Summer school on applied interval analysis[1]

Luc Jaulin
Laboratoire E3I2,
ENSIETA,
2, rue François Verny,
29806 Brest, Cedex 9
luc.jaulin@ensieta.fr
http://www.ensieta.fr/e3i2/Jaulin/
Tel : +33 (0)2 98 34 89 10

September 17, 2005

[1]This document is related to my slides written for the summer school on the applications of set computation that took place in Grenoble. It is based on the book *Applied Interval Analysis* by L. Jaulin, M. Kieffer, O. Didrit and E. Walter. New applications have also been considered. The name of the collaborators involved in these new applications are recalled in these document.

Basic notions on set theory

(Luc Jaulin, Monday, 10h15-10h45)

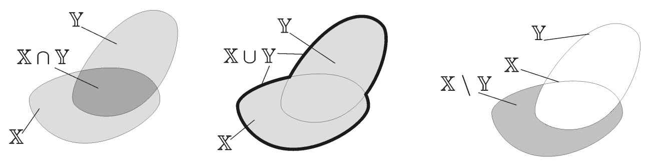
Basic operation on sets

Consider two sets $\mathbb{X}$ and $\mathbb{Y}$. We define

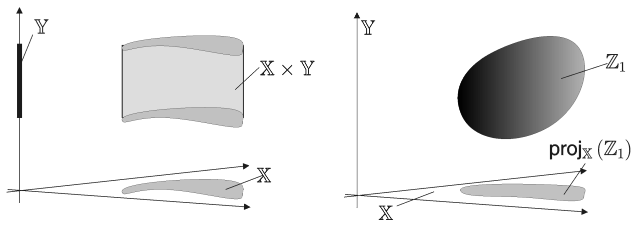
$$\begin{array}{rclc} \mathbb{X}\cap\mathbb{Y} & \overset{\text{def}}{=} & \{x \mid x\in\mathbb{X} \text{ and } x\in\mathbb{Y}\} & \text{(intersection)}\\ \mathbb{X}\cup\mathbb{Y} & \overset{\text{def}}{=} & \{x \mid x\in\mathbb{X} \text{ or } x\in\mathbb{Y}\} & \text{(union)}\\ \mathbb{X}\setminus\mathbb{Y} & \overset{\text{def}}{=} & \{x \mid x\in\mathbb{X} \text{ and } x\notin\mathbb{Y}\} & \text{(deprivation)}\\ \mathbb{X}\times\mathbb{Y} & \overset{\text{def}}{=} & \{(x,y) \mid x\in\mathbb{X} \text{ and } y\in\mathbb{Y}\} & \text{(Cartesian product)} \end{array} \tag{1}$$

If $\mathbb{Z}=\mathbb{X}\times\mathbb{Y}$, then the *projection* of a subset $\mathbb{Z}_1$ of $\mathbb{Z}$ onto $\mathbb{X}$ (with respect to $\mathbb{Y}$) is defined as

$$\text{proj}_{\mathbb{X}}(\mathbb{Z}_1) \overset{\text{def}}{=} \{x\in\mathbb{X} \mid \exists y\in\mathbb{Y} \text{ such that } (x,y)\in\mathbb{Z}_1\}. \tag{2}$$



$\mathbb{Y}$
$\mathbb{X}\cap\mathbb{Y}$
$\mathbb{X}$
$\mathbb{Y}$
$\mathbb{X}\cup\mathbb{Y}$
$\mathbb{X}$
$\mathbb{Y}$
$\mathbb{X}$
$\mathbb{X}\setminus\mathbb{Y}$



$\mathbb{Y}$
$\mathbb{X}\times\mathbb{Y}$
$\mathbb{X}$
$\mathbb{Y}$
$\mathbb{Z}_1$
$\text{proj}_{\mathbb{X}}(\mathbb{Z}_1)$
$\mathbb{X}$

Example 1: If $\mathbb{X}=\{a,b,c,d\}$ and $\mathbb{Y}=\{b,c,x,y\}$, then

$$\begin{array}{rcl} \mathbb{X}\cap\mathbb{Y} & = & \{b,c\}\\ \mathbb{X}\cup\mathbb{Y} & = & \{a,b,c,d,x,y\}\\ \mathbb{X}\setminus\mathbb{Y} & = & \{a,d\}\\ \mathbb{X}\times\mathbb{Y} & = & \{(a,b),(a,c),(a,x),(a,y),\dots,(d,b),(d,c),(d,x),(d,y)\}. \end{array} \tag{3}$$

If $\mathbb{Z}_1 \overset{\text{def}}{=} \{(a,c),(a,y),(b,c),(d,y)\} \subset \mathbb{X}\times\mathbb{Y}$, we have

$$\begin{array}{rcl} \text{proj}_{\mathbb{X}}(\mathbb{Z}_1) & = & \{a,b,d\},\\ \text{proj}_{\mathbb{Y}}(\mathbb{Z}_1) & = & \{c,y\}. \end{array} \tag{4}$$

■

Example 2: If

$$\mathbb{S}=\{(x,y,z)\in[1,5]\times[2,4]\times[6,10] \mid z=x+y\}, \tag{5}$$

then

$$\begin{array}{rcl} \text{proj}_{\mathbb{X}}(\mathbb{S}) & = & [6,9]\\ \text{proj}_{\mathbb{Y}}(\mathbb{S}) & = & [2,4]\\ \text{proj}_{\mathbb{Z}}(\mathbb{S}) & = & [2,5]. \end{array} \tag{6}$$

■

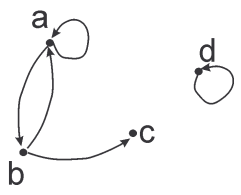
Relation (or binary constraint)

A relation in $\mathbb{X}$ is a subset of $\mathbb{X} \times \mathbb{X}$.

Example 1: If $\mathbb{X} = \{a, b, c, d\}$, the set

$$\mathcal{C} = \{(a,a), (a,b), (b,a), (b,c), (d,d)\} \tag{7}$$

is a relation or a binary constraint in $\mathbb{X}$. Since $\mathbb{X}$ is finite, it can be represented a directed graph.



or by the matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

■

Example 2: The set

$$\mathcal{C} = \left\{(x,y) \in \mathbb{R}^2 | y = \sin(x)\right\} \tag{9}$$

is a relation in $\mathbb{R}$. This relation can be written as "$y = \sin x$", or "$\sin(y,x)$" or "sin". ■

Example 3: The set

$$\mathcal{C} = \left\{(x,y) \in \mathbb{R}^2 | y \leq x\right\} \tag{10}$$

is a relation in $\mathbb{R}$. This relation can be written as "$y \leq x$", or "$\leq (y,x)$" or "$\leq$". ■

Example 4: The set

$$\mathcal{C} = \left\{(x,y) \in \mathbb{R}^2 | \sin(x+y) = 0\right\} \tag{11}$$

is a relation in $\mathbb{R}$. This relation can be written as "$\sin(x+y) = 0$". ■

Constraints
A constraint in $\mathbb{X}$ is a subset of $\mathbb{X} \times \mathbb{X} \times \cdots \times \mathbb{X}$.

Example 1: If $\mathbb{X} = \{a, b, c, d\}$, the set

$$\mathcal{C} = \{(a,a,a), (a,b,c), (c,c,a)\} \tag{12}$$

is a ternary constraint in $\mathbb{X}$. ■

Example 2: The set

$$\mathcal{C} = \left\{(x,y,z) \in \mathbb{R}^3 | z = x + y\right\} \tag{13}$$

is a ternary constraint in $\mathbb{R}$. This set can be written as "$z = x + y$", or "$+(z,y,x)$" or "+". ■

Functions
Consider a function $f : \mathbb{X} \to \mathbb{Y}$. If $\mathbb{X}_1 \subset \mathbb{X}$, the *direct image* of $\mathbb{X}_1$ by f is

$$f(\mathbb{X}_1) \overset{\text{def}}{=} \{f(x) \mid x \in \mathbb{X}_1\}. \tag{14}$$

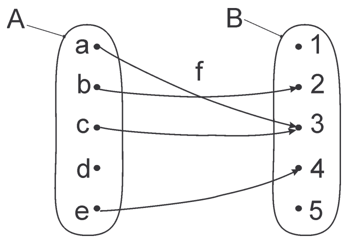
If $\mathbb{Y}_1 \subset \mathbb{Y}$, the *reciprocal image* of $\mathbb{Y}_1$ by f is

$$f^{-1}(\mathbb{Y}_1) \overset{\text{def}}{=} \{x \in \mathbb{X} \mid f(x) \in \mathbb{Y}_1\}. \tag{15}$$

If $\emptyset$ denotes the empty set, then the previous definitions imply that $f(\emptyset) = f^{-1}(\emptyset) = \emptyset$. It is trivial to show that if $\mathbb{X}_1$ and $\mathbb{X}_2$ are subsets of $\mathbb{X}$ and if $\mathbb{Y}_1$ and $\mathbb{Y}_2$ are subsets of $\mathbb{Y}$, then

$$\begin{gathered}
f(\mathbb{X}_1 \cap \mathbb{X}_2) \subset f(\mathbb{X}_1) \cap f(\mathbb{X}_2), \\
f(\mathbb{X}_1 \cup \mathbb{X}_2) = f(\mathbb{X}_1) \cup f(\mathbb{X}_2), \\
f^{-1}(\mathbb{Y}_1 \cap \mathbb{Y}_2) = f^{-1}(\mathbb{Y}_1) \cap f^{-1}(\mathbb{Y}_2), \\
f^{-1}(\mathbb{Y}_1 \cup \mathbb{Y}_2) = f^{-1}(\mathbb{Y}_1) \cup f^{-1}(\mathbb{Y}_2), \\
f\left(f^{-1}(\mathbb{Y})\right) \subset \mathbb{Y}, \\
\mathbb{X}_1 \ \subset f^{-1}(\mathbb{Y}) \Rightarrow f^{-1}(f(\mathbb{X}_1)) \supset \mathbb{X}_1 \\
\mathbb{X}_1 \subset \mathbb{X}_2 \Rightarrow f(\mathbb{X}_1) \subset f(\mathbb{X}_2), \\
\mathbb{Y}_1 \subset \mathbb{Y}_2 \Rightarrow f^{-1}(\mathbb{Y}_1) \subset f^{-1}(\mathbb{Y}_2), \\
\mathbb{X} \subset \mathbb{Y}_1 \times \mathbb{Y}_2 \Rightarrow \ \mathbb{X} \subset \mathrm{proj}_{\mathbb{Y}_1}(\mathbb{X}) \times \mathrm{proj}_{\mathbb{Y}_2}(\mathbb{X}).
\end{gathered} \tag{16}$$

Example 1: If f is defined as follows



then

$$\begin{aligned}
f(A) &= \{2,3,4\} = \mathrm{Im}(f). \\
f^{-1}(B) &= \{a,b,c,e\} = \mathrm{dom}(f). \\
f^{-1}(f(A)) &= \{a,b,c,e\} \subset A \\
f^{-1}(f(\{b,c\})) &= \{a,b,c\}.
\end{aligned} \tag{17}$$

■

Example 2: If $f(x) = x^2$, then

$$\begin{aligned}
f([2,3]) &= [4,9] \\
f^{-1}([4,9]) &= [-3,-2] \cup [2,3].
\end{aligned} \tag{18}$$

This is consistent with the property $f\left(f^{-1}(\mathbb{Y})\right) \subset \mathbb{Y}$. ■

Example 3: If $f(x) = \log$, then

$$f^{-1}(f([-3,-2])) = \emptyset. \tag{19}$$

Correct the error in the book page 13, line 6 of (2.10). ■

Interval computation

(Luc Jaulin , Monday, 12h00-12h30).

Intervals

A (closed) *interval* is a connected, closed subset of $\mathbb{R}$. The set of all intervals of $\mathbb{R}$ will be denoted by $\mathbb{IR}$. For example $[1,3]$, $\{1\}$, $]-\infty,6]$, $\mathbb{R}$ and $\emptyset$ are considered as intervals whereas $]1,3[$, $[3,2]$ and $[1,2]\cup[3,4]$ are not. The *lower bound* of the interval $[x]$ is defined by

$$\underline{x} = \mathrm{lb}([x]) = \inf\{x | x \in [x]\}. \tag{20}$$

The *upper bound* of the interval $[x]$ is defined by

$$\bar{x} = \mathrm{ub}([x]) = \sup\{x | x \in [x]\}. \tag{21}$$

By convention, $\mathrm{ub}(\emptyset) = -\infty$ and $\mathrm{lb}(\emptyset) = +\infty$. The *width* of the interval $[x]$ is defined by $w([x]) = \bar{x} - \underline{x}$. The *midpoint* of the interval $[x]$ is defined by

$$\mathrm{mid}([x]) = \frac{\bar{x}+\underline{x}}{2}. \tag{22}$$

The *enveloping interval* associated with a subset $\mathbb{X}$ of $\mathbb{R}$ is the smallest interval containing $\mathbb{X}$ and is denoted by $[\mathbb{X}]$. For instance

$$[\ [1,3]\cup[6,7[\ ] = [1,7]. \tag{23}$$

For two intervals $[x]$ and $[y]$, the *interval union* is defined by

$$[x] \sqcup [y] = [[x]\cup[y]]. \tag{24}$$

Binary operators

If $\diamond \in \{+,-,*,/,\max,\min\}$, where $*$ is the multiplication, and if $[x]$ and $[y]$ are two intervals, we define

$$[x]\diamond[y] \overset{\text{def}}{=} [\{x\diamond y \mid x\in[x], y\in[y]\}]. \tag{25}$$

Therefore,

$$\begin{array}{rl} [\underline{x},\bar{x}] + [\underline{y},\bar{y}] & = [\underline{x}+\underline{y}, \bar{x}+\bar{y}] \\ [\underline{x},\bar{x}]\cdot[\underline{y},\bar{y}] & = [\min(\underline{x}\underline{y}, \bar{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\bar{y}), \\ & \quad \max(\underline{x}\underline{y}, \bar{x}\underline{y}, \underline{x}\bar{y}, \bar{x}\bar{y})] \\ \max\left([\underline{x},\bar{x}],[\underline{y},\bar{y}]\right) & = [\max(\underline{x},\underline{y}), \max(\bar{x},\bar{y})]. \end{array} \tag{26}$$

For instance,

$$\begin{array}{rl} [-1,3]+[2,5] & = [1,8], \\ [-1,3].[2,5] & = [-5,15], \\ [-1,3]/[2,5] & = [-\frac{1}{2},\frac{3}{2}], \\ \max([-1,3],[2,5]) & = [2,5]. \end{array} \tag{27}$$

We have

$$([1,2]+[-3,4])*[-1,5] = [-2,6]*[-1,5] = [-10,30]. \tag{28}$$

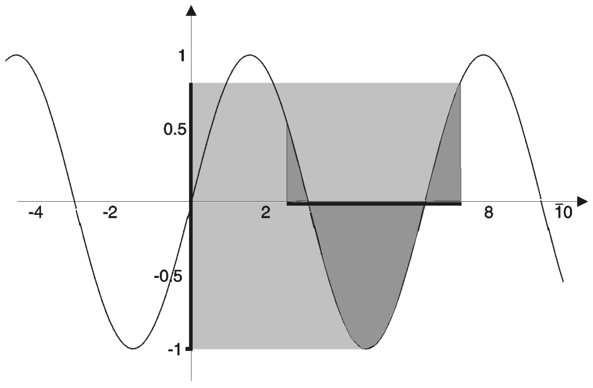
Elementary functions

If $f \in \{\cos, \sin, \mathrm{sqr}, \mathrm{sqrt}, \log, \exp, \dots\}$, is a function from $\mathbb{R}$ to $\mathbb{R}$, we define its interval extension as

$$f([x]) \overset{\text{def}}{=} [\{f(x) \mid x \in [x]\}]. \tag{29}$$

For instance

$$\begin{array}{rcl} \sin([0,\pi]) & = & [0,1], \\ \mathrm{sqr}([-1,3]) & = & [-1,3]^2 = [0,9], \\ \mathrm{abs}([-7,1]) & = & [0,7], \\ \mathrm{sqrt}([-10,4]) & = & \sqrt{[-10,4]} = [0,2], \\ \log([-2,-1]) & = & \emptyset. \end{array} \tag{30}$$



The sine function with an interval argument

Interpretation
If f is an expression then

$$f([x],[y]) = [z] \Rightarrow \forall x \in [x], \forall y \in [y], \exists z \in [z], z = f(x,y)$$

Modal intervals : handle proper intervals (such as $[1,2]$) and improper intervals (such as $[2,1]$).
For instance,

$$[1,4] + [2,1] = [3,5]$$

should be interpreted as

$$\forall x \in [1,4], \exists y[1,2], \exists z \in [3,5], z = x + y$$

and

$$[4,1] + [1,2] = [5,3]$$

should be interpreted as

$$\forall y \in [1,2], \forall z \in [3,5], \exists x \in [1,4], z = x + y.$$

Modal interval analysis can be useful to prove quickly propositions such as

$$\begin{aligned} \forall x_1 &\in [x_1], \forall x_2 \in [x_2], \\ \exists y_1 &\in [y_1], \exists y_2 \in [y_2], \exists z \in [z], \\ z &= \sin(x_1 x_2) + x_2 y_1 - y_2 x_2^2. \end{aligned}$$

Boxes
A *box*, or *interval vector,* $[\mathbf{x}]$ of $\mathbb{R}^n$ is the Cartesian product of n intervals *i.e.* a vector with interval components:

$$[\mathbf{x}] = [\underline{x}_1, \bar{x}_1] \times \cdots \times [\underline{x}_n, \bar{x}_n] = [x_1] \times \cdots \times [x_n]. \tag{31}$$

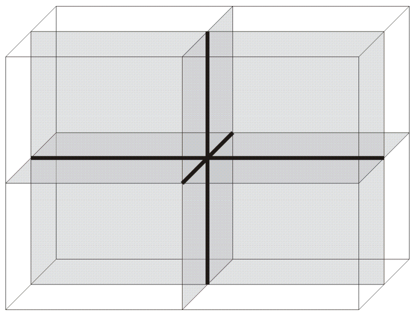
The set of all boxes of $\mathbb{R}^n$ will be denoted by $\mathbb{IR}^n$. The *width* $w([\mathbf{x}])$ of a box $[\mathbf{x}]$ is the length of its largest side

$$w([\mathbf{x}]) = \max_{i \in \{1,\dots,n\}} w([x_i]). \tag{32}$$

For instance

$$w([1,2] \times [-1,3]) = 4. \tag{33}$$

If $w([\mathbf{x}]) = 0$, $[\mathbf{x}]$ is said to be *degenerated.* In such a case, $[\mathbf{x}]$ is a singleton of $\mathbb{R}^n$ and will be denoted $\{\mathbf{x}\}$. The *principal plane* of $[\mathbf{x}]$ is the symmetric plane $[\mathbf{x}]$ perpendicular to its largest side.


A n-dimensional box has n symmetric planes

To *bisect* a box $[\mathbf{x}]$ means to split it in the two parts separated by its principal plane. For instance, the bisection of $[\mathbf{x}] = [1,2] \times [-1,3]$ generates the two boxes $\text{Left}([\mathbf{x}]) = [1,2] \times [-1,1]$ and $\text{Right}([\mathbf{x}]) = [1,2] \times [1,3]$.

Set inversion

(Luc Jaulin, Tuesday, 9h45-10h30).

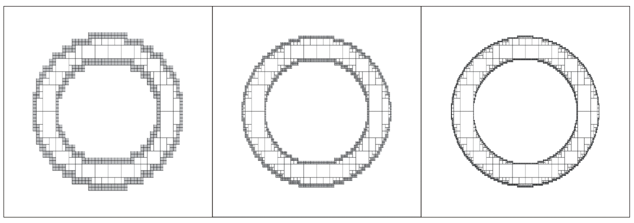
Subpavings
A *subpaving* of $\mathbb{R}^n$ is a set of non-overlapping boxes of $\mathbb{R}^n$. Compact sets $\mathbb{X}$ can be bracketed between inner and outer subpavings:

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+. \tag{34}$$

The following figure illustrates the bracketing of the set

$$\mathbb{X} = \left\{(x_1, x_2) \;\middle|\; x_1^2 + x_2^2 \in [1,2]\right\} \tag{35}$$

between subpavings with an increasing accuracy from left to right. The frame corresponds to the box $[-2,2] \times [-2,2]$. The subpaving $\Delta\mathbb{X}$ in grey contains the boundary of $\mathbb{X}$ whereas the subpaving $\mathbb{X}^-$ in white is inside $\mathbb{X}$.



Set operations such as $\mathbb{Z} := \mathbb{X} + \mathbb{Y},\ \mathbb{X} := \mathbf{f}^{-1}(\mathbb{Y}), \mathbb{Z} := \mathbb{X} \cap \mathbb{Y} \dots$ can be approximated by subpaving operations.

Stack-queue
A *list* is a (possibly empty) ordered finite set. If the list is nonempty, all its elements, except one (called the *last*), have a next element. The element of the list which is the next of nobody is called the *first* element. A *queue* is a list on which two operations are allowed : (i) add an element at the end (*push*) (ii) remove the first element (*pull*). A *stack* is a list on which two operations are allowed : (i) add an element at the beginning of the list (*stack*) (ii) remove the first element (*pop*).

Example: Let $\mathcal{L}$ be an empty list. The next table illustrates the evolution of the queue when different operations are performed.

k	operation	result
0		$\mathcal{L} = \emptyset$
1	$\text{push}(\mathcal{L}, a)$	$\mathcal{L} = \{a\}$
2	$\text{push}(\mathcal{L}, b)$	$\mathcal{L} = \{a, b\}$
3	$x := \text{pull}(\mathcal{L})$	$x = a, \mathcal{L} = \{b\}$
4	$x := \text{pull}(\mathcal{L})$	$x = b, \mathcal{L} = \emptyset.$

(36)

For a stack, the table becomes

k	operation	result
0		$\mathcal{L} = \emptyset$
1	$\text{stack}(\mathcal{L}, a)$	$\mathcal{L} = \{a\}$
2	$\text{stack}(\mathcal{L}, b)$	$\mathcal{L} = \{a, b\}$
3	$x := \text{pop}(\mathcal{L})$	$x = b, \mathcal{L} = \{a\}$
4	$x := \text{pop}(\mathcal{L})$	$x = a, \mathcal{L} = \emptyset.$

(37)

■

Set inversion
Let $\mathbf{f}$ be a possibly nonlinear function from $\mathbb{R}^n$ to $\mathbb{R}^m$ and let $\mathbb{Y}$ be a subset of $\mathbb{R}^m$ (for instance, a subpaving). Set inversion is the characterization of

$$\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) \in \mathbb{Y}\} = \mathbf{f}^{-1}(\mathbb{Y}). \tag{38}$$

For any $\mathbb{Y} \subset \mathbb{R}^m$ and for any function $\mathbf{f}$ admitting a convergent inclusion function $[\mathbf{f}](.)$, two subpavings $\mathbb{X}^-$ and $\mathbb{X}^+$ such that

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+, \tag{39}$$

can be obtained with the algorithm SIVIA (Set Inverter Via Interval Analysis), to be described now. To test if a box $[\mathbf{x}]$ is inside or outside $\mathbb{X}$, we shall use the following tests.

$$\begin{array}{lllll} \text{(i)} & [\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y} & \Rightarrow & [\mathbf{x}] \subset \mathbb{X} \\ \text{(ii)} & [\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset & \Rightarrow & [\mathbf{x}] \cap \mathbb{X} = \emptyset. \end{array} \tag{40}$$

Boxes for which these tests failed, will be bisected, except if they are too small (*i.e.,* smaller than the required accuracy ε). The box $[\mathbf{x}](0)$ is a box which is assumed to enclose the solution set $\mathbb{X}$. $\mathcal{L}$ is a queue of boxes.

```
Algo SIVIA(in: $[\mathbf{x}]$; out: $\mathcal{L}^-, \mathcal{L}^+$)
1  $\mathcal{L} := \{[\mathbf{x}]\}$; $\mathcal{L}^- = \emptyset$; $\mathcal{L}^+ := \emptyset$;
2  if $\mathcal{L} \neq \emptyset$, $[\mathbf{x}] :=$ pop $(\mathcal{L})$, else end;
3  if $[\mathbf{f}]([\mathbf{x}]) \subset \mathbb{Y}$, push($\mathcal{L}^-, [\mathbf{x}]$);push($\mathcal{L}^+, [\mathbf{x}]$); goto 2;
4  if $[\mathbf{f}]([\mathbf{x}]) \cap \mathbb{Y} = \emptyset$, goto 2;
5  if $w([\mathbf{x}]) < \varepsilon$, push($\mathcal{L}^+, [\mathbf{x}]$); goto 2;
6  stack($\mathcal{L}$,Left($[\mathbf{x}]$),Right($[\mathbf{x}]$)); goto 2.
```

If $\mathbb{X}^-$ denotes the union of boxes in $\mathcal{L}^-$ and if $\mathbb{X}^+$ is the union of boxes in $\mathcal{L}^+$ then :

$$\mathbb{X}^- \subset \mathbb{X} \subset \mathbb{X}^+. \tag{41}$$

Sivia with contractors
The constraint $\mathbf{f}(\mathbf{x}) \in \mathbb{Y}$ defining $\mathbb{X}$ can be translated into nonlinear inequalities:

$$\left\{ \begin{array}{lll} g_1(x_1, x_2, \ldots, x_n) & \leq & 0, \\ \vdots & \vdots & \vdots \\ g_m(x_1, x_2, \ldots, x_n) & \leq & 0. \end{array} \right. \tag{42}$$

Thus

$$\begin{array}{lll} \mathbb{X} & = & \{\mathbf{x} \in \mathbb{R}^n \mid \max(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \leq 0\} \text{ and} \\ \neg\mathbb{X} & = & \{\mathbf{x} \in \mathbb{R}^n \mid \max(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) > 0\}. \end{array} \tag{43}$$

A contractor $\mathcal{C}_{\mathbb{X}}$ for $\mathbb{X}$ and a contractor $\mathcal{C}_{\neg\mathbb{X}}$ for $\neg\mathbb{X}$ can easily be obtained (for instance, by a forward-backward propagation).

```
Algorithm SIVIAC(in: $[\mathbf{x}]$; out: $\mathcal{L}^-, \mathcal{L}^+$)
1  $\mathcal{L} := \{[\mathbf{x}]\}$; $\mathcal{L}^- = \emptyset$; $\mathcal{L}^+ := \emptyset$;
2  if $\mathcal{L} \neq \emptyset$ then $[\mathbf{x}] :=$ pop $(\mathcal{L})$ else end;
3  $[\mathbf{x}] := \mathcal{C}_{\mathbb{X}}([\mathbf{x}])$; if $[\mathbf{x}] = \emptyset$, goto 2
4  $[\mathbf{a}] := \mathcal{C}_{\neg\mathbb{X}}([\mathbf{x}])$;
5  if $[\mathbf{a}] \neq [\mathbf{x}]$, push($\mathcal{L}^-, [\mathbf{x}]\backslash[\mathbf{a}]$);push($\mathcal{L}^+, [\mathbf{x}]\backslash[\mathbf{a}]$);
6  if ($w([\mathbf{a}]) < \varepsilon$), push($\mathcal{L}^+, [\mathbf{a}]$); goto 2;
7  stack($\mathcal{L}$,Left($[\mathbf{a}]$),Right($[\mathbf{a}]$)); goto 2.
```

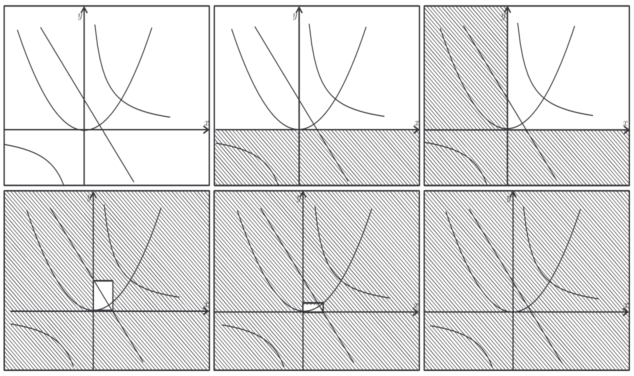
Unconstrained global optimization

(Luc Jaulin, Tuesday, 11h45-12h30).

