

# Introduction au traitement du signal

## Cours 2 : Conversion analogique/numérique

Laurent Oudre  
laurent.oudre@univ-paris13.fr

Université Paris 13, Institut Galilée  
Ecole d'ingénieurs Sup Galilée  
Parcours Télécommunications et Réseaux - 1<sup>ère</sup> année  
2018-2019

## Sommaire

### Signaux analogiques - signaux numériques

## Sommaire

### 1. Signaux analogiques - signaux numériques

#### 2. Échantillonnage

- 2.1 Échantillonnage uniforme
- 2.2 Reconstruction d'un signal échantillonné
- 2.3 Critère de Nyquist

#### 3. Quantification

- 3.1 Quantification uniforme
- 3.2 Erreur de quantification

## Signal analogique

- ▶ Nous avons vu que la plupart des signaux du monde réel (ondes, son, électricité, etc...) sont continus et ne peuvent pas être stockés et analysés sur un ordinateur.
- ▶ Pourquoi cela ? Prenons l'exemple d'un son d'une durée de 5 secondes et correspondant à la note *la* (celui du diapason).

$$x(t) = \sin(2\pi f_0 t) \quad t \in [0, 5[ \text{ avec } f_0 = 440 \text{ Hz}$$

- ▶ Le nombre de temps pour lequel le signal est défini est infini ! (même si le support temporel est ici borné)
- ▶ Les valeurs du signal sont réelles et contiennent une infinité de décimales  
Ex :  $x(0.001) = 0.368124552684678\dots$
- ▶ Tout ici est infini : le nombre de valeurs à stocker, mais aussi les valeurs elles-mêmes. On appelle un tel signal un **signal analogique**

## Signal numérique

- ▶ Pour qu'un signal puisse être stocké sur un ordinateur il faut :
  - ▶ Que le nombre d'échantillons qu'il contient soit fini : le signal doit être discret et à support temporel fini
  - ▶ Que le nombre de possibilités pour les valeurs du signal soit fini : chaque valeur du signal doit pouvoir être codée sur un nombre fini de bits
- ▶ Il faut donc un nombre fini de valeurs à stocker et un nombre fini de valeurs possibles. On appelle un tel signal un **signal numérique**

## Exemple

Supposons qu'on veuille stocker en mémoire le signal analogique suivant :

$$x(t) = \sin(2\pi f_0 t) \quad t \in [0, 5[ \text{ avec } f_0 = 440 \text{ Hz}$$

1. On va uniquement observer le signal à des instants  $t[n]$ . On prend une valeur toutes les  $10^{-3}$  secondes.

$$t[n] = 10^{-3}n \quad n \in \llbracket 0, 4999 \rrbracket \rightarrow 5000 \text{ échantillons}$$

2. On va coder chaque valeur  $x[n] = x(t[n])$  sur 4 bits. Comme on sait que les valeurs sont comprises entre -1 et 1, on définit  $2^4 = 16$  valeurs possibles de la façon suivante :  $-\frac{15}{16}, -\frac{13}{16}, \dots, -\frac{1}{16}, \frac{1}{16}, \dots, \frac{15}{16}$ . Chaque valeur est associée à un code binaire comportant 4 bits.

$$x[10] = x(10 \times 10^{-3}) = x(0.01) = 0.587785252292471... \rightarrow 0.5625 \rightarrow 1101$$

3. En tout, ce signal sera représenté numériquement sur

$$5000 \times 4 = 20000 \text{ bits} \approx 2.44 \text{ Ko}$$

## Conversion analogique/numérique

Pour pouvoir convertir un signal analogique en un signal numérique, il faut :

- ▶ **Échantillonnage.** Choisir un ensemble fini de  $N$  temps  $t[n]$  où l'on va stocker les valeurs (conversion continu vers discret).

$$x[n] = x(t[n]) \text{ où } t[n] \text{ représentent les temps où le signal va être enregistré}$$

$$n \in \llbracket 0, N - 1 \rrbracket$$

- ▶ **Quantification.** Choisir un ensemble fini de valeurs possibles et stocker en mémoire la valeur la plus proche de la valeur observée.

## Conversion analogique/numérique

Bilan :

- ▶ En entrée, un signal analogique physique du monde réel
- ▶ En sortie, après échantillonnage et quantification, un vecteur binaire composé de 0 et 1 stockable et analysable par ordinateur

Exemples :

- ▶ Enregistrement d'un son
- ▶ Enregistrement d'une photo numérique

## Sommaire

## Échantillonnage

- 2.1 Échantillonnage uniforme
- 2.2 Reconstruction d'un signal échantillonné
- 2.3 Critère de Nyquist

## Attention !

- ▶ Attention à ne pas confondre la notion de période/fréquence fondamentale ( $T_0$  et  $f_0$ ) qui est définie pour un signal périodique et la notion de période/fréquence d'échantillonnage ( $T_e$  et  $F_e$ ) !
- ▶ La fréquence d'échantillonnage  $F_e$  correspond au nombre d'échantillons qui seront enregistrés en 1 seconde.
- ▶ Dans certains cours, on note parfois  $F_e$  de façon différente (par exemple  $f_e$ ,  $f_s$  ou  $F_s$ ). L'indice  $s$  correspond au mot *sampling* en anglais qui veut dire *échantillonnage*

## Qu'est-ce que l'échantillonnage ?

- ▶ Principe : Convertir un signal continu en un signal discret en ne stockant que ce qui se passe à certains instants  $t[n]$

$$x[n] = x(t[n])$$

- ▶ On ne va considérer ici que l'échantillonnage uniforme, c'est à dire qu'on prend une valeur toutes les  $T_e$  secondes, où  $T_e$  est fixe

$$t[n] = nT_e = \frac{n}{F_e}$$

- ▶  $T_e$  est appelée la **période d'échantillonnage** (en secondes)
- ▶  $F_e = \frac{1}{T_e}$  est appelée la **fréquence d'échantillonnage** (en Hertz)

