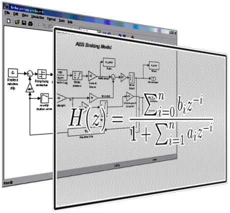
Optimal word-length allocation for the fixed-point implementation of linear filters and controllers

Thibault Hilaire, Hacène Ouzia and Benoit Lopez
Sorbonne Université, France
Inria, Paris Saclay

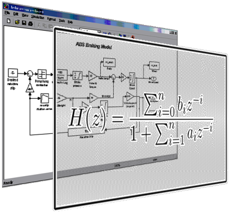
ARITH26 2019, June 2019, Kyoto

Context



Filters/Controllers
Algorithms

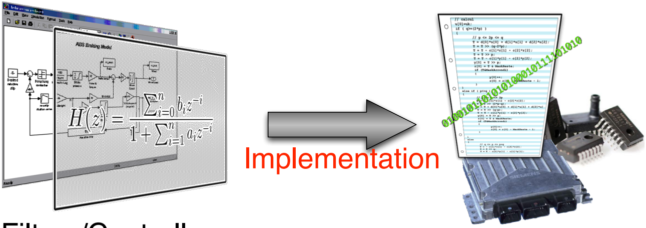
Context



Filters/Controllers
Algorithms

Embedded
Hardware

Context

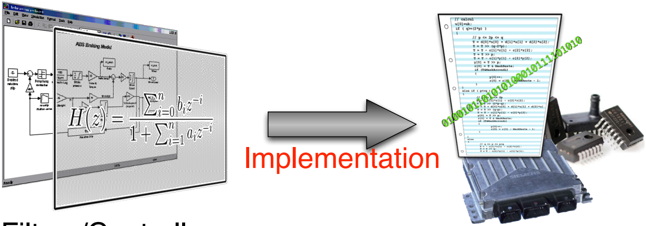


Filters/Controllers Algorithms

Embedded Hardware

☞*implementation*: transformation of the mathematical object into finite precision operations to be performed on a specific target

Context



Filters/Controllers Algorithms

Embedded Hardware

☞ *implementation*: transformation of the mathematical object into finite precision operations to be performed on a specific target

Specially for Fixed-Point arithmetic, the implementation process is:

- time-consuming and error-prone
- provides no guaranty on the the errors
- potentially non-*optimal*

We need a code generator!

Word-length optimization for FPGA/ASIC

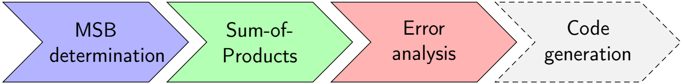
We will here focus on

- State-Space systems

Word-length optimization for FPGA/ASIC

We will here focus on

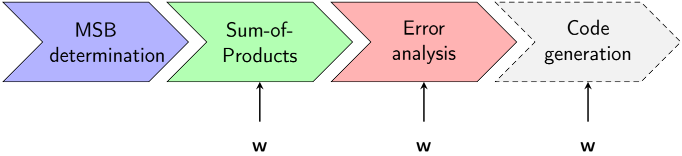
- State-Space systems
- the first parts of the flow, except the code generation



Word-length optimization for FPGA/ASIC

We will here focus on

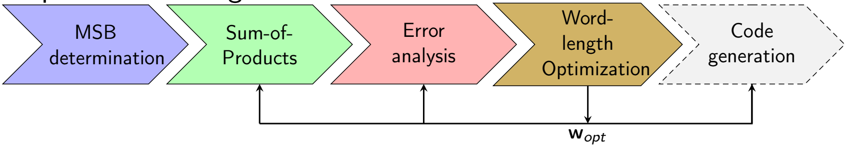
- State-Space systems
- the first parts of the flow, except the code generation
- for FPGAs or ASICs (multiple word-length paradigm)
  $\mathbf{w}$: word-lengths



Word-length optimization for FPGA/ASIC

We will here focus on

- State-Space systems
- the first parts of the flow, except the code generation
- for FPGAs or ASICs (multiple word-length paradigm)
  w: word-lengths
- optimal word-length allocation



