STABIBEX: V-Stability with IBEX 2.0

Groupe Robotique ENSTA-Bretagne 2013
P. Arnaudin, A. Bourquin, N. Brocheton, K. Bruget,
J. Cardon, Z. El Khallali, F. Jean, M. Kadanus, J. Nicola, M. Pinheiro, F. Rodier

15/02/2013

1. Project

The goal of the project is to develop a software called STABIBEX in order to study stability of robots. The software is programed in C++ with QT-CREATOR. It uses IBEX 2.0 [Cha13] to implement the principle of contractor programming [CJ09].

2. Theory

The robot to be considered satisfies a state equation [Jau05] of the form

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

With the controller $\mathbf{u} = \mathbf{g}(\mathbf{x})$, the robot becomes autonomous and now satisfies an equation of the form

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

Because of all uncertainties it is more realistic to represent the dynamics of the robot by

$$\dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x})$$

which is *a differential inclusion* [AF90]. The system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$$

is Lyapunov-stable (1892) [Kha02] is there exists $V(\mathbf{x}) \geq 0$ such that

$$\begin{array}{rcl} \dot{V}(\mathbf{x}) & < & 0 \text{ if } \mathbf{x} \neq \mathbf{0}, \\ V(\mathbf{x}) & = & 0 \text{ iff } \mathbf{x} = \mathbf{0}. \end{array}$$

This definition is now adapted in order to take into account the fact that the system is described by a differential inclusion [JB12].

Definition. Consider a differentiable function $V(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$. The system is (V, v^+)-stable if

$$\left(V(\mathbf{x}) \in [0, v^+] \;\Rightarrow\; \dot{V}(\mathbf{x}) < 0\right).$$

where $v^+ > 0$. An illustration is given on Figure 2.1 in the special case where $v^+ = +\infty$.

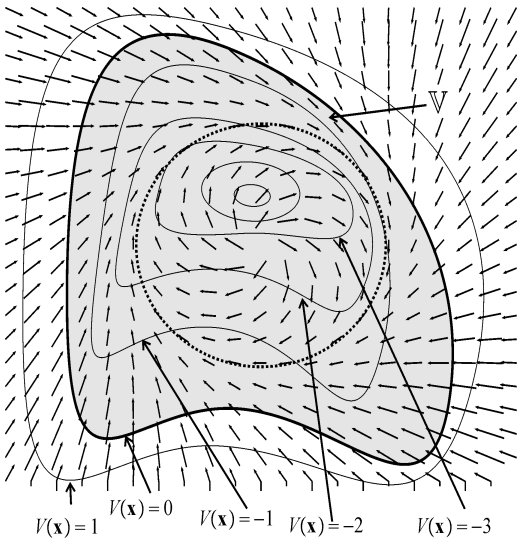


Figure 2.1: Vector field

Theorem. If the system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$ is (V, v^+)-stable then

(i) $\forall \mathbf{x}(0) \mid V(\mathbf{x}(0)) \leq v^+, \exists t \geq 0$ such that $V(\mathbf{x}(t)) < 0$
(ii) if $V(\mathbf{x}(t)) < 0$ then $\forall \tau > 0,\ V(\mathbf{x}(t+\tau)) < 0$.

The proof can be found in [JB12].

Theorem. We have

$$\begin{cases} \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{f}(\mathbf{x}) \geq 0 \\ V(\mathbf{x}) \in [0, v^+] \end{cases} \text{ inconsistent } \Leftrightarrow \dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \text{ is } V\text{-stable.}$$

Proof.

$$\begin{array}{ll} & \left(V(\mathbf{x}) \in [0, v^+] \ \Rightarrow\ \dot{V}(\mathbf{x}) < 0\right) \\ \Leftrightarrow & \left(V(\mathbf{x}) \in [0, v^+] \Rightarrow \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{f}(\mathbf{x}) < 0\right) \\ \Leftrightarrow & \forall \mathbf{x}, \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{f}(\mathbf{x}) < 0 \text{ or } V(\mathbf{x}) \notin [0, v^+] \\ \Leftrightarrow & \neg\left(\exists \mathbf{x}, \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{f}(\mathbf{x}) \geq 0 \text{ and } V(\mathbf{x}) \in [0, v^+]\right) \end{array}$$

Theorem. We have

$$\begin{cases} \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{a} \geq 0 \\ \mathbf{a} \in \mathbf{F}(\mathbf{x}) \\ V(\mathbf{x}) \in [0, v^+] \end{cases} \text{ inconsistent } \Leftrightarrow \dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}) \text{ is } V\text{-stable}$$

Or equivalently

$$\begin{cases} \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{a} \geq 0 \\ \mathbf{f}^+(\mathbf{x}) - \mathbf{a} \geq \mathbf{0} \\ \mathbf{a} - \mathbf{f}^-(\mathbf{x}) \geq 0 \\ V(\mathbf{x}) \in [0, v^+] \end{cases} \text{ inconsistent } \Leftrightarrow \dot{\mathbf{x}} \in \mathbf{F}(\mathbf{x}) \text{ is } V\text{-stable}$$

3. Software Stabibex

The goal of the project is to prove the (V, v^+)-stability for an uncertain system. For this purpose, STABIBEX shows that the following system has no solution:

$$\begin{cases} \frac{\partial V}{\partial \mathbf{x}}(\mathbf{x}).\mathbf{a} \geq 0 \\ \mathbf{f}^+(\mathbf{x}) - \mathbf{a} \geq \mathbf{0} \\ \mathbf{a} - \mathbf{f}^-(\mathbf{x}) \geq \mathbf{0} \\ V(\mathbf{x}) \in [0, v^+] \end{cases} \tag{3.1}$$

with $V : \mathbb{R}^n \to \mathbb{R}$, $\mathbf{f}^- : \mathbb{R}^n \to \mathbb{R}^n$, $\mathbf{f}^+ : \mathbb{R}^n \to \mathbb{R}^n$ and $\mathbf{a} \in \mathbb{R}^n$. In the software STABIBEX, we have restricted the system to $n = 2$. Functions V, $\mathbf{f}^-$, $\mathbf{f}^+$ have to be entered in a MINIBEX format [Cha13] and can be modified by the user. The software STABIBEX is able to:

Plot the differential inclusion. It draws on a grid the function $\mathbf{F}(\mathbf{x}) = [\mathbf{f}^-(\mathbf{x}), \mathbf{f}^+(\mathbf{x})]$ with $\mathbf{F} : \mathbb{R}^n \to \mathbb{IR}^n$.

Plot the function V. Level curves of V are displayed thanks to SIVIA. Sets $V^{-1}(\mathbb{R}^-)$ and $V^{-1}([0, v^+])$ are displayed in different colors.

Simulate. An initial condition is selected randomly and a simulation is done. Then the trajectory is displayed on the screen.

Resolution. From constraints (3.1), we make four contractors which are combined. If contractors result in the empty space, the proof of stability is made. If not, the bisections are in the space of $\mathbf{x}$ and the corresponding blocks are displayed.

4. Decomposition of the tasks

All groups who wrote code initially began with the same program `SiviaSimple`.

The group Interface (Pinhiero De Carvalho/ El Khallali) made the main program with the interface (buttons, trackbars, etc). The group has also performed the integration of the whole project. Four functions are called by the interface `DrawF`, `DrawV`, `Simu`, `Solve`.

The group DrawF (Arnaudin/Bourquin), made the plots the vector field of **F**.

The group DrawV (Murillo), made the plots of the level curves of **V**.

The group Simu (Brocheton/Bruget) did the simulation.

