Interval and affine computation

Iwona Skalna
AGH University of Science and Technology

AGH

Outline

- Floating-point numbers
- Rounding errors
- Interval computation
- Affine and revised affine forms
- Practical applications

Floating-point numbers

- Floating-point (FP) numbers are important because they are ubiquitous in scientific computing. A floating-point number is represented as

  $$\boldsymbol{s \times \beta^{e} \times d_0 . d_1\, d_2 \cdots d_{p-1}}$$

  Where $s \in \{-1,1\}$ is the sign, β is called the base, e is the exponent, p is the precision, and $d_0 . d_1 d_2 \cdots d_{p-1}$ part is called the significand (mantissa, fraction, coefficient).
  The significand is normalized to belong to the interval $[0, \beta)$.
- Why „floating point"? Because the exponent logically determines where the decimal point is placed within (or even outside) the significand.
- IEEE 754 standard requires β=2, p=24 for single precision and p=53 for double precision.
- Thanks to the use of β=2, only 23 bits and 52 bits must be stored for, respectively, single and double precision, since all normalized numbers starts with 1, so no need to store it.

Floating-point numbers

Floating-point numbers do not behave as do the real numbers encountered in mathematics:

- There is only a finite set of floating-point numbers, since computer representation must be finite.
- The set of floating-point numbers does not form a field under the usual set of arithmetic operations.
- Some common rules of real arithmetic are not always valid when applied to floating-point operations.

Floating-point arithmetic

Let a, b and c be floating-point numbers. Then

- $a + b$ may not be a floating-point number
  - $a + b$ may not always equal $a \oplus b$
  - Similarly for the operations $-$, $\times$ and /
  - Recall that floating-point numbers do not form a field
- $(a \oplus b) \oplus c$ may not be equal to $a \oplus (b \oplus c)$ (lack of associativity)
  - Similarly for other elementary operation
- $a \otimes (b \oplus c)$ may not be equal to $(a \otimes b) \oplus (a \otimes c)$ (lack of distributivity)
- $(1 \oslash a) \otimes a$ may not be equal to a.

Floating-point numbers

Consequences:

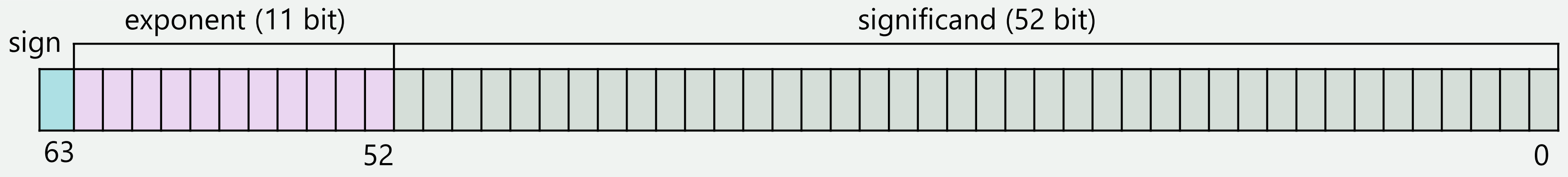
- Since there is only a finite set of floating-point numbers, there are rational numbers which are not floating-point numbers (transcendentals such as π, e, $\sqrt{2}$ cannot be represented exactly by a floating-point value regardless of format or precision).
- The decimal equivalent of any finite floating-point value contains a finite number of non-zero digits.
- Floating-point computation is not guaranteed because of rounding errors => the result that is not representable as machine number is rounded.

Rounding and truncation errors

What will be the result in double precision?

$2^{53}+1-2^{53} \neq 1$ (why?)

$2^{53}+1 = ?$



$A = 2^{53}, B = 1$

$$A = 2^{expA} * \left(1 + \frac{a_1}{2^1} + \frac{a_2}{2^2} + \cdots + \frac{a_{52}}{2^{52}}\right), B = 2^{expB} * \left(1 + \frac{b_1}{2^1} + \frac{b_2}{2^2} + \cdots + \frac{b_{52}}{2^{52}}\right)$$

Rounding and truncation errors

$$(A+B) = 2^{expA} * \left(1 + \frac{a_1}{2^1} + \frac{a_2}{2^2} + \dots + \frac{a_{52}}{2^{52}}\right) 2^{expB} * \left(1 + \frac{b_1}{2^1} + \frac{b_2}{2^2} + \dots + \frac{b_{52}}{2^{52}}\right)$$

The first thing to do is to make the exponents of A and B agree. The respective term is divided by the difference $xB = expA - expB$ ($expA > expB$).

$$(A+B) = 2^{expA} * \left(1 + \frac{a_1}{2^1} + \frac{a_2}{2^2} + \dots + \frac{a_{52}}{2^{52}}\right) + 2^{expA} * \frac{\left(1+\frac{b_1}{2^1}+\frac{b_2}{2^2}+\dots+\frac{b_{52}}{2^{52}}\right)}{2^{xB}}$$

$$(A+B) = 2^{expA} * \left(1 + \frac{a_1}{2^1} + \frac{a_2}{2^2} + \dots + \frac{a_{52}}{2^{52}} + \frac{1}{2^{xB}} + \frac{b_1}{2^{1+xB}} + \frac{b_2}{2^{2+x}} + \dots + \frac{b_{52}}{2^{52+}}\right)$$

In the cosidered case $xB = 53$, so all terms after $\frac{a_{52}}{2^{52}}$ will be neglected and the result will be

$$(A+B) = 2^{expA} * \left(1 + \frac{a_1}{2^1} + \frac{a_2}{2^2} + \dots + \frac{a_{52}}{2^{52}}\right) = A\,.$$

Remedy: transform $2^{53}+1-2^{53}$ => $2^{53}-2^{53}+1 = 1$.

Rounding and truncation errors

Consider the following quadratic equation:

$$x^2 + 2px - 1 = 0$$

The smaller (per modulus) root is $x_1 = -p + \sqrt{p^2 + 1}$.

In double precision floating-point arithmetic (binary64),
if $p \geq 10^8$, then $1/p^2 < 2^{-53}$ and $1+p^2=p^2(1+1/p^2)=p^2$.

So, $x_2 = 0$, whereas the correct value is $x_1 = 5 \cdot 10^{-9}$

Remedy: transform $x_2 = -p + \sqrt{p^2 + 1}$ => $x_2 = \frac{1}{2p}$. (correct!)

Rounding and truncation errors

Does increasing accuracy can help?

$f = 333.75b^6 + a^2(11a^2b^2 - b^6 - 121b^4 - 2) + 5.5b^8 + a/2^b,$

$a = 77617, b = 33096.$

single precision $f \approx 1.\underline{172603}...$

double precision $f \approx 1.\underline{1726039400531}...$

quadruple precision $f \approx 1.\underline{172603940053}178...$

correct value $f = -0.827386... = a/2b - 2$

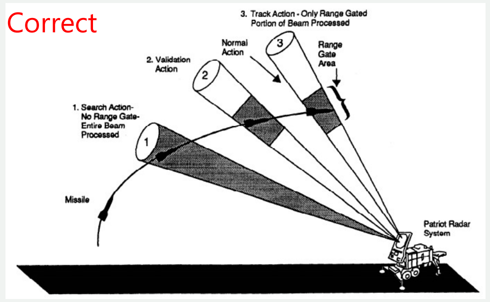
Remedy: transform => challenge

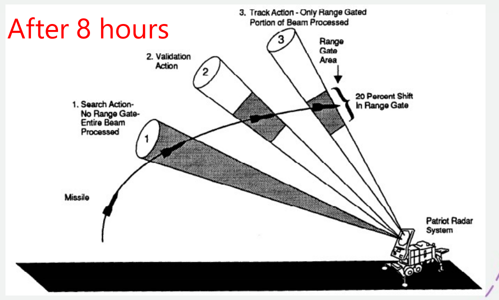
Rounding and truncation errors

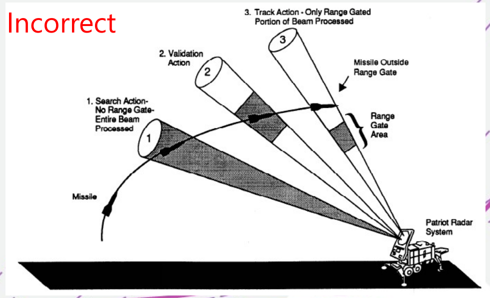
The Patriot Missile Failure

On February 25, 1991, during the Gulf War, an American Patriot Missile battery in Dhahran, Saudi Arabia, failed to track and intercept an incoming Iraqi Scud missile. The Scud struck American Army barracks, killing 28 soldiers and injuring around 100 other people.

- An investigation discovered a bug in the Patriot's tracking software that caused the system's internal clock to drift gradually from the real time.
- The time was stored as an integer number in a 24-bit register with an accuracy of 1/10 of a second. This resulted in some portion of the time value being lost as it incremented each 0.1 seconds.
- To calculate a target's location, the data had to be cast to real numbers
  1/10 is $1/2^4+1/2^5+1/2^8+1/2^9+1/2^{12}+1/2^{13}+\ldots$ In other words, binary expansion of the value 1/10 is 0.0001100110011001100110011001100.... That's why this value, stored in a 24-bit register, was rounded to 0.00011001100110011001100, resulting in a precision error of 0.000000000000000000000000011001100... in binary format, or about 0.000000095 in decimal format. During 100 hours of continuous operation, this error would build up to $0.000000095*100*60*60*10=0.34$ seconds.
- An R-17's velocity is 1676 m/s, so it covers over half a kilometer in 0.34 seconds, which is more than enough for the missile to slip past the Patriot's intercept range. The funny thing is that this time-calculation bug was fixed only in some parts of the software, but not in all of it.


