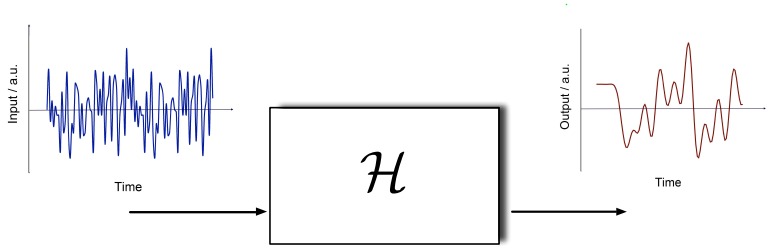
Towards reliable implementation of digital filters

Anastasia Volkova

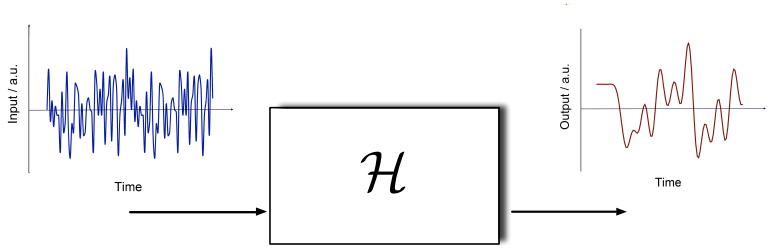
Sorbonne Universités, UPMC, LIP6

Seminar at ENS Lyon, LIP, AriC team
January 5, 2017

Context: digital filters



Context: digital filters

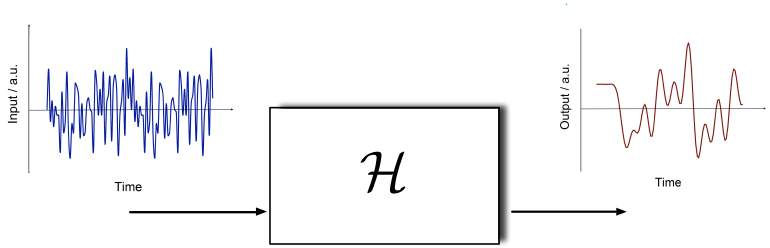


On the one hand

- LTI filter with Infinite Impulse Response
- Its transfer function:

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}$$

Context: digital filters



On the one hand

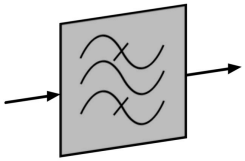
- LTI filter with Infinite Impulse Response
- Its transfer function:

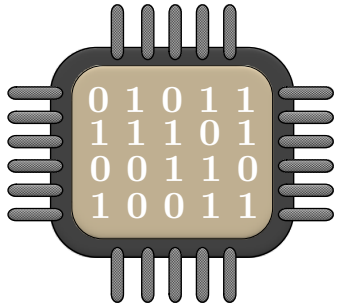
$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}$$

On the other hand

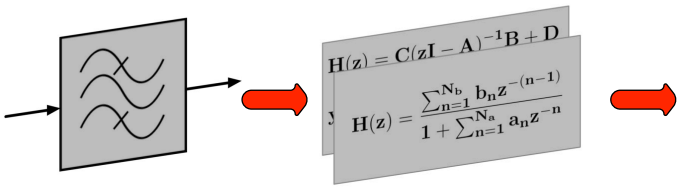
- Hardware or Software target
- Implementation in Fixed- or Floating-Point Arithmetic

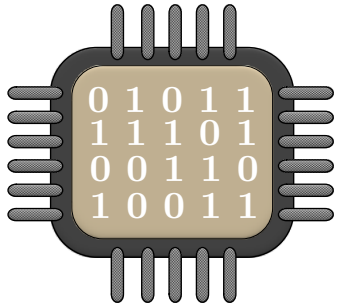
Context: implementation of LTI filters





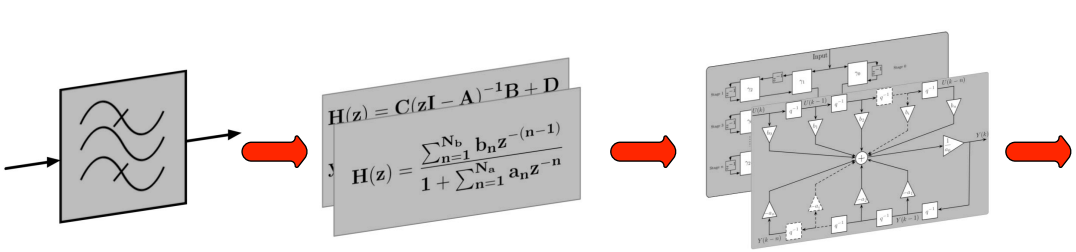
Context: implementation of LTI filters

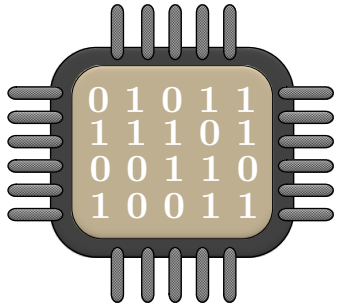




- Transfer function generation

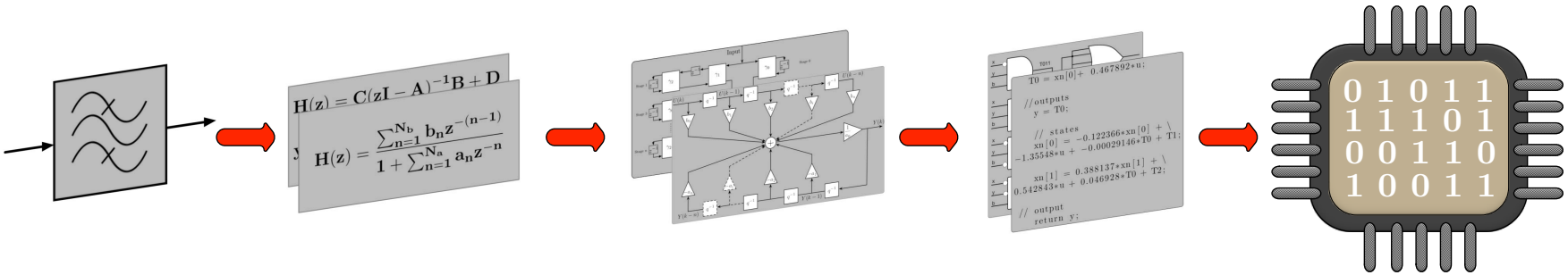
Context: implementation of LTI filters





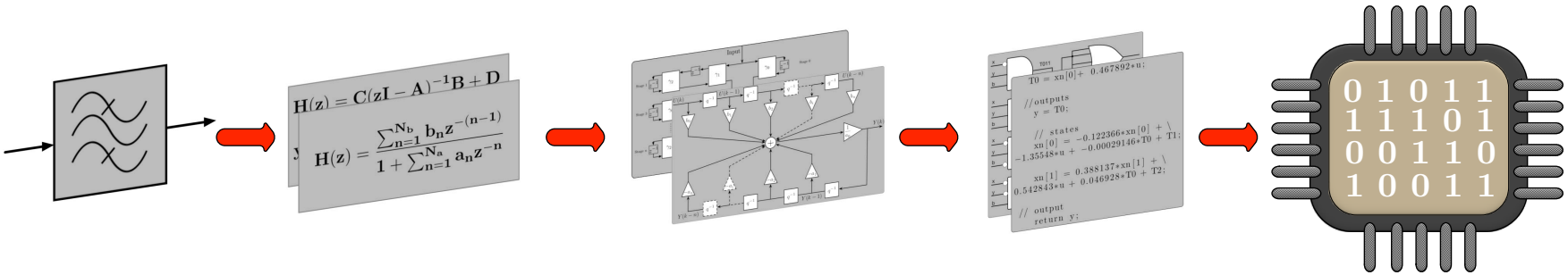
- Transfer function generation
- Algorithm choice: State-space, Direct Form I, Direct Form II, ...

Context: implementation of LTI filters



- Transfer function generation
- Algorithm choice: State-space, Direct Form I, Direct Form II, ...
- Software or Hardware implementation

Context: implementation of LTI filters



- Transfer function generation
  - ☞ Coefficient quantization
- Algorithm choice: State-space, Direct Form I, Direct Form II, ...
  - ☞ Large variety of structures with no common quality criteria
- Software or Hardware implementation
  - ☞ Constraints: power consumption, area, error, speed, etc.
  - ☞ Computational errors due to finite-precision implementation

Filter-to-code generator

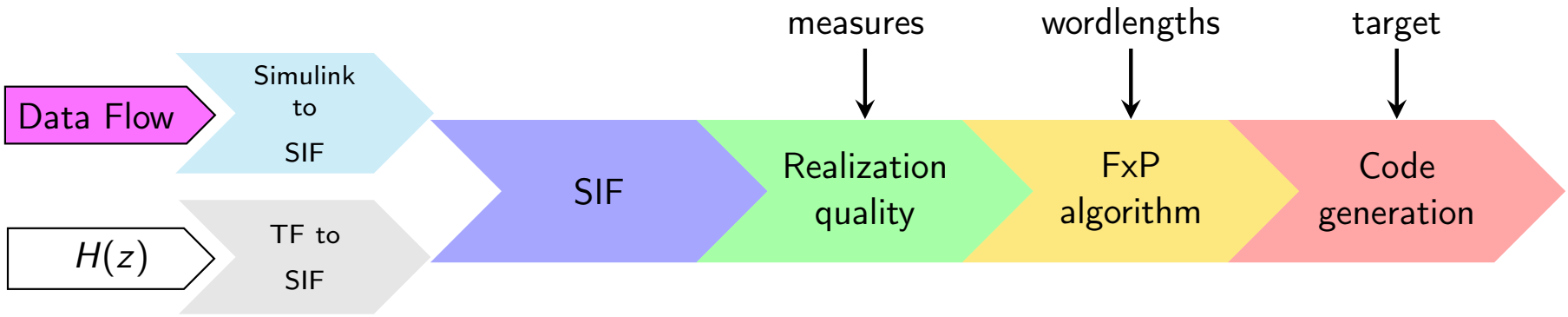


Figure: Automatic Filter Generator Flow

- Use unified *analytical* representation of linear data-flows
- Can describe any analytical and graphical representation
- Adopted numerous classical and developed new quality measures
- Provide **fully rigorous** and **reliable** implementation in Fixed-Point arithmetic
- Generate C (for μCs and DSPs) and VHDL code (for ASICs and FPGAs)

Filter-to-code generator

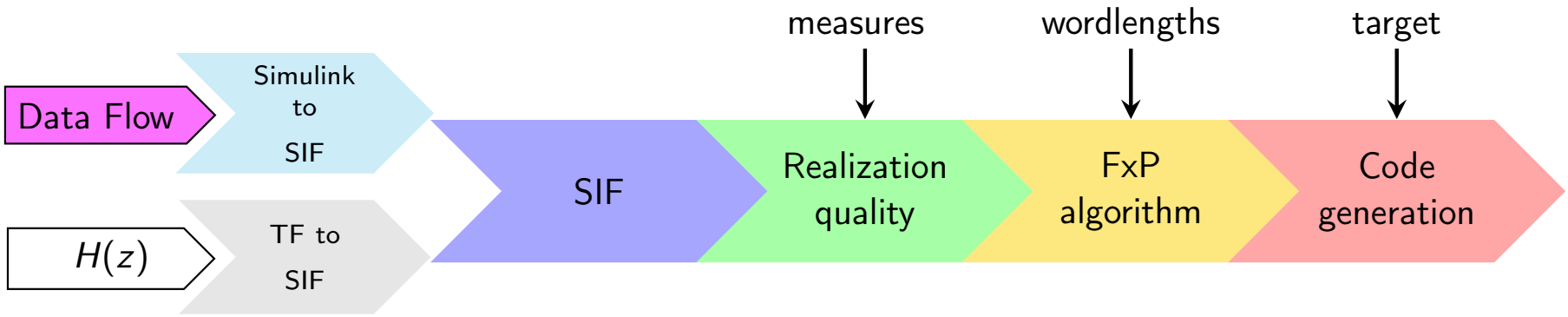


Figure: Automatic Filter Generator Flow

- Use unified *analytical* representation of linear data-flows
- Can describe any analytical and graphical representation
- Adopted numerous classical and developed new quality measures
- Provide **fully rigorous** and **reliable** implementation in Fixed-Point arithmetic
- Generate C (for μCs and DSPs) and VHDL code (for ASICs and FPGAs)