Constraints propagation (reminder)
Consider the three following constraints

$$\begin{array}{lll} (C_1) & : & y = x^2 \\ (C_2) & : & xy = 1 \\ (C_3) & : & y = -2x + 1 \end{array} \qquad (44)$$

To each variable, we associate the domain $]-\infty,\infty[$. A constraint propagation consists in projecting all constraints until equilibrium.



For more complex constraints, a decomposition is required. For instance, the CSP

$$\begin{array}{l} x + \sin(y) - xz \le 0, \\ x \in [-1,1], y \in [-1,1], z \in [-1,1] \end{array} \qquad (45)$$

can be decomposed into the following one.

$$\left\{ \begin{array}{l} a = \sin(y) \\ b = x + a \\ c = xz \\ b - c = d \end{array} \right. , \quad \begin{array}{ll} x \in [-1,1] & a \in]-\infty,\infty[\\ y \in [-1,1] & b \in]-\infty,\infty[\\ z \in [-1,1] & c \in]-\infty,\infty[\\ & d \in]-\infty,0] \end{array} \qquad (46)$$

Recall that since the decomposition introduces pessimism, the decomposition should only be done for constraints for which no projection procedure are available.

Minimization
The problem to be considered now is the minimization of a cost function $f(\mathbf{x})$ over a box $[\mathbf{x}] \in \mathbb{R}^n$:

$$\hat{f} = \min_{\mathbf{x} \in [\mathbf{x}]} f(\mathbf{x}). \qquad (47)$$

In the following algorithm, $\mathcal{L}$ is a list of boxes which contains all global minimizers. The real number f^+ is an upper bound for the global minimum $\hat{f}$. It is assumed that $f(\mathbf{x})$ is twice differentiable and has a global minimizer $\mathbf{x}^*$ which is not on the boundary of $[\mathbf{x}]$. Hence, the gradient $\nabla f(\mathbf{x}^*)$ at $\mathbf{x}^*$ should be zero and the Hessian $\mathbf{H}(\mathbf{x}^*)$ at $\mathbf{x}^*$ of f must be positive semi-definite ($\succeq 0$). If $\mathbf{H}([\mathbf{x}])$ is proved to contain no matrices $\succeq 0$, then $[\mathbf{x}]$ cannot contain any minimizer. This is the *nonconvexity check* used by Hansen to eliminate boxes $[\mathbf{x}]$. The test used by Hansen checks the signs of the diagonal entries of the interval matrix $[\mathbf{H}]$. If one of them is negative then the matrix cannot be $\succeq 0$.

```
Algorithm MINIMIZE(in: $[\mathbf{x}]$; out: $f^+, \mathcal{L}$)
1  $\mathcal{L} := \{[\mathbf{x}]\}$; $f^+ = \infty$;
2  if all boxes have a width $< \varepsilon$, return $(\mathcal{L})$;
3  pull the largest box $[\mathbf{x}]$ of $\mathcal{L}$;
4  $f^+ = \min(f^+, \text{localmin}(f, [\mathbf{x}]))$;
5  $[\mathbf{x}] := \mathcal{C}_{\{\mathbf{x} | f(\mathbf{x}) \le f^+, \nabla f(\mathbf{x}) = 0, \mathbf{H}_f(\mathbf{x}) \succeq 0\}}([\mathbf{x}])$;
6  if $[\mathbf{x}] = \emptyset$, goto 2;
7  if $(w([\mathbf{x}]) < \varepsilon)$ then push $[\mathbf{x}]$ into $\mathcal{L}$; goto 2;
8  bisect $[\mathbf{x}]$ and store the two resulting boxes in $\mathcal{L}$; goto 2.
```

Example (Collaboration with D. Henrion)
Consider the three-hump camel function (this is also the example chosen by Hansen to illustrate his nonconvexity check):

$$f(\mathbf{x}) = 2x_1^2 - 1.05x_1^4 + \frac{x_1^6}{6} - x_1x_2 + x_2^2. \tag{48}$$

Its gradient is

$$\nabla f(\mathbf{x}) = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{pmatrix} = \begin{pmatrix} 4x_1 - x_2 - 4.2x_1^3 + x_1^5 \\ 2x_2 - x_1 \end{pmatrix}. \tag{49}$$

Its Hessian is

$$\mathbf{H}(\mathbf{x}) = \begin{pmatrix} 4 + 5x_1^4 - 12.6x_1^2 & -1 \\ -1 & 2 \end{pmatrix}. \tag{50}$$

After decomposition, the constraints $f(\mathbf{x}) \le f^+, \nabla f(\mathbf{x}) = 0, \mathbf{H}_f(\mathbf{x}) \succeq 0$ become

$$\begin{array}{l} \text{(i)}x_{12} = x_1^2; \text{(ii)}x_{13} = x_1^3; \text{(iii)}x_{14} = x_1^4; \text{(iv)}x_{15} = x_1^5; \text{(v)}x_{16} = x_1^6; \text{(vi)}x_{22} = x_2^2; \text{(vii)}a = x_1x_2 \\ \text{(viii)} \left\{ \begin{array}{ccc} 2x_{12} - 1.05\ x_{14} + \frac{x_{16}}{6} - a + x_{22} & \le & f^+ \\ 4x_1 - x_2 - 4.2\ x_{13} + x_{15} & = & 0 \\ 2x_2 - x_1 & = & 0 \\ \begin{pmatrix} 4 + 5x_{14} - 12.6x_{12} & -1 \\ -1 & 2 \end{pmatrix} & \succeq & 0 \end{array} \right. \end{array} \tag{51}$$

Note that the eigen values of the Hessian matrix

$$\begin{pmatrix} 4 + 5x_{14} - 12.6x_{12} & -1 \\ -1 & 2 \end{pmatrix} \tag{52}$$

are given by:

$$\lambda_{1,2} = \frac{5}{2}x_{14} - 6.3x_{12} \pm \frac{1}{2}\sqrt{20x_{14} - 50.4x_{12} - 126x_{12}x_{14} + 158.76x_{12}^2 + 25x_{14}^2 + 8} + 3.$$

should be positive. Since the constraint (viii), which involves 7 variables, is an LMI, it can be projected and should not be decomposed.

LMIs
A linear matrix inequality (LMI) has the form

$$\mathbf{A}(\mathbf{x}) \overset{\text{def}}{=} \mathbf{A}_0 + x_1\mathbf{A}_1 + \cdots + x_m\mathbf{A}_m \succeq 0, \tag{53}$$

where $\mathbf{x} \in \mathbb{R}^m$ is a vector of variables and the $\mathbf{A}_i$ are square symmetric matrices. An *LMI set* is a subset of $\mathbb{R}^m$ which can be defined by an LMI. The *box-LMI problem,* which consists in finding the smallest box $[\mathbf{x}]$ which encloses a set $\mathbb{X}$ defined by an LMI constraint, has a polynomial complexity in the worst-case.

Example 1. A set of linear constraints (equalities or inequalities) is an LMI. For instance

$$\left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + b_1 & \ge 0 \\ a_{21}x_1 + a_{22}x_2 + b_2 & \ge 0 \end{array} \right. \tag{54}$$

is equivalent to the following LMI

$$\begin{pmatrix} a_{11}x_1 + a_{12}x_2 + b_1 & 0 \\ 0 & a_{21}x_1 + a_{22}x_2 + b_2 \end{pmatrix} \succeq 0, \tag{55}$$

i.e.,

$$\begin{pmatrix} b_1 & 0 \\ 0 & b_2 \end{pmatrix} + x_1 \begin{pmatrix} a_{11} & 0 \\ 0 & a_{21} \end{pmatrix} + x_2 \begin{pmatrix} a_{12} & 0 \\ 0 & a_{22} \end{pmatrix} \succeq 0. \tag{56}$$

■

Example 2. An ellipsoid of $\mathbb{R}^n$ is an LMI set. Consider for instance, the ellipse defined by $3x_1^2 + 2x_2^2 - 2x_1x_2 \leq 5$. We have

$$\begin{aligned} 3x_1^2 + 2x_2^2 - 2x_1x_2 \leq 5 \quad &\Leftrightarrow \quad \begin{pmatrix} 1 & x_1 & x_2 \\ x_1 & 2 & 1 \\ x_2 & 1 & 3 \end{pmatrix} \succeq 0 \\ &\Leftrightarrow \quad \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 3 \end{pmatrix} + x_1 \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} + x_2 \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \succeq 0. \end{aligned} \tag{57}$$

■

Minimax Optimization

(Luc Jaulin, Tuesday, 14h00-14h45).

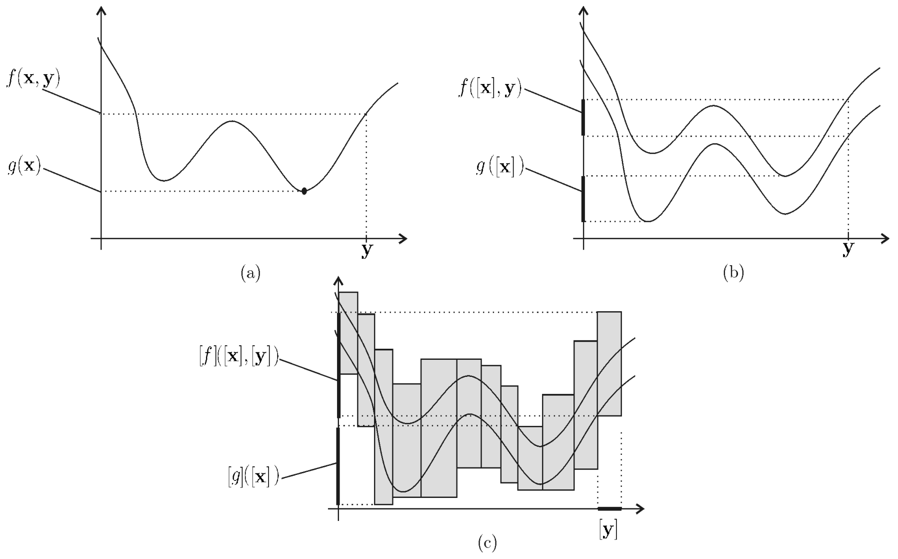
Perturbed minimization
Consider the function

$$g(\mathbf{x}) = \min_{\mathbf{y}\in[\mathbf{y}]} f(\mathbf{x},\mathbf{y}). \tag{58}$$

Assume that an inclusion function $[f]([\mathbf{x}],[\mathbf{y}])$ for $f(\mathbf{x},\mathbf{y})$ is available. An inclusion function $[g]([\mathbf{x}])$ for $g(\mathbf{x})$ can be obtained by the algorithm below. The real number f^+ represents the best known upper bound for $g([\mathbf{x}])$.

```
Algorithm PertMin(in: $[\mathbf{x}],[\mathbf{y}],[f]$; out: $[g]([\mathbf{x}])$)
1   $\mathcal{L} := \{[\mathbf{y}]\}$; $f^+ = \infty$;
2   if $\forall[\mathbf{y}] \in \mathcal{L}$, $w([\mathbf{y}]) < w([\mathbf{x}]) + \varepsilon$, return $\min_{[\mathbf{y}]\in\mathcal{L}}[f]([\mathbf{x}],[\mathbf{y}])$
3   take the largest box $[\mathbf{y}]$ of $\mathcal{L}$;
4   $f^+ = \min(f^+, \mathrm{ub}([f]([\mathbf{x}], \mathrm{center}([\mathbf{y}])))$;
5   if $[f]([\mathbf{x}],[\mathbf{y}]) > f^+$ then goto 2;
6   if $(w([\mathbf{y}]) \leq w([\mathbf{x}]) + \varepsilon$ then push $[\mathbf{y}]$ into $\mathcal{L}$; goto 2;
7   bisect $[\mathbf{y}]$ and store the two resulting boxes in $\mathcal{L}$; goto 2.
```

Note that in PertMin, the box $[\mathbf{x}]$ is not bisected. At step 2, the arguments of the min operators are intervals (for instance, $\min([3,7],[2,9],[4,5]) = [2,5]$). At step 6, ε is a small positive number which is useful for the particular case where $[\mathbf{x}]$ is a singleton. The behavior of the algorithm is illustrated by the following figure.



The perturbed maximization problem can be solved using PertMin: since

$$h(\mathbf{x}) = \max_{\mathbf{y}\in[\mathbf{y}]} f(\mathbf{x},\mathbf{y}) = -\min_{\mathbf{y}\in[\mathbf{y}]} -f(\mathbf{x},\mathbf{y}), \tag{59}$$

an inclusion function for $h(\mathbf{x})$ can be obtained by $[h]([\mathbf{x}]) = -\text{PertMin}([\mathbf{x}],[\mathbf{y}],[-f])$.

Perturbed minimization with constraints
The function

$$g(\mathbf{x}) = \min_{\substack{\mathbf{y}\in[\mathbf{y}] \\ \text{s.t. } \mathbf{h}(\mathbf{x},\mathbf{y})\leq 0}} f(\mathbf{x},\mathbf{y}) \tag{60}$$

can be rewritten as

$$g(\mathbf{x}) = \min_{\mathbf{y}\in[\mathbf{y}]} f(\mathbf{x},\mathbf{y}) + \eta(\mathbf{h}(\mathbf{x},\mathbf{y})), \tag{61}$$

where

$$\eta(\mathbf{v}) = \begin{cases} 0 & \text{if } \mathbf{v} \leq 0 \\ \infty & \text{otherwise} \end{cases} \tag{62}$$

The minimal inclusion function $[\eta]$ for η is $[\eta]=[\eta(\underline{v}),\eta(\bar{v})]$. For instance

$$\begin{array}{rcl} \eta(-1,-3,-2) & = & 0; \\ \eta(-1,-3,2) & = & \infty; \\ [\eta]([-3,-1],[-3,2],[-2,5]) & = & [0,\infty]; \\ [\eta]([-3,-1],[-3,-2],[-2,-1]) & = & [0,0]; \\ [\eta]([-3,-1],[1,2],[-2,5]) & = & [\infty,\infty]; \end{array} \tag{63}$$

As a consequence, an inclusion function for $f(\mathbf{x},\mathbf{y})+\eta(\mathbf{h}(\mathbf{x},\mathbf{y}))$ is $[f]([\mathbf{x}],[\mathbf{y}])+[\eta]([\mathbf{h}]([\mathbf{x}],[\mathbf{y}]))$. The previous algorithm can thus be used to get an inclusion function $[g]([\mathbf{x}])$ for $g(\mathbf{x})\overset{\text{def}}{=}\min\{f(\mathbf{x},\mathbf{y}),\mathbf{y}\in[\mathbf{y}],\mathbf{h}(\mathbf{x},\mathbf{y})\leq 0\}$ if inclusion functions for $f(\mathbf{x},\mathbf{y})$ and $\mathbf{h}(\mathbf{x},\mathbf{y})$ are available.

Minimax optimization
Consider the problem of computing an enclosure for

$$f_3=\min_{\substack{x_3\in[-1,2]\\ \sin(x_3)\leq 0}} \underbrace{\max_{\substack{x_2\in[-1,1]\\ x_3^2+x_2\leq 0}} \underbrace{\min_{\substack{x_1\in[0,10]\\ x_1^2+x_2x_3\leq 0}} \underbrace{x_1(x_2+x_3)}_{f_0(x_1,x_2,x_3)}}_{f_1(x_2,x_3)}}_{f_2(x_3)}. \tag{64}$$

It can be rewritten as

$$f_3=\min_{x_3\in[-1,2]} \underbrace{\eta(\sin x_3)+\max_{x_2\in[-1,1]} \underbrace{-\eta\left(x_3^2+x_2\right)+\min_{x_1\in[0,10]} \underbrace{\eta(x_1^2+x_2x_3)+x_1(x_2+x_3)}_{\varphi_0(x_1,x_2,x_3)}}_{\varphi_1(x_2,x_3)}}_{\varphi_2(x_3)}$$

An inclusion function for $\varphi_0(x_1,x_2,x_3)$ can be obtained using interval arithmetic. An inclusion function for $\varphi_1(x_2,x_3)$ can be obtained using PertMin. An inclusion function for $\varphi_2(x_3)$ can also be obtained using PertMin. An enclosure for the real number f_3 can thus be obtained using PertMin where the perturbed box is now degenerated.

Remark: The operators min and max cannot commute in general. For instance,

$$\max_{x\in\{-1,1\}}\min_{y\in\{-1,1\}} xy = \max_{x\in\{-1,1\}} x.(-sign(x))=-1 \tag{65}$$

$$\min_{y\in\{-1,1\}}\max_{x\in\{-1,1\}} xy = \min_{y\in\{-1,1\}} sign(y).y=1. \tag{66}$$

We always have

$$\max_{x\in X}\min_{y\in Y} f(x,y)\leq \min_{y\in Y}\max_{x\in X} f(x,y). \tag{67}$$

■

Set projection
Problems involving $\exists$ and $\forall$ are closely related to minimax problems. For instance, proving that

$$\forall p_3\in[1,3],\exists p_2\in[1,2],\forall p_1\in[0,1], p_1+p_2p_3\leqslant 1 \tag{68}$$

amounts to proving that

$$\max_{p_3\in[1,3]}\min_{p_2\in[1,2]}\max_{p_1\in[0,1]} p_1+p_2p_3\leqslant 1. \tag{69}$$

Consider the set

$$\mathbb{S} \overset{\text{def}}{=} \{\mathbf{x} \in [\mathbf{x}] | \exists \mathbf{y} \in [\mathbf{y}], \mathbf{f}(\mathbf{x},\mathbf{y}) \leq \mathbf{0}\}. \tag{70}$$

This problem is known as the set projection problem. The set $\mathbb{S}$ can be written as

$$\mathbb{S} = \left\{\mathbf{x} \in [\mathbf{x}] | \min_{\mathbf{y}\in[\mathbf{y}]} \max(f_1(\mathbf{x},\mathbf{y}),\ldots,f_m(\mathbf{x},\mathbf{y})) \leq 0\right\}. \tag{71}$$

If an inclusion function is available for all f_i, an inclusion function is also known for $\max(f_1(\mathbf{x},\mathbf{y}),\ldots,f_m(\mathbf{x},\mathbf{y}))$. From PertMin, an inclusion function $[g](\mathbf{x})$ for

$$g(\mathbf{x}) = \min_{\mathbf{y}\in[\mathbf{y}]} \max(f_1(\mathbf{x},\mathbf{y}),\ldots,f_m(\mathbf{x},\mathbf{y})). \tag{72}$$

can thus be obtained. This inclusion function can thus be used by SIVIA to characterize $\mathbb{S}$.

Epigraphs (Collaboration with M. Dao, M. Lhommeau)
We shall now give an illustration of the set projection methodology to a problem related to global optimization. Consider the following optimization problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \text{ s.t. } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}. \tag{73}$$

Its epigraph is defined by

$$\mathbb{S} = \{(\mathbf{x}, a) \in \mathbb{R}^n \times \mathbb{R} \mid a \geq f(\mathbf{x}) \text{ and } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}. \tag{74}$$

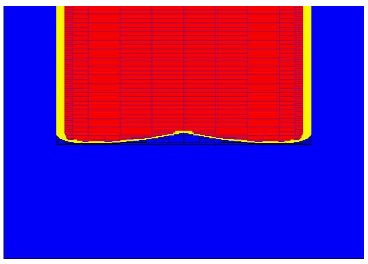
For many applications (such as detecting non-identifiability problems), it is of interest to have a guaranteed graphical representation of the sets

$$\mathbb{S}_i = \{(x_i, a) \in \mathbb{R} \times \mathbb{R} \mid \exists (x_1,\ldots,x_{i-1},x_i,\ldots,x_n) \mid a \geq f(\mathbf{x}) \text{ and } \mathbf{g}(\mathbf{x}) \leq \mathbf{0}\}, i \in \{1,\ldots,n\}. \tag{75}$$

The set $\mathbb{S}_i$ is called the ith *profile* of the epigraph.
An enclosure of $\mathbb{S}_i$ gives an interval containing the global minimum as well as some information about the possible existence of quasi-global minimizers. Consider, for instance, the following problem

$$\min_{\mathbf{x}\in\mathbb{R}^n} \sin x_1 x_2 \text{ s.t. } x_1^2 + x_2^2 \in [1,2]. \tag{76}$$

The sets $\mathbb{S}_1$ (and also $\mathbb{S}_2$, since the two sets are equal), represented below, has been obtained by Proj2D.



Example: Assume that the criterion to be minimized is given by

$$f(\mathbf{p}) = \max_{t\in\{1,2,3\}} \left| e^{-p_1 t} + 1.01.e^{-p_2 t} - y_t \right|$$

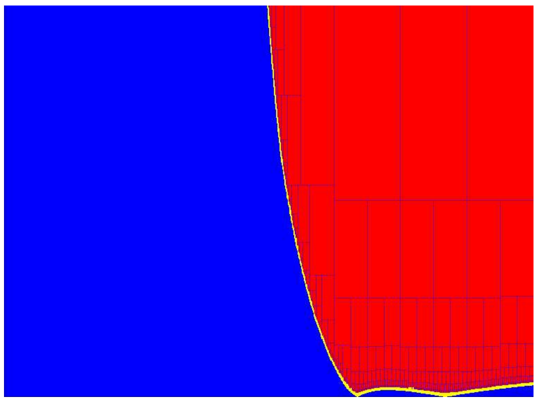
where

$$y_1 = 0.504,\ y_2 = 0.153 \text{ and } y_3 = 0.052.$$

This problem corresponds to a discrete minimax estimation problem in the case where the model is almost non-identifiable. In this example, the true values for the parameters are $p_1 = 1$ and $p_2 = 2$. Its epigraph projection can be obtained by the following Proj2D program.

```
Variables
  p1 in [-3,3]
  p2 in [-3,3]
  a in [0,1]
Constraints
 max(abs(exp(-p1*1)+1.01*exp(-p2*1)-0.504),
     abs(exp(-p1*2)+1.01*exp(-p2*2)-0.153),
     abs(exp(-p1*3)+1.01*exp(-p2*3)-0.052)) -a in [-1000,0]
Projected variables
  p1;a;
Epsilon
  0.05
EndOfFile
```

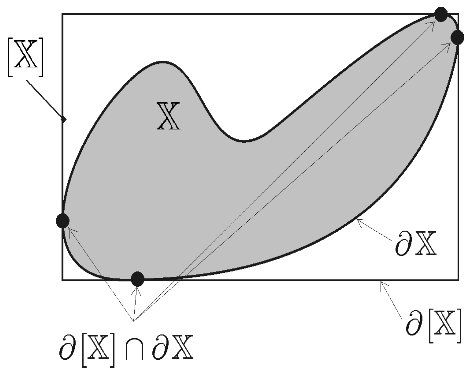
The corresponding picture on the (p_1, a)-space is as below. Even if there is here a unique global minimizer $\mathbf{p} = (1, 2)$, this picture shows that there is another local minimizer $\mathbf{p} = (2, 1)$ which is almost a global one.



Interval hull

Characterizing a (full) compact set $\mathbb{X}$ may turn out to be too costly when the dimension n of $\mathbf{x}$ is high and when $\mathbb{X}$ is large, because the paving of all the boxes generated by Sivia accumulates on the boundary of $\mathbb{X}$. In the hope of computing less if less if asked for, consider now the problem of finding the *interval hull* $[\mathbb{X}]$ of $\mathbb{X}$ (the smallest box that contains it) instead of requesting a detailed characterization of $\mathbb{X}$. Given a set $\mathbb{X}$ for which a contractor is available, compute two boxes $[\mathbf{x}_{\text{in}}]$ and $[\mathbf{x}_{\text{out}}]$ such that

$$[\mathbf{x}_{\text{in}}] \subset [\mathbb{X}] \subset [\mathbf{x}_{\text{out}}].$$



For this purpose, we shall compute a guaranteed enclosure $[\underline{x}_i^-, \underline{x}_i^+]$ and $[\bar{x}_i^-, \bar{x}_i^+]$ of $\min_{\mathbf{x}\in\mathbb{X}} x_i$ and $\max_{\mathbf{x}\in\mathbb{X}} x_i, i \in \{1, \ldots, n\}$. This amounts to solving $2n$ global optimization problems. Since

$$[\mathbb{X}] = \left[\min_{\mathbf{x}\in\mathbb{X}} x_1, \max_{\mathbf{x}\in\mathbb{X}} x_1\right] \times \cdots \times \left[\min_{\mathbf{x}\in\mathbb{X}} x_n, \max_{\mathbf{x}\in\mathbb{X}} x_n\right], \tag{77}$$

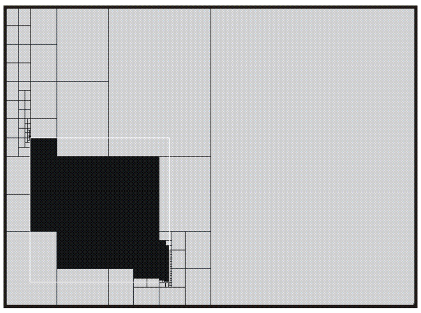
we will have

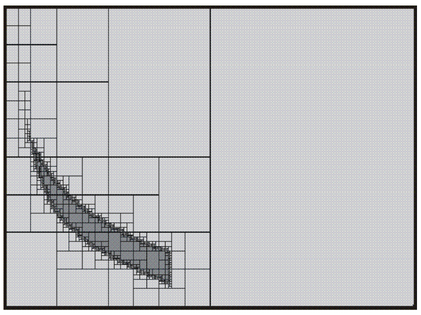
$$[\underline{x}_1^-, \underline{x}_1^+] \times \cdots \times [\underline{x}_n^-, \underline{x}_n^+] \subset [\mathbb{X}] \subset [\bar{x}_1^-, \bar{x}_1^+] \times \cdots \times [\bar{x}_n^-, \bar{x}_n^+] \tag{78}$$

Example: Consider the problem of characterizing the set

$$\mathbb{X} = \left\{ (x_1, x_2) \in [0,5]^2 \mid \forall t \in [0,1],\ |t^2 + 2t + 1 - x_1 e^{x_2 t}| \leqslant 1 \right\}. \tag{79}$$

The subpavings obtained when performing the four optimizations of are presented in figure (a) below. Compare with (b), which presents the subpavings generated by Sivia when solving the same example.


(a)


(b)

Constraints propagation for estimation

(Luc Jaulin, Wednesday, 14h30-15h45).

Constraint propagation (remainder)
A CSP is composed of

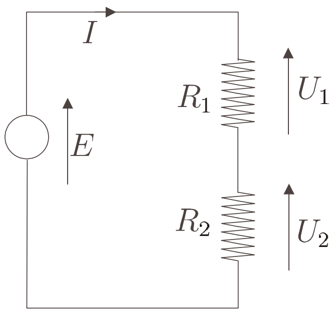
- a set of variables $\mathcal{V} = \{x_1, \dots, x_n\}$,
- a set of constraints $\mathcal{C} = \{c_1, \dots, c_m\}$ and
- a set of interval domains $\{[x_1], \dots, [x_n]\}$.

The aim of propagation techniques is to contract as much as possible the domains for the variables without loosing any solution. Denote by $[\mathbf{x}] \sqcap c_j$, the smallest box which contains all points in $[\mathbf{x}] = [x_1] \times \cdots \times [x_n]$ that satisfy c_j. The principle generally used to contract the $[x_i]$'s is *arc consistency*. It consists in computing the box

$$((((((([\mathbf{x}] \sqcap c_1) \sqcap c_2) \sqcap \dots) \sqcap c_m) \sqcap c_1) \sqcap c_2) \dots , \tag{80}$$

until a steady box (also called the fixed point) is reached.

