An efficient midpoint-radius implementation to handle symmetric fuzzy intervals

Manuel Marin, David Defour and Federico Milano

DALI, University of Perpignan Via Domitia,
Electricity Research Centre, University College Dublin

Introduction

- Interval arithmetic is a useful tool to provide reliable results in computations with uncertain data (e.g., the speed is between 100 and 120 km/h).

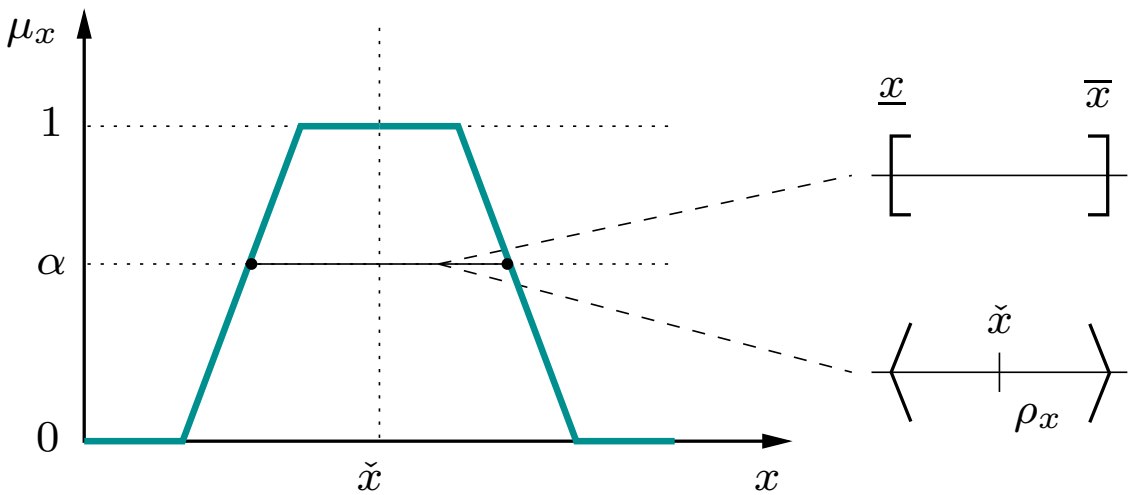
- In addition, *fuzzy* interval arithmetic provides an answer when the information in the knowledge base is more ambiguous and imprecise (e.g., the speed is high).

- Most fuzzy interval implementations are based on the lower-upper representation format. *This talk discusses the use of midpoint-radius to improve performance when dealing with symmetric fuzzy intervals.*

