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

- R. Moore: *Interval Analysis*, Prentice Hall, Englewood Cliffs, 1966.
- A. Neumaier: *Interval methods for systems of equations*, CUP, 1990.
- E. Hansen and W. Walster: *Global optimization using interval analysis*, MIT Press, 2004.
- R.B. Kearfott: *Rigorous global search: continuous problems*, Kluwer, 1996.
- V. Kreinovich, A. Lakeyev, J. Rohn, P. Kahl: *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Dordrecht, 1997.
- L.H. Figueiredo, J. Stolfi: *Affine arithmetic* http://www.ic.unicamp.br/~stolfi/EXPORT/projects/affine-arith/.
- *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 ?

Cf. `http://www.cs.utep.edu/interval-comp/`, click on *Early papers*.

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

Cf. `http://www.cs.utep.edu/interval-comp/`

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

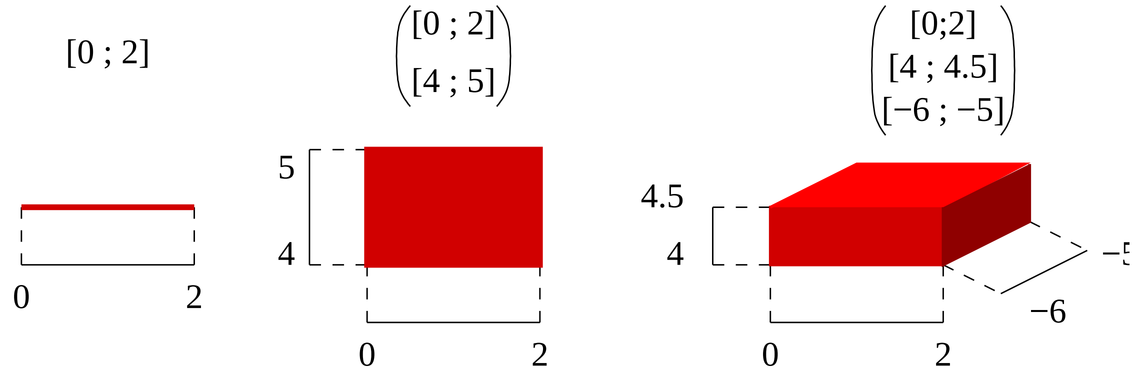
Agenda

- 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 π: $[3.14159, 3.14160]$
  - 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

$\boldsymbol{x} \diamond \boldsymbol{y} =$ **Hull**$\{x \diamond y \; : \; x \in \boldsymbol{x}, y \in \boldsymbol{y}\}$

Arithmetic and algebraic operations: use the monotony

$$
\begin{array}{rcl}
[\underline{x}, \overline{x}] + [\underline{y}, \overline{y}] & = & [\underline{x} + \underline{y}, \overline{x} + \overline{y}] \\
[\underline{x}, \overline{x}] - [\underline{y}, \overline{y}] & = & [\underline{x} - \overline{y}, \overline{x} - \underline{y}] \\
[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] & = & [\min(\underline{x} \times \underline{y}, \underline{x} \times \overline{y}, \overline{x} \times \underline{y}, \overline{x} \times \overline{y}), \max(\text{ibid.})] \\
[\underline{x}, \overline{x}]^2 & = & [\min(\underline{x}^2, \overline{x}^2), \max(\underline{x}^2, \overline{x}^2)] \text{ if } 0 \notin [\underline{x}, \overline{x}] \\
 & & [0, \max(\underline{x}^2, \overline{x}^2)] \text{ otherwise} \\
1/[\underline{y}, \overline{y}] & = & [\min(1/\underline{y}, 1/\overline{y}), \max(1/\underline{y}, 1/\overline{y})] \text{ if } 0 \notin [\underline{y}, \overline{y}] \\
[\underline{x}, \overline{x}] / [\underline{y}, \overline{y}] & = & [\underline{x}, \overline{x}] \times (1/[\underline{y}, \overline{y}]) \text{ if } 0 \notin [\underline{y}, \overline{y}] \\
\sqrt{[\underline{x}, \overline{x}]} & = & [\sqrt{\underline{x}}, \sqrt{\overline{x}}] \text{ if } 0 \leq \underline{x}, [0, \sqrt{\overline{x}}] \text{ otherwise}
\end{array}
$$

Definitions: operations

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

- subtraction is not the inverse of addition, in particular $\boldsymbol{x} - \boldsymbol{x} \neq [0]$
- 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 $\boldsymbol{f}$ of a function f satisfies

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

Elementary functions: again, use the monotony.

$$\begin{array}{lcl}
\exp \boldsymbol{x} & = & [\exp \underline{x}, \exp \overline{x}] \\
\log \boldsymbol{x} & = & [\log \underline{x}, \log \overline{x}] \text{ if } \underline{x} \geq 0, [-\infty, \log \overline{x}] \text{ if } \overline{x} > 0 \\
\sin[\pi/6, 2\pi/3] & = & [1/2, 1] \\
\ldots & &
\end{array}$$

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:

$\forall y \in \boldsymbol{x}, \forall \tilde{x} \in \boldsymbol{x},\; f(y) \in f(\tilde{x}) + (y-\tilde{x})\cdot \boldsymbol{f}'(\boldsymbol{x})$

$\Rightarrow f(\boldsymbol{x}) \subset f(\tilde{x}) + (\boldsymbol{x}-\tilde{x})\cdot \boldsymbol{f}'(\boldsymbol{x})$

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:

$\forall y \in \boldsymbol{x}, \forall \tilde{x} \in \boldsymbol{x},\; f(y) \in f(\tilde{x}) + (y-\tilde{x})\cdot f'(\tilde{x}) + \frac{(y-\tilde{x})^2}{2}\cdot \boldsymbol{f}''(\boldsymbol{x})$

$\Rightarrow f(\boldsymbol{x}) \subset f(\tilde{x}) + (\boldsymbol{x}-\tilde{x})\cdot f'(\tilde{x}) + \frac{(\boldsymbol{x}-\tilde{x})^2}{2}\cdot \boldsymbol{f}''(\boldsymbol{x})$

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 $\boldsymbol{f}$ of f, it holds $d(f(\boldsymbol{x}), \boldsymbol{f}(\boldsymbol{x})) \leq \mathcal{O}(w(\boldsymbol{x}))$
- for the first order Taylor extension $\boldsymbol{f_{T_1}}$ of f, it holds $d(f(\boldsymbol{x}), \boldsymbol{f_{T_1}}(\boldsymbol{x})) \leq \mathcal{O}(w(\boldsymbol{x})^2)$
- 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

