TP08_filtrage_2023-2024 (corrigé)

December 14, 2023

1 Etude de la réponse d'un filtre à un signal donné en entrée

2 Etude du filtrage d'un créneau par le filtre de Wien

2.1 Bibliothèques nécessaires

[1]:
```
import numpy as np #pour pouvoir manipuler les tableaux
from cmath import * #pour avoir les complexes
import matplotlib.pyplot as plt #pour tracer les graphiques
```

2.2 Fonction générant les signaux souhaités

La fonction `genesignal` permet de générer un signal connaissant son spectre en amplitude et en phase, c'est-à-dire les fréquences, amplitudes et phases à l'origine des temps des harmoniques.

Elle prend quatre arguments en entrée :

`temps` : liste des instants de calcul du signal

et 3 listes de même longueur (notée n) pour caractériser chaque harmonique :

`frequence` : liste qui contient les fréquences des signaux que l'on somme

`amplitude` : liste qui contient les amplitudes des signaux que l'on somme (dans le même ordre que fréquence)

`phase` : liste qui contient les phases à l'origine des temps des signaux que l'on somme (dans le même ordre que fréquence)

Le signal généré s'écrit alors :

$$\sum_{i=0}^{n-1} amplitude[i] \times \cos(2\pi \times frequence[i] * t + phase[i])$$

Par exemple, si on veut créer le signal :

$$u(t) = 1 \times \cos(2\pi \times 1000 \times t) + 2 \times \cos(2\pi \times 3000 \times t + \pi/2) + 0.5 \times \cos(2\pi \times 5000 \times t - \pi/5)$$

Les trois listes précédentes renseignées seraient :

`frequence=[1000 , 3000 , 5000]`

```
amplitude=[1 , 2 , 0.5]

phase=[0 , pi/2 , -pi/5]
```

[2]:
```
def gene_signal(temps,frequence,amplitude,phase):
    """
    temps : instants de calculs
    fréquence, amplitude, phase : listes des fréquences, amplitudes et phases␣
 ↪de chaque amplitude du signal
    """
    Nt=len(temps) #nombre d'instants de calcul du signal
    s=np.zeros(Nt) #intialisation du signal : tableau de Nt 0
    n=len(frequence) #nombre d'harmoniques
    # Somme sur tous les harmoniques voulues :
    for i in range(n):
        s=s+amplitude[i]*np.cos(2*pi*frequence[i]*temps+phase[i])
    return s
```

2.2.1 Définition du signal d'entrée

On donne la décomposition en série de Fourier du signal créneau de fréquence f :

$$s(t) = \sum_{k=1}^{\infty} \frac{4A}{(2k-1)\pi} \cos\left(2\pi \times (2k-1)ft + \frac{\pi}{2}\right)$$

[3]:
```
f= 15000    # (à compléter) fréquence du signal créneau
A= 2    # (à compléter) amplitude du signal créneau
n= 20   # (à compléter) nombre d'harmoniques utilisés dans le spectre du␣
 ↪créneau,
       #à modifier pour voir l'effet sur la qualité du créneau généré
```

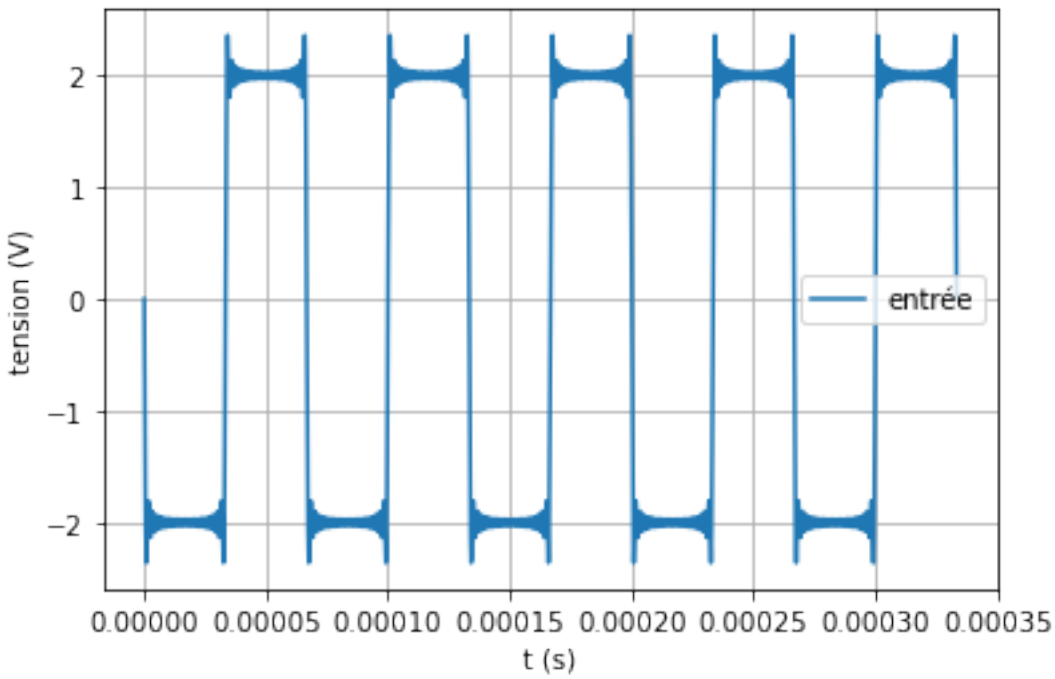
[4]:
```
#définition des harmoniques du signal d'entrée
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
 ↪fréquences des harmoniques
ampl_e=[ 4*A/((2*k-1) *pi)  for k in range(1,n+1)]     # (à compléter) défini␣
 ↪les amplitudes des harmoniques
phi_e=[np.pi/2 for k in range(1,n+1) ] # (à compléter)  défini les phases à␣
 ↪l'origine des temps de chaque harmonique
```

[5]:
```
# instants de calculs entre 0 et tf
tf=5*1/f # on calcule sur 5 périodes du signal
Nt=10000 # nombre d'instants de calculs
t=np.linspace(0,tf,Nt) # tableau de Nt valeurs réparties régulièrement entre 0␣
 ↪et tf
```

[6]:

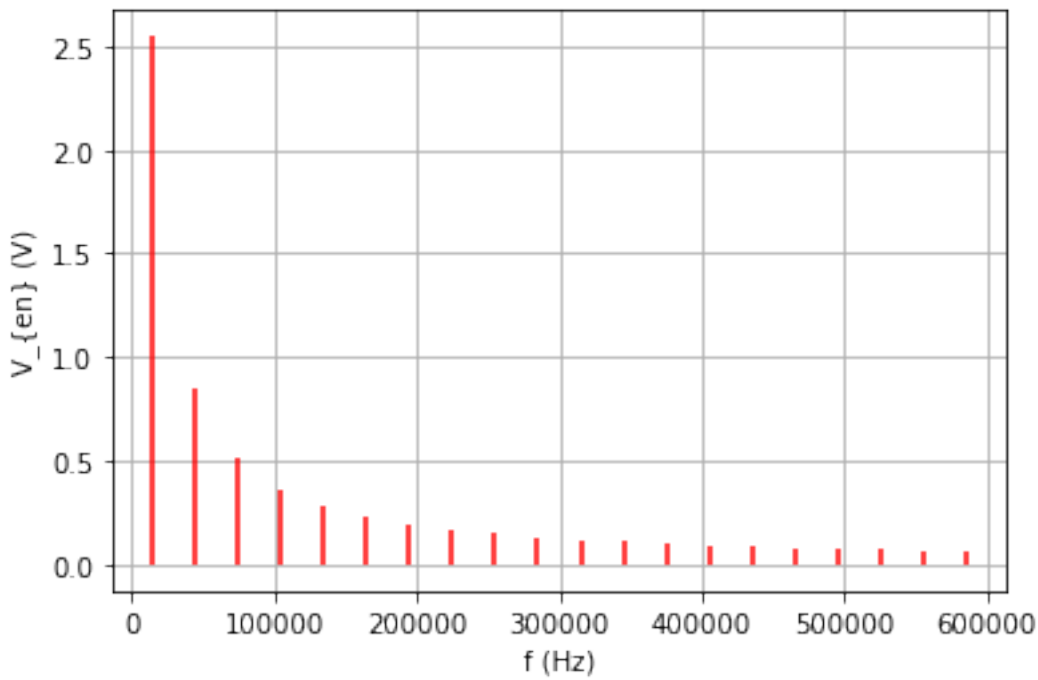
```
#signal d'entrée créé à partir de la fonction gene_signal et des paramètres␣
↪renseignés précédemment
entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
```

[7]:
```
# tracé du signal d'entrée
plt.plot(t,entree,label="entrée")
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension (V)')
plt.legend()
plt.show()
```