Constraint propagation for estimation (Collaboration with I. Braems, M. Kieffer, E. Walter)
Example 1: Consider the electronic circuit consisting of one battery and two resistors represented below



Assume that measures have been collected on this circuit, leading to the following relations :

$$\begin{aligned} E &\in [23V, 26V],\ I \in [4A, 8A],\ U_1 \in [10V, 11V], \\ U_2 &\in [14V, 17V],\ P \in [124W, 130W], \end{aligned} \tag{81}$$

where P is the power delivered by the battery. Nothing is known about the values of the resistors except that they are positive. Thus the prior domains for R_1 and R_2 are $]0, \infty[$. These quantities are related by the following constraints:

$$\begin{aligned} P &= EI;\ E = (R_1 + R_2) I; \\ U_1 &= R_1 I;\ U_2 = R_2 I;\ E = U_1 + U_2. \end{aligned} \tag{82}$$

Since the second constraint is not primitive, it is decomposed by introducing an auxiliary variable, say R, as follows:

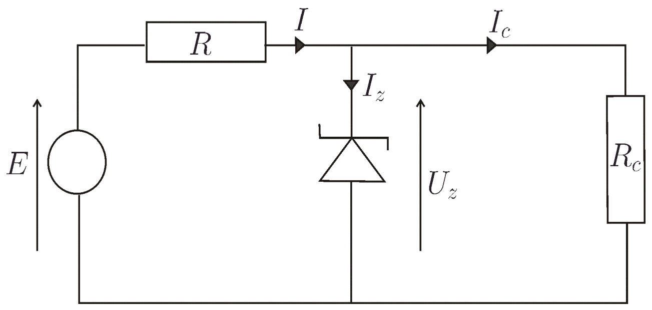
$$E = (R_1 + R_2) I \text{ is decomposed into } (E = RI;\ R = R_1 + R_2) . \tag{83}$$

The solver IntervalPeeler (`www.istia.univ-angers.fr/~baguenar/`) generates the following results

$$\begin{aligned} R_1 &\in [1.84\Omega, 2.31\Omega] ,\ R_2 \in [2.58\Omega, 3.35\Omega], \\ I &\in [4.769A, 5.417A] ,\ U_1 \in [10V; 11V] ,\ U_2 \in [14V; 16V] , \\ E &\in [24V; 26V] ,\ P \in [124W, 130W] . \end{aligned} \tag{84}$$

We got rather accurate intervals containing R_1 and R_2. The domains for I and U_2 have also been contracted, whereas the domains for U_1 and P have been left unchanged. ■

Example 2: Consider now the following circuit



where it is known that

$$\begin{array}{l} U_z \in [6V,7V], r \in [7,8]\Omega, U_0 \in [6,6.2]V \\ R \in [100,110]\Omega, E \in [18,20]V, I_z \in [0.001,\infty]A \\ I \in]-\infty,\infty[A,\ \ I_c \in]-\infty,\infty[A, R_c \in [50,60]\Omega. \end{array} \tag{85}$$

The constraints described by the circuit are given by

$$\begin{array}{ll} \text{Zener diode} & I_z = \max(0, \frac{U_z-U_0}{r}), \\ \text{Ohm rule} & U_z = R_c I_c, \\ \text{Current rule} & I = I_c + I_z, \\ \text{Voltage rule} & E = RI + U_z. \end{array} \tag{86}$$

The solver IntervalPeeler contracts the domains into:

$$\begin{array}{l} U_z \in [6,007;6,518], r \in [7,8]\Omega, U_0 \in [6,6.2]V \\ R \in [100,110]\Omega, E \in [18,20]V, I_z \in [0.001,0.398]A \\ I \in [0.11;0.14]A,\ \ I_c \in [0.1;0,13]A, R_c \in [50,60]\Omega \end{array} \tag{87}$$

■

Forward-backward propagation

Forward–backward propagation selects the primitive constraints to be used for contractions in an optimal order. Consider the constraint

$$f(\mathbf{x}) \in [y], \tag{88}$$

where

$$f(\mathbf{x}) = x_1 \exp(x_2) + \sin(x_3). \tag{89}$$

The domains for the variables x_1, x_2 and x_3 are denoted by $[x_1], [x_2]$ and $[x_3]$. To obtain an algorithm contracting these domains, first write an algorithm that computes $y = f(\mathbf{x})$, by a finite sequence of elementary operations.

$$\begin{array}{lll} a_1 & := & \exp(x_2); \\ a_2 & := & x_1 a_1; \\ a_3 & := & \sin(x_3); \\ y & := & a_2 + a_3. \end{array}$$

Then write an interval counterpart to this algorithm:

```
1    $[a_1] := \exp([x_2]);$
2    $[a_2] := [x_1] * [a_1];$
3    $[a_3] := \sin([x_3]);$
4    $[y] := [y] \cap [a_2] + [a_3].$
```

If $[y]$ as computed at Step 4 turns out to be empty, then we know that the CSP has no solution. Finally, a backward propagation is performed, updating the domains associated with all the variables to get

```
5    $[a_2] := ([y] - [a_3]) \cap [a_2];$    // see Step 4
6    $[a_3] := ([y] - [a_2]) \cap [a_3];$    // see Step 4
7    $[x_3] := \sin^{-1}([a_3]) \cap [x_3];$    // see Step 3
8    $[a_1] := ([a_2]/[x_1]) \cap [a_1];$    // see Step 2
9    $[x_1] := ([a_2]/[a_1]) \cap [x_1];$    // see Step 2
10   $[x_2] := \log([a_1]) \cap [x_2].$    // see Step 1
```

At Step 8, $\sin^{-1}([a_3]) \cap [x_3]$ returns the smallest interval containing $\{x_3 \in [x_3] \mid \sin(x_3) \in [a_3]\}$. The associated contractor is given below

```
Algorithm $\mathcal{C}_{\downarrow\uparrow}$(inout: $[\mathbf{x}]$)
1   $[a_1] := \exp([x_2])$;
2   $[a_2] := [x_1] * [a_1]$;
3   $[a_3] := \sin([x_3])$;
4   $[y] := [y] \cap ([a_2] + [a_3])$;
5   $[a_2] := ([y] - [a_3]) \cap [a_2]$;
6   $[a_3] := ([y] - [a_2]) \cap [a_3]$;
7   $[x_3] := \sin^{-1}([a_3]) \cap [x_3]$;
8   $[a_1] := ([a_2]/[x_1]) \cap [a_1]$;
9   $[x_1] := ([a_2]/[a_1]) \cap [x_1]$;
10  $[x_2] := \log([a_1]) \cap [x_2]$.
```

Application to state estimation

Consider the non-linear discrete-time system

$$\begin{cases} \begin{pmatrix} x_1(k) \\ x_2(k) \end{pmatrix} & = \begin{pmatrix} 0.1x_1(k-1) + x_2(k-1)\exp(x_1(k-1)) \\ x_1(k-1) + 0.1x_2^2(k-1) + \sin(k) \end{pmatrix}, \\ y(k) & = x_2(k)/x_1(k), \end{cases} \tag{90}$$

with $k \in \{1, \ldots, 15\}$. Interval data have been generated as follows. First, starting from the true value $\mathbf{x}^*(0) = (-1\ 0)^{\mathrm{T}}$ of the initial state vector, the true values $\mathbf{x}^*(k)$ and $y^*(k), k \in \{1, \ldots, 15\}$ were computed by simulation. To each noise-free output $y^*(k)$ a random error was then added, with a uniform distribution in $[-e, e]$, to generate noisy data $\check{y}(k)$. Finally, the prior domains for $y(k)$ was taken equal to $[\check{y}(k)] = [\check{y}(k) - e, \check{y}(k) + e]$. $[\check{y}(k)]$ is thus guaranteed to contain the unknown noise-free output $y^*(k)$. The problem to be solved is then: *given the equations of the system, the interval data $[\check{y}(k)]$, and bounded intervals $[\check{x}_1(0)]$ and $[\check{x}_2(0)]$ containing the initial state variables $x_1(0)$ and $x_2(0)$, compute (accurate) interval enclosures for the values of the variables $x_1(k)$, $x_2(k)$ and $y(k)$, $k = 1, \ldots, 15$.*

```
Algorithm $\phi$(in: $x_1(0), x_2(0)$; out: $y(1), \ldots, y(15)$)
1   for $k := 1$ to 15,
2       $x_1(k) := 0.1 * x_1(k-1) + x_2(k-1) * \exp(x_1(k-1))$;
3       $x_2(k) := x_1(k-1) + 0.1 * x_2^2(k-1) + \sin(k)$;
4       $y(k) := x_2(k)/x_1(k)$.
```

This simulator can be decomposed into primitive statements as follows

```
Algorithm $\phi$(in: $x_1(0), x_2(0)$; out: $y(1), \ldots, y(15)$)
1   for $k := 1$ to 15,
2       $z_1(k) := \exp(x_1(k-1))$;
3       $z_2(k) = x_2(k-1) * z_1(k)$;
4       $x_1(k) := 0.1 * x_1(k-1) + z_2(k)$;
5       $z_3(k) := 0.1 * \mathrm{sqr}(x_2(k-1))$;
6       $z_4(k) := z_3(k) + \sin(k)$;
7       $x_2(k) := x_1(k-1) + z_4(k)$;
8       $y(k) := x_2(k)/x_1(k)$.
```

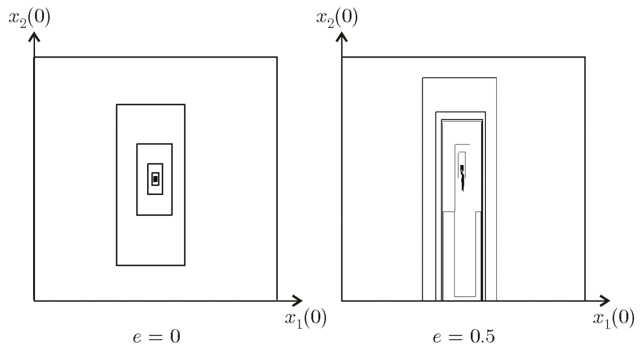
The contractor for the set $\hat{\mathbb{X}}(0)$ is in the following table.

```
Algorithm $\mathcal{C}_{\hat{\mathbb{X}}(0)}$(in: $[y(1)], \ldots, [y(15)]$; inout: $[x_1(0)], [x_2(0)]$)
1  for $k := 1$ to 15
2     $[x_1(k)] := [-\infty, \infty]$ ; $[x_2(k)] := [-\infty, \infty]$ ;
3     $[z_1(k)] := [-\infty, \infty]$ ; $[z_2(k)] := [-\infty, \infty]$ ;
4     $[z_3(k)] := [-\infty, \infty]$ ; $[z_4(k)] := [-\infty, \infty]$ ;
6  do
7     for $k := 1$ to 15,                                  // forward
8        $[z_1(k)] := [z_1(k)] \cap \exp([x_1(k-1)])$;
9        $[z_2(k)] := [z_2(k)] \cap ([x_2(k-1)] * [z_1(k)])$ ;
10       $[x_1(k)] := [x_1(k)] \cap (0.1 * [x_1(k-1)] + [z_2(k)])$ ;
11       $[z_3(k)] := [z_3(k)] \cap (0.1 * \mathrm{sqr}([x_2(k-1)]))$ ;
12       $[z_4(k)] := [z_4(k)] \cap ([z_3(k)] + \sin(k))$ ;
13       $[x_2(k)] := [x_2(k)] \cap ([x_1(k-1)] + [z_4(k)])$ ;
14       $[y(k)] := [y(k)] \cap ([x_2(k)]/[x_1(k)])$ ;
15    for $k := 15$ down to 1,                             // backward
16       $[x_2(k)] := [x_2(k)] \cap ([y(k)] * [x_1(k)])$ ;
17       $[x_1(k)] := [x_1(k)] \cap ([x_2(k)]/[y(k)])$ ;
18       $[x_1(k-1)] := [x_1(k-1)] \cap ([x_2(k)] - [z_4(k)])$ ;
19       $[z_4(k)] := [z_4(k)] \cap ([x_2(k)] - [x_1(k-1)])$ ;
20       $[z_3(k)] := [z_3(k)] \cap ([z_4(k)] - \sin(k))$ ;
21       $[x_2(k-1)] := [x_2(k-1)] \cap \left(0.1 * \mathrm{sqr}^{-1}([z_3(k)])\right)$ ;
22       $[x_1(k-1)] := [x_1(k-1)] \cap (10 * ([x_1(k)] - [z_2(k)]))$ ;
23       $[z_2(k)] := [z_2(k)] \cap ([x_1(k)] - 0.1 * [x_1(k-1)])$ ;
24       $[x_2(k-1)] := [x_2(k-1)] \cap ([z_2(k)]/[z_1(k)])$ ;
25       $[z_1(k)] := [z_1(k)] \cap ([z_2(k)]/[x_2(k-1)])$ ;
26       $[x_1(k-1)] := [x_1(k-1)] \cap \log([z_1(k)])$;
27 while contraction is significant.
```

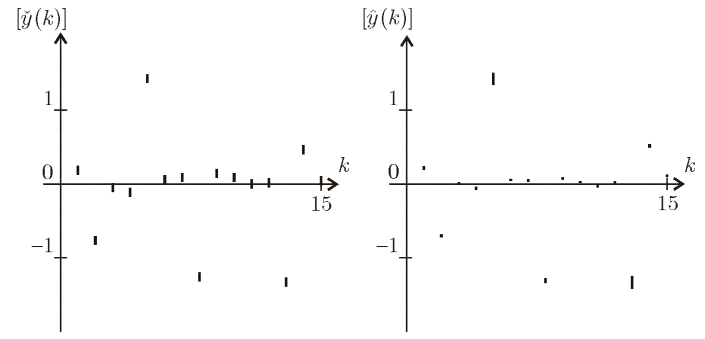
The prior domains for the components of the initial state vector were taken as

$$[\check{x}_1(0)] = [-1.2, -0.8], \qquad [\check{x}_2(0)] = [-0.2, 0.2]. \tag{91}$$

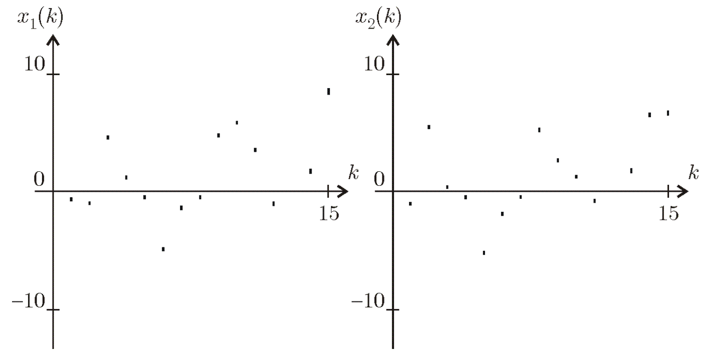
In the absence of noise (*i.e.*, $e = 0$), the contractor is able to find the actual values of all the variables with an accuracy of 8 digits in 0.1 s on a Pentium 133. No bisection turned out to be necessary to get this result. The boxes drawn on the left part of the figure are those obtained after each iteration of the contractor $\mathcal{C}_{\downarrow\uparrow}$. For $e = 0.5$ (*i.e.*, in the presence of noise), the volume of $\hat{\mathbb{X}}(0)$ is no longer equal to zero, and thus, even with an ideal contractor, bisections have to be performed (see the right part of the figure).



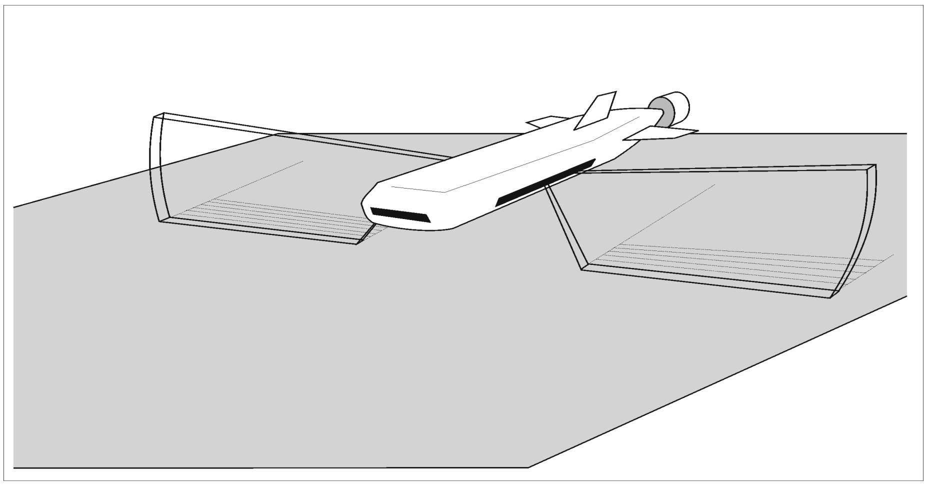
Always in the case where noise is present ($e = 0.5$), interval output data are on the following figure (left). The corresponding contracted interval outputs $[\hat{y}(k)]$ containing $y^*(k)$ are on the right.



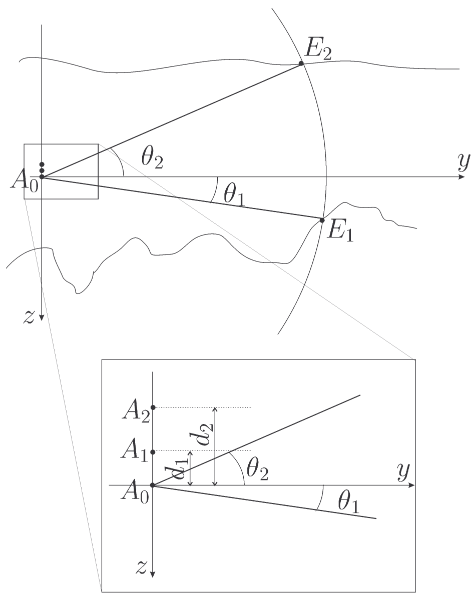
Contracted domains for $x_1(k)$ (left) and $x_2(k)$ (right) as functions of k are represented below.



Estimation of the bathymetry (or bottom relief) of the ocean with a sonar (Collaboration with M. Legris) The figure represents an autonomous underwater vehicle (AUV) equipped with two lateral sonars. At each sample, the sonar measures a signal resulting from the echo of the acoustic wave emitted by the sonar itself onto some obstacles. These echoes can be assumed to belong to a unique plane, as illustrated by the figure.



The sonar to be considered has three antennas A_0, A_1, A_2. The acoustic wave emitted by A_0 is $s(t) = e^{2\pi f_0 t}$, where $f_0 = 455$ kHz. Since the velocity of the wave in the water is $c = 1500$ ms^{-1}, the associated wave length is $\lambda = \frac{c}{f_0} = \frac{1500}{455000} = 3$ mm. The figure below represents a situation where the echo measured at time t results from two obstacles E_1,E_2. Note that we have $t = 2cr$ where $r = \text{dist}(A_0, E_1) = \text{dist}(A_0, E_2)$.



The sensors $A_m, m = 0, 1, 2$ receive the signal

$$\widehat{s}_m(t) = \sum_{n=1}^{n_{\max}} \alpha_n e^{j2\pi f_0 t + j2\pi f_0 \frac{r}{c} + j\varphi_n} e^{j2\pi f_0 \frac{d_m \sin\theta_n}{c}}, \tag{92}$$

where $n_{\max}$ represents the number of existing obstacles located at a distance r from A_0 and $\varphi_n(r)$ is an unknown phase difference resulting on the superposition of many microscopic reflections. We shall take $d_0 = 0, d_1 = 1.5\lambda = 4.94$mm and $d_2 = 4\lambda = 13.187$ mm. The Fresnel transformation associates to the signal $\widehat{s}(t)$, the signal $\widehat{s}(t)/e^{j2\pi f_0 t}$. Thus, the Fresnel transformation of the previous signal gives:

$$s_m(r) = \sum_{n=1}^{n_{\max}} \alpha_n e^{j2\pi f_0 \frac{r}{c} + j\varphi_n} e^{j2\pi f_0 \frac{d_m \sin\theta_n}{c}} = \sum_{n=1}^{n_{\max}} \alpha_n e^{j\rho_n} e^{j2\pi f_0 \frac{d_m \sin\theta_n}{c}},$$

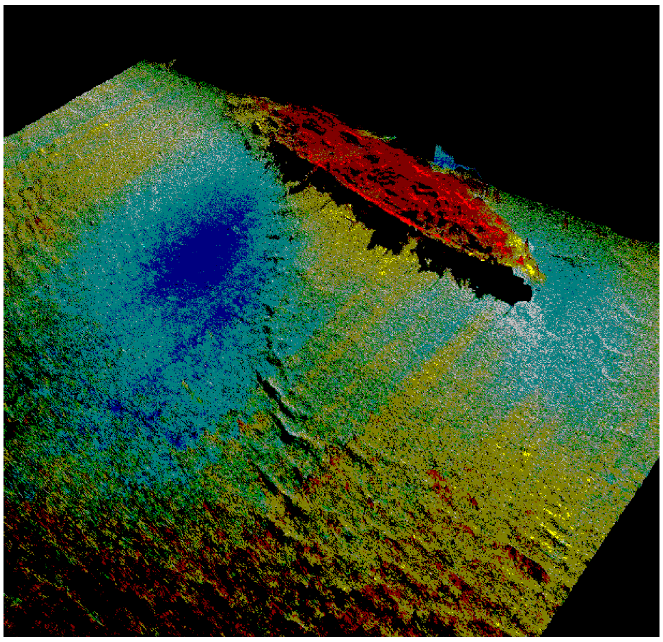
where $\rho_n(r) = 2\pi f_0 \frac{r}{c} + \varphi_n(r)$. For each r, we have a system of 6 equations with $3n_{\max}$ unknowns (the α_n's, the θ_n's and the ρ_n's). When $n_{\max} = 1$, an analytical resolution of these equations can be obtained and a three dimensional view of the bottom can be reconstructed.

$$\begin{aligned}
s_m(r) &= \alpha_1 e^{j\rho_1} e^{j2\pi f_0 \frac{d_m \sin\theta_1}{c}} \\
&= \alpha_1 (\cos\rho_1 + j\sin\rho_1)\left(\cos\left(2\pi f_0 \frac{d_m \sin\theta_1}{c}\right) + j\sin\left(2\pi f_0 \frac{d_m \sin\theta_1}{c}\right)\right) \\
&= \alpha_1 \left(\cos\rho_1 \cos\left(2\pi f_0 \frac{d_m \sin\theta_1}{c}\right) - \sin\rho_1 \sin\left(2\pi f_0 \frac{d_m \sin\theta_1}{c}\right)\right) \\
&\quad + j\alpha_n \left(\sin\rho_1 \cos\left(2\pi f_0 \frac{d_m \sin\theta_1}{c}\right) + \cos\rho_n \sin\left(2\pi f_0 \frac{d_m \sin\theta_1}{c}\right)\right).
\end{aligned} \tag{93}$$

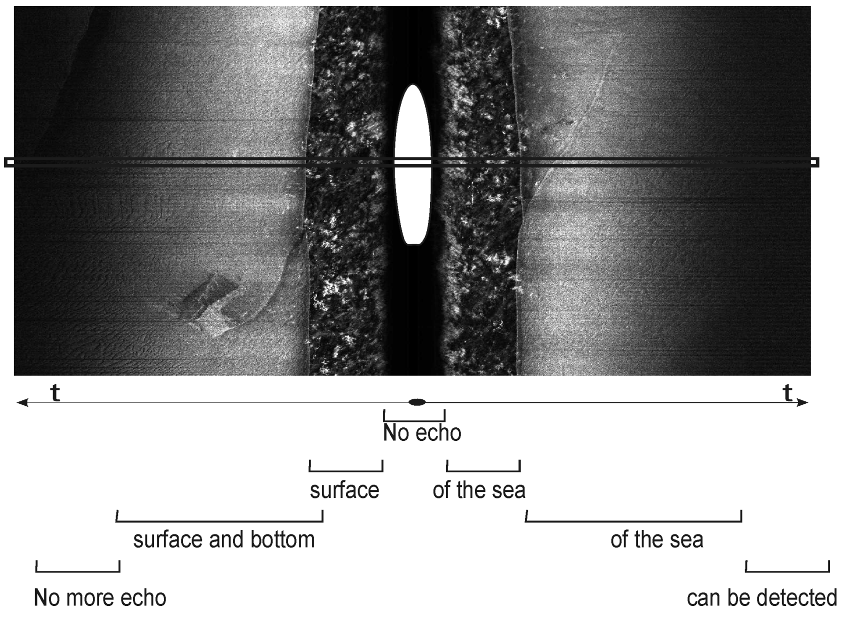
Since, $\cos a \cos b - \sin a \sin b = \cos(a+b)$ and $\sin a \cos b + \cos a \sin b = \sin(a+b)$, we get

$$\begin{aligned}
\mathrm{Re}(s_m) &= \alpha_1 \cos\left(\rho_1 + 2\pi f_0 \frac{d_m \sin\theta_1}{c}\right) \\
\mathrm{Im}(s_m) &= \alpha_1 \sin\left(\rho_1 + 2\pi f_0 \frac{d_m \sin\theta_1}{c}\right)
\end{aligned} \tag{94}$$

From the knowledge of $s_m(r)$ for $m = 1, 2$ we are then able to compute directly $\alpha_1(r), \rho_1(r), \theta_1(r)$. The results obtained by this method is illustrated by the figure below.



In practice, more that one obstacle should be taken into account as illustrated by the following picture. These data have been given by the GESMA (Groupe d'Etudes Sous Marines de l'Atlantique). They have been collected by a Klein 5000 with $f_0 = 455$ kHz and a resolution of 20 cm × 6cm. The AUV is represented in white. The black bar in the center illustrates the absence of echo. Then the surface of the sea is first detected. In the next step the echo corresponds to a superposition of an echo from the surface and an echo from the bottom. The black horizontal thin window corresponds to an echo signal.