Filter-to-code generator

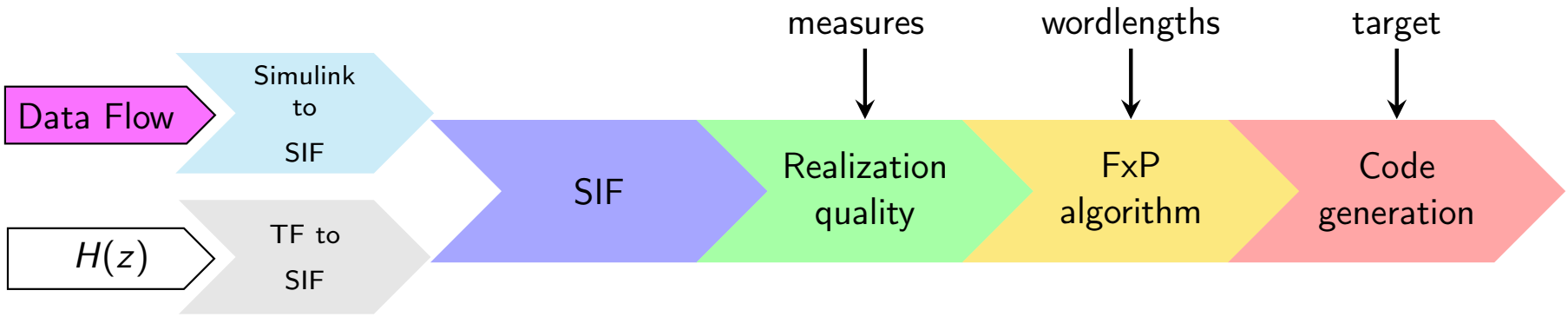


Figure: Automatic Filter Generator Flow

- Use unified *analytical* representation of linear data-flows
- Can describe any analytical and graphical representation
- Adopted numerous classical and developed new quality measures
- Provide **fully rigorous** and **reliable** implementation in Fixed-Point arithmetic
- Generate C (for μCs and DSPs) and VHDL code (for ASICs and FPGAs)

Reliable implementation of digital filters in Fixed-Point Arithmetic

LTI filters

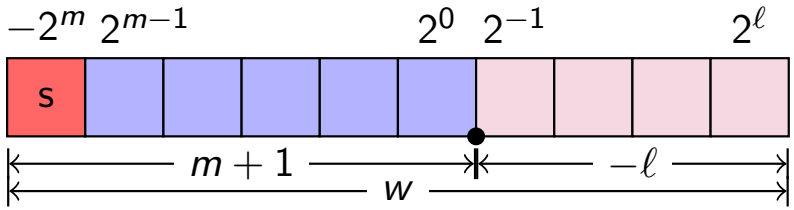
Let $\mathcal{H} := (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$ be a LTI filter:

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

The filter $\mathcal{H}$ is considered Bounded Input Bounded Output stable if

$$\rho(\boldsymbol{A}) < 1$$

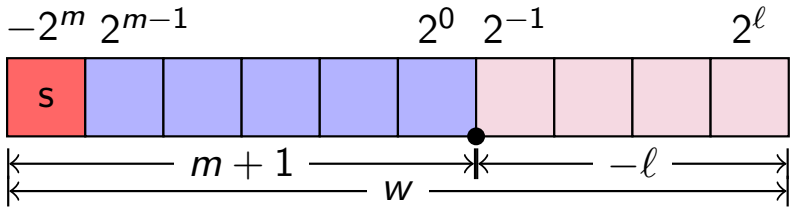
Two's complement Fixed-Point arithmetic



$$y = -2^m y_m + \sum_{i=\ell}^{m-1} 2^i y_i$$

- Wordlength: w
- **M**ost **S**ignificant **B**it position: m
- **L**east **S**ignificant **B**it position: $\ell := m - w + 1$

Two's complement Fixed-Point arithmetic



$$y = -2^m y_m + \sum_{i=\ell}^{m-1} 2^i y_i$$

- $y(k) \in \mathbb{R}$
- wordlength w bits
- minimal Fixed-Point Format (FPF) is the least m:

$$\forall k, \quad y(k) \in [-2^m; 2^m - 2^{m-w+1}]$$

Reliable Fixed-Point implementation

Input:

- $\mathcal{H} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$
- bound on the input interval
- wordlength constraints

Determine: the Fixed-Point Formats s.t.

- the least MSBs
- no overflows occur

  $\rightsquigarrow$ must take into account computational errors

Reliable Fixed-Point implementation

Input:

- $\mathcal{H} = (\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D})$
- bound on the input interval
- wordlength constraints

Determine: the Fixed-Point Formats s.t.

- the least MSBs
- no overflows occur

  $\rightsquigarrow$ must take into account computational errors

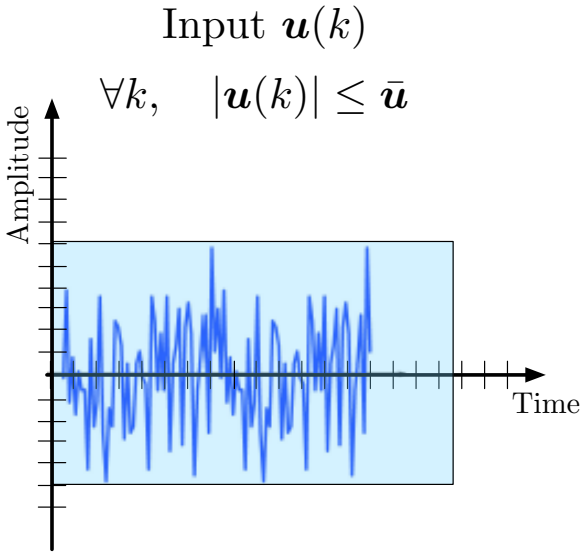
How to proceed:

1. determine the output interval of all variables
2. analyze propagation of the error in filter implementation and determine the FxPF

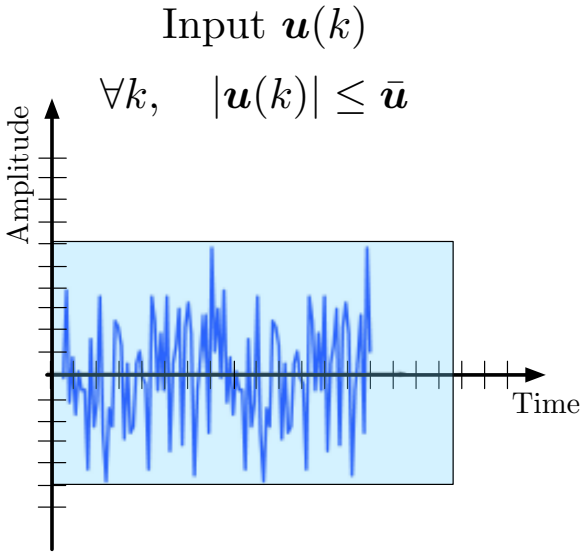
Deducing the output interval[1]

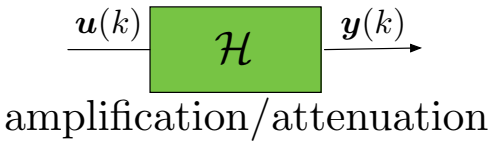
[1] A.V. et al., "Reliable Evaluation of the Worst-Case Peak Gain Matrix in Multiple Precision", ARITH22, 2015

Basic brick: the Worst-Case Peak Gain theorem



Basic brick: the Worst-Case Peak Gain theorem



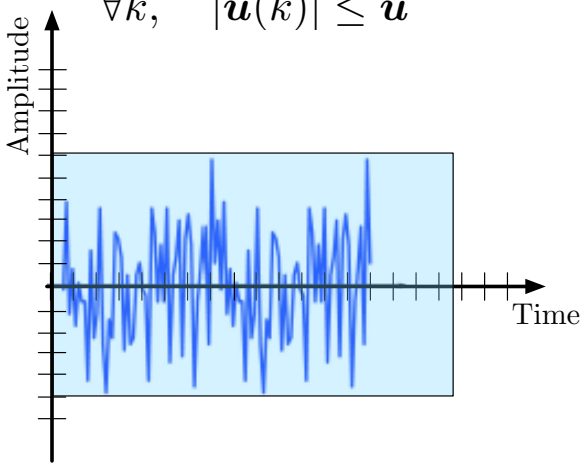


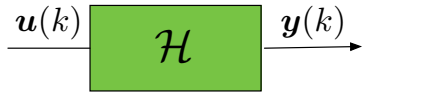
amplification/attenuation

Basic brick: the Worst-Case Peak Gain theorem

Input $\boldsymbol{u}(k)$

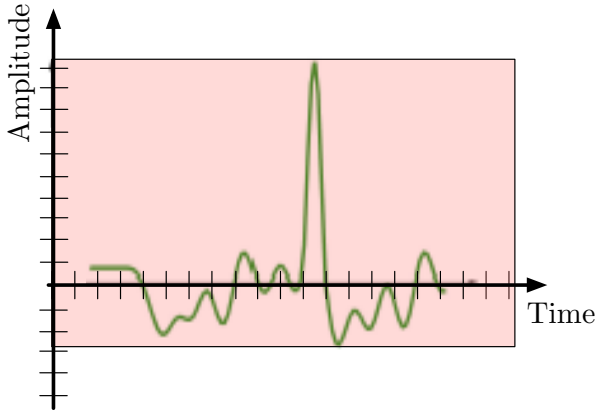
$\forall k, \quad |\boldsymbol{u}(k)| \leq \bar{\boldsymbol{u}}$





amplification/attenuation

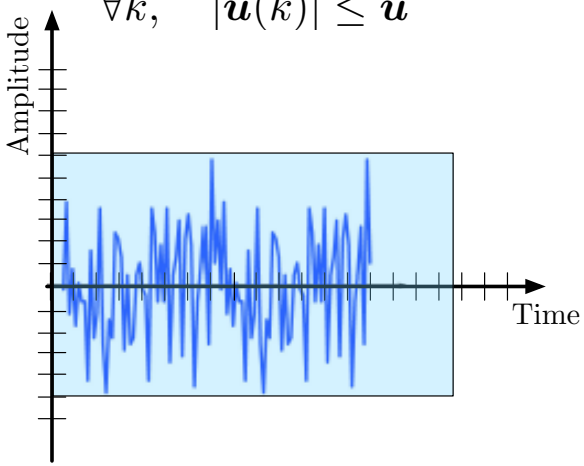
Output $\boldsymbol{y}(k)$

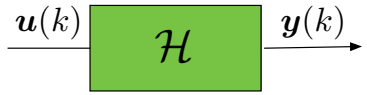


Basic brick: the Worst-Case Peak Gain theorem

Input $\boldsymbol{u}(k)$

$\forall k, \quad |\boldsymbol{u}(k)| \leq \bar{\boldsymbol{u}}$

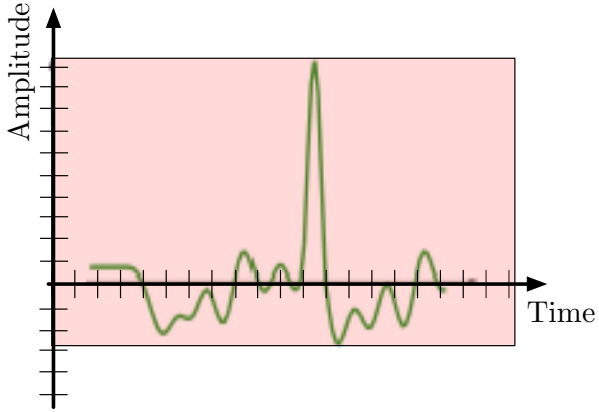




amplification/attenuation

Output $\boldsymbol{y}(k)$

$\forall k, \quad |\boldsymbol{y}(k)| \leq \langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}$



Worst-Case Peak Gain

$$\langle\langle \mathcal{H} \rangle\rangle = |\mathbf{D}| + \sum_{k=0}^{\infty} |\mathbf{C}\mathbf{A}^k\mathbf{B}|$$

Computing the Worst-Case Peak Gain

Problem: compute the Worst-Case Peak Gain with arbitrary precision.

$$\langle\langle \mathcal{H} \rangle\rangle = |\boldsymbol{D}| + \sum_{k=0}^{\infty} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

- Cannot sum infinitely $\Rightarrow$ need to truncate the sum
- Once the sum is truncated, evaluate it in multiple precision

Truncation

$$\sum_{k=0}^{\infty} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right| \quad \longrightarrow \quad \sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

Truncation

$$\left| \sum_{k=0}^{\infty} |\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}| \quad - \quad \sum_{k=0}^{N} |\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}| \right| \leq \varepsilon_1$$

Compute an approximate lower bound on truncation order N such that the truncation error is smaller than ε_1.

