

# Introduction au signal et bruit

## Travaux pratiques

Gabriel Dauphin

August 28, 2025

# Contents

<b>1</b>	<b>Séance 1 de travaux pratiques</b>	<b>3</b>
1.1	Préparation à faire avant la séance . . . . .	3
1.2	Travail à effectuer pendant la séance . . . . .	11
1.2.1	Préparation à l'utilisation de Python . . . . .	11
1.2.2	Visualisation de la réponse fréquentielle . . . . .	12
1.2.3	Détermination numérique de la réponse impulsionnelle . . . . .	16
1.2.4	Détermination de la réponse du système à $x_a(t) = \mathbb{T}(t)$ . . . . .	21
1.3	Travail à rendre une semaine après la séance . . . . .	23
<b>2</b>	<b>Séance 2 de travaux pratiques</b>	<b>25</b>
2.1	Préparation à faire avant la séance . . . . .	25
2.2	Travail à effectuer pendant la séance . . . . .	26
2.2.1	Signal étudié . . . . .	26
2.2.2	Calculs théoriques sur le signal $s_1(t)$ . . . . .	27

2.2.3	Echantillonnage du signal . . . . .	27
2.2.4	Simulation de la transformée de Fourier . . . . .	28
2.2.5	Simulation de la périodisation du spectre . . . . .	29
2.2.6	Périodisation de $s_1(t)$ . . . . .	30
2.2.7	Simulation de la transformée de Fourier . . . . .	31
2.2.8	Simulation de la périodisation de la transformée de Fourier . . . . .	32
2.2.9	Simulation de l'erreur quadratique moyenne en fonction de $f_e$ . . . . .	33
2.2.10	Reconstruction du signal par interpolation similaire à la formule de Shannon-Nyquist . . . . .	34
2.2.11	Simulation de l'erreur quadratique moyenne en fonction de $f_e$ après reconstruction . . . . .	35

### **3 Séance 3 de travaux pratiques 36**

3.1	. . . . .	36
-----	-----------	----

# Chapter 1

## Séance 1 de travaux pratiques

### 1.1 Préparation à faire avant la séance

Ceci constitue une partie théorique à faire avant la séance et à inclure dans le document pdf à rendre après la séance.

1. Choisissez un montage électronique simple pour lequel une source de tension ou d'intensité est considérée comme l'entrée d'un filtre et la tension aux bornes d'un composant ou l'intensité traversant un composant est considéré comme la sortie. Ce montage devra fonctionner ici en régime linéaire (par exemple l'amplificateur opérationnel devra être utilisé en rétro-action stable et non comme un comparateur).

Vous préciserez les valeurs numériques de chaque composant. Vous pouvez vous inspirer du montage présenté en section 1.1. Il est souhaitable que les caractéristiques du filtre soient numériquement simples à simuler, vous pouvez modifier les valeurs des composants pour que ce soit le cas.

2. En vous inspirant de la section 1.1 et à partir d'une analyse physique du montage proposé, justifiez que le filtre proposé peut être modélisé avec une relation linéaire et temps invariante.
3. En vous inspirant de la section 1.1, déterminez la réponse statique du filtre et le comportement limite à haute fréquence.
4. Calculez la réponse fréquentielle du filtre en utilisant vos connaissances d'électronique.
5. Déduisez une relation entrée-sortie décrite avec une équation différentielle.
6. Déduisez la réponse impulsionnelle théorique (il n'est pas nécessaire de conduire les calculs jusqu'au bout, il suffit qu'on puisse programmer le calcul de la réponse impulsionnelle).

Dans un premier temps, on considère dans ce TP ce montage et cette étude théorique.

## Montage étudié

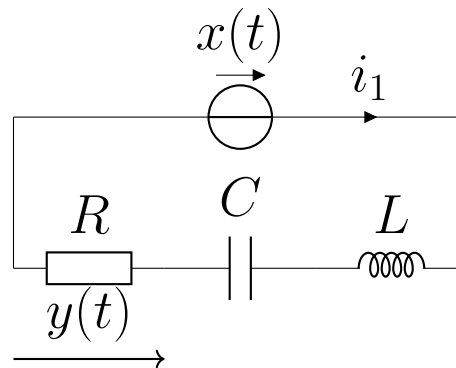


Figure 1.1: Montage correspondant au filtre étudié d'entrée la tension  $x(t)$  et de sortie la tension  $y(t)$ .  $R = 3\Omega$ ,  $C = 0.5\text{F}$ ,  $L = 1\text{H}$ .

## Analyse physique rapide du montage

Tous les composants sont linéaires entre l'intensité et la tension aussi la relation entre  $x(t)$  et  $y(t)$  est linéaire. On suppose ici que les composants ont des caractéristiques fixes, donc le système est temps invariant. À très basse fréquence, le condensateur se comporte comme un circuit ouvert et donc  $\widehat{Y}(f)$  est nul à très basse fréquence. À très haute fréquence, la bobine se comporte comme un circuit ouvert et donc  $\widehat{Y}(f)$  est nul à très haute fréquence.

## Calcul théorique de la réponse fréquentielle

En utilisant les impédances de chaque composant et en observant qu'il y a un diviseur de tension, on a

$$\widehat{H}(f) = \frac{\widehat{Y}(f)}{\widehat{X}(f)} = \frac{R}{R + j2\pi fL + \frac{1}{j2\pi fC}} = \frac{j2\pi fRC}{1 + j2\pi fRC - 4\pi^2 f^2 LC} \quad (1.1)$$

On peut remarquer que les affirmation de la section 1.1 sont confirmées par l'équation (1.1)

$$\left\{ \begin{array}{l} \lim_{f \rightarrow 0} \widehat{H}(f) = \lim_{f \rightarrow 0} \frac{j2\pi fRC}{1 + j2\pi fRC - 4\pi^2 f^2 LC} = 0 \\ \text{et} \quad \lim_{f \rightarrow +\infty} \widehat{H}(f) = \lim_{f \rightarrow +\infty} \frac{R}{R + j2\pi fL + \frac{1}{j2\pi fC}} = 0 \end{array} \right. \quad (1.2)$$

## Écriture de la relation entrée-sortie avec une équation différentielle

On remplace chaque terme  $j2\pi f$  par  $\frac{d}{dt}$

$$LC \frac{d^2}{dt^2} y(t) + RC \frac{d}{dt} y(t) + y(t) = RC \frac{d}{dt} x(t) \quad (1.3)$$

## Calcul théorique de la réponse impulsionnelle : solution 1

Les calculs qui suivent peuvent dans une certaine mesure être obtenus en utilisant la toolbox <sup>1</sup> `sympy` à installer sur Python <https://docs.sympy.org> et en util-

---

<sup>1</sup>Remarquez que certaines commandes de `sympy` commencent par une majuscule.

isant Wolfram <https://www.wolframalpha.com/>

Wolfram `Fourier transform of exp(-p*t)*heaviside(t)` *Il reste à multiplier par  $\sqrt{2\pi}$  et à remplacer  $\omega$  par  $2\pi f$ .*

En posant  $x(t) = \delta(t)$ , on sait que  $y(t)$  est la réponse impulsionnelle  $h(t)$  qui vérifie donc

$$LC \frac{d^2}{dt^2} h(t) + RC \frac{d}{dt} h(t) + h(t) = RC \frac{d}{dt} \delta(t) \quad (1.4)$$

Cette équation peut se mettre sous une forme plus classique

$$LC \frac{d^2}{dt^2} \tilde{h}(t) + RC \frac{d}{dt} \tilde{h}(t) + \tilde{h}(t) = \delta(t) \text{ et } h(t) = RC \frac{d}{dt} \tilde{h}(t) \quad (1.5)$$

Le polynôme  $Q(p) = LCp^2 + RCp + 1$  a deux racines réelles ( $R^2C^2 - 4LC > 0$ ) :  
 $p_1 = \frac{-RC - \sqrt{R^2C^2 - 4LC}}{2LC}$  et  $p_2 = \frac{-RC + \sqrt{R^2C^2 - 4LC}}{2LC}$ .

