

1/100

True and Accurate results on Computers¹

A Tutorial using INTLAB

Siegfried M. Rump, Hamburg

¹parts appeared in *Handbook on Accuracy and Reliability in Scientific Computation*, ed. Bo Einarsson, pages 195-240, SIAM, 2005; a shortened version also in *JSIAM Bulletin* in Japanese



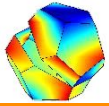
Back

Close

Proofs with the aid of computers



$2^{19937} - 1$ is a prime



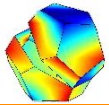
2/100



Back

Close

Proofs with the aid of computers



$2^{19937} - 1$ is a prime

IBM

Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

$2^{19937} - 1$ is a prime

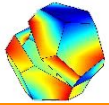
2/100



Back

Close

Proofs with the aid of computers



$2^{19937} - 1$ is a prime

IBM

Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

$2^{19937} - 1$ is a prime

2/100



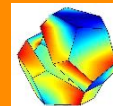
Four colours suffice



Back

Close

Proofs with the aid of computers



$2^{19937} - 1$ is a prime

IBM

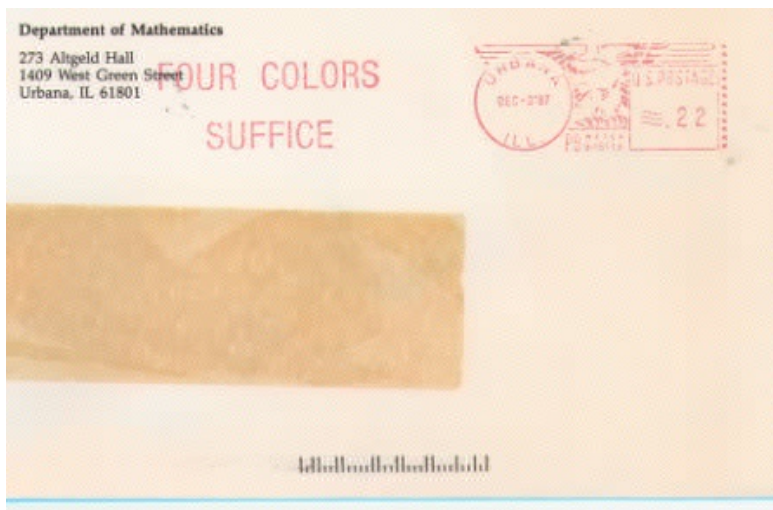
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

$2^{19937} - 1$ is a prime

2/100



Four colours suffice



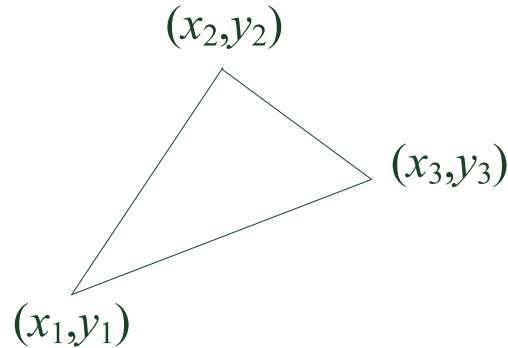
Back

Close

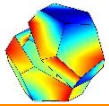
Proofs with the aid of computers (cont'd)

Tarski (1948): The elementary theory of real closed fields is decidable.

Example: Theorems from Geometry



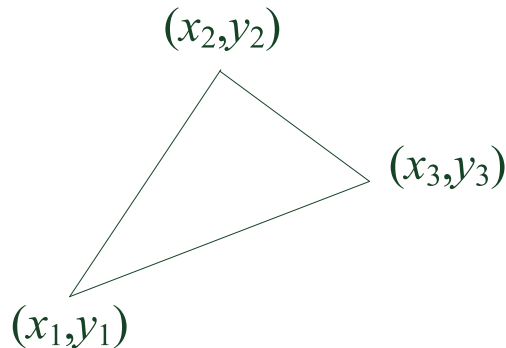
$$\forall x_1 \forall x_2 \forall x_3 \forall y_1 \forall y_2 \forall y_3 \dots \dots$$



Proofs with the aid of computers (cont'd)

Tarski (1948): The elementary theory of real closed fields is decidable.

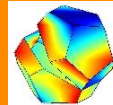
Example: Theorems from Geometry



$$\forall x_1 \forall x_2 \forall x_3 \forall y_1 \forall y_2 \forall y_3 \dots \dots$$

Theorem is true \Leftrightarrow Quantifier-free formula reduces to 1

(possibly with the aid of Gröbner bases)



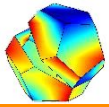
3/100



Back

Close

Proofs with the aid of computers (cont'd)



Risch (1970): Integration in finite terms of functions build out of basic arithmetic operations, roots, powers and elementary standard functions is decidable.

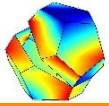
4/100



Back

Close

Proofs with the aid of computers (cont'd)



4/100

Risch (1970): Integration in finite terms of functions build out of basic arithmetic operations, roots, powers and elementary standard functions is decidable.

Example (Maple):

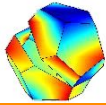
$$\text{a) } \int x e^{x^2} dx = \frac{1}{2} e^{x^2} + C$$



Back

Close

Proofs with the aid of computers (cont'd)



4/100

Risch (1970): Integration in finite terms of functions build out of basic arithmetic operations, roots, powers and elementary standard functions is decidable.

Example (Maple):

$$\text{a) } \int x e^{x^2} dx = \frac{1}{2} e^{x^2} + C$$

$$\text{b) } \int e^{x^2} dx \quad \text{not integrable}$$

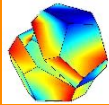
$$[= -\frac{1}{2} \sqrt{-\pi} \cdot \operatorname{erf}(\sqrt{-1} \cdot x) + C]$$



Back

Close

Proofs with the aid of computers (cont'd)



4/100

Risch (1970): Integration in finite terms of functions build out of basic arithmetic operations, roots, powers and elementary standard functions is decidable.

Example (Maple):

$$\text{a) } \int x e^{x^2} dx = \frac{1}{2} e^{x^2} + C$$

$$\text{b) } \int e^{x^2} dx \quad \text{not integrable}$$

$$[= -\frac{1}{2} \sqrt{-\pi} \cdot \operatorname{erf}(\sqrt{-1} \cdot x) + C]$$

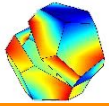
Computer algebra: [never-failing algorithms](#)



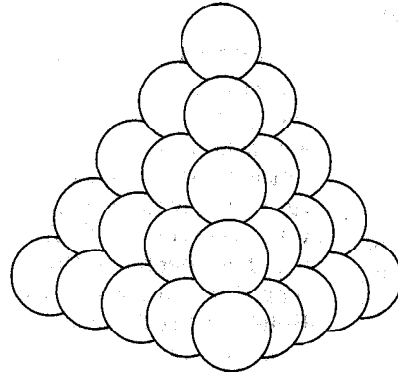
Back

Close

Proofs with the aid of computers (cont'd)



Conjecture (Kepler 1611): The face-centered cubic packing is the densest packing of equally sized balls.



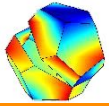
5/100



Back

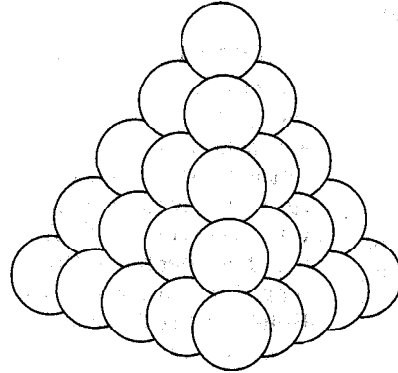
Close

Proofs with the aid of computers (cont'd)



5/100

Conjecture (Kepler 1611): The face-centered cubic packing is the densest packing of equally sized balls.

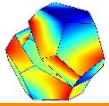


Proof (Hale 1998): by (numerical but validated) estimation the optimal value of certain nonlinear global optimization problems.



Back

Close



6/100

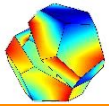
What is a proof?



Back

Close

1903 meeting of the American Mathematical Society,
"talk" by Frank N. Cole:



7/100

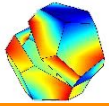
$$\begin{aligned} &147573952589676412927 \\ = &761838257287 \cdot 193707721 \\ = &2^{67} - 1 \end{aligned}$$



Back

Close

1903 meeting of the American Mathematical Society,
"talk" by Frank N. Cole:



7/100

$$\begin{aligned} & 147573952589676412927 \\ = & 761838257287 \cdot 193707721 \\ = & 2^{67} - 1 \end{aligned}$$

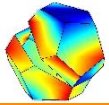
[2008 : $M_{46} = 2^{43112609} - 1$ is prime, 12 978 189 decimal digits]



Back

Close

Mathematical proofs:



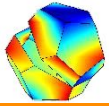
8/100



Back

Close

Mathematical proofs:



8/100

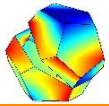
- Use of pocket calculators: widely accepted



Back

Close

Mathematical proofs:



8/100

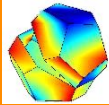
- Use of pocket calculators: widely accepted
- Use of Computer algebra systems (mostly integer arithmetic):
accepted?



Back

Close

Mathematical proofs:



8/100

- Use of pocket calculators: widely accepted
- Use of Computer algebra systems (mostly integer arithmetic): accepted?
- Use of floating-point arithmetic: questioned?

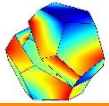


Back

Close

Pocket calculators

8 decimal digits, no exponents



9/100



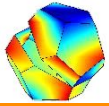
Back

Close

Pocket calculators

8 decimal digits, no exponents

10 000 000



9/100



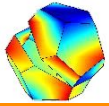
Back

Close

Pocket calculators

8 decimal digits, no exponents

$$\begin{array}{r} 10\ 000\ 000 \\ -\ 9\ 999\ 999.9 \\ \hline \end{array}$$



9/100



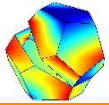
Back

Close

Pocket calculators

8 decimal digits, no exponents

$$\begin{array}{r} 10\ 000\ 000 \\ -\ 9\ 999\ 999.9 \\ \hline 0.1 \end{array}$$



9/100



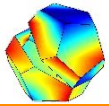
Back

Close

Pocket calculators

8 decimal digits, no exponents

$$\begin{array}{r} 10\ 000\ 000\ | \\ -\ 9\ 999\ 999\ |9 \\ \hline \end{array}$$



10/100



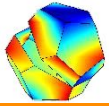
Back

Close

Pocket calculators

8 decimal digits, no exponents

$$\begin{array}{r} 10\ 000\ 000\ | \\ -\ 9\ 999\ 999\ |9 \\ \hline 1\ |0 \end{array}$$



11/100



Back

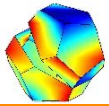
Close

Pocket calculators

8 decimal digits, no exponents

$$\begin{array}{r} 10\ 000\ 000\ | \\ -\ 9\ 999\ 999\ |9 \\ \hline 1\ |0 \end{array}$$

Accumulator same length as input precision



11/100

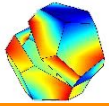


Back

Close

Floating-point arithmetic

even worse?



12/100



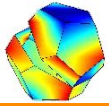
Back

Close

Floating-point arithmetic

even worse?

Pentium bug?



12/100



Back

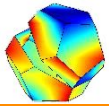
Close

Floating-point arithmetic

even worse?

Pentium bug?

Things happen



12/100



Back

Close

Floating-point arithmetic

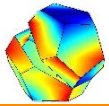
even worse?

Pentium bug?

Things happen

But...