Lower bound on truncation order N

$$N \geq \left\lceil \frac{\log \frac{\varepsilon_1}{\|\boldsymbol{M}\|_{min}}}{\log \rho(\boldsymbol{A})} \right\rceil, \quad \text{with } \boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1-|\lambda_l|} \frac{|\lambda_l|}{\rho(\boldsymbol{A})}$$

where

λ – eigenvalues of matrix $\boldsymbol{A}$

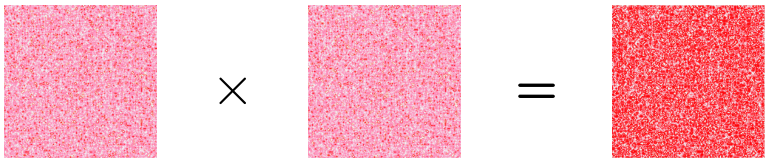
$\boldsymbol{R}_l$ – l^{th} residue matrix computed out of $\boldsymbol{C}, \boldsymbol{B}, \lambda$

Powering

$$\sum_{k=0}^{N} \left| C A^k B \right|$$

Powering

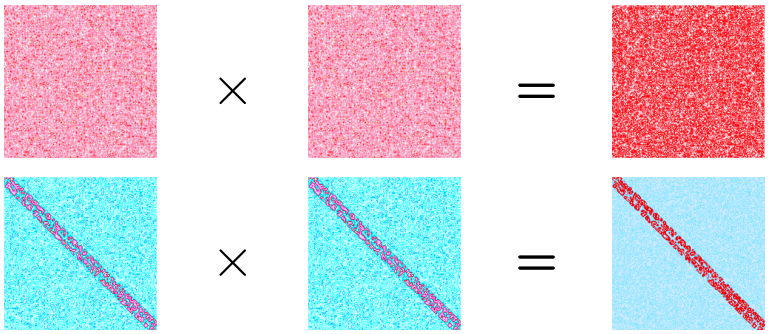
$$\sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^k \boldsymbol{B} \right|$$



cancellation

Powering

$$\sum_{k=0}^{N} |CA^k B|$$

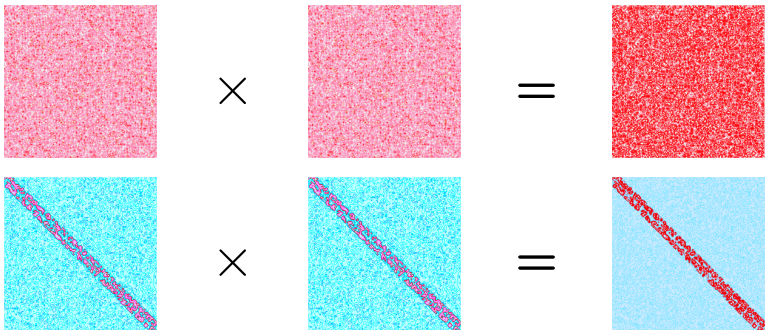


cancellation

less cancellation

Powering

$$\sum_{k=0}^{N} |CA^k B|$$



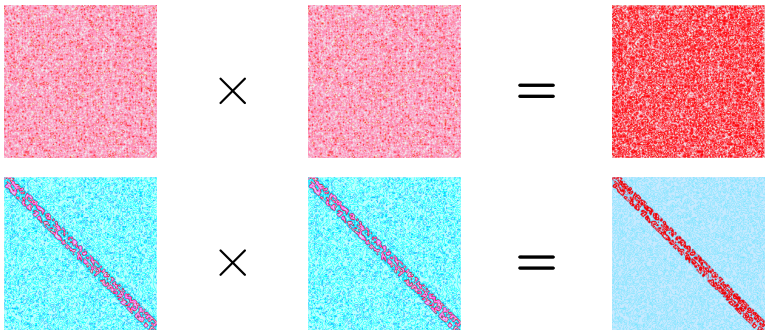
cancellation

less cancellation

$$A = XEX^{-1}$$

Powering

$$\sum_{k=0}^{N} |\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B}|$$



cancellation

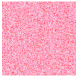
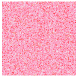
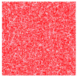
less cancellation

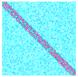
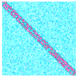
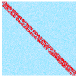
$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$$

$$\boldsymbol{V} \approx \boldsymbol{X} \text{ and } \boldsymbol{T} \approx \boldsymbol{E}$$

Powering

$$\sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

 $\times$  $=$  cancellation

 $\times$  $=$  less cancellation

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1} \qquad \boldsymbol{V} \approx \boldsymbol{X} \text{ and } \boldsymbol{T} \approx \boldsymbol{E}$$

$$\boldsymbol{T} \approx \boldsymbol{V}^{-1} \times \boldsymbol{A} \times \boldsymbol{V}$$

$$\downarrow$$

$$\boldsymbol{A}^k \approx \boldsymbol{V} \times \boldsymbol{T}^k \times \boldsymbol{V}^{-1}$$

Powering

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right| \quad - \quad \sum_{k=0}^{N} \left| \boldsymbol{C}\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}\boldsymbol{B} \right| \right| \leq \varepsilon_2$$

Given matrix $\boldsymbol{V}$ compute $\boldsymbol{T}$ such that the error of substitution of the product $\boldsymbol{V}\boldsymbol{T}^k\boldsymbol{V}^{-1}$ instead of $\boldsymbol{A}^k$ is less than ε_2.

Further steps

$$\left| \sum_{k=0}^{N} \left| C A^k B \right| \quad - \quad \sum_{k=0}^{N} \left| C V T^k V^{-1} B \right| \right| \leq \varepsilon_2$$

Apply the same approach for the other steps:

$$\left| \sum_{k=0}^{N} \left| C V T^k V^{-1} B \right| - \sum_{k=0}^{N} \left| C' T^k B' \right| \right| \leq \varepsilon_3$$

$$\left| \sum_{k=0}^{N} \left| C' T^k B' \right| - \sum_{k=0}^{N} \left| C' P_k B' \right| \right| \leq \varepsilon_4$$

$$\left| \sum_{k=0}^{N} \left| C' P_k B' \right| - \sum_{k=0}^{N} \left| L_k \right| \right| \leq \varepsilon_5$$

$$\left| \sum_{k=0}^{N} \left| L_k \right| - S_N \right| \leq \varepsilon_6$$

Basic bricks

Requirement:

Provide matrix operations which satisfy an element-by-element absolute error bound δ given in the argument.

Basic bricks

Requirement:

Provide matrix operations which satisfy an element-by-element absolute error bound δ given in the argument.

Problem:

In fixed-precision FP arithmetic such absolute bound is not generally possible.

Basic bricks

Requirement:

Provide matrix operations which satisfy an element-by-element absolute error bound δ given in the argument.

Problem:

In fixed-precision FP arithmetic such absolute bound is not generally possible.

Solution:

Use multiple-precision FP arithmetic and dynamically adapt precision of the result variables.

What if coefficients are not exact?

Cases when $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are not exact:

- coefficients are results of finite-precision computations (e.g. quantization, SIF $\leftrightarrow$ State-Space transformation etc.)

What if coefficients are not exact?

Cases when $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are not exact:

- coefficients are results of finite-precision computations
  (e.g. quantization, SIF $\leftrightarrow$ State-Space transformation etc.)

To take these properties into account we use Interval Arithmetic.
$\longrightarrow$ Need to compute the WCPG in interval arithmetic.

What if coefficients are not exact?

Cases when $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$ and $\boldsymbol{D}$ are not exact:

- coefficients are results of finite-precision computations (e.g. quantization, SIF $\leftrightarrow$ State-Space transformation etc.)

To take these properties into account we use Interval Arithmetic.
$\longrightarrow$ Need to compute the WCPG in interval arithmetic.

Notation: interval matrix $\boldsymbol{M}^{\mathcal{I}}$ is centered at $mid(\boldsymbol{M}^{\mathcal{I}})$ and has radius $rad(\boldsymbol{M}^{\mathcal{I}})$.
We suppose all interval arithmetic to be in multiple-precision.

Interval WCPG computation

Problem: compute the interval Worst-Case Peak Gain matrix

$$\left\langle\left\langle \mathcal{H}^{\mathcal{I}} \right\rangle\right\rangle = \left|\boldsymbol{D}^{\mathcal{I}}\right| + \sum_{k=0}^{\infty} \left| \boldsymbol{C}^{\mathcal{I}} \boldsymbol{A}^{\mathcal{I}k} \boldsymbol{B}^{\mathcal{I}} \right|.$$

Ensure:

- enclosure property: $\forall \mathcal{H} \in \mathcal{H}^{\mathcal{I}} \Longrightarrow \langle\langle \mathcal{H} \rangle\rangle \in \left\langle\left\langle \mathcal{H}_N^{\mathcal{I}} \right\rangle\right\rangle$
- if coefficients' radii $\to 0$ and the precision $\to \infty$, then $\left\langle\left\langle \mathcal{H}_N^{\mathcal{I}} \right\rangle\right\rangle$ is a ε-neighbourhood of the exact WCPG matrix for arbitrary $\varepsilon > 0$

Computing the eigensystem of interval matrix

Eigenvalues of interval matrix

Compute enclosures $\boldsymbol{\lambda}^{\mathcal{I}}$ such that $\forall \boldsymbol{A} \in \boldsymbol{A}^{\mathcal{I}}, \boldsymbol{\lambda}(\boldsymbol{A}) \in \boldsymbol{\lambda}^{\mathcal{I}}$

Approach

Following the works of Xu and Rachid (1996) and Rohn(1998), use the Generalized Gershgorin's Circles theorem.

Computing the eigensystem of interval matrix

Eigenvalues of interval matrix

Compute enclosures $\boldsymbol{\lambda}^{\mathcal{I}}$ such that $\forall \boldsymbol{A} \in \boldsymbol{A}^{\mathcal{I}}, \boldsymbol{\lambda}(\boldsymbol{A}) \in \boldsymbol{\lambda}^{\mathcal{I}}$

Approach

Following the works of Xu and Rachid (1996) and Rohn(1998), use the Generalized Gershgorin's Circles theorem.

Eigenvectors of interval matrix

Given the enclosures on eigenvalues $\boldsymbol{\lambda}^{\mathcal{I}}$, compute enclosures $\boldsymbol{V}^{\mathcal{I}}$ such that $\forall \boldsymbol{\lambda} \in \boldsymbol{\lambda}^{\mathcal{I}}, \forall \boldsymbol{A} \in \boldsymbol{A}^{\mathcal{I}}$ if $\boldsymbol{A}\boldsymbol{V} = \boldsymbol{\lambda}\boldsymbol{V}$, then $\boldsymbol{V} \in \boldsymbol{V}^{\mathcal{I}}$.

Approach

Use Rump's theory of Verified Inclusions.

Determining the Fixed-Point Formats[2]

[2] A.V. et al., "Determining Fixed-Point Formats for a Digital Filter Implementation using the Worst-Case Peak Gain Measure", Asilomar 49, 2015

Determining the Fixed-Point Formats

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i .$$

Determining the Fixed-Point Formats

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i .$$

We need to find the least integer $\boldsymbol{m}_y$ such that

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}.$$

Determining the Fixed-Point Formats

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i .$$

We need to find the least integer $\boldsymbol{m}_y$ such that

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}.$$

It easy to show that $\boldsymbol{m}_y$ can be computed with

$$\boldsymbol{m}_{y_i} = \left\lceil \log_2 \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i - \log_2 \left(1 - 2^{1 - \boldsymbol{w}_{y_i}}\right) \right\rceil .$$

Determining the Fixed-Point Formats

$$\mathcal{H} \begin{cases} \boldsymbol{x}(k+1) &= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) &= \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

We know that if $\forall k, |\boldsymbol{u}_i(k)| \leq \bar{\boldsymbol{u}}_i$, then

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i .$$

We need to find the least integer $\boldsymbol{m}_y$ such that

$$\forall k, \quad |\boldsymbol{y}_i(k)| \leq 2^{\boldsymbol{m}_{y_i}} - 2^{\boldsymbol{m}_{y_i} - \boldsymbol{w}_{y_i} + 1}.$$

It easy to show that $\boldsymbol{m}_y$ can be computed with

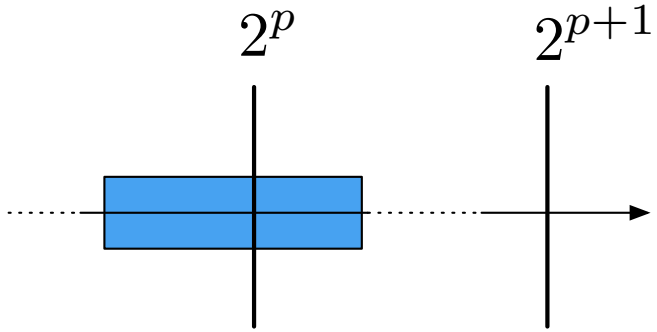
$$\boldsymbol{m}_{y_i} = \left\lceil \log_2 \left(\langle\langle \mathcal{H} \rangle\rangle \, \bar{\boldsymbol{u}}\right)_i - \log_2 \left(1 - 2^{1 - \boldsymbol{w}_{y_i}}\right) \right\rceil .$$

Control the accuracy of the WCPG such that $0 \leq \widehat{\boldsymbol{m}}_{y_i} - \boldsymbol{m}_{y_i} \leq 1$

Open question: the off-by-one problem

$$\widehat{\boldsymbol{m}_{y_i}} = \lceil \mathfrak{m} \rceil$$

Problem: interval $\mathfrak{m}$ contains a power of 2.

