Thibault
HILAIRE
thibault.hilaire@irisa.fr

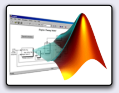
Bit Accurate Roundoff Noise Analysis of Fixed-Point Linear Controllers

IEEE Multi-Conference on Systems and Control
San Antonio, Texas
4 September 2008

CAIRN project
IRISA/INRIA
France

Context

filter/controller



$$H(z) = \frac{0.004708z^6 - 0.0251}{z^6 - 5.655}$$

$$\frac{0251z^5 + 0.05844z^4 - 0.07608}{5.653z^5 + 13.38z^4 - 16.98z^3 +}$$

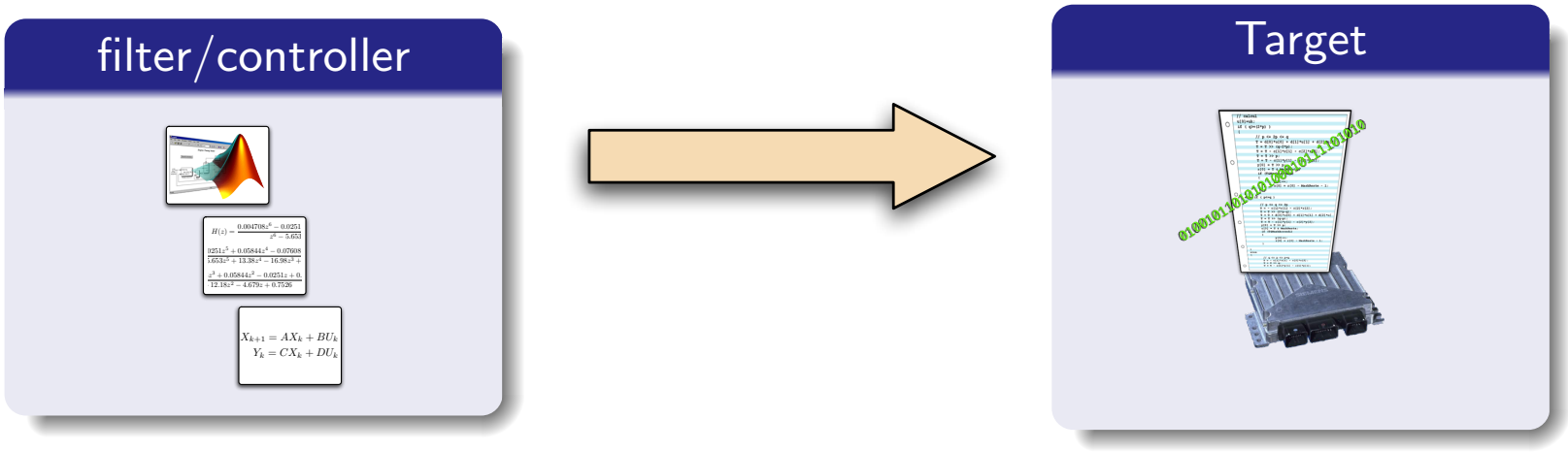
$$\frac{z^3 + 0.05844z^2 - 0.0251z + 0.}{12.18z^2 - 4.679z + 0.7526}$$

$$X_{k+1} = AX_k + BU_k$$
$$Y_k = CX_k + DU_k$$

Target

- Implementation of Linear Time Invariant controllers/filters
- Finite Word Length context (fixed-point)

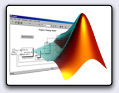
Context



- Implementation of Linear Time Invariant controllers/filters
- Finite Word Length context (fixed-point)

Context

filter/controller



$$H(z) = \frac{0.004708z^6 - 0.0251}{z^6 - 5.655}$$

$$\frac{9251z^5 + 0.05844z^4 - 0.07608}{5.653z^5 + 13.38z^4 - 16.98z^3 + z^3 + 0.05844z^2 - 0.0251z + 0.}$$

$$\frac{}{12.18z^2 - 4.679z + 0.7526}$$

$$X_{k+1} = AX_k + BU_k$$
$$Y_k = CX_k + DU_k$$

Target

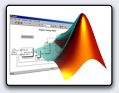
- Implementation of Linear Time Invariant controllers/filters
- Finite Word Length context (fixed-point)

Motivation

- Analysis (accurately) the roundoff noise errors in the implementation
- Compare various realizations and find an *optimal* one

Context

filter/controller



$$H(z) = \frac{0.004708z^6 - 0.0251}{z^6 - 5.655}$$

$$\frac{.9251z^5 + 0.05844z^4 - 0.07608}{5.653z^5 + 13.38z^4 - 16.98z^3 +}$$

$$\frac{z^3 + 0.05844z^2 - 0.0251z + 0.}{12.18z^2 - 4.679z + 0.7526}$$

$$X_{k+1} = AX_k + BU_k$$
$$Y_k = CX_k + DU_k$$

Target

- Implementation of Linear Time Invariant controllers/filters
- Finite Word Length context (fixed-point)

The roundoff will depend on

- the algorithmic relation to compute the output(s) from the input(s)
- the way the computations are implemented (wordlength, roundoff, etc.)

Outline

1. Implicit state-space framework
2. Roundoff noise analysis
3. Fixed-point implementation schemes
4. Optimal design
5. Conclusion

1

Implicit State-Space Framework

The need of a unifying framework

Various implementation forms have to be taken into consideration:

- shift-realizations
- δ-realizations
- observer-state-feedback
- direct form I or II
- cascade or parallel realizations
- etc...

