Exploiting the Eigenstructure of Linear Systems to speed up Reachability Computations

Alexandre Rocca[1,2], Thao Dang[1], and Eric Fanchon[2]

[1] VERIMAG/CNRS
2, avenue de Vignate 38610 GIERE, France
[2] UJF-Grenoble 1/CNRS, TIMC-IMAG,
UMR 5525, Grenoble, F-38041, France

Abstract. Reachability analysis has recently proved to be a useful technique for analysing the behaviour of under-specified biological models. In this paper, we propose a method exploiting the eigenstructure of a linear continuous system to efficiently estimate a bounded interval containing the time at which the system can reach a target set from an initial set. Then this estimation can be directly integrated in an existing algorithm for hybrid systems with linear continuous dynamics, to speed up reachability computations. Furthermore, it can also be used to improve time-efficiency of the hybridization technique that is based on a piecewise-linear approximation of non-linear continuous dynamics. The proposed method is illustrated on a number of examples including a biological model.

Keywords: reachability analysis, linear systems, biological systems

1 Introduction

Linear differential systems of the form $\dot{x}(t) = Ax(t)$, where A is a $n \times n$ matrix with real coefficients, constitute an important class of differential systems for which symbolic solutions are known. They have the form $x(t) = \exp(At)x_0$, where x_0 is an initial condition. An option is to compute numerically the matrix exponential at each time step. Another option is to write down explicitly the analytical expressions of the components $x_i(t)$ in terms of the eigenvalues and eigenvectors of A. In this paper we present an approach to take advantage of the eigenstructure of the matrix A to speed up reachability computations of linear systems. Furthemore, it can be applied to improve the time-efficiency of the dynamic hybridization of nonlinear systems [4].

The general idea is to use the analytical expressions of $x(t)$ to estimate the time intervals over which it is certain that the linear system from a given initial set does not reach a given fixed set. Knowing in advance that no such collision is possible over these time intervals allows avoiding some computations over these intervals, for example the intersection of the reachable set and some guard set, or even accurate computations of flowpipes (sets of trajectories). The intersection computation cost growing very fast with the number of dimensions, we need a method to avoid those computations for complex problems. If reachability analysis is greatly used for cyber-physical applications, it is less the case for biological applications because of the complexity of most of the biological systems. However, with improvements to speed it up, reachability analysis will become a powerful tool to check properties, and evaluate the robustness of biological models.

The rest of the paper is organized in two main parts. We begin the first part by presenting some preliminaries and the algorithm to estimate a set of time intervals, called Reachability Time Domain (RTD). Some experimental results are then described. Then we adapt the method of estimating RTD to speed up the dynamic hybridization of nonlinear systems. The adaptation is applied to a biological model, which shows the usefulness of the method in terms of gain in computation time. In the last section we describe related works which also exploit the eigenstructure of linear systems, and outline some directions for future work.

2 Reachability Time Domain Estimation

2.1 Preliminaries

In this section we consider a linear differential system:

$$\dot{x}(t) = Ax(t) \tag{1}$$

where A is an $n \times n$ matrix with real coefficients, and $x \in \mathbb{R}^n$. We assume that the matrix A is diagonalisable in $\mathbb{C}$ or, in other words, matrix A has n distinct eigenvalues $\Lambda = \{\lambda_1, \dots, \lambda_n\}$, and n associated eigenvectors $V = \{v_1, \dots, v_n\}$. If some of the eigenvalues are complex, then they occur in complex conjugate pairs $(\lambda, \bar{\lambda})$. We consider that there are r real eigenvalues, and c pairs of complex conjugate eigenvalues, such that $\Lambda = \{\lambda_1, \dots \lambda_r, \lambda_{r+1}, \bar{\lambda}_{r+1}, \dots, \lambda_{r+c}, \bar{\lambda}_{r+c}\}$ (the associated real and complex eigenvectors are indexed accordingly). Obviously, $n = r + 2c$.

In this situation (distinct eigenvalues), a basic theorem of linear algebra states that the matrix A can be put in block-diagonal form with blocks not bigger than 2×2. More formally, $(v_1, \dots, v_r, Im(v_{r+1}), Re(v_{r+1}), \dots, Im(v_{r+c}), Re(v_{r+c}))$ is a basis of $\mathbb{R}^n$ (by abuse of language we will call it the eigenbasis of A), the matrix

$$P = \begin{bmatrix} v_1 \dots v_r \; Im(v_{r+1}) \; Re(v_{r+1}) \dots Im(v_{r+c}) \; Re(v_{r+c}) \end{bmatrix} \tag{2}$$

is invertible, and $P^{-1}AP = diag[\lambda_1, \dots, \lambda_r, B_{r+1}, \dots, B_{r+c}]$, where B_j is a 2×2 real matrix

$$B_j = \begin{bmatrix} Re(\lambda_j) & -Im(\lambda_j) \\ Im(\lambda_j) & Re(\lambda_j) \end{bmatrix}. \tag{3}$$

The notation $diag[b_j]$ stands for a block-diagonal matrix with the elements b_j (b_j is either a real scalar, or a real 2×2 matrix) on the diagonal.

Now consider two convex H-polytopes [3] $Init_o$ and R_o in $\mathbb{R}^n$. In the following section, $Init_o$ represents the set of initial conditions and R_o the target set. The

[3] H-polytopes are polytopes defined by a set of linear constraints.

reachability problem we address now can be formulated as the following question: does the system (1), from the initial set $Init_o$, ever reach the target set R_o? If the answer to this question is yes, then the first time the system enters the target set is denoted t_{reach} (see Definition 2 below).

