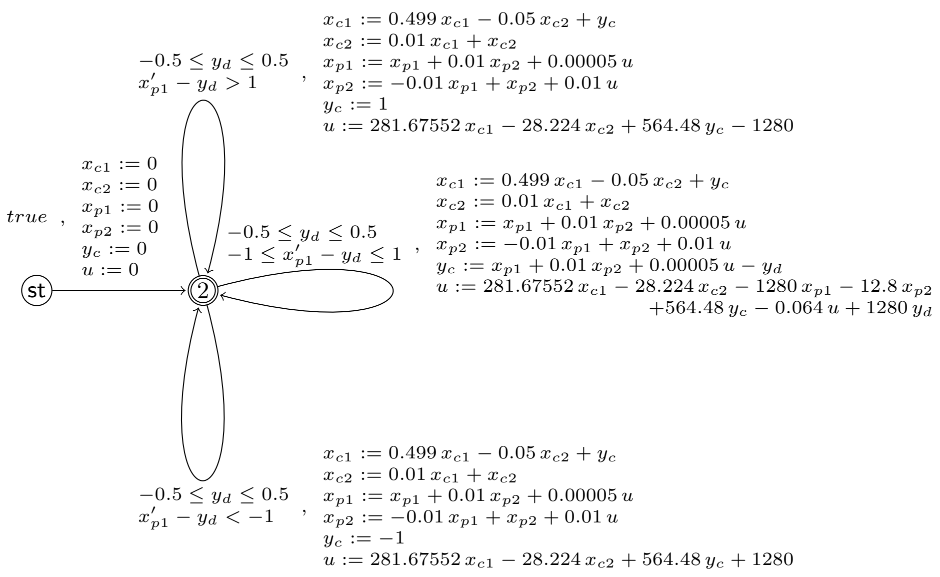
This matrix P is the one used in Section 3.2 in which it had been seen that policy iterations were able to refine the radius $r_{max} = 0.26$ down to 0.1754 and infer bounds for each dimension. Despite the fact that the disturbance y_d was abstracted to generate P, it is worth noting that policy iterations are performed on the complete system, with y_d.

REMARK 7. *Although quite heuristic, the choice for γ does not seem that difficult since any value in the interval $(2.40, 3.85)$ would also have led to a good template.*

5. REMOVING REDUNDANT VARIABLES

In previous sections, the analyzed closed loop system was written as a single set of equations mixing the controller and the model of the plant (c.f., code of Figures 2 and 4). However, it would be a lot more convenient to clearly separate the code of the controller (which is intended to be compiled and executed on the actual device) and the model of the plant (a model of the controlled physical system, part of the specification but not intended to be compiled nor executed). For our running example, this results in the control flow graph of Figure 6.

Along the rightmost edge of this graph, the assignment can be written $x_{k+1} = Ax_k + By_{d,k+1}$ where x denotes the



$-0.5 \le y_d \le 0.5$
$x'_{p1} - y_d > 1$,
$x_{c1} := 0.499\,x_{c1} - 0.05\,x_{c2} + y_c$
$x_{c2} := 0.01\,x_{c1} + x_{c2}$
$x_{p1} := x_{p1} + 0.01\,x_{p2} + 0.00005\,u$
$x_{p2} := -0.01\,x_{p1} + x_{p2} + 0.01\,u$
$y_c := 1$
$u := 281.67552\,x_{c1} - 28.224\,x_{c2} + 564.48\,y_c - 1280$
$true$,
$x_{c1} := 0$
$x_{c2} := 0$
$x_{p1} := 0$
$x_{p2} := 0$
$y_c := 0$
$u := 0$
st
2
$-0.5 \le y_d \le 0.5$
$-1 \le x'_{p1} - y_d \le 1$,
$x_{c1} := 0.499\,x_{c1} - 0.05\,x_{c2} + y_c$
$x_{c2} := 0.01\,x_{c1} + x_{c2}$
$x_{p1} := x_{p1} + 0.01\,x_{p2} + 0.00005\,u$
$x_{p2} := -0.01\,x_{p1} + x_{p2} + 0.01\,u$
$y_c := x_{p1} + 0.01\,x_{p2} + 0.00005\,u - y_d$
$u := 281.67552\,x_{c1} - 28.224\,x_{c2} - 1280\,x_{p1} - 12.8\,x_{p2}$
$+564.48\,y_c - 0.064\,u + 1280\,y_d$
$-0.5 \le y_d \le 0.5$
$x'_{p1} - y_d < -1$,
$x_{c1} := 0.499\,x_{c1} - 0.05\,x_{c2} + y_c$
$x_{c2} := 0.01\,x_{c1} + x_{c2}$
$x_{p1} := x_{p1} + 0.01\,x_{p2} + 0.00005\,u$
$x_{p2} := -0.01\,x_{p1} + x_{p2} + 0.01\,u$
$y_c := -1$
$u := 281.67552\,x_{c1} - 28.224\,x_{c2} + 564.48\,y_c + 1280$