Specialized Implicit Form (SIF)

Implicit specialized state-space form

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Specialized Implicit Form (SIF)

Implicit specialized state-space form

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

It corresponds to:

1. $J.T_{k+1} = M.X_k + N.U_k$
2. $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
3. $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Intermediate variables computation

Specialized Implicit Form (SIF)

Implicit specialized state-space form

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

It corresponds to:

1. $J.T_{k+1} = M.X_k + N.U_k$
2. $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
3. $Y_k = L.T_{k+1} + R.X_k + S.U_k$

State-space computation

Specialized Implicit Form (SIF)

Implicit specialized state-space form

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

It corresponds to:

1. $J.T_{k+1} = M.X_k + N.U_k$
2. $X_{k+1} = K.T_{k+1} + P.X_k + Q.U_k$
3. $Y_k = L.T_{k+1} + R.X_k + S.U_k$

Output(s) computation

Specialized Implicit Form (SIF)

Implicit specialized state-space form

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

It is equivalent to the system

$$H : z \mapsto C_Z(zI_n - A_Z)B_Z + D_Z$$

with

$$\begin{pmatrix} A_Z & B_Z \\ C_Z & D_Z \end{pmatrix} \triangleq \begin{pmatrix} K \\ L \end{pmatrix} J^{-1} \begin{pmatrix} M & N \end{pmatrix} + \begin{pmatrix} P & Q \\ R & S \end{pmatrix}$$

Intermediate variables

The intermediate variables introduced allow to

- make explicit all the computations done
- show the order of the computations
- express a larger parametrization

Implicit realization

The intermediate variables computation is expressed by

$$J.T_{k+1} = M.X_k + N.U_k$$

with J lower triangular with 1 on diagonal, so

- no need to compute J^{-1}
- an intermediate variable may be computed from another one previously computed (in the same stage)
  $\Rightarrow$ can express realizations like $Y_k = M_1.M_2...M_i U_k$

Intermediate variables

The intermediate variables introduced allow to

- make explicit all the computations done
- show the order of the computations
- express a larger parametrization

Implicit realization

The intermediate variables computation is expressed by

$$J.T_{k+1} = M.X_k + N.U_k$$

with J lower triangular with 1 on diagonal, so

- no need to compute J^{-1}
- an intermediate variable may be computed from another one previously computed (in the same stage)
  $\Rightarrow$ can express realizations like $Y_k = M_1.M_2...M_iU_k$

Example

A realization with the δ-operator is described by :

$$\begin{cases} \delta X_k &= A_\delta X_k + B_\delta U_k \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases} \qquad \delta \triangleq \frac{q-1}{\Delta}$$

It is computed with

$$\begin{cases} T &= A_\delta X_k + B_\delta U_k \\ X_{k+1} &= X_k + \Delta T \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases}$$

and it corresponds to the following implicit state-space :

$$\begin{pmatrix} I & 0 & 0 \\ -\Delta I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Example

A realization with the δ-operator is described by :

$$\begin{cases} \delta X_k &= A_\delta X_k + B_\delta U_k \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases} \qquad \delta \triangleq \frac{q-1}{\Delta}$$

It is computed with

$$\begin{cases} T &= A_\delta X_k + B_\delta U_k \\ X_{k+1} &= X_k + \Delta T \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases}$$

and it corresponds to the following implicit state-space :

$$\begin{pmatrix} I & 0 & 0 \\ -\Delta I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Example

A realization with the δ-operator is described by :

$$\begin{cases} \delta X_k &= A_\delta X_k + B_\delta U_k \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases} \qquad \delta \triangleq \frac{q-1}{\Delta}$$

It is computed with

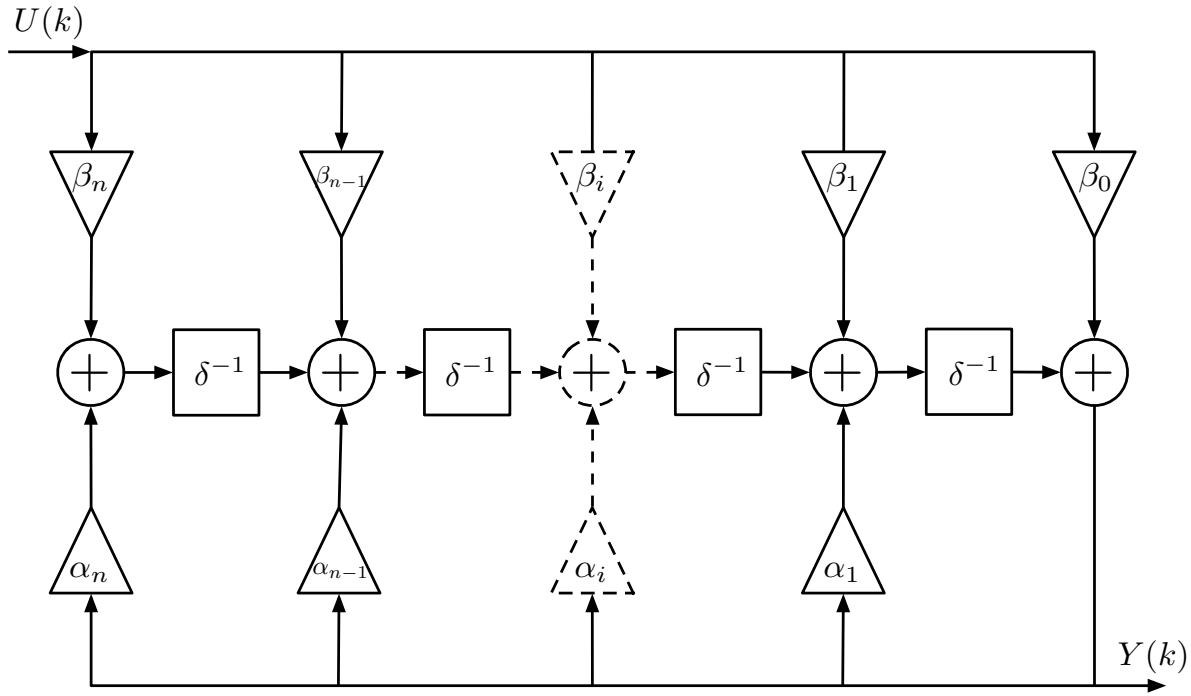
$$\begin{cases} T &= A_\delta X_k + B_\delta U_k \\ X_{k+1} &= X_k + \Delta T \\ Y_k &= C_\delta X_k + D_\delta U_k \end{cases}$$

and it corresponds to the following implicit state-space :

$$\begin{pmatrix} I & 0 & 0 \\ -\Delta I & I & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} T_{k+1} \\ X_{k+1} \\ Y_k \end{pmatrix} = \begin{pmatrix} 0 & A_\delta & B_\delta \\ 0 & I & 0 \\ 0 & C_\delta & D_\delta \end{pmatrix} \begin{pmatrix} T_k \\ X_k \\ U_k \end{pmatrix}$$