## Python :

Le calcul symbolique permet de faire faire la résolution.

```
import sympy
R=sympy.Symbol('R')
L=sympy.Symbol('L')
```



```

C=sympy.Symbol('C')
p=sympy.Symbol('p')
p1,p2=sympy.solve('L*C*p**2+R*C*p+1','p'); print(f"p1={p1}, p2={p2}")
sympy.simplify(p1+p2)
sympy.simplify(p1*p2)

```

Les valeurs de  $p_1$  et  $p_2$  sont négatives et distinctes. Aussi  $\tilde{h}(t)$  se met sous la forme

$$\tilde{h}(t) = (c_1 e^{p_1 t} + c_2 e^{p_2 t}) \llbracket t \geq 0 \rrbracket \quad (1.6)$$

Pour trouver  $c_1$  et  $c_2$ , on pourrait calculer sa transformée de Fourier et identifier avec  $\widehat{H}(f)$ . On peut aussi calculer les dérivées successives pour vérifier avec l'équation (1.5).

$$\begin{aligned} RC \frac{d}{dt} \tilde{h}(t) &= RC(c_1 p_1 e^{p_1 t} + c_2 p_2 e^{p_2 t}) \llbracket t \geq 0 \rrbracket + RC(c_1 + c_2) \delta(t) \\ LC \frac{d^2}{dt^2} \tilde{h}(t) &= LC(c_1 p_1^2 e^{p_1 t} + c_2 p_2^2 e^{p_2 t}) \llbracket t \geq 0 \rrbracket + (c_1 + c_2) \delta'(t) + LC(c_1 p_1 + c_2 p_2) \delta(t) \end{aligned} \quad (1.7)$$

Ces dérivées successives peuvent ensuite être réintroduites dans l'équation (1.5), on a alors l'égalité quand ceci est vérifié.

$$\left\{ \begin{array}{ll} c_1(1 + RCp_1 + L Cp_1^2) = 0 & \text{termes en facteur de } e^{p_1 t} \llbracket t \geq 0 \rrbracket \\ c_2(1 + RCp_2 + L Cp_2^2) = 0 & \text{termes en facteur de } e^{p_2 t} \llbracket t \geq 0 \rrbracket \\ RC(c_1 + c_2) + LC(c_1 p_1 + c_2 p_2) = 1 & \text{termes en facteur de } \delta(t) \\ c_1 + c_2 = 0 & \text{termes en facteur de } \delta'(t) \end{array} \right. \Rightarrow \left\{ \begin{array}{l} c_1 + c_2 = 0 \\ LC(c_1 p_1 + c_2 p_2) = 1 \end{array} \right.$$

La résolution de ce système donne

$$c_1 = \frac{1}{LC(p_1 - p_2)} \text{ et } c_2 = -\frac{1}{LC(p_1 - p_2)} \quad (1.9)$$

Et en utilisant  $c_1 + c_2 = 0$ , on a finalement

$$h(t) = RC \frac{d}{dt} \tilde{h}(t) = RC \frac{d}{dt} \tilde{h}(t) = RC(c_1 p_1 e^{p_1 t} + c_2 p_2 e^{p_2 t}) = \frac{R}{L} \frac{p_1}{p_1 - p_2} e^{p_1 t} \mathbb{I}[t \geq 0] - \frac{R}{L} \frac{p_2}{p_1 - p_2} e^{p_2 t} \mathbb{I}[t \geq 0]$$

### Python :

```
from sympy.matrices import *
M=Matrix([[1,1],[p2,p1]])
B=Matrix([[R/L],[0]])
X=M.solve(B); c1=X[0]; c2=X[1]
print(f"c1={c1} c2={c2}")
#Pour vérifier
sympy.simplify(c1+c2)
sympy.simplify(p2*c1+p1*c2)
```

## Calcul théorique de la réponse impulsionnelle : solution 2

Une seconde solution consiste à utiliser  $p_1$  et  $p_2$  calculé à partir de l'équation différentielle vérifiée par  $\tilde{h}(t)$  et à supposer qu'ils sont valables pour  $h(t)$  et que donc  $h(t)$  serait

de cette forme-là. Et n'étant pas tout à fait sûr, on rajoute un possible Dirac.

$$h(t) = d_1 e^{p_1 t} \llbracket t \geq 0 \rrbracket + d_2 e^{p_2 t} + d_3 \delta(t) \quad (1.11)$$

La transformée de Fourier de cette réponse impulsionnelle est

$$\widehat{H}(f) = d_1 \frac{1}{j2\pi f - p_1} + d_2 \frac{1}{j2\pi f - p_2} + d_3 \quad (1.12)$$

On identifie avec  $\widehat{H}(f)$  pour trois fréquences quelconques, par exemple  $f_0 = 0$ ,  $f_1 = 1$  et  $f_2 = 2$ .

$$\begin{cases} d_1 \frac{1}{j2\pi f_0 - p_1} + d_2 \frac{1}{j2\pi f_0 - p_2} + d_3 = \widehat{H}(f_0) \\ d_1 \frac{1}{j2\pi f_1 - p_1} + d_2 \frac{1}{j2\pi f_1 - p_2} + d_3 = \widehat{H}(f_1) \\ d_1 \frac{1}{j2\pi f_2 - p_1} + d_2 \frac{1}{j2\pi f_2 - p_2} + d_3 = \widehat{H}(f_2) \end{cases} \quad (1.13)$$

Et on demande à l'ordinateur de résoudre ce système en utilisant des valeurs numériques de  $R, L, C$ .

## 1.2 Travail à effectuer pendant la séance

### 1.2.1 Préparation à l'utilisation de Python

#### Python :

Pour démarrer l'utilisation de Python, je propose de choisir deux répertoires, le premier que j'appelle **rep\_prg** contient les programmes, notamment ceux disponibles sur mon site, et un autre répertoire que j'appelle **rep\_tra** où vous mettez le travail effectué notamment les données et les figures.

Je propose ensuite d'effectuer les lignes suivantes qui utilisent un module **seb** que j'ai écrit pour ce cours et qui est disponible sur <https://gabrieldauphin.neocities.org/L3SPI> ou sur <https://www-l2ti.univ-paris13.fr/~dauphin/L3SPI> **plt** et **np** sont les modules **matplotlib** et **numpy** qui servent à tracer des graphes et à manipuler des vecteurs.

```
import sys
sys.path.append('rep_prg')
import os
os.chdir('rep_tra')
import seb
plt,np,sig=seb.debut()
```

## 1.2.2 Visualisation de la réponse fréquentielle

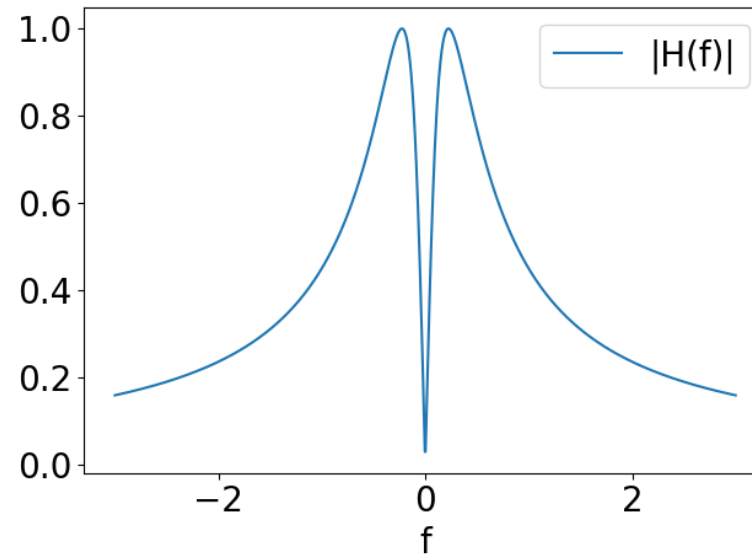


Figure 1.2: Module de la réponse fréquentielle

- Créez une échelle de fréquences avec le vecteur  $\mathbf{f}$  par exemple entre  $-3$  et  $3$  et représentez graphiquement la réponse fréquentielle

On appelle vecteur un tableau de valeurs composé d'une ligne (on parle alors de vecteur ligne) ou d'une seule colonne (on parle de vecteur colonne). La fonction `linspace` de `numpy` permet de générer un ensemble de valeurs en indiquant la première, la dernière et le nombre de ces valeurs. Ici ceci permet de construire l'échelle en fréquence.

```
f=np.linspace(-3,3,10**3)
```

Cette instruction génère 1000 valeurs entre -3 et 3. Cette notion est utile pour générer un graphique, elle permet d'obtenir le graphe 1.2 représentant le module de la réponse fréquentielle.