Figure 6: Control flow graph with extra variables (x'_{p1} denotes $x_{p1} + 0.01\,x_{p2} + 0.00005\,u$).

vector $[x_{c1}\ x_{c2}\ x_{p1}\ x_{p2}\ y_c\ u]^T$ and

$$A := \begin{bmatrix} 0.499 & -0.05 & 0 & 0 & 1 & 0 \\ 0.01 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0.01 & 0 & 0.00005 \\ 0 & 0 & -0.01 & 1 & 0 & 0.01 \\ 0 & 0 & 1 & 0.01 & 0 & 0.00005 \\ 281.67552 & -28.224 & -1280 & 12.8 & 564.48 & 0.064 \end{bmatrix},$$

$$B := \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -1 \\ 1280 \end{bmatrix}. \tag{13}$$

Matrix A is noticeably singular (for instance, its fifth row is equal to its third one) and we incur less precise or more computationally expensive invariants at best, serious numerical troubles at worse, if we try to compute invariants from it as described in Section 4. Thus, we first have to reduce the number of variables from six to four (the rank of matrix A).

EXAMPLE 1. *Considering the system defined by $x_0 = 0$ and $x_{k+1} = Ax_k + By_{k+1}$ where x denotes the vector $[u\ v]^T$ and*

$$A := \begin{bmatrix} 0.5 & 0.1 \\ 0.25 & 0.05 \end{bmatrix}, \qquad B := \begin{bmatrix} 0 \\ 4.5 \end{bmatrix},$$

it appears that $v_{k+1} = 0.5\,u_{k+1} + 4.5\,y_{k+1}$ for all k. Hence[3] $u_0 = 0$ and $u_{k+1} = 0.55\,u_k + 0.45\,y_k$. Assuming that y_k is bounded by 1, this enables to prove that u_k lies in the interval (ellipsoid of dimension 1) $[-1, 1]$ (hence v_k remains in $[-5, 5]$). In comparison, directly analyzing the two-variable system with an ellipsoid of dimension 2 would lead to much larger bounds as illustrated on Figure 7.

From edges of the control flow graph of Figure 6, matrices such as the one of (13) can be extracted. From such

[3]Assuming $y_0 = 0$, we have $v_k = 0.5\,u_k + 4.5\,y_k$ for all k.

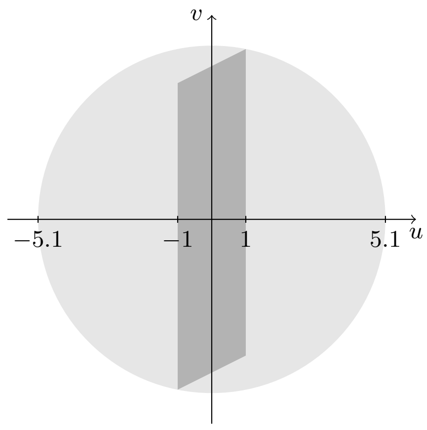


Figure 7: Illustration of Example 1. The dark gray parallelogram represents the invariant that can be computed after eliminating the redundant variable whereas the light gray ellipse is an invariant directly computed on the two variables system (other choices of ellipses are possible, we chose the one contained in the smallest possible disc). The latter is noticeably much larger than the former.

matrices, a Gaussian elimination implemented with rational arithmetic allows to discover linear dependencies such as $y_{c,k+1} = x_{p1,k+1} - y_{d,k+1}$ and $u_{k+1} = 564.48\, x_{c1,k+1} - 1280\, x_{p1,k+1} + 1280\, y_{d,k+1}$ along the rightmost edge of Figure 6. Unfolding these dependencies finally leads to a control flow graph without the redundant variables. For our running example, we obtain the graph of Figure 4.

After this preprocessing, the analysis can proceed from the freshly computed graph as in Sections 3 and 4.

