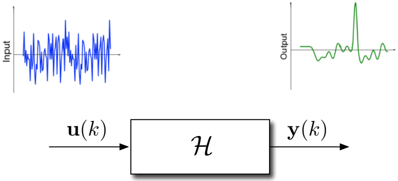
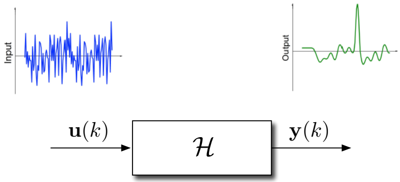
Towards reliable implementation of Digital Filters in Fixed-Point arithmetic

Anastasia Volkova, Thibault Hilaire, Christoph Lauter

RAIM 2016
June 30, 2016

Context: digital filters



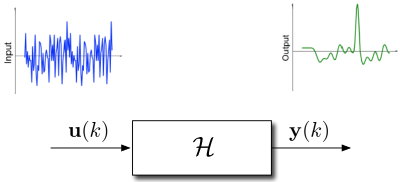


On the one hand

- 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-Point Arithmetic

LTI filters

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

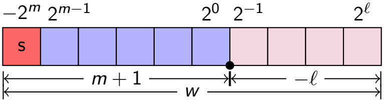
$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

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

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

The input $\boldsymbol{u}(k)$ is considered bounded by $\bar{\boldsymbol{u}}$.

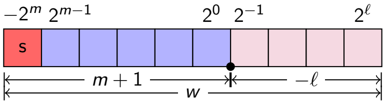
Two's complement Fixed-Point arithmetic



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

- 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:

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

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

[1] http://www.mathworks.com

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation

[1] http://www.mathworks.com

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation
2) deduce the magnitudes

[1] http://www.mathworks.com

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation
2) deduce the magnitudes
3) set *some* wordlength

[1] `http://www.mathworks.com`

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation
2) deduce the magnitudes
3) set *some* wordlength
4) compute the Fixed-Point formats

[1] `http://www.mathworks.com`

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation
2) deduce the magnitudes
3) set *some* wordlength
4) compute the Fixed-Point formats
5) compare to reference filter

[1] `http://www.mathworks.com`

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation
2) deduce the magnitudes
3) set *some* wordlength
4) compute the Fixed-Point formats
5) compare to reference filter
6) if not convinced, increase the wordlength and return to Step 4

[1] http://www.mathworks.com

Fixed-Point IIR filter implementation using Matlab®

Fixed-Point implementation in practice: simulation using Matlab/Simulink[1] tools:

1) random system simulation
2) deduce the magnitudes
3) set *some* wordlength
4) compute the Fixed-Point formats
5) compare to reference filter
6) if not convinced, increase the wordlength and return to Step 4

Unsatisfactory process!

Non-exhaustive simulations, using a floating-point simulation as reference
$\rightarrow$ no guarantees on the implementation

[1] `http://www.mathworks.com`

Example using Matlab

A random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

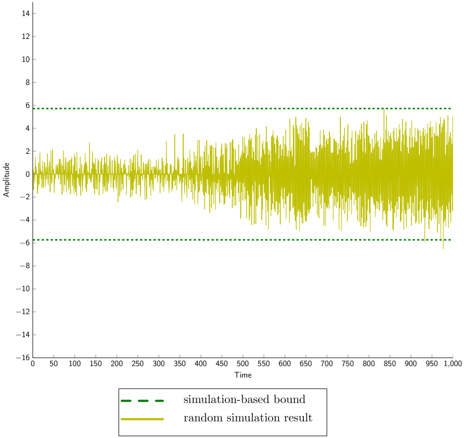
Example using Matlab

A random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Fixed-Point implementation:

- Simulating for $k = 0, \ldots, 1000$
- 1000 random input sequences
- $\bar{y}_{\text{sim}} = 5.72$



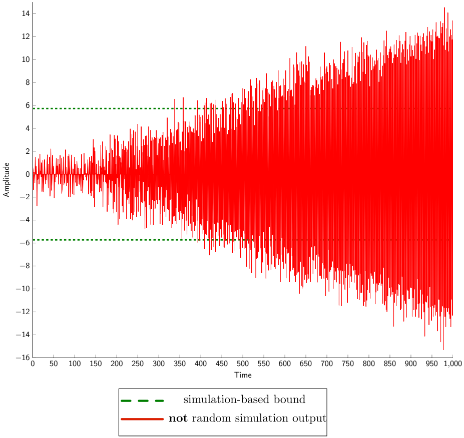
Example using Matlab

A random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Fixed-Point implementation:

- Simulating for $k = 0, \ldots, 1000$
- 1000 random input sequences
- $\bar{y}_{\text{sim}} = 5.72$



⚠ Simulation is not exhaustive

Simulation-based approach is not rigorous. What to do?

