Capacité numérique

À l'aide d'un langage de programmation, simuler l'évolution temporelle d'un signal généré par un oscillateur.

On s'intéresse dans ce document à la tension v générée par l'oscillateur à résistance négative dont le schéma est donné ci-dessous :

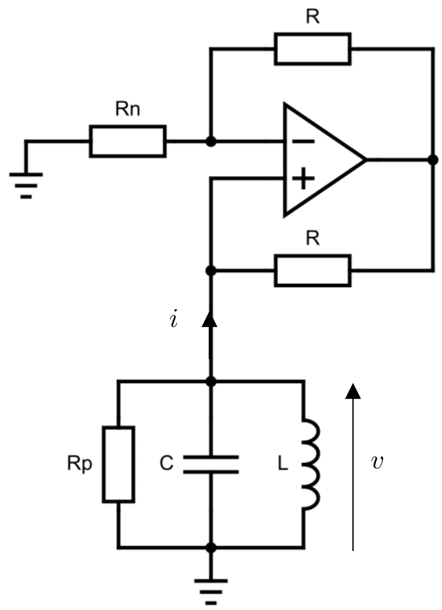

FIGURE 1 – Schéma de l'oscillateur à résistance négative

L'objectif est de simuler l'évolution temporelle de la tension v à l'aide de la méthode d'Euler explicite.

Le script `Python` utilisé est retranscrit dans l'annexe A du présent document et sa version exécutable est également jointe à cette ressource.

A. Mise en équation

Dans la suite, l'ALI est supposé idéal et on distingue trois régimes de fonctionnement :

- Le régime linéaire pour lequel la valeur absolue de la tension en sortie d'ALI est inférieure à la tension de saturation de celui-ci ($|v_s| < v_{sat}$). Les potentiels en entrée de l'ALI sont alors égaux ($v_+ = v_-$) et on en déduit que les deux résistances R sont traversées par le même courant. On peut donc écrire :

$$v = v_+ = v_- = \frac{R_n}{R_n + R} v_s = -R_n i$$

On en déduit que :

$$\text{si } |v| < \frac{R_n}{R_n + R} v_{sat} \text{ alors } v = -R_n i$$

- Le régime de saturation positive :

$$\text{si } v > \frac{R_n}{R_n + R} v_{sat} \text{ alors } v = Ri + v_{sat}$$

- Le régime de saturation négative :

$$\text{si } v < -\frac{R_n}{R_n + R} v_{sat} \text{ alors } v = Ri - v_{sat}$$

Déterminons alors l'équation différentielle vérifiée par v dans chaque cas :

- En régime linéaire ($|v| < \frac{R_n}{R_n+R} v_{sat}$) :

$$i + i_{R_p} + i_L + i_C = 0$$

Après dérivation, il vient :

$$\frac{d^2v}{dt^2} + \frac{1}{C}\left(\frac{1}{R_p} - \frac{1}{R_n}\right)\frac{dv}{dt} + \frac{1}{LC} v \ = \ 0$$

Finalement, en posant $\omega_0 = \frac{1}{\sqrt{LC}}$ et $m = \frac{1}{2}\left(\frac{1}{R_p} - \frac{1}{R_n}\right)\sqrt{\frac{L}{C}}$:

$$\begin{cases} \dfrac{d^2v}{dt^2} + 2m\omega_0 \dfrac{dv}{dt} + {\omega_0}^2 v \ = \ 0 \\ v_s = \dfrac{R_n + R}{R_n} v \end{cases}$$

- En régime de saturation ($|v| > \frac{R_n}{R_n+R} v_{sat}$), l'équation devient :

$$\frac{d^2v}{dt^2} + \frac{1}{C}\left(\frac{1}{R_p} + \frac{1}{R}\right)\frac{dv}{dt} + \frac{1}{LC} v \ = \ 0$$

En posant $p = \frac{1}{2}\left(\frac{1}{R_p} + \frac{1}{R}\right)\sqrt{\frac{L}{C}}$:

$$\begin{cases} \dfrac{d^2v}{dt^2} + 2p\omega_0 \dfrac{dv}{dt} + {\omega_0}^2 v \ = \ 0 \\ v_s = \pm\, v_{sat} \end{cases}$$

B. Schémas numériques

L'utilisation de la méthode d'Euler explicite implique de transformer l'équation différentielle du second ordre en un système d'équations différentielles du premier ordre. On introduit alors la fonction $w = \frac{dv}{dt}$.