Outline

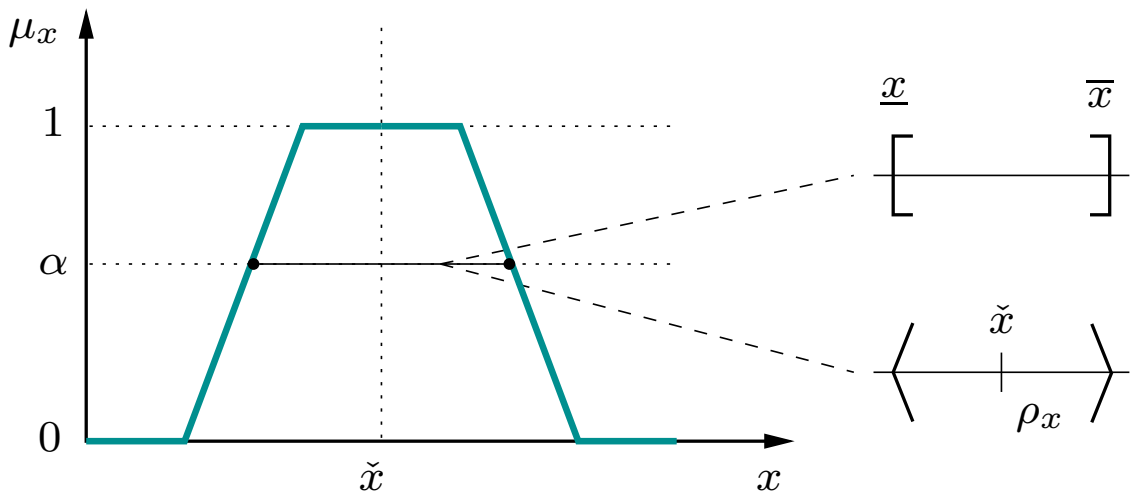
Fuzzy intervals

Fuzzy intervals are characterized by a membership function, $\mu : \mathbb{R} \to [0, 1]$.
For each $\alpha \in [0, 1]$, an α-*cut* is obtained:



Fuzzy intervals

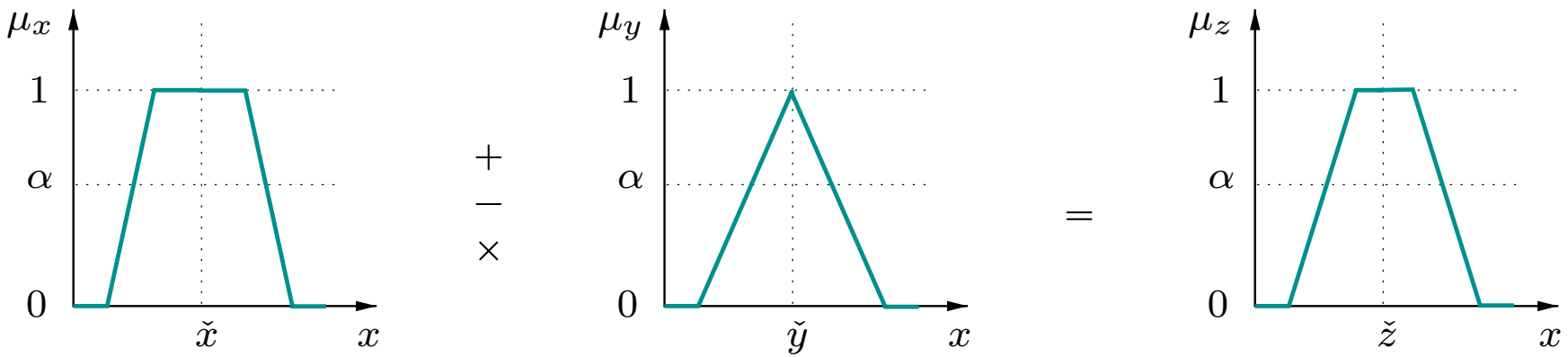
Fuzzy intervals are characterized by a membership function, $\mu : \mathbb{R} \to [0, 1]$.
For each $\alpha \in [0, 1]$, an α-*cut* is obtained:



If the fuzzy interval is symmetric, then all the α-cuts have the same midpoint.

