Interval arithmetic to handle uncertainties and to assess numerical quality

Nathalie Revol
INRIA - Université de Lyon
LIP (UMR 5668 CNRS - ENS Lyon - INRIA - UCBL)

Journée GAMNI-MAIRCI *Précision et incertitudes*, 1er février 2012

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Who invented Interval Arithmetic?

- **1962:** Ramon Moore defines IA in his PhD thesis and then a rather exhaustive study of IA in a book in 1966
- **1958:** Tsunaga, in his MSc thesis in Japanese
- **1956:** Warmus
- **1951:** Dwyer, in the specific case of closed intervals
- **1931:** Rosalind Cecil Young in her PhD thesis in Cambridge (UK) has used some formulas
- **1927:** Bradis, for positive quantities, in Russian
- **1908:** Young, for some bounded functions, in Italian
- **3rd century BC:** Archimedes, to compute an enclosure of π!

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

Historical remarks

Childhood until the seventies.

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

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

IEEE-1788 standard for interval arithmetic in 2014?
I hope so...

A brief introduction

Interval arithmetic:
instead of numbers, use 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.

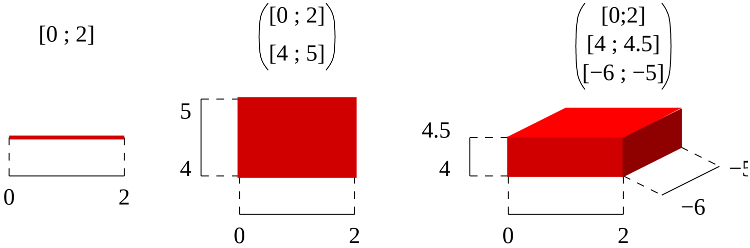
No result is lost, the computed interval is guaranteed to contain every possible result.

Agenda

Definitions: intervals

Objects:

- intervals of real numbers = closed connected sets of **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

$$\mathbf{x} \diamond \mathbf{y} = \mathbf{Hull}\{x \diamond y \ : \ x \in \mathbf{x}, y \in \mathbf{y}\}$$

Arithmetic and algebraic operations: use the monotonicity

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

Definitions: functions

Definition:
an interval extension $\mathbf{f}$ of a function f satisfies

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

Elementary functions: again, use the monotony.

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

Definitions: function extension

$f(x) = x^2 - x + 1 = x(x-1) + 1 = (x - 1/2)^2 + 3/4$ **on** $[-2, 1]$.

Using $x^2 - x + 1$, one gets
$[-2,1]^2 - [-2,1] + 1 = [0,4] + [-1,2] + 1 = [0,7]$.

Using $x(x-1)+1$, one gets
$[-2,1] \cdot ([-2,1] - 1) + 1 = [-2,1] \cdot [-3,0] + 1 = [-3,6] + 1 = [-2,7]$.

Using $(x-1/2)^2 + 3/4$, one gets
$([-2,1] - 1/2)^2 + 3/4 = [-5/2, 1/2]^2 + 3/4 = [0, 25/4] + 3/4 =$
$[3/4, 7] = f([-2,1])$.

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

How to choose the best extension? A good one?

Agenda

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:

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

Cons: overestimation (2/2)

Wrapping effect

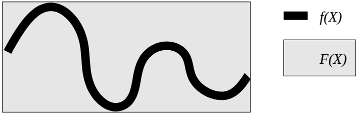
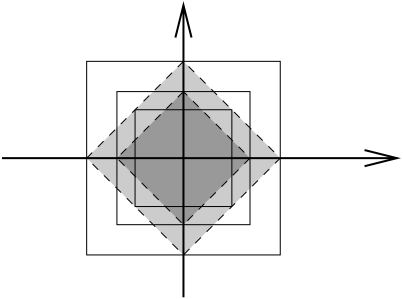


image of $f(\mathbf{x})$
with $f : \mathbf{R}^2 \to \mathbf{R}^2$



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

Cons: complexity and efficiency

Complexity: most problems are NP-hard **(Gaganov, Rohn, Kreinovich. . .)**

- evaluate a function on a box. . . even up to ε
- solve a linear system. . . even up to $1/4n^4$
- determine if the solution of a linear system is bounded

Efficiency

Implementation using floating-point arithmetic:
use directed roundings, towards $\pm\infty$.

