Série de Fourier d'un signal périodique et système linéaire

1. Série de Fourier

1.a. Définition

Soit u une fonction périodique d'une variable réelle (notée t) 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, appelée 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 :

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

Remarque : une autre définition courante du coefficient de Fourier complexe ne comporte pas le facteur 2, ce qui conduit à un facteur 2 dans l'expression (7). Nous adopterons cette définition, qui a l'avantage de donner directement l'amplitude des harmoniques, avec C_0 égal au double de la valeur moyenne.

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 :

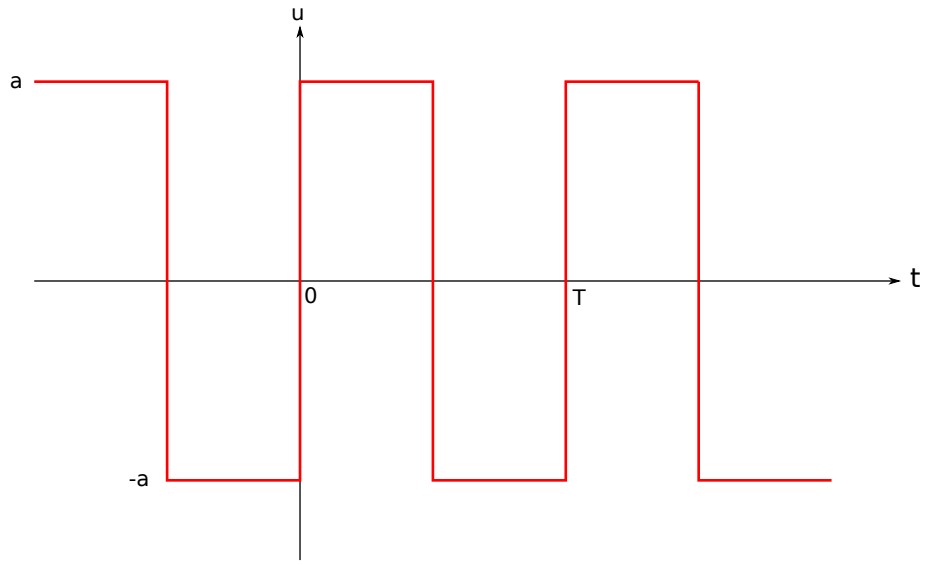
$$\frac{C_0}{2} = \frac{1}{T}\int_0^T u(t)\,dt \tag{13}$$

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 :

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

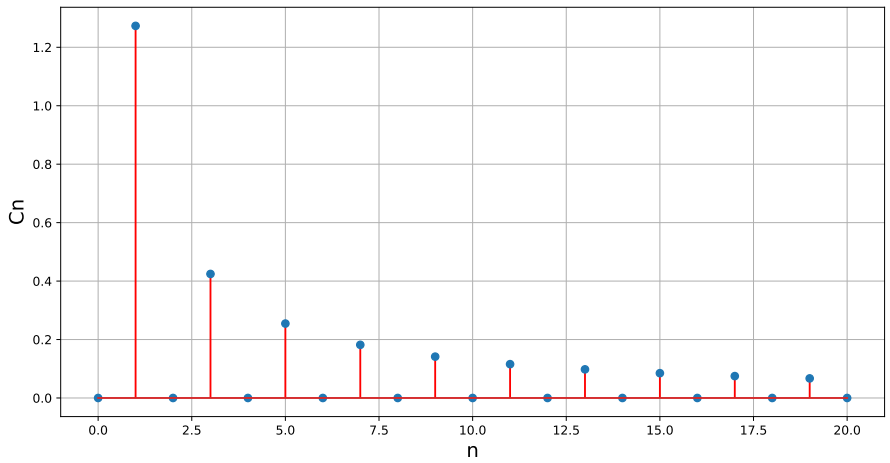
$$B_n = \frac{2a}{\pi n}\left(1-(-1)^n\right) \tag{15}$$