There is a precise definition of what should happen
[in contrast to pocket calculators]



12/100



Back

Close

Floating-point arithmetic

even worse?

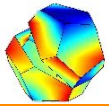
Pentium bug?

Things happen

But...

There is a precise definition of what should happen
[in contrast to pocket calculators]

⇒ IEEE 754 floating-point standard



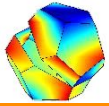
12/100



Back

Close

Floating-point arithmetic



13/100

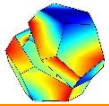


Back

Close

Floating-point arithmetic

IEEE 754 arithmetic standard (double precision)



13/100



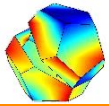
Back

Close

Floating-point arithmetic

IEEE 754 arithmetic standard (double precision)

$$f = \pm 1.m_1m_2 \dots m_{52} \cdot 2^e$$



13/100



Back

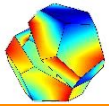
Close

Floating-point arithmetic

IEEE 754 arithmetic standard (double precision)

$$f = \pm 1.m_1m_2 \dots m_{52} \cdot 2^e$$

Extra quantities $\pm \infty$, NaN



13/100



Back

Close

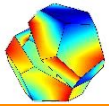
Floating-point arithmetic

IEEE 754 arithmetic standard (double precision)

$$f = \pm 1.m_1m_2\dots m_{52} \cdot 2^e$$

Extra quantities $\pm \infty$, NaN

Example: $1/10 = 0.000110011001100\dots$ binary



13/100



Back

Close

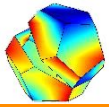
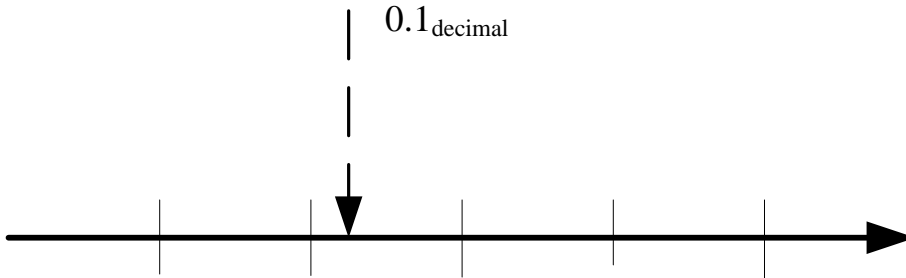
Floating-point arithmetic

IEEE 754 arithmetic standard (double precision)

$$f = \pm 1.m_1m_2 \dots m_{52} \cdot 2^e$$

Extra quantities $\pm \infty$, NaN

Example: $1/10 = 0.000110011001100 \dots$ binary



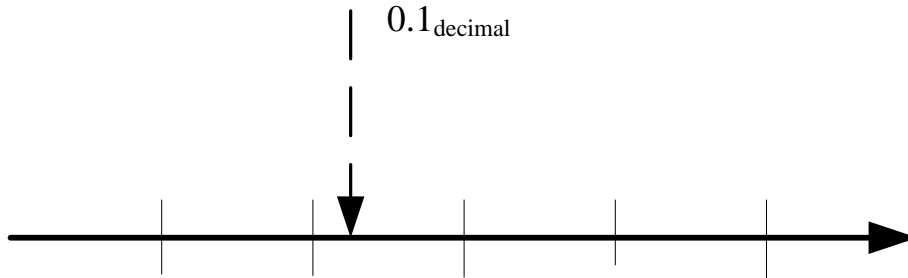
Floating-point arithmetic

IEEE 754 arithmetic standard (double precision)

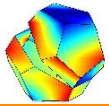
$$f = \pm 1.m_1m_2 \dots m_{52} \cdot 2^e$$

Extra quantities $\pm \infty$, NaN

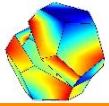
Example: $1/10 = 0.000110011001100 \dots$ binary



rounding to nearest $1 \boxed{/} 10$



Directed roundings



14/100

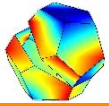


Back

Close

Directed roundings

$1 \nabla 10$ the largest floating-point number $f_1 \leq 1/10$



14/100



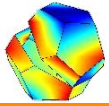
Back

Close

Directed roundings

$1 \nabla 10$ the largest floating-point number $f_1 \leq 1/10$

$1 \triangle 10$ the smallest floating-point number $1/10 \leq f_2$



14/100



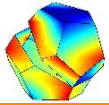
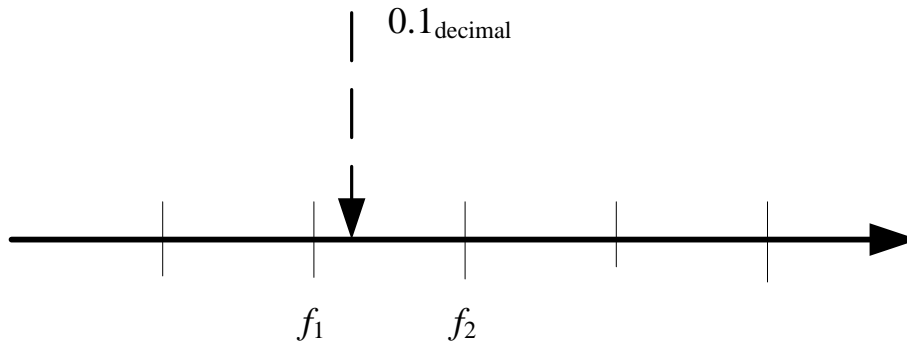
Back

Close

Directed roundings

$1 \nabla 10$ the largest floating-point number $f_1 \leq 1/10$

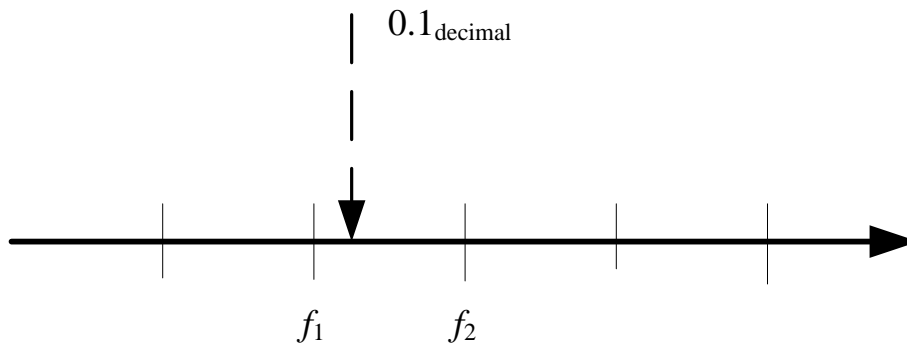
$1 \triangle 10$ the smallest floating-point number $1/10 \leq f_2$



Directed roundings

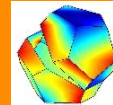
$1 \nabla 10$ the largest floating-point number $f_1 \leq 1/10$

$1 \triangle 10$ the smallest floating-point number $1/10 \leq f_2$



With mathematical certainty always

$$f_1 \leq 1/10 \leq f_2$$

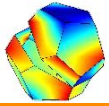


Directed roundings (cont'd)

More general $a, b \in \mathbb{F}$:

$$a \nabla b \leq a \circ b \leq a \triangle b$$

for $\circ \in \{+, -, \cdot, /\}$



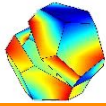
15/100



Back

Close

Directed roundings (cont'd)



15/100

More general $a, b \in \mathbb{F}$:

$$a \nabla b \leq a \circ b \leq a \triangle b$$

for $\circ \in \{+, -, \cdot, /\}$

Theorem. For all $a, b \in \mathbb{F}$

$$a \nabla b = a \triangle b \iff a \circ b \in \mathbb{F}.$$

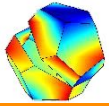


Back

Close

Directed roundings on the computer

INTLAB — The Matlab toolbox for Reliable Computing



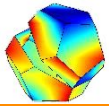
16/100



Back

Close

Directed roundings on the computer



INTLAB — The Matlab toolbox for Reliable Computing

16/100

`setround(-1)` rounding downwards

`setround(1)` rounding upwards

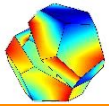
`setround(0)` rounding to nearest



Back

Close

Directed roundings on the computer



INTLAB — The Matlab toolbox for Reliable Computing

16/100

`setround(-1)` rounding downwards

`setround(1)` rounding upwards

`setround(0)` rounding to nearest

```
setround(-1)
```

```
x = 2/3
```

```
setround(1)
```

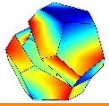
```
y = 2/3
```



Back

Close

Directed roundings on the computer



INTLAB — The Matlab toolbox for Reliable Computing

16/100

`setround(-1)` rounding downwards

`setround(1)` rounding upwards

`setround(0)` rounding to nearest

`setround(-1)`

`x = 2/3`

`setround(1)`

`y = 2/3`

$\Rightarrow x \leq \frac{2}{3} \leq y$ guaranteed

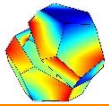


Back

Close

Interval arithmetic

$$A = [a_1, a_2] := \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$



17/100



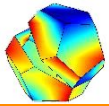
Back

Close

Interval arithmetic

$$A = [a_1, a_2] := \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

Note: a_1, a_2 floating-point numbers, but
 A continuous interval



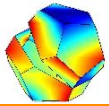
17/100



Back

Close

Interval arithmetic



17/100

$$A = [a_1, a_2] := \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

Note: a_1, a_2 floating-point numbers, but
 A continuous interval

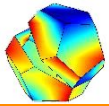
Example: $[2.5, 3.25] = \{x \in \mathbb{R} : 2.5 \leq x \leq 3.25\}$



Back

Close

Interval arithmetic



17/100

$$A = [a_1, a_2] := \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

Note: a_1, a_2 floating-point numbers, but
 A continuous interval

Example: $[2.5, 3.25] = \{x \in \mathbb{R} : 2.5 \leq x \leq 3.25\}$

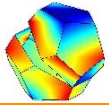
$A + B$ “the set of all results”



Back

Close

Interval addition/subtraction



18/100



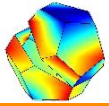
Back

Close

Interval addition/subtraction

$$A = [a_1, a_2] = \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

$$B = [b_1, b_2] = \{y \in \mathbb{R} : b_1 \leq y \leq b_2\}$$



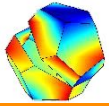
18/100



Back

Close

Interval addition/subtraction



18/100

$$A = [a_1, a_2] = \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

$$B = [b_1, b_2] = \{y \in \mathbb{R} : b_1 \leq y \leq b_2\}$$

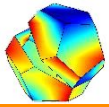
$$\Rightarrow a_1 + b_1 \leq x + y \leq a_2 + b_2$$



Back

Close

Interval addition/subtraction



18/100

$$A = [a_1, a_2] = \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

$$B = [b_1, b_2] = \{y \in \mathbb{R} : b_1 \leq y \leq b_2\}$$

$$\Rightarrow a_1 + b_1 \leq x + y \leq a_2 + b_2$$

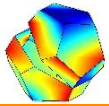
$$\Rightarrow A + B = [a_1 + b_1, a_2 + b_2]$$



Back

Close

Interval addition/subtraction



18/100

$$A = [a_1, a_2] = \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

$$B = [b_1, b_2] = \{y \in \mathbb{R} : b_1 \leq y \leq b_2\}$$

$$\Rightarrow a_1 + b_1 \leq x + y \leq a_2 + b_2$$

$$\Rightarrow A + B = [a_1 + b_1, a_2 + b_2]$$

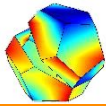
$$-B = \{-y : y \in B\} = [-b_2, -b_1]$$



