EASIBEX-MATLAB: a simple tool to begin with interval contractors

Fabrice Le Bars[1], Jeremy Nicola[1,2] and Luc Jaulin[1]

[1] ENSTA Bretagne (ex ENSIETA), STIC/OSM, Lab-STICC/CID/IHSEV,
2 Rue F. Verny, 29806, Brest Cedex 9, France
{fabrice.le_bars, luc.jaulin}@ensta-bretagne.fr
[2] iXBlue,
Rue Rivoalon, Sainte Anne du Portzic, 29200, Brest, France
jeremy.nicola@gmail.com

Keywords: EASIBEX, MATLAB, interval, contractors

Introduction

EASIBEX-MATLAB is a simple tool to start using interval arithmetic and contractors. It uses IBEX (see [1]) as internal library. It is designed for people that do not feel comfortable with C++, Python, object-oriented programming, but want to quickly prototype and test programs using interval arithmetic and contractors, while manipulating easily the results thanks to MATLAB.

Description

The main goals of EASIBEX-MATLAB are the following:

- Start using interval arithmetic and contractors.
- Quickly prototype and test new algorithms.

The target users that would find EASIBEX-MATLAB useful are:

- Students.

- Scientists that do not know advanced computer science languages and paradigms such as C++, Python and object-oriented programming, but would like to use classical interval algorithms for their own problems, or to prototype new algorithms.

The base and philosophy of EASIBEX-MATLAB can be sum up as these ideas:

- It is designed to be a very simple MATLAB layer of IBEX to benefit from several advanced, efficient and already tested algorithms.
- The naming conventions and way of use is strongly based on the very simple interval library used in several existing samples (see [2], [3]).
- It can be also used with VIBes ([4]) to easily draw the results of interval computations.

Due to its design, EASIBEX-MATLAB has some inherent limitations:

- To keep it simple, not all the features provided by IBEX are available.
- Even if the computations are made by IBEX, no study has been made to check if the guarantee of the results (w.r.t. rounding, etc.) is lost when passing parameters and retrieving results through MATLAB functions and shared library calls.
- The function calls are simplified and different from IBEX to avoid the difficulties of object-oriented paradigm (IBEX uses notions such as inheritance, polymorphism, etc.).

If C++ is needed, EASIBEX-CPP provides a very simple way to start using IBEX and benefit from its features in C++, without extended knowledge of object-oriented programming. See [5] for several guides and examples. Once you are comfortable with the notions of intervals, contractors as well as C++, you can be more efficient by using directly IBEX.

Quick tutorial

To start using EASIBEX-MATLAB, download and extract
https://github.com/ENSTABretagneRobotics/EASIBEX-MATLAB/archive/master.zip:

- In MATLAB, go to File\Set Path...\Add Folder... and add this folder as well as x86 folder if you use MATLAB 32 bit or x64 folder if you use MATLAB 64 bit.
- Run sivia_easibex.m to test. A red ring on a blue background and with yellow borders should appear.

To define an EASIBEX-MATLAB interval :

```
x=[-2,2]
```

`x(1,2)` would be 2. An empty interval would be:

```
x=[NaN,NaN]
```

and infinity:

```
x=[-Inf,Inf]
```

To define a box:

```
x=[[-2,2];[2,4];[-4,1]]
```

`x(2,:)` would be $[2, 4]$.

2 intervals can be added using:

```
Z=i_Add([0,2],[-1,2])
```

and 2 boxes:

```
Z=i_Add([[0,1];[0,10];[0,10]],[[-1,0];[2,5];[-1,0]])
```

To contract 3 intervals $Z = [-10, 1]$, $X = [0, 2]$, $Y = [-1, 2]$ knowing the constraint $Z = X + Y$:

```
[Z,X,Y]=i_Cadd([-10,1],[0,2],[-1,2])
```

To contract the vector $\mathbf{x} = [-10, 10] \times [0, 10] \times [-10, 0]$ w.r.t. the q-relaxed intersection (see e.g. [6]) of the 4 vectors $[-2, 2] \times [2, 4] \times [-4, 1]$, $[-1, 5] \times [-5, 8] \times [-7, 2]$, $[-1, 1] \times [0, 2] \times [1, 2]$, $[-2, 2] \times [2, 8] \times [-1, 2]$, with $q = 2$:

```
x = [[-10,10];[0,10];[-10,0]]
y_j = {[[-2,2];[2,4];[-4,1]];[[-1,5];[-5,8];[-7,2]];
[[-1,1];[0,2];[1,2]];[[-2,2];[2,8];[-1,2]]}
x = i_C_q_in(x, 2, y_j)
```

Quick reference

The current operations, functions and contractors available include $+$, $-$, $*$, $/$, intersection, union, $\sqrt{}$, exp, sin, arctan, min, abs, sign, determinant, scalar product, norm, distance, q-intersection, not inside, inside segment, inside circle, inside ring...

References

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

[**2**] L. Jaulin, M. Kieffer, O. Didrit and E. Walter, *Applied Interval Analysis with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, 2001.

[**3**] https://github.com/ENSTABretagneRobotics/interval/

[**4**] V. Drevelle and J. Nicola, VIBes: A Visualizer for Intervals and Boxes, *Mathematics in Computer Science* 8(3–4):563–572, 2014.

[**5**] https://www.ensta-bretagne.fr/jaulin/easibex.html

[**6**] L. Jaulin, Robust set membership state estimation ; Application to Underwater Robotics, *Automatica* 45(1):202–206, 2009.