

# An asymptotic minimal contractor for non-linear equations provided in the Codac library

Simon Rohou

ENSTA Bretagne, Lab-STICC, UMR CNRS 6285, Brest, France

SWIM 2024, Maastricht  
6<sup>th</sup> June 2024



## Section 1

# Introduction

## Introduction

# Problem statement

We consider the problem of approximating the solutions of the system:

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

where  $\mathbf{f} : \mathbb{R}^n \rightarrow \mathbb{R}^p$  is a non-linear and differentiable function.  
Possibly non continuous.

In particular, we will consider systems where:

$$p < n$$

..for which the solution set  $\mathbb{X} = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{f}(\mathbf{x}) = \mathbf{0}\}$  has infinitely many solutions.

## Introduction

# Set-inversion problem

Set inversion can be done with

- **forward** evaluations of  $\mathbf{f}$   
(see *e.g.* SIVIA)
  - limits in high dimensions due to bisections
- **forward/reverse** evaluations of  $\mathbf{f}$   
(such as contractors + pavers)
  - lower complexity, bisections used as a last resort
  - efficient for constraint propagation

## Introduction

# Reverse evaluations

### Forward evaluations:

`fwd_plus([x],[y])`: returns  $\{z \mid \exists x \in [x], \exists y \in [y], z = x + y\}$

Nathalie Revol. Introduction to the IEEE 1788-2015 standard for interval arithmetic. International Workshop on Numerical Software Verification. pp. 14-21, Springer, Germany (2017)

## Introduction

# Reverse evaluations

### Forward evaluations:

`fwd_plus([x],[y])`: returns  $\{z \mid \exists x \in [x], \exists y \in [y], z = x + y\}$

and also `fwd_cos`, `fwd_exp`, `fwd_matrix_product`, *etc*

Nathalie Revol. Introduction to the IEEE 1788-2015 standard for interval arithmetic. International Workshop on Numerical Software Verification. pp. 14-21, Springer, Germany (2017)

## Introduction

## Reverse evaluations

**Forward evaluations:**

`fwd_plus([x],[y])`: returns  $[\{z \mid \exists x \in [x], \exists y \in [y], z = x + y\}]$

and also `fwd_cos`, `fwd_exp`, `fwd_matrix_product`, *etc*

**Reverse (backward) evaluations:**

`bwd_plus([z],[x],[y])`: returns  $[\{(x, y) \in ([x], [y]) \mid x + y \in [z]\}]$

Nathalie Revol. Introduction to the IEEE 1788-2015 standard for interval arithmetic. International Workshop on Numerical Software Verification. pp. 14-21, Springer, Germany (2017)

## Introduction

## Reverse evaluations

**Forward evaluations:**

`fwd_plus([x],[y])`: returns  $\{z \mid \exists x \in [x], \exists y \in [y], z = x + y\}$

and also `fwd_cos`, `fwd_exp`, `fwd_matrix_product`, *etc*

**Reverse (backward) evaluations:**

`bwd_plus([z],[x],[y])`: returns  $\{(x, y) \in ([x], [y]) \mid x + y \in [z]\}$

and also `bwd_cos`, `bwd_exp`, `bwd_matrix_product`, *etc*

Nathalie Revol. Introduction to the IEEE 1788-2015 standard for interval arithmetic. International Workshop on Numerical Software Verification. pp. 14-21, Springer, Germany (2017)



## Section 2

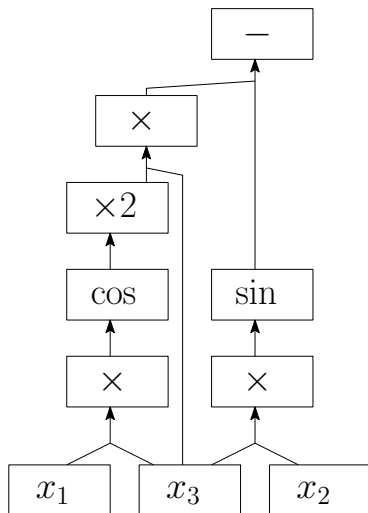
# The HC4Revise algorithm

The HC4Revise algorithm

"Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$$



F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244

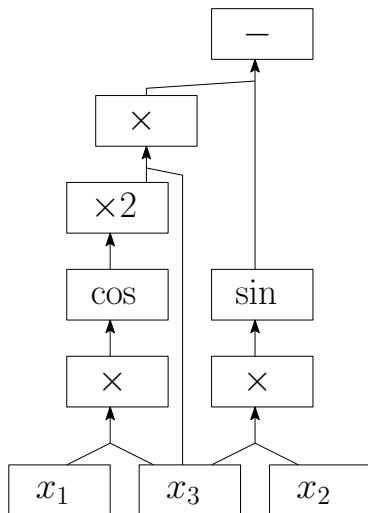
The HC4Revise algorithm

## "Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$$

Considered constraint:  $f(\mathbf{x}) \in [y]$



F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244

The HC4Revise algorithm

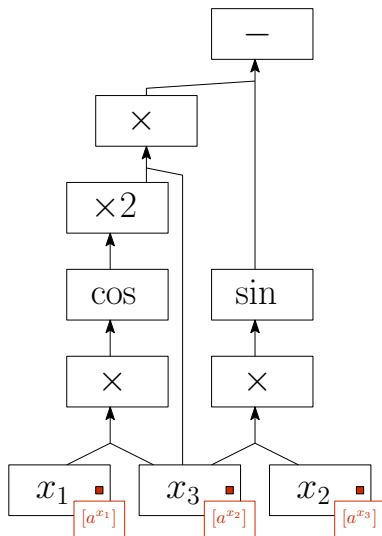
## "Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$$

Considered constraint:  $f(\mathbf{x}) \in [y]$

1. initializing ( $[a^{x_1}]$ ,  $[a^{x_2}]$ ,  $[a^{x_3}]$ )