Correct
3. Track Action - Only Range Gated Portion of Beam Processed
Normal Action
Range Gate Area
2. Validation Action
1. Search Action- No Range Gate- Entire Beam Processed
Missile
Patriot Radar System


After 8 hours
3. Track Action - Only Range Gated Portion of Beam Processed
Range Gate Area
20 Percent Shift In Range Gate
2. Validation Action
1. Search Action- No Range Gate- Entire Beam Processed
Missile
Patriot Radar System


Incorrect
3. Track Action - Only Range Gated Portion of Beam Processed
Missile Outside Range Gate
Range Gate Area
2. Validation Action
1. Search Action- No Range Gate- Entire Beam Processed
Missile
Patriot Radar System

Alternative representation

+ Consider a real number :

  $x = 5.9$ // not a machine number

+ We can substitute x with a pair of machine numbers (very rough representation)

  $x \in [5.75,6]$ // both 5.74 and 6 are machine numbers

+ Floating-point representation of x is:

  0 10000000001 0111100110011001100110011001100110011001100110011001|100 …
  0 10000000001 0111100110011001100110011001100110011001100110011001 (round down)
  0 10000000001 0111100110011001100110011001100110011001100110011010 (round up)

Tight interval $x =$ [5.8999999999999995, 5.9000000000000004] that represents $x = 5.9$.

Why interval arithmetic is so useful?

+ Measurement errors – measured value ± accuracy of a measurement tool.
+ Dynamical model parameters – the distribution probability is not known, but the range for the parameters can be given.
+ Truncation errors – for example when approximating a function with Taylor series or when approximating derivatives with finite differences.
+ Sensitivity analysis – examining the influence of parameter changes on the system behavior.
+ Infinite numer of states – impossible to process them separately.

Intervals

avoid some of FP drawbacks, but have some other problems

The main idea is to represent an unknown value $\tilde{x}$ with a known absolute error $\leq r$ as a closed and bounded interval

$$\boldsymbol{x} = [\underline{x}, \bar{x}] = [\tilde{x} - r, \tilde{x} + r] = \{x \in \mathrm{R} \mid \underline{x} \leq x \leq \bar{x}\}.$$

It is guaranteed that $\tilde{x}$ is „somewhere" between the lower bound $\underline{x}$ and the upper bound $\bar{x}$.

On the other hand, given an interval $[\underline{x}, \bar{x}]$, its midpoint

$$x^c = (\underline{x} + \bar{x})/2,$$

can be tread as approximation of unknown value $\tilde{x}$ and its radius

$$x^\Delta = (\bar{x} - \underline{x})/2,$$

can be treated as an approximation error:

$$\tilde{x} \in [\underline{x}, \bar{x}] \Leftrightarrow |\tilde{x} - x^c| \leq x^\Delta.$$

The set of all intervals is denoted as IR.

Interval arithmetic (IA)

Given the interval values

$$\boldsymbol{x} = [\underline{x}, \bar{x}] \text{ and } \boldsymbol{y} = [\underline{y}, \bar{y}],$$

the elementary operations $\circ \in \{+, -, \times, /\}$ are defined as

$$\boldsymbol{x} \circ \boldsymbol{y} = \mathrm{hull}\{x \circ y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\},$$

where hull S = [inf S, sup S] (for $S \neq \emptyset$).

Elementary arithmetic operations on intervals are inclusion isotone, i.e.,

$$[x] \subseteq [x'] \wedge [y] \subseteq [y'] \Longrightarrow [x] \circ [y] \subseteq [x'] \circ [y'], \circ \in \{+, -, \times, /\}$$

Interval arithmetic

Endpoint form of elementary operations (Moore):

$$\boldsymbol{x} + \boldsymbol{y} = \left[\underline{x} + \underline{y}, \bar{x} + \bar{y}\right]$$

$$\boldsymbol{x} - \boldsymbol{y} = \left[\underline{x} - \bar{y}, \bar{x} - \underline{y}\right]$$

$$\boldsymbol{x} * \boldsymbol{y} = \left[\min\left\{\underline{x} * \underline{y}, \underline{x} * \bar{y}, \bar{x} * \underline{y}, \bar{x} * \bar{y}\right\}, \max\left\{\underline{x} * \underline{y}, \underline{x} * \bar{y}, \bar{x} * \underline{y}, \bar{x} * \bar{y}\right\}\right]$$

$$1/\boldsymbol{y} = \left[1/\bar{y}, 1/\underline{y}\right] \ (0 \notin \boldsymbol{y})$$

$$\boldsymbol{x}/\boldsymbol{y} = \boldsymbol{x} * (1/\boldsymbol{y}) \ (0 \notin \boldsymbol{y})$$

Interval arithmetic

The algebraic system of interval arithmetic is only an abelian monoid under both addition and multiplication, additive and multiplicative inverses exist only for intervals of zero radius.

Interval arithmetic operations only preserve some of the algebraic laws valid for real numbers. In particular, the following laws hold true for arbitrary intervals $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$:

$\boldsymbol{x}+\boldsymbol{y}=\boldsymbol{y}+\boldsymbol{x}$	$\boldsymbol{x}*\boldsymbol{y}=\boldsymbol{y}*\boldsymbol{x}$	(commutativity)
$(\boldsymbol{x}+\boldsymbol{y})+\boldsymbol{z}=(\boldsymbol{x}+\boldsymbol{y})+\boldsymbol{z}$	$(\boldsymbol{x}*\boldsymbol{y})*\boldsymbol{z}=\boldsymbol{x}*(\boldsymbol{y}*\boldsymbol{z})$	(associativity)
$\boldsymbol{x}+0=0+\boldsymbol{x}$	$1*\boldsymbol{x}=\boldsymbol{x}*1$	(neutral element)

Interval arithmetic

The distributivity and cancellation are present in the interval arithmetic in a weaker form as sub-distributivity and sub-cancellation, respectively.

For arbitrary intervals $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, the following holds true:

- sub-distributivity:

  $\boldsymbol{x} * (\boldsymbol{y} \pm \boldsymbol{z}) \subseteq \boldsymbol{x} * \boldsymbol{y} \pm \boldsymbol{x} * \boldsymbol{z}$ $(\boldsymbol{x} \pm \boldsymbol{y}) * \boldsymbol{z} \subseteq \boldsymbol{x} * \boldsymbol{z} \pm \boldsymbol{y} * \boldsymbol{z}$

- sub-cancellation:

  $\boldsymbol{x} - \boldsymbol{y} \subseteq (\boldsymbol{x} + \boldsymbol{z}) - (\boldsymbol{y} + \boldsymbol{z})$ $\boldsymbol{x}/\boldsymbol{y} \subseteq (\boldsymbol{x} * \boldsymbol{z})/(\boldsymbol{y} * \boldsymbol{z})$

  $0 \in \boldsymbol{x} - \boldsymbol{x}$ $1 \in \boldsymbol{x}/\boldsymbol{x}$

Interval arithmetic

Example. Let $\boldsymbol{x} = [-5,3]$ and $\boldsymbol{y} = [1,2]$. Then

$$[-5,3] + [1,2] = [-5 + 1, 3 + 2] = [-4,5]$$

$$[-5,3] - [1,2] = [-5 - 2, 3 - 1] = [-7,2]$$

$$[-5,3] * [1,2] \ = [\min\{-5 * 1, -5 * 2, 3 * 1, 3 * 2\}, \max\{-5 * 1, -5 * 2, 3 * 1, 3 * 2\}] = [-10,6]$$

$$1/\ [1,2] = 1 * [1/2,1] = [1/2, 1]$$

$$[-5,3]/[1,2] \ = [-5,3] * [1/2, 1] = [-5,3]$$

Interval arithmetic

Example. Let $\boldsymbol{x} = [-6,1]$, $\boldsymbol{y} = [1,2]$ and $\boldsymbol{z} = [-3,1]$.

Sub-distributivity

$$[-6,1] * [1,2] + [-6,1] * [-3,1] = [-12,2] + [-6,18] = [-18,20]$$

$$[-6,1] * ([1,2] + [-3,1]) = [-6,1] * [-2,3] = [\mathbf{-18,12}] \subset [\mathbf{-18,20}]$$

Sub-cancellation

$$([-6,1] + [-3,1]) - ([1,2] + [-3,1]) = [-9,2] - [-2,3] = [-12,4]$$

$$[-6,1] - [1,2] = [\mathbf{-8,0}] \subset [\mathbf{-12,4}]$$

$$[-6,1] - [-6,1] = [\mathbf{-7,7}] \neq \mathbf{0}$$

$$[1,2]/[1,2] = [\mathbf{1/2,2}] \neq \mathbf{1}$$

Computer interval arithmetic

In computer interval arithmetic, computations are performed on intervals where the lower bound $\underline{x}$ and the upper $\bar{x}$ are machine numbers.