Consider now the case $n_{\max} = 2$. The equations become

$$\begin{aligned}
s_m(r) &= \sum_{n=1}^{n_{\max}} \alpha_n e^{j\rho_n} e^{j2\pi f_0 \frac{d_m \sin\theta_n}{c}} \\
&= \sum_{n=1}^{n_{\max}} \alpha_n \left(\cos\rho_n + j\sin\rho_n\right)\left(\cos\left(2\pi f_0 \frac{d_m\sin\theta_n}{c}\right) + j\sin\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right)\right) \\
&= \sum_{n=1}^{n_{\max}} \alpha_n \left(\cos\rho_n \cos\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right) - \sin\rho_n\sin\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right)\right) \\
&\quad + j\sum_{n=1}^{n_{\max}} \alpha_n\left(\sin\rho_n\cos\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right) + \cos\rho_n\sin\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right)\right).
\end{aligned} \tag{95}$$

Thus

$$\begin{aligned}
\mathrm{Re}(s_m) &= \sum_{n=1}^{n_{\max}} \alpha_n\left(\cos\rho_n\cos\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right) - \sin\rho_n\sin\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right)\right) \\
\mathrm{Im}(s_m) &= \sum_{n=1}^{n_{\max}} \alpha_n\left(\sin\rho_n\cos\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right) + \cos\rho_n\sin\left(2\pi f_0\frac{d_m\sin\theta_n}{c}\right)\right).
\end{aligned} \tag{96}$$

Since, $\cos a\cos b - \sin a \sin b = \cos(a+b)$ and $\sin a\cos b + \cos a\sin b = \sin(a+b)$, we get

$$\begin{aligned}
\mathrm{Re}(s_m) &= \sum_{n=1}^{n_{\max}} \alpha_n \cos\left(\rho_n + 2\pi f_0\frac{d_m\sin\theta_n}{c}\right) \\
\mathrm{Im}(s_m) &= \sum_{n=1}^{n_{\max}} \alpha_n \sin\left(\rho_n + 2\pi f_0\frac{d_m\sin\theta_n}{c}\right)
\end{aligned} \tag{97}$$

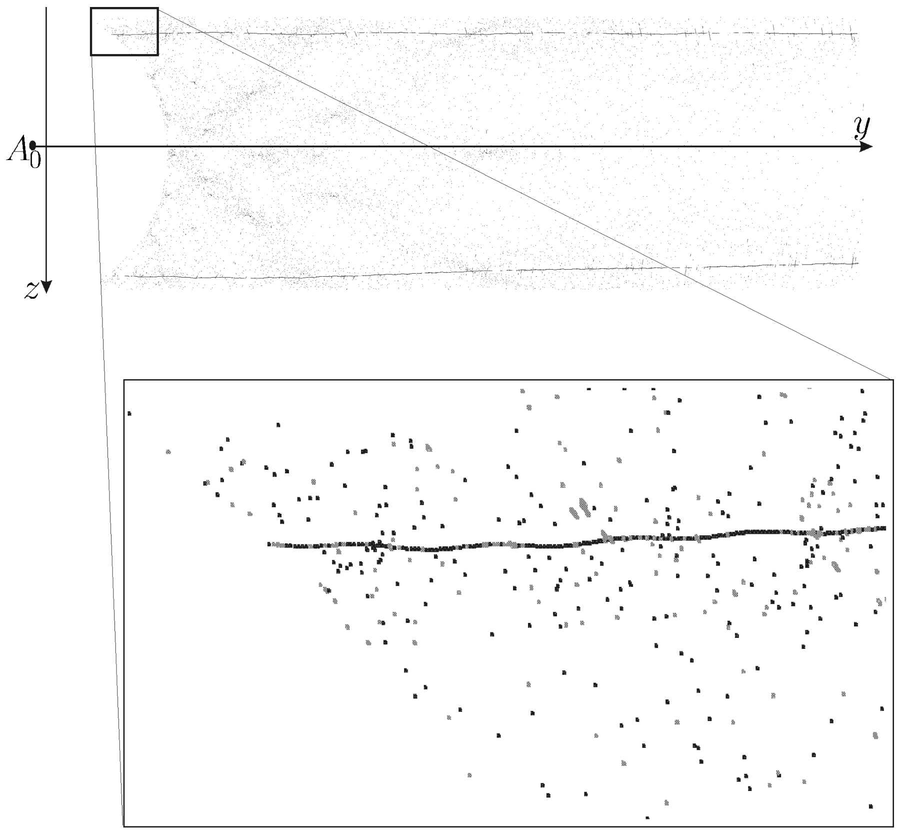
The equations to be solved for each $r \in \{15m, 15.03m, 15.06m, \ldots, 150m\}$ are

$$\begin{aligned}
s_0^{\mathrm{Re}} &= \alpha_1\cos\rho_1 + \alpha_2\cos\rho_2 \\
s_0^{\mathrm{Im}} &= \alpha_1\sin\rho_1 + \alpha_2\sin\rho_2 \\
s_1^{\mathrm{Re}} &= \alpha_1\cos\left(\rho_1 + 2\pi f_0\frac{d_1\sin\theta_1}{c}\right) + \alpha_2\cos\left(\rho_2 + 2\pi f_0\frac{d_1\sin\theta_2}{c}\right) \\
s_1^{\mathrm{Im}} &= \alpha_1\sin\left(\rho_1 + 2\pi f_0\frac{d_1\sin\theta_1}{c}\right) + \alpha_2\sin\left(\rho_2 + 2\pi f_0\frac{d_1\sin\theta_2}{c}\right) \\
s_2^{\mathrm{Re}} &= \alpha_1\cos\left(\rho_1 + 2\pi f_0\frac{d_2\sin\theta_1}{c}\right) + \alpha_2\cos\left(\rho_2 + 2\pi f_0\frac{d_2\sin\theta_2}{c}\right) \\
s_2^{\mathrm{Im}} &= \alpha_1\sin\left(\rho_1 + 2\pi f_0\frac{d_2\sin\theta_1}{c}\right) + \alpha_2\sin\left(\rho_2 + 2\pi f_0\frac{d_2\sin\theta_2}{c}\right).
\end{aligned} \tag{98}$$

Set $\mathbf{y} = \left(s_0^{\mathrm{Re}}, s_0^{\mathrm{Im}}, s_1^{\mathrm{Re}}, s_1^{\mathrm{Im}}, s_2^{\mathrm{Re}}, s_2^{\mathrm{Im}}\right)$, and $\mathbf{x} = (\theta_1, \theta_2)$, we get the following state equations

$$\begin{aligned}
\mathbf{x}(r+dr) &= \mathbf{x}(r) + \mathbf{b}_{\mathbf{x}}(r) \\
\mathbf{y}(r) &= \begin{pmatrix}
\alpha_1\cos\rho_1 + \alpha_2\cos\rho_2 \\
\alpha_1\sin\rho_1 + \alpha_2\sin\rho_2 \\
\alpha_1\cos\left(\rho_1 + 2\pi f_0\frac{d_1\sin x_1}{c}\right) + \alpha_2\cos\left(\rho_2 + 2\pi f_0\frac{d_1\sin x_2}{c}\right) \\
\alpha_1\sin\left(\rho_1 + 2\pi f_0\frac{d_1\sin x_1}{c}\right) + \alpha_2\sin\left(\rho_2 + 2\pi f_0\frac{d_1\sin x_2}{c}\right) \\
\alpha_1\cos\left(\rho_1 + 2\pi f_0\frac{d_2\sin x_1}{c}\right) + \alpha_2\cos\left(\rho_2 + 2\pi f_0\frac{d_2\sin x_2}{c}\right) \\
\alpha_1\sin\left(\rho_1 + 2\pi f_0\frac{d_2\sin x_1}{c}\right) + \alpha_2\sin\left(\rho_2 + 2\pi f_0\frac{d_2\sin x_2}{c}\right)
\end{pmatrix}
\end{aligned} \tag{99}$$

where $\mathbf{b}_{\mathbf{x}}(r)$ represents the feasible variations for $\mathbf{x}$ and $\rho_1, \rho_2, \alpha_1, \alpha_2$ represent the non-additive noise output. In the case where $[\mathbf{b}_{\mathbf{x}}] = [-\infty, \infty]^2$, the state estimation amounts to solving $\frac{150-15}{0.03} = 4500$ systems of 6 nonlinear equations with 6 unknowns. For a realistic situation, the solutions, obtained in less than 1 hour, have been represented on the figure below. For dark points, the existence of a unique solution has be proved using the Newton operator. The small circle represents the sonar. For $[\mathbf{b}_{\mathbf{x}}] = [0.1, 0.1]^2$, the two curves representing the surface of the sea and the bottom can be retrieved without any parasite solutions.


y
A_0
z

Robust stability of linear systems

(Luc Jaulin, Thursday, 10h00-11h00).

Routh criterion

Let $P(s) = a_n s^n + \cdots + a_1 s + a_0$ be a polynomial. Its Routh table is defined by

a_n	a_{n-2}	a_{n-4}	a_{n-6}	$\ldots$	0	0
a_{n-1}	a_{n-3}	a_{n-5}	a_{n-7}	$\ldots$	0	0
b_1	b_2	b_3			0	0
c_1	c_2	c_3			0	0
$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$

with

$$\begin{array}{ccc} b_1 = \frac{a_{n-1}a_{n-2}-a_n a_{n-3}}{a_{n-1}} & b_2 = \frac{a_{n-1}a_{n-4}-a_n a_{n-5}}{a_{n-1}} & \ldots \\ c_1 = \frac{b_1 a_{n-3}-a_{n-1}b_2}{b_1} & c_2 = \frac{b_1 a_{n-5}-a_{n-1}b_3}{b_1} & \ldots \\ \ldots & \vdots & \end{array} \qquad (100)$$

Note that the first two lines are the coefficients of $P(s)$. The other elements of the table t_{ij} are obtained by the following relation

$$t_{ij} = \frac{t_{i-1,1}t_{i-2,j+1} - t_{i-2,1}t_{i-1,j+1}}{t_{i-1,1}}. \qquad (101)$$

The roots of $P(s)$ are all stable (i.e., with negative real parts) if and only if all entries of the first column of the table have the same sign.

Stability domain

The *stability domain* $\mathbb{S}_{\mathrm{p}}$ of the polynomial

$$P(s,\mathbf{p}) = s^n + a_{n-1}(\mathbf{p})s^{n-1} + \ldots + a_1(\mathbf{p})s + a_0(\mathbf{p}) \qquad (102)$$

is the set of all $\mathbf{p}$ such that $P(s,\mathbf{p})$ is stable. Consider for instance the polynomial $P(s,\mathbf{p})$ given by

$$s^3 + (p_1+p_2+2)s^2 + (p_1+p_2+2)s + 2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2, \qquad (103)$$

$\sigma = 0.5$. Its Routh table is given by

1	p_1+p_2+2
p_1+p_2+2	$2p_1p_2+6p_1+6p_2+2+\sigma^2$
$\frac{(p_1+p_2+2)^2-2p_1p_2+6p_1+6p_2+2+\sigma^2}{p_1+p_2+2} = \frac{(p_1-1)^2+(p_2-1)^2-\sigma^2}{p_1+p_2+2}$	0
$2p_1p_2+6p_1+6p_2+2+\sigma^2 = 2(p_1+3)(p_2+3)-16+\sigma^2$	0

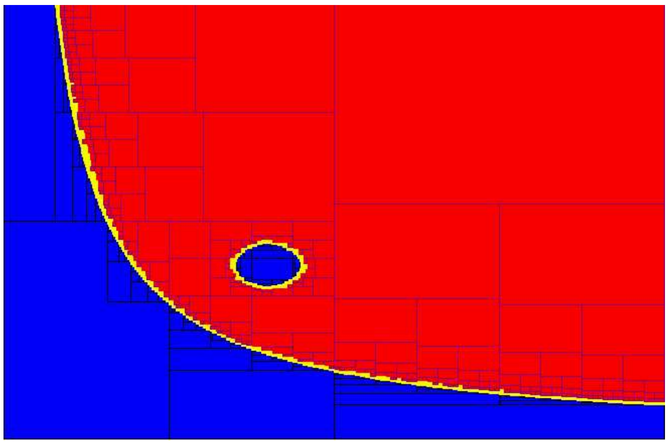
Its stability domain is thus defined by

$$\mathbb{S}_{\mathrm{p}} \overset{\text{def}}{=} \{\mathbf{p} \in \mathbb{R}^n \mid \mathbf{r}(\mathbf{p}) > \mathbf{0}\} = \mathbf{r}^{-1}\left(]0,+\infty[^{\times n}\right). \qquad (104)$$

where

$$\mathbf{r}(\mathbf{p}) = \begin{pmatrix} p_1+p_2+2 \\ (p_1-1)^2+(p_2-1)^2-\sigma^2 \\ 2(p_1+3)(p_2+3)-16+\sigma^2 \end{pmatrix}. \qquad (105)$$

The corresponding set, obtained by PROJ2D (`www.istia.univ-angers.fr/~dao/`) is represented on the following figure.



Robust stability of a controlled motorbike (Collaboration with M. Christie, L. Granvilliers, X. Baguenard)
A CSP is *infallible* if any arbitrary instantiation of all variables in their domains is a solution, *i.e.,* its solution set is equal to the Cartesian product of all its domains. To prove that a CSP is infallible, it suffices to prove that its negation has an empty solution set.
Consider for instance the CSP

$$\begin{array}{lll}\mathcal{V} & = & \{x,y\} \\ \mathcal{D} & = & \{[x],[y]\} \\ \mathcal{C} & = & \{\ f(x,y)\leq 0,\ g(x,y)\leq 0\}\,. \end{array} \tag{106}$$

The CSP is infallible if

$$\begin{array}{rl} & \forall x\in[x],\forall y\in[y],\ f(x,y)\leq 0 \text{ and } g(x,y)\leq 0, \\ \Leftrightarrow & \{(x,y)\in[x]\times[y] \mid f(x,y)>0 \text{ or } g(x,y)>0\}=\emptyset \\ \Leftrightarrow & \{(x,y)\in[x]\times[y] \mid \max\left(f(x,y),g(x,y)\right)>0\}=\emptyset. \end{array} \tag{107}$$

This task can be performed efficiently using interval constraint propagation techniques. As an illustration, consider a motorbike with a speed of 1m/s. The input of the system is the angle θ of the handlebars and the output is the rolling angle ϕ of the bike. The transfer function is

$$\phi(s)=\frac{1}{s^2-\alpha_1}\theta(s). \tag{108}$$

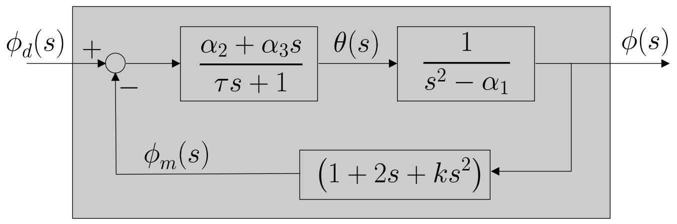
Because of the small velocity of the bike, the system is unstable (the gyroscopic effect of the front wheel is not sufficient to maintain the stability). In order to stabilize the system, we add the following controller

$$\theta(s)=\frac{\alpha_2+\alpha_3 s}{\tau s+1}\left(\phi_d(s)-\phi_m(s)\right), \tag{109}$$

where ϕ_d is the wanted rolling angle and ϕ_m is the measured rolling angle. Since the sensor is not perfect ϕ_m is not identical to the rolling angle ϕ of the bike. These quantities are related by the relation :

$$\phi_m(s)=\left(1+2s+ks^2\right)\phi(s). \tag{110}$$

The whole system is depicted on the following picture.



The input-output relation of the closed-loop system is :

$$\phi(s)=\frac{\alpha_2+\alpha_3 s}{\left(s^2-\alpha_1\right)(\tau s+1)+(\alpha_2+\alpha_3 s)\left(1+2s+ks^2\right)}\phi_d(s). \tag{111}$$

Its characteristic polynomial is thus

$$\left(s^2-\alpha_1\right)(\tau s+1)+(\alpha_2+\alpha_3 s)\left(1+2s+ks^2\right)=a_3s^3+a_2s^2+a_1s+a_0, \tag{112}$$

with $a_3=\tau+\alpha_3 k$, $a_2=\alpha_2 k+2\alpha_3+1$, $a_1=\alpha_3-\alpha_1\tau+2\alpha_2$ and $a_0=-\alpha_1+\alpha_2$. The associated Routh table is :

$$\begin{array}{|c|c|}\hline a_3 & a_1 \\ \hline a_2 & a_0 \\ \hline \frac{a_2a_1-a_3a_0}{a_2} & 0 \\ \hline a_0 & 0 \\ \hline \end{array} \tag{113}$$

The closed-loop system is thus stable if $a_3, a_2, \frac{a_2a_1-a_3a_0}{a_2}$ and a_0 have the same sign. Assume that it is known that

$$\begin{array}{rcl} \alpha_1 & \in & [\alpha_1]=[8.8;9.2]\,, \alpha_2\in[\alpha_2]=[2.8;3.2]\,, \alpha_3\in[\alpha_3]=[0.8;1.2]\,, \\ \tau & \in & [\tau]=[1.8;2.2]\,, k\in[k]=[-3.2;-2.8]. \end{array} \tag{114}$$

The system is robustly stable if it is stable for all feasible parameters, i.e.,

$$\begin{array}{l} \forall\alpha_1\in[\alpha_1]\,,\forall\alpha_2\in[\alpha_2]\,,\forall\alpha_3\in[\alpha_3]\,,\forall\tau\in[\tau]\,,\forall k\in[k]\,, \\ a_3,\ a_2,\ \frac{a_2a_1-a_3a_0}{a_2} \text{ and } a_0 \text{ have the same sign.} \end{array} \tag{115}$$

Now, we have the equivalence

$$\begin{array}{rc} & b_1,\ b_2,\ b_3 \text{ and } b_4 \text{ have the same sign} \\ \Leftrightarrow & \max\left(\min\left(b_1,b_2,b_3,b_4\right), -\max\left(b_1,b_2,b_3,b_4\right)\right)>0 \end{array} \tag{116}$$

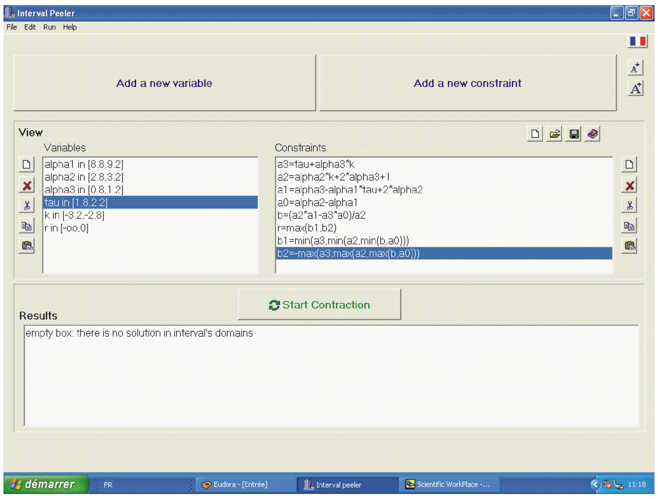
The robust stability condition amounts to proving that

$$\begin{array}{l} \exists\alpha_1\in[\alpha_1]\,,\exists\alpha_2\in[\alpha_2]\,,\exists\alpha_3\in[\alpha_3]\,,\exists\tau\in[\tau]\,,\exists k\in[k]\,, \\ \max\left(\min\left(a_3,a_2,\frac{a_2a_1-a_3a_0}{a_2},a_0\right), -\max\left(a_3,a_2,\frac{a_2a_1-a_3a_0}{a_2},a_0\right)\right)\leq 0 \end{array} \tag{117}$$

is false, i.e., that the following CSP

$$\begin{array}{rcl} \mathcal{V} & = & \{a_0,a_1,a_2,a_3,\alpha_1,\alpha_2,\alpha_3,\tau,k\}, \\ \mathcal{D} & = & \{[\alpha_0]\,,[\alpha_1]\,,[\alpha_2]\,,[\alpha_3]\,,[\alpha_2]\,,[\alpha_3]\,,[\tau]\,,[k]\}\,, \\ \mathcal{C} & = & \left\{\begin{array}{l} a_3=\tau+\alpha_3 k\ ;\ a_2=\alpha_2 k+2\alpha_3+1\ ;\ a_1=\alpha_3-\alpha_1\tau+2\alpha_2, \\ a_0=-\alpha_1+\alpha_2\ ;\ b=\frac{a_2a_1-a_3a_0}{a_2}; \\ \max\left(\min\left(a_3,a_2,b,a_0\right), -\max\left(a_3,a_2,b,a_0\right)\right)\leq 0. \end{array}\right\} \end{array} \tag{118}$$

has no solution. This task has been performed using IntervalPeeler (see the figure below).



Variables

```
alpha1 in [8.8,9.2]
alpha2 in [2.8,3.2]
alpha3 in [0.8,1.2]
tau in [1.8,2.2]
k in [-3.2,-2.8]
r in [-oo,0]
```

Constraints

```
a3=tau+alpha3*k
a2=alpha2*k+2*alpha3+1
a1=alpha3-alpha1*tau+2*alpha2
a0=alpha2-alpha1
b=(a2*a1-a3*a0)/a2
r=max(b1,b2)
b1=min(a3,min(a2,min(b,a0)))
b2=-max(a3,max(a2,max(b,a0)))
```

Results

```
empty box: there is no solution in interval's domains
```

Analysis of a time-delay system (Collaboration with M. Dao, M. Di Loreto, J.F. Lafay and J.J. Loiseau)
Consider the following linear system

$$\ddot{y}(t) - \ddot{y}(t-1) + 2\dot{y}(t) - \dot{y}(t-1) + y(t) = u(t). \tag{119}$$

Its transfer function is

$$H(s) = \frac{1}{(s+1)\,(s(1-e^{-s})+1)}. \tag{120}$$

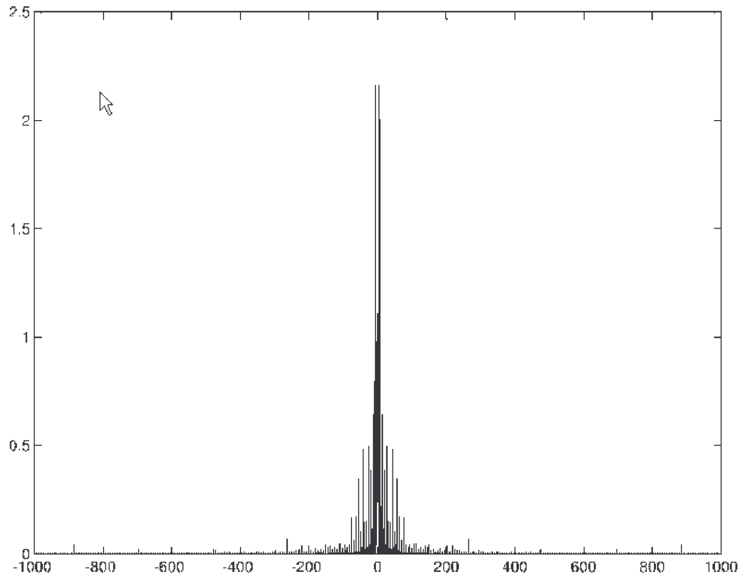
Its magnitude transfer function is

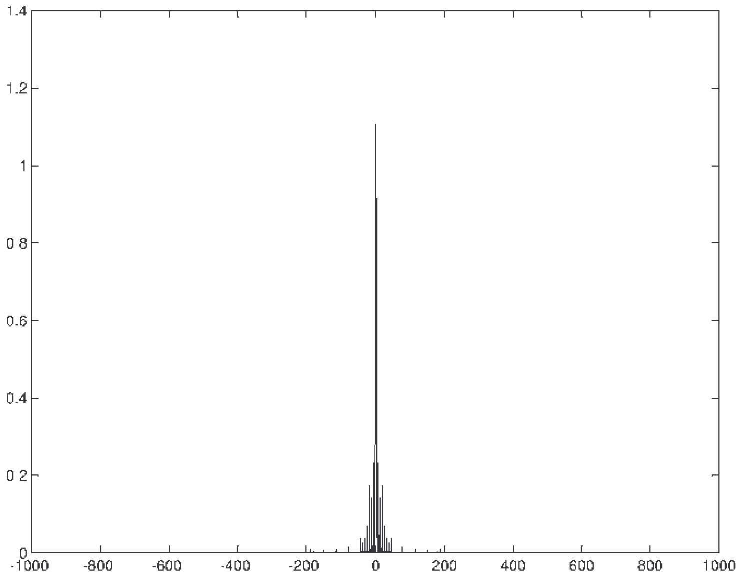
$$G(\omega) = |H(j\omega)| = \frac{1}{\sqrt{(1+\omega^2)}.\sqrt{(1-\omega\sin(\omega))^2 + \omega^2\,(1-\cos(\omega))^2}}. \tag{121}$$

The Bode diagram is defined by

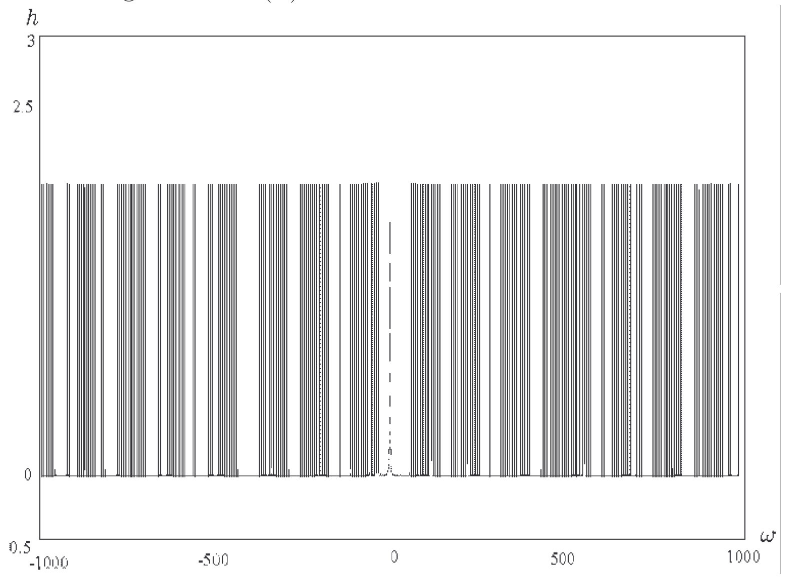
$$\mathbb{S} = \left\{(\omega, h) \in \mathbb{R}^2 | G(\omega) = h\right\}. \tag{122}$$

The Bode diagram has picks every 2π. The thickness of the picks decreases exponentially when ω increases. The four following pictures show that an elementary interval algorithm makes it possible to draw the Bode diagram in a reliable way whereas MATLAB has some difficulties, even for a very high precision.

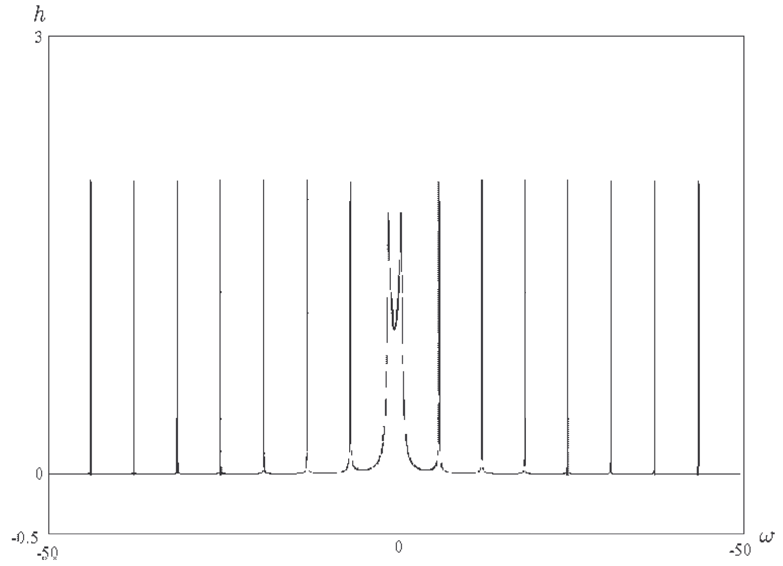

Bode diagram $h = G(\omega)$ with MATLAB with $\Delta\omega = 0.1 rad.s^{-1}$



Bode diagram $h = G(\omega)$ with MATLAB with $\Delta\omega = 0.001 rad.s^{-1}$



Bode diagram obtained by PROJ2D for $\omega \in [-1000, 1000]$

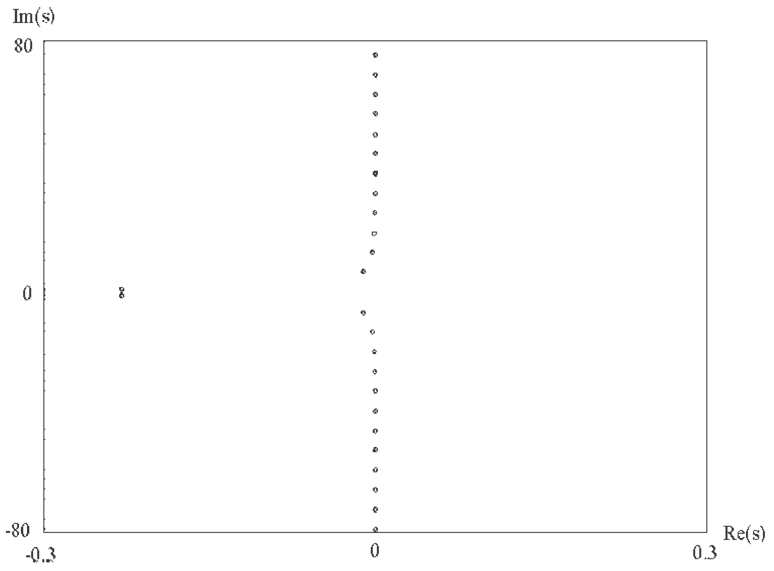


Bode diagram obtained by PROJ2D for $\omega \in [-50, 50]$

Moreover, PROJ2D has been able to bracket that the H_∞ norm of the system around 2. The set of all feasible roots (or *root locus*) of the system is given by

$$\begin{aligned} \mathbb{S} &= \left\{ s \in \mathbb{C} | \, (s+1)\left(s(1-e^{-s})+1\right) = 0 \right\} \\ &= \left\{ x + jy \in \mathbb{C} | \begin{pmatrix} x - (x\cos y + y \sin y)e^{-x} + 1 \\ x + (x \sin y - y \cos y)e^{-x} \end{pmatrix} = \mathbf{0} \right\}. \end{aligned} \tag{123}$$

The following picture representing $\mathbb{S}$ has been obtained by PROJ2D. It illustrates the asymptotic direction of the poles of the system.