We also want it to be reliable:
☞classical Signal Processing approach models errors as *noises* and perform statistical error analysis
☞we want to use *worst case* analysis (rigorous bounds)

Outline

State-Space

We consider Linear Time Invariant filters (or controllers) expressed with State-Space systems:

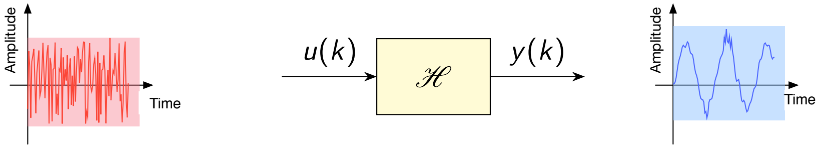
$$\mathscr{H}\left\{\begin{array}{rcl}\mathbf{x}(k+1) & = & \mathbf{A}\mathbf{x}(k)+\mathbf{B}\mathbf{u}(k)\\ \mathbf{y}(k) & = & \mathbf{C}\mathbf{x}(k)+\mathbf{D}\mathbf{u}(k)\end{array}\right.$$

where

- $\mathbf{u}(k)$ and $\mathbf{y}(k)$ are the input and output vector *at time* k ;
- $\mathbf{x}(k)$ is the state vector ;
- $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are matrices defining the system.

Worst-Case Peak Gain

Input $u(k)$ Stable filter Output $y(k)$



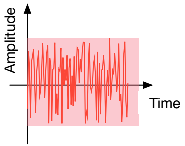
[1] Volkova, H., and Lauter, "Reliable evaluation of the Worst-Case Peak Gain matrix in multiple precision," in 22nd IEEE Symposium on Computer Arithmetic, 2015

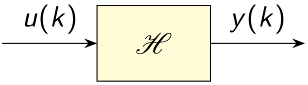
Worst-Case Peak Gain

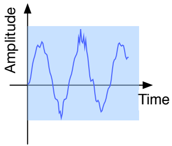
Input $u(k)$

Stable filter

Output $y(k)$





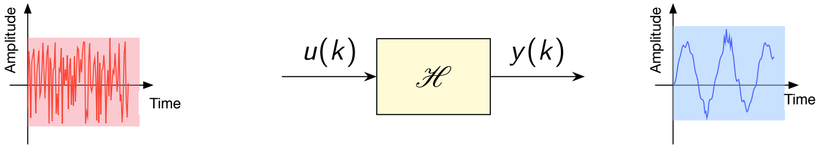


$$\forall k, |\mathbf{u}(k)| \leqslant \bar{\mathbf{u}}$$

[1] Volkova, H., and Lauter, "Reliable evaluation of the Worst-Case Peak Gain matrix in multiple precision," in 22nd IEEE Symposium on Computer Arithmetic, 2015

Worst-Case Peak Gain

Input $u(k)$ Stable filter Output $y(k)$



$\forall k, |\mathbf{u}(k)| \leqslant \bar{\mathbf{u}}$ $\forall k, |\mathbf{y}(k)| \leqslant \langle\!\langle \mathscr{H} \rangle\!\rangle \bar{\mathbf{u}}$

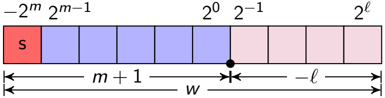
Worst-Case Peak Gain: $\langle\!\langle \mathscr{H} \rangle\!\rangle = |\mathbf{D}| + \sum_{k=0}^{\infty} \left|\mathbf{C}\mathbf{A}^k\mathbf{B}\right|$

The WCPG matrix can be computed at any arbitrary precision[1].