Since we want to exploit the analytical solutions of the system (1), from now on we work in the eigenbasis. This means that the two polytopes have to be transformed as follows: $Init = P^{-1}Init_o$ and $R = P^{-1}R_o$ ($Init$ is the set of initial conditions, and R the target set, expressed in the eigenbasis). The Fundamental Theorem for Linear Systems [12] states that for $x_0 \in \mathbb{R}^n$ the initial value problem for the equation (1) and $x(0) = x_0$ has a unique solution for all $t \in \mathbb{R}$ which is given by $x(t) = e^{At}x(0)$.

For $t \geq 0$, let $E(t) = \{e^{At}x(0) \mid x(0) \in Init\}$. With this notation: $E(0) = Init$, and $E(t) = e^{At}E(0)$. From the computational point of view, the computation of $x(t)$ can be reduced to the computation of the exponential of a matrix, and to do so numerous algorithms are known [3]. In addition it is known that the image of a convex polytope by a linear operator is a convex polytope.

Definition 1. *We define the reach time interval $T_{overlap}$ as the set of times t for which $E(t)$ intersects with R (under the condition that such an intersection occurs, otherwise $T_{overlap} = \emptyset$).*

$$T_{overlap} = \{t \mid E(t) \bigcap R \neq \emptyset\} \tag{4}$$

Definition 2. *If $T_{overlap}$ is not empty, we define the reachability time $t_{reach} \in \mathbb{R}^+$ as the first instant t for which $E(t)$ intersects R.*

$$t_{reach} = min\{T_{overlap}\} \tag{5}$$

Definition 3. *Let T be a union of disjoint time intervals. T is said to be a* Reachability Time Domain (RTD) *if t_{reach} (when it exists) does not belong to the complement of T, then T satisfies: $T_{overlap} \neq \emptyset \implies t_{reach} \in T$. Obviously, the largest RTD in all cases is $\mathbb{R}^+$, and the smallest is $\{t_{reach}\}$ when R is reachable from $Init$. Note that $T_{overlap}$ is also an RTD.*

We can now restate informally our goal as follows: we want a fast algorithm to compute a useful RTD T. It would be for example useless to give $\mathbb{R}^+$ as an answer. On the other hand, one could design an algorithm which computes t_{reach} directly by using reachability computation, and of course this is not what we intend to do here. The idea is to perform fast computations to determine an RTD T. Since by construction $E(t)$ cannot intersect R on the complement of T, then it is possible to avoid the test whether $E(t)$ intersects R for all time t in the complement of T. Thus the computation of T is rewarded by avoiding heavier computations.

2.2 Algorithm for Reachability Time Domain Estimation

We take advantage of the fact that, as mentioned above, the matrix $P^{-1}AP$ is block-diagonal in the eigenbasis, a block being just a scalar (in the case of a real eigenvalue) or a 2×2 submatrix (in the case of a pair of complex eigenvalues). This means that the system (1) can be decomposed into smaller subsystems of 1 or at most 2 variables. Remember that we assume in this work that all eigenvalues are distinct. The principle of our method is to use the analytic expressions of the solutions, expressed in the eigenbasis, and to make simple over-approximations of the convex polytopes $E(t)$ and R in order to work on 1-dimensional or 2-dimensional projections.

The algorithm is divided in three parts: first, the exploitation of the real eigenvalues; second, the radial motion associated to the complex eigenvalues; third, the rotation motion associated to the complex eigenvalues. Since the differential system is decoupled when expressed in the eigenbasis, the time information extracted from the projections are independent one from the other. One could thus choose to exploit only the information associated with the real eigenvalues (assuming there is at least one). This would provide an approximation of RTD. But of course exploiting also the information associated with the complex eigenvalues provides additional constraints and generally leads to a smaller RTD.

Recall that λ_i for $i \in \{1, \ldots, r\}$ are the real eigenvalues of A, and that $(\lambda_i, \bar{\lambda}_i)$, $i \in \{r+1, \ldots, r+c\}$ are pairs of conjugate eigenvalues.

Part 1. We first extract information from the real eigenvalues. The case of complex eigenvalues (presented in the next two parts) is a generalization of the basic idea presented here.

We consider each real eigenvalue λ_i, and its associated eigenvector v_i. The analytic solution along this axis is: $y_i(t) = e^{\lambda_i t} y_i(0)$. Now we define $proj(P, i)$ as the function that gives the projection of the polytope P on the i^{th} real eigenvalue subspace (subtended by v_i), and we call T_i the time interval during which the intervals $proj(E(t), i)$ and $proj(R, i)$ overlap. The time interval T_i is defined formally by:

$$T_i = \{t \mid (e^{\lambda_i t} proj(Init, i)) \bigcap proj(R, i) \neq \emptyset\} \tag{6}$$

The bounds t_i^{min} and t_i^{max} of T_i ($i \in \{1, \ldots, r\}$) are easily computable as we will see shortly. If R is reachable from $Init$ then it is clear that the time of the first encounter t_{reach} belongs to all T_i (because the point of contact between the two polytopes belongs to all the projections).

We define accordingly T^{real} as the intersection of all the time intervals associated with real eigenvalues:

$$T^{real} = \bigcap_{0 \leq i \leq r} T_i \tag{7}$$

From what we have just said, T^{real} is an RTD. Let us call $outer(X)$ the smallest box containing the polytope X. Note that, from its definition, T^{real} is bounded if at least one T_i is bounded. Note also that even if there is a point of contact between $outer(E(t))$ and $outer(R)$ at some time t, we cannot conclude that R is reachable from $Init$, since working on projections amounts to over-approximating the polytopes by boxes (in the subspace subtended by the real eigenvectors). In other words, if T^{real} is not empty, we cannot be sure that R is reachable from $Init$. But we can be sure that if R is reachable, then t_{reach} cannot be outside T^{real}. This is true independently of the existence of complex eigenvalues. In addition, if a T_i is empty then we can conclude immediately that R is unreachable.

