Validation for scientific computations
Interval arithmetic

Cours de recherche master informatique
Nathalie Revol
Nathalie.Revol@ens-lyon.fr

26 January 2007
References for today’s lecture

• *Taylor models arith.*: M. Berz and K. Makino, N. Nedialkov, M. Neher.
Historical remarks

Who invented Interval Arithmetic?

- Ramon Moore in 1962 - 1966?
- T. Sunaga in 1958?
- Rosalind Cecil Young in 1931?


Popularization in the 1980, German school (U. Kulisch).

IEEE-754 standard for floating-point arithmetic in 1985: directed roundings are standardized and available (?).

Since the nineties: interval algorithms.
A brief introduction

**Interval arithmetic:** replace numbers by intervals and compute.

**Fundamental theorem of interval arithmetic:**  
(or “Thou shalt not lie”):  
the exact result (number or set) is contained in the computed interval.

No result is lost, the computed interval is guaranteed to contain every possible result.
A brief introduction

Interval Arithmetic and validated scientific computing:

two directions

1. replace floating-point arithmetic by interval arithmetic to bound from above roundoff errors;

2. replace floating-point arithmetic and algorithms by interval ones to compute guaranteed enclosures.
A brief introduction

**Interval arithmetic:** replace numbers by intervals and compute.

Initially: introduced to take into account roundoff errors (Moore 1966) and also uncertainties (on the physical data...). Then: computations “in the large”, computations with sets.

**Interval analysis:** develop algorithms for reliable (or verified, or guaranteed) computing, that are suited for interval arithmetic, i.e. different from the algorithms from classical numerical analysis.
A brief introduction: examples of applications

• control the roundoff errors, cf. computational geometry

• solve several problems with verified solutions: linear and nonlinear systems of equations and inequations, constraints satisfaction, (non/convex, un/constrained) global optimization, integrate ODEs e.g. particules trajectories...

• mathematical proofs: cf. Hales’ proof of the Kepler’s conjecture

Agenda

- Definitions of interval arithmetic (operations, function extensions)
- Cons (overestimation, complexity) and pros (contractant iterations: Brouwer’s theorem)
- Some algorithms
  - solving linear systems
  - Newton
  - global optimization wo/with constraints
  - constraints programming
- Variants: affine arithmetic, Taylor models arithmetic
Agenda

• Definitions of interval arithmetic (operations, function extensions)
• Cons (overestimation, complexity)
  and pros (contractant iterations: Brouwer’s theorem)
• Some algorithms
  – solving linear systems
  – Newton
  – global optimization wo/with constraints
  – constraints programming
• Variants: affine arithmetic, Taylor models arithmetic
Definitions: intervals

Objects:

• intervals of real numbers = closed connected sets of \( \mathbb{R} \)
  
  - interval for \( \pi \): \([3.14159, 3.14160]\)
  
  - data \( d \) measured with an absolute error less than \( \pm \varepsilon \): \([d - \varepsilon, d + \varepsilon]\)

• interval vector: components = intervals; also called box

• interval matrix: components = intervals.
Definitions: operations

\[ x \diamond y = \text{Hull}\{x \diamond y : x \in x, y \in y\} \]

**Arithmetic and algebraic operations:** use the monotony

\[
\begin{align*}
[x, \bar{x}] + [y, \bar{y}] &= [x + y, \bar{x} + \bar{y}] \\
[x, \bar{x}] - [y, \bar{y}] &= [x - \bar{y}, \bar{x} - y] \\
[x, \bar{x}] \times [y, \bar{y}] &= [\min(x \times y, \bar{x} \times \bar{y}, \bar{x} \times y, \bar{x} \times \bar{y}), \max(\text{ibid.})] \\
[x, \bar{x}]^2 &= [\min(x^2, \bar{x}^2), \max(x^2, \bar{x}^2)] \text{ if } 0 \not\in [x, \bar{x}] \\
&\quad [0, \max(x^2, \bar{x}^2)] \text{ otherwise} \\
1/ [y, \bar{y}] &= [\min(1/y, 1/\bar{y}), \max(1/y, 1/\bar{y})] \text{ if } 0 \not\in [y, \bar{y}] \\
[x, \bar{x}] / [y, \bar{y}] &= [x, \bar{x}] \times (1/ [y, \bar{y}]) \text{ if } 0 \not\in [y, \bar{y}] \\
\sqrt{[x, \bar{x}]} &= [\sqrt{\bar{x}}, \sqrt{x}] \text{ if } 0 \leq x, [0, \sqrt{x}] \text{ otherwise}
\end{align*}
\]
Definitions: operations