F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244

The HC4Revise algorithm

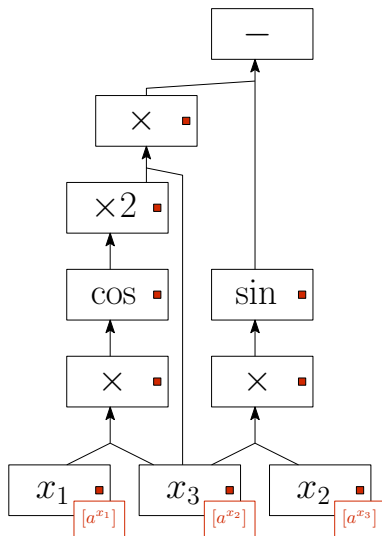
## "Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1 x_3) - \sin(x_2 x_3)$$

Considered constraint:  $f(\mathbf{x}) \in [y]$

1. initializing ( $[a^{x_1}]$ ,  $[a^{x_2}]$ ,  $[a^{x_3}]$ )
2. forward propagation:  
from  $(x_1, x_2, x_3)$  to  $f(\mathbf{x})$



F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244

The HC4Revise algorithm

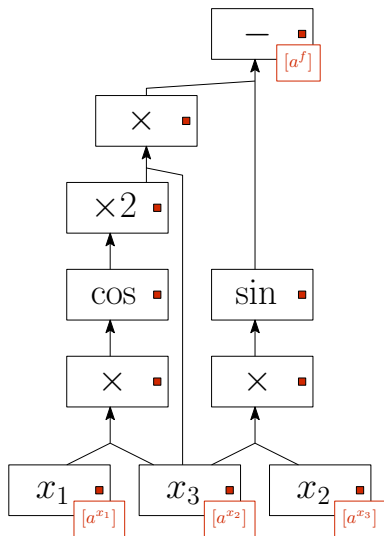
## "Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$$

Considered constraint:  $f(\mathbf{x}) \in [y]$

1. initializing ( $[a^{x_1}]$ ,  $[a^{x_2}]$ ,  $[a^{x_3}]$ )
2. forward propagation:  
from  $(x_1, x_2, x_3)$  to  $f(\mathbf{x})$
3. intersecting  $[a^f]$  with  $[y]$



F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244

The HC4Revise algorithm

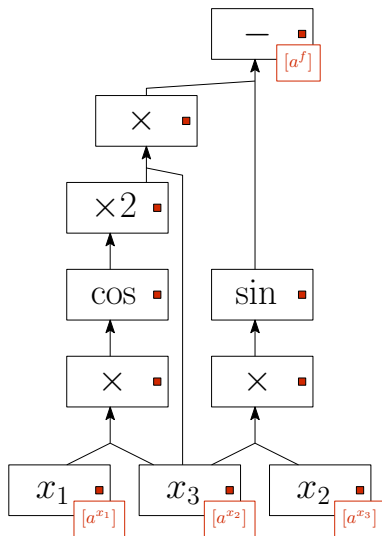
## "Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$$

Considered constraint:  $f(\mathbf{x}) \in [y]$

1. initializing ( $[a^{x_1}]$ ,  $[a^{x_2}]$ ,  $[a^{x_3}]$ )
2. forward propagation:  
from  $(x_1, x_2, x_3)$  to  $f(\mathbf{x})$
3. intersecting  $[a^f]$  with  $[y]$
4. reverse (backward) propagation:  
from  $f(\mathbf{x})$  to  $(x_1, x_2, x_3)$



F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244

The HC4Revise algorithm

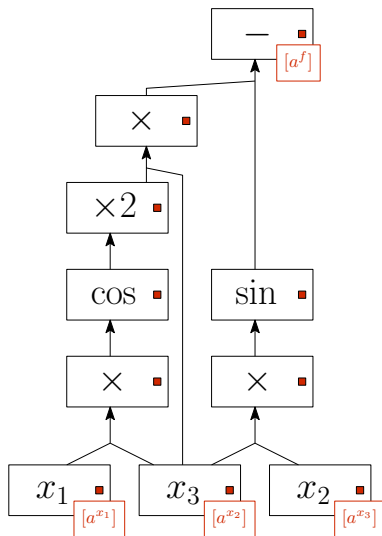
## "Forward-backward" algorithm

DAG associated with:

$$f(\mathbf{x}) = 2x_3 \cos(x_1 x_3) - \sin(x_2 x_3)$$

Considered constraint:  $f(\mathbf{x}) \in [y]$

1. initializing ( $[a^{x_1}]$ ,  $[a^{x_2}]$ ,  $[a^{x_3}]$ )
2. forward propagation:  
from  $(x_1, x_2, x_3)$  to  $f(\mathbf{x})$
3. intersecting  $[a^f]$  with  $[y]$
4. reverse (backward) propagation:  
from  $f(\mathbf{x})$  to  $(x_1, x_2, x_3)$
5. possible contraction  
of ( $[a^{x_1}]$ ,  $[a^{x_2}]$ ,  $[a^{x_3}]$ )



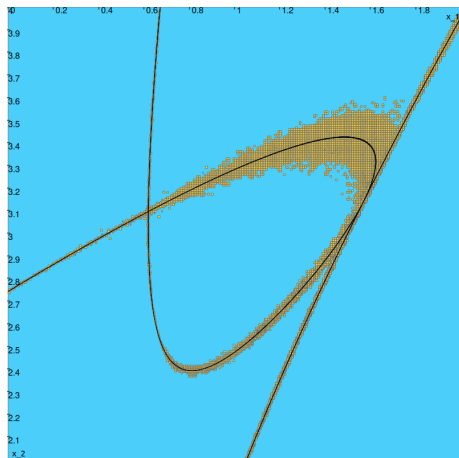
F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget, "Revising hull and box consistency," in Logic Programming: The 1999 International Conference, Las Cruces, New Mexico, USA, November 29 - December 4, 1999, D. D. Schreye, Ed. MIT Press, 1999, pp. 230-244