Now concerning the computation of the bounds t_i^{min} and t_i^{max} of T_i, we consider a point $y_i(0)$ belonging to $proj(Init, i)$. If the configuration is such that $y_i(t)$ moves toward R, and the origin 0 does not lie between $y_i(0)$ and $proj(R, i)$ then it is trivial to compute the time at which $y_i(0)$ will reach $proj(R, i)$. As an example, we consider the following case: $\lambda_i < 0$ (the trajectories in this 1-dimensional subspace converge to 0), we suppose that $proj(R, i) = [z_{i,min}, z_{i,max}]$ is strictly above 0, and $y_i(0) > z_{i,max}$. Then the entry time of this point is given by: $t_i^{min} = (1/\lambda_i)\ln(z_{i,max}/y_i(0))$, and the exit time by: $t_i^{max} = (1/\lambda_i)\ln(z_{i,min}/y_i(0))$. The logarithm is negative and consequently the computed times are positive, as expected. The key property here is the monotonicity of the function $e^{\lambda_i t}$. This is just an example and a number of cases must be considered depending on: the sign of λ_i, the relative position of $proj(E(t), i)$ and $proj(R, i)$, and the position of the origin with respect to these intervals. Depending on the case, T_i may be empty (meaning that R is unreachable and thus the problem is solved); it may be bounded as in the above example; or it may be semi-infinite ($[t_i^{min}, -\infty]$). The lower and upper bounds of T^{real} are: $t^{lb} = \max_i\{t_i^{min}\}$ and $t^{ub} = \min_i\{t_i^{max}\}$ (if at least one t_i^{max} is finite).

Consider now the case where the origin 0 belongs to the box over-approximation $outer(R)$ of the target set R. If there is a real eigenvalue λ_i which is negative, then the points of $(e^{\lambda_i t} proj(Init, i))$ never exit $proj(R, i)$ after entering in it, and consequently t_i^{max} is infinite. We would like to obtain a *bounded* interval which is an RTD. If at least one real eigenvalue λ_i is positive (and 0 does not belong to $outer(Init)$), then t^{ub} as defined above is finite. If all the real eigenvalues are negative more work is required to get a bounded RTD. Two subcases need to be considered when all the real eigenvalues are negative. Either 0 belongs to R, or 0 belongs to $outer(R)$ but not to R itself (we assume here that 0 does not belong to the boundary of R). In the first subcase we define a box containing 0 and contained in R, which we call $inner1(R)$. We then apply the same method as above, just replacing the outer box by the inner box $inner1(R)$: t_i^{inner1} is defined as the time at which $proj(E(t), i)$ makes the first contact with $proj(inner1(R), i)$, and $t^{inner1} = \max_i\{t_i^{inner1}\}$. If $t \geq t^{inner1}$ then at least one point of the moving polytope $E(t)$ has entered the inner box $inner1(R)$. Since it is included in R this point is necessarily inside R. This time t^{inner1} thus occurs necessarily after t_{reach}, and can thus be taken as an upper bound for t_{reach}: $t^{ub} = t^{inner1}$. In the second subcase, where 0 belongs to $outer(R) \setminus R$, we define a box containing 0, contained in $outer(R)$, and disjoint from R. We call $inner2(R)$ a box having these properties. The time t_i^{incl} is defined as the time at which $proj(E(t), i)$ is completely included in $proj(inner2(R), i)$, and globally $t^{incl} = \max_i\{t_i^{incl}\}$. If $t \geq t^{incl}$ then the moving polytope $E(t)$ is completely included in the inner box $inner2(R)$, and due to the monotonicity property, it will always remain in it. The box $\mathit{inner}2(R)$ being disjoint from R, R cannot be reached after t^{incl}. Consequently t_{reach}, if it exists, is necessarily smaller than t^{incl}. We conclude that t^{incl} can be taken as an upper bound for t_{reach}: $t^{ub} = t^{incl}$.

Similar reasoning can be made if 0 belongs to Init (or to $\mathit{outer}(\mathit{Init}) \setminus \mathit{Init}$) and all the real eigenvalues are positive (case where all the t_i^{max}'s are infinite). The cases are too numerous to give the details here, but in the end it is only under very special conditions that the RTD resulting from the presented method is unbounded.There are basically two classes of conditions for which the above method may not provide a bounded RTD: (i) there exists only one real eigenvalue λ_i and it is equal to 0 (the corresponding component y_i is constant); (ii) there is a projection i such that 0 is at an extremity of $\mathit{proj}(R, i)$ and λ_i is stricly negative (or 0 is at an extremity of $\mathit{proj}(\mathit{Init}, i)$ and λ_i is stricly positive).

The core of this part of the algorithm is straightforward: first compute $\mathit{outer}(R)$; if 0 does not belong to $\mathit{outer}(R)$, then perform the following loop for $i \in \{1, \dots, r\}$:

- compute t_i^{min} and t_i^{max};
- keep the value of this t_i^{min} if it is greater than the current stored value;
- keep the value of this t_i^{max} if it is smaller than the current stored value.

If 0 belongs to R (resp. if 0 belongs to $\mathit{outer}(R) \setminus R$) and if all the real eigenvalues are negative, then compute an inner box $\mathit{inner}1(R)$ (resp. $\mathit{inner}2(R)$). Then perform a similar loop in which t^{inner1} (or t^{incl} depending on the case) is computed instead of t_i^{max}, and the maximum value is retained at each step.