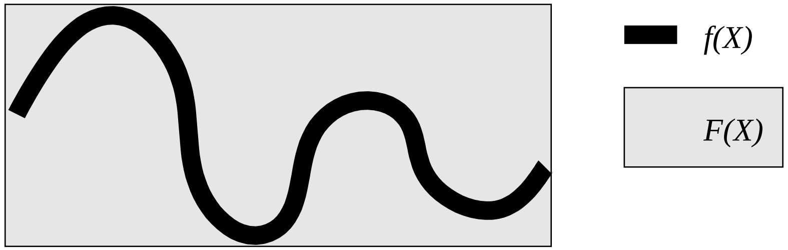
Cons: overestimation (1/2)

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:

$$\boldsymbol{x} - \boldsymbol{x} = \{x - y \; : \; x \in \boldsymbol{x}, y \in \boldsymbol{x}\} \neq \{x - x \; : \; x \in \boldsymbol{x}\} = \{0\}.$$

Cons: overestimation (2/2)



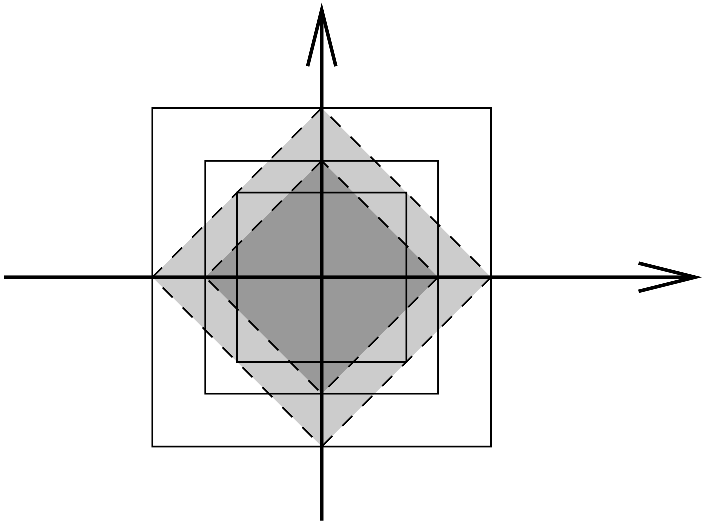


image of $f(\boldsymbol{x})$
with $f\ :\ \mathbb{R}^2 \longrightarrow \mathbb{R}^2$

2 successives rotations of $\pi/4$
of the little central square

Cons: Complexity: almost every problem is NP-hard

Gaganov 1982, Rohn 1994 ff, Kreinovich. . .

- evaluate a function on a box (cartesian product of intervals)
- evaluate a function on a box up to ε
- solve a linear system
- solve a linear system up to $1/4n^4$ ($n =$ dim. of the system)
- determine if the solution of a linear system is bounded
- compute the matrix norm $\|\boldsymbol{A}\|_{\infty,1}$
- 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

$$f = \bigwedge_{i=1}^{m} f_i \text{ with } f_i = \bigvee_{j=1}^{1,2 or 3} r_{i,j}$$

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
  $\Rightarrow$ express f and the f_i using conjonctions and negations
  $\Rightarrow$ 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
$$\begin{array}{rclcrcl} r_{i,j} & = & q_{k_{i,j}} & \rightarrow & y_{i,j}(x) & = & 1 - x_{k_{i,j}} \\ r_{i,j} & = & \neg q_{k_{i,j}} & \rightarrow & y_{i,j}(x) & = & x_{k_{i,j}} \end{array}$$

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 α is a boolean vector and β is the associated $0-1$ vector, then

$$\begin{array}{rclcrcl} f(\alpha) & = & T & \Rightarrow & p(\beta) & = & 1 \\ f(\alpha) & = & F & \Rightarrow & p(\beta) & = & 0. \end{array}$$

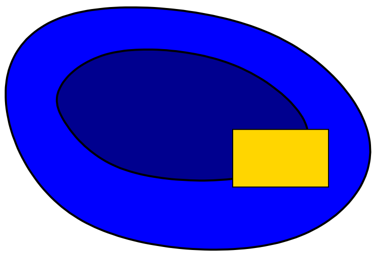
3. if f is not feasible, then $\forall x \in [0,1]^n$, $p(x) \leq 7/8$.

Proof of (3): (proving (1) and (2) is easy).
$\forall x \in [0,1]^n$, let us consider β 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 β 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 ε) is NP-hard.

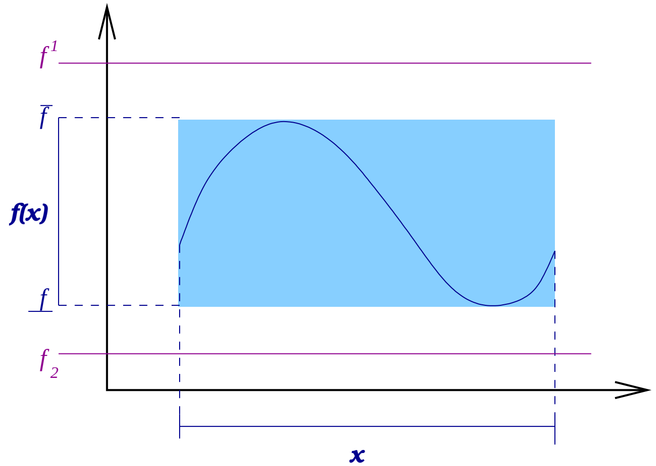
Pros: set computing

Behaviour safe? controllable? dangerous?



always controllable.

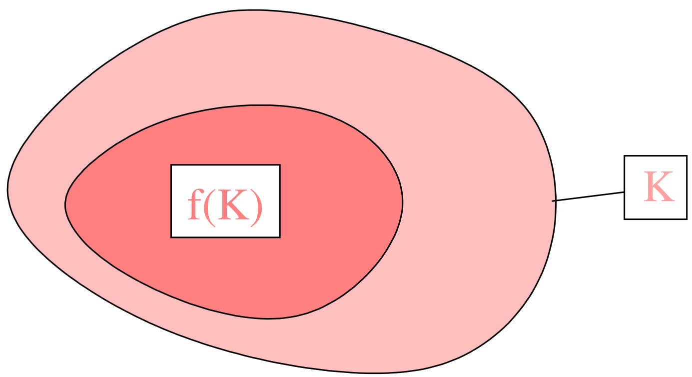
On $\boldsymbol{x}$, are the extrema of the function f $> f^1$, $< f_2$?