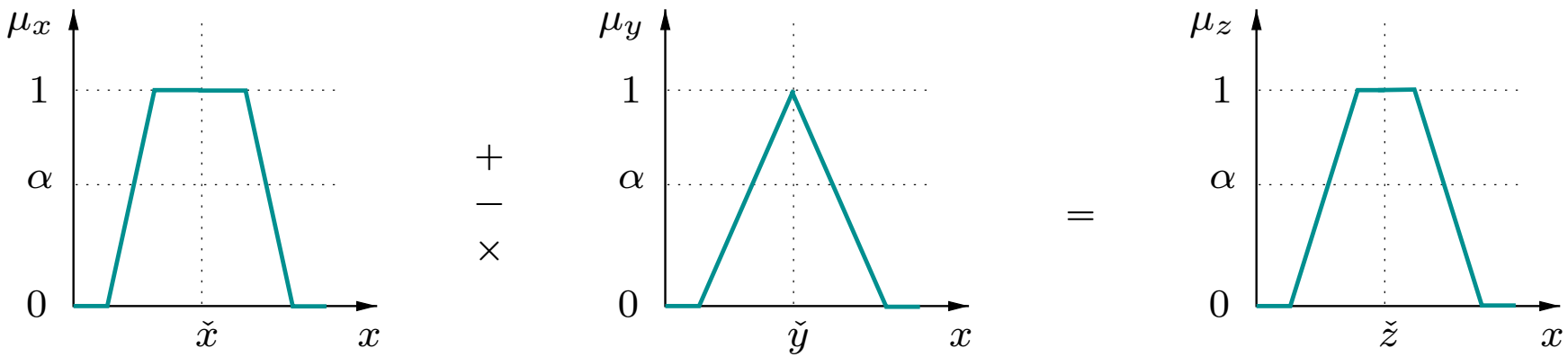
Fuzzy interval arithmetic

Fuzzy arithmetic operations are decomposed into a series of interval operations, one per α-cut:



Fuzzy interval arithmetic

Fuzzy arithmetic operations are decomposed into a series of interval operations, one per α-cut:



Addition, subtraction and multiplication of fuzzy intervals preserve symmetry.

Lower-upper representation

$$[\underline{x}, \overline{x}] = \{x \in \mathbb{R} \colon \underline{x} \leqslant x \leqslant \overline{x}\}.$$

Lower-upper representation

$$[\underline{x}, \overline{x}] = \{x \in \mathbb{R} \colon \underline{x} \leqslant x \leqslant \overline{x}\}.$$

Rounded interval arithmetic in the lower-upper format

$$\begin{aligned}
x + y &= [\bigtriangledown(\underline{x} + \underline{y}), \bigtriangleup(\overline{x} + \overline{y})], \\
x - y &= [\bigtriangledown(\underline{x} - \overline{y}), \bigtriangleup(\overline{x} - \underline{y})], \\
x \cdot y &= [\bigtriangledown(\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})), \bigtriangleup(\max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y})], \\
\frac{1}{y} &= \left[\bigtriangledown\left(\frac{1}{\overline{y}}\right), \bigtriangleup\left(\frac{1}{\underline{y}}\right)\right], \quad 0 \notin y.
\end{aligned}$$

$\bigtriangledown$: rounding downwards
$\bigtriangleup$: rounding upwards

Midpoint-radius representation

$$\langle \check{x}, \rho_x \rangle = \{x \in \mathbb{R} \colon |x - \check{x}| \leqslant \rho_x\}$$

Midpoint-radius representation

$$\langle \check{x}, \rho_x \rangle = \{x \in \mathbb{R} \colon |x - \check{x}| \leqslant \rho_x\}$$

Rounded interval arithmetic in the midpoint-radius format

$$\begin{aligned}
x \pm y &= \langle \square(\check{x} \pm \check{y}), \triangle(\epsilon' |\check{x} \pm \check{y}| + \rho_x + \rho_y) \rangle, \\
x \cdot y &= \langle \square(\check{x}\check{y}), \triangle(\eta + \epsilon' |\check{x}\check{y}| + (|\check{x}| + \rho_x)\rho_y + |\check{y}|\rho_x) \rangle, \\
\frac{1}{y} &= \left\langle \square\left(\frac{1}{\check{y}}\right), \triangle\left(\eta + \epsilon' \left|\frac{1}{\check{y}}\right| + \frac{-\rho_y}{|\check{y}|(\rho_y - |\check{y}|)}\right) \right\rangle, \quad 0 \notin y.
\end{aligned}$$

$\square$: rounding to nearest
$\triangle$: rounding upwards
ϵ': relative rounding error divided by 2
η: smallest representable floating-point number

Outline

Algorithms

Algorithm 1 Fuzzy multiplication in the lower-upper format.

Input: Fuzzy operands x and y.
Output: Fuzzy result $z = x \cdot y$.
1: **for** i in $1, \ldots, N$ **do**
2: $\quad \underline{z}_i = \bigtriangledown(\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}))$
3: $\quad \overline{z}_i = \bigtriangleup(\max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}))$

Requires $14N$ floating-point operations.