Overhead in execution time:
in theory, at most 4, or 8, cf.

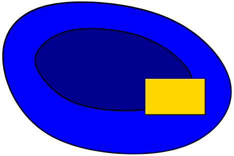
$$[\underline{x}, \overline{x}] \times [\underline{y}, \overline{y}] = [\min(RD(\underline{x} \times \underline{y}), RD(\underline{x} \times \overline{y}), RD(\overline{x} \times \underline{y}), RD(\overline{x} \times \overline{y})), \max(RU(\underline{x} \times \underline{y}), RU(\underline{x} \times \overline{y}), RU(\overline{x} \times \underline{y}), RU(\overline{x} \times \overline{y}))$$

in practice, around 20: changing the rounding modes implies flushing the pipelines (on most architectures and implementations).

Pros: set computing

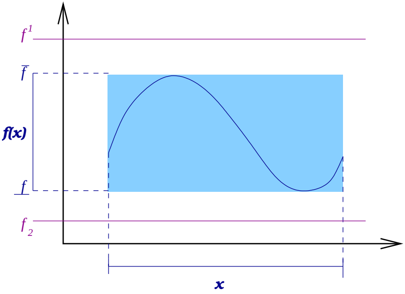
Computing with whole sets or with sets enclosing uncertainties.

Behaviour safe?
controllable? dangerous?



always controllable.

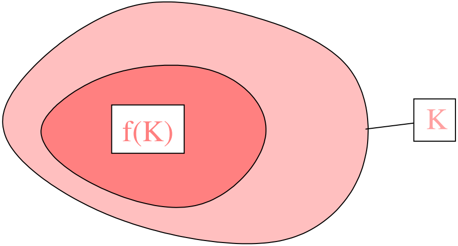
On $\mathbf{x}$, are the extrema of the function f
$> f^1$, $< f_2$?



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

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 $f(B) \subset \text{int}B$ then f has a unique fixed point on B.



The theorem remains valid if B is replaced by a compact K and in particular an interval.

Agenda

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:

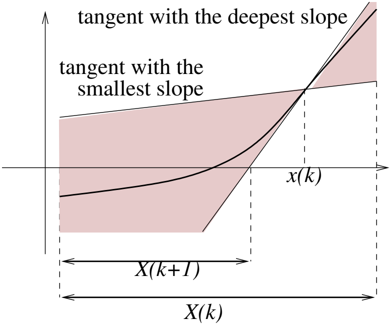
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{f(\mathbf{x}_k)}{f'(\mathbf{x}_k)}$$

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

divergence!

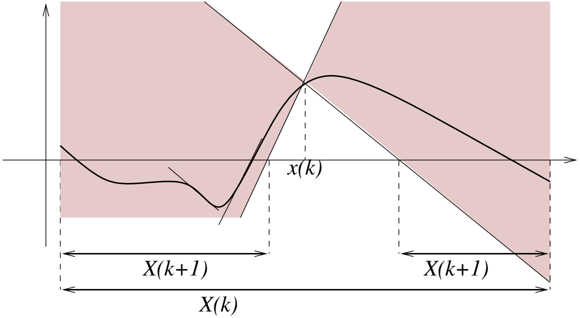
Algorithm: interval Newton principle of an iteration

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



$$\mathbf{x}_{k+1} := \left(x_k - \frac{\mathbf{f}(\{x_k\})}{\mathbf{f}'(\mathbf{x}_k)} \right) \bigcap \mathbf{x}_k$$

Algorithm: interval Newton principle of an iteration



$$(\mathbf{x}_{k+1,1}, \mathbf{x}_{k+1,2}) := \left(x_k - \frac{\mathbf{f}(\{x_k\})}{\mathbf{f}'(\mathbf{x}_k)} \right) \bigcap \mathbf{x}_k$$

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

Agenda

Preliminary remarks

- Complexity: polynomial in time . . . for an NP-hard problem: no guarantee on the accuracy of the solution, failure is possible.
- This algorithm is usually employed to verify the solution of a linear system with floating-point coefficients: interval arithmetic is used as a verification tool.