In order to to guarantee that the obtained interval solution is rigorous, i.e., the unknown exact result is included in the resulting interval, the so-called outward rounding procedure is used:

+ the lower bound of the resulting interval must be rounded downward (towards -∞)
+ the upper bound of the resulting interval must be rounded upward (towards +∞)

The outward rounding is optimal when

$$\boldsymbol{x} = \bigcap \left\{ \boldsymbol{y} = \left[\underline{y}, \bar{y}\right] \mid \boldsymbol{y} \supseteq \boldsymbol{x},\ \underline{y}, \bar{y} \text{ are machine numbers} \right\}.$$

Computer interval arithmetic

Endpoint form of elementary operations (Moore) in computer arithmetic is as follows:

$$\boldsymbol{x}+\boldsymbol{y}=\left[\nabla\left(\underline{x}+\underline{y}\right),\Delta(\bar{x}+\bar{y})\right]$$

$$\boldsymbol{x}-\boldsymbol{y}=\left[\nabla(\underline{x}-\bar{y}),\Delta\left(\bar{x}-\underline{y}\right)\right]$$

$$\boldsymbol{x}*\boldsymbol{y}=\left[\min\left\{\nabla\left(\underline{x}*\underline{y}\right),\nabla(\underline{x}*\bar{y}),\nabla\left(\bar{x}*\underline{y}\right),\nabla(\bar{x}*\bar{y})\right\},\max\left\{\Delta\left(\underline{x}*\underline{y}\right),\Delta(\underline{x}*\bar{y}),\Delta\left(\bar{x}*\underline{y}\right),\Delta(\bar{x}*\bar{y})\right\}\right]$$

$$\boldsymbol{1}/\,\boldsymbol{y}=\left[\nabla(1/\bar{y}),\Delta\left(1/\underline{y}\right)\right]\ (0\notin\boldsymbol{y})$$

$$\boldsymbol{x}/\,\boldsymbol{y}=\boldsymbol{x}*\left(\frac{\boldsymbol{1}}{\boldsymbol{y}}\right)\ (0\notin\boldsymbol{y})$$

where ∇ denote downward rounding and Δ denote upward rounding.

Computer interval arithmetic

The following examples illustrate interval operations and outward rounding. The base is $B=10$ and the length of significand is $L=2$.

$$[1.1,1.2] + [-2.1,0.2] = [-1.0,1.4]$$

$$[1.1,1.2] - [-2.1,0.2] = [0.9,3.3]$$

$[1.1,1.2]*[-2.1,0.2] = [-2.52,0.24]$, rounded: $[-2.6,0.24]$

$[1.1,1.2]/[-2.1,0.2]$ not defined since $0\in\boldsymbol{y}$

$[-2.1,0.2]/[1.1,1.2] = [-21/11,2/11]$, rounded: $[-2.0,0.19]$

$[-2.1,0.2]/[1.1,1000\,] = [-21/11,2/11]$, rounded: $[-2.0,0.19]$

Outward rounding

In 32-bit architecture, the FPU (floating-point unit) 16-bit control word (CW) allows to set, among others, the rounding mode.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			X	RC		PC				PM	UM	OM	ZM	DM	IM
			Infinity control	Rounding control		Precision control				Precision	Underflow	Overflow	Zero divide	Denormal Operand	Invalid Operand

Exception masks

Outward rounding

RC are rounding mode bits; they can be set as follows:

00 = rounding to nearest or to even if equally distant (default)

01 = rounding down (toward -infinity)

10 = rounding up (toward +infinity)

11 = truncation (toward 0)

Exemplary assembler implementation (Microsof VS):

```
static unsigned short CRoundDown = 0x0400;

static unsigned short CRoundUp = 0x0800;

static unsigned short CRoundNear = 0xF3FF;

static unsigned short Ctrunc = 0x0C00;
```

```
void RoundDown() {
   __asm{ fstcw oldcw
   // to ensure the storage instruction is completed
   fwait
   mov ax, oldcw
   // clears only the RC bits, leaving all other bits unchanged
   // not necessary here because both bits will be set
   and ax, CRoundNear
   // this will set both bits of the RC field to the truncating mode
   // without affecting any of the other field's bits
   or ax, CRoundUp
   mov newcw, ax
   // load the modified Control Word
   fldcw newcw
}
```

Range bounding

The problem of range bounding is often called the main problem of interval computation.

The problem of computing the range of a function over a given interval (or a box in general case) can be formalized as follows: given a real-valued function $f\colon \mathrm{R}^n \supseteq D_f \to \mathrm{R}$ and a box $\boldsymbol{x}$, we are interested in the range

$$Rge(f \mid \boldsymbol{x}) = \{f(x) \mid x \in \boldsymbol{x} \cap D_f\}.$$

If f is continuous, then $Rge(f \mid \boldsymbol{x})$ is a closed interval.

For many classes of functions, computing $Rge(f \mid \boldsymbol{x})$ is an NP-hard problem.

Therefore, in practical situations, we often ask for an interval enclosure for $Rge(f \mid \boldsymbol{x})$, i.e., an interval $\boldsymbol{y}$ such that $Rge(f \mid \boldsymbol{x}) \subseteq \boldsymbol{y}$.

The tightest possible interval enclosure is simply $\mathrm{hull}\, Rge(f \mid \boldsymbol{x})$. If function is monotone, the hull can be obtained by computing the values of the function at respective endpoints.

Range bounding

If a real-valued function f is represented by an arithmetic expression f, then the interval enclosure $\boldsymbol{y}$ can be obtained by a (straightforward) interval evaluation of f over $\boldsymbol{x}$.

The interval evaluation relies on replacing the real variables by the corresponding interval variables and the real arithmetic operations by the corresponding interval operations.

Interval valued function $\boldsymbol{f} : \mathrm{IR}^n \supseteq \boldsymbol{D} \rightarrow \mathrm{IR}$ induced by expression f, is said to be the natural interval extension of f.

If the interval evaluation is defined, then $f(x) \in \boldsymbol{f}(\boldsymbol{x})$.

If $\boldsymbol{x}$ contains a point $x \notin D_f$, then the interval evaluation is not defined.

Range bounding

Sometimes, the interval evaluation is not defined even if $\boldsymbol{x} \subseteq D_f$…

Example. Consider a real-valued function $f(x) = 1/(x^2 + 2)$. The function is defined for all real numbers and takes values from the interval (0,1].

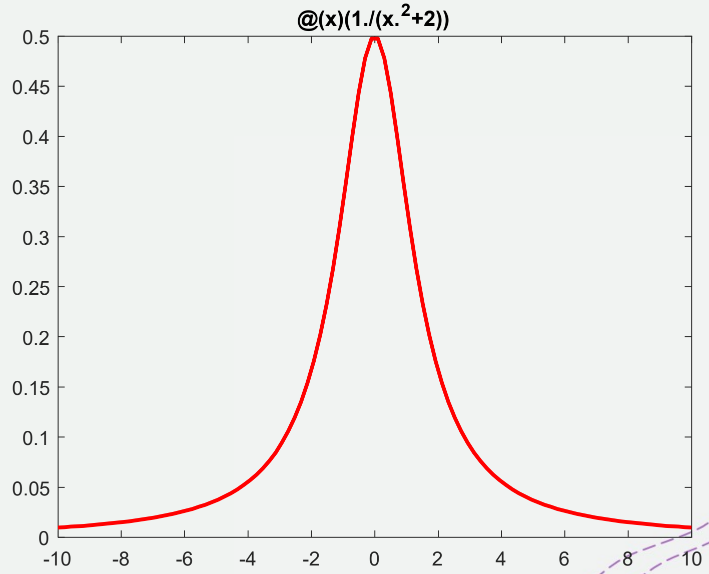
As it is well known $a^n = \underbrace{a * a \cdots * a}_{n-times}$, so we have

$$\boldsymbol{f}([-2,2]) = \frac{1}{([-2,2] * [-2,2] + 2)} = \frac{1}{([-4,4] + 2)} = \frac{1}{[-2,6]}$$

which means that $\boldsymbol{f}([-2,2])$ is not defined for $\boldsymbol{x} = [-2,2]$, (the division cannot be performed since the denominator contains 0) despite $[-2,2] \subset D = \mathcal{R}$.

Having properly defined the square function we will obtain:

$$\boldsymbol{f}([-2,2]) = \frac{1}{([-2,2]^2 + 2)} = \frac{1}{([0,4] + 2)} = \frac{1}{[2,6]} = \left[\frac{1}{6}, \frac{1}{2}\right]$$



Range bounding

…and the result much depends on the syntactic formulation of the expression for f. The difference between exact range and the computed range is called the overestimation.

Example. Consider a real-valued function $f(x) = x^2 - 2x + 1$. The function is defined for all real numbers.

