Implémentation optimale de filtres linéaires en arithmétique virgule fixe

Soutenance de thèse de

Benoit Lopez (LIP6, UPMC)

Directeur de thèse : L.-S. Didier (IMATH, Univ. de Toulon)
Co-encadrant de thèse : T. Hilaire (LIP6, UPMC)

Cursus

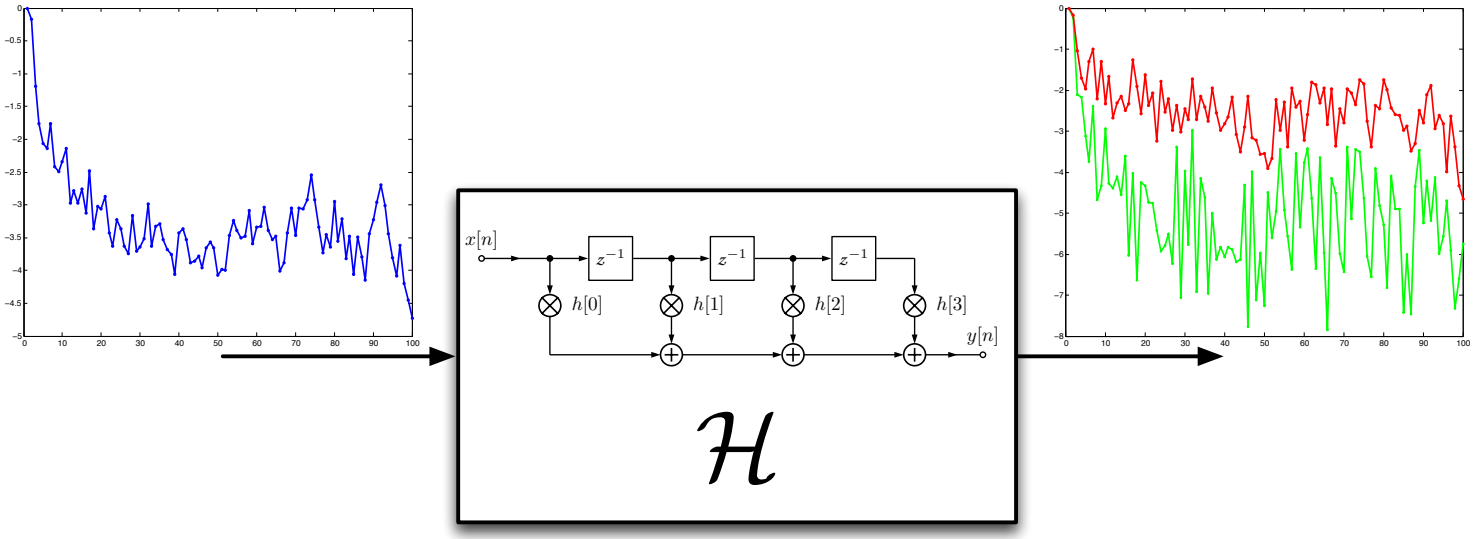
- 2007 : Licence de mathématique, UPVD
- 2010 : Master de mathématique, parcours Maths-Info, UM2
- 2014 : Doctorat d'informatique, UPMC, financé par le projet ANR DEFIS

Enseignement

- 2011-12 : 64h, UPEC.
- 2012-15 : 2 x 64h + 192h, Polytech'Paris UPMC

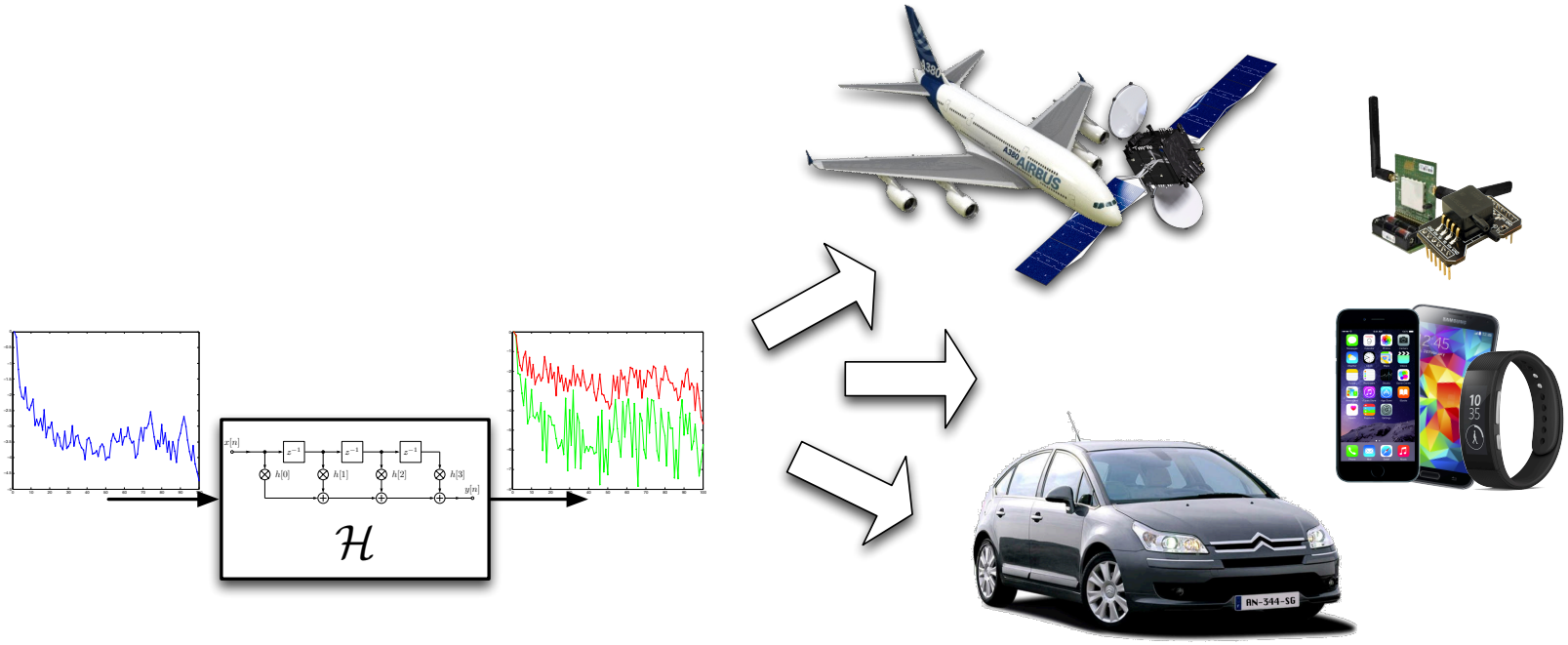
Soit 384h d'enseignement en informatique, principalement des langages de programmation (Python, C, Fortran), de la programmation web (PHP, HTML, SQL), et l'environnement Linux. Responsabilité de deux modules en 2014-2015.

D'un côté un filtre...



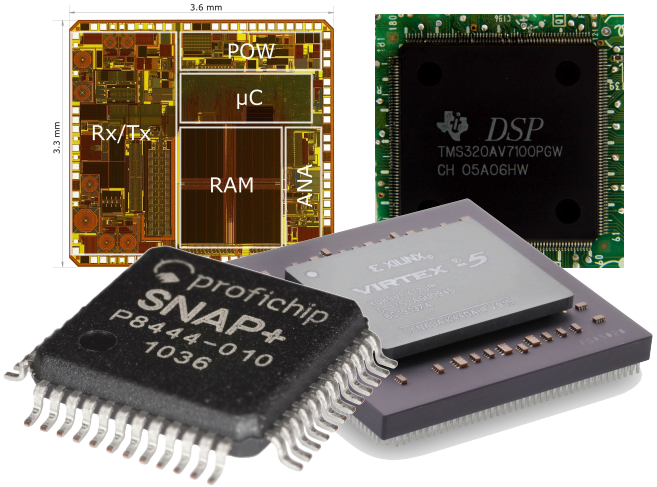
- Traitement du signal
- Filtre linéaire et invariant dans le temps
- À coefficients réels

D'un côté un filtre...



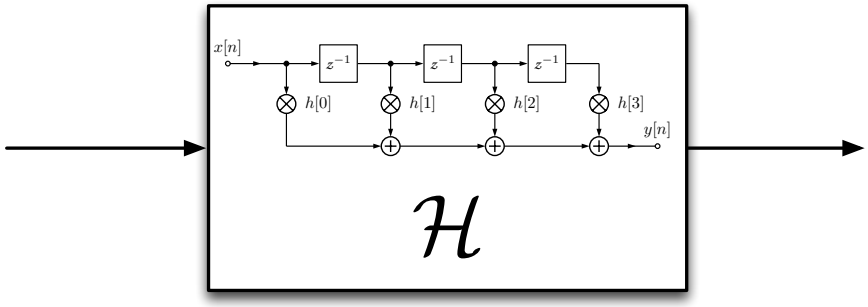
- Traitement du signal
- Filtre linéaire et invariant dans le temps
- À coefficients réels

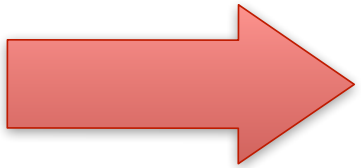
... de l'autre une cible – 1

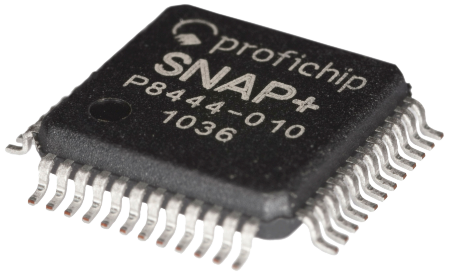


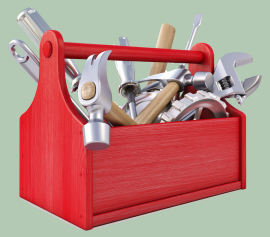
- Système embarqué matériel ou logiciel
- On ne peut pas représenter les réels, et pour des raisons de ressources on utilise l'arithmétique virgule fixe.

... de l'autre une cible – 2









On a besoin d'une méthodologie et d'outils pour implémenter ces filtres linéaires réels en virgule fixe.

L'implémentation en virgule fixe est difficile

Implémentation à partir d'un algorithme réel $\Longrightarrow$ dégradations numériques

L'implémentation en virgule fixe est difficile

Implémentation à partir d'un algorithme réel $\Longrightarrow$ dégradations numériques

Origines des dégradations

- Quantification des coefficients réels
- Arrondis dans les calculs

L'implémentation en virgule fixe est difficile

Implémentation à partir d'un algorithme réel $\Longrightarrow$ dégradations numériques

Origines des dégradations

- Quantification des coefficients réels
- Arrondis dans les calculs

Les dégradations dépendent de

- l'algorithme utilisé pour décrire le filtre
- la façon dont sont effectués les calculs

Problématique

Question :

Comment implémenter *au mieux* un filtre linéaire réel en virgule fixe ?

Problématique

Question :

Comment implémenter *au mieux* un filtre linéaire réel en virgule fixe ?

Pour répondre à cette question, plusieurs étapes sont nécessaires :

- Formaliser précisément l'arithmétique virgule fixe

Problématique

Question :

Comment implémenter *au mieux* un filtre linéaire réel en virgule fixe ?

Pour répondre à cette question, plusieurs étapes sont nécessaires :

- Formaliser précisément l'arithmétique virgule fixe
- Exploiter les propriétés des filtres linéaires

Problématique

Question :

Comment implémenter *au mieux* un filtre linéaire réel en virgule fixe ?

Pour répondre à cette question, plusieurs étapes sont nécessaires :

- Formaliser précisément l'arithmétique virgule fixe
- Exploiter les propriétés des filtres linéaires
- Décrire une méthodologie pour le *logiciel* et une autre pour le *matériel*

Problématique

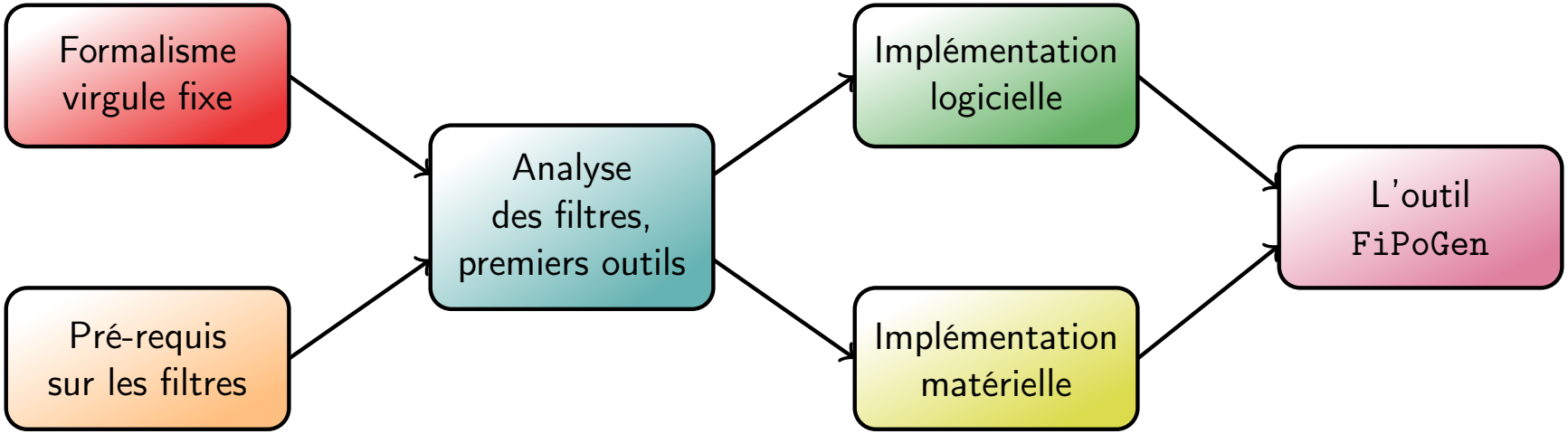
Question :

Comment implémenter *au mieux* un filtre linéaire réel en virgule fixe ?

