BUBBIBEX with IBEX

Option ROB ENSTA Bretagne 2014
ENSTA Bretagne, 2 rue F. Verny, 29200, Brest.

14/02/2014

Authors. A. Akkouche, J.-B. Bénéfice, Q. Bréfort, F. Carbonera, B. Desrochers, T. Issautier, M. Laranjeira-Moreira, V. Le Doze, D. Monnet, A. Oubelhaj.

Supervisors. L. Jaulin, J. Ninin, M. Saad and S. Le Menec

Abstract. This project aims to developing a simple software called BUBBIBEX to study the stability of a autonomous robot from its state equations. The study of the stability is based on an interval approach. We were 10 students divided into 5 teams and we all program in C++ on Qt-Creator, from an existing software called SIVIA [8] and IBEX 2.0 [3] [2] which is a C++ numerical library based on interval arithmetic and constraint programming.

1 Introduction

A dynamic system can often be described by a state equation $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t)$ where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{u} \in \mathbb{R}^m$ is the control vector and $\mathbf{f} : \mathbb{R}^n \times \mathbb{R}^p \times \mathbb{R} \to \mathbb{R}^n$ is the evolution function. Assume that the control low $\mathbf{u} = \mathbf{g}(\mathbf{x}, t)$ is known, the system becomes autonomous and will be called a robot. If we define $\mathbf{f}(\mathbf{x}, t) = \mathbf{f}(\mathbf{x}, \mathbf{g}(\mathbf{x}, t), t)$, we get an the following equation.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t).$$

The validation of the stability of this robot is an important and difficult problem [9] which can be transformed into proving the inconsistency of a constraint satisfaction problem. This has be shown by using the V-stability approach introduced in the previous work for *innovation Lab* and used in several applications such as the validation of the control low of sailboats [1]. Here, we extend this work to systems where the target to be reached depends on time and we will try to show that the approach is able to deal with large dimensional problems.

The document proposes a simple formulation for one robot to satisfy some set-membership time dependant constraints. The approach will be based on interval analysis which is a efficient tool to deal with uncertainties when the variables are linked by nonlinear equations **??**.

2 Problem statement

Consider an autonomous robot described by a state equation with some target conditions. This problem is assumed to be described by

$$\begin{cases} \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t) & \text{(evolution equation)} \\ \mathbf{g}(\mathbf{x}, t) \leq \mathbf{0} & \text{(target condition)} \end{cases} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector, $\mathbf{f} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ is the evolution function and $\mathbf{g} : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$ is the target function. The set

$$\mathbb{G}(t) = \{\mathbf{x}, \mathbf{g}(\mathbf{x}, t) \leq \mathbf{0}\}$$

is called a *target tube*. A trajectory $\mathbf{x}(t)$ is *safe* if it satisfies the two conditions (1). The system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ is said to be *safe* if for all $\mathbf{x}(0) \in \mathbb{G}(0)$, the corresponding trajectory is safe. The notion can be intepreted as an extention of the V-stability when the system is time dependant [7].

Theorem 1. Consider a trajectory $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ with $\mathbf{x}(0) \in \mathbb{G}(0)$. If the system of constraints

$$\begin{cases} \text{(i)} & \frac{\partial g_i}{\partial \mathbf{x}}(\mathbf{x}, t) . \mathbf{f}(\mathbf{x}, t) + \frac{\partial g_i}{\partial t}(\mathbf{x}, t) \geq 0 \\ \text{(ii)} & g_i(\mathbf{x}, t) = 0 \\ \text{(iii)} & \mathbf{g}(\mathbf{x}, t) \leq 0 \end{cases} \tag{2}$$

is inconsistent for all $\mathbf{x}$, all $t \geq 0$ and all $i \in \{1, \ldots, m\}$ then the system is safe.

Proof. The proof is by contradiction. Assume that the system is not safe. Then, there exists $\mathbf{x}(0) \in \mathbb{G}(0)$ and $t_1 > 0$ such that $\mathbf{x}(t)$ get out of the target tube at time t_1. Thus, there exists one i such that

$$g_i(\mathbf{x}, t_1) = 0 \text{ and } \dot{g}_i(\mathbf{x}, t_1) \geq 0 \text{ and } \mathbf{g}(\mathbf{x}, t_1) \leq 0,$$

which means that the trajectory crosses the boundary of the target tube from inside to outside. Equivalently, we have

$$\begin{cases} g_i(\mathbf{x}, t_1) = 0 \\ \frac{\partial g_i}{\partial \mathbf{x}}(\mathbf{x}, t_1) . \mathbf{f}(\mathbf{x}, t_1) + \frac{\partial g_i}{\partial t}(\mathbf{x}, t_1) \geq 0 \\ \mathbf{g}(\mathbf{x}, t_1) \leq 0 \end{cases}$$

which is inconsistent with the assumption of the theorem.