**Algebraic properties:** associativity, commutativity hold, some are lost:

- subtraction is not the inverse of addition, in particular $x - x \neq [0]$
- division is not the inverse of multiplication
- squaring is tighter than multiplication by oneself
- multiplication is only sub-distributive wrt addition
Definitions: functions

Definition:
an interval extension $f$ of a function $f$ satisfies

$$\forall x, f(x) \subset f(x), \text{ and } \forall x, f\{x\} = f\{x\}.$$ 

Elementary functions: again, use the monotony.

$$\exp x = [\exp \underline{x}, \exp \overline{x}]$$
$$\log x = [\log \underline{x}, \log \overline{x}] \text{ if } x \geq 0, [-\infty, \log \overline{x}] \text{ if } \overline{x} > 0$$
$$\sin[\pi/6, 2\pi/3] = [1/2, 1]$$

...
Definitions: function extension

Example: \( f(x) = x^2 - x + 1 \) with \( x \in [-2, 1] \).

\([-2, 1]^2 - [-2, 1] + 1 = [0, 4] + [-1, 2] + 1 = [0, 7] \).

Since \( x^2 - x + 1 = x(x - 1) + 1 \), we get \([-2, 1] \cdot ([-2, 1] - 1) + 1 = [-2, 1] \cdot [-3, 0] + 1 = [-3, 6] + 1 = [-2, 7] \).

Since \( x^2 - x + 1 = (x - 1/2)^2 + 3/4 \), we get \(([-2, 1] - 1/2)^2 + 3/4 = [-5/2, 1/2]^2 + 3/4 = [0, 25/4] + 3/4 = [3/4, 7] = f([-2, 1]) \).

Problem with this definition: infinitely many interval extensions, syntactic use (instead of semantic).

How to choose the best extension? How to choose a good one?
Definitions: function extension

Mean value theorem of order 1 (Taylor expansion of order 1):
\[ \forall x, \forall y, \exists \xi_{x,y} \in (x, y) : f(y) = f(x) + (y - x) \cdot f'(\xi_{x,y}) \]
Interval interpretation:
\[ \forall y \in x, \forall \tilde{x} \in x, f(y) \in f(\tilde{x}) + (y - \tilde{x}) \cdot f'(x) \]
\[ \Rightarrow f(x) \subset f(\tilde{x}) + (x - \tilde{x}) \cdot f'(x) \]

Mean value theorem of order 2 (Taylor expansion of order 2):
\[ \forall x, \forall y, \exists \xi_{x,y} \in (x, y) : f(y) = f(x) + (y - x) \cdot f'(x) + \frac{(y-x)^2}{2} \cdot f''(\xi_{x,y}) \]
Interval interpretation:
\[ \forall y \in x, \forall \tilde{x} \in x, f(y) \in f(\tilde{x}) + (y - \tilde{x}) \cdot f'(\tilde{x}) + \frac{(y-\tilde{x})^2}{2} \cdot f''(x) \]
\[ \Rightarrow f(x) \subset f(\tilde{x}) + (x - \tilde{x}) \cdot f'(\tilde{x}) + \frac{(x-\tilde{x})^2}{2} \cdot f''(x) \]
Definitions: function extension

No need to go further:

- it is difficult to compute (automatically) the derivatives of higher order, especially for multivariate functions;
- there is no (theoretical) gain in quality.

Theorem:

- for the natural extension $f$ of $f$, it holds $d(f(x), f(x)) \leq O(w(x))$
- for the first order Taylor extension $f_{T_1}$ of $f$, it holds $d(f(x), f_{T_1}(x)) \leq O(w(x)^2)$
- getting an order higher than 3 is impossible without the squaring operation, is difficult even with it...
Agenda

- Definitions of interval arithmetic (operations, function extensions)
- Cons (overestimation, complexity)
  and pros (contractant iterations: Brouwer’s theorem)
- Some algorithms
  - solving linear systems
  - Newton
  - global optimization wo/with constraints
  - constraints programming
- Variants: affine arithmetic, Taylor models arithmetic
Cons: overestimation (1/2)

The result encloses the true result, but it is too large: overestimation phenomenon. Two main sources: variable dependency and wrapping effect.