Our approach: reliable Fixed-Point implementation

Input:

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

Determine rigorously the Fixed-Point Formats s.t.

- the least MSBs
- **no overflows**

  $\rightsquigarrow$ pay attention to computational errors

Our approach: reliable Fixed-Point implementation

Input:

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

Determine rigorously the Fixed-Point Formats s.t.

- the least MSBs
- **no overflows**

  $\rightsquigarrow$ pay attention to computational errors

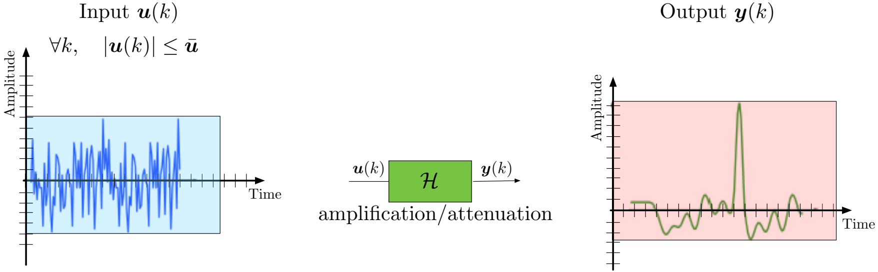
Our approach:

1) determine analytically the output interval of all variables
2) analyze propagation of the error in filter implementation and determine the Fixed-point formats

Deducing the output interval[2]

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

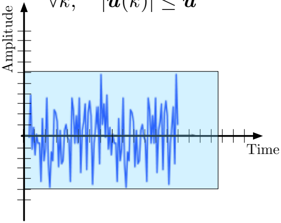
Basic brick: the Worst-Case Peak Gain theorem

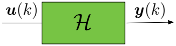


Basic brick: the Worst-Case Peak Gain theorem

Input $\boldsymbol{u}(k)$

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

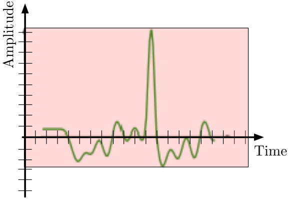




amplification/attenuation

Output $\boldsymbol{y}(k)$

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



Worst-Case Peak Gain

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

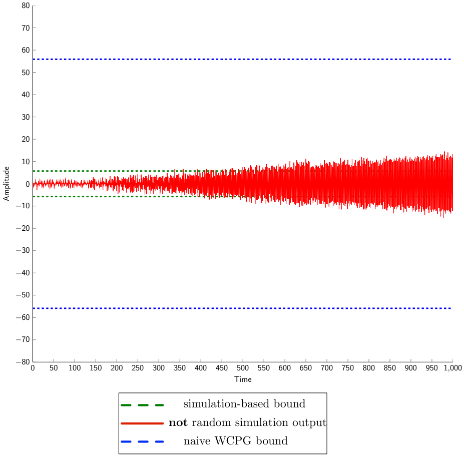
Example (continued)

Our random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Naive WCPG computation

- sum over 1000 terms
- $\bar{y}_{\mathrm{WCPG}} = 55.91 \; \left(\bar{y}_{\mathrm{sim}} = 5.72\right)$



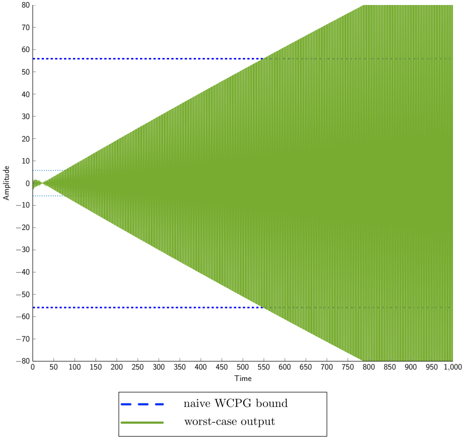
Example (continued)