Example. Consider an autonomous robot described by a state equation with some target conditions.

$$\begin{cases} \dot{x} = -x + t^2 & \text{(evolution equation)} \\ (x - t^2)^2 - 1 \leq 0 & \text{(bubble condition)} \end{cases}$$

The system is bubble-stable if, initialized in the bubble for $t = 0$, it always stays inside the bubble. Let is now find the condition on (x, t) to have a system which is bubble-stable. From the Theorem, we study the system

$$\begin{cases} \text{(i)} & \frac{\partial g}{\partial x}(x, t) . f(x, t) + \frac{\partial g}{\partial t}(x, t) \geq 0 & \Leftrightarrow & 2(x - t^2)(-x + t^2) + 2(x - t^2)(-2t) \geq 0 \\ \text{(ii)} & g(x, t) = 0 & \Leftrightarrow & (x - t^2)^2 - 1 = 0 \\ \text{(iii)} & g(x, t) \leq 0 & \Leftrightarrow & (x - t^2)^2 - 1 \leq 0 \end{cases}$$

i.e.,

$$\left\{ \begin{array}{ll} \text{(i)} & \left(x-t^2\right)^2 + 2\left(x-t^2\right)t \leq 0 \\ \text{(ii)} & x - t^2 = 1 \end{array} \right.$$

i.e.

$$\left\{ \begin{array}{ll} \text{(i)} & 1 + 2t \leq 0 \\ \text{(ii)} & x - t^2 = 1. \end{array} \right.$$

The set of solution is depicted below. Since there is no solution, the system is bubble stable.

In this simple example, the system is simple to solve by hand. But is general, one has to use interval analysis methods [10] to show that we do not have any solution. Note that interval analysis has been used in many applications involving robots [5], [11], [4].

3 System

We decided to develop our software on the example of a char robot [6], which aims to reach a moving target. However, this software can be reused with other types of robot (and hence other state equations). We only use this example to have a first approach to design this software. The state equations of the robot are the following

$$\left\{ \begin{array}{rcc} \dot{x} & = & u_1 \cos\theta \\ \dot{y} & = & u_1 \sin\theta \\ \dot{\theta} & = & u_2 \end{array} \right.$$

where $u_1 > 0$ is the speed, u_2 is the yaw, θ is its heading and (x, y) are the coordinates of its center. The state vector is $\mathbf{x} = (x, y, \theta)$. Assume that the yaw should satisfy $u_2 \in [-1, 1]$. This system is non holonomous because we cannot move toward any direction of the space (x, y, θ). For instance, the direction $\left(\dot{x} = 1, \dot{y} = 0, \dot{\theta} = 1\right)$ is not possible. If we want the robot to go toward the direction θ_d, a natural control strategy is to take

$$u_2 = -K \sin\left(\theta - \theta_d\right).$$

The heading θ would thus satisfy the differential equation

$$\dot{\theta} = -K \sin\left(\theta - \theta_d\right),$$

which is stable. This means at time 0 that $\theta \simeq \theta_d$ then θ will converge toward θ_d, if θ_d is constant. Therefore, the closed loop system satisfies

$$\left\{ \begin{array}{rcc} \dot{x} & = & u_1 \cos\theta \\ \dot{y} & = & u_1 \sin\theta \\ \dot{\theta} & = & -K \sin\left(\theta - \theta_d\right). \end{array} \right.$$

This system still has two inputs: u_1 and θ_d. We want now want the robot goes toward the moving target (x_d, y_d), we can take the following proportional controller

$$\left\{\begin{array}{lll} \text{(i)} & \mathbf{n} & = \left(\begin{array}{c} x_d - x \\ y_d - y \end{array}\right) + \left(\begin{array}{c} \dot{x}_d \\ \dot{y}_d \end{array}\right) \\ \text{(ii)} & u_1 & = \|\mathbf{n}\| \\ \text{(iii)} & \theta_d & = \operatorname{atan2}(\mathbf{n}) \end{array}\right.$$

The reason is that in the case where $\theta = \theta_d$ (which is approximately the case), we have

$$\begin{array}{ccccc} \left(\begin{array}{c} \dot{x} \\ \dot{y} \end{array}\right) & \overset{(??)}{=} & \left(\begin{array}{c} u_1 \cos\theta \\ u_1 \sin\theta \end{array}\right) & \overset{(??,(ii),(iii))}{=} & \left(\begin{array}{c} \|\mathbf{n}\| \cos(\operatorname{atan2}(\mathbf{n})) \\ \|\mathbf{n}\| \sin(\operatorname{atan2}(\mathbf{n})) \end{array}\right) = \mathbf{n} \\ & \overset{(??,(i))}{=} & -\left(\begin{array}{c} x - x_d \\ y - y_d \end{array}\right) + \left(\begin{array}{c} \dot{x}_d \\ \dot{y}_d \end{array}\right) & & \end{array}$$

Therefore, if $e_x = x - x_d$ and $e_y = y - y_d$, we get

$$\left\{\begin{array}{lll} \dot{e}_x + e_x & = & 0 \\ \dot{e}_y + e_y & = & 0 \end{array}\right.$$