Here **verification** corresponds to **precision's assessment.**

joint work with H. D. Nguyen

Problem: verified solution of a linear system

Goals: For a linear system $Ax = b$ with $A \in \mathbb{F}^{n \times n}$ non-singular and $b \in \mathbb{F}^n$, we want to

1. compute an approximation $\tilde{x} \in \mathbb{F}^n$ of the exact solution x^*,
2. simultaneously bound the error upon $\tilde{x}$, or enclose it in an interval

$$\mathbf{e} \ni x^* - \tilde{x}.$$

Remark: denote by e the error $x^* - \tilde{x}$.
Then e is the solution of the residual system $Ae = b - A\tilde{x}$.
Indeed, $Ae = A(x^* - \tilde{x}) = Ax^* - A\tilde{x} = b - A\tilde{x}$.

Agenda

Introduction to interval arithmetic
- operations, function extensions
- cons and pros
- interval Newton

Verified solutions of linear systems
- stating the problem
- iterative refinement
- concluding remarks

Variants of interval arithmetic
- higher precision, affine arithmetic, Taylor models

Conclusions

Method: contractant iteration
Classical iterative refinement

Wilkinson (1963), Higham (2000), Demmel et al. (2006) . .

Algorithm (Classical iterative refinement)

Input: $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$
$\tilde{x} = A \setminus b$ % *MatLab-like syntax*
while(not converged)
  $\tilde{r} = b - A\,\tilde{x}$
  $\tilde{e} = A \setminus \tilde{r}$
  $\tilde{x} = \tilde{x} + \tilde{e}$
end
Output: $\tilde{x}$

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$
$\tilde{x} = A \setminus b$ % *MatLab-like syntax*
while(not converged)
  $\tilde{r} = b - A\,\tilde{x}$
  $\tilde{e} = A \setminus \tilde{r}$
  $\tilde{x} = \tilde{x} + \tilde{e}$
end
Output: $\tilde{x}$

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$
$\tilde{x} = A \setminus b$ % *MatLab-like syntax*
while(not converged)
  $\mathbf{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \mathbf{r}$
  $\tilde{e} = A \setminus \tilde{r}$
  $\tilde{x} = \tilde{x} + \tilde{e}$
end
Output: $\tilde{x}$

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}$, $b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like syntax*

while(not converged)

$\mathbf{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \tilde{e}$

end

Output: $\tilde{x}$

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

while(not converged)

  $\mathbf{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \mathbf{r}$

  $\mathbf{e} = A \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

  $\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}$

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like syntax*

while(not converged)

$\mathbf{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}, \mathbf{e}$

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like syntax*

while(not converged)

$\mathbf{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = A \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}, \mathbf{e}$

Solving interval residual system?

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

while(not converged)

$\quad \mathbf{r} = [b - A\,\tilde{x}] \qquad \% \quad A(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = A \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}, \mathbf{e}$

K is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \% \quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = A \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}, \mathbf{e}$

$\mathbf{K}$ is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \% \quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}, \mathbf{e}$

$\mathbf{K}$ is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n \times n}$, $b \in \mathbb{F}^n$
$\tilde{x} = A \setminus b$, $\quad R = inv(A)$, $\quad \mathbf{K} = [RA]$
while(not converged)
$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ $\qquad$ % $\quad RA(x^* - \tilde{x}) \in \mathbf{r}$
$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r}$ $\qquad$ % $\quad x^* - \tilde{x} \in \mathbf{e}$
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$, $\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$
end
Output: $\tilde{x}, \mathbf{e}$

$\mathbf{K}$ is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method: contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \% \quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

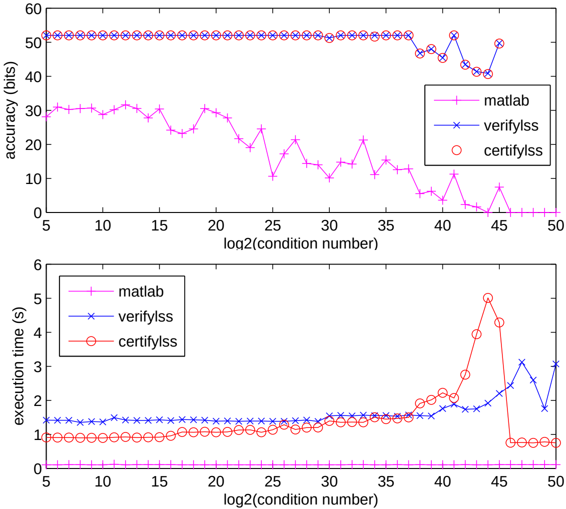
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\tilde{x}, \mathbf{e}$

This algorithm can fail, if it fails to solve the interval linear system.