## The HC4Revise algorithm

## Limits of the algorithm

$$\begin{pmatrix} -x_3^2 + 2x_3 \sin(x_3x_1) + \cos(x_3x_2) \\ 2x_3 \cos(x_3x_1) - \sin(x_3x_2) \end{pmatrix} = \mathbf{0}$$



## Example from:

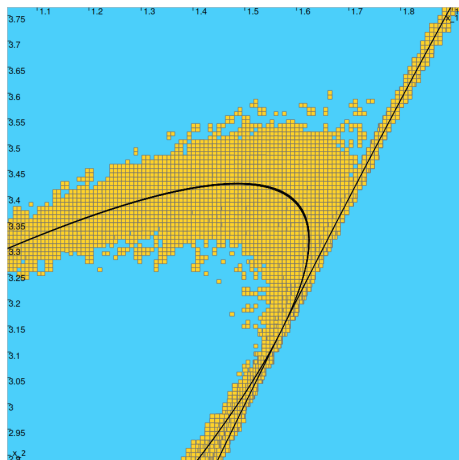
R. Malti, M. Rapaić, and V. Turkulov.  
A unified framework for robust  
stability analysis of linear  
irrational systems in the parametric  
space. *Annual Reviews in Control*,  
vol. 57, 2024.

- delayed systems
- $\mathbf{x} \in [0, 2] \times [2, 4] \times [0, 10]$
- projected approxima-  
tions for  $\mathbf{x}_{1,2}$

## The HC4Revise algorithm

## Limits of the algorithm

$$\begin{pmatrix} -x_3^2 + 2x_3 \sin(x_3 x_1) + \cos(x_3 x_2) \\ 2x_3 \cos(x_3 x_1) - \sin(x_3 x_2) \end{pmatrix} = \mathbf{0}$$



## Example from:

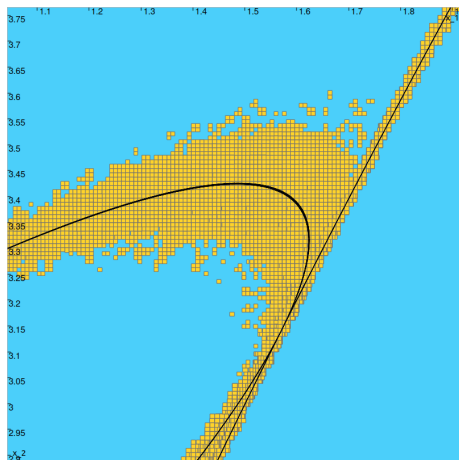
R. Malti, M. Rapaić, and V. Turkulov.  
A unified framework for robust  
stability analysis of linear  
irrational systems in the parametric  
space. *Annual Reviews in Control*,  
vol. 57, 2024.

- delayed systems
- $\mathbf{x} \in [0, 2] \times [2, 4] \times [0, 10]$
- projected approximations for  $\mathbf{x}_{1,2}$

## The HC4Revise algorithm

## Limits of the algorithm

$$\begin{pmatrix} -x_3^2 + 2x_3 \sin(x_3 x_1) + \cos(x_3 x_2) \\ 2x_3 \cos(x_3 x_1) - \sin(x_3 x_2) \end{pmatrix} = \mathbf{0}$$



## Example from:

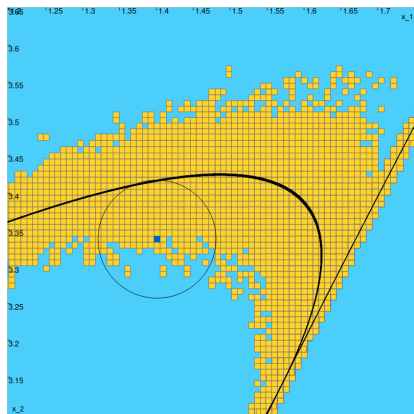
R. Malti, M. Rapaić, and V. Turkulov.  
A unified framework for robust  
stability analysis of linear  
irrational systems in the parametric  
space. Annual Reviews in Control,  
vol. 57, 2024.

- delayed systems
- $\mathbf{x} \in [0, 2] \times [2, 4] \times [0, 10]$
- projected approximations for  $\mathbf{x}_{1,2}$

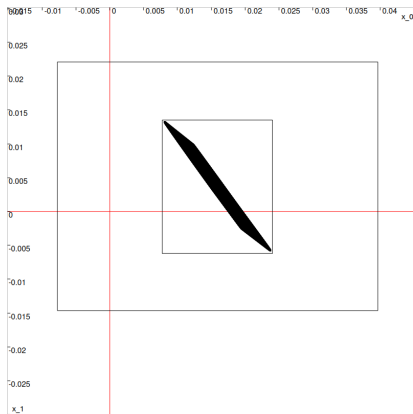
**Two problems**

The HC4Revise algorithm

Clustering effect due to: [1] dependency problem



$\mathbb{R}^n$  space (projection)

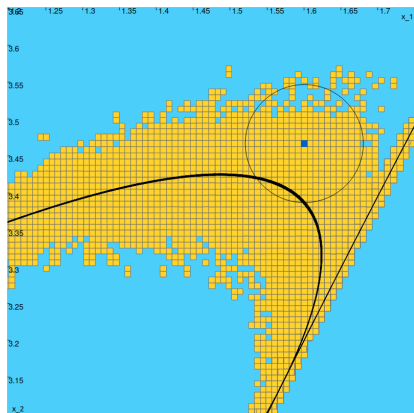


$\mathbb{R}^p$  space

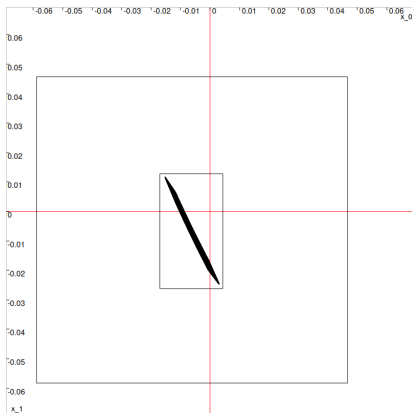
$f$

The HC4Revise algorithm

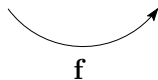
Clustering effect due to: [2] wrapping effect



