Série de Fourier d'un signal périodique et système linéaire

1. Série de Fourier

1.a. Définition

Soit $u(t)$ une fonction périodique d'une variable réelle et à valeurs réelles, qui représente par exemple une grandeur physique dépendant du temps. Soit T la plus petite période de cette fonction. La fréquence fondamentale est par définition :

$$f_1 = \frac{1}{T} \tag{1}$$

On utilisera aussi la pulsation fondamentale :

$$\omega_1 = \frac{2\pi}{T} \tag{2}$$

La fonction u est supposée de classe C^∞ par morceaux. Le théorème de Fourier établit que cette fonction peut s'écrire comme la somme d'une série de fonctions, nommée série de Fourier :

$$u(t) = \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \cos(n\omega_1 t) + B_n \sin(n\omega_1 t) \tag{3}$$

Les nombres réels A_n et B_n sont les coefficients de Fourier. Ils peuvent être calculés par les intégrales suivantes (pour tout n entier positif ou nul) :

$$A_n = \frac{2}{T} \int_0^T u(t) \cos(n\omega_1 t)\, dt \tag{4}$$

$$B_n = \frac{2}{T} \int_0^T u(t) \sin(n\omega_1 t)\, dt \tag{5}$$

Pour $n = 0$ on a $B_0 = 0$ et :

$$\frac{A_0}{2} = \frac{1}{T} \int_0^T u(t)\, dt \tag{6}$$

qui est la valeur moyenne de u.
Si la fonction est de classe C^∞ alors la somme peut être arrêtée à un rang fini car les coefficients de Fourier sont nuls à partir d'un certain rang.
Pour les problèmes de traitement du signal ou de réponse des systèmes linéaires, on préfère généralement l'écriture suivante de la série de Fourier :

$$u(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos(n\omega_1 t + \Phi_n) \tag{7}$$

Le terme de rang n de la somme est une sinusoïde de fréquence nf_1, appelée *harmonique de rang n*. Le coefficient C_n est donc l'amplitude de l'harmonique de rang n et Φ_n est sa phase à l'origine. En développant le cosinus et en identifiant à la première forme, on montre que :

$$A_n = C_n \cos(\Phi_n) \tag{8}$$
$$B_n = -C_n \sin(\Phi_n) \tag{9}$$

ce qui conduit à définir un coefficient de Fourier complexe par :

$$\underline{C_n} = A_n - jB_n = \frac{2}{T}\int_0^T u(t)\exp\left(-jn\omega_1 t\right)\,dt \tag{10}$$

L'amplitude et la phase à l'origine de l'harmonique de rang n se déduisent de ce coefficient complexe :

$$C_n = |\underline{C_n}| \tag{11}$$
$$\Phi_n = arg(\underline{C_n}) \tag{12}$$

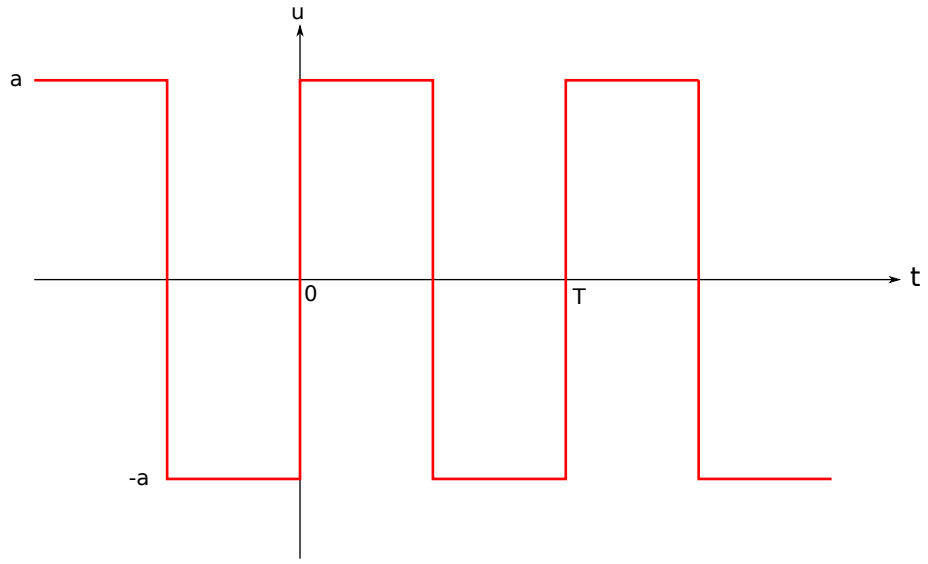
Pour $n = 0$, on a :

$$\frac{C_0}{2} = \frac{1}{T}\int_0^T u(t)\,dt \tag{13}$$

qui est la valeur moyenne de u.
Remarque sur le programme de MP : l'expression de la série de Fourier (7) doit être connue mais les intégrales permettant de calculer les coefficients de Fourier ne sont pas à connaître.

1.b. Exemple : signal carré

Soit le signal carré (ou signal créneau) défini sur la figure suivante :



Les coefficients de Fourier de cette fonction sont :