Example

One can find the Direct Form II transposed with δ-operator



with

$$A_\delta = \begin{pmatrix} -\alpha_n & 1 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & \vdots \\ \vdots & 0 & \ddots & \ddots & \\ \vdots & \vdots & \ddots & \ddots & 0 \\ -\alpha_1 & 0 & \dots & 0 & 1 \\ -\alpha_0 & 0 & \dots & \dots & 0 \end{pmatrix} \quad B_\delta = \begin{pmatrix} \beta_n \\ \vdots \\ \vdots \\ \vdots \\ \beta_1 \\ \beta_0 \end{pmatrix} \quad C_\delta = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}$$

2

Roundoff Noise Analysis

Fixed point representation

The real numbers are represented by fixed point numbers.

Fixed-point representation

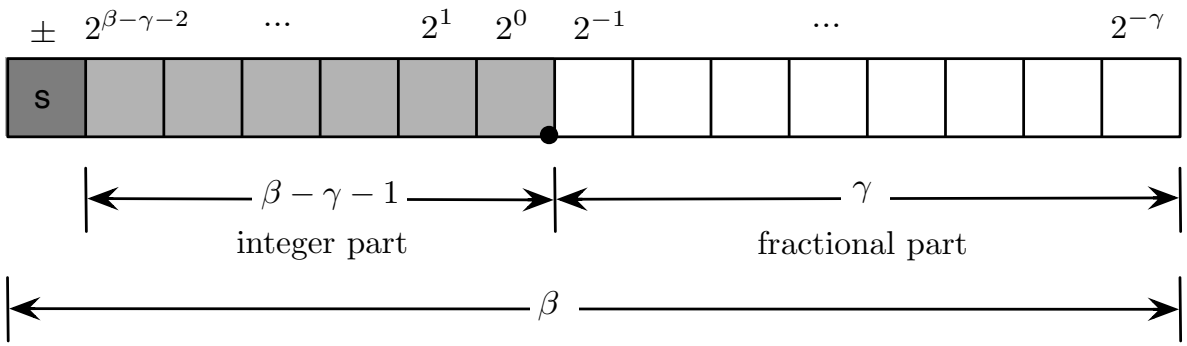
- A number is represented by

  $2^{-\gamma}.N$

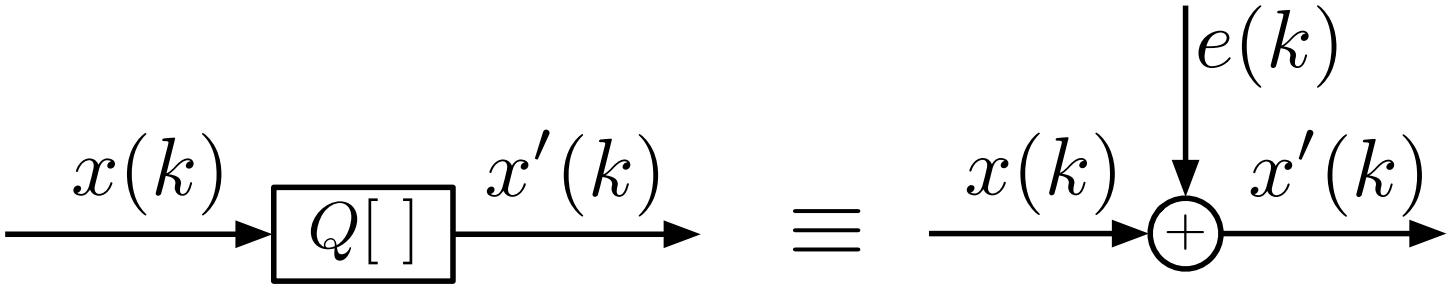
  N : signed integer with β bits

  γ : fixed integer (scaling)

- The quantization step $2^{-\gamma}$ is fixed, the dynamic is fixed (and limited)

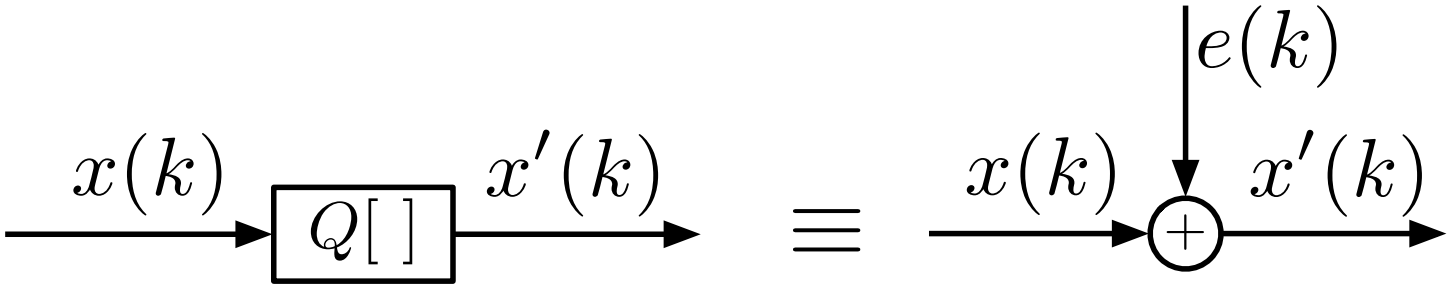


Quantization



To Quantize a signal $x(k)$ is equivalent to add a independent white noise $e(k)$.
Its first and second-order moments characterize it.