The group Solve (Cardon/Nicola) made the inteval resolution.

The group Test (Jean/Rodier) prepared some tests.

Finally, **each group** wrote its part of the report.

5. Interface task

During the project `Stabibex`, the work was divided into different activities. We have chosen to focus on the interface of project. The project had initially a very simple interface which allowed the user to observe the image, change the observation function and restart the program. Various functions have been added:

A button "DrawF": calls the function to draw a grid on the differential inclusion F and displays the vector fields on the Interface.

A button "DrawV": calls the function to trace the differential inclusion V and displays the V curves by Sivia on the interface.

A check-box "trajectory": can draw a trajectory from the simulation if the check box is checked.

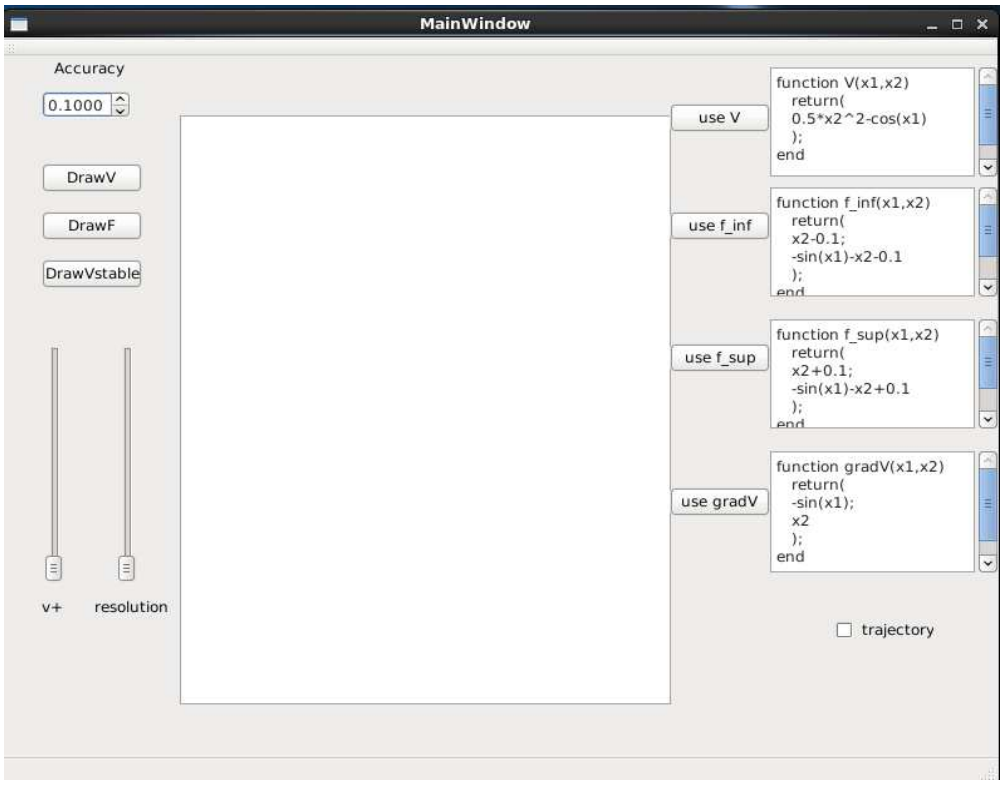


Figure 5.1: Stabibex's Interface

Four text boxes: allows the user to enter the functions "f_inf" "f_sup" "V" and "gradV" and to recover them in the format .txt after clicking on the respective buttons quads "use V" "use f_inf "" f_sup use "and" use grav_V". These functions, once recovered . txt format will be used for the simulation.

A button "Draw V-stable" in the case when the proof of stability is checked, this function plots the bisections in space x and corresponding pavement are displayed on the interface.

Track-bar "resolution": Sets the resolution when plotting the differential inclusion F.

Track-bar "V +": allows to increment or decrement the parameter v bar of the V function to be tested.

The figure 5.1 shows us the final interface of the program. We'll explain how it works with more details below.

All calls received from the interface had their functions in the file 'mainwindow.cpp'. The buttons of the right are where we can find the functions $V(\mathbf{x})$, $f^-(\mathbf{x})$, $f^+(\mathbf{x})$ and grad$V(\mathbf{x})$, where $\mathbf{x} = [x_1, x_2]^{\mathrm{T}}$ a vector used in the program. This configuration allows the user to change the equations easily, in case he doesn't want to use the ' pendulum' example as initially suggested by the program. After writing the chosen equations, the user must click in each button beside the boxes to load them so their specific files. So they can be written by the different parts of the program. The user must keep the format used :

```
function name(x1,x2)
return(
//write equation here
);
end
```

We can see in Figure 5.2 the function that is called when the button corresponding to the V function is clicked.

We can see that `a .txt` file is created and it receives the desired function.

```
void MainWindow::on_pushButtonV_clicked()
{
    QString s1 = ui->plainTextEditV->toPlainText();

    QFile file("V.txt");
        if (!file.open(QIODevice::WriteOnly | QIODevice::Text))
            return;

        QTextStream out(&file);
        out << s1 << "\n";
        file.close();
}
```

Figure 5.2: Code for the V Button

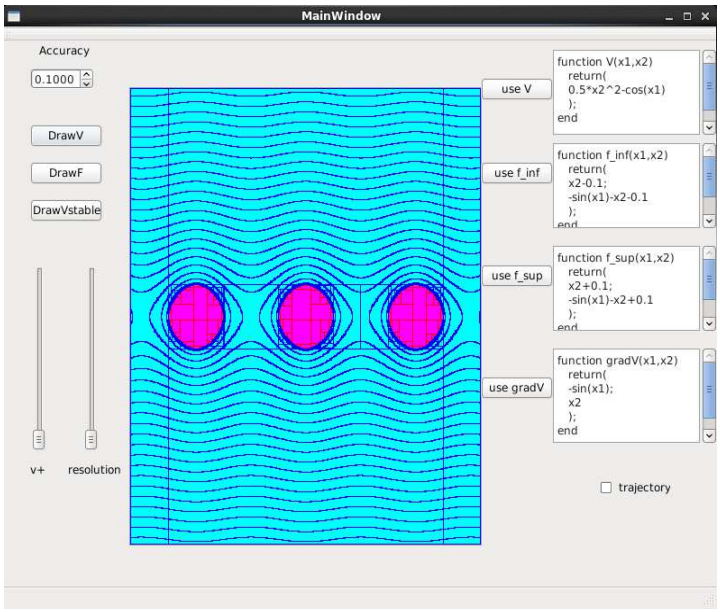


Figure 5.3: Result of DrawV Button

All buttons call functions very similar to this one. Because some actions must happen only when the buttons are clicked. One equations have been loaded, the next step is to choose the method to be applied in each case. The user can choose between three different buttons. We created different classes and libraries to implement the buttons . Inside each one of these functions the specific classes were called to implement the action desired.

DrawV : Draws the graphic corresponding to the function V: `Drawv tmp(v, *frame, epsilon);`. In Figure 5.3 X, we can see the generated screenshot.

DrawF : Draws the vector field via the function F : `Drawf(*frame,xmin,xmax,ymin,ymax, ui->verticalSliderResolution-` An illustration is given by Figure 5.4.

When called after the V-function, STABIBEX superposes both displays as illustrated by Figure 5.5.

The scrollbar called 'resolution' allows the user to increase, or decrease the size of the vector field (see Figure 5.5 and compare it with Figure 5.3). DrawVstable draws the result of a test with the V function. As we can see in the figure 5.6.