$$A_n = 0 \tag{14}$$

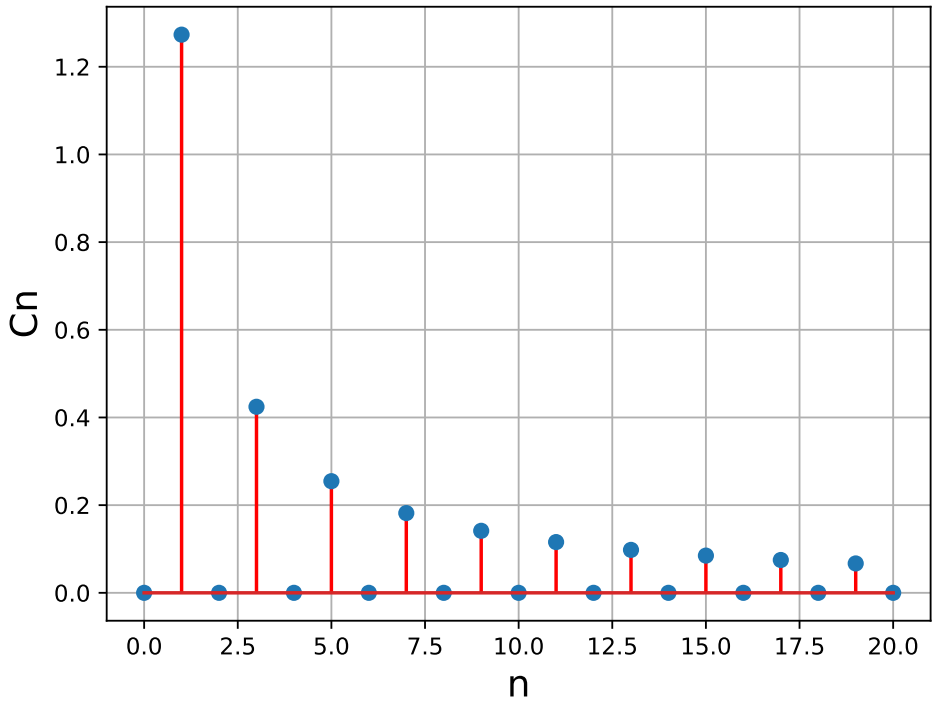
$$B_n = \frac{2a}{\pi n}\left(1-(-1)^n\right) \tag{15}$$

L'origine de t a été choisie sur un front du signal, ce qui a pour conséquence que la fonction est impaire et que les coefficients A_n sont nuls. On remarque que les coefficients de rang pair sont nuls. C'est une propriété due à la symétrie demi-onde :

$$u\left(t+\frac{T}{2}\right) = -u(t) \tag{16}$$

La représentation du spectre du signal consiste à tracer C_n pour les différents harmoniques. La fonction `Cn_carree` renvoie les coefficients $\underline{C_n}$ jusqu'au rang P, avec $C_0 = 0$.

```
from matplotlib.pyplot import *
import numpy as np

def Cn_carre(P,a=1):
    n = np.arange(1,P+1)
    cn = -1j*2*a/(n*np.pi)*(1-(-1)**n)
    return np.concatenate(([0],cn))

Cn_entree = Cn_carre
P=20
spectre = np.absolute(Cn_entree(P))
figure()
stem(np.arange(P+1),spectre,linefmt='r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```

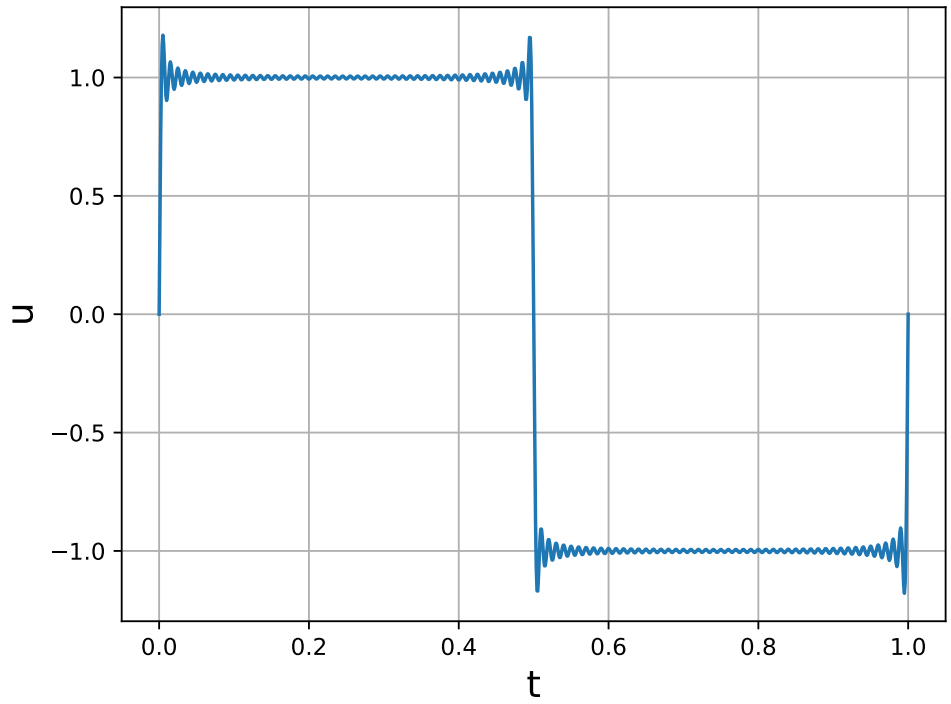


Le spectre du signal carré est caractérisé par une décroissance de l'amplitude des harmoniques en $1/n$, ce qui constitue une décroissance très lente, caractéristique des fonctions présentant une ou plusieurs discontinuités.
Pour reconstituer numériquement la fonction $u(t)$ à partir de ses coefficients de Fourier, on doit calculer la somme partielle de la série de Fourier, c'est-à-dire la somme stoppée à un rang P fini. Pour obtenir une bonne représentation graphique de cette somme, il faut échantillonner assez finement l'harmonique de rang P. Par exemple, si l'on veut 10 points par période pour cet harmonique, il faudra $N = 10P$ points sur l'intervalle $[0, T]$.
La fonction `somme` calcule la somme partielle à partir du tableau des coefficients de Fourier `Cn`, de la fréquence `f1` et du tableau des temps `t` :