(Loss of) Variable dependency:

\[ x - x = \{x - y : x \in x, y \in x\} \neq \{x - x : x \in x\} = \{0\}. \]
Cons: overestimation (2/2)

image of $f(x)$ with $f : \mathbb{R}^2 \rightarrow \mathbb{R}^2$

2 successives rotations of $\pi/4$
of the little central square
Cons: Complexity: almost every problem is NP-hard


- evaluate a function on a box (cartesian product of intervals)
- evaluate a function on a box up to $\varepsilon$
- solve a linear system
- solve a linear system up to $1/4n^4$ ($n = \text{dim. of the system}$)
- determine if the solution of a linear system is bounded
- compute the matrix norm $\|A\|_{\infty,1}$
- determine if an interval matrix (a matrix with interval coefficients) is regular, i.e. if every possible punctual matrix in it is regular
  
  . . .
**Cons: Complexity: Gaganov 1982**

evaluation of a multivariate polynomial with rational coeff. on a box is NP-hard

**Idea: reduce polynomially the CNF-3 problem to this problem.**

On $n$ boolean variables $q_1, \cdots, q_n$, a formula $f$ in CNF-3 is defined by

$$f = \bigwedge_{i=1}^m f_i \text{ with } f_i = \bigvee_{j=1}^{1,2or3} r_{i,j}$$

with $r_{i,j} = q_{k_{i,j}}$ or $r_{i,j} = \neg q_{k_{i,j}}$.

1. to each boolean variable $q_i$, let us associate a real variable $x_i \in [0, 1]$. Meaning: $x_i = 0$ if $q_i = F$ and $x_i = 1$ if $q_i = T$. 

* Goal: get a polynomial which takes only values in $[0, 1]$
i.e. allow only product of terms or sums of the form ($1 -$ term).
A product corresponds to a conjunction and $1 - x$ to a negation
⇒ express $f$ and the $f_i$ using conjunctions and negations
⇒ express the $f_i$ as $\neg \bigwedge_{j=1}^{1,2or3} \neg r_{i,j}$.

2. to each $r_{i,j}$ let us associate a polynomial $y_{i,j}$ (corresponding to the
negation of $r_{i,j}$) defined by

$$r_{i,j} = q_{k_{i,j}} \rightarrow y_{i,j}(x) = 1 - x_{k_{i,j}}$$
$$r_{i,j} = \neg q_{k_{i,j}} \rightarrow y_{i,j}(x) = x_{k_{i,j}}$$

3. to each $f_i$, let us associate a polynomial $p_i$ (corresponding to the
negation of $f_i$) defined by $f_i = \bigwedge r_{i,j} \rightarrow p_i(x) = \prod y_{i,j}(x)$.
4. to $f$, let us associate the polynomial $p$ defined by $f = \bigwedge_{i=1}^{m} f_i \rightarrow$

$p(x) = \prod_{i=1}^{m} (1 - p_i(x))$. 
Cons: Complexity: Gaganov 1982

evaluation of a multivariate polynomial with rational coeff. on a box is NP-hard

Lemma:

1. $\forall x \in [0, 1], \ p(x) \in [0, 1]$.
2. if $\alpha$ is a boolean vector and $\beta$ is the associated $0 - 1$ vector, then

   \[
   f(\alpha) = T \Rightarrow p(\beta) = 1
   \]

   \[
   f(\alpha) = F \Rightarrow p(\beta) = 0.
   \]

3. if $f$ is not feasible, then $\forall x \in [0, 1]^n, \ p(x) \leq 7/8$. 
Proof of (3): (proving (1) and (2) is easy).
\( \forall x \in [0,1]^n, \) let us consider \( \beta \) the 0-1 vector obtained by rounding \( x \) to the nearest.
Since \( f \) is not feasible, \( p(\beta) = 0. \)
Since \( p(x) = \prod_{i=1}^{m}(1 - p_i(x)), \exists i_0 \) such that \( 1 - p_{i_0}(\beta) = 0. \)
One can prove that \( p_{i_0}(x) \geq 1/8, \) using the fact that it is the product of at most three terms, each of them \( \leq 1/2, \) using the fact that \( \beta \) is the rounding to nearest of \( x. \) Thus \( 1 - p_{i_0}(x) \leq 7/8. \)
The remaining factors \( 1 - p_j(x) \) are less or equal to 1.
Thus \( p(x) = \prod_{i=1}^{m}(1 - p_i(x)) \leq 7/8. \)