No if $f(\boldsymbol{x}) = [\underline{f}, \overline{f}] \subset [f_2, f^1]$.

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 $f(B) \subset \mathrm{int}B$ 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 $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ or equivalently:

$$\boldsymbol{A}_{i,1}\boldsymbol{x}_1 + \ldots + \boldsymbol{A}_{i,i}\boldsymbol{x}_i + \ldots + \boldsymbol{A}_{i,n}\boldsymbol{x}_n = \boldsymbol{b}_i \text{ for } 1 \leq i \leq n$$

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

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

Algorithm: apply Gauss-Seidel iteration

while convergence not reached loop

for $i = 1$ to n do

$$\boldsymbol{x}_i := \left(\boldsymbol{b}_i - \sum_{j \neq i} \boldsymbol{A}_{i,j}\boldsymbol{x}_j\right) / \boldsymbol{A}_{i,i}$$

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:

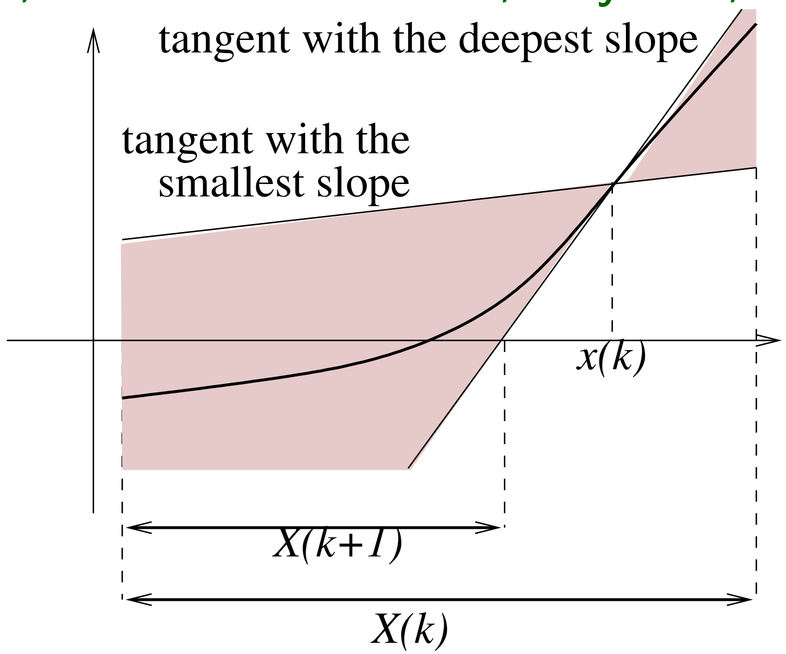
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{f(\boldsymbol{x}_k)}{f'(\boldsymbol{x}_k)}$$

$$w(\boldsymbol{x}_{k+1}) = w(\boldsymbol{x}_k) + w\left(\frac{f(\boldsymbol{x}_k)}{f'(\boldsymbol{x}_k)}\right) > w(\boldsymbol{x}_k)$$

divergence!

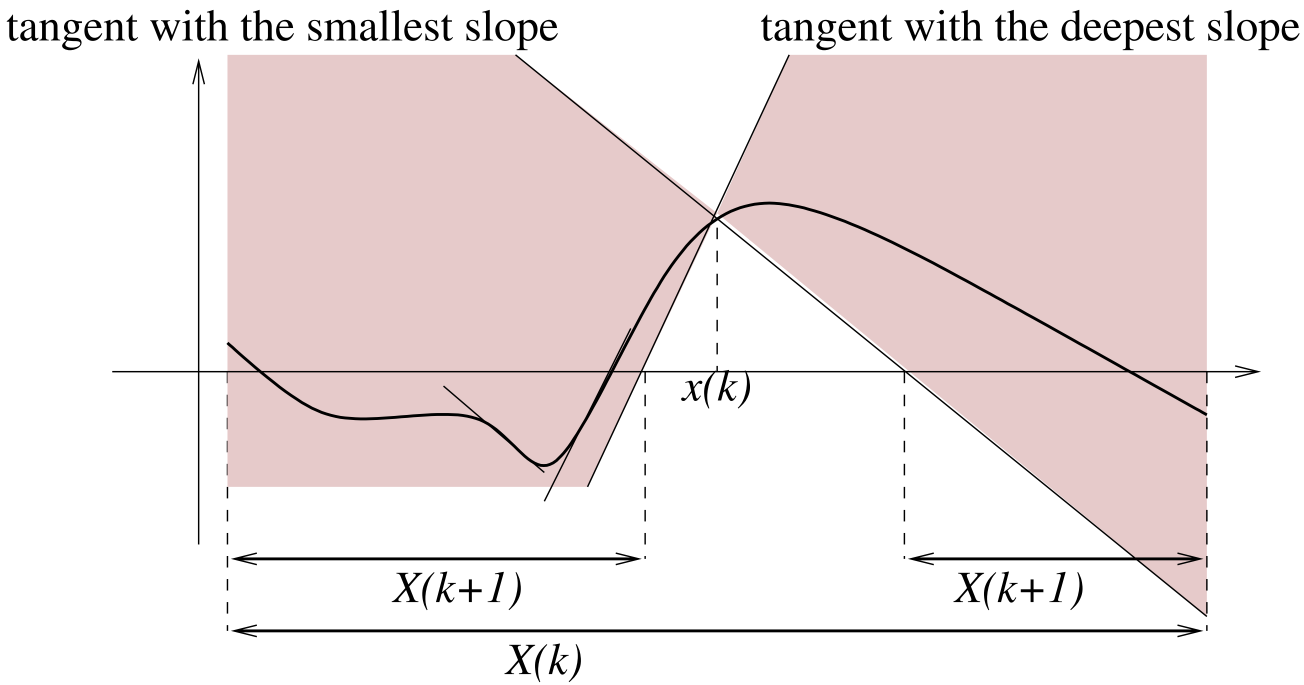
Algorithm: interval Newton
principle of an iteration

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



$$\boldsymbol{x}_1 := \left(x - \frac{F(\{x\})}{F'(\boldsymbol{x})} \right) \bigcap \boldsymbol{x}$$

