Codac and Vibes manual

Benoît Desrochers, Luc Jaulin and Simon Rohou

Errata.

The `codac` library replaces `pyIbex` as a consequence
`from pyIbex import *` should be replaced by `from codac import *`

1 Codac

CODAC (http://codac.io/) is a PYTHON library which makes it possible to use IBEX under a PYTHON environment. A documentation on IBEX can be found at http://ibex-team.github.io/ibex-lib/

Elementary interval functions. Basic functions for real numbers such as `sin`, `cos`, `tan`, `acos`, `asin`, `atan`, `log`, `exp` are extended to intervals. For instance:

```
sin(Interval(0,3))
```

returns the interval $[0, 1]$.

Import. Use the `import` statement to import CODAC. For instance, to import the classes `Interval` and `IntervalVector` from CODAC, write:

```
from codac import Interval, IntervalVector
```

Function. To create a new function, use `Function` from IBEX. For instance,

```
f = Function("x[2]","(x[0]-1)^2+(x[1]-2)^2")
```

creates the function

$$\mathbf{f}: \begin{array}{ccc} \mathbb{R}^2 & \mapsto & \mathbb{R} \\ \mathbf{x} & \to & (x_1-1)^2+(x_2-2)^2 \end{array}$$

Equivalently, we could have written

```
f = Function("x1","x2","(x1-1)^2+(x2-2)^2")
```

or

```
f = Function("x1","x2","(x1-%f)^2+(x2-%f)^2"%(1.0,2.0))
```

For the evaluation

```
f.eval(IntervalVector([[1,2],[3,4]])))
```

For vector valued functions, the syntax is similar. For instance a translation by the vector $(1, -2)$ is defined by:

```
f=Function("x1","x2","(x1+1;x2-2)")
```

For the evaluation

```
f.eval_vector(IntervalVector([[1,2],[3,4]])))
```

Image. The optimal contractor for an image is obtained as follows

```
C = CtcRaster(image, -5, 5, 0.1, -0.1)
```

An example is given here: https://replit.com/@aulin/ctcimage

Interval. An interval is defined by its lower and upper bound. For instance, to define the interval $x = [-2, 4]$, write:

```
x=Interval(-2,4)
```

Note that the lower bound must be smaller than upper bound. The statement:

```
y=Interval(5)
```

defines the degenerated interval (or singleton) $y = [5, 5] = \{5\}$. To define the interval $z = \mathbb{R} = [-\infty, \infty]$, write

```
z=Interval.(-oo,oo)
```

To define the intervals $[\pi, \pi]$, $\emptyset$ we write: `Interval.PI`, `Interval.EMPTY_SET.`
The basic methods for intervals are `lb()` which returns the lower bound, `ub()` which returns the upper bound, `diam()` which returns the width, `mid()` which returns center and `is_empty()` which returns true if empty.

IntervalVector. An `IntervalVector` (also called a box) is a Cartesian product of intervals. In Codac, it can be defined from a 2D array or a tuple. For instance, to create the box $x = [1, 3] \times [-2, 9] \times [1, 10^3]$, write

```
x=IntervalVector([[1,3],[-2,9],[1,10**3]])
```

or equivalently

```
x=IntervalVector((Interval(1,3),Interval(-2, 9),Interval(1, 10**3)))
```

To create the box $[-2, 3]^{\otimes n} = [-2, 3] \times \cdots \times [-2, 3]$ write:

```
IntervalVector(n,Interval(-2,3)).
```

To create the degenerated box $[1, 1] \times [2, 2] \times [3, 3]$, write:

```
x=IntervalVector([1,2,3]).
```

To access the ith element of an `IntervalVector x` write `x[i-1]`. For instance, the first element of `x` previously defined can be obtained by `x[0]`.