Quantization



To Quantize a signal $x(k)$ is equivalent to add a independent white noise $e(k)$.

Its first and second-order moments characterize it.

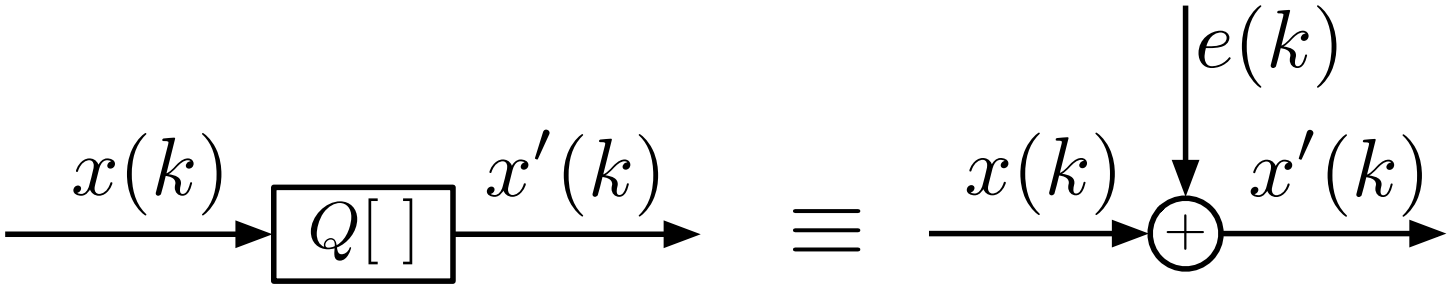
The first (μ) and second (σ, ψ) order moments are defined by

$$\mu_e \triangleq E\{e(k)\}$$

$$\psi_e \triangleq E\left\{(e(k)-\mu_e)(e(k)-\mu_e)^\top\right\}$$

$$\sigma_e^2 \triangleq E\left\{(e(k)-\mu_e)^\top(e(k)-\mu_e)\right\} = tr(\psi_e)$$

Quantization



To Quantize a signal $x(k)$ is equivalent to add a independent white noise $e(k)$.

Its first and second-order moments characterize it.

Right shifting of d bits :

	truncation	best roundoff
μ_e	$2^{-\gamma-1}(1-2^{-d})$	$2^{-\gamma-d-1}$
σ_e^2	$\frac{2^{-2\gamma}}{12}(1-2^{-2d})$	$\frac{2^{-2\gamma}}{12}(1-2^{-2d})$

Roundoff noise analysis

When implemented, the algorithm used is:

$$\begin{cases} J.T_{k+1} & \leftarrow & M.X_k + N.U_k \\ X_{k+1} & \leftarrow & K.T_{k+1} + P.X_k + Q.U_k \\ Y_k & \leftarrow & L.T_{k+1} + R.X_k + S.U_k \end{cases}$$

Quantizations during computations lead to noise addition ξ_k:

$$\xi_k \triangleq \begin{pmatrix} \xi_{T_k} \\ \xi_{X_k} \\ \xi_{Y_k} \end{pmatrix}$$

Roundoff noise analysis

When implemented, the algorithm used is:

$$\begin{cases} J.T_{k+1} & \leftarrow & M.X_k + N.U_k + \xi_{T_k} \\ X_{k+1} & \leftarrow & K.T_{k+1} + P.X_k + Q.U_k + \xi_{X_k} \\ Y_k & \leftarrow & L.T_{k+1} + R.X_k + S.U_k + \xi_{Y_k} \end{cases}$$

Quantizations during computations lead to noise addition ξ_k:

$$\xi_k \triangleq \begin{pmatrix} \xi_{T_k} \\ \xi_{X_k} \\ \xi_{Y_k} \end{pmatrix}$$

Roundoff noise analysis

Output Roundoff noise power

The output roundoff noise power is defined as the power of the noises added on the output(s):

$$P \triangleq \sigma_{\xi'}^2$$

The implemented system is equivalent to

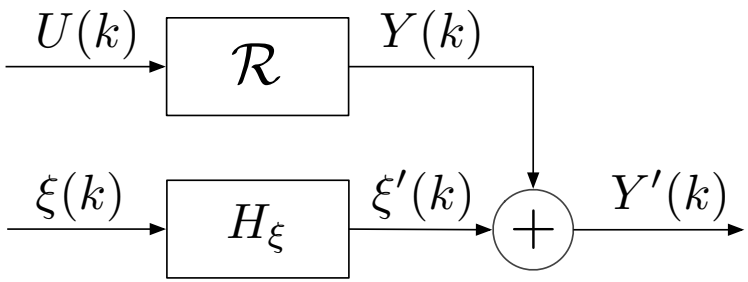
Roundoff noise analysis

Output Roundoff noise power

The output roundoff noise power is defined as the power of the noises added on the output(s):

$$P \triangleq \sigma^2_{\xi'}$$

The implemented system is equivalent to



Roundoff noise analysis

Considering the moments of ξ_k through G, we've got P:

$$P = tr\left(\psi_\xi \left(M_2^\top M_2 + M_1^\top W_o M_1\right)\right) + \mu_{\xi'}^\top \mu_{\xi'}$$

where $\mu_{\xi'} = H_1(0)\mu_\xi = (C_Z(I - A_Z)^{-1}M_1 + M_2)\mu_\xi$

ψ_ξ and μ_ξ depends only on HW/SW considerations, whereas the other terms depends on the realizations.

Roundoff noise analysis

Considering the moments of ξ_k through G, we've got P:

$$P = tr\left(\psi_\xi \left(M_2^\top M_2 + M_1^\top W_o M_1\right)\right) + \mu_{\xi'}^\top \mu_{\xi'}$$

where $\mu_{\xi'} = H_1(0)\mu_\xi = (C_Z(I - A_Z)^{-1}M_1 + M_2)\mu_\xi$
ψ_ξ and μ_ξ depends only on HW/SW considerations, whereas the other terms depends on the realizations.