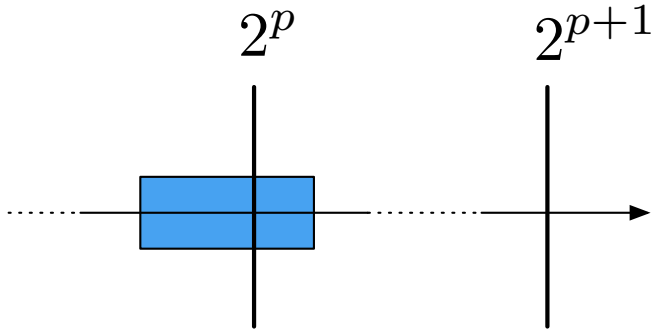


Open question: the off-by-one problem

$$\widehat{\boldsymbol{m}_{y_i}} = \lceil \mathfrak{m} \rceil$$

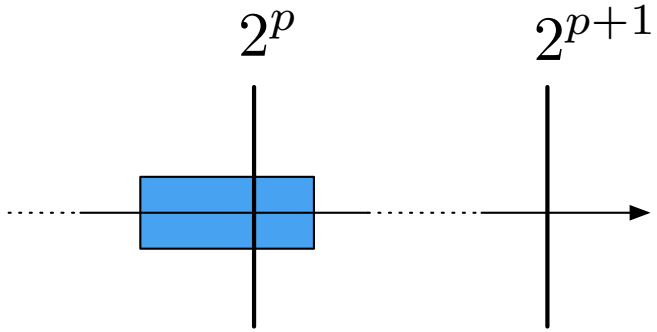
Problem: interval $\mathfrak{m}$ contains a power of 2.

Technique: Ziv's strategy to reduce interval



Open question: the off-by-one problem

$$\widehat{\boldsymbol{m}_{y_i}} = \lceil \mathfrak{m} \rceil$$



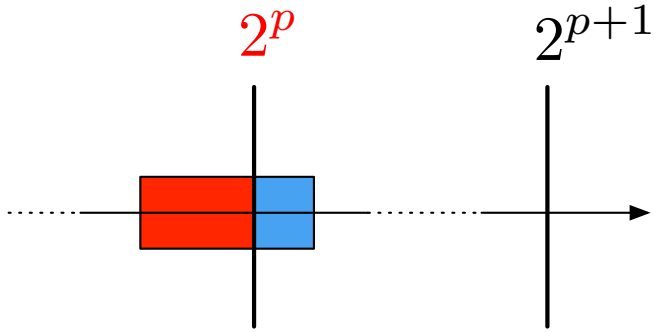
Problem: interval $\mathfrak{m}$ contains a power of 2.
Technique: Ziv's strategy to reduce interval

Dilemma:

- propagation of computational errors
- or overestimation in linear filter decomposition?

Open question: the off-by-one problem

$$\widehat{\boldsymbol{m}_{y_i}} = \lceil \mathbb{m} \rceil$$



Problem: interval $\mathbb{m}$ contains a power of 2.
Technique: Ziv's strategy to reduce interval

Dilemma:

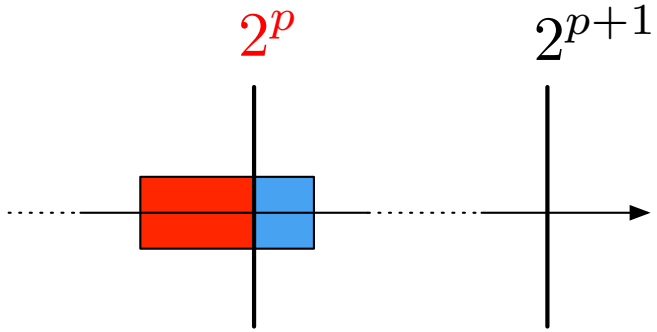
- propagation of computational errors
- or overestimation in linear filter decomposition?

Possible approach:

- Assume the format $\widehat{\boldsymbol{m}} = p$
- Does there exist a *reachable* $\boldsymbol{x}^{\Diamond}(k)$ s.t. $\boldsymbol{y}^{\Diamond}(k)$ overflows ?

Open question: the off-by-one problem

$$\widehat{\boldsymbol{m}_{y_i}} = \lceil \mathbb{m} \rceil$$



Problem: interval $\mathbb{m}$ contains a power of 2.
Technique: Ziv's strategy to reduce interval

Dilemma:

- propagation of computational errors
- or overestimation in linear filter decomposition?

Possible approach:

- Assume the format $\widehat{\boldsymbol{m}} = p$
- Does there exist a *reachable* $\boldsymbol{x}^{\Diamond}(k)$ s.t. $\boldsymbol{y}^{\Diamond}(k)$ overflows ?

Technique: SMT? integer linear programming? lattice algorithms?

Taking the quantization errors into account

The exact filter $\mathcal{H}$ is:

$$\mathcal{H} \begin{cases} \boldsymbol{x}\ (k+1) &= & \boldsymbol{A}\boldsymbol{x}\ (k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}\ (k) &= & \boldsymbol{C}\boldsymbol{x}\ (k) + \boldsymbol{D}\boldsymbol{u}(k) \end{cases}$$

Taking the quantization errors into account

The actually implemented filter $\mathcal{H}^{\Diamond}$ is:

$$\mathcal{H}^{\Diamond} \begin{cases} \boldsymbol{x}^{\Diamond}(k+1) &= \Diamond_{m_x}(\boldsymbol{A}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{B}\boldsymbol{u}(k)) \\ \boldsymbol{y}^{\Diamond}(k) &= \Diamond_{m_y}(\boldsymbol{C}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{D}\boldsymbol{u}(k)) \end{cases}$$

where $\Diamond_m$ is some operator ensuring faithful rounding:

$$|\Diamond_m(x) - x| \leq 2^{m-w+1}.$$

Taking the quantization errors into account

The actually implemented filter $\mathcal{H}^{\Diamond}$ is:

$$\mathcal{H}^{\Diamond} \begin{cases} \boldsymbol{x}^{\Diamond}(k+1) &= \boldsymbol{A}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{B}\boldsymbol{u}(k) &+ \varepsilon_x(k) \\ \boldsymbol{y}^{\Diamond}(k) &= \boldsymbol{C}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{D}\boldsymbol{u}(k) &+ \varepsilon_y(k) \end{cases}$$

with

$$|\varepsilon_x(k)| \leq 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1} \quad \text{and} \quad |\varepsilon_y(k)| \leq 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}.$$

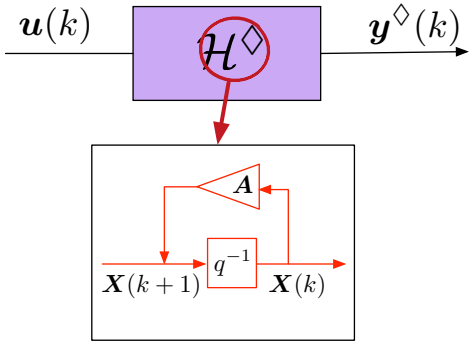
Taking the quantization errors into account

The actually implemented filter $\mathcal{H}^{\Diamond}$ is:

$$\mathcal{H}^{\Diamond} \begin{cases} \boldsymbol{x}^{\Diamond}(k+1) &= \boldsymbol{A}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{B}\boldsymbol{u}(k) &+ \varepsilon_x(k) \\ \boldsymbol{y}^{\Diamond}(k) &= \boldsymbol{C}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{D}\boldsymbol{u}(k) &+ \varepsilon_y(k) \end{cases}$$

with

$$|\varepsilon_x(k)| \leq 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1} \quad \text{and} \quad |\varepsilon_y(k)| \leq 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}.$$



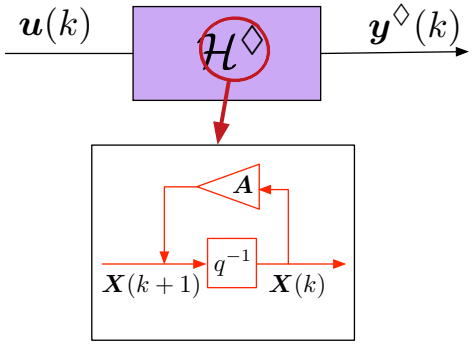
Taking the quantization errors into account

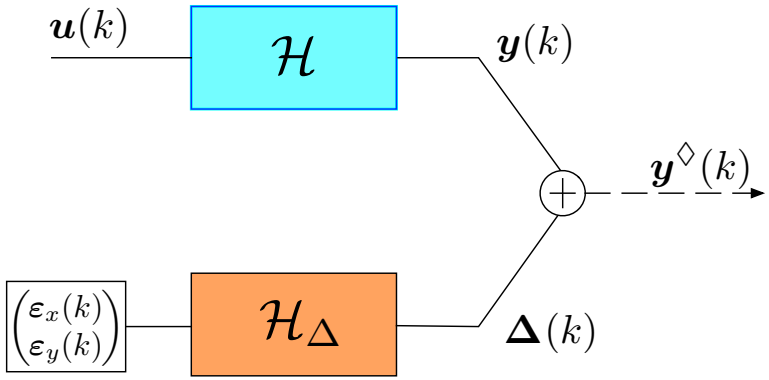
The actually implemented filter $\mathcal{H}^{\Diamond}$ is:

$$\mathcal{H}^{\Diamond} \begin{cases} \boldsymbol{x}^{\Diamond}(k+1) &= \boldsymbol{A}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{B}\boldsymbol{u}(k) &+ \varepsilon_x(k) \\ \boldsymbol{y}^{\Diamond}(k) &= \boldsymbol{C}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{D}\boldsymbol{u}(k) &+ \varepsilon_y(k) \end{cases}$$

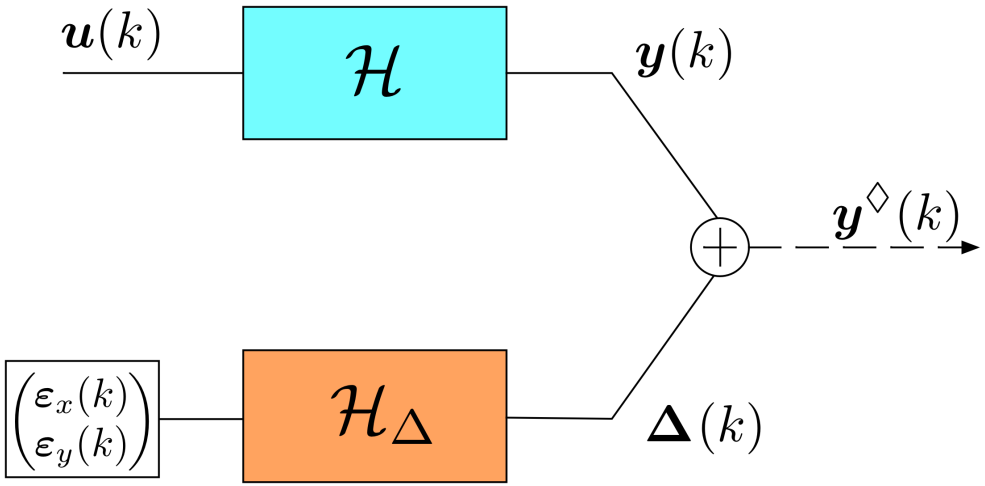
with

$$|\varepsilon_x(k)| \leq 2^{\boldsymbol{m}_x - \boldsymbol{w}_x + 1} \quad \text{and} \quad |\varepsilon_y(k)| \leq 2^{\boldsymbol{m}_y - \boldsymbol{w}_y + 1}.$$

