

Computer aided proofs in Analysis

Ferenc Bartha

Department of Applied Mathematics
University of Bergen

MaGIC, 2009

Floating-point Numbers

IEEE Standards

- The `single` format consists of 32 bits.
7 significant decimal digits.
- The `double` format consists of 64 bits.
16 significant decimal digits.
- The `INTEL extended` format consists of 80 bits.
19 significant decimal digits.
- The `SPARC extended` format consists of 128 bits.
34 significant decimal digits.

Floating-point Numbers

Drawbacks

- Real Numbers
 - ∞ significant digits.
 - ∞ numbers.
- Floating-point Numbers
 - finite number of significant digits
 - finite representable numbers
- We map real numbers to machine numbers
 - Rounding (to nearest even) \rightarrow Round-off errors
 - Possibility of overflow

Floating-point Numbers

Drawbacks

- Real Numbers
 - ∞ significant digits.
 - ∞ numbers.
- Floating-point Numbers
 - finite number of significant digits
 - finite representable numbers
- We map real numbers to machine numbers
 - Rounding (to nearest even) \rightarrow Round-off errors
 - Possibility of overflow

Floating-point Numbers

Drawbacks

- Real Numbers
 - ∞ significant digits.
 - ∞ numbers.
- Floating-point Numbers
 - finite number of significant digits
 - finite representable numbers
- We map real numbers to machine numbers
 - Rounding (to nearest even) \rightarrow Round-off errors
 - Possibility of overflow

Floating-point Numbers

Drawbacks

- Real Numbers
 - ∞ significant digits.
 - ∞ numbers.
- Floating-point Numbers
 - finite number of significant digits
 - finite representable numbers
- We map real numbers to machine numbers
 - Rounding (to nearest even) \rightarrow Round-off errors
 - Possibility of overflow

Floating-point Numbers

Drawbacks

- Real Numbers
 - ∞ significant digits.
 - ∞ numbers.
- Floating-point Numbers
 - finite number of significant digits
 - finite representable numbers
- We map real numbers to machine numbers
 - Rounding (to nearest even) \rightarrow Round-off errors
 - Possibility of overflow

Notorious Examples

increase accuracy until the result is stabilized

```
restart;  
Digits := 10;  
m := 333.75 * y^6 + x^2 * (11 * x^2 * y^2 - y^6 - 121 * y^4 - 2) + 5.5 * y^8 + x / (2 * y);  
x := 77617;  
y := 33096;  
evalf(m);  
Digits := 18;  
evalf(m);  
Digits := 50;  
evalf(m);
```

Digits := 10

$$m := 333.75 y^6 + x^2 (11 x^2 y^2 - y^6 - 121 y^4 - 2) + 5.5 y^8 + \frac{1}{2} \frac{x}{y}$$

1.172603940

Digits := 18

1.17260394005317863

Digits := 50

-0.8273960599468213681411650954798162919990331157844

Notorious Examples

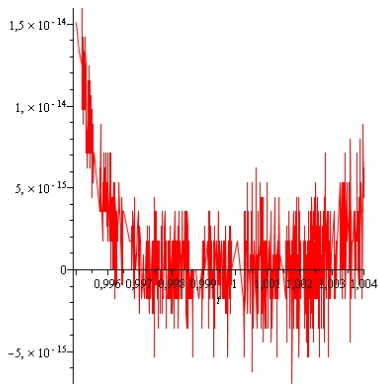
plotting functions

```
restart;
```

```
s := t^6 - 6*t^5 + 15*t^4 - 20*t^3 + 15*t^2 - 6*t + 1;
```

```
plot(s, t=0.995..1.004);
```

$$s = t^6 - 6t^5 + 15t^4 - 20t^3 + 15t^2 - 6t + 1$$



A possible solution: Interval Arithmetics

Instead of rounding, we keep track that the current `exact` result is between which machine numbers.

- this results in counting with intervals instead of numbers
- in order to evaluate functions, we have to implement
 - arithmetic operations over intervals
 - the commonly used functions

A possible solution: Interval Arithmetics

Instead of rounding, we keep track that the current `exact` result is between which machine numbers.

- this results in counting with intervals instead of numbers
- in order to evaluate functions, we have to implement
 - arithmetic operations over intervals
 - the commonly used functions

A possible solution: Interval Arithmetics

Instead of rounding, we keep track that the current `exact` result is between which machine numbers.

- this results in counting with intervals instead of numbers
- in order to evaluate functions, we have to implement
 - arithmetic operations over intervals
 - the commonly used functions

Implementing IA

Example: addition, multiplication

Let $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$ be intervals, \oplus an arithmetic operation. $I_1 \oplus I_2$ is the narrowest interval containing all $i_1 \oplus i_2 : i_1 \in I_1, i_2 \in I_2$.

- $I_1 + I_2 = [\nabla(a_1 + a_2), \Delta(b_1 + b_2)]$
- $I_1 \times I_2 =$
 $[\min\{\nabla(a_1 \times a_2), \nabla(a_1 \times b_2), \nabla(b_1 \times a_2), \nabla(b_1 \times b_2)\},$
 $\max\{\Delta(a_1 \times a_2), \Delta(a_1 \times b_2), \Delta(b_1 \times a_2), \Delta(b_1 \times b_2)\}]$

Implementing IA

Example: addition, multiplication

Let $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$ be intervals, \oplus an arithmetic operation. $I_1 \oplus I_2$ is the narrowest interval containing all $i_1 \oplus i_2 : i_1 \in I_1, i_2 \in I_2$.