Thus e_x and e_y will converge to zero, i.e., the robot will converge to the target. Now, θ and θ_d are not exactly equal and the convergence is not guaranteed. The autonomous robot satisfies

$$\left\{\begin{array}{lll} \dot{x} & = & \|\mathbf{n}\| \cos\theta \\ \dot{y} & = & \|\mathbf{n}\| \sin\theta \\ \dot{\theta} & = & -K \sin(\theta - \operatorname{atan2}(\mathbf{n})) \end{array}\right.$$

Now, since

$$\sin(a - b) = \sin a \cos b - \cos a \sin b,$$

we get

$$\begin{array}{rcl} \dot{\theta} & = & -K \sin\theta \cdot \cos(\operatorname{atan2}(\mathbf{n})) + K \cos\theta \cdot \sin(\operatorname{atan2}(\mathbf{n})) \\ & = & -K \sin\theta \dfrac{n_x}{\|\mathbf{n}\|} + K \cos\theta \dfrac{n_y}{\|\mathbf{n}\|}. \end{array}$$

Since

$$\|\mathbf{n}\| \overset{(??,(i))}{=} \sqrt{(x_d - x + \dot{x}_d)^2 + (y_d - y + \dot{y}_d)^2},$$

the state equation are thus

$$\left\{\begin{array}{lll} \dot{x} & = & \sqrt{(x_d - x + \dot{x}_d)^2 + (y_d - y + \dot{y}_d)^2} \cdot \cos\theta \\ \dot{y} & = & \sqrt{(x_d - x + \dot{x}_d)^2 + (y_d - y + \dot{y}_d)^2} \cdot \sin\theta \\ \dot{\theta} & = & \dfrac{\cos\theta(y_d - y + \dot{y}_d) - \sin\theta(x_d - x + \dot{x}_d)}{\sqrt{(x_d - x + \dot{x}_d)^2 + (y_d - y + \dot{y}_d)^2}} \end{array}\right. \tag{3}$$

which corresponds to a time dependant system. Note that the target trajectory $(x_d(t), y_d(t))$ is completely known with their derivatives $\dot{x}_d(t)$ and $\dot{y}_d(t)$. For instance $x_d(t)$ and $y_d(t)$ may be polynomials.

Target tube. We want the robot to stay inside the set

$$\mathbb{G}(t) = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}, t) \leq \mathbf{0}\},$$

with

$$\left\{ \begin{array}{rcl} g_1(\mathbf{x}, t) & = & (x - x_d)^2 + (y - y_d)^2 - r^2 \\ g_2(\mathbf{x}, t) & = & \left(\cos\theta - \frac{x_d - x + \dot{x}_d}{\sqrt{(x_d - x + \dot{x}_d)^2 + (y_d - y + \dot{y}_d)^2}}\right)^2 + \left(\sin\theta - \frac{y_d - y + \dot{y}_d}{\sqrt{(x_d - x + \dot{x}_d)^2 + (y_d - y + \dot{y}_d)^2}}\right)^2 - \varepsilon_\theta. \end{array} \right. \quad (4)$$

The first inequality asks the center of the robot to stay inside a disk with a radius r and a center (x_d, y_d). The second inequality tells us that the heading of the robot should correspond to $\mathbf{n}$.

Target. The target is assumed to have a known equation. In our example, we will take

$$\left\{ \begin{array}{rcl} x_d(t) & = & 7.t \\ y_d(t) & = & \sin(\nu t) \end{array} \right.$$

where ν is a parameter that are assumed to be fixed by the user. For the derivative, we get

$$\left\{ \begin{array}{rcl} \dot{x}_d(t) & = & 7 \\ \dot{y}_d(t) & = & \nu\cos(\nu t) \end{array} \right.$$

Parameters. In what follows, we will consider the following parameters

parameter	**value**	**meaning**
ε_θ	0.2	error on the heading
r	1	radius of the target circle
ν	0.1	frequency of the target cycle

Resolution. For the resolution, we will call a solver based on interval analysis. More precisely, we will have to prove twice (i.e., for $i \in \{1, 2\}$) the inconsistency of the system (5), where $\mathbf{f}$ and $\mathbf{g}$ are given by (3) and (4). The research space $(\mathbf{x}, t)$ will then be of dimension 4. We will thus have to enter inside the solver (see the next section) the two following functions

$$\mathbf{f}(\mathbf{x}, t) = \left(\begin{array}{c} \sqrt{(7t - x_1 + 7)^2 + \left(\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}\right)^2} \cdot \cos x_3 \\ \sqrt{(7t - x_1 + 7)^2 + \left(\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}\right)^2} \cdot \sin x_3 \\ \frac{(\cos x_3)\left(\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}\right) - (\sin x_3)(7t - x_1 + 7)}{\sqrt{(7t - x_1 + 7)^2 + \left(\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}\right)^2}} \end{array} \right)$$

$$\mathbf{g}(\mathbf{x}, t) = \left(\begin{array}{l} (x_1 - 7t)^2 + (x_2 - \sin\frac{t}{10})^2 - 1 \\ \left(\cos x_3 - \frac{7t - x_1 + 7}{\sqrt{(7t - x_1 + 7)^2 + \left(\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}\right)^2}}\right)^2 \cdots \\ \quad \cdots + \left(\sin x_3 - \frac{\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}}{\sqrt{(7t - x_1 + 7)^2 + \left(\sin\frac{t}{10} - x_2 + \frac{1}{10}\cos\frac{t}{10}\right)^2}}\right)^2 - 0.2 \end{array} \right)$$