On définit les valeurs des variables. En Python il est possible d'allouer plusieurs variables en même temps.

```
R,C,L = 3,0.5,1
```

On obtient la réponse fréquentielle, ( $\pi$ ,  $j$  et le carré sont implémentés avec `np.pi` et `1j` et `**2`).

```
H=1j*2*np.pi*f*R*C/(1+1j*2*np.pi*f*R*C-4*(np.pi**2)*(f**2)*L*C)
```

On obtient alors un vecteur ligne de même taille que `f` et contenant successivement toutes les valeurs complexes de  $H$  pour chacune des valeurs du vecteur `f`.

Pour faire le graphe, on commence par

```
fig,ax = plt.subplots()
```

Ensuite pour la courbe on rajoute <sup>2</sup> On implémente le module avec `np.abs`. Le troisième argument permet de rajouter une légende.

```
ax.plot(f,np.abs(H),label='|H(f)|')
```

---

<sup>2</sup>Et si on avait plusieurs courbes, il suffit de rajouter une ligne par courbe.

Sur le graphe on peut préciser ce que signifie l'axe des abscisses

```
ax.set_xlabel('f')
```

L'affichage de la légende est déclenchée par

```
ax.legend()
```

La gestion de la taille de la figure est faite avec

```
plt.tight_layout()
```

Je propose de sauvegarde la figure

```
fig.savefig('nom_figure.png')
```

La figure apparaît lorsqu'on exécute cette commande

```
fig.show()
```

Et comme on va utiliser à nouveau ces calculs il est intéressant de construire une fonction qui calcule  $\widehat{H}(f)$ . Notez qu'il est très important ici de laisser deux espaces au début de chaque ligne après la première ligne pour indiquer que les instructions font parti de la fonction appelée **H1**.

```
def H1(R,L,C,f):
    """fonction de la reponse frequentielle obtenue à partir du montage da
    import numpy as np
    H=1j*2*np.pi*f*R*C/(1+1j*2*np.pi*f*R*C-4*(np.pi**2)*(f**2)*L*C)
    return H
```

On peut faire appel à cette fonction avec

```
ax.plot(f,np.abs(H1(f)),label=' |H(f)| ')
```



### 1.2.3 Détermination numérique de la réponse impulsionnelle

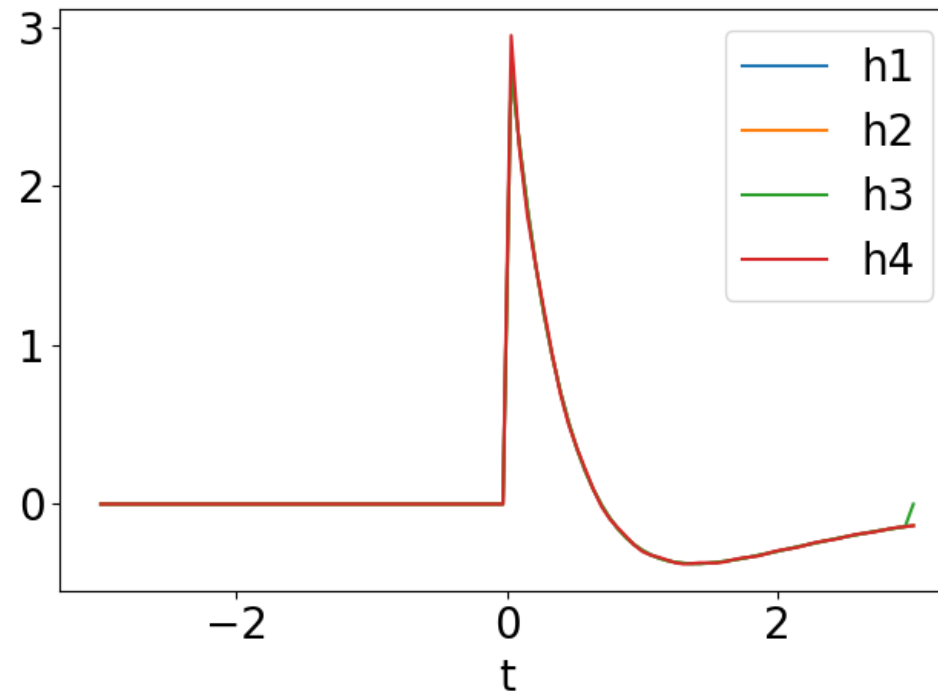


Figure 1.3: Réponse impulsionnelle  $h(t)$ .

- Créez une échelle de temps avec le vecteur  $\mathbf{t}$  par exemple entre  $-3$  et  $3$  et représentez graphiquement cette réponse impulsionnelle (voir section ??, p. ??) soit avec **h1** notée  $h_1(t)$  soit avec **h2** notée  $h_2(t)$ .
- Déterminez  $h_3(t)$  la réponse impulsionnelle à partir de l'équation différentielle (1.5) en s'inspirant de la section ??.

- Déterminez  $h_4(t)$  la réponse impulsionnelle cette fois-ci à partir de la réponse fréquentielle (1.1) en s'inspirant de la section ??.

La figure 1.3 montre  $h_1(t)$ ,  $h_2(t)$ ,  $h_3(t)$ ,  $h_4(t)$ . Ici l'implémentation est indiquée dans ??.

## Implémentation de la première solution

La partie théorique permet de calculer la réponse impulsionnelle liée à la première solution

### Python :

On calcule d'abord  $p_1$  et  $p_2$ .

```
p1=(-C*R - np.sqrt(C*(C*R**2 - 4*L)))/(2*C*L)
p2=(-C*R + np.sqrt(C*(C*R**2 - 4*L)))/(2*C*L)
```

On peut ensuite vérifier si les formules sont justes.

```
assert np.abs(L*C*p1**2+R*C*p1+1)<1e-8
assert np.abs(L*C*p2**2+R*C*p2+1)<1e-8
```

On calcule ensuite  $c_1$  et  $c_2$

$$c1, c2 = R/L * p1 / (p1 - p2), -R/L * p2 / (p1 - p2)$$

Puis on en déduit la réponse impulsionnelle.

```
y=c1*np.exp(t*p1)*(t>=0)+c2*np.exp(t*p2)*(t>=0)
y[t<0]=0
```

La dernière ligne a l'objectif de donner une valeur nulle lorsque  $t < 0$  y compris quand les expressions de  $y(t)$  pour  $t \geq 0$  n'ont pas de sens.

Je propose de mettre toutes ces lignes de code dans une fonction notée **h1** dépendant de  $R, L, C, t$ .

## Implémentation de la deuxième solution

La deuxième solution utilise le calcul déjà effectué de  $p_1$  et  $p_2$ , que l'on peut retrouver avec

```
p1,p2=np.roots(np.array([L*C,R*C,1]))
```

Pour simplifier le code, je propose de définir une fonction notée **Hp** calculant  $\frac{1}{j2\pi f - p}$

```
def Hp(f,p):
    return 1/(1j*2*np.pi*f-p)
```

Je considère trois fréquences

```
f=np.array([0,0.1,0.2])
```

La matrice contenant les paramètres à gauche de l'équation matricielle (1.13)

```
M = np.array([[Hp(f[0],p1),Hp(f[0],p2),1],[Hp(f[1],p1),Hp(f[1],p2),1],[Hp(f[2],p1),Hp(f[2],p2),1]])
```

Le vecteur colonne contenant les valeurs complexes souhaitées de la réponse fréquentielle sont les valeurs à droite de l'équation matricielle (1.13)

```
B = np.array([[H1(R,L,C,f[0])],[H1(R,L,C,f[1])],[H1(R,L,C,f[2])]])
```

L'ordinateur résout le système d'équations linéaires

```
d1,d2,d3=np.linalg.solve(M,B); d1,d2,d3=d1[0],d2[0],d3[0]
```

On vérifie qu'il était inutile de considérer un Dirac supplémentaire

```
assert np.abs(d3)<1e-10, (f"d3={d3:.2e}")
```

La réponse impulsionnelle trouvée est alors

```
y=d1*np.exp(p1*t)*(t>=0)+d2*np.exp(p2*t)*(t>=0)  
y[t<0]=0
```

Je propose de la même façon de mettre ces lignes de code dans une deuxième fonction `h2`.