Fixed point implementation

Fixed point implementation

We supposed the following wordlengths known

- β_Z : coefficient's wordlength
- β_U, β_Y, β_T, β_X : intputs, outputs, intermediate variables and states' wordlength
- β_{ADD} : accumulator's wordlength

γ_U is also known ($\gamma_U = \beta_U - 1 - \left\lfloor \log_2 \overset{\max}{U} \right\rfloor$).

It is also supposed that the accumulations (in each scalar product) are done on the same fixed-point format (no shift between two additions).

Fixed point implementation

We supposed the following wordlengths known

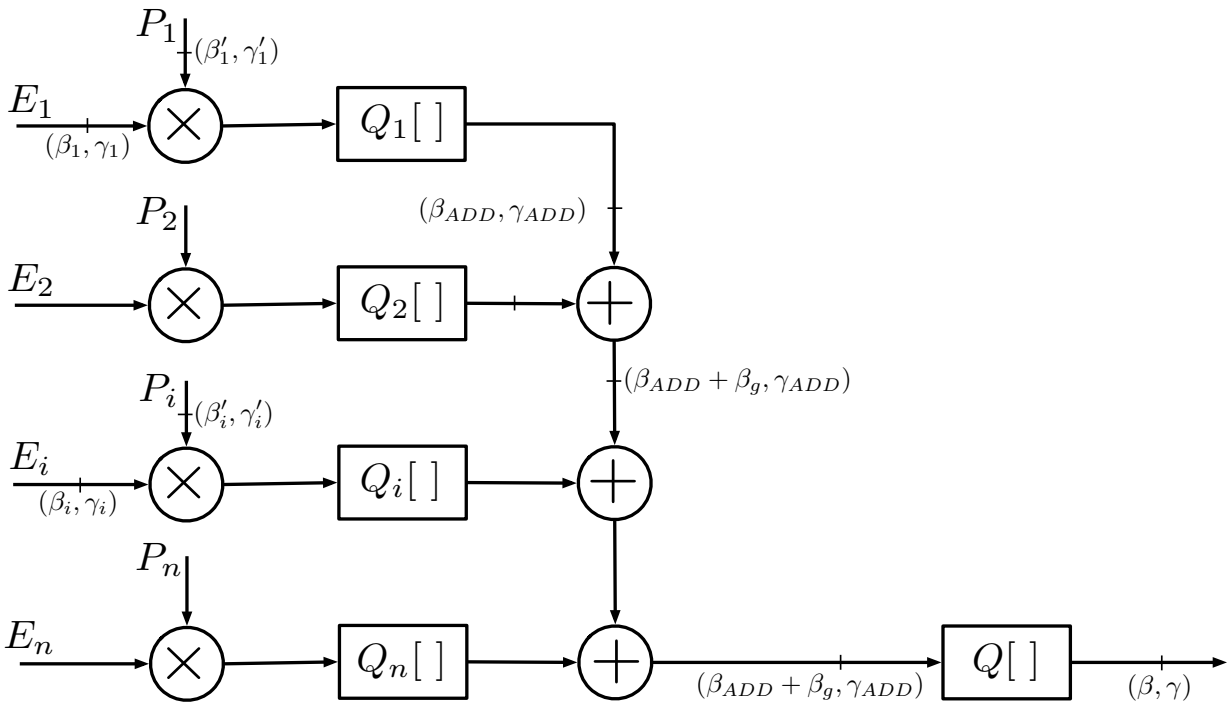
- β_Z : coefficient's wordlength
- β_U, β_Y, β_T, β_X : intputs, outputs, intermediate variables and states' wordlength
- β_{ADD} : accumulator's wordlength

γ_U is also known ($\gamma_U = \beta_U - 1 - \left\lfloor \log_2 \overset{\max}{U} \right\rfloor$).

It is also supposed that the accumulations (in each scalar product) are done on the same fixed-point format (no shift between two additions).

Scalar product

$$S = \sum_{i=1}^{n} P_i E_i$$



Fixed point implementation

Peak value estimation

$$\begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_Y\end{pmatrix} = \begin{pmatrix}\beta_T\\ \beta_X\\ \beta_Y\end{pmatrix} - 2.\mathbb{1}_{l+n+p,1} - \left\lfloor \log_2\left(\|H_{\max}\|_{l_1} \overset{\max}{|U|} \right) \right\rfloor$$

The common fixed-point format of each accumulator γ_{ADD} can be set, in order to represent

- the dynamic of each product without overflow
- the final result without overflow

$$\gamma_{ADD} = \beta_{ADD} - \max_{row}\left(\begin{pmatrix}\beta_T\\ \beta_X\\ \beta_Y\end{pmatrix} - \beta_g - \begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_Y\end{pmatrix}, \alpha \right)$$

where

$$\alpha = \max_{row}\left(\beta_Z - \gamma_Z + \mathbb{1}_{l+n+p,1}\left(\begin{pmatrix}\beta_T\\ \beta_X\\ \beta_U\end{pmatrix} - \begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_U\end{pmatrix} \right)^{\top} \right)$$

Fixed point implementation

Peak value estimation

$$\begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_Y\end{pmatrix} = \begin{pmatrix}\beta_T\\ \beta_X\\ \beta_Y\end{pmatrix} - 2.\mathbb{1}_{l+n+p,1} - \left\lfloor \log_2\left(\|H_{\max}\|_{l_1} \overset{\max}{|U|} \right) \right\rfloor$$

The common fixed-point format of each accumulator γ_{ADD} can be set, in order to represent

- the dynamic of each product without overflow
- the final result without overflow

$$\gamma_{ADD} = \beta_{ADD} - \max_{row}\left(\begin{pmatrix}\beta_T\\ \beta_X\\ \beta_Y\end{pmatrix} - \beta_g - \begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_Y\end{pmatrix}, \alpha \right)$$

where

$$\alpha = \max_{row}\left(\beta_Z - \gamma_Z + \mathbb{1}_{l+n+p,1}\left(\begin{pmatrix}\beta_T\\ \beta_X\\ \beta_U\end{pmatrix} - \begin{pmatrix}\gamma_T\\ \gamma_X\\ \gamma_U\end{pmatrix} \right)^{\top} \right)$$

Fixed point implementation

In order to align the results of the products, 2 computational schemes are possible

Computational scheme

- Roundoff After Multiplication : the result of the product is quantized
- Roundoff Before Multiplication : the coefficient is quantized

$\Rightarrow \gamma_Z$ is then deduced

Since the wordlengths, the binary point positions and the quantizations are know, the moments ψ_ξ and μ_ξ can be (analytically) expressed.