4 Software features

- The software BUBBIBEX has as inputs:
  - Two files `f.txt` and `g.txt` corresponding to the functions $\mathbf{f}(\mathbf{x},t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ and $\mathbf{g}(\mathbf{x},t) : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^m$.
  - A value of $t_{\max}$ and a box for $[\mathbf{x}] \subset \mathbb{R}^n$.

  The state space will have a dimension $n \geq 2$. The two variables x_1, x_2 will have a special status: the paving in $\mathbb{R}^n$ will always be projected in the (x_1, x_2) space to be painted on the 2D screen of the computer.
- The interface contains one trackbar for $t \in [0, t_{\max}]$ and three checkbox (*draw paving, draw simulation, draw field*).
- The BUBBIBEX software allows to perform the following tasks:
  - *Check capture:* A SIVIA like algorithm compute an enclosure for the set

    $$\mathbb{S} = \{(\mathbf{x},t) \in [\mathbf{x}] \times [0, t_{\max}] \mid (5) \text{ is satisfied}\}$$

    If SIVIA concludes that $\mathbb{S}$ is empty,it will write on the screen that the system is safe and it will store the resulting subpavings (list of boxes) on $\mathbb{S}^{\text{out}}$ and $\mathbb{S}^{?}$. The subpaving $\mathbb{S}^{\text{out}}$ contains boxes that are outside $\mathbb{S}$ and boxes in $\mathbb{S}^{?}$ are uncertain (empty is the system is proved to be safe).
  - *Simulation:* Using a Monte-Carlo method, BUBBIBEX selects a cloud of initial conditions in $\mathbb{G}_0$. For each particle, a simulation is run and the trajectory is stored in the memory.
  - *Draw simulation:* If the corresponding checkbox is selected, the trajectory is painted in the (x_1, x_2) plane (black if the target tube is satisfied, red otherwise). The trackbar allows the user to see the position of the robot at the current time in the trajectory.
  - *Draw Paving:* If the checkbox is selected, when the trackbar changes its position, BUBBIBEX takes all boxes $[\mathbf{x}] \times [t]$ in $\mathbb{S}^{\text{out}}$ and $\mathbb{S}^{?}$. If t_1 denotes the time of the trackbar, we select only boxes $[\mathbf{x}] \times [t]$ such that $t_1 \in [t]$. Then all corresponding $[x_1] \times [x_2]$ are painted. The color should be white for boxes coming from $\mathbb{S}^{\text{out}}$ and grey for boxes coming from $\mathbb{S}^{?}$.
  - *Draw field:* If the checkbox is selected, the time t is selected on the trackbar. From a griding on the $\mathbf{x}$ space, the arrow corresponding to $\mathbf{f}(\mathbf{x},t)$ is computed for each point of the griding. The projection of the arrow on the (x_1, x_2) plane is painted.

5 Development process

We divided the work and allocate it into the teams. Each team presents its work in this part.

5.1 Scenario definition by T. Issautier and D. Monnet

Our group is responsible for initializing the functions f and g and for creating different scenario . It consists in automatically create two .txt files: f.txt and g.txt. Each of these files contains the concerned function expression with respect to the Minibex syntax. These .txt files will be reused by other groups for the calculation of subpaving for example. We implemented, in order to later use, the possibility for the user to define his own functions. Indeed, two text-editing areas are present on the GUI and are intended to capture expressions of desired functions. A "Save" button is also provided to generate .txt files from the expressions of the functions contained in the text boxes. By default, functions present in the project statement automatically appear in the text-editing boxes.

However this scenario is not trivial and we added two basic scenarios to test the functionality and reliability of our program. We also can improve the debugging with these additional scenarii.