$\mathbb{R}^n$  space (projection)



$\mathbb{R}^p$  space



**f**

The HC4Revise algorithm

Solutions for these clustering effects

**For [1]: dependency problem**

**For [2]: wrapping effect**

The HC4Revise algorithm

## Solutions for these clustering effects

### **For [1]: dependency problem**

- centered form provides minimal results with narrow boxes

### **For [2]: wrapping effect**

The HC4Revise algorithm

## Solutions for these clustering effects

### **For [1]: dependency problem**

- centered form provides minimal results with narrow boxes
- not used classically for reverse (backward) propagations

### **For [2]: wrapping effect**



The HC4Revise algorithm

## Solutions for these clustering effects

### **For [1]: dependency problem**

- centered form provides minimal results with narrow boxes
- not used classically for reverse (backward) propagations

### **For [2]: wrapping effect**

- the hull box of the image evaluation may unfortunately contain zero

The HC4Revise algorithm

## Solutions for these clustering effects

### **For [1]: dependency problem**

- centered form provides minimal results with narrow boxes
- not used classically for reverse (backward) propagations

### **For [2]: wrapping effect**

- the hull box of the image evaluation may unfortunately contain zero
- a preconditioning allows to attenuate the wrapping effect

The HC4Revise algorithm

## Solutions for these clustering effects

### For [1]: dependency problem

- centered form provides minimal results with narrow boxes
- not used classically for reverse (backward) propagations

### For [2]: wrapping effect

- the hull box of the image evaluation may unfortunately contain zero
- a preconditioning allows to attenuate the wrapping effect

These solutions come from a recent contribution with proof:

L. Jaulin (2024). Asymptotically minimal interval contractors based on the centered form; Application to the stability analysis of linear time-delayed differential equations, Acta Cybernetica.

## Section 3

# A contractor involving the centered form

A contractor involving the centered form

## Forward evaluations with the "centered form"

The classical formula is given as:

$$[\mathbf{f}_c]([\mathbf{x}]) = \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}})$$

with  $\bar{\mathbf{x}}$  the center of the box  $[\mathbf{x}]$ ,

and  $[\mathbf{J}_f]([\mathbf{x}])$  the interval Jacobian matrix of  $\mathbf{f}$  evaluated over  $[\mathbf{x}]$ .

R. Moore, Methods and Applications of Interval Analysis  
Society for Industrial and Applied Mathematics, jan 1979.

- traditionally used to enclose the range of a function over narrow intervals
- asymptotically small overestimation for sufficiently narrow boxes on scalar functions

A contractor involving the centered form

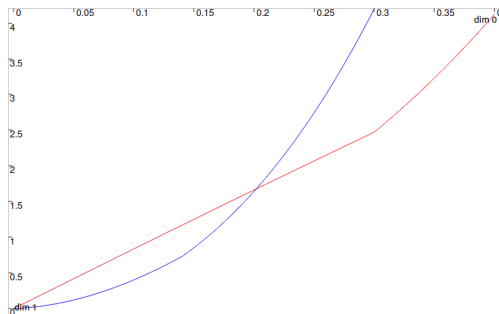
## The dependency problem: example

Consider for instance this expression with multiple occurrences of  $x$ :

$$2x^5 + x^3 - 3x^2$$

Let us try the evaluation for several growing inputs:

$$[x] = 0.7 + [-\epsilon, \epsilon], \epsilon = 0 \dots 0.5$$



Comparing pessimism: natural evaluation (red) vs centered form evaluation (blue).

A contractor involving the centered form

Forward evaluations with the "centered form"

The constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$  is then expressed using a centered form expression:

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

A contractor involving the centered form

## Forward evaluations with the "centered form"

The constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$  is then expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right)$$



A contractor involving the centered form

## Forward evaluations with the "centered form"

The constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$  is then expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right) \cap [\mathbf{f}]([\mathbf{x}])$$

A contractor involving the centered form

## Forward evaluations with the "centered form"

The constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$  is then expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + \mathbf{J}_f([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right) \cap \mathbf{f}([\mathbf{x}])$$

A contractor involving the centered form

## Forward evaluations with the "centered form"

The constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$  is then expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right) \cap [\mathbf{f}]([\mathbf{x}])$$

terms to be computed  
simultaneously during the  
forward DAG evaluation

A contractor involving the centered form

## Forward evaluations with the "centered form"

The constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$  is then expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right) \cap [\mathbf{f}]([\mathbf{x}])$$

terms to be computed  
simultaneously during the  
forward DAG evaluation

Simultaneous evaluation  $\Rightarrow$  operator overloading

A contractor involving the centered form

## Interval Automatic Differentiation (IAD)

### Contribution of this work:

We use IAD to automatically compute the term  $[\mathbf{J}_f]([\mathbf{x}])$ .

A contractor involving the centered form

## Interval Automatic Differentiation (IAD)

### Contribution of this work:

We use IAD to automatically compute the term  $[\mathbf{J}_f]([\mathbf{x}])$ .