En régime linéaire ($|v| < \frac{R_n}{R_n+R} v_{sat}$), on peut écrire :

$$\begin{cases} \dfrac{dv}{dt}(t) = w(t) \\ \dfrac{dw}{dt}(t) = -2m\omega_0 w(t) - {\omega_0}^2 v(t) \end{cases}$$

En introduisant le pas temporel Δt, la méthode d'Euler explicite permet d'obtenir le schéma numérique suivant :

$$\begin{cases} v[k+1] = v[k] + w[k]\Delta t \\ w[k+1] = (1 - 2m\omega_0 \Delta t)w[k] - ({\omega_0}^2 \Delta t)v[k] \end{cases}$$

Le régime étant linéaire, on détermine les valeurs prises par la tension v_s via :

$$v_s[k+1] = \frac{R_n + R}{R_n} v[k+1]$$

En régime de saturation, le schéma numérique devient :

$$\begin{cases} v[k+1] = v[k] + w[k]\Delta t \\ w[k+1] = (1 - 2p\omega_0 \Delta t)w[k] - ({\omega_0}^2 \Delta t)v[k] \end{cases}$$

La tension v_s prend alors les valeurs $\pm\ v_{sat}$.

C. Résultats de simulation

On choisit $R_n = 800\ \Omega$, $R_p = 1000\ \Omega$, $R = 1500\ \Omega$, $C = 100$ nF et $L = 40$ mH et on initialise la tension v avec les valeurs de 0 V et 0,001 V. Pour ce jeu de paramètres, $m = \frac{1}{2}\left(\frac{1}{R_p} - \frac{1}{R_n}\right)\sqrt{\frac{L}{C}} = -0{,}079 < 0$ et les oscillations peuvent apparaître à partir du bruit.

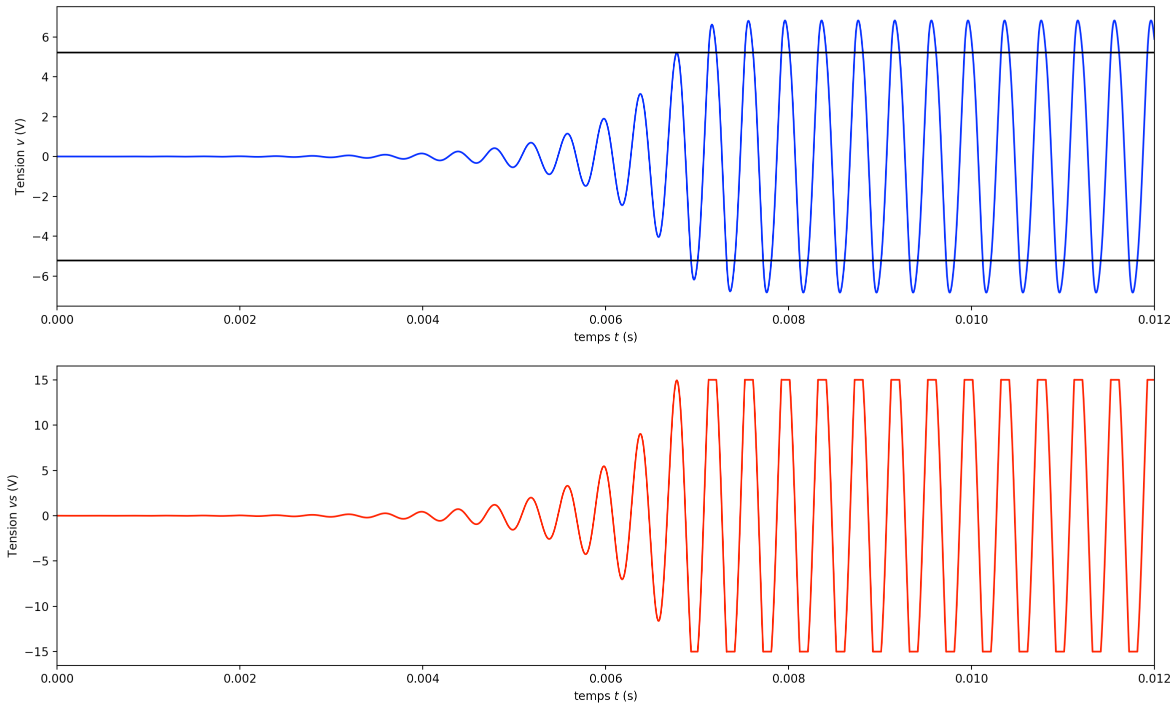

FIGURE 2 – Tensions v et v_s en fonction du temps pour une valeur de $m = -0{,}024$.

La valeur de m est suffisamment faible pour que la fréquence des oscillations ($f_{sim} = 2496\ Hz$), soit très proche de $\frac{\omega_0}{2\pi}$ ($= 2516\ Hz$). Pour des valeurs de m négatives et plus grandes en valeur absolue ($R_n = 500\ \Omega$ donne m = - 0,316), on observe que l'ALI arrive plus vite en régime de saturation et y reste plus longtemps.

