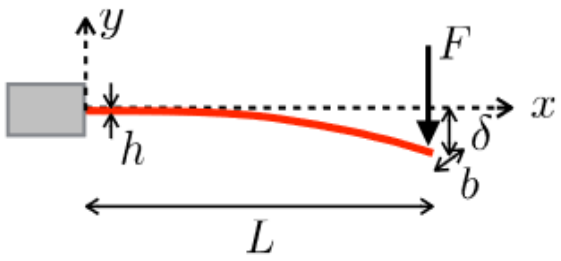
TIPE - Santé et prévention
Bras Elévateur Aérien

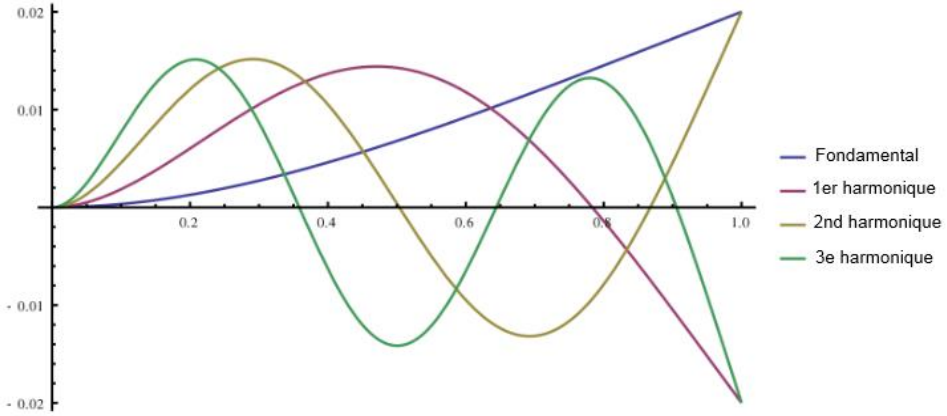
Sommaire

- Introduction
- Modélisation
- Caractéristiques du système
- Dimensionnement et élaboration du filtre réjecteur
- Expériences



Flexion d'une poutre encastrée, chargée à son extrémité.

Déplacement maximal par mode en fonction de la position de la poutre encastrée

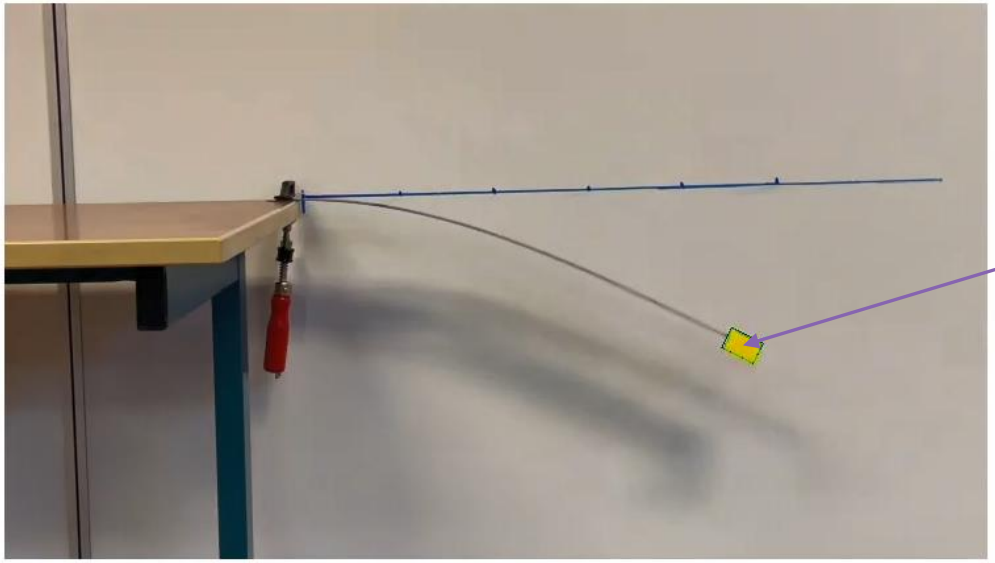


Phénomènes d'oscillations

source [3] https://blog.espci.fr/mecaflu/files/2019/02/vibrations.pdf

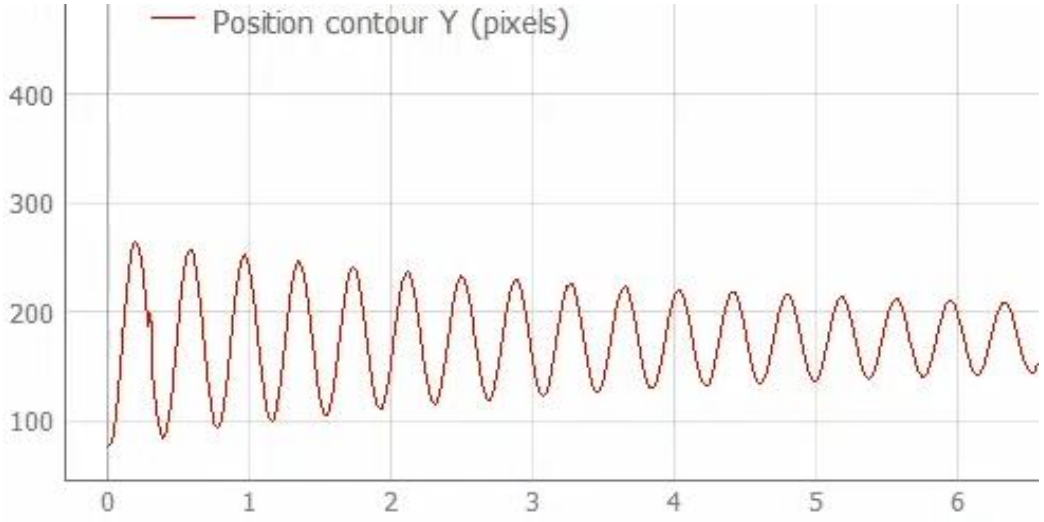
Phénomènes d'oscillations

Expérience mettant en évidence la présence d'oscillations sur une règle métallique



Repère coloré permettant le pointage par un logiciel

Vidéo d'une règle métallique oscillante



image

Pointage numérique de la vidéo

Objectif

- Réduire les oscillations au bout du bras : la nacelle

Solution

- Implanter un filtre réjecteur numérique qui modifie la consigne de position pour réduire les oscillations

Image: source [6] Flexible multibody system modelling of an aerial rescue ladder using Lagrange's equations

Filtre réjecteur

Fonction de transfert du filtre réjecteur

Fonction de transfert du moteur

Fonction de transfert du système oscillant

Problématique

Dans quelle mesure le filtre réjecteur permet de modifier la consigne afin de réduire les oscillations ?

Image: source [6] Flexible multibody system modelling of an aerial rescue ladder using Lagrange's equations