IntervalMatrix. An `IntervalMatrix` (also called a box) is matrix of intervals. For instance, to create the interval matrix

$$\mathbf{M} = \begin{pmatrix} 1 & [2,3] \\ [4,5] & [6,7] \end{pmatrix}$$

write

```
J=IntervalMatrix(2,2)
J[0][0]=Interval(1,1) J[0][1]=Interval(2,3)
J[1][0]=Interval(4,5) J[1][1]=Interval(6,7)
```

Interval operators. The intersection $\cap$, union hull $\sqcup$, sum $+$, difference $-$, multiplication $*$ etc. are made using the overloaded operators `&`, |,`+`, `-`, `*`, etc. For instance to perform $([1, 2] + [3, 5]) \cup [9, 10]$, write:

```
(Interval(1,2)+Interval(3,5)) | Interval(9,10)
```

and you will get the interval $[4, 10]$.

Relaxed intersection. If L is a list of separators, the q relaxed intersection of L is performed using SepQInter. For instance is $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ are three separators, we obtain the separator

$$\mathcal{S} = \bigcap^{\{1\}} \mathcal{S}_i$$

as follows:

```
L=[S1,S2,S3]
S=SepQInter(L)
S.q=1
```

Own contractor. To build your own contractor, you should build a class that inherits from the IBEX main class Ctc. For instance if you want to build from scratch the contractor associated with the equation

$$x_1^2 + x_2^2 \in [4,5]\,,$$

```
class myCtc(Ctc):
        def __init__(C):
                Ctc.__init__(C, 2)
        def contract(C, X):
                x, y = X[0], X[1]
                r2 = Interval(4,5)
                x2,y2 = sqr(x),sqr(y)
                bwd_add(r2,x2,y2)
                bwd_sqr(x2,x)
                bwd_sqr(y2,y)
```

To create an instance of myCtc, write

```
C1= myCtc()
```

Since this contractor inherits from Ctc, you will be able to compose it as an actual contractor.

Own flattened contractor. A flattened contractor is a contractor whose input/output of which is a list of intervals (instead of a box). To build a your own flattened contractor, you should build a class for it. For instance, assume that you want to build a contractor for the distance equation

$$(x-a)^2 + (y-b)^2 \in [d]^2$$

where a, b, x, y are the variables the domains of which should be contracted. The interval $[d]$ is an interval parameter. You should first build the associated class

```
class my_flattenedcontractor():
        def __init__(C):
                f=Function("x","y","a","b","d","(x-a)^2+(y-b)^2-d^2")
                C.Cm=CtcFwdBwd(f)
        def contract(C,x,y,a,b,d):
                X=IntervalVector(5)
                X[0],X[1],X[2],X[3],X[4]=x,y,a,b,d
                C.Cm.contract(X)
```

```
        x,y,a,b =X[0],X[1],X[2],X[3]
        return x,y,a,b
```

Then, you create you contractor:

```
Cdist=my_flattenedcontractor()
```

and you use it as follows:

```
for k in range(0,10):
    X[k],Y[k],A[k],B[k]=Cdist.contract(X[k],Y[k],A[k],B[k],D[k])
```

Polar. The optimal separator for the set

$$\mathbb{P} = \left\{(x,y) \in \mathbb{R}^2 \mid \exists \rho \in [\rho], \exists \theta \in [\theta] \text{ s.t. } x = \rho \cos\theta \text{ and } y = \rho \sin\theta\right\}$$

is defined as follows:

```
S=SepPolarXY(Rho,Theta)
```

where `Rho,Theta` correspond to the intervals $[\rho], [\theta]$.

Polygon. The optimal separator of the following polygon

$$P = \begin{pmatrix} 6 & 7 & 0 & -9 & -8 \\ -6 & 9 & 5 & 8 & -9 \end{pmatrix}$$

is constructed as follows:

```
S = SepPolygon([[6, -6], [7, 9], [0, 5], [-9, 8], [-8, -9]])
```

Projection. We can build a separator S2 associated to the projection of a set defined by the separator S1 using `SepProj`. The following example provides a separator S2 associated with the set

$$\left\{\mathbf{x} \in \mathbb{R}^2 \,| \exists \mathbf{a} \in [0,1]^2,\, (x_1 - a_1)^2 + (x_2 - a_2)^2 \in [4,9]\right\}$$

```
f = Function("x1","x2","a1","a2","(x1-a1)^2+(x2-a2)^2")
S1=SepFwdBwd(f,Interval(4,9))
A=IntervalVector([[-1,1],[-1,1]])
S2=SepProj(S1,A,0.001)
```

In this example, S1 is a separator of dimension 4 whereas S2 is of dimension 2. In the same manner, we can build the projection for a contractor.