**Consequence:** since checking the feasibility of a CNF-3 formula is NP-hard, evaluating a multivariate polynomial (up to a small \( \varepsilon \)) is NP-hard.
**Pros: set computing**

**Behaviour safe? controllable? dangerous?**

always controllable.

On $x$, are the extrema of the function $f > f^1, < f_2$?

No if $f(x) = [\_f, f] \subset [f_2, f^1]$. 

Validation in scientific computing - Nathalie Revol 25 26-01-2007
Pros: Brouwer-Schauder theorem

A function $f$ which is continuous on the unit ball $B$ and which satisfies $f(B) \subset B$ has a fixed point on $B$.
Furthermore, if $f(B) \subset \text{int}B$ then $f$ has a unique fixed point on $B$.

The theorem remains valid if $B$ is replaced by a box $K$. 
Agenda

• Definitions of interval arithmetic (operations, function extensions)
• Cons (overestimation, complexity)
  and pros (contractant iterations: Brouwer’s theorem)
• Some algorithms
  – solving linear systems
  – Newton
  – global optimization wo/with constraints
  – constraints programming
• Variants: affine arithmetic, Taylor models arithmetic
Algorithm: linear systems solving (Hansen-Sengupta)

Problem: solve $Ax = b$ or equivalently:

$$A_{i,1}x_1 + \ldots + A_{i,i}x_i + \ldots + A_{i,n}x_n = b_i \text{ for } 1 \leq i \leq n$$

Determine Hull ($\sum_{\exists\exists}(A, b)$) = Hull ($\{x : \exists A \in A, \exists b \in b, Ax = b\}$).

Pre-processing: multiply the system by an approximate $\midd(A)^{-1}$. New system = $\midd(A)^{-1}Ax = b$. Hope: contracting iteration.

Algorithm: apply Gauss-Seidel iteration

while convergence not reached loop

for $i = 1$ to $n$ do

$$x_i := \left( b_i - \sum_{j \neq i} A_{i,j}x_j \right) / A_{i,i}$$
Algorithm: solving a nonlinear system: Newton
Why a specific iteration for interval computations?

Usual formula:

\[ x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \]

Direct interval transposition:

\[ x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \]

\[ w(x_{k+1}) = w(x_k) + w\left(\frac{f(x_k)}{f'(x_k)}\right) > w(x_k) \]

divergence!
Algorithm: interval Newton principle of an iteration

(Hansen & Greenberg 83, Baker Kearfott 95-97, Mayer 95, van Hentenryck et al. 97)

\[ x_{1} := \left( x - \frac{F(x)}{F'(x)} \right) \cap x \]
Algorithm: interval Newton principle of an iteration

\[
(x_1, x_2) := \left( x - \frac{F\{x\}}{F'(x)} \right) \cap \mathbb{x}
\]
Algorithm: interval Newton

Input: \( F, F', x_0 \)  // \( x_0 \) initial search interval

Initialization: \( L = \{x_0\}, \alpha = 0.75 \)  //any value in \( ]0.5, 1[ \) is suitable

Loop: while \( L \neq \emptyset \)

Suppress \((x, L)\)

\( x := \text{mid}(x) \)

\( (x_1, x_2) := \left( x - \frac{F(x)}{F'(x)} \right) \cap x \)  // \( x_1 \) and \( x_2 \) can be empty

if \( w(x_1) > \alpha w(x) \) or \( w(x_2) > \alpha w(x) \) then \( (x_1, x_2) := \text{bisect}(x) \)

if \( x_1 \neq \emptyset \) and \( F(x_1) \ni 0 \) then

if \( w(x_1)/|\text{mid}(x_1)| \leq \varepsilon_x \) or \( w(F(x_1)) \leq \varepsilon_Y \) then Insert \( x_1 \) in \( \text{Res} \)

else Insert \( x_1 \) in \( L \)

same handling of \( x_2 \)

Output: \( \text{Res} \), a list of intervals that may contain the roots.
Algorithm: interval Newton

Existence and uniqueness of a root are proven:
if there is no hole and if the new iterate (before \( \bigcap \)) is contained in the interior of the previous one.

