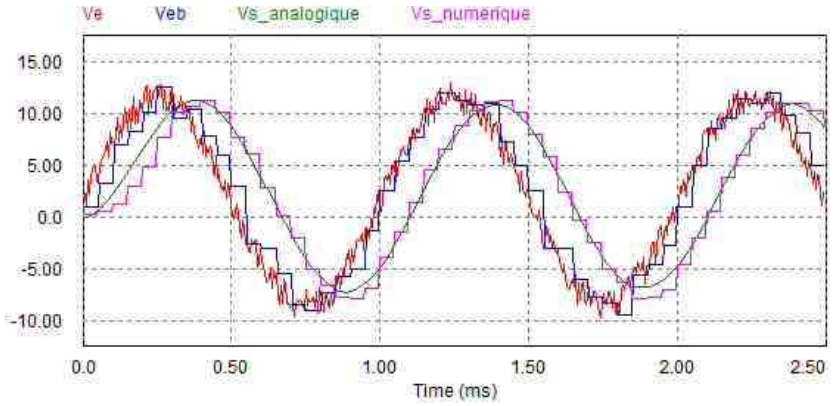
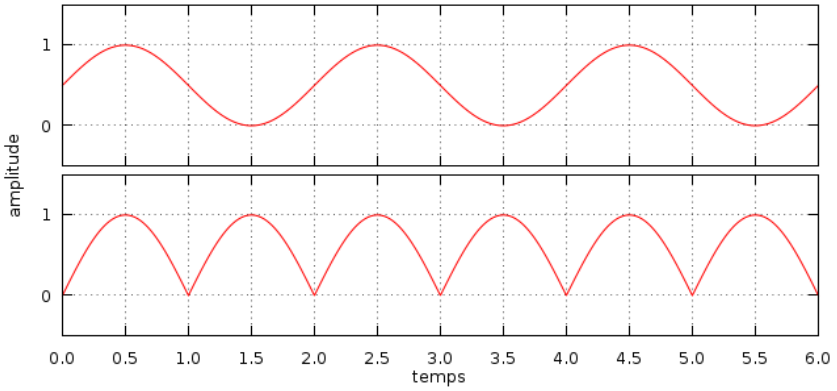
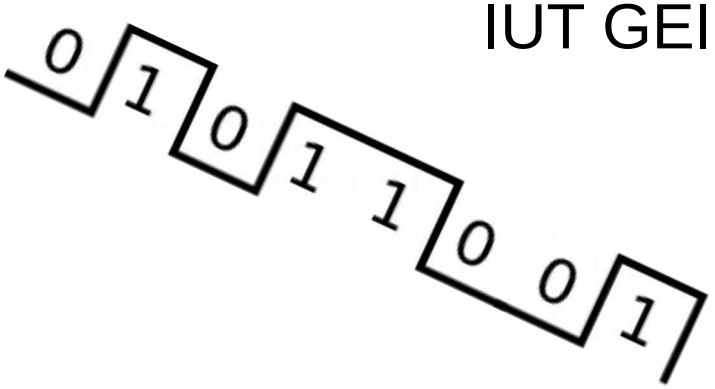




Architecture pour le Traitement Numérique du Signal (TNS)

Bertrand Vandeportaele, Hugues Gilliard
IUT GEII TOULOUSE
2018



Objectifs

- Savoir décrire une chaîne de traitement du signal (modélisation système)
- Savoir choisir/utiliser une représentation des nombres et mettre en œuvre les opérateurs associés
- Savoir implémenter un filtrage d'un signal monodimensionnel (1D) par logiciel (gestion des données, structure algorithmique, organisation du calcul)
- Savoir séquencer temporellement le calcul et garantir des contraintes temps-réel
- Disposer de quelques éléments de dimensionnement pour le choix d'une architecture

Déroulement

- 1 séances de cours (1x2h)
  - Aspects logiciels et calculatoire
- 2 séances de TD (2x2h)
  - Préparation des TP
- 7 séances de TP (7x2h)
  - Mise en œuvre sur PC pour traitement hors ligne en virgule flottante
  - Mise en œuvre sur Micro contrôleur pour traitement en ligne en virgule flottante (plateforme Nucleo)