Back

Close

Interval addition/subtraction



18/100

$$A = [a_1, a_2] = \{x \in \mathbb{R} : a_1 \leq x \leq a_2\}$$

$$B = [b_1, b_2] = \{y \in \mathbb{R} : b_1 \leq y \leq b_2\}$$

$$\Rightarrow a_1 + b_1 \leq x + y \leq a_2 + b_2$$

$$\Rightarrow A + B = [a_1 + b_1, a_2 + b_2]$$

$$-B = \{-y : y \in B\} = [-b_2, -b_1]$$

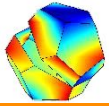
$$\Rightarrow A - B = A + (-B) = [a_1 - b_2, a_2 - b_1]$$



Back

Close

Quality of the result



19/100

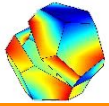


Back

Close

Quality of the result

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$



19/100



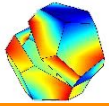
Back

Close

Quality of the result

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$



19/100



Back

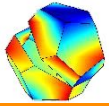
Close

Quality of the result

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$

$$\text{diam}(A - B) = \text{diam}([a_1 - b_2, a_2 - b_1])$$



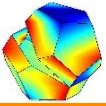
19/100



Back

Close

Quality of the result



19/100

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$

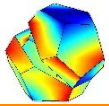
$$\begin{aligned}\text{diam}(A - B) &= \text{diam}([a_1 - b_2, a_2 - b_1]) \\ &= (a_2 - b_1) - (a_1 - b_2)\end{aligned}$$



Back

Close

Quality of the result



19/100

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$

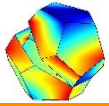
$$\begin{aligned}\text{diam}(A - B) &= \text{diam}([a_1 - b_2, a_2 - b_1]) \\ &= (a_2 - b_1) - (a_1 - b_2) \\ &= (a_2 - a_1) + (b_2 - b_1)\end{aligned}$$



Back

Close

Quality of the result



19/100

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$

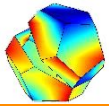
$$\begin{aligned}\text{diam}(A - B) &= \text{diam}([a_1 - b_2, a_2 - b_1]) \\ &= (a_2 - b_1) - (a_1 - b_2) \\ &= (a_2 - a_1) + (b_2 - b_1) \\ &= \text{diam}(A) + \text{diam}(B)\end{aligned}$$



Back

Close

Quality of the result



19/100

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$

$$\begin{aligned} \text{diam}(A - B) &= \text{diam}([a_1 - b_2, a_2 - b_1]) \\ &= (a_2 - b_1) - (a_1 - b_2) \\ &= (a_2 - a_1) + (b_2 - b_1) \\ &= \text{diam}(A) + \text{diam}(B) \end{aligned}$$

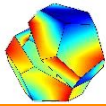
$$\Rightarrow \text{diam}(A \pm B) = \text{diam}(A) + \text{diam}(B)$$



Back

Close

Quality of the result



19/100

$$\text{diam}(A) = \text{diam}([a_1, a_2]) = a_2 - a_1$$

Obviously $\text{diam}(A + B) = \text{diam}(A) + \text{diam}(B)$

$$\begin{aligned}\text{diam}(A - B) &= \text{diam}([a_1 - b_2, a_2 - b_1]) \\ &= (a_2 - b_1) - (a_1 - b_2) \\ &= (a_2 - a_1) + (b_2 - b_1) \\ &= \text{diam}(A) + \text{diam}(B)\end{aligned}$$

$$\Rightarrow \text{diam}(A \pm B) = \text{diam}(A) + \text{diam}(B)$$

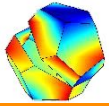
Careful: Intervals may become wide



Back

Close

Interval multiplication/division



20/100

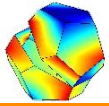


Back

Close

Interval multiplication/division

$$[2, 3] * [4, 5] = [8, 15]$$



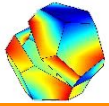
20/100



Back

Close

Interval multiplication/division



20/100

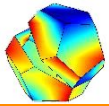
$$\begin{aligned} [2, 3] * [4, 5] &= [8, 15] \\ [-2, 3] * [4, 5] &= [-10, 15] \end{aligned}$$



Back

Close

Interval multiplication/division



20/100

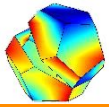
$$\begin{aligned} [2, 3] * [4, 5] &= [8, 15] \\ [-2, 3] * [4, 5] &= [-10, 15] \\ [2, 3] * [-4, 5] &= [-12, 15] \end{aligned}$$



Back

Close

Interval multiplication/division



20/100

$$\begin{aligned}[2, 3] * [4, 5] &= [8, 15] \\ [-2, 3] * [4, 5] &= [-10, 15] \\ [2, 3] * [-4, 5] &= [-12, 15]\end{aligned}$$

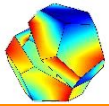
$$[a_1, a_2] * [b_1, b_2] = \min / \max(a_1b_1, a_1b_2, a_2b_1, a_2b_2)$$



Back

Close

Interval multiplication/division



20/100

$$\begin{aligned} [2, 3] * [4, 5] &= [8, 15] \\ [-2, 3] * [4, 5] &= [-10, 15] \\ [2, 3] * [-4, 5] &= [-12, 15] \end{aligned}$$

$$[a_1, a_2] * [b_1, b_2] = \min / \max(a_1b_1, a_1b_2, a_2b_1, a_2b_2)$$

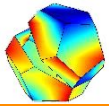
Details and improvements in INTLAB



Back

Close

A more sophisticated example



21/100

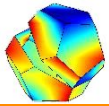


Back

Close

A more sophisticated example

$$\text{res} = a * b + c * d$$



21/100



Back

Close

A more sophisticated example

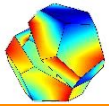
$$\text{res} = a * b + c * d$$

transforms into

$$f_1 = a * b$$

$$f_2 = c * d$$

$$\text{res} = f_1 + f_2$$



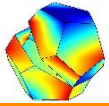
21/100



Back

Close

A more sophisticated example



21/100

$$\mathbf{res} = a * b + c * d$$

transforms into

$$f_1 = a * b$$

$$f_2 = c * d$$

$$\mathbf{res} = f_1 + f_2$$

setround(-1)

$$f_1 \leq ab$$

$$f_2 \leq cd$$

$$\mathbf{res} \leq f_1 + f_2 \leq ab + cd$$

setround(1)

$$ab \leq f_1$$

$$cd \leq f_2$$

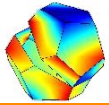
$$ab + cd \leq f_1 + f_2 \leq \mathbf{res}$$



Back

Close

Be careful



22/100

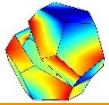


Back

Close

Be careful

$$\text{res} = a * b - c * d$$



22/100



Back

Close

Be careful

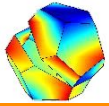
$$\text{res} = a * b - c * d$$

transforms into

$$f_1 = a * b$$

$$f_2 = c * d$$

$$\text{res} = f_1 - f_2$$



22/100



Back

Close

Be careful

$$\text{res} = a * b - c * d$$

transforms into

$$f_1 = a * b$$

$$f_2 = c * d$$

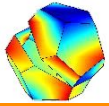
$$\text{res} = f_1 - f_2$$

setround(-1)

$$f_1 \leq ab$$

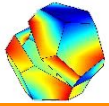
$$f_2 \leq cd$$

$$\text{res} \leq f_1 - f_2 \not\leq ab - cd$$



Dot products

x, y vectors, $d = x^T y$



23/100



Back

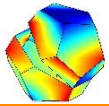
Close

Dot products

x, y vectors, $d = x^T y$

$$d_1 = x_1 y_1, d_2 = x_2 y_2, \dots, d_n = x_n y_n$$

$$\text{res} = d_1 + d_2 + \dots + d_n$$



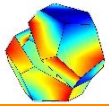
23/100



Back

Close

Dot products



23/100

x, y vectors, $d = x^T y$

$d_1 = x_1 y_1, d_2 = x_2 y_2, \dots, d_n = x_n y_n$

$\text{res} = d_1 + d_2 + \dots + d_n$

```
setround(-1)
```

```
dlow = x' * y
```

```
setround(1)
```

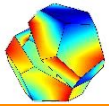
```
dup = x' * y
```



Back

Close

Dot products



23/100

x, y vectors, $d = x^T y$

$d_1 = x_1 y_1, d_2 = x_2 y_2, \dots, d_n = x_n y_n$

$\text{res} = d_1 + d_2 + \dots + d_n$

`setround(-1)`

`dlow = x' * y`

`setround(1)`

`dup = x' * y`

$\Rightarrow \quad \text{dlow} \leq x^T y \leq \text{dup}$



Back

Close

Matrix multiplication

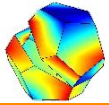
Given matrices A, B

```
setround(-1)
```

```
Cinf = A * B
```

```
setround(1)
```

```
Csup = A * B
```



24/100



Back

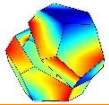
Close

Matrix multiplication

Given matrices A, B

```
setround(-1)
Cinf = A * B
setround(1)
Csup = A * B
```

$\Rightarrow C_{\text{inf}} \leq AB \leq C_{\text{sup}}$ entrywise



24/100



Back

Close

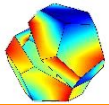
Matrix multiplication

Given matrices A, B

```
setround(-1)
Cinf = A * B
setround(1)
Csup = A * B
```

$$\Rightarrow \quad C_{\text{inf}} \leq AB \leq C_{\text{sup}} \quad \text{entrywise}$$

fast!



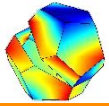
24/100



Back

Close

Computation with real numbers



25/100

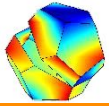
$$\pi \in [3.14159, 3.14160]$$



Back

Close

Computation with real numbers



25/100

$$\pi \in [3.14159, 3.14160]$$

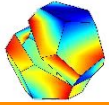
Definitely π is not a floating-point number



Back

Close

Computation with real numbers



25/100

$$\pi \in [3.14159, 3.14160]$$

Definitely π is not a floating-point number

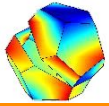
$$\pi + 2 \in [5.14159, 5.14160]$$



Back

Close

Computation with real numbers



25/100

$$\pi \in [3.14159, 3.14160]$$

Definitely π is not a floating-point number

$$\pi + 2 \in [5.14159, 5.14160]$$

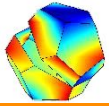
$$\pi^2 \in [9.8695877281, 9.8696505600]$$



Back

Close

Computation with real numbers



25/100

$$\pi \in [3.14159, 3.14160]$$

Definitely π is not a floating-point number

$$\pi + 2 \in [5.14159, 5.14160]$$

$$\pi^2 \in [9.8695877281, 9.8696505600]$$

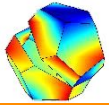
Safe, mathematically correct computations with real numbers



Back

Close

The range of a function



26/100

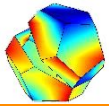


Back

Close

The range of a function

$$f(x) = 3x - \frac{3x^3 - x^2 - 3x - 2}{x + 3}$$



26/100



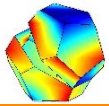
Back

Close

The range of a function

$$f(x) = 3x - \frac{3x^3 - x^2 - 3x - 2}{x + 3}$$

What is the range of f over $X = [0, 1]$?



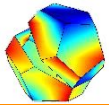
26/100



Back

Close

The range of a function



26/100

$$f(x) = 3x - \frac{3x^3 - x^2 - 3x - 2}{x + 3}$$

What is the range of f over $X = [0, 1]$?