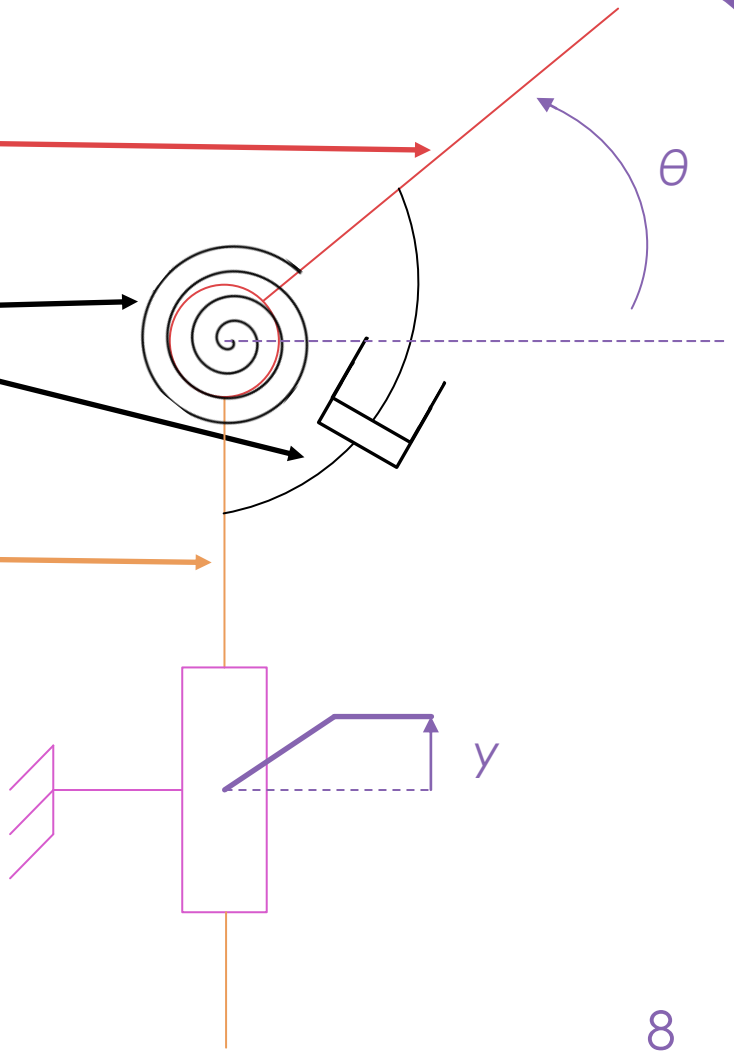
Modélisation – 1ère approche

Bras élévateur indéformable

Système ressort de torsion/ amortisseur modélisant la déformation du bras élévateur

Avant-bras indéformable

Bâti



Modélisation – 2ème approche

Fonction de transfert du filtre réjecteur

Fonction de transfert approximée du moteur choisi (moteur pas à pas)

Fonction de transfert du système oscillant

Modélisation – 2ème approche

$$H(p) = \frac{1}{1 + \frac{2\xi}{\omega_0}p + \frac{1}{\omega_0^2}p^2}$$

Fonction de transfert estimée du bras oscillant

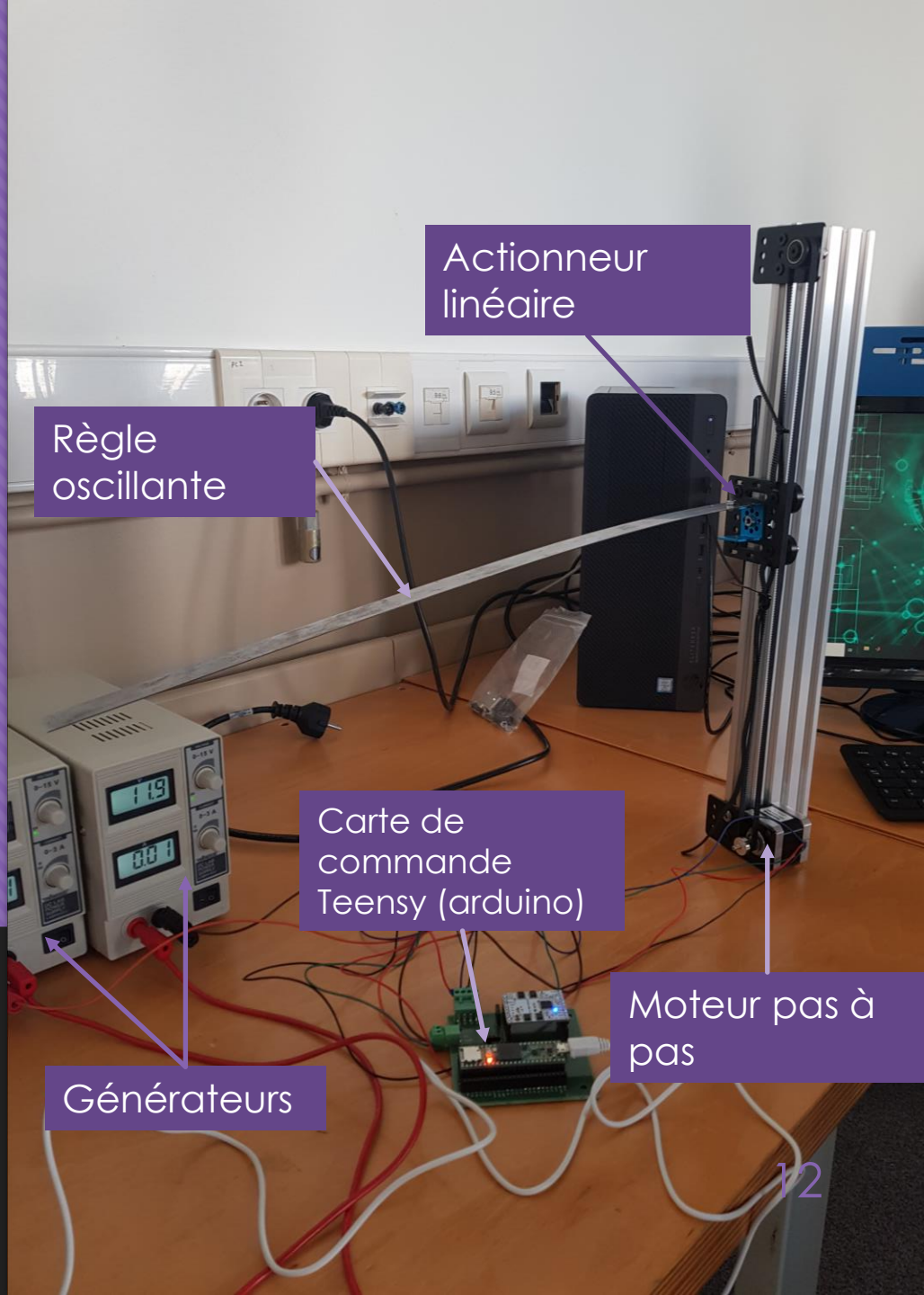
Modélisation – 2ème approche

$$F(p) = \frac{1 + \frac{2\xi}{\omega_0}p + \frac{1}{\omega_0^2}p^2}{1 + \frac{2\xi_1}{\omega_0}p + \frac{1}{\omega_0^2}p^2}$$