Using the scrollbar 'v+', we can increase, or decrease the value of the v bar in the V function. We can see the result of the increasing of this value in the next image: In the corner of the right there is a checkbox called trajectory. When

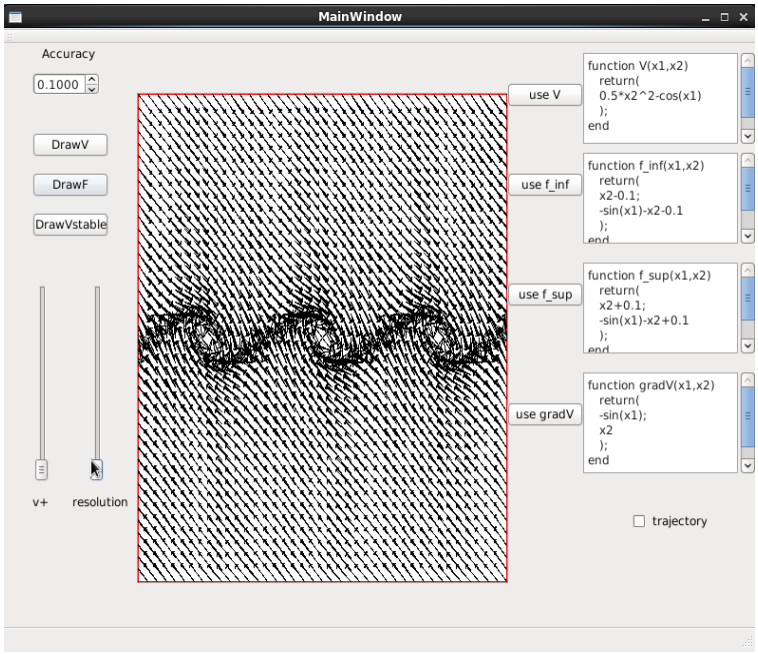


Figure 5.4: Result of DrawF Button

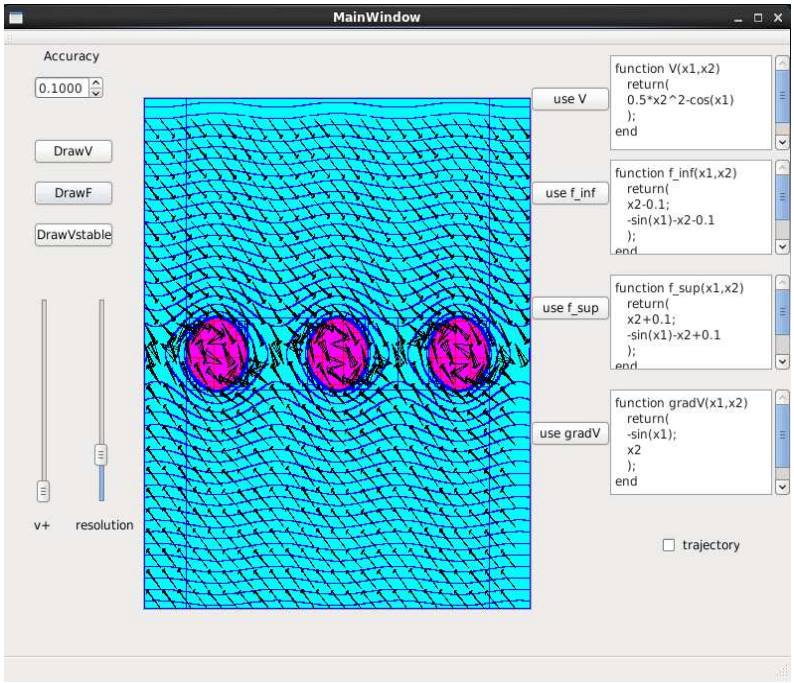


Figure 5.5: Both function on the same frame

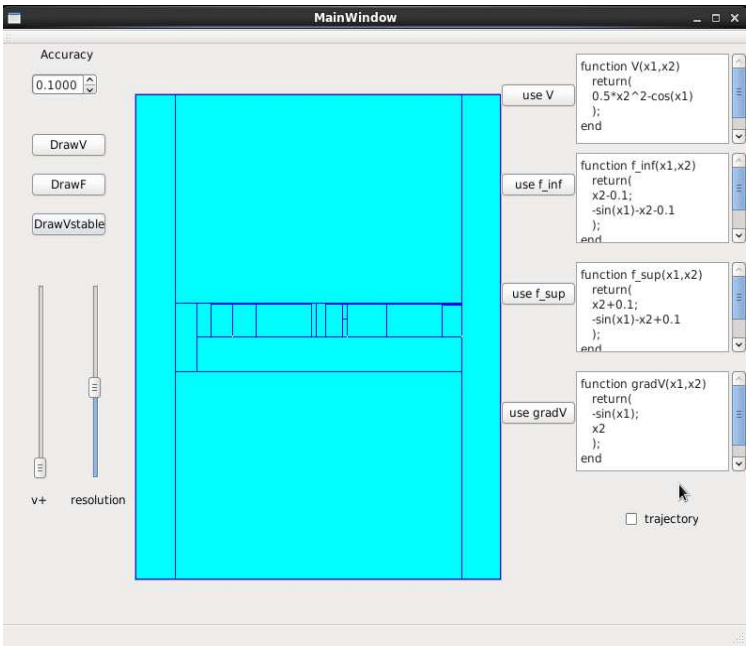


Figure 5.6: Result of DrawVstable Button

clicked, the user is allowed to choose some (x,y) position clicking the mouse in the image to design the trajectory of some robot going to the final position. This part of the function to receive the mouse events is responsible to catch these coordinates.

```
if(ui->trajc->isChecked()){
            double xM = (double) (mouseEvent->x()-140)/(391/20) -10;
            double yM = (double) -(mouseEvent->y()-50)/(491/20) +10;
            Simu simulation("f_inf.txt","f_sup.txt",*frame,xM,yM);
```

Figure 5.7: Code catching coordinate of click of the mouse

Figure 5.8 shows the result of this implementation.

6. DrawF task

6.1. Presentation

The aim of this task is to plot the interval vector field $\mathbf{F}(\mathbf{x}) = [\mathbf{f}^-(\mathbf{x}), \mathbf{f}^+(\mathbf{x})]$ with $\mathbf{F} : \mathbb{R}^2 \to \mathbb{IR}^2$. For each point of a grid defined by discretizing $\mathbb{R}^2$, four vectors are drawn as presented on the following scheme. Each coordinate of the vectors have a minimam and a maximal max value. One arrow is drawn for each case. See Figure 6.1 for an illustration.

6.2. Code explanation

To implement this functionality, we start by defining the grid with two loops. For each point of this grid, we evaluate functions ($f^-(\mathbf{x})$ and $f^+(\mathbf{x})$) and then we normalize the vectors before plotting them.