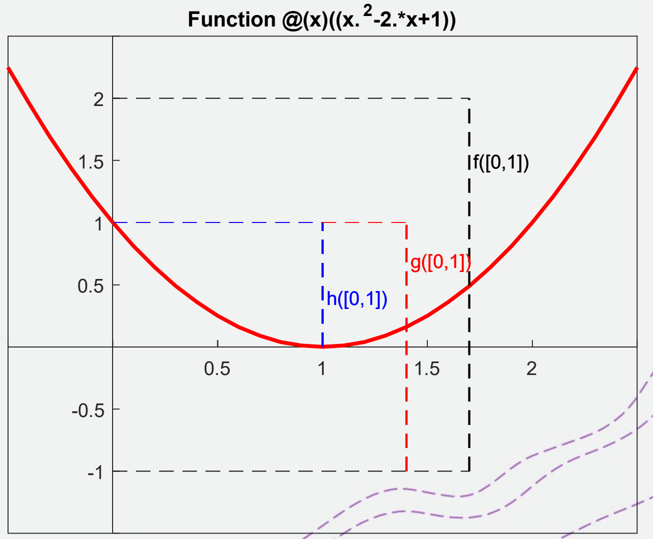
Let $\boldsymbol{x} = [0,1]$. Then
$\boldsymbol{f}([0,1]) = [0,1] * [0,1] - 2 * [0,1] + 1 = [-1,2]$

We can write $f(x)$ in the following two equivalent forms:
$g(x) = x * (x - 2) + 1$
$h(x) = (x - 1)^2$

Then

$\boldsymbol{g}([0,1]) = [0,1] * ([0,1] - 2) + 1 = [-1,1]$ (50% overestimation)

$\boldsymbol{h}([0,1]) = ([0,1] - 1)^2 = [-1,1]^2 = [0,1]$ (hull)



Range bounding

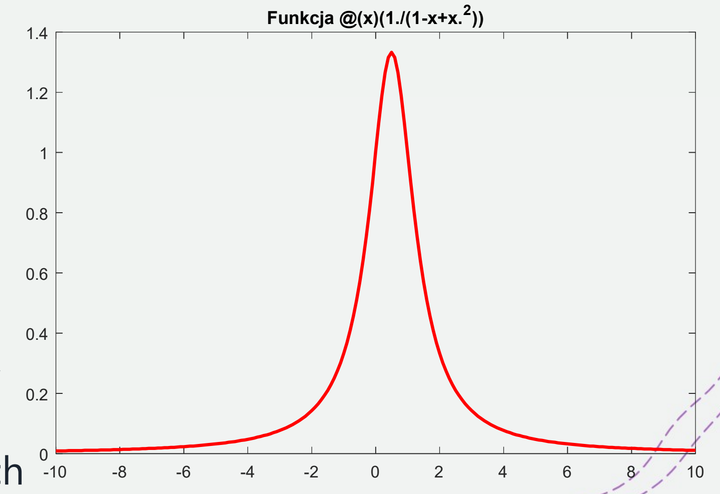
The overestimation can be arbitrarily large.

Example. Consider a real-valued function $f(x) = 1/(1 - x + x^2)$. The function is defined for all real numbers and takes value from (0,4/3].

However, for intervals $[0, t]$, the value

$$\boldsymbol{f}([0,t]) = \frac{1}{(1-[0,t]+[0,t]^2)} = \left[\frac{1}{1+t^2}, \frac{1}{1-t}\right]$$

If $t \rightarrow 1$ then the upper bound of the resulting interval tends to $+\infty$.



This huge overestimation is caused by multiple occurrences of variable x in the considered function. Subexpressions $1 - x$ and x^2 are calculated as if they were independent, i.e., x is treated as a different variable in each occurrence. Therefore, the range of function f is calculated from the expression $f(x_1, x_2) = 1/(1 - x_1 + x_2^2)$ over box $[0, t] \times [0, t]$.

We can rewrite f into an equivalent form as $g(x) = 4/(3 + (1 - 2 * x)^2)$. Then $\boldsymbol{g}(x) = [0,4/3]$ (hull).

Range bounding

The following theorem is one of the most important theorems of interval analysis.

Theorem. (Moore, Neumaier) *Let* $\boldsymbol{f}$ *be a natural interval extension of real function* f *represented by the arithmetic the expression* f *in which each variable* $x_i, (i = 1, \cdots, n)$ *occurs only once. Then, for all* $\boldsymbol{x} \subseteq D_f$

$$\boldsymbol{f}(\boldsymbol{x}) = \mathrm{hull}\, Rge(f \mid \boldsymbol{x}).$$

Unfortunately, for many expressions, it is difficult to rearrange them so that the assumptions of the above theorem are fulfilled.

The problem of multiple occurrence of variables in an expression is known as the dependency problem.

If it is not possible to obtain a single occurrence of each variable, we should at least try to reduce the number of occurrences of all variables as much as possible.

Symbolic manipulation of arithmetic expressions has proven to be useful in the detection and removal of multiple occurrences of variables.

Obviously, we can never be sure to obtain the best form of an expression, but even a small reduction in the number of occurrences of variables will improve the accuracy of the final enclosure.

Range bounding

The natural interval extension is only one of the infinitely many inclusion monotonic interval extensions of a real function.

If the reduction of the number of occurrences of variables is not satisfactory, some more-sophisticated interval extensions must be used to reduce the effect of the interval dependency (and overestimation). Below are mentioned the most known interval extensions:

+ centered form, generalized centered form
+ bi-centered form
+ mean value form
+ Taylor expansion
+ Bernstein expansion

They are useful, but often require automatic differentiation or the underlying formulas are quite complex.

Other ways to improve the results of interval computation?

Affine arithmetic (AA)

keeps track of first-order correlations between quantities

One of the possible ways to handle dependency/overestimation problem of IA is to use affine arithmetic, which is a model of self-validated numerical computation.

In affine arithmetic an unknown value $\tilde{x}$ is represented by an affine form, which is a 1-degree polynomial

$$\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n,$$

where the central value x_0 and the partial deviations x_i are finite floating-point numbers; the noise symbols (or dummy variables) ε_i are unknown but assumed to vary independently within the interval $[-1,1]$.

Each noise symbol ε_i stands for an independent component of the total uncertainty of the ideal quantity $\tilde{x}$ and the corresponding coefficient x_i gives the magnitude of that component.

Affine arithmetic

The sources of uncertainty in an affine form can be divided into:

+ External sources – already present in the input data,
+ Internal sources – due to affine approximations and rounding errors.

The semantics of affine forms is formalized by

The fundamental invariant of affine arithmetic states that, at any instant between AA operations, there is a single assignment of values from the interval $[-1,1]$ to each of the noise variables in use that makes the value of every affine form $\hat{x}$ equal to the true value of the corresponding ideal quantity $\tilde{x}$.

Affine forms ⇔ Intervals

Every affine form implies the interval for an unknown $\tilde{x}$:

$\tilde{x} \in [\hat{x}] = [x_0 - r, x_0 + r]$, where $r = \sum_{i=1}^{n} |x_i|$ is the total deviation.

The interval $[\hat{x}]$ is the tightest interval that contains all possible values of $\hat{x}$ assuming all ε_i vary independently within the interval $[-1,1]$.

Conversely, if $\tilde{x} \in [a, b]$, then it can be represented by the affine form

$$\hat{x} = \frac{a+b}{2} + \frac{b-a}{2}\varepsilon_k,$$

where ε_k is a new noise symbol (not present in any previous computation).

Affine forms ⇔ Intervals

Example. Consider two affine forms

$$\hat{x} = 20 - 4\varepsilon_1 \qquad\qquad + 2\varepsilon_3 + 3\varepsilon_4$$

$$\hat{y} = 10 - 2\varepsilon_1 + 1\varepsilon_2 \qquad\qquad - 1\varepsilon_4$$

The first form implies the interval $\boldsymbol{x} = [11,29]$, the second one implies the interval $\boldsymbol{y} = [6,14]$.

Example. Let $\boldsymbol{x} = [2,8]$, $\boldsymbol{y} = [-5,3]$, $\boldsymbol{z} = [-4,-1]$. Then, they can be replaced with the following affine forms

$$\hat{x} = 5 + 3\varepsilon_1$$

$$\hat{y} = -1 + 4\varepsilon_2$$

$$\hat{z} = -2.5 + 1.5\varepsilon_3$$

Affine arithmetic

The key feature of AA is that the same noise symbol ε_i may contribute to the uncertainty of two or more quantities (inputs, outputs, or intermediate results) $\hat{x}$ and $\hat{y}$ arising in the evaluation of an expression.

Thanks to the use of noise symbols AA keeps track of first-order correlations between quantities; therefore, it can provide much tighter intervals than conventional interval arithmetic, especially in long chained computations.

The sharing of noise symbols indicates some partial dependency between the underlying quantities $\tilde{x}$ and $\tilde{y}$ determined by the corresponding coefficients x_i and y_i.

The signs of x_i and y_i are not meaningful in themselves, because the sign of ε_i is arbitrary; but the relative sign of x_i and y_i defines the direction of the correlation.

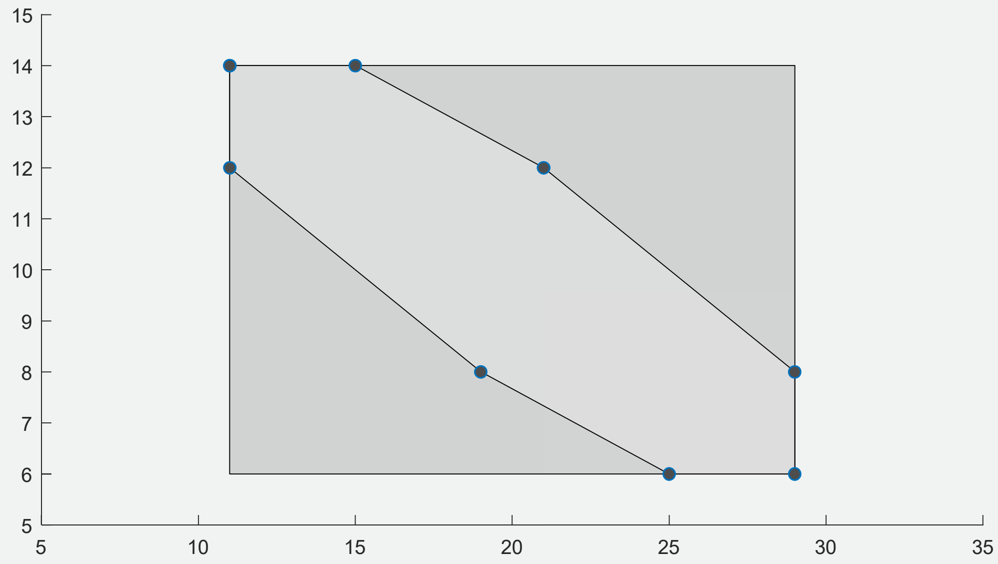
Affine arithmetic

Example. Consider two affine forms

$$\hat{x} = 20 - 4\varepsilon_1 \qquad + 2\varepsilon_3 + 3\varepsilon_4$$

$$\hat{y} = 10 - 2\varepsilon_1 + 1\varepsilon_2 \qquad - 1\varepsilon_4$$

From this data, we can tell that $\tilde{x}$ lies in the interval $\boldsymbol{x} = [11,29]$, and $\tilde{y}$ lies in the interval $\boldsymbol{y} = [6,14]$, i.e., the pair $(\tilde{x}, \tilde{y})$ lies somewhere in the box $= [11,29] \times [11,29]$ (dark gray rectangle).