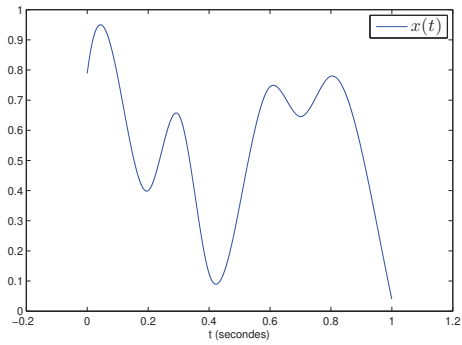
## Echantillonnage uniforme

- ▶ Si on souhaite échantillonner un signal avec une fréquence d'échantillonnage  $F_e$ , on stocke ceci :

Echantillon	Temps	Valeur stockée
$n$	$t[n]$	$x[n]$
0	0	$x(0)$
1	$T_e$	$x(T_e)$
2	$2T_e$	$x(2T_e)$
3	$3T_e$	$x(3T_e)$
$\vdots$	$\vdots$	$\vdots$

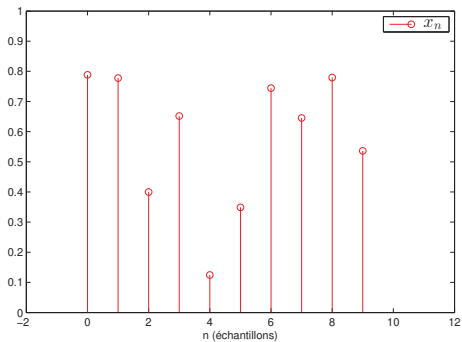
- ▶ Moyen mnémotechnique : une seconde de signal correspond à  $F_e$  échantillons
- ▶ Si on considère un signal d'une durée de  $d$  secondes, il faut donc prévoir de stocker  $d \times F_e$  échantillons (plus éventuellement les temps correspondant dans un vecteur temps).

## Exemple



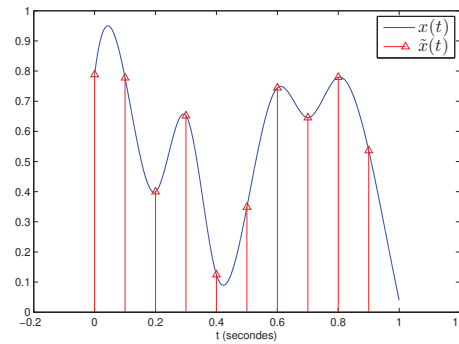
- ▶ Signal continu  $x(t)$  défini sur  $t \in [0, 1[$

## Exemple



- ▶ On range chaque valeur
 
$$x(t[n]) = x(nT_e) = x\left(\frac{n}{F_e}\right)$$
 dans un vecteur (ou un tableau)
- ▶  $x[n] = x(t[n])$  avec  $n \in \llbracket 0, 9 \rrbracket$
- ▶ Le signal est stocké sur  $N = 10$  échantillons

## Exemple



- ▶ On prend une valeur toutes les 0.1 secondes en commençant par  $t = 0$  et en s'arrêtant à  $t = 0.9$  :

- ▶  $T_e = 0.1$  secondes
- ▶  $F_e = 10$  Hz

- ▶ Temps  $t[n]$  définis par

$$t[n] = nT_e = \frac{n}{F_e} \text{ pour } n \in \llbracket 0, 9 \rrbracket$$

$$t_0 = 0, t_1 = 0.1, t_2 = 0.2, \dots$$

- ▶ Le signal continu obtenu peut s'écrire :

$$\begin{aligned} \tilde{x}(t) &= \sum_{n=0}^9 x(nT_e) \delta(t - nT_e) \\ &= x(t) \times \sum_{n=0}^9 \delta(t - nT_e) \end{aligned}$$

## Limites de l'échantillonnage : approche qualitative

- ▶ Intuitivement si  $T_e$  est trop grand (donc  $F_e$  trop petite), on va perdre de l'information.
- ▶ En particulier, si le signal  $x(t)$  varie très rapidement, si l'on veut garder toute l'information, il va falloir prendre une fréquence d'échantillonnage très élevée
- ▶ A l'inverse, si le signal  $x(t)$  varie lentement, on n'aura pas besoin de prendre beaucoup de points

→ Comment choisir la fréquence d'échantillonnage ?

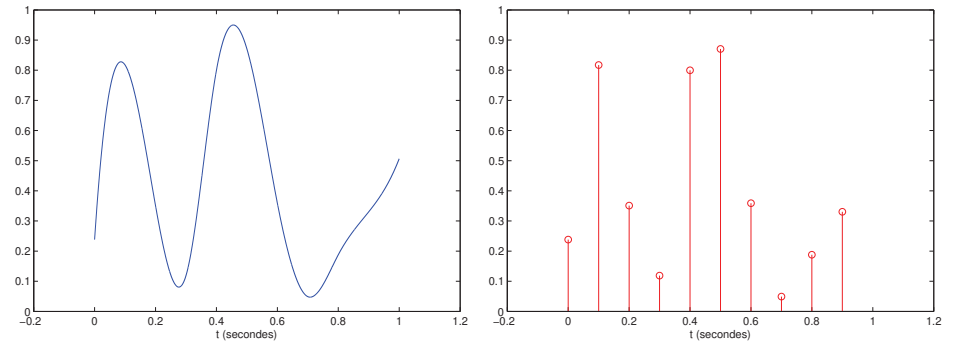
## Reconstruction d'un signal échantillonné

Pour tester l'intuition que nous avons sur le choix de la fréquence d'échantillonnage, nous allons considérer l'expérience suivante :

- ▶ Prendre un signal continu et l'échantillonner
- ▶ Essayer de reconstruire le signal continu à partir des échantillons
- ▶ Voir si le signal reconstruit est éloigné ou pas du signal original

→ Si la fréquence d'échantillonnage a été correctement choisie, le signal reconstruit devrait être proche du signal original.