Our random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Naive WCPG computation

- sum over 1000 terms
- $\bar{y}_{\text{WCPG}} = 55.91 \ \left(\bar{y}_{\text{sim}} = 5.72\right)$



Example (continued)

Our random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

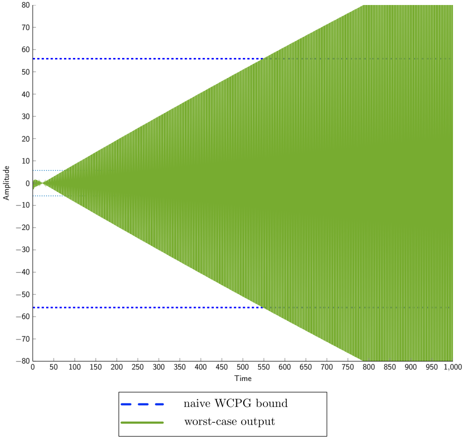
- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Naive WCPG computation

- sum over 1000 terms
- $\bar{y}_{\text{WCPG}} = 55.91 \; \left(\bar{y}_{\text{sim}} = 5.72\right)$

⚠ Still not reliable. Why?

$\rightsquigarrow$ not enough terms for the WCPG



Example (continued)

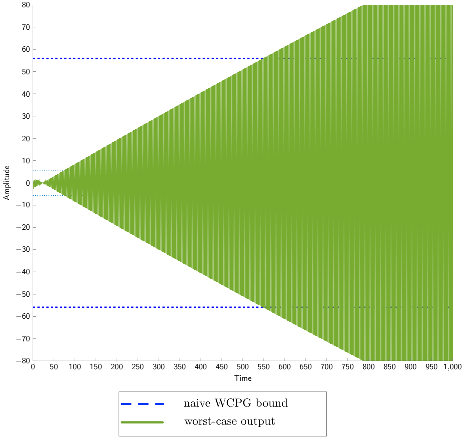
Our random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Naive WCPG computation

- sum over 1000 terms
- $\bar{y}_{\text{WCPG}} = 55.91 \ \left(\bar{y}_{\text{sim}} = 5.72\right)$

⚠ Still not reliable. Why?
$\rightsquigarrow$ not enough terms for the WCPG



How to compute the WCPG matrix **reliably**?

Computing the Worst-Case Peak Gain

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

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

- Deduce reliable lower bound on truncation order
- 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

$$\left| \sum_{k=0}^{\infty} |CA^kB| \quad - \quad \sum_{k=0}^{N} |CA^kB| \right| \leq \varepsilon_1$$

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

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

where

λ – eigenvalues of matrix A

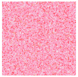
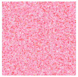
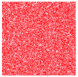
R_l – l^{th}residue matrix computed out of C, B, λ

Powering

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

Powering

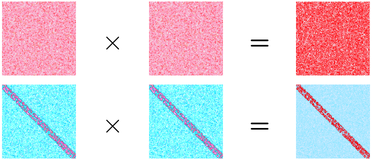
$$\sum_{k=0}^{N} \left| \boldsymbol{C} \boldsymbol{A}^{k} \boldsymbol{B} \right|$$

 $\times$  $=$ 

cancellation

Powering

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

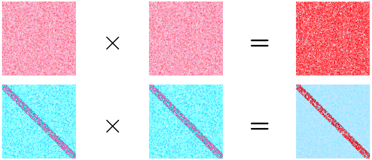


cancellation

less cancellation

Powering

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

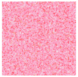
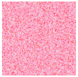
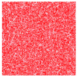


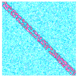
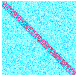
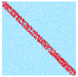
cancellation

less cancellation

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

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

 $\times$  $=$  cancellation

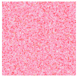
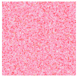
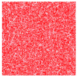
 $\times$  $=$  less cancellation

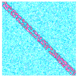
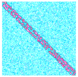
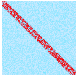
$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{E}\boldsymbol{X}^{-1}$$

$\boldsymbol{V} \approx \boldsymbol{X}$ 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

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

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

Apply the same approach for the other steps:

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

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

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{P}_k \boldsymbol{B}' \right| - \sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| \right| \leq \varepsilon_5$$

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| - \quad \boldsymbol{S}_N \right| \leq \varepsilon_6$$

Further steps

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

Apply the same approach for the other steps:

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

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

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{C}' \boldsymbol{P}_k \boldsymbol{B}' \right| - \sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| \right| \leq \varepsilon_5$$

$$\left| \sum_{k=0}^{N} \left| \boldsymbol{L}_k \right| - \quad \boldsymbol{S}_N \right| \leq \varepsilon_6$$

We can determine the output interval of a filter with arbitrary precision.