However, since the two affine forms include the same noise variables, $\tilde{x}$ and $\tilde{y}$ are not independent, so they must lie in the light gray region, which is the set of all possible values of $(\hat{x}, \hat{y})$, when the noise variables $\varepsilon_1, \cdots, \varepsilon_4$ are independently chosen in $[-1,1]$. This set is called the joint range of the forms $\tilde{x}$ and $\tilde{y}$ and is denoted as $\langle \hat{x}, \hat{y} \rangle$.

Affine arithmetic

In order to perform computation on affine forms, all standard arithmetic operations, as well as other classical functions must be redefined for affine forms.

Affine-linear operations, such as addition and subtraction, on affine forms can be defined straightforwardly. Given two affine forms $\hat{x}, \hat{y}$ and $\alpha, \beta, \gamma \in \mathrm{R}$, the following holds

$$\alpha\hat{x} + \beta\hat{y} + \gamma = (\alpha x_0 + \beta y_0 + \gamma) + \sum_{i=1}^{n}(\alpha x_i + \beta y_i)\varepsilon_i.$$

Nonlinear operations usually result in nonlinear functions of the noise symbols, so they must be approximated by a respective affine form and the approximation error must be considered. Given $\hat{x}, \hat{y}$ and some non-linear function $z = f(x, y)$, we have

$$z = f(x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n, x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n) = f^*(\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n).$$

Then, we must find some affine function

$$f^a(\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n) = z_0 + \sum_{i=1}^{n} z_i\varepsilon_i + z_m\varepsilon_m.$$

that approximates f^* reasonably well on its domain and $z_m\varepsilon_m$ is the error of this approximation. For simplicity, the dependencies in higher order terms are neglected.

Revised affine arithmetic

In revised affine arithmetic (RAA) a revised affine form of length n is defined as a sum of a standard affine form and a term that represents all errors introduced during a computation (approximation and rounding errors), i.e.,

$$\hat{x} = x_0 + x_1\varepsilon_1 + \cdots + x_n\varepsilon_n + x_r[-1,1],$$

where $x_r[-1,1]$ with $x_r \geq 0$ is the so-called accumulation error.

Thanks to the use of accumulation error, the length of affine form is constant during the whole computation.

The accuracy of computation of revised affine forms is the same as the accuracy.

Revised affine arithmetic

Affine-linear operation

$$\alpha\hat{x} + \beta\hat{y} + \gamma = (\alpha x_0 + \beta y_0 + \gamma) + \sum_{i=1}^{n}(\alpha x_i + \beta y_i)\varepsilon_i + (|\alpha| x_i + |\beta| y_i)[-1,1],$$

Multiplication is challenging, there are several approaches:

+ Trivial approximation (quite rough but fast)

$$\hat{z} = x_0 y_0 + \sum_{i=1}^{n}(x_0 y_i + y_0 x_i)\varepsilon_i + z_r[-1,1],$$
$$z_r = |y_0| x_r + |x| y_r + (\|x\|_1 + x_r) \cdot (\|y\|_1 + y_r),$$

+ Min-error approximation (minimizes error but is more time consuming)

$$\hat{z} = x_0 y_0 + 0.5(R_{min} + R_{max}) + \sum_{i=1}^{n}(x_0 y_i + y_0 x_i)\varepsilon_i + z_r[-1,1],$$
$$z_r = |y_0| x_r + |x| y_r + 0.5(R_{max} - R_{min}),$$

where R_{min} and R_{max} are extremal values of f^* on the joint range $\langle\hat{x}, \hat{y}\rangle$.

Revised affine arithmetic

Example. Comparison of multiplication formulas. Consider two affine forms

$\hat{x} = 3.4 - 5.6\varepsilon_1 + \varepsilon_3 - \varepsilon_4 + 2.3\varepsilon_6 + 0.6\varepsilon_7,$

$\hat{y} = 1.4 + 3.6\varepsilon_2 - \varepsilon_3 + 2\varepsilon_4 + 2.3\varepsilon_7 - 4.5\varepsilon_8.$

The results of various multiplication formulas:

min-error: [-174.52, 156.32]

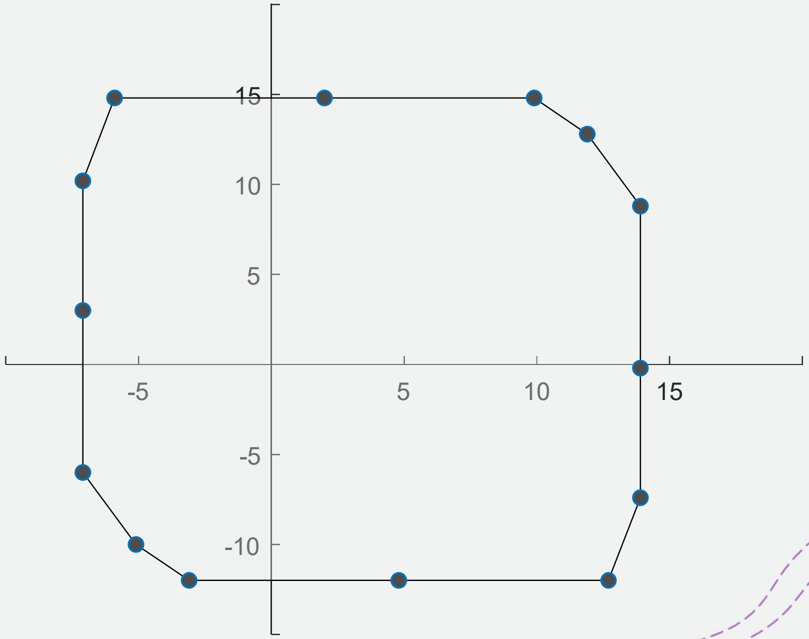
trivial: [-190.6, 200.12] (15%)

Kolev (standard): [-189.22, 197.12] (14%)

Miyajima: [-185.62, 193.52] (13%)

Rump & Kashiwagi: [-184.93, 192.83] (12%)

Min range for the product [-152.4,152.32]



Amount of overestimation of $\boldsymbol{b}$ over $\boldsymbol{a}$: $O(\boldsymbol{a},\boldsymbol{b}) = \left(1 - \frac{a^\Delta}{b^\Delta}\right) \cdot 100\%$

Revised affine arithmetic

Division

$$\frac{\hat{x}}{\hat{y}} = \frac{x_0}{y_0} + \sum_{i=1}^{n}\left(x_i - \frac{x_0}{y_0}y_i\right)\varepsilon_i + \left(x_r + \left|\frac{x_0}{y_0}\right|y_r\right)[-1,1].$$

The division can also be defined as $\hat{x} * (1/\hat{y})$, where the term $1/\hat{y}$ can be computed from the approximation of the function $f(x) = 1/x$, which is monotonic in its subdomains $(-\infty, 0)$ and $(0, +\infty)$.

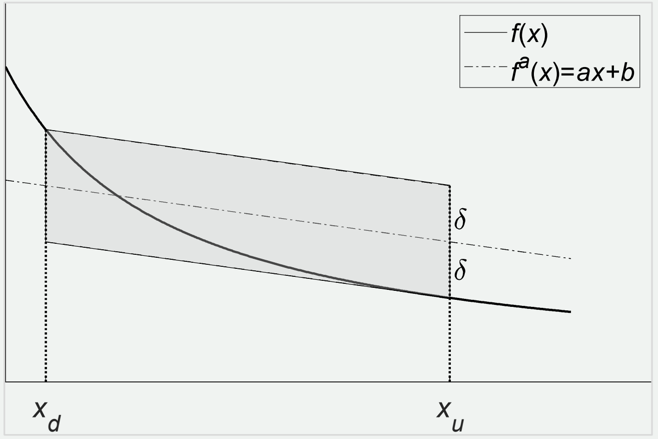
Monotonic functions

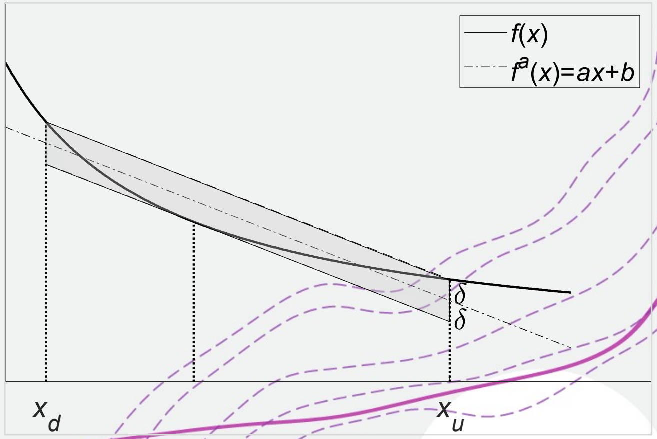
There are generally two approaches to computing the affine approximation of monotonic functions: the min-range approximation (left) and min-error approximation (right).

