Journée des doctorants

Silviu Filip Adeline Langlois Vincent Neiger
Philippe Theveny

Aric Team, LIP, ENS de Lyon, France

June 3, 2014

AriC

Exact

Exact arithmetic

Computer algebra

Approximate

Floating-point arithmetic

Interval arithmetic

Structured linear algebra

Lattices

Polynomial

Euclidean

Numerical Linear Algebra

Multivariate Interpolation

Learning With Errors structured version

Numerical functions and digital filters

Interval Linear Algebra

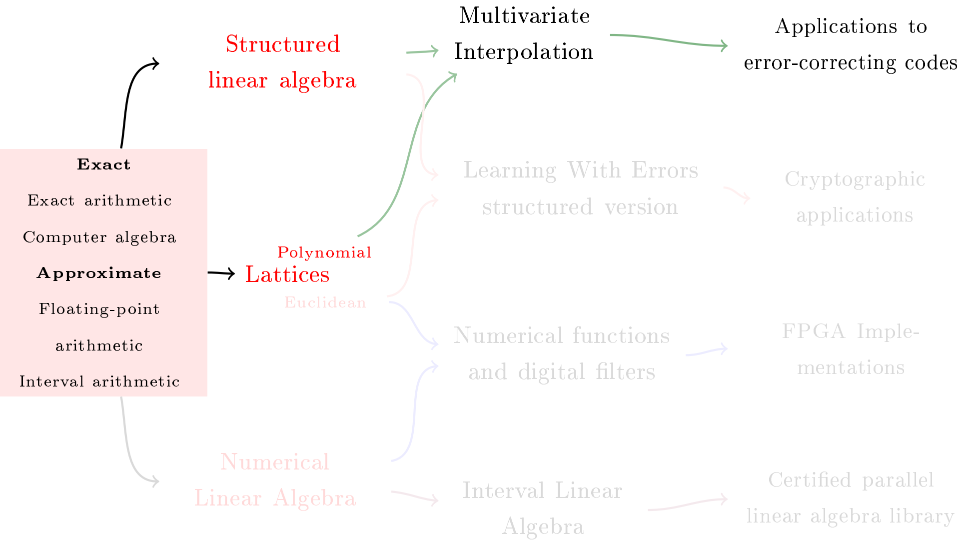
Applications to error-correcting codes

Cryptographic applications

FPGA Implementations

Certified parallel linear algebra library

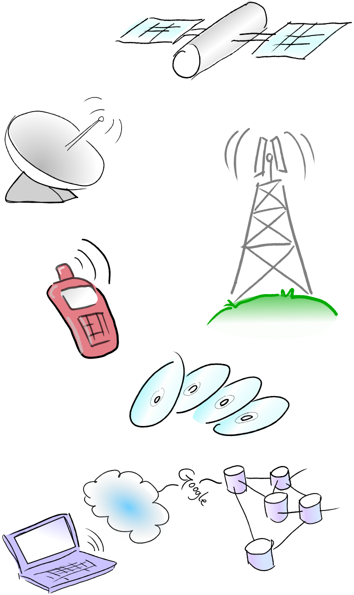
AriC


Structured
linear algebra
Multivariate
Interpolation
Applications to
error-correcting codes
Exact
Exact arithmetic
Computer algebra
Approximate
Floating-point
arithmetic
Interval arithmetic
Polynomial
Lattices
Euclidean
Learning With Errors
structured version
Cryptographic
applications
Numerical functions
and digital filters
FPGA Imple-
mentations
Numerical
Linear Algebra
Interval Linear
Algebra
Certified parallel
linear algebra library