Part 2. Now we consider pairs of complex conjugate eigenvalues $(\lambda_j, \overline{\lambda}_j)$. Each such pair is associated to a 2×2 submatrix A_j. A trajectory defined by this matrix (and an initial condition) in the corresponding eigenplane is a spiral, or a circle if $Re(\lambda_j) = 0$, and can be decomposed into a radial and an angular component. To exploit this decomposition we use polar coordinates and we over-approximate the sets $\mathit{proj}(R, j)$ and $\mathit{proj}(I, j)$ by sectors (interval description in a polar system). In this second part we extract time information from the radial evolution of (the projection of) moving set.

The polar coordinates of a point x in the eigenplane associated to $(\lambda_j, \overline{\lambda}_j)$ are noted (γ, θ). E being a polytope in $\mathbb{R}^n$, we define the radial part of $\mathit{proj}(E, j)$ by:

$$\Gamma_j(E) = \{\gamma(x) \mid x \in \mathit{proj}(E, j)\} \tag{8}$$

The sets $\Gamma_j(\mathit{Init})$ and $\Gamma_j(R)$ are intervals and we apply the same method as in Part 1. We compute for each pair $j \in \{r+1, \dots, r+c\}$ of complex conjugate eigenvalues the time interval T_j where the sector approximation of $\mathit{proj}(R, j)$ is reached following the radial decomposition of the motion. If there exists a pair of eigenvalues j, such that T_j is empty then, R is unreachable. Else, we compute T^{rad} the intersection of all the T_j for $j \in \{r+1, \dots, r+c\}$. Again, if T^{rad} is empty then R is unreachable.

$$T_j = \{t \mid (e^{Re(\lambda_j)t} \Gamma_j(Init)) \bigcap \Gamma_j(R) \neq \emptyset\} \tag{9}$$

$$T^{rad} = \bigcap_{r+1 \leq j \leq r+c} T_j \tag{10}$$

The upper bound of the interval T^{rad} can be infinite. The conditions under which this occurs are similar to those of Part 1. If the real part of all the complex eigenvalues is equal to zero, then the point trajectories lie on a product of circles (the radii depend on the initial conditions and are constant). If in addition the intersection of this set with R is non empty, then the upper bound of T^{rad} is infinite.

It is clear that the set $T^{real \cap rad}$ defined as the intersection of T^{rad} and T^{real} is an RTD. If $T^{real \cap rad}$ is empty, then R is unreachable.

$$T^{real \cap rad} = T^{rad} \bigcap T^{real} \tag{11}$$

The computation of T^{rad} is similar to that of T^{real} in Part 1.

Part 3. In this last part, we extract time information from the angular motion of the reachable set. For each complex eigenvalue pair $j \in \{r+1, \ldots, r+c\}$ we define, $\theta_j(E(t))$ the angular representation of the projection of the polytope $E(t)$ on the complex plane (a circular arc). Then we compute T_j^{ang} the union of time intervals representing all the instant t for which

$$\theta_j(E(t)) \bigcap \theta_j(R) \neq \emptyset.$$

Because of the periodicity of the angular motion, we describe T_j^{ang} by the first interval and the period π_j.

$$T_j^{ang} = \{\ t \mid \theta_j(R) \ \bigcap\ \{e^{Bt}x_0 \mid x_0 \in \theta_j(Init)\} \neq \emptyset\} \tag{12}$$

where

$$B = \begin{bmatrix} Re(\lambda_j) & -Im(\lambda_j) \\ Im(\lambda_j) & Re(\lambda_j) \end{bmatrix}.$$

The theoretical output is the intersection of all these unions of time intervals and $T^{real \cap rad}$:

$$T^{ang} = \bigcap_{r+1 \leq j \leq r+c} T_j^{ang} \tag{13}$$

Combining all the information, the final output is:

$$T^{final} = T^{real \cap rad} \bigcap T^{ang} \tag{14}$$

In practice, the intersection to compute T^{ang} is done on the fly. It is possible, mathematically, that the periods π_j are not commensurable. In such a case, the trajectories are quasiperiodic, and T^{ang} is an infinite union of intervals. The implementation handles only floating-point numbers and consequently this case is not considered.

We can thus compute the lower common multiple of all the periods, which will be the global period Π of the system (note that Π can be very large). Then, even if $T^{real \cap rad}$ is not bounded (which is a very special case), the computation of T^{ang} is finite in time, and T^{final} can be easily represented as a finite union of time intervals, and the period Π.

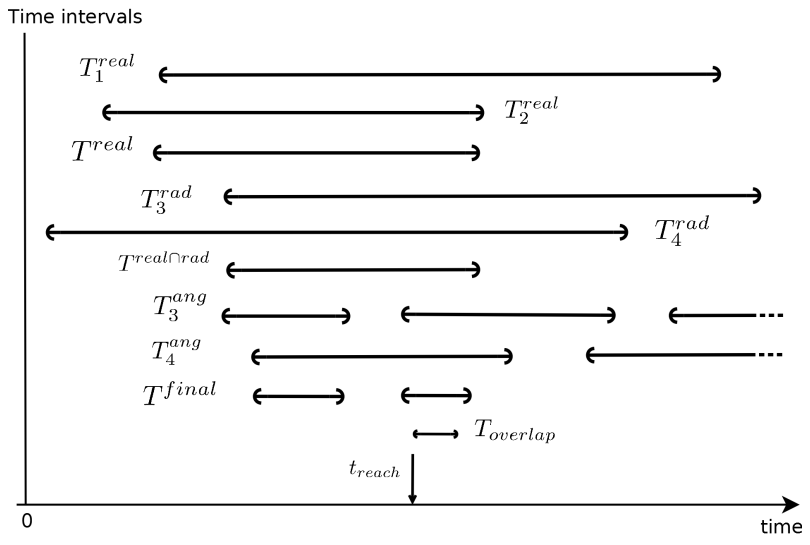

Fig. 1. This figure shows the different steps to construct the T^{final} union of intervals for a 6-dimensional example with two real eigenvalues, and two pairs of conjugated complex eigenvalues.

