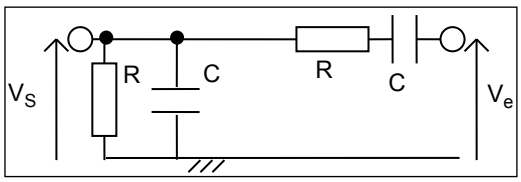
TP 4 : Oscillateur sinusoïdal : l'oscillateur à Pont de Wien

Un oscillateur est un montage autonome (pas de signal de commande) qui génère spontanément un signal alternatif lors de la mise sous tension. Il est quasi-sinusoïdal s'il délivre un signal quasiment dénué d'harmoniques (l'équation différentielle de ce signal approche celle de l'oscillateur harmonique). Il comporte toujours un élément actif (circuit amplificateur alimenté par un signal continu) associé à un circuit passif (filtre).
On peut citer l'oscillateur à résistance négative, l'oscillateur à réseau déphaseur et ***l'oscillateur à «pont de Wien»*** *étudié lors de cette séance de TP.*
On associera une simulation théorique (Python) aux observations et mesures expérimentales.

A. Éléments théoriques

1. On pourra démontrer (lors du compte-rendu) que le filtre suivant (pont de Wien) a pour fonction de transfert :

$$\underline{H}(j\omega) = \frac{1}{3 + j\left(RC\omega - \frac{1}{RC\omega}\right)}$$



Donner son facteur de qualité Q théorique et sa fréquence caractéristique f_0 théorique.
Ecrire l'équation différentielle reliant Vs(t) et Ve(t)

2. Lorsque le schéma de principe est une simple boucle fermée contenant l'amplificateur de gain $\underline{A}(\omega)$ et le filtre que constitue le réseau de réaction de fonction de transfert $\underline{H}(\omega)$, la condition limite d'oscillation est connue sous le nom de «critère de BARKHAUSEN» :

$$\underline{A}(\omega).\underline{H}(\omega) = 1 \Leftrightarrow \begin{cases} |\underline{A}(\omega)|.|\underline{H}(\omega)| = 1 \\ \arg\left(\underline{A}(\omega).\underline{H}(\omega)\right) = 0 \end{cases}$$

-Interprétez cette condition en imaginant les cas où $|\underline{A}(\omega)|.|\underline{H}(\omega)| < 1$ ou $|\underline{A}(\omega)|.|\underline{H}(\omega)| > 1$

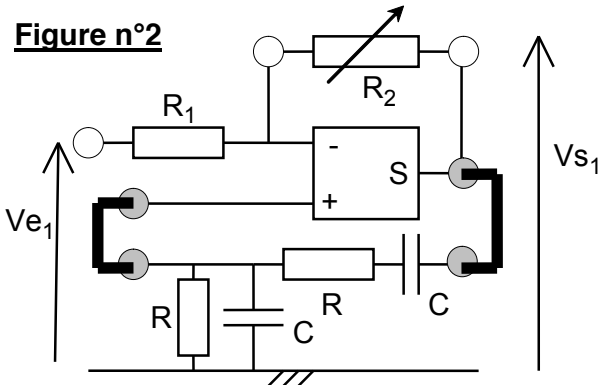
3. L'amplificateur non-inverseur utilisé a pour coefficient d'amplification :

$$\underline{A}(\omega) = (1 + k) \text{ avec } k = \frac{R_2}{R_1}$$

(sous réserve que l'ALI fonctionne bien en régime d'amplification)

Écrire l'équation différentielle régissant l'évolution de $V_{s1}(t)$ dans le circuit bouclé **($v_{e1}(t)=0$).**
Quelle condition doit être vérifiée pour avoir un oscillateur harmonique ?
Quelle équation différentielle ***(1)*** pour $V^+(t)$?

Figure n°2



En déduire un temps caractéristique d'évolution de l'amplitude du signal en cas de divergence ou de convergence vers zéro.

En cas de divergence de l'amplitude : $Vs_1(t)=\pm V_{sat}$ et est donc constante pendant cette phase.
Que vaut sa dérivée temporelle ?
Comment s'écrit alors l'équation différentielle ***(2)*** gouvernant l'évolution de la tension $V^+(t)$?

Pour la simulation en langage Python, on pourra préparer les expressions donnant la dérivée seconde fonction du reste :
$d^2V^+(t)/dt^2 = f(V^+(t), dV^+(t)/dt)$ dans les situations ***(1)*** *et* ***(2)***

B.Étude expérimentale

1.Filtre de Wien seul :

Obtenez une courbe de gain en dB suffisamment précise pour avoir f_{0exp} et Q_{exp} très précisément. (interrupteur en position I1)

La tension **d'entrée** dans le filtre gardait-elle son amplitude quelque soit la fréquence ? Interpréter.