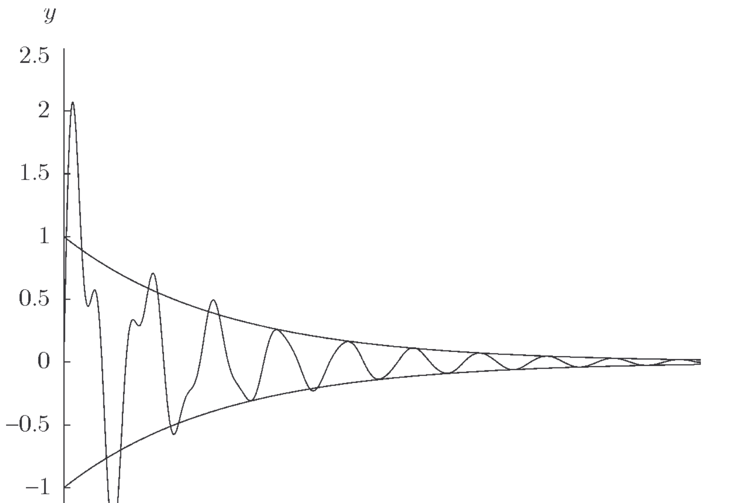
Stability degree
The location of the roots of $P(s)$ provides more information that just the stability or instability of the system Σ. When Σ is stable, the real parts of the roots of its characteristic polynomial are related to the speed with which $\mathbf{x}$ converges to $\mathbf{0}$ in the absence of input, and complex roots are responsible for oscillations in the process, if any. For instance, if the roots of $P(s)$ are

$$(-0.2-3j, -0.2+3j, -0.5-7j, -0.5+7j, -1-3j, -1+3j), \tag{124}$$

then, in the absence of input, all the components $y_i(t)$ of the output vector $\mathbf{y}(t)$ of the system Σ have the form

$$\begin{aligned} y_i(t) &= \alpha_1 \sin(3t+\phi_1)\exp(-0.2t) + \alpha_2 \sin(7t+\phi_2)\exp(-0.5t) \\ &\quad + \alpha_3 \sin(3t+\phi_3)\exp(-t), \end{aligned} \tag{125}$$

where the coefficients $\alpha_1, \alpha_2, \alpha_3, \phi_1, \phi_2, \phi_3$ depend on the initial conditions. The function $y_i(t)$ is depicted below for $\alpha_1 = \alpha_2 = \alpha_3 = 1$ and $\phi_1 = \phi_2 = \phi_3 = 0$. The asymptotic enveloping exponential curves correspond to $\pm\exp(-0.2t)$. The location of the roots is directly related to the temporal behavior of the outputs of the system.



A system Σ is said to be δ-*stable* if all its roots are on the left of the vertical line $\mathrm{Re}(s) = -\delta$. For our example Σ is

0.1-stable but 0.3-unstable. To check the δ-stability of $P(s)$, it suffices to check the stability of the polynomial

$$\begin{aligned} Q_\delta(s) &= P(s-\delta) \\ &= (s-\delta)^n + a_{n-1}(s-\delta)^{n-1} + \ldots + a_1(s-\delta) + a_0 \\ &= s^n + b_{n-1}(\delta)s^{n-1} + \ldots + b_1(\delta)s + b_0(\delta), \end{aligned} \tag{126}$$

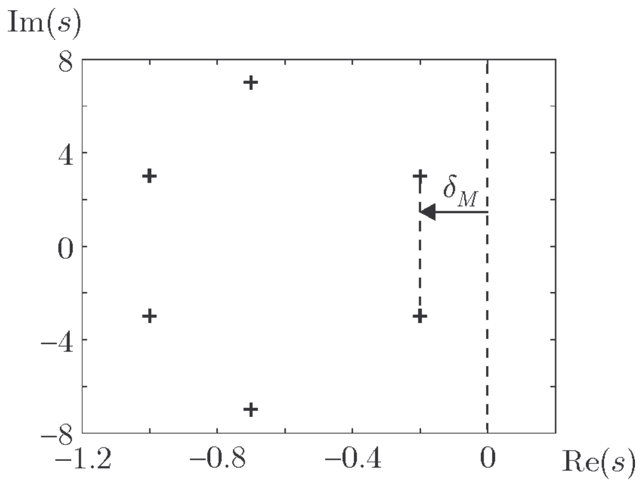
i.e., $P(s)$ is δ-stable if and only if $P(s-\delta)$ is stable. The stability of $Q_\delta(s)$ can be tested using the Routh criterion. Define the δ-Routh vector $\mathbf{r}(\delta)$ as the Routh vector associated with $Q_\delta(s)$. Then

$$P(s) \text{ is } \delta\text{-stable} \quad \Leftrightarrow \quad Q_\delta(s) \text{ is stable} \quad \Leftrightarrow \quad \mathbf{r}(\delta) > \mathbf{0}. \tag{127}$$

The *stability degree* of $P(s)$ is

$$\delta_M \overset{\text{def}}{=} \max_{\mathbf{r}(\delta)\geqslant\mathbf{0}} \delta. \tag{128}$$

The larger the stability degree of Σ is, the faster the state of Σ will return to equilibrium in the absence of inputs. The figure illustrates the significance of the stability degree δ_M which is found here to be equal to 0.2. This means that asymptotically the dominant component of $y_i(t)$ is $\alpha_1 \sin(3t+\phi_1)\exp(-0.2t)$.



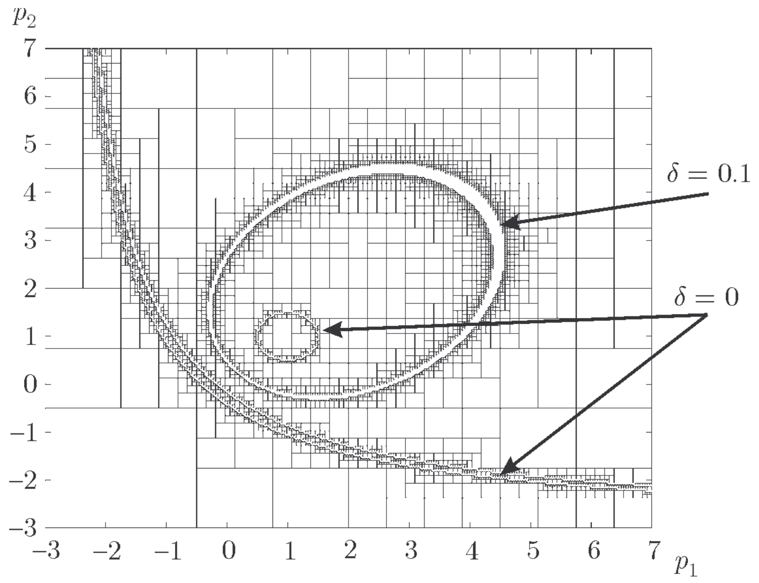
Robust stability degree
When Σ depends on a vector $\mathbf{p}$ of parameters, so does its δ-Routh vector $\mathbf{r}(\mathbf{p},\delta)$. The *stability degree* of $\Sigma(\mathbf{p})$ is defined as

$$\delta_M(\mathbf{p}) \overset{\text{def}}{=} \max_{\mathbf{r}(\mathbf{p},\delta)\geqslant\mathbf{0}} \delta. \tag{129}$$

Example 1: If the characteristic polynomial for $\Sigma(\mathbf{p})$ is given by

$$P(s,\mathbf{p}) = s^3 + (p_1+p_2+2)s^2 + (p_1+p_2+2)s + 2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2, \tag{130}$$

the level sets associated with $\delta \in \{0; 0.1\}$ are as follows.



■

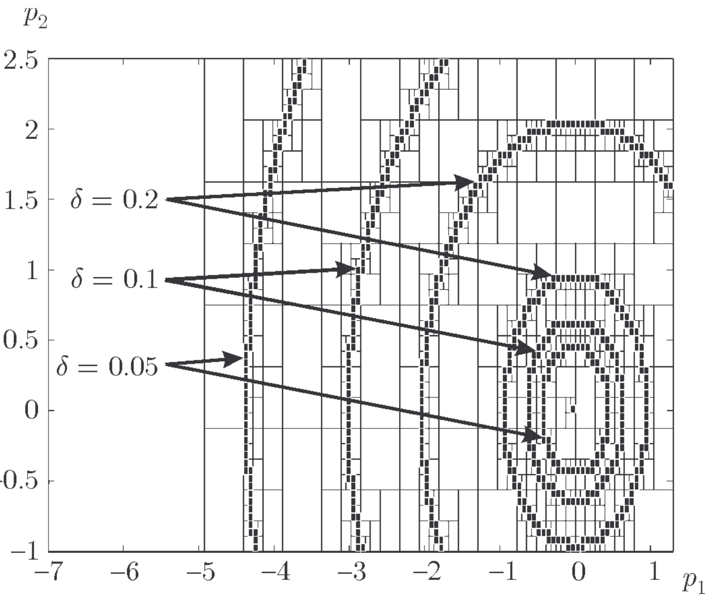
Example 2: Consider now the uncertain system $\Sigma(\mathbf{p})$ given by

$$\dot{\mathbf{x}} = \begin{pmatrix} 0 & 1 & -p_1 \\ 1 & 0 & -p_2 \\ p_1 & p_2 & 1 \end{pmatrix} \mathbf{x},\ p_1 \in [-7, 1.3],\ p_2 \in [-1, 2.5]. \tag{131}$$

Its characteristic polynomial is

$$P(s, \mathbf{p}) = s^3 + s^2 + (p_1^2 + p_2^2 + 1)s + 1. \tag{132}$$

The only unstable point is $\mathbf{p} = \mathbf{0}$. Some level sets are given on the figure below.



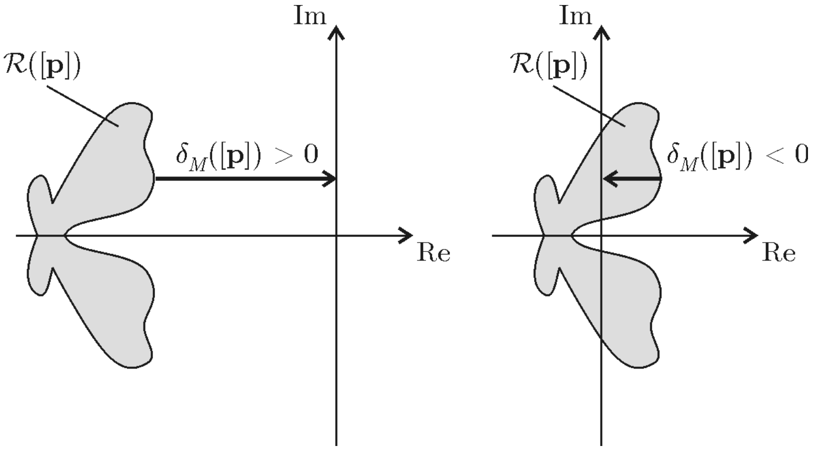
■

The *robust stability degree* $\delta_M([\mathbf{p}])$ of $\Sigma(\mathbf{p})$ for $\mathbf{p}$ in $[\mathbf{p}]$ is the stability degree in the worst case:

$$\delta_M([\mathbf{p}]) = \min_{\mathbf{p}\in[\mathbf{p}]} \max_{\mathbf{r}(\mathbf{p},\delta)\geqslant \mathbf{0}} \delta. \tag{133}$$

If $\delta_M([\mathbf{p}]) > 0$, then all the roots of $\Sigma(\mathbf{p})$ are in $\mathbb{C}^-$ and the uncertain system $\Sigma([\mathbf{p}])$ is robustly stable (*i.e.*, it is stable for any $\mathbf{p}$ in $[\mathbf{p}]$), as illustrated by the subfigure on the left., where $\mathcal{R}([\mathbf{p}])$ is the set of all feasible roots for

$\Sigma(\mathbf{p}), \mathbf{p} \in [\mathbf{p}]$. If $\delta_M([\mathbf{p}]) \leqslant 0$, then there exists some $\mathbf{p}$ in $[\mathbf{p}]$ such that $\Sigma(\mathbf{p})$ is unstable, as illustrated by the right subfigure.


Im
$\mathcal{R}([\mathbf{p}])$
$\delta_M([\mathbf{p}]) > 0$
Re
Im
$\mathcal{R}([\mathbf{p}])$
$\delta_M([\mathbf{p}]) < 0$
Re

Example 3. Consider the uncertain system

$$\dot{\mathbf{x}} = \begin{pmatrix} \frac{p_2}{1+p_2} & 2 \\ \frac{p_2}{1+p_1} & \frac{p_1}{1+p_2^2} \end{pmatrix} .\mathbf{x}. \tag{134}$$

For $[\mathbf{p}] = [1,2] \times [0, 0.5]$, the interval algorithm PertMin finds that the robust stability degree satisfies

$$-2.01590 \leqslant \delta_M([\mathbf{p}]) \leqslant -2.01451.$$

The system is thus not robustly stable. ■

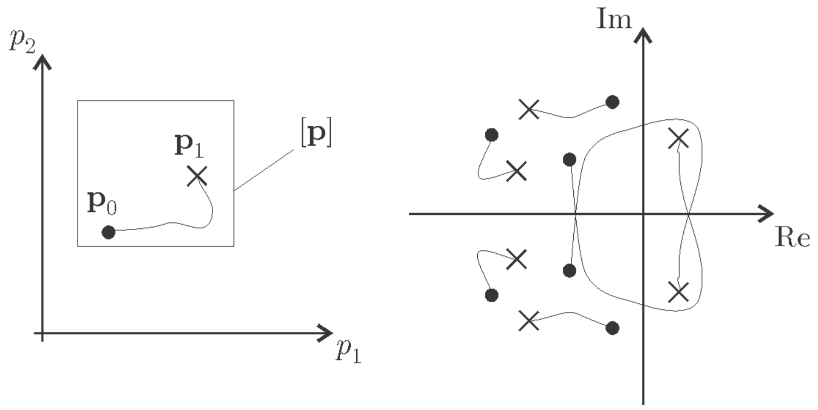
Value-set approach
The concept of *value set* allows a simple geometrical interpretation of robust stability in the complex plane. Recall that the uncertain system $\Sigma(\mathbf{p})$ with characteristic polynomial

$$P(s,\mathbf{p}) = a_n(\mathbf{p})s^n + a_{n-1}(\mathbf{p})s^{n-1} + \ldots + a_1(\mathbf{p})s + a_0(\mathbf{p}) \tag{135}$$

is robustly stable in $[\mathbf{p}]$ if the CSP

$$\mathcal{H} : (P(s,\mathbf{p}) = 0, \mathbf{p} \in [\mathbf{p}], \mathrm{Re}(s) \geqslant 0) \tag{136}$$

has no solution. Since s is complex, the dimension of the search space is dim $\mathbf{p} + 2$. It is often possible to reduce this dimension to dim $\mathbf{p} + 1$ by taking advantage of the continuity of the roots of $P(s,\mathbf{p})$ with respect to its coefficients. Assume that $[\mathbf{p}]$ contains a stable vector $\mathbf{p}_0$ and an unstable vector $\mathbf{p}_1$. The roots associated with $\mathbf{p}_0$ all have negative real parts and at least one of the roots associated with $\mathbf{p}_1$ has a positive real part. Assume also that the coefficients of $P(s,\mathbf{p})$ are continuous in $\mathbf{p}$. When $\mathbf{p}$ moves from $\mathbf{p}_0$ to $\mathbf{p}_1$ in $[\mathbf{p}]$, at least one of the roots crosses the imaginary axis, *i.e.*, there exist $\mathbf{p}$ in $[\mathbf{p}]$ and ω in $\mathbb{R}$ such that $P(j\omega,\mathbf{p}) = 0$.



Theorem: If the coefficients of $P(s, \mathbf{p})$ are continuous functions of $\mathbf{p}$, if the leading coefficient $a_n(\mathbf{p})$ never vanishes over $[\mathbf{p}]$ and if there exists $\mathbf{p}_0$ in $[\mathbf{p}]$ such that $P(s, \mathbf{p}_0)$ is stable, then $P(s, [\mathbf{p}])$ is robustly stable if and only if the CSP

$$\mathcal{H} : (P(j\omega, \mathbf{p}) = 0, \mathbf{p} \in [\mathbf{p}],\ \omega \in \mathbb{R}) \tag{137}$$

has no solution. ■

The stability of $P(s, \mathbf{p}_0)$ can be checked with the Routh criterion. The domain for ω can be restricted to $\omega \geqslant 0$, because if $(\mathbf{p}, \omega)$ is a solution of (137), so is $(\mathbf{p}, -\omega)$. The dimension of search space is now dim $\mathbf{p} + 1$ instead of dim $\mathbf{p} + 2$. To apply interval methods to prove that (137) has no solution, it is important to bound the domain for ω. As the module of $P(j\omega, \mathbf{p})$ tends to infinity with ω, there exists an angular frequency ω_c (*cutoff frequency*) beyond which $P(j\omega, \mathbf{p})$ will never be equal to zero for any $\mathbf{p}$ in $[\mathbf{p}]$. The following theorem provides a mean for computing an upper bound for ω_c.

Theorem: All the roots of $P(s) = a_n s^n + \cdots + a_1 s + a_0$, with $a_n \neq 0$, are inside the disk with centre zero and radius

$$\beta = 1 + \frac{\max\{|a_0|, |a_1|, \ldots, |a_{n-1}|\}}{|a_n|}. \tag{138}$$

■

When $P(s)$ depends on $\mathbf{p}$, β becomes a function of $\mathbf{p}$. An inclusion function for $\beta(\mathbf{p})$ is thus

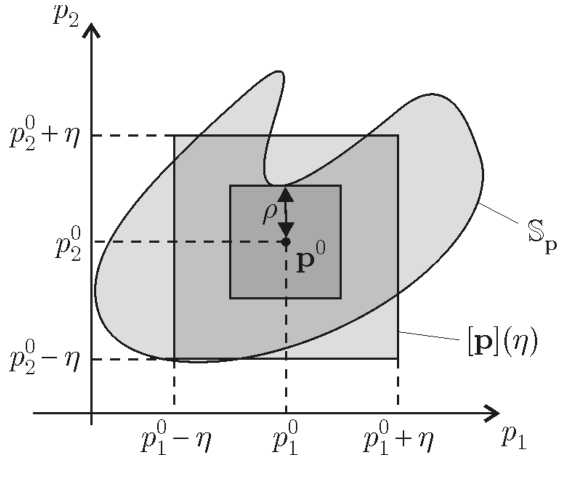
$$[\beta]([\mathbf{p}]) = 1 + \frac{\max(|[a_0]([\mathbf{p}])|, \ldots, |[a_{n-1}]([\mathbf{p}])|)}{|[a_n]([\mathbf{p}])|}. \tag{139}$$

If $\overline{\beta}$ is the upper bound of the interval $[\beta]([\mathbf{p}])$, the uncertain polynomial $P([\mathbf{p}], s)$ has all its roots inside the disk with centre 0 and radius $\overline{\beta}$. $\overline{\beta}$ is thus an upper bound of the cutoff frequency ω_c. The domain for ω is now taken as $[0, \overline{\beta}]$, which is finite.

Stability radius
The *stability radius* of $\Sigma(\mathbf{p})$ at $\mathbf{p}^0$ is

$$\begin{aligned} \rho &\overset{\text{def}}{=} \sup\{\eta \geqslant 0 \mid \Sigma(\mathbf{p}) \text{ is stable for all } \mathbf{p} \in [\mathbf{p}](\eta)\}, \\ &= \min\{\eta \geqslant 0 \mid \Sigma(\mathbf{p}) \text{ is unstable for one } \mathbf{p} \in [\mathbf{p}](\eta)\}, \end{aligned} \tag{140}$$

where $[\mathbf{p}](\eta)$ is the box with centre $\mathbf{p}^0$ such that $w([\underline{p}_j, \overline{p}_j]) = 2\eta w_j$ for some prespecified positive number w_j. The quantity η is thus the radius of the hypercube $[\mathbf{p}](\eta)$ in the L_∞ norm weighted by the w_js. The figure illustrates this notion for dim $\mathbf{p} = 2$ and $w_1 = w_2 = 1$.



Now, since

$$\mathbf{p} \in [\mathbf{p}](\eta) \Leftrightarrow \forall j \in \{1, \ldots, n\}, (p_j^0 - \eta w_j \leqslant p_j \leqslant p_j^0 + \eta w_j) \tag{141}$$

and since

$$\begin{aligned}\Sigma(\mathbf{p}) \text{ is unstable} &\Leftrightarrow \exists i \text{ such that } r_i(\mathbf{p}) \leqslant 0 &(142)\\ &\Leftrightarrow (r_1(\mathbf{p}) \leqslant 0) \vee \cdots \vee (r_n(\mathbf{p}) \leqslant 0)\,, \\ &\Leftrightarrow \max(r_1(\mathbf{p})), \ldots, r_n(\mathbf{p})) \leqslant 0, &(143)\end{aligned}$$

the stability radius can also be defined as

$$\left\{\begin{array}{l}\rho = \min\limits_{\eta \geqslant 0} \eta, \\ \text{subject to } \left\{\begin{array}{l}\max(r_1(\mathbf{p})), \ldots, r_n(\mathbf{p})) \leqslant 0 \\ \wedge \left(\forall j \in \{1, \ldots, n\}, \left\{\begin{array}{l} p_j^0 - \eta w_j - p_j \leqslant 0 \\ -p_j^0 - \eta w_j + p_j \leqslant 0\end{array}\right.\right)\end{array}\right.\end{array}\right. \tag{144}$$

Example 1: Consider the polynomial

$$\begin{aligned}P(s, \mathbf{p}) &= s^3 + (p_1 + p_2 + 2)s^2 + (p_1 + p_2 + 2)s \\ &\quad +2p_1p_2 + 6p_1 + 6p_2 + 2 + \sigma^2.\end{aligned}$$

When p_1 and p_2 are positive, this polynomial is stable for all parameter vectors outside the disk with centre $\mathbf{p}^0 = (1, 1)^{\mathrm{T}}$ and radius σ. An interval optimization algorithm is now used to compute the stability radius for different values of σ. The nominal value for $\mathbf{p}$ is taken as $\mathbf{p}^0 = (1.4, 0.85)^{\mathrm{T}}$, and the weights are $w_1 = 1.1$ and $w_2 = 0.85$. The box $[\mathbf{p}](\eta)$ is thus defined by

$$\begin{array}{rcccl}1.4 - 1.1\eta & \leqslant & p_1 & \leqslant & 1.4 + 1.1\eta, \\ 0.85 - 0.85\eta & \leqslant & p_2 & \leqslant & 0.85 + 0.85\eta.\end{array} \tag{145}$$

The results, obtained on a PENTIUM 90 for different values of σ are given in the next table.

σ	10^{-1}	10^{-3}	10^{-5}	10^{-7}
Number of iterations	66	113	55	63
Computing time (s)	0.44	0.55	0.44	0.49
Number of solution boxes	1	5	1	2
Stability radius	0.2727	0.3627	0.3636	0.3636

■

Example 2: Consider the polynomial

$$P(s, \mathbf{p}) = s^3 + a_2(\mathbf{p})s^2 + a_1(\mathbf{p})s + a_0(\mathbf{p}), \tag{146}$$

with

$$\begin{aligned}a_0(\mathbf{p}) &= \sin(p_2)e^{p_2} + p_1p_2 - 1, \\ a_1(\mathbf{p}) &= 2p_1 + 0.2p_1e^{p_2}, \\ a_2(\mathbf{p}) &= p_1 + p_2 + 4.\end{aligned} \tag{147}$$

The coefficient function $\mathbf{a}(\mathbf{p})$ is neither linear nor polynomial. A computation of the stability radius, using an interval optimization algorithm, finds it to be approximately equal to 2.025 at $\mathbf{p}^0 = (1.5, 1.5)^{\mathrm{T}}$. ■

Nonlinear control of a sailboat

(Luc Jaulin, Thursday, 11h15-11h45).

Projection of an equality.
We shall first provide a methodology to compute an inner and an outer approximation of the set

$$\mathbb{S} \overset{\text{def}}{=} \{\mathbf{p} \in \mathbf{P} \mid \exists \mathbf{q} \in \mathbf{Q}, f(\mathbf{p},\mathbf{q}) = 0\}. \tag{148}$$

where $\mathbf{P}$ and $\mathbf{Q}$ are boxes and f is a continuous function. This methodology can only be used if only one equation is involved and cannot not be generalized to the case where $\dim f \geq 2$.
Since $\mathbf{Q}$ is a connected set and f is continuous, we have

$$\mathbb{S} = \{\mathbf{p} \in \mathbf{P} \mid (\exists \mathbf{q}_1 \in \mathbf{Q}, f(\mathbf{p},\mathbf{q}_1) \leq 0) \text{ and } (\exists \mathbf{q}_2 \in \mathbf{Q}, f(\mathbf{p},\mathbf{q}_2) \geq 0)\}. \tag{149}$$

i.e.,

$$\mathbb{S} = \{\mathbf{p} \in \mathbf{P} \mid (\exists \mathbf{q}_1 \in \mathbf{Q}, f(\mathbf{p},\mathbf{q}_1) \leq 0)\} \cap \{\mathbf{p} \in \mathbf{P} \mid (\exists \mathbf{q}_2 \in \mathbf{Q}, f(\mathbf{p},\mathbf{q}_2) \geq 0)\}. \tag{150}$$

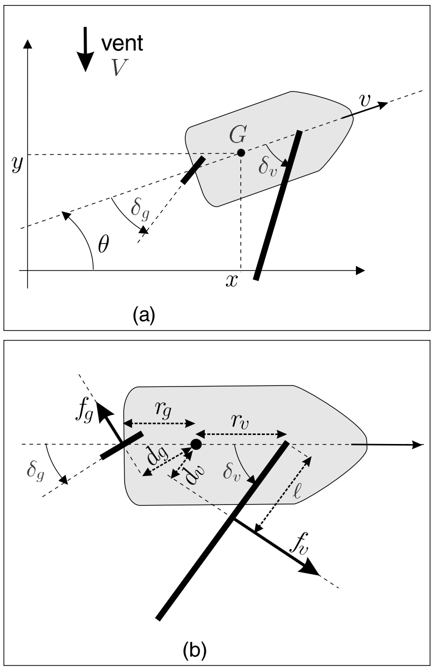
or equivalently

$$\mathbb{S} = \left\{\mathbf{p} \in \mathbf{P} \mid \exists (\mathbf{q}_1,\mathbf{q}_2) \in \mathbf{Q}^2, f(\mathbf{p},\mathbf{q}_1) \leq 0 \text{ and } f(\mathbf{p},\mathbf{q}_2) \geq 0\right\}. \tag{151}$$

Polar speed diagram of a sailboat (Collaboration with M. Dao, M. Lhommeau, P. Herrero, J. Vehi and M. Sainz)
The sailboat represented on the figure is described by the following state equations

$$\left\{\begin{array}{lll} \dot{x} & = & v\cos\theta, \\ \dot{y} & = & v\sin\theta - \beta V, \\ \dot{\theta} & = & \omega, \\ \dot{\delta}_s & = & u_1, \\ \dot{\delta}_r & = & u_2, \\ \dot{v} & = & \frac{f_s \sin\delta_s - f_r \sin\delta_r - \alpha_f v}{m}, \\ \dot{\omega} & = & \frac{(\ell - r_s\cos\delta_s) f_s - r_r \cos\delta_r f_r - \alpha_\theta \omega}{J}, \\ f_s & = & \alpha_s \left(V\cos(\theta+\delta_s) - v\sin\delta_s\right), \\ f_r & = & \alpha_r v \sin\delta_r. \end{array}\right. \tag{152}$$

The state vector $\mathbf{x} = (x,y,\theta,\delta_s,\delta_r,v,\omega)^{\mathrm{T}} \in \mathbb{R}^7$.The inputs u_1 and u_2 of the system are the derivatives of the angles δ_s and δ_r.