2.3 Experimental Results

We performed two sets of experiments: the goal of the first one is to evaluate the time-efficiency of the method, and the goal of the second is to evaluate the efficiency of the method in terms of speeding up reachability computations. The experimentation was done on an Intel Pentium 4 3.60Ghz, with 2 GBytes of memory.

The first set of experiments were carried out on randomly generated systems of dimensions 50 and 200, and the average computation times are around 4s and 654s respectively. The main cost of the computation comes from the computation of the box over-approximations of the initial set and of the target set.

Besides the box approximations and their projections, the computation of the lower bound of the reach time is fast (0.005s for the systems of 200 dimensions), which shows the advantage of working on low dimensional projections.

The second set of experiments was carried out on a helicopter model with 28 variables, which is a benchmark treated by the tool `SpaceEx` [1]. The initial set is defined by $x_i = 0.1$ for $1 \leq i \leq 8$, and $\forall i\{1, \ldots, 28\} : x_i \in [10 - 10^{-6}, 10 + 10^{-6}]$. We searched for the time at which the system reaches a target set defined by $\forall i\{1, \ldots, 28\} : x_i \in [-2, 2]$. Our method found a reachability time at $t = 655$. This result, which is clearly smaller than the exact reach time because of the over-approximations, allowed reducing the total reachability computation time. Indeed, to compute the reachable set from the the initial up to the time point $t = 655$ `SpaceEx` took 397s, while our computation of the reach time took only 0.241s; thus we reduced the computation time by roughly $(397 - 0.241)$s.

We can see that our method is useful in improving time-efficiency of the existing reachability algorithms, especially when the time to reach the target set from the initial set is large. In addition, to improve the accuracy of our method, the boxes may need to be subdivided, as done in [6]. Another way is to compute around the initial set the largest box that does not intersect with the target set, and then use a variant of our method for computing a lower bound on the time at which the system leaves the box. This variant is described in Section 3.

3 Application to Dynamic Hybridization

Another application of our method of reachability time domain estimation is to speed up the reachable set computation for non-linear differential systems using dynamic hybridization [5, 4]. The main idea of hybridization is to construct around the current set a domain, called *hybridization domain*, within which the non-linear system is approximated by an affine system with uncertain additive input. The input here is used to account for the approximation error. When the trajectory set is inside the domain, the affine approximate system can be used to yield the analysis result for the original system with some guaranteed bounded error. To compute the reachable set of the linear approximate system inside each domain, we can use a variety of existing techniques (such as [7] and see references there in). Basically most of these techniques are based on a discretization of time into a set of consecutive small time intervals, and in each step the reachable set is approximated for the corresponding time interval. It is important to note that as soon as the trajectory set leaves the domain, this approximate system is no longer valid and a new domain and a new approximate system need to be constructed. We can see that "hybridization" here means approximating a non-linear system by a piecewise-linear system (which is a hybrid system). The hybridization technique requires therefore checking the validity of the current approximate system by testing whether the trajectory set is not entirely included in the current domain. To avoid this intersection test, we can estimate a lower bound on the first exit time, say τ_e, and for any time $t < \tau_e$ the system is guaranteed to stay inside the current domain and no intersection test is needed. After the time τ_e, either we stay with the current approximate system and perform intersection tests, or we construct a new hybridization domain and a new approximate system.

To estimate a lower bound on the exit time, we adapt the method for reachability time domain estimation (described in the previous section), and we then show how the time-efficiency of the reachable set computation can be enhanced by avoiding the intersection test at each step.

3.1 Dynamic Hybridization

First we recall the dynamic hybridization technique[5, 4]. We consider the following autonomous non-linear system:

$$\dot{x}(t) = f(x(t)) \tag{15}$$

where $x \in \mathbb{X} \subseteq \mathbb{R}^n$ is the state variables and $Init \subset \mathbb{X}$ is a set of initial states.

The essential idea of the hybridization technique is as follows. It first constructs a simplicial domain Δ containing the initial set and inside Δ the dynamics f is approximated by an affine system l. For all $x \in \Delta$:

$$l(x) = Ax + b \tag{16}$$

where A is a matrix of size $n \times n$ and b is a vector in $\mathbb{R}^n$. The error bound μ between the original dynamics f and the approximate one, a, is:

$$\mu = \max_{x \in \Delta} \|f(x) - l(x)\|_\infty \tag{17}$$

This bound is used to define the input set $U_\mu \subset \mathbb{R}^n$:

$$U_\mu : \{u \mid u \in \mathbb{R}^n \wedge \|u\|_\infty \leq \mu\} \tag{18}$$

To obtain a conservative approximate system, an input u is added to the above affine system. For all t such that $x(t) \in \Delta$, the non-linear system can be over-approximated by the following affine system with input:

$$\dot{x}(t) = A(x(t)) + b + u(t), u(t) \in U_\mu, x(t) \in \Delta \tag{19}$$

We denote the above system as (Δ, l, U). It is of great interest to estimate a time τ_e such that before that time: the trajectory of the approximate affine system is guaranteed to stay within the hybridization domain. To this end, we need to adapt the algorithm for reachability time domain estimation, which is explained in the next section.

3.2 Exit Time Prevision

From now on, we work in the transformed basis, as defined in section 2.1, with the domain Δ (centered around the current set X) and the approximate system calculated as in (19).

To estimate a lower bound on the time at which the system intersects with the domain boundary $\partial(\Delta)$, we adapt the technique presented in the previous section. This adaptation should take into account the presence of uncertain input in the approximate dynamics. We recall that the domain Δ is a simplex.

The main idea is still to project on low dimension spaces associated with either the pairs of conjugated complex eigenvalues, or the real eigenvalues. To cope with the over-approximation due to the low dimension projection, we will use box under-approximation of the domain to stay conservative.