- $I_1 + I_2 = [\nabla(a_1 + a_2), \Delta(b_1 + b_2)]$
- $I_1 \times I_2 =$
 $[\min\{\nabla(a_1 \times a_2), \nabla(a_1 \times b_2), \nabla(b_1 \times a_2), \nabla(b_1 \times b_2)\},$
 $\max\{\Delta(a_1 \times a_2), \Delta(a_1 \times b_2), \Delta(b_1 \times a_2), \Delta(b_1 \times b_2)\}]$

Implementing IA

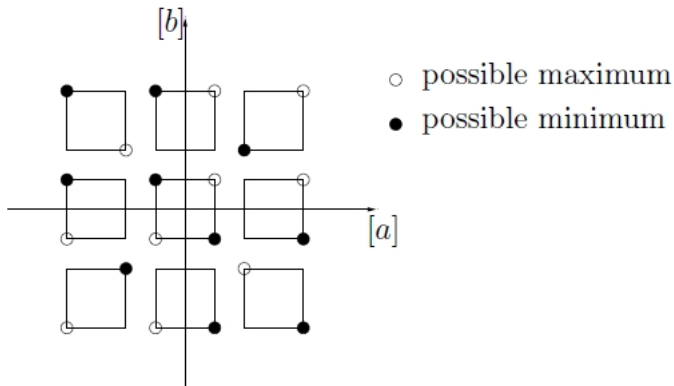
Example: addition, multiplication

Let $I_1 = [a_1, b_1]$, $I_2 = [a_2, b_2]$ be intervals, \oplus an arithmetic operation. $I_1 \oplus I_2$ is the narrowest interval containing all $i_1 \oplus i_2 : i_1 \in I_1, i_2 \in I_2$.

- $I_1 + I_2 = [\nabla(a_1 + a_2), \Delta(b_1 + b_2)]$
- $I_1 \times I_2 =$
 $[\min\{\nabla(a_1 \times a_2), \nabla(a_1 \times b_2), \nabla(b_1 \times a_2), \nabla(b_1 \times b_2)\},$
 $\max\{\Delta(a_1 \times a_2), \Delta(a_1 \times b_2), \Delta(b_1 \times a_2), \Delta(b_1 \times b_2)\}]$

Implementing IA

Advanced multiplication scheme



Evaluating functions

Given a function f that consists of arithmetic operations and IA implemented elementary functions, we get **one** interval extension F by simply replacing each operations and functions with their interval versions.

Given an interval X , we have that for all $x \in X : f(x) \in f(X) \subseteq F(X)$

Evaluating functions

Reducing overestimation

- Many usual convenient laws are violated!
- The form of the function does matter!

For example in general $X^2 \neq X \times X$.

Counting with $X = [-1, 1]$, we get $X^2 = [0, 1]$, while $X \times X = [-1, 1]$.

- Reduce the number of X-s appearing!

Evaluating functions

Reducing overestimation

- Many usual convenient laws are violated!
- The form of the function does matter!

For example in general $X^2 \neq X \times X$.

Counting with $X = [-1, 1]$, we get $X^2 = [0, 1]$, while $X \times X = [-1, 1]$.

- Reduce the number of X-s appearing!

Evaluating functions

Reducing overestimation

- Many usual convenient laws are violated!
- The form of the function does matter!

For example in general $X^2 \neq X \times X$.

Counting with $X = [-1, 1]$, we get $X^2 = [0, 1]$, while $X \times X = [-1, 1]$.

- Reduce the number of X-s appearing!

Rigorous results

Range check

Does $f(x)$ attains $c \in \mathbb{R}$ when $x \in X$?

Certainly not if $c \notin F(X)$.

Rigorous results

Range check

Does $f(x)$ attains $c \in \mathbb{R}$ when $x \in X$?

Certainly not if $c \notin F(X)$.

Rigorous results

Poincaré map

Let $P(x)$ be a Poincaré map for the flow Φ and section X , homeomorphic to a ball (an interval box for example).

- $P(X) \subseteq X$, then Φ has a periodic point in X
- $P(X) \cap X = \emptyset$, then surely such point doesn't exist.

Rigorous results

Poincaré map

Let $P(x)$ be a Poincaré map for the flow Φ and section X , homeomorphic to a ball (an interval box for example).

- $P(X) \subseteq X$, then Φ has a periodic point in X
- $P(X) \cap X = \emptyset$, then surely such point doesn't exist.

What is the value of $f'(x_0)$?

We work with ordered pairs instead of scalars: $\vec{u} = (u, u')$.
Here u means the value of the function evaluated at x_0 ,
 u' means the value of the derivative evaluated at x_0 .

Basic rules:

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$
- $\vec{u} \div \vec{v} = (u/v, (u' - (u/v)v')/v)$

Replace constants with: $\vec{c} = (c, 0)$, variables with $\vec{x} = (x, 1)$.

What is the value of $f'(x_0)$?

We work with ordered pairs instead of scalars: $\vec{u} = (u, u')$.
Here u means the value of the function evaluated at x_0 ,
 u' means the value of the derivative evaluated at x_0 .