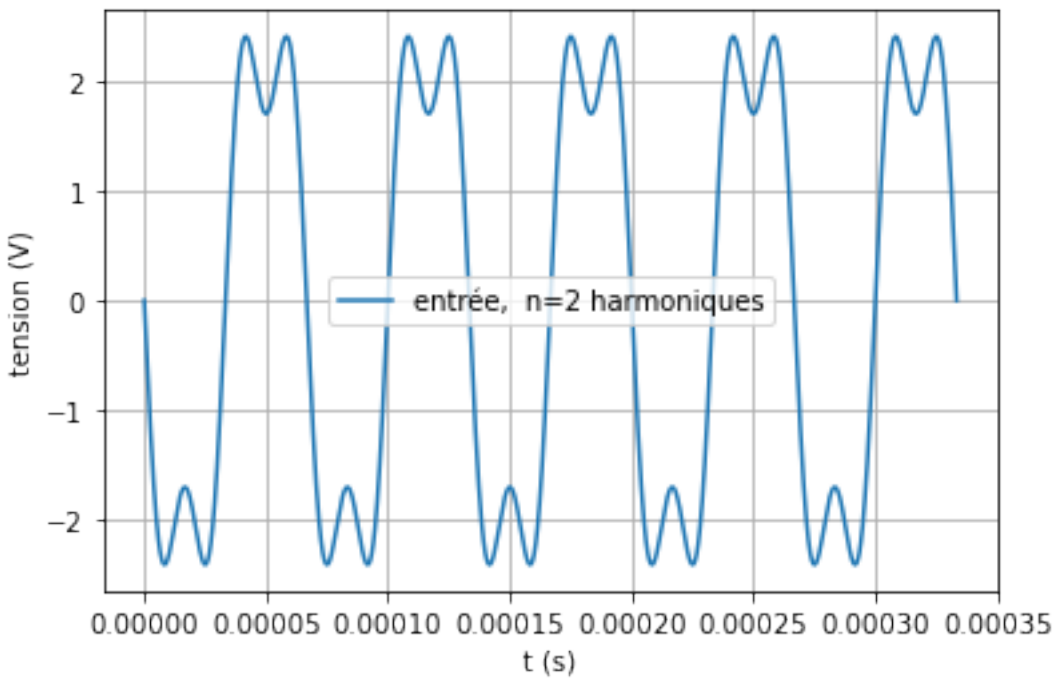
2.2.2 Spectre du signal d'entrée

[8]:
```
# tracé des harmoniques d'abscisses les fréquences de la liste freq et entre 0␣
↪et ampl/np.sqrt(2)
plt.vlines(freq,[0]*len(freq),ampl_e, linestyle = 'solid', color = 'red')
plt.xlabel('f (Hz)')
plt.ylabel('V_{en} (V)')
plt.grid()
plt.show()
```

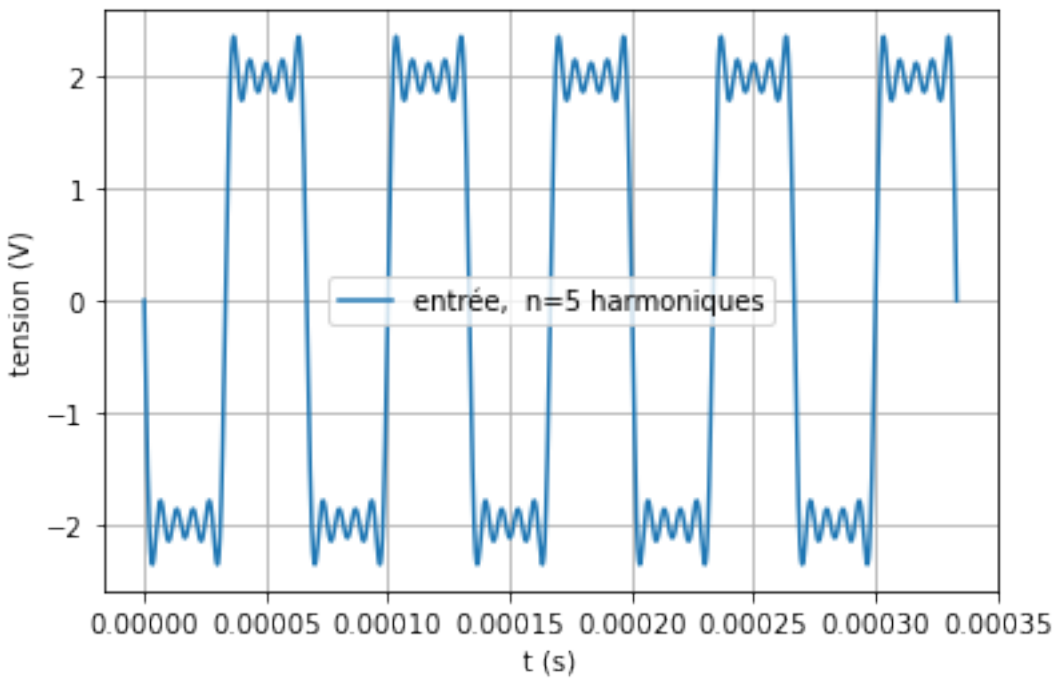


2.2.3 Influence du nombre d'harmoniques

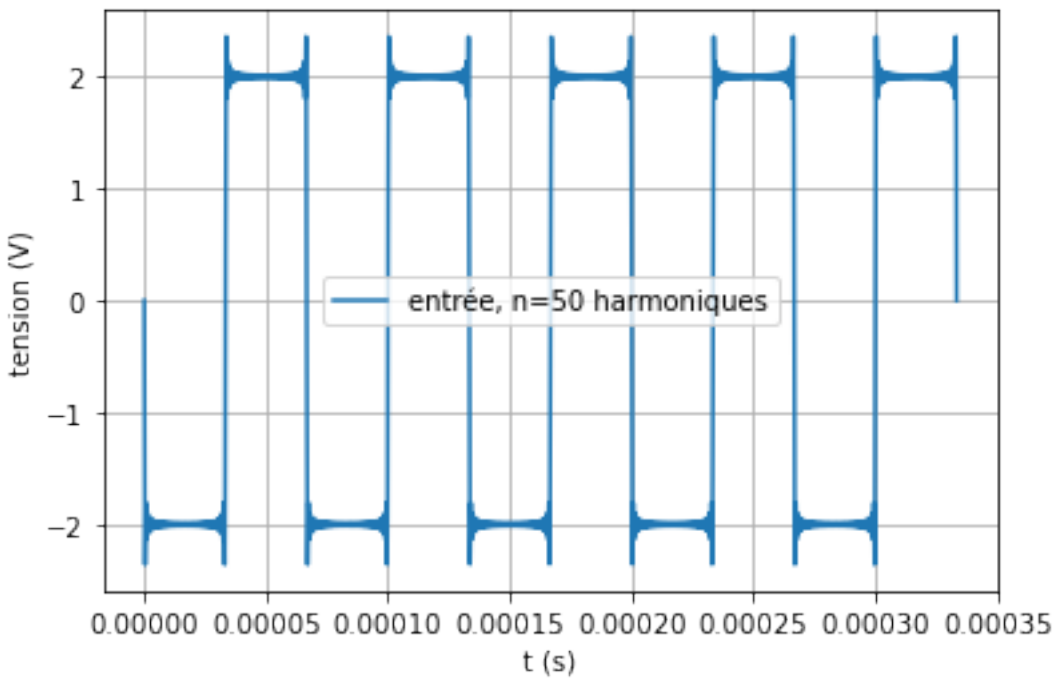
[9]:
```
n=2
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
↪fréquences des harmoniques
ampl_e=[ 4*A/((2*k-1) *pi)  for k in range(1,n+1)]     # (à compléter) défini␣
↪les amplitudes des harmoniques
phi_e=[np.pi/2 for k in range(1,n+1) ] # (à compléter)  défini les phases à␣
↪l'origine des temps de chaque harmonique
entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
# tracé du signal d'entrée
plt.plot(t,entree,label="entrée,  n=2 harmoniques")
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension (V)')
plt.legend()
plt.show()
```



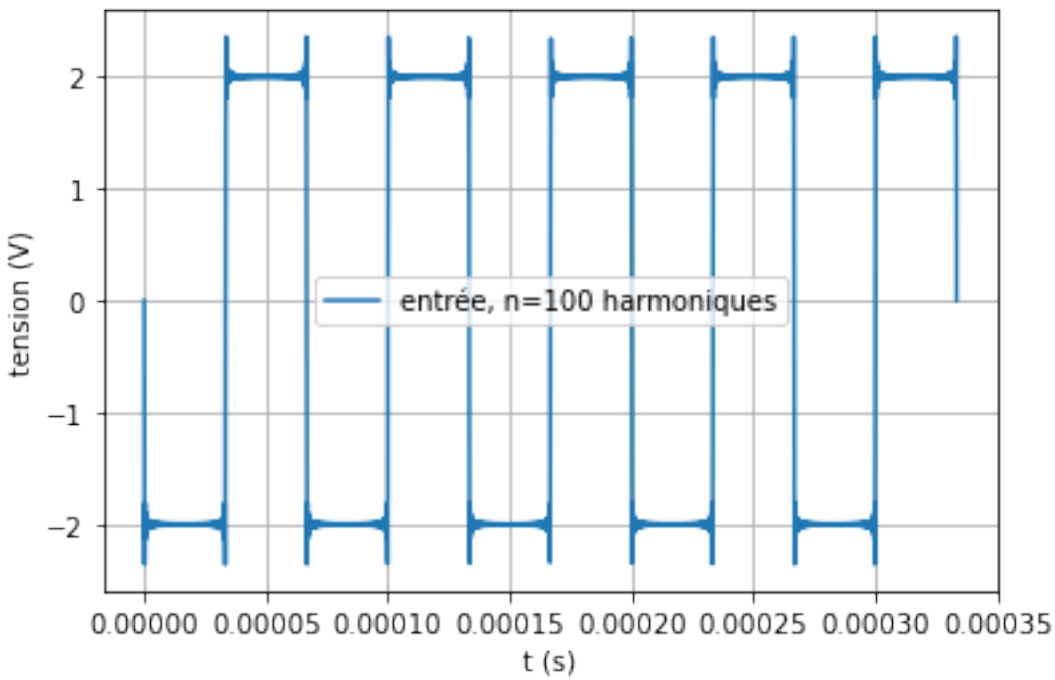
[10]:
```
n=5
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
↪fréquences des harmoniques
ampl_e=[ 4*A/((2*k-1) *pi)  for k in range(1,n+1)]     # (à compléter) défini␣
↪les amplitudes des harmoniques
phi_e=[np.pi/2 for k in range(1,n+1) ] # (à compléter)  défini les phases à␣
↪l'origine des temps de chaque harmonique
entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
# tracé du signal d'entrée
plt.plot(t,entree,label="entrée,  n=5 harmoniques")
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension (V)')
plt.legend()
plt.show()
```