Fixed point implementation

In order to align the results of the products, 2 computational schemes are possible

Computational scheme

- Roundoff After Multiplication : the result of the product is quantized
- Roundoff Before Multiplication : the coefficient is quantized

$\Rightarrow \gamma_Z$ is then deduced

Since the wordlengths, the binary point positions and the quantizations are know, the moments ψ_ξ and μ_ξ can be (analytically) expressed.

Example

$$\begin{cases} X(k+1) &= \begin{pmatrix} 0.58399 & -0.42019 \\ 0.42019 & 0.1638 \end{pmatrix} X(k) + \begin{pmatrix} 0.64635 \\ -0.23982 \end{pmatrix} \\ Y(k) &= \begin{pmatrix} 0.64635 & 0.23982 \end{pmatrix} X(k) + 0.13111 U(k) \end{cases}$$

Example

$$\begin{cases} X(k+1) &= \begin{pmatrix} 0.58399 & -0.42019 \\ 0.42019 & 0.1638 \end{pmatrix} X(k) + \begin{pmatrix} 0.64635 \\ -0.23982 \end{pmatrix} \\ Y(k) &= \begin{pmatrix} 0.64635 & 0.23982 \end{pmatrix} X(k) + 0.13111 U(k) \end{cases}$$

$\beta_U = \beta_X = \beta_Y = 16$

$\beta_Z = 16 I_3$

$$\beta_{ADD} = \begin{pmatrix} 32 \\ 32 \\ 32 \end{pmatrix}$$

$\overset{\max}{U} = 10 \quad \Rightarrow \quad \gamma_U = 11$

$$\gamma_{ADD} = \begin{pmatrix} 26 \\ 27 \\ 26 \end{pmatrix}$$

$$\gamma_Z = \begin{pmatrix} 15 & 16 & 15 \\ 16 & 17 & 17 \\ 15 & 17 & 17 \end{pmatrix}$$

$$\gamma_X = \begin{pmatrix} 11 \\ 11 \end{pmatrix}$$

Roundoff After Multiplication

Intermediate variables

$Acc \leftarrow (xn(1) * 19136);$
$Acc \leftarrow Acc + (xn(2) * -27537) >> 1;$
$Acc \leftarrow Acc + (u * 21179);$
$xnp(1) \leftarrow Acc >> 15;$
$Acc \leftarrow (xn(1) * 27537);$
$Acc \leftarrow Acc + (xn(2) * 21470) >> 1;$
$Acc \leftarrow Acc + (u * -31433) >> 1;$
$xnp(2) \leftarrow Acc >> 16;$

Outputs

$Acc \leftarrow (xn(1) * 21179);$
$Acc \leftarrow Acc + (xn(2) * 31433) >> 2;$
$Acc \leftarrow Acc + (u * 17184) >> 2;$
$y \leftarrow Acc >> 15;$

Permutations

$xn \leftarrow xnp;$

Example

$$\begin{cases} X(k+1) &= \begin{pmatrix} 0.58399 & -0.42019 \\ 0.42019 & 0.1638 \end{pmatrix} X(k) + \begin{pmatrix} 0.64635 \\ -0.23982 \end{pmatrix} \\ Y(k) &= \begin{pmatrix} 0.64635 & 0.23982 \end{pmatrix} X(k) + 0.13111 U(k) \end{cases}$$

$$\beta_U = \beta_X = \beta_Y = 16$$
$$\beta_Z = 16 I_3$$
$$\beta_{ADD} = \begin{pmatrix} 32 \\ 32 \\ 32 \end{pmatrix}$$
$$\overset{\max}{U} = 10 \quad \Rightarrow \quad \gamma_U = 11$$
$$\gamma_{ADD} = \begin{pmatrix} 26 \\ 27 \\ 26 \end{pmatrix}$$
$$\gamma_Z = \begin{pmatrix} 15 & 16 & 15 \\ 16 & 17 & 17 \\ 15 & 17 & 17 \end{pmatrix}$$
$$\gamma_X = \begin{pmatrix} 11 \\ 11 \end{pmatrix}$$

Roundoff After Multiplication

Intermediate variables

$Acc \leftarrow (xn(1) * 19136);$
$Acc \leftarrow Acc + (xn(2) * -27537) >> 1;$
$Acc \leftarrow Acc + (u * 21179);$
$xnp(1) \leftarrow Acc >> 15;$
$Acc \leftarrow (xn(1) * 27537);$
$Acc \leftarrow Acc + (xn(2) * 21470) >> 1;$
$Acc \leftarrow Acc + (u * -31433) >> 1;$
$xnp(2) \leftarrow Acc >> 16;$

Outputs

$Acc \leftarrow (xn(1) * 21179);$
$Acc \leftarrow Acc + (xn(2) * 31433) >> 2;$
$Acc \leftarrow Acc + (u * 17184) >> 2;$
$y \leftarrow Acc >> 15;$