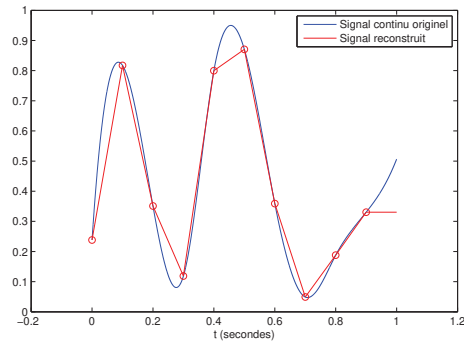
## Exemple 1



Le signal étudié ici varie de façon lente. On échantillonne le signal à  $F_e = 10$  Hz et on va essayer de reconstruire le signal à partir des seuls échantillons. Comment faire ?

## Exemple 1

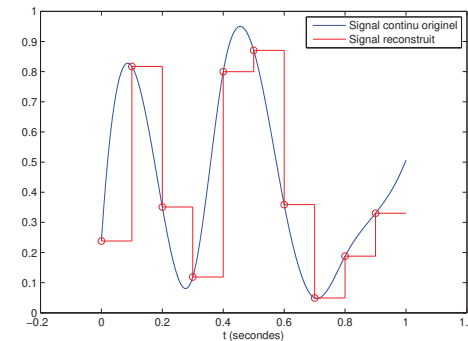
Idee 1 : On va relier les points entre eux ! (Bloqueur d'ordre 1)



Problème : il faut calculer chaque équation de droite et faire des interpolations qui peuvent être coûteuses en temps de calcul

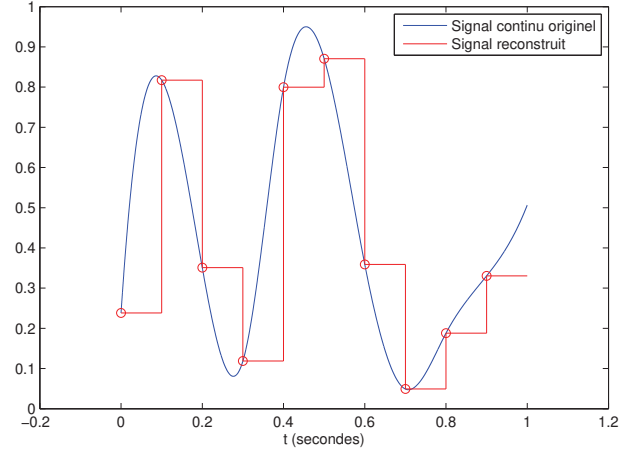
## Exemple 1

Idee 2 : On va dire que tant que l'on a pas un nouvel échantillon, le signal reste constant (Bloqueur d'ordre 0)



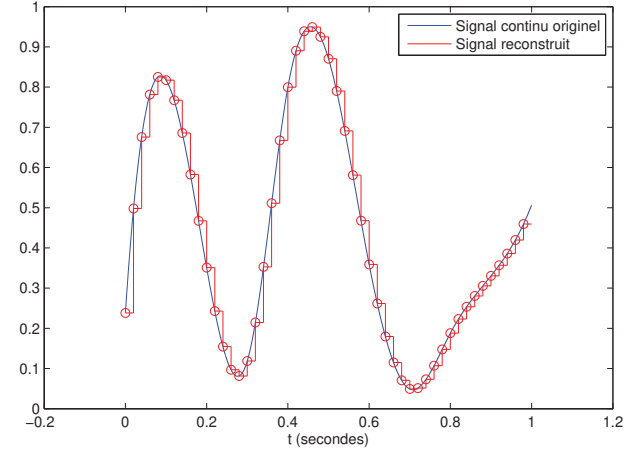
Pas extrêmement précis, mais très facile à implémenter

## Exemple 1 : bloqueur d'ordre 0



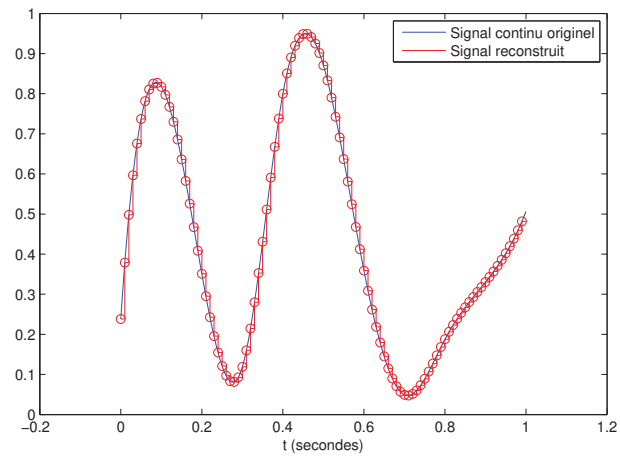
$F_e = 10 \text{ Hz}$ ,  $T_e = 0.1 \text{ secondes}$ . Reconstruction très grossière.