REMARK 8. *Removing redundant variables can also be seen as the discovery and use of linear equality invariants. This symbolic analysis is of particular interest for the following numerical analysis not well suited to handle equalities.*

6. FLOATING-POINT ROUNDING ERRORS

Two fundamentally different issues arise with floating-point arithmetic:

The analysis itself is carried out with floating-point computations for the sake of efficiency, this usually works well in practice but might give erroneous results, hence the need for some a-posteriori validation;

The analyzed system uses floating-point arithmetic with rounding errors, making it behave differently from the way it would using real arithmetic.

Proofs of the mathematical results used in this section being rather painful and error prone, they were mechanically checked using the proof assistant Coq [4] which gives us a very high level of confidence in these results. Our development (3.8 kloc of Coq) is available at `http://cavale.enseeiht.fr/formalbounds2014/` and based on the Flocq library [1] for the formal definition of floating-point arithmetic.

6.1 Floating-Point Arithmetic in the Analyzer

For the sake of efficiency, the SDP solvers used perform all their computations on floating-point numbers and do not offer any strict soundness guarantee on their results. To address this issue, we adopt the following strategy:

- first perform policy iterations with unsound solvers, just padding the equations to hopefully get a correct result;
- then check the soundness of previous result.

From a control flow graph and a set of templates t_j, policy iterations return a vector of values $b_{v,j} \in \overline{\mathbb{R}}$ such that, at each vertex v of the control flow graph, $\bigwedge_j t_j \le b_{v,j}$ should be an invariant. Since this result was computed using floating-point arithmetic, we have to check it. This amounts to check that for each edge from v to v' in the control flow graph and for each template t_j, the following inequality holds

$$b_{v',j} \ge \max\left\{ r(t_j) \;\middle|\; e \le c \wedge \bigwedge_{j'} (t_{j'} \le b_{v,j'}) \right\} \tag{14}$$

where $e \le c$ and r are respectively the constraint and the assignments associated to the edge between v and v'. Informally, this inequalities mean that, if for all j the constraints $t_j \le b_j$ hold in vertex v, then they also hold in vertex v'.

This can be checked efficiently [19], although details are outside the scope of this paper.

6.2 Floating-Point Arithmetic in the Analyzed Program

Up to this point, all computations performed by the analyzed code were assumed to be done in the real field $\mathbb{R}$. However, developers of control systems commonly resort to floating-point numbers.

Floating-point arithmetic can potentially lead to far different results than the ones expected with real numbers [10, 15, 22], thus floating-point computations must be taken into account in our analysis.

Definition 1. $\mathbb{F} \subset \mathbb{R}$ denotes the set of floating-point values and $\mathrm{fl}(e) \in \mathbb{F}$ represents the floating-point evaluation of expression e with any rounding mode and any order of evaluation[4].

EXAMPLE 2. *The value* $\mathrm{fl}(1+2+3)$ *can be either* $\mathrm{round}(1+\mathrm{round}(2+3))$ *or* $\mathrm{round}(\mathrm{round}(1+2)+3)$ *with* round *any valid rounding mode (toward* $+\infty$ *or to nearest for instance).*

Taking floating-point arithmetic into account, (14) becomes

$$b_{v',j} \ge \max\left\{ \mathrm{fl}(r(t_j)) \;\middle|\; \mathrm{fl}(e) \le \mathrm{fl}(c) \wedge \bigwedge_{j'} (t_{j'} \le b_{v,j'}) \right\} \tag{15}$$

since guards $e \le c$ and assignments r are now performed in $\mathbb{F}$. All the remaining of (14) is kept unchanged since it only corresponds to mathematical expressions (in $\mathbb{R}$) and not to parts of the analyzed program (in $\mathbb{F}$).

We will first see how to handle the guards $\mathrm{fl}(e) \le \mathrm{fl}(c)$ then the assignments $\mathrm{fl}(r(t_j))$. Our goal is to obtain a slightly modified version of (14) to be able to proceed as in the previous Section 6.1.

Guards.

For all guards $e \le c$, the actually implemented guard is $\mathrm{fl}(e) \le \mathrm{fl}(c)$ and there can be values of program variables such that the later holds but not the former. Our goal is to define a $c' \ge c$ such that $\mathrm{fl}(e) \le \mathrm{fl}(c)$ implies $e \le c'$. We will only consider the case of linear guards $a^T x \le c$ with $a \in \mathbb{R}^n$, $c \in \mathbb{R}$, $x \in \mathbb{F}^n$.