The affine approximation of $z = f(x)$ on the interval is given by

$$\hat{z} = ax_0 + b + \sum_{i=1}^{n} ax_i\varepsilon_i + \delta[-1,1],$$

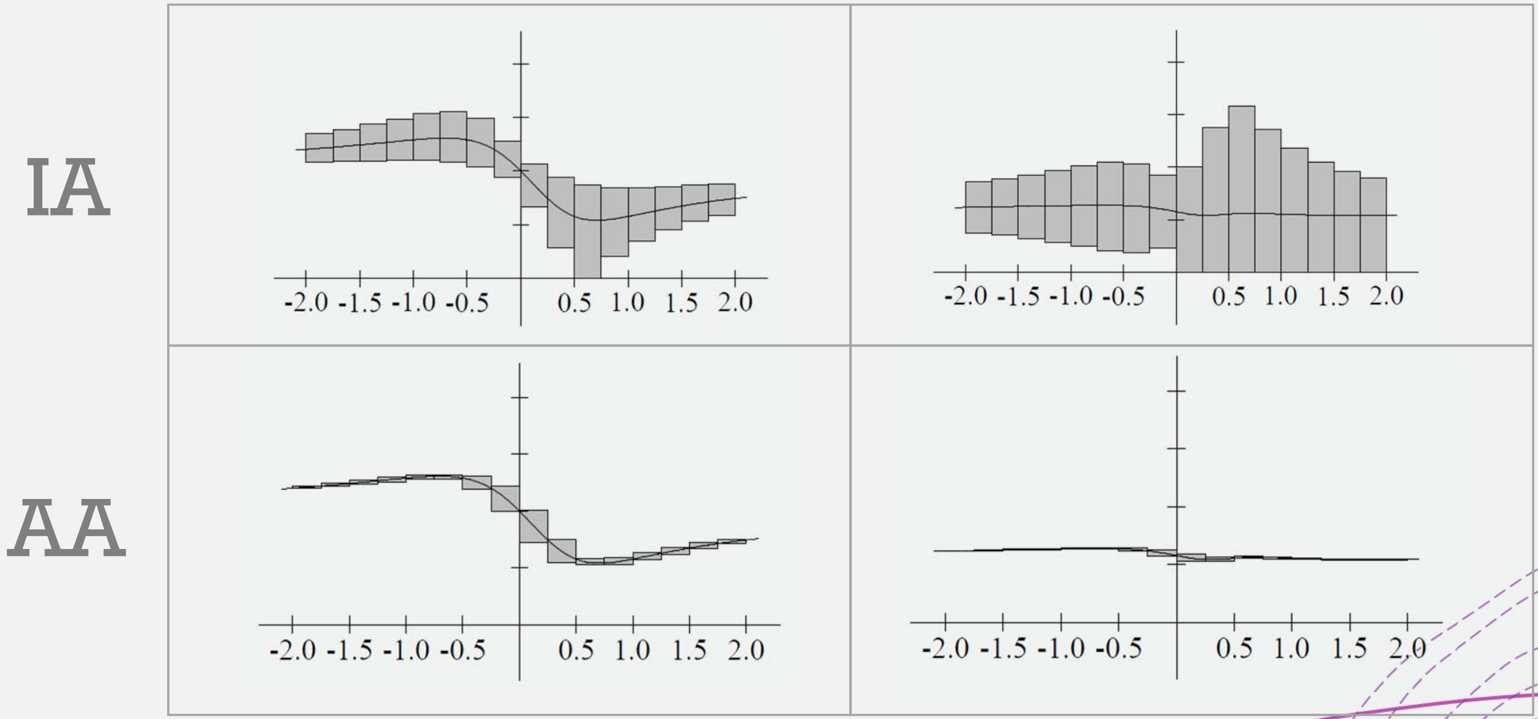
where a and b are, respectively, the slope and the intercept of the straight line $f^a(x)$ that approximates $f(x)$, and δ is the error of this approximation.





Comparison on IA and AA

Example. Consider the function $g(x) = \sqrt{x^2 - x + 1}/\sqrt{x^2 + 1/2}$ (black line). The figure below shows the interval evaluation of the natural extension of $g(x)$ over the interval $[-2,2]$ (left top) and the evaluation of the iterate
$h(x) = g(g\ (x))$ (right top). We can see error explosion. The same computation with affine arithmetic gives tight bounds ($g(x)$ – bottom left, $h(x)$ – bottom right); the iteration contracts to the line.



Practical problems

Drawing implicit curves

An implicit curve is given by the formula $f(x,y) = 0$.

If the formula for $f(x,y)$ is simple, then we can represent y as a function of x and draw a graph.

If the formula for $f(x,y)$ is complex, like for example,

$$f(x,y) = \left(\frac{15}{4}\right) + 8x - 16x^2 + 8y - 112xy + 128.0x^2y - 16y^2 + 128xy^2 - 128x^2y^2 = 0,$$

then it is difficult to represent y as a function of x.

So, in general case the problem of drawing implicit curves is considered as to the problem of finding all zeros of f in a box.

Practical problems

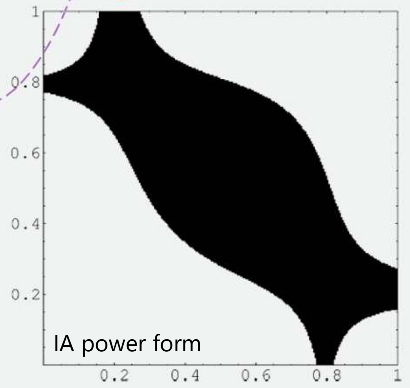
Drawing implicit curves

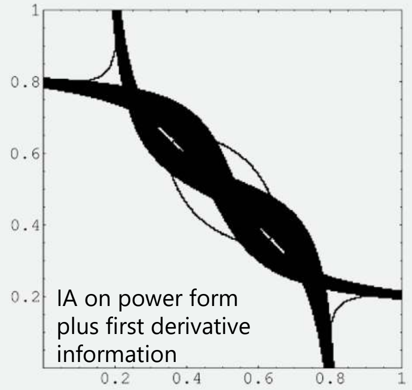
```
PROCEDURE   Quadtree($\underline{x}, \overline{x}, \underline{y}, \overline{y}$):
  $\mathbf{F}$ = RangeEvaluation($\underline{x}, \overline{x}, \underline{y}, \overline{y}$);
  if $\underline{F} \leqslant 0 \leqslant \overline{F}$ then
    if $\overline{x} - \underline{x} <$ PixelSize AND $\overline{y} - \underline{y} <$ PixelSize then
      PlotPixel($\underline{x}, \overline{x}, \underline{y}, \overline{y}$)
    else Subdivide($\underline{x}, \overline{x}, \underline{y}, \overline{y}$).
```

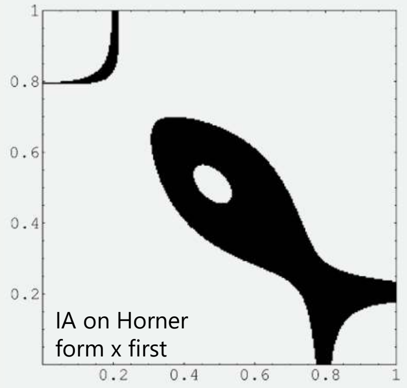
```
PROCEDURE   Subdivide($\underline{x}, \overline{x}, \underline{y}, \overline{y}$):
  $\check{x} = (\underline{x} + \overline{x})/2$;
  $\check{y} = (\underline{y} + \overline{y})/2$;
  Quadtree($\underline{x}, \check{x}, \underline{y}, \check{y}$);
  Quadtree($\underline{x}, \check{x}, \check{y}, \overline{y}$);
  Quadtree($\check{x}, \overline{x}, \check{y}, \overline{y}$);
  Quadtree($\check{x}, \overline{x}, \underline{y}, \check{y}$).
```