Basic rules:

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$
- $\vec{u} \div \vec{v} = (u/v, (u' - (u/v)v')/v)$

Replace constants with: $\vec{c} = (c, 0)$, variables with $\vec{x} = (x, 1)$.

What is the value of $f'(x_0)$?

We work with ordered pairs instead of scalars: $\vec{u} = (u, u')$.
Here u means the value of the function evaluated at x_0 ,
 u' means the value of the derivative evaluated at x_0 .

Basic rules:

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$
- $\vec{u} \div \vec{v} = (u/v, (u' - (u/v)v')/v)$

Replace constants with: $\vec{c} = (c, 0)$, variables with $\vec{x} = (x, 1)$.

What is the value of $f'(x_0)$?

We work with ordered pairs instead of scalars: $\vec{u} = (u, u')$.
Here u means the value of the function evaluated at x_0 ,
 u' means the value of the derivative evaluated at x_0 .

Basic rules:

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$
- $\vec{u} \div \vec{v} = (u/v, (u' - (u/v)v')/v)$

Replace constants with: $\vec{c} = (c, 0)$, variables with $\vec{x} = (x, 1)$.

What is the value of $f'(x_0)$?

We work with ordered pairs instead of scalars: $\vec{u} = (u, u')$.
Here u means the value of the function evaluated at x_0 ,
 u' means the value of the derivative evaluated at x_0 .

Basic rules:

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$
- $\vec{u} \div \vec{v} = (u/v, (u' - (u/v)v')/v)$

Replace constants with: $\vec{c} = (c, 0)$, variables with $\vec{x} = (x, 1)$.

What is the value of $f'(x_0)$?

We work with ordered pairs instead of scalars: $\vec{u} = (u, u')$.
Here u means the value of the function evaluated at x_0 ,
 u' means the value of the derivative evaluated at x_0 .

Basic rules:

- $\vec{u} + \vec{v} = (u + v, u' + v')$
- $\vec{u} - \vec{v} = (u - v, u' - v')$
- $\vec{u} \times \vec{v} = (uv, uv' + u'v)$
- $\vec{u} \div \vec{v} = (u/v, (u' - (u/v)v')/v)$

Replace constants with: $\vec{c} = (c, 0)$, variables with $\vec{x} = (x, 1)$.

What is the value of $f'(x_0)$?

an example

Let $f(x) = \frac{(x+1)(x-2)}{x+3}$. Evaluate $f(3)$ and $f'(3)$!

$$\vec{f}(\vec{x}) = \frac{((x, 1) + (1, 0)) \times ((x, 1) - (2, 0))}{(x, 1) + (3, 0)}$$

$$\begin{aligned}\vec{f}(3, 1) &= \frac{((3, 1) + (1, 0)) \times ((3, 1) - (2, 0))}{(3, 1) + (3, 0)} = \\ &= \frac{(4, 1) \times (1, 1)}{(6, 1)} = \frac{(4, 5)}{(6, 1)} = \left(\frac{2}{3}, \frac{13}{18}\right)\end{aligned}$$

What is the value of $f'(x_0)$?

an example

Let $f(x) = \frac{(x+1)(x-2)}{x+3}$. Evaluate $f(3)$ and $f'(3)$!

$$\vec{f}(\vec{x}) = \frac{((x, 1) + (1, 0)) \times ((x, 1) - (2, 0))}{(x, 1) + (3, 0)}$$

$$\begin{aligned}\vec{f}(3, 1) &= \frac{((3, 1) + (1, 0)) \times ((3, 1) - (2, 0))}{(3, 1) + (3, 0)} = \\ &= \frac{(4, 1) \times (1, 1)}{(6, 1)} = \frac{(4, 5)}{(6, 1)} = \left(\frac{2}{3}, \frac{13}{18}\right)\end{aligned}$$

What is the value of $f'(x_0)$?

an example

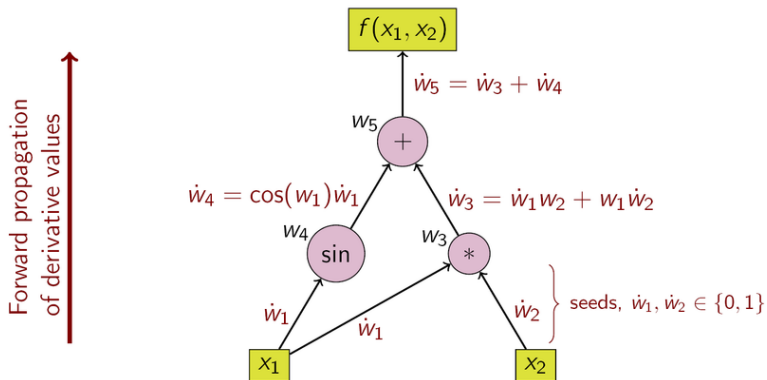
Let $f(x) = \frac{(x+1)(x-2)}{x+3}$. Evaluate $f(3)$ and $f'(3)$!

$$\vec{f}(\vec{x}) = \frac{((x, 1) + (1, 0)) \times ((x, 1) - (2, 0))}{(x, 1) + (3, 0)}$$

$$\begin{aligned}\vec{f}(3, 1) &= \frac{((3, 1) + (1, 0)) \times ((3, 1) - (2, 0))}{(3, 1) + (3, 0)} = \\ &= \frac{(4, 1) \times (1, 1)}{(6, 1)} = \frac{(4, 5)}{(6, 1)} = \left(\frac{2}{3}, \frac{13}{18}\right)\end{aligned}$$