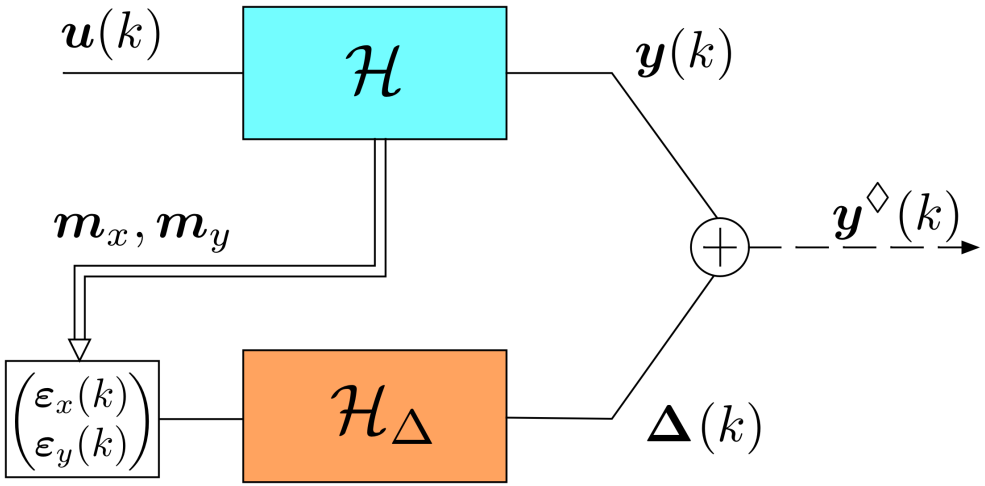




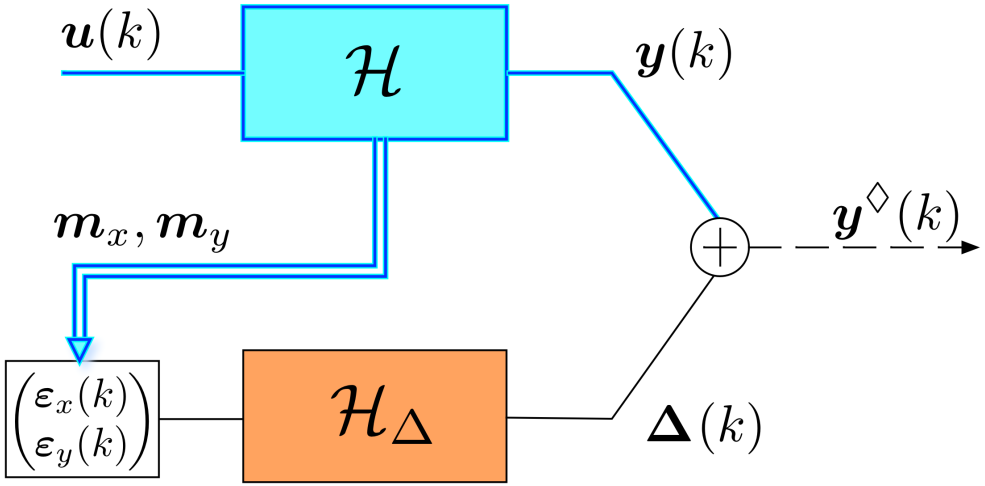
Algorithm


$\boldsymbol{u}(k)$
$\mathcal{H}$
$\boldsymbol{y}(k)$
$\boldsymbol{y}^{\Diamond}(k)$
$\begin{pmatrix}\varepsilon_x(k)\\ \varepsilon_y(k)\end{pmatrix}$
$\mathcal{H}_\Delta$
$\boldsymbol{\Delta}(k)$

Algorithm

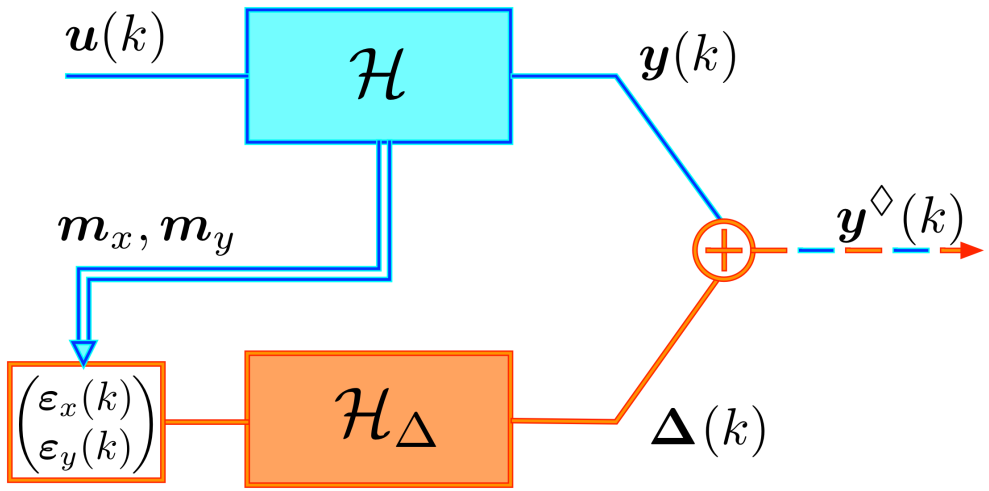

$\boldsymbol{u}(k)$
$\mathcal{H}$
$\boldsymbol{y}(k)$
$\boldsymbol{m}_x, \boldsymbol{m}_y$
$\boldsymbol{y}^{\diamondsuit}(k)$
$\begin{pmatrix}\varepsilon_x(k)\\ \varepsilon_y(k)\end{pmatrix}$
$\mathcal{H}_\Delta$
$\boldsymbol{\Delta}(k)$

Algorithm



Step 1: Determine the initial guess MSBs $\boldsymbol{m}_y$ for the exact filter $\mathcal{H}$

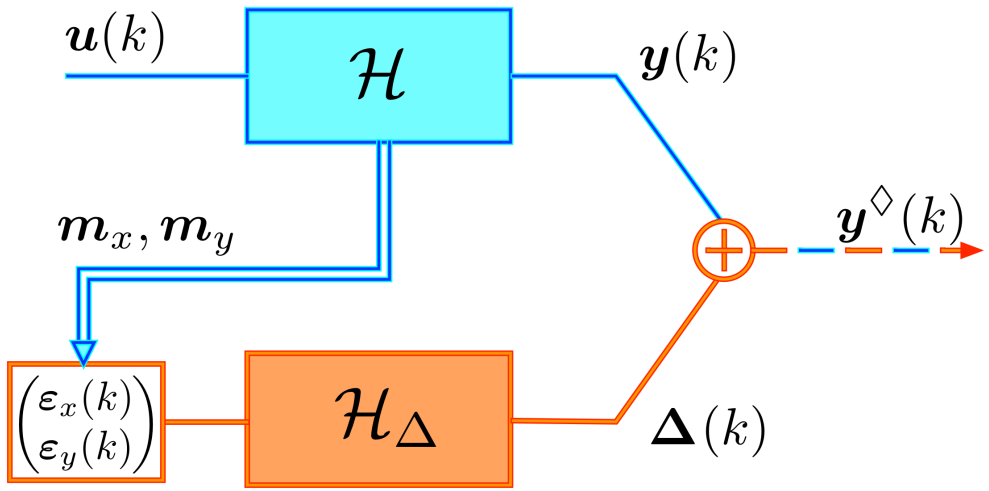
Algorithm



Step 1: Determine the initial guess MSBs $\boldsymbol{m}_y$ for the exact filter $\mathcal{H}$

Step 2: Compute the error-filter $\mathcal{H}_\Delta$, induced by the format $\boldsymbol{m}_y$ and deduce the MSBs $\boldsymbol{m}_\zeta^{\Diamond}$

Algorithm



Step 1: Determine the initial guess MSBs $\boldsymbol{m}_y$ for the exact filter $\mathcal{H}$

Step 2: Compute the error-filter $\mathcal{H}_\Delta$, induced by the format $\boldsymbol{m}_y$ and deduce the MSBs $\boldsymbol{m}_\zeta^\Diamond$

Step 3: If $\boldsymbol{m}_{y_i}^\Diamond == \boldsymbol{m}_{y_i}$ then return $\boldsymbol{m}_{y_i}^\Diamond$
otherwise $\boldsymbol{m}_{y_i} \leftarrow \boldsymbol{m}_{y_i} + 1$ and go to Step 2.

Numerical example

Example:

- Random filter with 3 states, 1 input, 1 output
- $\bar{u} = 5.125$, wordlengths set to 7 bits

	states			output
	$\boldsymbol{x}_1(k)$	$\boldsymbol{x}_2(k)$	$\boldsymbol{x}_3(k)$	$\boldsymbol{y}(k)$
Step 1	6	7	5	6
Step 2	6	7	6	6
Step 3	6	7	6	6

Table: Evolution of the MSB positions

Numerical examples

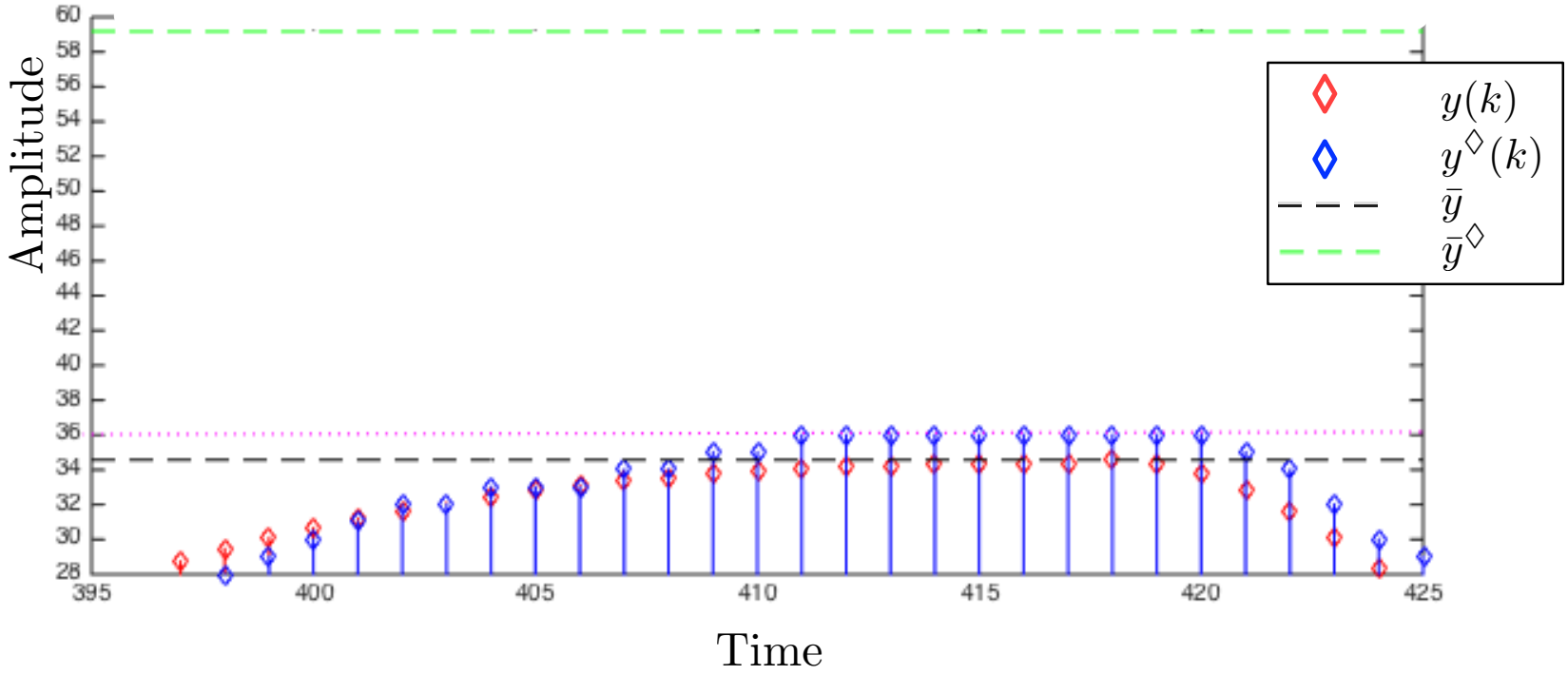


Figure: The exact and quantized outputs of the example.
Quantized output does not pass over to the next binade.