Complex eigenvalues For each pair of conjugate complex eigenvalues, we consider their 2-dimensional subspaces. We consider the radial evolution of the projected system to bound the exit time. To do so, we need an inner-ball approximation of the domain Δ, and then we search for the time at which the current set leaves this ball, by considering the radial evolution of the system.

Let j be the j^{th} pair of complex conjugated eigenvalues. Let c be the centroid of $Init$. We construct $B(c, \rho_b)$, the biggest ball centered at c and contained in Δ and let $B_j = proj(B(c, \rho_b), j)$ be the projection of this ball on the pair j of the corresponding dimensions, and $c_j = proj(c, j)$. Let A_j be the matrix in this 2-dimensional system.

We are now working in a 2-dimensional subspace. We perform a translation of the coordinate subsystem so that $c_j \in \mathbb{R}^2$ becomes the origin in the new coordinate system. Let $z = y - c_j$, where y is the variables of the 2-dimensional subsystem. In this new coordinate system, the dynamics of z is given by:

$$\dot{z}(t) = A_j z(t) + u_c + u(t), u(t) \in U_\mu \tag{Ez}$$

where $u_c = A_j c_j$. The solution of (Ez) is:

$$z(t) = e^{A_j t} z(0) + \int_0^t e^{A_j(t-\tau)} u_c \, d\tau + \int_0^t e^{A_j(t-\tau)} u(t) \, d\tau \tag{20}$$

The exit time is the solution given by:

$$t_{exit} = min(t : ||z(t)|| \geq \rho_b)$$

To compute this time, we need a good over-approximation of $\mathcal{I} = \int_0^t e^{A_j(t-\tau)} u(t) d\tau$. To do so, we use a time discretization of step h and proceed from time $t = 0$ until $||z(t)|| \geq \rho_b$. Under the uncertain input, the solution can be over-approximated by:

$$\|z(t)\| \leq \|e^{A_j t} z(0) + A_j^{-1}(e^{A_j t} - I) u_c\| + \| \int_0^t e^{A_j(t-\tau)} u(t) d\tau\|$$

We can prove [8] that this integral $\mathcal{I}$ for the interval $[0 \; ; \; h]$ is bounded by:

$$||\mathcal{I}|| \leq h \, ||A_j|| \, e^{h||A_j||} \, (2 \frac{\mu}{||A_j||} + (\frac{1}{2} + h) ||z(0)||). \tag{21}$$

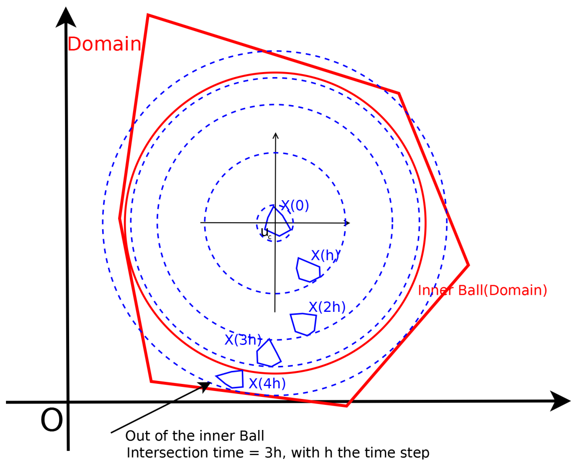

Fig. 2. Complex eigenvalues: we search for the intersection between the inner circle of the domain and the radial evolution of the system (in the basis centered at u_c). We use a stepwise computation, and in this example the intersection is found at $t = 3h$, with h the time step.

Real eigenvalues Now we show how to handle real eigenvalues. The projection of the simplex on one dimension creates a large over-approximation of the domain. To keep a conservative approximation, we find a single box under-approximation $\mathcal{B} = inner(\Delta)$ of the simplex, using the algorithm in [2], and centered at the centroid of the initial set $Init$. Similarly let $\mathcal{B}_{\mathcal{X}} = inner(X)$ where X is the current set.

Let $proj_r$ be the operator of projecting a set on the dimensions corresponding to the real eigenvalues. Once a box under-approximation $\mathcal{B}$ of Δ is determined, we can now use the projection of $\mathcal{B}_r = proj_r(\mathcal{B})$ on each dimension associated with each real eigenvalue λ_i. Let the projection $proj(\mathcal{B}_r, i)$ be represented by two constraints $x_i \leq M$ and $x_i \geq m$ where $m, M \in \mathbb{R}$. As previously in Section 2.2, we can easily compute a lower bound on the exit time for the system without input, and then for the system with input, by replacing $||A_j||$ in (21) by λ_i.

3.3 Dynamical Hybridization with Exit Time Prevision

In the dynamic hybridization [13], the domains are dynamically constructed. Our exit time prevision method can be integrated in the hybridization algorithm to avoid polytopic inclusion tests (which in general require solving LP problems).

The main steps of the original hybridization algorithm are as follows. Given an initial $Init$. For each iteration, the algorithm performs the following steps. First, we compute an approximation domain Δ and its associated linear approximate system (Δ, l, U) as in (19). We then compute the reachable set Rn from R using the step-by-step algorithm with the time step h. We test if $newReach$ intersects with the boundary $\partial(\Delta)$ of Δ. If so, we discard the set $newReach$. Otherwise, we continue with the next iteration.

Now we explain how the above-described exit time prevision method allows reducing the number of intersection tests between the set Rn and the boundary $\partial(\Delta)$, by predicting a lower bound τ_e on the exit time (see Algorithm 1). If τ_e is not larger than the time step, that is $\tau_e \leq h$, we ignore this result and use the original algorithm (with intersection test at each step) until the next domain is needed. Otherwise, we can compute the reachable set for the linear approximate system (Δ, l, U) without intersection test until τ_e.