2. Circuit bouclé

On utilisera une boite à décades 10 kΩ de résistances en série avec un potentiomètre de 1kΩ branché en rhéostat en série (fiché dans la plaquette LAB) pour réaliser la résistance notée R_2

2.1. Où branche-t-on le GBF ?

2.2. A-t-on des oscillations quelque soit la valeur de résistance du potentiomètre ? (faire de multiples essais pour estimer la valeur limite d'oscillations)

2.3. Imprimer et commenter l'allure des oscillations établies pour différentes valeurs de R_2.

2.4.Qu'est-ce qui semble fixer l'amplitude des oscillations ?

2.5.Imprimer et commenter la FFT du signal oscillant pour $R_{2limite}$ puis pour 10kΩ. Commenter

2.6. On cherche alors à observer le régime transitoire d'installation des oscillations.

2.6.1. Décrivez très précisément la méthode de déclenchement en mode monocoup (single)
2.6.2. Imprimer plusieurs régimes transitoires divergents d'amorçage des oscillations pour des valeurs de k proches de celles envisagées dans la partie théorique. Utiliser REGRESSI pour évaluer les temps caractéristiques de l'équation différentielle.

3.Portrait de phase de l'oscillateur.

3.1. Rappeler la définition d'un portrait de phase et ses propriétés

3.2. L'oscillo HP peut-il dériver le signal ? (difficultés rencontrées) Peut-il présenter le portrait de phase à l'écran ?

3.3.Utiliser REGRESSI pour tenter un relevé de portrait de phase. Quelle limitation nous empêche d'observer un portrait de phase «lisse» et continu ?

3.4. Proposer un montage à ALI supplémentaire pour réaliser ensuite un relevé propre du portrait de phase de l'oscillateur pour R_2=10 kΩ. (Des composants sont disponibles sur la paillasse professeur)

C.Simulation sous Python

1.Utilisation du module d'intégration d'équations différentielles ordinaires **scipy.integrate.odeint**

Dans ce TP nous ne chercherons pas à coder nous-même un schéma d'intégration d'équation différentielle (par Euler explicite ou implicite, Verlet ou encore Runge Kutta). Nous profiterons de cette modélisation d'une situation d'électronique pour maitriser la syntaxe du module **odeint** de cette librairie scientifique de Python.

Le « document technique » en Annexe 1 détaille la syntaxe et propose de l'illustrer sur la problématique du pendule simple vu en PTSI (équation d'ordre 2 donc comme dans notre cas)

Pour résumer, il faut **définir une fonction f** qui « calculera » les dérivées premières à partir de leurs équations (différentielle ou pas). Pour nous, les expressions seront notées **vplus** et **dvplusdt**. Cette fonction devra renvoyer une liste de deux valeurs constitués dans l'ordre de l'expression de la dérivée première dvplusdt puis l'expression de la dérivée seconde (obtenue en fin de partie A) (dérivée première de dvplusdt)

On remarque que la possibilité de deux régimes soit deux équations différentielles implique l'emploi d'une condition de type « if … return … else return …. » au sein de cette fonction.

Après avoir préparé un vecteur représentant les instants t d'évaluation des tensions et avoir choisi des valeurs numériques initiales (très faibles) pour vplus0 et sa dérivée dvplusdt0 (bruit électronique) , on peut alors appliquer :

sol = sp.integrate.odeint(f,[vplus0,dvplusdt0],t)

Le résultat **sol** est une matrice de deux « vecteurs » donnant les valeurs numériques de vplus aux instants t dans la première colonne (**sol[:,0]**) et dvplusdt dans la seconde (**sol[:,1]**).

Comme il s'agira également de représenter graphiquement Vs1(t) , on pourra par exemple utiliser **np.where** pour proposer **Vs1=A*vplus** lorsque l'ALI n'est pas saturé et **Vs1=vsat*np.sign(vplus)** lorsqu'il l'est.

Les représentations graphiques issues de cette simulation (chronogrammes, caractéristique et portrait de phase) seront prêtes à l'emploi dans un programme python à compléter.

2.Modifications pertinentes des paramètres de la simulation

On modifiera les valeurs numériques pour coller au mieux à la courbe expérimentale obtenue en mode single à l'oscillo numérique.

On proposera également des valeurs numériques éloignées de notre situation expérimentale pour extrapoler des comportements.

Valeurs par défaut dans le code proposé :

```
R=1e3                   # Resistance du Pont de Wien
C=100e-9                # Capacite du Pont de Wien
A=3.1                   # Gain de l'etage d'amplification
tfin=15e-3              # instant final de la simulation
dt=0.01e-3              # pas
vsat=13.2               # tension de saturation de l'ALI
```

Annexe 1 : syntaxe de **scipy.integrate.odeint**

Document technique DT1 : Fonction ODEINT de Scipy

Description

`sol=scipy.integrate.odeint(func, y0, t, args=())`

Integrate a system of ordinary differential equations. Solve a system of ordinary differential equations using `lsoda` from the FORTRAN library `odepack`. Solves the initial value problem for stiff or non-stiff systems of first order ode-s:

`dy/dt = func(y, t0, ...)`, where `y` can be a vector.