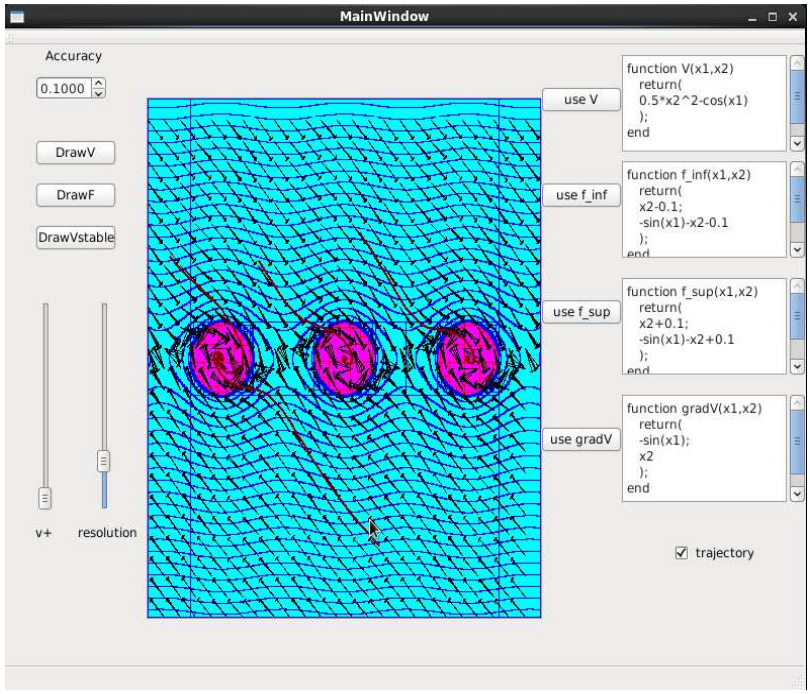


Figure 5.8: Trajectories implementation

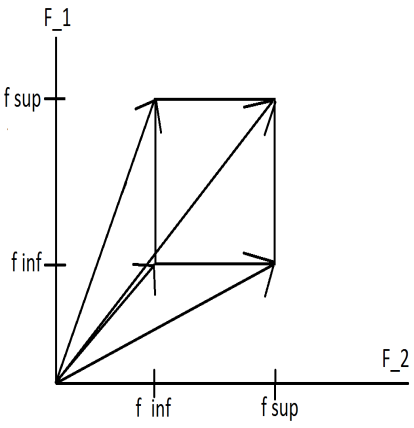


Figure 6.1: Explanatory diagram for drawing arrows

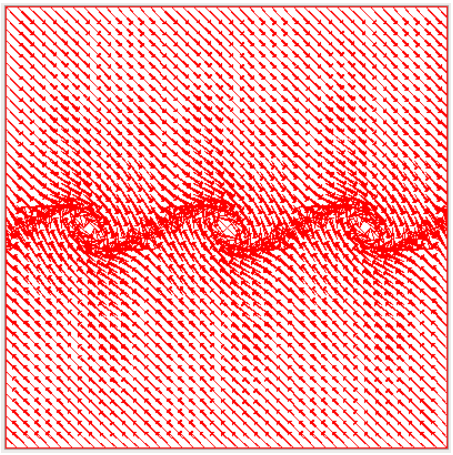


Figure 6.2: Field vectors of F (simple pendulum)

Figure 6.2 represents the plot of the vector fields for the following functions:

$$\begin{aligned} \mathbf{f}^{-}(\mathbf{x}) &= \begin{pmatrix} x_2 + \alpha_1^{-} \\ -\sin x_1 - x_2 + \alpha_2^{-} \end{pmatrix} \\ \mathbf{f}^{+}(\mathbf{x}) &= \begin{pmatrix} x_2 + \alpha_1^{+} \\ -\sin x_1 - x_2 + \alpha_2^{+} \end{pmatrix}. \end{aligned}$$

We notice that the result seems to be consistent. Indeed, we can observe three focal points that are three stable points for the simple pendulum (π, 0 and – π). The direction of arrows seems also to be coherent with equations of pendulum. Furthermore, our result is in adequacy with the plot of V-gradient. To integrate our program in the project, we implemented it in a new class called `drawf`. Moreover, during the integration, we added a scrollbar to change the number of arrows by modifying the step of our loops.

7. DrawV task

The `DrawV` function draws in the screen the line curves of the mathematical function V inserted in the box first box of the Interface located in the top right. This function can be executed by clicking in the button named `DrawV` located on the top left of the interface. One example of the result of this function can be viewed in Figure 7.1.

The algorithm of this function can be viewed in the figure 7.2. It uses the functions of the library IBEX (Interval-Based Explorer) to make the computation and uses the libraries of the QT-creator to draw the lines in the screen.

To calculate the line curves of a mathematical function, the `DrawV` function is based on the algorithm *Set Inverter via Interval Analysis* (SIVIA) [JW93] used with interval contractors. These contractor use interval analysis [Moo66] to contract boxes of the research space without loosing any solutions. The lines drawn in blue (with the background in light blue) corresponds to the set $V^{-1}([0, v^{+}])$, while the lines painted yellow (with the background in light red) corresponds to the set $V^{-1}(\mathbb{R}^{-})$. For the first set, the function uses the following contractor associated with the equation sin(10∗sqrt(V)) = 0. For the second set, the function uses two contractors. The first one is associated to$V < 0$ and the second contractor to $V > 0$. The function is dependent of the value of the accuracy box. Smaller is the value of the accuracy, better and more precise is the final result, but more time is necessary to execute this function. The function will only show anything in the screen after it finish all of the calculations (depending on the value of the accuracy box and the power of calculation of the machine, it may takes a few seconds to finish).

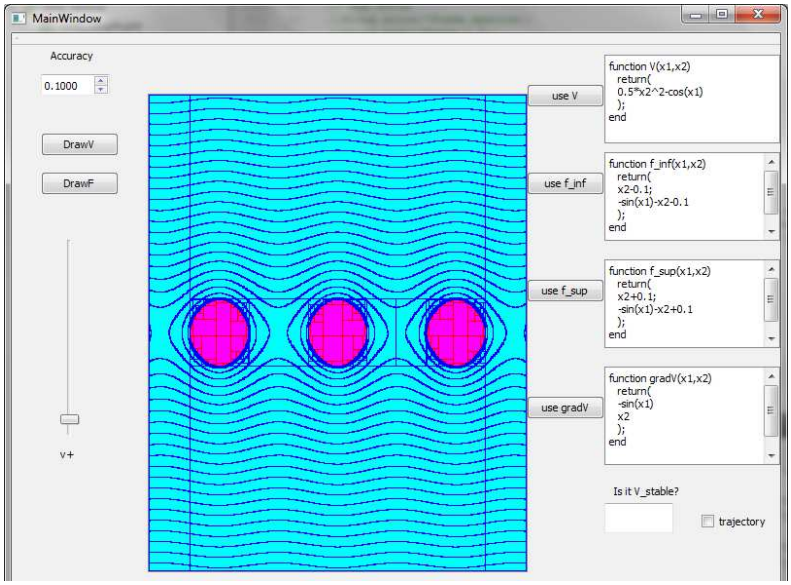


Figure 7.1: The result of the function DrawV for the pendulum equation

```
Drawv::Drawv(Function func, Frame& frame, double epsilon)
{
    //Declare the variables x1 and x2
    Variable x1,x2;

    //Create the NumConstrait for func(x1,x2) < 0
    NumConstraint cInside_1(x1,x2, func(x1,x2)<0);
    NumConstraint cOutside_1(x1,x2, func(x1,x2)>0);

    //Create the Forward Backward Constraints
    CtcFwdBwd t1 (cInside_1);
    CtcFwdBwd t2 (cOutside_1);
    //Execute Sivia with the created constraints and draw the lines
    Sivia captureSet(frame, epsilon, t1,t2, QColor(Qt::blue), QColor(Qt::cyan),
                     QColor(Qt::red), QColor(Qt::magenta), QColor(Qt::blue),  QColor(Qt::blue)  );

    //Create the NumConstrait for func(x1,x2) > 0
    NumConstraint c4(x1, sqr(x1)>-1);
    NumConstraint c2(x1,x2, sin(10*sqrt(func(x1,x2)))=0);
    //Create the Forward Backward Constraints
    CtcFwdBwd t3 (c2);
    CtcFwdBwd t4 (c4);
    //Execute Sivia with the created constraints and draw the lines
    Sivia LevelSet(frame, epsilon, t3,t4, QColor(Qt::transparent), QColor(Qt::transparent),
                   QColor(Qt::transparent), QColor(Qt::transparent), QColor(Qt::blue),  QColor(Qt::blue)  );

}
```

Figure 7.2: Code of the fonction wrote in C++

8. Simulation task

The aim of this part is to realize a simulation function which draws, from a randomly chosen point on the figure, its trajectory.