Fonction de transfert choisie du filtre réjecteur

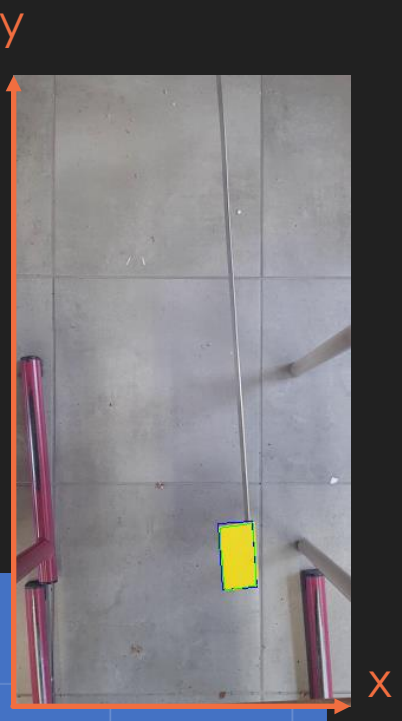
Caractéristiques du système

Photo du système réel

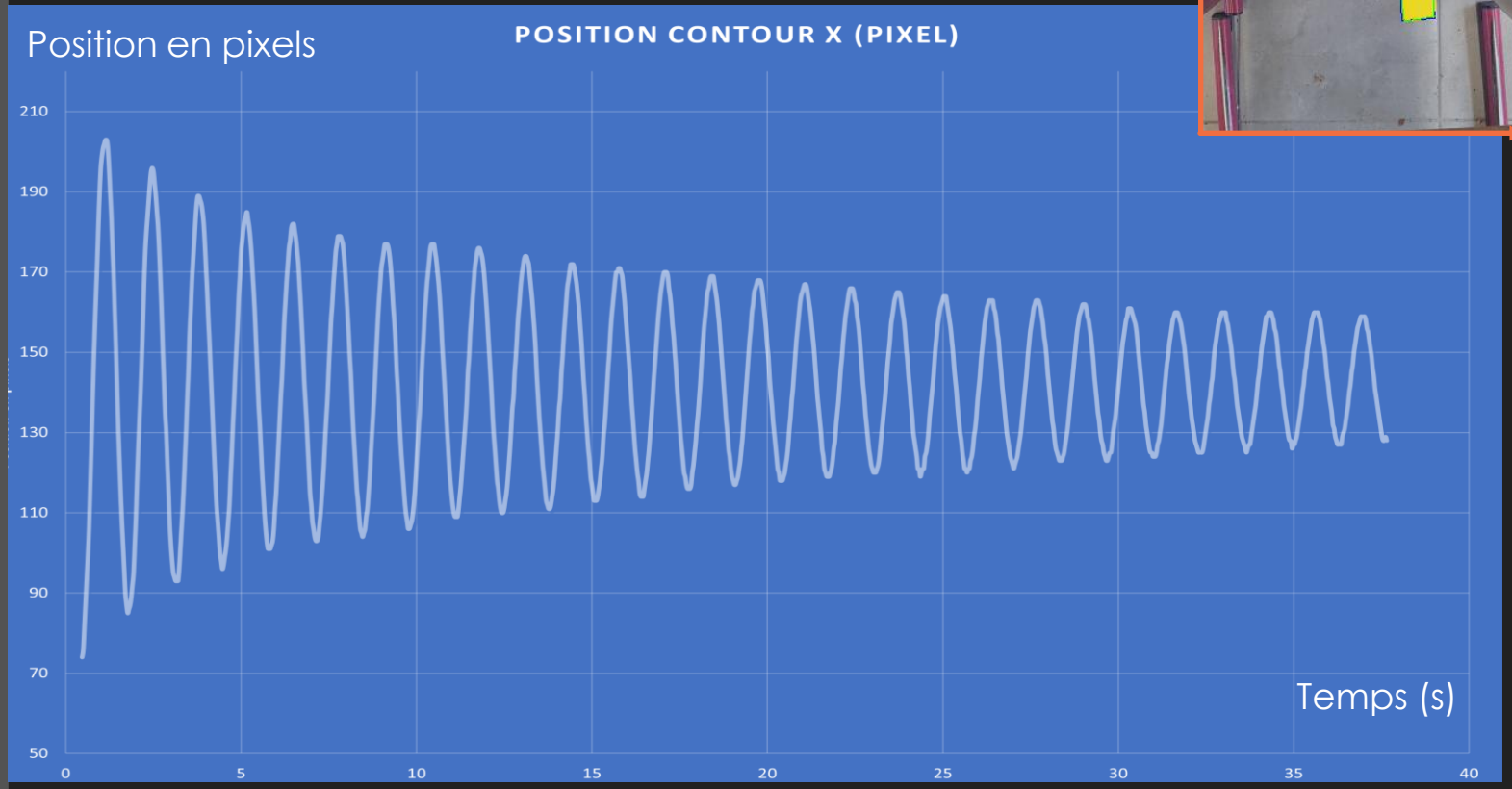


Caractéristiques du système

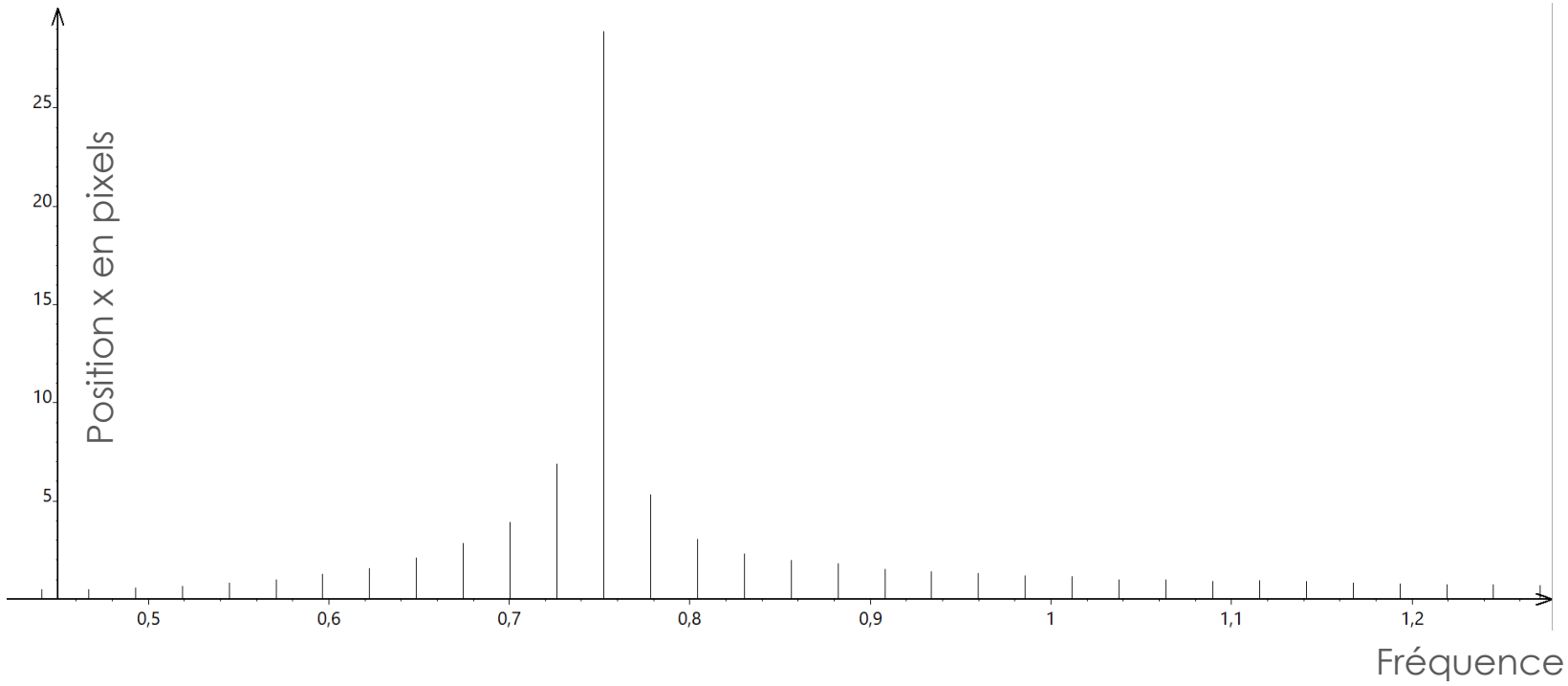
Vidéo de l'acquisition avec repères



Position x du bout de la règle en fonction du temps, pour une excitation sans amortissement