Parameters

`func` : callable(y, t0, ...), computes the derivative of `y` at `t0`.
`y0` : array, initial condition on `y` (can be a vector).
`t` : array, a sequence of time points for which to solve for `y`. The initial value point should be the first element of this sequence.
`args` : tuple, optional, extra arguments to pass to function.

Returns

`sol` : array, `shape (len(t), len(y0))`, array containing the value of `y` for each desired time in `t`, with the initial value `y0` in the first row.

Example

The second order differential equation for the angle theta of a pendulum acted on by gravity with friction can be written:

$$\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} + c.\sin(\theta) = 0$$

Where `b` and `c` are positive constants. To solve this equation with **`odeint`**, we must first convert it to a system of first order equations. By defining the angular velocity `Ω(t)=θ'(t)`, we obtain the system:

$$\begin{cases} \dfrac{d\theta}{dt} = \Omega(t) \\ \dfrac{d\Omega}{dt} = -b\dfrac{d\theta}{dt} - c.\sin(\theta) \end{cases}$$

Let `y` be the vector [theta, omega]. We implement this system in Python as:

```
def pend(y,t,b,c):
    theta,omega=y
    dydt=[omega,-b*omega-c*math.sin(theta)]
    return dydt
```

For initial conditions, we assume the pendulum is nearly vertical with `theta(0)=pi-0.1`, and it initially at rest, so `omega(0)=0`. Then the vector of initial conditions is:

```
y0=[math.pi-0.1,0.0]
```

We generate a solution 101 evenly spaced samples in the interval 0 <= t <= 10. So our array of times is:

```
t=numpy.linspace(0,10,101)
```

Call `odeint` to generate the solution. To pass the parameters `b` and `c` to the **pend** function; we give them to `odeint` using the `args` argument:

```
from scipy.integrate import odeint
sol=odeint(pend,y0,t,args=(b,c))
```

Annexe 2 : Programme Python « oscillateur à pont de Wien »

```python
import numpy as np              # Outils numeriques
import scipy as sp              # Outils scientifiques
import scipy.integrate          # pour l'integration
import matplotlib.pyplot as plt # Outils graphiques

R=1e3                           # Resistance du Pont de Wien
C=100e-9                        # Capacite du Pont de Wien
A=3.1                           # Gain de l'etage d'amplification
tfin=15e-3                      # instant final de la simulation
dt=0.01e-3                      # pas
vsat=13.2                       # tension de saturation de l'ALI

vplus0   = 0.01 # entree initiale de l'etage d'amplitication ; 0.01 simule le bruit initial
dvplusdt0= 5 # derivee de vplus initiale ; la valeur de variation (modifiable) simule le bruit en entree

def f(y,t):
    vplus,dvplusdt = y
    # à completer
    # à completer
    # à completer
    # à completer

# Determination de l'entree de l'etage d'amplification
t   =    # à completer
sol = sp.integrate.odeint(   # à completer)
vplus   = sol[:,0]
dvplusdt= sol[:,1]

# Pour la sortie, on utilise np.where pour ecreter là où vplus est trop grand
vs = np.where(   # à completer)

# Traces des courbes
plt.figure(1,figsize=(14,8))
plt.subplot(2,2,1)      # Figure du haut à gauche
plt.plot(t,vs,'r')          # La sortie de l'ALI
plt.title("Démarrage des oscillations")
plt.xlabel("$t$ (s)")
plt.ylabel("$vs$ (V)")
plt.ylim(-15,15)        # Pour bien voir l'ecretage
plt.subplot(2,2,2)      # Figure du haut à droite
plt.plot(vplus,vs,'k')          # La sortie de l'ALI
plt.title("Caractéristique balayée lors des oscillations")
plt.xlabel("$v+$ (V)")
plt.ylabel("$vs$ (V)")
plt.ylim(-15,15)        # Pour bien voir les saturations
plt.axis('equal')
plt.subplot(2,2,3)      # Figure du bas à gauche
plt.plot(t,vplus,'g')          # L'entree
plt.xlabel("$t$ (s)")
plt.ylabel("$v+$ (V)")
plt.subplot(2,2,4) # Figure du bas à droite
plt.plot(vplus,dvplusdt,'b')       # Le portrait de phase
plt.xlabel("$v+$ (V)")
plt.ylabel("d$v+$/d$t$ (V/s)")
plt.title('Portrait de phase')
plt.tight_layout()   # Pour ajuster les espaces autour des sous-figures
# plt.savefig('PNG/PSI_pont_de_wien_odeint.png')
```

```python
# Sauf mention explicite du contraire par la suite, ce travail a été fait par
# Jean-Julien Fleck, professeur de physique/IPT en PCSI1 au lycée Kléber.
# Vous êtes libres de le réutiliser et de le modifier selon vos besoins.


"""
Version légèrement modifiée à l'usage des PT Benjam ORLEANS'
"""
```