## Exemple 1 : bloqueur d'ordre 0



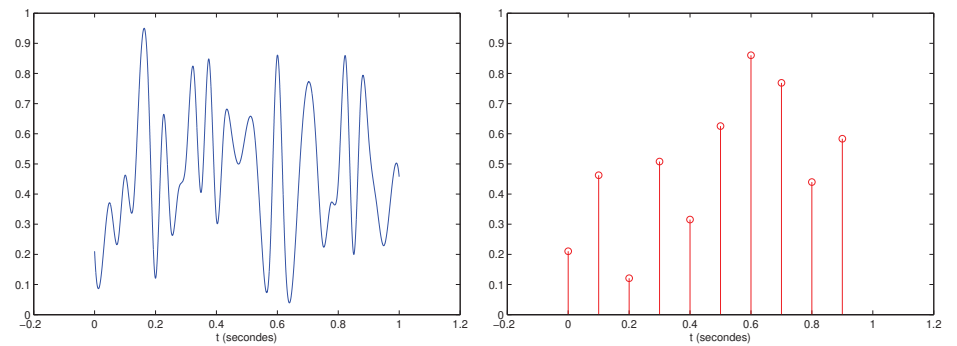
$F_e = 50 \text{ Hz}$ ,  $T_e = 0.02 \text{ secondes}$ . Reconstruction encore imparfaite.

## Exemple 1 : bloqueur d'ordre 0



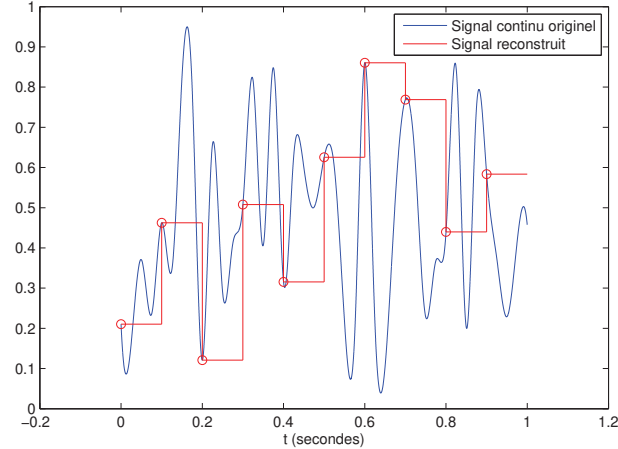
$F_e = 100 \text{ Hz}$ ,  $T_e = 0.01 \text{ secondes}$ . Reconstruction de qualité très acceptable.

## Exemple 2



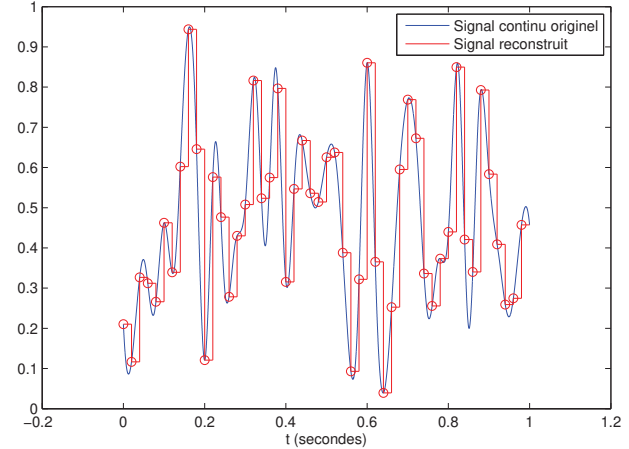
Cette fois-ci le signal étudié ici varie de façon rapide. Cela va-t-il influencer la reconstruction du signal ?

## Exemple 2 : bloqueur d'ordre 0



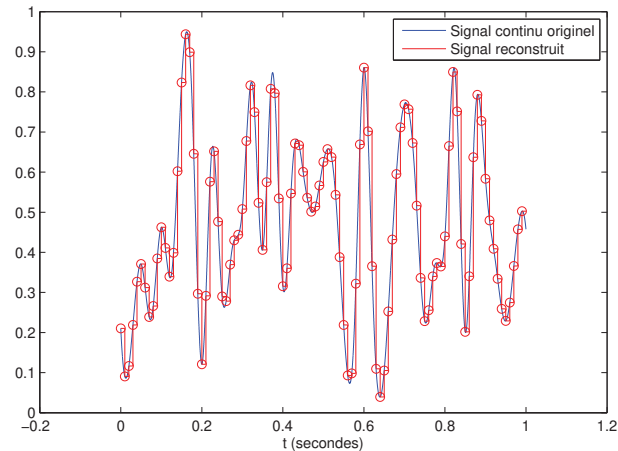
$F_e = 10$  Hz,  $T_e = 0.1$  secondes. Reconstruction très grossière.

## Exemple 2 : bloqueur d'ordre 0



$F_e = 50$  Hz,  $T_e = 0.02$  secondes. Reconstruction de mauvaise qualité.

## Exemple 2 : bloqueur d'ordre 0



$F_e = 100$  Hz,  $T_e = 0.01$  secondes. Reconstruction toujours trop approximative surtout dans les zones à variations rapides.

## Limites de l'échantillonnage : approche quantitative

- ▶ Nous avons validé par des exemples que la fréquence d'échantillonnage à utiliser pour ne pas perdre trop d'information dépend du signal que l'on veut échantillonner. Y a-t-il une formule permettant de savoir quelle fréquence d'échantillonnage utiliser ?
- ▶ Nous allons considérer ici le cas simple d'une sinusoïde  $x(t)$  de fréquence fondamentale  $f_0$ . Plus  $f_0$  est grand, plus le signal est rapide. Plus  $f_0$  est petit, plus le signal est lent.
- ▶ Nous allons tester plusieurs fréquences d'échantillonnage et voir quelle est la valeur minimale à utiliser pour ne pas perdre d'information.