Algorithms

Algorithm 1 Fuzzy multiplication in the lower-upper format.

Input: Fuzzy operands x and y.
Output: Fuzzy result $z = x \cdot y$.
1: **for** i in $1, \ldots, N$ **do**
2: $\quad \underline{z}_i = \bigtriangledown(\min(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}))$
3: $\quad \overline{z}_i = \bigtriangleup(\max(\underline{x}\underline{y}, \underline{x}\overline{y}, \overline{x}\underline{y}, \overline{x}\overline{y}))$

Requires $14N$ floating-point operations.

Algorithm 2 Symmetric fuzzy multiplication in the midpoint-radius format.

Input: Symmetric fuzzy operands x and y.
Output: Symmetric fuzzy result $z = x \cdot y$.
1: $\check{z} = \square(\check{x}\check{y})$
2: **for** i in $1, \ldots, N$ **do**
3: $\quad \rho_{z,i} = \bigtriangleup(\eta + \epsilon'|\check{z}| + (|\check{x}| + \rho_{x,i})\rho_{y,i} + |\check{y}|\rho_{x,i})$

Requires $6 + 5N$ floating-point operations.

Performance considerations

Table 1: Number of floating-point instructions per arithmetical operation, for different data types.

Data type	**Addition**	**Multiplication**	**Inversion**
Scalar	1	1	1
Lower-upper interval	2	14	2
Midpoint-radius interval	5	11	9
Lower-upper fuzzy	$2N$	$14N$	$2N$
Midpoint-radius fuzzy	$3 + 2N$	$6 + 5N$	$5 + 5N$

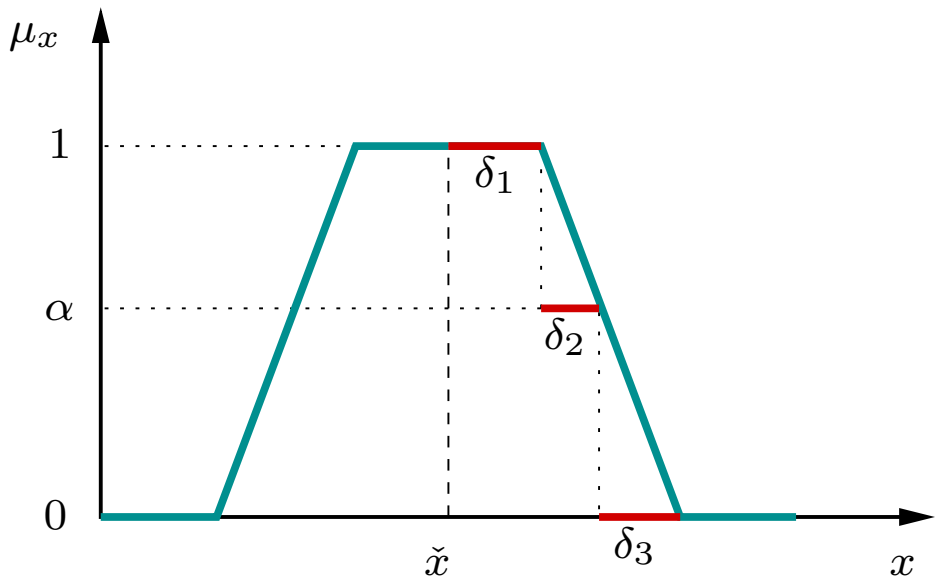
Table 2: Memory requirements of different data types.

Data type	**Memory usage**
Scalar	1
Lower-upper interval	2
Midpoint-radius interval	2
Lower-upper fuzzy	$2N$
Midpoint-radius fuzzy	$1 + N$

N: number of α-cuts

Radius increments

Fuzzy intervals can also be represented in terms of midpoint and radius increments:



$$\rho_i = \sum_{k=1}^{i} \delta_k = \rho_{i-1} + \delta_i.$$

Algorithms

Algorithm 3 Symmetric fuzzy addition in the midpoint-increment format.