The *polar speed diagram* of the sailboat is the set $\mathbb{S}$ of all pairs (θ, v) that can be reached by the boat. During a cruising behavior of the boat, the speed of the boat, its course, its angular velocity, ... are constant, i.e.,

$$\dot{\theta} = 0, \dot{\delta}_s = 0, \dot{\delta}_r = 0, \dot{v} = 0, \dot{\omega} = 0. \tag{153}$$

From Equation (152), we get

$$\left\{ \begin{array}{lll} 0 & = & \frac{f_s \sin \delta_s - f_r \sin \delta_r - \alpha_f v}{m}, \\ 0 & = & \frac{(\ell - r_s \cos \delta_s) f_s - r_r \cos \delta_r f_r}{J}, \\ f_s & = & \alpha_s \left(V \cos\left(\theta + \delta_s\right) - v \sin \delta_s\right), \\ f_r & = & \alpha_r v \sin \delta_r. \end{array} \right. \tag{154}$$

An elimination of f_s, f_r and δ_r yields

$$\begin{array}{ll} \left((\alpha_r + 2\alpha_f) v - 2\alpha_s V \cos(\theta + \delta_s) \sin \delta_s + 2\alpha_s v \sin^2 \delta_s \right)^2 & \\ + \left(\frac{2\alpha_s}{r_r} (\ell - r_s \cos \delta_s) (V \cos(\theta + \delta_s) - v \sin \delta_s) \right)^2 - \alpha_r^2 v^2 & = \quad 0 \end{array} \tag{155}$$

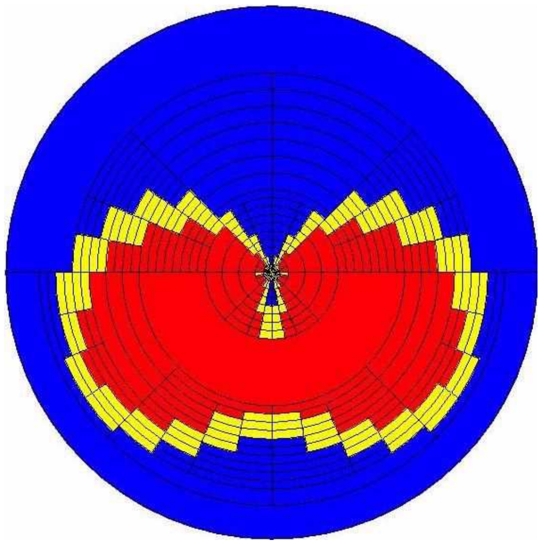
The polar speed diagram can thus be written as

$$\mathbb{S} = \left\{ (\theta, v) | \exists \delta_s \in [-\frac{\pi}{2}, \frac{\pi}{2}] \mid f(\theta, v, \delta_s) = 0 \right\}. \tag{156}$$

An inner and an outer approximation of the polar speed diagram is given below in the case where the parameters are given by

$$\begin{array}{lll} L & = & 1, \alpha_f = 60, \alpha_\theta = 500, \alpha_s = 500, \alpha_r = 300, \beta = 0.05, \\ r_s & = & 1, r_r = 2, V = 10, m = 1000, J = 2000. \end{array} \tag{157}$$

This diagram has been obtained with a modal version of the above methodology. Since the theory of modal interval analysis has not been introduced in this lesson, a nonmodal algorithm has been given here.



Feedback linearization

To illustrate how feedback control can be applied to our sailboat, we shall consider a normalized version of the state equations where all parameters are equal to 1, i.e.,

$$\left\{ \begin{array}{lll} \dot{\theta} & = & \omega, \\ \dot{\delta}_s & = & u_1, \\ \dot{\delta}_r & = & u_2, \\ \dot{v} & = & f_s \sin \delta_s - f_r \sin \delta_r - v, \\ \dot{\omega} & = & (1 - \cos \delta_s) f_s - \cos \delta_r f_r - \omega, \\ f_s & = & \cos(\theta + \delta_s) - v \sin \delta_s, \\ f_r & = & v \sin \delta_r. \end{array} \right. \tag{158}$$

Denote by $\mathcal{F}(\mathbf{x},\mathbf{u})$ the set of all variables that can be written as algebraic functions of $\mathbf{x}$ and $\mathbf{u}$. We have

$$\left(\dot{\theta},\dot{\delta}_s,\dot{\delta}_r,\dot{v},\dot{\omega},f_s,f_r\right)\in\mathcal{F}(\mathbf{x},\mathbf{u}), \tag{159}$$

but $\ddot{\delta}_s=\dot{u}_1\notin\mathcal{F}(\mathbf{x},\mathbf{u})$. Since

$$\left\{\begin{array}{rcl}\ddot{\theta} & = & \dot{\omega},\\ \ddot{v} & = & \dot{f}_s\sin\delta_s+f_s u_1\cos\delta_s-\dot{f}_r\sin\delta_r-f_r u_2\cos\delta_r-\dot{v}\\ \ddot{\omega} & = & u_1\sin\delta_s f_s+(1-\cos\delta_s)\,\dot{f}_s+u_2\sin\delta_r f_r-\cos\delta_r\dot{f}_r-\dot{\omega}.\\ \dddot{\theta} & = & \ddot{\omega}\\ \dot{f}_s & = & -(\omega+u_1)\sin(\theta+\delta_s)-\dot{v}\sin\delta_s-vu_1\cos\delta_s\\ \dot{f}_r & = & \dot{v}\sin\delta_r+vu_2\cos\delta_r.\end{array}\right. \tag{160}$$

we have

$$\left(\ddot{\theta},\ddot{v},\ddot{\omega},\dot{f}_s,\dot{f}_r,\dddot{\theta}\right)\in\mathcal{F}(\mathbf{x},\mathbf{u}). \tag{161}$$

Denote by $\mathcal{F}_u(\mathbf{x},\mathbf{u})$ the set of all variables of $\mathcal{F}(\mathbf{x},\mathbf{u})$ which are not constant with respect to $\mathbf{u}$. We have

$$\left(u_1,u_2,\dot{\delta}_s,\dot{\delta}_r,\ddot{v},\ddot{\omega},\dot{f}_s,\dot{f}_r,\dddot{\theta}\right)\in\mathcal{F}_u(\mathbf{x},\mathbf{u}). \tag{162}$$

Take two state variables and store them into $\mathbf{y}$. For instance

$$\mathbf{y}=(\delta_s,\theta)^{\mathrm{T}}. \tag{163}$$

Note that $\left(\dot{\delta}_s,\dddot{\theta}\right)\in\mathcal{F}_u(\mathbf{x},\mathbf{u})$:

$$\begin{array}{rcll}\left(\begin{array}{c}\dot{y}_1\\ \dddot{y}_2\end{array}\right) & = & \left(\begin{array}{c}\dot{\delta}_s\\ \dddot{\theta}\end{array}\right) & (164)\\ & = & \left(\begin{array}{cc}1 & 0\\ f_s\sin\delta_s+(\cos\delta_s-1)(v\cos\delta_s+\sin(\theta+\delta_s)) & f_r\sin\delta_r-v\cos^2\delta_r\end{array}\right)\mathbf{u} & (165)\\ & & +\left(\begin{array}{cc}0 & 0\\ 1-\cos\delta_s & -\cos\delta_r\end{array}\right)\left(\begin{array}{c}-\omega\sin(\theta+\delta_s)-\dot{v}\sin\delta_s\\ \dot{v}\sin\delta_r\end{array}\right)+\left(\begin{array}{c}0\\ -\dot{\omega}\end{array}\right) & (166)\\ & = & \mathbf{A}(\mathbf{x})\mathbf{u}+\mathbf{b}(\mathbf{x}).\end{array}$$

If we take

$$\mathbf{u}=\mathbf{A}^{-1}(\mathbf{x})\left(\mathbf{v}-\mathbf{b}(\mathbf{x})\right), \tag{167}$$

The closed loop system becomes linear:

$$\left\{\begin{array}{rcl}\dot{\delta}_s & = & v_1,\\ \dddot{\theta} & = & v_2.\end{array}\right. \tag{168}$$

This linear and decoupled system should now be stabilized.

Linear control.
Denote by $\mathbf{w}=(w_1,w_2)=\left(\hat{\delta}_s,\hat{\theta}\right)$ the wanted values for $\mathbf{y}=(\delta_s,\theta)$. Classical PD^n controllers are given by :

$$\left\{\begin{array}{rcl}v_1 & = & \alpha_{\mathrm{P}}(w_1-\delta_s),\\ v_2 & = & \beta_{\mathrm{P}}(w_2-\theta)+\beta_{\mathrm{D}}\left(\dot{w}_2-\dot{\theta}\right)+\beta_{\mathrm{D}^2}\left(\ddot{w}_2-\ddot{\theta}\right).\end{array}\right. \tag{169}$$

If w_2 is assumed to be constant, the closed loop system can be written:

$$\left\{\begin{array}{rcl}\dot{\delta}_s & = & \alpha_{\mathrm{P}}(w_1-\delta_s)\\ \dddot{\theta} & = & \beta_{\mathrm{P}}(w_2-\theta)-\beta_{\mathrm{D}}\dot{\theta}-\beta_{\mathrm{D}^2}\ddot{\theta}.\end{array}\right. \tag{170}$$

The transfer matrix is

$$\mathbf{M}(s)=\left(\begin{array}{cc}\frac{\alpha_{\mathrm{P}}}{s+\alpha_{\mathrm{P}}} & 0\\ 0 & \frac{\beta_{\mathrm{P}}}{s^3+\beta_{\mathrm{D}^2}s^2+\beta_{\mathrm{D}}s+\beta_{\mathrm{P}}}\end{array}\right). \tag{171}$$

The characteristic polynomial is

$$P(s) = (s+\alpha_{\mathrm{P}})\left(s^3+\beta_{\mathrm{D}^2}s^2+\beta_{\mathrm{D}}s+\beta_{\mathrm{P}}\right), \tag{172}$$

If we want all roots to be equal to -1, we should solve:

$$\begin{aligned}(s+\alpha_{\mathrm{P}})\left(s^3+\beta_{\mathrm{D}^2}s^2+\beta_{\mathrm{D}}s+\beta_{\mathrm{P}}\right) &= (s+1)^4 \\ &= (s+1)\left(s^3+3s^2+3s+1\right)\end{aligned} \tag{173}$$

Thus the linear controller to be taken is

$$\begin{cases} v_1 = w_1-\delta_s, \\ v_2 = (w_2-\theta)-3\dot{\theta}-3\ddot{\theta}. \end{cases} \tag{174}$$

Control with wanted inputs (Collaboration with M. Dao, P. Herrero, J. Vehi, M. Sainz)
The system to be controlled is described by :

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u}). \tag{175}$$

As seen before, for some rather large conditions on the system and for specific output vector $\mathbf{y}=\mathbf{g}(\mathbf{x})$, feedback linearization methods make it possible to find a controller of the form

$$\mathbf{u} = \mathcal{R}_u(\mathbf{x},\bar{\mathbf{y}}), \tag{176}$$

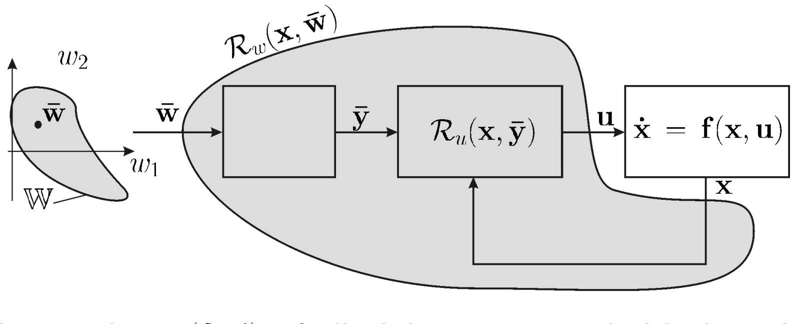
such that the output $\mathbf{y}$ converges to $\bar{\mathbf{y}}$. Now, in many cases, the user wants to choose its own output vector $\mathbf{w}=\mathbf{h}(\mathbf{x})$ and not to have an output vector $\mathbf{y}$ imposed by the structure of the system to make the feedback linearization method working. The problem of interest is to find a controller

$$\mathbf{u} = \mathcal{R}_w(\mathbf{x},\bar{\mathbf{w}}) \tag{177}$$

such that the $\mathbf{w}$ converges to the *wanted vector* $\bar{\mathbf{w}}$. Define the set of all feasible wanted vectors by

$$\mathbb{W} = \left\{\mathbf{w}\in\mathbb{R}^m \mid \exists\mathbf{x}\in\mathbb{R}^n, \exists\mathbf{u}\in\mathbb{R}^m, \mathbf{f}(\mathbf{x},\mathbf{u})=\mathbf{0}, \mathbf{w}=\mathbf{h}(\mathbf{x})\right\}. \tag{178}$$

Since $\mathbb{W}$ is defined by $n+m$ equations for $n+2m$ variables (where $n=\dim\mathbf{x}$ and $m=\dim\mathbf{u}=\dim\mathbf{w}$), except for atypical situations, the set $\mathbb{W}$ has a nonempty volume.
We first have to characterize an inner and an outer approximation of $\mathbb{W}$. Then, the user will be allowed to choose any point $\bar{\mathbf{w}}$ inside $\mathbb{W}$. From $\bar{\mathbf{w}}$, we will then compute some corresponding $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$ such that $\mathbf{f}(\bar{\mathbf{x}},\bar{\mathbf{u}})=\mathbf{0}, \bar{\mathbf{w}}=\mathbf{h}(\bar{\mathbf{x}})$. Note the solution pair $(\bar{\mathbf{x}},\bar{\mathbf{u}})$ may be not unique. From $\bar{\mathbf{x}}$, we shall compute $\bar{\mathbf{y}}=\mathbf{g}(\bar{\mathbf{x}})$. Then the controller $\mathcal{R}_u(\mathbf{x},\bar{\mathbf{y}})$ will compute $\mathbf{u}$ such that $\mathbf{y}$ converges to $\bar{\mathbf{y}}$. As a consequence, $\mathbf{x}$ will tend to $\bar{\mathbf{x}}$ and $\mathbf{w}$ to $\bar{\mathbf{w}}$.


w_2
$\bar{\mathbf{w}}$
w_1
$\mathbb{W}$
$\mathcal{R}_w(\mathbf{x},\bar{\mathbf{w}})$
$\bar{\mathbf{w}}$
$\bar{\mathbf{y}}$
$\mathcal{R}_u(\mathbf{x},\bar{\mathbf{y}})$
$\mathbf{u}$
$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x},\mathbf{u})$
$\mathbf{x}$

For the normalized sailboat, with $\mathbf{y}=(\boldsymbol{\delta}_s,\theta)$ a feedback linearization method leads to the controller $\mathcal{R}_u(\mathbf{x},\bar{\mathbf{y}}) = \mathcal{R}_u(\mathbf{x},\hat{\delta}_s,\hat{\theta})$ given by

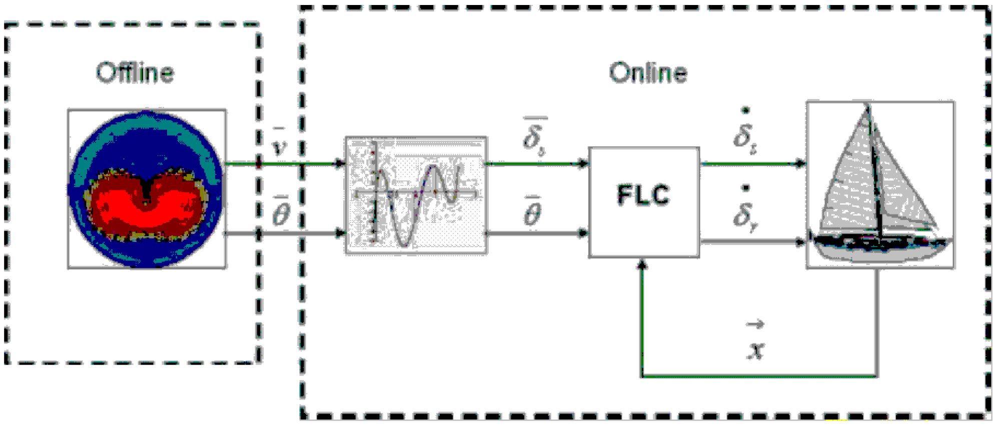
$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x})\left(\begin{pmatrix}\hat{\delta}_s-\delta_s \\ \hat{\theta}-\theta-3\omega-3\dot{\omega}\end{pmatrix}-\mathbf{b}(\mathbf{x})\right) \tag{179}$$

where

$$\begin{aligned}\mathbf{A}(\mathbf{x}) &= \begin{pmatrix}1 & 0 \\ f_s\sin\delta_s+(v\cos\delta_s+\sin(\theta+\delta_s))(\cos\delta_s-1) & f_r\sin\delta_r-v\cos^2\delta_r\end{pmatrix} \\ \mathbf{b}(\mathbf{x}) &= \begin{pmatrix}0 \\ (v\sin\delta_s+\omega\sin(\theta+\delta_s))(\cos\delta_s-1)-v\cos\delta_r\sin\delta_r\end{pmatrix}-\begin{pmatrix}0 \\ \dot{\omega}\end{pmatrix}\end{aligned} \tag{180}$$

The principle of the control is as follows.

1. Choose $\bar{\mathbf{w}} = (\bar{v}, \bar{\theta})$ in the polar speed diagram.
2. Using an interval algorithm compute $\bar{\mathbf{y}} = (\bar{\delta}_s, \bar{\theta})$ such that $\exists \mathbf{x}$, $\mathbf{f}(\mathbf{x}, \bar{\mathbf{y}}) = \mathbf{0}$ and $\mathbf{w} = \mathbf{h}(\mathbf{x})$. We know from (178) that there exists at least one solution.
3. Apply the control (179) based on feedback linearization.



Control of a wheeled stair-climbing robot

(Luc Jaulin, Thursday, 11h45-12h15).
(Collaboration with students and colleagues from ENSIETA)

Control problem

We shall now consider the class of constrained dynamic systems which can be described by

$$\begin{array}{ll} \text{(i)} & \dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \\ \text{(ii)} & (\mathbf{x}(t), \mathbf{v}(t)) \in \mathbb{V}, \end{array} \tag{181}$$

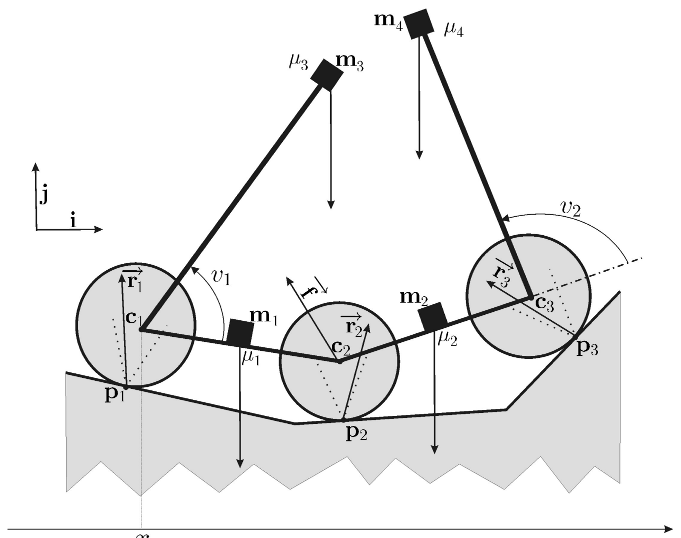
where $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the *evolution input vector*, $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is the *state vector*, $\mathbf{f} : \mathbb{R}^{n_x+n_u} \to \mathbb{R}^{n_x}$ is the *evolution function*, $\mathbf{v}(t) \in \mathbb{R}^{n_v}$ is the *viable input vector* and $\mathbb{V}$ is the *viable set*. We have assumed here that the evolution inputs and viable inputs are independent, *i.e.*, $\mathbf{u}$ acts only in (i) whereas $\mathbf{v}$ affects only (ii). This is not always the case. However, for many robotics applications, it is possible to conceive the robot and to insert the actuators at the right place in order to get a decoupling between the evolution and the viable inputs. If the value for $\mathbf{v}(t)$, chosen at time t, is such that $(\mathbf{x}, \mathbf{v}) \in \mathbb{V}$ then the system is said to be *alive*. Otherwise, it is *dead*. In practice, the condition $(\mathbf{x}, \mathbf{v}) \notin \mathbb{V}$ translates into events such as "an element of the system brakes down" or "the robot is sliding", etc. In such situations the differential equation $\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}, \mathbf{u})$ is not satisfied anymore and the behavior of the system becomes unpredictable. Consequently, all should be done to avoid such a situation.
From a control viewpoint, two problems should be treated. The first one consists in finding the evolution input vector $\mathbf{u}$ in order to get a satisfactory behavior of the system. This problem can be solved using classical techniques even when $\mathbf{f}$ is nonlinear. The second problem to be considered is to find a viable input vector $\mathbf{v}$ such that the set-membership relation (ii) is satisfied for the given state vector $\mathbf{x}$. This research should be done for each t in a very short time.

Wheel stair-climbing robot

The figure below presents a two-dimensional robot made by three driving wheels and two pendulums. The center of the ith wheel will be denoted by $\mathbf{c}_i$. The back wheel and the middle wheel are linked by a platform with weight μ_1. The front wheel and the middle wheel are linked by a platform with weight μ_2. At the top of each pendulum are located two weights μ_3 and μ_4. We shall denote by $\mathbf{m}_1$ and $\mathbf{m}_2$ the centers of the two platforms and by $\mathbf{m}_3$ and $\mathbf{m}_4$ the vertices of the pendulums. The angles v_1 and v_2 between the pendulum and the corresponding platform can be tuned as desired and will be used to prevent the robot from any sliding. The abscissa of the center of the back wheel is denoted by x. The robot is equipped with a strong electric motor which is linked to each of the three wheels. It can be assumed that when the robot is not sliding, x satisfies the following differential equation

$$\dot{x} = u, \tag{182}$$

where the evolution input u can be chosen arbitrarily. Here, we shall choose u small enough to be allowed to assume that the robot has a quasi-static motion. We shall assume that all wheels keep a contact with the ground. Thus, our robot has three degrees of freedom: x, v_1 and v_2. The contact point $\mathbf{p}_i$ between the ith wheel and the ground will be assumed to be unique.



Two dimensional three-wheeled climbing robot to be moved in an uneven urban environment

Assume that the robot is immobile and denote by $\overrightarrow{\mathbf{r}}_i$ the reaction force generated by the ground onto the ith wheel. The Coulomb cone $\mathcal{C}_i$ associated with the ith wheel and represented with dotted lines on the figure is the cone with vertex $\mathbf{p}_i$ whose symmetric axis in orthogonal to the ground and with an half aperture angle $\phi = 0.54$ (corresponding to a tire/concrete contact). If there exist $\overrightarrow{\mathbf{r}}_1^0 \in \mathcal{C}_1, \overrightarrow{\mathbf{r}}_2^0 \in \mathcal{C}_2$, $\overrightarrow{\mathbf{r}}_3^0 \in \mathcal{C}_3$ such that the fundamental static conditions are fulfilled then the robot is known to be at a steady position and will not slide. Note that the actual values for $\overrightarrow{\mathbf{r}}_1, \overrightarrow{\mathbf{r}}_2, \overrightarrow{\mathbf{r}}_3$ cannot be computed without an in-depth knowledge of the internal tensions inside the robot (which is unrealistic here) and may be different from $\overrightarrow{\mathbf{r}}_1^0$, $\overrightarrow{\mathbf{r}}_2^0$, $\overrightarrow{\mathbf{r}}_3^0$.