The additional first scenario is as follows:

$$\begin{aligned} \mathbf{f}(\mathbf{x},t) &= \begin{pmatrix} -x_1+t \\ -x_2 \\ -x_3 \end{pmatrix} \\ \mathbf{g}(\mathbf{x},t) &= \begin{pmatrix} ((x_1-t)^2+(x_2)^2-1) \\ (\cos(x_3)-1)^2+(\sin(x_3))^2-0.2 \end{pmatrix} \end{aligned}$$

The additional second scenario is as follows:

$$\begin{aligned} \mathbf{f}(\mathbf{x},t) &= \begin{pmatrix} \sqrt{(7t-x_1+7)^2+x_2^2}cos(x_3) \\ \sqrt{(7t-x_1+7)^2+x_2^2}sin(x_3) \\ 10\cos(x_3)(\frac{(-x_2)}{\sqrt{((7t-x_1+7)^2+(-x_2)^2)}})-\sin(x_3)(\frac{(7t-x_1+7)}{\sqrt{((7t-x_1+7)^2+(-x_2)^2)}}) \end{pmatrix} \\ \mathbf{g}(\mathbf{x},t) &= \begin{pmatrix} ((x_1-t)^2+(x_2)^2-1) \\ (\sin(x_3)-(\frac{(7t-x_1+7)}{\sqrt{((7t-x_1+7)^2+(-x_2)^2)}}))^2+(\sin(x_3)-(\frac{(-x_2)}{\sqrt{((7t-x_1+7)^2+(-x_2)^2)}}))^2-0.01 \end{pmatrix} \end{aligned}$$

5.2 Simulation, by Q. Bréfort and V. Le Doze

The purpose here is to draw the path of the robot and its position through time, for a random initial position.

5.2.1 The safe state condition

To be in a safe state, the robot must satisfy the equations: $\mathbf{g}(\mathbf{x},t) \leq 0$.

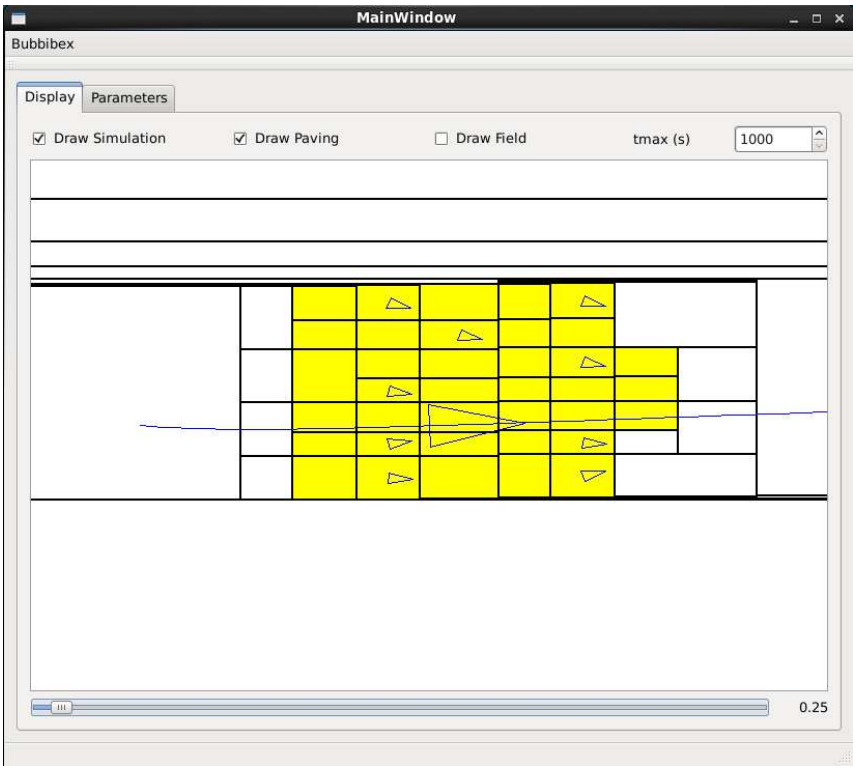
However, if the robot has an initial state ($t = 0$) which do not satisfy this condition, it will certainly not satisfy it in its future states. So, when we generate a random initial position for the robot, we need to make sure that: $\mathbf{g}(\mathbf{x},0) \leq 0$.

In the simulation part, BUBBIBEX uses a Monte-Carlo method to show the safe state condition of a system. So, it generates automatically a robot with a random initial state in respect with the condition and draws the robot's path according to its state equations. In addition, the program allows us to draw the state of the robot at a given time with the track bar in the bottom.

5.2.2 The limits of the "Bubble"

The BUBBIBEX program draws only a bubble which shows the G_0 condition: $g_1(\mathbf{x}, t) \leq 0$. So, even if a robot satisfies this condition, we have no graphic information about the G_1 condition: $g_2(\mathbf{x}, t) \leq 0$. Thus, it is hard to understand the behavior of a robot on the limits of the Bubble. For example, a robot could be inside the Bubble, but having a direction pointing toward the outside of it. In this case, the robot will certainly not be safe, because it will diverge from its target.

Therefore, the BUBBIBEX program draws 10 random robots on the edge of the Bubble at a given time, according to the results of $\mathbb{S}^?$, and allows the user to see if they will or not be safe.



The larger triangle represents the actual robot at its given position for time t and its path in blue.
The smaller triangles represent robots position in a state
in which the stability could not be established (either G0 or G1 is not satisfied).