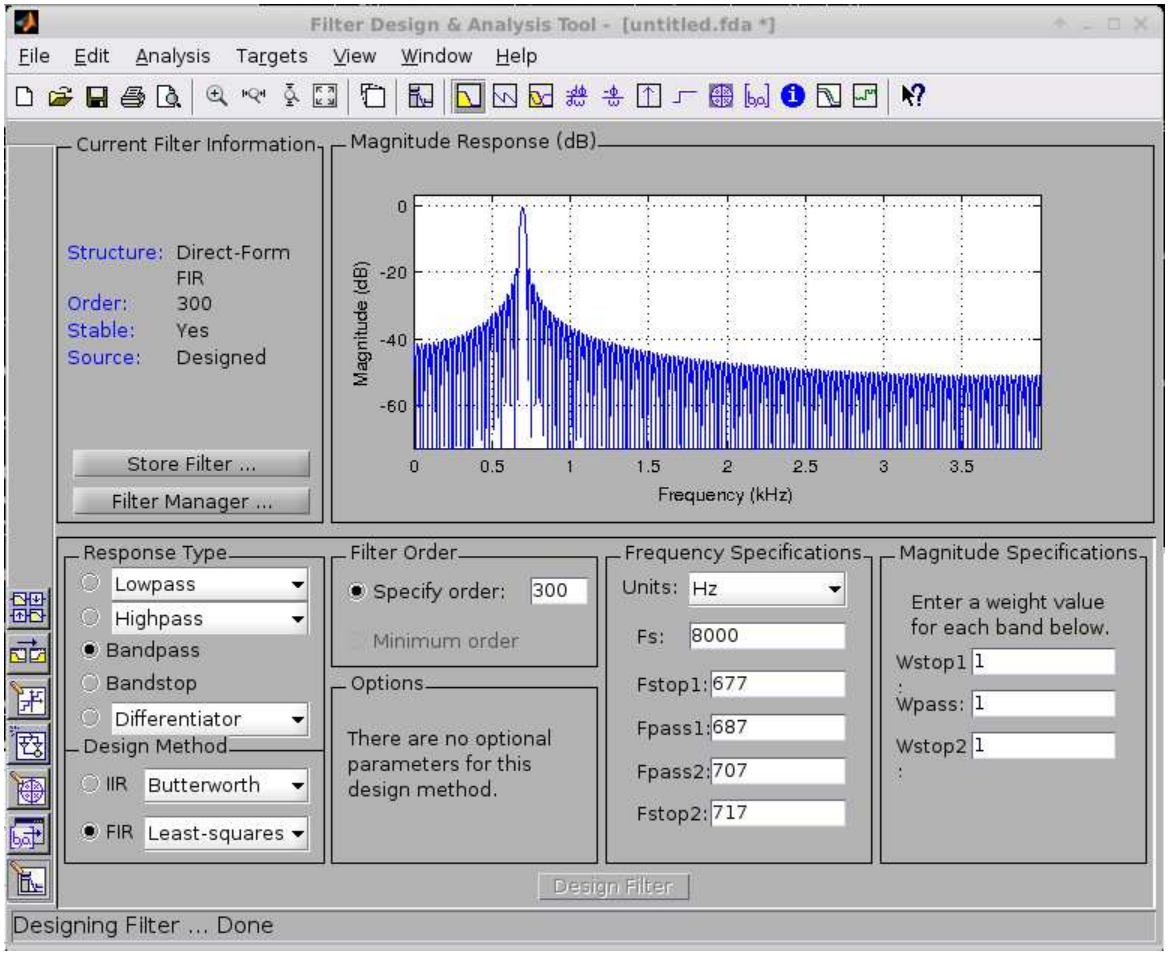
Plan du cours

- Introduction
- Rappels sur les filtres numériques RII et RIF
- Présentation d'une chaîne générique de traitement numérique du signal
- Représentation des données
- Traitement du signal par logiciel

Introduction

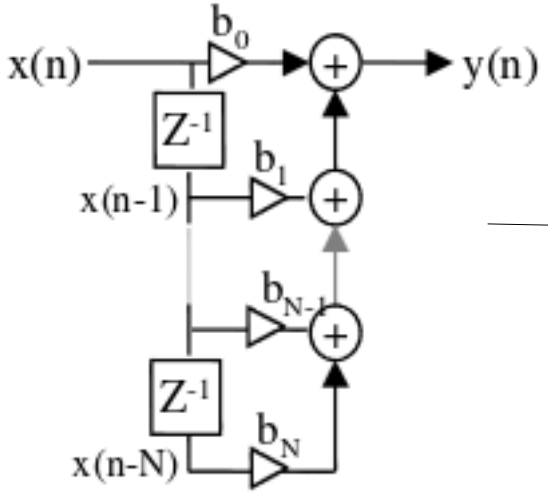
- L'objectif de ce cours n'est pas la **conception** du filtre

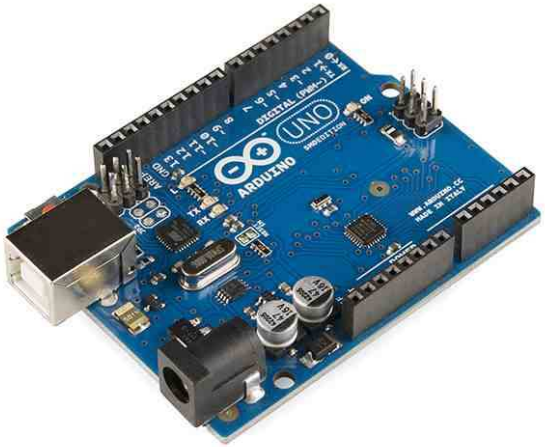
Exemple de conception avec un outils matlab :



Introduction

- L'objectif de ce cours n'est pas la **conception** du filtre mais sa **réalisation**
- Permet de passer du schéma de la structure du filtre à une réalisation réelle





- Démarche de développement utilisant des **tests** comparatifs

•Rappels sur les filtres numériques RIF

- Réponse Impulsionnelle Finie
  - Plusieurs structures pour un même filtre
    - Structure directe : les coefficients multiplicateurs sont les coefficients du filtre
    - Filtre d'ordre N :
      - N+1 multiplications de 2 opérandes
      - N additions de 2 opérandes
      - N valeurs à mémoriser

$$s_k = \sum_{i=0}^{N} b_i \cdot e_{k-i}$$

  - Stabilité
    - La sortie ne diverge jamais
  - Données nécessaires pour le calcul d'un échantillon :
    - Valeurs précédentes et courante des entrées (et les coefficients du filtre)
  - La réponse impulsionnelle h(k) correspond aux coefficients de l'équation de récurrence

•Rappels sur les filtres numériques RII

- Réponse Impulsionnelle Infinie
  - Plusieurs structures pour un même filtre
    - Filtre d'ordre M (ordre du dénominateur) et N (ordre du numérateur) :
      - (M)+(N+1) multiplications de 2 opérandes
      - M+N additions de 2 opérandes
    - Structure directe 1: les coefficients multiplicateurs sont les coefficients du filtre
      - Données nécessaires pour le calcul d'un échantillon :
        - Valeurs précédentes et courante des entrées ek
        - Valeurs précédentes des sorties sk
        - N+M valeurs à mémoriser

$$s_k = \sum_{i=0}^{N} b_i \cdot e_{k-i} - \sum_{j=1}^{M} a_j \cdot s_{k-j}$$

    - Structure directe 2: les coefficients multiplicateurs sont les coefficients du filtre
      - Données nécessaires pour le calcul d'un échantillon :
        - Valeur courante de l'entrée ek
        - Valeurs précédentes et courante de termes intermédiaires vk
        - max(N,M) valeurs à mémoriser