Definition 2. `eps` is the precision of the floating-point format $\mathbb{F}$ and `eta` its precision in case of underflows. In particular, we have for all $x, y \in \mathbb{F}$

$$\exists \delta \in \mathbb{R}, |\delta| \le \texttt{eps} \wedge \mathrm{fl}(x+y) = (1+\delta)(x+y)$$

and

$$\exists \delta, \eta \in \mathbb{R}, |\delta| \le \texttt{eps} \wedge |\eta| \le \texttt{eta} \wedge \mathrm{fl}(x \times y) = (1+\delta)(x \times y) + \eta.$$

REMARK 9. *Those are fairly classic notations and results [10, 22].* `eps` *and* `eta` *are very small constants defined by the floating-point format in use. For instance,* `eps` $= 2^{-53} (\simeq 10^{-16})$ *and* `eta` $= 2^{-1075} (\simeq 10^{-323})$ *for the IEE754 binary64 format with a rounding to nearest*[5] *[11].*

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

[5]Usual implementation of type `double` in C.

THEOREM 1. *Assuming* $2(n+1)\texttt{eps} < 1$, *we have for all* $a \in \mathbb{R}^n$ *and* $x \in \mathbb{F}^n$

$$\left|\mathrm{fl}\left(\sum_{i=1}^n a_i x_i\right) - \sum_{i=1}^n a_i x_i\right| \le \gamma_{n+1} \sum_{i=1}^n |a_i x_i| + 2\left(n + \sum_{i=1}^n |x_i|\right)\texttt{eta}$$

where $\gamma_{n+1} := \frac{(n+1)\texttt{eps}}{1-(n+1)\texttt{eps}}$.

This theorem gives us the desired property: for all $c' \ge \mathrm{fl}(c) + \gamma_{n+1}\,|a|^T|x| + 2(n + \|x\|_1)\texttt{eta}$, the inequality $\mathrm{fl}\left(a^T x\right) \le \mathrm{fl}(c)$ implies $a^T x \le c'$. Since we used templates x_i^2 for each variable x_i of the analyzed program, we actually have a bound on $\|x\|_1$ and it is easy to compute such an appropriate c' (for instance with floating-point arithmetic and rounding toward $+\infty$).

Assignments.

Things are a bit more involved than in the case of guards. We are now looking for a $b'_{v',j} \le b_{v',j}$ such that $r(t_j) \le b'_{v',j}$ implies $\mathrm{fl}(r(t_j)) \le b_{v',j}$, that is $[x^T\ 1]R_r^T P_{t_j} R_r [x^T\ 1]^T \le b'_{v',j}$ implies $\mathrm{fl}\left([x^T\ 1]R_r^T\right) P_{t_j}\,\mathrm{fl}\left(R_r[x^T\ 1]^T\right) \le b_{v',j}$ where R_r and P_{t_j} are matrix representations of assignment r and template t_j respectively[6]. The next theorem will guarantee us that this property holds for any $b'_{v',j} \le \left(\sqrt{b_{v',j}} - \sqrt{s}\|e\|_2\right)^2$ with $s \in \mathbb{R}$ such that $P_{t_j} \preceq s\,I$ and $e_i := \gamma_{n+1}\,|R_{r\,i,\cdot}|\,[|x|^T\ \ 1]^T + 2\,(n + 2 + \|x\|_1)\,\texttt{eta}$. Again, such a $b'_{v',j}$ is easy to compute (with a SDP solver for s and rounding toward $+\infty$ for e).

THEOREM 2. *Given matrices* $P, R \in \mathbb{R}^{(n+1)\times(n+1)}$, *with* $2(n+2)\texttt{eps} < 1$, *and scalars* $s, b \in \mathbb{R}$ *such that* P *is symmetric positive semi-definite (i.e.,* $P^T = P$ *and* $P \succeq 0$*) and* $P \preceq s\,I$, *for any* $x \in \mathbb{F}^n$, *denoting* $e_i := \gamma_{n+2}\,|R_{i,\cdot}|\,[\,|x|^T\ \ 1]^T + 2\,(n + 2 + \|x\|_1)\,\texttt{eta}$, *if* $s\|e\|_2^2 \le b$ *and*

$$\begin{bmatrix} x \\ 1 \end{bmatrix}^T R^T P\, R \begin{bmatrix} x \\ 1 \end{bmatrix} \le \left(\sqrt{b} - \sqrt{s}\|e\|_2\right)^2$$

then

$$\mathrm{fl}\left(\begin{bmatrix} x \\ 1 \end{bmatrix}^T R^T\right) P\,\mathrm{fl}\left(R\begin{bmatrix} x \\ 1 \end{bmatrix}\right) \le b$$

where $R_{i,\cdot}$ *denotes the* i*-th row of the matrix* R.