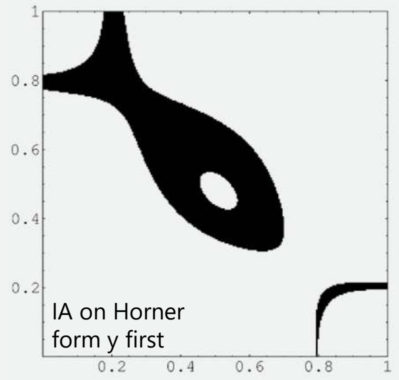
Practical problems

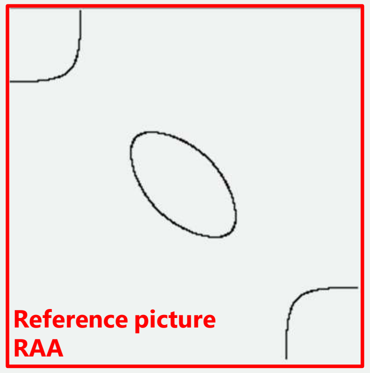
Drawing implicit curves

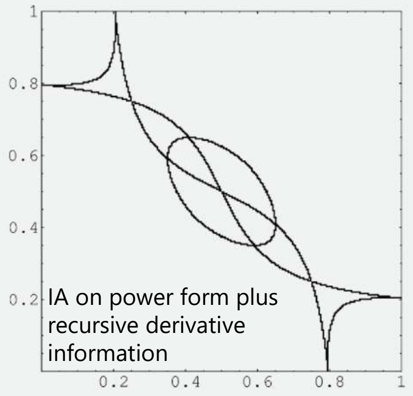


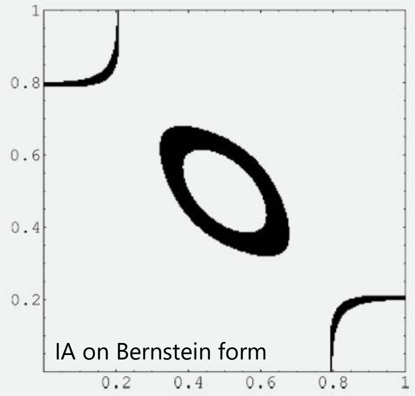


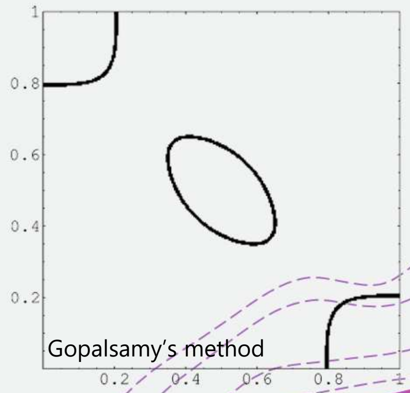












Practical problems

Interval Global Optimization

is useful for finding global extrema of functions, especially when function has many global optima.

The nonlinear optimization problem is usually formulated as a minimization problem of a nonlinear function with additional constraints:

$$f(x) \to \min, x \in \mathrm{R}^n$$

$$p(x) \le 0, p: \mathrm{R}^n \to \mathrm{R}^m$$
$$q(x) = 0, p: \mathrm{R}^n \to \mathrm{R}^r$$

where $f(x)$ is the function for which the minimum value is sought, x is the independent variable, $p(x)$ is a set of inequality constraints, and $q(x)$ a set of equality constraints.

Practical problems

Interval Global Optimization

Branch & Bound is an algorithm design paradigm for discrete and combinatorial optimization problems.

During the branch and bound loop, sub-boxes that do not contain the global minimum are deleted, while boxes that could contain the global minimum are contracted and divided.

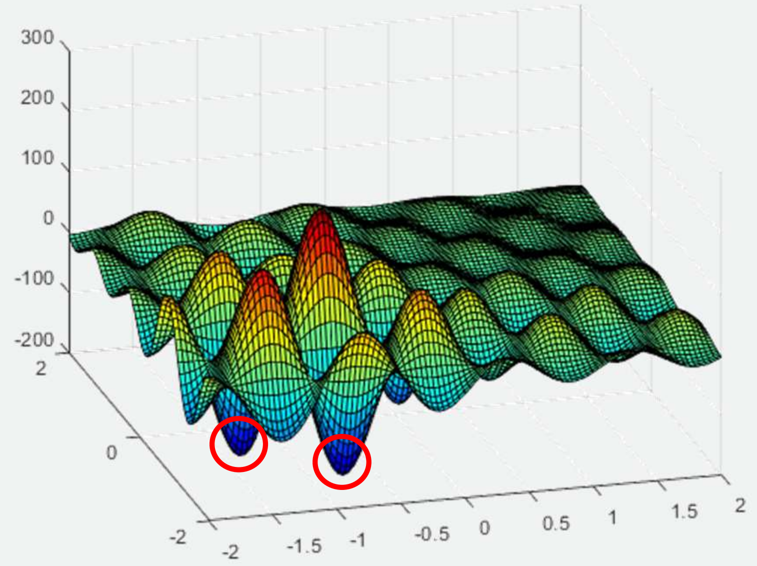
```
0.    Initializaton: Determine integer search space and fill $L_{start} = [\boldsymbol{\alpha}_{ij}]$
OUTERLOOP
1.    Initialize list $L = L_{start}$
      INNERLOOP
      a.    Take the interval vector $[\boldsymbol{\alpha}_{ij}]$ from the list $L$
      b.    Determine the circle intersection $[\boldsymbol{\delta}_{ij}]$
      c.    If $[\boldsymbol{\delta}_{ij}] = \emptyset$, proceed with the next interval vector in the list $L$ else
            goto step (d).
      d.    If $[\boldsymbol{\alpha}_{ij}]$ corresponds to a single integer per dimension then store
            $([\boldsymbol{\alpha}_{ij}], [\boldsymbol{\delta}_{ij}])$ in $L_{final}$,
            else bisect $[\boldsymbol{\alpha}_{ij}]$ and put it in the list $L$.
      e.    If $L = \emptyset$ then break innerloop
      END INNERLOOP
2.    If multiple solutions exist then use data from the next epoch
      with $L_{start} = L_{final}$, else break outerloop
END OUTERLOOP
```

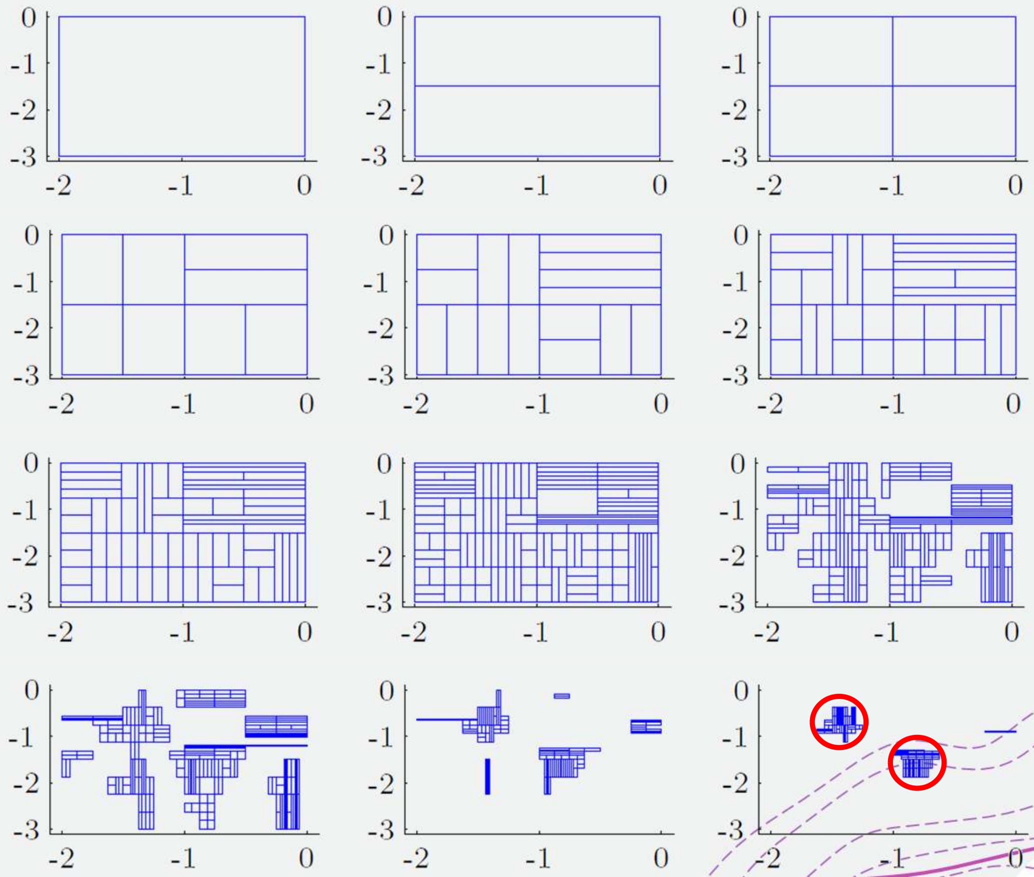
Practical problems

Interval Global Optimization

Consider Shubert's function

$$f(\mathbf{x}) = \left(\sum_{i=1}^{5} i\cos((i+1)x_1 + i)\right)\left(\sum_{i=1}^{5} i\cos((i+1)x_2 + i)\right)$$





Practical problems

Solving interval parametric linear systems (IPLS)

IPLS are encountered, e.g., in electrical engineering, structural engineering

If uncertainty is introduced into a model described by a system of linear equations, then the coefficients of the matrix are functions of parameters that vary within prescribed intervals. Assume that the system has the following form

$$\begin{pmatrix} 1 & p_2^2 & p_2 \\ p_1 & 2 & p_1 \\ p_2 & p_1 & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ p_1^2 \\ p_2^2 \end{pmatrix}, \quad p_1 \in [0,1], p_2 \in [0,0.9]$$