$$s_k = \sum_{i=0}^{N} b_i \cdot v_{k-i}$$

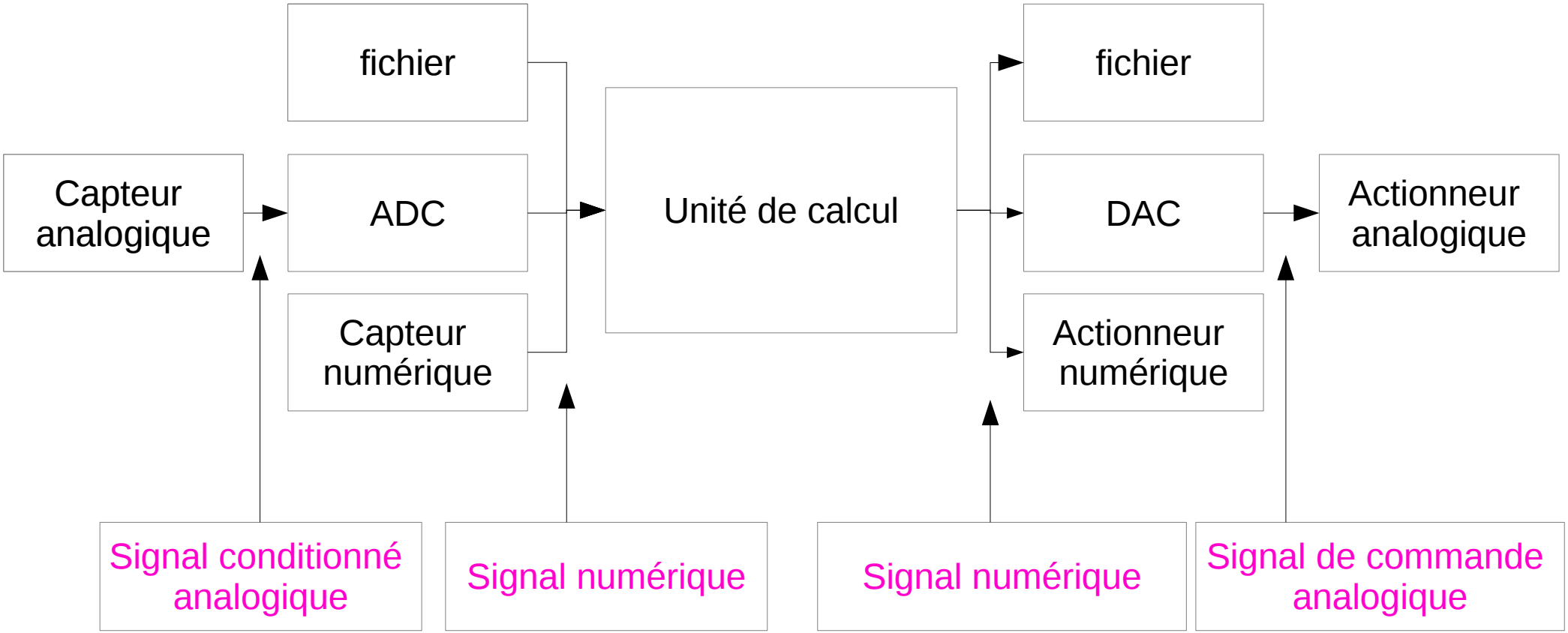
$$v_k = e_k - \sum_{j=1}^{M} a_j \cdot v_{k-j}$$

  - Stabilité
    - La sortie peut diverger

•Rappels sur les filtres numériques

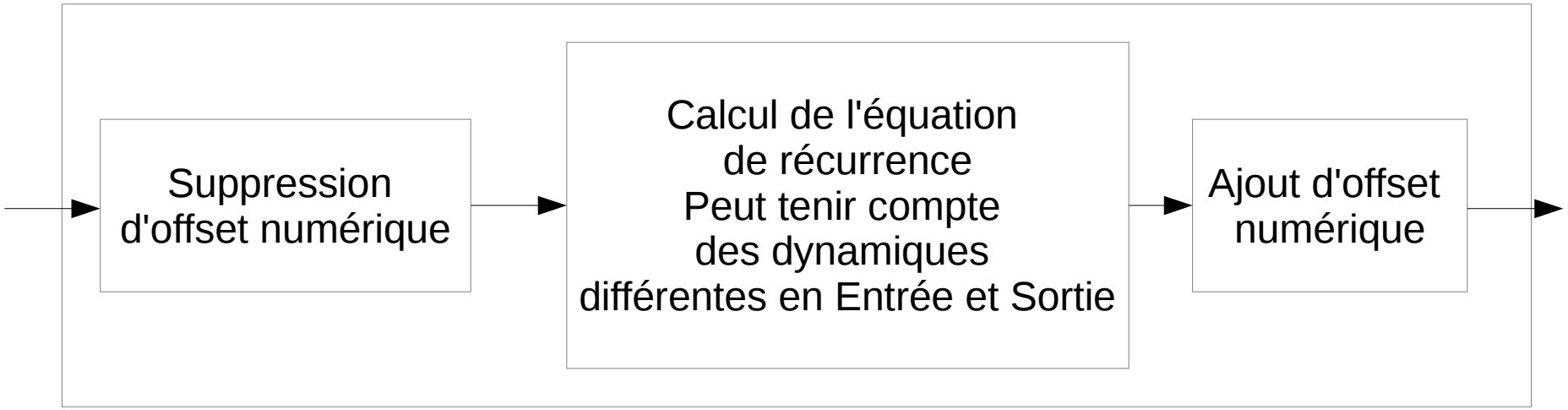
- Structure canonique si le nombre de délais dans le filtre est égal à l'ordre de la fonction de transfert
- On considère que le filtre numérique est dimensionné pour traiter des échantillons référencés par rapport à une valeur donnée qui est 0
  - Mise en place d'ajout et de suppression d'offset numérique