## Détermination de $h(t)$ à partir de l'équation différentielle

On peut trouver  $\tilde{h}$  en utilisant directement la fonction `sol_eq_diff` du module `seb` à partir de l'équation (1.4).

```
y1=seb.sol_eq_diff((L*C,R*C,1),t)
```

Notez que le premier argument est un **tuple** composé de trois arguments qui sont respectivement les coefficients devant  $\frac{d^2}{dt^2}y(t)$ ,  $\frac{d}{dt}y(t)$ ,  $y(t)$ .

On trouve alors  $h(t)$  avec l'équation (1.5)

```
y2=R*C*seb.deriver(t,y1)
```

Je propose de mettre ces lignes de code dans une fonction `h3`.

## Détermination de la réponse impulsionnelle à partir de la réponse fréquentielle

On utilise ici la fonction `H1` donnant la réponse fréquentielle du filtre étudié en utilisant (1.1). On évalue cette réponse fréquentielle en utilisant un très grand nombre de valeurs de fréquences. Puis on lui applique la transformée de Fourier inverse `TFI` définie dans `seb.py`.

```
f1=np.linspace(-20,20,10**5)
y1=seb.TFI(f1,H1(R,L,C,f1),t)
```

Comme on sait qu'avant  $t = 0$ , cette réponse impulsionnelle est nulle et que cette réponse impulsionnelle est réelle, on rajoute

```
y1[t<0]=0  
y2=np.real(y1)
```

Je propose de mettre ces instructions dans une fonction **h4** et d'afficher sur un même graphique ces trois ou quatre fonctions.

#### 1.2.4 Détermination de la réponse du système à $x_a(t) = \mathbb{T}(t)$

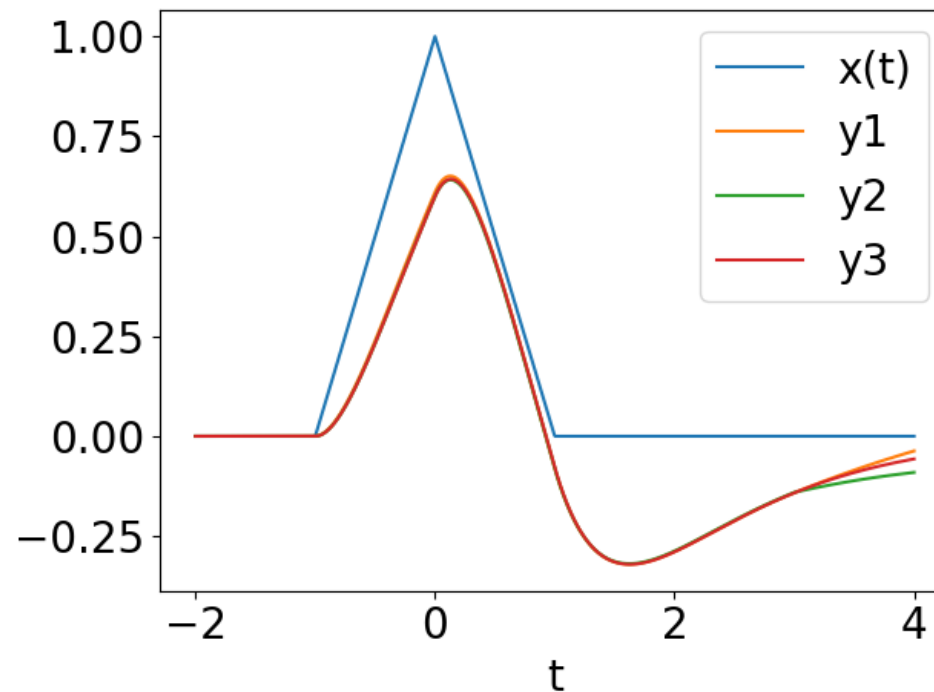


Figure 1.4

On remarque que la fonction triangle dérivée une fois est une fonction porte avancée et une fonction porte retardée, (la porte étant définie  $\Pi(t) = \llbracket |t| \leq 0.5 \rrbracket$ ).

$$RC \frac{d}{dt} \mathbb{T}(t) = RC \Pi(t + 0.5) - RC \Pi(t - 0.5) \quad (1.14)$$

Dérivée deux fois, ce sont trois, l'un avancé, le deuxième au milieu et un retardé.

$$RC \frac{d^2}{dt^2} \mathbb{T}(t) = RC \delta(t - 1) - 2RC \delta(t) + RC \delta(t + 1) \quad (1.15)$$

En intégrant cette expression, on trouve alors que

$$RC \frac{d}{dt} \mathbb{T}(t) = \int_{-\infty}^t RC \delta(\tau - 1) - 2RC \delta(\tau) + RC \delta(\tau + 1) d\tau \quad (1.16)$$

Ceci permet de donner une nouvelle définition de  $y_{a2}(t)$

$$\begin{cases} LC \frac{d^2}{dt^2} \tilde{y}_{a2}(t) + RC \frac{d}{dt} \tilde{y}_{a2}(t) + \tilde{y}_{a2}(t) = \delta(t) \\ y_{a2}(t) = \int_{-\infty}^t RC \tilde{y}_{a2}(\tau - 1) - 2RC \tilde{y}_{a2}(\tau) + RC \tilde{y}_{a2}(\tau + 1) d\tau \end{cases} \quad (1.17)$$

1. Déterminez  $y_{a1}(t)$  en utilisant la réponse impulsionnelle  $h_1(t)$  ou  $h_2(t)$  et la définition de  $x_a(t)$  en s'inspirant des sections ?? et ?? (p. ??, ??).
2. Utilisez l'équation (1.17) pour déterminer  $y_{a2}(t)$  la réponse du filtre à  $\mathbb{T}(t)$ .

3. Calculez la réponse fréquentielle de la sortie  $y_{a3}(t)$  en utilisant la réponse fréquentielle calculée en section 1.2.2 et la transformée Fourier de la fonction triangle pour en déduire  $y_{a3}(t)$  avec

$$y_{a3}(t) = \text{TF}^{-1} [\widehat{H}(f) \text{TF} [\mathbb{T}(t)] (f)] (t) \quad (1.18)$$

Comme on sait que la sortie est réelle, il est intéressant de prendre la partie réelle de la sortie.

La figure 1.4 montre  $y_{a1}(t)$ ,  $y_{a2}(t)$  et  $y_{a3}$ . Ici l'implémentation est indiquée dans ??.

## 1.3 Travail à rendre une semaine après la séance

En vous inspirant du travail effectué précédemment, répondez aux questions suivantes. Cette partie est à mettre après la partie correspondant à la section 1.1.

7. Créez une échelle de fréquences avec le vecteur **f** par exemple entre  $-3$  et  $3$  et représentez graphiquement cette réponse impulsionnelle (voir section ??, p. ??).
8. Créez une échelle de temps avec le vecteur **t** par exemple entre  $-3$  et  $3$  et représentez graphiquement cette réponse impulsionnelle (voir section ??, p. ??) soit avec **h1** notée  $h_1(t)$  soit avec **h2** notée  $h_2(t)$ .
9. Déterminez  $h_3(t)$  la réponse impulsionnelle à partir de l'équation différentielle (1.5) en s'inspirant de la section ??.