AriC — Error-correcting codes

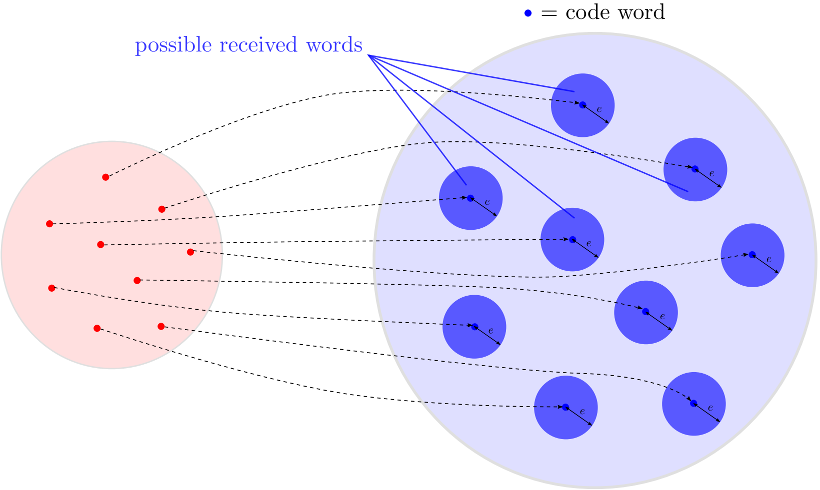
Goal:
Enable reliable delivery of data over unreliable communication channels

Strategy:
add redundancy to the message
add redundancy to the message
add redundancy to the message



(courtesy of J.S.R. Nielsen)

AriC — Error-correcting codes

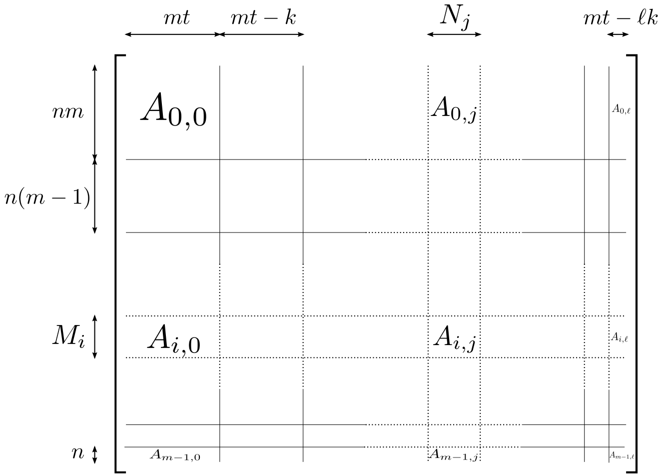


polynomials of degree $\leqslant k$ $\longrightarrow$ their evaluation at $x_1, \ldots, x_n$

$w = w_0 + w_1 X + \cdots + w_k X^k$ $\qquad$ $(w(x_1), \ldots, w(x_n))$

AriC — (list-)Decoding

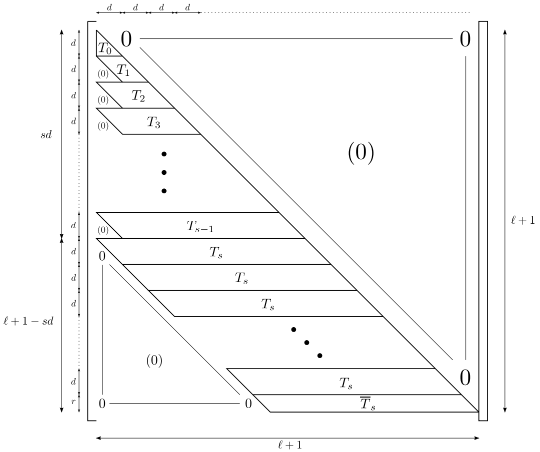
Find a solution of a structured linear system,



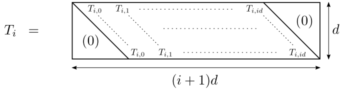
where $A_{i,j}$ is a Toeplitz / Hankel / Vandermonde / … matrix

AriC — (list-)Decoding

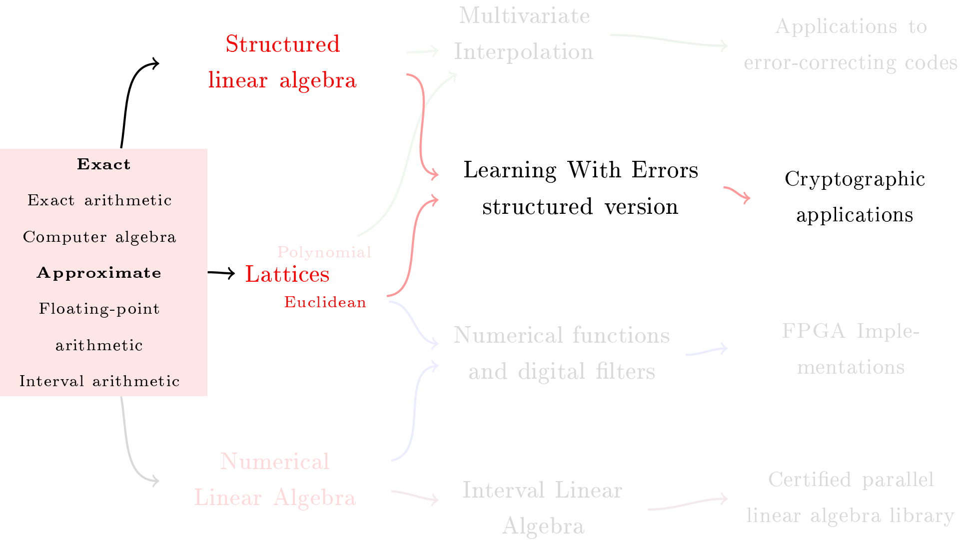
Find a short vector in a (structured) polynomial lattice,