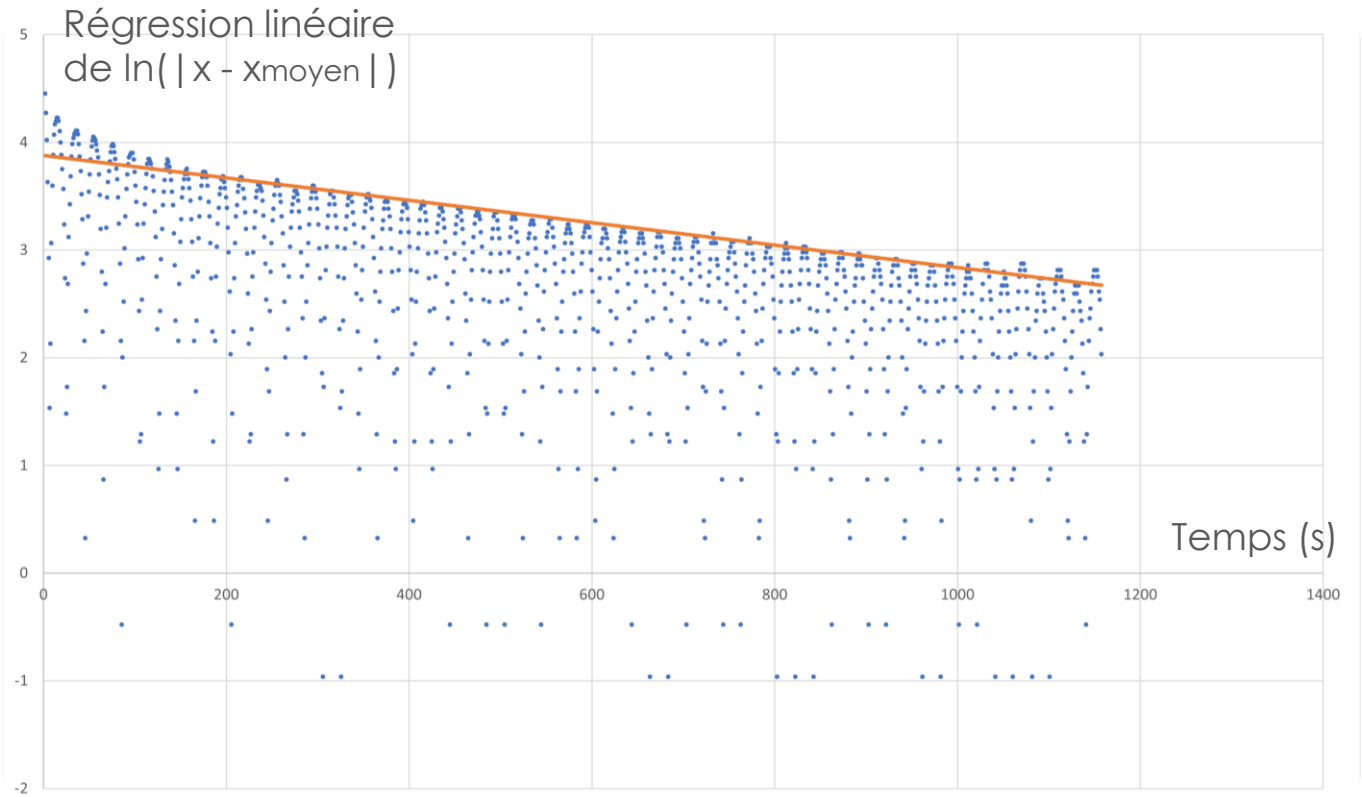


Caractéristiques du système



Analyse de Fourier de l'acquisition vidéo

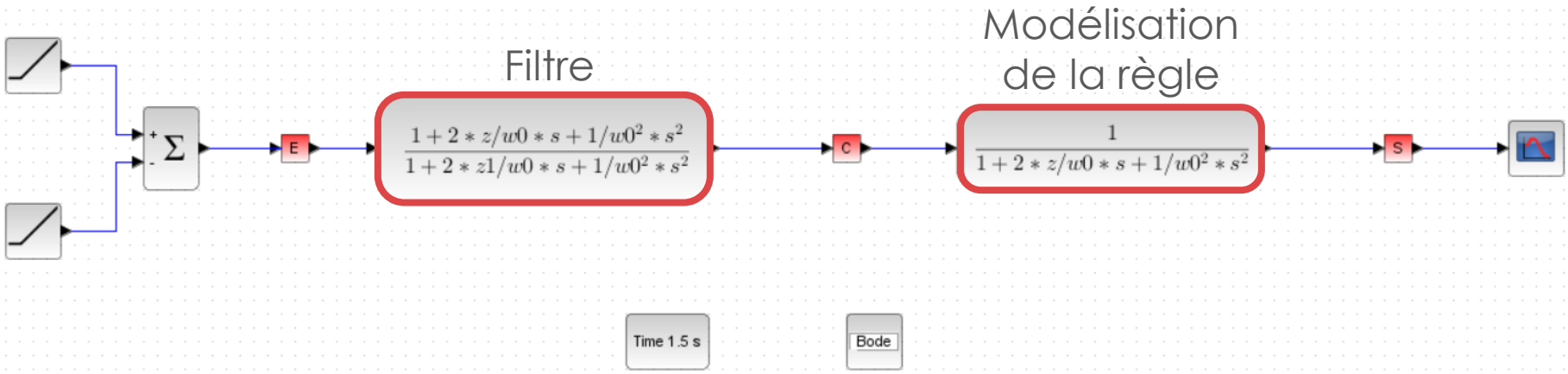
Caractéristiques du système



Analyse de la linéarité de la décroissance exponentielle

Dimensionner le filtre réjecteur

- Utilisation du logiciel Scilab



Implantation du filtrage dans Scilab

Dimensionner le filtre réjecteur

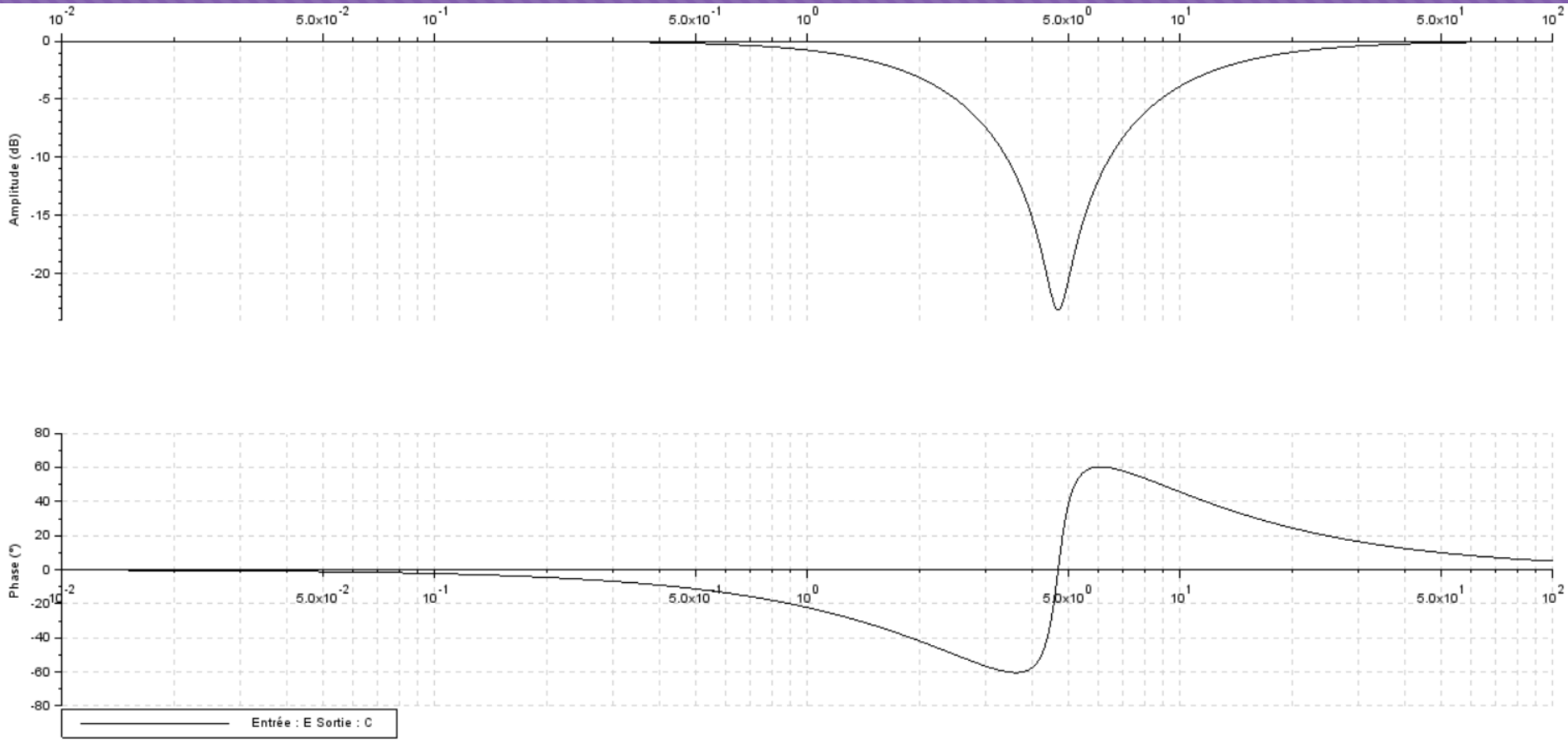
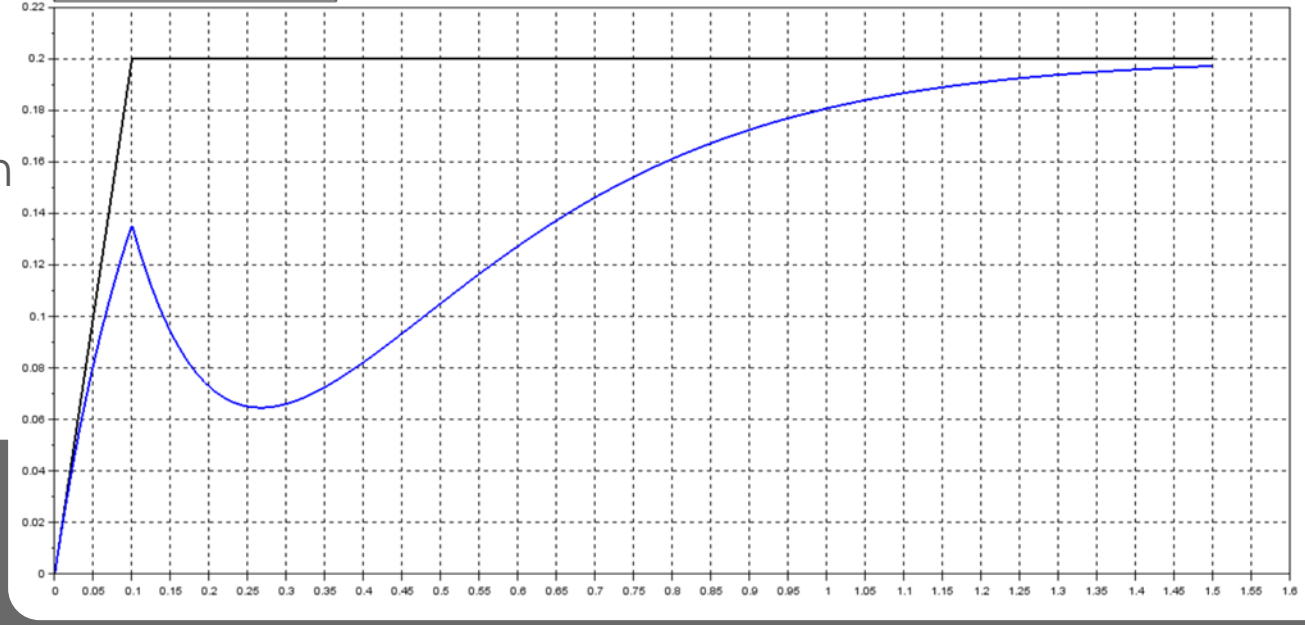