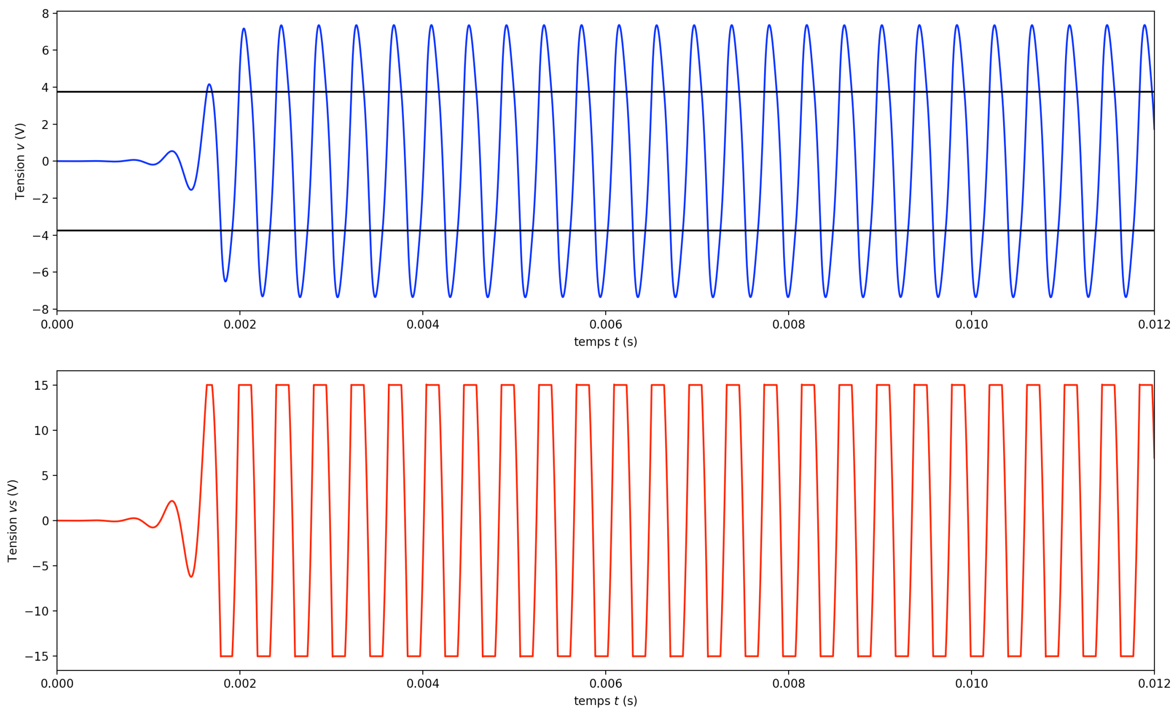

FIGURE 3 – Tensions v et v_s en fonction du temps pour une valeur de $m = -0{,}024$.

La fréquence observée ($f_{sim} = 2438\ Hz$) s'éloigne davantage de $\frac{\omega_0}{2\pi}$ ($= 2516\ Hz$).

Enfin, pour des valeurs positives de m ($R_n = 1100\ \Omega$ donne $m = 0{,}029$), l'ALI ne permet pas l'amplification du bruit.