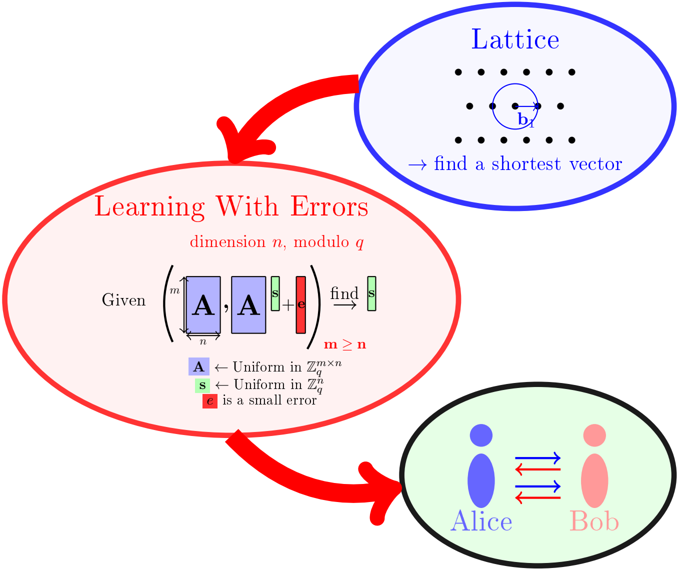
where T_i has a **Toeplitz** structure:



AriC


Structured
linear algebra
Multivariate
Interpolation
Applications to
error-correcting codes
Exact
Exact arithmetic
Computer algebra
Approximate
Floating-point
arithmetic
Interval arithmetic
Polynomial
Lattices
Euclidean
Learning With Errors
structured version
Cryptographic
applications
Numerical functions
and digital filters
FPGA Imple-
mentations
Numerical
Linear Algebra
Interval Linear
Algebra
Certified parallel
linear algebra library

AriC – Lattice-Based Cryptography



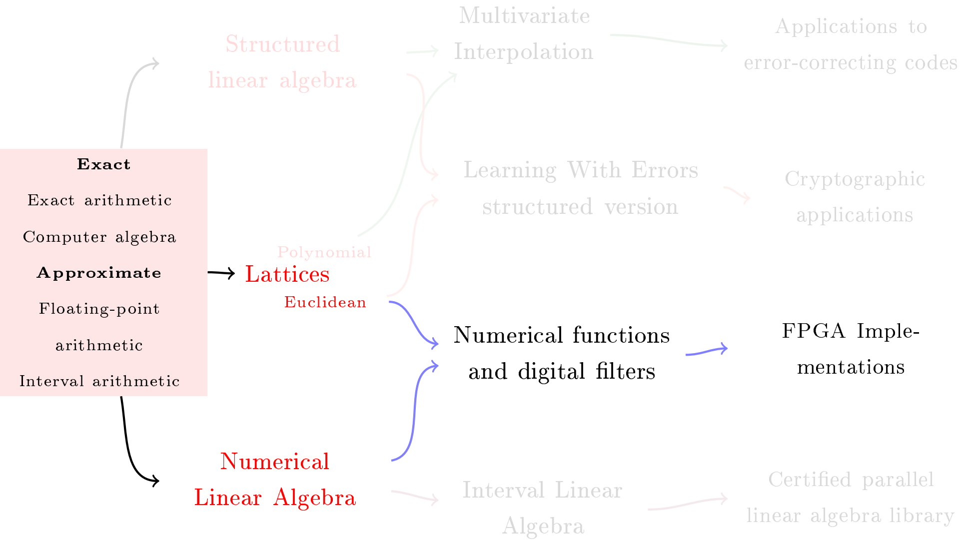
AriC – Lattice-Based Cryptography

- Public Key Encryption
- Identity Based Encryption
- Fully Homomorphic Encryption