Numerical examples

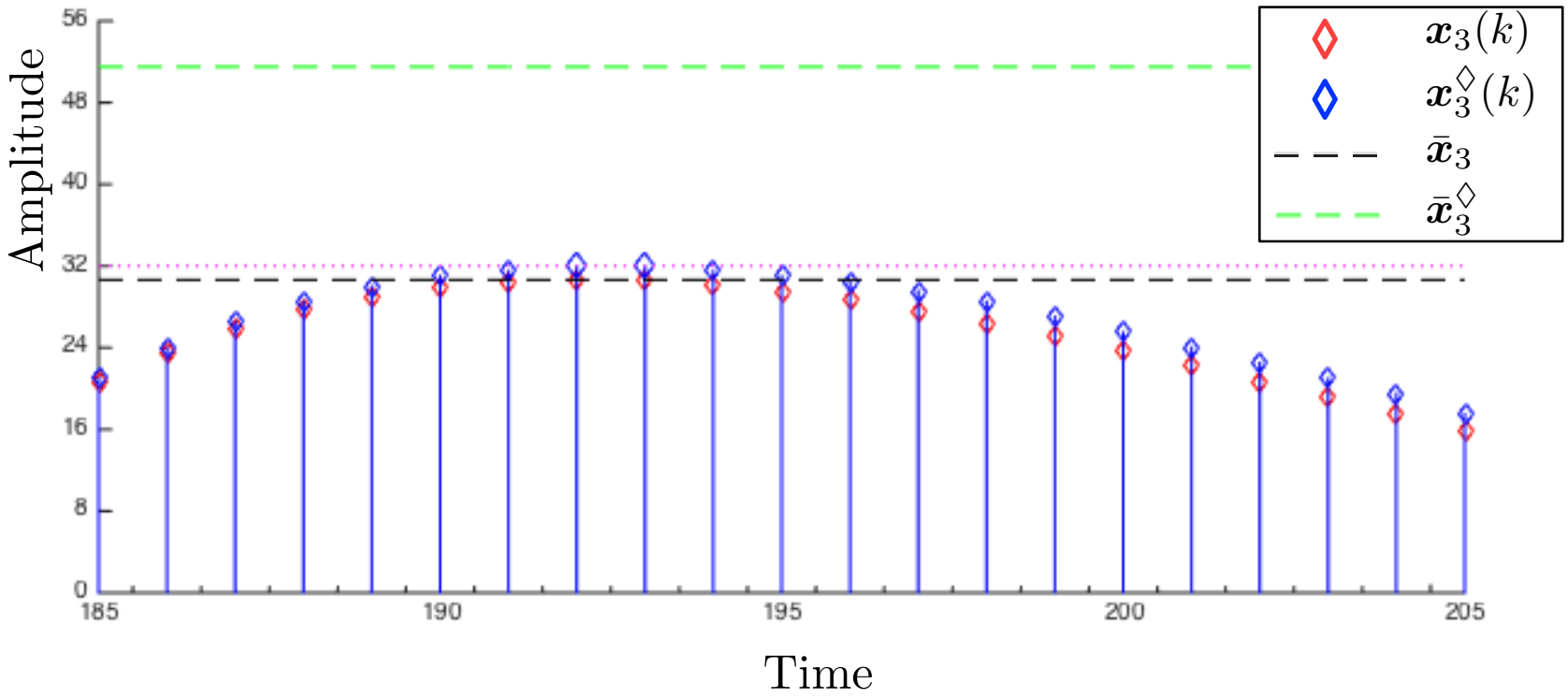


Figure: The exact and quantized third state of the example.
Quantized state passes over to the next binade.

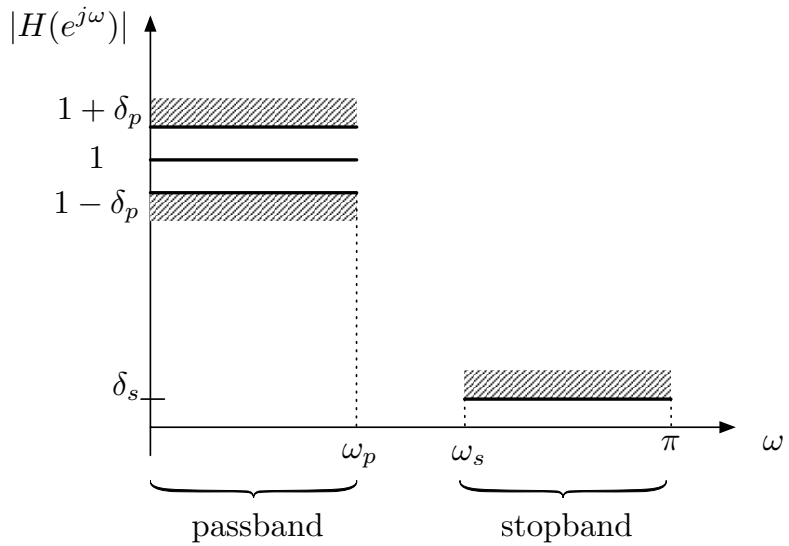
Work in progress:
"Verification of digital *implemented* filters against specification."

Filter specifications

The filters are designed from transfer function specifications in frequency domain:

$$\underline{\beta} \leq \left|H(e^{i\omega})\right| \leq \overline{\beta}, \quad \forall \omega \in [\omega_1, \omega_2]$$

Example: lowpass filter



$$\begin{cases} 1-\delta_p \leq \left|H(e^{i\omega})\right| \leq 1+\delta_p & \forall \omega \in [0, \omega_p] \quad \text{(passband)} \\ \left|H(e^{i\omega})\right| \leq \delta_s & \forall \omega \in [\omega_s, \pi] \quad \text{(stopband)} \end{cases}$$

Reformulating the problem

Need to show that $\forall z \in \{e^{j\omega} | \omega \in \Omega \subset [0, \pi]\}$

$$\underline{\beta} \leq |H(z)| \leq \overline{\beta}$$

Reformulating the problem

Need to show that $\forall z \in \{e^{j\omega} | \omega \in \Omega \subset [0, \pi]\}$

$$\underline{\beta}^2 \leq |H(z)|^2 \leq \overline{\beta}^2$$

Reformulating the problem

Need to show that $\forall z \in \{e^{j\omega} | \omega \in \Omega \subset [0, \pi]\}$

$$\underline{\beta}^2 \leq |H(z)|^2 \leq \overline{\beta}^2$$

We have that

$$|H(z)|^2 = \frac{|b(z)|^2}{|a(z)|^2} = \frac{b(z)\overline{b(\overline{z})}}{a(z)\overline{a(\overline{z})}} = \frac{b(z)b(\frac{1}{z})}{a(z)a(\frac{1}{z})} =: \frac{P(z)}{Q(z)},$$

where $P(z)$ and $Q(z)$ polynomials with real coefficients.

Reformulating the problem

Need to show that $\forall z \in \{e^{j\omega} | \omega \in \Omega \subset [0, \pi]\}$

$$\underline{\beta}^2 \leq |H(z)|^2 \leq \overline{\beta}^2$$

We have that

$$|H(z)|^2 = \frac{|b(z)|^2}{|a(z)|^2} = \frac{b(z)\overline{b(\overline{z})}}{a(z)\overline{a(\overline{z})}} = \frac{b(z)b(\frac{1}{z})}{a(z)a(\frac{1}{z})} =: \frac{P(z)}{Q(z)},$$

where $P(z)$ and $Q(z)$ polynomials with real coefficients.
Therefore, we need to show:

$$\underline{\beta}' \leq \frac{P(z)}{Q(z)} \leq \overline{\beta}'$$

We use the Sollya tool to prove it.

Basic brick: verifying the bounds of a rational function

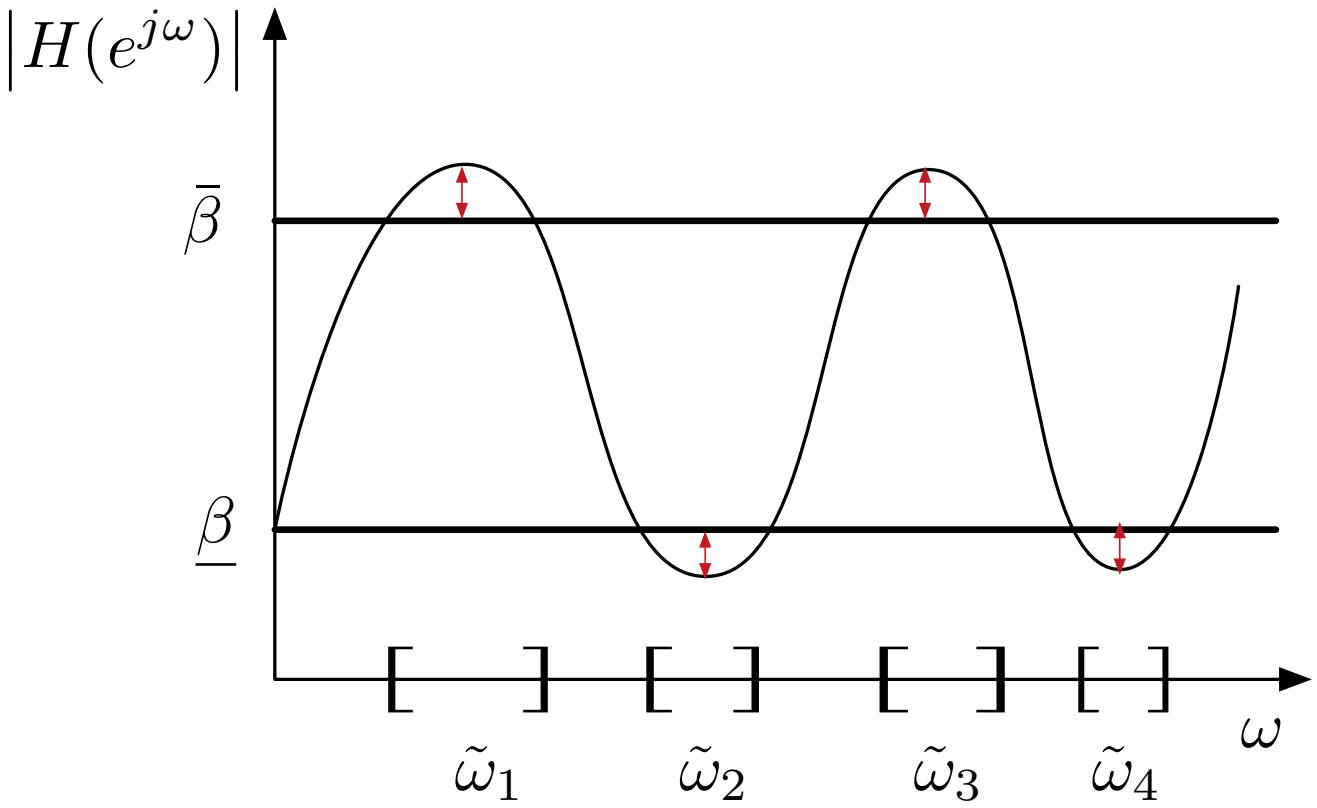


Figure: If a specification is not satisfied, our algorithm returns the problematic frequencies as small intervals $\tilde{\omega}_i$ and maximum overflows within those frequencies.

Verifying a LTI filter implementation

Given a filter implementation compute the corresponding transfer function.

Transfer function of a single-input single-output state-space system:

$$H(z) = \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d.$$

Verifying a LTI filter implementation

Given a filter implementation compute the corresponding transfer function.

Transfer function of a single-input single-output state-space system:

$$H(z) = \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d.$$

Using the eigendecomposition $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{E}\boldsymbol{V}^{-1}$:

$$H(z) = \frac{P(z)}{Q(z)} + d$$

$$P(z) = \sum_{i=1}^{n} (\boldsymbol{c}\boldsymbol{V})_i (\boldsymbol{V}^{-1}\boldsymbol{b})_i \prod_{j \neq i} (z - \lambda_j)$$

$$Q(z) = \prod_{j=1}^{n} (z - \lambda_j)$$

Verifying a LTI filter implementation

Given a filter implementation compute the corresponding transfer function.

Transfer function of a single-input single-output state-space system:

$$H(z) = \boldsymbol{c}(z\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{b} + d.$$

Using the eigendecomposition $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{E}\boldsymbol{V}^{-1}$:

$$H(z) = \frac{P(z)}{Q(z)} + d$$

$$P(z) = \sum_{i=1}^{n} (\boldsymbol{c}\boldsymbol{V})_i (\boldsymbol{V}^{-1}\boldsymbol{b})_i \prod_{j \neq i} (z - \lambda_j)$$

$$Q(z) = \prod_{j=1}^{n} (z - \lambda_j)$$

We can compute an approximation $\widehat{H}(z)$ in Multiple Precision arithmetic.