## Critère de Nyquist : cas d'une sinusoïde

- ▶ Sinusoïde originelle :

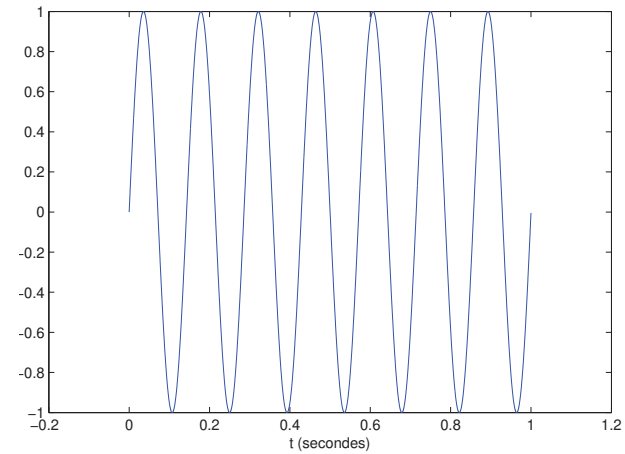
$$x(t) = \sin(2\pi f_0 t) \quad t \in [0, 1[ \text{ avec } f_0 = 7 \text{ Hz}$$

- ▶ Signal échantillonné avec  $N = F_e$  échantillons (on suppose pour simplifier que  $F_e$  est un entier) :

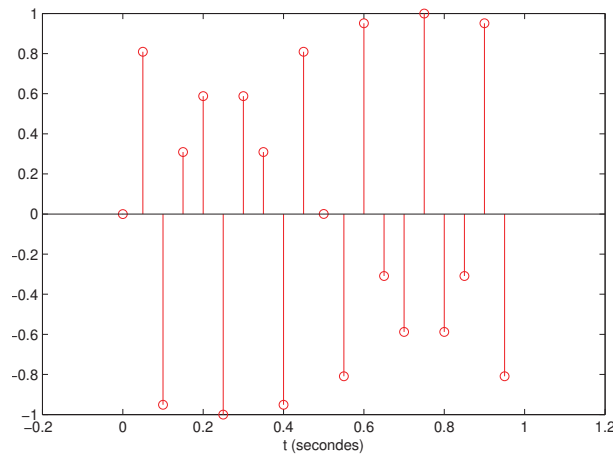
$$x[n] = \sin\left(2\pi f_0 \frac{n}{F_e}\right) \quad \text{avec } n \in \llbracket 0, N - 1 \rrbracket$$

- ▶ Nous allons tester plusieurs fréquences d'échantillonnage, et tenter de reconstruire une sinusoïde à partir des points échantillonnés

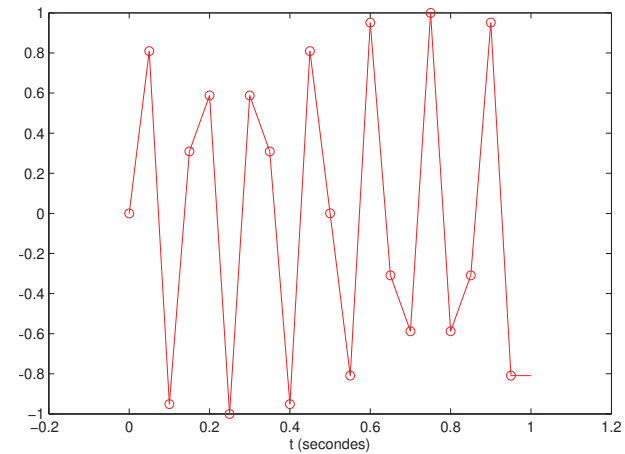
## Sinusoïde originelle



$t \in [0, 1[ \text{ avec } f_0 = 7 \text{ Hz}$

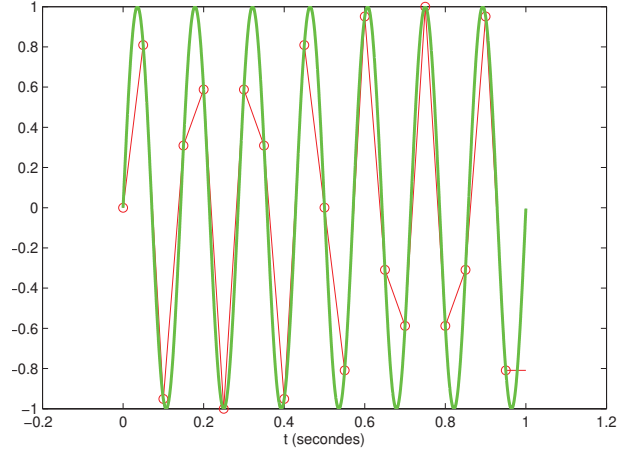
Sinusoïde échantillonnée à  $F_e = 20 \text{ Hz}$ 

$F_e = 20 \text{ Hz}, T_e = 0.05 \text{ secondes}$

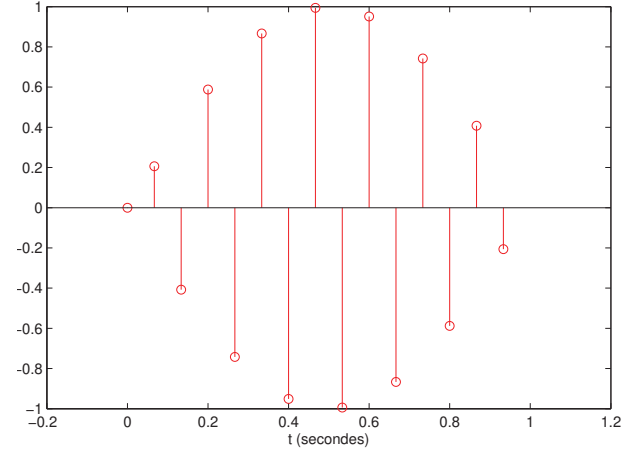
Signal reconstruit ( $F_e = 20 \text{ Hz}$ )