### About Automatic Differentiation:

- techniques to automatically evaluate the partial derivatives of a function
- HC4Revise  $\Rightarrow$  execution of a sequence of elementary algorithms:  
    *fwd\_plus, fwd\_cos, etc* (and their reverse counterparts)
- using **chain rule** and interval analysis: compute AD for approximating  $[\mathbf{J}_f]([\mathbf{x}])$

A contractor involving the centered form

## Interval Automatic Differentiation (IAD)

### Contribution of this work:

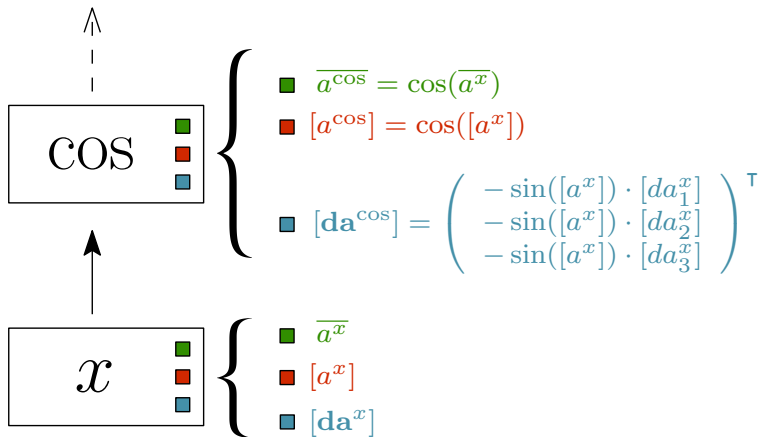
We use IAD to automatically compute the term  $[\mathbf{J}_f]([\mathbf{x}])$ .

### About Automatic Differentiation:

- techniques to automatically evaluate the partial derivatives of a function
  - HC4Revise  $\Rightarrow$  execution of a sequence of elementary algorithms:  
    *fwd\_plus, fwd\_cos, etc* (and their reverse counterparts)
  - using **chain rule** and interval analysis: compute AD for approximating  $[\mathbf{J}_f]([\mathbf{x}])$
- $\Rightarrow$  operator overloading in programming languages

A contractor involving the centered form

## Examples of operator overloading

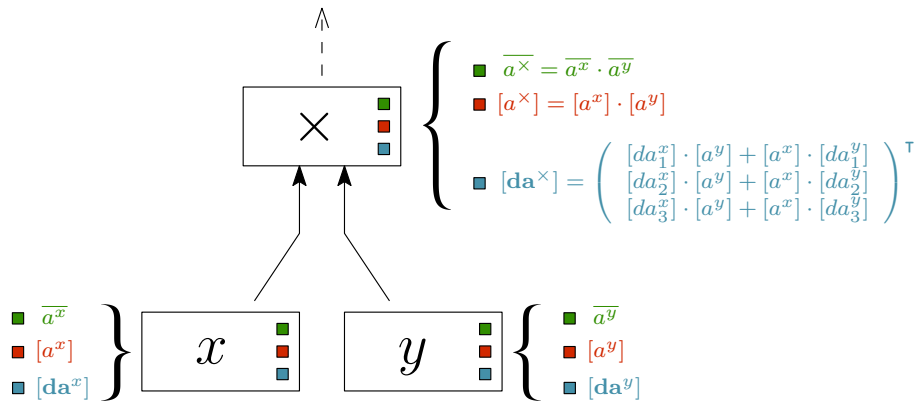


Overloading the  $\cos$  operator with automatic differentiation



A contractor involving the centered form

## Examples of operator overloading



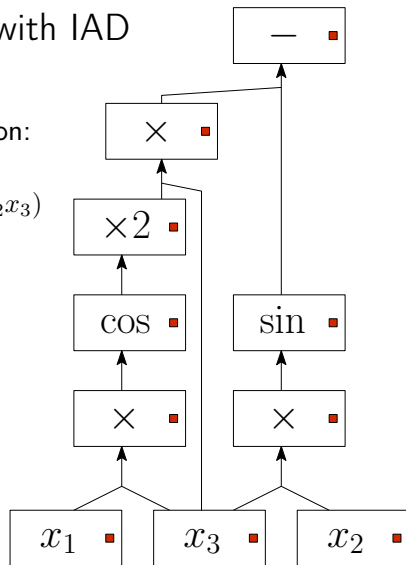
Overloading the product operator with automatic differentiation

A contractor involving the centered form

## Directed acyclic graph, now with IAD

Directed acyclic graph (DAG)  
involving automatic differentiation:

Ex:  $f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$

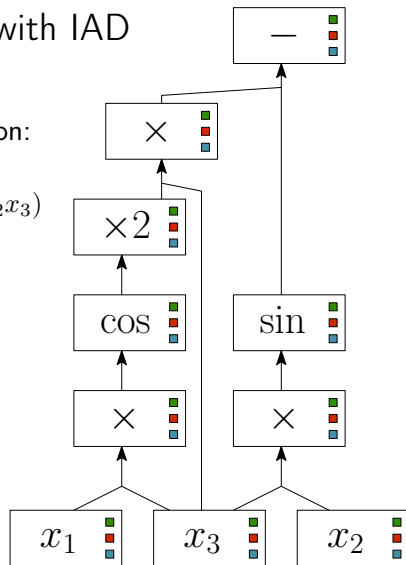


A contractor involving the centered form

## Directed acyclic graph, now with IAD

Directed acyclic graph (DAG)  
involving automatic differentiation:

Ex:  $f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$

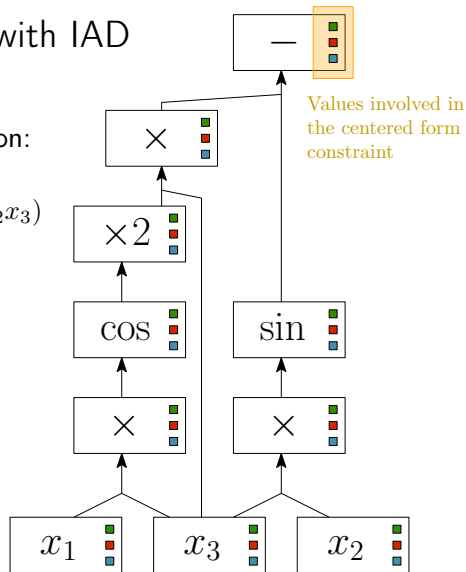


A contractor involving the centered form

## Directed acyclic graph, now with IAD

Directed acyclic graph (DAG)  
involving automatic differentiation:

Ex:  $f(\mathbf{x}) = 2x_3 \cos(x_1x_3) - \sin(x_2x_3)$



A contractor involving the centered form

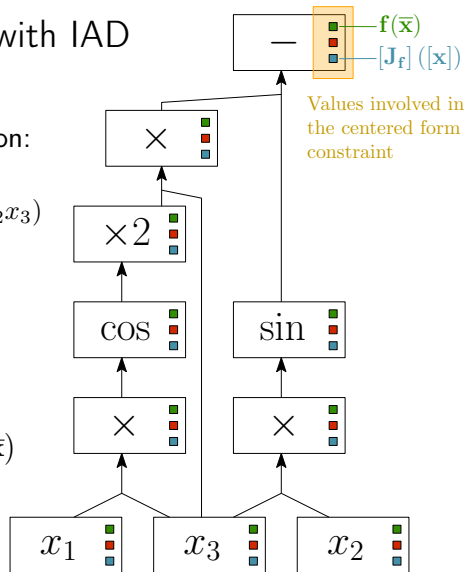
## Directed acyclic graph, now with IAD

Directed acyclic graph (DAG)  
involving automatic differentiation:

$$\text{Ex: } f(\mathbf{x}) = 2x_3 \cos(x_1 x_3) - \sin(x_2 x_3)$$

Towards centered form for  
forward interval evaluation of  $f$

$$f([\mathbf{x}]) \subset \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}})$$



A contractor involving the centered form

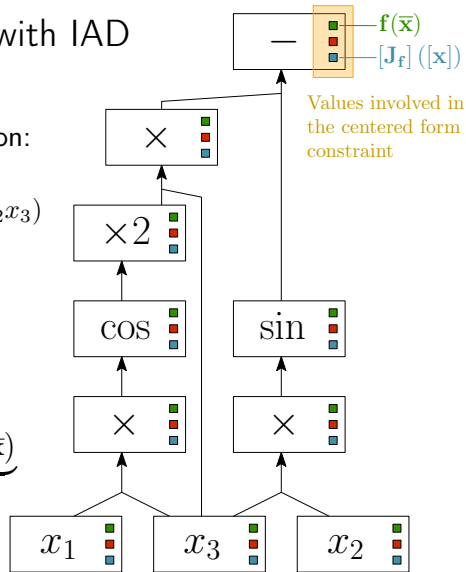
## Directed acyclic graph, now with IAD

Directed acyclic graph (DAG)  
involving automatic differentiation:

$$\text{Ex: } f(\mathbf{x}) = 2x_3 \cos(x_1 x_3) - \sin(x_2 x_3)$$

Towards centered form for  
forward interval evaluation of  $f$

$$f([\mathbf{x}]) \subset \underbrace{f(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}]) \cdot ([\mathbf{x}] - \bar{\mathbf{x}})}_{\text{linear system}}$$



## Section 4

# Top level algorithm for the CtcInverse contractor

## Top level algorithm for the CtcInverse contractor

Recall the constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$ , expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right)$$



## Top level algorithm for the CtcInverse contractor

Recall the constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$ , expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right) \cap [\mathbf{f}]([\mathbf{x}])$$

## Top level algorithm for the CtcInverse contractor

Recall the constraint  $\{\mathbf{f}(\mathbf{x}) \in [\mathbf{y}], \mathbf{x} \in [\mathbf{x}]\}$ , expressed using a centered form expression:

$$\mathbf{f}(\mathbf{x}) \in [\mathbf{y}] \cap \left( \mathbf{f}(\bar{\mathbf{x}}) + [\mathbf{J}_f]([\mathbf{x}])([\mathbf{x}] - \bar{\mathbf{x}}) \right) \cap [\mathbf{f}]([\mathbf{x}])$$

This can be expressed under the form of a linear constraint:

$$\mathbf{b} = \mathbf{A} \cdot \mathbf{p}$$

with

- $\mathbf{b} \in ([\mathbf{y}] \cap [\mathbf{f}]([\mathbf{x}])) - \mathbf{f}(\bar{\mathbf{x}}), \in \mathbb{IR}^p$
- $\mathbf{A} \in [\mathbf{J}_f]([\mathbf{x}]), \in \mathbb{IR}^{p \times n}$
- $\mathbf{p} \in [\mathbf{x}] - \bar{\mathbf{x}}, \in \mathbb{IR}^n$

## Top level algorithm for the CtcInverse contractor

Resulting algorithm for the contractor involving the centered form,  
for dealing with the constraint  $\mathbf{f}(\mathbf{x}) \in [\mathbf{y}]$

$\text{CtcInverse}(in : \mathbf{f}, [\mathbf{y}], in/out : [\mathbf{x}])$

```
 $\bar{\mathbf{x}} \leftarrow \text{mid}([\mathbf{x}])$   
 $(\mathbf{a}^f, [\mathbf{a}^f], [\mathbf{d}\mathbf{a}^f]) \leftarrow \text{fwd\_dag\_eval}(\mathbf{f}, [\mathbf{x}], \bar{\mathbf{x}})$   
 $[\mathbf{p}] \leftarrow [\mathbf{x}] - \bar{\mathbf{x}}$   
 $[\mathbf{b}] \leftarrow ([\mathbf{y}] \cap [\mathbf{a}^f]) - \mathbf{a}^f$   
 $[\mathbf{A}] \leftarrow [\mathbf{d}\mathbf{a}^f]$   
 $([\mathbf{A}], [\mathbf{p}]) \leftarrow \text{reverse\_mul}([\mathbf{b}], [\mathbf{A}], [\mathbf{p}])$   
 $[\mathbf{x}] \leftarrow [\mathbf{p}] + \bar{\mathbf{x}}$ 
```

with the terms  $\mathbf{a}^f$ ,  $[\mathbf{a}^f]$  and  $[\mathbf{d}\mathbf{a}^f]$  computed during the forward evaluation of the DAG.