```
def somme(Cn,f1,t):
    # Cn : tableau des coefficients de Fourier
    # f1 : fréquence fondamentale
    # t : tableau des instants
    P = len(Cn)-1
    C = np.absolute(Cn)
    Phi = np.angle(Cn)
    w1 = 2*np.pi*f1
    x = np.ones(len(t))*np.real(Cn[0])/2
    for n in range(1,P+1):
        x += C[n]*np.cos(n*w1*t+Phi[n])
    return x
```

Voici le tracé de la somme jusqu'au rang $P = 100$, pour une fréquence $f_1 = 1$:

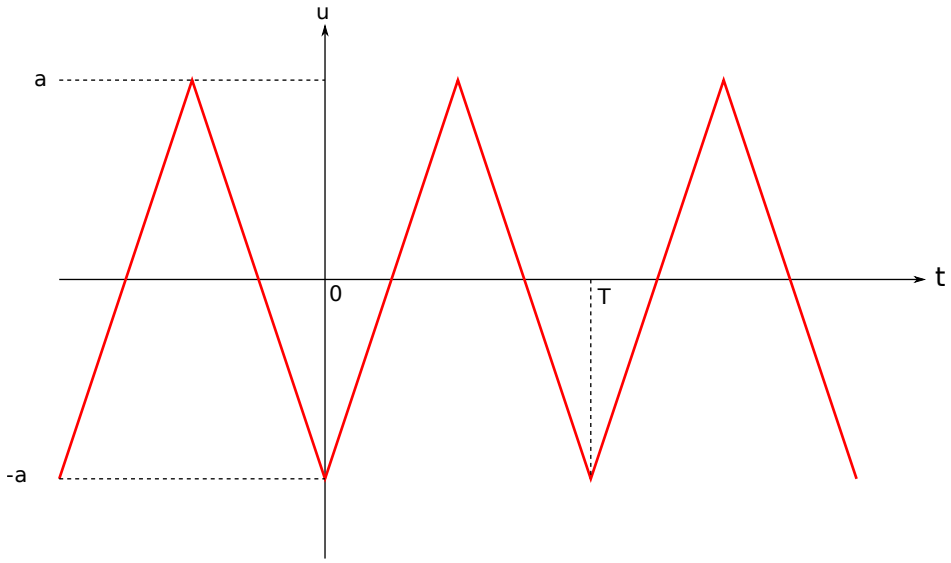
```
P=100
f1 = 1
t = np.linspace(0,1/f1,P*10)
Cn = Cn_entree(P)
figure()
plot(t,somme(Cn,f1,t))
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



Lorsque la fonction $u(t)$ comporte des discontinuités, la somme partielle présente un phénomène oscillatoire (phénomène de Gibbs) au voisinage de ces discontinuités, ce qui rend très mauvaise la représentation de la fonction par une somme partielle, même de rang très élevé.

1.c. Exemple : signal triangulaire

Considérons le signal défini sur la figure suivante :



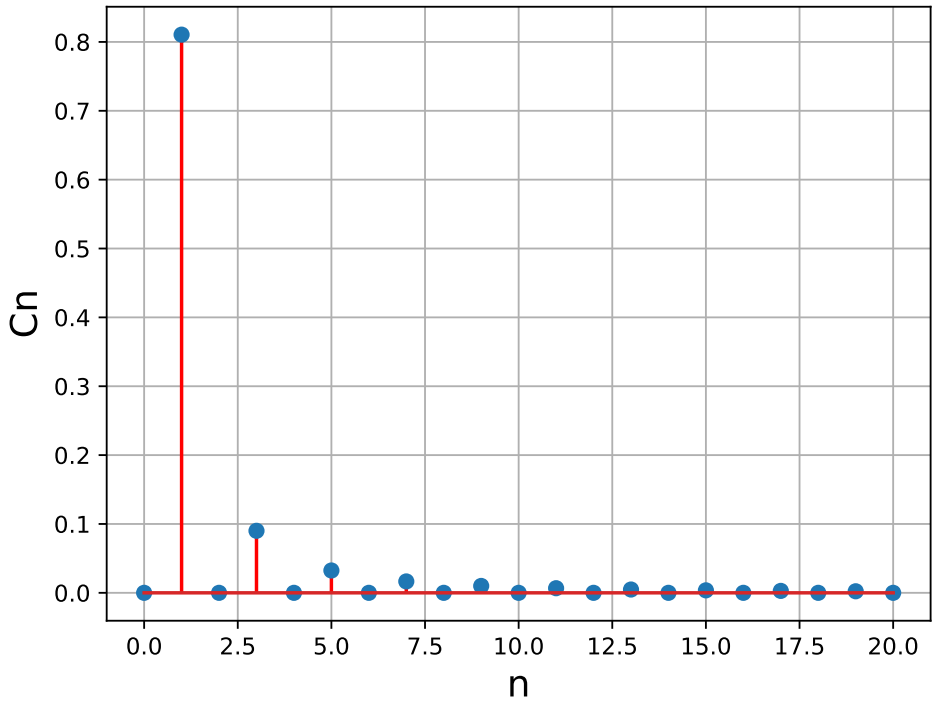
Avec l'origine du temps ainsi placée, la fonction est paire, ce qui fait que $B_n = 0$. Les coefficients de Fourier sont :