Existence of a root is proven:

• using the mean value theorem:
  OK if \( f(\inf(x)) \) and \( f(\sup(x)) \) have opposite signs.
  (Miranda theorem in higher dimensions).
• using Schauder theorem: if the new iterate (before \( \bigcap \)) in contained in the previous one.
Algorithm: optimize a continuous function

Problem: \( f : \mathbb{R}^n \to \mathbb{R} \), determine \( x^* \) and \( f^* \) that verify

\[
 f^* = f(x^*) = \min_x f(x) 
\]

Assumptions:

- search within a box \( x_0 \)
- \( x^* \in \) in the interior of \( (x_0) \), not at the boundary
- \( f \) continuous enough: \( C^2 \)
Algorithm: optimize a continuous function

(Ratschek and Rokne 1988, Hansen 1992, Kearfott 1996. . . )

Goal: determine the minimum of $f$, continuous function on a box $x_0$.

$x_0$ current box
$ar{f}$ current upper bound of $f^*$

while there is a box in the waiting list
  if $f(x) > \bar{f}$ then
    reject $x$
  otherwise
    update $\bar{f}$: if $f(mid(x)) < \bar{f}$ then $\bar{f} = f(mid(x))$
bisect $x$ into $x_1$ and $x_2$
examine $x_1$ and $x_2$
Algorithm: optimize a continuous function
the rejection procedure

$F(X1)$

$X1 \ X2 \ X3$

$f$ non convexe sur $X3$

$f$ trop haute : $F(X1) > \overline{f}$

$0$ n’est pas dans $G(X2)$

$f$ non convexe sur $X3$
Algorithm: optimize a continuous function
the reduction procedure
Algorithm: optimize a continuous function

Hansen algorithm  

\[ \mathcal{L} = \text{list of not yet examined boxes} := \{ x_0 \} \]

while \( \mathcal{L} \neq \emptyset \) loop

remove \( x \) from \( \mathcal{L} \)

reject \( x \)?
  yes if \( f(x) > \bar{f} \)
  yes if \( \text{Grad} f(x) \neq 0 \)
  yes if \( Hf(x) \) has its diagonal non \( > 0 \)

reduce \( x \)
  Newton applied to the gradient
  solve \( y \subset x \) such that \( f(y) \leq \bar{f} \)

bisect \( y \): insert the resulting \( y_1 \) and \( y_2 \) in \( \mathcal{L} \)
Algorithm: constrained optimization

Problem: \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) and \( c : \mathbb{R}^n \rightarrow \mathbb{R}^m \), determine \( x^* \) and \( f^* \) that verify

\[
 f^* = f(x^*) = \min_{\{x | c(x) \leq 0\}} f(x)
\]

Assumptions:

- search within a box \( x_0 \)
- \( f \) continuous enough: \( C^2 \)
- \( c \) continuous enough: \( C^1 \)
Algorithm: constrained optimization $c(x) \leq 0$

the rejection procedure
Algorithm: constrained optimization $c(x) \leq 0$

the reduction procedure
Algorithm: constrained optimization \( c(x) \leq 0 \)

\[
\mathcal{L} := \{ x_0 \}
\]

while \( \mathcal{L} \neq \emptyset \) loop

remove \( x \) from \( \mathcal{L} \)

reject \( x \)?

yes if \( f(x) > \bar{f} \)

yes if \( \text{Grad} f(x) \neq 0 \)

yes if \( f \) not convex on \( x \)

reduce \( x \)

solve \( y \subset x | f(y) \leq \bar{f} \)

Newton applied to the gradient

bisect \( y \) into \( y_1 \) and \( y_2 \)

insert \( y_1 \) and \( y_2 \) in \( \mathcal{L} \)

\[
\mathcal{L} := \{ x_0 \}
\]

while \( \mathcal{L} \neq \emptyset \) loop

remove \( x \) from \( \mathcal{L} \)

reject \( x \)?