Top level algorithm for the CtcInverse contractor

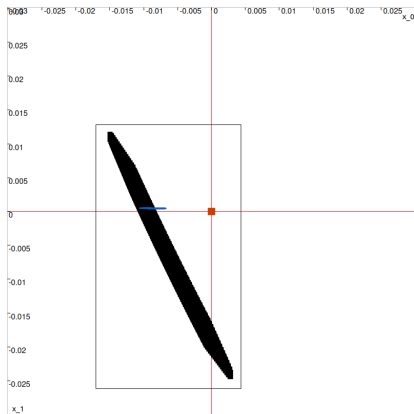
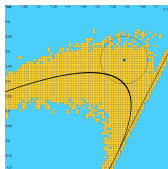
## Dealing efficiently with the linear constraint

About the **reverse matrix-vector product**:

$$([\mathbf{A}], [\mathbf{p}]) \leftarrow \text{reverse\_mul}([\mathbf{b}], [\mathbf{A}], [\mathbf{p}])$$

Some preconditioning allows to significantly reduce wrapping effect.

This can be achieved using a Gauss Jordan band diagonalization method.

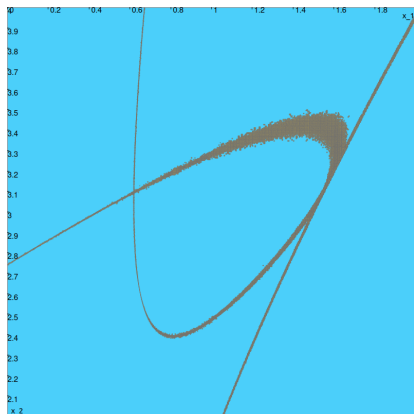


## Section 5

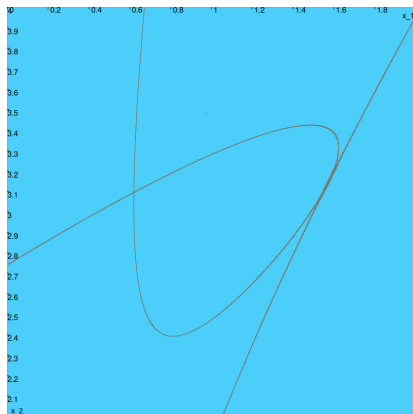
# Results of CtcInverse provided in the Codac library

Results of CtcInverse provided in the Codac library

Previous example using the centered form



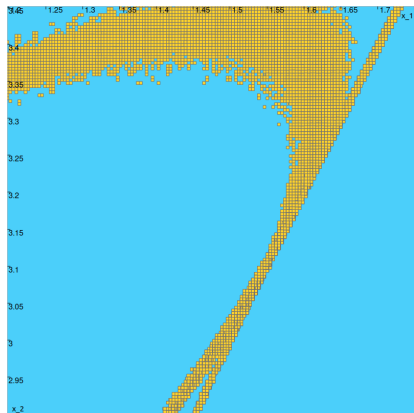
$X$  computed with HC4Revise.  
Computation time: 4.51s. 27430 boxes.



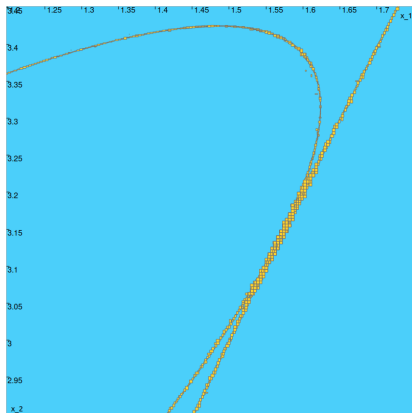
$X$  computed with CtcInverse.  
Computation time: 0.69s. 3713 boxes.

Results of CtcInverse provided in the Codac library

Previous example using the centered form



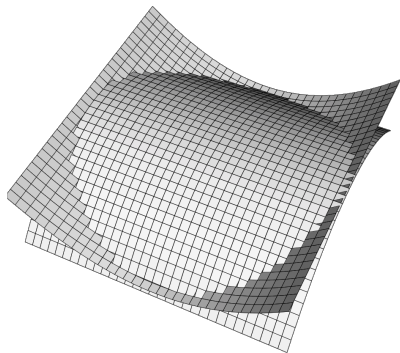
$\times$  computed with HC4Reverse.  
Computation time: 4.51s. 27430 boxes.



$\times$  computed with CtcInverse.  
Computation time: 0.69s. 3713 boxes.