$$A_n = -\frac{4a}{\pi^2 n^2}(1-(-1)^n) \tag{17}$$

$$B_n = 0 \tag{18}$$

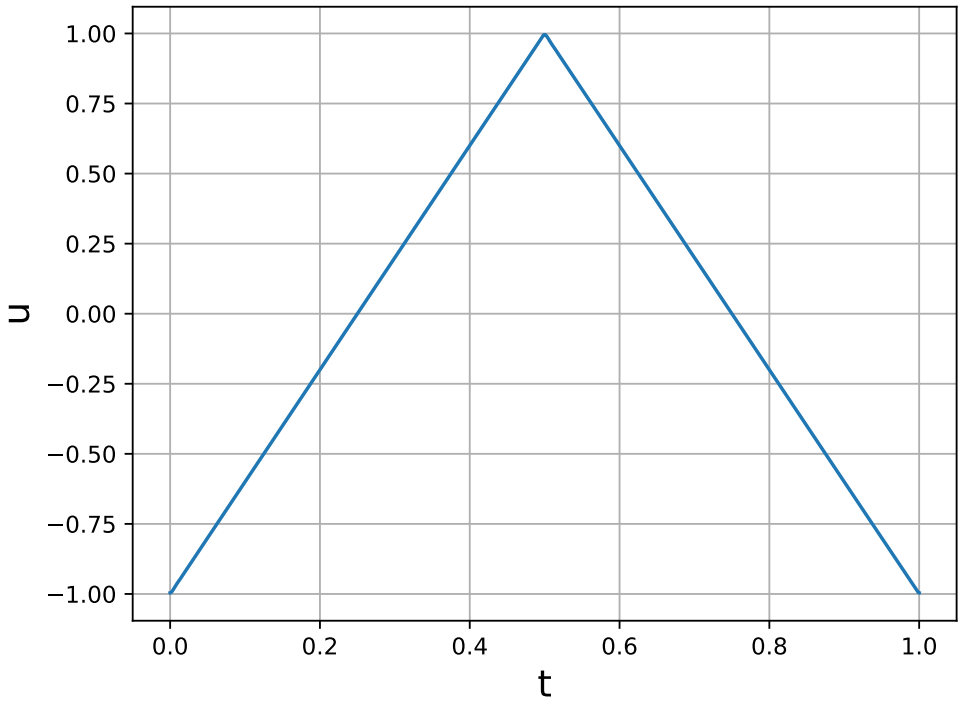
La série de Fourier ne comporte que des harmoniques de rangs impairs car ce signal possède la symétrie demi-onde (16). La décroissance est en $1/n^2$. Elle est beaucoup plus rapide que pour le signal carré car le signal triangulaire est continu.

```
def Cn_triangle(P,a=1):
    n = np.arange(1,P+1)
    cn = -4*a/(n*np.pi)**2*(1-(-1)**n)
    return np.concatenate(([0],cn))

Cn_entree = Cn_triangle
P=20
spectre = np.absolute(Cn_entree(P))
figure()
stem(np.arange(P+1),spectre,linefmt='r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```



Voici le tracé de la somme partielle jusqu'au rang P=100 :

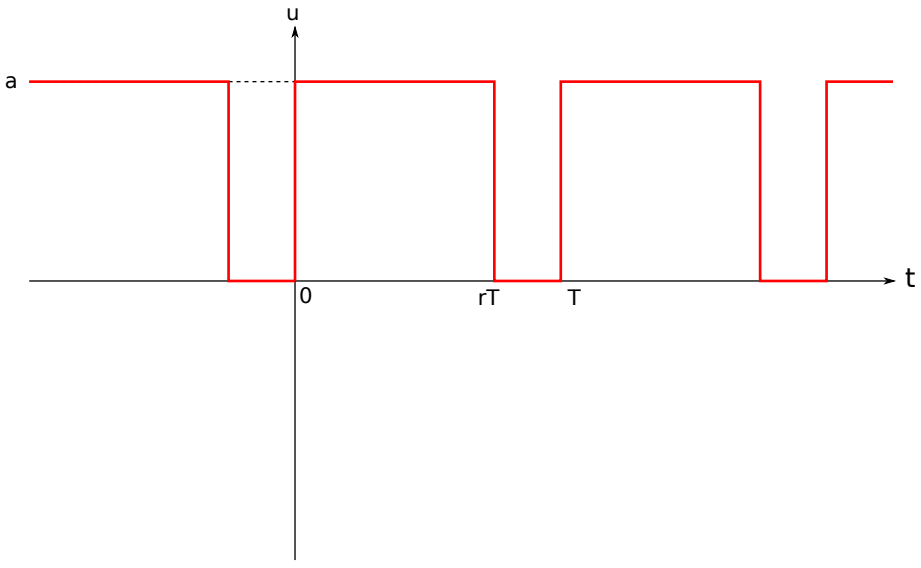
```
P=100
t = np.linspace(0,1/f1,P*10)
Cn = Cn_entree(P)
figure()
plot(t,somme(Cn,f1,t))
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



Contrairement au cas du signal carré, la somme partielle jusqu'au rang 100 donne une très bonne représentation de la forme du signal, ce qui signifie que la contribution des harmoniques de rang supérieur à 100 est négligeable.

1.d. Exemple : signal rectangulaire de rapport cyclique variable

Considérons le signal suivant :



Le coefficient r est le rapport cyclique, compris entre 0 et 1. La valeur moyenne de la fonction u est ra. En faisant varier le rapport cyclique, on fait donc varier la valeur moyenne.

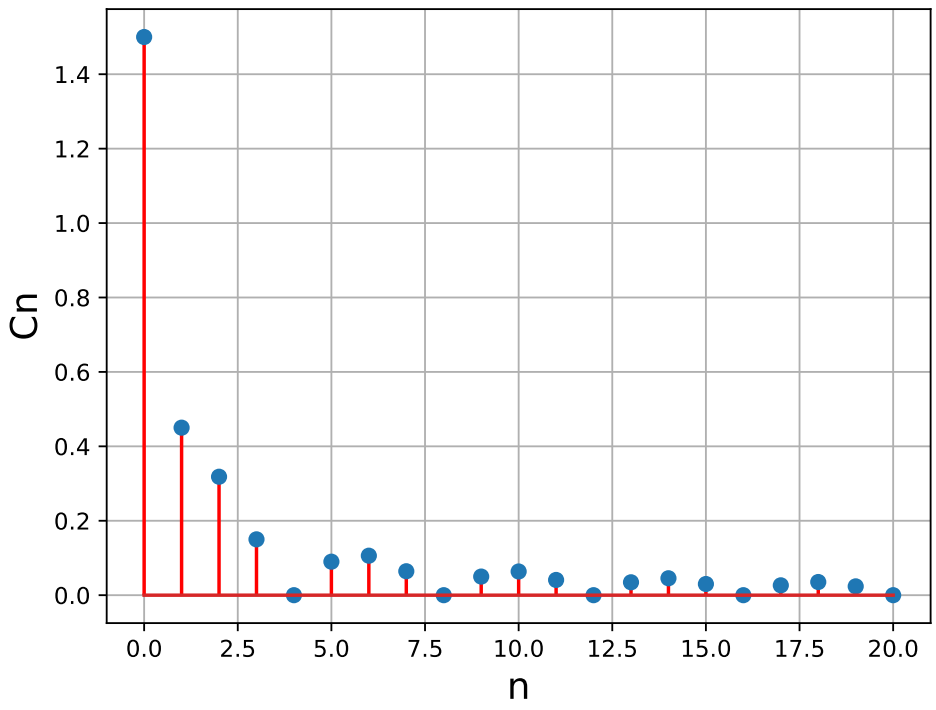
Voici les coefficients de Fourier complexes de cette fonction :

$$C_0 = 2ra \tag{19}$$

$$\underline{C_n} = 2ra \exp(-j\pi nr) \frac{\sin(\pi nr)}{\pi nr} \text{ pour } n \geq 1 \tag{20}$$