Chaîne générique de TNS



Chaîne générique de TNS

- Détail de la fonction de Calcul



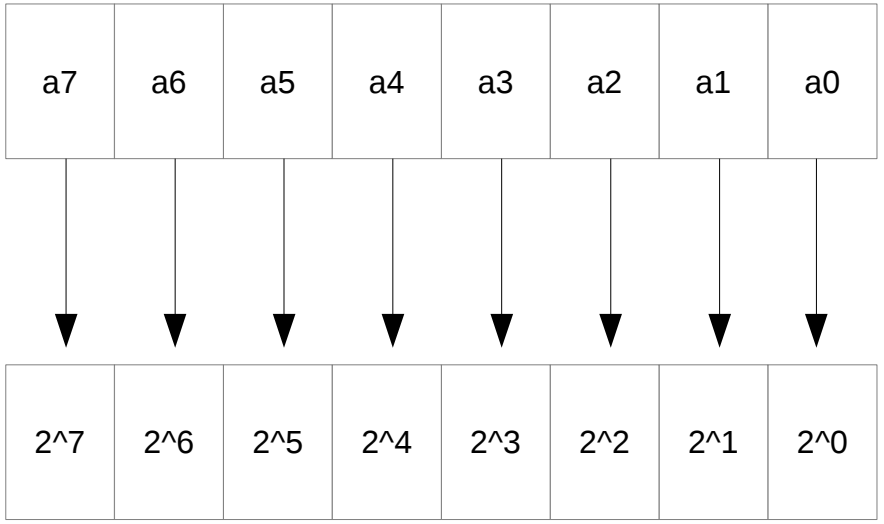
Chaîne générique de TNS

- Différentier chaîne matérielle et traitement logiciel hors ligne
  - Contraintes différentes (fréquence, latence, accès au données)
- Convertisseurs ADC / DAC
  - Caractéristiques adaptées à l'application (nombre de bits, fréquence d'échantillonnage...)
- Différents types d'unité de calcul
  - Microprocesseurs généralistes
  - Microprocesseurs dédiés (DSP)
  - Architectures reconfigurables (FPGA)
  - Circuits dédiées (ASIC)

Représentation des données

- Codage binaire sur les entiers non signés
  - Chaque bit du nombre est associé à un poids (exemple octet):

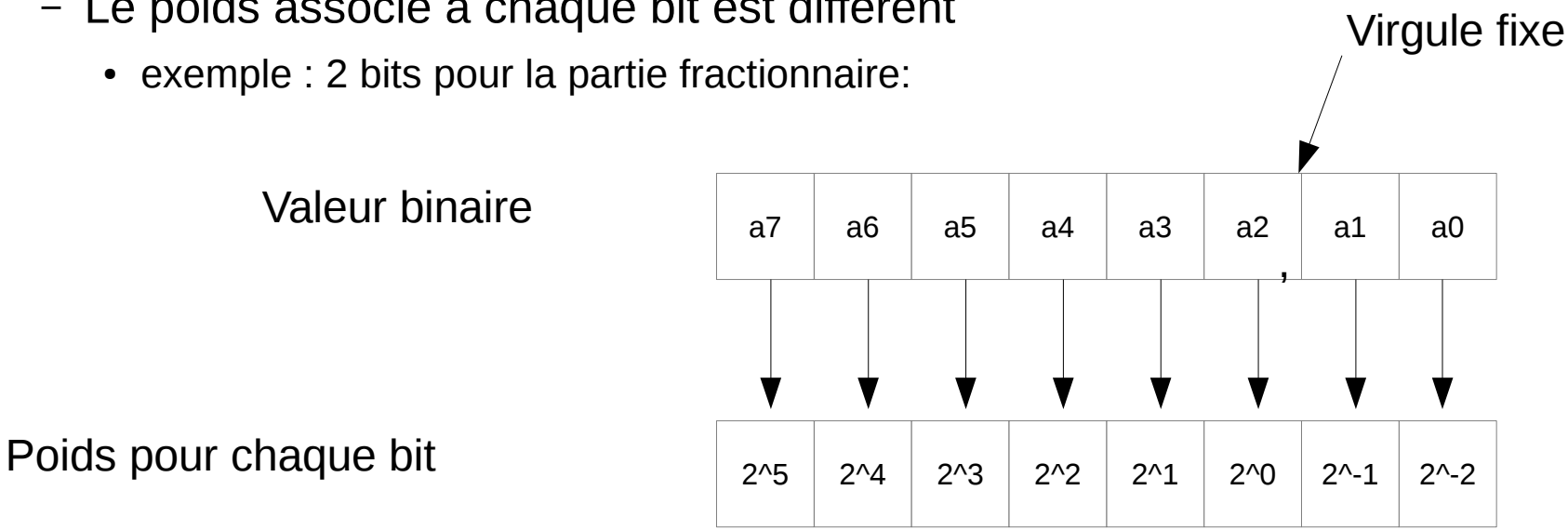
Valeur binaire



Poids pour chaque bit

Virgule fixe non signé (unsigned fixed-point)

- Représentation en virgule fixe non signés:
  - Le poids associé à chaque bit est différent
    - exemple : 2 bits pour la partie fractionnaire:



- Plusieurs notations, mais nous utiliserons:
  - **UQm.f** m pour le nombre de bits de la partie entière
    f pour le nombre de bits de la partie fractionnaire

Virgule fixe signé (signed fixed-point)

- Utilise le complément à deux de la valeur absolue pour coder les nombres négatifs
  - Le bit de poids fort indique le signe (1 → négatif)
- Plusieurs notations, mais nous utiliserons:
  - **Qm.f** m pour le nombre de bits de la partie entière
    f pour le nombre de bits de la partie fractionnaire
    1 bit implicite pour coder le signe
  - Se code sur m+f+1 bits
  - Exemples :
    - 16bits dont 4 pour la partie fractionnaire : Q11.4
    - 16bits en signed int : Q15.0
    - 16bits avec 15 bits pour la partie fractionnaire Q0.15 (intérêt : le produit de deux Q0.15 peut se coder sur un Q0.15 si on exclus -1*-1 et qu'on arrondi le résultat)
- Autres notations : http://en.wikipedia.org/wiki/Fixed-point_arithmetic#Notation
- Valeurs min : $-2^{(m-1)}$ max : $2^{(m-1)}-\frac{1}{2^f}$ Précision : $\frac{1}{2^f}$

Virgule fixe (fixed-point)