Permutations

$xn \leftarrow xnp;$

Example

$$\begin{cases} X(k+1) &= \begin{pmatrix} 0.58399 & -0.42019 \\ 0.42019 & 0.1638 \end{pmatrix} X(k) + \begin{pmatrix} 0.64635 \\ -0.23982 \end{pmatrix} \\ Y(k) &= \begin{pmatrix} 0.64635 & 0.23982 \end{pmatrix} X(k) + 0.13111 U(k) \end{cases}$$

$$\beta_U = \beta_X = \beta_Y = 16$$
$$\beta_Z = 16 I_3$$
$$\beta_{ADD} = \begin{pmatrix} 32 \\ 32 \\ 32 \end{pmatrix}$$
$$\overset{\max}{U} = 10 \quad \Rightarrow \quad \gamma_U = 11$$
$$\gamma_{ADD} = \begin{pmatrix} 26 \\ 27 \\ 26 \end{pmatrix}$$
$$\gamma_Z = \begin{pmatrix} 15 & 15 & 15 \\ 16 & 16 & 16 \\ 15 & 15 & 15 \end{pmatrix}$$
$$\gamma_X = \begin{pmatrix} 11 \\ 11 \end{pmatrix}$$

Roundoff Before Multiplication

Intermediate variables

$Acc \leftarrow (xn(1) * 19136);$
$Acc \leftarrow Acc + (xn(2) * -13769);$
$Acc \leftarrow Acc + (u * 21179);$
$xnp(1) \leftarrow Acc >> 15;$
$Acc \leftarrow (xn(1) * 27537);$
$Acc \leftarrow Acc + (xn(2) * 10735);$
$Acc \leftarrow Acc + (u * -15717);$
$xnp(2) \leftarrow Acc >> 16;$

Outputs

$Acc \leftarrow (xn(1) * 21179);$
$Acc \leftarrow Acc + (xn(2) * 7858);$
$Acc \leftarrow Acc + (u * 4296);$
$y \leftarrow Acc >> 15;$

Permutations

$xn \leftarrow xnp;$

4

Optimal fixed-point implementation

Optimal design

It is possible to analytically describe equivalent classes of realization (*Inclusion Principle*)

Equivalent realization

Consider a realization $\mathcal{R}_0$. All realizations $\mathcal{R}_1$ such that

$$\begin{pmatrix} -J_1 & M_1 & N_1 \\ K_1 & P_1 & Q_1 \\ L_1 & R_1 & S_1 \end{pmatrix} = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} \begin{pmatrix} -J_0 & M_0 & N_0 \\ K_0 & P_0 & Q_0 \\ L_0 & R_0 & S_0 \end{pmatrix} \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix}$$

are equivalent (with $\mathcal{U} \in \mathbb{R}^{n \times n}$, $\mathcal{Y} \in \mathbb{R}^{l \times l}$ and $\mathcal{W} \in \mathbb{R}^{l \times l}$ non-singular matrices).

State-space : $(A, B, C, D) \rightarrow (\mathcal{T}^{-1}A\mathcal{T}, \mathcal{T}^{-1}B, C\mathcal{T}, D)$

Optimal design

It is possible to analytically describe equivalent classes of realization (*Inclusion Principle*)

Equivalent realization

Consider a realization $\mathcal{R}_0$. All realizations $\mathcal{R}_1$ such that

$$\begin{pmatrix} -J_1 & M_1 & N_1 \\ K_1 & P_1 & Q_1 \\ L_1 & R_1 & S_1 \end{pmatrix} = \begin{pmatrix} \mathcal{Y} & & \\ & \mathcal{U}^{-1} & \\ & & I_p \end{pmatrix} \begin{pmatrix} -J_0 & M_0 & N_0 \\ K_0 & P_0 & Q_0 \\ L_0 & R_0 & S_0 \end{pmatrix} \begin{pmatrix} \mathcal{W} & & \\ & \mathcal{U} & \\ & & I_m \end{pmatrix}$$

are equivalent (with $\mathcal{U} \in \mathbb{R}^{n \times n}$, $\mathcal{Y} \in \mathbb{R}^{l \times l}$ and $\mathcal{W} \in \mathbb{R}^{l \times l}$ non-singular matrices).

Optimal realization problem

The optimal design problem consists in finding the realization $\mathcal{R}^{\text{opt}}$ that minimizes $\mathcal{J}$

$$\mathcal{R}^{\text{opt}} = \underset{\mathcal{R} \in \mathscr{R}_H}{\arg\min}\ \mathcal{J}(\mathcal{R})$$

Example

We consider the following low-pass butterworth filter

$$H(z) = \frac{0.003622z^2 + 0.007243z + 0.003622}{z^2 - 1.823z + 0.8372}$$

And the following realizations

- Z_1: direct form II with shift-operator,
- Z_2: roundoff noise-optimal state-space realization,
- Z_3: roundoff noise-optimal δ-realization.

16 bits for the coefficients and variables. 32 bits for the accumulator.
Roundoff Before Multiplication

Example

The optimizations are done with Adaptative Simulated Annealing method.

realization	**Roundoff**	**Nb. operations**
Z_1	$3.914e-3$	$4+\ 5\times$
Z_2	$3.903e-7$	$6+\ 9\times$
Z_3	$3.540e-7$	$8+\ 11\times$

5

Conclusions

Conclusions

Conclusion

Some tools are exhibited to help to answer the question:

How *optimally* implement filter/controllers ?

- Implicit state-space framework
- Roundoff noise analysis
- Two bit-accurate fixed-point implementation schemes

A Matlab's toolbox (*Finite Wordlength Realization Toolbox*) was developed, with some others Finite WordLength measures (sensitivity):

`http://fwrtoolbox.gforge.inria.fr`

Conclusions

Conclusion

Some tools are exhibited to help to answer the question:

How *optimally* implement filter/controllers ?

- Implicit state-space framework
- Roundoff noise analysis
- Two bit-accurate fixed-point implementation schemes

A Matlab's toolbox (*Finite Wordlength Realization Toolbox*) was developed, with some others Finite WordLength measures (sensitivity):

`http://fwrtoolbox.gforge.inria.fr`

Any questions ?