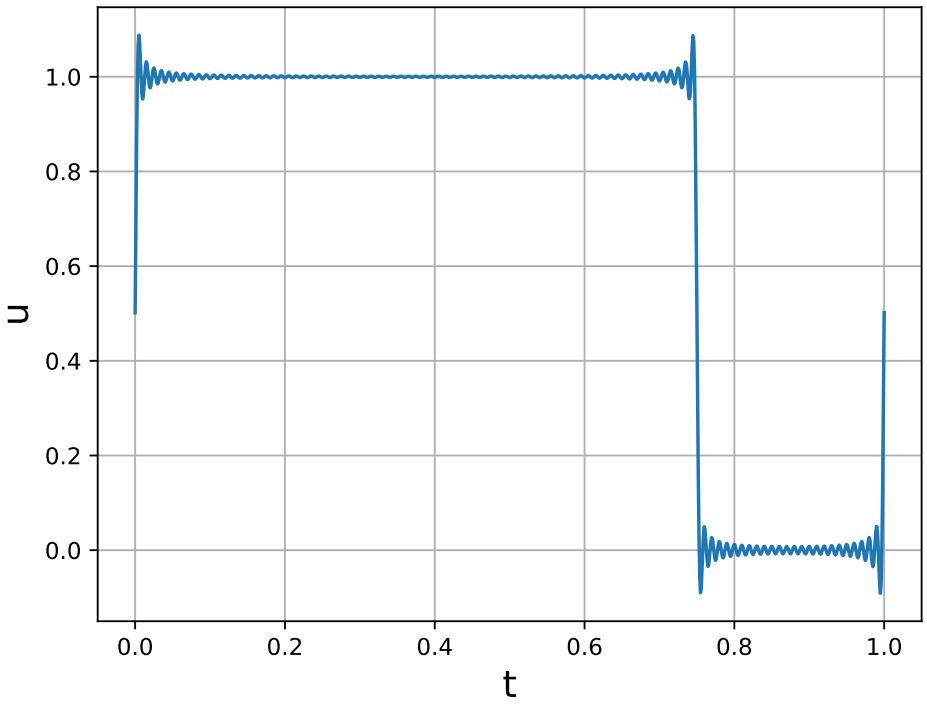
Voici le spectre de ce signal pour $r = 0{,}75$:

```
r = 0.75

def Cn_rectangle(P,a=1):
    # r en variable globale
    n = np.arange(1,P+1)
    u = np.pi*n*r
    cn = 2*r*a*np.exp(-1j*u)*np.sin(u)/u
    return np.concatenate(([2*r*a],cn))

Cn_entree = Cn_rectangle
P=20
spectre = np.absolute(Cn_entree(P))
figure()
stem(np.arange(P+1),spectre,linefmt='r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```



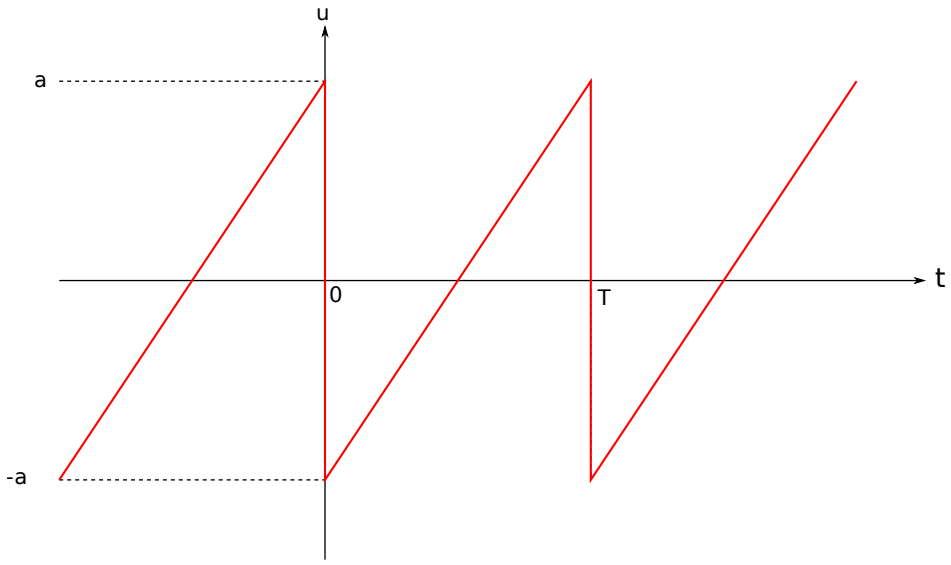
Voici la somme partielle jusqu'au rang 100 :

```
P=100
f1 = 1
t = np.linspace(0,1/f1,P*10)
Cn = Cn_entree(P)
figure()
plot(t,somme(Cn,f1,t))
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



1.e. Exemple : signal en dents de scie

Pour certaines applications, il est nécessaire de disposer d'un signal possédant tous les harmoniques (pairs et impairs). Le signal en dents de scie, représenté ci-dessous, ne vérifie par la symétrie demi-onde (16) :