Experimental Results: dim $= 1000$ $b = [1, \ldots, 1]^T$



Method: contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}] \qquad \% \quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method: contractant iteration Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input: $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$, $\quad R = inv(A)$, $\quad \mathbf{K} = [RA]$,

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\quad$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ $\quad$ % $\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r}$ $\quad$ % $\quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$, $\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method: contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$ % *cost: 1 floating-point matrix-vector product*

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method: contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input: $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$, $\quad R = inv(A)$, $\quad \mathbf{K} = [RA]$,

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\quad$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ $\quad$ % $\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

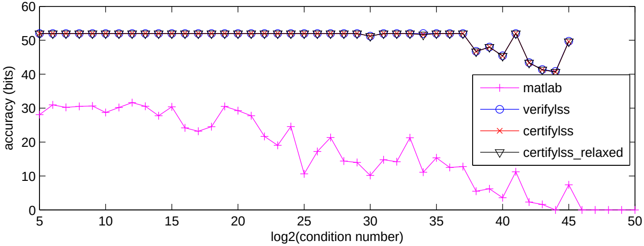
$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$ $\quad$ % *cost: 1 floating-point matrix-vector product*

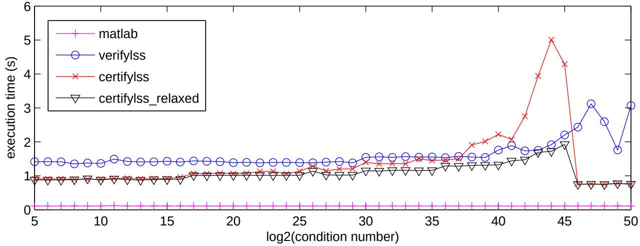
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$, $\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Relaxed method, results: $\dim = 1000 \qquad b = [1, \ldots, 1]^T$





Method: contractant iteration
Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\qquad$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method: contractant iteration
Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n \times n}$, $b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$, $\quad R = inv(A)$, $\quad \mathbf{K} = [RA]$,

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\quad$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ $\quad$ % $\mathbf{r}$ *in twice the working precision*

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method: contractant iteration
Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $\mathbf{r}$ *in twice the working precision*

$\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ % $\tilde{x}$ *in twice the working precision*

$\mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method: contractant iteration
Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \quad R = inv(A), \quad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ *% $\mathbf{r}$ in twice the working precision*

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ *% $\tilde{x}$ in twice the working precision*

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

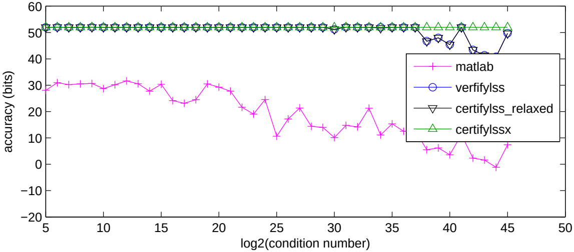
Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

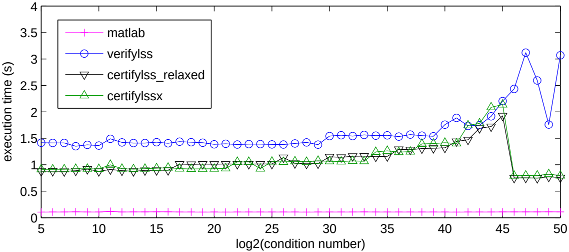
Implementation: careful tuning of the precision of each variable (no doubling for $\mathbf{e}$: useless and costly).