[1] Volkova, H., and Lauter, "Reliable evaluation of the Worst-Case Peak Gain matrix in multiple precision," in 22nd IEEE Symposium on Computer Arithmetic, 2015

Outline

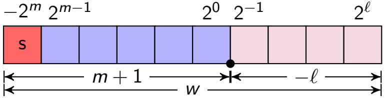
Fixed-Point Arithmetic



$$x = M \cdot 2^\ell$$

$$M : \text{mantissa (two's complement)} \in [-2^{w-1}; 2^{w-1} - 1]$$

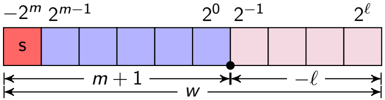
Fixed-Point Arithmetic



$$w = m - \ell + 1$$

w	wordlength
m	Most Significant Bit
ℓ	Least Significant Bit

Fixed-Point Arithmetic



$$w = m - \ell + 1$$

w	wordlength
m	Most Significant Bit
ℓ	Least Significant Bit

The MSB/LSB (or MSB/word-length) must be chosen carefully:

- we choose MSB such that no overflow occurs:

$$\forall k, x(k) \in [-2^m; 2^m - 2^\ell]$$

- we choose LSB such that we achieve a certain accuracy

Determining the MSB position

We consider the vector of our variables $\zeta(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{y}(k) \end{pmatrix}$.

We want to determine their MSB **m** such that

$$\forall k, \quad \zeta_i(k) \in [-2^{\mathbf{m}_i}, 2^{\mathbf{m}_i}(1 - 2^{1-\mathbf{w}_i})].$$

Determining the MSB position

We consider the vector of our variables $\zeta(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{y}(k) \end{pmatrix}$.

We want to determine their MSB $\mathbf{m}$ such that

$$\forall k, \quad \zeta_i(k) \in [-2^{\mathbf{m}_i}, 2^{\mathbf{m}_i}(1 - 2^{1-\mathbf{w}_i})].$$

Our system is rewritten as

$$\mathscr{H}_\zeta \left\{ \begin{array}{rcl} \mathbf{x}(k+1) & = & \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \zeta(k) & = & \mathbf{M}_1\mathbf{x}(k) + \mathbf{M}_2\mathbf{u}(k), \end{array} \right.$$

Determining the MSB position

We consider the vector of our variables $\zeta(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{y}(k) \end{pmatrix}$.

We want to determine their MSB $\mathbf{m}$ such that

$$\forall k, \quad \zeta_i(k) \in [-2^{\mathbf{m}_i}, 2^{\mathbf{m}_i}(1 - 2^{1-\mathbf{w}_i})].$$

Our system is rewritten as

$$\mathscr{H}_\zeta \left\{ \begin{array}{rcl} \mathbf{x}(k+1) & = & \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \zeta(k) & = & \mathbf{M}_1\mathbf{x}(k) + \mathbf{M}_2\mathbf{u}(k), \end{array} \right.$$

And then the least MSB is obtained with

$$\mathbf{m}(\mathbf{w}) = \left\lceil \log_2\left(\overline{\zeta}\right) - \log_2\left(\mathbb{1} - 2^{\mathbb{1}-\mathbf{w}}\right) \right\rceil, \quad \overline{\zeta} = \langle\!\langle \mathscr{H}_\zeta \rangle\!\rangle\, \overline{\mathbf{u}}$$

Determining the MSB position

We consider the vector of our variables $\zeta(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{y}(k) \end{pmatrix}$.

We want to determine their MSB $\mathbf{m}$ such that

$$\forall k, \quad \zeta_i(k) \in [-2^{\mathbf{m}_i}, 2^{\mathbf{m}_i}(1 - 2^{1-\mathbf{w}_i})].$$

Our system is rewritten as

$$\mathscr{H}_\zeta \left\{ \begin{array}{rcl} \mathbf{x}(k+1) & = & \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \zeta(k) & = & \mathbf{M}_1\mathbf{x}(k) + \mathbf{M}_2\mathbf{u}(k), \end{array} \right.$$

And then the least MSB is obtained with

$$\mathbf{m}(\mathbf{w}) = \left\lceil \log_2 \left(\overline{\zeta}\right) \right\rceil + \delta(\mathbf{w}),$$

Determining the MSB position

We consider the vector of our variables $\zeta(k) = \begin{pmatrix} \mathbf{x}(k) \\ \mathbf{y}(k) \end{pmatrix}$.