10. Déterminez  $h_4(t)$  la réponse impulsionnelle cette fois-ci à partir de la réponse fréquentielle (1.1) en s'inspirant de la section ??.
11. Choisissez un signal d'entrée noté  $x(t)$ .
12. Déterminez  $y_1(t)$  la réponse du filtre à  $x(t)$  en utilisant la réponse impulsionnelle  $h_1(t)$  ou  $h_2(t)$  et  $x(t)$  en s'inspirant des sections ?? et ?? (p. ??, ??).
13. Utilisez un calcul similaire à l'équation (1.17) pour déterminer  $y_2(t)$  la réponse du filtre à  $\mathbb{T}(t)$ .
14. Calculez la réponse fréquentielle de la sortie  $y_3(t)$  en utilisant la réponse fréquentielle calculée en section ?? et la transformée Fourier de la fonction triangle pour en déduire  $y_3(t)$  avec

$$y_3(t) = \text{TF}^{-1} [\widehat{H}(f) \text{TF} [\mathbb{T}(t)] (f)] (t) \quad (1.19)$$

Comme on sait que la sortie est réelle, il est intéressant de prendre la partie réelle de la sortie.

Le document à rendre à un **pdf** dont la première page doit lister toutes les figures, toutes les formules et toutes les réponses aux questions. La suite du document expliquant la justification des réponses et/ou les programmes utilisés.

# Chapter 2

## Séance 2 de travaux pratiques

### 2.1 Préparation à faire avant la séance

1. Choisissez la représentation graphique d'un signal simple que vous allez étudier à la place de  $s_1(t)$  représenté sur la figure [2.1](#). Ce signal doit non-nul seulement sur un intervalle petit (il n'est donc pas périodique). On appelle ce signal choisi  $s_2(t)$ .
2. Explicitez  $s_2(t)$  au moyen des fonctions de base présentées dans le cours de façon similaire à l'équation ([2.1](#)).
3. Calculez l'intégrale de ce signal  $\int_{-\infty}^{+\infty} s_2(t) dt$ .

4. Si possible calculez la transformée de Fourier  $\widehat{S}_2(f)$ .

## 2.2 Travail à effectuer pendant la séance

### 2.2.1 Signal étudié

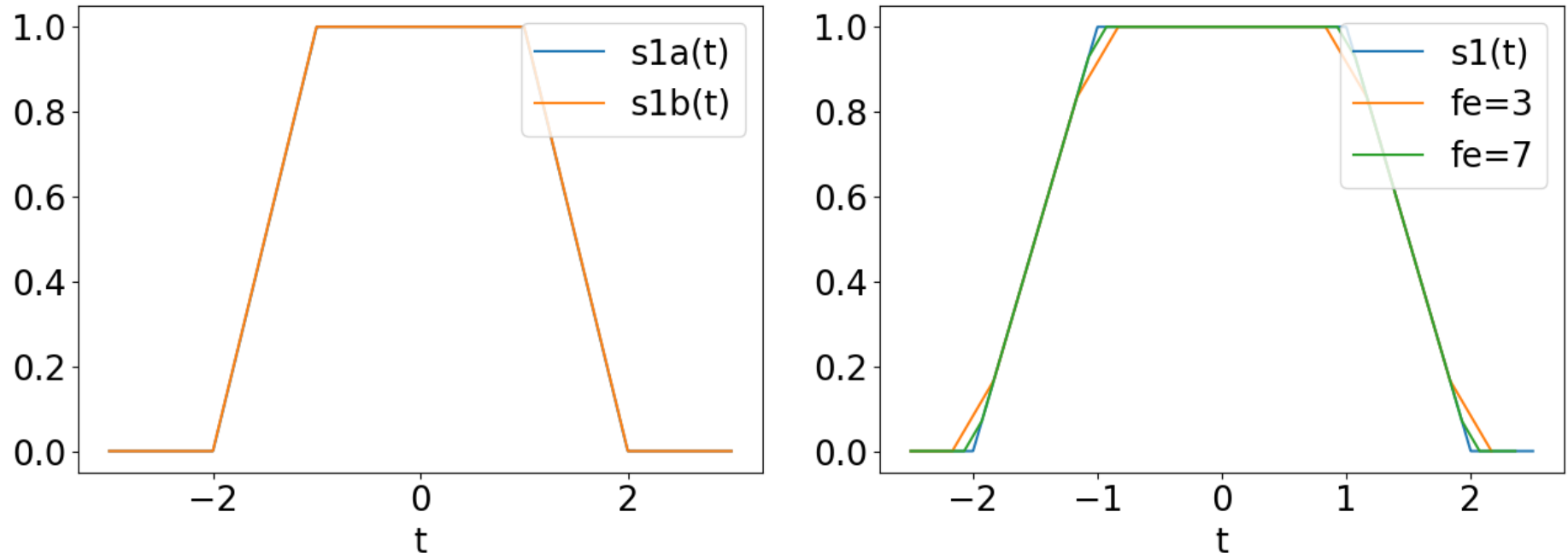


Figure 2.1: Signal étudié  $s_1(t)$  et deux échantillonnages

Graphiquement on voit que le signal peut se mettre sous deux formes (la première notée  $s_{1a}(t)$  et la deuxième notée  $s_{1b}(t)$ ).

$$s_1(t) = 2\mathbb{T}(t/2) - \mathbb{T}(t) = \mathbb{C}(t + 1.5) + \Pi(t/2) + \mathbb{D}(t - 1.5) \quad (2.1)$$

La gauche de la figure 2.1 montre le signal étudié  $s_1(t)$  avec ces deux formes. La droite montre l'échantillonnage avec deux fréquences d'échantillonnages. L'implémentation est réalisée dans ??.

## 2.2.2 Calculs théoriques sur le signal $s_1(t)$

Graphiquement on peut voir la surface du signal qui est composé de deux triangles et d'un rectangle, il a donc une surface de  $2(1 \times 1/2) + (1 \times 2) = 3$ .

$$\int_{-\infty}^{+\infty} s_1(t) dt = 3 \quad (2.2)$$

Je propose d'utiliser l'expression de  $s_1(t)$  en fonction de  $\mathbb{T}(t)$  dont la transformée de Fourier vaut  $\text{sinc}^2(f)$  :

$$\widehat{S}_1(f) = 2\text{TF} [\mathbb{T}(t/2)] (f) - \text{TF} [\mathbb{T}(t)] (f) = 4\text{TF} [\mathbb{T}(t)] (2f) - \text{TF} [\mathbb{T}(t)] (f) = 4 \text{sinc}^2(2f) -$$

On remarque que ces deux équations sont cohérentes :  $3 = \widehat{S}_1(0) = 4 - 1$

## 2.2.3 Echantillonnage du signal

La droite de la figure 2.1 montre le signal étudié  $s_1(t)$  et deux échantillonnages faits à deux fréquences différentes, ici  $fe_a = 3\text{Hz}$  et  $fe_b = 7\text{Hz}$ . L'implémentation est réalisée dans ??.