[11]:
```
n=50
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
↪fréquences des harmoniques
ampl_e=[ 4*A/((2*k-1) *pi)  for k in range(1,n+1)]     # (à compléter) défini␣
↪les amplitudes des harmoniques
phi_e=[np.pi/2 for k in range(1,n+1) ] # (à compléter)  défini les phases à␣
↪l'origine des temps de chaque harmonique
entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
# tracé du signal d'entrée
plt.plot(t,entree,label="entrée, n=50 harmoniques")
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension (V)')
plt.legend()
plt.show()
```



[12]:
```
n=100
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
↪fréquences des harmoniques
ampl_e=[ 4*A/((2*k-1) *pi)  for k in range(1,n+1)]     # (à compléter) défini␣
↪les amplitudes des harmoniques
phi_e=[np.pi/2 for k in range(1,n+1) ] # (à compléter)  défini les phases à␣
↪l'origine des temps de chaque harmonique
entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
# tracé du signal d'entrée
plt.plot(t,entree,label="entrée, n=100 harmoniques")
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension (V)')
plt.legend()
plt.show()
```



L'augmentation du nombre d'harmoniques pris en compte permet de mieux générer le signal périodique.

2.3 Définitions des fonctions de transfert

Passe-bas du 1er ordre :

$$\underline{H} = \frac{H_0}{1 + j\dfrac{f}{f_c}}$$

Passe-haut du 1er ordre :

$$\underline{H} = \frac{H_0 \dfrac{f}{f_c}}{1 + j\dfrac{f}{f_c}}$$

Passe-bas du 2è ordre :

$$\underline{H} = \frac{H_0}{1 - \dfrac{f^2}{f_0^2} + j\dfrac{f}{f_0 Q}}$$

Passe-bande du 2è ordre :

$$\underline{H} = \frac{H_0}{1 + jQ\left(\dfrac{f}{f_0} - \dfrac{f_0}{f}\right)}$$

[13]:
```python
# 1j est le nombre complexe tel que son carré vaut -1
def passe_bas1(H0,fc,f):
    """
    H0 : gain à basse fréquence
    fc : fréquence de coupure
    f : fréquence du signal sinusoïdal en entrée du filtre
    """
    return H0/(1+1j*f/fc)

def passe_haut1(H0,fc,f):
    """
    H0 : gain à haute fréquence
    fc : fréquence de coupure
    f : fréquence du signal sinusoïdal en entrée du filtre
    """
    return H0*1j*f/fc/(1+1j*f/fc)

def passe_bas2(H0,f0,Q,f):
    """
    H0 : gain à basse fréquence
    f0 : fréquence propre
    Q : facteur de qualité
    f : fréquence du signal sinusoïdal en entrée du filtre
    """
    return H0/(1-(f/fc)**2+1j*f/(f0*Q))

def passe_bande2(H0,f0,Q,f):
    """
    H0 : gain à la résonance
    f0 : fréquence propre (et de résonance)
    Q : facteur de qualité
    f : fréquence du signal sinusoïdal en entrée du filtre
    """
    return H0/(1+1j*Q*(f/f0-f0/f))
```

2.4 Etude du filtrage d'un créneau par le filtre de Wien à haute fréquence

On étudie la réponse du filtre de Wien de fonction de transfert $\underline{H} = \dfrac{1/3}{1 + \dfrac{j}{3}\left(\dfrac{f}{f_0} - \dfrac{f_0}{f}\right)}$, de fréquence de coupure $f_0 \approx 720$ Hz, à un signal créneau de fréquence $f = 20$ kHz.

2.4.1 Générer le signal de sortie

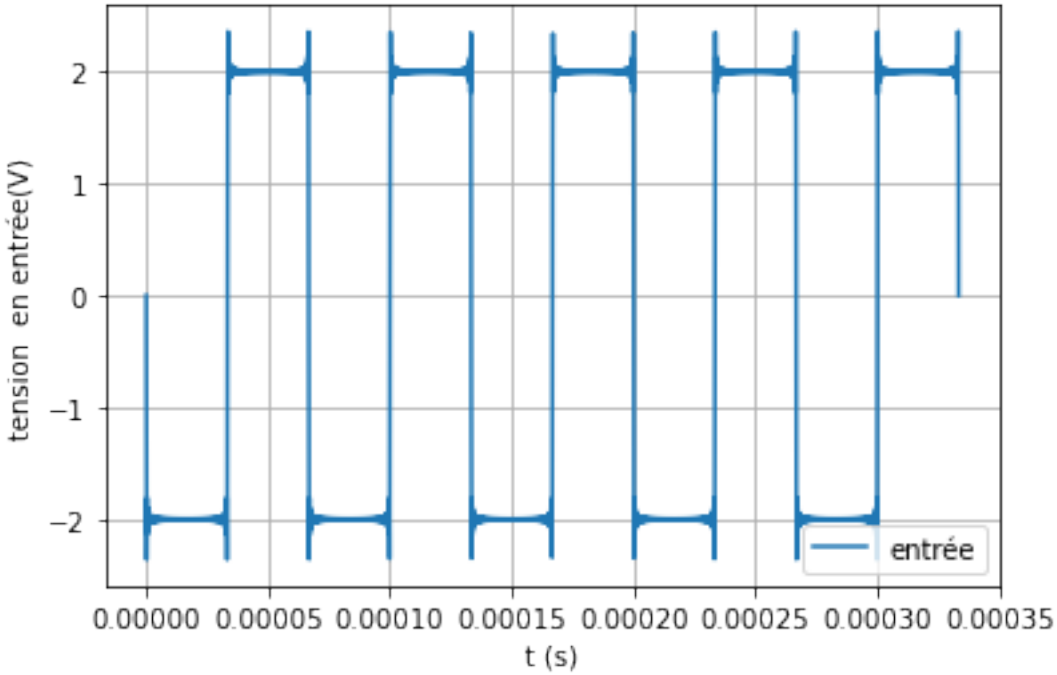
[14]:
```
# initialisation du signal de sortie avec Nt 0 (Nt : nombre d'instants de␣
↪calculs)
sortie =  np.zeros(Nt)
```

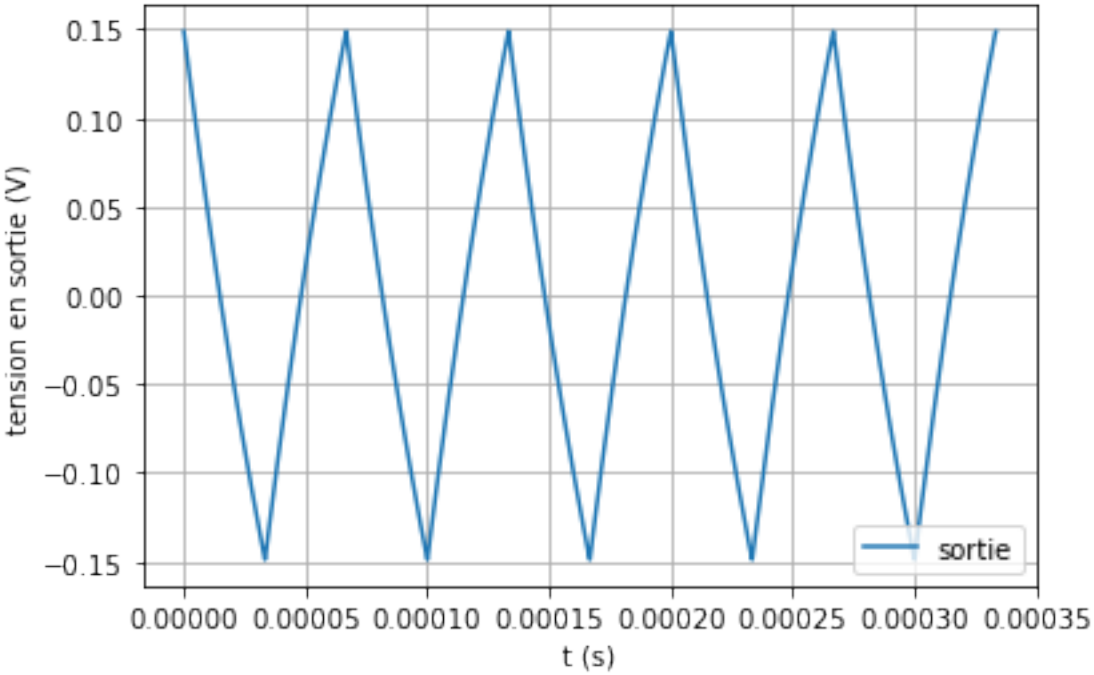
[15]:
```
# paramètres du filtre de Wien étudié expérimentalement
H0= 1/3   # (à compléter) gain statique
f0=  723     # (à compléter) fréquence propre mesurée expérimentalement
Q=1/3
```