Forward Automatic differentiation

An example: $\sin(x_1) + x_1 x_2$



The Taylor Coefficients

Let $(u_j)_i$ and $(v_j)_i$ be the i th Taylor coefficients of $u(t)$ and $v(t)$ at t_j . Then it is true that

- $(u_j \pm v_j)_i = (u_j)_i \pm (v_j)_i$
- $(u_j v_j)_i = \sum_{r=0}^i (u_j)_r (v_j)_{i-r}$
- $\left(\frac{u_j}{v_j}\right)_i = \frac{1}{v_j} \left\{ (u_j)_i - \sum_{r=1}^i (v_j)_r \left(\frac{u_j}{v_j}\right)_{i-r} \right\}$

Similar formulas can be derived for the standard functions.

The Taylor Coefficients

Let $(u_j)_i$ and $(v_j)_i$ be the i th Taylor coefficients of $u(t)$ and $v(t)$ at t_j . Then it is true that

- $(u_j \pm v_j)_i = (u_j)_i \pm (v_j)_i$
- $(u_j v_j)_i = \sum_{r=0}^i (u_j)_r (v_j)_{i-r}$
- $\left(\frac{u_j}{v_j}\right)_i = \frac{1}{v_j} \left\{ (u_j)_i - \sum_{r=1}^i (v_j)_r \left(\frac{u_j}{v_j}\right)_{i-r} \right\}$

Similar formulas can be derived for the standard functions.

The Taylor Coefficients

Let $(u_j)_i$ and $(v_j)_i$ be the i th Taylor coefficients of $u(t)$ and $v(t)$ at t_j . Then it is true that

- $(u_j \pm v_j)_i = (u_j)_i \pm (v_j)_i$
- $(u_j v_j)_i = \sum_{r=0}^i (u_j)_r (v_j)_{i-r}$
- $\left(\frac{u_j}{v_j}\right)_i = \frac{1}{v_j} \left\{ (u_j)_i - \sum_{r=1}^i (v_j)_r \left(\frac{u_j}{v_j}\right)_{i-r} \right\}$

Similar formulas can be derived for the standard functions.

The Taylor Coefficients

Let $(u_j)_i$ and $(v_j)_i$ be the i th Taylor coefficients of $u(t)$ and $v(t)$ at t_j . Then it is true that

- $(u_j \pm v_j)_i = (u_j)_i \pm (v_j)_i$
- $(u_j v_j)_i = \sum_{r=0}^i (u_j)_r (v_j)_{i-r}$
- $\left(\frac{u_j}{v_j}\right)_i = \frac{1}{v_j} \left\{ (u_j)_i - \sum_{r=1}^i (v_j)_r \left(\frac{u_j}{v_j}\right)_{i-r} \right\}$

Similar formulas can be derived for the standard functions.

The Taylor Coefficients

Let $(u_j)_i$ and $(v_j)_i$ be the i th Taylor coefficients of $u(t)$ and $v(t)$ at t_j . Then it is true that

- $(u_j \pm v_j)_i = (u_j)_i \pm (v_j)_i$
- $(u_j v_j)_i = \sum_{r=0}^i (u_j)_r (v_j)_{i-r}$
- $\left(\frac{u_j}{v_j}\right)_i = \frac{1}{v_j} \left\{ (u_j)_i - \sum_{r=1}^i (v_j)_r \left(\frac{u_j}{v_j}\right)_{i-r} \right\}$

Similar formulas can be derived for the standard functions.

Very simple implementation

multiplication

Here the class `taylor` is an array of `TayN` intervals, representing the appropriate taylor coefficients.

```
taylor operator * (taylor tay1, taylor tay2)
{
    int k,i;
    taylor temp;
    for (i=0; i<TayN; i++) {
        temp.coeff[i].set_values(0);
        for (k=0; k<=i; k++) {
            temp.coeff[i] = temp.coeff[i]+ (tay1.coeff[k] * tay2.coeff[i-k]);
        }
    }
    return temp;
}
```

The Taylor method

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

- We want to integrate this ODE, starting from time t_0 .
- The Taylor method of order N :
$$x(t) = \sum_{i=0}^{N-1} (x_0)_i (t - t_0)^i + r_N(\xi) t^N$$
- where $\xi \in [t_0, t]$

The Taylor method

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

- We want to integrate this ODE, starting from time t_0 .
- The Taylor method of order N :
$$x(t) = \sum_{i=0}^{N-1} (x_0)_i (t - t_0)^i + r_N(\xi) t^N$$
- where $\xi \in [t_0, t]$

No need for symbolic derivation!

For computing $(x_0)_i$ infact we derivate f , and use the ODE to obtain the coefficients. To see this, let's consider the sequence:

- $f^{[1]} = f$
- $f^{[i]} = \frac{1}{i} \left(\frac{\partial f^{[i-1]}}{\partial x} f \right)$

$$(x_j)_i = f^{[i]}(x_j)$$

Notice, that we don't need the formulas for derviations of f !
They can grow extremely fast in complexity, but using automatic differentiation we won't encounter them.