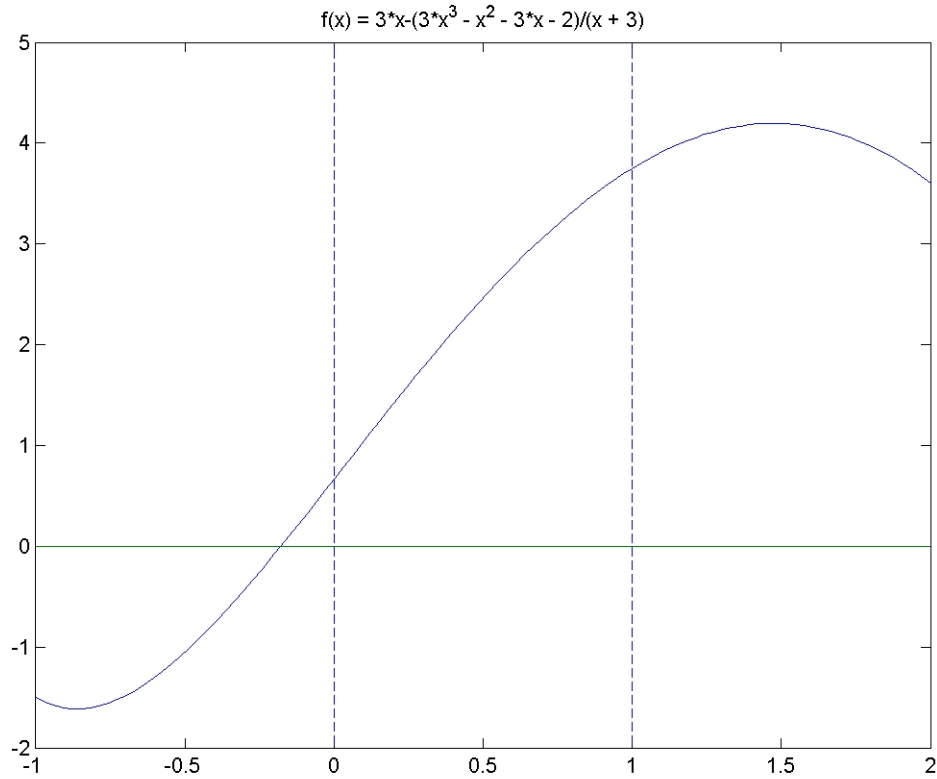
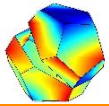
```
f = inline('3*x-(3*x^1 3 - x^1 2 - 3*x - 2)/(x + 3)')
X = infsup(0,1)
Y_1 = f(X)
```

```
Y_1 =
[-0.3334, 5.0000]
```



Back

Close



Back

Close

The range of a function

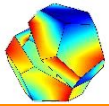
$$f(x) = 3x - \frac{3x^3 - x^2 - 3x - 2}{x + 3}$$

What is the range of f over $X = [-1, 2]$?

$$Y_2 = f(\text{infsup}(-1, 2))$$

$$Y_2 =$$

$$[-15.5000, 13.5000]$$



28/100



Back

Close

The range of a function

$$f(x) = 3x - \frac{3x^3 - x^2 - 3x - 2}{x + 3}$$

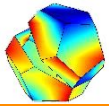
What is the range of f over $X = [-1, 2]$?

$$Y_2 = f(\text{infsup}(-1, 2))$$

$$Y_2 =$$

$$[-15.5000, 13.5000]$$

True, but significant overestimation.



28/100



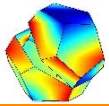
Back

Close

Self-validating methods — a simple example

Theory Let A be a matrix, R be some preconditioner

$$\|I - RA\| < 1 \quad \Rightarrow \quad A \text{ nonsingular}$$



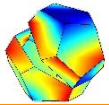
29/100



Back

Close

Self-validating methods — a simple example



Theory Let A be a matrix, R be some preconditioner

$$\|I - RA\| < 1 \quad \Rightarrow \quad A \text{ nonsingular}$$

29/100

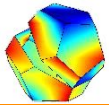
Proof Suppose A is singular. Then $Ax = 0$ for a vector $x \neq 0$.



Back

Close

Self-validating methods — a simple example



Theory Let A be a matrix, R be some preconditioner

$$\|I - RA\| < 1 \quad \Rightarrow \quad A \text{ nonsingular}$$

29/100

Proof Suppose A is singular. Then $Ax = 0$ for a vector $x \neq 0$.

Then

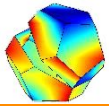
$$(I - RA)x = x$$



Back

Close

Self-validating methods — a simple example



Theory Let A be a matrix, R be some preconditioner

$$\|I - RA\| < 1 \quad \Rightarrow \quad A \text{ nonsingular}$$

29/100

Proof Suppose A is singular. Then $Ax = 0$ for a vector $x \neq 0$.

Then

$$(I - RA)x = x$$

so that $I - RA$ has an eigenvalue 1. But then

$$\|I - RA\| \geq 1.$$

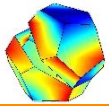
□



Back

Close

Self-validating methods — a simple example



29/100

Theory Let A be a matrix, R be some preconditioner

$$\|I - RA\| < 1 \quad \Rightarrow \quad A \text{ nonsingular}$$

Proof Suppose A is singular. Then $Ax = 0$ for a vector $x \neq 0$.

Then

$$(I - RA)x = x$$

so that $I - RA$ has an eigenvalue 1. But then

$$\|I - RA\| \geq 1. \quad \square$$

Practice Choose an approximate inverse R and verify

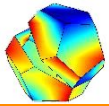
$$(*) \quad \|I - RA\| < 1$$



Back

Close

Self-validating methods — a simple example



29/100

Theory Let A be a matrix, R be some preconditioner

$$\|I - RA\| < 1 \quad \Rightarrow \quad A \text{ nonsingular}$$

Proof Suppose A is singular. Then $Ax = 0$ for a vector $x \neq 0$.

Then

$$(I - RA)x = x$$

so that $I - RA$ has an eigenvalue 1. But then

$$\|I - RA\| \geq 1. \quad \square$$

Practice Choose an approximate inverse R and verify

$$(*) \quad \|I - RA\| < 1$$

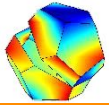
Verification Verify $(*)$ using INTLAB



Back

Close

Verification of nonsingularity using INTLAB



30/100

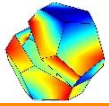


Back

Close

Verification of nonsingularity using INTLAB

Given a matrix A of dimension n ,



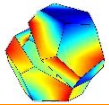
30/100



Back

Close

Verification of nonsingularity using INTLAB



Given a matrix A of dimension n ,

```
R = inv(A)
C = eye(n) - R*intval(A)
nonsingular = ( norm(C,1) < 1 )
```

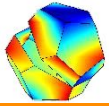
30/100



Back

Close

Verification of nonsingularity using INTLAB



30/100

Given a matrix A of dimension n ,

```
R = inv(A)
C = eye(n) - R*intval(A)
nonsingular = ( norm(C,1) < 1 )
```

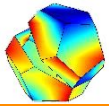
If nonsingular = 1, then nonsingularity of A is **proved**.



Back

Close

Verification of nonsingularity using INTLAB



30/100

Given a matrix A of dimension n ,

```
R = inv(A)
C = eye(n) - R*intval(A)
nonsingular = ( norm(C,1) < 1 )
```

If nonsingular = 1, then nonsingularity of A is **proved**.

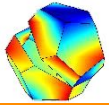
If nonsingular = 0, then we don't know



Back

Close

Self-validating methods - more sophisticated



31/100

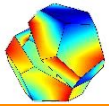


Back

Close

Self-validating methods - more sophisticated

Theory Let A be a real or complex matrix, R be some preconditioner.



31/100



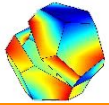
Back

Close

Self-validating methods - more sophisticated

Theory Let A be a real or complex matrix, R be some preconditioner.

If the spectral radius $\rho(I - RA)$ is less than 1, then A (and R) are nonsingular.



31/100



Back

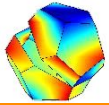
Close

Self-validating methods - more sophisticated

Theory Let A be a real or complex matrix, R be some preconditioner.

If the spectral radius $\rho(I - RA)$ is less than 1, then A (and R) are nonsingular.

By Perron-Frobenius Theory, $\rho(I - RA) \leq \rho(|I - RA|)$, and $|I - RA| \cdot x < x$ for a positive vector x implies $\rho(|I - RA|) < 1$



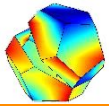
31/100



Back

Close

Self-validating methods - more sophisticated



31/100

Theory Let A be a real or complex matrix, R be some preconditioner.

If the spectral radius $\rho(I - RA)$ is less than 1, then A (and R) are nonsingular.

By Perron-Frobenius Theory, $\rho(I - RA) \leq \rho(|I - RA|)$, and $|I - RA| \cdot x < x$ for a positive vector x implies $\rho(|I - RA|) < 1$

Practice Choose numerical approximations for R and x , and verify

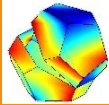
$$|I - RA| \cdot x < x \quad (*)$$



Back

Close

Self-validating methods - more sophisticated



31/100

Theory Let A be a real or complex matrix, R be some preconditioner.

If the spectral radius $\rho(I - RA)$ is less than 1, then A (and R) are nonsingular.

By Perron-Frobenius Theory, $\rho(I - RA) \leq \rho(|I - RA|)$, and $|I - RA| \cdot x < x$ for a positive vector x implies $\rho(|I - RA|) < 1$

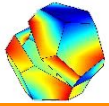
Practice Choose numerical approximations for R and x , and verify

$$|I - RA| \cdot x < x \quad (*)$$

Implementation Verify $(*)$ in floating-point



Verification of nonsingularity



32/100

$$C_1 = \nabla(R \cdot A - I), \quad C_2 = \Delta(R \cdot A - I)$$

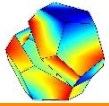
$$[\Rightarrow C_1 \leq RA - I \leq C_2]$$



Back

Close

Verification of nonsingularity



32/100

$$C_1 = \nabla(R \cdot A - I), \quad C_2 = \Delta(R \cdot A - I)$$

$$[\Rightarrow C_1 \leq RA - I \leq C_2]$$

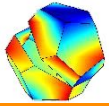
$$C = \max(|C_1|, |C_2|) \quad (\text{componentwise})$$



Back

Close

Verification of nonsingularity



32/100

$$C_1 = \nabla(R \cdot A - I), \quad C_2 = \Delta(R \cdot A - I)$$

$$[\Rightarrow C_1 \leq RA - I \leq C_2]$$

$$C = \max(|C_1|, |C_2|) \quad (\text{componentwise})$$

$$\Delta(C \cdot x) < x \quad \text{for } 0 < x \in \mathbb{F}^n$$

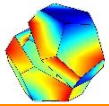
[implies A and R are nonsingular]



Back

Close

Verification of nonsingularity



32/100

$$C_1 = \nabla(R \cdot A - I), \quad C_2 = \Delta(R \cdot A - I)$$

$$[\Rightarrow C_1 \leq RA - I \leq C_2]$$

$$C = \max(|C_1|, |C_2|) \quad (\text{componentwise})$$

$$\Delta(C \cdot x) < x \quad \text{for } 0 < x \in \mathbb{F}^n$$

[implies A and R are nonsingular]

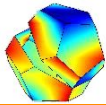
- proves all $\tilde{A} \in U_\varepsilon(A)$ to be nonsingular



Back

Close

Verification of nonsingularity



32/100

$$C_1 = \nabla(R \cdot A - I), \quad C_2 = \Delta(R \cdot A - I)$$

$$[\Rightarrow C_1 \leq RA - I \leq C_2]$$

$$C = \max(|C_1|, |C_2|) \quad (\text{componentwise})$$

$$\Delta(C \cdot x) < x \quad \text{for } 0 < x \in \mathbb{F}^n$$

[implies A and R are nonsingular]

- proves all $\tilde{A} \in U_\varepsilon(A)$ to be nonsingular
- "Rounded operations" create speed with rigour!