yes if \( f(x) > \bar{f} \)

yes if \( c(x) > 0 \)

reduce \( x \)

'solve \( y \subset x \) such that \( c(y) \leq 0 \)

Newton applied to the Lagrangian

bisect \( y \) into \( y_1 \) and \( y_2 \)

insert \( y_1 \) and \( y_2 \) in \( \mathcal{L} \)
Algorithm: constraints programming

Cleary 1987, Benhamou et al. 1999, Jaulin et al. 2001

Problem:
\[
\begin{align*}
    c_1(x_1, \ldots, x_n) &= 0 \\
    \vdots \\
    c_p(x_1, \ldots, x_n) &= 0
\end{align*}
\]

expressed as:

\[
\begin{align*}
    y_i &= x_i \quad \text{for } 1 \leq i \leq n \\
    y_k &= y_i \diamond y_j \quad \text{for } n + 1 \leq k \leq m \text{ and } i, j < k \\
    y_k \text{ auxiliary variable}
\end{align*}
\]

where \( y_k = \varphi(y_i) \) \( \text{for } n + 1 \leq k \leq m \text{ and } i < k \)
Algorithm: constraints programming

Initializations:  \( y_1 := x_1, \ldots, y_n := x_n \)

Propagation: forward mode
for \( k = n + 1 \) to \( m \) loop
\[ y_k := y_i \odot y_j \text{ or } y_k := \varphi(y_i) \]

Propagation: backward mode
for \( k = m \) to \( n \) loop
if \( y_k \) is defined as \( y_i \odot y_j \) then
\[ y_i := (y_k \odot^{-r} y_j) \cap y_i \]
\[ y_j := (y_i \odot^{-l} y_k) \cap y_j \]
else if \( y_k \) is defined as \( \varphi(y_i) \) then
\[ y_i := \varphi^{-1}(y_k) \cap y_i \]
Algorithm: constraints programming:

\[ \begin{align*}
  x_1 x_2^2 - 2x_3 &= 0 \\
  \cos x_1 + x_3 &= 0
\end{align*} \]
\( x_1 = [0, 2\pi/3], \ x_2 = [-1, 1], \ x_3 = [-1/2, 3] \)

**iter. 1 : forward**

\[
\begin{align*}
y_4 &= y_2^2 \\
y_5 &= y_1 y_4 \\
y_6 &= 2y_3 \\
y_7 &= y_5 - y_6 \\
y_8 &= \cos y_1 \\
y_9 &= y_8 + y_3
\end{align*}
\]

**backward**

\[
\begin{align*}
y_9 &= y_8 + y_3 \\
y_8 &= \cos y_1 \\
y_7 &= y_5 - y_6 \\
y_6 &= 2y_3 \\
y_5 &= y_1 y_4 \\
y_4 &= y_2^2
\end{align*}
\]

\[
\begin{align*}
y_1 &= [0, 2\pi/3], \ y_2 = [-1, 1], \ y_3 = [-1/2, 3] \\
y_4 &= [0, 1] \\
y_5 &= [0, 2\pi/3] \\
y_6 &= [-1, 6] \\
y_7 &= [-6, 1 + 2\pi/3] \ni 0 \\
y_8 &= [-1/2, 1] \\
y_9 &= [-1, 4] \ni 0 \\
\left\{ \begin{align*}
y_8 &= (y_9 - y_3) \cap y_8 = [-1/2, 1/2] \\
y_3 &= (y_9 - y_8) \cap y_3 = [-1/2, 1/2]
\end{align*} \right\}
\end{align*}
\]

\[
\begin{align*}
y_1 &= \cos^{-1} y_8 \cap y_1 = [\pi/3, 2\pi/3] \\
y_5 &= (y_7 + y_6) \cap y_5 = [0, 2\pi/3] \\
y_6 &= (y_5 - y_7) \cap y_6 = [0, 2\pi/3] \\
y_3 &= (1/2y_6) \cap y_3 = [0, 1/2] \\
y_1 &= (y_5/y_4) \cap y_1 = [\pi/3, 2\pi/3] \\
y_4 &= (y_5/y_1) \cap y_4 = [0, 1] \\
y_2 &= \pm \sqrt{y_4} \cap y_2 = [-1, 1]
\end{align*}
\]
Algorithm: constraints programming: \[
\begin{align*}
    x_1 x_2^2 - 2x_3 &= 0 \\
    \cos x_1 + x_3 &= 0
\end{align*}
\]
\[x_1 = [0, \frac{2\pi}{3}], \ x_2 = [-1, 1], \ x_3 = [-\frac{1}{2}, 3]\]