Finally, if the following holds

$$b'_{v',j} \ge \max\left\{ r(t_j)\ \middle|\ e \le c' \wedge \bigwedge_{j'} (t_{j'} \le b_{v,j'}) \right\} \tag{16}$$

then (15) holds and we can now proceed as in Section 6.1 by just replacing (14) with the above (16). The check should still succeed since the differences between the values in (14) and (16) are orders of magnitude smaller than the accuracy the $b_{v,j}$ were initially computed with using SDP solvers.

A possible improvement would be to avoid over-approximating the computations of the model of the plant since they do not need to be performed with floating-point values like the computations of the controller.

[6]Again, the assignment r is actually computed with floating-point arithmetic whereas $r(t_j) \le b_{v',j}$ is a purely mathematical property. That's why we don't need $\mathrm{fl}\left([x\ 1]R_r^T P_{t_j} R_r [x\ 1]^T\right) \le b_{v',j}$.

REMARK 10 (ABOUT FIXED-POINT ARITHMETIC). *In some sense, floating-point arithmetic can seem harder to analyze than fixed-point arithmetic due to the relative error (δ terms). However, the fact that floating-point usually only induce tiny errors, makes the* proof search[7] then proof checking *scheme, used in this section, practical. Fixed-point arithmetic may introduce larger errors which may require the use of more involved proof search techniques, actually taking rounding errors into account [24, 27].*

7. CONCLUSION AND PERSPECTIVES

The presented analyses extended previous work of the authors focused on the software part of a controlled system. In the present work addressing the complete system, the choice has been made to follow the usual approach of control engineers and consider the behavior of the controller with respect to a linear approximation of the plant, i.e., a linearized plant. It differs from the vast state of the art of the analysis of hybrid systems.

However this simplified setting, where the non linearity of the plant is hidden, remains an interesting challenge. First it is meaningful with respect to the design process of the controller, it is important to ensure that the stability properties targeted at design level remain valid at the implementation level. Second the proposed analysis achieves an over-approximation of all reachable states while arguing about a control level property, closed-loop stability of the controlled system. Third, it is able to handle more than linear systems, e.g., including saturations or constructs leading to non globally stable systems. It also opens the possibility to evaluate a control law within its safety architecture.

We are currently completing the implementation[8] of the template generation method of Section 4.2.2 in our static analyzer. Everything is available at `http://cavale.enseeiht.fr/closedloop2014/`.

To the best of authors' knowledge, none of the existing static analysis tools performs a computation by taking the plant into account and computing an over-approximation of all reachable states. Furthermore our proposal also considers the mix of floating-point computations at the software level with real computations for the plant part. The extension of previous work on stability of the controller to the complete system also enlightened the need for new Lyapunov-based heuristics to deal with saturations.

In terms of future work, the presented analysis would have to be applied on more examples of control systems as shared by our academic and industrial partners (e.g., control command of aircraft, full-authority digital engine control (FADEC), etc). It would also be interesting to add sensor and actuation noise to the model of the plant[9]. Another exciting outcome of this work is the now open possibility to consider control level properties when dealing with the actual code. Our next goal would address the analysis of robustness and performances properties with a similar approach.

8. REFERENCES

[1] S. Boldo and G. Melquiond. Flocq: A Unified Library for Proving Floating-point Algorithms in Coq. In

[7]Completely ignoring rounding errors.

[8]Currently only prototyped as a Scilab script.

[9]Handling sensor noise is very easy but actuation noise seems a bit less trivial.

Proceedings of the 20th IEEE Symposium on Computer Arithmetic, pages 243–252, Tübingen, Germany, July 2011.

[2] O. Bouissou, E. Goubault, S. Putot, K. Tekkal, and F. Védrine. Hybridfluctuat: A static analyzer of numerical programs within a continuous environment. In A. Bouajjani and O. Maler, editors, *Computer Aided Verification, 21st International Conference, CAV 2009, Grenoble, France, June 26 - July 2, 2009. Proceedings*, volume 5643 of *Lecture Notes in Computer Science*, pages 620–626. Springer, 2009.