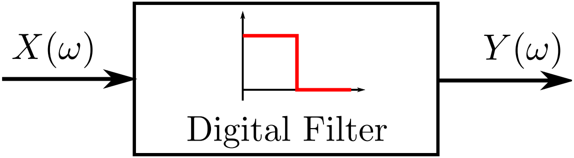
- Signature
- Group Signature
- Hash Function

- Cryptographic Multilinear Maps

AriC


Structured linear algebra
Multivariate Interpolation
Applications to error-correcting codes
Exact
Exact arithmetic
Computer algebra
Approximate
Floating-point arithmetic
Interval arithmetic
Polynomial
Lattices
Euclidean
Learning With Errors structured version
Cryptographic applications
Numerical functions and digital filters
FPGA Implementations
Numerical Linear Algebra
Interval Linear Algebra
Certified parallel linear algebra library

AriC – Digital Filter Design



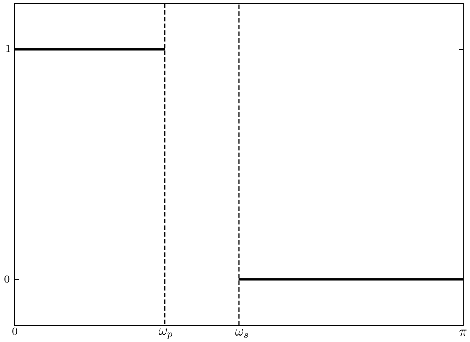
$$Y(\omega) = H_d(\omega)X(\omega), \omega \in [0, \pi]$$

Two types of filters:

- finite impulse response (**FIR**) $\Rightarrow H_d(\omega)$ **polynomial**
- infinite impulse response (**IIR**) $\Rightarrow H_d(\omega)$ **rational function**

AriC – Digital Filter Design

FIR case: $H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$



Steps:

1. Optimal filter computation:

$$H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$$

Naive rounding:

$$\overline{H}_d(\omega) = \sum_{k=0}^{L} \overline{a}_k \cos(\omega k)$$

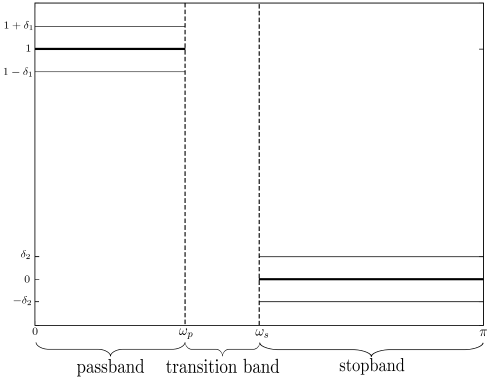
2. Coefficient quantization:

$$H_d^*(\omega) = \sum_{k=0}^{L} a_k^* \cos(\omega k)$$

Goal: filter synthesis toolchain for embedded and FPGA targets

AriC – Digital Filter Design

FIR case: $H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$



Steps:

1. Optimal filter computation:

$$H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$$

Naive rounding:

$$\overline{H}_d(\omega) = \sum_{k=0}^{L} \overline{a}_k \cos(\omega k)$$

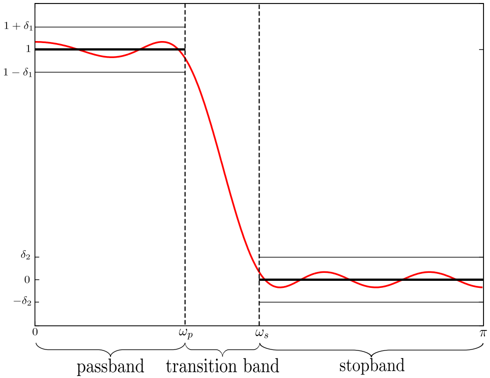
2. Coefficient quantization:

$$H_d^*(\omega) = \sum_{k=0}^{L} a_k^* \cos(\omega k)$$

Goal: filter synthesis toolchain for embedded and FPGA targets

AriC – Digital Filter Design

FIR case: $H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$



Steps:

1. Optimal filter computation:

$$H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$$

Naive rounding:

$$\overline{H}_d(\omega) = \sum_{k=0}^{L} \overline{a}_k \cos(\omega k)$$

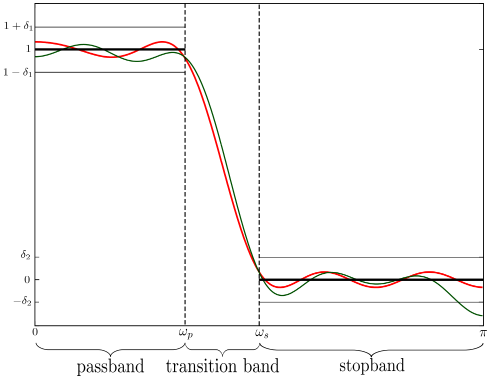
2. Coefficient quantization:

$$H_d^*(\omega) = \sum_{k=0}^{L} a_k^* \cos(\omega k)$$

Goal: filter synthesis toolchain for embedded and FPGA targets

AriC – Digital Filter Design

FIR case: $H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$



Steps:

1. Optimal filter computation:

$$H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$$

Naive rounding:

$$\overline{H}_d(\omega) = \sum_{k=0}^{L} \overline{a}_k \cos(\omega k)$$

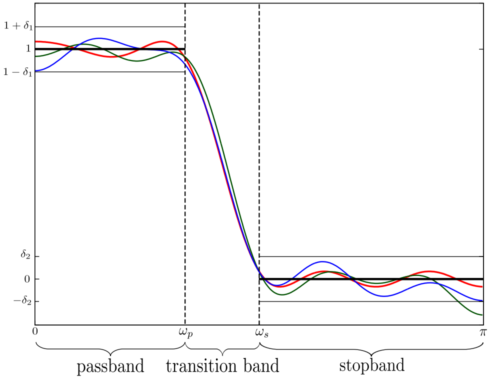
2. Coefficient quantization:

$$H_d^*(\omega) = \sum_{k=0}^{L} a_k^* \cos(\omega k)$$

Goal: filter synthesis toolchain for embedded and FPGA targets

AriC – Digital Filter Design

FIR case: $H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$



Steps:

1. Optimal filter computation:

$$H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$$

Naive rounding:

$$\overline{H}_d(\omega) = \sum_{k=0}^{L} \overline{a}_k \cos(\omega k)$$

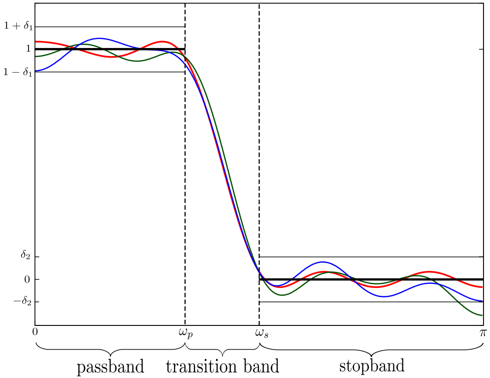
2. Coefficient quantization:

$$H_d^*(\omega) = \sum_{k=0}^{L} a_k^* \cos(\omega k)$$

Goal: filter synthesis toolchain for embedded and FPGA targets

AriC – Digital Filter Design

FIR case: $H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$



Steps:

1. Optimal filter computation:

$$H_d(\omega) = \sum_{k=0}^{L} a_k \cos(\omega k)$$

Naive rounding:

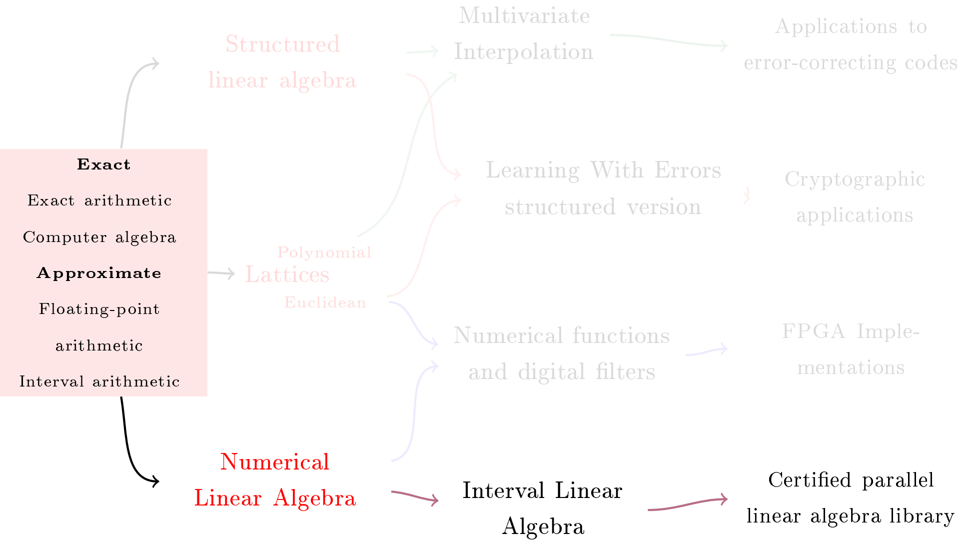
$$\overline{H}_d(\omega) = \sum_{k=0}^{L} \overline{a}_k \cos(\omega k)$$

2. Coefficient quantization:

$$H_d^*(\omega) = \sum_{k=0}^{L} a_k^* \cos(\omega k)$$

Goal: filter synthesis toolchain for embedded and FPGA targets

AriC



Exact
Exact arithmetic
Computer algebra
Approximate
Floating-point arithmetic
Interval arithmetic
Structured linear algebra
Lattices
Polynomial
Euclidean
Numerical Linear Algebra
Multivariate Interpolation
Learning With Errors structured version
Numerical functions and digital filters
Interval Linear Algebra
Applications to error-correcting codes
Cryptographic applications
FPGA Implementations
Certified parallel linear algebra library

Numerical Linear Algebra

$$\begin{pmatrix} 1.20 & 0.40 \\ 8.00 & 2.50 \end{pmatrix} \begin{pmatrix} 44.3 & 2.10 \cdot 10^{+3} \\ 12.6 & 2.60 \cdot 10^{-3} \end{pmatrix}$$
$$\approx \begin{pmatrix} 58.2 & 2.52 \cdot 10^{+3} \\ 386 & 1.68 \cdot 10^{+4} \end{pmatrix}$$

Numerical Interval Matrix Multiplication

$$\begin{pmatrix} [1,2] & [0,4] \\ [8,8] & [2,3] \end{pmatrix} \begin{pmatrix} [44,45] & [2\cdot 10^{+3}, 3\cdot 10^{+3}] \\ [12,13] & [2\cdot 10^{-3}, 3\cdot 10^{-3}] \end{pmatrix}$$

$$\subseteq \begin{pmatrix} [44,142] & [2\cdot 10^{+3}, 6\cdot 10^{+3}] \\ [376,399] & [1.6\cdot 10^{+4}, 2.4\cdot 10^{+4}] \end{pmatrix}$$

Classical 3-loops Algorithm

Input: $\boldsymbol{A} = [\underline{A}, \overline{A}] \in \mathbb{IF}^{m \times k}$, $\boldsymbol{B} = [\underline{B}, \overline{B}] \in \mathbb{IF}^{k \times n}$

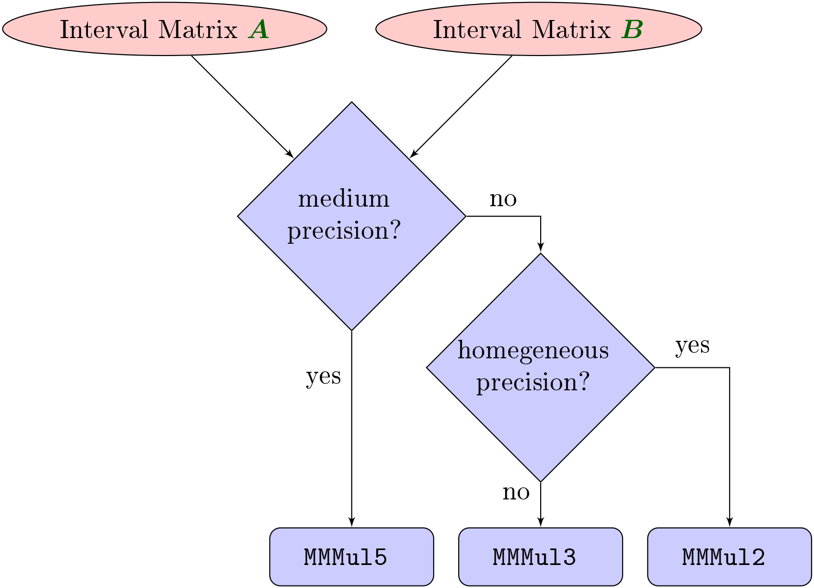
Output: $\boldsymbol{C} \in \mathbb{IF}^{m \times n}$, $\boldsymbol{C} \supseteq \boldsymbol{AB}$