Input: Symmetric fuzzy operands x and y.
Output: Symmetric fuzzy result $z = x + y$.

1: $\check{z} = \square(\check{x} + \check{y})$
2: $\delta_{z,1} = \triangle(\frac{1}{2}\epsilon|\check{z}| + \delta_{x,1} + \delta_{y,1})$
3: **for** i in $2, \ldots, N$ **do**
4: $\quad \delta_{z,i} = \triangle(\delta_{x,i} + \delta_{y,i})$

Requires $1 + 4N$ operations, fewer than midpoint-radius.

Algorithms

Algorithm 4 Symmetric fuzzy multiplication in the midpoint-increment format.

Input: Symmetric fuzzy operands x and y.
Output: Symmetric fuzzy result $z = x \cdot y$.

1: $\check{z} = \square(\check{x}\check{y})$
2: $t_{x,1} = \triangle(|\check{x}| + \delta_{x,i})$
3: $t_{y,1} = \triangle(|\check{y}|)$
4: $\delta_{z,1} = \triangle(\eta + \frac{1}{2}\epsilon|\check{z}| + t_{x,1}\delta_{y,1} + t_{y,1}\delta_{x,1})$
5: **for** i in $2, \ldots, N$ **do**
6: $\quad t_{x,i} = \triangle(t_{x,i-1} + \delta_{x,i})$
7: $\quad t_{y,i} = \triangle(t_{y,i-1} + \delta_{y,i})$
8: $\quad \delta_{z,i} = \triangle(\eta + \frac{1}{2}\epsilon|\check{z}| + t_{x,i}\delta_{y,i} + t_{y,i}\delta_{x,i})$

Requires $6 + 5N$ operations, the same as midpoint-radius.

Performance considerations

Table 3: Number of floating-point instructions per arithmetical operation, for different data types.

Data type	Addition	Multiplication	Inversion
Scalar	1	1	1
Lower-upper interval	2	14	2
Midpoint-radius interval	5	11	9
Lower-upper fuzzy	$2N$	$14N$	$2N$
Midpoint-radius fuzzy	$3 + 2N$	$6 + 5N$	$5 + 5N$
Midpoint-increment fuzzy	$4 + N$	$6 + 5N$	$5 + 5N$

Table 4: Memory requirements of different data types.

Data type	Memory usage
Scalar	1
Lower-upper interval	2
Midpoint-radius interval	2
Lower-upper fuzzy	$2N$
Midpoint-radius fuzzy	$1 + N$
Midpoint-increment fuzzy	$1 + N$

N: number of α-cuts

Error analysis

Absolute error in the midpoint-radius format ($i \geqslant 2$)

$$\rho_{z,i} = (\epsilon'|\check{z}| + \rho_{x,i}) + \rho_{y,i}.$$

$$\begin{aligned} E(\rho_{z,i}) &\leqslant \epsilon \left(\left| \epsilon'|\check{z}| + \rho_{x,i}) \right| + \left| \epsilon'|\check{z}| + \rho_{x,i} + \rho_{y,i} \right| \right) \\ &= \epsilon \left(\epsilon|\check{z}| + 2\rho_{x,i} + \rho_{y,i} \right). \end{aligned}$$

Error analysis

Absolute error in the midpoint-radius format ($i \geqslant 2$)

$$\rho_{z,i} = (\epsilon'|\check{z}| + \rho_{x,i}) + \rho_{y,i}.$$

$$\begin{aligned} E(\rho_{z,i}) &\leqslant \epsilon \left(\left| \epsilon'|\check{z}| + \rho_{x,i}) \right| + \left| \epsilon'|\check{z}| + \rho_{x,i} + \rho_{y,i} \right| \right) \\ &= \epsilon \left(\epsilon|\check{z}| + 2\rho_{x,i} + \rho_{y,i} \right). \end{aligned}$$