Example (continued)

Our random 5^{th} order Butterworth:

5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

Example (continued)

Our random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$

We computed WCPG with $\varepsilon = 2^{-64}$:

Approach	N	$\bar{y}$
Simulation	-	5.72
Naive WCPG	1 000	55.91
Our WCPG	**352 158**	**772.04**

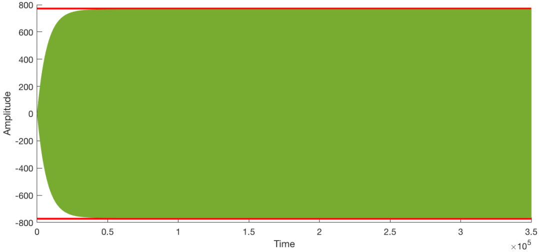


Figure: Output $y(k)$ reaches a ε-neighborhood of $\bar{y}$.

Determining the Fixed-Point Formats[3]

[3] 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

$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

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 $\boldsymbol{m}_y$ 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

$$\mathcal{H}\left\{\begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{D}\boldsymbol{u}(k) \end{array}\right.$$

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 $\boldsymbol{m}_y$ 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

$$\mathcal{H}\left\{\begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k)+\boldsymbol{D}\boldsymbol{u}(k) \end{array}\right.$$

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 $\boldsymbol{m}_y$ 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 $0 \leq \widehat{\boldsymbol{m}}_{y_i} - \boldsymbol{m}_{y_i} \leq 1$

Taking the quantization errors into account

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

$$\mathcal{H} \left\{ \begin{array}{rcl} \boldsymbol{x}(k+1) & = & \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) \\ \boldsymbol{y}(k) & = & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{u}(k) \end{array} \right.$$

Taking the quantization errors into account

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

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

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:

$$\mathcal{H}^{\Diamond} \left\{ \begin{array}{rcl} \boldsymbol{x}^{\Diamond}(k+1) & = & \Diamond_{m_x}(\boldsymbol{A}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{B}\boldsymbol{u}(k)) \; + \; \varepsilon_x(k) \\ \boldsymbol{y}^{\Diamond}(k) & = & \Diamond_{m_y}(\boldsymbol{C}\boldsymbol{x}^{\Diamond}(k) + \boldsymbol{D}\boldsymbol{u}(k) \; + \varepsilon_y(k) \end{array} \right.$$

with

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

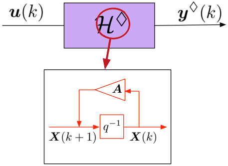
Taking the quantization errors into account

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

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

with

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



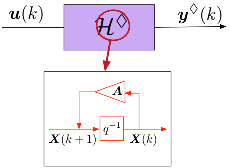
Taking the quantization errors into account

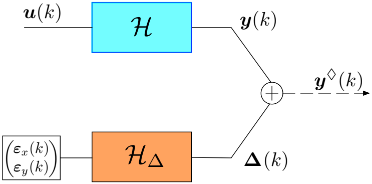
The actually implemented filter $\mathcal{H}^\Diamond$ is:

$$\mathcal{H}^\Diamond \left\{ \begin{array}{rcl} \boldsymbol{x}^\Diamond(k+1) & = & \Diamond_{m_x}(\boldsymbol{A}\boldsymbol{x}^\Diamond(k) + \boldsymbol{B}\boldsymbol{u}(k)) \; + \; \varepsilon_x(k) \\ \boldsymbol{y}^\Diamond(k) & = & \Diamond_{m_y}(\boldsymbol{C}\boldsymbol{x}^\Diamond(k) + \boldsymbol{D}\boldsymbol{u}(k) \; + \varepsilon_y(k) \end{array} \right.$$