[16]:
```
#calculs des amplitudes et phases des harmoniques du signal de sortie
ampl_s=[] #liste des amplitudes du signal de sortie
phi_s=[] #liste des phases à l'origne des temps du signal de sortie
for i in range(len(freq)): #parcours des harmoniques
    #pour chaque harmonique i de fréquence freq[i]:
    #calculer la fonction de transfert à la fréquence freq[i]:
    H= passe_bande2(H0,f0,1/3,freq[i])  # (à compléter)
    #en déduire le gain :
    G=  abs(H)         # (à compléter)
    #la phase:
    phi=phase(H)                # (à compléter)
    # calcul de l'amplitude de l'harmonique i de sortie
    ampl_si = ampl_e[i]*G      # (à compléter)
    #ajout de l'amplitude ampl_si à la liste des amplitudes :
    ampl_s.append(ampl_si)
    # calcul de la phase à l'origine des temps de l'harmonique i en sortie
    phi_si = phi_e[i]+phi    # (à compléter)
    #ajout de la phase phi_si à la liste des phases :
    phi_s.append(phi_si)               # (à compléter)

# génération du signal de sortie
sortie=gene_signal(t,freq,ampl_s,phi_s)
```

[17]:
```
#tracer des signaux d'entrée et de sortie
plt.figure(1)
plt.plot(t,entree,label="entrée")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension  en entrée(V)')
plt.figure(2)
plt.plot(t,sortie,label="sortie")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension en sortie (V)')
```

```
plt.show()
```

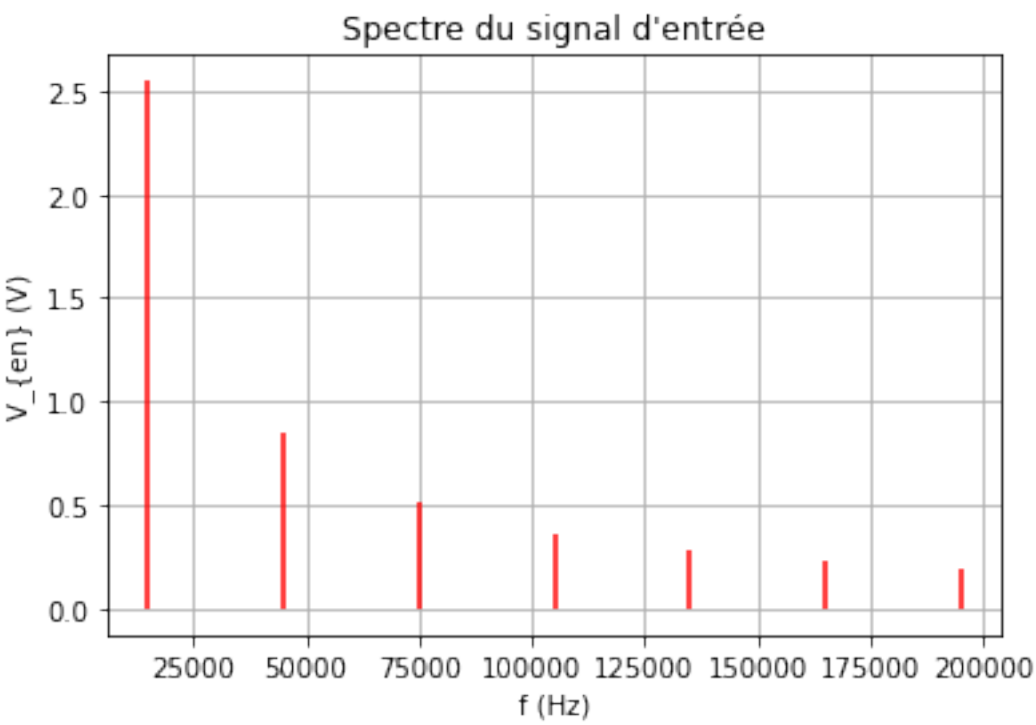


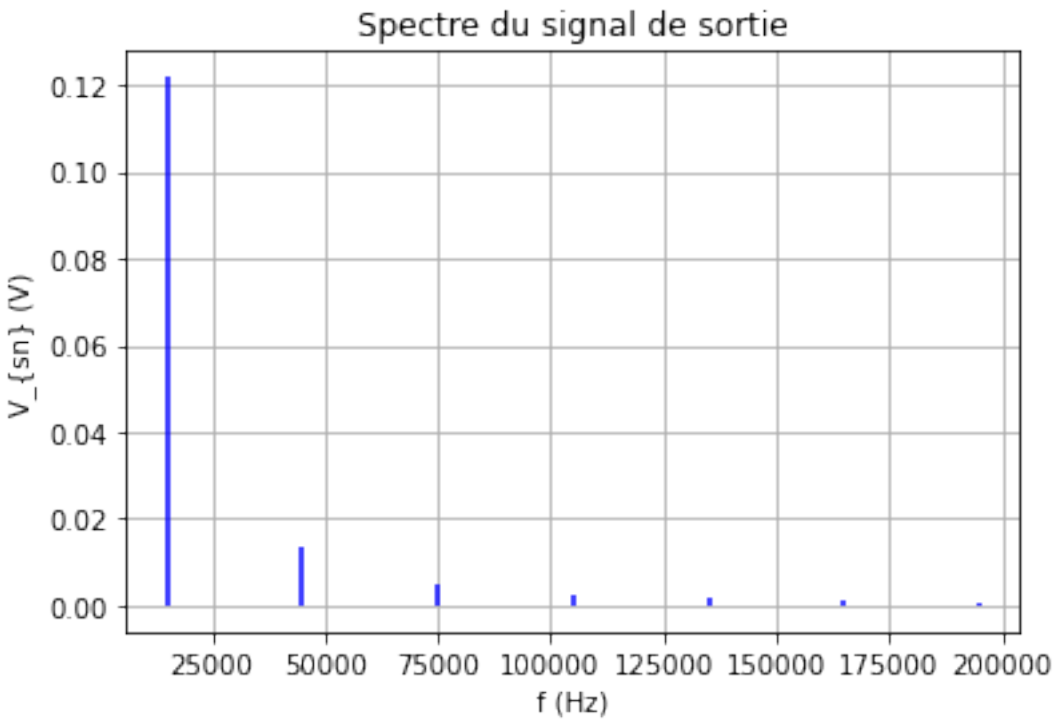


Le filtre intègre le signal d'entrée !

2.4.2 Spectre du signal de sortie

[18]:
```
# spectre du signal d'entrée
plt.figure(1)
plt.vlines(freq[:7],[0]*len(freq[:7]),ampl_e[:7], linestyle = 'solid', color =␣
 ↪'red')
plt.xlabel('f (Hz)')
plt.ylabel('V_{en} (V)')
plt.title("Spectre du signal d'entrée")
plt.grid()

#spectre du signal de sortie :
plt.figure(2)
plt.vlines(freq[:7],[0]*len(freq[:7]),ampl_s[:7], linestyle = 'solid', color =␣
 ↪'blue')
plt.xlabel('f (Hz)')
plt.ylabel('V_{sn} (V)')
plt.title('Spectre du signal de sortie')
plt.grid()
plt.show()
```





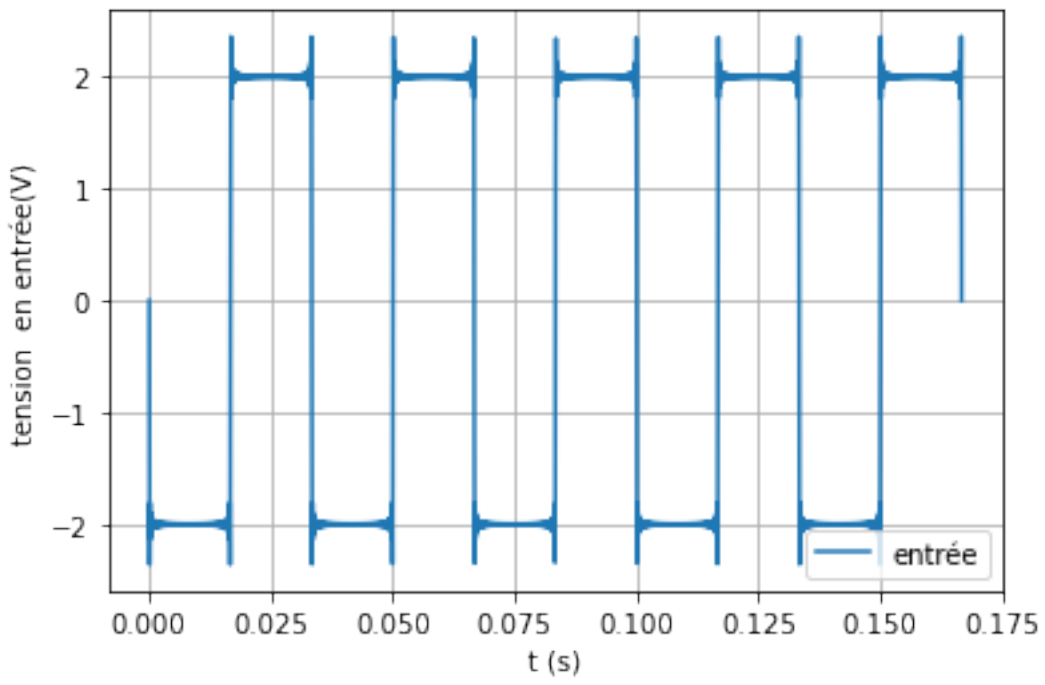
2.5 Etude du filtrage d'un créneau par le filtre de Wien à haute fréquence