No need for symbolic derivation!

For computing $(x_0)_i$ infact we derivate f , and use the ODE to obtain the coefficients. To see this, let's consider the sequence:

- $f^{[1]} = f$
- $f^{[i]} = \frac{1}{i} \left(\frac{\partial f^{[i-1]}}{\partial x} f \right)$

$$(x_j)_i = f^{[i]}(x_j)$$

Notice, that we don't need the formulas for derviatives of f !
They can grow extremely fast in complexity, but using automatic differentiation we won't encounter them.

No need for symbolic derivation!

For computing $(x_0)_i$ infact we derivate f , and use the ODE to obtain the coefficients. To see this, let's consider the sequence:

- $f^{[1]} = f$
- $f^{[i]} = \frac{1}{i} \left(\frac{\partial f^{[i-1]}}{\partial x} f \right)$

$$(x_j)_i = f^{[i]}(x_j)$$

Notice, that we don't need the formulas for derviatives of f !
They can grow extremely fast in complexity, but using automatic differentiation we won't encounter them.

Combining what we have

Instead of working with numbers, switch to machine represented **intervals!**

$$x(t) = \sum_{i=0}^{N-1} (x)_i (t - t_0)^i + r_N(\xi)(t - t_0)^N$$

transforms into an enclosure:

$$x(t) \in \sum_{i=0}^{N-1} (x)_i (t - t_0)^i + R_N([t_0, t])(t - t_0)^N$$

in the second version the coefficients are evaluated **rigorously!**
We get a polynomial enclosure of x and a remainder.

Combining what we have

Instead of working with numbers, switch to machine represented **intervals!**

$$x(t) = \sum_{i=0}^{N-1} (x)_i (t - t_0)^i + r_N(\xi)(t - t_0)^N$$

transforms into an enclosure:

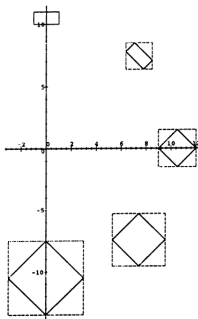
$$x(t) \in \sum_{i=0}^{N-1} (x)_i (t - t_0)^i + R_N([t_0, t])(t - t_0)^N$$

in the second version the coefficients are evaluated **rigorously!**
We get a polynomial enclosure of x and a remainder.

Why can't we be happy

The wrapping effect

As our interval is transformed during one step to a set, to enclose that in an interval, can and usually will produce huge



overestimation.

Change the way of representing sets!

The wrapping effect comes from the behaviour of intervals against transformations.

Possible solutions:

- Instead of intervals lets consider other representable sets and base our arithmetics on them
- Try to store the information of a transformed interval in a simple way

Change the way of representing sets!

The wrapping effect comes from the behaviour of intervals against transformations.

Possible solutions:

- Instead of intervals lets consider other representable sets and base our arithmetics on them
- Try to store the information of a transformed interval in a simple way

Change the way of representing sets!

The wrapping effect comes from the behaviour of intervals against transformations.

Possible solutions:

- Instead of intervals lets consider other representable sets and base our arithmetics on them
- Try to store the information of a transformed interval in a simple way

Different representations for intervals

The GOOD

Parallelepiped

$$[x] = x + [B] \times [r]$$

$[B]$ is an interval matrix, $[r]$ is an interval vector

Different representations for intervals

The BAD

Cuboid

$$[q] = x + [Q] \times [r]$$

special case of parallelepiped where the interval matrix $[Q]$ contains an orthogonal matrix.

Different representations for intervals

The UGLY

Doubleton

$$[x] = x + C \times [r_0] + [s]$$

here C is a representable matrix, $[s]$ is a `small` interval.
Formally it is the sum of a parallelepiped and a small error. And we want to keep it this way.

Enclosing the transformed set

example through doubletons

Let's find an enclosure for $f(x + C \times [r_0] + s)$.

Let $f(x) = g(x) + w(x)$, where g is a good rational approximation, and w is called the error. For example a taylor polynomial, and the remainder would do the job.

Enclosing the transformed set

example through doubletons

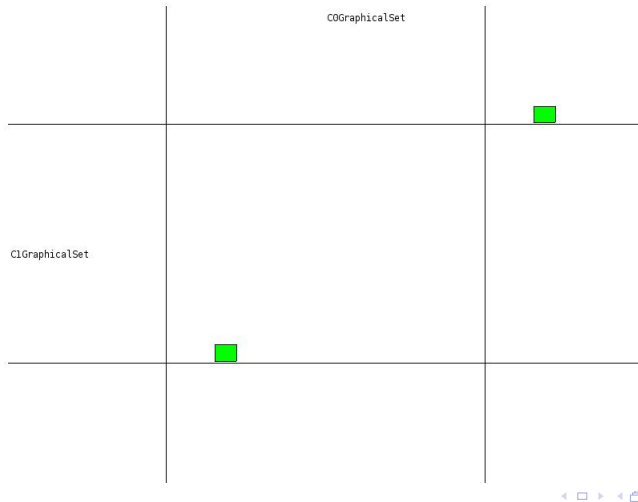
Theorem

$$f(x + C \times [r_0] + s) \subset x_1 + C_1 \times [r_0] + \\ + \text{encl}(J_{[x]}g \times [s] + ([w] + [x_1]^0 + [C_1]^0 \times [r_0]))$$

$J_{[x]}g$ is the Jacobian of g evaluated over $[x]$ and $C_1, [C_1]^0$ is a decomposition of $J_{[x]}g \times C$ and $x_1, [x_1]^0$ is a decomposition of $G([x])$.