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 :

```
from matplotlib.pyplot import *
import numpy as np
def C(n):
    return 2/(np.pi*n)*(1-(-1)**n)
spectre = [0]
P=20
for n in range(1,P+1):
    spectre.append(C(n))
figure(figsize=(12,6))
stem(np.arange(P+1),spectre,'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. Une décroissance très lente de ce type se produit lorsque la fonction présente une ou plusieurs discontinuités.

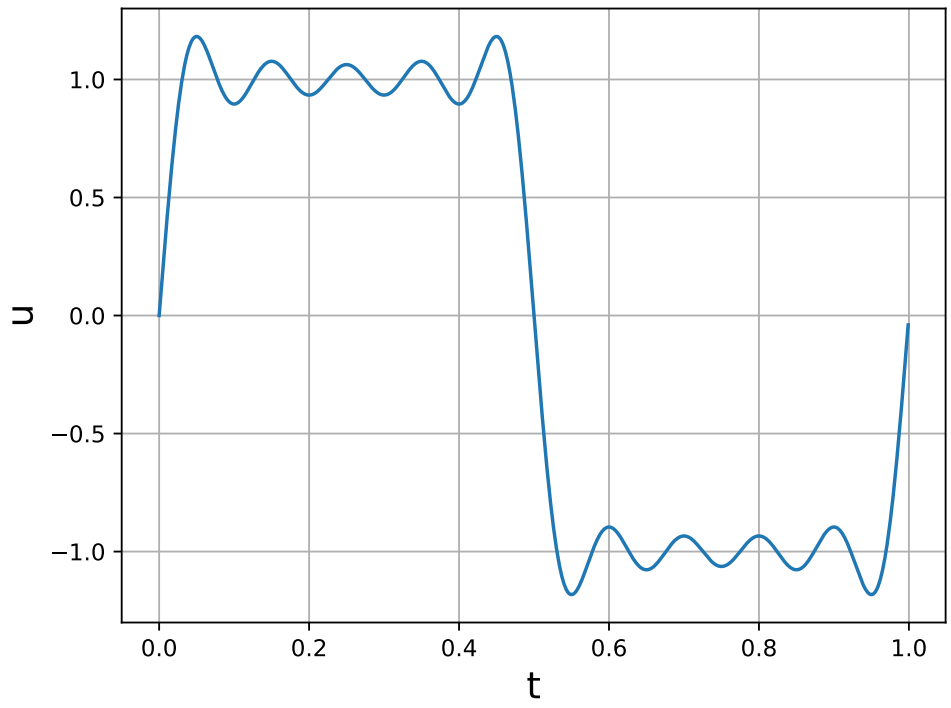
Il est intéressant de tracer la somme partielle de la série de Fourier, c'est-à-dire la somme stoppée à un rang P fini. Pour obtenir une bonne représentation graphique de cette somme,

il faut échantillonner assez finement l'harmonique de rang P. Par exemple, si l'on veut 100 points par période pour cet harmonique, il faudra $N = 100P$ points sur l'intervalle $[0, T]$. Pour chacun de ces points, il faut calculer les P termes de la somme, ce qui fait au total $100P^2$ termes cosinus à calculer. Lorsque P est grand, il est préférable d'utiliser un algorithme plus efficace que le calcul direct de tous ces termes. La meilleure méthode consiste à calculer la somme partielle par transformée de Fourier discrète, au moyen de l'algorithme de transformée de Fourier rapide (dont la complexité est $N \ln(N)$). Dans ce document, nous donnons la méthode sans la justifier. On commence par remplir un tableau contenant les coefficients de Fourier complexes $\underline{C_n}$ jusqu'au rang P. Ce tableau est de taille $100P$ mais tous les coefficients pour $n > P$ sont nuls.

```
P=10
N=100*P
Cn = np.zeros(N,dtype=np.complex)
for n in range(1,P):
    Cn[n]=-1j*C(n)
```

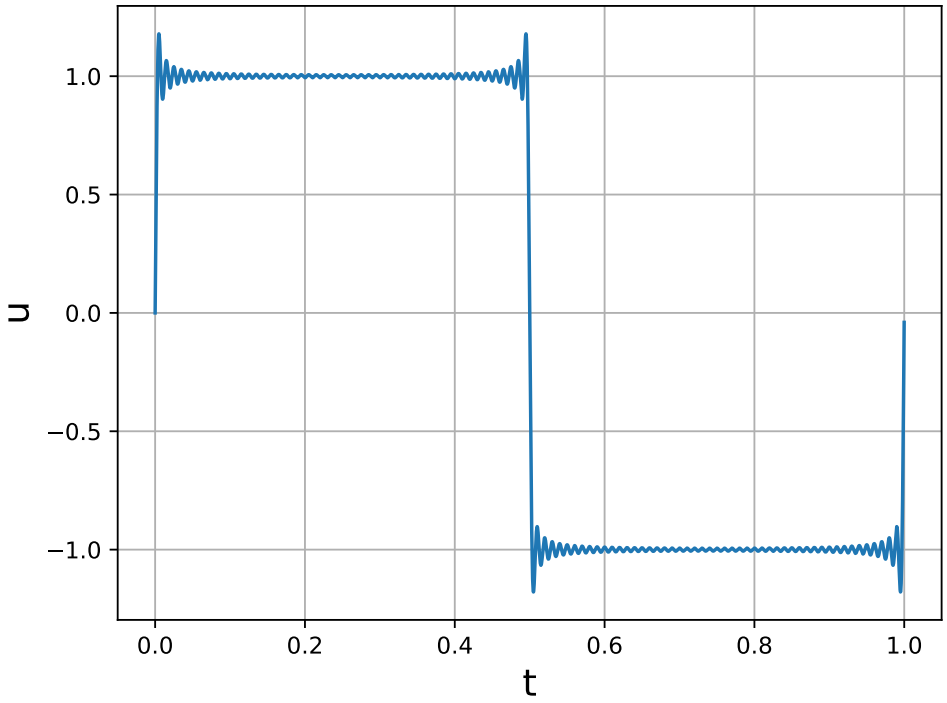
La transformée de Fourier discrète inverse permet de calculer les $100P$ échantillons de la somme partielle sur l'intervalle [0,T]. On utilise pour cela la fonction `numpy.fft.ifft`.