On utilise un bloqueur d'ordre 1 (interpolation linéaire)

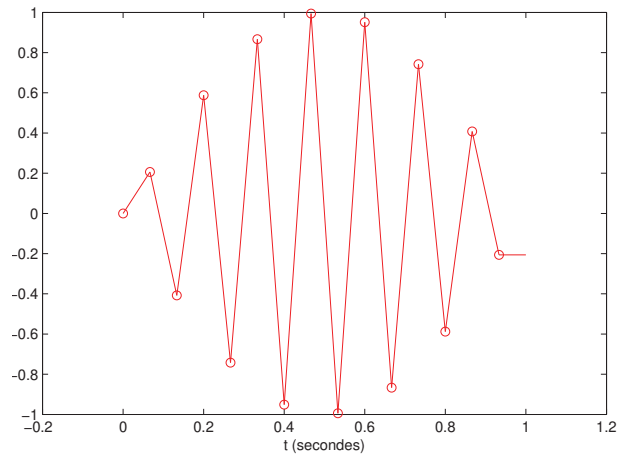


Sinusoïde reconstruite ( $F_e = 20$  Hz)

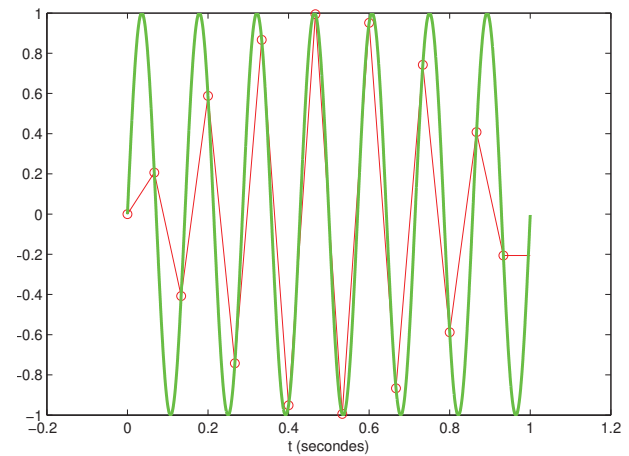
On retrouve une sinusoïde égale à celle d'origine !

Sinusoïde échantillonnée à  $F_e = 15$  Hz

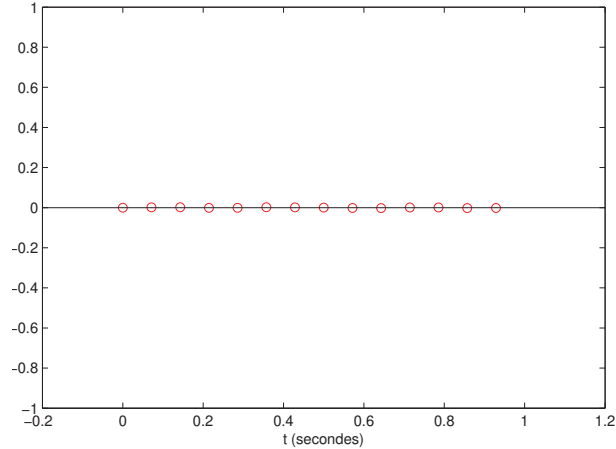
$F_e = 15$  Hz,  $T_e = 0.0667$  secondes

Signal reconstruit ( $F_e = 15$  Hz)

On utilise un bloqueur d'ordre 1 (interpolation linéaire)

Sinusoïde reconstruite ( $F_e = 15$  Hz)

On retrouve une sinusoïde égale à celle d'origine !

Sinusoïde échantillonnée à  $F_e = 14$  Hz

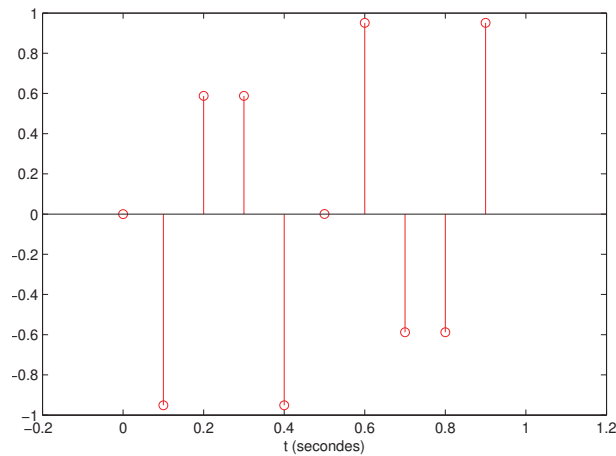
$F_e = 14$  Hz,  $T_e = 0.0714$  secondes