Algorithm: interval Newton principle of an iteration



$$(\boldsymbol{x}_1, \boldsymbol{x}_2) := \left(x - \frac{F(\{x\})}{F'(\boldsymbol{x})} \right) \bigcap \boldsymbol{x}$$

Algorithm: interval Newton

Input: F, F', $\boldsymbol{x}_0$ // $\boldsymbol{x}_0$ initial search interval

Initialization: $\mathcal{L} = \{\boldsymbol{x}_0\}$, $\alpha = 0.75$ //any value in $]0.5, 1[$ is suitable

Loop: while $\mathcal{L} \neq \emptyset$

Suppress $(\boldsymbol{x}, \mathcal{L})$

$x :=$ mid$(\boldsymbol{x})$

$(\boldsymbol{x}_1, \boldsymbol{x}_2) := \left(x - \frac{F(\{x\})}{F'(\boldsymbol{x})}\right) \bigcap \boldsymbol{x}$ // $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ can be empty

if $w(\boldsymbol{x}_1) > \alpha w(\boldsymbol{x})$ or $w(\boldsymbol{x}_2) > \alpha w(\boldsymbol{x})$ then $(\boldsymbol{x}_1, \boldsymbol{x}_2) :=$ bisect$(\boldsymbol{x})$

if $\boldsymbol{x}_1 \neq \emptyset$ and $F(\boldsymbol{x}_1) \ni 0$ then

if $w(\boldsymbol{x}_1)/|\text{mid}(\boldsymbol{x}_1)| \leq \varepsilon_{\boldsymbol{x}}$ or $w(F(\boldsymbol{x}_1)) \leq \varepsilon_Y$ then Insert $\boldsymbol{x}_1$ in Res

else Insert $\boldsymbol{x}_1$ in $\mathcal{L}$

same handling of $\boldsymbol{x}_2$

Output: Res, a list of intervals that may contain the roots.

Algorithm: interval Newton

properties

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 $f(\inf(\boldsymbol{x}))$ and $f(\sup(\boldsymbol{x}))$ have opposite signs.
  (Miranda theorem in higher dimensions).
- using Schauder theorem: if the new iterate (before $\bigcap$) in contained in the previous one.

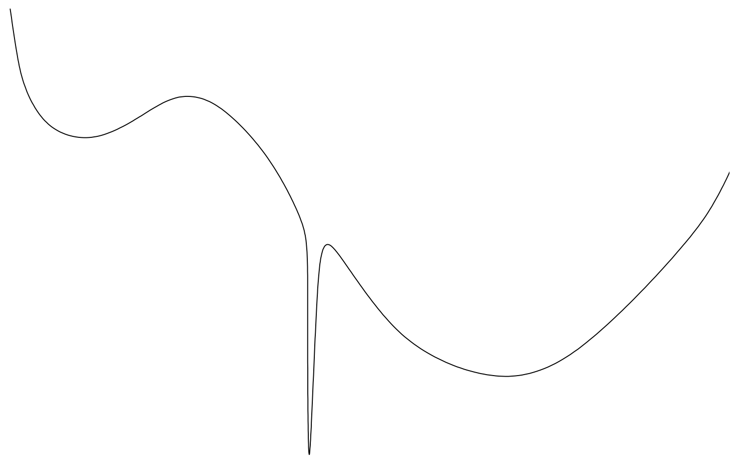
Algorithm: optimize a continuous function

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 $\boldsymbol{x}_0$
- $x^* \in$ in the interior of $(\boldsymbol{x}_0)$, not at the boundary
- f continuous enough: $\mathcal{C}^2$



Algorithm: optimize a continuous function

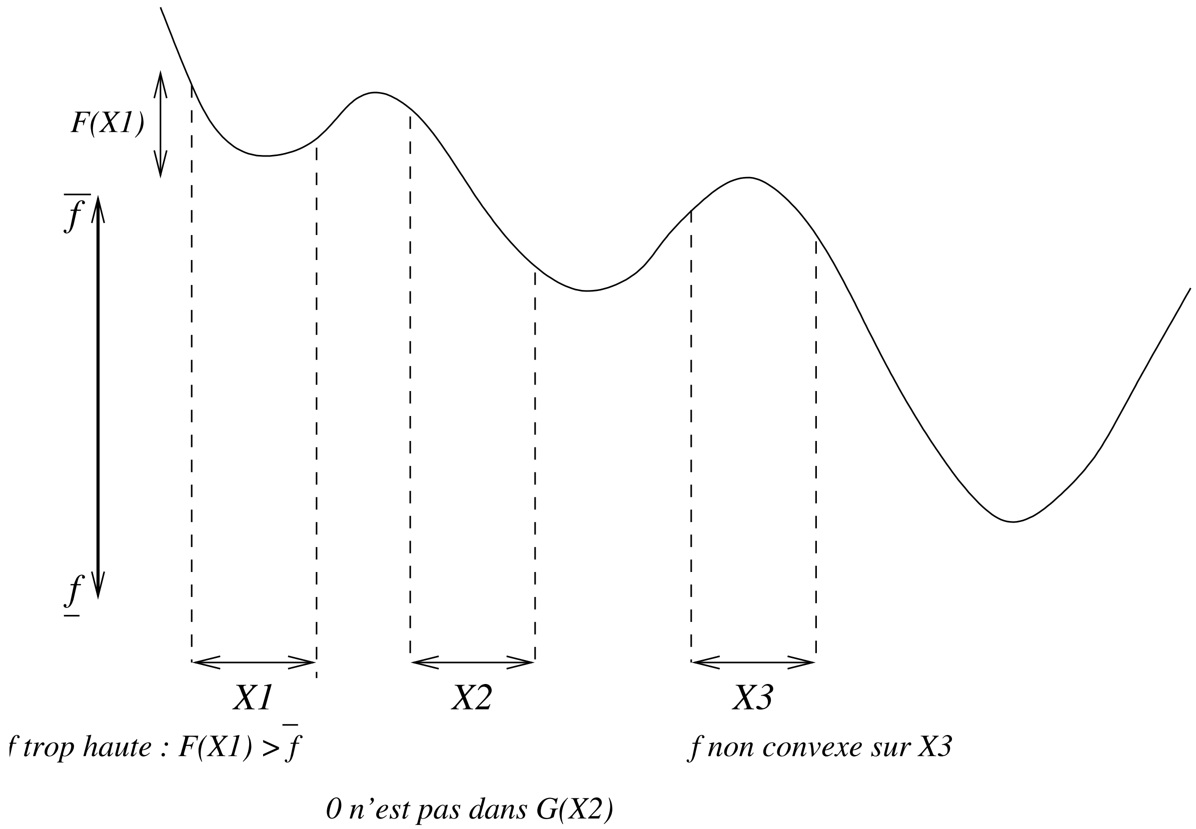
(Ratschek and Rokne 1988, Hansen 1992, Kearfott 1996. . .)