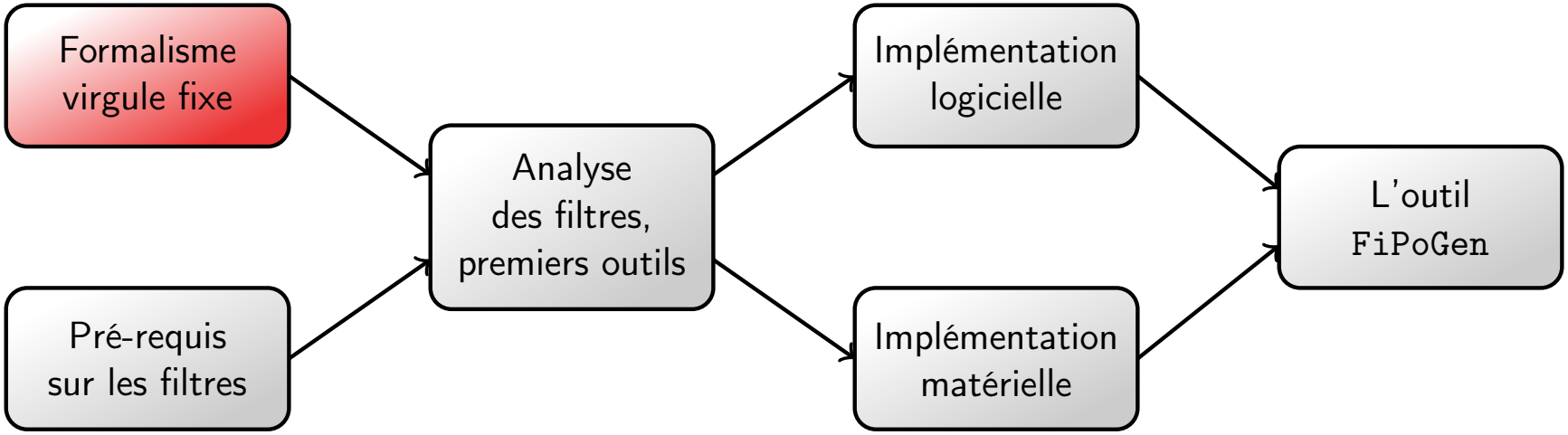
Pour répondre à cette question, plusieurs étapes sont nécessaires :

- Formaliser précisément l'arithmétique virgule fixe
- Exploiter les propriétés des filtres linéaires
- Décrire une méthodologie pour le *logiciel* et une autre pour le *matériel*
- Proposer un outil pour mettre en œuvre l'ensemble de cette démarche

Plan

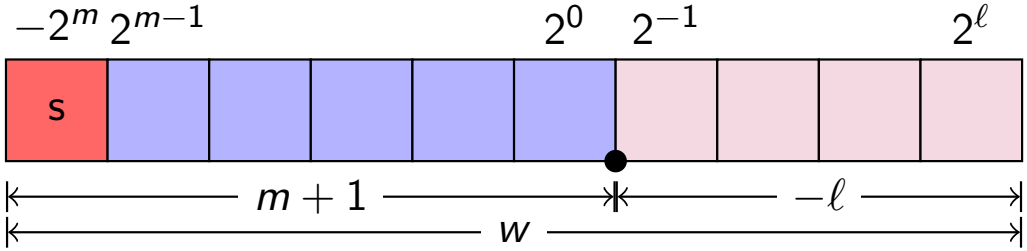

Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

Virgule Fixe

Formalisme virgule fixe

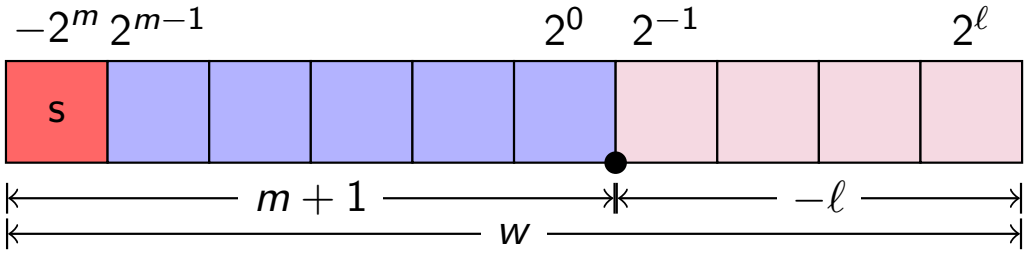


- un entier multiplié par un facteur d'échelle fixe : $x = X \cdot 2^\ell$
- un format virgule fixe : (m, ℓ)
- une largeur : w
- arithmétique signée en complément à deux

Le facteur d'échelle est implicite : le développeur sait où sont les virgules, mais le code implanté est uniquement basé sur des calculs en nombres entiers.

Virgule Fixe

Formalisme virgule fixe

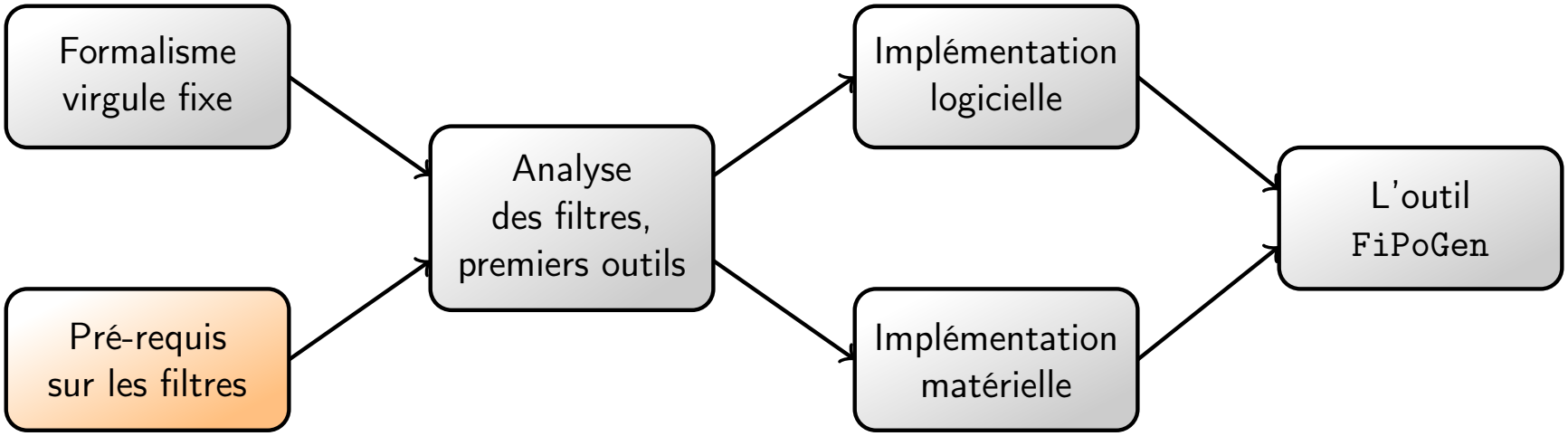


- un entier multiplié par un facteur d'échelle fixe : $x = X \cdot 2^\ell$
- un format virgule fixe : (m, ℓ)
- une largeur : w
- arithmétique signée en complément à deux

Première contribution

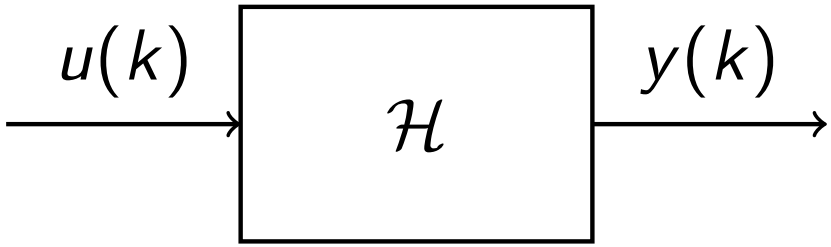
Une formalisation détaillée de l'arithmétique virgule fixe pour les algorithmes de filtres linéaires.

Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

Filtres Linéaires

Pré-requis sur les filtres



- LTI : Filtres linéaires invariants dans le temps
  - FIR (réponse impulsionnelle finie) :

$$H(z) = \sum_{i=0}^{n} b_i z^{-i}, \quad \Rightarrow y(k) = \sum_{i=0}^{n} b_i u(k-i)$$

  - IIR (réponse impulsionnelle infinie) :

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}, \quad \Rightarrow y(k) = \sum_{i=0}^{n} b_i u(k-i) - \sum_{i=1}^{n} a_i y(k-i)$$

Filtres Linéaires

Pré-requis sur les filtres

- LTI : Filtres linéaires invariants dans le temps
  - FIR (réponse impulsionnelle finie) :

$$H(z) = \sum_{i=0}^{n} b_i z^{-i}, \quad \Rightarrow y(k) = \sum_{i=0}^{n} b_i u(k-i)$$

  - IIR (réponse impulsionnelle infinie) :

$$H(z) = \frac{\sum_{i=0}^{n} b_i z^{-i}}{1 + \sum_{i=1}^{n} a_i z^{-i}}, \quad \Rightarrow y(k) = \sum_{i=0}^{n} b_i u(k-i) - \sum_{i=1}^{n} a_i y(k-i)$$

On a un premier algorithme (ou réalisation) exprimant à chaque instant la sortie $y(k)$ en fonction de l'entrée $u(k)$, mais ce n'est pas le seul.

Réalisations équivalentes

Pré-requis sur les filtres

Pour un filtre LTI donné, il y a de nombreuses façons de décrire la relation entre $u(k)$ et $y(k)$.

Réalisations équivalentes

Pré-requis sur les filtres

Pour un filtre LTI donné, il y a de nombreuses façons de décrire la relation entre $u(k)$ et $y(k)$.

Toutes ces réalisations sont équivalentes en précision infinie mais ne le sont plus en précision finie.

☞ La dégradation en précision finie dépend du choix de la réalisation.

Réalisations équivalentes

Pré-requis sur les filtres

Pour un filtre LTI donné, il y a de nombreuses façons de décrire la relation entre $u(k)$ et $y(k)$.

Toutes ces réalisations sont équivalentes en précision infinie mais ne le sont plus en précision finie.

☞ La dégradation en précision finie dépend du choix de la réalisation.

Par exemple

- Forme directe I
- State-space
- Forme directe II transposée avec opérateur ρ
- Structure LGS
- etc.

Une structure unificatrice

Pré-requis sur les filtres

On peut décrire toutes ces réalisations sous la forme d'une structure commune appelée forme implicite spécialisée (SIF).

- Description matricielle des calculs
- Explicite les opérations et leurs liens

$$\begin{pmatrix} J & 0 & 0 \\ -K & I & 0 \\ -L & 0 & I \end{pmatrix} \begin{pmatrix} T(k+1) \\ X(k+1) \\ Y(k) \end{pmatrix} = \begin{pmatrix} 0 & M & N \\ 0 & P & Q \\ 0 & R & S \end{pmatrix} \begin{pmatrix} T(k) \\ X(k) \\ U(k) \end{pmatrix}$$

Une structure unificatrice

Pré-requis sur les filtres

On peut décrire toutes ces réalisations sous la forme d'une structure commune appelée forme implicite spécialisée (SIF).

- Description matricielle des calculs
- Explicite les opérations et leurs liens

$$\begin{cases} J \cdot T(k+1) & \leftarrow & M \cdot X(k) + N \cdot U(k) \\ X(k+1) & \leftarrow & K \cdot T(k+1) + P \cdot X(k) + Q \cdot U(k) \\ Y(k) & \leftarrow & L \cdot T(k+1) + R \cdot X(k) + S \cdot U(k) \end{cases}$$

Une structure unificatrice

Pré-requis sur les filtres

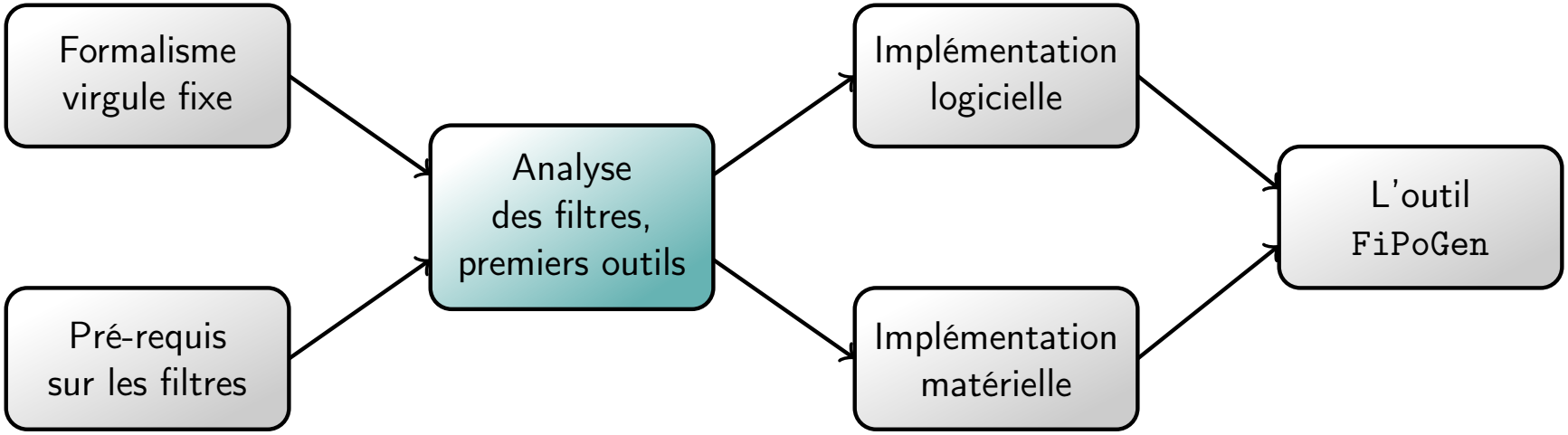
On peut décrire toutes ces réalisations sous la forme d'une structure commune appelée forme implicite spécialisée (SIF).

- Description matricielle des calculs
- Explicite les opérations et leurs liens

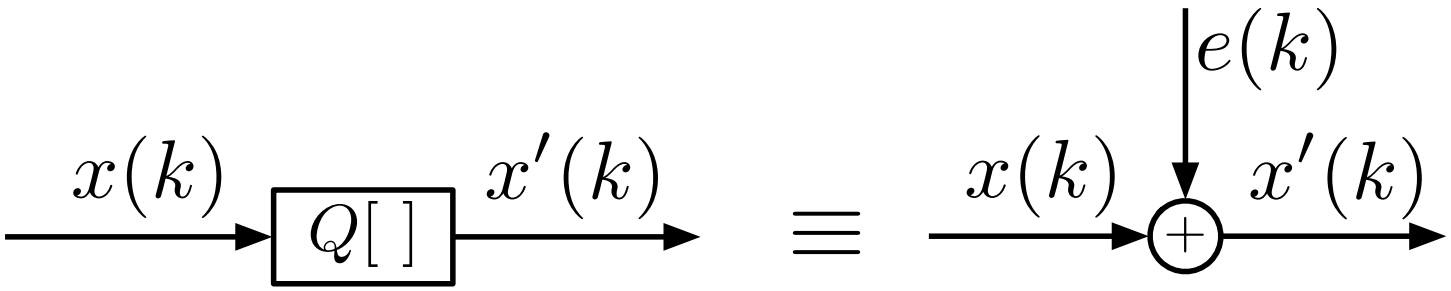
$$\begin{cases} J \cdot T(k+1) & \leftarrow & M \cdot X(k) + N \cdot U(k) \\ X(k+1) & \leftarrow & K \cdot T(k+1) + P \cdot X(k) + Q \cdot U(k) \\ Y(k) & \leftarrow & L \cdot T(k+1) + R \cdot X(k) + S \cdot U(k) \end{cases}$$

En développant une méthodologie autour de la forme implicite spécialisée, on rend cette méthodologie applicable à toutes les réalisations.

Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

Analyse d'erreur



Quantifier un signal $x(k)$ est équivalent à ajouter une erreur $e(k)$ à ce signal.