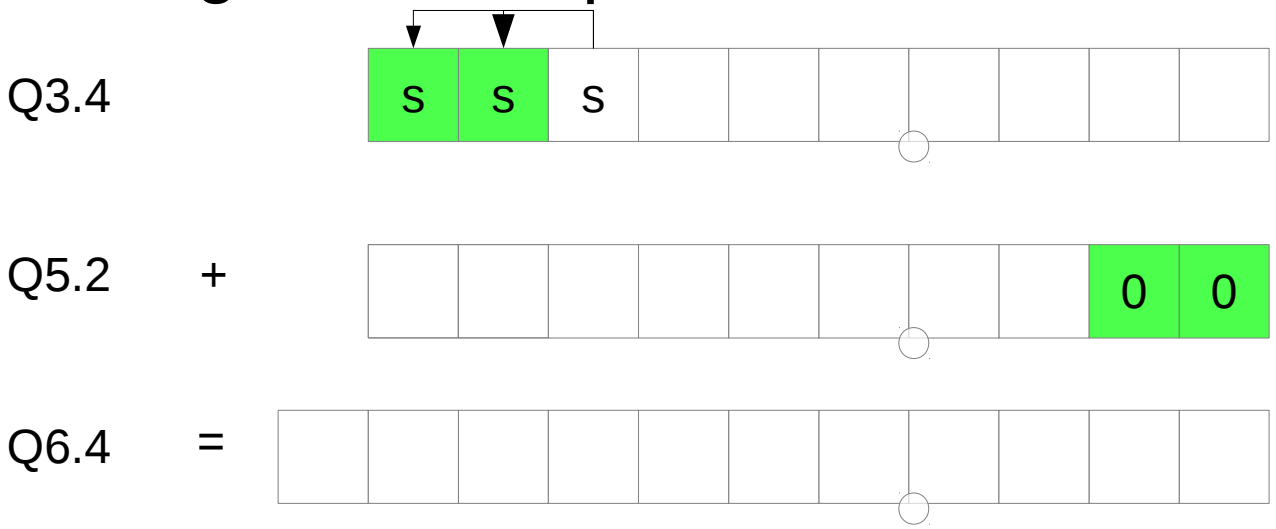
- Avantages :
  - Les calculs sur les opérandes en virgule fixe utilisent les mêmes opérateurs que les nombres entiers :
    - Vitesse, coût...
  - Représentation sans erreur des nombres entiers codables sur m bits
  - Interprétation aisée des valeurs
  - Précision prédictible :
    - le pas est constant entre 2 valeurs
- Problèmes & inconvénients
  - Dimensionnement nécessaire
    - En fonction des valeurs que l'on souhaite coder (pour des entrées par exemple)
    - En fonction des résultats que l'on souhaite obtenir (pour des sorties, ou des calculs intermédiaires)
  - Le résultat d'une opération sur 2 opérandes n'est pas forcement codable sur le même nombre de bits que les opérandes (débordement)
- A utiliser si on ne dispose pas d'une unité matérielle de calcul sur les flottants et que les contraintes de temps de calcul sont fortes.

Virgule fixe (fixed-point)

- Dimensionnement
  - On considère arbitrairement que les données échangées avec les ADC/DAC sur n bits sont à interpréter au format UQn.0
  - Choix des formats le long de la chaîne de traitement
  - Troncatures en milieu et/ou fin de traitement

Règles de calculs en Virgule fixe

- Qa.b + Qc.d a pour taille Q(max(a,c)+1).(max(b,d)) si on veut garder la précision de l'opérande le plus précis
- Addition de 2 nombres de formats différents
  - Aligner les opérandes en ajoutant des 0 avant de faire l'addition sur les poids faibles et propager le bit de signe sur les poids forts