Verifying a LTI filter implementation

Bounding the approximation error:

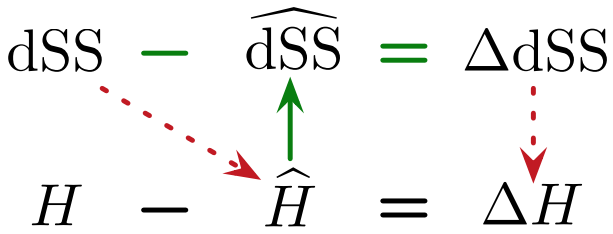


Figure: In red: inexact transformation. In green: exact transformation.

Verifying a LTI filter implementation

Bounding the approximation error:

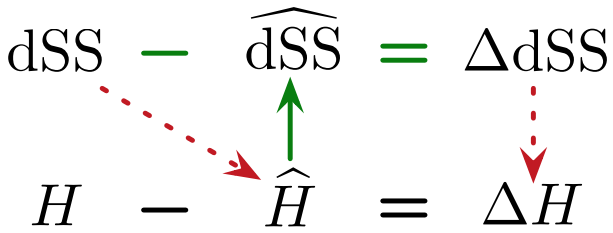


Figure: In red: inexact transformation. In green: exact transformation.

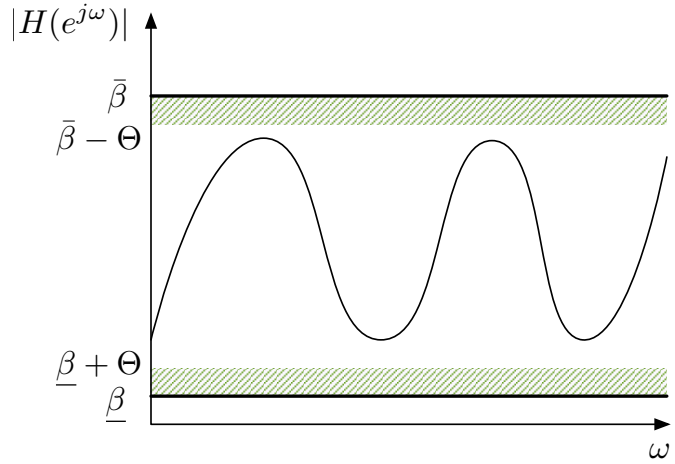
Lemma

Given a discrete state-space system, the error of a multiple precision approximation $\widehat{H}(z)$ on its transfer function is bounded by:

$$\left|H(z) - \widehat{H}(z)\right| \leq \Theta,$$

where $\Theta = \langle\langle \Delta dSS \rangle\rangle + \varepsilon$ is the Worst-Case Peak Gain of the system ΔdSS computed with arbitrary small absolute error bounded by the $\varepsilon > 0$.

Verifying a LTI filter implementation



Lemma

If the approximation $\widehat{H}(z)$ satisfies

$$\underline{\beta} + \Theta \leq \left|\widehat{H}(e^{i\omega})\right| \leq \overline{\beta} - \Theta, \quad \forall \omega \in \Omega \tag{1}$$

Then the exact transfer function $H(z)$ verifies the initial specifications

$$\underline{\beta} \leq \left|H(e^{i\omega})\right| \leq \overline{\beta}, \quad \forall \omega \in \Omega \tag{2}$$

Numerical results: comparing filter design tools

		Butterworth	Chebyshev	Elliptic
lowpass	MATLAB	1.29e-17	7.93e-17	✓
	SciPy	2.14e-15	**4.48e-2**	**4.48e-2**
highpass	MATLAB	2.77e-16	6.94e-17	**4.48e-2**
	SciPy	3.02e-15	2.29e-16	**4.48e-2**
bandpass	MATLAB	3.04e-17	✓	✓
	SciPy	✓	**4.48e-2**	**4.48e-2**
bandstop	MATLAB	4.59e-16	3.09e-15	✓
	SciPy	✓	6.36e-15	**7.02e-6**

Table: Quality of the filters designd with MATLAB vs. SciPy for some simple filters. For example, a lowpass filter here has $\delta_p = 10^{-\frac{1}{20}}$, $\delta_s = 10^{-1}$, $\omega_p = 0.4\pi$, $\omega_s = 0.5\pi$

Numerical results: comparing filter structures

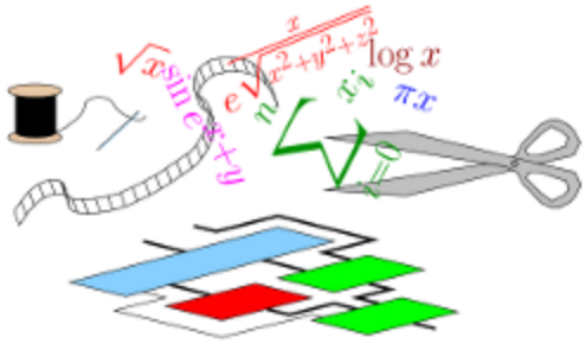
	filter	Butterworth			Chebyshev		
	wordlength	32	16	8	32	16	8
DFIIt	margin	✓	✓	1.17e-2	1.97e-10	7.03e-6	7.42e-3
	time	12s	15s	1min17s	1min22s	35s	18s
ρDFIIt	margin	✓	6.78e-2	✗	1.21e-9	4.72e-4	8.35e-2
	time	13s	3min17s	unstable	1min26s	1min02s	17s
SS	margin	✓	9.11e-7	9.64e-3	2.06e-10	7.64e-6	2.23e-3
	time	15s	2min4s	1min01s	23s	49s	18s

Table: Filter structures for a lowpass MATLAB-designed filter.

Work in progress: "Code generation for FPGAs"

with Matei Istoan and Florent de Dinechin

FloPoCo



Circuits computing just right

Binding with FloPoCo

FloPoCo
synthesis of arithmetic units
(i.e. Sum-of-Products)

$+$

SIF
unifying representation
(a bunch of Sum-of-Products)

$\Longrightarrow$

FPGA
implementation

Binding with FloPoCo

FloPoCo		**SIF**		
synthesis of arithmetic units (i.e. Sum-of-Products)	+	unifying representation (a bunch of Sum-of-Products)	$\Longrightarrow$	**FPGA** implementation

FloPoCo: requires the the desired Fixed-Point Format of the output of a Sum-of-Product.

SIF: we deduce a lower bound on the error of computation of each Sum-of-Product s.t. the error-bound *on the filter's output* is respected.

Naive approach: the error-budget is distrubuted equally

Binding with FloPoCo

FloPoCo		**SIF**		
synthesis of arithmetic units (i.e. Sum-of-Products)	+	unifying representation (a bunch of Sum-of-Products)	$\Longrightarrow$	**FPGA** implementation

FloPoCo: requires the the desired Fixed-Point Format of the output of a Sum-of-Product.

SIF: we deduce a lower bound on the error of computation of each Sum-of-Product s.t. the error-bound *on the filter's output* is respected.

Naive approach: the error-budget is distrubuted equally

We can do better

Take into account different impact of different Sums-of-Products.
BUT need to solve non-linear optimization problem

Conclusion

- Proposed a new completely rigorous approach for the Fixed-Point implementation of linear digital filters
- Provided reliable evaluation of the WCPG measure
- Applied the WCPG measure to determine the FxPF that guarantee that overflow occurs
- Proposed a new approach for the accurate computation of the transfer function of an implemented filter
- Proposed a rigorous approach on the verification of a LTI filter implementation against its specification
- Binding with FloPoCo gives possibility to fairly compare different filter realizations

Perspectives

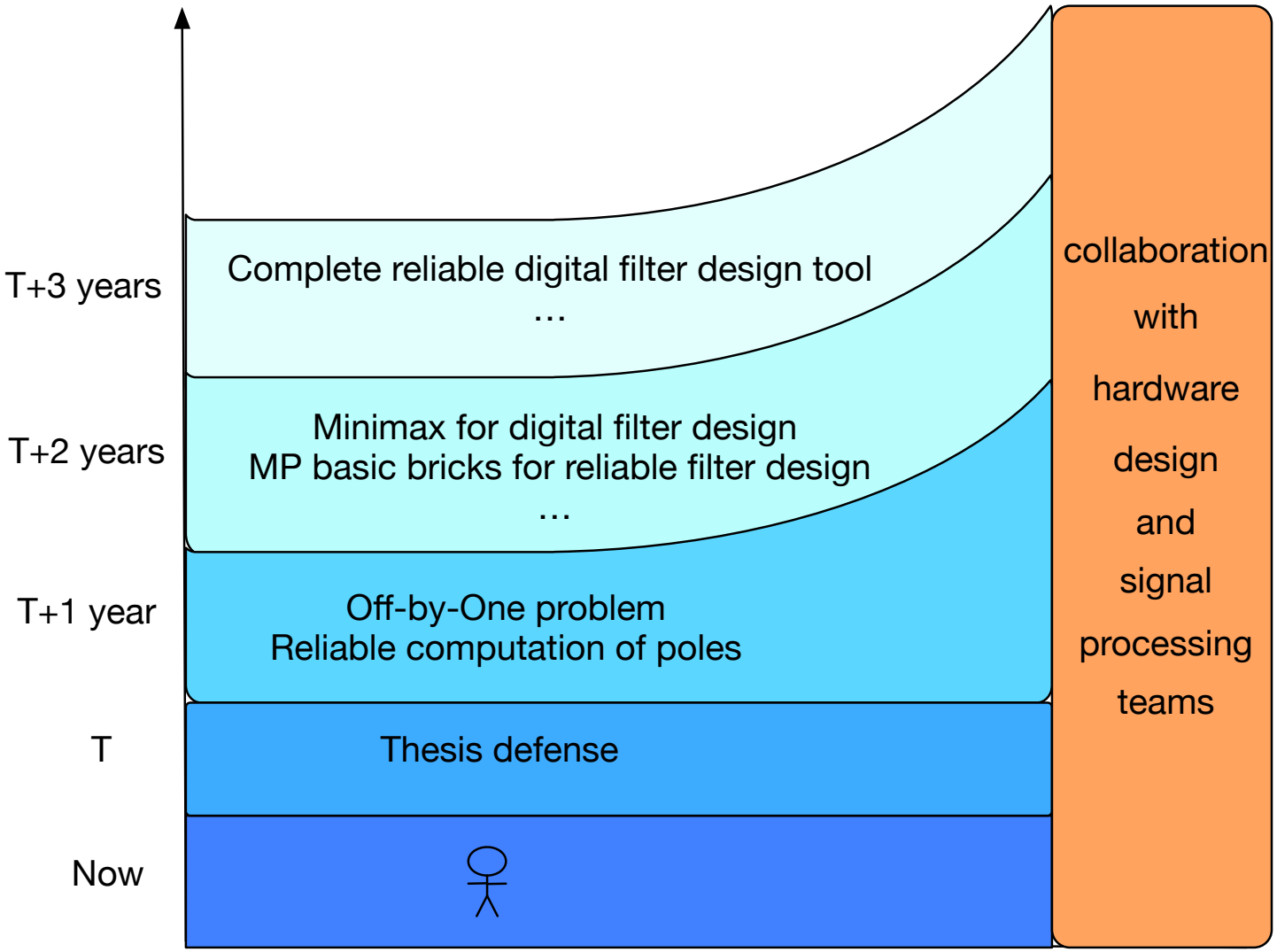
Filter design and implementation:

- Plug all the stages of the generator into optimization routines
- Fairly compare finite precision implementation under various constraints
- Analysis of (short) floating-point implementations

From computer arithmetic point of view:

- Eigendecomposition with a priori absolute error bound
- Off-by-one problem
- Optimize the quantization of filter coefficients using our specification verification algorithm
- Basic bricks multiple precision algorithms (transfer function, Lyapunov equations,...)

Future plans...



Thank you!
Questions?

L_2-norm evaluation

Another related problem is the reliable evaluation of the L_2-norm.
If $\boldsymbol{H}$ is a transfer function, then its L_2-norm is defined by

$$\|\boldsymbol{H}\|_2 \triangleq \sqrt{\frac{1}{2\pi}\int_0^{2\pi} \|\boldsymbol{H}(e^{j\omega})\|_F^2 \, d\omega}$$

Parseval's theorem gives another expression when $\boldsymbol{H}$ is described with state-space matrices $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{D}$:

$$\begin{aligned} \|\boldsymbol{H}\|_2 &= \sqrt{tr(\boldsymbol{C}\boldsymbol{W}_c\boldsymbol{C}^\top + \boldsymbol{D}\boldsymbol{D}^\top)} \\ &= \sqrt{tr(\boldsymbol{B}^\top\boldsymbol{W}_o\boldsymbol{B} + \boldsymbol{D}^\top\boldsymbol{D})} \end{aligned}$$

where $\boldsymbol{W}_c$ and $\boldsymbol{W}_o$ are the controllability and observability Gramians of the system.