6

Appendix

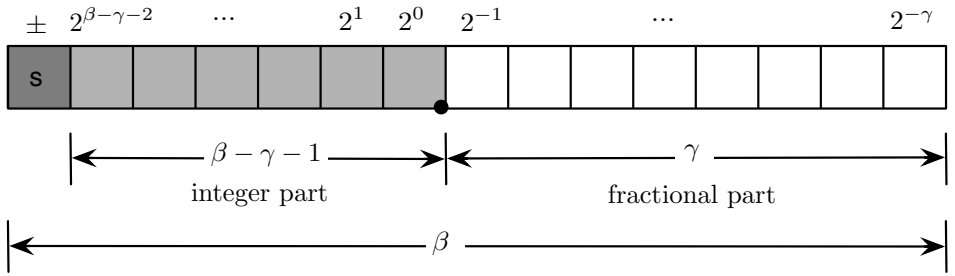
Example

Roundoff Before Multiplication scheme.

- 16 bits for the coefficients, states, inputs, outputs, intermediate variables
- Accumulator 32 bits (with 4 guard bits)

Input format (16,11)

Output format (16,10)



Example

Direct Form Directe II

```
// intermediate variables
Acc ← xn(1) * 5788 + xn(2) * -13703 + xn(3) * 17387 + xn(4) * -12469 + xn(5) * 4791 + xn(6) * -771 + u(i)
* 1;
xnp(1) ← Acc >>10;
xnp(2) ← xn(1);
xnp(3) ← xn(2);
xnp(4) ← xn(3);
xnp(5) ← xn(4);
xnp(6) ← xn(5);
// outputs
Acc ← xn(1) * 6336 + xn(2) * -19119 + xn(3) * 16167 + xn(4) * 4681 + xn(5) * -12891 + xn(6) * 4886 + u(i)
* 1; y(i) ← Acc >> 4;
// permutations
xn ← xnp;
```

Exemple

Balanced state-space form

```
// intermediate variables
Acc ← xn(1) * 32370 + xn(2) * -3673 + Acc0 + xn(3) * 42 + xn(4) * 873 + xn(5) * -171 + xn(6) * 51 + u(i) *
458;
xnp(1) ← Acc >> 15;
Acc ← xn(1) * 3688 + xn(2) * 31858 + xn(3) * 4405 + xn(4) * 785 + xn(5) * -451 + xn(6) * 74 + u(i) * -605;
xnp(2) ← Acc >> 15;
Acc ← xn(1) * 318 + xn(2) * -4416 + xn(3) * 31250 + xn(4) * -3922 + xn(5) * 499 + xn(6) * -120 + u(i) *
-790;
xnp(3) ← Acc >> 15;
Acc ← xn(1) * -900 + xn(2) * 546 + xn(3) * 3833 + xn(4) * 30304 + xn(5) * 1828 + xn(6) * -196 + u(i) * 742;
xnp(4) ← Acc >> 15;
Acc ← xn(1) * -551 + xn(2) * 1425 + xn(3) * 1760 + xn(4) * -7483 + xn(5) * 29956 + xn(6) * 1961 + u(i) *
868;
xnp(5) ← Acc >> 15;
Acc ← xn(1) * -786 + xn(2) * 1182 + xn(3) * 2572 + xn(4) * -4839 + xn(5) * -7995 + xn(6) * 29485 + u(i) *
956;
xnp(6) ← Acc >> 15;
//outputs
Acc ← xn(1) * 14733 + xn(2) * 21060 + xn(3) * -23783 + xn(4) * -22615 + xn(5) * 7488 + xn(6) * -1780 +
u(i) * 77;
y(i) ← Acc >> 15;
// permutations
xn ← xnp;
```

Exemple

Direct form II with δ-operator

```
// intermediate variables
Acc ← xn(1) * -11383 + xn(2) * -31123 + xn(3) * -22773 + xn(4) * -13468 + xn(5) * -9425 + xn(6) * -1852 +
u(i) << 8;
T0 ← Acc >>13;
// states
Acc ← T0 + xn(1) <<2;
xn(1) ← Acc >>2;
Acc ← xn(1) + xn(2) <<3;
xn(2) ← Acc >>3;
Acc ← xn(2) + xn(3) <<2;
xn(3) ← Acc >>2;
Acc ← xn(3) + xn(4) <<2;
xn(4) ← Acc >>2;
Acc ← xn(4) + xn(5) <<3;
xn(5) ← Acc >>3;
Acc ← xn(5) + xn(6) <<2;
xn(6) ← Acc >>2; // outputs
Acc ← xn(1) * 792 + xn(2) * 12559 + xn(3) * 12190 + xn(4) * 29211 + xn(5) * 22483 + xn(6) * 30784 + u(i)
* 19;
y(i) ← Acc >>13 );
```

Exemple

Direct form II with δ-operator : *Roundoff After Multiplication*

```
// intermediate variables
Acc ← (xn(1) * -22767) >> 1 + xn(2) * -31123 + xn(3) * -22773 +(xn(4) * -26936) >> 1 + (xn(5) * -18851)
>> 1 + (xn(6) * -29635) >> 4 + u(i) << 8;
T0 ← Acc >>13;
// states
Acc ← T0 + xn(1) <<2;
xn(1) ← Acc >>2;
Acc ← xn(1) + xn(2) <<3;
xn(2) ← Acc >>3;
Acc ← xn(2) + xn(3) <<2;
xn(3) ← Acc >>2;
Acc ← xn(3) + xn(4) <<2;
xn(4) ← Acc >>2;
Acc ← xn(4) + xn(5) <<3;
xn(5) ← Acc >>3;
Acc ← xn(5) + xn(6) <<2;
xn(6) ← Acc >>2; // outputs
Acc ← (xn(1) * 25342) >> 5 + (xn(2) * 25118) >> 1 + (xn(3) * 24379) >> 1 + xn(4) * 29211 + xn(5) *
22483 + xn(6) * 30784 + (u(i) * 19746) >> 10;
y(i) ← Acc >>13
```