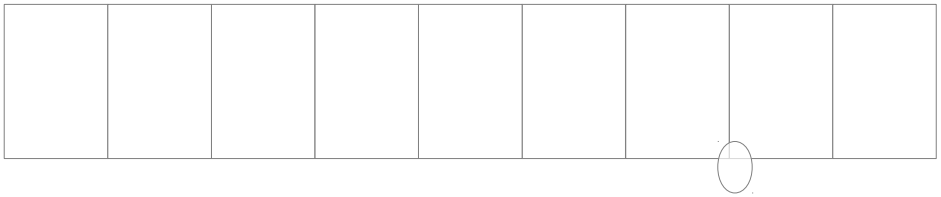


-

Règles de calculs en Virgule fixe

- Troncature d'un résultat
  - Ne pas tenir compte des bits de poids faible

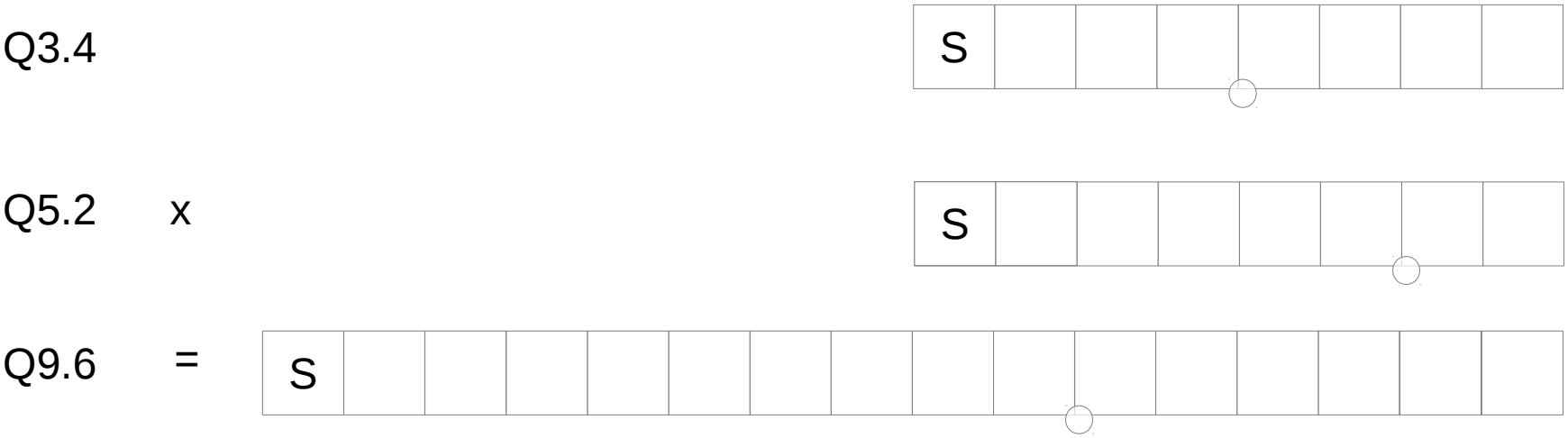
Q6.2 =



  - Utilisation des opérateurs de décalage binaire
  - Peut se faire à priori (avant le calcul du résultat) dans certains cas
- Arrondi d'un résultat
  - Tient compte du premier bit de poids faible éliminé
    - Si ce bit est à 1, ajouter 1 au résultat tronqué

Règles de calculs en Virgule fixe

- Qa.b x Qc.d a pour taille Q(a+c+1).(b+d)
  - Q0.b x Q0.d a pour taille Q1.(b+d)
    - Dans le cas où les 2 opérandes Q0.b et Q0.d sont <1 en module, le résultat se code en Q0.(b+d)
- Multiplication de 2 nombres de formats différents : $(n.2^{-b}).(m.2^{-d})=(n.m).2^{-(b+d)}$



Dimensionnement en Virgule fixe

- Dimension au pire cas du résultats de N multiplications et (N-1) additions sur format

  Qa.b x Qc.d = Q(a+c+N-1).(b+d)

- Potentiellement très grand (exemple RIF à 300 coefficients)
- Bornage haut des valeurs en sortie pour un RIF

$$b_i \cdot e_{k-i} \leqslant |b_i \cdot e_{k-i}| \leqslant |b_i| \cdot e_{max} \qquad \sum_{i=0}^{N} b_i \cdot e_{k-i} \leqslant e_{max} \cdot \sum_{i=0}^{N} |b_i|$$

Virgule flottante (floating-point)

- Représentation en virgule flottante:
  - Différents standards
  - Une norme : IEEE 754
  - Les 2 standards les plus courants (source Wikipedia)

Précision	Encodage	Signe	*Exposant*	*Mantisse*	Valeur d'un nombre	Précision	Chiffres significatifs
Simple précision	32 bits	1 bit	8 bits	23 bits	$(-1)^S \times M \times 2^{(E-127)}$	24 bits	environ 7
Double précision	64 bits	1 bit	11 bits	52 bits	$(-1)^S \times M \times 2^{(E-1023)}$	53 bits	environ 16

Attention, le M de cette table n'est pas la mantisse codée !

  - Démonstration : http://www.h-schmidt.net/FloatConverter/IEEE754.html

Virgule flottante IEEE754 32 bits

31	30.....23	22.....0
Bit de signe	Exposant codé	Mantisse codée
1 bit	8 bits	23 bits

- valeur = signe × mantisse réelle × 2^(exposant réel)
- Signe sur 1 bit
- Exposant codé sur 8bits non signé
  - Exposant réel = exposant codé− 127
  - exposant codé = 0 → mantisse non normalisée au format UQ1.22 (2>valeur mantisse réelle>=0)
  - 0<exposant codé<255 → mantisse normalisée au format UQ1.23 avec bit pour la partie entière =1 (2>valeur mantisse réelle>=1)
  - exposant codé = 255 → mantisse indique un code spécial (+/- Inf, Nan...)
- Mantisse codée sur les 23 bits restants en non signé
  - Mantisse réelle =mantisse codée (non normalisée) si exposant codé=0
  - Mantisse réelle =1 concaténé avec mantisse codée (normalisée) sinon
    - bit de poids fort (24ieme) implicitement à 1 sans qu'il ne soit codé sur la valeur 23 bits
  - Code spécial si exposant codé=255

Virgule flottante 32 bits : exemples

- -Inf : 0xff800000
- +Inf : 0x7f800000
- Nan : 0x7f800001
- 0 n'est pas normalisé, le bit de poids fort n'est pas implicitement à 1 (avec exposant codé=0)

  +0 : 0x00000000

  -0 : 0x80000000
- Plus petit nombre non nul avec mantisse dénormalisée (avec exposant codé=0)

  1.4E-45 : 0x00000001
- Plus grand nombre avec mantisse dénormalisée (avec exposant codé=0)

  1.1754942E-38 : 0x007fffff
- Plus petit nombre avec mantisse normalisée (avec exposant codé=1)

  1.17549435E-38 : 0x00800000
- Plus grand nombre avec mantisse normalisée (avec exposant codé=254)

  3.4028235E38 : 0x7f7fffff

  1.0 : 0x3f800000

  -1.0 : 0xbf800000

Virgule flottante (floating-point)

- Peut être implémenté en logiciel (émulation par calculs sur des entiers) ou en matériel
- Avantages :
  - Possibilité de représenter un intervalle de valeur plus grand qu'en virgule fixe à nombre de bits équivalent
  - Retardement des phénomènes de débordement
  - Le résultat d'une opération sur 2 opérandes est codable sur le même nombre de bits que les opérandes (sauf débordement)
  - Adaptation de la précision du codage à l'ordre de grandeur du nombre
  - Dans certains cas, on peut « faire confiance » aux calculs en flottants (surtout en double précision...)
  - Intègre la représentation de +/- Infini, Not A Number...