Algorithm 1 Hybridisation with Exit Time Prevision.

```
1: function Reach($(Init, f, h)$)
2:     $Reach = \emptyset$
3:     $t = 0$
4:     $R = Init$
5:     repeat
6:         $(\Delta, l, U) = Domain(R, f)$
7:         $\tau_e = ExitTimePrevision(R, \Delta)$
8:         if $(\tau_e > h)$ then
9:             /* Computing the reachable set without intersection test */
10:            for all $t \leq \tau_e$ do
11:                $Rn = LinReach(R, l, U, h)$
12:                $Reach = Rn \cup Rn$
13:                $R = Rn$
14:                $t = t + h$
15:            end for
16:        else
17:            /* Computing the reachable set with intersection test */
18:            $newDomain = false$
19:            repeat
20:                $Rn = LinReach(R, l, U, h)$
21:                if $(Rn \cap \delta(\Delta) = \emptyset)$ then
22:                    $Reach = Reach \cup Rn$
23:                    $R = Rn$
24:                    $t = t + h$
25:                else
26:                    $newDomain = true$
27:                end if
28:            until $newDomain$
29:        end if
30:    until $t \geq t_{max}$
31:    return $Reach$
32: end function
```

3.4 Experimental Results

To show how the hybridization algorithm with exit time prevision (HPA) is more time-efficiency than the original hybridization algorithm (HA) [13], we used the

7-dimensional polynomial biological model *Dictyostelium discoideum* [10], also used in [13]. This model, extracted from the molecular network, describes the aggregation stage of Dictyostellium, and its spontaneous oscillations during its development process. The equations are given bellow and the parameters value can be found in [10].

$$\begin{aligned}
\frac{d[ACA]}{dt} &= k1[ERK2] - k2[ACA] \\
\frac{d[PKA]}{d}t &= k3[internal\ cAMP] - k4[PKA] \\
\frac{d[ERK2]}{dt} &= k5[CAR1] - k6[ERK2][PKA] \\
\frac{d[REG\ A]}{dt} &= k7 - k8[REG\ A][ERK2] \\
\frac{d[internal\ cAMP]}{dt} &= k9[ACA] - k10[REG\ A][internal\ cAMP] \\
\frac{d[external\ cAMP]}{dt} &= k11[ACA] - k12[external\ cAMP] \\
\frac{d[CAR1]}{d}t &= k13[external\ cAMP] - k14[CAR1][PKA]
\end{aligned}$$

We studied the oscillating behaviours of the process. In fact, our reachability results show that for some initial states, the system can stop oscillating. This behaviour can be seen on the Figure 3.4 representing the evolution of the variable $CAR1$ in function of the variable $internal\ cAMP$.
Again, the experimention was done on an Intel i7 720QM quad core 1.60Ghz, with 4 GBytes of memory. The reachable set computed by HPA are coherent with the one computed by HA, and they can be seen in Figure 3.4 and Figure 3.4. If we compare the total execution times, for 2000 iterations, HPA took 88 seconds, while HA took 128 seconds. The gain is 31.25%. HPA still needs intersection tests when the estimated exit time is smaller than the time step h (at about 5% of the total number of iterations), but these tests took on 0.43s, while HA needed 7.21s for intersection tests. The total time of exit time estimation was 0.24s.

Using the exit time estimation, we greatly reduced the computation time of the intersection detection. As future work, we plan to reduce the domain computation cost by using domains which are rectangular in the eigenbasis to reduce over-approximation due to the low dimension projections.

Conclusion

The essential idea of the methods presented in this paper is to extract time information from the symbolic expressions of the components $x_i(t)$ expressed in the eigenbasis of the matrix A of the linear system. Our goal is to compute what we call an RTD. This allows, in a set-based simulation or a reachability analysis, to skip the parts of the time domain which corresponds to the complement of the computed RTD. Consequently it can be seen as an acceleration

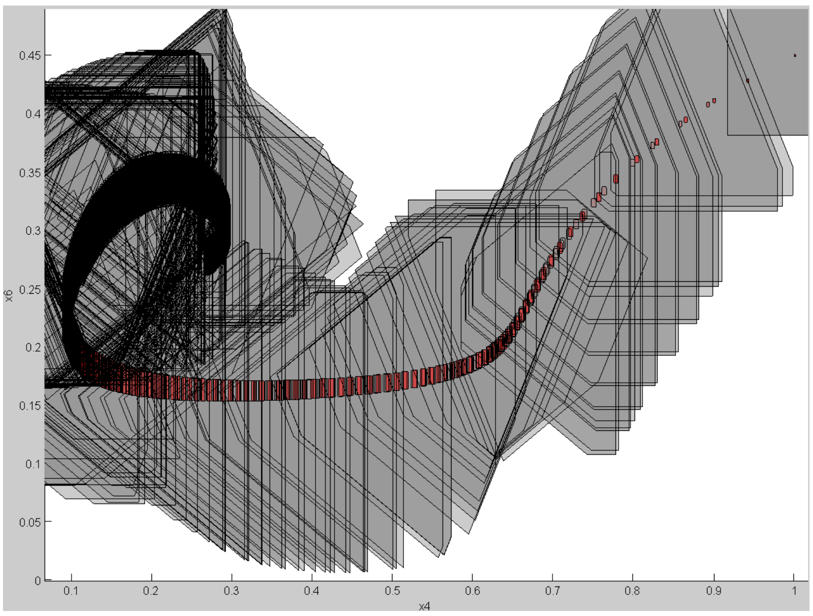

Fig. 3. The picture shows the projection on the plan ($CAR1$, *internal* $cAMP$) of the reachable set computed with [13] implementation.