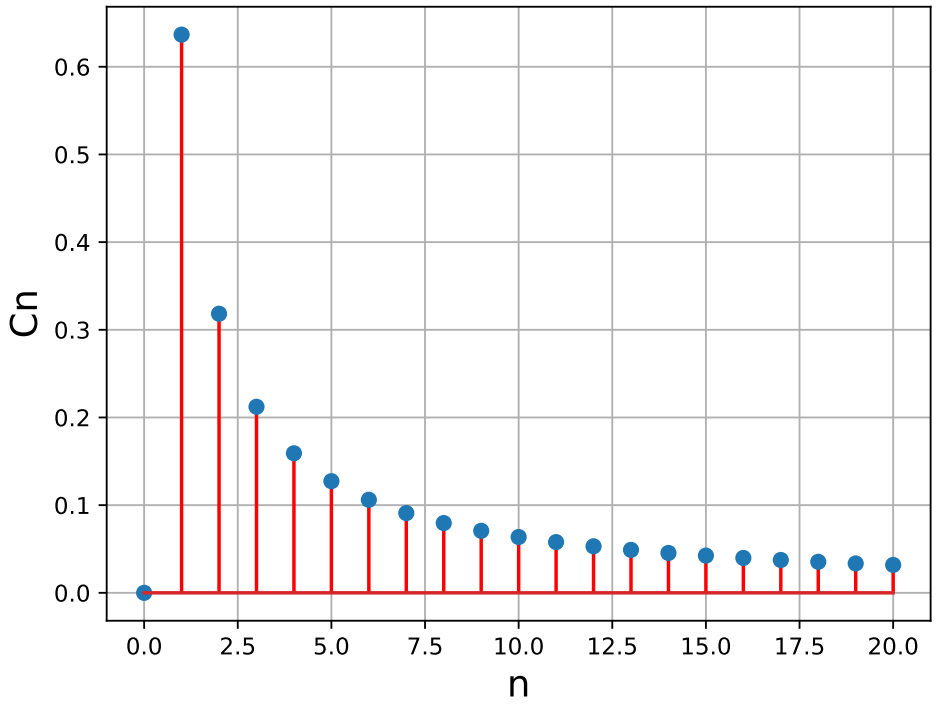
Ses coefficients de Fourier sont (pour $n > 0$) :

$$A_n = 0 \tag{21}$$

$$B_n = -\frac{2a}{\pi n} \tag{22}$$

Voici son spectre :

```
def Cn_dents(P,a=1):
    n = np.arange(1,P+1)
    cn = 1j*2*a/(n*np.pi)
    return np.concatenate(([0],cn))

Cn_entree = Cn_dents
P=20
spectre = np.absolute(Cn_entree(P))
figure()
stem(np.arange(P+1),spectre,linefmt='r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```



La présence d'une discontinuité implique une décroissance lente (en $1/n$). On a donc un spectre très riche en harmoniques, qui peut servir à générer une grande variété de signaux par filtrage.

2. Système linéaire

2.a. Définition

Soit un système effectuant la transformation d'un signal $e(t)$ (signal d'entrée) en un signal $s(t)$ (signal de sortie) :

$$e(t) \rightarrow s(t) \tag{23}$$

Considérons deux signaux différente et leurs transformés par le système :

$$e_1(t) \rightarrow s_1(t)$$
$$e_2(t) \rightarrow s_2(t)$$

Par définition, le système est linéaire si, quels que soient ces deux signaux et quelles que soient deux constantes α et β, on a :

$$\alpha e_1(t) + \beta e_2(t) \rightarrow \alpha s_1(t) + \beta s_2(t) \tag{24}$$

2.b. Action d'un système linéaire sur un signal périodique

Considérons l'action d'un système linéaire (par exemple un filtre électronique ou un système mécanique) sur un signal périodique. Si le signal d'entrée d'un système linéaire est une sinusoïde de pulsation ω alors le signal de sortie (en régime permanent) est aussi une sinusoïde de pulsation ω mais généralement d'amplitude différente et de phase à l'origine différente. On définit donc pour un système linéaire une *fonction de transfert harmonique* $\underline{H}(\omega)$, qui permet de calculer le gain et le déphasage en fonction de la pulsation :

$$G(\omega) = |\underline{H}(\omega)| \tag{25}$$
$$\varphi(\omega) = arg(\underline{H}(\omega)) \tag{26}$$

Notons $e(t)$ le signal d'entrée et $s(t)$ le signal de sortie du système linéaire. Si $e(t)$ est sinusoïdal, on a :

$$e(t) = C\cos(\omega t + \Phi) \tag{27}$$
$$s(t) = CG(\omega)\cos\left(\omega t + \Phi + \varphi(\omega)\right) \tag{28}$$

Si le signal $e(t)$ est périodique, nous utilisons sa série de Fourier :

$$e(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos(n\omega_1 t + \Phi_n) \tag{29}$$

La propriété de linéarité du système implique que si l'entrée est une combinaison linéaire de fonctions sinusoïdales (comme l'est la série de Fourier) alors la sortie est la même combinaison linéaire des sorties correspondant aux différentes fonctions sinusoïdales :

$$s(t) = \frac{C_0}{2}G(0)\cos(\varphi(0)) + \sum_{n=1}^{\infty} C_n G(n\omega_1)\cos(n\omega_1 t + \Phi_n + \varphi(n\omega_1)) \tag{30}$$

En conséquence, l'harmonique de rang n de la sortie est égal à l'harmonique de rang n de l'entrée multiplié par le gain à la pulsation correspondante ($n\omega_1$) et déphasé du déphasage à cette pulsation. Le terme de fréquence nulle, c'est-à-dire la valeur moyenne du signal, est multiplié par le gain à pulsation nulle et par le cosinus du déphasage à pulsation nulle. Une conséquence de ce résultat est que si un harmonique est absent dans le signal d'entrée alors il est aussi absent dans le signal de sortie. Les systèmes non linéaires sont au contraire susceptibles de faire apparaître des harmoniques dans le signal. Inversement, un harmonique présent dans le signal d'entrée peut être absent en sortie si le gain à la pulsation correspondante est nul (ou très faible).
Un système linéaire est entièrement défini par sa *fonction de transfert harmonique* $\underline{H}(\omega)$. Nous venons en effet de démontrer que la connaissance de cette fonction permet de déterminer la réponse à un signal périodique quelconque. Ce résultat se généralise à un signal quelconque (non périodique) au moyen de la transformée de Fourier.

2.c. Exemple : filtrage d'un signal carré

Considérons comme exemple un filtre passe-bas du premier ordre agissant sur un signal carré auquel une valeur moyenne est ajoutée. Le filtre passe-bas est défini par la fonction de transfert harmonique suivante :

$$\underline{H}(\omega) = \frac{1}{1 + j\frac{\omega}{\omega_c}} \tag{31}$$

où ω_c est la pulsation de coupure à -3dB.