Separator. To initialize a separator from a function, use the IBEX syntax. For instance a separator associated with the set

$$\mathbb{X} = \{\mathbf{x} | \mathbf{f}(\mathbf{x}) \in [1,2]\}$$

obtained by a forward-backward procedure is performed as follows:

```
S=SepFwdBwd(f,Interval(1,2))
```

Separator operations. The intersection, the union and the complement of separators is obtained using & , | and $\sim$. For instance if $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$ are three separators, then, the separator $\mathcal{S} = (\mathcal{S}_1 \cap \mathcal{S}_2) \cup (\mathcal{S}_2 \cap \mathcal{S}_3) \cup \left(\mathcal{S}_1 \cap \overline{\mathcal{S}_3}\right)$ is obtained by:

```
S=(S1&S2)|(S2&S3)|(S1&(~S3)).
```

SIVIA. Sivia admits as an input an initial box $[\mathbf{x}]$, a separator $\mathcal{S}$, and an accuracy ε. For instance to run Sivia with an initial box `X`, a separator `S` and an accuracy `0.1`, write

```
pySIVIA(X,S,0.1)
```

If we want to change the colors :

```
color = {'color_in':'black[red]', 'color_out':'blue[cyan]', 'color_maybe':'white[
pySIVIA(X, S, 0.1, **color)
```

With `codac` use `SIVIA` instead of `pySIVIA`. For instance:

```
SIVIA(X,S,0.1,fig_name='name',color_map={SetValue.IN:"red[magenta]",
SetValue.OUT:"blue[cyan]",SetValue.UNKNOWN:"yellow[white]"}).
```

Transformation of a separator. If $\mathbf{f}$ and $\mathbf{g}$ are two functions from $\mathbb{R}^n \to \mathbb{R}^n$ such that $\mathbf{g}$ is the reciprocal function of $\mathbf{f}$ and if $\mathcal{S}_1$ is a separator, then we define the transformation of $\mathcal{S}$ by $\mathbf{f}$ as follows

```
S2=SepTransform(S1,f,g)
```

If $\mathcal{S}_1$ is a separator for $\mathbb{S}_1$ then $\mathcal{S}_2$ is a separator for $\mathbb{S}_2 = \mathbf{f}(\mathbb{S}_1)$.

Own separator from two contractors. From two complementary contractors C_{in}, C_{out}, you can build a separator S. You should first build the associated class

```
class mySep(Sep):
    def __init(S):
        Sep.init__(S, 2)
    def separate(S, Xin, Xout):
        Cout.contract(Xout)
        Cin.contract(Xin)
```

Then, you create you separator:

```
S=mySep()
```

2 Vibes

The library VIBES is used only for drawing. For more details about VIBES, see:

`http://enstabretagnerobotics.github.io/VIBES/`

Initialization. First, import all functions from VIBES, initialize VIBES, create a new figure and set the properties of the figure:

```
from vibes import *
vibes.beginDrawing()
vibes.newFigure('name')
vibes.setFigureProperties({'x':200, 'y':100,'width':800, 'height':800})
```

where `x`, `y` corresponds to the upper left corner (in pixel), `width`, `height` are also in pixel.

Draw a box. To draw the box $[1, 2] \times [3, 4]$ with the boundary blue and painted cyan inside, write

```
vibes.drawBox(1,2,3,4,'blue[cyan]')
```

Draw a circle. To draw a circle with center $(1, 2)$ and a radius 3 with the boundary in red and painted magenta inside, write

```
vibes.drawCircle(1,2,3,'red[magenta]')
```

Moving in Vibes. Zoom in: press '+'; Zoom out: press '−'; Move left: left arrow; Move right: right arrow.

For more details, have a look to Vibes C++ API

```
http://enstabretagnerobotics.github.io/VIBES/doxygen/cxx/
```