## 2.2.4 Simulation de la transformée de Fourier

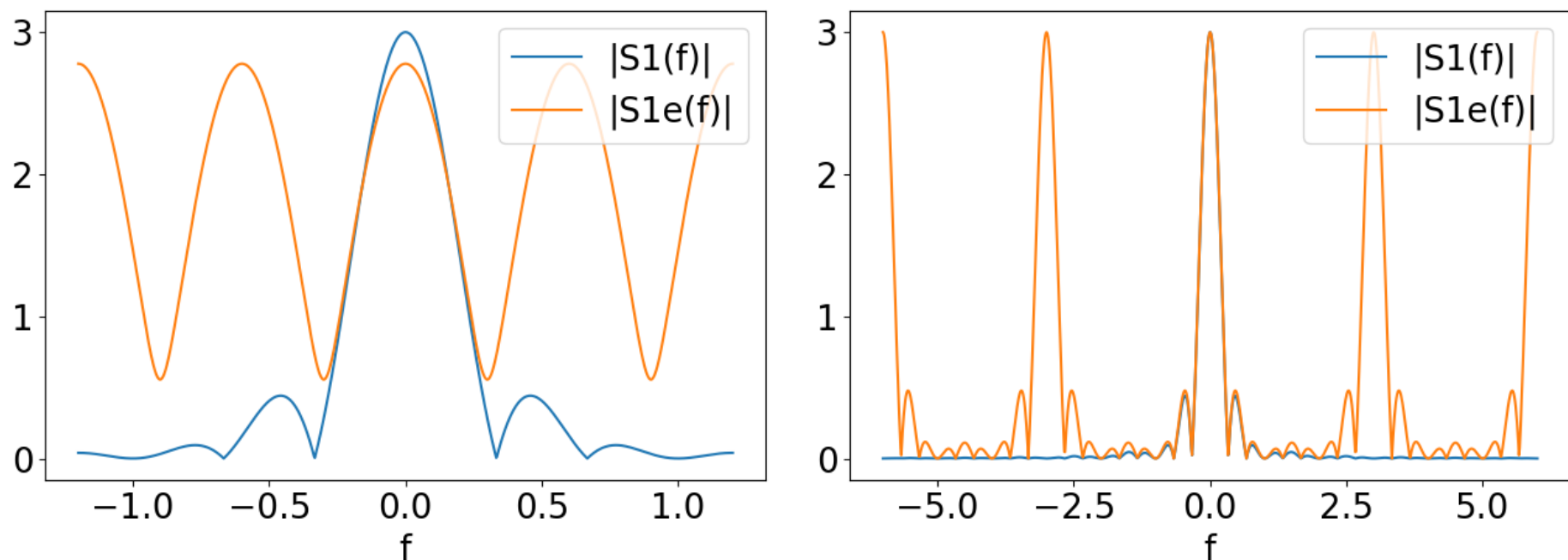


Figure 2.2: Spectre de  $s_1(t)$  et à gauche de  $s_1(t)$  échantillonné à 0.6Hz et à droite à 3Hz.

- Choisissez deux fréquence d'échantillonnage  $fe_a$  et  $fe_b$  et montrez que les transformées de Fourier des signaux échantillonnés ne sont pas les mêmes que celle du signal non-échantillonné.

La figure 2.2 montre à gauche et à droite deux spectres périodique, ce sont les transformées de Fourier des signaux échantillonnés. C'est le même spectre en bleu et non-périodique qui est représenté à gauche et à droite. L'implémentation est réalisée dans ??.

## 2.2.5 Simulation de la périodisation du spectre

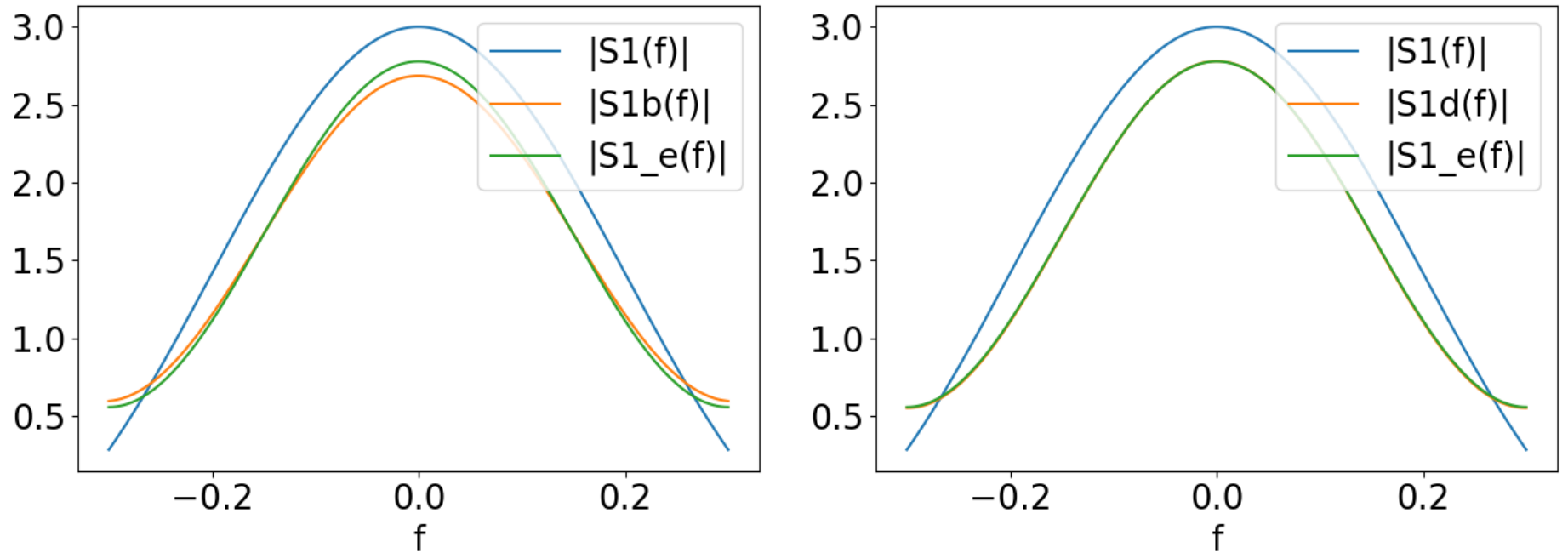


Figure 2.3: Spectre de  $s_1(t)$  et à gauche de  $s_1(t)$  échantillonné à 0.6Hz et à droite à 3Hz.

- Représentez  $|\widehat{S}_1(f)|$ ,  $|\widehat{S}_1(f) + \widehat{S}_1(f + f_e) + \widehat{S}_1(f - f_e)|$ ,  $|\widehat{S}_1(f) + \widehat{S}_1(f + f_e) + \widehat{S}_1(f - f_e)|$  et  $\frac{1}{f_e} |\widehat{S}e_1(f)|$  avec une représentation centrée. La notion de représentation centrée est définie dans ???. Vous pouvez utiliser une version approchée ou exacte de  $\widehat{S}_1(f)$ .

La figure 2.3 montre dans les deux figures en haut  $S_1(f)$  et avec une forme plus en cloche  $\widehat{S}_1^\#(f) = \frac{1}{f_e} \text{TFTD}[s_1[n]](f)$ . À gauche,  $\widehat{S}_1^\#(f)$  est approchée avec  $|\widehat{S}_1(f) + \widehat{S}_1(f + f_e) + \widehat{S}_1(f - f_e)|$  on voit la différence entre les deux courbes. Cette différence avec  $\widehat{S}_1^\#(f)$  disparaît à droite en visualisant  $|\widehat{S}_1(f) + \widehat{S}_1(f + f_e) + \widehat{S}_1(f - f_e) + \widehat{S}_1(f + 2f_e) + \widehat{S}_1(f - 2f_e)|$ . L'implémentation est réalisée dans ??.

## 2.2.6 Périodisation de $s_1(t)$

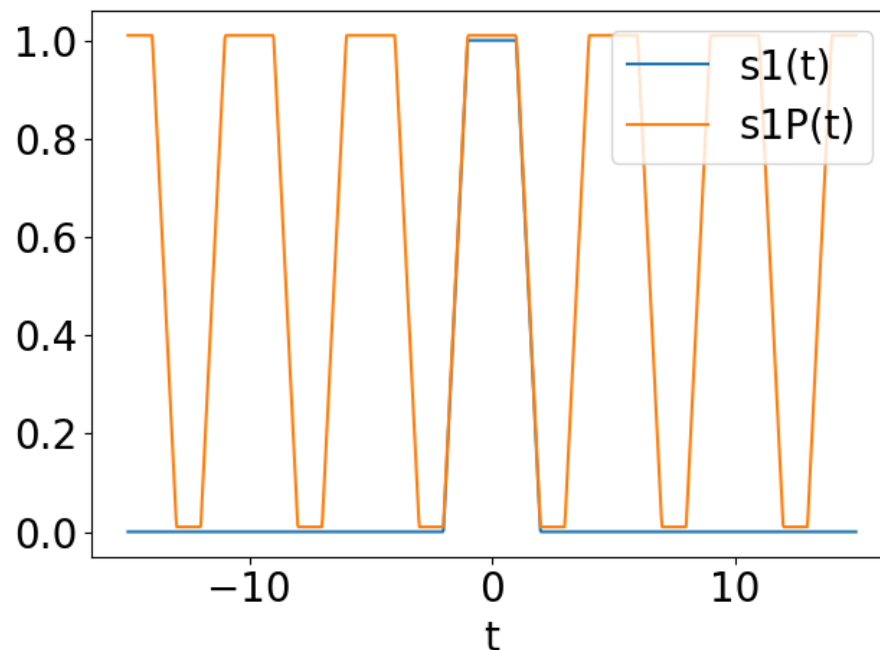


Figure 2.4: Signal  $s_1(t)$  en bleu périodisé en  $s_1^P(t)$  en répétant le motif  $[-\frac{5}{2}, \frac{5}{2}]$ .