5.3 Check Capture, by F. Carbonera and M. Laranjeira-Moreira

The goal of this part is to check if the system is safe for all state vectors ($\mathbf{x}$) and for all time ($\mathbf{t}$). To do so, we will compute an inclusion test in a set $\mathbf{S}$. The set $\mathbf{S}$ is composed by the state vector and

the time vector. Hence, **S** has dimension 4 : x1, x2, θ and t. The constraints functions and the test for prove the safety of the system are described below:

Theorem 1. Consider a trajectory $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, t)$ with $\mathbf{x}(0) \in \mathbb{G}(0)$. If the system of constraints

$$\begin{cases} \text{(i)} & \frac{\partial g_i}{\partial \mathbf{x}}(\mathbf{x}, t).\mathbf{f}(\mathbf{x}, t) + \frac{\partial g_i}{\partial t}(\mathbf{x}, t) \geq 0 \\ \text{(ii)} & g_i(\mathbf{x}, t) = 0 \\ \text{(iii)} & \mathbf{g}(\mathbf{x}, t) \leq 0 \end{cases} \tag{5}$$

is inconsistent for all $\mathbf{x}$, all $t \geq 0$ and all $i \in \{1, \ldots, m\}$ then the system is safe.

So, we will store the safe regions of **S** in a subset **Sout** and the regions of **S** that are uncertain will be stored in the subset **Sp**.

As we have two functions $\mathbf{g}(\mathbf{x}, t)$ in our test, we have as a final contractor a union of two sub-contractors. In other words we will compute the test described by **Theorem 1** twice, once for each function $\mathbf{g}(\mathbf{x}, t)$. Then, the final test will be the union of these sub-tests.

5.4 Draw Paving, by A. Akkouche and A. Oubelhaj

Our first task in this project was to create a program that allows us to show in a window, the decomposition of the space in gray and white. White being the code color for the «out» boxes and gray for the «possible» boxes.

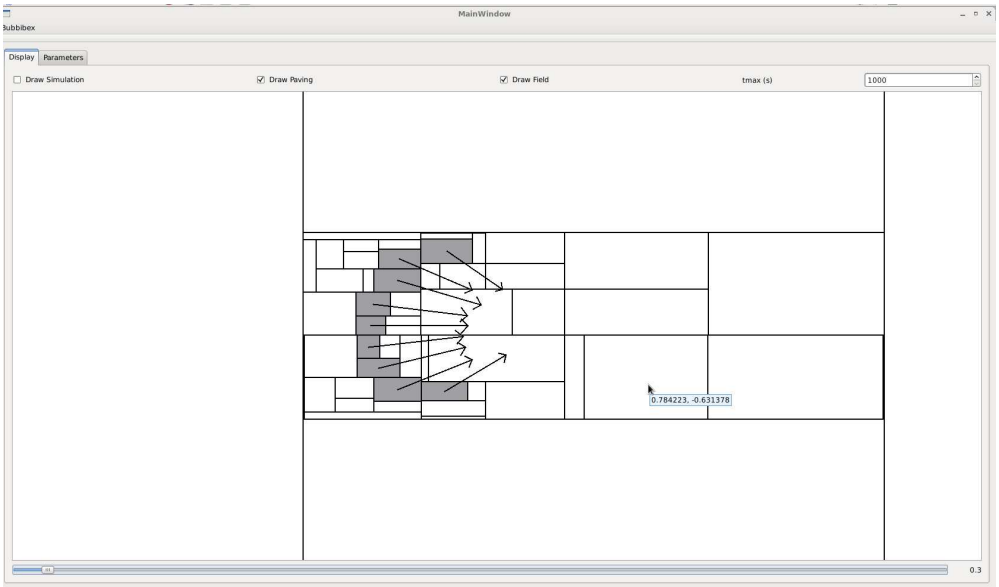
The first information is the time in which we wish to show the situation («out» and «possible» boxes represented in code colors). The time is set trough a trackbar. The second information,coming from the sivia file, is in the form of two lists of IntervalVectors, one for the «out» boxes, the other for «possible» boxes. The IntervalVectors are in the form of $[[x], [y], [\theta], [t]]$.

Once our program has the data, it goes trough each one of the two lists and checks whether or not the time indicated in the trackbar belongs in the interval $[t]$. If it is the case, we will represent $[x] * [y]$ in the main frame, if not, the IntervalVector is not represented.

To do so, we first programmed a function `update()`, which given a «repere», an `IntervalVector`, a color code and a time t_1, will check if t_1 belongs to $[t]$, then if it is the case will represent $[x] * [y]$ in the main frame. We then programmed the main function, which will go trough both lists Sout and Spossible, extracting the IntervalVectors and using the `update()` function to represent them. The illustration below represents the drawing of the boxes in gray and white in the time indicated by the trackbar.

Boxes out(white) and possible(gray)

The second task consisted on drawing the field of vectors inside the probable boxes. To do this, our program loads the function f from a `*.txt` file to have its expression. And just as our first program, this one uses the same data from the same two other programs. It receives a list of IntervalVectors of the possible boxes as well as the current time set by the trackbar. The program then evaluates the **f** function in the middle point of the possible boxes. We then use the `DrawArrow()` function to draw the arrows representing the values of the function in these points. The illustration below represents the drawing of the field's arrows in the possible boxes.