Diagramme de Bode du filtre

Dimensionner le filtre réjecteur

— Consigne de position

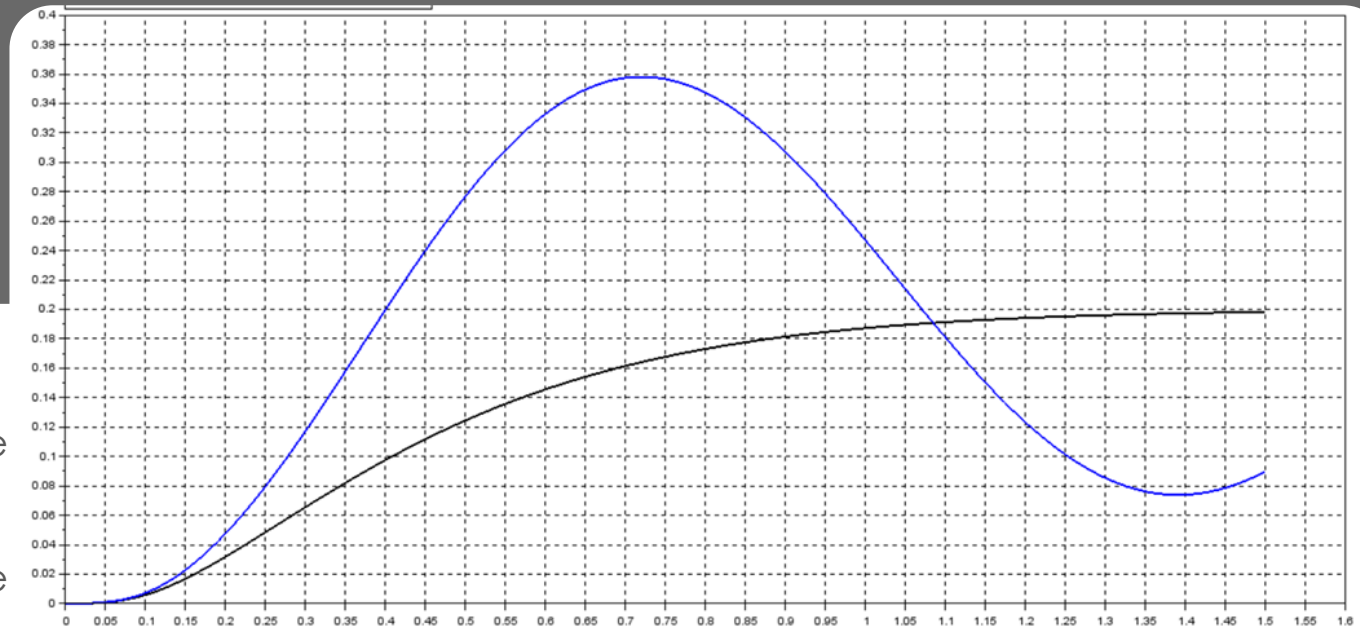
— Position filtrée



Positions de consigne en fonction du temps

— Position en bout de règle filtrée

— Position en bout de règle sans filtre



Positions en bout de règle en fonction du temps

$$F(p) = \frac{1 + \frac{2\xi}{\omega_0}p + \frac{1}{\omega_0^2}p^2}{1 + \frac{2\xi_1}{\omega_0}p + \frac{1}{\omega_0^2}p^2}$$

Elaborer le filtre réjecteur

F(p) = S/E

$$s(t) + c\frac{d}{dt}s(t) + b\frac{d^2}{dt^2}s(t) = e(t) + a\frac{d}{dt}e(t) + b\frac{d^2}{dt^2}e(t)$$

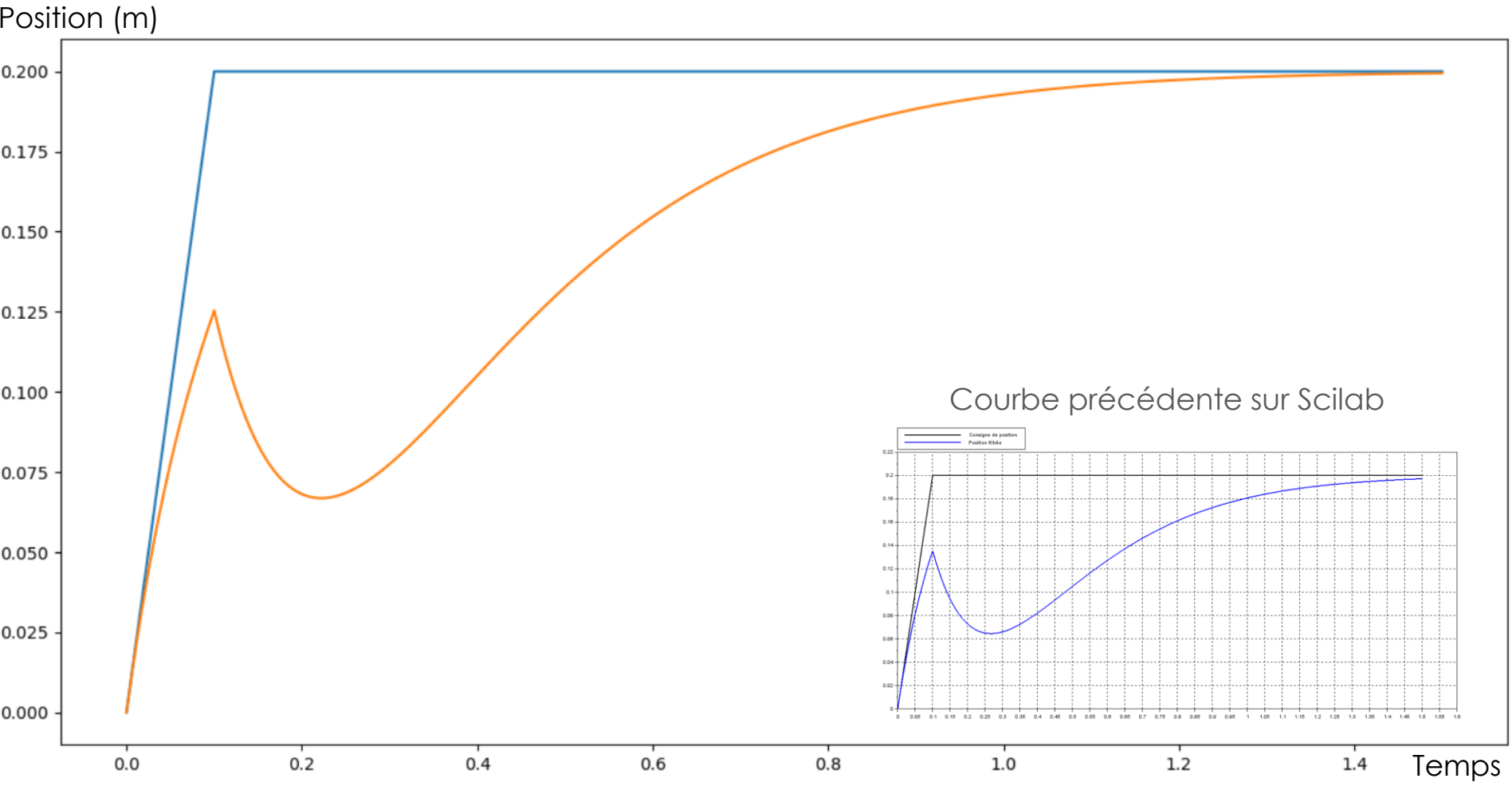
$$\frac{d}{dt}s(kT_e) \approx \frac{s_k - s_{k-1}}{T_e}$$