Analyse d'erreur

À l'implémentation, l'algorithme

$$\mathcal{H}\left\{\begin{array}{rcl} J\cdot T(k+1) & \leftarrow & M\cdot X(k)+N\cdot U(k) \\ X(k+1) & \leftarrow & K\cdot T(k+1)+P\cdot X(k)+Q\cdot U(k) \\ Y(k) & \leftarrow & L\cdot T(k+1)+R\cdot X(k)+S\cdot U(k) \end{array}\right.$$

Analyse d'erreur

À l'implémentation, l'algorithme

$$\mathcal{H}\left\{\begin{array}{rcl} J\cdot T(k+1) & \leftarrow & M\cdot X(k)+N\cdot U(k) \\ X(k+1) & \leftarrow & K\cdot T(k+1)+P\cdot X(k)+Q\cdot U(k) \\ Y(k) & \leftarrow & L\cdot T(k+1)+R\cdot X(k)+S\cdot U(k) \end{array}\right.$$

devient

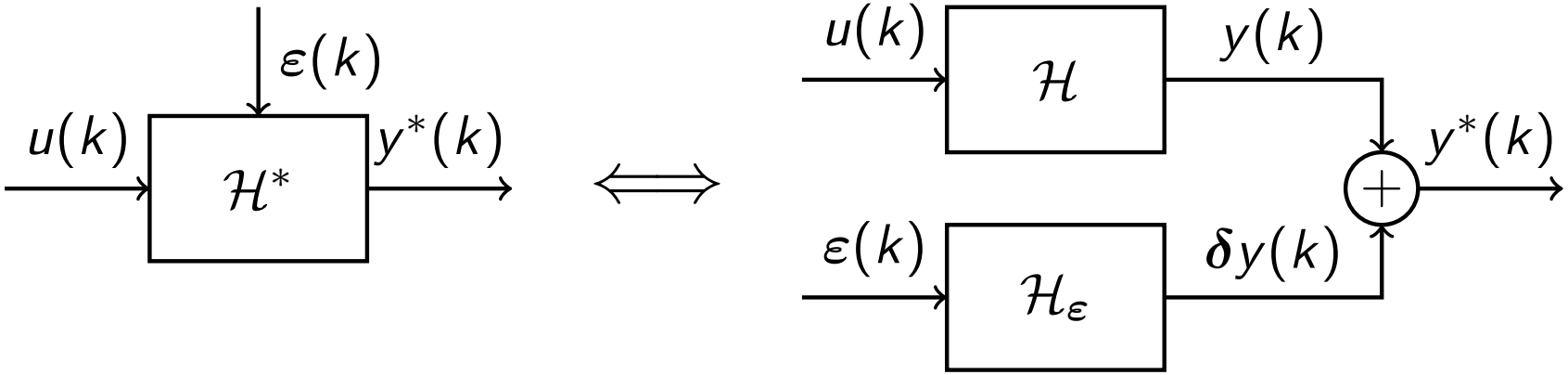
$$\mathcal{H}^*\left\{\begin{array}{rcl} J\cdot T^*(k+1) & \leftarrow & M\cdot X^*(k)+N\cdot U(k)+\varepsilon_T(k) \\ X^*(k+1) & \leftarrow & K\cdot T^*(k+1)+P\cdot X^*(k)+Q\cdot U(k)+\varepsilon_X(k) \\ Y^*(k) & \leftarrow & L\cdot T^*(k+1)+R\cdot X^*(k)+S\cdot U(k)+\varepsilon_Y(k) \end{array}\right.$$

Les erreurs d'arrondi mènent à l'addition d'une erreur globale $\varepsilon(k)$:

$$\varepsilon(k) \triangleq \begin{pmatrix} \varepsilon_T(k) \\ \varepsilon_X(k) \\ \varepsilon_Y(k) \end{pmatrix}$$

Analyse d'erreur

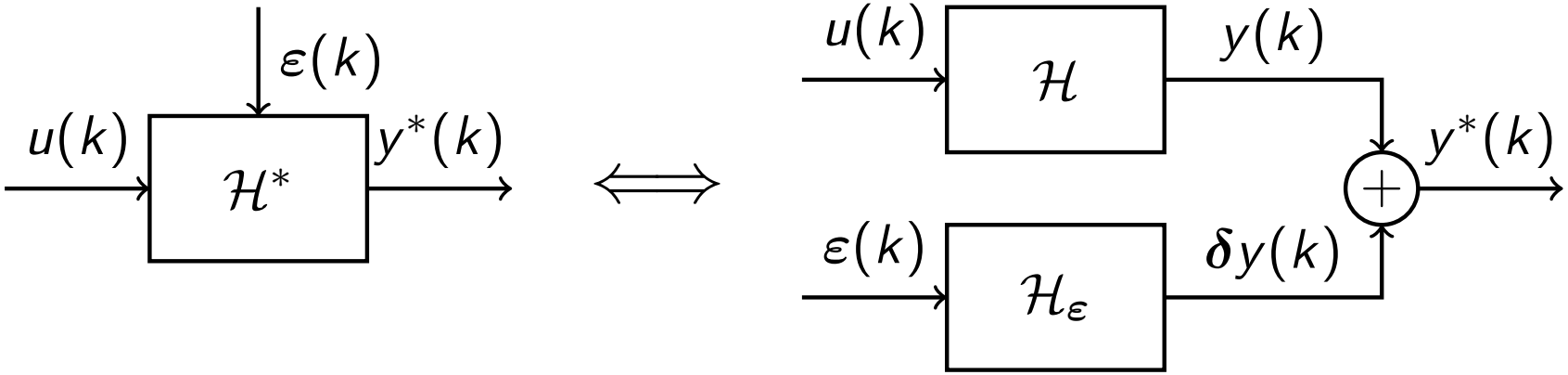
Analyse des filtres, premiers outils



On veut déterminer $\delta y(k) = y^*(k) - y(k)$.

Analyse d'erreur

Analyse des filtres, premiers outils



On veut déterminer $\delta y(k) = y^*(k) - y(k)$.

- on a besoin d'évaluer $\varepsilon(k)$
- on a besoin de connaître le comportement de $\varepsilon(k)$ à travers $\mathcal{H}_\varepsilon$

Analyse d'erreur

Évaluer l'erreur de calcul $\varepsilon(k)$, c'est déterminer les calculs qui produisent des erreurs d'arrondi.

$$\mathcal{H}\left\{\begin{array}{rcl} J\cdot T(k+1) & \leftarrow & M\cdot X(k)+N\cdot U(k) \\ X(k+1) & \leftarrow & K\cdot T(k+1)+P\cdot X(k)+Q\cdot U(k) \\ Y(k) & \leftarrow & L\cdot T(k+1)+R\cdot X(k)+S\cdot U(k) \end{array}\right.$$

Analyse d'erreur

Évaluer l'erreur de calcul $\varepsilon(k)$, c'est déterminer les calculs qui produisent des erreurs d'arrondi.

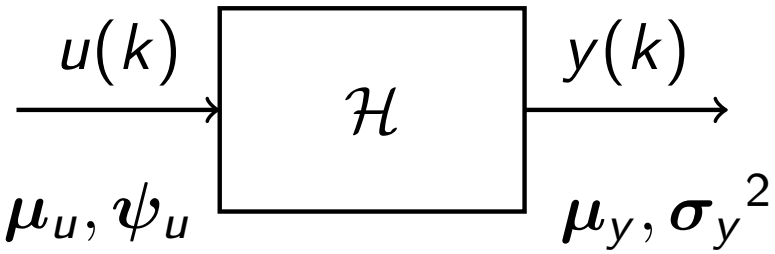
$$\mathcal{H}\left\{\begin{array}{rcl} J\cdot T(k+1) & \leftarrow & M\cdot X(k)+N\cdot U(k) \\ X(k+1) & \leftarrow & K\cdot T(k+1)+P\cdot X(k)+Q\cdot U(k) \\ Y(k) & \leftarrow & L\cdot T(k+1)+R\cdot X(k)+S\cdot U(k) \end{array}\right.$$

On a besoin de connaître les positions des virgules des variables t, x et y.
☞ Connaître le comportement du signal u à travers $\mathcal{H}$.

Propriété des filtres linéaires

Analyse des filtres, premiers outils

Approche classique du comportement d'un signal d'entrée u à travers un filtre linéaire $\mathcal{H}$.



Traitement du signal

$\boldsymbol{\mu}_u$	:	moyenne
$\boldsymbol{\psi}_u$	:	variance
$\langle\!\langle \mathcal{H} \rangle\!\rangle_{\mathrm{DC}}$	:	DC-Gain de $\mathcal{H}$
$\lVert \mathcal{H} \rVert_2$	:	norme $\mathcal{L}_2$ de $\mathcal{H}$

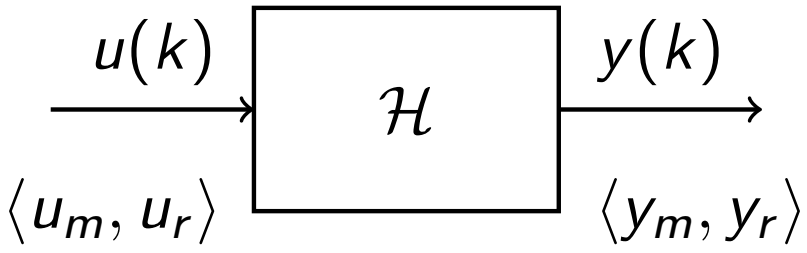
Proposition

$$\begin{aligned} \boldsymbol{\mu}_y &= \langle\!\langle \mathcal{H} \rangle\!\rangle_{\mathrm{DC}} \cdot \boldsymbol{\mu}_u \\ \boldsymbol{\sigma}_y^2 &= \lVert \mathcal{H} \cdot \boldsymbol{\varphi}_u \rVert_2^2 \end{aligned}$$

Propriété des filtres linéaires

Analyse des filtres, premiers outils

Nouvelle approche du comportement d'un signal d'entrée u à travers un filtre linéaire $\mathcal{H}$.



Intervalle

u_m	:	centre
u_r	:	rayon
$\langle\langle \mathcal{H} \rangle\rangle_{\mathrm{DC}}$	:	DC-Gain de $\mathcal{H}$
$\langle\langle \mathcal{H} \rangle\rangle_{\mathrm{WCPG}}$	:	WCPG de $\mathcal{H}$

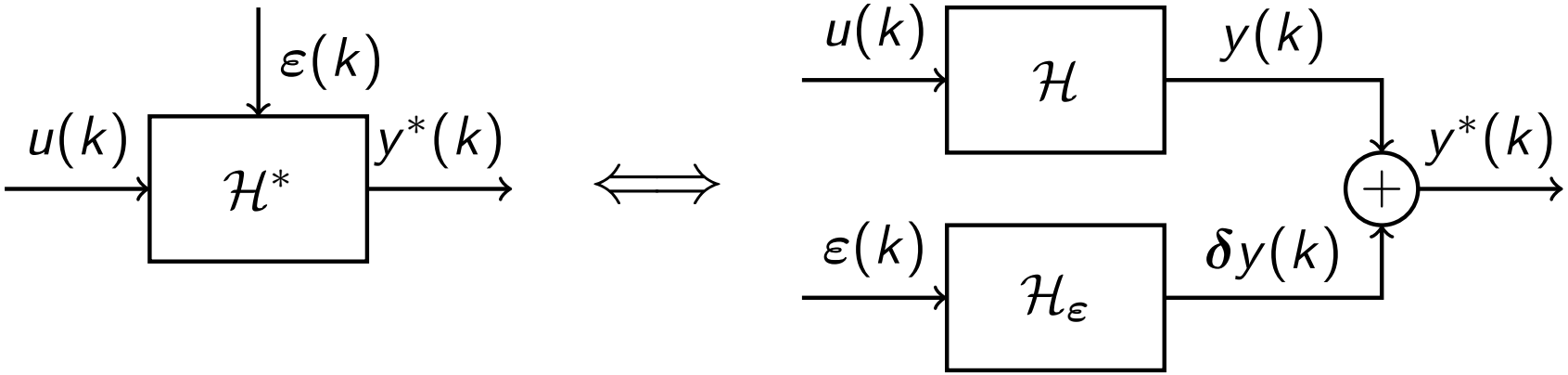
Théorème du WCPG

$$\begin{aligned} y_m &= \langle\langle \mathcal{H} \rangle\rangle_{\mathrm{DC}} \cdot u_m \\ y_r &= \langle\langle \mathcal{H} \rangle\rangle_{\mathrm{WCPG}} \cdot u_r \end{aligned}$$

en régime permanent

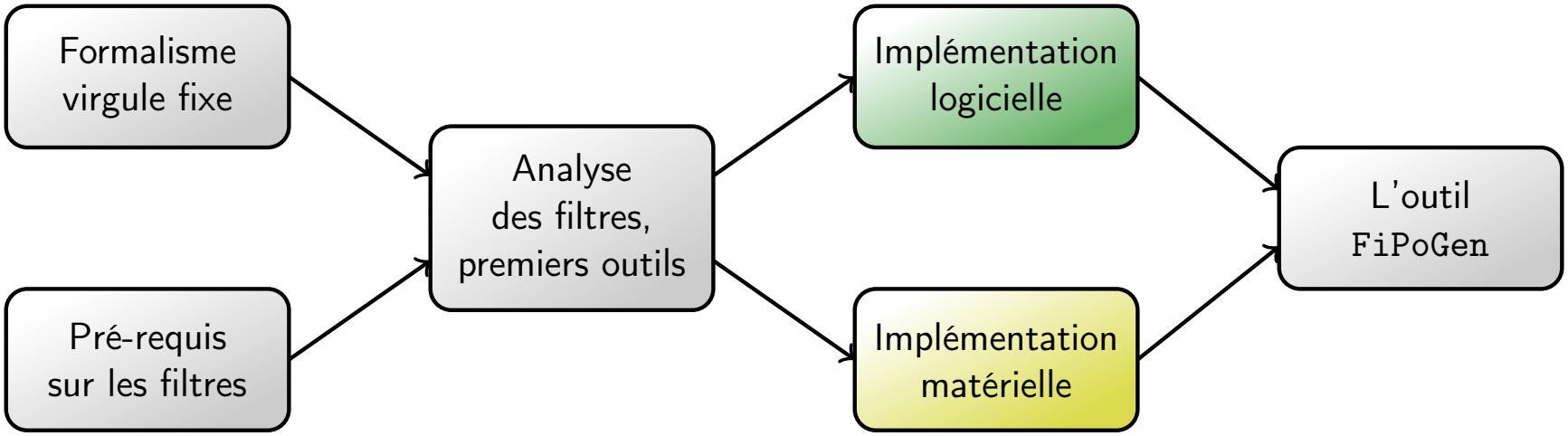
Analyse d'erreur

Analyse des filtres, premiers outils

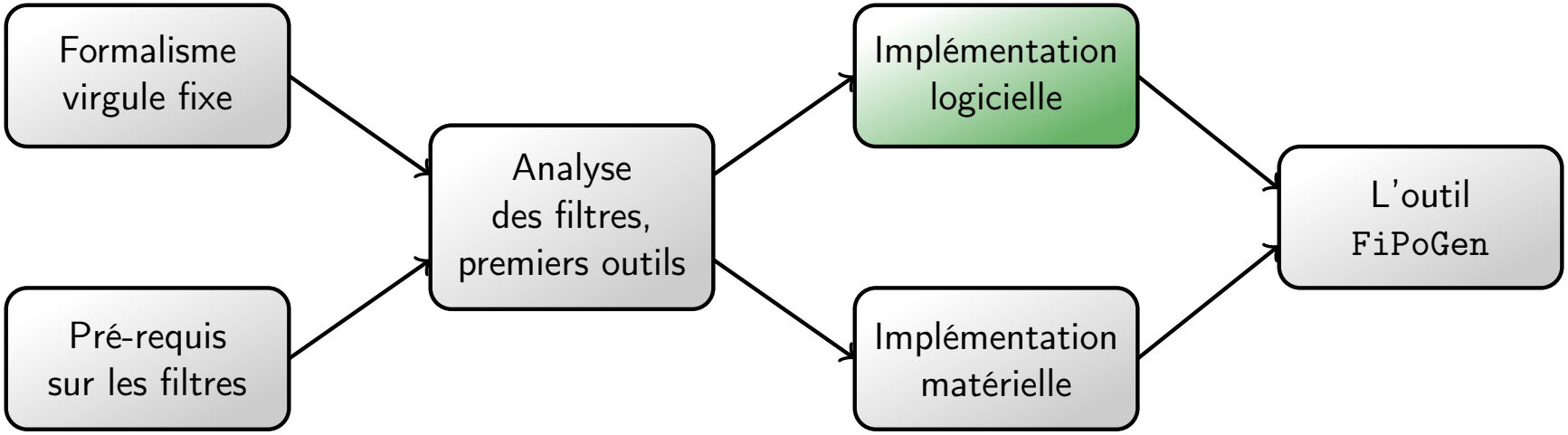


Avec notre nouvelle approche, nous sommes donc capable de déterminer l'intervalle de l'erreur $\delta y(k)$.

Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

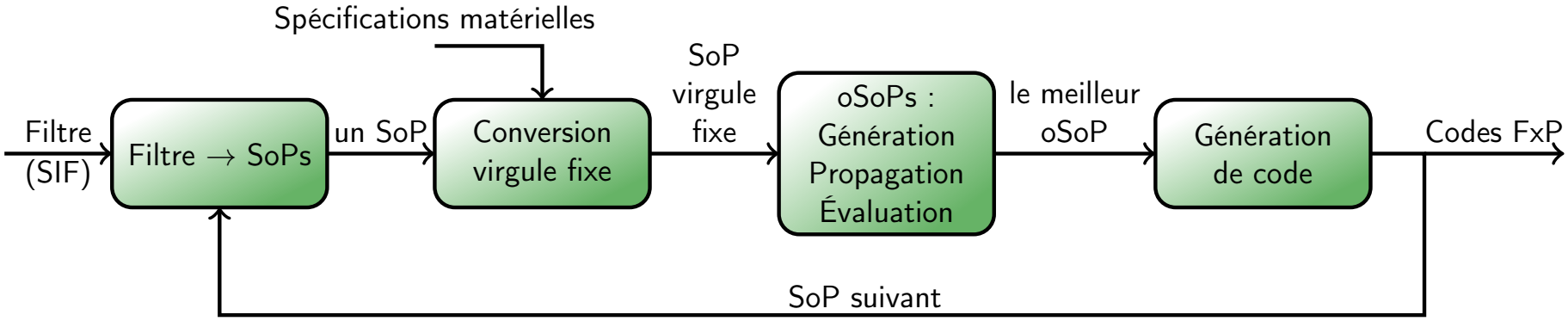
Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

Implémentation logicielle

Implémentation logicielle

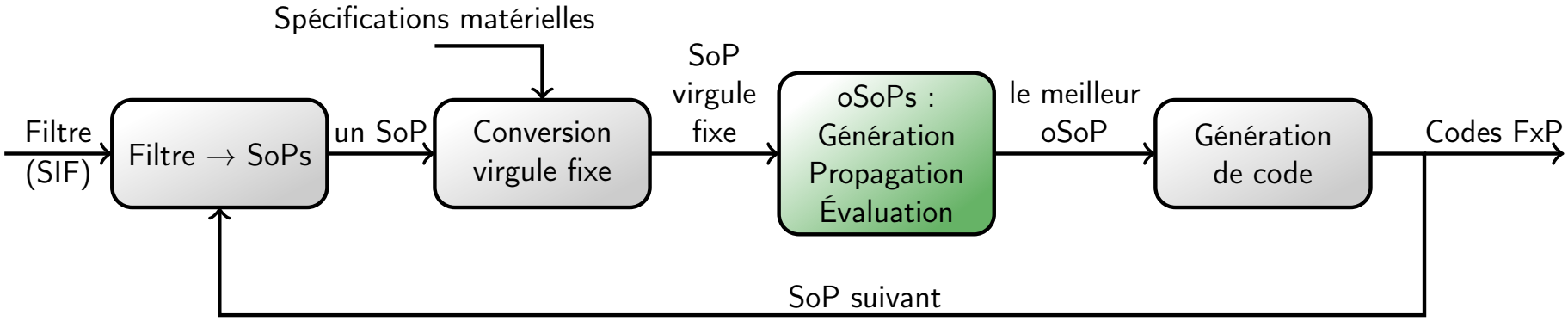
Méthodologie pour l'implémentation logicielle



Implémentation logicielle

Implémentation logicielle

Méthodologie pour l'implémentation logicielle



Implémentation logicielle

Les seules opérations nécessaires sont des sommes-de-produits :

$$S = \sum_{i=1}^{n} c_i \cdot v_i$$

où les c_i sont des constantes connues et les v_i des variables (d'entrée, d'état, ou intermédiaires).

Implémentation logicielle

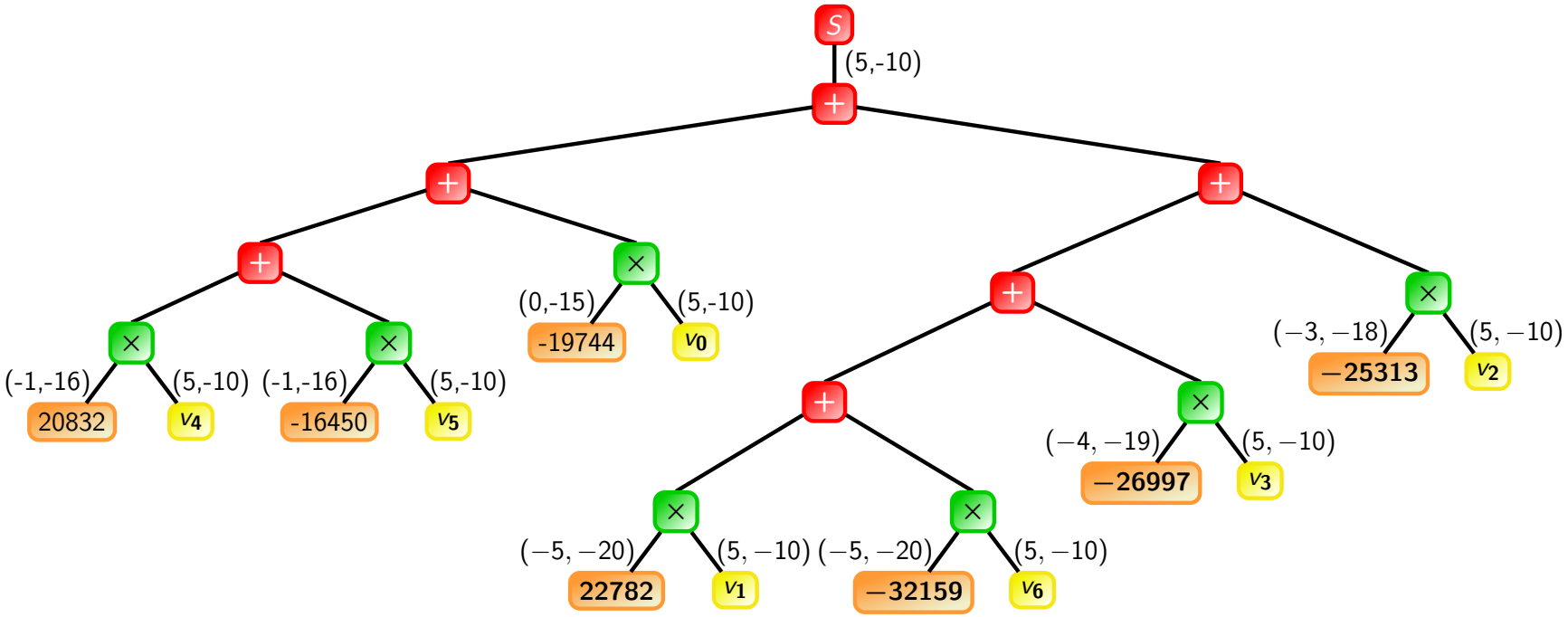
Les seules opérations nécessaires sont des sommes-de-produits :

$$S = \sum_{i=1}^{n} c_i \cdot v_i$$

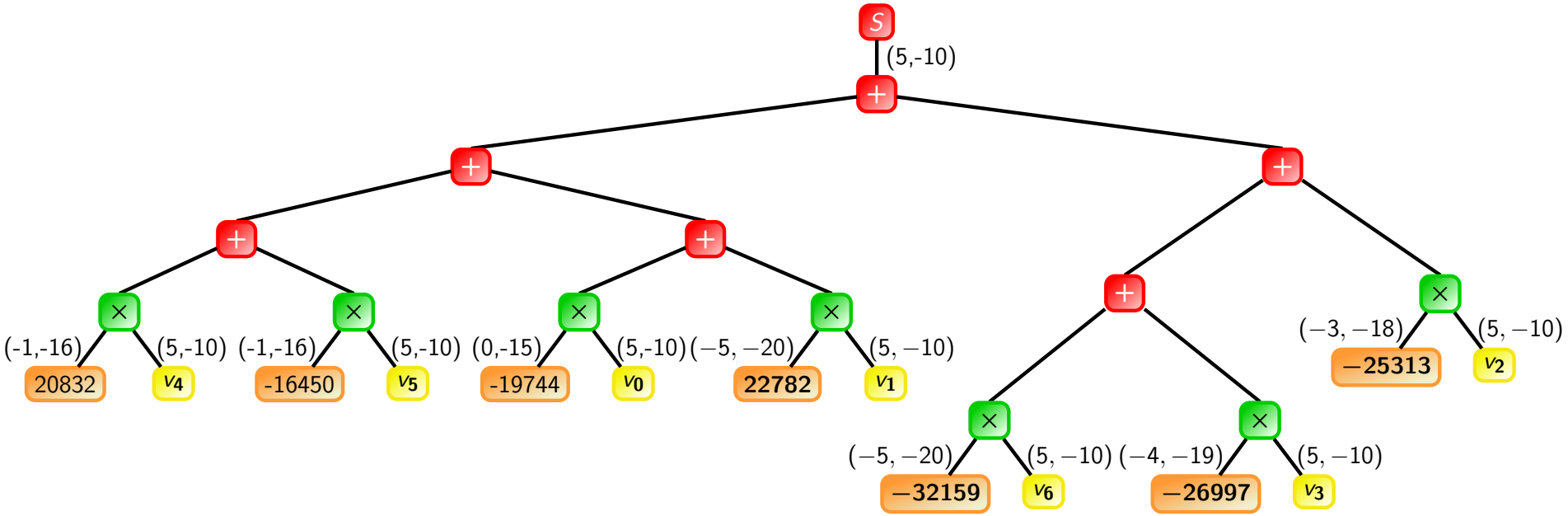
où les c_i sont des constantes connues et les v_i des variables (d'entrée, d'état, ou intermédiaires).

L'addition en logiciel peut ne pas être associative à cause des décalages.
☞ On considère alors des sommes-de-produits ordonnées (oSoP).