Viability constraints
Let us now write the fundamental static conditions of the robot in order to translate the non-sliding condition into the form $(x, v_1, v_2) \in \mathbb{V}$, **(**since a quasi-static motion is assumed, the dependency with respect to $\dot{x}$ does not exist**).** When the robot does not move, we have

$$\left\{\begin{array}{lcl} -\overrightarrow{\mathbf{p}_1\mathbf{m}_1} \wedge \mu_1\mathbf{j} + \overrightarrow{\mathbf{p}_1\mathbf{c}_2} \wedge \overrightarrow{\mathbf{f}} - \overrightarrow{\mathbf{p}_1\mathbf{m}_3} \wedge \mu_3\mathbf{j} & = & 0 \\ -\overrightarrow{\mathbf{p}_2\mathbf{m}_2} \wedge \mu_2\mathbf{j} - \overrightarrow{\mathbf{p}_2\mathbf{c}_2} \wedge \overrightarrow{\mathbf{f}} + \overrightarrow{\mathbf{p}_2\mathbf{p}_3} \wedge \overrightarrow{\mathbf{r}}_3 - \overrightarrow{\mathbf{p}_2\mathbf{m}_4} \wedge \mu_4\mathbf{j} & = & 0 \\ \overrightarrow{\mathbf{r}}_1 - (\mu_1 + \mu_3)\mathbf{j} + \overrightarrow{\mathbf{f}} & = & 0 \\ \overrightarrow{\mathbf{r}}_2 - \overrightarrow{\mathbf{f}} - (\mu_2 + \mu_4)\mathbf{j} + \overrightarrow{\mathbf{r}}_3 & = & 0, \end{array}\right. \tag{183}$$

where $\tilde{\mathbf{f}}$ denotes the reaction force generated by the second platform onto the first platform. A scalar decomposition yields

$$\left\{\begin{array}{lcl} -\mu_1 (m_{1x} - p_{1x}) + (c_{2x} - p_{1x}) f_y - (c_{2y} - p_{1y}) f_x - \mu_3 (m_{3x} - p_{1x}) & = & 0 \\ -\mu_2 (m_{2x} - p_{2x}) - (c_{2x} - p_{2x}) f_y + (c_{2y} - p_{2y}) f_x & & \\ +(p_{3x} - p_{2x}) r_{3y} - (p_{3y} - p_{2y}) r_{3x} - \mu_4 (m_{4x} - p_{2x}) & = & 0 \\ r_{1x} + f_x & = & 0 \\ r_{1y} - \mu_1 - \mu_3 + f_y & = & 0 \\ r_{2x} - f_x + r_{3x} & = & 0 \\ r_{2y} - f_y - \mu_2 - \mu_4 + r_{3y} & = & 0. \end{array}\right. \tag{184}$$

This system can be written into a matrix form as

$$\mathbf{A}_1(x).\mathbf{y} = \mathbf{b}_1(x), \tag{185}$$

where

$$\begin{array}{rcl} \mathbf{A}_1(x) & = & \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & p_{1y} - c_{2y} & c_{2x} - p_{1x} & -\mu_3 & 0 \\ 0 & 0 & 0 & 0 & p_{2y} - p_{3y} & p_{3x} - p_{2x} & c_{2y} - p_{2y} & p_{2x} - c_{2x} & 0 & -\mu_4 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & -1 & 0 & 0 \end{pmatrix} \\ \mathbf{b}_1(x) & = & \begin{pmatrix} \mu_1 (m_{1x} - p_{1x}) - \mu_3 p_{1x} \\ \mu_2 (m_{2x} - p_{2x}) - \mu_4 p_{2x} \\ 0 \\ \mu_1 + \mu_3 \\ 0 \\ \mu_2 + \mu_4 \end{pmatrix} \\ \mathbf{y} & = & (r_{1x}, r_{1y}, r_{2x}, r_{2y}, r_{3x}, r_{3y}, f_x, f_y, m_{3x}, m_{4x})^{\mathrm{T}}. \end{array}$$

The system of equations (185) has ten unknowns for six equations. If $\mathbf{v} = (v_1, v_2)$ is known, then it remains two more unknowns than equations. This is due to the fact that our robot is a statically undetermined structure with a second order hyperstatic equilibrium.
None of the wheels will slide if all $\overrightarrow{\mathbf{r}}_i$ belong to their corresponding Coulomb cones. Thus, we should have

$$\det(\overrightarrow{\mathbf{r}}_i, \mathbf{u}_i^-) \leq 0 \text{ and } \det(\mathbf{u}_i^+, \overrightarrow{\mathbf{r}}_i) \leq 0, \tag{186}$$

where $\mathbf{u}_i^-$ and $\mathbf{u}_i^+$ denote the two vectors supporting the ith Coulomb cone $\mathcal{C}_i$. These two inequalities can be rewritten into a matrix form as

$$\begin{pmatrix} u_{iy}^- & -u_{ix}^- \\ -u_{iy}^+ & u_{ix}^+ \end{pmatrix} \overrightarrow{\mathbf{r}}_i \leq \mathbf{0}. \tag{187}$$

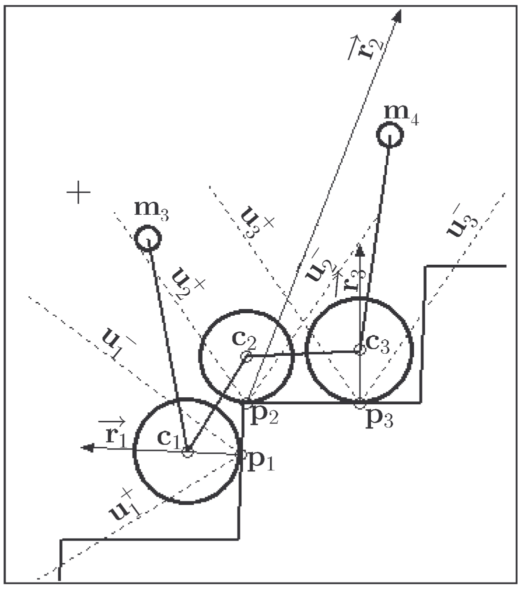
Thus, we should have,

$$\mathbf{A}_2(x).\mathbf{y} \leq \mathbf{0}, \tag{188}$$

where

$$\mathbf{A}_2(x) = \begin{pmatrix} u_{1y}^- & -u_{1x}^- & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -u_{1y}^+ & u_{1x}^+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & u_{2y}^- & -u_{2x}^- & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -u_{2y}^+ & u_{2x}^+ & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & u_{3y}^- & -u_{3x}^- & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -u_{3y}^+ & u_{3x}^+ & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{189}$$

The figure below represents a computed situation where the robot is immobile and not sliding. Vectors $\mathbf{u}_i^-, \mathbf{u}_i^+$ delimiting the Coulomb cones are represented with dotted lines. Since all ground reaction forces $\overrightarrow{\mathbf{r}}_i$ are inside their corresponding cones, the robot cannot slide.



First, we have

$$\begin{array}{rcl} m_{3x} & = & c_{1x} + \ell_1 \cos\left(v_1 + \arctan\left(\frac{c_{2y}-c_{1y}}{c_{2x}-c_{1x}}\right)\right), \\ m_{4x} & = & c_{3x} + \ell_2 \cos\left(v_2 + \arctan\left(\frac{c_{3y}-c_{2y}}{c_{3x}-c_{2x}}\right)\right). \end{array} \tag{190}$$

Moreover the variables m_{3x} and m_{4x} should be such that the pendulums be at feasible locations : the pendulums should not intersect the ground or the robot itself. This condition can be translated into

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} .\mathbf{y} \in \begin{pmatrix} [m_{3x}^{\min}, m_{3x}^{\max}] \\ [m_{4x}^{\min}, m_{4x}^{\max}] \end{pmatrix}, \tag{191}$$

where the bounds $m_{3x}^{\min}, m_{3x}^{\max}, m_{4x}^{\min}, m_{4x}^{\max}$ can be obtained with sensors. Our robot can be described by

$$\begin{array}{lrcl} \text{(i)} & \dot{x} & = & u \\ \text{(ii)} & (x, v_1, v_2) & \in & \mathbb{V} \end{array} \tag{192}$$

where the condition $(x, v_1, v_2) \in \mathbb{V}$ is equivalent to

$$\exists\, (r_{1x}, r_{1y}, r_{2x}, r_{2y}, r_{3x}, r_{3y}, f_x, f_y, m_{3x}, m_{4x}) \text{ such that (185), (188), (191) and (190) are satisfied}$$

The control problem we wish to solve now is the following: find the angles v_1 and v_2 such that the fundamental principle of static, the non-sliding conditions and the feasible conditions for the pendulums are satisfied. To achieve this task, we shall act as follows:

1. Estimate the vectors $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3, \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{m}_1, \mathbf{m}_2$ and the two intervals $[m_{3x}^{\min}, m_{3x}^{\max}], [m_{4x}^{\min}, m_{4x}^{\max}]$.

2. Solve the following linear program

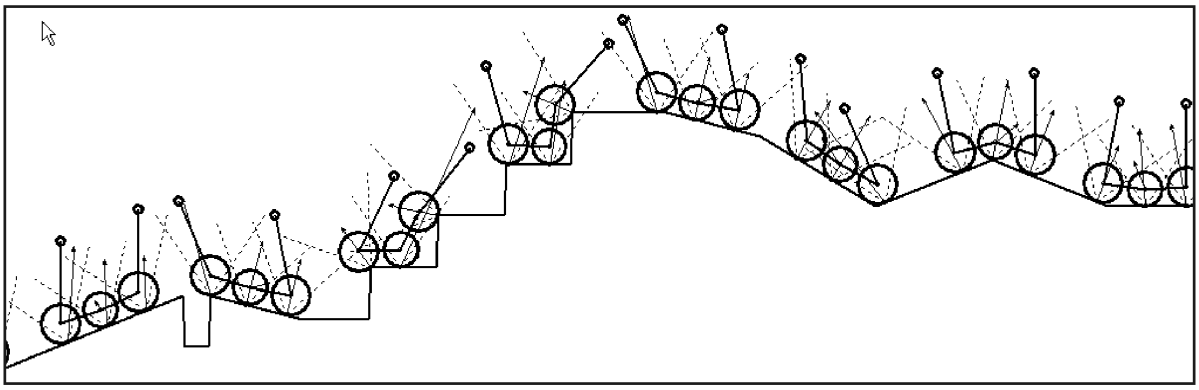
$$\begin{array}{rl} & \min e, \\ s.t. & \text{(i) } \mathbf{A}_1(x).\mathbf{y} = \mathbf{b}_1(x) \\ & \text{(ii) } \mathbf{A}_2(x).\mathbf{y} \le e.\mathbf{1}, \\ & \text{(iii) } \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} .\mathbf{y} \in \begin{pmatrix} [m_{3x}^{\min}, m_{3x}^{\max}] \\ [m_{4x}^{\min}, m_{4x}^{\max}] \end{pmatrix} \end{array} \tag{193}$$

where $\mathbf{1} = (1\ 1\ 1\ 1\ 1\ 1)^{\mathrm{T}}$. Since the rank of $\mathbf{A}_1$ is full, the linear program always has a solution. If the minimum $\hat{e}$ is negative, the non-sliding condition (188) can be satisfied and the robot will not slide. If $\hat{e}$ is positive, the robot will slide.

3. From $\mathbf{y}$, compute the viable inputs v_1 and v_2 such that (190) is satisfied.

4. Compute independently the control $\mathbf{u}$ using a classical control method.

Let us move the robot slowly in a urban irregular ground. As illustrated by the figure below the pendulums react in order to avoid any sliding from the robot. The simulation of the control has been performed with SCILAB and the source code is available at `www.ensieta.fr/e3i2/Jaulin/etas.sce`. For each sample, the computation of the control $\mathbf{v}$ required less than 0.01 sec on a Pentium 1.5 GHz. The shape of the ground has been chosen so as to be as chaotic as possible while still preserving the possibility to move the pendulums in order to avoid any sliding. For the simulation, the parameters that have been chosen for the robot are $\phi = 0.54$ for the angle friction coefficient, $\rho_1 = 85$mm, $\rho_2 = 75$mm, $\rho_3 = 85$mm for the radius of the wheels and $\ell_1 = \ell_2 = 350$mm for the lengths of the pendulums. The weights of the platforms and the pendulum are given by $\mu_1 = \mu_2 = 70$N, $\mu_3 = \mu_4 = 20$N. The height and the width of the stairs have been chosen equal to 220mm and 280mm, respectively. On the figure, dotted lines represent the Coulomb cones. Since each cone contains the corresponding reaction force created by the ground onto the wheel, the robot never slides.



Different configurations for the robot; Since the Coulomb conditions are always satisfied, the robot never slides

The figure below represents the robot built by the robotics team of the ENSIETA engineering school that has won the 2005 robot cup organized by ETAS, France. The robot can be seen as a three-dimensional version of the robot treated above. It has been proven to be very competitive on irregular grounds but failed to cross over some compulsory obstacles (such as stairs). For the 2006 competition, we are planning to implement the mass transfer system on the robot. Carrying out the mass transfer as well as the localization of the robot with respect to the ground remain to be developed.

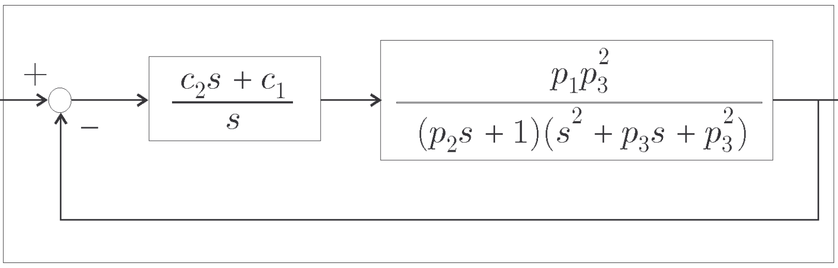
Robot on which the mass transfer system will be implemented

Robust control

(Luc Jaulin and Michel Kieffer, Thursday, 14h00-15h15).

Robust control of a linear system
The robust control problem to be considered is find a controller that stabilizes a parametric model as on the figure. We assume that $\mathbf{p} \in [\mathbf{p}] = [0.9, 1.1]^{\times 3}$ and that $\mathbf{c}$ has to be chosen arbitrarily in a box $[\mathbf{c}] = [0,1]^2$. Denote by $\Sigma(\mathbf{c},\mathbf{p})$ the closed-loop system. It is easy to find, for instance via the Routh criterion, a function $r\colon \mathbb{R}^2 \times \mathbb{R}^3 \to \mathbb{R}$ such that

$$\Sigma(\mathbf{p},\mathbf{c}) \text{ is stable } \Leftrightarrow r(\mathbf{c},\mathbf{p}) > 0. \tag{194}$$



Finding all robust controllers amounts to characterizing the set

$$\mathbb{T}_c = \{\mathbf{c} \in [\mathbf{c}] \mid \forall \mathbf{p} \in [\mathbf{p}], r(\mathbf{c},\mathbf{p}) > 0\}. \tag{195}$$

The complementary set of $\mathbb{T}_c$ in $[\mathbf{c}]$:

$$\neg\mathbb{T}_c = \{\mathbf{c} \in [\mathbf{c}] \mid \exists \mathbf{p} \in [\mathbf{p}], r(\mathbf{c},\mathbf{p}) \leq 0\} \tag{196}$$

is thus the projection of a set defined by a nonlinear inequality. The transfer function of $\Sigma(\mathbf{p},\mathbf{c})$ is

$$H(s) = \frac{(c_2 s + c_1)\, p_1 p_3^2}{p_2 s^4 + (p_2 p_3 + 1)\, s^3 + (p_2 p_3^2 + p_3)\, s^2 + (p_3^2 + c_2 p_1 p_3^2)\, s + c_1 p_1 p_3^2}. \tag{197}$$

The first column of the corresponding Routh table is

$$\begin{pmatrix} p_2 \\ p_2 p_3 + 1 \\ p_2 p_3^2 + p_3 - \frac{p_2\left(p_3^2 + c_2 p_1 p_3^2\right)}{p_2 p_3 + 1} \\ p_3^2 + c_2 p_1 p_3^2 - \frac{(p_2 p_3+1)^2\left(c_1 p_1 p_3^2\right)}{\left(p_2 p_3^2 + p_3\right)(p_2 p_3 + 1) - p_2\left(p_3^2 + c_2 p_1 p_3^2\right)} \\ c_1 p_1 p_3^2 \end{pmatrix}. \tag{198}$$

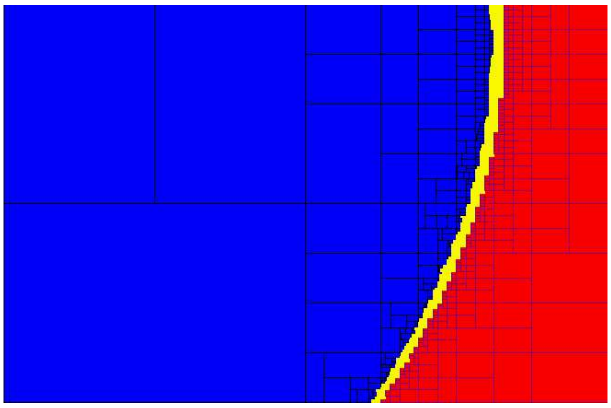
Since $p_2 > 0$, the closed-loop system $\Sigma(\mathbf{p},\mathbf{c})$ is asymptotically stable if and only if

$$r(\mathbf{c},\mathbf{p}) \overset{\text{def}}{=} \min \begin{pmatrix} p_2 p_3 + 1 \\ p_2 p_3^2 + p_3 - \frac{p_2\left(p_3^2 + c_2 p_1 p_3^2\right)}{p_2 p_3 + 1} \\ p_3^2 + c_2 p_1 p_3^2 - \frac{(p_2 p_3+1)^2\left(c_1 p_1 p_3^2\right)}{\left(p_2 p_3^2 + p_3\right)(p_2 p_3 + 1) - p_2\left(p_3^2 + c_2 p_1 p_3^2\right)} \\ c_1 p_1 p_3^2 \end{pmatrix} > 0. \tag{199}$$

The complementary set

$$\neg\mathbb{T}_c \overset{\text{def}}{=} \left\{\mathbf{c} \in [0,1]^2 \mid \exists \mathbf{p} \in [0.9; 1.1]^{\times 3}, r(\mathbf{c},\mathbf{p}) \leq 0\right\} \tag{200}$$

of the set of robust controller $\mathbb{T}_c$ computed by Proj2D is depicted on the figure below.


Approximation of the set $\neg\mathbb{T}_c$

Optimal robust control
The problem to be now studied is the computation of the set $\mathbb{S}_c$ of the vectors $\mathbf{c}$ that maximize the stability degree in the worst case. This set satisfies

$$\mathbb{S}_c = \arg \max_{\mathbf{c}\in[\mathbf{c}]} \min_{\mathbf{p}\in[\mathbf{p}]} \max_{\mathbf{r}(\mathbf{p},\mathbf{c},\delta)\geqslant \mathbf{0}} \delta. \tag{201}$$

The rightmost *max* corresponds to the definition of the stability degree. The *min* ensures the worst-case conditions. The leftmost *max* corresponds to the optimality requirement.

Example: For the closed-loop system $\Sigma(\mathbf{p},\mathbf{c})$ considered immediately before, PERTMIN gives the results of the table below. In this table, $\#\overline{\mathbb{S}}_c$ is the number of boxes in the subpaving $\overline{\mathbb{S}}_c$ containing all the values of $\mathbf{c}$ corresponding to optimal robust controllers, $[\overline{\mathbb{S}}_c]$ is the interval hull of $\overline{\mathbb{S}}_c$, and $[\delta_M^c]$ is an interval guaranteed to contain the associated optimal robust stability degree. The times are indicated for a PENTIUM 90.

$[\mathbf{p}]$	Time (s)	$\#\overline{\mathbb{S}}_c$	$[\overline{\mathbb{S}}_c]$	$[\delta_M^c]$
$(1,1,1)^{\mathrm{T}}$	5.5	66	$[0.257, 0.273] \times [0.305, 0.354]$	$[0.300, 0.326]$
$[0.99, 1.01]^{\times 3}$	85	69	$[0.239, 0.274] \times [0.264, 0.382]$	$[0.288, 0.299]$
$[0.95, 1.05]^{\times 3}$	339	37	$[0.207, 0.277] \times [0.179, 0.437]$	$[0.261, 0.282]$
$[0.9, 1.1]^{\times 3}$	345	17	$[0.207, 0.254] \times [0.191, 0.367]$	$[0.230, 0.246]$

■

Control of a time-delay system (Collaboration with M. Dao, M. Di Loreto, J.F. Lafay and J.J. Loiseau)
Consider the unstable system

$$\dot{x}(t) = x(t) + u(t-1). \tag{202}$$

Let us try to stabilize this system using the following control law:

$$u(t) = \alpha x(t) + \beta x(t-1). \tag{203}$$

We have

$$\dot{x}(t) = x(t) + \alpha x(t-1) + \beta x(t-2). \tag{204}$$

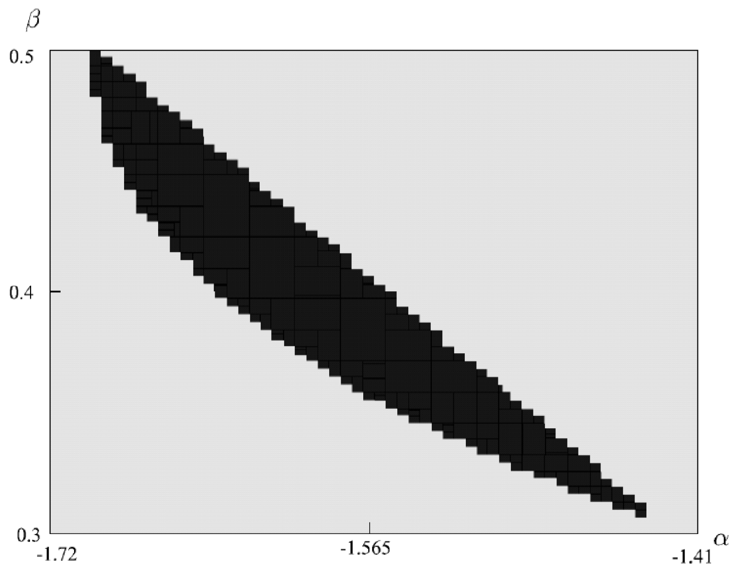
The characteristic equation is

$$s - 1 - \alpha e^{-s} - \beta e^{-2s} = 0. \tag{205}$$

The stability domain is

$$\mathbb{S} = \left\{(\alpha,\beta) | \nexists s \in \mathbb{C}^+, s - 1 - \alpha e^{-s} - \beta e^{-2s} = 0\right\}. \tag{206}$$

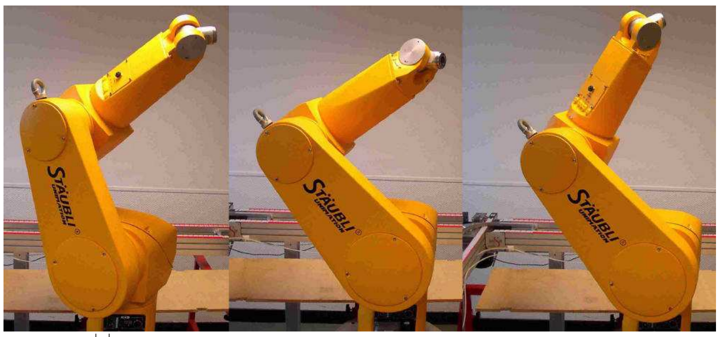
Proving that a box is inside $\mathbb{S}$ can be performed using interval constraint propagation methods. The black boxes in the figure below have been proved to belong to $\mathbb{S}$ using PROJ2D.

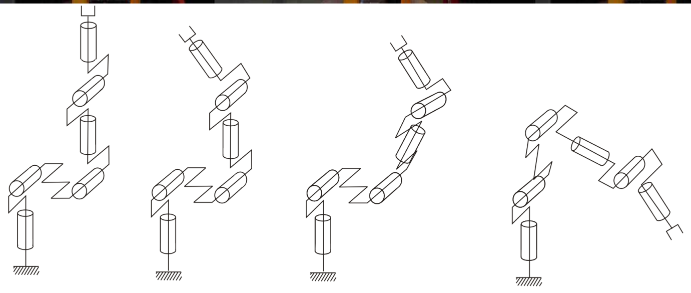

β
0.5
0.4
0.3
-1.72
-1.565
-1.41
α

Robot Calibration

(Luc Jaulin and Nacim Ramdani, Friday, 14h00-14h30).

Presentation of the robot (Staubli RX90). (Collaboration with X. Baguenard, P. Lucidarme and W. Khalil) Consider the following robot





Its configuration vector is

$$\mathbf{q} = (q_1, ..., q_6) \in \mathbb{R}^6, \tag{207}$$

where the q_i's are the angles of the articulations of the robot in radian. The tool, is represented by 3 points A_1, A_2, A_3 forming the vector

$$\mathbf{x} = \left(a_x^1, a_y^1, a_z^1, a_x^2, a_y^2, a_z^2, a_x^3, a_y^3, a_z^3\right)^t \in \mathbb{R}^9. \tag{208}$$

The parameter vector of the robot is given by

$$\mathbf{p} = (r_0, \alpha_1, d_1, r_1, \dots, \alpha_5, d_5, r_5, \alpha_6, d_6, \theta_0, \theta_1^o, \dots, \theta_5^o, b_x^1, b_y^1, b_z^1, b_x^2, b_y^2, b_z^2, b_x^3, b_y^3, b_z^3) \tag{209}$$

contains all geometric constants which characterize the robot. For instance, d_i correspond to the length of the ith arm. The *direct geometric model* is given by

$$\mathbf{x} = \mathbf{f}(\mathbf{p}, \mathbf{q}). \tag{210}$$

where $\mathbf{f}$ is given by the following algorithm.

```
Algorithm f
inputs : $\mathbf{q} = (q_1, ..., q_6)^{\mathrm{T}}$,
         $\mathbf{p} = \left(\alpha_j, d_j, r_j, \theta_0, \theta_j^o, b_x^i, b_y^i, b_z^i, ...\right)^{\mathrm{T}}$.
outputs : $\mathbf{x} = \left(a_x^1, a_y^1, a_z^1, a_x^2, a_y^2, a_z^2, a_x^3, a_y^3, a_z^3\right)^{\mathrm{T}}$.
1  $\mathbf{M} := \begin{pmatrix} 1 & 0 & 0 & d_6 \\ 0 & \cos\alpha_6 & -\sin\alpha_6 & 0 \\ 0 & \sin\alpha_6 & \cos\alpha_6 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} \cos q_6 & -\sin q_6 & 0 & 0 \\ \sin q_6 & \cos q_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$;
2  for $j := 5$ to 1,
3      $\theta := \theta_j^o + q_j$;
4      $\mathbf{M} := \begin{pmatrix} 1 & 0 & 0 & d_j \\ 0 & \cos\alpha_j & -\sin\alpha_j & 0 \\ 0 & \sin\alpha_j & \cos\alpha_j & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & r_j \\ 0 & 0 & 0 & 1 \end{pmatrix} .\mathbf{M}$;
5  endfor
6  $\mathbf{M} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \begin{pmatrix} \cos\theta_0 & -\sin\theta_0 & 0 & 0 \\ \sin\theta_0 & \cos\theta_0 & 0 & 0 \\ 0 & 0 & 1 & r_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} .\mathbf{M}$;
7  for $i := 1$ to 3, $\mathbf{b}^i = \begin{pmatrix} b_x^i & b_y^i & b_z^i & 1 \end{pmatrix}^{\mathrm{T}}$;
8  $\mathbf{x} := \begin{pmatrix} \mathbf{M} & 0 & 0 \\ 0 & \mathbf{M} & 0 \\ 0 & 0 & \mathbf{M} \end{pmatrix} \begin{pmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \\ \mathbf{b}^3 \end{pmatrix}$.
```

Principle of the calibration

1. Choose r different configuration vectors $\mathbf{q}(1), ..., \mathbf{q}(r)$.
2. For these r configurations, measure the coordinates of the three points characterizing the tool: $\mathbf{x}(1), ..., \mathbf{x}(r)$ (recall that the $\mathbf{x}(i)$'s all belong to $\mathbb{R}^9$),
3. Generate the constraints
$$\mathbf{x}(k) = \mathbf{f}(\mathbf{p}, \mathbf{q}(k)),\ k = \{1, ..., r\}. \tag{211}$$
4. Contract the prior domains for all variables $\mathbf{p}, \mathbf{q}(k), \mathbf{x}(k)$ $k = \{1, ..., r\}$.

DAG (Directed Acyclic Graph)
Our problem is a CSP with a huge number of variables and constraints. It is important to rewrite our constraints in an optimal way in order to make the propagation more efficient. Consider for instance the constraints

$$\begin{array}{rcl} y_1 & = & \cos(i_1 + i_2).\sin(i_1 + i_2), \\ y_2 & = & i_3.\sin^2(i_1 + i_2). \end{array} \tag{212}$$

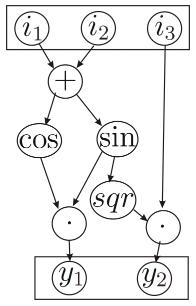
They can be decomposed into primitive constraints as follows

$$\begin{array}{ll} a_1 = i_1 + i_2, & \\ a_2 = \cos(a_1), & a_5 = i_1 + i_2, \\ a_3 = i_1 + i_2, & a_6 = \sin(a_5), \\ a_4 = \sin(a_3), & a_7 = a_6^2, \\ y_1 = a_2.a_4. & y_2 = i_3.a_7. \end{array} \tag{213}$$

A more efficient representation is

$$\begin{array}{ll} a_1 = i_1 + i_2, & \\ a_2 = \cos(a_1), & \\ a_4 = \sin(a_1), & a_7 = a_4^2, \\ y_1 = a_2.a_4. & y_2 = i_3.a_7. \end{array} \tag{214}$$

which is associated to the following DAG



An automatic way to get an optimal decomposition use the notions of DAG (Directed Acyclic Graph) and hatching table.