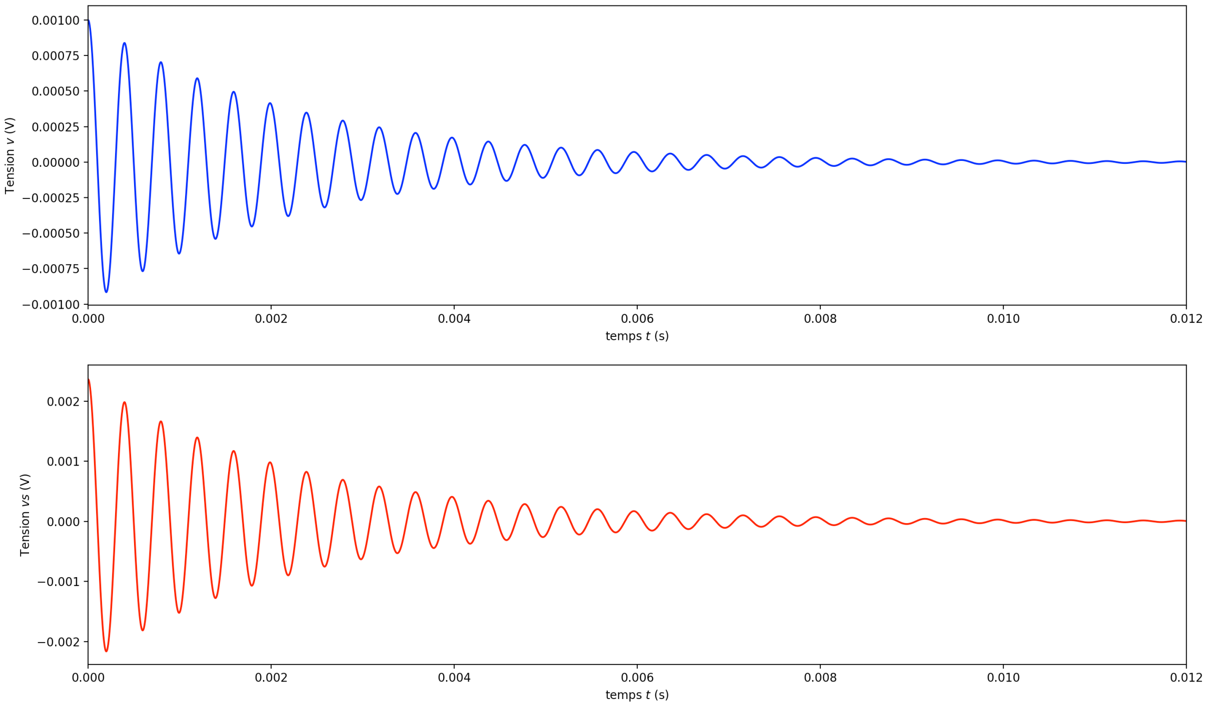

FIGURE 4 – Tensions v et v_s en fonction du temps pour une valeur de $m = 0{,}029$

Annexe A : Script Python

```python
## Importation des bibliothèques

import numpy as np
import matplotlib.pyplot as plt


## Paramétrage des grandeurs électriques

Rn = 800 # En Ohm
Rp = 1000 # En Ohm
R = 1500 # En Ohm
C = 10**(-7) # En farad
L = 0.04# En henry
v_sat = 15 # En Volt

m = 0.5 * ((1 / Rp) - (1 / Rn)) * np.sqrt(L / C)
p = 0.5 * ((1 / Rp) + (1 / R)) * np.sqrt(L / C)
om = 1 / np.sqrt(L * C) # en sec^(-1)
v_com = (Rn / (R + Rn)) * v_sat # Valeur de v pour laquelle il y a changement de régime de l'ALI (en Volt)


## Conditions initiales

v_0 = 0.001 # En Volt
w_0 = 0.001 # En Volt par seconde


## Paramétrage et initialisation de la simulation

duree = 0.012 # Durée de la simulation en seconde
dt = 10**(-7) # Pas temporel en seconde

N = 1 + int(duree / dt) # Nombres d'échantillons

t = np.array([i * dt for i in range(N)])
v = np.array([v_0] + [0 for i in range(N - 1)])
w = np.array([w_0] + [0 for i in range(N - 1)])
v_s = np.array([v_0 * (1 + R/Rn) ] + [0 for i in range(N - 1)]) # On suppose qu'à l'instant initial le régime est linéaire

v_com_pos = np.array([v_com for i in range(N)])
v_com_neg = np.array([- v_com for i in range(N)])



## Génération des tensions v et vs

for i in range(0, N - 1):

        if v[i] < v_com and v[i] > - v_com :

            v[i + 1] = v[i] + w[i] * dt
            w[i + 1] = w[i] * (1 - 2 * m * om * dt) - (om**2) * dt * v[i] # Schéma numérique en régime linéaire
            v_s[i +1] = v[i + 1] * (1 + R/Rn)

        else :

            v[i + 1] = v[i] + w[i] * dt
            w[i + 1] = w[i] * (1 - 2 * p * om * dt) - (om**2) * dt * v[i] # Schéma numérique en régime saturé

            if v[i] < - v_com :

                v_s[i +1] = - v_sat

            else :

                v_s[i +1] = v_sat
```

```python
## Affichage des tensions

plt.subplot(211)
plt.plot(t,v,'b')
plt.plot(t,v_com_pos,'k')
plt.plot(t,v_com_neg,'k')
plt.axis([t[0] - dt, t[N - 1] + dt, 1.1 * min(v), 1.1 * max(v)])
plt.xlabel("temps $t$ (s)")
plt.ylabel("Tension $v$ (V)")

plt.subplot(212)
plt.plot(t,v_s,'r')
plt.axis([t[0] - dt, t[N - 1] + dt, 1.1 * min(v_s), 1.1 * max(v_s)])
plt.xlabel("temps $t$ (s)")
plt.ylabel("Tension $vs$ (V)")

plt.show()
```