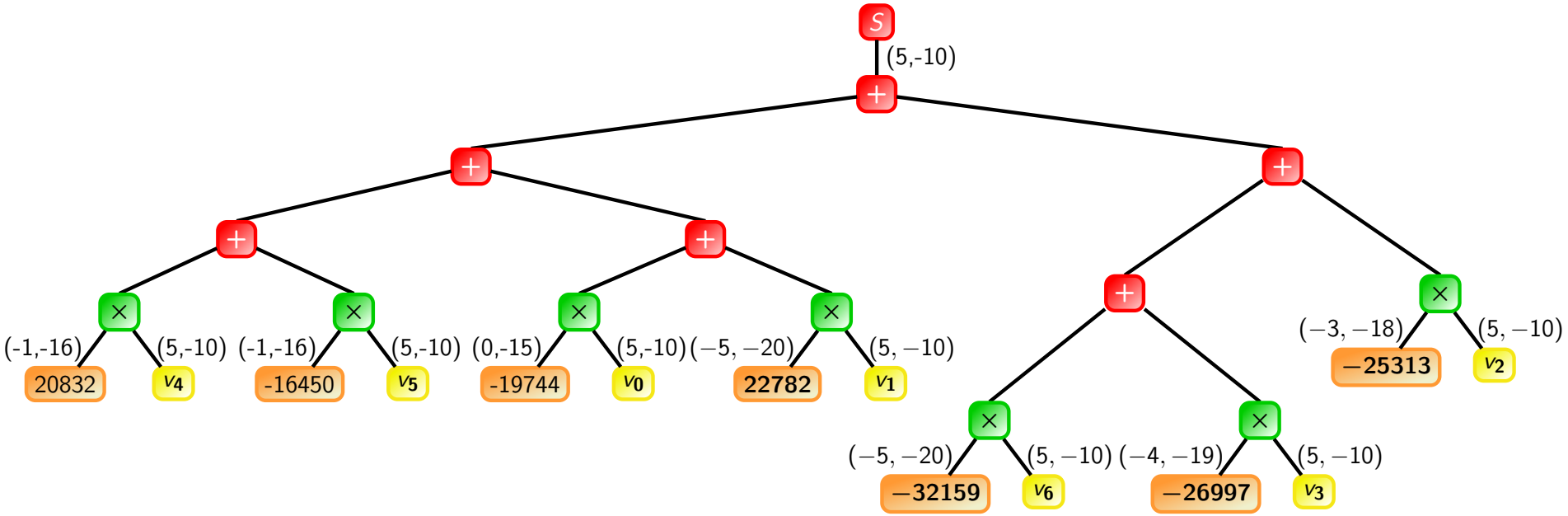
Implémentation logicielle - oSoP



Implémentation logicielle - oSoP



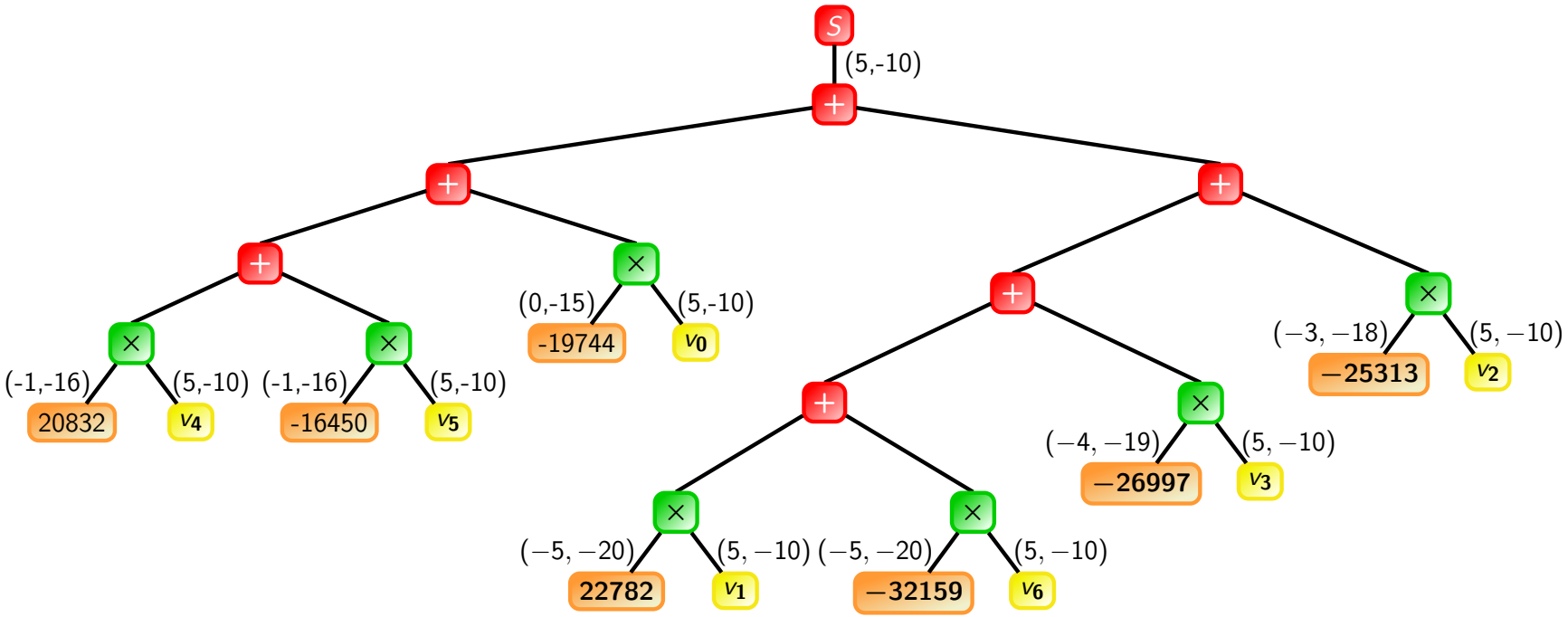
Implémentation logicielle - oSoP



Il y a $\prod_{i=1}^{n}(2i-1)$ oSoP à considérer.

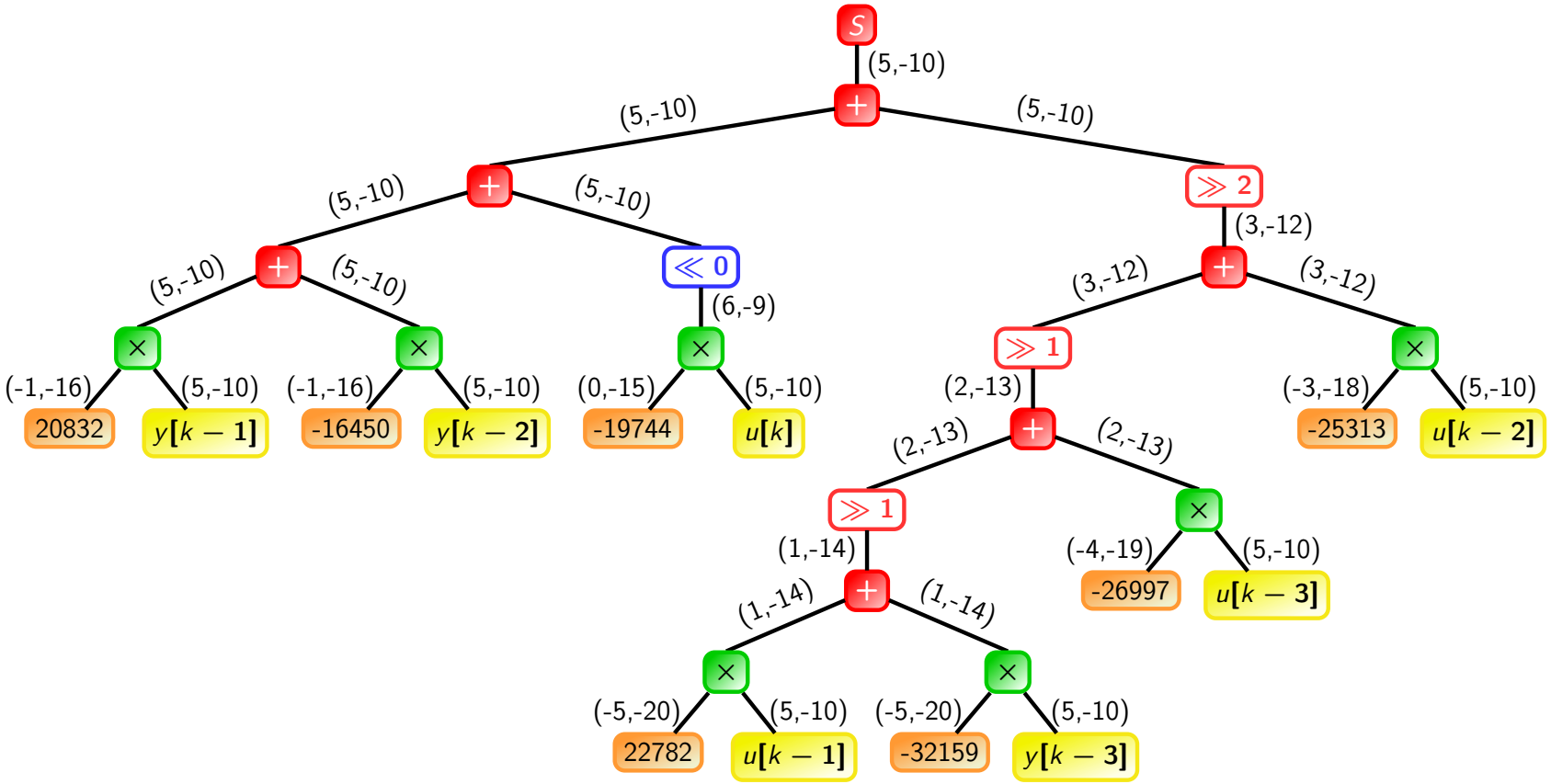
Implémentation logicielle - Propagation

Pour chaque oSoP, on propage les intervalles des variables vers la sortie en utilisant l'arithmétique d'intervalle et l'arithmétique virgule fixe.



Implémentation logicielle - Propagation

Pour chaque oSoP, on propage les intervalles des variables vers la sortie en utilisant l'arithmétique d'intervalle et l'arithmétique virgule fixe.



Implémentation logicielle - Évaluation / Sélection

Évaluation

À chaque formatage correspond une erreur (un bruit ou un intervalle) que l'on sait évaluer.

☞ On en déduit l'erreur cumulée globale pour chaque schéma.

Implémentation logicielle - Évaluation / Sélection

Évaluation

À chaque formatage correspond une erreur (un bruit ou un intervalle) que l'on sait évaluer.

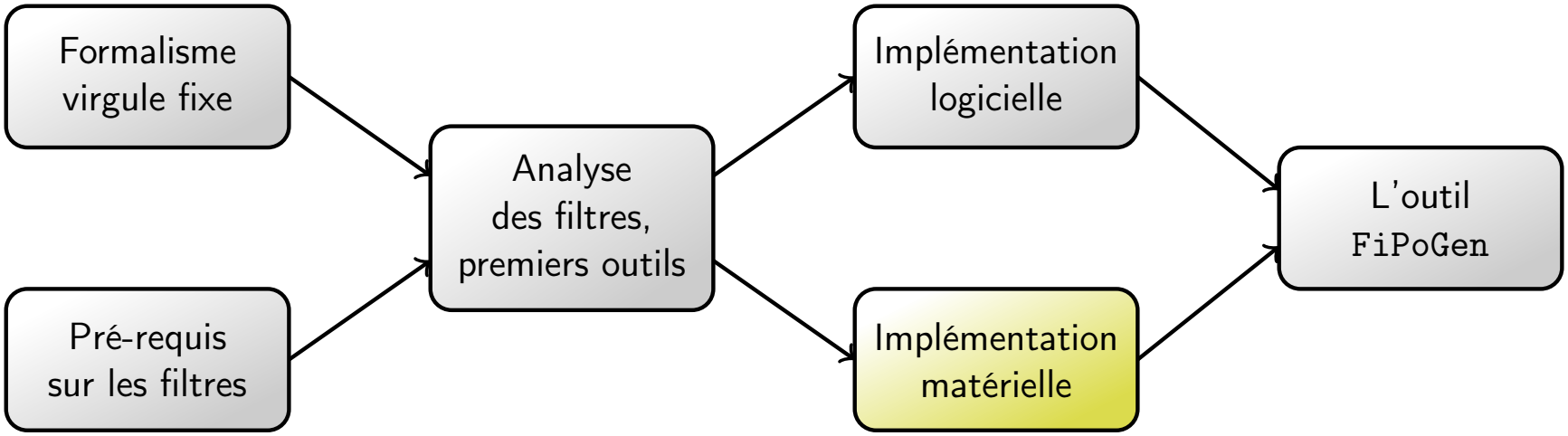
☞ On en déduit l'erreur cumulée globale pour chaque schéma.

Sélection

Plusieurs critères pour choisir le meilleur schéma :

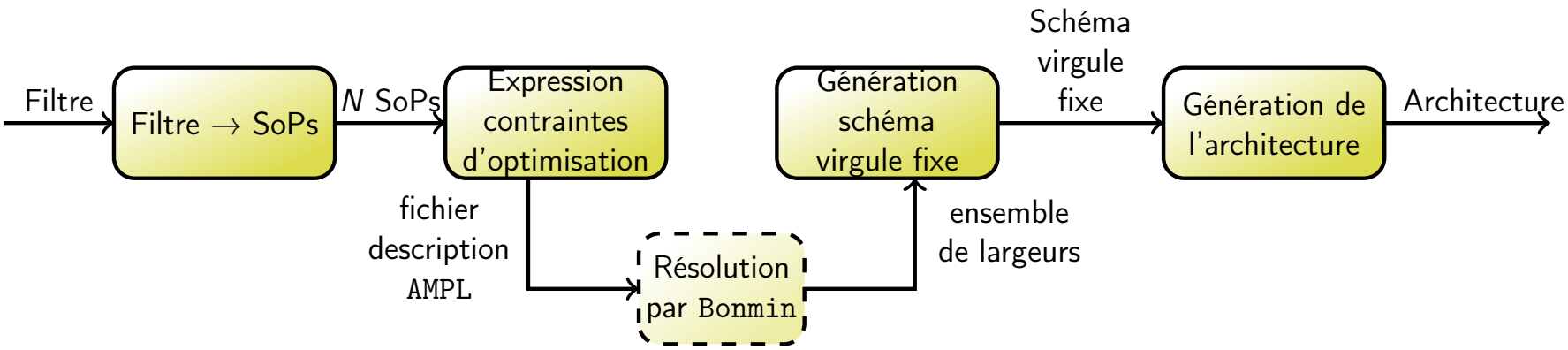
- l'erreur globale
- le parallélisme

Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

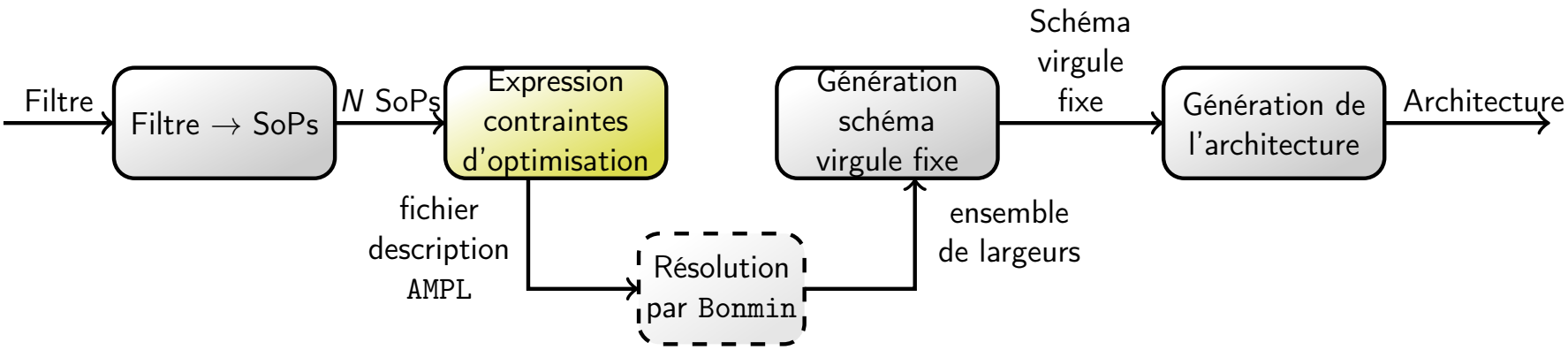
Implémentation matérielle

Méthodologie pour l'implémentation matérielle



Implémentation matérielle

Méthodologie pour l'implémentation matérielle



Implémentation matérielle

Choix sur les largeurs $\Longrightarrow$ Problème d'optimisation combinatoire

Notations

$$v = Z \cdot v_{in}$$

$$v \triangleq \begin{pmatrix} t(k+1) \\ x(k+1) \\ y(k) \end{pmatrix}, \quad Z \triangleq \begin{pmatrix} -J' & M & N \\ K & P & Q \\ L & R & S \end{pmatrix}, \quad \text{et} \quad v_{in} \triangleq \begin{pmatrix} t(k) \\ x(k) \\ u(k) \end{pmatrix}.$$

Implémentation matérielle

Choix sur les largeurs $\Longrightarrow$ Problème d'optimisation combinatoire

Notations

$$v = Z \cdot v_{in}$$

$$v \triangleq \begin{pmatrix} t(k+1) \\ x(k+1) \\ y(k) \end{pmatrix}, \quad Z \triangleq \begin{pmatrix} -J' & M & N \\ K & P & Q \\ L & R & S \end{pmatrix}, \quad \text{et} \quad v_{in} \triangleq \begin{pmatrix} t(k) \\ x(k) \\ u(k) \end{pmatrix}.$$