```
from numpy.fft import ifft
signal = ifft(Cn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



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

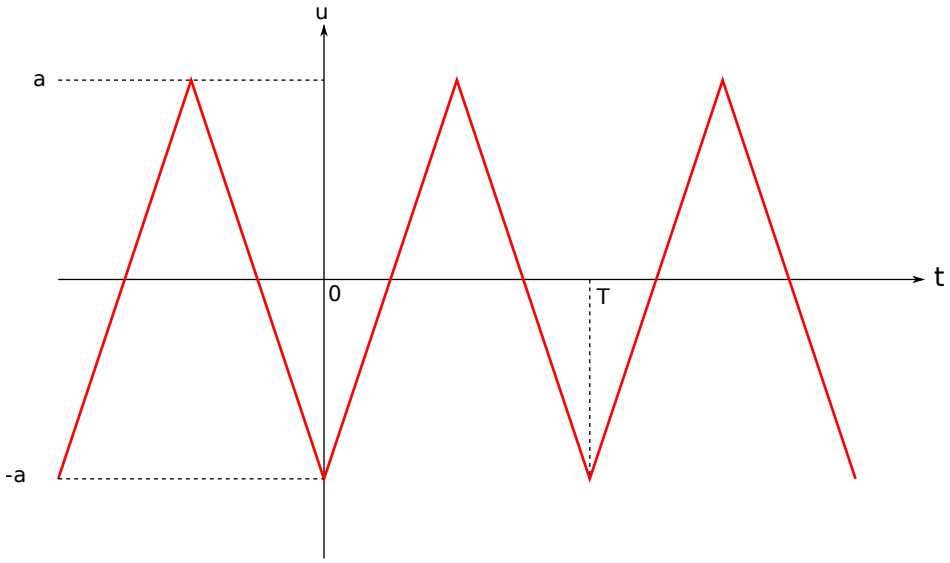
```
P=100
N=100*P
Cn = np.zeros(N,dtype=np.complex)
for n in range(1,P):
    Cn[n]=-1j*C(n)
signal = ifft(Cn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



Dans le cas du signal carré (qui présente des discontinuités), la somme partielle présente un phénomène oscillatoire (appelé 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 :

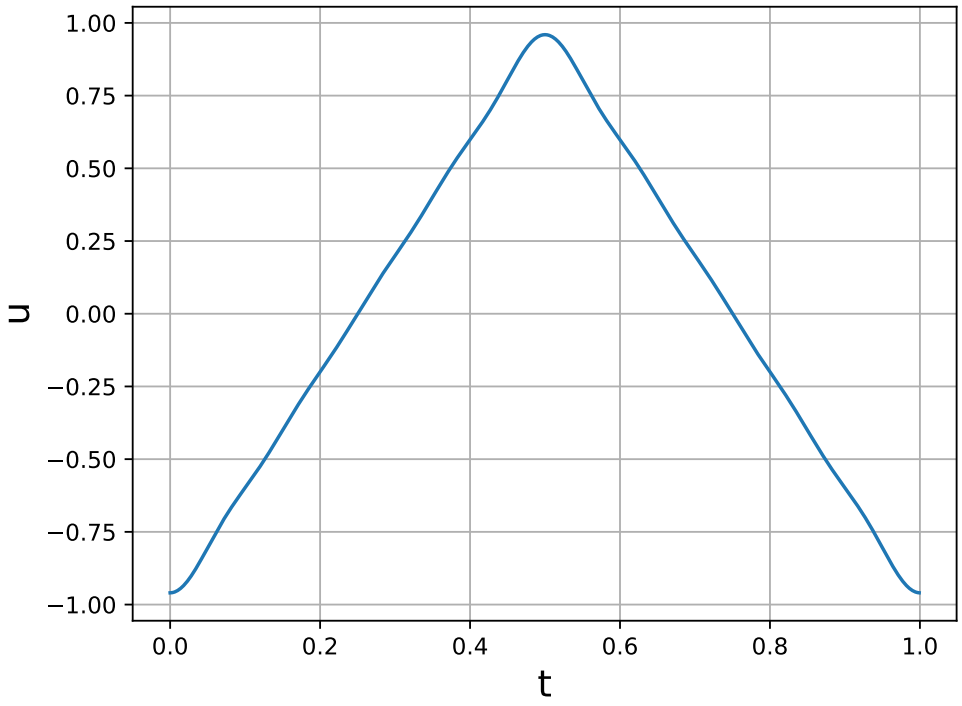
$$A_n = -\frac{4a}{\pi^2 n^2}(1-(-1)^n) \tag{17}$$

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

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.

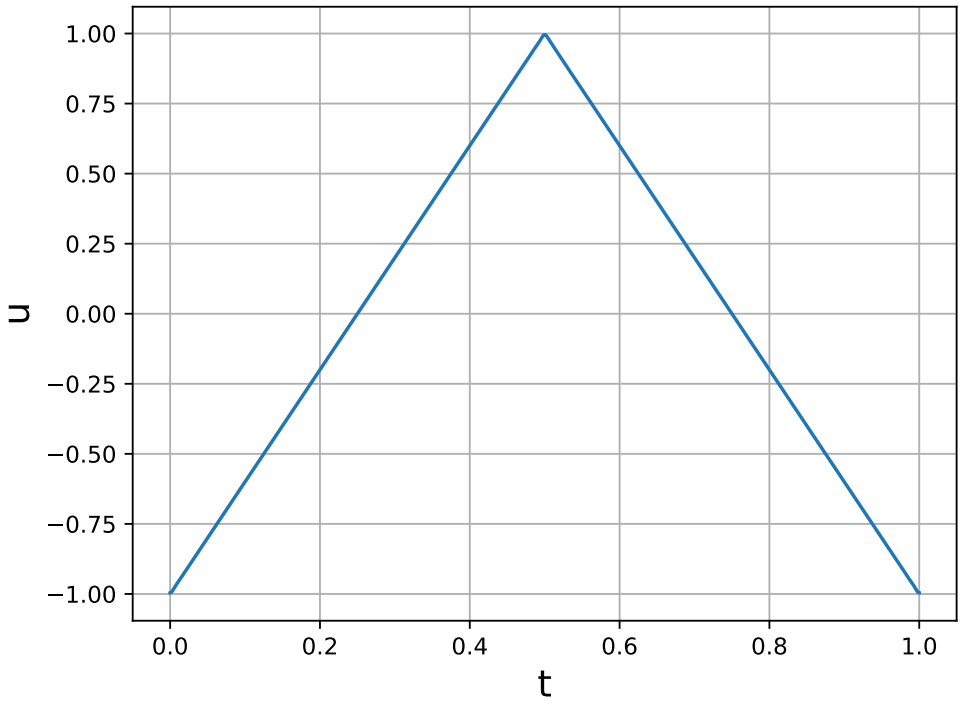
Voici le tracé de la somme partielle jusqu'au rang P=10 :