8.1. Code explanation

In order to allow this feature, two functions were created. A first in the `sivia.cpp` file called `simu` which takes as parameters a point of the current frame, calculates then draws the course of the point. The second function in `mainwindow.cpp` file is a `mouseEvent` function which catches the coordinates of a point designated by the current position of the mouse and then calls the first function. The final integration part will be the creation of a check-box in order to activate, or switch off, the drawing of the course. In order to implement the simulation function, a time step is first declared. Then, function `f_inf` and `f_sup` are implemented by calling them from their files created in specific folders into the build directory. Next we defined all the needed variables, mainly the current position vector and the IntervalVector F which will represent the estimation on the point's position.

The principal action of the function is defined in a loop which called our two functions to define the bottom left corner and the up right corner of the box F which correspond to the worst approximation, respectively the lowest and the highest cases. In order to respect the parameters expected by the Ibex functions, we use intervals even if in reality they are simple singletons. So we take the middle of each intervals to get the coordinates of each point. With these two points we create a trust interval on each axis which match the estimated area of the real position of the point. Next, we randomly choose a point into this area to calculate its next position on the trajectory and then draw the line between the current and the next positions. Finally, we replace the current position by the new one and run again the algorithm.

The mouse event function is realized when the user leftclicks on the main frame. In order to get the start position of the course, we catch the current mouse position in the window, transform it to be into the frameview and next alter it to obtain a position in the correct coordinate system.

8.2. Results

The function has been checked and tested in order to validate its behavior. The displayed results in figure 8.2 seem consistent with theory and expected trajectory for a pendulum.

9. Solve task

This part's purpose is to confirm or not the V-stability of a system. To do so we defined here the contractors that verify inconsistency of the constraints' system (3.1).

9.1. Code explanation

Before everything, we define the variables we want to contract : the position x_1, x_2, and the function a_1, a_2. Then we implement the constraints enumerated previously (NumConstraint). Then we transform those into the contractors Forward-Backward (CtcFwdBwd) that we compose or unify according of the system (`CtcUnion` & `CtcCompo`). It is now time to define the box to contract. We choose to define a 4 dimension's box (one per variable in the same order as they are defined). Lastly we use the classic SIVIA algorithm (with stack and cutting the largest first) to contract the box.

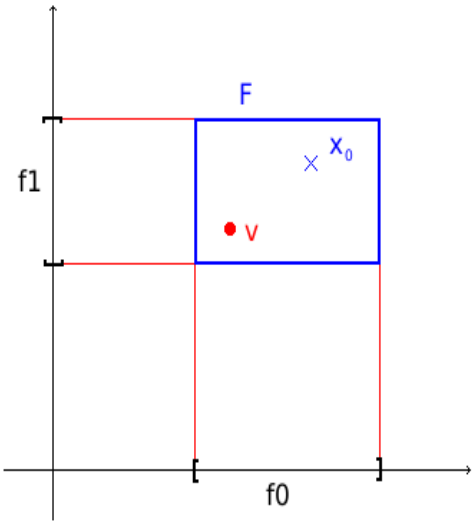


Figure 8.1:

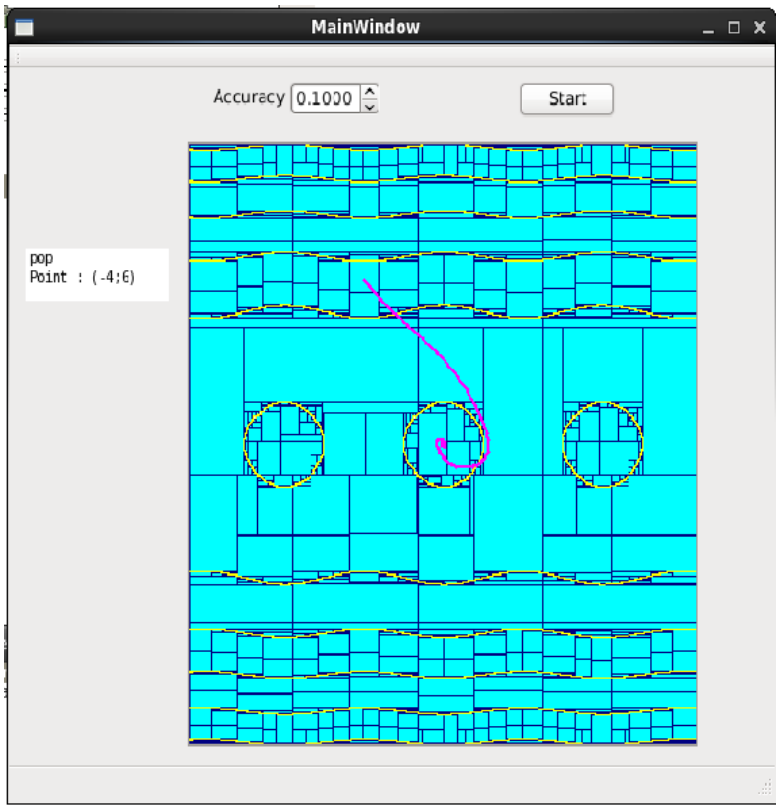


Figure 8.2: Result of Trajectory Simulation

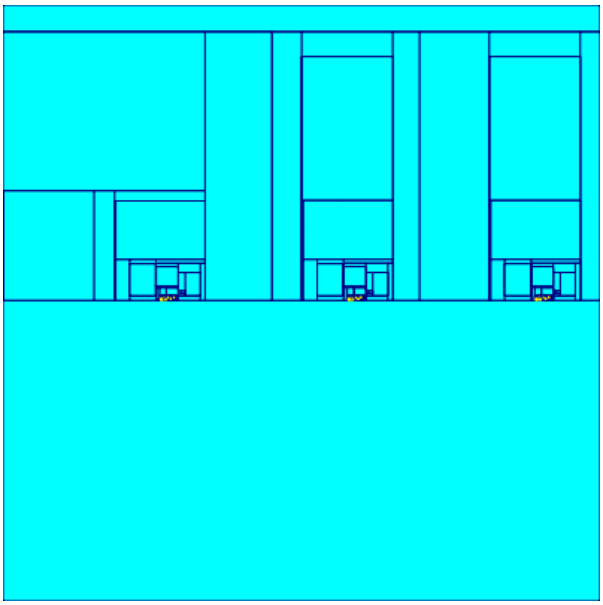


Figure 9.1: Paving of Pendulum System

9.2. Results

We can observe on the example of the pendulum that we find a nor stable or unstable zone that correspond to the state the pendulum is to the vertical ($x_1 = (2k+1) * \pi$) without velocity. This zone is proportional to α_1 and α_2, the uncertainty of the function.

As a result, using set-membership methods, we can't conclude about the V-Stability of a single pendulum.

10. Test task

Some examples were taken in order to test our software. The first two examples are simple and allow us to deduce stability easily. Then, we have dealt with the pendulum system, with mass-spring system and finally the Van Der Pol oscillator.