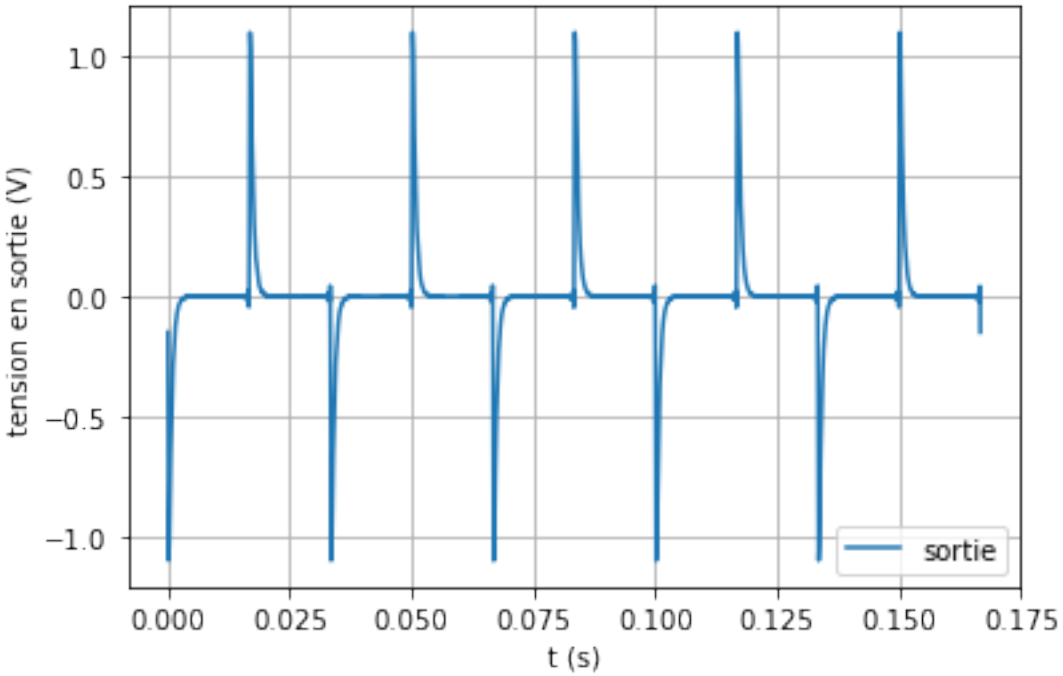
[19]:
```
n=100
f=30 # Hz : <<f0=723 Hz
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
 ↪fréquences des harmoniques
ampl_e=[ 4*A/((2*k-1) *pi)  for k in range(1,n+1)]     # (à compléter) défini␣
 ↪les amplitudes des harmoniques
phi_e=[np.pi/2 for k in range(1,n+1) ] # (à compléter)  défini les phases à␣
 ↪l'origine des temps de chaque harmonique

# instants de calculs entre 0 et tf
tf=5*1/f # on calcule sur 5 périodes du signal
Nt=10000 # nombre d'instants de calculs
t=np.linspace(0,tf,Nt) # tableau de Nt valeurs réparties régulièrement entre 0␣
 ↪et tf

entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
```

[20]:
```
#calculs des amplitudes et phases des harmoniques du signal de sortie
ampl_s=[] #liste des amplitudes du signal de sortie
phi_s=[] #liste des phases à l'origne des temps du signal de sortie
for i in range(len(freq)): #parcours des harmoniques
```

```
    #pour chaque harmonique i de fréquence freq[i]:
    #calculer la fonction de transfert à la fréquence freq[i]:
    H= passe_bande2(H0,f0,1/3,freq[i])  # (à compléter)
    #en déduire le gain :
    G=  abs(H)         # (à compléter)
    #la phase:
    phi=phase(H)                # (à compléter)
    # calcul de l'amplitude de l'harmonique i de sortie
    ampl_si = ampl_e[i]*G      # (à compléter)
    #ajout de l'amplitude ampl_si à la liste des amplitudes :
    ampl_s.append(ampl_si)
    # calcul de la phase à l'origine des temps de l'harmonique i en sortie
    phi_si = phi_e[i]+phi    # (à compléter)
    #ajout de la phase phi_si à la liste des phases :
    phi_s.append(phi_si)               # (à compléter)

# génération du signal de sortie
sortie=gene_signal(t,freq,ampl_s,phi_s)
```

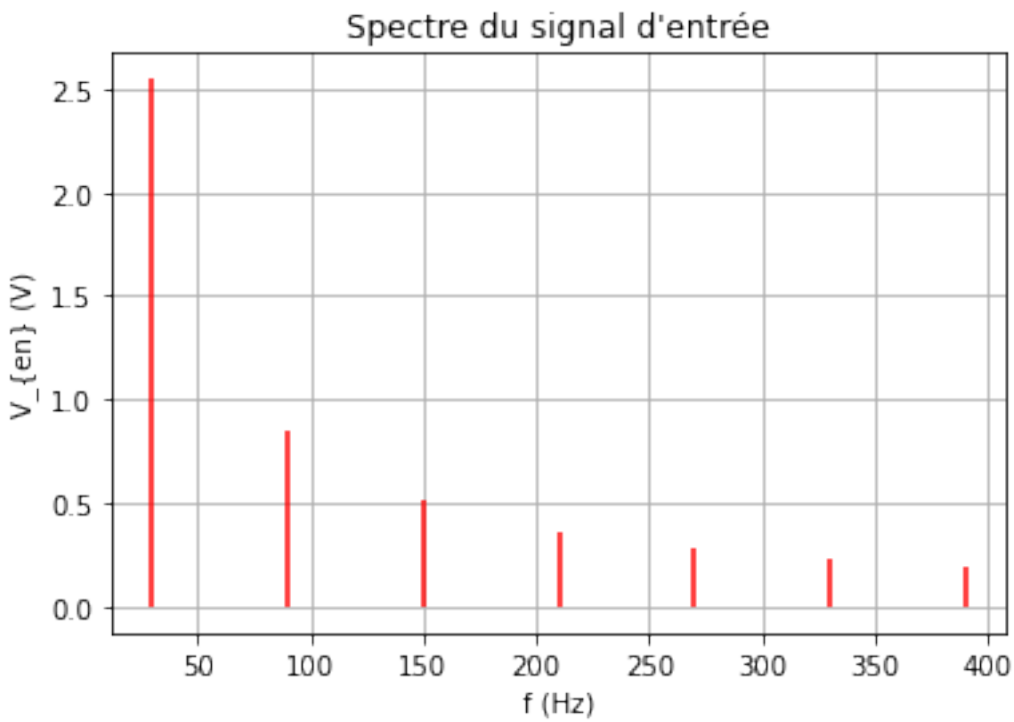
[21]:
```
#tracer des signaux d'entrée et de sortie
plt.figure(1)
plt.plot(t,entree,label="entrée")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension  en entrée(V)')
plt.figure(2)
plt.plot(t,sortie,label="sortie")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension en sortie (V)')
plt.show()
```

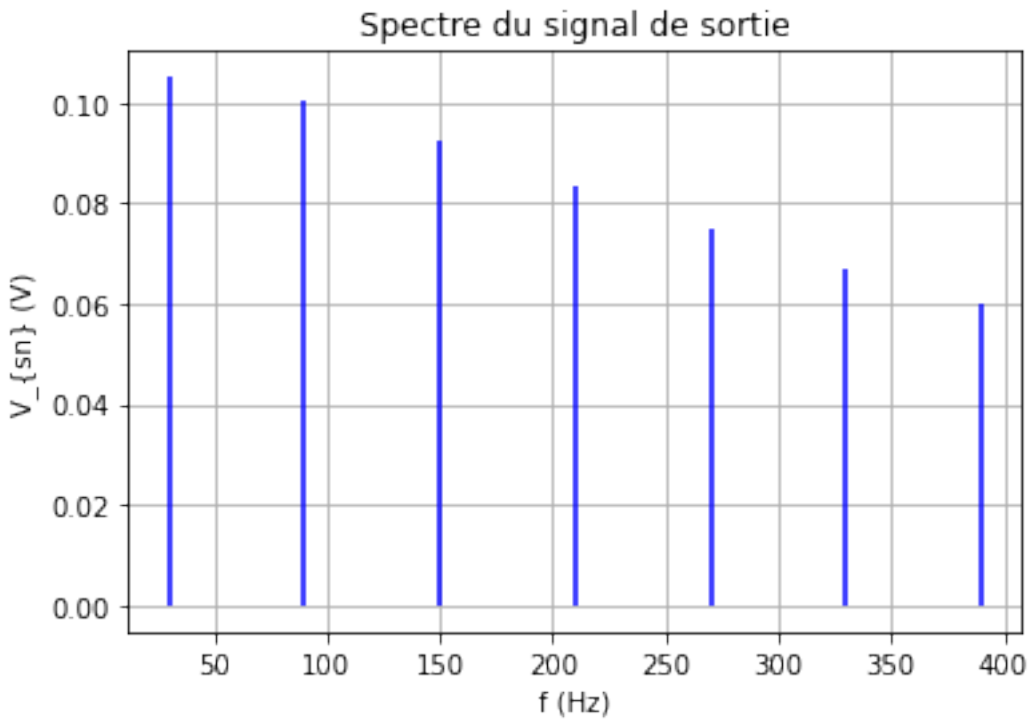




Le filtre dérive ici le signal d'entrée !