Extra-precise relaxed method: Results

$\dim = 1000$

$b = [1, \dots, 1]^T$





Agenda

Morale

Hidden under the carpet in this talk: proofs that
full accuracy is reached (when no failure),
at most width $= 2\times$ width of HBRKN (most used method)...

- keep your goals in mind (accuracy, efficiency)
- reuse optimized blocks (BLAS3)
- build algorithms by assembling building blocks
- interval arithmetic can be a tool for verification purposes

Future work:

- push further the condition number limits
- propose a verified BLAS / Lapack library
- implemented on multicores

Agenda

Introduction to interval arithmetic
- operations, function extensions
- cons and pros
- interval Newton

Verified solutions of linear systems
- stating the problem
- iterative refinement
- concluding remarks

Variants of interval arithmetic
- higher precision, affine arithmetic, Taylor models

Conclusions

Higher precision: extended / arbitrary

Extended precision (double-double, triple-double): (Moler, Priest, Dekker, Knuth, Shewchuk, Bailey. . .)
a number is represented as the sum of 2 (or 3 or . . .) floating-point numbers. Do not evaluate the sum using floating-point arithmetic!
Double-double arith. is implemented using IEEE-754 FP arith.

Arbitrary precision: the precision is chosen by the user, the only limit being the computer's memory.
Arithmetic is implemented in software, e.g. MPFR (**Zimmermann et al.**), MPFI (**Revol, Rouillier et al., Yamamoto, Krämer et al.**).

Tradeoff between accuracy and efficiency (and memory):
double-double: accuracy "$\times 2$", ≤ 1 order of magnitude slower
arbitrary precision: accuracy "∞", $\geq$ 1-2 order of magnitude slower
(provided Higham's rule of thumb applies).

Affine arithmetic (Comba, Stolfi and Figueiredo (1993, 2004), Messine and Ninin (2009), Goubault, Martel and Putot (Fluctuat))

Definition: each input or computed quantity x is represented by
$x = x_0 + \alpha_1 \varepsilon_1 + \alpha_2 \varepsilon_2 + \cdots + \alpha_n \varepsilon_n$
where x_0, α_1, $\ldots \alpha_n$ are known real / floating-point numbers,
and $\varepsilon_1 \ldots \varepsilon_n$ are symbolic variables for uncertainties, $\in [-1, +1]$.
Example: $x \in [3, 7]$ is represented by $x = 5 + 2\varepsilon$.

Operations:
$(x + \sum_k \alpha_k \varepsilon_k) + (y + \sum_k \beta_k \varepsilon_k) = (x + y) + \sum_k (\alpha_k + \beta_k)\varepsilon_k$.
$(x + \sum_k \alpha_k \varepsilon_k) \times (y + \sum_k \beta_k \varepsilon_k) = (x \times y) + \sum_k (x\beta_k + y\alpha_k)\varepsilon_k + \gamma_l \varepsilon_l$
with ε_l a new variable.

Roundoff errors: compute δ_l an upper bound of all roundoff errors and add it to γ_l.

Computing precision: Fluctuat uses arbitrary precision, internally.

Taylor models

Berz, Hoefkens and Makino 1998, Nedialkov, Neher, Tucker, Wittig

Principle: represent a function $f(x)$ for $x \in [-1, 1]$ by a polynomial part $p(x)$ and a reminder part (a big bin) I such that $\forall x \in [-1, 1]$, $f(x) \in p(x) + I$.

Operations:

- affine operations: straigthforward;
- non-affine operations: enclose the nonlinear terms and add this enclosure to the reminder.

Roundoff errors: determine an upper bound b on the roundoff errors and add $[-b, b]$ to the reminder.

Computing precision: use of double-double arithmetic to increase the accuracy (ongoing work).

Agenda

Conclusions

Interval algorithms

- can solve problems that other techniques are not able to solve
- are 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
- are less efficient than floating-point arithmetic (theoretical factor: 4, practical factor: 20 to 100)
  $\Rightarrow$ solve "small" problems.
- can be used to **verify floating-point computations**.