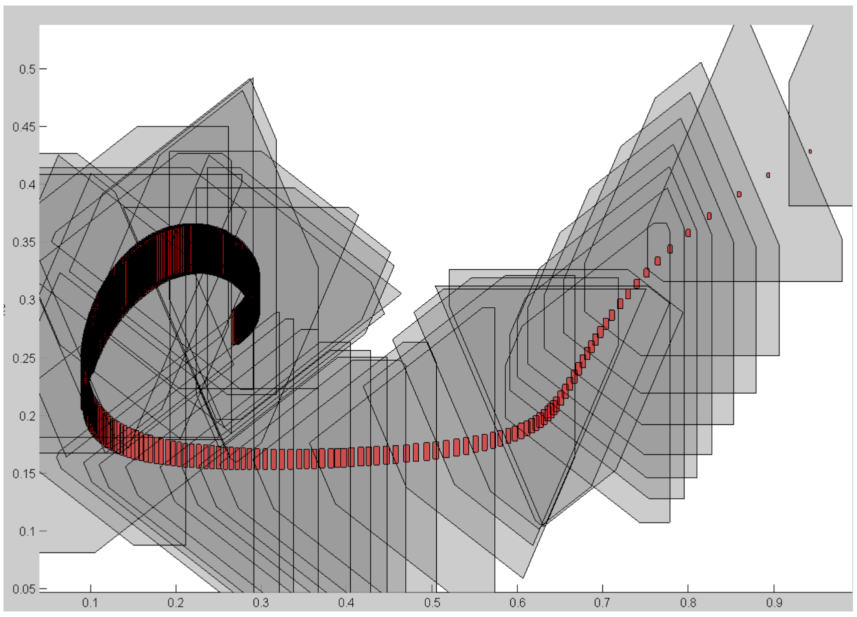

Fig. 4. The picture shows the projection on the plan ($CAR1$, *internal* $cAMP$) of the reachable set computed with the new implementation.

technique that can be integrated in reachability tools (for example `SpaceEx` or [13]). The authors of [9] used the symbolic expressions of the components $x_i(t)$ expressed in the eigenbasis, but in a very different way. Their goal is to define new decidable classes of linear hybrid systems. Their method, based on quantifier elimination, applies when either A is nilpotent, or its eigenvalues are either all reals or all purely imaginary. Although important from the theoretical viewpoint, these classes are too restricted for practical problems. The work [11] is closer to our approach. It also uses also analytical solutions $x_i(t)$ (in the eigenbasis), and makes a piecewise linear approximation of the natural logarithm function on the real axis in order to find linear relations involving time and state variables. In that way it produces an abstraction of the solution (called time-aware relational abstraction), and then use bounded model checking to verify the linear hybrid systems. The abstraction can be refined by increasing the number of points in the piecewise linear approximation. The recent paper [6] describes a safety verification tool for linear systems also based on the idea of using symbolic expressions of the components $x_i(t)$. Their goal is to perform safety verification using a counterexample guided abstraction refinement (CEGAR) procedure. So their goal is different from ours, and consequently the way they exploit the time information contained in the $x_i(t)$ components is different, too. At the present time they use only the real eigenvalues, and plan to extend the approach to what they call quadratic eigenforms. This will allow them to extract time information from the real part of complex eigenvalues. In the present work we exploit the time information contained in real eigenvalues, and in the real and imaginary part of the complex eigenvalues. However, because of rough approximation of the initial set and target set in our methods, the application to solve the linear reachability problems suffers from some precision loss. Our future plan includes an improvement of the precision by using a more refined approximation for these sets, and by compensating the precision loss due to the projection. Combining both the dynamic hybrization technique and the linear reachability methods will give a powerful reachability tool which is also valid for non-linear systems.

References

1. Spaceex: State space explorer. In *http://spaceex.imag.fr/* (2010).
2. Bemporad, A., Filippi, C., and Torrisi, F. D. Inner and outer approximations of polytopes using boxes. *Computational Geometry 27*, 2 (2004), 151–178.
3. Cleve Moler, C., and Van Loan, C. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review 45*, 1 (2003), 3–49.
4. Dang, T., Le Guernic, C., and Maler, O. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology* (2009), Springer, pp. 126–141.
5. Dang, T., Maler, O., and Testylier, R. Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control* (2010), ACM, pp. 11–20.
6. Duggirala, P. S., and Tiwari, A. Safety verification for linear systems. In *Embedded Software (EMSOFT), 2013 Proceedings of the International Conference on* (2013), IEEE, pp. 1–10.

7. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., and Maler, O. Spaceex: Scalable verification of hybrid systems. In *Computer Aided Verification* (2011), Springer, pp. 379–395.
8. Guernic, C. L. *Reachability analysis of hybrid systems with linear continuous dynamics*. PhD thesis, Université Grenoble 1 - Joseph Fourier, 2009.
9. Lafferriere, G., Pappas, G. J., and Yovine, S. A new class of decidable hybrid systems. In *Hybrid Systems: Computation and Control*. Springer, 1999, pp. 137–151.
10. Laub, M. T., and Loomis, W. F. A molecular network that produces spontaneous oscillations in excitable cells of dictyostelium. *Molecular biology of the cell 9*, 12 (1998), 3521–3532.
11. Mover, S., Cimatti, A., Tiwari, A., and Tonetta, S. Time-aware relational abstractions for hybrid systems. In *Proceedings of the Eleventh ACM International Conference on Embedded Software* (2013), IEEE Press, p. 14.
12. Perko, L. Linear systems. In *Differential Equations and Dynamical Systems*. Springer, 1991, pp. 1–63.
13. Testylier, R., and Dang, T. NLTOOLBOX: A library for reachability computation of nonlinear dynamical systems. In *Automated Technology for Verification and Analysis*. Springer, 2013, pp. 469–473.