10.1. A simple V-stable system

First, we created a simple stable system described by the equation:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1^2 \\ -x_2 \end{pmatrix}$$

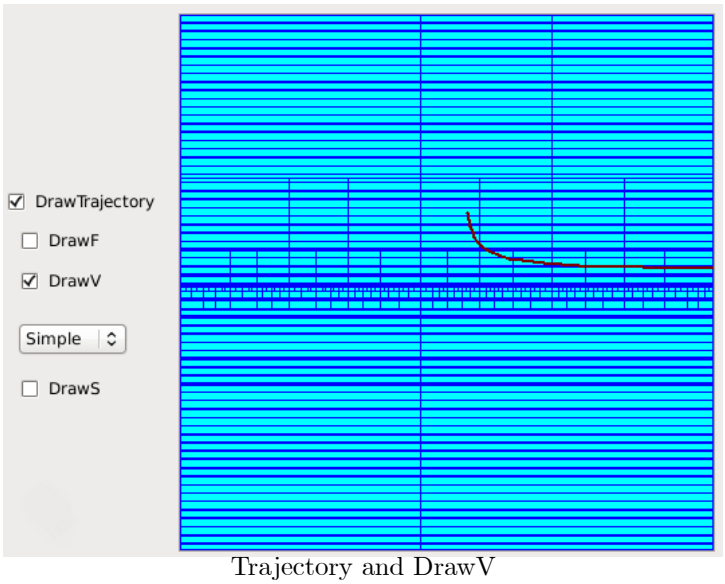
and chose the following function V to get the stability:

$$V(x) = x_2^2$$

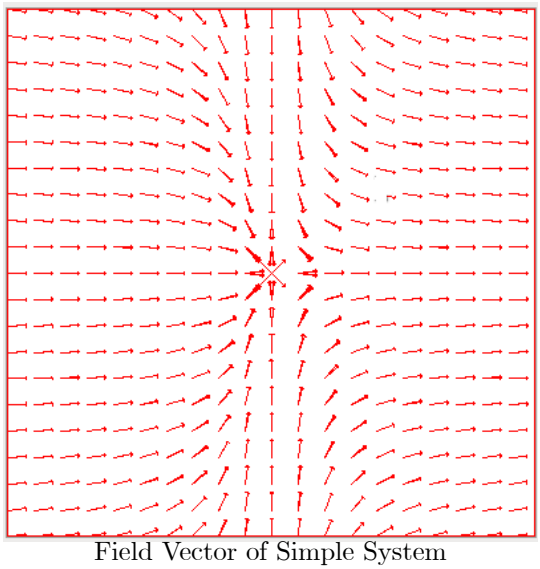
In STABIBEX, we take:

$$\begin{aligned} V(\mathbf{x}) &= x_2 + v^+ \\ \mathbf{f}^-(\mathbf{x}) &= \begin{pmatrix} x_1^2 + \alpha_1^- \\ -x_2 + \alpha_2^- \end{pmatrix} \\ \mathbf{f}^+(\mathbf{x}) &= \begin{pmatrix} x_1^2 + \alpha_1^+ \\ -x_2 + \alpha_2^+ \end{pmatrix}. \end{aligned}$$

We plot a random trajectory and the vector field:



Trajectory and DrawV


Field Vector of Simple System

According to the vector field, we can presume that our system will be V-stable. Indeed, trajectories seem to converge to the line $y = 0$. Finally by testing our system with the software, nothing appear in the window. We can conclude that our system is V-stable.

10.2. A V-unstable system

We created an system described by the equation:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ -x_2 \end{pmatrix},$$

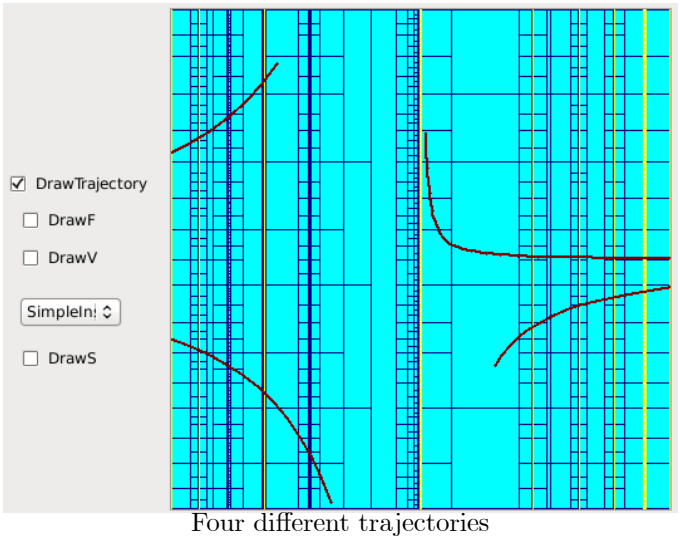
We have chosen the following V function:

$$V(x) = x_1^2.$$

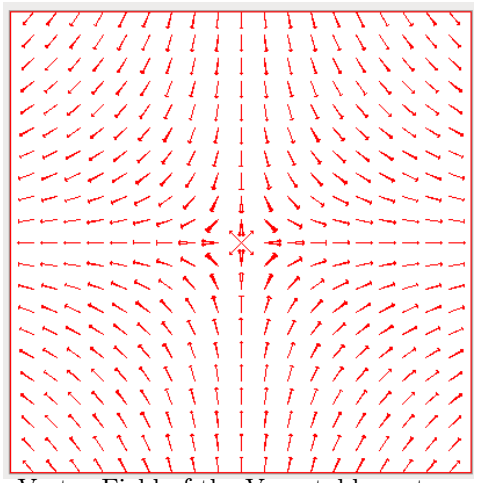
Using STABIBEX, we take:

$$\begin{aligned} V(\mathbf{x}) &= x_1^2 + \bar{v} \\ \mathbf{f}^-(\mathbf{x}) &= \begin{pmatrix} x_2 + \alpha_1^- \\ -x_1 + \alpha_2^- \end{pmatrix} \\ \mathbf{f}^+(\mathbf{x}) &= \begin{pmatrix} x_2 + \alpha_1^+ \\ -x_1 + \alpha_2^+ \end{pmatrix}. \end{aligned}$$

We have to choose v^+ so that the system always converges to a negative value.

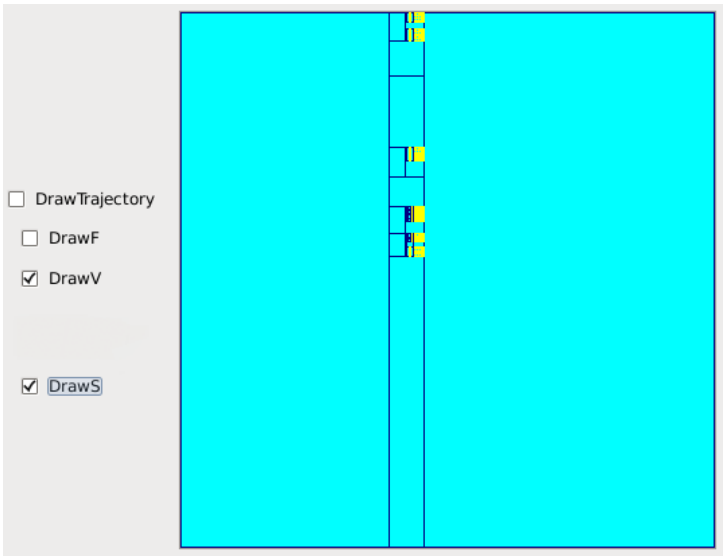


Four different trajectories



Vector Field of the V-unstable system

Oncer STABIBEX terminates we got a figure showing that there is actually some bisections. To conclude, the system is not V-stable.