We want to determine their MSB $\mathbf{m}$ such that

$$\forall k, \quad \zeta_i(k) \in [-2^{\mathbf{m}_i}, 2^{\mathbf{m}_i}(1 - 2^{1-\mathbf{w}_i})].$$

Our system is rewritten as

$$\mathscr{H}_\zeta \left\{ \begin{array}{rcl} \mathbf{x}(k+1) & = & \mathbf{A}\mathbf{x}(k) + \mathbf{B}\mathbf{u}(k) \\ \zeta(k) & = & \mathbf{M}_1\mathbf{x}(k) + \mathbf{M}_2\mathbf{u}(k), \end{array} \right.$$

And then the least MSB is obtained with

$$\mathbf{m}(\mathbf{w}) = \left\lceil \log_2 \left(\overline{\zeta}\right) \right\rceil + \delta(\mathbf{w}),$$

$$\delta_i(\mathbf{w}) = \begin{cases} 0 & \text{if } \mathbf{w}_i \geqslant \widetilde{\mathbf{w}}_i \\ 1 & \text{if } \mathbf{w}_i < \widetilde{\mathbf{w}}_i \end{cases}, \quad \widetilde{\mathbf{w}} \triangleq \mathbb{1} + \left\lceil \log_2 \left(\overline{\zeta}\right) \right\rceil - \left\lfloor \log_2 \left(2^{\left\lceil \log_2 \left(\overline{\zeta}\right) \right\rceil} - \overline{\zeta} \right) \right\rfloor$$

$\widetilde{\mathbf{w}}$ can be seen as a threshold. How many bits such that $\zeta_i \notin [2^{\mathbf{m}_i}(1-2^{\mathbf{w}_i}); 2^{\mathbf{m}_i}]$

Sum-of-Products by real Constants

In the State-Space evaluation, the Sum of Products by real Constants is the basic brick operation $r = \sum_{i=1}^{N} c_i v_i$

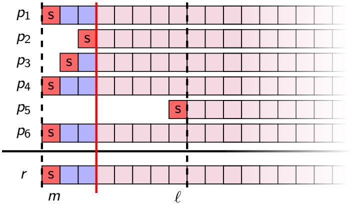
We want to compute r with last bit accuracy with format (m, ℓ)

[2] Volkova, Istoan, De Dinechin and H. "Towards hardware IIR filters computing just right: Direct form I case study," IEEE Transactions on Computers, 2019

Sum-of-Products by real Constants

In the State-Space evaluation, the Sum of Products by real Constants is the basic brick operation $r = \sum_{i=1}^{N} c_i v_i$

We want to compute r with last bit accuracy with format (m, ℓ)

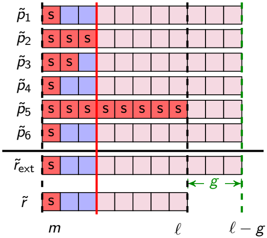


[2] Volkova, Istoan, De Dinechin and H. "Towards hardware IIR filters computing just right: Direct form I case study," IEEE Transactions on Computers, 2019

Sum-of-Products by real Constants

In the State-Space evaluation, the Sum of Products by real Constants is the basic brick operation $r = \sum_{i=1}^{N} c_i v_i$

We want to compute r with last bit accuracy with format (m, ℓ)



It can be done using g extra guard bits.

If g is large enough[2], the error $|r - \tilde{r}|$ is then bounded by 2^ℓ

☞Can be done using Table-based Constant Multiplier for FPGAs

[2] Volkova, Istoan, De Dinechin and H. "Towards hardware IIR filters computing just right: Direct form I case study," IEEE Transactions on Computers, 2019

Outline

Error analysis – 1

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

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

Error analysis – 1

The actually implemented filter $\mathscr{H}^\star$ is:

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

where $\varepsilon(k) = \begin{pmatrix} \varepsilon_x(k) \\ \varepsilon_y(k) \end{pmatrix}$ collects the errors due to the Sum-of-Products.