```
P=10
N=100*P
Cn = np.zeros(N,dtype=np.complex)
for n in range(1,P):
    Cn[n]=-4/(np.pi**2*n**2)*(1-(-1)**n)
signal = ifft(Cn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



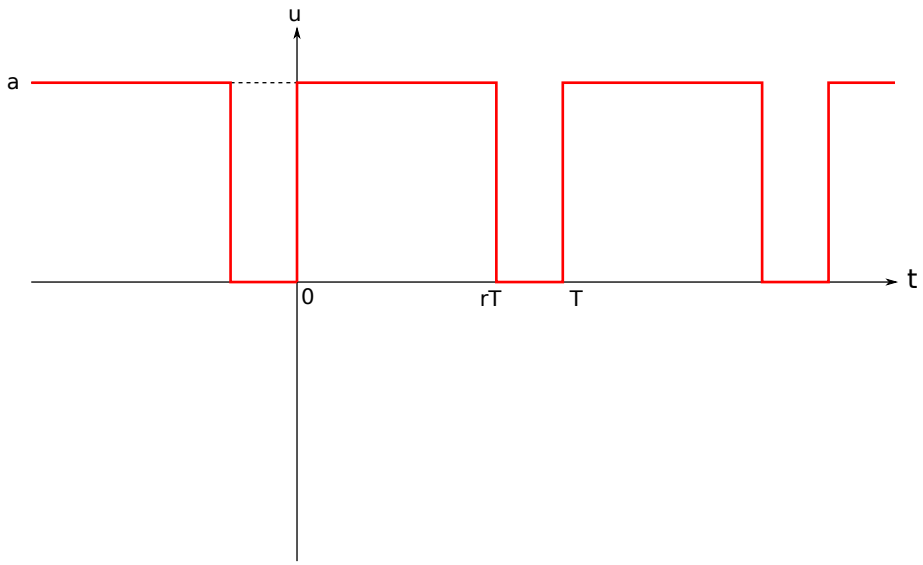
Contrairement au cas du signal carré, la somme partielle jusqu'au rang 10 donne déjà une bonne représentation de la forme du signal, ce qui signifie que la contribution des harmoniques de rang supérieur à 10 est faible. Voici le tracé pour P=100 :

```
P=100
N=100*P
Cn = np.zeros(N,dtype=np.complex)
for n in range(1,P):
    Cn[n]=-4/(np.pi**2*n**2)*(1-(-1)**n)
signal = ifft(Cn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
grid()
```



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.

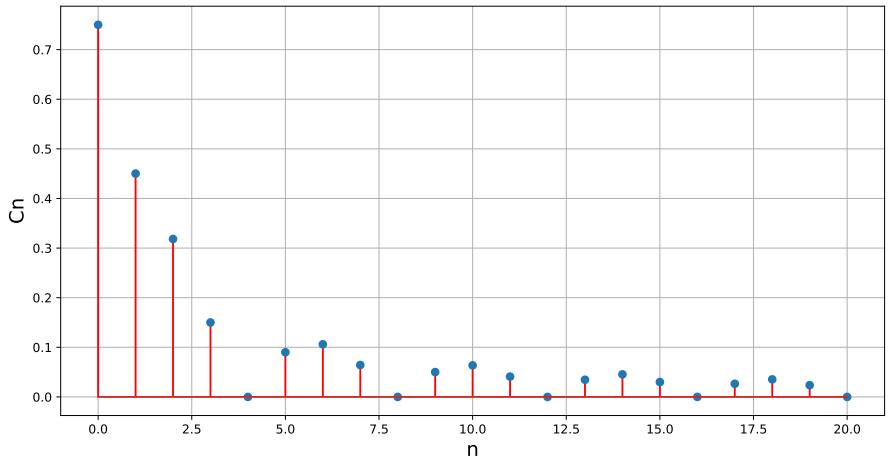
Voici les coefficients de Fourier complexes de cette fonction :