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

[4] T. Coq development team. *The Coq proof assistant reference manual*, 2012. Version 8.4.

[5] J. Feret. Static Analysis of Digital Filters. In *ESOP*, number 2986, 2004.

[6] E. Féron. From Control Systems to Control Software. *Control Systems, IEEE*, 30(6), dec. 2010.

[7] G. F. Franklin, M. L. Workman, and D. Powell. *Digital Control of Dynamic Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1990.

[8] T. M. Gawlitza, H. Seidl, A. Adjé, S. Gaubert, and E. Goubault. Abstract interpretation meets convex optimization. *J. Symb. Comput.*, 47(12), 2012.

[9] H. Herencia-Zapana, R. Jobredeaux, S. Owre, P.-L. Garoche, E. Feron, G. Perez, and P. Ascariz. Pvs linear algebra libraries for verification of control software algorithms in c/acsl. In *NASA Formal Methods*, pages 147–161, 2012.

[10] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

[11] IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. *IEEE Standard 754-2008*, 2008.

[12] W. S. Levine. *The control handbook*. The electrical engineering handbook series. CRC Press New York, Boca Raton (Fl.), 1996.

[13] A. M. Lyapunov. Problème général de la stabilité du mouvement. *Annals of Mathematics Studies*, 17, 1947.

[14] E. Möhlmann and O. E. Theel. Stabhyli: a tool for automatic stability verification of non-linear hybrid systems. In C. Belta and F. Ivancic, editors, *Proceedings of the 16th international conference on Hybrid systems: computation and control, HSCC 2013, April 8-11, 2013, Philadelphia, PA, USA*, pages 107–112. ACM, 2013.

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

[16] A. Podelski and S. Wagner. Region stability proofs for hybrid systems. In J. Raskin and P. S. Thiagarajan, editors, *Formal Modeling and Analysis of Timed Systems, 5th International Conference, FORMATS 2007, Salzburg, Austria, October 3-5, 2007, Proceedings*, volume 4763 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 2007.

[17] P. Prabhakar and M. G. Soto. Abstraction based model-checking of stability of hybrid systems. In N. Sharygina and H. Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 280–295. Springer, 2013.

[18] S. Ratschan and Z. She. Providing a basin of attraction to a target region of polynomial systems by computation of lyapunov-like functions. *SIAM J. Control and Optimization*, 48(7):4377–4394, 2010.

[19] P. Roux and P. Garoche. Computing quadratic invariants with min- and max-policy iterations: A practical comparison. In C. B. Jones, P. Pihlajasaari, and J. Sun, editors, *FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings*, volume 8442 of *Lecture Notes in Computer Science*, pages 563–578. Springer, 2014.

[20] P. Roux and P.-L. Garoche. Integrating policy iterations in abstract interpreters. In D. V. Hung and M. Ogawa, editors, *ATVA*, volume 8172 of *Lecture Notes in Computer Science*, pages 240–254. Springer, 2013.

[21] P. Roux, R. Jobredeaux, P.-L. Garoche, and E. Féron. A generic ellipsoid abstract domain for linear time invariant systems. In *HSCC*. ACM, 2012.

[22] S. M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, May 2010.

[23] S. Sankaranarayanan and A. Tiwari. Relational abstractions for continuous and hybrid systems. In G. Gopalakrishnan and S. Qadeer, editors, *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, volume 6806 of *Lecture Notes in Computer Science*, pages 686–702. Springer, 2011.

[24] A. Suardi, S. Longo, E. C. Kerrigan, and G. A. Constantinides. Robust explicit mpc design under finite precision arithmetic. In *World Congress of the International Federation of Automatic Control*, volume 19, pages 2939–2944, 2014.

[25] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996.

[26] T. Wang, R. Jobredeaux, H. Herencia-Zapana, P.-L. Garoche, A. Dieumegard, E. Feron, and M. Pantel. From design to implementation: an automated, credible autocoding chain for control systems. *CoRR*, abs/1307.2641, 2013.

[27] Y. Xia, J. Yan, P. Shi, and M. Fu. Stability analysis of discrete-time systems with quantized feedback and measurements. *Industrial Informatics, IEEE Transactions on*, 9(1):313–324, Feb 2013.

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

APPENDIX

A. PROOFS

This appendix contains proofs not included in the paper.