Problème d'optimisation

$$\min \quad f(w_Z, w_v)$$

$$\text{tel que} \quad g_i(w_Z, w_v) \leq 0$$

Implémentation matérielle

Choix sur les largeurs $\Longrightarrow$ Problème d'optimisation combinatoire

Notations

$$v = Z \cdot v_{in}$$

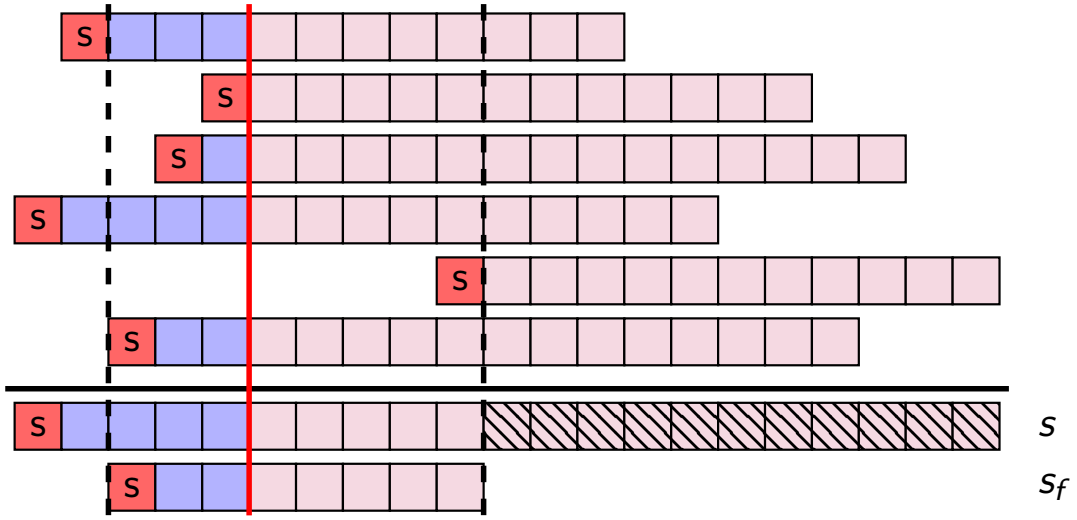
$$v \triangleq \begin{pmatrix} t(k+1) \\ x(k+1) \\ y(k) \end{pmatrix}, \quad Z \triangleq \begin{pmatrix} -J' & M & N \\ K & P & Q \\ L & R & S \end{pmatrix}, \quad \text{et} \quad v_{in} \triangleq \begin{pmatrix} t(k) \\ x(k) \\ u(k) \end{pmatrix}.$$

Problème d'optimisation

$$\begin{aligned} \min \quad & f(w_Z, w_v) = \sum_{i,j} w_{Z_{i,j}} + \sum_i w_{v_i} \\ \text{tel que} \quad & g_i(w_Z, w_v) \leq 0 \end{aligned}$$

Contraintes - Formatage de bits

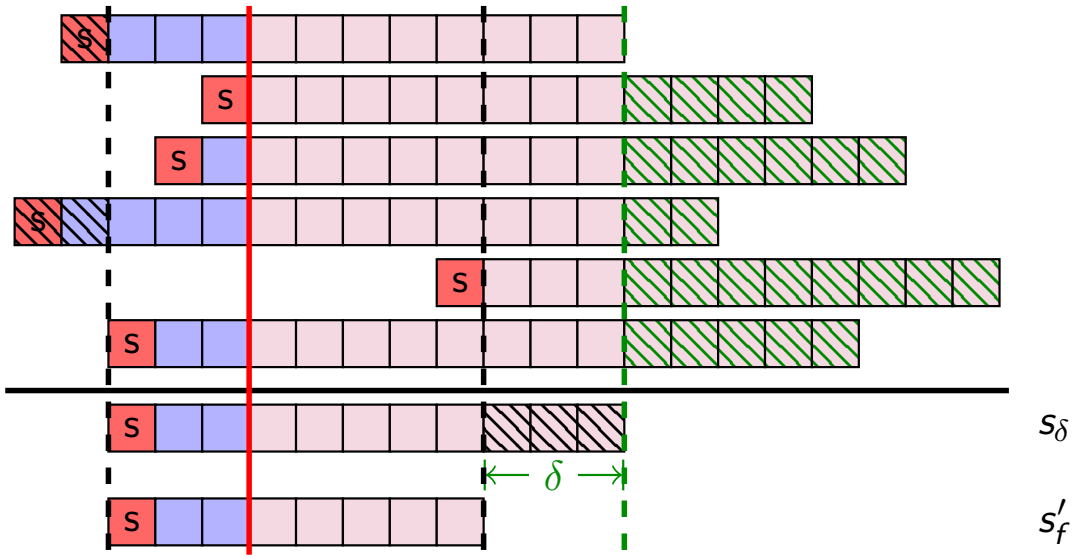
Implémentation matérielle



Contexte

Une somme de N termes $(p_i)_{1 \leq i \leq N}$ avec différents formats, et le FPF de la somme finale connu, dont la largeur est plus petite que la largeur de la somme exacte.

Contraintes - Formatage de bits

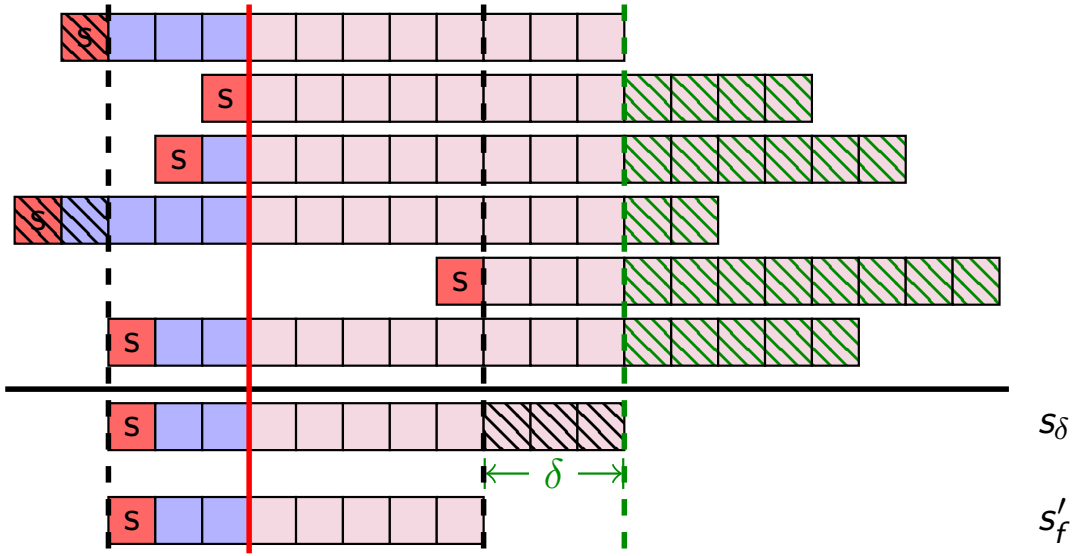


Formatage

On peut limiter le nombre de bits et obtenir un arrondi fidèle.

Contraintes - Formatage de bits

Implémentation matérielle



Calcul de δ

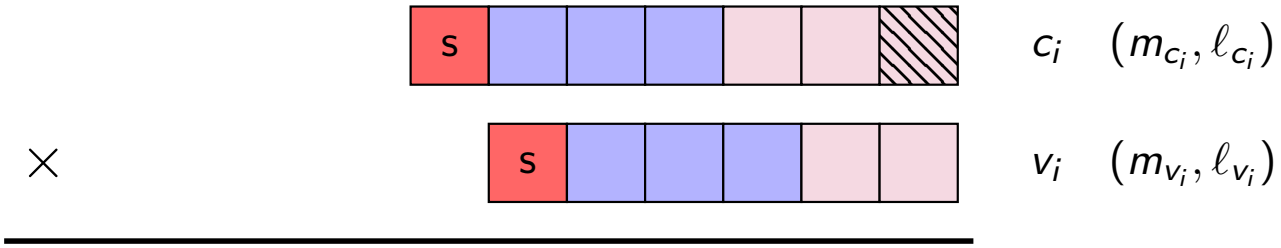
Arrondi	δ
Troncature	$\lfloor \log_2(N-1) \rfloor + 1$
Au plus proche	$\lfloor \log_2(N) \rfloor + 1$

Contraintes - Formatage de bits

Produit constante / variable

Les constantes c_i sont des constantes réelles arrondies.

☞ L'erreur sur c_i est plus petite que le bit en position ℓ_{c_i}.



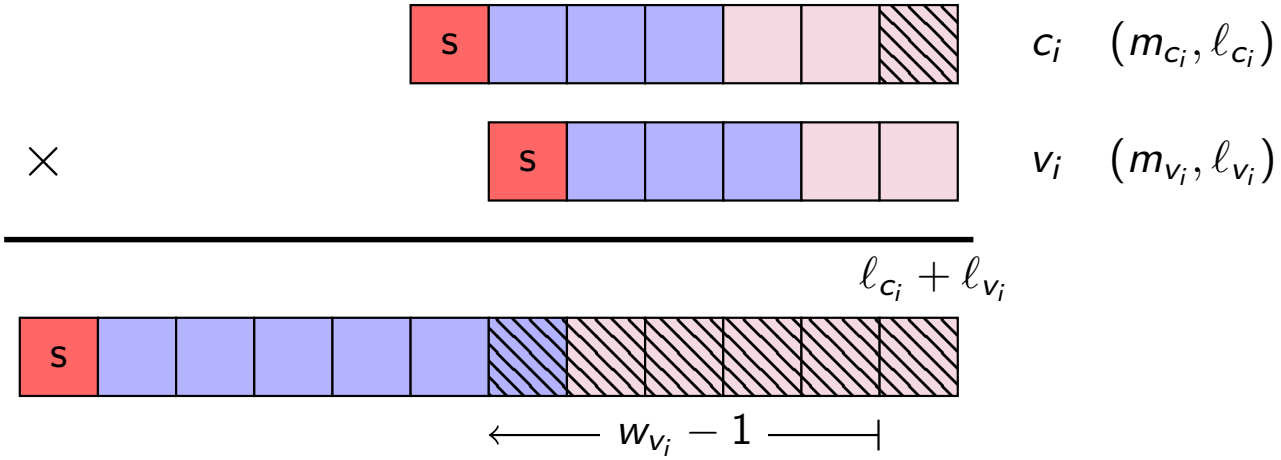
Contraintes - Formatage de bits

Implémentation matérielle

Produit constante / variable

Les constantes c_i sont des constantes réelles arrondies.

☞ L'erreur sur c_i est plus petite que le bit en position ℓ_{c_i}.



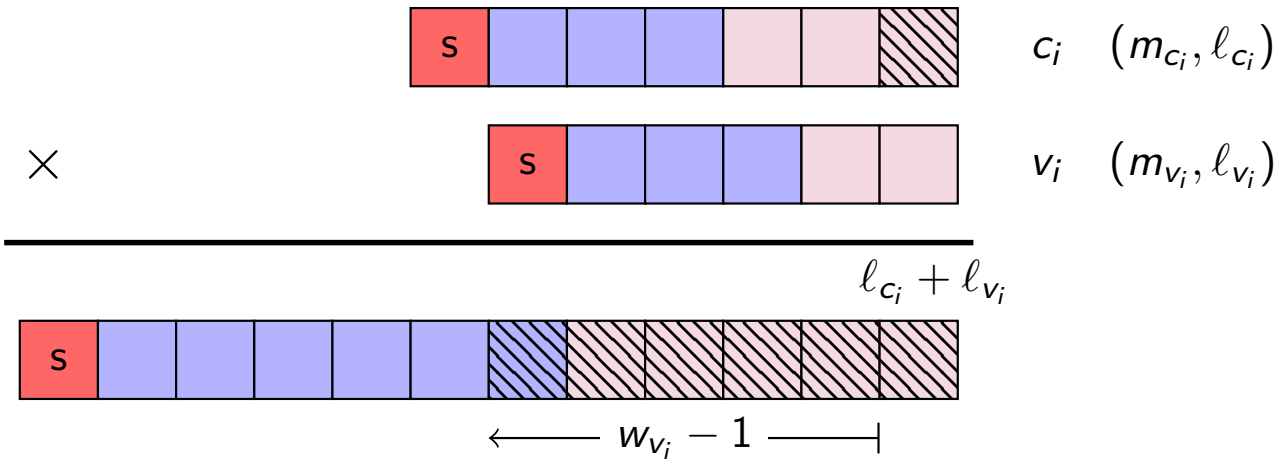
Contraintes - Formatage de bits

Implémentation matérielle

Produit constante / variable

Les constantes c_i sont des constantes réelles arrondies.

☞ L'erreur sur c_i est plus petite que le bit en position ℓ_{c_i}.

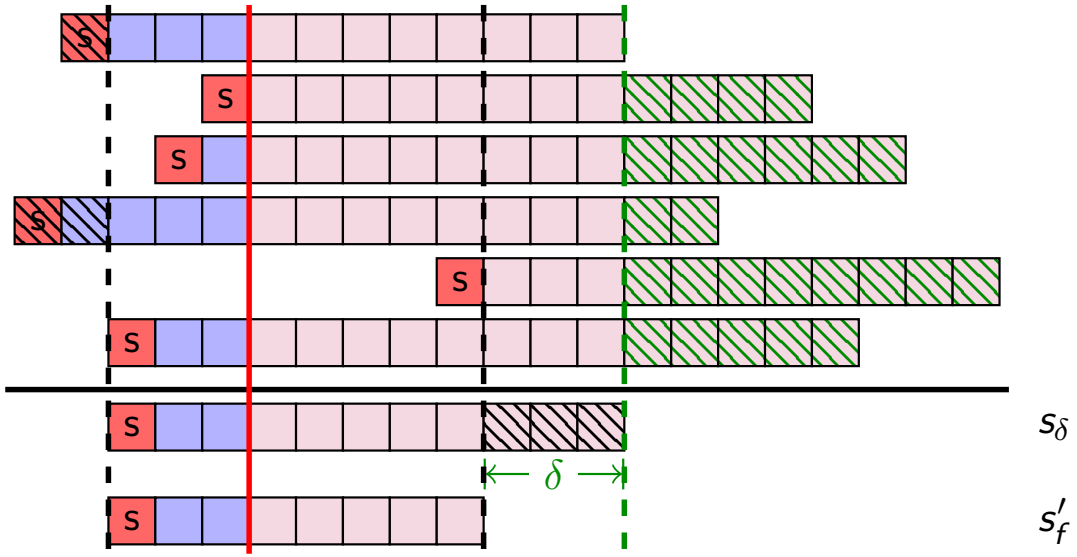


L'erreur se propage sur les bits en position $\ell_{c_i} + m_{v_i}$ et inférieures.