Equations de récurrence pour l'implantation séquentielle

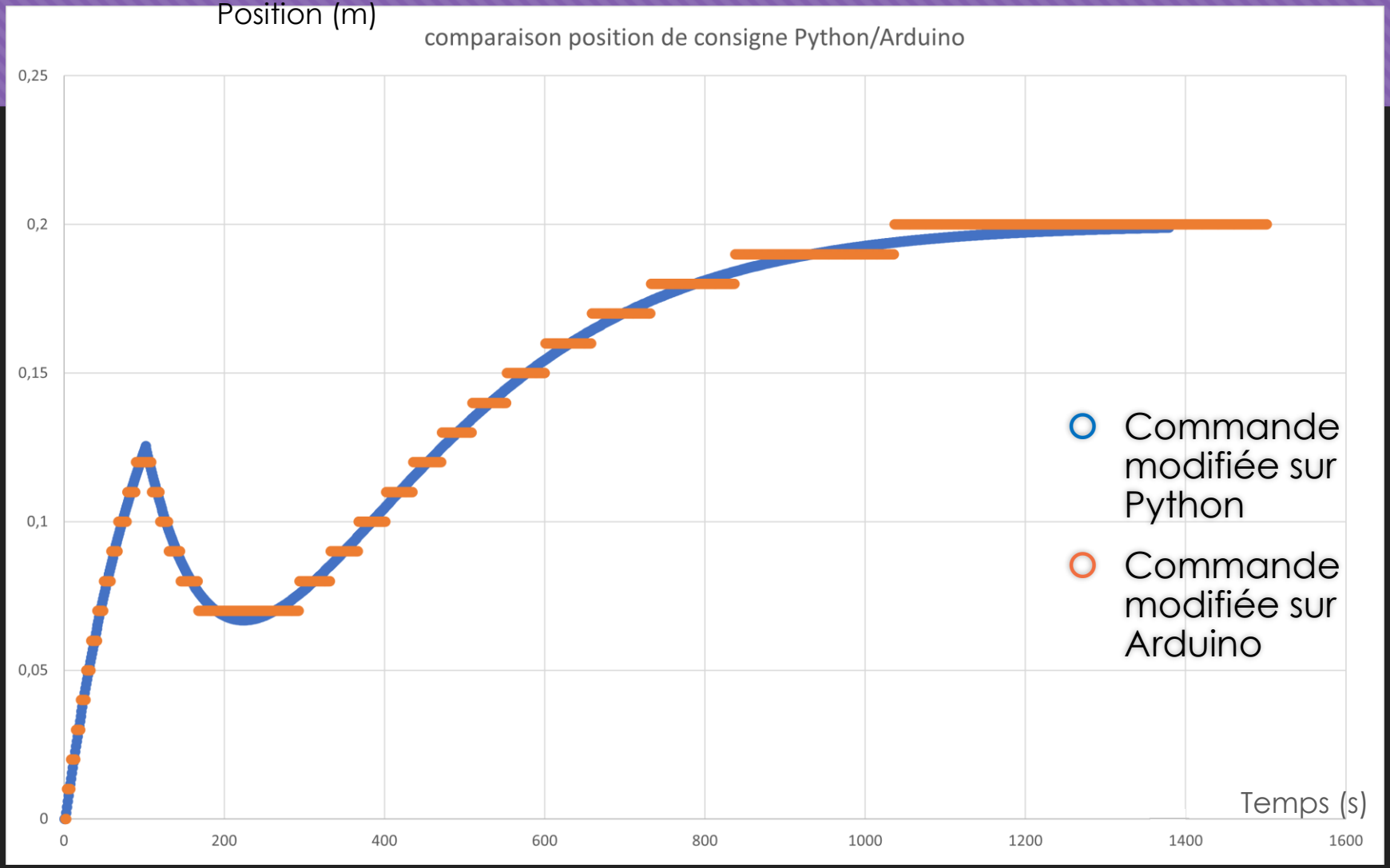
$$\frac{d^2}{dt^2}s(kT_e) \approx \frac{s_k - 2s_{k-1} + s_{k-2}}{T_e^2}$$

Dimensionner le filtre réjecteur

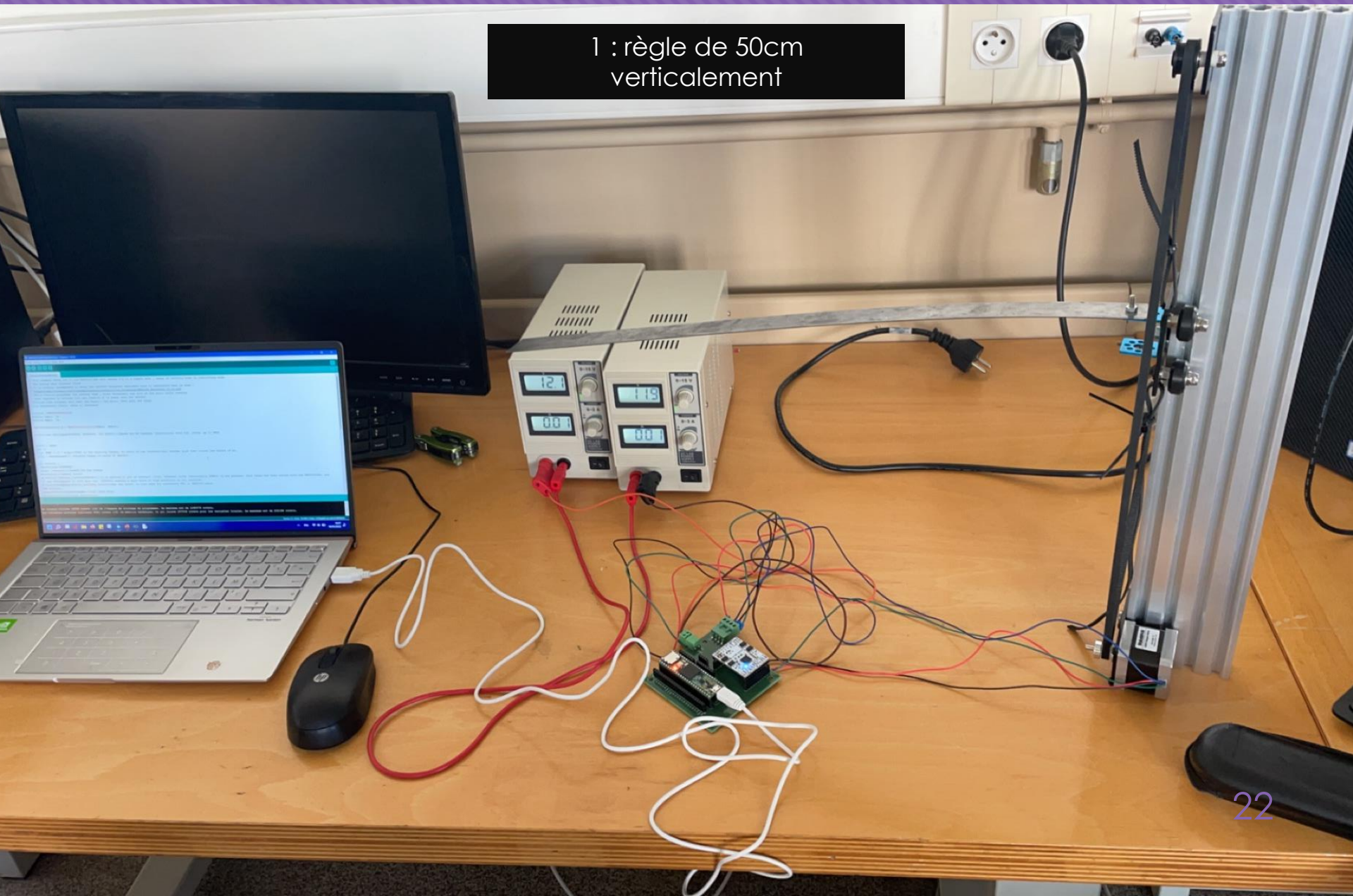
Utilisation de Python pour l'implantation séquentielle