**iter. 2: forward**

\[y_4 = y_2^2\]
\[y_5 = y_1 y_4\]
\[y_6 = 2y_3\]
\[y_8 = \cos y_1\]

**backward**

\[y_9 = y_8 + y_3\]
\[y_8 = \cos y_1\]
\[y_7 = y_5 - y_6\]
\[y_6 = 2y_3\]
\[y_5 = y_1 y_4\]
\[y_4 = y_2^2\]

\[y_1 = [\frac{\pi}{3}, \frac{2\pi}{3}], \ y_2 = [-1, 1], \ y_3 = [0, \frac{1}{2}]\]
\[y_4 = [0, 1], \ y_5 = [0, 2\pi/3], \ y_6 = [0, 1]\]
\[y_7 = 0, \ y_8 = [-1/2, 1/2], \ y_9 = 0\]
\[y_4 = [0, 1]\]
\[y_5 = [0, 2\pi/3]\]
\[y_6 = [0, 1]\]
\[y_8 = [-1/2, 1/2]\]

\[\{\ y_8 = (y_9 - y_3) \cap y_8 = [-1/2, 0]\]
\[\ y_3 = (y_9 - y_8) \cap y_3 = [0, 1/2]\]
\[y_1 = \cos^{-1} y_8 \cap y_1 = [\pi/2, 2\pi/3]\]

\[\{\ y_5 = (y_7 + y_6) \cap y_5 = [0, 1]\]
\[\ y_6 = (y_5 - y_7) \cap y_6 = [0, 1]\]
\[y_3 = (1/2 y_6) \cap y_3 = [0, 1/2]\]

\[\{\ y_1 = (y_5/y_4) \cap y_1 = [\pi/2, 2\pi/3]\]
\[\ y_4 = (y_5/y_1) \cap y_4 = [0, 2/\pi]\]
\[y_2 = \pm \sqrt{y_4} \cap y_2 = [-\sqrt{2/\pi}, \sqrt{2/\pi}]\]
\[y_1 = [\frac{\pi}{2}, \frac{2\pi}{3}], \ y_2 = [-\sqrt{2/\pi}, \sqrt{2/\pi}], \ y_3 = [0, \frac{1}{2}]\]
Problem: \[
\begin{align*}
    x_1 x_2^2 - 2x_3 &= 0 \\
    \cos x_1 + x_3 &= 0
\end{align*}
\]

with \( x_1 = [0, \frac{2\pi}{3}] \), \( x_2 = [-1, 1] \), \( x_3 = [-\frac{1}{2}, 3] \).

Optimal solution obtained after two iterations:
\( x_1 = [\frac{\pi}{2}, \frac{2\pi}{3}] \), \( x_2 = [-\sqrt{\frac{2}{\pi}}, \sqrt{\frac{2}{\pi}}] \), \( x_3 = [0, \frac{1}{2}] \).
Agenda

- Definitions of interval arithmetic (operations, function extensions)
- Cons (overestimation, complexity) and pros (contractant iterations: Brouwer’s theorem)
- Some algorithms
  - solving linear systems
  - Newton
  - global optimization wo/with constraints
  - constraints programming
- Variants: affine arithmetic, Taylor models arithmetic
Conclusions

Interval algorithms

• can solve problems that other techniques are not able to solve
• is a simple version of set computing
• give effective versions of theorems which did not seem to be effective (Brouwer)
• can determine all zeros or all extrema of a continuous function
• overestimate the result
• is less efficient than floating-point arithmetic (theoretical factor: 4, practical factor: 20)
  \Rightarrow\text{ solve “small” problems.}
Philosophical conclusion

Morale

- forget one’s biases:
  - do not use without thinking algorithms which are supposed to be good ones (Newton)
  - do not reject without thinking algorithm which are supposed to be bad ones (Gauss-Seidel)
- prefer contracting iterations whenever possible