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

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

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

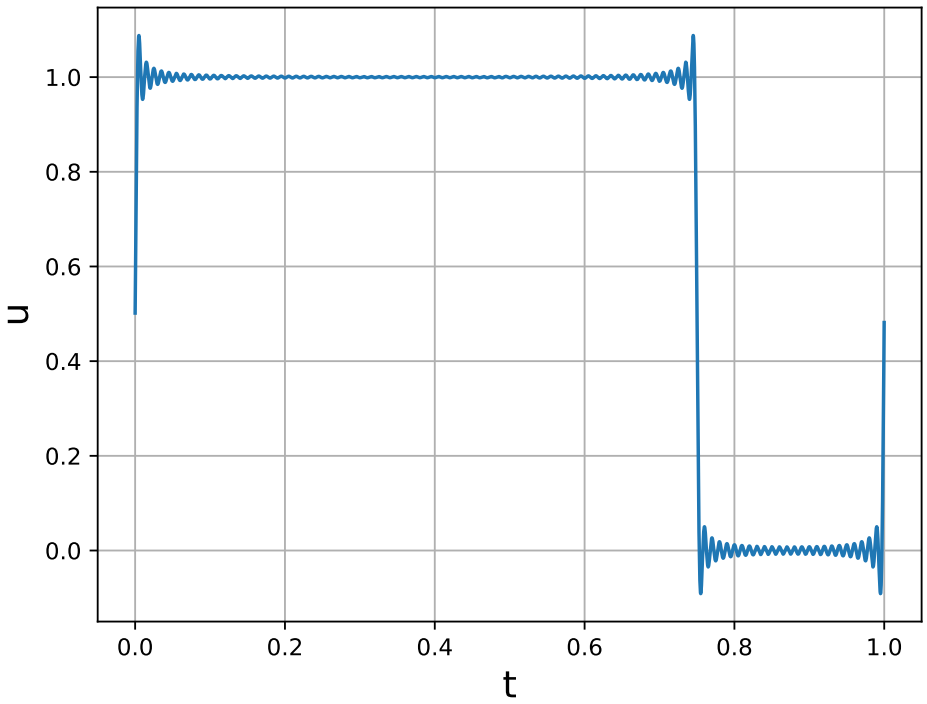
```
r=0.75
def C(n):
    return 2*r*np.exp(-1j*np.pi*n*r)*np.sin(np.pi*n*r)/(np.pi*n*r)
spectre = [r]
P = 20
for n in range(1,P+1):
    spectre.append(np.abs(C(n)))
figure(figsize=(12,6))
stem(np.arange(P+1),spectre,'r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```



On remarquera que dans cette représentation le terme de rang 0 représenté est $C_0/2$, c'est-à-dire la valeur moyenne. Ce spectre est caractérisé par une décroissance très lente en 1/n. Pour calculer la somme partielle par transformée de Fourier discrète, il faut placer la valeur moyenne au rang 0 (et non pas son double). Voici la somme partielle jusqu'au rang 100 :

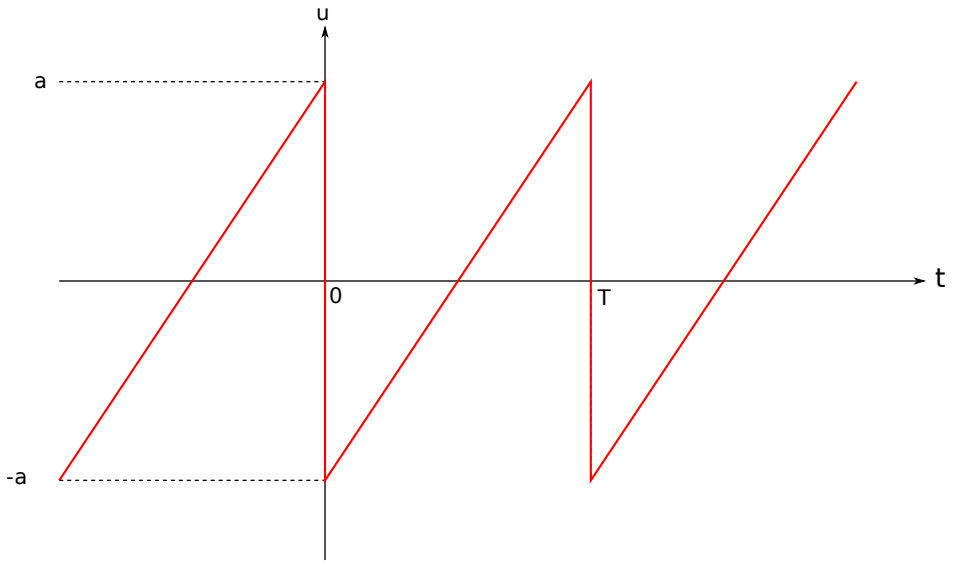
```
P=100
N=100*P
Cn = np.zeros(N,dtype=np.complex)
Cn[0] = r
for n in range(1,P):
    Cn[n]=C(n)
signal = ifft(Cn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
```

```
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) :



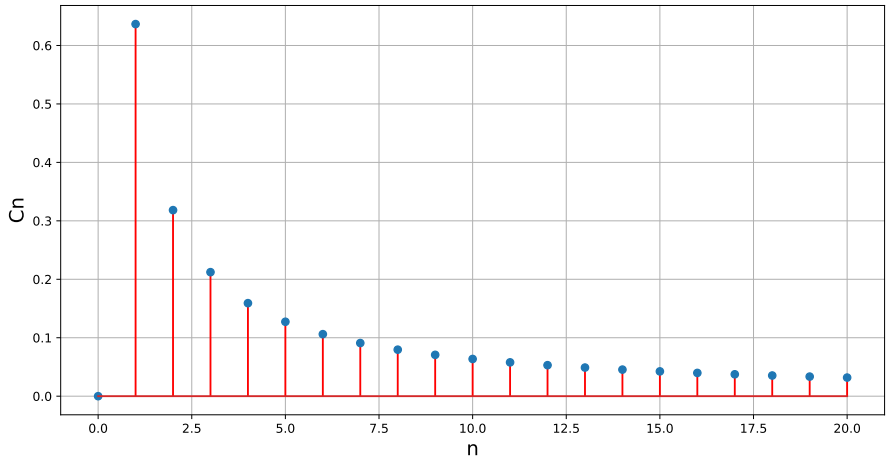
Ses coefficients de Fourier sont (pour $n > 0$) :