LEMMA 1. *Let $C \in \mathbb{R}^{1\times n}$ be a non-zero row vector, $P \in \mathbb{R}^{n\times n}$ a positive definite matrix, and $x \in \mathbb{R}^{n\times 1}$ a column vector. Assume $x^{\mathrm{T}}Px \leq 1$. Then*

$$|Cx| \leq \sqrt{CP^{-1}C^{\mathrm{T}}}$$

PROOF LEMMA 1. To look for an upper and lower bound on Cx given the ellipsoid constraint, consider the following extremization problem:

$$\begin{aligned} &\underset{x}{\text{extremize}} && Cx \\ &\text{subject to} && x^{\mathrm{T}}Px - 1 \leq 0 \end{aligned}$$

Introducing the Lagrangian $\mathcal{L} = Cx - \lambda(x^{\mathrm{T}}Px - 1)$, an extremal solution x^* must satisfy $\frac{\partial \mathcal{L}}{\partial x} = 0$, that is:

$$C - \lambda {x^*}^{\mathrm{T}} P = 0 \tag{17}$$

Multiply (17) by x^* to the right and solve for λ:

$$\lambda = \frac{Cx^*}{{x^*}^{\mathrm{T}}Px^*}$$

(Note that $x^* \neq 0$,otherwise (17) would yield $C = 0$). Then, transposing (17) and multiplying by CP^{-1} on the left:

$$\begin{aligned} &CP^{-1}C^{\mathrm{T}} - \lambda Cx^* = 0 \\ \iff &CP^{-1}C^{\mathrm{T}} - (Cx^*)(Cx^*) = 0 \\ \iff &Cx^* = \pm\sqrt{CP^{-1}C^{\mathrm{T}}} \end{aligned}$$

The two solutions yield the required upper and lower bound on Cx. □

To prove Theorem 1, we first need the following lemma.

LEMMA 2. *Assuming $(n-1)\texttt{eps} < 1$, for all $x \in \mathbb{F}^n$, there exists $\theta \in \mathbb{R}^n$ such that for all i, $|\theta_i| \leq \gamma_{n-1}$ and*

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n}(1+\theta_i)x_i.$$

PROOF LEMMA 2. According to Definition 2, if the sum is computed from left to right, there exists $\delta_{n-1} \in \mathbb{R}$ such that $|\delta_{n-1}| \leq \texttt{eps}$ and

$$\begin{aligned} \mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) &= \mathrm{fl}\left(\mathrm{fl}\left(\sum_{i=1}^{n-1} x_i\right) + x_n\right) \\ &= (1+\delta_{n-1})\left(\mathrm{fl}\left(\sum_{i=1}^{n-1} x_i\right) + x_n\right). \end{aligned}$$

Then, by an immediate induction, there exists $\delta \in \mathbb{R}^{n-1}$ such that for all i, $|\delta_i| \leq \texttt{eps}$ and

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = \left(\prod_{j=1}^{n-1}(1+\delta_j)\right)x_1 + \sum_{i=2}^{n}\left(\left(\prod_{j=i-1}^{n-1}(1+\delta_j)\right)x_i\right).$$

According to classic results [10, Lemma 3.3] about the terms $\gamma_k := \frac{k\texttt{eps}}{1-k\texttt{eps}}$, for all i, there exists $\theta_i \in \mathbb{R}$ such that $|\theta_i| \leq \gamma_{n-i+1}$ and $\prod_{j=i-1}^{n-1}(1+\delta_j) = 1+\theta_i$, hence the result[10]. □

[10]A similar proof can be performed if the sum is not computed in this left-right order.

PROOF THEOREM 1. According to Lemma 2, there exists $\theta \in \mathbb{R}^n$ such that for all i, $|\theta_i| \leq \gamma_{n-1}$ and

$$\mathrm{fl}\left(\sum_{i=1}^{n} a_i x_i\right) = \sum_{i=1}^{n}(1+\theta_i)\mathrm{fl}(a_i x_i).$$

Then, according to Definition 2, there exist $\delta, \eta \in \mathbb{R}^n$ such that for all i, $|\delta_i| \leq \texttt{eps}$, $|\eta_i| \leq \texttt{eta}$ and

$$\mathrm{fl}\left(\sum_{i=1}^{n} a_i x_i\right) = \sum_{i=1}^{n}(1+\theta_i)\left((1+\delta_i)\mathrm{fl}(a_i)\,\mathrm{fl}(x_i) + \eta_i\right).$$