La figure 2.4 montre  $s_1(t)$  et  $s_1^P(t)$  périodisés. La courbe de  $s_1^P(t)$  est légèrement surélevée pour montrer la différence avec  $s_1(t)$ . L'implémentation est dans ??.

## 2.2.7 Simulation de la transformée de Fourier

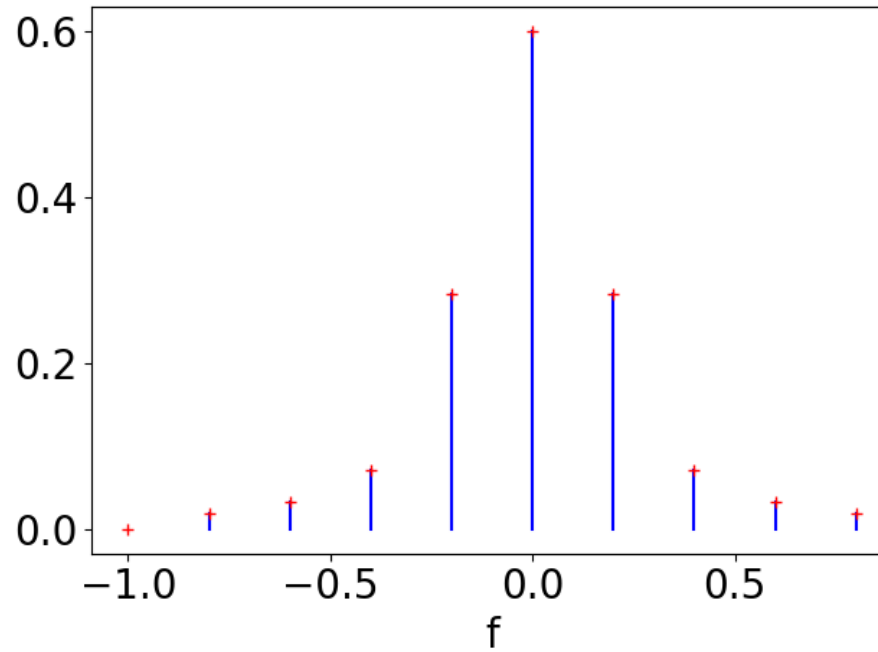


Figure 2.5: Signal  $s_1(t)$  en bleu périodisé en  $s_1^P(t)$  en répétant le motif  $[-\frac{5}{2}, \frac{5}{2}]$ .

La figure 2.5 montre des raies représentant les coefficients de la série de Fourier de  $s_1^P(t)$ . Comme la restriction de  $s_1^P(t)$  sur  $[-\frac{5}{2}, \frac{5}{2}]$  coïncide avec  $s_1(t)$ , ces raies peuvent être obtenues avec  $\widehat{S}_1(f)$  pour  $f_k = \frac{k}{T}$ , ces points sont les plus indiqués en rouge. L'implémentation est dans ??.



## 2.2.8 Simulation de la périodisation de la transformée de Fourier

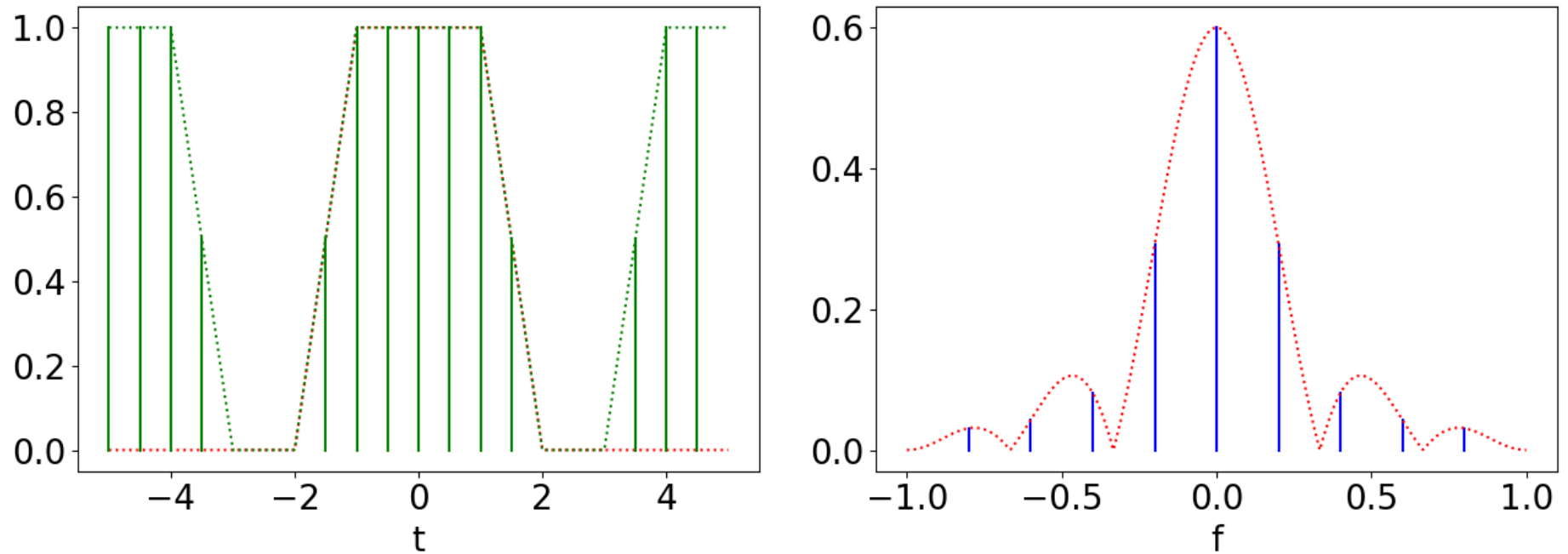


Figure 2.6

- Choisissez une fréquence d'échantillonnage  $f_{ec}$  tel que la période  $T$  soit un multiple de  $\frac{1}{f_{ec}}$ . On note  $N = T f_{ec}$ .
- On note  $s_1^{\#P}[n]$  le signal  $s_1^P(t)$  échantillonné à  $f_{ec}$ . Représentez  $s_1^{\#P}[n]$ ,  $s_1^{\#}[n]$ ,  $s_1^P(t)$  sur le même graphique.
- Représentez sur la même figure  $\text{TFD}[s_1^{\#P}[n]][k]$  et  $\frac{1}{N} \text{TFTD}[s_1^{\#P}[n]] \left( k \frac{f_{ec}}{N} \right)$

La figure 2.6 est implémentée dans ??