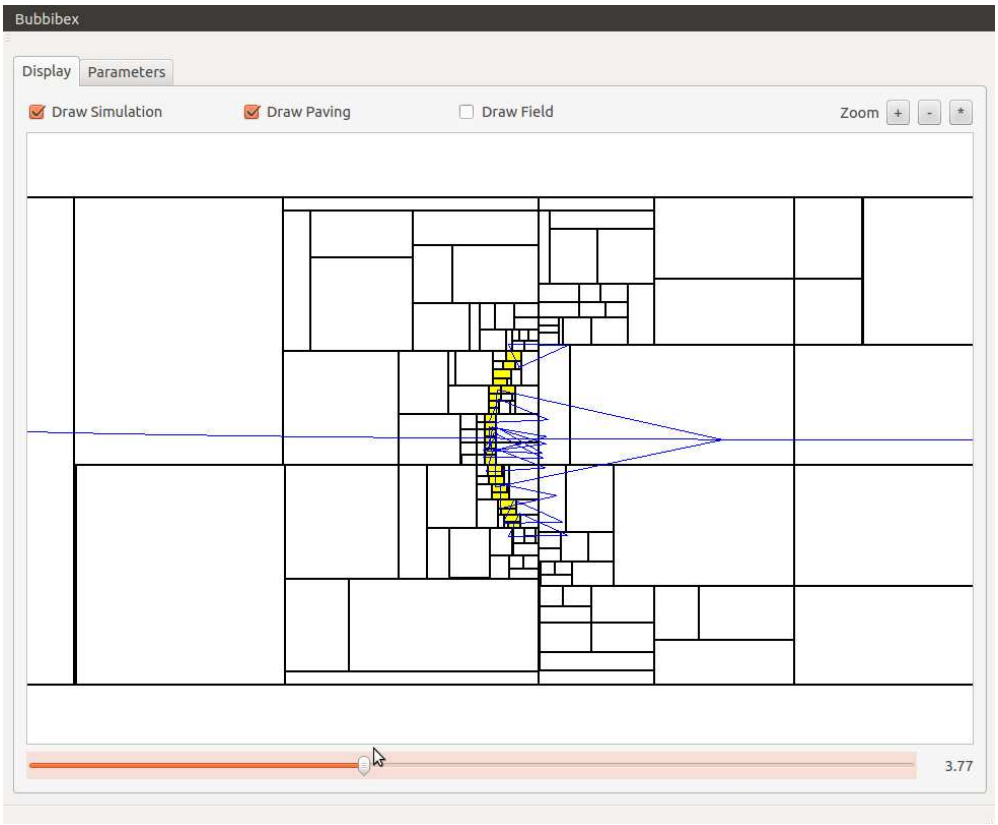


Arrows of the field

5.5 Interface, by J.-B Bénéfice and B. Desrochers

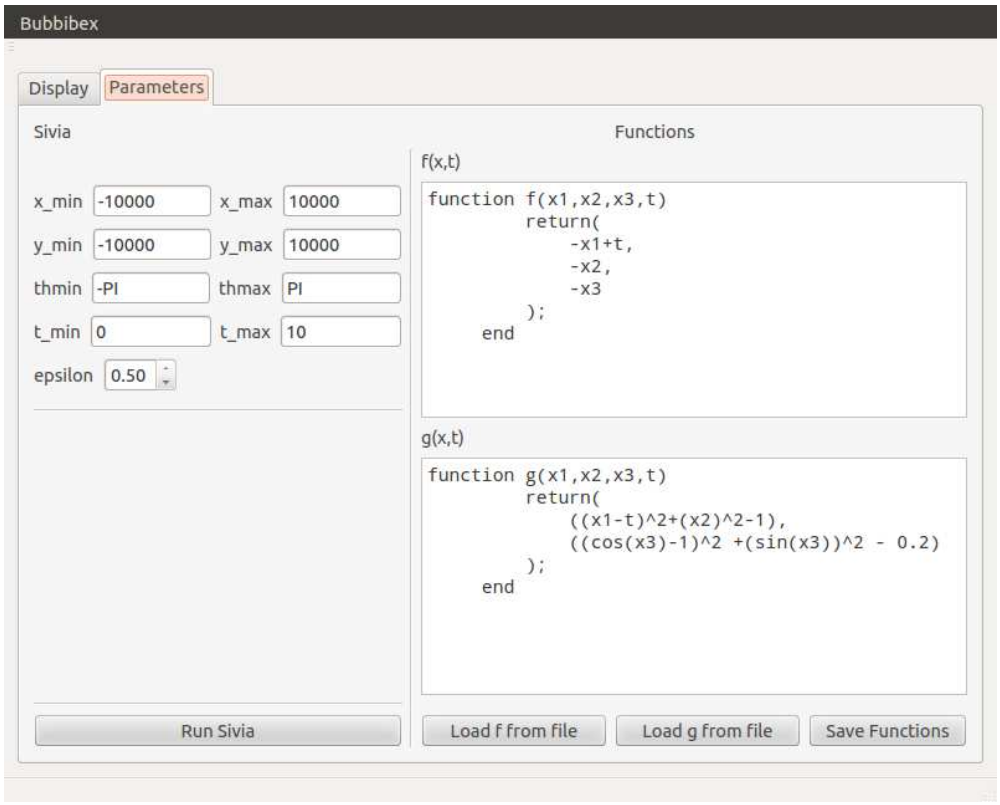
5.5.1 GUI

Our first task was to design a GUI.
According to the specifications, the GUI should have a display field, as well as checkbox to select what will be displayed and a track bar used to show results at different times. In addition, we had programmed zoom functions both with buttons and the mouse (see figure).



Display panel

We have also decided to add a parameter panel (see figure), which will improve the control on the Sivia's algorithm. For that purpose, the initial conditions of the algorithm can be set up. Furthermore, functions **f** and **g** can be rewritten in dedicated fields or load from existing files. Saving functions into files is imperative to run a simulation.



Parameter panel

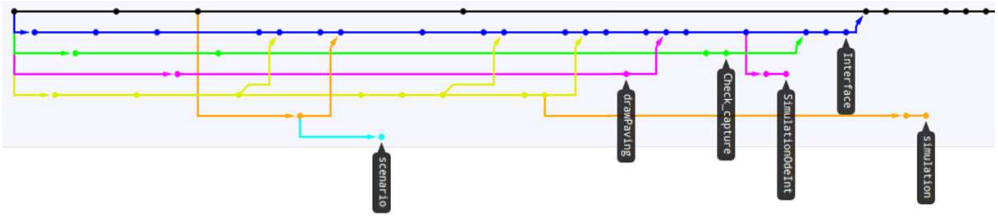
In addition, an informative window (see figure) pop-up on running Sivia's algorithm, returning informations on elapsed time, initial boxes, and informations about the number of unsafe boxes computed.

Information window

5.5.2 Integration

In order to integrate the different parts of the project, we use a git repository shared with all teams. Each group created a branch for its task, wrote its code and push into the server. When it was done,

we merge the branch in order to integrate different parts. The figure below shows the evolution of the project for the first days. After that, improvement was almost only brought to the master branch.



Development flow

As we were in charge of the GUI, we were also responsible for the global integration. In terms of programming, it's mainly in our part that classes were created functions called. We had so to see other teams and insure that their procedures will be compatible with our own graphic structures.

6 Results