Generation of simulated measurements
The nominal values chosen for the geometric parameters of the robot STAUBLI RX90 are

j	α_j	d_j	θ_j^o	r_j
0	-	-	$\frac{\pi}{2}$	0.5
1	0.1	0	0	0
2	$-\frac{\pi}{2}$	0	0	0
3	0	0.5	0	0
4	$\frac{\pi}{2}$	0	0	0.5
5	$-\frac{\pi}{2}$	0	0	0
6	$\frac{\pi}{2}$	0	-	-

The three points of the tool with coordinates b_x^i, b_y^i and b_y^i in the terminal arm frame are chosen as

i	b_x^i	b_y^i	b_z^i
1	0.1	0.2	0.1
2	0.1	0.1	0.2
3	0.2	0.1	0.1

For 50 random configuration vector $\mathbf{q}(k)$, we computes $\mathbf{x}(k) = \mathbf{f}(\mathbf{p}, \mathbf{q}(k)) + \mathbf{e}(k)$ where $\mathbf{e}(k)$ is a random bounded noise. To each variable, we associated an interval which contains the true values for the variables.

Results
It took about $10s$ to the solver IntervalPeeler (`www.istia.univ-angers.fr/~baguenar/`) to perform the propagation. The file containing all constraints takes about 837Ko. The results obtained are given below.

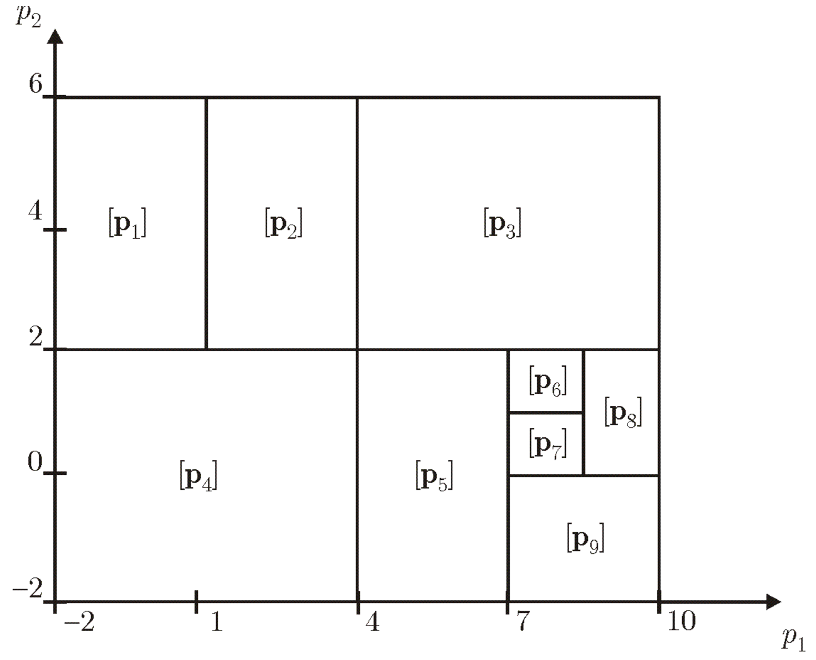
	initial domains	contracted domains
r_0	$[0.4, 0.6]$	$[0.494046, 0.50101]$
d_1	$[0, 0.1]$	$[0, 0.000558009]$
r_1	$[0, 0.1]$	$[0, 0.00693694]$
d_3	$[0.49, 0.51]$	$[0.498385, 0.501133]$
r_4	$[0.49, 0.51]$	$[0.499216, 0.50114]$
b_x^1	$[0, 0.2]$	$[0.0996052, 0.100629]$
b_y^1	$[0.1, 0.3]$	$[0.199502, 0.200455]$
b_z^1	$[0, 0.2]$	$[0.0997107, 0.100714]$
b_x^2	$[0, 0.2]$	$[0.0996747, 0.100712]$
b_y^2	$[0, 0.2]$	$[0.0994585, 0.10031]$
b_z^2	$[0.1, 0.3]$	$[0.199535, 0.200642]$
b_x^3	$[0.1, 0.3]$	$[0.199689, 0.200578]$
b_y^3	$[0, 0.2]$	$[0.0997562, 0.100319]$
b_z^3	$[0, 0.2]$	$[0.0995661, 0.100557]$

Path planning

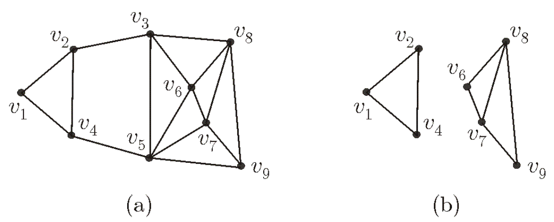
(Luc Jaulin, Friday, 15h15-16h00).

We shall present a recent approach to finding a collision-free path for an object in a environment cluttered with known obstacles. Most of the methods available in the literature are based on the concept of *configuration space* (or C-space). Each coordinate in C-space represents a degree of freedom of the object. The number of independent parameters needed to specify the configuration of this object corresponds to the dimension of C-space. The initial configuration and desired final configuration of the object become two points **a** and **b** in C-space.
The *feasible configuration space* $\mathbb{S}$ is a subset of C-space that only contains configuration vectors for which the object does not collide with obstacles. Path planning amounts to finding a path belonging to $\mathbb{S}$ from the initial point **a** to the desired point **b**.

Graph discretization of the configuration space
A guaranteed characterization of the feasible configuration space $\mathbb{S}$ can be obtained using a subdivision algorithm such as SIVIA.
The following figure describes a paving $\mathcal{P} = \{[\mathbf{p}_1], [\mathbf{p}_2], \dots, [\mathbf{p}_9]\}$ of the box $[\mathbf{p}_0] = [-2, 10] \times [-2, 6]$.



Two boxes in $\mathcal{P}$ are *neighbors* if they share, at least partly, an $(n-1)$-dimensional face. For instance, $[\mathbf{p}_1]$ and $[\mathbf{p}_4]$ are neighbors but $[\mathbf{p}_2]$ and $[\mathbf{p}_5]$ are not. Any paving $\mathcal{P}$ of a box $[\mathbf{p}_0]$ can be represented by a graph $\mathcal{G}$. Each element $[\mathbf{p}_i]$ of $\mathcal{P}$ is associated with a vertex v_i of $\mathcal{G}$. If two boxes $[\mathbf{p}_i]$ and $[\mathbf{p}_j]$ of $\mathcal{P}$ are neighbors, then $\mathcal{G}$ contains the edge $v_i v_j$. For instance, the graph $\mathcal{G}$ associated with the paving of the figure above is given in Subfigure a. Subpavings can also be represented by graphs (see Subfigure b).



Graph G and one of its subgraph.

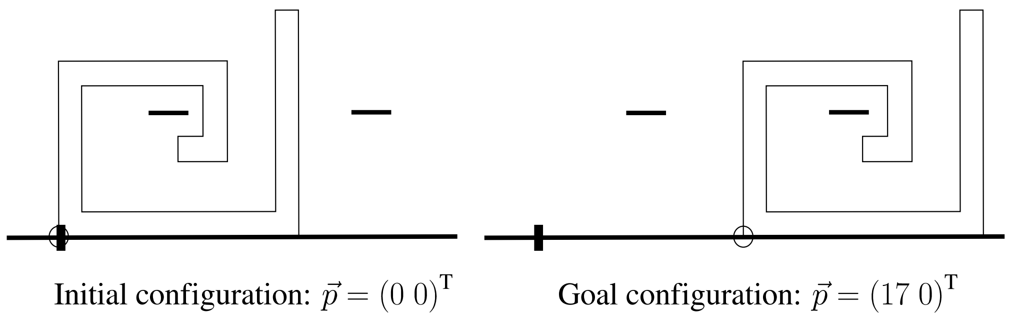
Among the algorithms that have been proposed for finding the shortest path between two specified vertices v_a and v_b in a graph $\mathcal{G}$, one of the most efficient is due to Dijkstra. With each vertex v of $\mathcal{G}$ is associated an integer $d(v)$

representing the minimum number of edges in a path from v_a to v. $\mathcal{G}(i)$, $i \in \mathbb{N}$ denotes the set of all vertices of $\mathcal{G}$ such that $d(v) = i$. If the algorithm DIJKSTRA returns an empty list $\mathcal{L}$, then v_a and v_b are not in the same connected component of $\mathcal{G}$. Otherwise, it returns one of the shortest paths from v_a to v_b, in terms of the number of vertices crossed.
Running DIJKSTRA$(\mathcal{G}, v_1, v_6)$ on the graph of Subfigure a, one gets $d(v_1) = 0$, $d(v_2) = d(v_4) = 1$, $d(v_3) = d(v_5) = 2$, $d(v_6) = d(v_7) = d(v_8) = d(v_9) = 3$. DIJKSTRA returns either the path $\{v_1, v_2, v_3, v_6\}$ or the path $\{v_1, v_4, v_5, v_6\}$.
A first approach to obtain a path from $\mathbf{a}$ to $\mathbf{b}$ is to use the procedure SIVIA to build a paving $\mathbb{P}$ and two subpavings $\underline{\mathbb{P}}$ and $\overline{\mathbb{P}}$ of $\mathbb{P}$ satisfying

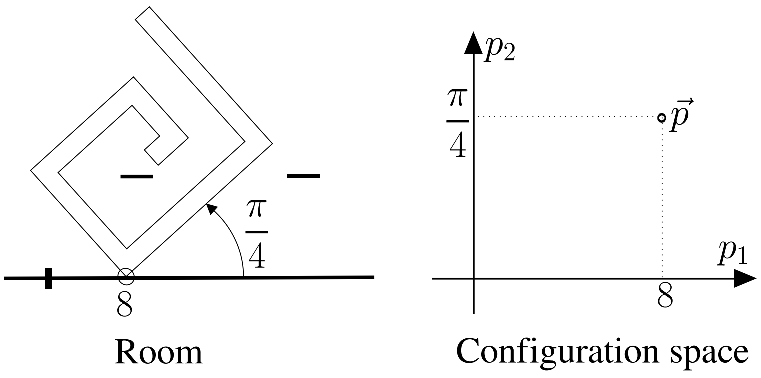
$$\underline{\mathbb{P}} \subset \mathbb{S} \subset \overline{\mathbb{P}}. \tag{215}$$

The graphs $\underline{\mathcal{G}}$ and $\overline{\mathcal{G}}$ associated with $\underline{\mathbb{P}}$ and $\overline{\mathbb{P}}$ are then built, and two boxes $[\mathbf{p}_a]$ and $[\mathbf{p}_b]$ of $\overline{\mathbb{P}}$ are selected, such that $\mathbf{a} \in [\mathbf{p}_a]$ and $\mathbf{b} \in [\mathbf{p}_b]$. Let v_a and v_b be the two vertices of $\overline{\mathcal{G}}$ associated with $[\mathbf{p}_a]$ and $[\mathbf{p}_b]$. One then calls the procedure DIJKSTRA to get a path $\overline{\mathcal{L}}$ of $\overline{\mathcal{G}}$ from v_a to v_b. If no such path is found, then $\mathbf{a}$ and $\mathbf{b}$ have been proved to belong to disconnected components of $\mathbb{S}$ and one reports that there can be no path. If a non-empty $\overline{\mathcal{L}}$ is found, then DIJKSTRA is run again to find a path $\underline{\mathcal{L}}$ of $\underline{\mathcal{G}}$ connecting v_a to v_b. If such a path $\underline{\mathcal{L}} = \{v_a, v_1, \ldots, v_{\ell-1}, v_b\}$ is found, then the associated box path in $\underline{\mathbb{P}}$ is included in $\mathbb{S}$ and a point path can thus be generated. If $\underline{\mathcal{L}}$ is empty, the algorithm reports failure because nothing has been proved yet about the existence or inexistence of a feasible path from $\mathbf{a}$ to $\mathbf{b}$. One may then run the algorithm again with a smaller ε.
A second approach takes into account that the time spent running DIJKSTRA is very short compared to that required to build a detailed characterization of the feasible configuration space. During each iteration, DIJKSTRA is used to locate the regions of configuration space that seem most promising, and the algorithm stops as soon as a feasible path has been found. Let $\mathbb{P}$ be the current paving of the search box $[\mathbf{p}_0]$. As in the first approach, we first look for a shortest path $\overline{\mathcal{L}}$ in the graph associated with an available subpaving $\overline{\mathbb{P}}$ of $\mathbb{P}$, which satisfies $\mathbb{S} \subset \overline{\mathbb{P}}$. If no such path is found, then $\mathbf{a}$ and $\mathbf{b}$ are not in the same connected component of $\mathbb{S}$ and the algorithm reports that there can be no path. If the path exists, then we try to find the shortest path $\underline{\mathcal{L}}$ in the graph $\underline{\mathcal{G}}$ associated with a subpaving $\underline{\mathbb{P}}$ of $\mathbb{P}$. If such a path is found, it is returned. Otherwise, since the box path corresponding to $\overline{\mathcal{L}}$ may nevertheless contain a feasible path, all subboxes of this path are bisected and a new paving $\mathbb{P}$ is thus obtained.

Test case
The configuration space of this test case was chosen two-dimensional, so that the feasible configuration set S can be visualized easily, but it suffices to try to find a solution to realize that the problem is nevertheless quite complicated.


Initial configuration: $\vec{p} = (0\ 0)^{\mathrm{T}}$
Goal configuration: $\vec{p} = (17\ 0)^{\mathrm{T}}$

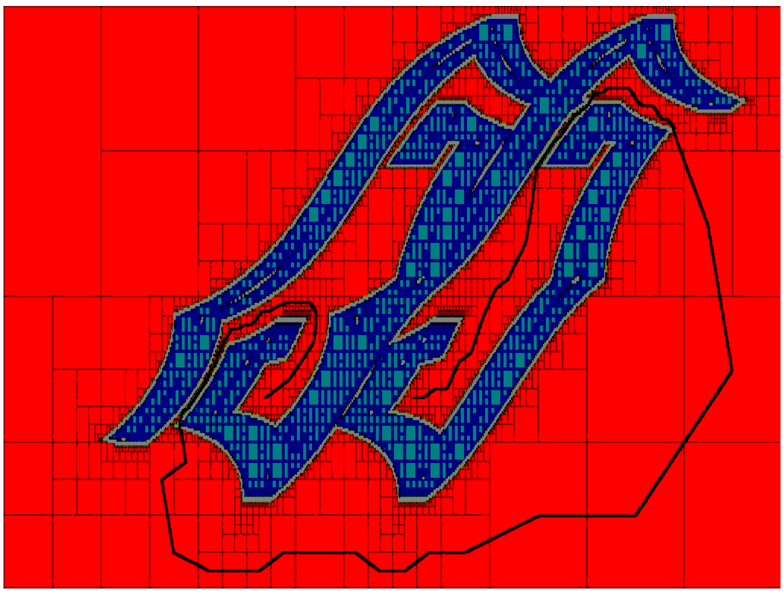
Consider a two-dimensional room that contains $\overline{j}$ segment obstacles. The extreme points of the jth obstacle are denoted by $\mathbf{a}_j$ and $\mathbf{b}_j$ for $j \in \mathcal{J} = \{1, \ldots, \overline{j}\}$. The object to be moved is a non-convex polygon with $\overline{i}$ vertices, denoted by $\mathbf{s}_i \in \mathbb{R}^2, i \in \mathcal{I} = \{1, \ldots, \overline{i}\}$. In the example to be treated, $\overline{j} = 2$ and $\overline{i} = 14$. The vertex $\mathbf{s}_1$ is constrained to stay on the horizontal axis of the room frame. The configuration of the object is thus represented by a two-dimensional vector $\mathbf{p} = (p_1, p_2)^{\mathrm{T}}$, where p_1 is the coordinate of $\mathbf{s}_1$ along the horizontal axis and p_2 is the heading angle of the object (in radians).
A vector $\mathbf{p}$ associated with a given configuration is feasible if and only if none of the edges of the object intersects any of the segment obstacles and the extreme points of each segment obstacle lay outside the object. As illustrated by below, $\mathbf{p} = (8,\ \pi/4)^{\mathrm{T}}$ is feasible.



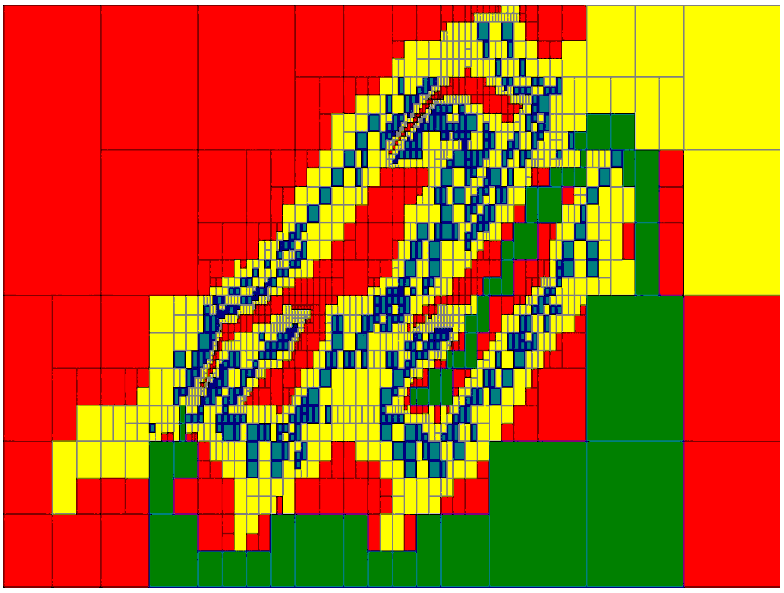
In what follows, $\mathrm{segm}(\mathbf{a},\mathbf{b})$ denotes the segment with end points $\mathbf{a}$ and $\mathbf{b}$, and $\mathrm{line}(\mathbf{a},\mathbf{b})$ is the straight line passing through $\mathbf{a}$ and $\mathbf{b}$. Since $[\mathbb{A}]$ denotes the interval hull of $\mathbb{A}$ (*i.e.*, the smallest box containing $\mathbb{A}$), $[\mathbf{a}\cup\mathbf{b}]$ will represent the smallest box that contains $\mathbf{a}$ and $\mathbf{b}$. If $\mathbf{s}_{\bar{i}+1}$ is taken equal to $\mathbf{s}_1$, then

$$(\mathbf{p}\in\mathbb{S})\Leftrightarrow\left(\begin{array}{l}\forall i\in\mathcal{I},\forall j\in\mathcal{J},\ \mathrm{segm}(\mathbf{s}_i,\mathbf{s}_{i+1})\cap\ \mathrm{segm}(\mathbf{a}_j,\mathbf{b}_j)=\emptyset\\ \text{and } \mathbf{a}_j \text{ and } \mathbf{b}_j \text{ are outside the object}\end{array}\right). \tag{216}$$

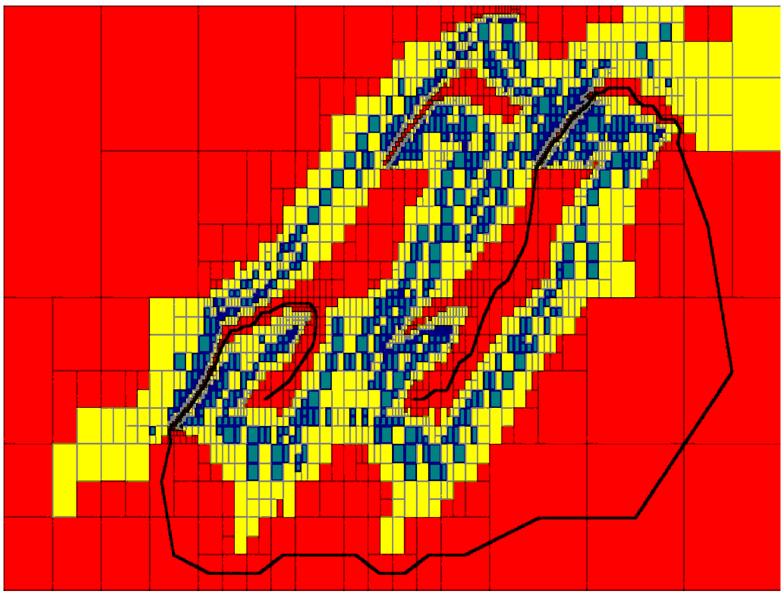
From this equivalence, we can obtain inequalities describing the set $\mathbb{S}$. The SIVIA algorithm generates the following paving. Then DIJKSTRA is then able to compute the path from a to $\mathbf{b}$.



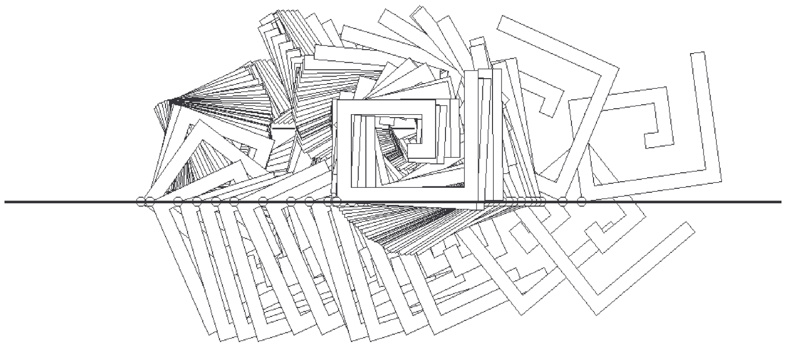
The principle of the second approach is described by the following figure. The DIJKSTRA find the green path. All ambiguous boxes of this green path will be bisected at the next iteration.



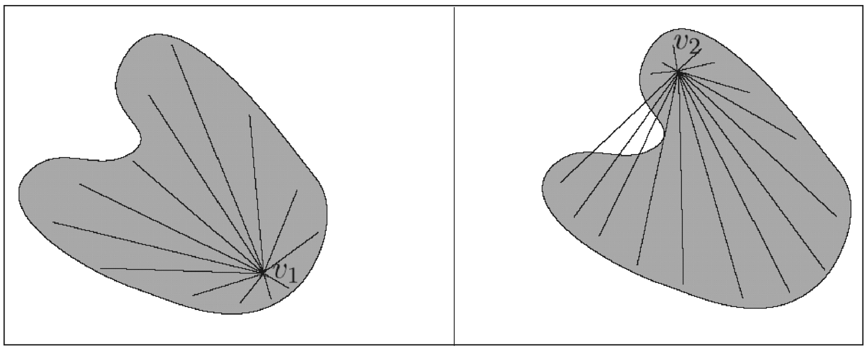
The final paving of the second approach as well as the corresponding path is represented below.



A display of the motion corresponding to the path is given on the picture.



Computing the number of connected components of a set (Collaboration with N. Delanoue and B. Cottenceau)
The point $\mathbf{v}$ is a *star* for $\mathbb{S} \subset \mathbb{R}^n$ if $\forall \mathbf{x} \in \mathbb{S}$, $\forall \alpha \in [0,1]$, $\alpha\mathbf{v} + (1-\alpha)\mathbf{x} \in \mathbb{S}$. For instance, in the figure below $\mathbf{v}_1$ is a star for $\mathbb{S}$ whereas $\mathbf{v}_2$ is not.

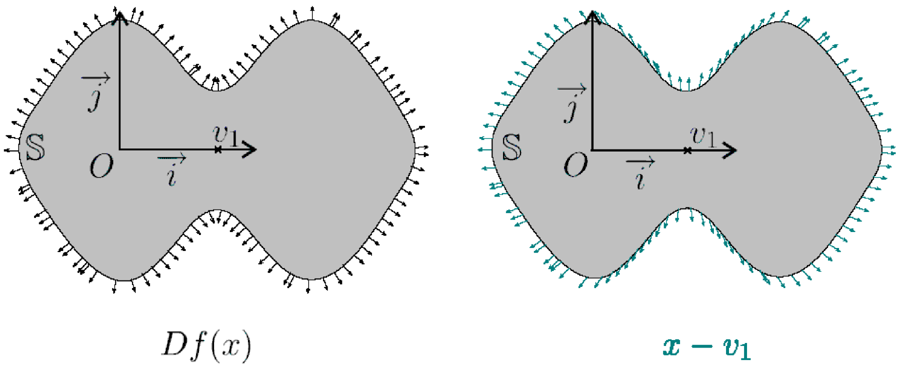


The set $\mathbb{S} \subset \mathbb{R}^n$ is *star-shaped* is there exists $\mathbf{v}$ such that $\mathbf{v}$ is a star for $\mathbb{S}$.
Theorem: Define the set

$$\mathbb{S} \overset{\text{def}}{=} \{\mathbf{x} \in [\mathbf{x}] | f(\mathbf{x}) \leq 0\} \tag{217}$$

where f is differentiable. We have the following implication

$$\{\mathbf{x} \in [\mathbf{x}] \mid f(\mathbf{x}) = 0, Df(\mathbf{x}).(\mathbf{x}-\mathbf{v}) \leq 0\} = \emptyset \Rightarrow \mathbf{v} \text{ is a star for } \mathbb{S}. \tag{218}$$



If $\mathbf{v}$ is a star for $\mathbb{S}_1$ and a star for $\mathbb{S}_2$ then it is a star for $\mathbb{S}_1 \cap \mathbb{S}_2$ and for $\mathbb{S}_1 \cup \mathbb{S}_2$. As a consequence, interval methods can be used to prove that a point $\mathbf{v}$ is a star for a set defined by a conjunction or a disjunction of inequalities.

Consider a subpaving $\mathcal{P} = \{[\mathbf{p}_1], [\mathbf{p}_2], \dots\}$ covering $\mathbb{S}$. The relation $\mathcal{R}$ defined by

$$[\mathbf{p}]\mathcal{R}[\mathbf{q}] \Leftrightarrow \mathbb{S} \cap [\mathbf{p}] \cap [\mathbf{q}] \neq \emptyset \tag{219}$$

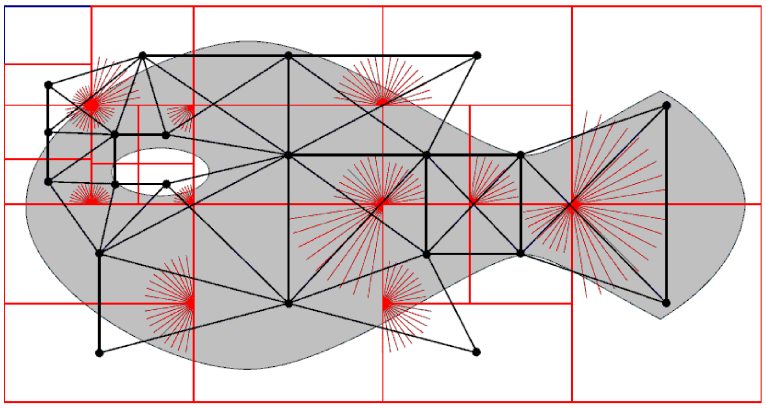
is *star-spangled graph* of the set $\mathbb{S}$ if

$$\forall [\mathbf{p}] \in \mathcal{P}, \mathbb{S} \cap [\mathbf{p}] \text{ is star-shaped.} \tag{220}$$

For instance, a star-spangled graph for the set

$$\mathbb{S} \overset{\text{def}}{=} \left\{ (x,y) \in \mathbb{R}^2 \mid \begin{pmatrix} x^2 + 4y^2 - 16 \\ 2\sin x - \cos y + y^2 - \frac{3}{2} \\ -(x+\frac{5}{2})^2 - 4(y-\frac{2}{5})^2 + \frac{3}{10} \end{pmatrix} \leq 0 \right\}, \tag{221}$$

obtained using the solver CIA (`http://www.istia.univ-angers.fr/~delanoue/`), is given below.



For each $[\mathbf{p}]$ of the paving $\mathcal{P}$, a common star located at the corner of $[\mathbf{p}]$ (represented in red) has been found for all three constraints.
Theorem: The number of connected components of the star-spangled graph of $\mathbb{S}$ is equal to that of $\mathbb{S}$.
An extension of this approach has also been developed by N. Delanoue to compute a triangulation homeomorphic to $\mathbb{S}$.