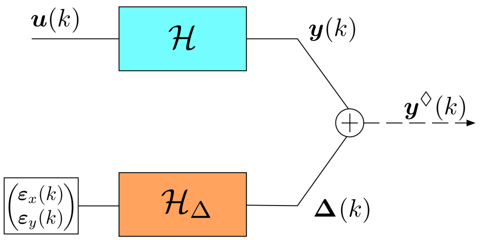
with

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



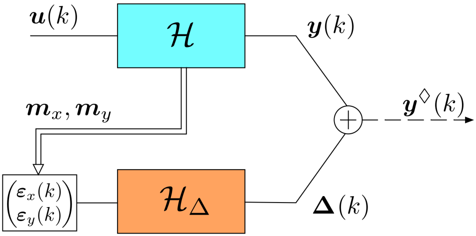


Algorithm



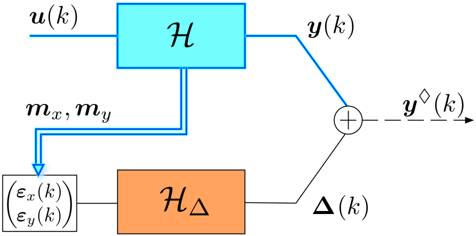
$\boldsymbol{u}(k)$
$\mathcal{H}$
$\boldsymbol{y}(k)$
$\boldsymbol{y}^{\diamondsuit}(k)$
$\begin{pmatrix} \varepsilon_x(k) \\ \varepsilon_y(k) \end{pmatrix}$
$\mathcal{H}_{\Delta}$
$\boldsymbol{\Delta}(k)$

Algorithm



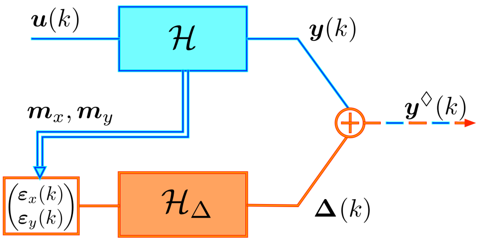
$\boldsymbol{u}(k)$
$\mathcal{H}$
$\boldsymbol{y}(k)$
$\boldsymbol{m}_x, \boldsymbol{m}_y$
$\boldsymbol{y}^{\diamond}(k)$
$\begin{pmatrix} \varepsilon_x(k) \\ \varepsilon_y(k) \end{pmatrix}$
$\mathcal{H}_{\Delta}$
$\boldsymbol{\Delta}(k)$

Algorithm



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

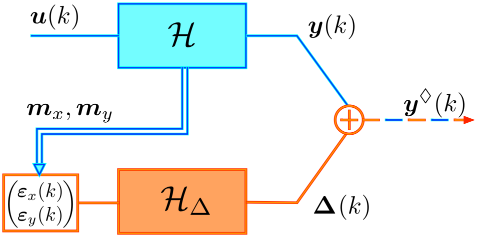
Algorithm



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 $\boldsymbol{m}_{\zeta}^{\diamondsuit}$

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.

Example (continued)

Our random 5^{th} order Butterworth:
5 states, 1 input, 1 output.

- $\bar{u} = 1$
- $\rho(\boldsymbol{A}) = 1 - 1.44 \times 10^{-4}$
- wordlengths set to 7 bits

	states					output
	$\boldsymbol{x}_1(k)$	$\boldsymbol{x}_2(k)$	$\boldsymbol{x}_3(k)$	$\boldsymbol{x}_4(k)$	$\boldsymbol{x}_5(k)$	$\boldsymbol{y}(k)$
Matlab	8	9	9	9	8	7
Our approach	11	12	12	12	11	11

Table: Resulting MSB positions

Conclusion and Perspectives

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 Fixed-Point Formats that guarantee no overflow

Perspectives:

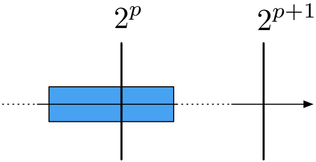
- Solve the off-by-one problem for the MSBs
- Accuracy of the algorithms for the design of IIR filters
  - $\rightsquigarrow$ develop approaches to take the quantization error into account
- Formalize proofs in a Formal Proof Checker

Thank you!

Off-by-one problem

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

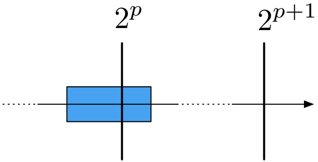
Problem: interval $\mathfrak{m}$ contains a power of 2.