```
def H(w,param):
    # w : pulsation
    # param : liste de paramètres
    wc = param[0]
    return 1/(1+1j*w/wc)
```

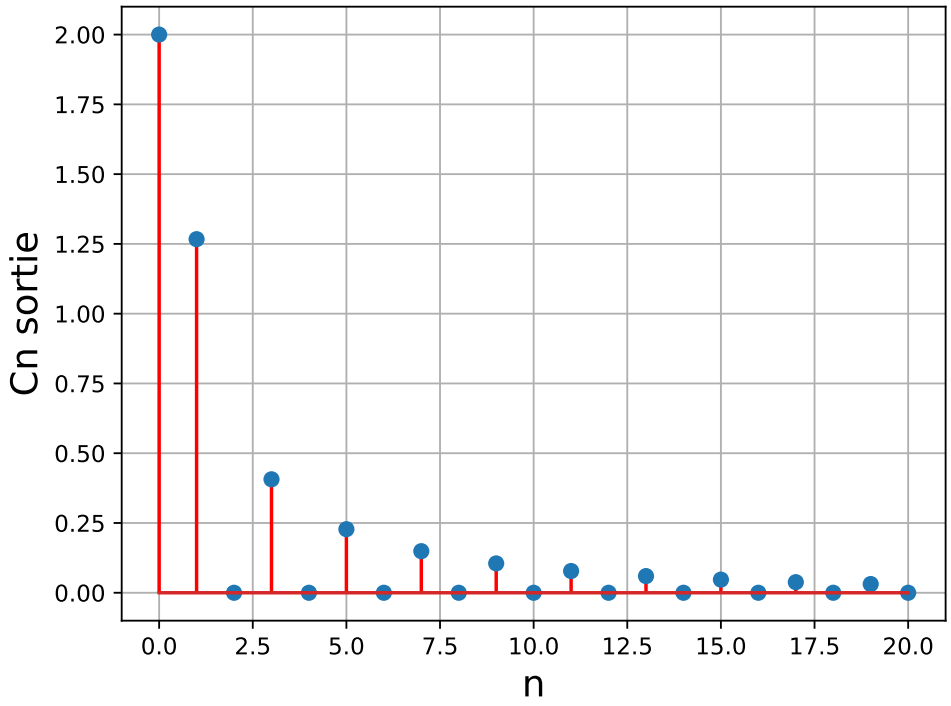
Les coefficients de Fourier complexes de la sortie sont obtenus en les multipliant par la valeur de la fonction de transfert aux fréquences correspondantes. La fonction suivante calcule les coefficients de Fourier de la sortie :

```
def Cn_sortie(Cn,f1,H,param):
    # Cn : tableau des coefficients de Fourier de l'entrée
    # f1 : fréquence fondamentale
    # H : fonction de transfert
    # param : paramètres de la fonction de transfert
    P = len(Cn)-1
    n = np.arange(0,P+1)
    return Cn*H(n*2*np.pi*f1,param)
```

Voici le spectre du signal de sortie :

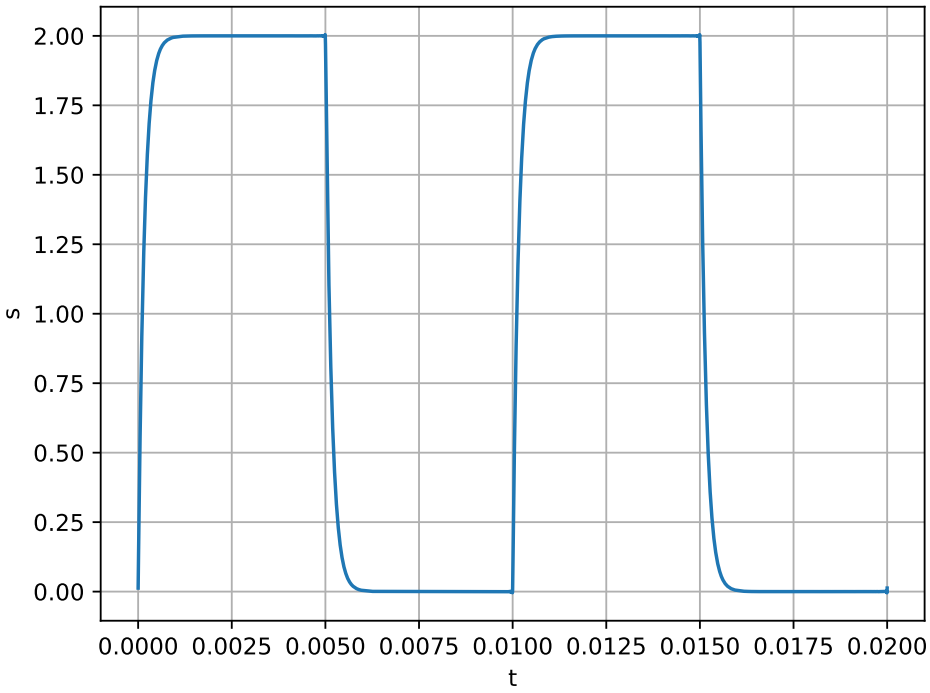
```
Cn_entree = Cn_carre
f1 = 100
fc = 1000
```