Since $x_i \in \mathbb{F}$, $\mathrm{fl}(x_i) = x_i$ but $a_i \in \mathbb{R}$ hence $\mathrm{fl}(a_i) = (1+\delta_i')a_i + \eta_i'$ for some $\delta_i', \eta_i' \in \mathbb{R}$, $|\delta_i'| \leq \texttt{eps}$ and $|\eta_i'| \leq \texttt{eta}$. Hence

$$\begin{aligned} \mathrm{fl}\left(\sum_{i=1}^{n} a_i x_i\right) &= \sum_{i=1}^{n}(1+\theta_i)(1+\delta_i)(1+\delta_i')a_i x_i \\ &\quad + (1+\theta_i)(1+\delta_i)\eta_i' x_i + (1+\theta_i)\eta_i. \end{aligned}$$

According to classic results [10, Lemma 3.3] about the terms γ_k, for all i, there exists $\theta_i' \in \mathbb{R}$ such that $|\theta_i'| \leq \gamma_{n+1}$ and $(1+\theta_i)(1+\delta_i)(1+\delta_i') = 1+\theta_i'$. Similarly, there exists $\theta_i'' \in \mathbb{R}$ such that $|\theta_i''| \leq \gamma_n$ and $(1+\theta_i)(1+\delta_i) = (1+\theta_i'')$, which gives

$$\mathrm{fl}\left(\sum_{i=1}^{n} a_i x_i\right) = \sum_{i=1}^{n}\left((1+\theta_i')a_i x_i + (1+\theta_i'')\eta_i' x_i + (1+\theta_i)\eta_i\right).$$

Then

$$\begin{aligned} \mathrm{fl}\left(\sum_{i=1}^{n} a_i x_i\right) - \sum_{i=1}^{n} a_i x_i &= \sum_{i=1}^{n} \theta_i' a_i x_i \\ &\quad + \sum_{i=1}^{n}\left((1+\theta_i'')\eta_i' x_i + (1+\theta_i)\eta_i\right). \end{aligned}$$

We can notice that

$$\left|\sum_{i=1}^{n} \theta_i' a_i x_i\right| \leq \sum_{i=1}^{n} |\theta_i'|\,|a_i x_i| \leq \sum_{i=1}^{n} \gamma_{n+1}\,|a_i x_i| = \gamma_{n+1}\sum_{i=1}^{n}|a_i x_i|$$

and similarly

$$\left|\sum_{i=1}^{n}\left((1+\theta_i'')\eta_i' x_i + (1+\theta_i)\eta_i\right)\right| \leq 2\left(n + \sum_{i=1}^{n}|x_i|\right)\texttt{eta}$$

since $|\theta_i''| \leq \gamma_n \leq 1$ and $|\theta_i| \leq \gamma_{n-1} \leq 1$, which finally gives the result. □

PROOF THEOREM 2. Denoting $y := R[x\ \ 1]^T$ we have, thanks to Theorem 1, $|\mathrm{fl}(y_i) - y_i| \leq e_i$, hence $\mathrm{fl}(y)_i = y_i + \delta_i e_i$ for some $\delta_i \in \mathbb{R}$ such that $|\delta_i| \leq 1$. Thus, denoting D the diagonal matrix such that for all i, $D_{i,i} = \delta_i$, we have

$$\begin{aligned} \mathrm{fl}(y)^T\,P\,\mathrm{fl}(y) &= (y + De)^T\,P\,(y + De) \\ &= y^T\,P\,y + e^T\,D^T\,P\,D\,e + 2y^T\,P\,D\,e. \end{aligned}$$

Then, by Cauchy-Schwarz inequality

$$\mathrm{fl}(y)^T\,P\,\mathrm{fl}(y) \leq y^T\,P\,y + e^T\,D^T\,P\,D\,e + 2\sqrt{y^T\,P\,y}\sqrt{e^T\,D^T\,P\,D\,e}$$

and since $P \preceq s\,I$

$$\mathrm{fl}(y)^T\,P\,\mathrm{fl}(y) \leq y^T\,P\,y + s\|e\|_2^2 + 2\sqrt{y^T\,P\,y}\sqrt{s}\|e\|_2.$$

Hence the result, since $y^T\,P\,y \leq \left(\sqrt{b} - \sqrt{s}\|e\|_2\right)^2$. □