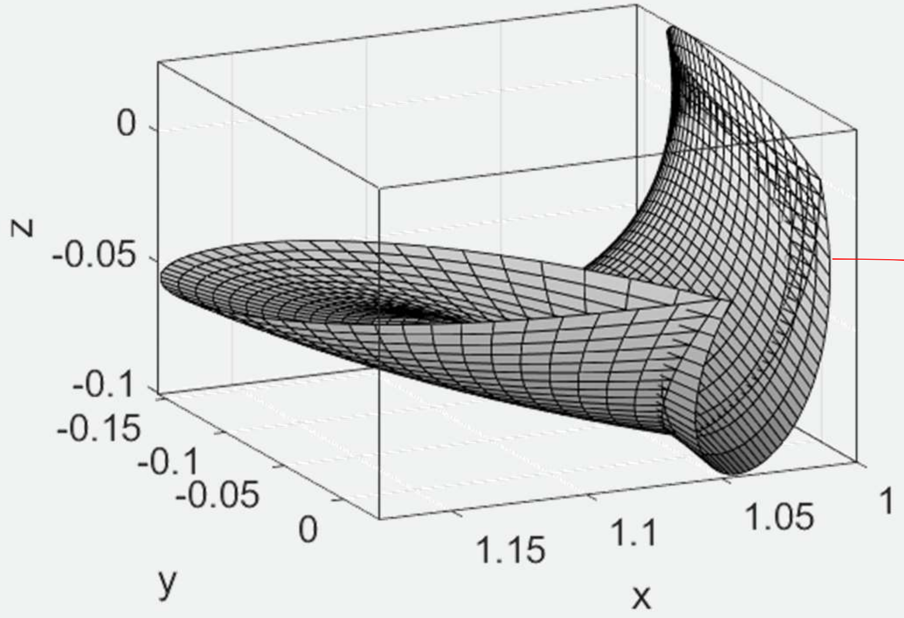
Corresponding interval (nonparametric) linear system

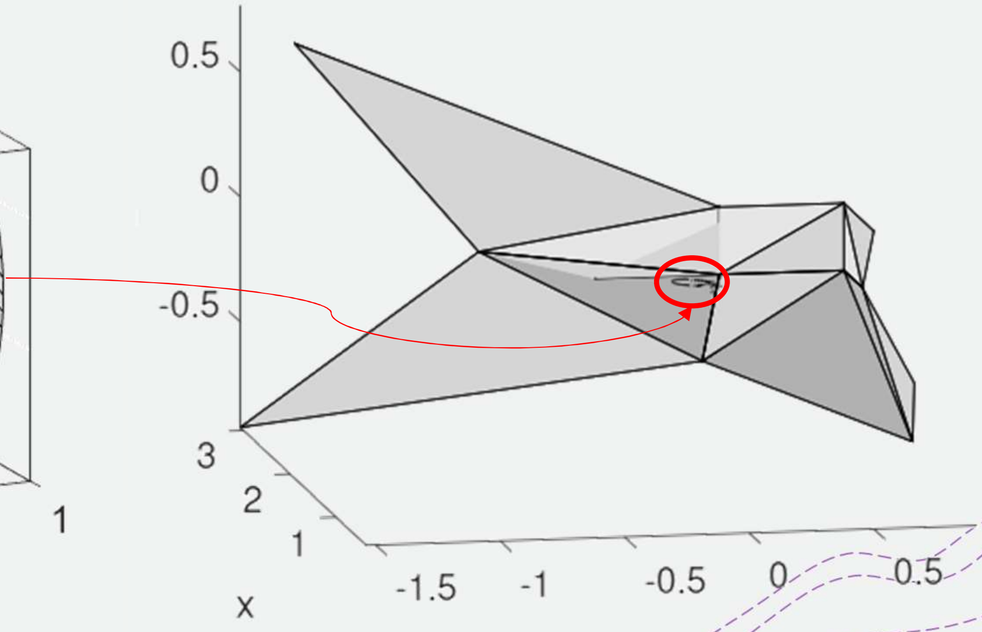
$$\begin{pmatrix} 1 & [0.81] & [0,0.9] \\ [0,1] & 2 & [0,1] \\ [0,0.9] & [0,1] & 3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ [0,1] \\ [0,0.81] \end{pmatrix}$$

Practical problems

Solving interval and interval parametric linear systems

Comparison of solution sets: parametric (left) and non-parametric with (right) (notice the parametric solution inside the red ellipse).





Practical problems

Game theory

Cooperative games under incomplete information (partially defined cooperative games) were first considered by Willson (1993).

Two-person zero-sum game with uncertain payoffs ([1], October 2020).

If the payoffs are uncertain, then the entries of the payoffs matrix are uncertain.

If the payoff matrix has a saddle point, then the optimal strategy can be obtained as follows:

1. Select the minimum element of each row of the payoff matrix using ordering of intervals and choose the maximum value among them. It is called the maximin value.
2. Select the maximum element of each column of the payoff matrix using ordering of intervals and choose the minimum value among them. It is called the minimax value.
3. The position where the maximin value equals the minimax value in a game is called the saddle point and the corresponding strategies of the saddle point are called optimum strategies. The payoff at the saddle point is called the value of the game.

[1] https://www.researchgate.net/publication/346090311_Research_article_game_theory_problems_using_interval_parameters

Practical problems

Game theory

In order to find minimax and maximin values, we must know how to compare intervals.

Let $\boldsymbol{a} = [\underline{a}, \bar{a}]$, $\boldsymbol{b} = [\underline{b}, \bar{b}]$, then (according to [1])

$\boldsymbol{a} < \boldsymbol{b}$ iff $a^c < b^c$.

$\boldsymbol{a} > \boldsymbol{b}$ iff $a^c > b^c$.

$\boldsymbol{a} < \boldsymbol{b}$ iff $a^\Delta < b^\Delta$ whenever $a^c = b^c$.

$\boldsymbol{a} > \boldsymbol{b}$ iff $a^\Delta > b^\Delta$ whenever $a^c = b^c$.

$\boldsymbol{a} = \boldsymbol{b}$ whenever $a^\Delta = b^\Delta$ and $a^c = b^c$.

[1] https://www.researchgate.net/publication/346090311_Research_article_game_theory_problems_using_interval_parameters

Practical problems

Game theory

If there is no saddle point, the optimum strategies and the value of the game can be computed as follows. Let the interval payoff matrix be given as

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a} & \boldsymbol{b} \\ \boldsymbol{c} & \boldsymbol{d} \end{bmatrix}.$$

Then, the optimum mixed strategies are given by

$$S_A = \begin{bmatrix} A_1 & A_2 \\ p_1 & p_2 \end{bmatrix} \text{ and } S_B = \begin{bmatrix} B_1 & B_2 \\ q_1 & q_2 \end{bmatrix},$$

where

$$p_1 = \frac{\boldsymbol{d}-\boldsymbol{c}}{(\boldsymbol{a}+\boldsymbol{d})-(\boldsymbol{b}+\boldsymbol{c})} \quad \text{and} \quad p_2 = 1 - p_1,$$

$$q_1 = \frac{\boldsymbol{d}-\boldsymbol{b}}{(\boldsymbol{a}+\boldsymbol{d})-(\boldsymbol{b}+\boldsymbol{c})} \quad \text{and} \quad q_2 = 1 - q_1,$$

and the value of the game is $V = \frac{\boldsymbol{ad}-\boldsymbol{bc}}{(\boldsymbol{a}+\boldsymbol{d})-(\boldsymbol{b}+\boldsymbol{c})}$.

[1] https://www.researchgate.net/publication/346090311_Research_article_game_theory_problems_using_interval_parameters

Practical problems

Game theory

Example. Consider the following 2×2 game with payoff as intervals. We solve it by existing method.

$$\boldsymbol{A} = \begin{array}{c} \\ A_1 \\ A_2 \end{array} \begin{array}{cc} B_1 & B_2 \\ \end{array} \begin{bmatrix} [1,3] & [4,6] \\ [6,8] & [2,4] \end{bmatrix}.$$

Test for the saddle point:

Row minima: row1: [1,3] (2<5), row2: [2,4] (3<7), which gives minimax = [2,4] (3>2)

Columns maxima: col1: [6,8] (7>2), col2: [4,6] (5>3), which gives maximin = [4,6] (5<7)

There is no saddle point: [2,4] ≠ [4,6]. We must compute optimum strategies and the value of the game.

$$p_1 = \frac{[2,4]-[6,8]}{[3,7]-[10,14]} = \left[\frac{2}{11}, 2\right],\ p_2 = \left[-1, \frac{9}{11}\right],\ q_1 = \frac{[2,4]-[4,6]}{[3,7]-[10,14]} = \left[0, \frac{4}{3}\right],\ q_2 = \left[-\frac{1}{3}, 1\right].$$

$$S_A = \begin{bmatrix} A_1 & A_2 \\ \left[\frac{2}{11}, 2\right] & \left[-1, \frac{9}{11}\right] \end{bmatrix}, S_A = \begin{bmatrix} B_1 & B_2 \\ \left[0, \frac{4}{3}\right] & \left[-\frac{1}{3}, 1\right] \end{bmatrix}, V = \frac{[1,3][2,4]-[4,6][6,8]}{[-11,3]} = \frac{[2,12] \quad [24,48]}{[-11,3]} = \frac{[-46,-12]}{[-11,-3]} = \left[\frac{12}{11}, \frac{46}{3}\right].$$

[1] https://www.researchgate.net/publication/346090311_Research_article_game_theory_problems_using_interval_parameters

Interval and affine software

- INTLAB (INTerval LABoratory) – The Matlab/Octave toolbox for Reliable Computing (interval and affine computation) (copyright)
  Developed by Prof. Dr. Siegfried M. Rump form Institute for Reliable Computing of Hamburg University of Technology
- Interval - The Octave based interval package for real-valued interval arithmetic (GPL-3.0+)
  Developed by Oliver Heimlich
- libaffa – C++ Affine Arithmetic library for GNU/Linux (GNU Lesser General Public License v2.1).
  Developed by Olivier Gay, David Coeurjolly, Nathan Hurst
- aaflib - an Affine Arithmetic C++ Library (free software)
  Developed at the Institute of Microelectronic Systems in Hannover
- YalAA (Yet Another Library for Affine Arithmetic)
  Developed by Stefan Kiel from University of Duisburg-Essen