Philosophical conclusion

Morale

- don't be naive when using interval arithmetic
- 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

Appendix: References on interval arithmetic

- R. Moore: *Interval Analysis*, Prentice Hall, Englewood Cliffs, 1966.
- A. Neumaier: *Interval methods for systems of equations*, CUP, 1990.
- R. Moore, R.B. Kearfott, M.J. Cloud: *Introduction to interval analysis*, SIAM, 2009.
- S.M. Rump: *Computer-assisted proofs and Self-Validating Methods*. In B. Einarsson ed., Handbook on Accuracy and Reliability in Scientific Computation, pp. 195-240. SIAM, 2005.
- S.M. Rump: *Verification methods: Rigorous results using floating-point arithmetic*, Acta Numerica, vol. 19, pp. 287-449, 2010.

Appendix: References on interval arithmetic

- J. Rohn: *A Handbook of Results on Interval Linear Problems*, `http://www.cs.cas.cz/rohn/handbook` 2006.
- 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.

Appendix: more operations

$\mathbf{x} \diamond \mathbf{y} = \mathbf{Hull}\{x \diamond y \ : \ x \in \mathbf{x}, y \in \mathbf{y}\}$

Arithmetic and algebraic operations: use the monotonicity

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

Definitions: operations

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

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

Appendix: some more about 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:

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

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

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:

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

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

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

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:

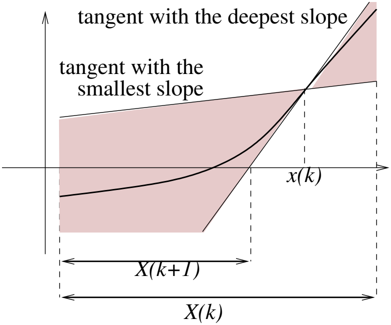
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{f(\mathbf{x}_k)}{f'(\mathbf{x}_k)}$$

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

divergence!

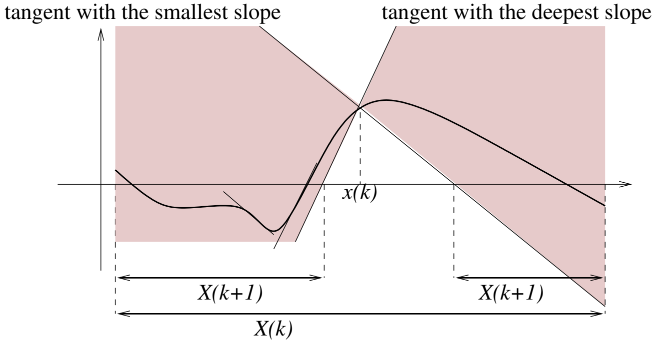
Algorithm: interval Newton principle of an iteration

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



$$\mathbf{x}_{k+1} := \left(x_k - \frac{\mathbf{f}(\{x_k\})}{\mathbf{f}'(\mathbf{x}_k)} \right) \bigcap \mathbf{x}_k$$

Algorithm: interval Newton principle of an iteration



$$(\mathbf{x}_{k+1,1}, \mathbf{x}_{k+1,2}) := \left(x_k - \frac{\mathbf{f}(\{x_k\})}{\mathbf{f}'(\mathbf{x}_k)}\right) \bigcap \mathbf{x}_k$$

Algorithm: interval Newton

Input: $\mathbf{f}$, $\mathbf{f'}$, $\mathbf{x}_0$ // $\mathbf{x}_0$ initial search interval
Initialization: $\mathcal{L} = \{\mathbf{x}_0\}$, $\alpha = 0.75$ //any value in $]0.5, 1[$ is suitable
Loop: while $\mathcal{L} \neq \emptyset$
  Suppress $(\mathbf{x}, \mathcal{L})$
  $x := \text{mid}(\mathbf{x})$
  $(\mathbf{x}_1, \mathbf{x}_2) := \left(x - \frac{\mathbf{f}(\{x\})}{\mathbf{f'}(\mathbf{x})}\right) \bigcap \mathbf{x}$ // $\mathbf{x}_1$ and $\mathbf{x}_2$ can be empty
  if $w(\mathbf{x}_1) > \alpha w(\mathbf{x})$ or $w(\mathbf{x}_2) > \alpha w(\mathbf{x})$ then $(\mathbf{x}_1, \mathbf{x}_2) := \text{bisect}(\mathbf{x})$
  if $\mathbf{x}_1 \neq \emptyset$ and $\mathbf{f}(\mathbf{x}_1) \ni 0$ then
    if $w(\mathbf{x}_1)/|\text{mid}(\mathbf{x}_1)| \leq \varepsilon_{\mathbf{X}}$ or $w(\mathbf{f}(\mathbf{x}_1)) \leq \varepsilon_Y$ then Insert $\mathbf{x}_1$ in Res
    else Insert $\mathbf{x}_1$ in $\mathcal{L}$
  same handling of $\mathbf{x}_2$
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 $f(\inf(\mathbf{x}))$ and $f(\sup(\mathbf{x}))$ have opposite signs.
  (Miranda theorem in higher dimensions).
- using Brouwer theorem: if the new iterate (before $\bigcap$) in contained in the previous one.