Sinusoïde échantillonnée à  $F_e = 14$  Hz

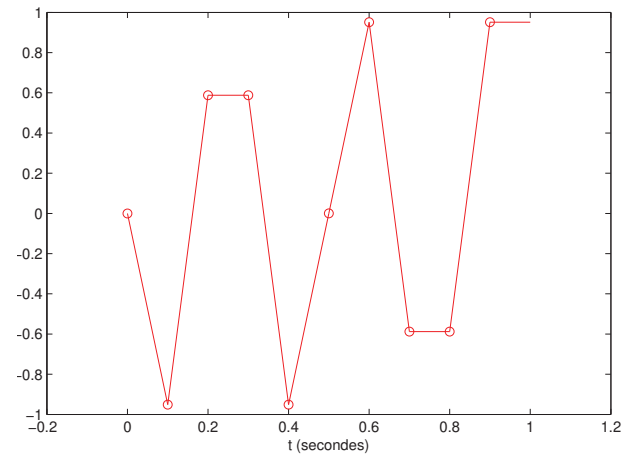
Pour  $F_e = 2f_0$ , tous les échantillons sont à 0 !

$$\begin{aligned} x[n] &= \sin\left(2\pi f_0 \frac{n}{F_e}\right) \\ &= \sin\left(2\pi f_0 \frac{n}{2f_0}\right) \\ &= \sin(\pi n) \\ &= 0 \end{aligned}$$

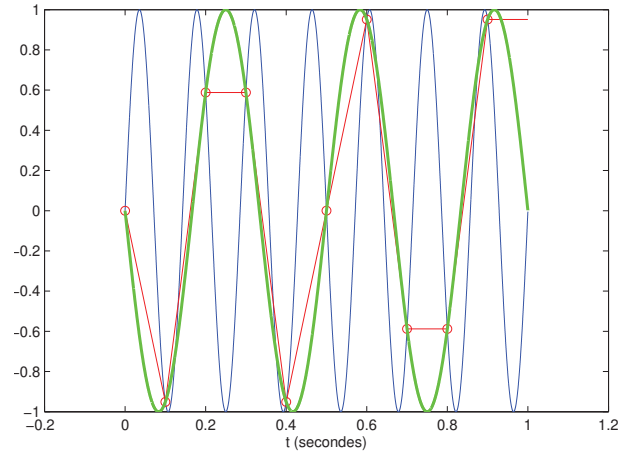
Reconstruction impossible si  $F_e = 2f_0$  !

Sinusoïde échantillonnée à  $F_e = 10$  Hz

$F_e = 10$  Hz,  $T_e = 0.1$  secondes

Signal reconstruit ( $F_e = 10$  Hz)

On utilise un bloqueur d'ordre 1 (interpolation linéaire)

Sinusoïde reconstruite ( $F_e = 10$  Hz)

On retrouve une autre sinusoïde de fréquence  $f_0 = 3$  Hz ! Pourquoi cela ?

## Critère de Nyquist : généralisation

- ▶ Etant donnée une sinusoïde de fréquence fondamentale  $f_0$  que l'on souhaite échantillonner à une fréquence d'échantillonnage  $F_e$ , on a vu que :
  - ▶ Si  $F_e > 2f_0$ , on est capable de reconstruire parfaitement la sinusoïde à partir des échantillons
  - ▶ Si  $F_e = 2f_0$ , tous les échantillons sont à 0 et il est impossible de reconstruire la sinusoïde
  - ▶ Si  $F_e < 2f_0$ , une sinusoïde avec une autre fréquence fondamentale semble apparaître après reconstruction : il est impossible de reconstruire la sinusoïde
- ▶ Critère de Nyquist : Pour reconstruire parfaitement après échantillonnage une sinusoïde de fréquence fondamentale  $f_0$ , il faut :

$$F_e > 2f_0$$

Différences entre  $f_0 = 7$  Hz et  $f_0 = 3$  Hz pour  $F_e = 10$  Hz

Cas  $f_0 = 7$  Hz

$$x[n] = \sin\left(2\pi \times 7 \times \frac{n}{F_e}\right)$$

- ▶  $x_0 = 0$
- ▶  $x_1 = \sin\left(2\pi \times 7 \times \frac{1}{10}\right) \approx -0.9511$
- ▶  $x_2 = \sin\left(2\pi \times 7 \times \frac{2}{10}\right) \approx 0.5878$
- ▶  $x_3 = \sin\left(2\pi \times 7 \times \frac{3}{10}\right) \approx 0.5878$
- ▶ ...

- ▶ On a  $x[n] = -y_n$  ! C'est pour cela que, lorsque l'on relie les points, on a l'impression de voir une sinusoïde de fréquence fondamentale  $f_0 = 3$  Hz au lieu de la sinusoïde originelle avec  $f_0 = 7$  Hz
- ▶ Problème : Si on a accès uniquement aux échantillons, nous n'avons aucun moyen de savoir si la fréquence fondamentale de la sinusoïde originelle était  $f_0 = 3$  Hz ou  $f_0 = 7$  Hz.

Cas  $f_0 = 3$  Hz

$$y_n = \sin\left(2\pi \times 3 \times \frac{n}{F_e}\right)$$

- ▶  $y_0 = 0$
- ▶  $y_1 = \sin\left(2\pi \times 3 \times \frac{1}{10}\right) \approx 0.9511$
- ▶  $y_2 = \sin\left(2\pi \times 3 \times \frac{2}{10}\right) \approx -0.5878$
- ▶  $y_3 = \sin\left(2\pi \times 3 \times \frac{3}{10}\right) \approx -0.5878$
- ▶ ...

## Critère de Nyquist : exemple audio

Illustrons ceci par des exemples audio :

- ▶ Prenons une sinusoïde de fréquence fondamentale  $f_0 = 880$  Hz.
- ▶ Testons plusieurs valeurs pour la fréquence d'échantillonnage  $F_e = 2000$  Hz, 1800 Hz, 1600 Hz, 1400 Hz
- ▶ Reconstituons un signal audio et... écoutons!