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

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

Voici son spectre :

```
def C(n):
    return 2/(np.pi*n)
spectre = [0]
P = 20
for n in range(1,P+1):
    spectre.append(C(n))
figure(figsize=(12,6))
stem(np.arange(P+1),spectre,'r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```



La présence d'une discontinuité implique une décroissance très 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. 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. Un système linéaire est complètement caractérisé par la réponse qu'il donne d'un signal sinusoïdal. 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 :

$$G(\omega) = |\underline{H}(\omega)| \tag{23}$$
$$\varphi(\omega) = arg(\underline{H}(\omega)) \tag{24}$$

Pour le cas particulier d'une pulsation nulle, on définit le gain par :

$$G(0) = \underline{H}(0) \tag{25}$$

Il s'agit d'un nombre réel, éventuellement négatif.

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

$$e(t) = C\cos(\omega t) \tag{26}$$
$$s(t) = CG(\omega)\cos\left(\omega t + \varphi(\omega)\right) \tag{27}$$

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

$$e(t) = \frac{C_0}{2} + \sum_{n=1}^{\infty} C_n \cos(n\omega_1 t + \Phi_n) \tag{28}$$

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 :

$$s(t) = \frac{C_0}{2}G(0) + \sum_{n=1}^{\infty} C_n G(n\omega_1)\cos(n\omega_1 t + \Phi_n + \varphi(n\omega_1)) \tag{29}$$

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. Une conséquence de ce résultat est que si un harmonique est absent du 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).

2.b. Exemple : filtrage d'un signal rectangulaire

Considérons comme exemple un filtre passe-bas du premier ordre agissant sur le signal rectangulaire à rapport cyclique variable (défini plus haut). Par convention, la fréquence du signal est égale à 1. Le filtre passe-bas est défini par la fonction de transfert harmonique suivante :

$$\underline{H}(\omega) = \frac{1}{1 + j\frac{\omega}{\omega_c}} \tag{30}$$

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

Considérons le cas où la fréquence de coupure est égale à 10 fois la fréquence fondamentale, c'est-à-dire 10.

```
fc=10
def H(f):
    return 1/(1+1j*f/fc)
```

Les coefficients de Fourier complexes (jusqu'au rang 100) sont placés dans un tableau de taille $100P$:

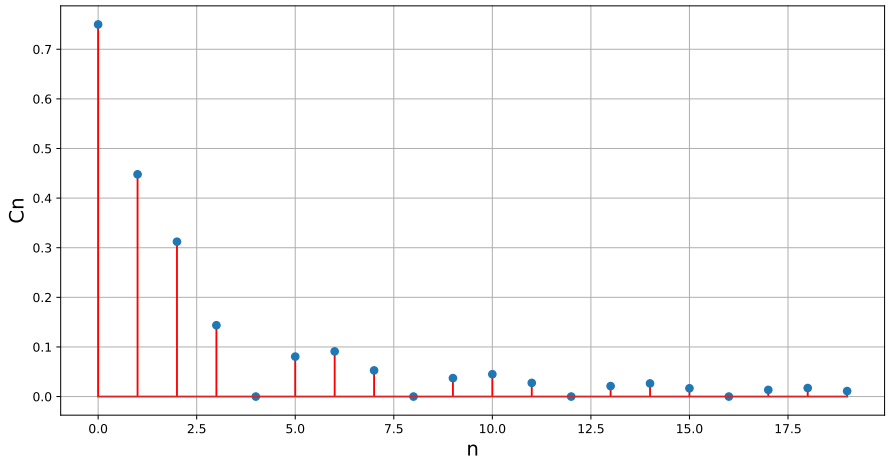
```
r=0.75
def C(n):
    return 2*r*np.exp(-1j*np.pi*n*r)*np.sin(np.pi*n*r)/(np.pi*n*r)
P=100
N=100*P
Cn = np.zeros(N,dtype=np.complex)
Cn[0] = r
for n in range(1,P):
    Cn[n]=C(n)
```

Les coefficients de Fourier complexes de la sortie sont obtenus en les multipliant par la valeur de la fonction de transfert aux fréquences correspondantes, sachant que la fréquence de l'harmonique de rang n est n :

```
Dn = np.array(Cn)
for n in range(0,P):
    Dn[n] *= H(n)
```

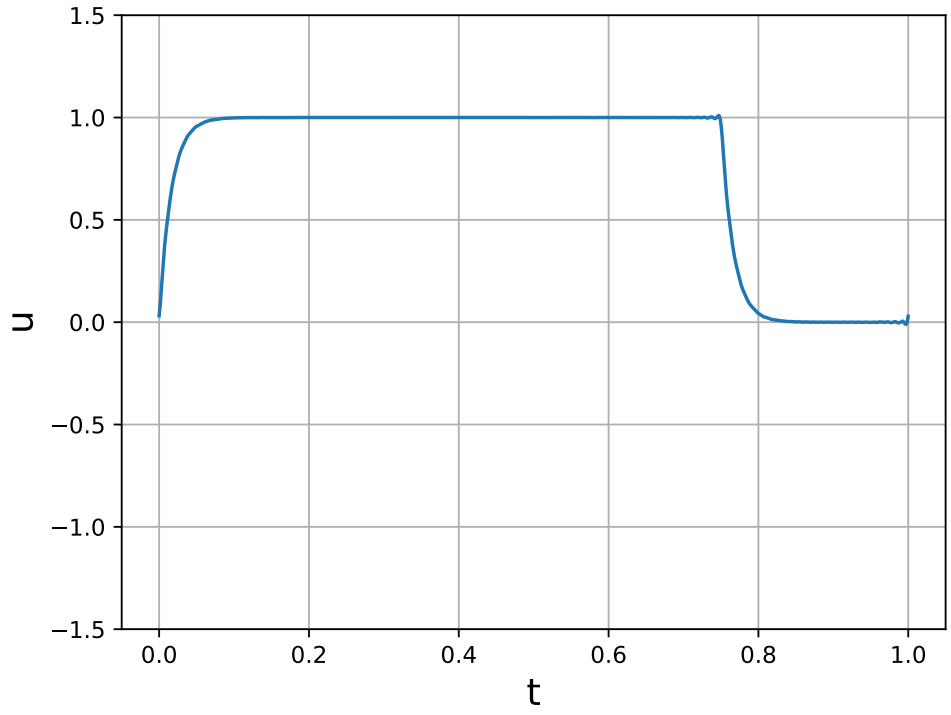
Voici le spectre du signal de sortie :