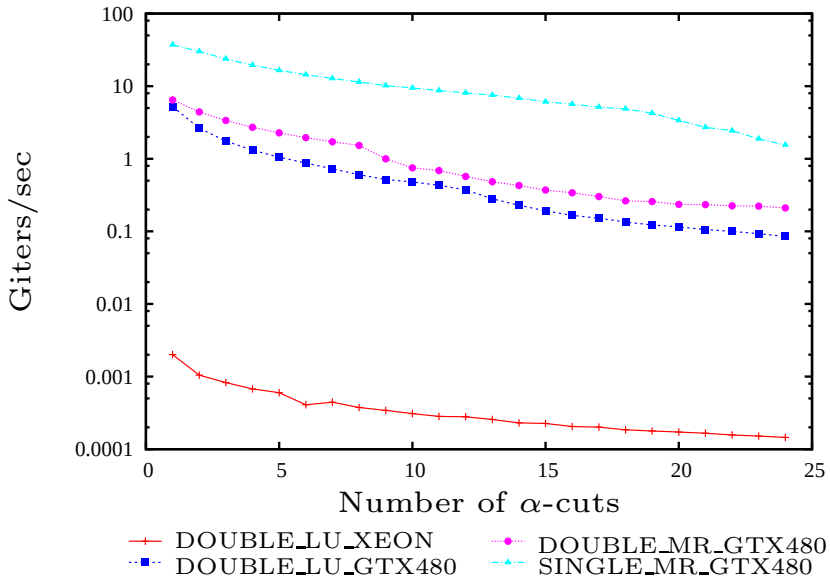
Absolute error in the midpoint-increment format ($i \geqslant 2$)

$$\rho_{z,i} = \rho_{z,i-1} + \delta_{z,i}.$$

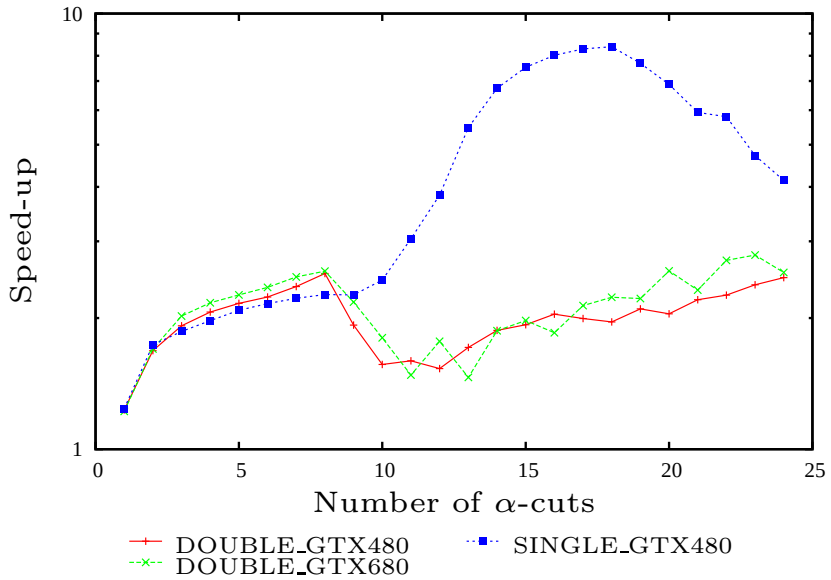
$$\begin{aligned} E(\rho_{z,i}) &= E(\rho_{z,i-1}) + E(\delta_{z,i}) \\ &\leqslant \epsilon \left(\epsilon|\check{z}| + 2\rho_{x,i-1} + \rho_{y,i-1} \right) + \epsilon \left| \delta_{x,i} + \delta_{y,i} \right| \\ &= \epsilon \left(\epsilon|\check{z}| + \rho_{x,i-1} + \rho_{x,i} + \rho_{y,i} \right). \end{aligned}$$

Outline

Compute-bound benchmark: AXPY loop



(a)