Goal: determine the minimum of f, continuous function on a box $\boldsymbol{x}_0$.

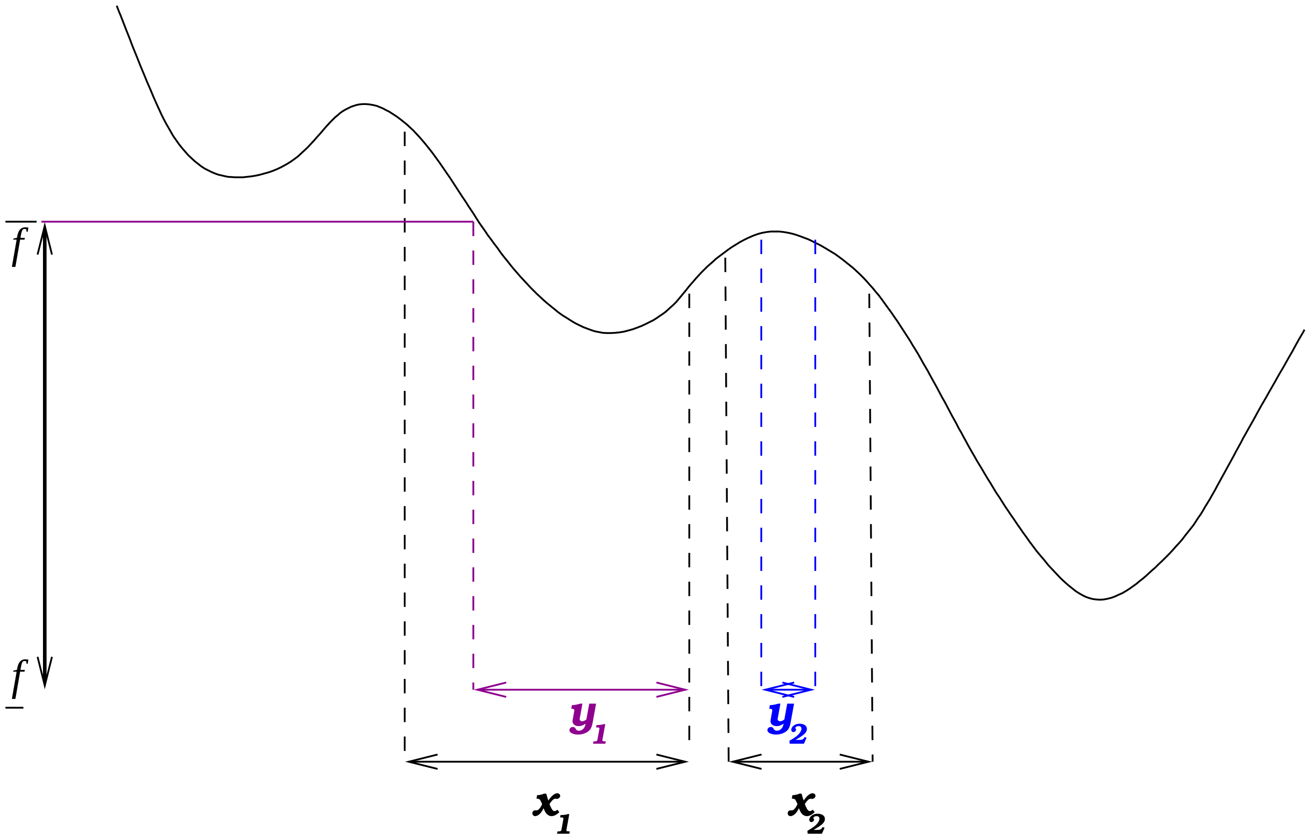
$\boldsymbol{x}_0$ current box
$\bar{f}$ current upper bound of f^*

while there is a box in the waiting list
    if $f(\boldsymbol{x}) > \bar{f}$ then
        reject $\boldsymbol{x}$
    otherwise
        update $\bar{f}$: if $f(mid(\boldsymbol{x})) < \bar{f}$ then $\bar{f} = f(mid(\boldsymbol{x}))$
        bisect $\boldsymbol{x}$ into $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$
        examine $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$