Comments on `certifylss`

Iterative refinement performed on the interval residual.

- initialization of $\mathbf{e}$: heuristic trying to determine $\mathbf{e}_0$, based on
  Proposition: let $A \in \mathbb{F}^{n \times n}$ and $R \in \mathbb{F}^{n \times n}$ be a floating-point approximate inverse of A.
  If $< [RA] > u \geq v > 0$ for some $u > 0$ then

  $$\begin{array}{lcl} |A^{-1}\mathbf{r}| & \leq & \|R\mathbf{r}\|_v u \\ A^{-1}\mathbf{r} & \subset & \|R\mathbf{r}\|_v [-u, u]. \end{array}$$

  Idea: start from $u = e = (1, 1, \dots 1)^t$ and modify u if v is not ≥ 0.
  Failure of the algo if failure of this step.
- solve $\mathbf{Ke} = \mathbf{r}$ using Gauss-Seidel iteration: known to converge quicker than Krawczyk.

Method: contractant iteration Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$, $R = inv(A)$, $\mathbf{K} = [RA]$,

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matri x*

while(not converged)

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

4 $\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$ % *cost: 1 floating-point matrix-vector product*

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$, $\mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Relaxed method: details

The product of $\mathbf{A}$ centered in zero by $\mathbf{B}$ is $[-\bar{A} * \text{mag}\mathbf{B}, \bar{A} * \text{mag}\mathbf{B}]$, i.e. 1 FP matrix product.

Let us decompose $\mathbf{A}$ as $\mathbf{D} + \mathbf{L} + \mathbf{U}$. Then

Jacobi:	$\mathbf{e}' = \mathbf{D}^{-1}(\mathbf{b} - (\mathbf{L} + \mathbf{U})\mathbf{e})$
Gauss-Seidel:	$\mathbf{e}' = \mathbf{D}^{-1}(\mathbf{b} - \mathbf{L}\mathbf{e}' - \mathbf{U}\mathbf{e})$

If $\mathbf{L}$ and $\mathbf{U}$ are inflated so as to be centered in 0:

$$\mathbf{L}' = [-|\mathbf{L}|, |\mathbf{L}|] \quad \text{and} \quad \mathbf{U}' = [-|\mathbf{U}|, |\mathbf{U}|]$$

then Jacobi or Gauss-Seidel costs 1 FP matrix-vector product.
A BLAS2 routine can be used.
Convergence remains linear.
Accuracy of the solution: at most twice as wide as HBRNK.

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b,$

$R = inv(A),$

$\mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while (not converged)

$\quad \mathbf{r} = [b - A\,\tilde{x}]$

$\quad \mathbf{r} = [R\mathbf{r}]$

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ $\qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b,$ $\qquad \frac{2}{3}n^3$

$R = inv(A),$

$\mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\qquad$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

while (not converged)

$\qquad \mathbf{r} = [b - A\,\tilde{x}]$

$\qquad \mathbf{r} = [R\mathbf{r}]$

$\qquad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\qquad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}) \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b,$ $\quad \frac{2}{3}n^3$

$R = inv(A),$ $\quad \frac{4}{3}n^3$

$\mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\quad$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while (not converged)

$\quad \mathbf{r} = [b - A\,\tilde{x}]$

$\quad \mathbf{r} = [R\mathbf{r}]$

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ $\qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b,$ $\quad \frac{2}{3}n^3$