Résultats :

- ▶  $F_e = 2000$  Hz, 1800 Hz ( $F_e > 2f_0$ ) : OK !
- ▶  $F_e = 1600$  Hz, 1400 Hz ( $F_e < 2f_0$ ) : on entend des notes plus graves. Peut-on savoir quelles fréquences apparaissent ?

## Notion de fréquence apparente

On pourrait démontrer que, lorsqu'on échantillonne une sinusoïde de fréquence fondamentale  $f_0$ , si la fréquence d'échantillonnage ne vérifie pas le critère de Nyquist, il apparaît après échantillonnage une sinusoïde (éventuellement déphasée) de fréquence apparente  $f_{app}$  :

$$f_{app} = \min_{k \in \mathbb{Z}} \{|f_0 + kF_e|\}$$

## Exemple

$$f_{app} = \min_{k \in \mathbb{Z}} \{|f_0 + kF_e|\}$$

Dans le cas du signal audio considéré précédemment (avec  $f_0 = 880$  Hz) :

- ▶ Si  $F_e = 2000$  Hz ,  $f_{app} = 880$  Hz (correspond à  $k = 0$ )
- ▶ Si  $F_e = 1800$  Hz ,  $f_{app} = 880$  Hz (correspond à  $k = 0$ )
- ▶ Si  $F_e = 1600$  Hz ,  $f_{app} = 720$  Hz (correspond à  $k = -1$ )
- ▶ Si  $F_e = 1400$  Hz ,  $f_{app} = 520$  Hz (correspond à  $k = -1$ )

Remarque : si  $\frac{F_e}{2} < f_0 < F_e$ , on a  $f_{app} = F_e - f_0$

## Sommaire

### Quantification

- 3.1 Quantification uniforme
- 3.2 Erreur de quantification

## Qu'est-ce que la quantification ?

- ▶ Principe : Au lieu d'avoir un signal pouvant prendre n'importe quelle valeur, on va définir un ensemble fini de valeurs que le signal peut prendre
- ▶ Nous allons définir des intervalles de valeurs, et associer toutes les valeurs du signal comprises dans l'intervalle à une valeur quantifiée correspondant au milieu de l'intervalle  
Exemple : toutes les valeurs comprises entre 0.1 et 0.3 seront associées à la valeur 0.2
- ▶ Chacune des valeurs quantifiée sera associée à un code binaire composé de 0 et 1
- ▶ En partant d'un signal discret quelconque, on arrive ainsi à un signal numérique composé de 0 et de 1

## Quantification uniforme

- ▶ Il existe de nombreuses façons de choisir les intervalles pour quantifier un signal, qui dépendent fortement du type du signal
- ▶ Pour définir au mieux les intervalles, il faut savoir l'ordre de grandeur des valeurs prises par le signal
- ▶ Nous n'allons voir ici que la quantification uniforme : chaque intervalle de valeurs a la même taille.
- ▶ Largeur d'un intervalle constante  $q$  appelée **pas de quantification**

## Quantification uniforme

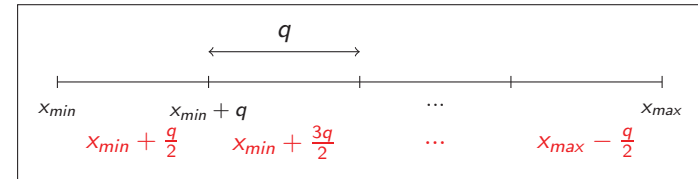
- ▶ Supposons que notre signal prend des valeurs comprises entre  $x_{min}$  et  $x_{max}$  et que l'on souhaite coder ces valeurs sur  $b$  bits.
- ▶ Si on veut coder sur  $b$  bits, on va définir  $2^b$  intervalles.
- ▶ Afin de couvrir toutes les valeurs possibles du signal, la taille de chaque intervalle sera :

$$q = \frac{x_{max} - x_{min}}{2^b}$$

- ▶ Les intervalles seront donc définis comme  $[x_{min}, x_{min} + q], [x_{min} + q, x_{min} + 2q], \dots, [x_{max} - q, x_{max}]$

- ▶ Les valeurs une fois quantifiées seront choisies sur une grille

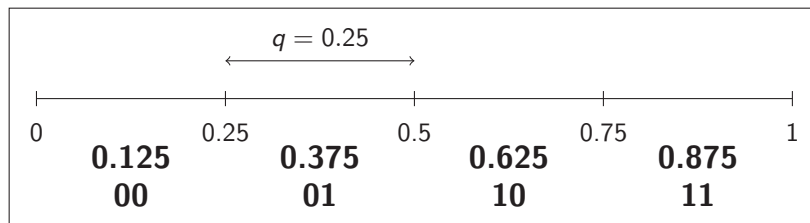
$$x_{min} + \frac{q}{2}, x_{min} + \frac{3q}{2}, \dots, x_{max} - \frac{q}{2}$$



## Exemple

Supposons ici que les valeurs à quantifier sont comprises entre 0 et 1, et que l'on souhaite coder chaque valeur sur 2 bits.

- ▶  $2^2 = 4$  intervalles chacun de longueur  $q = \frac{1 - 0}{2^2} = 0.25$
- ▶ 4 valeurs quantifiées possibles : 0.125, 0.375, 0.625, 0.875 (milieux des intervalles)



## Erreur de quantification

- ▶ Plus on quantifie sur peu de bits, plus on perd de l'information
- ▶ Erreur de quantification : différence entre la valeur originelle et la valeur quantifiée
- ▶ Dans le cas d'une quantification uniforme, l'erreur de quantification maximale pour une valeur est

$$\text{erreur maximale} = \frac{q}{2}$$