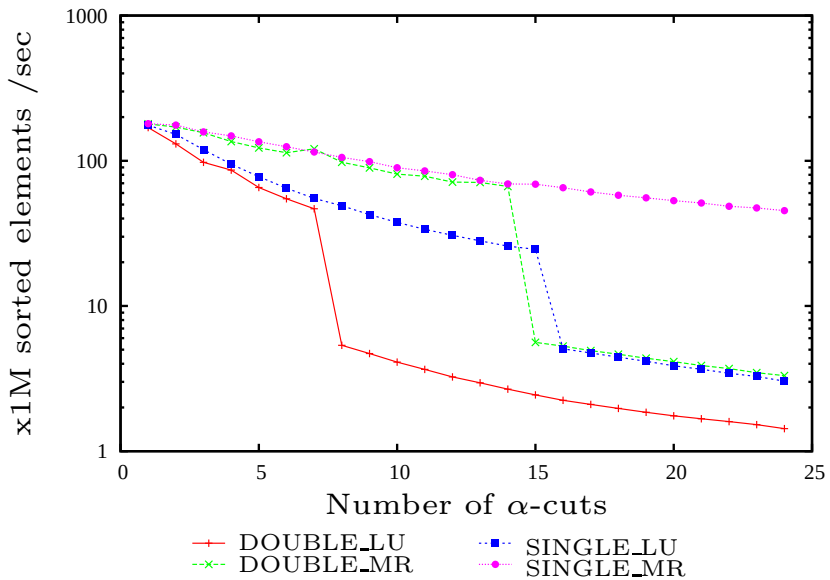
(b)

Figure 1: (a) Performance comparison of different representation formats and architectures. (b) Speed-up of midpoint-radius over lower-upper.

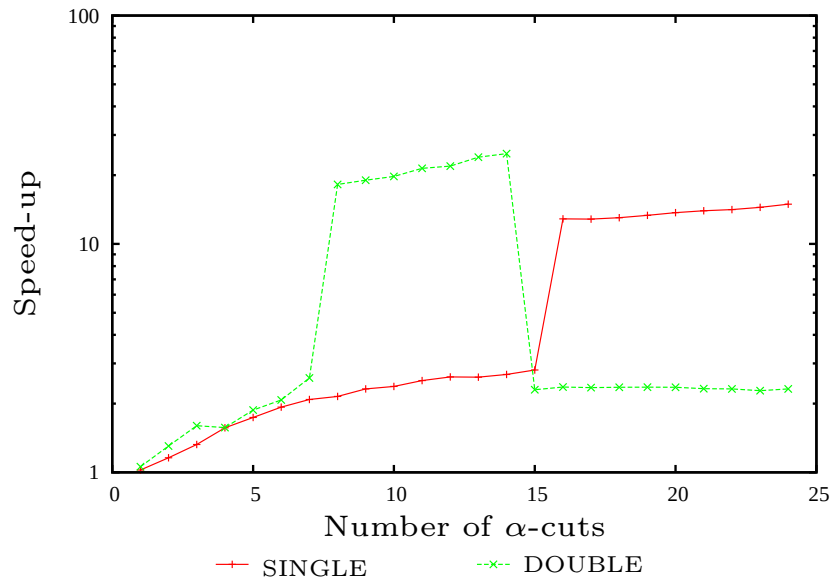
For fewer than 8 α-cuts the speed-up curve follows the theoretical ratio:

$$\frac{16N}{9+7N}$$

Memory-bound benchmark: RADIX sort



(a)



(b)

Figure 2: (a) Performance comparison of different representation formats and architectures. (b) Speed-up of midpoint-radius over lower-upper.

For fewer than 7 α-cuts the speed-up curve follows the theoretical ratio:

$$\frac{2N}{1+N}$$

Conclusions and future work

- Midpoint-radius representation is an attractive alternative to handle symmetric fuzzy intervals. It achieves a speed-up of 2 to 20 over lower-upper in both compute and memory-bound benchmarks.
- Fuzzy arithmetic library written in CUDA, available at `https://github.com/mmarin-upvd/fuzzy-gpu/`.
- Future work will consider the implementation of the midpoint-increment format and further assessment of accuracy issues.