[22]:
```
# spectre du signal d'entrée
plt.figure(1)
plt.vlines(freq[:7],[0]*len(freq[:7]),ampl_e[:7], linestyle = 'solid', color =␣
↪'red')
plt.xlabel('f (Hz)')
plt.ylabel('V_{en} (V)')
plt.title("Spectre du signal d'entrée")
plt.grid()

#spectre du signal de sortie :
plt.figure(2)
plt.vlines(freq[:7],[0]*len(freq[:7]),ampl_s[:7], linestyle = 'solid', color =␣
↪'blue')
plt.xlabel('f (Hz)')
plt.ylabel('V_{sn} (V)')
plt.title('Spectre du signal de sortie')
plt.grid()
plt.show()
```





3 Etude du filtrage d'un signal triangulaire avec le filtre de Wien à haute fréquence

Reprendre toute l'étude précédente aussi bien expérimentale que de modélisation avec un signal triangulaire en entrée. On donne la décomposition en série de Fourier du signal triangle de fréquence f :

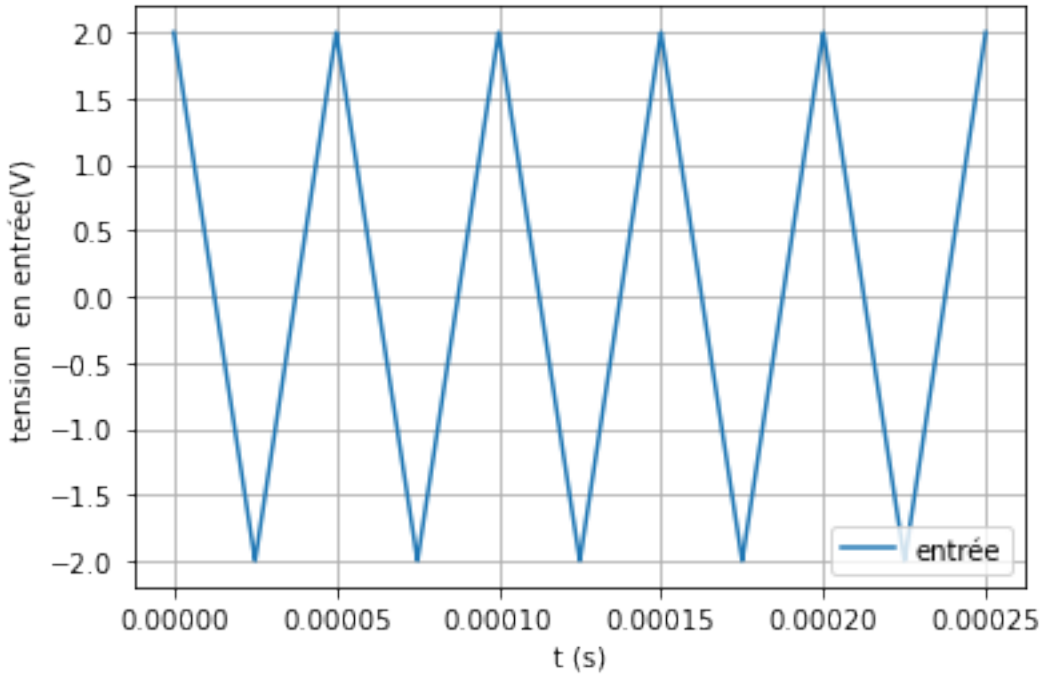
$$s(t) = \sum_{k=1}^{\infty} \frac{8A}{((2k-1)\pi)^2} \cos\Big(2\pi \times (2k-1)ft\Big)$$

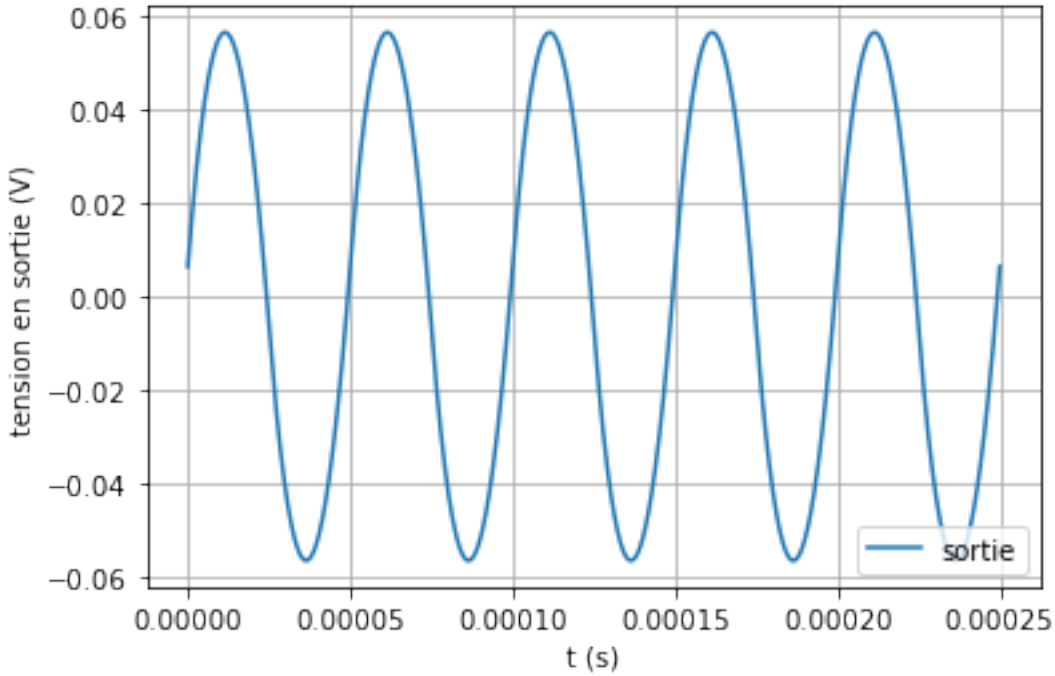
[23]:
```
n=100
f=20000 # Hz : <<f0=723 Hz
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
 ↪fréquences des harmoniques
ampl_e=[ 8*A/((2*k-1) *pi)**2  for k in range(1,n+1)]     # (à compléter)␣
 ↪défini les amplitudes des harmoniques
phi_e=[0 for k in range(1,n+1) ] # (à compléter)  défini les phases à l'origine␣
 ↪des temps de chaque harmonique

# instants de calculs entre 0 et tf
tf=5*1/f # on calcule sur 5 périodes du signal
Nt=10000 # nombre d'instants de calculs
```

```
t=np.linspace(0,tf,Nt) # tableau de Nt valeurs réparties régulièrement entre 0␣
↪et tf

entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
```

[24]:
```
#calculs des amplitudes et phases des harmoniques du signal de sortie
ampl_s=[] #liste des amplitudes du signal de sortie
phi_s=[] #liste des phases à l'origne des temps du signal de sortie
for i in range(len(freq)): #parcours des harmoniques
    #pour chaque harmonique i de fréquence freq[i]:
    #calculer la fonction de transfert à la fréquence freq[i]:
    H= passe_bande2(H0,f0,1/3,freq[i])  # (à compléter)
    #en déduire le gain :
    G=  abs(H)          # (à compléter)
    #la phase:
    phi=phase(H)                # (à compléter)
    # calcul de l'amplitude de l'harmonique i de sortie
    ampl_si = ampl_e[i]*G      # (à compléter)
    #ajout de l'amplitude ampl_si à la liste des amplitudes :
    ampl_s.append(ampl_si)
    # calcul de la phase à l'origine des temps de l'harmonique i en sortie
    phi_si = phi_e[i]+phi    # (à compléter)
    #ajout de la phase phi_si à la liste des phases :
    phi_s.append(phi_si)               # (à compléter)

# génération du signal de sortie
sortie=gene_signal(t,freq,ampl_s,phi_s)
```