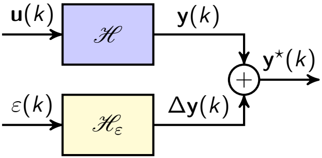
Error analysis – 1

The actually implemented filter $\mathscr{H}^\star$ is:

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

where $\varepsilon(k) = \begin{pmatrix} \varepsilon_x(k) \\ \varepsilon_y(k) \end{pmatrix}$ collects the errors due to the Sum-of-Products.

It can be shown that the implemented system $\mathscr{H}^\star$ can be seen as



with $\mathscr{H}_\varepsilon$ the *error filter* (state-space).

Error analysis – 2

Using last-bit accuracy Sum-of-Products, then the error $\varepsilon(k)$ is bounded by

$$\bar{\varepsilon} = 2^{\mathbf{m}-\mathbf{w}+\mathbb{1}}$$

Error analysis – 2

Using last-bit accuracy Sum-of-Products, then the error $\varepsilon(k)$ is bounded by

$$\overline{\varepsilon} = 2^{\mathbf{m}-\mathbf{w}+\mathbb{1}}$$

So, the output error $\Delta\mathbf{y}(k)$ is bounded by

$$\Delta\overline{\mathbf{y}} = \langle\!\langle \mathscr{H}_\varepsilon \rangle\!\rangle \, \overline{\varepsilon}$$

And finally

$$\Delta\overline{\mathbf{y}} = \langle\!\langle \mathscr{H}_\varepsilon \rangle\!\rangle \, 2^{\left\lceil \log_2\left(\langle\!\langle \mathscr{H}_\zeta \rangle\!\rangle \overline{\mathbf{u}} \right) \right\rceil + \mathbb{1} - \mathbf{w} + \delta(w)}$$

Error analysis – 2

Using last-bit accuracy Sum-of-Products, then the error $\varepsilon(k)$ is bounded by

$$\overline{\varepsilon} = 2^{\mathbf{m}-\mathbf{w}+\mathbb{1}}$$

So, the output error $\Delta\mathbf{y}(k)$ is bounded by

$$\Delta\overline{\mathbf{y}} = \langle\!\langle \mathscr{H}_\varepsilon \rangle\!\rangle\, \overline{\varepsilon}$$

And finally

$$\Delta\overline{\mathbf{y}} = \langle\!\langle \mathscr{H}_\varepsilon \rangle\!\rangle\, 2^{\left\lceil \log_2\left(\langle\!\langle \mathscr{H}_\zeta \rangle\!\rangle \overline{\mathbf{u}}\right)\right\rceil + \mathbb{1} - \mathbf{w} + \delta(w)}$$

The output error bound depends on the word-lengths $\mathbf{w}$.

Word-length allocation problem – 1

We want to minimize the total word-length involved *while* guarantying a certain accuracy ϵ.

Word-length allocation problem – 1

We want to minimize the total word-length involved *while* guarantying a certain accuracy ϵ.
It leads to the following optimization problem

$$\begin{aligned} &\mathbf{w}_{opt} = \arg\min \textstyle\sum_i \mathbf{w}_i \\ &\text{subject to} \\ &\quad \Delta\bar{\mathbf{y}}_j \leqslant \epsilon_j \end{aligned}$$

Word-length allocation problem – 1

We want to minimize the total word-length involved *while* guarantying a certain accuracy ϵ.
It leads to the following optimization problem

$$\begin{aligned} &\mathbf{w}_{opt} = \arg\min \textstyle\sum_i \mathbf{w}_i \\ &\text{subject to} \\ &\quad \textstyle\sum_j \mathbf{E}_{ij} 2^{-\mathbf{w}_j + \delta_j} \leqslant \epsilon_j \end{aligned}$$

with $\mathbf{E}_{ij} \triangleq \langle\!\langle \mathscr{H}_\varepsilon \rangle\!\rangle_{ij} \, 2^{\left\lceil \log_2 \left(\langle\!\langle \mathscr{H}_\zeta \rangle\!\rangle \overline{\mathbf{u}} \right)_j \right\rceil}$