Elaborer le filtre réjecteur



Expériences



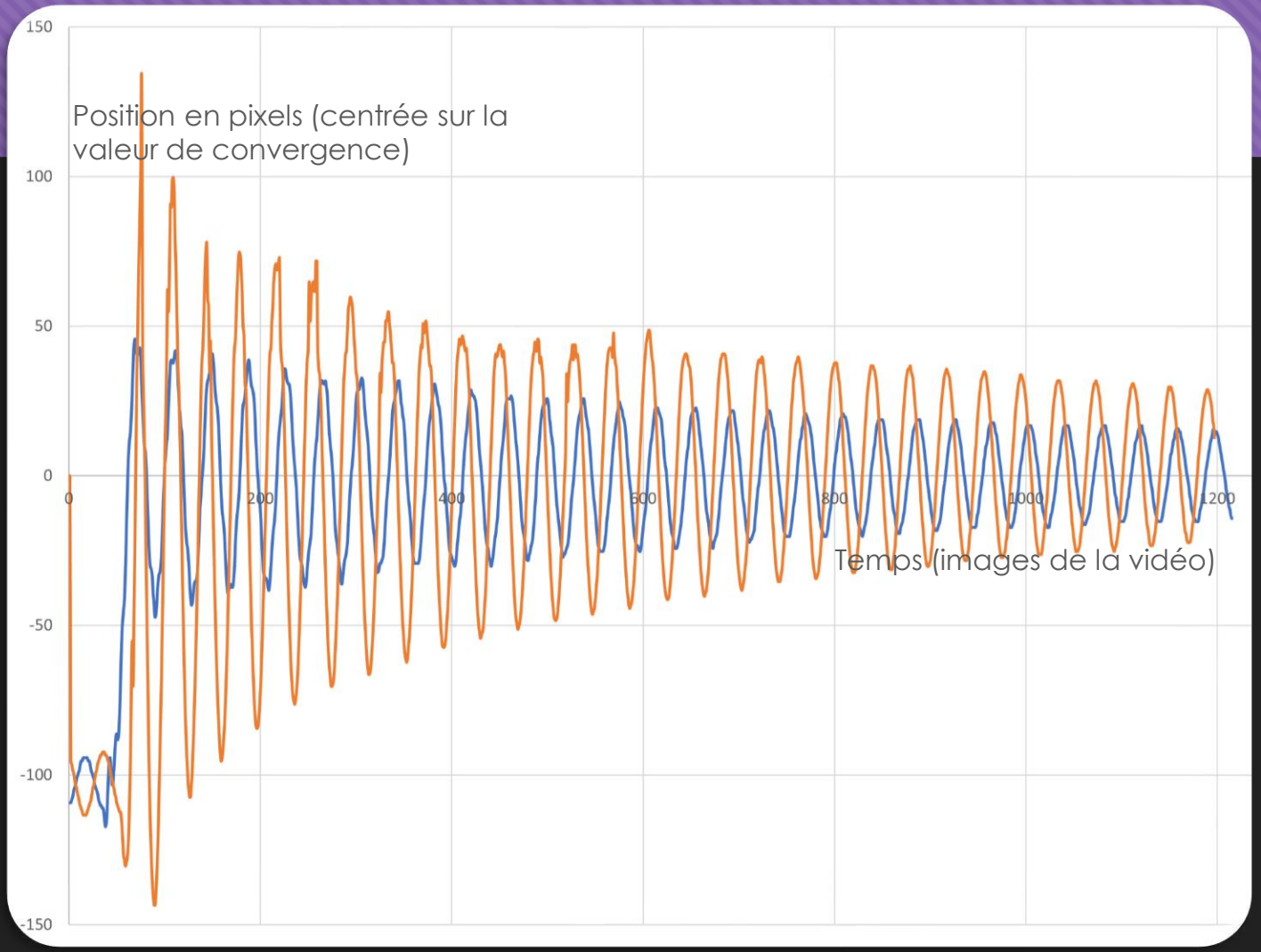
Expériences

2 : règle d’ 1m positionnée horizontalement

(plus facile à amortir car pulsation de résonance plus faible)

Conclusion

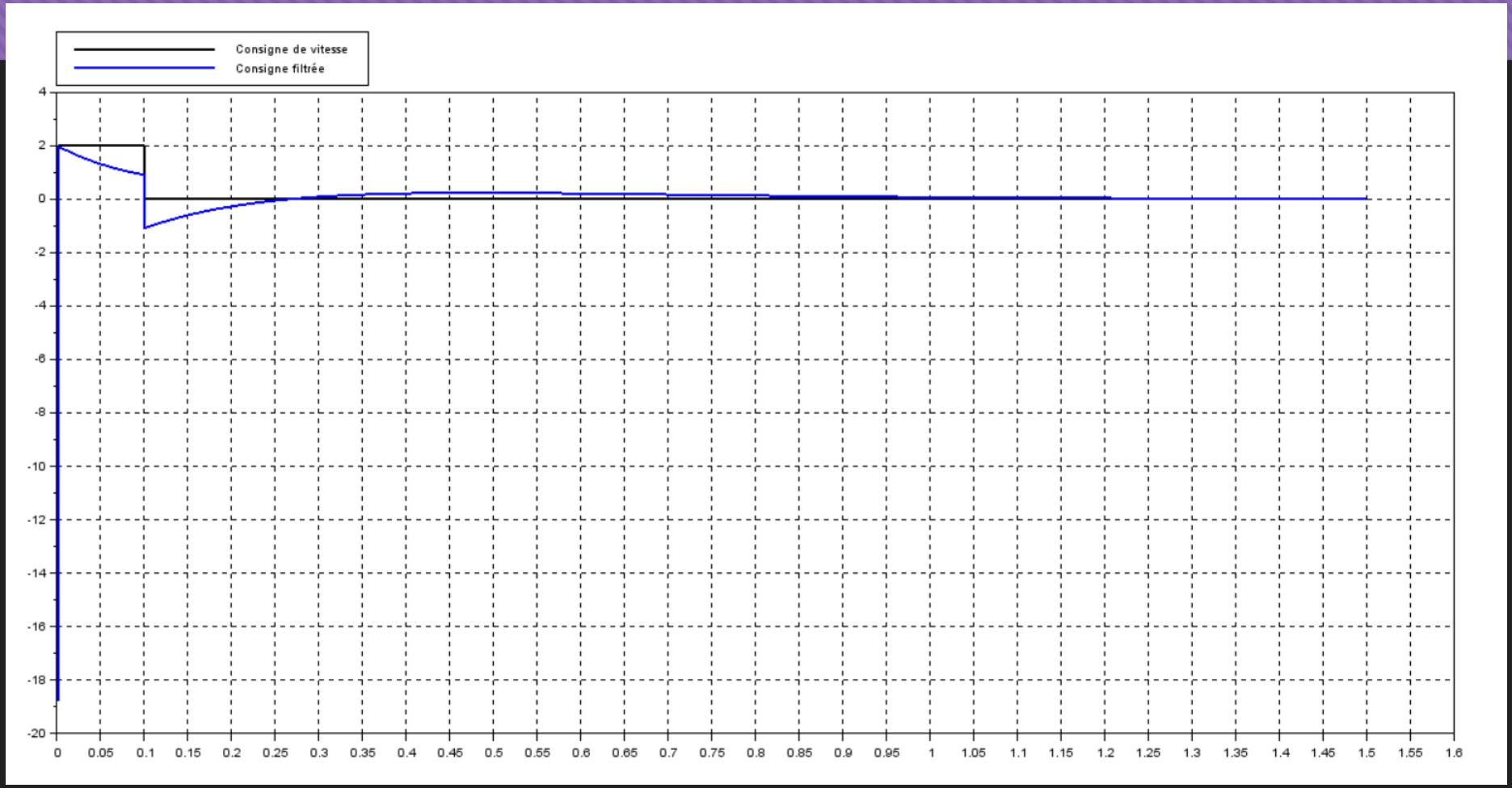
- Sans filtre réjecteur (consigne en rampe puis saturation)
- Avec filtre réjecteur