Back

Close

A convenient implementation

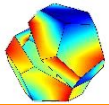
The INTerval LABoratory (INTLAB) is a free and fast toolbox entirely written under Matlab.

It **uses** interval arithmetic (by no means mandatory for SV-methods)

Sample implementation of previous algorithm

(Check $|I-RA| \ x < x$)

```
>> R = inv(A);  
C = abs(eye(n)-R*intval(A));  
x = ones(n,1);  
setround(+1)  
nonsingular = ( C*x < x )
```



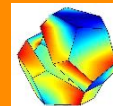
33/100



Back

Close

Computer Algebra vs. Self-validating methods



34/100

Computer Algebra algorithms yield **exact** answers

- Integration in finite terms
- Quantifier elimination
- Gröbner bases

... ..

SV-methods only applicable to **well-posed** problems, e.g.

- prove **nonsingularity** of matrix
- error bounds for cluster of zeros

... ..

not applicable to

- prove singularity of matrix
- prove multiplicity > 1

... ..

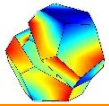
because these problems are **ill-posed**:

arbitrary small perturbations of input data may
change answer

[speed of SV-methods is created by "inaccuracy" of computation]



Self-validating methods with INTLAB



Verify assumptions of mathematical theorems on the computer.

35/100

Objectives:

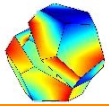
- computation of valid bounds for solution
- possibly with proof of uniqueness
- valid in mathematical sense
(including all possible errors)
- fast



Back

Close

Self-validating methods with INTLAB



35/100

Verify assumptions of mathematical theorems on the computer.

Objectives:

- computation of valid bounds for solution
- possibly with proof of uniqueness
- valid in mathematical sense
(including all possible errors)
- fast

INTLAB — The Matlab toolbox for Reliable Computing



Back

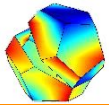
Close

Bounds for the range of a function

Griewank function

```
G = inline('(x^2+y^2)/4000+cos(x)*cos(y)/sqrt{2}+1')
```

over $-60 \leq x, y \leq 60$



36/100



Back

Close

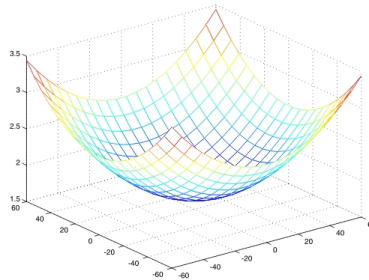
Bounds for the range of a function

Griewank function

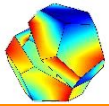
```
G = inline('(x^2+y^2)/4000+cos(x)*cos(y)/sqrt{2}+1')
```

over $-60 \leq x, y \leq 60$

A plot with 20 meshpoints in each coordinate:



$$1.7119 \leq G(x, y)$$



36/100



Back

Close

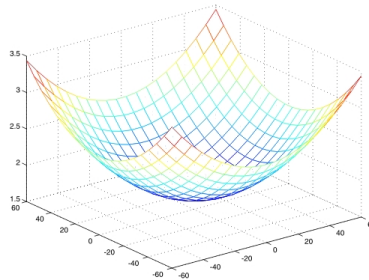
Bounds for the range of a function

Griewank function

```
G = inline('(x^2+y^2)/4000+cos(x)*cos(y)/sqrt{2}+1')
```

over $-60 \leq x, y \leq 60$

A plot with 20 meshpoints in each coordinate:

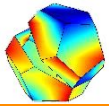


$$1.7119 \leq G(x, y)$$

```
>> X = infsup(-60,60); Y = X; G(X,Y)
```

```
inval ans =
```

```
[ 0.2928, 3.5072]
```



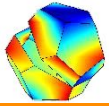
36/100



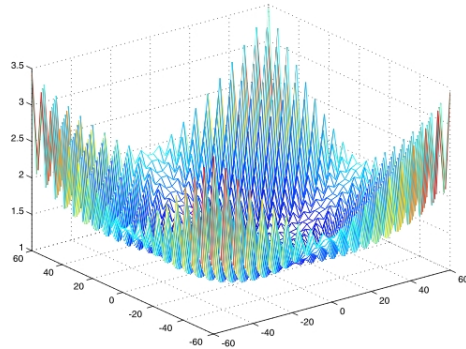
Back

Close

A plot with 40 meshpoints in each coordinate:



37/100



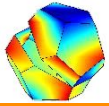
$$1.0019 \leq G(x, y)$$



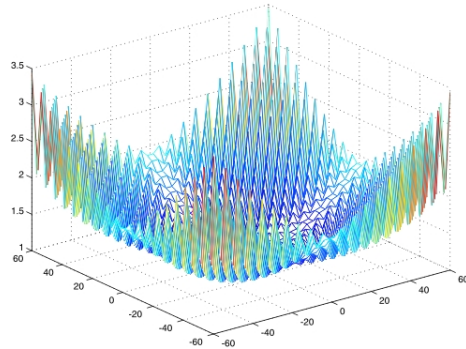
Back

Close

A plot with 40 meshpoints in each coordinate:



37/100



$$1.0019 \leq G(x, y)$$

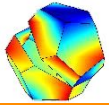
”true” minimum ~ 0.2957 (2000 meshpoints in each coordinate)
computing time 16 sec



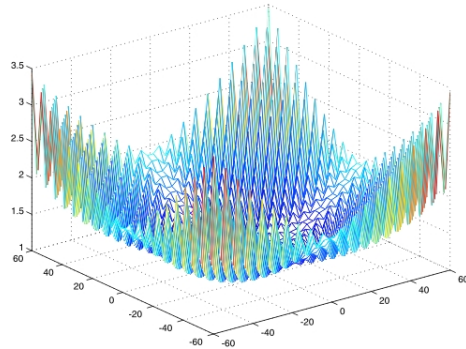
Back

Close

A plot with 40 meshpoints in each coordinate:



37/100



$$1.0019 \leq G(x, y)$$

”true” minimum ~ 0.2957 (2000 meshpoints in each coordinate)
computing time 16 sec

vs. $[0.2928, 3.5072]$ by INTLAB
computing time 0.03 sec

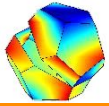


Back

Close

Common misuse — an example

Consider $X_{n+1} = 2 - 2x_n - 3x_{n-1}$ for $x_1 = x_2 = 1/3$



38/100



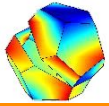
Back

Close

Common misuse — an example

Consider $x_{n+1} = 2 - 2x_n - 3x_{n-1}$ for $x_1 = x_2 = 1/3$

Obviously $x_i \equiv 1/3$ for all $i \geq 1$



38/100



Back

Close

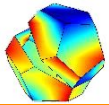
Common misuse — an example

Consider $x_{n+1} = 2 - 2x_n - 3x_{n-1}$ for $x_1 = x_2 = 1/3$

Obviously $x_i \equiv 1/3$ for all $i \geq 1$

However, $1/3$ is not a floating-point number

$$\text{Hence } x_3 = 2 - 2 \square 2 \square x_2 - 3 \square x_1 \neq 1/3$$



38/100



Back

Close

Common misuse — an example

Consider $X_{n+1} = 2 - 2x_n - 3x_{n-1}$ for $x_1 = x_2 = 1/3$

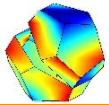
Obviously $x_i \equiv 1/3$ for all $i \geq 1$

However, $1/3$ is not a floating-point number

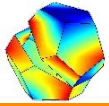
$$\text{Hence } x_3 = 2 - 2 \cdot 1/3 - 3 \cdot 1/3 \neq 1/3$$

Result in floating-point (up to $n = 30$):

```
ans =  
0.333333333333333  
0.333333333333333  
0.333333333333333  
0.333333333333333  
0.333333333333333  
... ..  
0.333333333378139  
0.333333333241904  
0.333333333381774  
0.333333333510738  
0.333333332833201
```



Common misuse using interval arithmetic



39/100

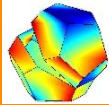
```
X(1) = intval(1/3)
X(2) = X(1)
for i = 3: 30
    X(i) = 2 - 2*X(i-1) - 3*X(i-2)
end
X
```



Back

Close

Common misuse using interval arithmetic



39/100

```
X(1) = intval(1/3)
X(2) = X(1)
for i = 3: 30
    X(i) = 2 - 2*X(i-1) - 3*X(i-2)
end
X
```

Result

```
intval =
[ 0.333333333333333, 0.333333333333334]
[ 0.333333333333333, 0.333333333333334]
[ 0.333333333333333, 0.333333333333334]
[ 0.333333333333333, 0.333333333333334]
[ 0.333333333333333, 0.333333333333334]
... ..
[ 0.33331968770671, 0.33334697902040]
[ 0.33329239632799, 0.33337427026905]
[ 0.33321052240071, 0.33345614422387]
[ 0.33296490074513, 0.33370176621459]
[ 0.33222803489921, 0.33443863130758]
```



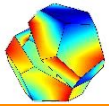
Back

Close

Formulation as a linear system

Remember $x_{n+1} = 2 - 2x_n - 3x_{n-1}$ for $x_1 = x_2 = 1/3$

$$\begin{array}{rcccccc} x_1 & & & & & & = & 1/3 \\ & x_2 & & & & & = & 1/3 \\ -3x_1 & -2x_2 & + x_3 & & & & = & 2 \\ & -3x_2 & -2x_3 & + x_4 & & & = & 2 \\ & & -3x_3 & -2x_4 & + x_5 & & = & 2 \\ & & & \dots & \dots & \dots & & \end{array}$$



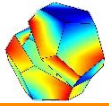
40/100



Back

Close

Formulation as a linear system



Remember $x_{n+1} = 2 - 2x_n - 3x_{n-1}$ for $x_1 = x_2 = 1/3$

40/100

$$\begin{array}{rcccccc} x_1 & & & & & & = & 1/3 \\ & x_2 & & & & & = & 1/3 \\ -3x_1 & -2x_2 & + x_3 & & & & = & 2 \\ & -3x_2 & -2x_3 & + x_4 & & & = & 2 \\ & & -3x_3 & -2x_4 & + x_5 & & = & 2 \\ & & & \dots & \dots & \dots & & \end{array}$$

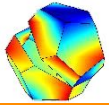
or $Ax = b$ with

$$A = \begin{pmatrix} 1 & & & & & \\ & 1 & & & & \\ -3 & -2 & 1 & & & \\ & -3 & -2 & 1 & & \\ & & -3 & -2 & 1 & \\ & & & \dots & \dots & \dots \end{pmatrix}, \quad b = \begin{pmatrix} 1/3 \\ 1/3 \\ 2 \\ 2 \\ 2 \\ \dots \end{pmatrix}$$



Verified solution of linear systems

```
verifylss(A,b)
```



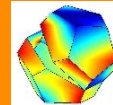
41/100



Back

Close

Verified solution of linear systems



41/100

```
verifylss(A,b)
```

```
intval  =  
[ 0.333333333333333, 0.333333333333334]  
[ 0.333333333333333, 0.333333333333334]  
[ 0.333333333333333, 0.333333333333334]  
[ 0.333333333333333, 0.333333333333334]  
[ 0.333333333333333, 0.333333333333334]  
    ... ..  
[ 0.33333333327687, 0.33333333336157]  
[ 0.33333333329299, 0.33333333341402]  
[ 0.33333333327482, 0.33333333339585]  
[ 0.33333333307524, 0.33333333346238]  
[ 0.33333333308727, 0.33333333382545]
```