```
spectre = np.absolute(Dn)
figure(figsize=(12,6))
stem(np.arange(20),spectre[0:20],'r')
xlabel('n',fontsize=16)
ylabel('Cn',fontsize=16)
grid()
```



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

```
signal = ifft(Dn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
ylim(-1.5,1.5)
grid()
```

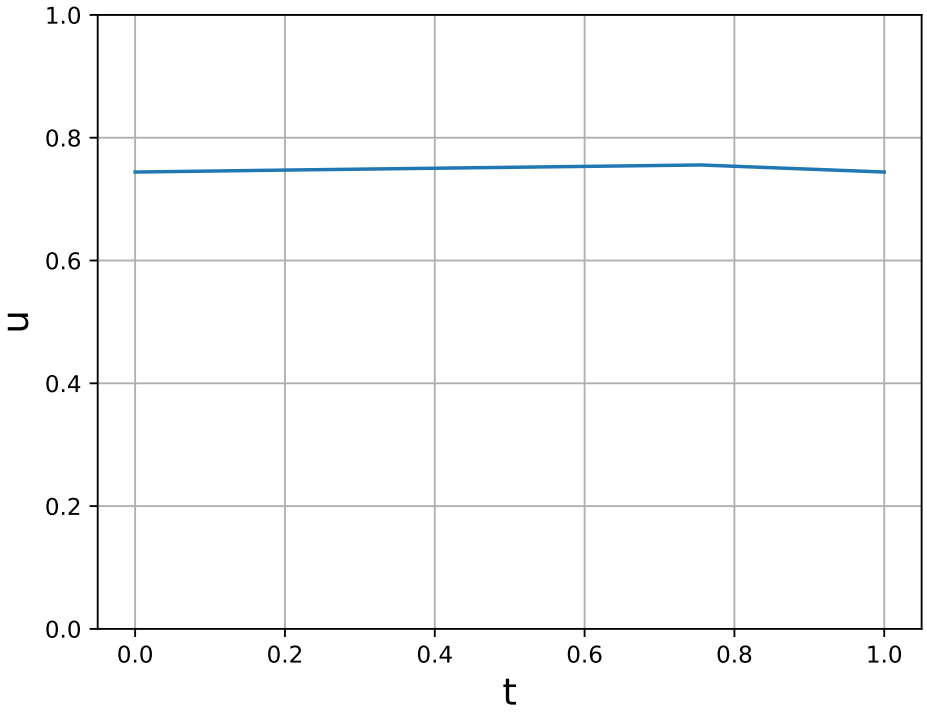


Le filtrage passe-bas a pour effet de remplacer les fronts de pente infini par des variations continues. Contrairement au signal d'entrée, la somme partielle de rang 100 suffit largement

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.

Pour finir, 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.

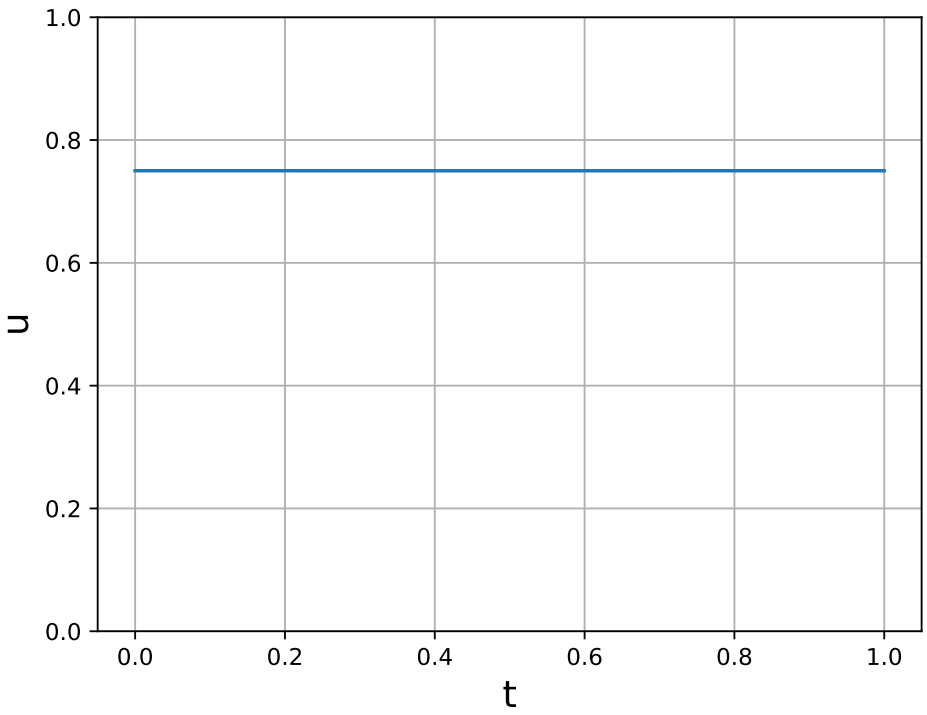
```
fc = 0.01
Dn = np.array(Cn)
for n in range(0,P):
    Dn[n] *= H(n)
signal = ifft(Dn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
ylim(0,1)
grid()
```



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 :

$$\underline{H}(\omega) = \frac{1}{1 + j\sqrt{2}\frac{\omega}{\omega_c} - \left(\frac{\omega}{\omega_c}\right)^2} \tag{31}$$