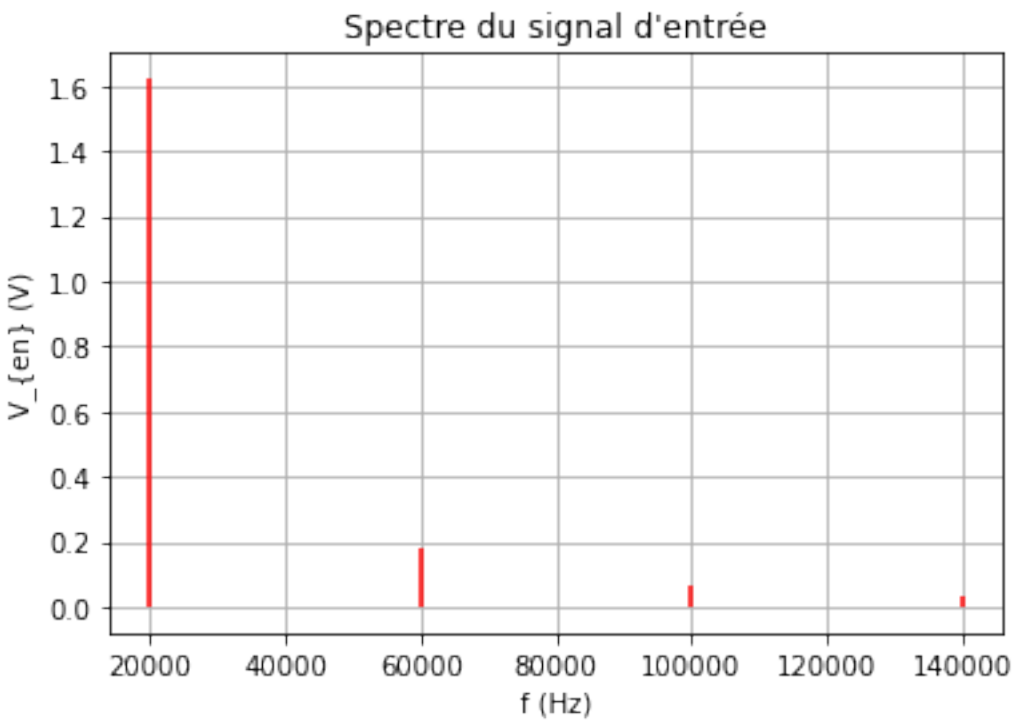
[25]:
```
#tracer des signaux d'entrée et de sortie
plt.figure(1)
plt.plot(t,entree,label="entrée")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension  en entrée(V)')
plt.figure(2)
plt.plot(t,sortie,label="sortie")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension en sortie (V)')
plt.show()
```

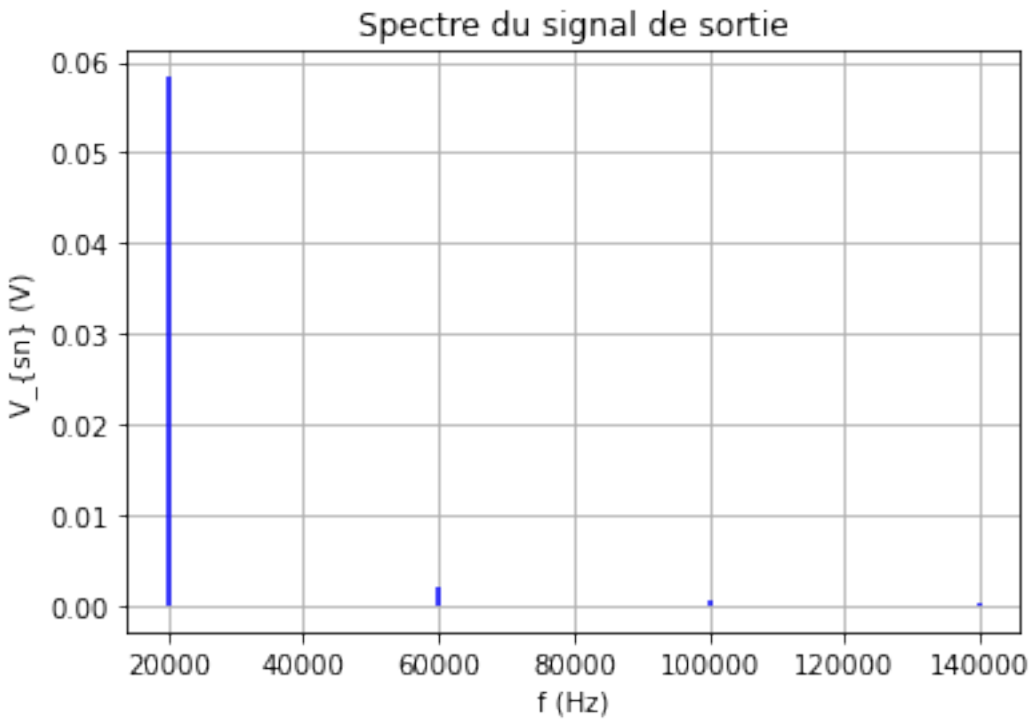




Le signal triangulaire est intégré, ATTENTION ce n'est pas un signal sinusoïdal en sortie, c'est un signal constitué de portion de paraboles : parabole tournée vers le haut quand le triangle est décroissant, parabole tournée vers le bas quand le triangle est croissant.

[26]:
```
# spectre du signal d'entrée
plt.figure(1)
plt.vlines(freq[:4],[0]*len(freq[:4]),ampl_e[:4], linestyle = 'solid', color =␣
↪'red')
plt.xlabel('f (Hz)')
plt.ylabel('V_{en} (V)')
plt.title("Spectre du signal d'entrée")
plt.grid()

#spectre du signal de sortie :
plt.figure(2)
plt.vlines(freq[:4],[0]*len(freq[:4]),ampl_s[:4], linestyle = 'solid', color =␣
↪'blue')
plt.xlabel('f (Hz)')
plt.ylabel('V_{sn} (V)')
plt.title('Spectre du signal de sortie')
plt.grid()
plt.show()
```





Le spectre du signal de sortie se rapproche de plus en plus au spectre d'un sinus, mais c'en est pas un, c'est une succession de portions de paraboles.

4 Etude du filtrage d'un signal triangulaire avec le filtre de Wien à basse fréquence

Reprendre toute l'étude précédente aussi bien expérimentale que de modélisation avec un signal triangulaire en entrée. On donne la décomposition en série de Fourier du signal triangle de fréquence f :

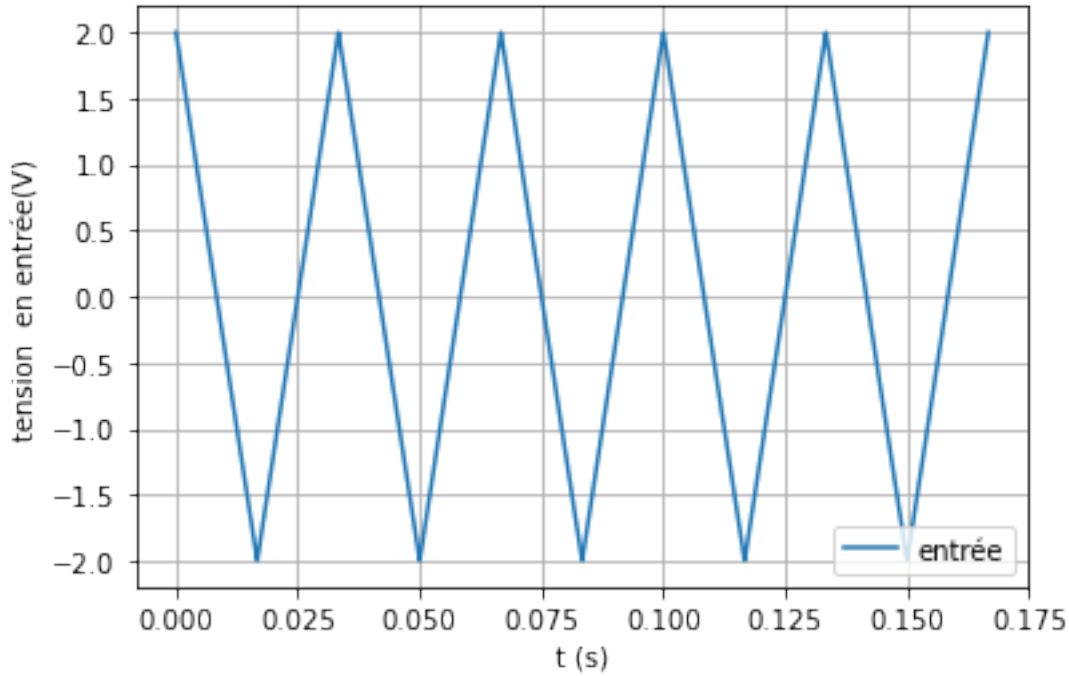
$$s(t) = \sum_{k=1}^{\infty} \frac{8A}{((2k-1)\pi)^2} \cos\Big(2\pi \times (2k-1)ft\Big)$$