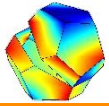


Back

Close

Common misuse - an example

$$A = \begin{pmatrix} 1 & & 0 \\ & 1 & \\ & & \ddots \\ 1 & & & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \pm 10^{-10}.$$



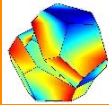
42/100



Back

Close

Common misuse - an example



42/100

$$A = \begin{pmatrix} 1 & & 0 \\ & 1 & \\ & & \dots \\ 1 & & & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \pm 10^{-10}.$$

Solution of $Ax = b$ by interval forward substitution

INTLAB code:

```
n = 10; A = tril(ones(n));
b = midrad(ones(n,1),1e-10);
X = b;
for i = 1:n
    X(i) = b(i) - A(i,1:i-1)*X(1:i-1);
end
X
```



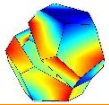
Back

Close

Common misuse - an example (cont'd)

$$A = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 1 & & & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \pm 10^{-10}.$$

Solution of $Ax = b$ by forward substitution



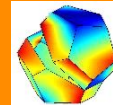
43/100



Back

Close

Common misuse - an example (cont'd)



43/100

$$A = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 1 & & & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \pm 10^{-10}.$$

Solution of $Ax = b$ by forward substitution

Result (midpoint-radius representation):

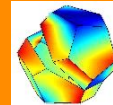
intval X =	true radius
< 1.0000e+000, 1.0001e-010>	1.0001e-10
< 0.0000e+000, 2.0001e-010>	2.0001e-10
< 0.0000e+000, 4.0001e-010>	2.0001e-10
< 0.0000e+000, 8.0001e-010>	2.0001e-10
< 0.0000e+000, 1.6001e-009>	2.0001e-10
< 0.0000e+000, 3.2001e-009>	2.0001e-10
< 0.0000e+000, 6.4001e-009>	2.0001e-10
< 0.0000e+000, 1.2801e-008>	2.0001e-10
< 0.0000e+000, 2.5601e-008>	2.0001e-10
< 0.0000e+000, 5.1201e-008>	2.0001e-10



Back

Close

Common misuse - an example (cont'd)



43/100

$$A = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & \ddots & \\ 1 & & & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} \pm 10^{-10}.$$

Solution of $Ax = b$ by forward substitution

Result (midpoint-radius representation):

intval X =	true radius
< 1.0000e+000, 1.0001e-010>	1.0001e-10
< 0.0000e+000, 2.0001e-010>	2.0001e-10
< 0.0000e+000, 4.0001e-010>	2.0001e-10
< 0.0000e+000, 8.0001e-010>	2.0001e-10
< 0.0000e+000, 1.6001e-009>	2.0001e-10
< 0.0000e+000, 3.2001e-009>	2.0001e-10
< 0.0000e+000, 6.4001e-009>	2.0001e-10
< 0.0000e+000, 1.2801e-008>	2.0001e-10
< 0.0000e+000, 2.5601e-008>	2.0001e-10
< 0.0000e+000, 5.1201e-008>	2.0001e-10

In a standard numerical algorithm do not try to substitute floating-point operations by corresponding interval operations.

This (naive) approach will almost certainly fail!



Back

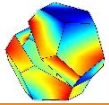
Close

Interval standard functions

An example

$$\left| x - \frac{x^3}{3!} - \sin x \right| \leq \left| \frac{x^5}{5!} \right| \quad \text{for } x \in \mathbb{R}$$

$$\begin{aligned} \Rightarrow \sin(X) &= \{\sin x : x \in X\} \\ &\subseteq \text{hull}\left(X - \frac{X^3}{3!}, X - \frac{X^3}{3!} + \frac{X^5}{5!}\right) \quad \text{for } X \in \mathbb{IR} \end{aligned}$$



44/100

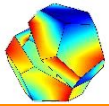


Back

Close

Interval standard functions

An example



44/100

$$\left| x - \frac{x^3}{3!} - \sin x \right| \leq \left| \frac{x^5}{5!} \right| \quad \text{for } x \in \mathbb{R}$$

$$\begin{aligned} \Rightarrow \sin(X) &= \{ \sin x : x \in X \} \\ &\subseteq \text{hull}\left(X - \frac{X^3}{3!}, X - \frac{X^3}{3!} + \frac{X^5}{5!}\right) \quad \text{for } X \in \mathbb{IR} \end{aligned}$$

Much better algorithms available to bound $f(X)$,

$f \in \{$ exp, log, sqrt,
sin, cos, tan, cot,
arcsin, arccos, arctan, arccot,
sinh, cosh, tanh, coth,
arcsinh, arccosh, arctanh, arccoth,
Gamma}

and $X \in \mathbb{IR}, X \in \mathbb{IC}$



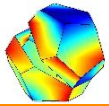
Back

Close

A simple SV-approach

$f : \mathbb{R}^n \rightarrow \mathbb{R}^n \in C^1, \tilde{x} \in \mathbb{R}^n$ with $f(\tilde{x}) \approx 0$

Compute bounds for $\tilde{x} \approx \hat{x}$ with $f(\hat{x}) = 0$.



45/100



Back

Close

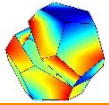
A simple SV-approach

$f : \mathbb{R}^n \rightarrow \mathbb{R}^n \in C^1, \tilde{x} \in \mathbb{R}^n$ with $f(\tilde{x}) \approx 0$

Compute bounds for $\tilde{x} \approx \hat{x}$ with $f(\hat{x}) = 0$.

$$f(x) = 0 \quad \Leftrightarrow \quad g(x) = x,$$

where $g(x) := x - R \cdot f(x), \det R \neq 0$.



45/100



Back

Close

A simple SV-approach

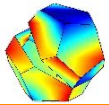
$f : \mathbb{R}^n \rightarrow \mathbb{R}^n \in C^1, \tilde{x} \in \mathbb{R}^n$ with $f(\tilde{x}) \approx 0$

Compute bounds for $\tilde{x} \approx \hat{x}$ with $f(\hat{x}) = 0$.

$$f(x) = 0 \quad \Leftrightarrow \quad g(x) = x,$$

where $g(x) := x - R \cdot f(x), \det R \neq 0$.

$$\begin{aligned} X \in \mathbb{I}\mathbb{R}^n, g(X) \subseteq X &\stackrel{\text{Brouwer}}{\implies} \exists \hat{x} \in X : g(\hat{x}) = \hat{x} \\ &\implies f(\hat{x}) = 0 \end{aligned}$$



45/100



Back

Close

A simple SV-approach

$f : \mathbb{R}^n \rightarrow \mathbb{R}^n \in C^1, \tilde{x} \in \mathbb{R}^n$ with $f(\tilde{x}) \approx 0$

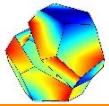
Compute bounds for $\tilde{x} \approx \hat{x}$ with $f(\hat{x}) = 0$.

$$f(x) = 0 \quad \Leftrightarrow \quad g(x) = x,$$

where $g(x) := x - R \cdot f(x), \det R \neq 0$.

$$\begin{aligned} X \in \mathbb{I}\mathbb{R}^n, g(X) \subseteq X &\stackrel{\text{Brouwer}}{\implies} \exists \hat{x} \in X : g(\hat{x}) = \hat{x} \\ &\implies f(\hat{x}) = 0 \end{aligned}$$

Check $g(X) \subseteq X$ and prove $\det R \neq 0$.



45/100



Back

Close

A simple SV-approach

$f : \mathbb{R}^n \rightarrow \mathbb{R}^n \in C^1, \tilde{x} \in \mathbb{R}^n$ with $f(\tilde{x}) \approx 0$

Compute bounds for $\tilde{x} \approx \hat{x}$ with $f(\hat{x}) = 0$.

$$f(x) = 0 \quad \Leftrightarrow \quad g(x) = x,$$

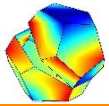
where $g(x) := x - R \cdot f(x), \det R \neq 0$.

$$\begin{aligned} X \in \mathbb{I}\mathbb{R}^n, g(X) \subseteq X &\stackrel{\text{Brouwer}}{\implies} \exists \hat{x} \in X : g(\hat{x}) = \hat{x} \\ &\implies f(\hat{x}) = 0 \end{aligned}$$

Check $g(X) \subseteq X$ and prove $\det R \neq 0$.

Naive approach:

$$g(X) \subseteq X - R \cdot f(X) \not\subseteq X$$



45/100



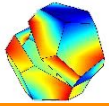
Back

Close

Automatic differentiation

INTLAB example (Broyden's function):

```
>> f = inline(' [ .5*sin(x*y) - y/(4*pi) - x/2 ;  
    (1-1/(4*pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
    - 2*exp(1)*x ]')
```



46/100



Back

Close

Automatic differentiation

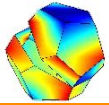
INTLAB example (Broyden's function):

```
>> f = inline(' [ .5*sin(x*y) - y/(4*pi) - x/2 ;  
    (1-1/(4*pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
    - 2*exp(1)*x ]')
```

```
>> f( [ .5 ; 3 ] )
```

```
ans =  
    0.0100  
   -0.1225
```

```
>> f( gradientinit( [ .5 ; 3 ] ) )
```



46/100



Back

Close

gradient value ans.x =

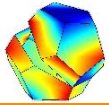
0.0100

-0.1225

gradient derivative(s) ans.dx =

-0.3939 -0.0619

-0.4326 0.8653



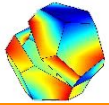
47/100



Back

Close

Automatic differentiation with interval arithmetic



INTLAB example (Broyden's function):

```
>> f = inline(' [ .5*sin(x*y) - y/(4*pi) - x/2 ;  
    (1-1/(4*pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
                - 2*exp(1)*x ] ')  
  
>> f( [ .5 ; infsup(2.9,3.1) ] )
```

```
intval ans =  
    [ -0.0004,    0.0192]  
    [ -0.2091,   -0.0359]
```

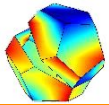
48/100



Back

Close

Automatic differentiation with interval arithmetic



INTLAB example (Broyden's function):

```
>> f = inline(' [ .5*sin(x*y) - y/(4*pi) - x/2 ;  
    (1-1/(4*pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
                - 2*exp(1)*x ]')
```

```
>> f( [ .5 ; infsup(2.9,3.1) ] )
```

```
intval ans =  
    [ -0.0004,    0.0192]  
    [ -0.2091,   -0.0359]
```

```
>> f( gradientinit( [ .5 ; infsup(2.9,3.1) ] ) )
```

```
intval gradient value ans.x =  
    [ -0.0004,    0.0192]  
    [ -0.2091,   -0.0359]  
intval gradient derivative(s) ans.dx =  
    [ -0.4699,   -0.3132] [ -0.0744,   -0.0494]  
    [ -0.4327,   -0.4326] [  0.8652,    0.8653]
```

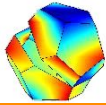
48/100



Back

Close

Bounds for the solution of nonlinear systems



Prove $g(X) = \{x - R \cdot f(x) : x \in X\} \subseteq X$

49/100

$$g(x) = g(\tilde{x}) + M \cdot (x - \tilde{x})$$

$$\text{where } M_i = \frac{\partial g}{\partial x}(\zeta_i), \quad \zeta_i \in x \cup \tilde{x}$$

$$= \tilde{x} - R \cdot f(\tilde{x}) + \{I - R \cdot M\}(x - \tilde{x})$$

$$\subseteq \tilde{x} - R \cdot f(\tilde{x}) + \{I - R \cdot M\}(X - \tilde{x})$$

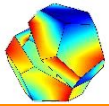
$$=: K_f(\tilde{x}, X) \quad \text{Krawczyk operator}$$