Contraintes - Formatage de bits

On combine le formatage avec l'observation précédente pour obtenir la contrainte suivante :

$$\ell_s - \delta > \ell_{c_i} + m_{v_i}, \quad \forall i$$



Contraintes - Formatage de bits

On combine le formatage avec l'observation précédente pour obtenir la contrainte suivante :

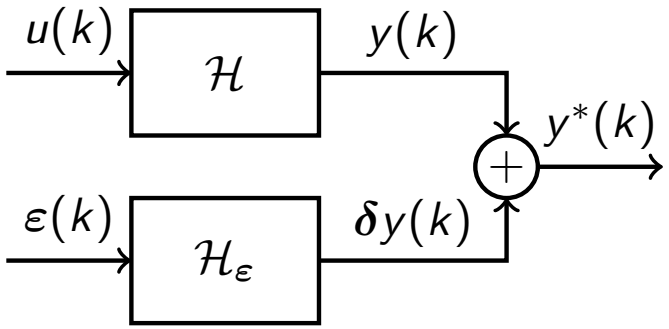
$$\ell_s - \delta > \ell_{c_i} + m_{v_i}, \quad \forall i$$

Contraintes de formatage de bits

Avec les notations de la SIF

$$w_{Z_{i,j}} - w_{v_i} \geq C_{i,j}, \quad \forall i, j.$$

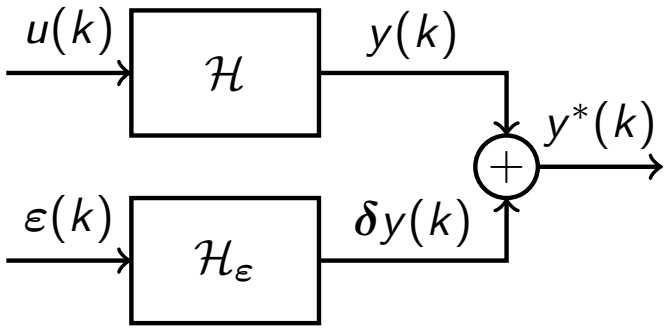
Contraintes - Erreur sur la sortie



On veut borner l'erreur sur la sortie y par ξ :

$$[\underline{\delta y}; \overline{\delta y}] \subset [-\xi; \xi]$$

Contraintes - Erreur sur la sortie



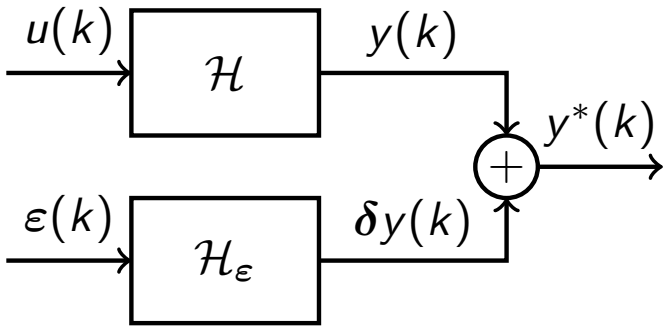
On veut borner l'erreur sur la sortie y par ξ :

$$[\underline{\delta y}; \overline{\delta y}] \subset [-\xi; \xi]$$

$$\left|\underline{\delta y}\right| < \xi$$

$$\left|\overline{\delta y}\right| < \xi$$

Contraintes - Erreur sur la sortie



On veut borner l'erreur sur la sortie y par ξ :

$$[\underline{\delta y}; \overline{\delta y}] \subset [-\xi; \xi]$$

$$\left|\underline{\delta y}\right| < \xi \quad \rightarrow \quad \sum_j \frac{a_j}{2^{w_{v_j}}} < 1$$

$$\left|\overline{\delta y}\right| < \xi \quad \rightarrow \quad \sum_j \frac{d_j}{2^{w_{v_j}}} < 1$$

Contraintes - Bilan

Contraintes de formatage de bits

Avec les notations de la SIF

$$w_{Z_{i,j}} - w_{v_i} \geq C_{i,j}, \quad \forall i, j.$$

Contraintes sur l'erreur

$$\sum_j \frac{a_j}{2^{w_{v_j}}} < 1, \qquad \sum_j \frac{d_j}{2^{w_{v_j}}} < 1.$$

Contraintes - Bilan

Contraintes de formatage de bits

Avec les notations de la SIF

$$w_{Z_{i,j}} \geq w_{v_i} + C_{i,j}, \quad \forall i, j.$$

Contraintes sur l'erreur

$$\sum_j \frac{a_j}{2^{w_{v_j}}} < 1, \qquad \sum_j \frac{d_j}{2^{w_{v_j}}} < 1.$$

Problème d'optimisation

Finalement, le problème se formalise ainsi :

$$\min \quad f(w_v) = \sum_i w_{v_i}$$

$$\text{tel que} \quad \sum_j \frac{a_j}{2^{w_{v_j}}} < 1, \qquad \sum_j \frac{d_j}{2^{w_{v_j}}} < 1.$$

Problème Convexe Non-Linéaire en Nombres Entiers

Problème d'optimisation

Finalement, le problème se formalise ainsi :

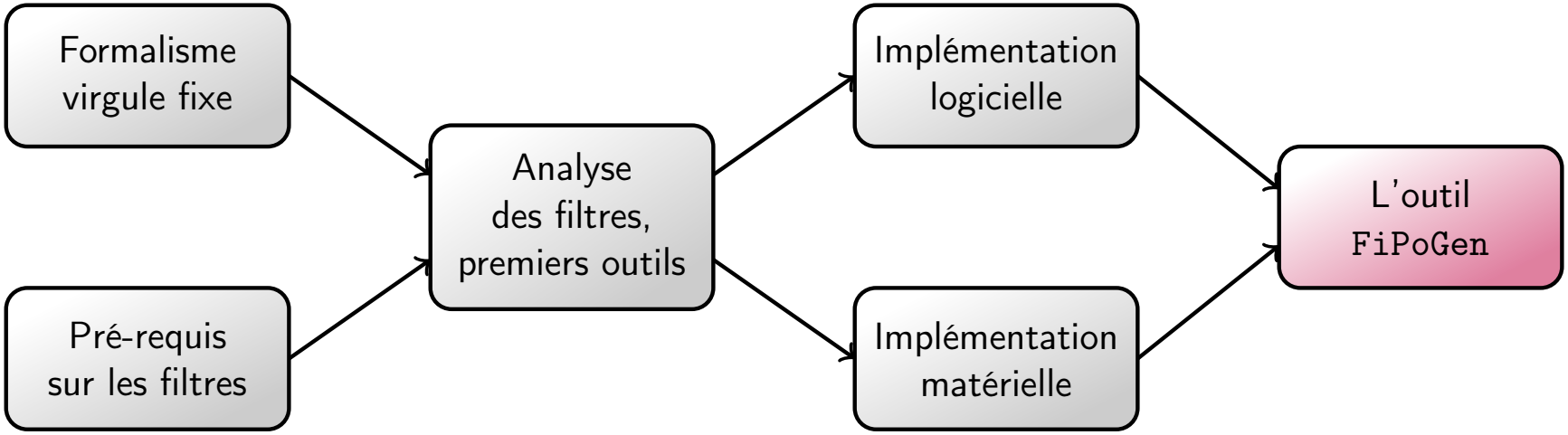
$$\min \quad f(w_v) = \sum_i w_{v_i}$$

$$\text{tel que} \quad \sum_j \frac{a_j}{2^{w_{v_j}}} < 1, \qquad \sum_j \frac{d_j}{2^{w_{v_j}}} < 1.$$

Problème Convexe Non-Linéaire en Nombres Entiers

☞ Résolution par le solveur `Bonmin`.

Plan


Formalisme virgule fixe
Pré-requis sur les filtres
Analyse des filtres, premiers outils
Implémentation logicielle
Implémentation matérielle
L'outil FiPoGen

FiPoGen (*Fixed-Point code Generator*)

Cet outil réalise la méthodologie proposée pour l'implémentation d'un filtre linéaire en virgule fixe.

- Développé en Python
- Actuellement spécialisé dans les algorithmes de filtres linéaires
- Met en œuvre aussi bien la méthodologie logicielle que matérielle

Exemple

L'outil FiPoGen

Soit $\mathcal{H}$ un filtre aléatoire stable généré avec `Matlab` d'ordre 8 et H sa fonction de transfert :

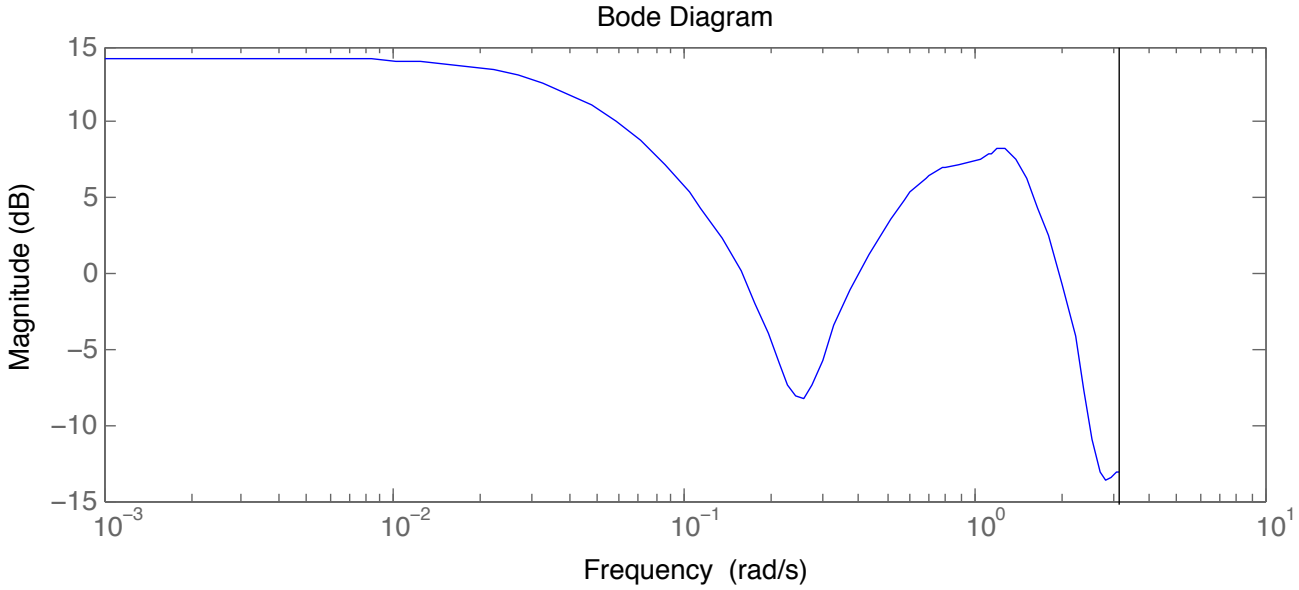
$$H(z) = \frac{\sum_{i=0}^{8} b_i z^{-i}}{1 + \sum_{i=0}^{8} a_i z^{-i}}, \quad \forall z \in \mathbb{C},$$

$$\begin{array}{lclclcl}
b_0 & = & -0.940140, & & & & \\
b_1 & = & 1.04326, & a_1 & = & -2.06925, \\
b_2 & = & 0.292108, & a_2 & = & 1.95404, \\
b_3 & = & -0.693159, & a_3 & = & -1.04616, \\
b_4 & = & 0.183105, & a_4 & = & 0.0898125, \\
b_5 & = & 0.270429, & a_5 & = & 0.172354, \\
b_6 & = & -0.296554, & a_6 & = & -0.0730519, \\
b_7 & = & -0.0311837, & a_7 & = & -0.00696194, \\
b_8 & = & 0.0363021, & a_8 & = & 0.00556424.
\end{array}$$

Exemple

L'outil FiPoGen

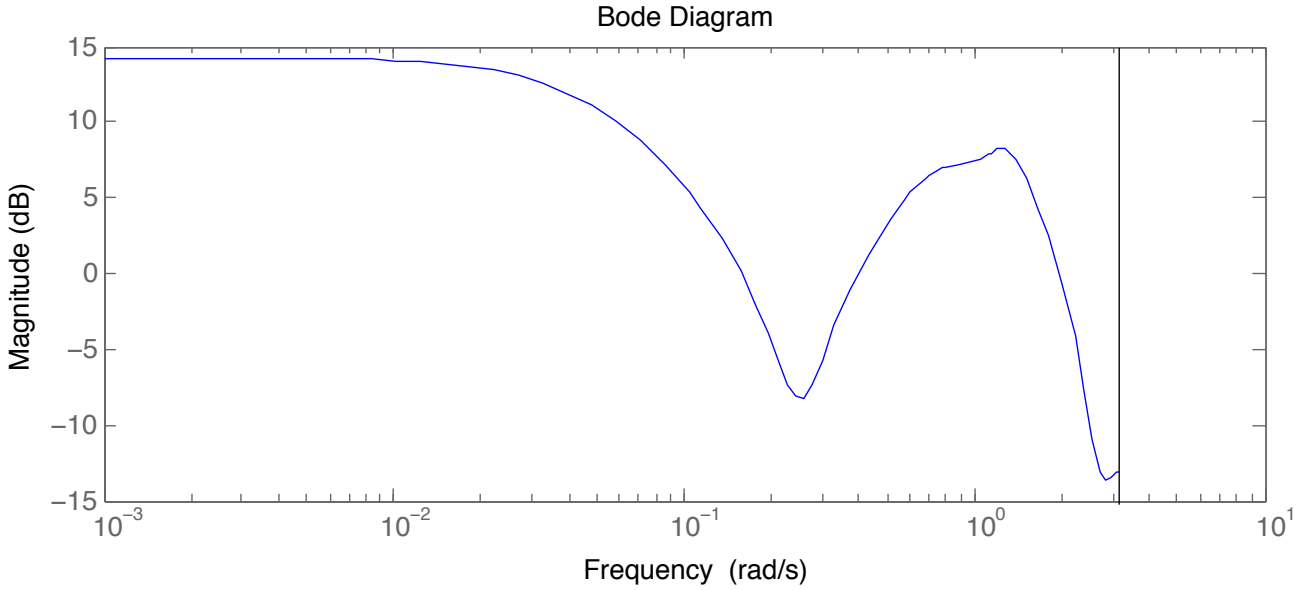
Soit $\mathcal{H}$ un filtre aléatoire stable généré avec `Matlab` d'ordre 8



Exemple

L'outil FiPoGen

Soit $\mathcal{H}$ un filtre aléatoire stable généré avec `Matlab` d'ordre 8



Intervalle de sortie

On considère une entrée aléatoire $u(k) \in [-237; 162] \subset [-256; 255]$.
On obtient la sortie $y(k) \in [-1235.97; 1622.63] \subset [-2048; 2047]$.