$R = inv(A),$ $\quad \frac{4}{3}n^3$

$\mathbf{K} = [RA],$ $\quad 4n^3$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\quad$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while (not converged)

$\quad \mathbf{r} = [b - A\,\tilde{x}]$

$\quad \mathbf{r} = [R\mathbf{r}]$

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$ $\qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b,$	$\frac{2}{3}n^3$
$R = inv(A),$	$\frac{4}{3}n^3$
$\mathbf{K} = [RA],$	$4n^3$
$\hat{\mathbf{K}} = inflated(\mathbf{K})$	$2n^2$

while (not converged)

  $\mathbf{r} = [b - A\,\tilde{x}]$

  $\mathbf{r} = [R\mathbf{r}]$

  $\mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

  $\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}) \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$	
$\tilde{x} = A \setminus b,$	$\frac{2}{3}n^3$
$R = inv(A),$	$\frac{4}{3}n^3$
$\mathbf{K} = [RA],$	$4n^3$
$\hat{\mathbf{K}} = inflated(\mathbf{K})$	$2n^2$
while (not converged)	
$\quad \mathbf{r} = [b - A\,\tilde{x}]$	$2n^2$
$\quad \mathbf{r} = [R\mathbf{r}]$	
$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$	
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}) \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$	
end	
Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$	

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$,	$\frac{2}{3}n^3$
$R = inv(A)$,	$\frac{4}{3}n^3$
$\mathbf{K} = [RA]$,	$4n^3$
$\hat{\mathbf{K}} = inflated(\mathbf{K})$	$2n^2$
while (not converged)	
$\quad \mathbf{r} = [b - A\,\tilde{x}]$	$2n^2$
$\quad \mathbf{r} = [R\mathbf{r}]$	$4n^2$
$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$	
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}) \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$	
end	

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$	
$\tilde{x} = A \setminus b,$	$\frac{2}{3}n^3$
$R = inv(A),$	$\frac{4}{3}n^3$
$\mathbf{K} = [RA],$	$4n^3$
$\hat{\mathbf{K}} = inflated(\mathbf{K})$	$2n^2$
while (not converged)	
$\quad \mathbf{r} = [b - A\,\tilde{x}]$	$2n^2$
$\quad \mathbf{r} = [R\mathbf{r}]$	$4n^2$
$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$	$2n^2$
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}) \qquad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$	
end	
Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$	

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b,$		$\frac{2}{3}n^3$
$R = inv(A),$		$\frac{4}{3}n^3$
$\mathbf{K} = [RA],$		$4n^3$
$\hat{\mathbf{K}} = inflated(\mathbf{K})$		$2n^2$
while (not converged)		
$\quad \mathbf{r} = [b - A\,\tilde{x}]$		$2n^2$
$\quad \mathbf{r} = [R\mathbf{r}]$		$4n^2$
$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$		$2n^2$
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$	$\mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$	$\mathcal{O}(n)$
end		

Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$

Complexity of `certifylss`

Algorithm (certifylssx)

Input: $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$		
$\tilde{x} = A \setminus b,$		$\frac{2}{3}n^3$
$R = inv(A),$		$\frac{4}{3}n^3$
$\mathbf{K} = [RA],$		$4n^3$
$\hat{\mathbf{K}} = inflated(\mathbf{K})$		$2n^2$
while (not converged)		
$\quad \mathbf{r} = [b - A\,\tilde{x}]$		$2n^2$
$\quad \mathbf{r} = [R\mathbf{r}]$		$4n^2$
$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$		$2n^2$
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$	$\mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$	$\mathcal{O}(n)$
end		
Output: $\mathbf{x} = \tilde{x} + \mathbf{e}$		

Complexity of `certifylss`

Reminder:

$6n^3 + 2n^2$ for the initialization

$8n^2 + \mathcal{O}(n)$ for each iteration

Number of iterations:

- starting with $p - \log_2 \kappa(A)$ correct bits
- linear convergence
- ending with p correct bits

$\Rightarrow \frac{p}{p - \log_2 \kappa(A)}$ iterations.

Total complexity:

$6n^3 + 2n^2 + 8\frac{p}{p - \log_2 \kappa(A)} n^2 + \mathcal{O}(n)$ operations using p bits.