Back

Close

Bounds for the solution of nonlinear systems



49/100

Prove $g(X) = \{x - R \cdot f(x) : x \in X\} \subseteq X$

$$g(x) = g(\tilde{x}) + M \cdot (x - \tilde{x})$$

$$\text{where } M_i = \frac{\partial g}{\partial x}(\zeta_i), \quad \zeta_i \in x \cup \tilde{x}$$

$$= \tilde{x} - R \cdot f(\tilde{x}) + \{I - R \cdot M\}(x - \tilde{x})$$

$$\subseteq \tilde{x} - R \cdot f(\tilde{x}) + \{I - R \cdot M\}(X - \tilde{x})$$

$$=: K_f(\tilde{x}, X) \quad \text{Krawczyk operator}$$

Theorem. $\tilde{x} \in X$ and $K_f(\tilde{x}, X) \subseteq \text{int}(X)$

$\Rightarrow \exists! \hat{x} \in X : f(\hat{x}) = 0.$

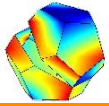
Computation of M by automatic differentiation



INTLAB example (Broyden's function):

```
>> f = inline('[ .5*sin(x*) - y/(4*pi) - x/2 ;  
    (1-1/(4+pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
    - 2*exp(1)*x ]')  
  
>> xs = [ .5 ; 3 ]; verifynlss(f,xs)
```

```
intval ans =  
[    0.4999, 0.5001]  
[    3.1415, 3.1416]
```



50/100



Back

Close

INTLAB example (Broyden's function):

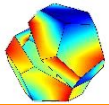
```
>> f = inline('[ .5*sin(x*) - y/(4*pi) - x/2 ;  
    (1-1/(4*pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
                - 2*exp(1)*x ]')
```

```
>> xs = [ .5 ; 3 ]; verifynlss(f,xs)
```

```
intval ans =  
[    0.4999, 0.5001]  
[    3.1415, 3.1416]
```

```
>> xs = [ 0 ; 0 ]; X = verifynlss(f,xs)
```

```
intval X =  
[ -0.2606, -0.2605]  
[  0.6225,  0.6226]
```



50/100



Back

Close

INTLAB example (Broyden's function):

```
>> f = inline('[ .5*sin(x*) - y/(4*pi) - x/2 ;  
    (1-1/(4+pi))*(exp(2*x)-exp(1)) + exp(1)*y/pi  
                - 2*exp(1)*x ]')
```

```
>> xs = [ .5 ; 3 ]; verifynlss(f,xs)
```

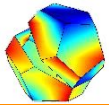
```
intval ans =  
[ 0.4999, 0.5001]  
[ 3.1415, 3.1416]
```

```
>> xs = [ 0 ; 0 ]; X = verifynlss(f,xs)
```

```
intval X =  
[ -0.2606, -0.2605]  
[ 0.6225, 0.6226]
```

```
>> format long; X
```

```
intval X =  
[ -0.26059929002248, -0.26059929002247]  
[ 0.62253089661391, 0.62253089661392]
```



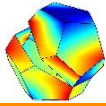
50/100



Back

Close

Nonlinear functions with uncertain parameters



51/100

$$f : \mathbb{R}^{k \times n} \rightarrow \mathbb{R}^n, \quad f(\tilde{p}, \tilde{x}) \approx 0.$$

Interval extension $F(P, X)$ for $P \in \mathbb{IR}^k$, $X \in \mathbb{IR}^n$

$$L_F(P, \tilde{x}, X) := -R \cdot F(P, \tilde{x}) + \{I - R \cdot M\}X$$

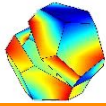
where M is computed by automatic differentiation applied to F at P and X .



Back

Close

Nonlinear functions with uncertain parameters



51/100

$$f : \mathbb{R}^{k \times n} \rightarrow \mathbb{R}^n, \quad f(\tilde{p}, \tilde{x}) \approx 0.$$

Interval extension $F(P, X)$ for $P \in \mathbb{IR}^k$, $X \in \mathbb{IR}^n$

$$L_F(P, \tilde{x}, X) := -R \cdot F(P, \tilde{x}) + \{I - R \cdot M\}X$$

where M is computed by automatic differentiation applied to F at P and X .

Theorem. $\tilde{x} \in X$ and $L_F(P, \tilde{x}, X) \subseteq \text{int}(X)$

$$\Rightarrow \forall \hat{p} \in P \exists! \hat{x} = \hat{x}(\hat{p}) \in \tilde{x} + X : f(\hat{x}) = 0.$$



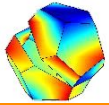
Back

Close

INTLAB example (Broyden's function):

```
>> Pi = midrad( 3.141592653589793, 1e-15 )
```

```
>> xs = [ .6 ; 3 ];  verifynlss(f,xs)
```



52/100



Back

Close

INTLAB example (Broyden's function):

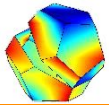
```
>> Pi = midrad( 3.141592653589793, 1e-15 )
```

```
>> xs = [ .6 ; 3 ]; verifynlss(f,xs)
```

```
intval ans =
```

```
[ 0.499999999999740, 0.500000000000260]
```

```
[ 3.14159265357934, 3.14159265360025]
```



52/100



Back

Close

INTLAB example (Broyden's function):

```
>> Pi = midrad( 3.141592653589793, 1e-15 )
```

```
>> xs = [ .6 ; 3 ]; verifynlss(f,xs)
```

```
intval ans =
```

```
[ 0.49999999999740, 0.50000000000260]
```

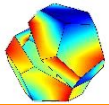
```
[ 3.14159265357934, 3.14159265360025]
```

```
>> xs = [ 0 ; 0 ]; y = verifynlss(f,xs)
```

```
intval ans =
```

```
[ -0.26059929002903, -0.26059929001592]
```

```
[ 0.62253089659741, 0.62253089663041]
```



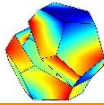
52/100



Back

Close

Automatic slopes (Krawczyk, Neumaier, R.)



Given $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$, $\tilde{x} \in \mathbb{R}^n$, $X \in \mathbb{I}\mathbb{R}^n$

"slope" arithmetic uses triples $(f_c, f_r, f_s) \in \mathbb{I}\mathbb{R}^n \times \mathbb{I}\mathbb{R}^n \times \mathbb{I}\mathbb{R}^{n \times n}$ with

$$f(\tilde{x}) \in f_c$$

$$f(X) \subseteq f_r$$

$$\forall x \in X \exists s \in f_s : f(x) = f(\tilde{x}) + s \cdot (x - \tilde{x})$$

$\tilde{x} \in \mathbb{R}^n$, $X \in \mathbb{I}\mathbb{R}^n$ given,

y computed slope expansion of f w.r.t. \tilde{x}, X ,

$$L_f(\tilde{x}, X) = -R \cdot y_c + \{I - R \cdot y_s\}X.$$

Theorem. $L_f(\tilde{x}, X) \subseteq \text{int}(X)$

$\Rightarrow \exists \hat{x} \in \tilde{x} + L_f(\tilde{x}, X) : f(\hat{x}) = 0.$

Remark. f continuous, $\tilde{x} \in X$ not necessary,

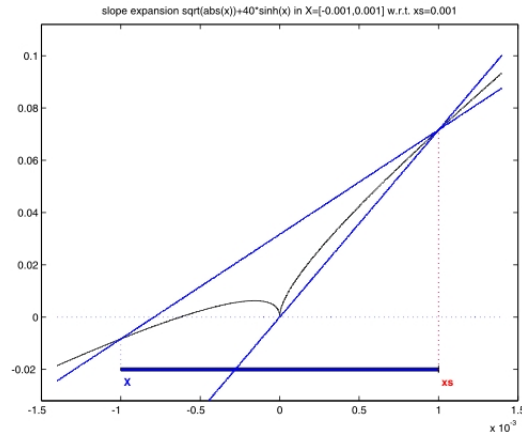
\hat{x} not necessarily unique in X .



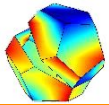
Intlab example (with tentatively large inclusion interval):

```
>> f = inline('sqrt(abs(x))+40*sinh(x)'), xs = 1e-3;  
X = infsup(-3e-3,0); xs+Lf(xs,X)
```

```
intval ans =  
[ -0.00198031444750, 0.00099016265108]
```



Inclusion of (real) zeros 1 simple + 1 double *or*
3 simple *or*
1 simple



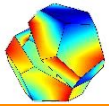
54/100



Back

Close

A linear example



55/100

$$f(x) := Ax - b$$

$$g(x) = x - R(Ax - b) = Rb + (I - RA)x$$

$$g(X) = Rb + (I - RA) \cdot X \subseteq \text{int}(X) \quad \Rightarrow$$

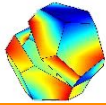
A, R nonsingular and $A^{-1}b \in X$



Back

Close

A linear example



55/100

$$\begin{aligned} f(x) &:= Ax - b \\ g(x) &= x - R(Ax - b) = Rb + (I - RA)x \\ g(X) &= Rb + (I - RA) \cdot X \subseteq \text{int}(X) \Rightarrow \\ &A, R \text{ nonsingular and } A^{-1}b \in X \end{aligned}$$

INTLAB computing time ($n = 500$, 1.2Ghz Laptop)

0.2 sec for $A \setminus b$ (built-in solver)
1.4 sec with verification as above
0.7 sec verification by Oishi's method



Back

Close

Self-validating methods:

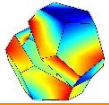
Correct answer *or* error message

Wright/Foster (1994):

$$\dot{x} = x - 1 \quad \text{with} \quad x(T) = x(0) \quad \text{in} \quad [0, T]$$

[obviously $x \equiv 1$]

Integration trapezoidal rule yields linear system



56/100



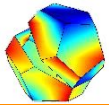
Back

Close

Solution of linear system with $n = 70$

approximation	intval	X =
1.000000000000000	[0.999999999999999,	1.000000000000001]
1.000000000000000	[0.999999999999999,	1.000000000000001]
...
1.00111607142857	[0.999999999999994,	1.000000000000006]
1.00111607142857	[0.999999999999994,	1.000000000000006]
1.01211734693878	[0.999999999999993,	1.000000000000007]
1.01211734693878	[0.999999999999993,	1.000000000000007]
0.99011479591837	[0.999999999999993,	1.000000000000007]
0.96811224489796	[0.999999999999992,	1.000000000000007]
0.88010204081633	[0.999999999999992,	1.000000000000008]
1.05612244897959	[0.999999999999992,	1.000000000000008]
0.70408163265306	[0.999999999999992,	1.000000000000008]
0.70408163265306	[0.999999999999992,	1.000000000000009]
1.40816326530612	[0.999999999999991,	1.000000000000009]
0	[0.999999999999990,	1.000000000000009]
0	[0.999999999999990,	1.000000000000010]
0	[0.999999999999989,	1.000000000000011]
0	[0.999999999999986,	1.000000000000012]
-22.53061224489796	[0.999999999999999,	1.000000000000001]

No warning!



57/100



Back

Close

Data with tolerances

Theorem 1. $A, R \in M_n(\mathbb{R})$, $b \in \mathbb{R}^n$, $X \in \mathbb{IR}^n$ and

$$Rb + (I - RA) \cdot X \subseteq \text{int}(X)$$

$\Rightarrow A$ and R are nonsingular and $A^{-1}b \in X$.

Theorem 2. $\mathcal{A} \in \mathbb{IM}_n(\mathbb{R})$, $\mathbf{b} \in \mathbb{IR}^n$, $R \in M_n(\mathbb{R})$, $X \in \mathbb{IR}^n$ and

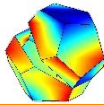
$$R\mathbf{b} + (I - RA)X \subseteq \text{int}(X)$$

$\Rightarrow \forall A \in \mathcal{A} \quad \forall b \in \mathbf{b} : A$ and R are nonsingular
and $A^{-1}b \in X$.