Virgule flottante (floating-point)

- Problèmes & inconvénients
  - Complexité accrue des calculs
    - Éventuellement palliée par la présence d'une unité matérielle de calcul sur les flottants (FPU : Floating Point Unit)
  - Interprétation non aisée des valeurs binaires
  - Précision difficilement prédictible :
    - le pas n'est pas constant entre 2 valeurs successives
  - Non commutativité des calculs :
    - a.(b.c)!=(a.b).c
  - Dimensionnement nécessaire
    - Choix demi/simple/double précision
  - Problème en cas de **mauvaise utilisation** : par exemple **l'absorption** d'un incrément sur un compteur
- A utiliser si on dispose de temps de calcul et/ou d'une unité matérielle de calcul sur les flottants

Traitement du signal par logiciel

- Utilisation d'un processeur pour réaliser les calculs
- Itération de calcul répétée soit :
  - à la fréquence d'acquisition/restitution des échantillons (traitement en ligne)
  - aussi vite que possible (traitement hors ligne)
- Étapes d'une itération:
  - Acquisition de ek
    - Lecture depuis un ADC, un fichier....
  - Gestion du stockage des échantillons (commutation ou buffer circulaire)
  - Calcul numérique de l'équation de récurrence
  - Utilisation de sk
    - Écriture sur un DAC, un fichier....

Traitement du signal par logiciel

- Architecture du processeur capable de manipuler en une instruction des entiers sur:
  - 8bits : 80C51, ATMEGA328
  - 16bits : C167
  - 32bits : PIC32,Nucleo (STM32)
  - 64bits : processeurs de PC x64
  - + ?
- En TP, nous utiliserons une carte Nucleo avec FPU pour traiter des échantillons codé en flottant simple précision

Traitement du signal par logiciel

- Gestion du temps pour traitement en ligne
  - Utilisation d'un timer matériel
  - Fonctionnement par interruption matérielle pour avoir la maîtrise du temps
  - Le temps de calcul du « pire cas » doit tenir dans le temps disponible entre deux interruptions
  - Utilisation d'une mesure du temps
    - Tic/toc ou broche GPIO
  - Entrelacement des étapes acquisition/calcul/restitution si les périphériques d'acquisition/restitution sont lents

Traitement du signal par logiciel

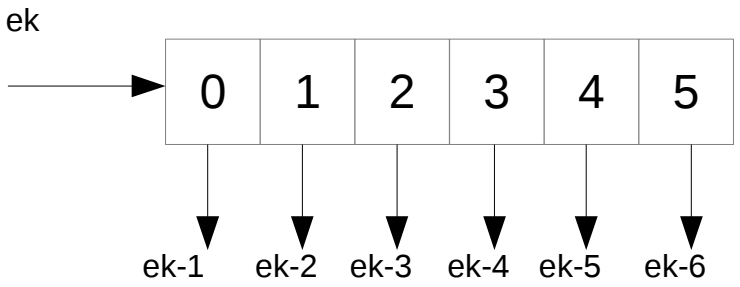
- Selon le filtre et la forme selon laquelle on l'écrit, il faut 1 ou 2 zones de stockage pour mémoriser les échantillons

$$s_k = \sum_{i=0}^{N} b_i \cdot e_{k-i} - \sum_{j=1}^{M} a_j \cdot s_{k-j}$$

$$s_k = \sum_{i=0}^{N} b_i \cdot v_{k-i} \qquad\qquad v_k = e_k - \sum_{j=1}^{M} a_j \cdot v_{k-j}$$

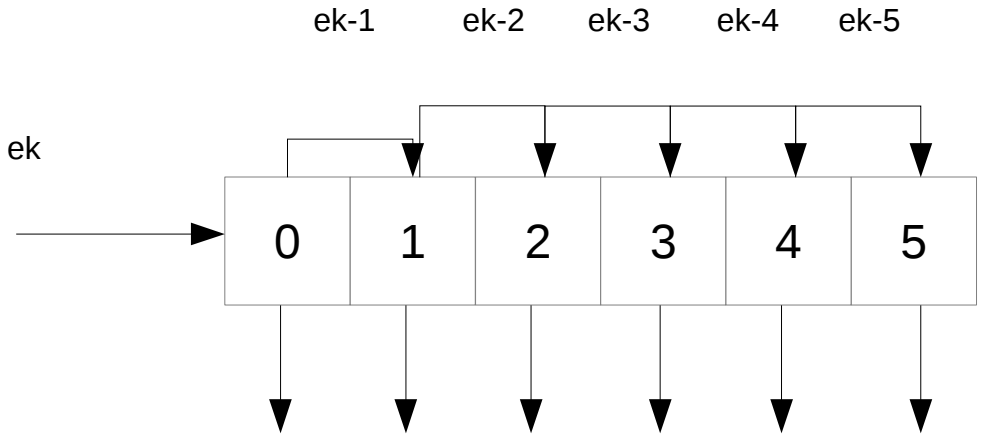
Traitement du signal par logiciel

- Gestion du stockage des échantillons
- Objectif : faire rentrer l'échantillon ek dans le buffer et avoir accès aux échantillons précédents
- Utilisation d'un tableau pour stocker des échantillons :
  - Implémentation naïve, l’indice de la case est relié à la date de l’échantillon :



Traitement du signal par logiciel

- Gestion par décalage des échantillons
  - L'échantillon ek entre toujours au même indice
  - Les données sont recopiées une à une vers la case adjacente