Word-length allocation problem – 1

We want to minimize the total word-length involved *while* guarantying a certain accuracy ϵ.
It leads to the following optimization problem

$$\mathbf{w}_{opt} = \arg\min \textstyle\sum_i \mathbf{w}_i$$
subject to
$$\textstyle\sum_j \mathbf{E}_{ij} 2^{-\mathbf{w}_j + \delta_j} \leqslant \epsilon_j$$

Recall that $\delta_j = \begin{cases} 0 & \text{if } w_j \geqslant \tilde{w}_j \\ 1 & \text{if } w_j < \tilde{w}_j \end{cases}$

And this condition is equivalent to (w_j and $\tilde{w}_j \in [2, u]$)

$$(2 - \tilde{w}_j)\,\delta_j + 1 \leq w_j - \tilde{w}_j + 1 \leq (1 - \delta_j)\,(u - \tilde{w}_j + 1)\,, \quad \delta_j \in \{0, 1\}$$

Word-length allocation problem – 1

We want to minimize the total word-length involved *while* guarantying a certain accuracy ϵ.
It leads to the following optimization problem

$$
\begin{aligned}
&\mathbf{w}_{opt} = \arg\min \textstyle\sum_i \mathbf{w}_i \\
&\text{subject to} \\
&\quad \textstyle\sum_j \mathbf{E}_{ij} 2^{-\mathbf{w}_j + \delta_j} \leqslant \epsilon_j \\
&\quad (2 - \widetilde{\mathbf{w}}_j)\, \delta_j + 1 \leqslant \mathbf{w}_j - \widetilde{\mathbf{w}}_j + 1 \\
&\quad \mathbf{w}_j - \widetilde{\mathbf{w}}_j + 1 \leqslant (1 - \delta_j)\, (\mathbf{u}_j - \widetilde{\mathbf{w}}_j + 1) \\
&\quad 2 \leqslant \mathbf{w}_j \leqslant \mathbf{u}_j \\
&\quad \mathbf{w}_j \in \mathbb{Z},\ \ \delta_j \in \{0, 1\}
\end{aligned}
$$

Word-length allocation problem – 2

☞ Separable convex non-linear integer optimization problem
Generaly solved using branch-and-bound and outer approximation methods.
Existing solvers (like Bonmin, Artylis-Knitro, etc.) can be used

Word-length allocation problem – 2

☞ Separable convex non-linear integer optimization problem
Generaly solved using branch-and-bound and outer approximation methods.
Existing solvers (like Bonmin, Artylis-Knitro, etc.) can be used

Two sub-optimal problem can be defined and solved:

- Uniform word-lengths

$$w_{uni} = \max\left(\left\lceil \log_2\left(\mathbf{E}\mathbb{1}\right) - \log_2(\epsilon)\right\rceil\right) + 1$$

Word-length allocation problem – 2

☞ Separable convex non-linear integer optimization problem
Generaly solved using branch-and-bound and outer approximation methods.
Existing solvers (like Bonmin, Artylis-Knitro, etc.) can be used

Two sub-optimal problem can be defined and solved:

- Uniform word-lengths

$$w_{uni} = \max\left(\left\lceil \log_2\left(\mathbf{E}\mathbb{1}\right) - \log_2(\epsilon)\right\rceil\right) +1$$

- Equitably distributed budget error
Constraints $\sum_{j=1}^{N} \mathbf{E}_{ij} 2^{-\mathbf{w}_j+\delta_j} \leqslant \epsilon_i$ are transformed in N stricter but simpler constraints $\mathbf{E}_{ij} 2^{-\mathbf{w}_j+\delta_j} \leqslant \frac{\epsilon_i}{N}$

Word-length allocation problem – 2

☞ Separable convex non-linear integer optimization problem
Generaly solved using branch-and-bound and outer approximation methods.
Existing solvers (like Bonmin, Artylis-Knitro, etc.) can be used

Two sub-optimal problem can be defined and solved:

- Uniform word-lengths
$$w_{uni} = \max\left(\left\lceil \log_2\left(\mathbf{E}\mathbb{1}\right) - \log_2(\epsilon)\right\rceil\right) + 1$$
- Equitably distributed budget error
Constraints $\sum_{j=1}^{N} \mathbf{E}_{ij} 2^{-\mathbf{w}_j+\delta_j} \leqslant \epsilon_i$ are transformed in N stricter but simpler constraints $\mathbf{E}_{ij} 2^{-\mathbf{w}_j+\delta_j} \leqslant \frac{\epsilon_i}{N}$

☞ Similar results, if the cost function is the total number of bits involved in the computations

Example – 1

A 10^{th} order active controller of longitudinal oscillation[3].
Designed to remove the unpleasant oscillations of the vehicle (acting on the engine torque).

☞ $\overline{\mathbf{u}} = 10$ and $\epsilon = 2^{-6}$

	oscillation controller											
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	y	$f(\mathbf{w})$
$\mathbf{m}$	9	8	9	9	8	8	8	6	6	2	4	
$\widetilde{\mathbf{w}}$	3	3	3	3	3	3	3	4	8	3	3	
optimal	17	15	17	17	15	15	15	11	11	3	13	**149**
uniform	16	16	16	16	16	16	16	16	16	16	16	**176**
eq. distributed	17	15	18	17	15	15	15	12	12	4	14	**154**

$\overline{\zeta}_9 \approx 63.412$

[3] Lefebvre, Chevrel, and Richard, "An H_∞ based control design methodology dedicated to the active control of longitudinal oscillations," IEEE Trans. on Control Systems Technology, 2003

Example – 2

A random stable 4th order State-Space, 5 inputs, 7 outputs

	random controller											
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$	$\mathbf{y}_4$	$\mathbf{y}_5$	$\mathbf{y}_6$	$\mathbf{y}_7$	$f(\mathbf{w})$
$\mathbf{m}$	6	6	5	6	16	15	16	16	16	15	16	
$\widetilde{\mathbf{w}}$	3	4	3	6	4	3	6	3	4	3	6	
optimal	13	11	14	12	4	3	5	4	4	3	5	**78**
uniform	31	31	31	31	31	31	31	31	31	31	31	**341**
eq. distributed	33	32	34	32	26	25	26	26	26	25	26	**311**

Example – 2

A random stable 4^{th} order State-Space, 5 inputs, 7 outputs

	random controller											
	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$	$\mathbf{x}_4$	$\mathbf{y}_1$	$\mathbf{y}_2$	$\mathbf{y}_3$	$\mathbf{y}_4$	$\mathbf{y}_5$	$\mathbf{y}_6$	$\mathbf{y}_7$	$f(\mathbf{w})$
$\mathbf{m}$	6	6	5	6	16	15	16	16	16	15	16	
$\widetilde{\mathbf{w}}$	3	4	3	6	4	3	6	3	4	3	6	
optimal	13	11	14	12	4	3	5	4	4	3	5	**78**
uniform	31	31	31	31	31	31	31	31	31	31	31	**341**
eq. distributed	33	32	34	32	26	25	26	26	26	25	26	**311**

For optimal: $\mathbf{w}_3 < \widetilde{\mathbf{w}}_3$ and $\mathbf{w}_7 < \widetilde{\mathbf{w}}_7$ (so $\delta_3 = \delta_7 = 0$)

Conclusion

- Reliable Fixed-Point implementation of State-Space systems
  - MSB determination
  - Last-bit accuracy sum-of-Products
- Error analysis
- Word-length allocation problem, solved with three heuristics

Conclusion

- Reliable Fixed-Point implementation of State-Space systems
  - MSB determination
  - Last-bit accuracy sum-of-Products
- Error analysis
- Word-length allocation problem, solved with three heuristics

Perspectives:

- Now consider the code generation, using dedicated tools (FloPoCo, etc.)
- Allow a more realistic cost function
- Extend this work to full class of linear filters/controllers, in order to compare various algorithms and implementations

Thank you
Any questions ?