Program solving

As a result, using set-membership methods, we cannot conclude about the V-Stability of these system.

10.3. Pendulum

The pendulum is represented by the following equations:

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} x_2 + \alpha_1 \\ -\sin x_1 - x_2 + \alpha_2 \end{pmatrix},$$

where x_1 is the angular displacement (x_1 is equal to zero when the pendulum is in equilibrium state) and x_2 is the angular velocity. In these equations, $\alpha_i \in [-0.1, 0.1]$ are disturbances (on masses, friction, the wind, ...). The mechanical energy of the pendulum is:

$$V(x) = \underbrace{\frac{1}{2}x_2^2}_{\text{kinetic energy}} + \underbrace{-\cos x_1}_{\text{potential energy}}$$

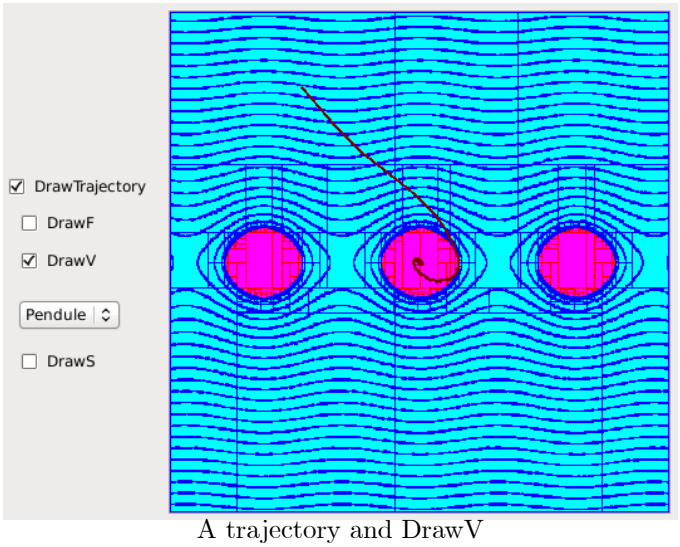
We note that:

$$\begin{aligned} \frac{dV}{dt} &= \dot{x}_2 x_2 + \dot{x}_1 \sin x_1 = (-\sin x_1 - x_2 + \alpha_2) x_2 + (x_2 + \alpha_1) \sin x_1 \\ &= -x_2^2 + \alpha_2 x_2 + \alpha_1 \sin x_1 \end{aligned}$$

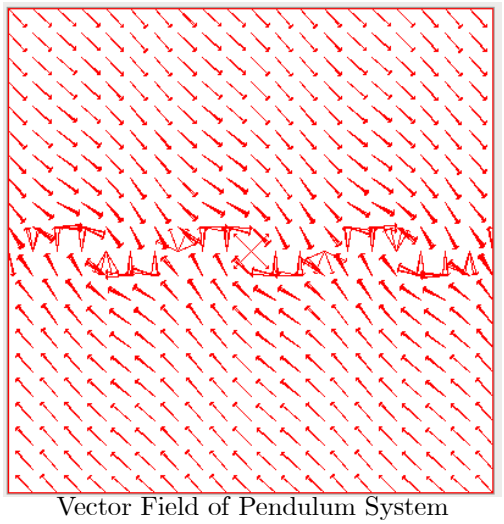
In STABIBEX, we will type:

$$\begin{aligned} V(\mathbf{x}) &= \frac{1}{2} * x_2^2 - \cos x_1 + \bar{v} \\ \mathbf{f}^-(\mathbf{x}) &= \begin{pmatrix} x_2 + \alpha_1^- \\ -\sin x_1 - x_2 + \alpha_2^- \end{pmatrix} \\ \mathbf{f}^+(\mathbf{x}) &= \begin{pmatrix} x_2 + \alpha_1^+ \\ -\sin x_1 - x_2 + \alpha_2^+ \end{pmatrix}. \end{aligned}$$

We have to choose v^+ so that the system always converges to a negative value. We plot a random trajectory and the vector field in the Figure

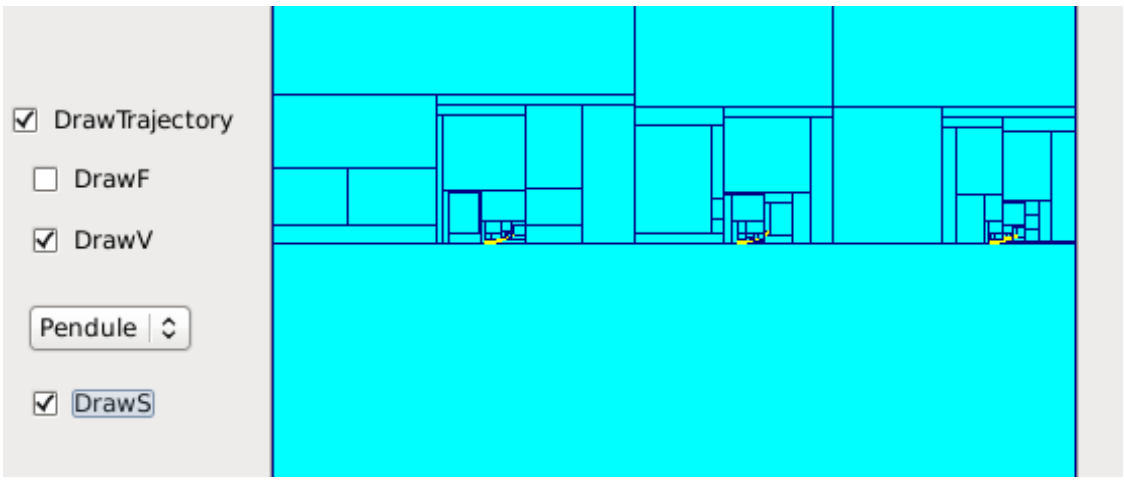


A trajectory and DrawV



Vector Field of Pendulum System

Thanks to the program, we have tested this system and found that, using set-membership methods, we cannot conclude about the V-Stability of a single pendulum.



10.4. Mass-spring system

This system can be described by the equation:

$$\frac{d^2x}{dt^2}(t) + w_0^2 * x(t) = 0$$

By taking the state vector

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \dot{x} \\ x \end{pmatrix}$$

we get

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -w_0^2 * x_2 + \alpha_1 \\ x_1 + \alpha_2 \end{pmatrix},$$

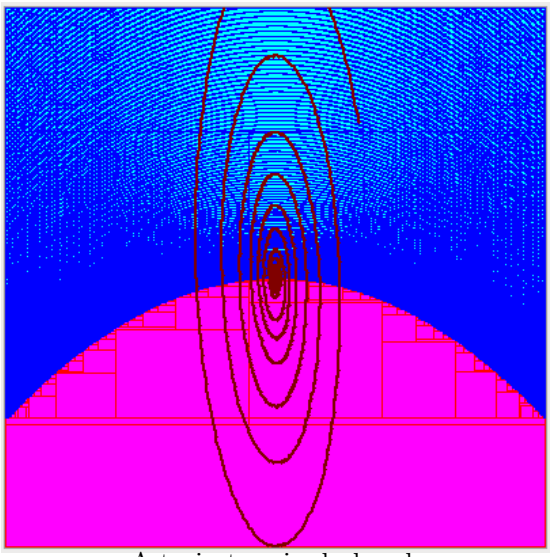
where x_2 is the position of the mass, x_2 its velocity and w_0 the angular frequency of the harmonic oscillator which is $\sqrt{\frac{k}{m}}$ (k is the spring constant and m the mass). In these equations, $\alpha_i \in -0.1, 0.1$ are disturbances. The mechanical energy of the system is:

$$V(x) = \underbrace{\frac{1}{2}mx_1^2}_{\text{kinetic energy}} + \underbrace{mgx_2 + \frac{1}{2}kx_2^2}_{\text{potential energy}}$$