for $i = 1$ to m **do**
    for $j = 1$ to n **do**
        $\underline{C}_{ij} \leftarrow 0$; $\overline{C}_{ij} \leftarrow 0$
        for $l = 1$ to k **do**
            $\underline{C}_{ij} \leftarrow$
            $\texttt{rounddown}\left(\underline{C}_{ij} + \min\left\{\underline{A}_{il}\underline{B}_{lj}, \underline{A}_{il}\overline{B}_{lj}, \overline{A}_{il}\underline{B}_{lj}, \overline{A}_{il}\overline{B}_{lj}\right\}\right)$
            $\overline{C}_{ij} \leftarrow$
            $\texttt{roundup}\left(\overline{C}_{ij} + \max\left\{\underline{A}_{il}\underline{B}_{lj}, \underline{A}_{il}\overline{B}_{lj}, \overline{A}_{il}\underline{B}_{lj}, \overline{A}_{il}\overline{B}_{lj}\right\}\right)$
        end for
    end for
end for
return $[\underline{C}, \overline{C}]$

Variant Interval Algorithms

Algorithm	Computed Radius	Cost
`MMMul3`	at most $1.5\times$ exact radius	about 3 `gemm`'s
`MMMul5`	at most $1.18\times$ exact radius	about 5 `gemm`'s

Criterion of Choice

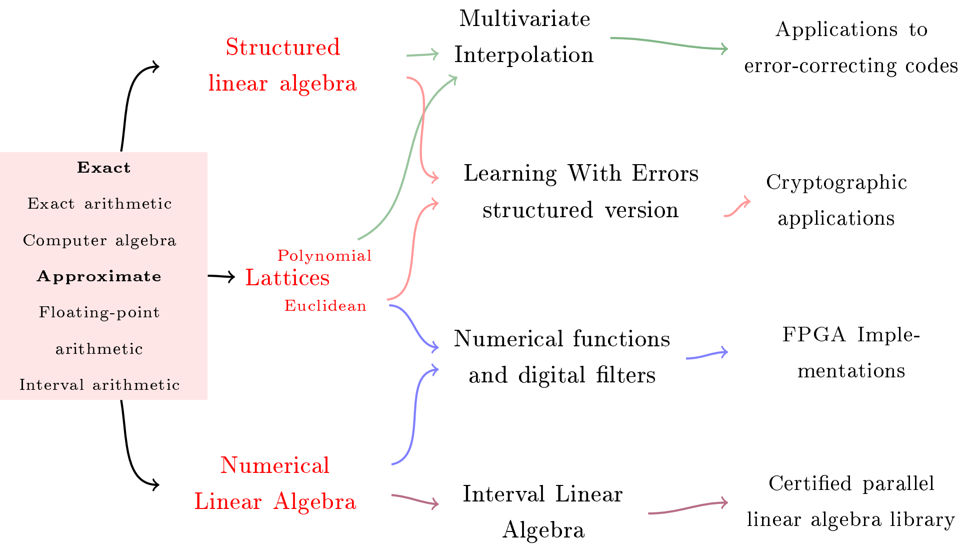


Parallel Implementation

Problematic Rounding Mode Support:

- compiler
- numerical library
- multi-thread management

AriC


Exact
Exact arithmetic
Computer algebra
Approximate
Floating-point arithmetic
Interval arithmetic
Structured linear algebra
Lattices
Polynomial
Euclidean
Numerical Linear Algebra
Multivariate Interpolation
Learning With Errors structured version
Numerical functions and digital filters
Interval Linear Algebra
Applications to error-correcting codes
Cryptographic applications
FPGA Implementations
Certified parallel linear algebra library