Algorithm: optimize a continuous function
the rejection procedure



Algorithm: optimize a continuous function
the reduction procedure



Algorithm: optimize a continuous function

Hansen algorithm **Hansen 1992**

$\mathcal{L}$ = list of not yet examined boxes $:= \{\boldsymbol{x}_0\}$

while $\mathcal{L} \neq \emptyset$ **loop**

remove $\boldsymbol{x}$ from $\mathcal{L}$

reject $\boldsymbol{x}$**?**

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

yes if Grad$f(\boldsymbol{x}) \not\ni 0$

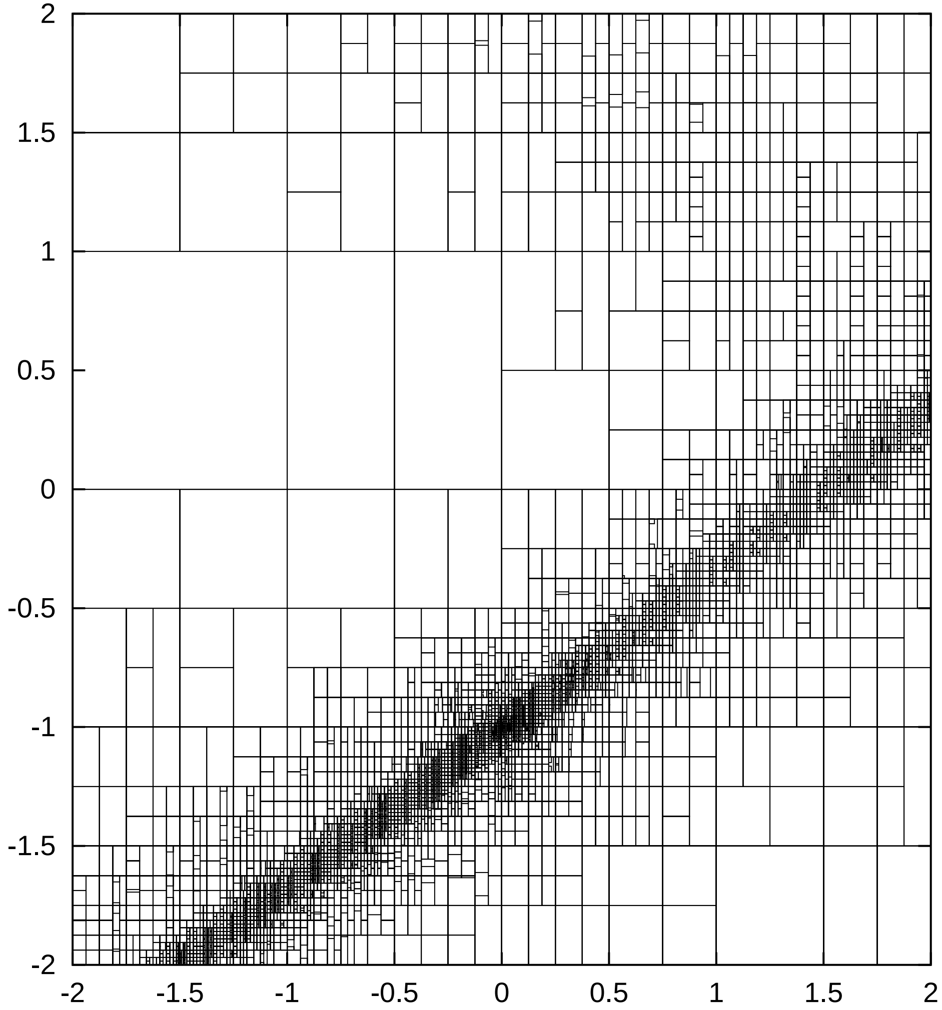
yes if $Hf(\boldsymbol{x})$ has its diagonal non > 0

reduce $\boldsymbol{x}$

Newton applied to the gradient

solve $\boldsymbol{y} \subset \boldsymbol{x}$ such that $f(\boldsymbol{y}) \leq \bar{f}$

bisect $\boldsymbol{y}$: insert the resulting $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ in $\mathcal{L}$



2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Algorithm: constrained optimization

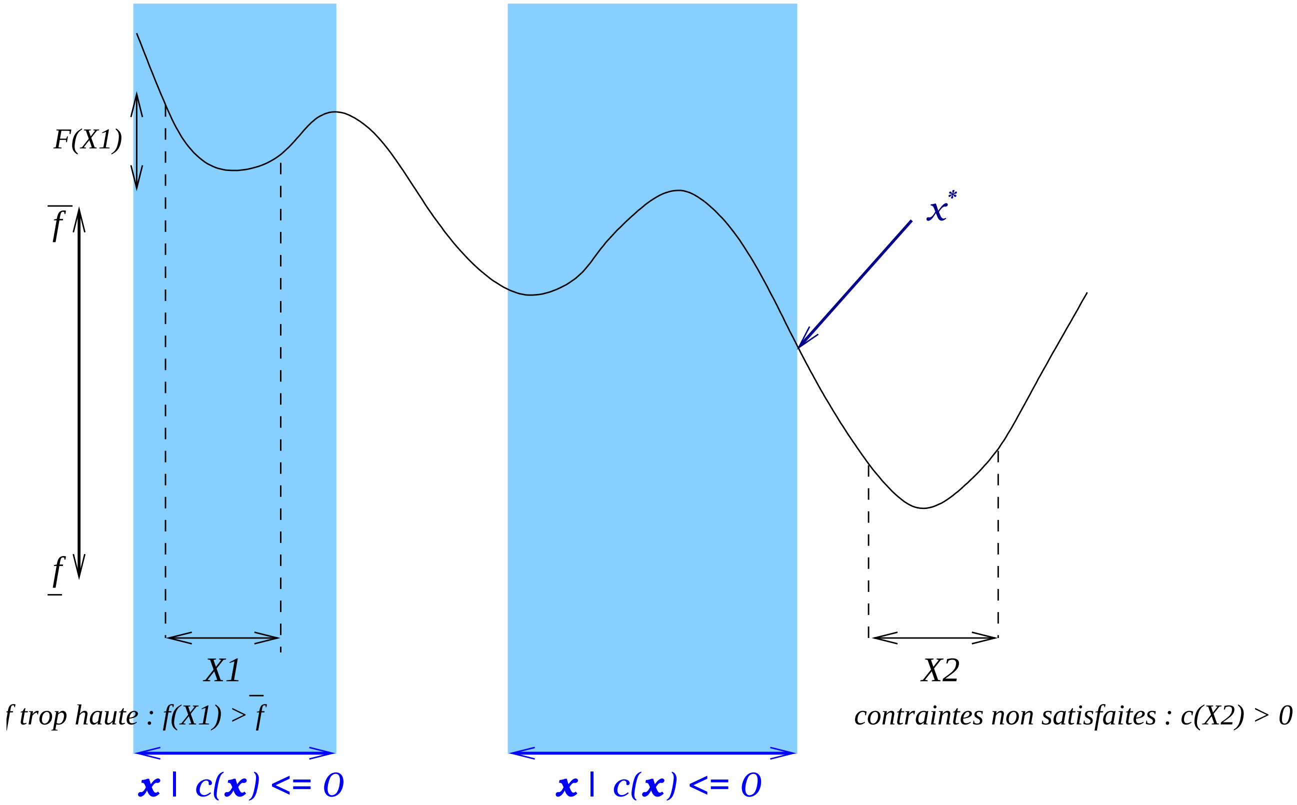
Problem: $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ and $c : \mathbb{R}^n \longrightarrow \mathbb{R}^m$,
determine x^* and f^* that verify