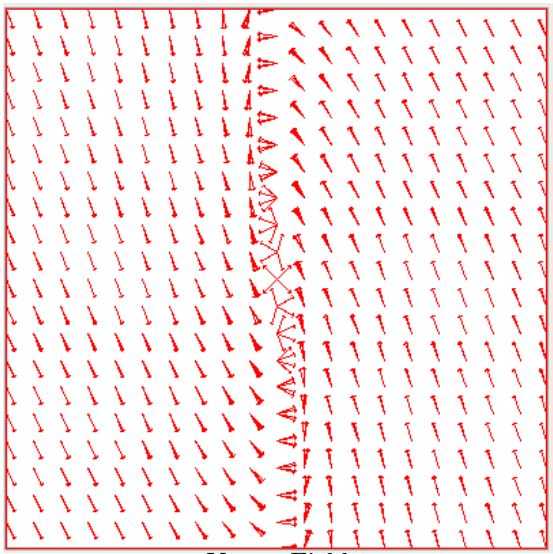
For STABIBEX, we take:

$$\begin{aligned} V(\mathbf{x}) &= \frac{1}{2} * m * x_1^2 + m * g * x_2 + \frac{1}{2} * k * x_2 + \bar{v} \\ \mathbf{f}^-(\mathbf{x}) &= \begin{pmatrix} -w_0^2 * x_2 + \alpha_1^- \\ x_1 + \alpha_2^- \end{pmatrix} \\ \mathbf{f}^+(\mathbf{x}) &= \begin{pmatrix} -w_0^2 * x_2 + \alpha_1^+ \\ x_1 + \alpha_2^+ \end{pmatrix}. \end{aligned}$$

We have to choose v^+ so that the system always converges to a negative value. We can plot a random trajectory and the differentiable inclusion.



A trajectory in dark red.



Vector Field

According to the software, contractors lead to the empty space, so the mass-spring system is V-stable.

10.5. Van der Pol oscillator

Van der Pol oscillator is a non-conservative oscillator with non-linear damping. It evolves in time according to the second order differential equation:

$$\ddot{x}(t) + \lambda(x^2(t) - 1)\dot{x}(t) + x(t) = 0$$

where x is the position coordinate and λ is a scalar parameter indicating the nonlinearity and the strength of the damping. We take $\lambda = 0.5$. By taking the state vector

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \dot{x} \\ x \end{pmatrix},$$

we have

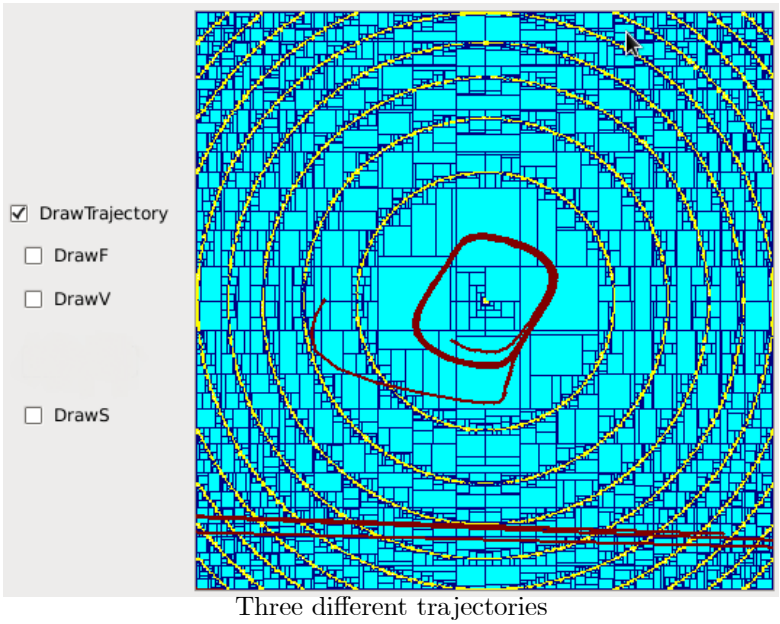
$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} -x_2 + 0.5(x_2^2 - 1)x_1 + \alpha_1 \\ x_1 + \alpha_2 \end{pmatrix}.$$

As a Lyapunov function we have taken

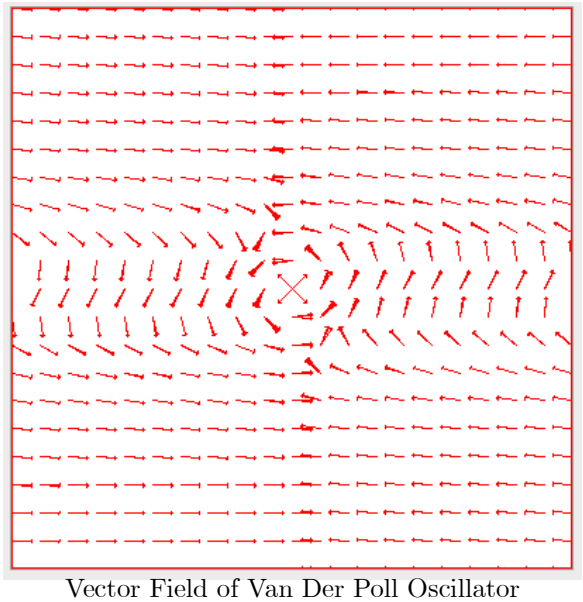
$$V(x) = 0.5(x_1^2 + x_2^2)$$

In STABIBEX, we take

$$\begin{aligned} V(\mathbf{x}) &= 0.5 * (x_1^2 + x_2^2) + \bar{v} \\ \mathbf{f}^-(\mathbf{x}) &= \begin{pmatrix} -x_2 + 0.5 * (x_2^2 - 1) * x_1 + \alpha_1^- \\ x_1 + \alpha_2^- \end{pmatrix} \\ \mathbf{f}^+(\mathbf{x}) &= \begin{pmatrix} -x_2 + 0.5 * (x_2^2 - 1) * x_1 + \alpha_1^+ \\ x_1 + \alpha_2^+ \end{pmatrix}. \end{aligned}$$



Three different trajectories


Vector Field of Van Der Poll Oscillator

The software gives contractors equal to empty space, so theVan Der Pol system is V-stable.

References

[AF90] J.P. Aubin and H. Frankowska. *Set-Valued Analysis*. Birkhäuser, Boston, 1990.

[Cha13] G. Chabert. *IBEX 2.0, available at* , `http://www.emn.fr/z-info/ibex/`. Ecole des mines de Nantes, 2013.

[CJ09] G. Chabert and L. Jaulin. Contractor Programming. *Artificial Intelligence*, 173:1079–1100, 2009.

[Jau05] L. Jaulin. *Représentation d'état pour la modélisation et la commande des systèmes (Coll. Automatique de base)*. Hermès, London, 2005.

[JB12] L. Jaulin and F. Le Bars. An interval approach for stability analysis; Application to sailboat robotics. *IEEE Transaction on Robotics*, 27(5), 2012.

[JW93] L. Jaulin and E. Walter. Guaranteed nonlinear parameter estimation via interval computations. *Interval Computation*, pages 61–75, 1993.

[Kha02] H.K. Khalil. *Nonlinear Systems, Third Edition*. Prentice Hall, 2002.

[Moo66] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1966.