Gramians

- $\boldsymbol{W}_c$ is the controllability Gramian of the system.

$$\boldsymbol{W}_c \triangleq \sum_{k=0}^{\infty} (\boldsymbol{A}^k \boldsymbol{B})(\boldsymbol{A}^k \boldsymbol{B})^\top$$

$\boldsymbol{W}_c$ is the solution of the discrete-time Lyapunov equation

$$\boldsymbol{W}_c = \boldsymbol{A}\boldsymbol{W}_c\boldsymbol{A}^\top + \boldsymbol{B}\boldsymbol{B}^\top$$

- $\boldsymbol{W}_o$ is the observability Gramian of the system.

$$\boldsymbol{W}_o \triangleq \sum_{k=0}^{\infty} (\boldsymbol{C}\boldsymbol{A}^k)^\top (\boldsymbol{C}\boldsymbol{A}^k)$$

$\boldsymbol{W}_o$ is the solution of the discrete-time Lyapunov equation

$$\boldsymbol{W}_o = \boldsymbol{A}^\top \boldsymbol{W}_o \boldsymbol{A} + \boldsymbol{C}^\top \boldsymbol{C}$$

Computation of the Gramians

The Gramians are usually computed by solving the discrete-time Lyapunov equation $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^\top + \boldsymbol{Q}$
The following methods can be used:

- solve $(\boldsymbol{I} - \boldsymbol{A} \otimes \boldsymbol{A})\boldsymbol{x} = \boldsymbol{q}$
  where $\boldsymbol{x} = Vec(\boldsymbol{X})$ and $\boldsymbol{q} = Vec(\boldsymbol{Q})$
  $\rightarrow$ numerically inefficient
- use infinite sum $\sum_{k=0}^{\infty} \boldsymbol{A}^k \boldsymbol{Q} \boldsymbol{A}^{k\top}$
  $\rightarrow$ may required a lot of computation
- use Hammarling's method, based on Schur decomposition of matrix $\boldsymbol{A}$
  $\rightarrow$ efficient, but required a deep analysis of the computational errors of the algorithm

see "Computational methods for linear matrix equations", V. Simoncini

Reliable computation of the L_2-norm

Questions

- How to have a reliable evaluation of the L_2-norm in multiple precision
- How to proceed when $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are interval matrices (small radii, containing previously computed errors)

Step 1. Bound on truncation error

Truncation error is the tail of the infinite sum:

$$\sum_{k>N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

Step 1. Bound on truncation error

Truncation error is the tail of the infinite sum:

$$\sum_{k>N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

Suppose $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$, where $\boldsymbol{E} = diag(\lambda_1, \ldots, \lambda_n)$ is the eigenvalue matrix and $\boldsymbol{X}$ is the eigenvector matrix. Then,

$$\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{E}^k\boldsymbol{X}^{-1}\boldsymbol{B} = \sum_{l=1}^{n} \boldsymbol{R}_l\lambda_l^k$$

Step 1. Bound on truncation error

Truncation error is the tail of the infinite sum:

$$\sum_{k>N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

Suppose $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$, where $\boldsymbol{E} = diag(\lambda_1, \ldots, \lambda_n)$ is the eigenvalue matrix and $\boldsymbol{X}$ is the eigenvector matrix. Then,

$$\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{E}^k\boldsymbol{X}^{-1}\boldsymbol{B} = \sum_{l=1}^{n} \boldsymbol{R}_l \lambda_l^k$$

Bound on truncation error

$$\sum_{k>N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right| \leq \rho(\boldsymbol{A})^{N+1} \boldsymbol{M}$$

$$\boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1 - |\lambda_l|} \frac{|\lambda_l|}{\rho(\boldsymbol{A})}$$

Step 1. Bound on truncation error

Truncation error is the tail of the infinite sum:

$$\sum_{k>N} \left| \boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} \right|$$

Suppose $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$, where $\boldsymbol{E} = diag(\lambda_1, \ldots, \lambda_n)$ is the eigenvalue matrix and $\boldsymbol{X}$ is the eigenvector matrix. Then,

$$\boldsymbol{C}\boldsymbol{A}^k\boldsymbol{B} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{E}^k\boldsymbol{X}^{-1}\boldsymbol{B} = \sum_{l=1}^{n} \boldsymbol{R}_l \lambda_l^k$$

Bound on truncation error

$$\rho(\boldsymbol{A})^{N+1}\boldsymbol{M} \overset{!}{\leq} \varepsilon_1$$

$$\boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1-|\lambda_l|} \frac{|\lambda_l|}{\rho(\boldsymbol{A})}$$

Step 1. Bound on truncation order

Lower bound on truncation order

$$N \geq \left\lceil \frac{\log \frac{\varepsilon_1}{m}}{\log \rho(\boldsymbol{A})} \right\rceil$$

$$\boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1 - |\lambda_l|} \frac{|\lambda_l|}{\rho(\boldsymbol{A})}$$

where m is defined as $m := \min\limits_{i,j} |\boldsymbol{M}_{i,j}|$.

Step 1. Bound on truncation order

Lower bound on truncation order

$$N \geq \left\lceil \frac{\log \frac{\varepsilon_1}{m}}{\log \rho(\boldsymbol{A})} \right\rceil$$

$$\boldsymbol{M} := \sum_{l=1}^{n} \frac{|\boldsymbol{R}_l|}{1 - |\lambda_l|} \frac{|\lambda_l|}{\rho(\boldsymbol{A})}$$

where m is defined as $m := \min_{i,j} |\boldsymbol{M}_{i,j}|$.

Reliable evaluation

Interval Arithmetic and Rump's Theory of Verified Inclusions are used to determine a rigorous bound of N.

Step 2. "Diagonalization" of matrix $\mathbf{A}$

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2$$

Step 2. "Diagonalization" of matrix $\boldsymbol{A}$

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2$$

- $\boldsymbol{V}$ is some approximation on $\boldsymbol{X}$
- $\boldsymbol{\Delta}_2$ represents the element-by-element errors due to the two matrix multiplications and the inversion of matrix $\boldsymbol{V}$

Step 2. "Diagonalization" of matrix $\boldsymbol{A}$

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2$$

- $\boldsymbol{V}$ is some approximation on $\boldsymbol{X}$
- $\boldsymbol{\Delta}_2$ represents the element-by-element errors due to the two matrix multiplications and the inversion of matrix $\boldsymbol{V}$
- $\boldsymbol{T}$ diagonal in dominant with very small other elements
- $\|\boldsymbol{T}\|_2 \leq 1$

Step 2. "Diagonalization" of matrix $\boldsymbol{A}$

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2$$
$$\boldsymbol{A}^k = \boldsymbol{V}(\boldsymbol{T} + \boldsymbol{\Delta}_2)^k \boldsymbol{V}^{-1}$$

The error of substitution of $\boldsymbol{A}$ by $\boldsymbol{V}\boldsymbol{T}\boldsymbol{V}^{-1}$:

$$\sqrt{n}(N+1)(N+2)\,\|\boldsymbol{\Delta}_2\|_F\,\|\boldsymbol{C}\boldsymbol{V}\|_F\,\left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F$$

Step 2. "Diagonalization" of matrix $\boldsymbol{A}$

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2$$
$$\boldsymbol{A}^k = \boldsymbol{V}(\boldsymbol{T} + \boldsymbol{\Delta}_2)^k \boldsymbol{V}^{-1}$$

The error of substitution of $\boldsymbol{A}$ by $\boldsymbol{V}\boldsymbol{T}\boldsymbol{V}^{-1}$:

$$\sqrt{n}(N+1)(N+2)\,\|\boldsymbol{\Delta}_2\|_F\,\|\boldsymbol{C}\boldsymbol{V}\|_F\,\left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F \overset{!}{\leq} \varepsilon_2$$

Step 2. "Diagonalization" of matrix $\boldsymbol{A}$

$$\boldsymbol{T} := \boldsymbol{V}^{-1}\boldsymbol{A}\boldsymbol{V} - \boldsymbol{\Delta}_2$$
$$\boldsymbol{A}^k = \boldsymbol{V}(\boldsymbol{T} + \boldsymbol{\Delta}_2)^k \boldsymbol{V}^{-1}$$

The error of substitution of $\boldsymbol{A}$ by $\boldsymbol{V}\boldsymbol{T}\boldsymbol{V}^{-1}$:

$$\sqrt{n}(N+1)(N+2)\,\|\boldsymbol{\Delta}_2\|_F\,\|\boldsymbol{C}\boldsymbol{V}\|_F\,\left\|\boldsymbol{V}^{-1}\boldsymbol{B}\right\|_F \overset{!}{\leq} \varepsilon_2$$

A condition on the error-matrix $\boldsymbol{\Delta}_2$:

$$\|\boldsymbol{\Delta}_2\|_F \leq \frac{1}{\sqrt{n}(N+1)(N+2)} \frac{\varepsilon_2}{\|\boldsymbol{C}\boldsymbol{V}\|_F\,\|\boldsymbol{V}^{-1}\boldsymbol{B}\|_F}$$

Step 3. Computing products $\boldsymbol{C}'$ and $\boldsymbol{B}'$

$$\boldsymbol{C}' := \boldsymbol{C}\boldsymbol{V} + \boldsymbol{\Delta}_{3_C}$$
$$\boldsymbol{B}' := \boldsymbol{V}^{-1}\boldsymbol{B} + \boldsymbol{\Delta}_{3_B}$$

where $\boldsymbol{\Delta}_{3_C} \in \mathbb{C}^{p\times n}$ and $\boldsymbol{\Delta}_{3_B} \in \mathbb{C}^{n\times q}$ are error-matrices.

Bound on the multiplication errors $\boldsymbol{\Delta}_{3_C}$ and $\boldsymbol{\Delta}_{3_B}$:

$$\|\boldsymbol{\Delta}_{3_C}\|_F \leq \frac{1}{3\sqrt{n}} \cdot \frac{1}{N+1} \frac{\varepsilon_3}{\|\boldsymbol{C}'\|_F}$$
$$\|\boldsymbol{\Delta}_{3_B}\|_F \leq \frac{1}{3\sqrt{n}} \cdot \frac{1}{N+1} \frac{\varepsilon_3}{\|\boldsymbol{B}'\|_F}.$$

Step 4. Powering $\boldsymbol{T}$

$$\boldsymbol{P}_k := \boldsymbol{T}^k - \boldsymbol{\Delta}_{4_k}$$

$\boldsymbol{\Delta}_{4_k} \in \mathbb{C}^{n\times n}$ error-matrix on matrix powers, including error propagation from the first to the last power.

$$\boldsymbol{P}_k = \boldsymbol{T}\boldsymbol{P}_{k-1} + \boldsymbol{\Gamma}_k,$$

where $\boldsymbol{\Gamma}_k \in \mathbb{C}^{n\times n}$ is the error-matrix on the error of the matrix multiplication at step k.

Bound on the error-matrix $\boldsymbol{\Gamma}_k$

$$\|\boldsymbol{\Gamma}_k\|_F \leq \frac{1}{\sqrt{n}} \cdot \frac{1}{N-1} \cdot \frac{1}{N+1} \cdot \frac{\varepsilon_4}{\|\boldsymbol{C'}\|_F \, \|\boldsymbol{B'}\|_F}$$

Step 5. Computing L_k

$$\boldsymbol{L}_k := \boldsymbol{C}'\boldsymbol{P}_k\boldsymbol{B}' + \boldsymbol{\Delta}_{5_k},$$

where $\boldsymbol{\Delta}_{5_k} \in \mathbb{C}^{p\times q}$ is the matrix of element-by-element errors for the two matrix multiplications.

Bound on the error-matrix $\boldsymbol{\Delta}_{5_k}$

$$|\boldsymbol{\Delta}_{5_k}| \leq \frac{1}{N+1} \cdot \varepsilon_5.$$

Step 6. Summation

$$S_N = |\boldsymbol{D}| + \sum_{l=0}^{N} |\boldsymbol{L}_l| + \boldsymbol{\Delta}_6,$$

where the error-matrix $\boldsymbol{\Delta}_6 \in \mathbb{C}^{p \times q}$ represents the error of $N+1$ absolute value accumulations.

Bound on the error matrix $\boldsymbol{\Delta}_{6_k}$

$$\boldsymbol{\Delta}_{6_k} \leq \frac{1}{N}\varepsilon_6, \qquad k = 1 \ldots N$$