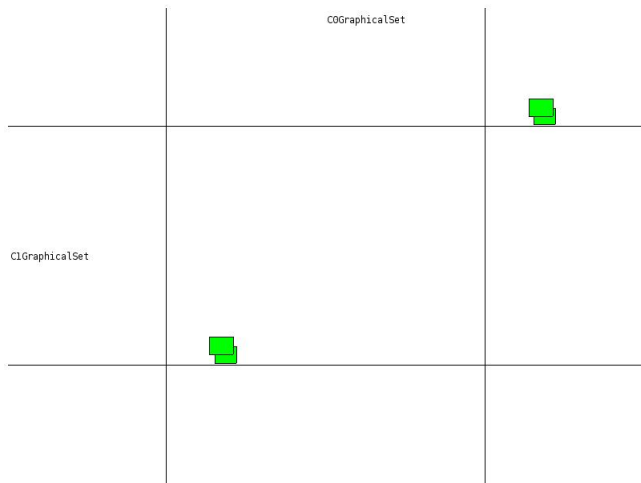
Doubletons in action

The harmonic oscillator



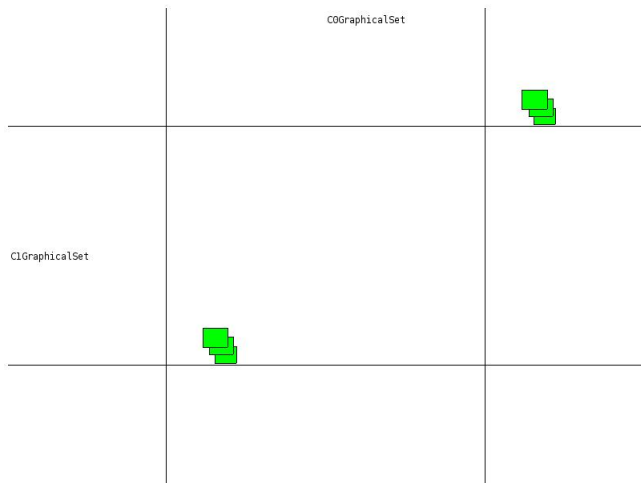
Doubletons in action

The harmonic oscillator



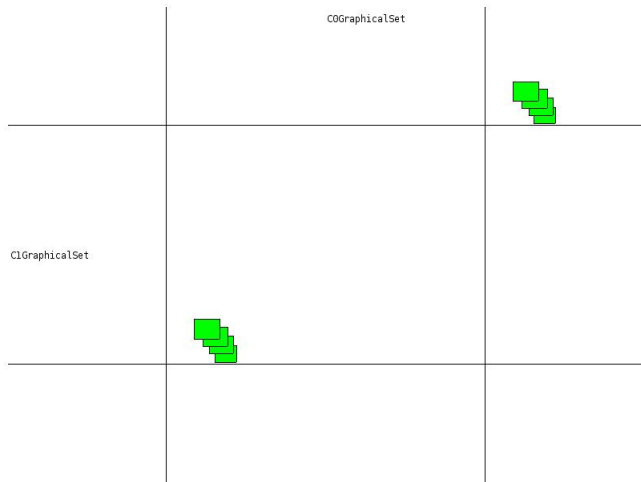
Doubletons in action

The harmonic oscillator



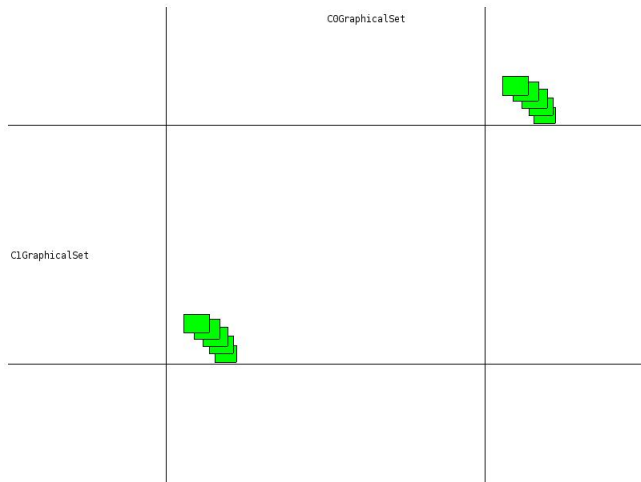
Doubletons in action

The harmonic oscillator



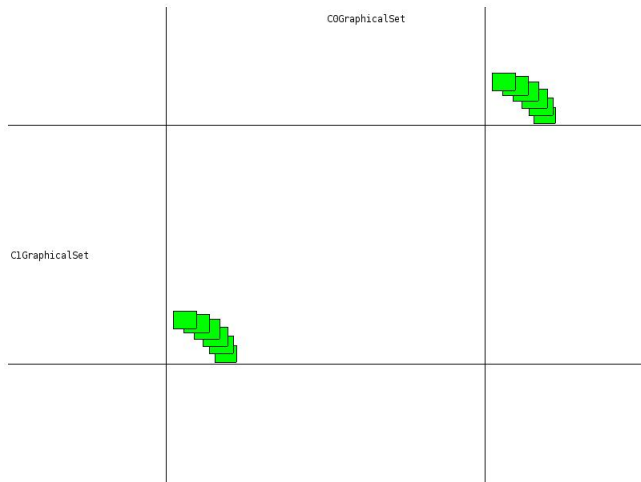
Doubletons in action

The harmonic oscillator



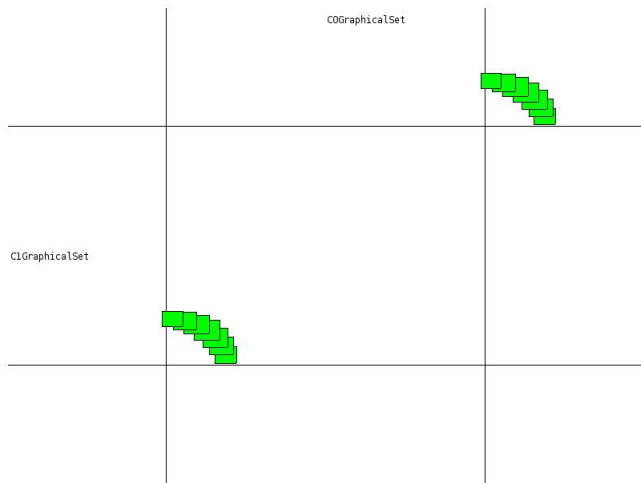
Doubletons in action

The harmonic oscillator



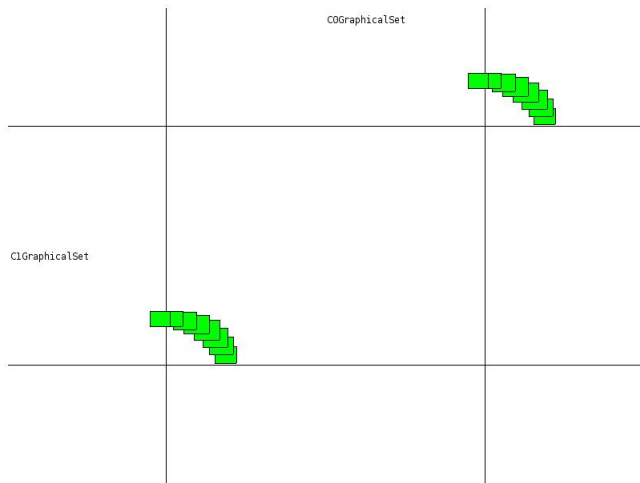
Doubletons in action

The harmonic oscillator



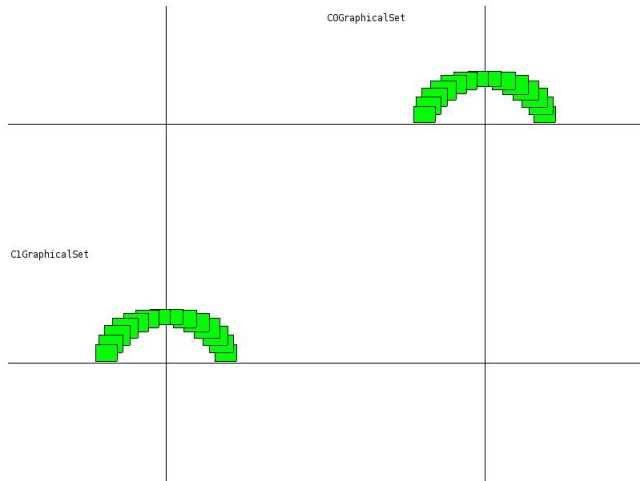
Doubletons in action

The harmonic oscillator



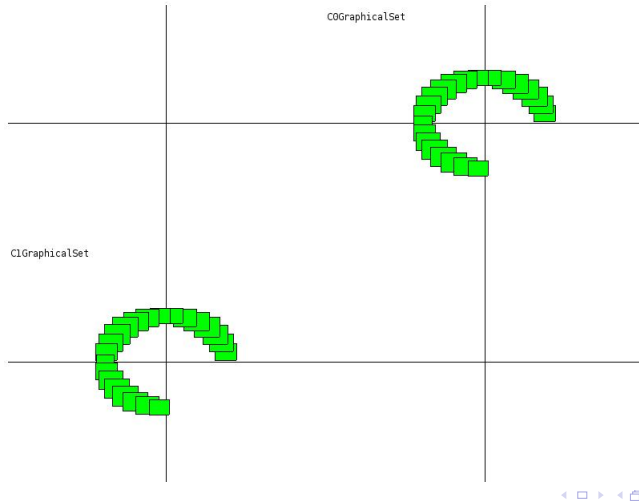
Doubletons in action

The harmonic oscillator



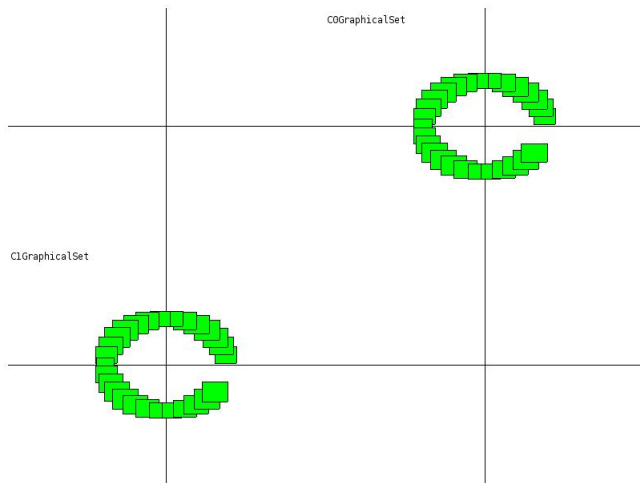
Doubletons in action

The harmonic oscillator