Exemple - Plusieurs réalisations

Plusieurs réalisations sont considérées :

- Forme Directe I
  ☞ une seule somme-de-produits de 17 produits
- Forme Directe II transposée avec opérateur ρ
  ☞ une matrice de taille 17×17 creuse
  40 coefficients non triviaux
- Structure LGS
  ☞ une matrice de taille 185×185 creuse
  73 coefficients non triviaux
- State-space minimisant la sensibilité $\mathcal{L}_2$
  ☞ une matrice de taille 9×9 dense

Cas logiciel

On considère les largeurs fixées suivantes

- Largeurs des constantes et des différents types de variables : 16 bits
- Largeurs additionneurs et multiplieurs : 32 bits

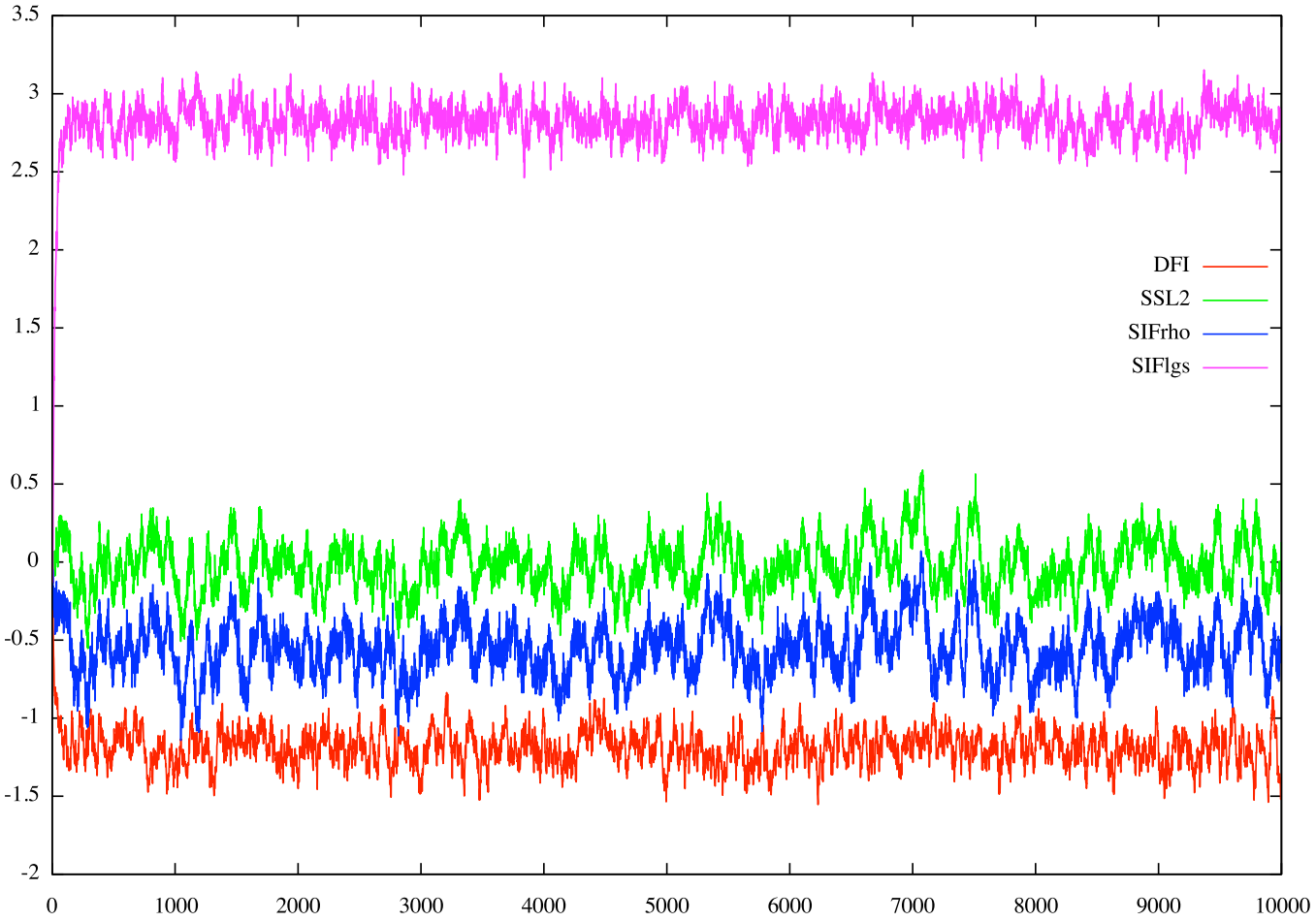
Cas logiciel

On considère les largeurs fixées suivantes

- Largeurs des constantes et des différents types de variables : 16 bits
- Largeurs additionneurs et multiplieurs : 32 bits

Réalisations	Temps de génération	Nombre d'opérations	Intervalles d'erreurs
DFI	–	17 $\times$ et 16 $+$	$[-2.37205; 4.21885e\text{-}15]$
SIF_ρ	$00'00''03$	40 $\times$ et 24 $+$	$[-1.62242; 0.176747]$
SIF_{LGS}	$02'27''61$	73 $\times$ et 34 $+$	$[-0.608507; 5.66752]$
$SS_{\mathcal{L}_2}$	$05'48''87$	81 $\times$ et 72 $+$	$[-1.31393; 0.311331]$

L'outil
FiPoGen



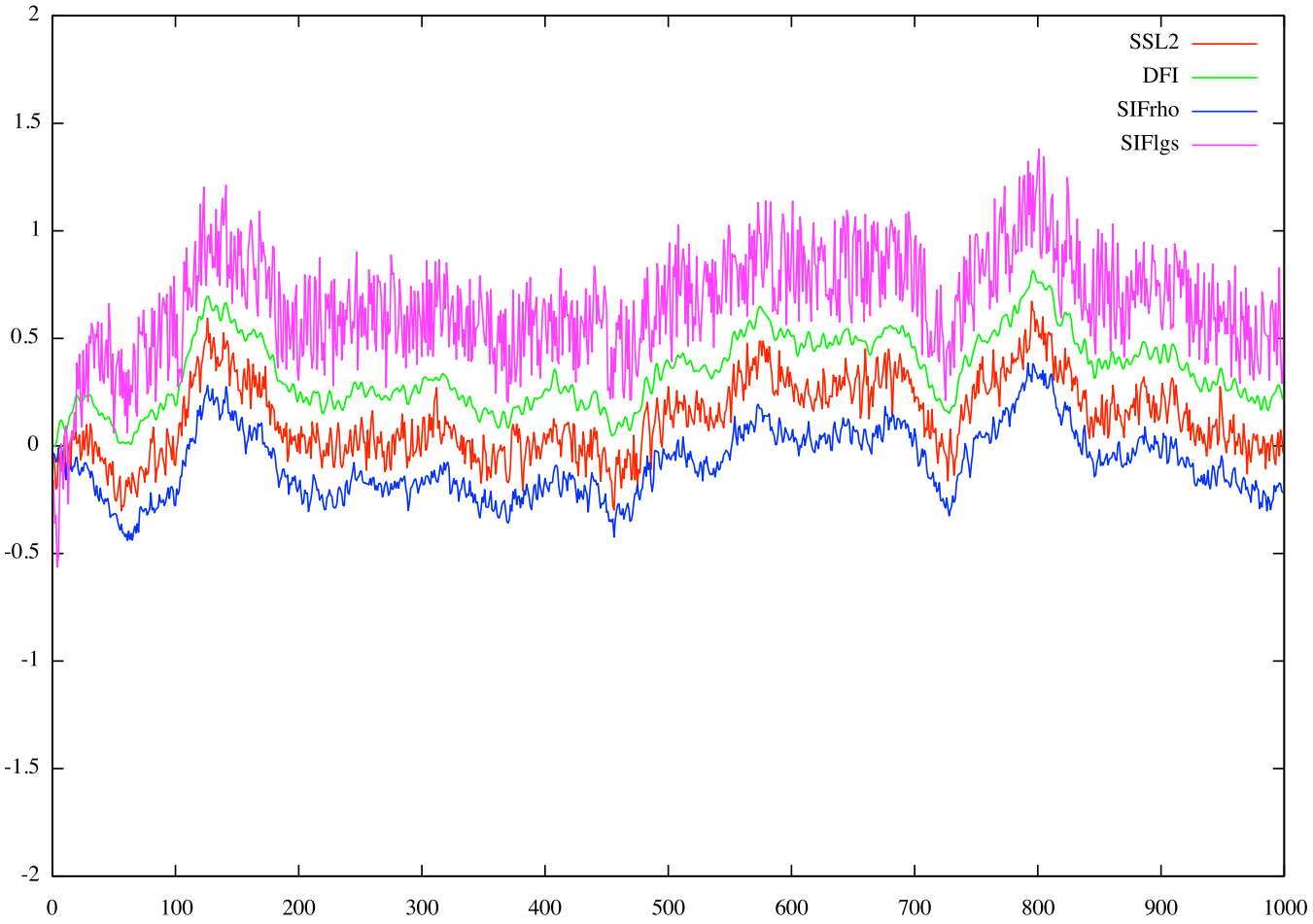
Cas matériel

L'outil FiPoGen

Dans cet exemple, on souhaite borner l'erreur par l'intervalle $[-2; 2]$.

Réalisations	Temps de résolution (sec)	Optimalité	Nombre de bits des variables	Nombre de bits total
DFI	–	–	–	670
SIF_{ρ}	836.15	Oui	293	1025
SIF_{LGS}	600	?	749	1702
$SS_{\mathcal{L}_2}$	0.28	Oui	142	1642

L'outil FiPoGen



Conclusion

Conclusion

Durant cette thèse, j'ai répondu à la question suivante :

Pour un filtre linéaire donné, comment produire l'implémentation virgule fixe *optimale* ?

Plusieurs résultats ont pu être obtenus :

- Formalisation détaillée de l'arithmétique virgule fixe
- Analyse du comportement d'un intervalle à travers un filtre linéaire

Conclusion

Conclusion

Durant cette thèse, j'ai répondu à la question suivante :

Pour un filtre linéaire donné, comment produire l'implémentation virgule fixe *optimale* ?

Plusieurs résultats ont pu être obtenus :

- Formalisation détaillée de l'arithmétique virgule fixe
- Analyse du comportement d'un intervalle à travers un filtre linéaire
- Méthodologie pour les implémentations matérielle et logicielle
- `FiPoGen`, un outil pour générer du code virgule fixe en suivant ces méthodologies

Perspectives

Dans un futur proche :

- Rendre `FiPoGen` totalement autonome du filtre au code
- Fonction de coût plus réaliste : coût d'implémentation, surface, etc.
- Génération des oSoPs plus performante selon les critères de sélection
- Connecter `FiPoGen` à des outils tels que `FloPoCo` ou `Stratus`

Perspectives

Dans un futur proche :

- Rendre `FiPoGen` totalement autonome du filtre au code
- Fonction de coût plus réaliste : coût d'implémentation, surface, etc.
- Génération des oSoPs plus performante selon les critères de sélection
- Connecter `FiPoGen` à des outils tels que `FloPoCo` ou `Stratus`

Dans un second temps :

- Chercher de nouvelles réalisations pour les filtres LTI
- Considérer d'autres filtres linéaires (paramétrés, etc.)
- S'intéresser à d'autres classes d'algorithmes

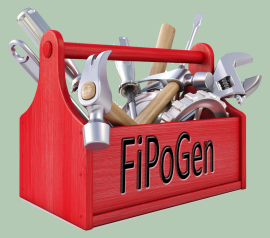
Perspectives

Dans un futur proche :

- Rendre `FiPoGen` totalement autonome du filtre au code
- Fonction de coût plus réaliste : coût d'implémentation, surface, etc.
- Génération des oSoPs plus performante selon les critères de sélection
- Connecter `FiPoGen` à des outils tels que `FloPoCo` ou `Stratus`

Dans un second temps :

- Chercher de nouvelles réalisations pour les filtres LTI
- Considérer d'autres filtres linéaires (paramétrés, etc.)
- S'intéresser à d'autres classes d'algorithmes



Faire de `FiPoGen` un outil générique pour la génération de code virgule fixe, et non uniquement pour les filtres.

Publications

Trois publications dans des conférences avec comité de lecture

- B. Lopez, T. Hilaire, L.-S. Didier. "Sum-of-products evaluation schemes with fixed-point arithmetic, and their application to IIR filter implementation", *DASIP 2012*.
- T. Hilaire, B. Lopez. "Reliable Implementation of Linear Filters with Fixed-Point Arithmetic", *SiPS 2013.*
- B. Lopez, T. Hilaire, L.-S. Didier. "Formatting Bits to Better Implement Signal Processing Algorithms". *PECCS 2014.*

Une Demo Night

- D. Menard *et al.* "Design of fixed-point embedded systems (DEFIS) French ANR project", *DASIP 2012.*

En cours

Une soumission à IEEE Circuits & Systems I est prévue prochainement.

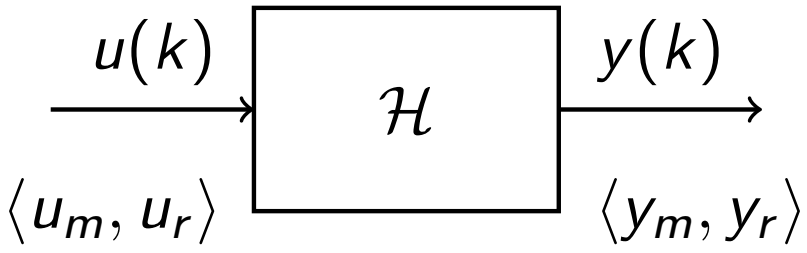
Merci

Annexe

Propriété des filtres linéaires

Analyse des filtres, premiers outils

Nouvelle approche du comportement d'un signal d'entrée u à travers un filtre linéaire $\mathcal{H}$.



Intervalle

u_m	:	centre
u_r	:	rayon
$\langle\langle \mathcal{H} \rangle\rangle_{\mathrm{DC}}$	:	DC-Gain de $\mathcal{H}$
$\langle\langle \mathcal{H} \rangle\rangle_{\mathrm{WCPG}}$	:	WCPG de $\mathcal{H}$

Théorème du WCPG

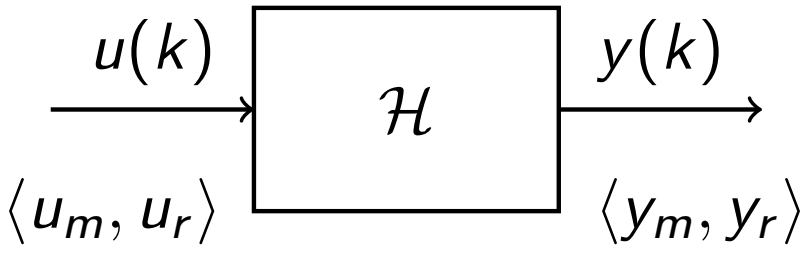
$$\begin{aligned} y_m &= \langle\langle \mathcal{H} \rangle\rangle_{\mathrm{DC}} \cdot u_m \\ y_r &= \langle\langle \mathcal{H} \rangle\rangle_{\mathrm{WCPG}} \cdot u_r \end{aligned}$$

en régime permanent

Propriété des filtres linéaires

Analyse des filtres, premiers outils

Nouvelle approche du comportement d'un signal d'entrée u à travers un filtre linéaire $\mathcal{H}$.



Intervalle

u_m	:	centre
u_r	:	rayon
$\langle\langle \mathcal{H} \rangle\rangle_{\mathrm{DC}}$	:	DC-Gain de $\mathcal{H}$
$\langle\langle \mathcal{H} \rangle\rangle_{\mathrm{WCPG}}$	:	WCPG de $\mathcal{H}$

Théorème du WCPG

$$\begin{aligned} y_m &= \langle\langle \mathcal{H} \rangle\rangle_{\mathrm{DC}} \cdot u_m \\ y_r &= \langle\langle \mathcal{H} \rangle\rangle_{\mathrm{WCPG}} \cdot u_r + \varepsilon \cdot u_m \end{aligned}$$

$\exists K$ tel que $\forall k > K,\ y(k) \in \langle y_m, y_r \rangle$