## 2.2.9 Simulation de l'erreur quadratique moyenne en fonction de $f_e$

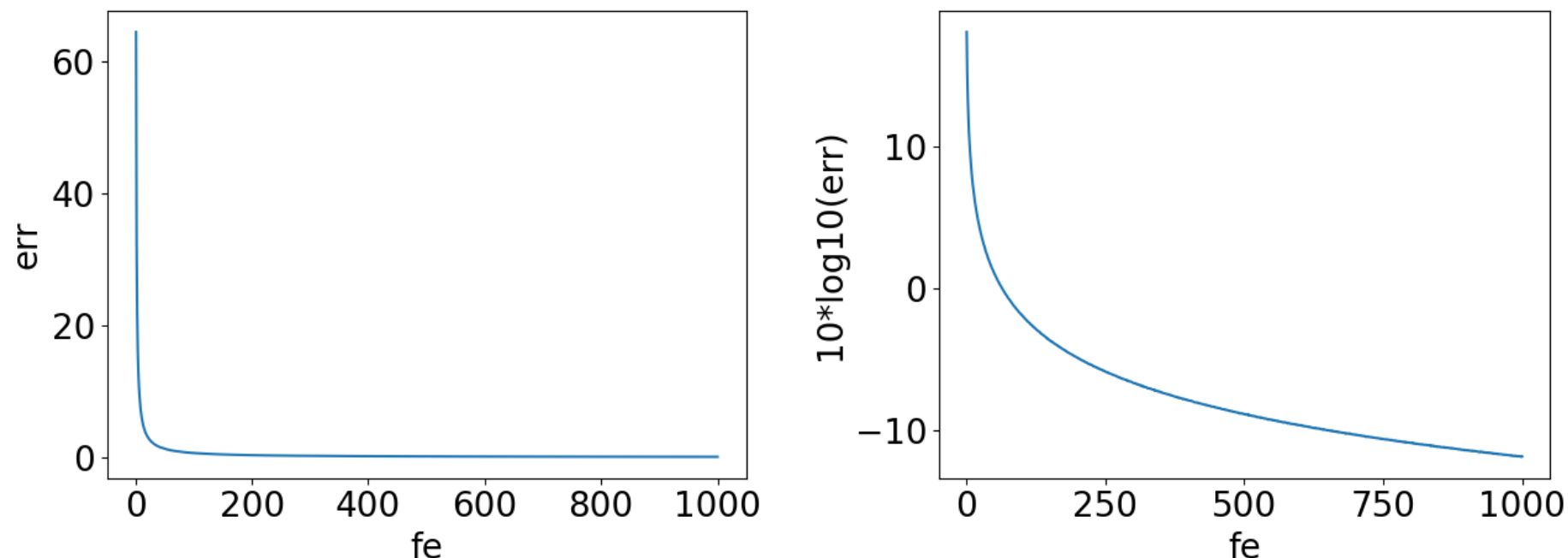


Figure 2.7: Signal étudié  $s_1(t)$  et deux échantillonnages

L'erreur quadratique moyenne se calcule normalement en faisant une intégrale sur le temps du carré de la différence. Mais pour que l'évaluation numérique soit moins sensible à des valeurs particulières utilisées pour approcher l'intégrale, je considère une variable aléatoire notée  $\tilde{T}$  qui suit une loi uniforme sur le support du signal (i.e. l'intervalle ou les intervalles sur lesquels le signal est non-nul).

La figure 2.7 montre la façon dont l'erreur quadratique dépend de la fréquence d'échantillonnage. L'implémentation est réalisée dans ??.

## 2.2.10 Reconstruction du signal par interpolation similaire à la formule de Shannon-Nyquist

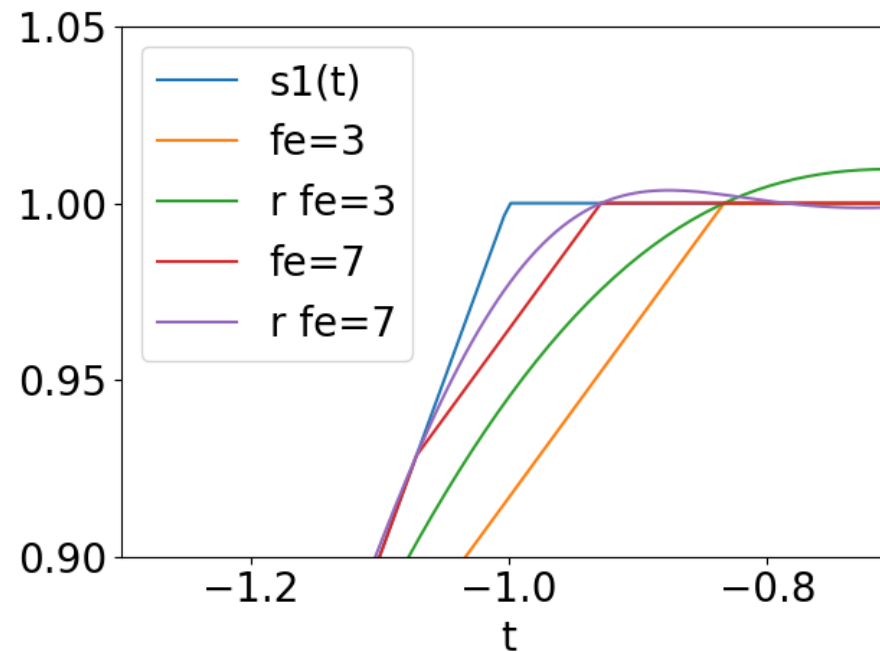


Figure 2.8: Signal étudié  $s_1(t)$  et deux échantillonnages

La figure 2.8 un zoom sur la reconstruction du signal. L'implémentation est réalisée dans ??.

### 2.2.11 Simulation de l'erreur quadratique moyenne en fonction de $f_e$ après reconstruction

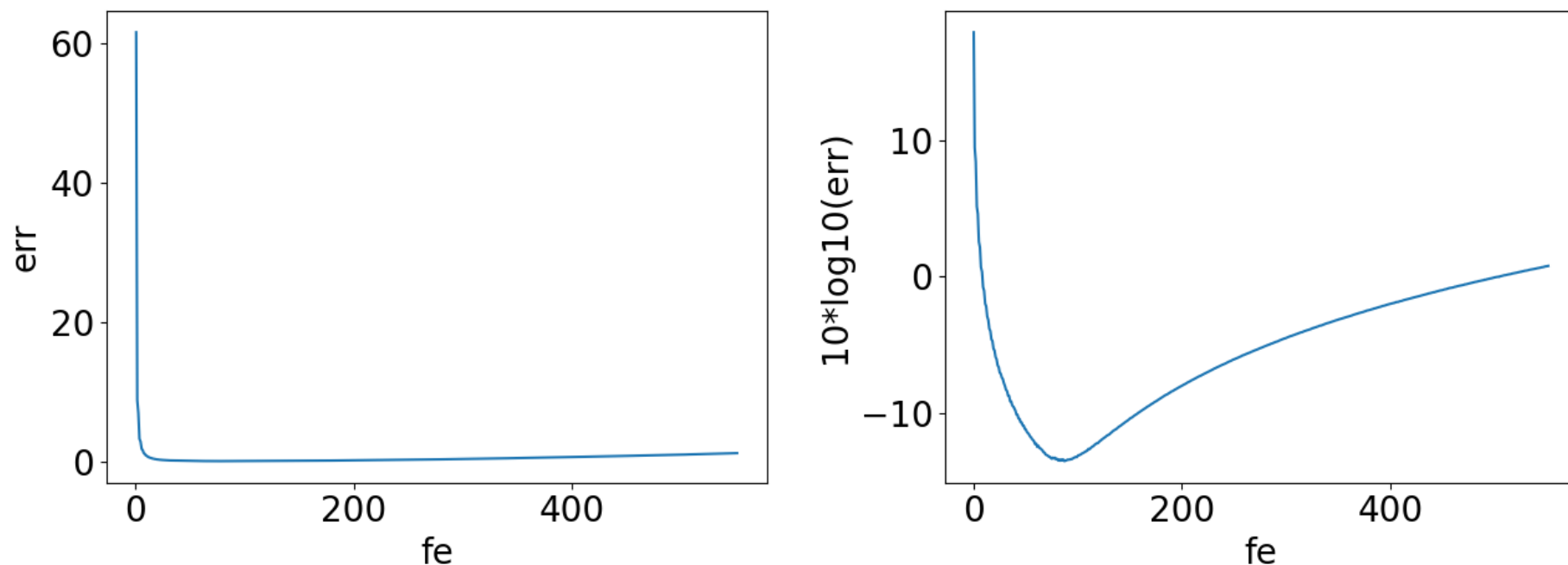


Figure 2.9: Signal étudié  $s_1(t)$  et deux échantillonnages

La figure 2.9 un zoom sur la reconstruction du signal. L'implémentation est réalisée dans ??.

# Chapter 3

## Séance 3 de travaux pratiques

### 3.1