When we considered the simplest example, sivia satified the expectations. Indeed, it informed if the system is safe or not. When we considered the two other example, BUBBIBEX does not allow to determine if the system is safe or not, even if we change the size of the bubble reduce the constraints of safety. This might comes from the fact that controller is not accurate, since BUBBIBEX gives good results with the simplest example.

References

[1] F. L. BARS AND L. JAULIN. An experimental validation of a robust controller with the VAIMOS autonomous sailboat. In "5th International Robotic Sailing Conference", pp. 74–84, Cardiff, Wales, England (2012). Springer.

[2] G. CHABERT. "IBEX 2.0, available at , `http://www.emn.fr/z-info/ibex/`". Ecole des mines de Nantes (2013).

[3] G. CHABERT AND L. JAULIN. Contractor Programming. *Artificial Intelligence* **173**, 1079–1100 (2009).

[4] J. A. DIT SANDRETTO, G. TROMBETTONI, D. DANEY, AND G. CHABERT. Certified calibration of a cable-driven robot using interval contractor programming. In F. THOMAS AND A. P. GRACIA, editors, "Computational Kinematics, Mechanisms and Machine Science", Spinger (2014).

[5] V. DREVELLE AND P. BONNIFAIT. High integrity gnss location zone characterization using interval analysis. In "ION GNSS" (2009).

[6] L. Jaulin. "Représentation d'état pour la modélisation et la commande des systèmes (Coll. Automatique de base)". Hermès, London (2005).

[7] L. Jaulin and F. L. Bars. An interval approach for stability analysis; Application to sailboat robotics. *IEEE Transaction on Robotics* **27**(5) (2012).

[8] L. Jaulin and E. Walter. Guaranteed nonlinear parameter estimation via interval computations. *Interval Computation* pp. 61–75 (1993).

[9] S. L. Menec. Linear differential game with two pursuers and one evader. In M. Breton and K. Szajowski, editors, "Advances in Dynamic Games", vol. 11, pp. 209–226 (2011).

[10] R. E. Moore. "Interval Analysis". Prentice-Hall, Englewood Cliffs, NJ (1966).

[11] J. Sliwka, F. L. Bars, O. Reynet, and L. Jaulin. Using interval methods in the context of robust localization of underwater robots. In "NAFIPS 2011", El Paso, USA (2011).