```
param = [2*np.pi*fc]
P=20
Cn = Cn_entree(P)
Cn[0] =2 # valeur moyenne = 1
Cn_s = Cn_sortie(Cn,f1,H,param)
figure()
stem(np.arange(P+1),np.absolute(Cn_s),linefmt='r')
xlabel('n',fontsize=16)
ylabel('Cn sortie',fontsize=16)
grid()
```



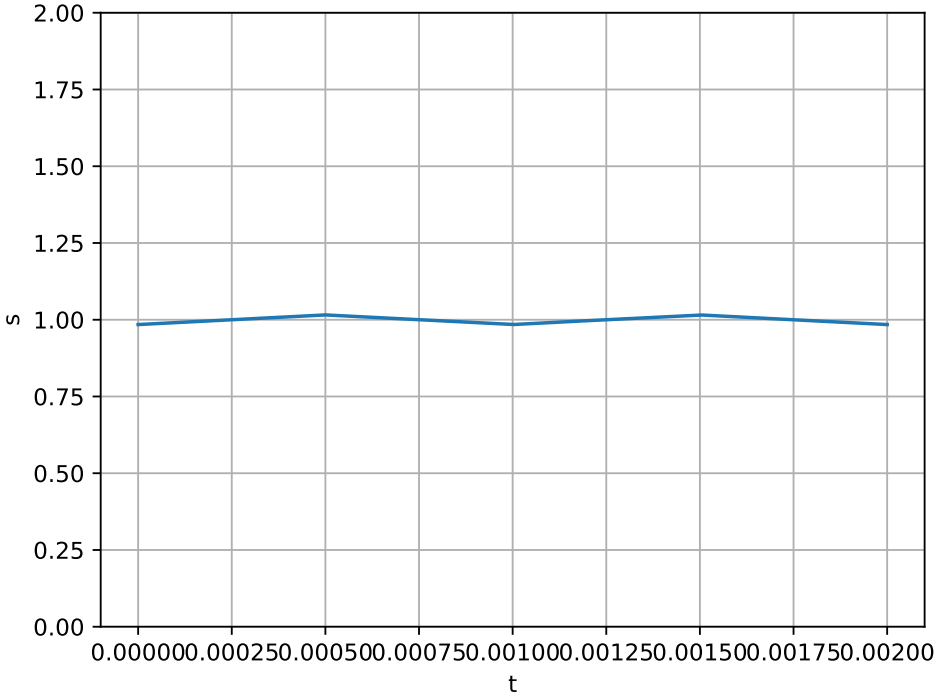
Finalement, on reconstitue le signal de sortie en calculant la somme partielle :

```
P = 500
Cn = Cn_entree(P)
Cn[0] = 2
Cn_s = Cn_sortie(Cn,f1,H,param)
t = np.linspace(0,2/f1,P*20)
figure()
plot(t,somme(Cn_s,f1,t),label='P=%d'%P)
grid()
xlabel('t')
ylabel('s')
```



Le filtrage passe-bas a pour effet de remplacer les fronts de pente infini par des variations continues. Contrairement au signal d'entrée, la somme partielle de rang 100 suffit largement à représenter le signal de sortie avec une très bonne précision. En effet, le filtrage passe-bas a pour effet d'augmenter la vitesse de décroissance des harmoniques : les harmoniques de rang supérieur à 100 sont complètement négligeables dans le signal de sortie.
Appliquons un filtrage passe-bas qui permet d'obtenir en sortie la valeur moyenne du signal. Il faut pour cela que la fréquence de coupure soit beaucoup plus faible que la fréquence fondamentale du signal. Voici le filtrage lorsque la fréquence de coupure est 100 fois plus faible, ce qui permet d'avoir un gain de -40 dB pour le fondamental.

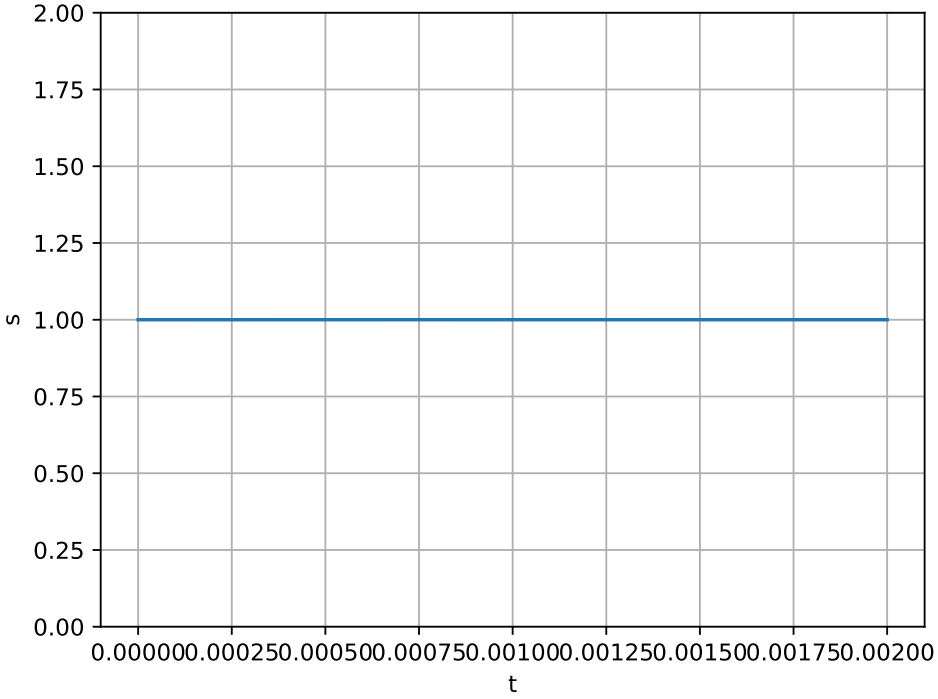
```
f1 = 1000
fc = 10
param = [2*np.pi*fc]
P = 500
Cn = Cn_entree(P)
Cn[0] = 2
Cn_s = Cn_sortie(Cn,f1,H,param)
t = np.linspace(0,2/f1,P*20)
figure()
plot(t,somme(Cn_s,f1,t),label='P=%d'%P)
grid()
xlabel('t')
ylabel('s')
ylim(0,2)
```



Le signal en sortie est presque constant, égal à la valeur moyenne du signal d'entrée. Il reste néanmoins une légère ondulation en sortie car les harmoniques (à partir du rang 1) ne sont pas assez atténués. Un filtre du second ordre sera plus efficace :

$$\underline{H}(\omega) = \frac{1}{1 + j\sqrt{2}\frac{\omega}{\omega_c} - \left(\frac{\omega}{\omega_c}\right)^2} \tag{32}$$

```
def H(w,param):
    wc = param[0]
    return 1/(1+1j*np.sqrt(2)*w/wc-(w/wc)**2)

f1 = 1000
fc = 10
param = [2*np.pi*fc]
P = 500
Cn = Cn_entree(P)
Cn[0] = 2
Cn_s = Cn_sortie(Cn,f1,H,param)
t = np.linspace(0,2/f1,P*20)
figure()
plot(t,somme(Cn_s,f1,t),label='P=%d'%P)
grid()
xlabel('t')
ylabel('s')
ylim(0,2)
```



La sortie est bien constante (l'ondulation est négligeable), égale à la valeur moyenne du signal d'entrée.