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

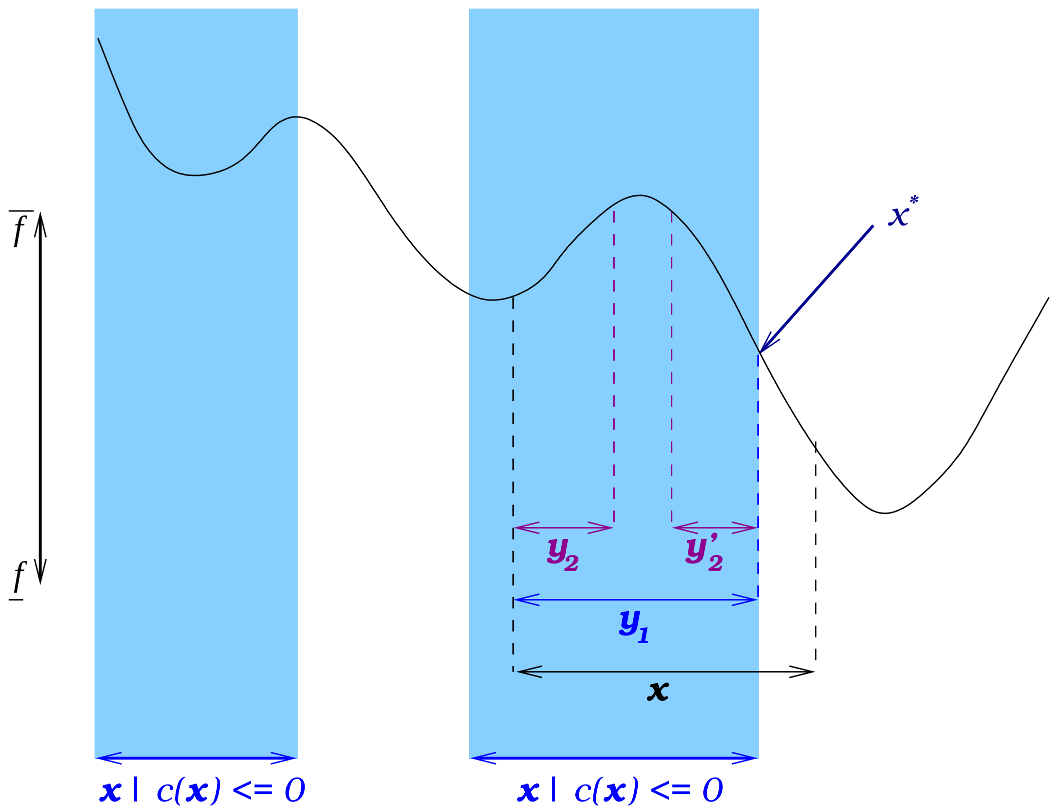
Assumptions:

- search within a box $\boldsymbol{x}_0$
- f continuous enough: $\mathcal{C}^2$
- c continuous enough: $\mathcal{C}^1$

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



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



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

$\mathcal{L} := \{\boldsymbol{x}_0\}$
while $\mathcal{L} \neq \emptyset$ **loop**
remove $\boldsymbol{x}$ from $\mathcal{L}$
reject $\boldsymbol{x}$**?**
yes if $f(\boldsymbol{x}) > \bar{f}$
yes if $\mathrm{Grad} f(\boldsymbol{x}) \not\ni 0$
yes if f not convex on $\boldsymbol{x}$
reduce $\boldsymbol{x}$
solve $\boldsymbol{y} \subset \boldsymbol{x} \mid f(\boldsymbol{y}) \leq \bar{f}$
Newton applied to the gradient
bisect $\boldsymbol{y}$ into $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$
insert $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ in $\mathcal{L}$

$\mathcal{L} := \{\boldsymbol{x}_0\}$
while $\mathcal{L} \neq \emptyset$ **loop**
remove $\boldsymbol{x}$ from $\mathcal{L}$
reject $\boldsymbol{x}$**?**
yes if $f(\boldsymbol{x}) > \overline{f}$
yes if $c(\boldsymbol{x}) > 0$
reduce $\boldsymbol{x}$
'solve $\boldsymbol{y} \subset \boldsymbol{x}$ such that $c(\boldsymbol{y}) \leq$
Newton applied to the Lagran
bisect $\boldsymbol{y}$ into $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$
insert $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ in $\mathcal{L}$

Algorithm: constraints programming

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

Problem:

$$\left\{\begin{array}{rcl} c_1(x_1, \ldots, x_n) & = & 0 \\ \vdots & & \\ c_p(x_1, \ldots, x_n) & = & 0 \end{array}\right.$$

expressed as:

$$\begin{array}{lrcll} & y_i & = & x_i & \text{for } 1 \leq i \leq n \\ & y_k & = & y_i \diamond y_j & \text{for } n+1 \leq k \leq m \text{ and } i, j < k \\ & & & & y_k \text{ auxiliary variable} \\ \text{where} & y_k & = & \varphi(y_i) & \text{for } n+1 \leq k \leq m \text{ and } i < k \end{array}$$

Algorithm: constraints programming

Initializations: $\boldsymbol{y}_1 := \boldsymbol{x}_1, \ldots, \boldsymbol{y}_n := \boldsymbol{x}_n$

Propagation: forward mode

for $k = n + 1$ to m loop

$\boldsymbol{y}_k := \boldsymbol{y}_i \diamond \boldsymbol{y}_j$ or $\boldsymbol{y}_k := \varphi(\boldsymbol{y}_i)$

Propagation: backward mode

for $k = m$ to n loop

if $\boldsymbol{y}_k$ is defined as $\boldsymbol{y}_i \diamond \boldsymbol{y}_j$ then

$\boldsymbol{y}_i := (\boldsymbol{y}_k \diamond^{-r} \boldsymbol{y}_j) \cap \boldsymbol{y}_i$

$\boldsymbol{y}_j := \left(\boldsymbol{y}_i \diamond^{-l} \boldsymbol{y}_k\right) \cap \boldsymbol{y}_j$

else if $\boldsymbol{y}_k$ is defined as $\varphi(\boldsymbol{y}_i)$ then

$\boldsymbol{y}_i := \varphi^{-1}(\boldsymbol{y}_k) \cap \boldsymbol{y}_i$

Algorithm: constraints programming: $\left\{ \begin{array}{lcl} x_1 x_2^2 - 2x_3 & = & 0 \\ \cos x_1 + x_3 & = & 0 \end{array} \right.$