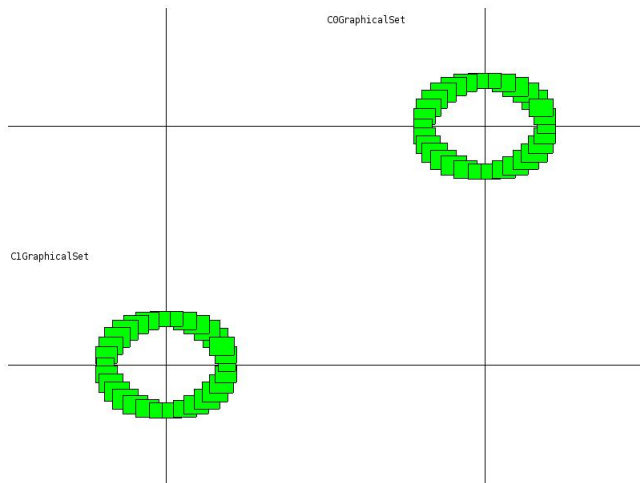
Doubletons in action

The harmonic oscillator



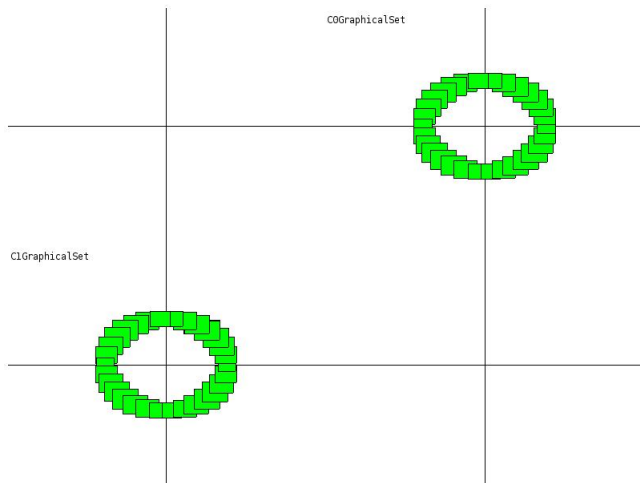
Doubletons in action

The harmonic oscillator



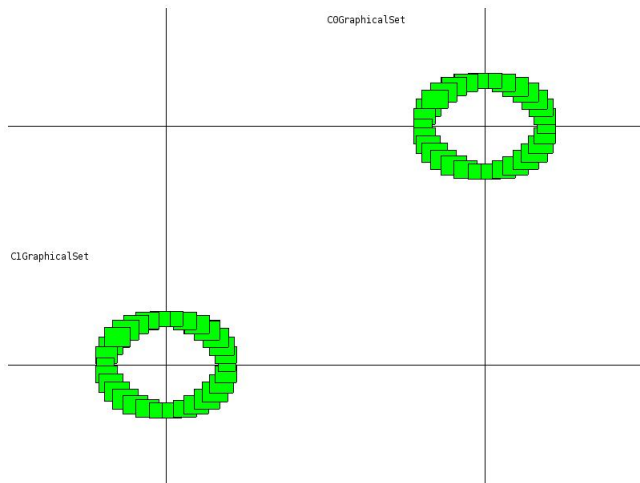
Doubletons in action

The harmonic oscillator



Doubletons in action

The harmonic oscillator



Summary

- Doing our calculation in `interval arithmetics`, **we get rigorous results.**
- Computing `taylor coefficients with automatic differentiation` is fast and requires low resources.
- `Lohner type representations` help to reduce wrapping effect

Integrating ODEs rigorously is possible for larger time intervals, though the war against the wrapping effect goes on!

Summary

- Doing our calculation in `interval arithmetics`, **we get rigorous results.**
- Computing `taylor coefficients` **with automatic differentiation is fast and requires low resources.**
- `Lohner type representations` help to reduce wrapping effect

Integrating ODEs rigorously is possible for larger time intervals, though the war against the wrapping effect goes on!

Summary

- Doing our calculation in `interval arithmetics`, we get rigorous results.
- Computing `taylor coefficients` with `automatic differentiation` is fast and requires low resources.
- `Lohner type representations` help to reduce wrapping effect

Integrating ODEs rigorously is possible for larger time intervals, though the war against the wrapping effect goes on!

Summary

- Doing our calculation in `interval arithmetics`, we get rigorous results.
- Computing taylor coefficients with `automatic differentiation` is fast and requires low resources.
- Lohner type representations help to reduce wrapping effect

Integrating ODEs rigorously is possible for larger time intervals, though the war against the wrapping effect goes on!

Thank You! / Tusen takk!