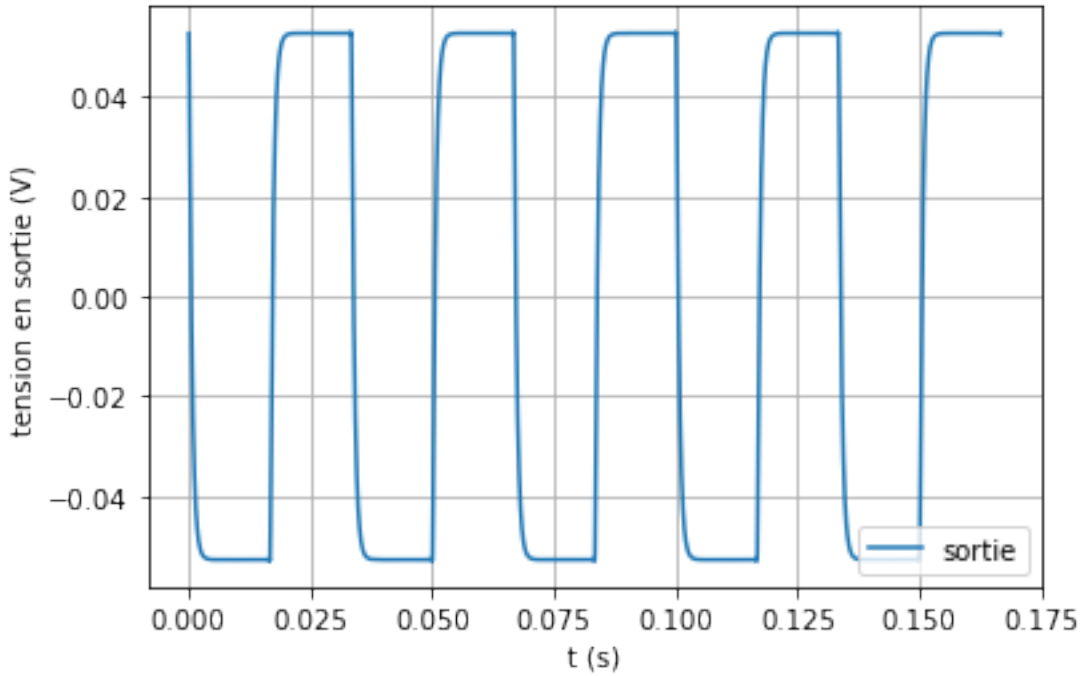
[27]:
```
n=100
f=30 # Hz : <<f0=723 Hz
freq=[  (2*k-1)*f    for k in range(1,n+1)] # (à compléter)  défini les␣
 ↪fréquences des harmoniques
ampl_e=[ 8*A/((2*k-1) *pi)**2  for k in range(1,n+1)]     # (à compléter)␣
 ↪défini les amplitudes des harmoniques
phi_e=[0 for k in range(1,n+1) ] # (à compléter)  défini les phases à l'origine␣
 ↪des temps de chaque harmonique

# instants de calculs entre 0 et tf
```

```
tf=5*1/f # on calcule sur 5 périodes du signal
Nt=10000 # nombre d'instants de calculs
t=np.linspace(0,tf,Nt) # tableau de Nt valeurs réparties régulièrement entre 0␣
 ↪et tf

entree=  gene_signal(t,freq,ampl_e,phi_e)              # (à compléter)
```

[28]:
```
#calculs des amplitudes et phases des harmoniques du signal de sortie
ampl_s=[] #liste des amplitudes du signal de sortie
phi_s=[] #liste des phases à l'origne des temps du signal de sortie
for i in range(len(freq)): #parcours des harmoniques
    #pour chaque harmonique i de fréquence freq[i]:
    #calculer la fonction de transfert à la fréquence freq[i]:
    H= passe_bande2(H0,f0,1/3,freq[i])  # (à compléter)
    #en déduire le gain :
    G=  abs(H)         # (à compléter)
    #la phase:
    phi=phase(H)                # (à compléter)
    # calcul de l'amplitude de l'harmonique i de sortie
    ampl_si = ampl_e[i]*G      # (à compléter)
    #ajout de l'amplitude ampl_si à la liste des amplitudes :
    ampl_s.append(ampl_si)
    # calcul de la phase à l'origine des temps de l'harmonique i en sortie
    phi_si = phi_e[i]+phi    # (à compléter)
    #ajout de la phase phi_si à la liste des phases :
    phi_s.append(phi_si)               # (à compléter)

# génération du signal de sortie
sortie=gene_signal(t,freq,ampl_s,phi_s)
```

[29]:
```
#tracer des signaux d'entrée et de sortie
plt.figure(1)
plt.plot(t,entree,label="entrée")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension  en entrée(V)')
plt.figure(2)
plt.plot(t,sortie,label="sortie")
plt.legend(loc='lower right')
plt.grid()
plt.xlabel('t (s)')
plt.ylabel('tension en sortie (V)')
plt.show()
```

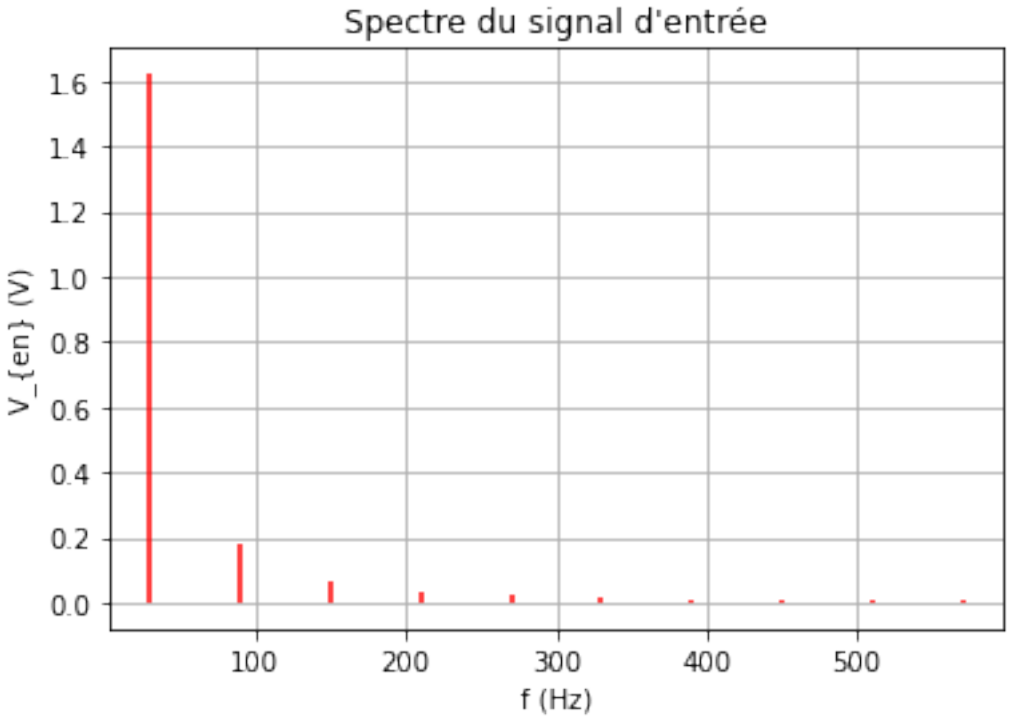


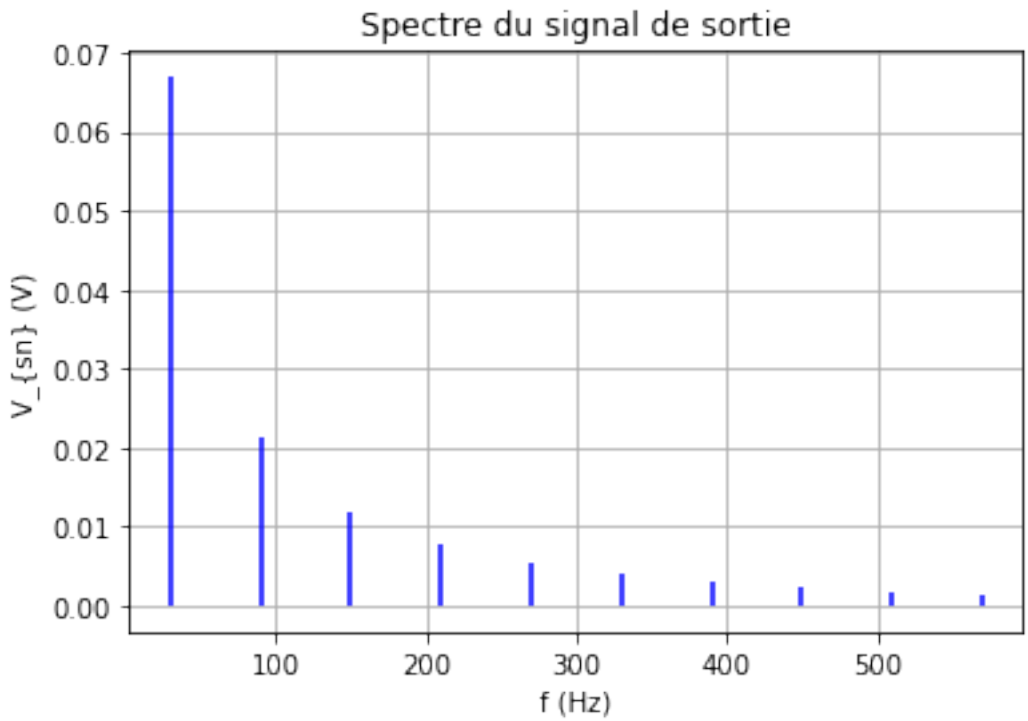


Le signal triangulaire est dérivé, on obtient un signal créneau (ou presque...). Le filtre est dérivateur à BASSE FREQUENCE, mais des harmoniques du triangle ne sont pas à basse fréquence par rapport à la fréquence de résonance, et ne sont donc pas dérivés. Ce sont les harmoniques élevés,

responsables des changements brusques de tension qui ne sont pas bien dérivées, d'où le créneau déformé lors des bascules.

[30]:
```
# spectre du signal d'entrée
plt.figure(1)
plt.vlines(freq[:10],[0]*len(freq[:10]),ampl_e[:10], linestyle = 'solid', color␣
 ↪= 'red')
plt.xlabel('f (Hz)')
plt.ylabel('V_{en} (V)')
plt.title("Spectre du signal d'entrée")
plt.grid()

#spectre du signal de sortie :
plt.figure(2)
plt.vlines(freq[:10],[0]*len(freq[:10]),ampl_s[:10], linestyle = 'solid', color␣
 ↪= 'blue')
plt.xlabel('f (Hz)')
plt.ylabel('V_{sn} (V)')
plt.title('Spectre du signal de sortie')
plt.grid()
plt.show()
```





Le spectre du signal de sortie ressemble bien à celui d'un créneau.