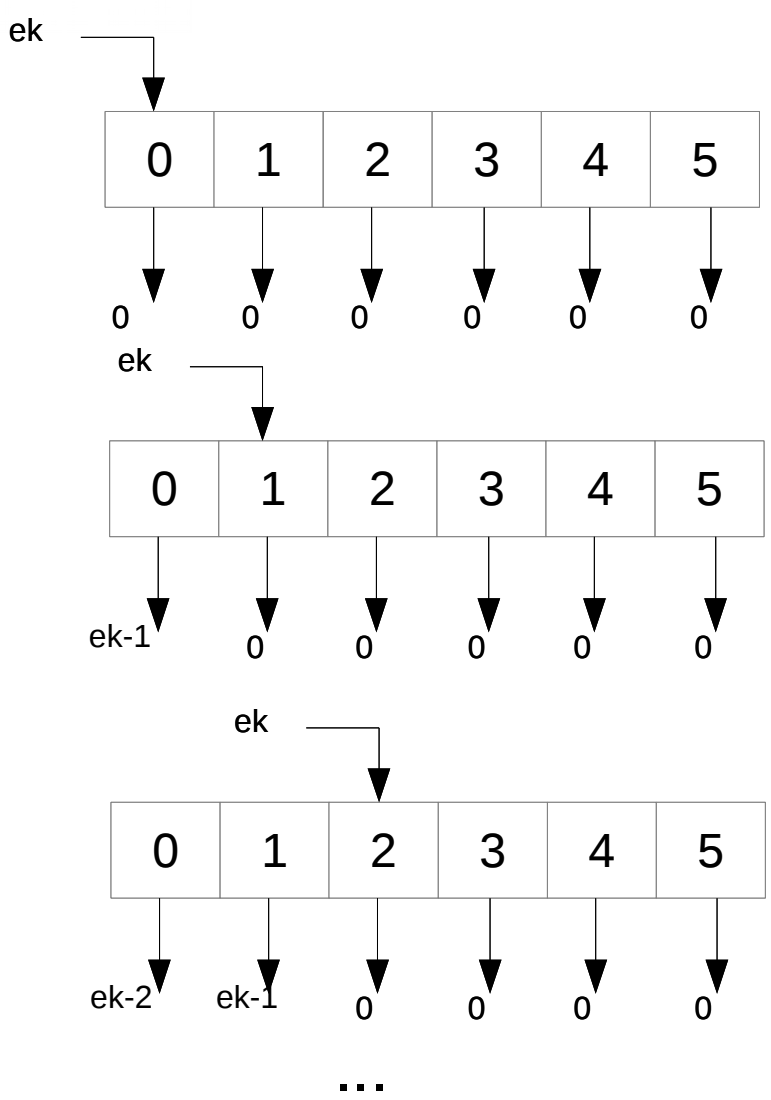


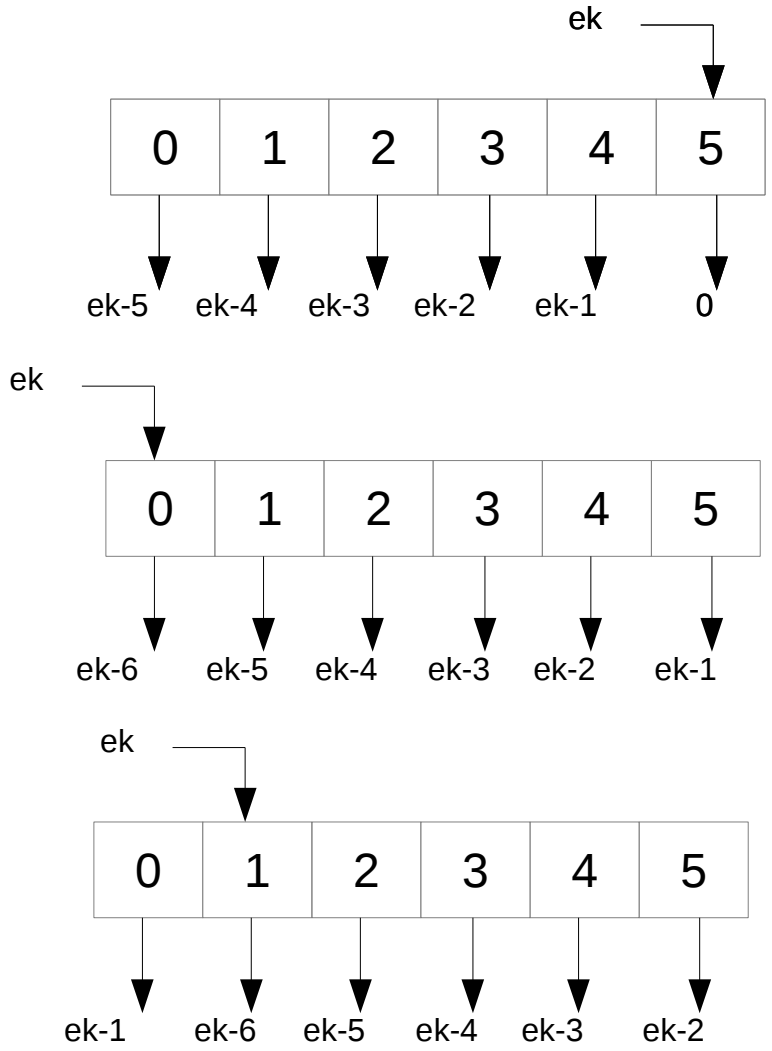
Très inefficace si grand nombre d'échantillons à stocker

Traitement du signal par logiciel

- Gestion par buffer circulaire
  - L'échantillon ek entre à un indice modifié à chaque itération (avec gestion des limites si sortie du tableau) : utilisation d'un **indice d'écriture**
  - L'interprétation des indices des cases du tableau dépend de l'indice d'écriture
- Avantage : Les échantillons stockés n'ont pas à être recopiés
- C’est l’approche que nous utiliserons en TP

Traitement du signal par logiciel





Démarche proposée pour développer

- Objectif : Réalisation d'un filtre en virgule flottante pour micro contrôleur
- Conception du filtre déjà réalisée
- Implémentation et test sur PC hors ligne
  - Échantillons connus en entrée
  - Application des fonctions de calculs en double précision pour obtenir des résultats servant de « vérité terrain » sur les données disponibles en entrée
  - Codage des structures de données permettant la gestion des échantillons (ex : buffer circulaire)
  - Comparaison des résultats entre une version fournie par les enseignants et une version développée par l'étudiant
- Test du comportement du filtre hors ligne
  - Génération de signaux de test en entrée et analyse des sorties
- Portage des composants logiciels sur le micro contrôleur
  - Gestion du temps (par interruption)
  - Gestion des entrées/sorties (ADC/DAC/GPIO)
    - Test simple : recopie des entrées sur les sorties à la fréquence d'échantillonnage
  - Utilisation directe des fonctions de calcul déjà testées sur PC
- Test du comportement du système complet (GBF+Oscilloscope/analyseur spectral)

Démarche proposée pour développer

- Fonction de calcul avec interface unifiée
  - Permet le remplacement transparent de la fonction de filtrage
  - Permet l'utilisation de cette fonction pour le développement/test sur PC et sur micro contrôleur
  - Utilisation de l’approche objet
    - La classe filtre contient tout ce qui est nécessaire à son fonctionnement