Off-by-one problem

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

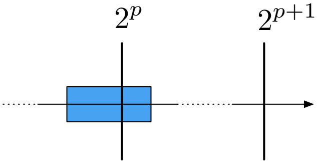
Problem: interval $\mathfrak{m}$ contains a power of 2.
Technique: Ziv's strategy to reduce interval



Off-by-one problem

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

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



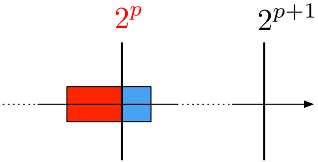
Dilemma:

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

Off-by-one problem

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

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?

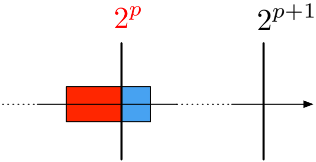
Possible approach:

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

Off-by-one problem

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

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



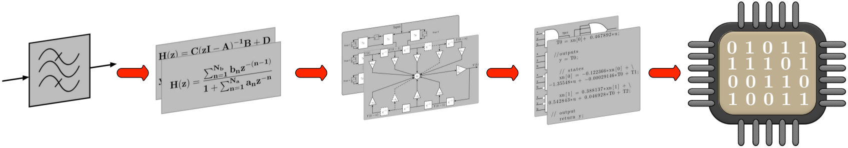
Dilemma:

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

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? LLL?



- 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

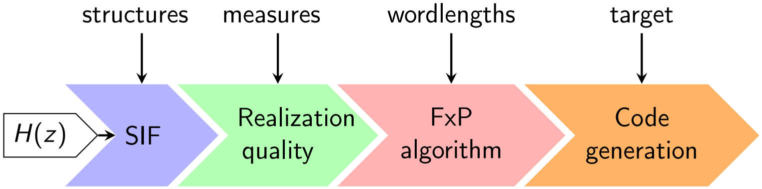


Figure: Automatic filter generator flow.

Stage 1: analytical filter realization representation (SIF)

Stage 2: filter quality measures

Stage 3: fixed-point algorithm (rigorous approach, computational errors are taken into account, no onverflows)

Stage 4: Fixed-Point Code Generator

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 example

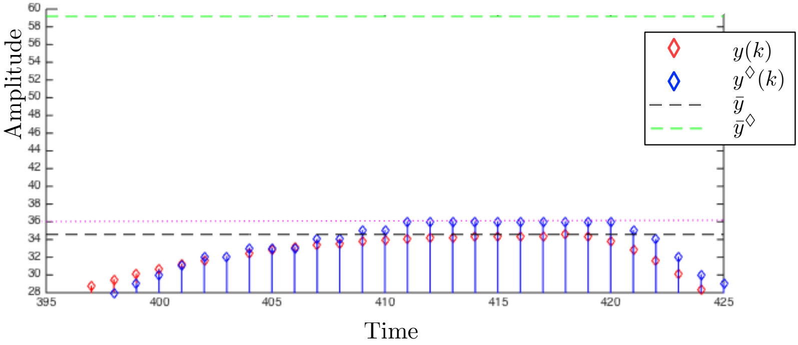


Figure: The exact and quantized outputs of the example.
Quantized output does not pass over to the next binade.

Numerical example

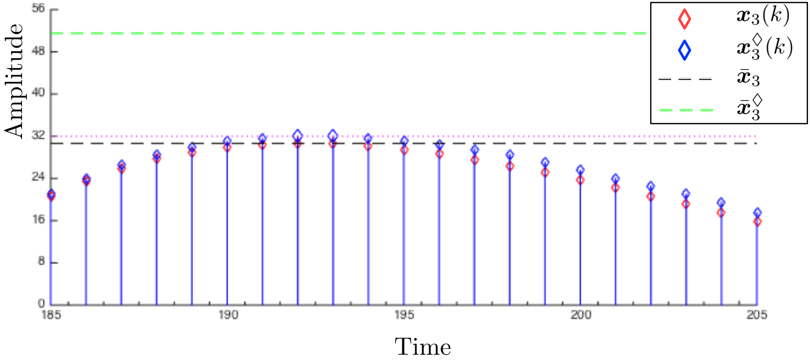


Figure: The exact and quantized third state of the example.

Quantized state passes over to the next binade.