$\boldsymbol{x}_1 = [0, 2\pi/3],\ \boldsymbol{x}_2 = [-1, 1],\ \boldsymbol{x}_3 = [-1/2, 3]$		$\boldsymbol{y}_1 = [0, 2\pi/3],\ \boldsymbol{y}_2 = [-1, 1],\ \boldsymbol{y}_3 = [-1/2, 3]$
iter. 1 : forward	$\boldsymbol{y}_4 = \boldsymbol{y}_2^2$	$\boldsymbol{y}_4 = [0, 1]$
	$\boldsymbol{y}_5 = \boldsymbol{y}_1 \boldsymbol{y}_4$	$\boldsymbol{y}_5 = [0, 2\pi/3]$
	$\boldsymbol{y}_6 = 2\boldsymbol{y}_3$	$\boldsymbol{y}_6 = [-1, 6]$
	$\boldsymbol{y}_7 = \boldsymbol{y}_5 - \boldsymbol{y}_6$	$\boldsymbol{y}_7 = [-6, 1 + 2\pi/3] \ni 0$
	$\boldsymbol{y}_8 = \cos \boldsymbol{y}_1$	$\boldsymbol{y}_8 = [-1/2, 1]$
	$\boldsymbol{y}_9 = \boldsymbol{y}_8 + \boldsymbol{y}_3$	$\boldsymbol{y}_9 = [-1, 4] \ni 0$
backward	$\boldsymbol{y}_9 = \boldsymbol{y}_8 + \boldsymbol{y}_3$	$\begin{cases} \boldsymbol{y}_8 = (\boldsymbol{y}_9 - \boldsymbol{y}_3) \cap \boldsymbol{y}_8 = [-1/2, 1/2] \\ \boldsymbol{y}_3 = (\boldsymbol{y}_9 - \boldsymbol{y}_8) \cap \boldsymbol{y}_3 = [-1/2, 1/2] \end{cases}$
	$\boldsymbol{y}_8 = \cos \boldsymbol{y}_1$	$\boldsymbol{y}_1 = \cos^{-1} \boldsymbol{y}_8 \cap \boldsymbol{y}_1 = [\pi/3, 2\pi/3]$
	$\boldsymbol{y}_7 = \boldsymbol{y}_5 - \boldsymbol{y}_6$	$\begin{cases} \boldsymbol{y}_5 = (\boldsymbol{y}_7 + \boldsymbol{y}_6) \cap \boldsymbol{y}_5 = [0, 2\pi/3] \\ \boldsymbol{y}_6 = (\boldsymbol{y}_5 - \boldsymbol{y}_7) \cap \boldsymbol{y}_6 = [0, 2\pi/3] \end{cases}$
	$\boldsymbol{y}_6 = 2\boldsymbol{y}_3$	$\boldsymbol{y}_3 = (1/2\boldsymbol{y}_6) \cap \boldsymbol{y}_3 = [0, 1/2]$
	$\boldsymbol{y}_5 = \boldsymbol{y}_1 \boldsymbol{y}_4$	$\begin{cases} \boldsymbol{y}_1 = (\boldsymbol{y}_5/\boldsymbol{y}_4) \cap \boldsymbol{y}_1 = [\pi/3, 2\pi/3] \\ \boldsymbol{y}_4 = (\boldsymbol{y}_5/\boldsymbol{y}_1) \cap \boldsymbol{y}_4 = [0, 1] \end{cases}$
	$\boldsymbol{y}_4 = \boldsymbol{y}_2^2$	$\boldsymbol{y}_2 = \pm\sqrt{\boldsymbol{y}_4} \cap \boldsymbol{y}_2 = [-1, 1]$

Algorithm: constraints programming: $\left\{ \begin{array}{rcl} x_1 x_2^2 - 2x_3 & = & 0 \\ \cos x_1 + x_3 & = & 0 \end{array} \right.$

	$x_1 = [0, \frac{2\pi}{3}],\ x_2 = [-1, 1],\ x_3 = [-\frac{1}{2}, 3]$	$y_1 = [\frac{\pi}{3}, \frac{2\pi}{3}],\ y_2 = [-1, 1],\ y_3 = [0, \frac{1}{2}]$
		$y_4 = [0, 1],\ y_5 = [0, 2\pi/3],\ y_6 = [0, 1]$
		$y_7 = 0,\ y_8 = [-1/2, 1/2],\ y_9 = 0$
iter. 2: forward	$y_4 = y_2^2$	$y_4 = [0, 1]$
	$y_5 = y_1 y_4$	$y_5 = [0, 2\pi/3]$
	$y_6 = 2 y_3$	$y_6 = [0, 1]$
	$y_8 = \cos y_1$	$y_8 = [-1/2, 1/2]$
backward	$y_9 = y_8 + y_3$	$\begin{cases} y_8 = (y_9 - y_3) \cap y_8 = [-1/2, 0] \\ y_3 = (y_9 - y_8) \cap y_3 = [0, 1/2] \end{cases}$
	$y_8 = \cos y_1$	$y_1 = \cos^{-1} y_8 \cap y_1 = [\pi/2, 2\pi/3]$
	$y_7 = y_5 - y_6$	$\begin{cases} y_5 = (y_7 + y_6) \cap y_5 = [0, 1] \\ y_6 = (y_5 - y_7) \cap y_6 = [0, 1] \end{cases}$
	$y_6 = 2 y_3$	$y_3 = (1/2 y_6) \cap y_3 = [0, 1/2]$
	$y_5 = y_1 y_4$	$\begin{cases} y_1 = (y_5 / y_4) \cap y_1 = [\pi/2, 2\pi/3] \\ y_4 = (y_5 / y_1) \cap y_4 = [0, 2/\pi] \end{cases}$
	$y_4 = y_2^2$	$y_2 = \pm\sqrt{y_4} \cap y_2 = [-\sqrt{2/\pi}, \sqrt{2/\pi}]$
	$x_1 = [0, \frac{2\pi}{3}],\ x_2 = [-1, 1],\ x_3 = [-\frac{1}{2}, 3]$	$y_1 = [\frac{\pi}{2}, \frac{2\pi}{3}],\ y_2 = [-\sqrt{2/\pi}, \sqrt{2/\pi}],\ y_3 = [0, \frac{1}{2}]$

Problem: $\begin{cases} x_1 x_2^2 - 2x_3 & = & 0 \\ \cos x_1 + x_3 & = & 0 \end{cases}$

with $\boldsymbol{x}_1 = [0, \frac{2\pi}{3}]$, $\boldsymbol{x}_2 = [-1, 1]$, $\boldsymbol{x}_3 = [-\frac{1}{2}, 3]$.

Optimal solution obtained after two iterations:
$\boldsymbol{x}_1 = [\frac{\pi}{2}, \frac{2\pi}{3}]$, $\boldsymbol{x}_2 = [-\sqrt{\frac{2}{\pi}}, \sqrt{\frac{2}{\pi}}]$, $\boldsymbol{x}_3 = [0, \frac{1}{2}]$.

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)
  $\Rightarrow$ 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