That means

$$\Sigma(\mathcal{A}, \mathbf{b}) := \{x \mid \exists A \in \mathcal{A} \exists b \in \mathbf{b} : Ax = b\} \subseteq X$$

Poljak/Rohn (1993): Computation of exact bounds for $\Sigma(\mathcal{A}, \mathbf{b})$ is NP-hard

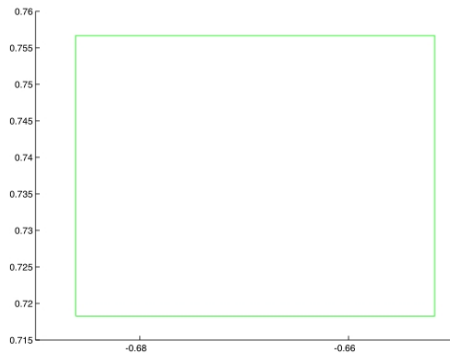


A trivial example

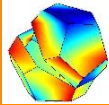
$$A = \begin{pmatrix} [-0.5796, -0.5771] & [0.2469, 0.2581] \\ [0.2469, 0.2581] & [-0.4370, -0.4365] \end{pmatrix},$$
$$b = \begin{pmatrix} 0.5731 \\ -0.4910 \end{pmatrix}$$

```
>> A = infsup( [-0.5796 0.2469 ; 0.2469 -0.4370 ] ,  
               [-0.5771 0.2581; 0.2581 -0.4365 ] );  
           b = [ 0.5731 ; -0.4910 ];  
>> X = verifylss(A,b)
```

```
intval X =  
[ -0.6862, -0.6517]  
[ 0.7182, 0.7567]
```



Overestimation?



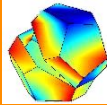
59/100



Back

Close

Data with tolerances - inner inclusion (Neumaier, R.)

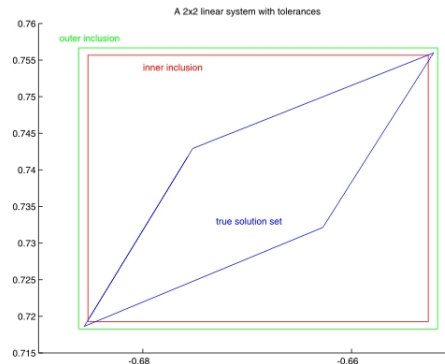


Theorem. $\mathcal{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$, $\tilde{x} \in \mathbb{R}^n$, $R \in \mathbb{R}^{n \times n}$,
 $X \in \mathbb{R}^n$,

$$Z := R(\mathbf{b} - \mathcal{A}\tilde{x}) \quad \text{and} \quad \Delta := \{I - R\mathcal{A}\} \cdot X$$

$$Z + \Delta \subseteq \text{int}(X) \quad \Rightarrow$$

$$\begin{aligned} \inf Z_i + \inf \Delta_i &\leq \inf \sum (\mathcal{A}, \mathbf{b})_i \leq \inf Z_i + \sup \Delta_i \\ \sup Z_i + \inf \Delta_i &\leq \sup \sum (\mathcal{A}, \mathbf{b})_i \leq \sup Z_i + \sup \Delta_i. \end{aligned}$$



Monte Carlo approach

```
for  $i = 1 : K$   
  choose  $A \in \mathcal{A}$   
  choose  $b \in \mathfrak{b}$   
  compute  $x = A \setminus b$   
   $x_{\min} = \min(x_{\min}, x)$   
   $x_{\max} = \max(x_{\max}, x)$   
end
```

Linear system with random matrix A and random b , $n = 100$

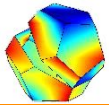
Tolerances 10^{-4} in each component of A and b

$K = 100$ sample linear systems

Computing time

self-validating method 0.39 sec

Monte Carlo 11.9 sec

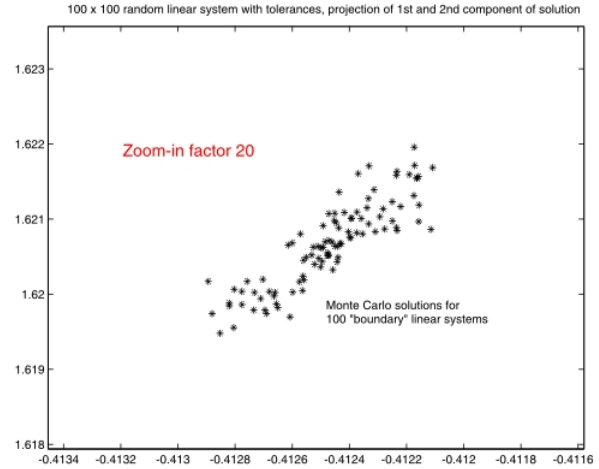
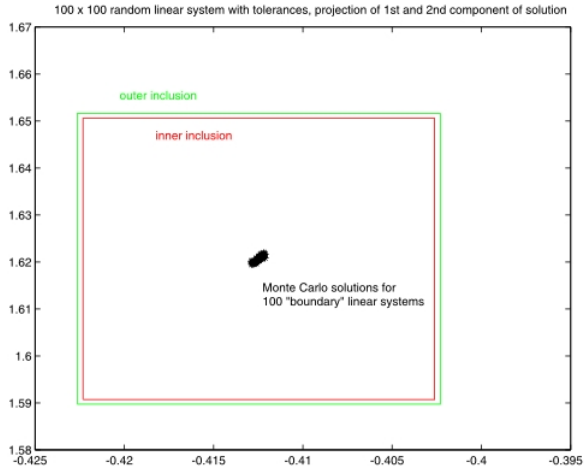
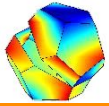


61/100



Back

Close

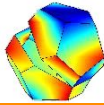


Data with tolerances and dependencies

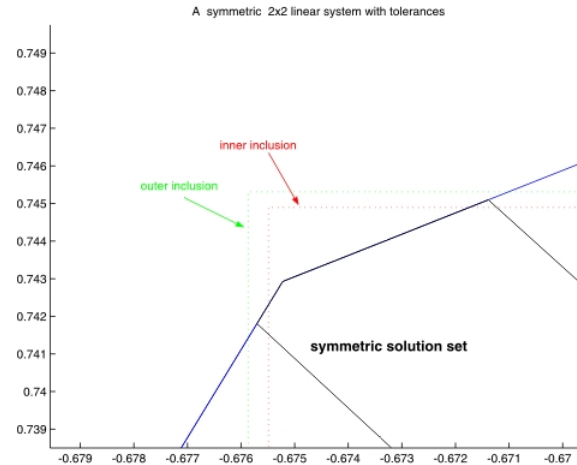
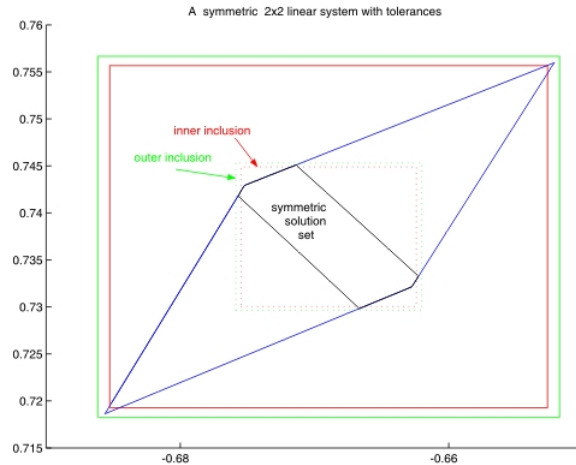
$$\Sigma(\mathcal{A}, \mathfrak{b}) := \{x \mid \exists A \in \mathcal{A} \exists b \in \mathfrak{b} : Ax = b\}$$

$$\Sigma_{\text{sym}}(\mathcal{A}, \mathfrak{b}) = \{x \mid \exists A \in \mathcal{A} \exists b \in \mathfrak{b} : A = A^T, Ax = b\}$$

similarly for Toeplitz, Hankel, circulant ...



63/100



Back

Close

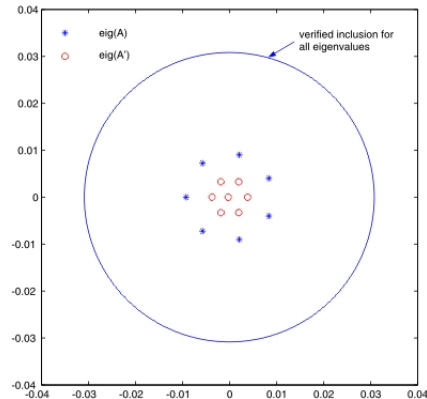
Self-validating methods:

Correct answer or error message

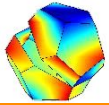
What are the eigenvalues of the following 7×7 matrix?

A =

```
3.2704  4.2268 -2.9925  2.5902 -0.1306  1.3495  0.4708
2.7041  0.9369 -0.7639  2.9434  1.1338 -0.5135 -1.2424
2.6288  2.7926 -1.1863  2.4664 -0.4605  0.9760 -1.1777
-5.3920 -3.9846  3.7885 -5.2513 -2.7607  0.2610  0.0214
3.3226  0.8233 -1.1560  3.3884  2.0575 -0.3869 -1.1942
-0.4403 -0.9826  0.4109 -0.3525  0.2232 -1.0126 -0.2296
-1.3145  1.8579 -0.4431 -1.6898 -1.9447  1.6597  1.1853
```



no warning!



64/100



Back

Close

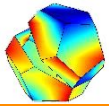
Programming robustness

- Construct problem with integer entries and known solution, e.g. π
- A programming error may produce

[3.141592653589792, 3.141592653589793]

Floating-point approximations of superb quality.

The bounds coincide to 16 decimal places
but are **WRONG!**



65/100

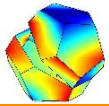


Back

Close

Successful applications

- The Kepler conjecture (Hales)
- The double-bubble conjecture (Hass, Hutchings, Schlafly)
- Dynamics of the Jouanolou foliation (Camacho, Figureido)
- Verified bound for the Feigenbaum constant (Eckmann, Wittwer, Lanford)
- Existence of transversal homocyclic points for a dynamical system with diffeomorphism (Neumaier, Rage)
- Existence of eigenvalue below the essential spectrum of the Sturm-Liouville problem (Brown, Mc Cormack, Zettl)
- Eigenfrequencies of turbine (Kulisch et al.)
- SPICE program for circuit analysis (R.)
- Extreme currents in Lake Constance (R.)
- Forest planning (Jansson)
-



66/100



Back

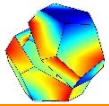
Close

Self-validating methods for

- global optimization
(Csendes, Jansson, Kearfott, Ratz, ...)
- *all* zeros of nonlinear system in a box
(Kearfott, Knüppel)
- least squares problems (Krawczyk, R.)
- sparse linear and nonlinear problems (R.)
- initial value and boundary value problems
(Lohner, Nedialkov, Jackson)
- partial differential equations (Nakao, Plum)

Recent developments:

Very fast self-validating methods for linear problems, eigenproblems and others (Oishi)



67/100



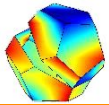
Back

Close

Conclusion

- Self-validating methods provide rigorous answers to numerical problems (using numerical approximations)
- in a computing time not too far from a standard numerical algorithm
- SV methods are used in so-called "computer-assisted proofs"

INTLAB is freely available from www.ti3.tu-harburg.de/rump/intlab/



68/100



Back

Close