Results of CtcInverse provided in the Codac library

## Comparison with affine arithmetic



**Intersection of two lofted paraboloids:**

L.H. De Figueiredo. Surface Intersection using Affine Arithmetic, Proceedings of Graphics Interface'96, 168-175. 1996

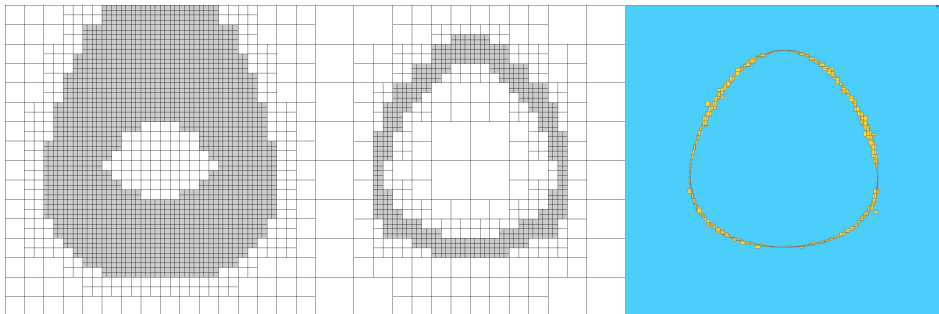
$$\begin{aligned}\alpha(u) &= \mathbf{a}_0(1-u)^2 + 2\mathbf{a}_1u(1-u) + \mathbf{a}_2u^2 \\ \beta(u) &= \mathbf{b}_0(1-u)^2 + 2\mathbf{b}_1u(1-u) + \mathbf{b}_2u^2 \\ \mathbf{f}(u,v) &= (1-v)\alpha(u) + v\beta(u)\end{aligned}$$



Results of CtcInverse provided in the Codac library

## Comparison with affine arithmetic

$$\mathbf{f} : \mathbb{R}^4 \rightarrow \mathbb{R}^3$$



Results projected on the  $(u_1, v_1)$  space.

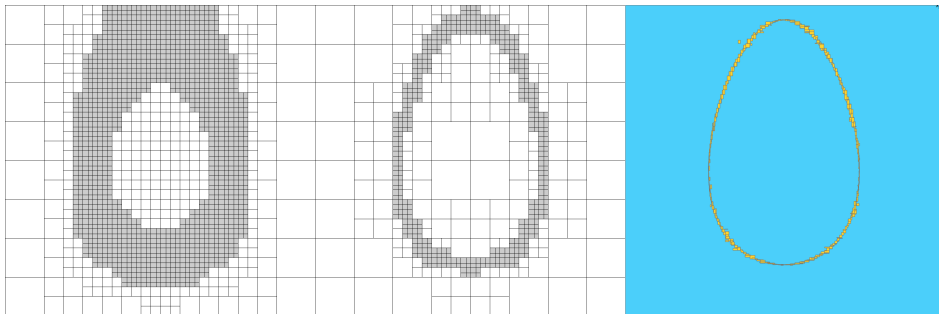
Left: natural evaluation – center: affine evaluation – right: CtcInverse

Using the same  $\epsilon$  in the branch-and-band algorithm.

Results of CtcInverse provided in the Codac library

## Comparison with affine arithmetic

$$\mathbf{f} : \mathbb{R}^4 \rightarrow \mathbb{R}^3$$



Results projected on the  $(u_2, v_2)$  space.

Left: natural evaluation – center: affine evaluation – right: CtcInverse

Using the same  $\epsilon$  in the branch-and-band algorithm.

## Section 6

# Conclusion

Conclusion

## Contribution of this work

- **proposed algorithm involving IAD**
  - operator overloading:
  - implementation simple to maintain or to extend

## Conclusion

# Contribution of this work

- **proposed algorithm involving IAD**
  - operator overloading:
  - implementation simple to maintain or to extend
- **embedded in the contractor framework**
  - simple to use for a non-advised user

## Conclusion

# Contribution of this work

- **proposed algorithm involving IAD**
  - operator overloading:
  - implementation simple to maintain or to extend
- **embedded in the contractor framework**
  - simple to use for a non-advised user
- **code available in C++, Python, Matlab**
  - and Linux, Windows, MacOS..
  - many thanks to Fabrice Le Bars for his help

## Conclusion

# Contribution of this work

- **proposed algorithm involving IAD**
  - operator overloading:
  - implementation simple to maintain or to extend
- **embedded in the contractor framework**
  - simple to use for a non-advised user
- **code available in C++, Python, Matlab**
  - and Linux, Windows, MacOS..
  - many thanks to Fabrice Le Bars for his help
- **comparison of a centered form algorithm with affine arithmetic**

## Conclusion

## Python example (Codac library)



```
from codac import *

x = VectorVar(3)
f = AnalyticFunction([x], vec(
    -sqr(x[2])+2*x[2]*sin(x[2]*x[0])+cos(x[2]*x[1]),
    2*x[2]*cos(x[2]*x[0])-sin(x[2]*x[1])
))

ctc = CtcInverse(f, [[0],[0]])
pave([[0,2],[2,4],[0,10]], ctc, 0.004)
```

---

Using pre-release Codac 2.0.0:

```
pip install codac --pre
```

<http://codac.io>



## Conclusion

## Matlab example (Codac library)



```
import py.codac4matlab.*

x = VectorVar(3);
f = AnalyticFunction({x}, vec( ...
    -sqr(x(3))+2*x(3)*sin(x(3)*x(1))+cos(x(3)*x(2)), ...
    2*x(3)*cos(x(3)*x(1))-sin(x(3)*x(2)) ...
));

ctc = CtcInverse(f, {0,0});
pave(IntervalVector({{0,2},{2,4},{0,10}}), ctc, 0.004);
```

---

Using pre-release Codac 2.0.0 dedicated to Matlab:

```
pip install codac4matlab --pre
```

<http://codac.io>

## Conclusion

## C++ example (Codac library)



```
#include <codac-core.h>

using namespace std;
using namespace codac2;

int main()
{
    VectorVar x(3);
    AnalyticFunction f({x}, vec(
        -sqr(x[2])+2*x[2]*sin(x[2]*x[0])+cos(x[2]*x[1]),
        2*x[2]*cos(x[2]*x[0])-sin(x[2]*x[1])
    ));

    CtcInverse_<IntervalVector> ctc(f, {0.,0.});
    pave(IntervalVector({{0,2},{2,4},{0,10}}), ctc, 0.004);
}
```

<http://codac.io>