```
fc=0.01
def H(f):
    return 1/(1-(f/fc**2)+1j*np.sqrt(2)*f/fc)
Dn = np.array(Cn)
for n in range(0,P):
    Dn[n] *= H(n)
signal = ifft(Dn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
ylim(0,1)
grid()
```



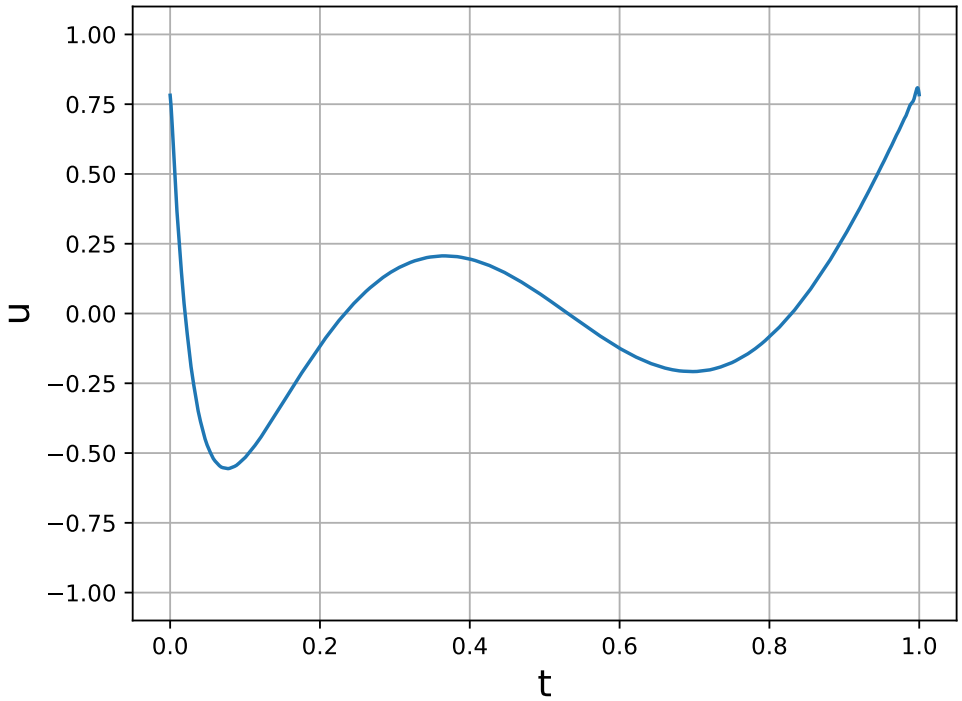
La sortie est bien constante, égale à la valeur moyenne du signal d'entrée, elle-même proportionnelle au rapport cyclique. Le filtrage passe-bas d'un signal rectangulaire à rapport cyclique variable est une technique utilisée en électronique de puissance pour obtenir une tension continue dont la valeur est contrôlée par le rapport cyclique. On peut citer comme application les alimentations régulées à découpage, où un asservissement permet de faire varier le rapport cyclique de manière à maintenir une tension constante quel que soit le courant débité par l'alimentation.

2.c. Exemple : annulation du fondamental

Nous allons appliquer un filtrage au signal en dents de scie, consistant à annuler son fondamental sans modifier les harmoniques et à lui appliquer un filtrage passe-bas du premier ordre. On trace la somme partielle de rang 100.

```
P=100
N=100*P
Cn = np.zeros(N,dtype=np.complex)
for n in range(1,P):
    Cn[n]=1j*2/(np.pi*n)
Dn = np.array(Cn)
Dn[1] = 0

fc=5
def H(f):
    return 1/(1+1j*f/fc)
for n in range(0,P):
    Dn[n] *= H(n)
signal = ifft(Dn)*N
t = np.arange(N)/N
figure()
plot(t,signal.real)
xlabel('t',fontsize=16)
ylabel('u',fontsize=16)
ylim(-1.1,1.1)
grid()
```



Ce signal a la particularité d'avoir un fondamental d'amplitude nulle. Le premier harmonique non nul est celui de rang 2. Lorsqu'un son riche en harmoniques possède cette propriété, la hauteur perçue par l'oreille est bien celle qui correspond à la fréquence fondamentale, même si celle-ci a une amplitude nulle. Le cerveau reconstitue le fondamental à partir de la perception des harmoniques. L'harmonique de rang 2 correspond à un intervalle de une octave par rapport au fondamental et le son une octave plus haut que le fondamental est aussi nettement perçu.

C'est principalement la perception de l'harmonique de rang 3 (une quinte plus haut que le rang 2) qui permet au cerveau de reconstituer la fréquence fondamentale.