Résultats par pointage numérique des expériences

Annexes

Dimensionner le filtre réjecteur



Elaborer le filtre réjecteur

```
"""
Correcteur pour une consigne en "trapèze"
"""

import matplotlib.pyplot as plt
import numpy as np

##valeurs
w0 = 6
z = 0.07
z1 = 1

##pas
Te =0.001
nbp = int(1.5/Te)

##abscisse ordonnée
T=np.linspace(0, 1.5, nbp)
X=[0,2*Te]
Y=[0,2*Te]


##fonction de récurrence
def rec(s1,e1,s2,e2,e):
    a = 2*z/(w0*Te)
    b = 1/(w0*Te)**2
    c = 2*z1/(w0*Te)
    s = s1*(2*b + c) - s2*b + e*(b+a+1) + e1*(-(2*b)-a) + e2*b
    return s*(Te**2)/(Te**2+(2*z1/w0)*Te+(1/w0**2))

for i in range(2, nbp):
    if X[-1] >= 0.2:
        X.append(X[-1])
    else:
        X.append(X[-1]+2*Te)
    Y.append(rec(Y[-1],X[-2],Y[-2],X[-3],X[-1]))

plt.plot(T,X)
plt.plot(T,Y)
plt.show(block= False)
```

Elaborer le filtre réjecteur

```
#include <TMC5160BOBLib.h>
#define CSpin  10
#define ENpin  25

TMC5160Controller m = TMC5160Controller(CSpin, ENpin);

SPISettings settingsA(3000000, MSBFIRST, SPI_MODE3);//Speed can be changed, theorically, with int. clock, up to 4MHz


uint32_t amax;
char c;
float VUSI = 14 * m.pi;//VUSI is the velocity target, in units of the international system; m.pi just stores the values of pi.
float vvusi = 0;
int32_t consvsigned;// Velocity target in units of tmc5160


//caractéristiques de la règle et du filtre
float z = 0.07;
float z1 = 1;
float w0 = 4.72;
float Te = 0.001; //s

//variables utilisées dans la fonction correction
float s = 0; float s1 = 0; float s2 = 0;
float e = 0; float e1 = 0; float e2 = 0;
float a = 0; float b = 0; float ce = 0; float d = 0;

float posmVUSI = 0.2; //position de consigne en m
float nouveau = 0;
float ancien = 0;
```

Elaborer le filtre réjecteur

```
void setup() {

  vvusi = VUSI;
  a = 2 * z / (w0 * Te); b = 1/(w0 * Te); b = b*b; ce = 2 * z1/(w0 * Te); d = b;

  //nécessaire pour le moteur

  Serial.begin(250000);
  while (!Serial);//needed for the teensy
  m.enable();//enable driver
  m.TMC5160_INTERNAL_CLOCK=12180000;// it is advised to you an external clock, internal clock (theorically 12MHz) is not precise, this
  // may determinate it with your own. 12000000 remains a good value if high precision is not required.
  m.setCLOCKSPEED(m.TMC5160_INTERNAL_CLOCK);//Set the clock, is only used for conversion USI to TMC5160 units.
  SPI.begin();
  m.setSPISettings(settingsA);//Call this first
  m.dontBeQuiet();//activate stealthchop
  m.setDefaultCHOPCONF();//see datasheet. The values are // TOFF=3, HSTRT=4, HEND=1, TBL=2, CHM=0 (spreadCycle™). Good values for most
  m.setStepRes(STEP256);//Step resolution. Other available are STEP128, STEP64, STEP32, STEP16, STEP8, STEP4, STEP2 and FULLSTEP
  m.setDefaultPWMCONF();//see datasheet. The values are TMC default reset (datasheet p. 47). Good values for most applications
  m.setIHOLD(1);//Hold current from 0 t0 31 (more than an amp in this case).
  m.setIRUN(31);//Running current. from 0 to 31.
  m.setIHOLDDELAY(5);//See datasheet
  m.setTPOWERDOWN(1);//See datasheet
  m.setTPWMTHRS(0); //time elapsed between 2 microsteps above witch stealthchop2 is activated. under, spreadcycle is activated. Set ze
  //warning : some current peaks can appear if sc2 is badly configured. Use sc2 for precise positionning at low speed, results are bet
  m.configRAMPMODE(VMODE_MINUS);// configuring the TMC5160 in velocity mode, negative value
  amax = 60000;//amax
  m.setAMAX(amax);//between 0 and 2^16-1 . A value out of limits is not written to the register
  m.setVMAX(0);//Speed target

}
```

Elaborer le filtre réjecteur

```
void loop() {
  //réinitialisation des variables
  vvusi = 0;
  s = 0; s1 = 0; s2 = 0;
  e = 0; e1 = 0; e2 = 0;
  ancien = 0;
  nouveau = 0;

  if (Serial.available()) {//open serial monitor, type anything
    m.enable();//enable the driver
    c = (char) Serial.read();//clean
    consvsigned = m.VUSITo5160(vvusi);//convert
    if (consvsigned >= 0) {
      m.configRAMPMODE(VMODE_PLUS);//in case of positive value
    }
    else {
      m.configRAMPMODE(VMODE_MINUS);//or negative
      consvsigned = -consvsigned;//then, always send positive
    }

    //correction
    float temps = 0;
    while (temps < 1500){
      correction();
      consvsigned = m.VUSITo5160(vvusi);
      if (consvsigned >= 0) {
        m.configRAMPMODE(VMODE_PLUS);//in case of positive value
      }
      else {
        m.configRAMPMODE(VMODE_MINUS);//or negative
        consvsigned = -consvsigned;//then, always send positive
      }
      m.setVMAX(consvsigned);//must be between 0 and 2^23-512.A value out of limits is not written to the register

      delay(Te*1000);//wait WAIT Te !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      temps = temps + Te*1000;//le temps est en ms
    }
    //stop : après un epsilon de la valeur finale
    Serial.println("STOP");//we want to stop the rotation now!
    m.setVMAX(0);//Stop it
    delay(1000);
    m.disable();//disable!
```

Elaborer le filtre réjecteur

```
void correction(){
  if (e >= posmVUSI){
    e2 = e1; e1 = e; /*e = posmmVUSI;*/
  }
  else {
    e2 = e1; e1 = e; e = e + 2*Te;
  }
  s = s1*(2*d + ce) - s2*d + e*(b + a + 1) - e1*(2*b + a) + e2*b;
  s = s/(1/((w0*w0)*(Te*Te)) + 2*z1/(w0*Te) +1);
  s2 = s1; s1 = s;
  ancien = nouveau;
  nouveau = s;
  vvusi = (nouveau - ancien)*175 /Te;  //175 est le rapport vitesse de rotation/ vitesse réelle
  Serial.println(vvusi);
}
```