Casque à détection de chute

Numéro de candidat : 37399

Plan et objecifs

Objectifs : - Construire un prototype de casque à détection de chute
- Collecter et analyser les données dynamiques d'un cycliste, afin de concevoir un système de détection performant

I. Présentation du prototype :

A) Exigences et présentation du système

B) Étude préliminaire des matériaux de fixation

C) Protocole et mesures

II. Analyse des mesures pour discriminer les chutes d'une course classique :

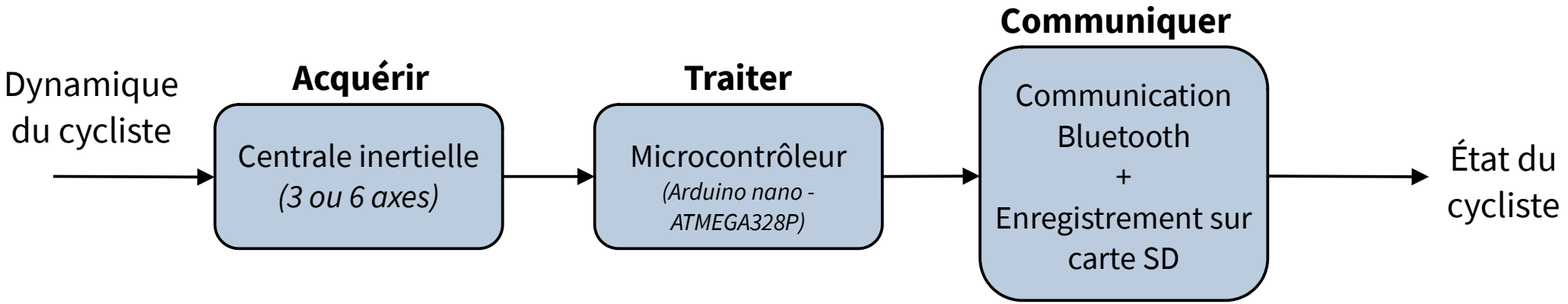
A) Étude des normes

B) Critère de blessure à la tête (HIC)

C) Indice de chute (Fall Index)

I.A) Description du système embarqué

Chaîne d'information du système :



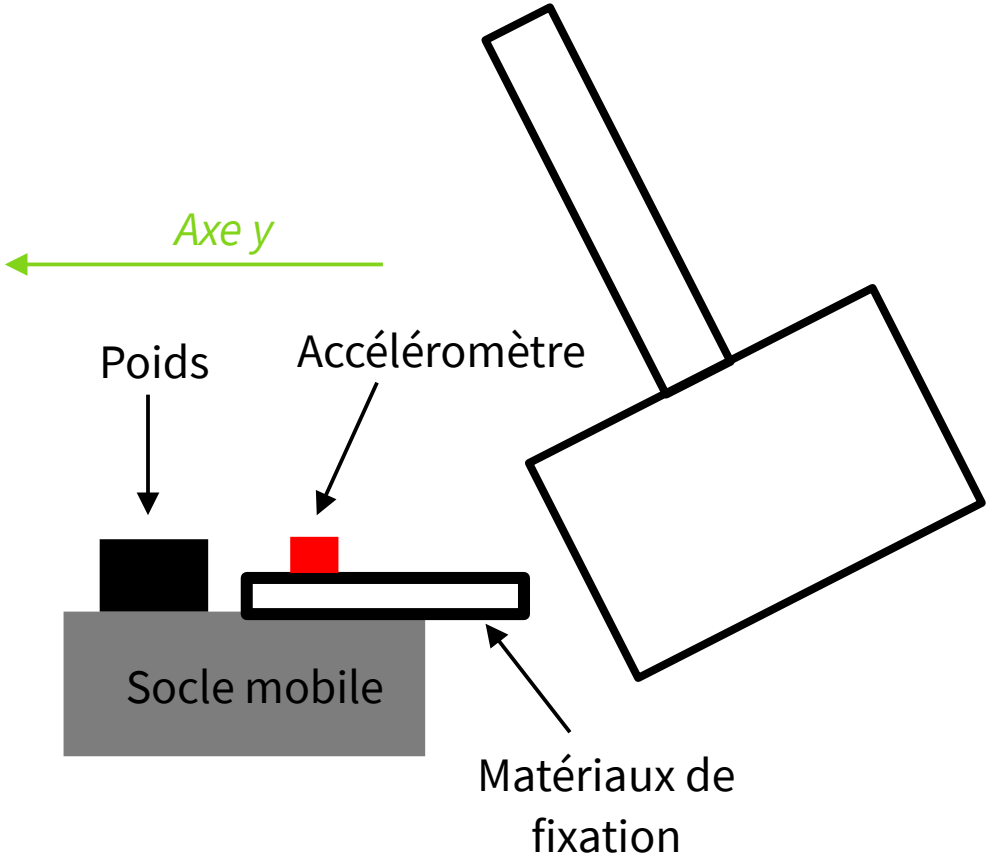
Cahier des charges :

- Système portable peu encombrant
- Acquisition correcte des données dynamiques du porteur
- Enregistrement des données sur un support physique pour traitement ultérieur
  (Pour ensuite remplacer ce système par une communication Bluetooth avec le téléphone de l'usager)
- Pouvoir déterminer l'état du cycliste en fonction des données récoltées

I.A) Prototype du système embarqué

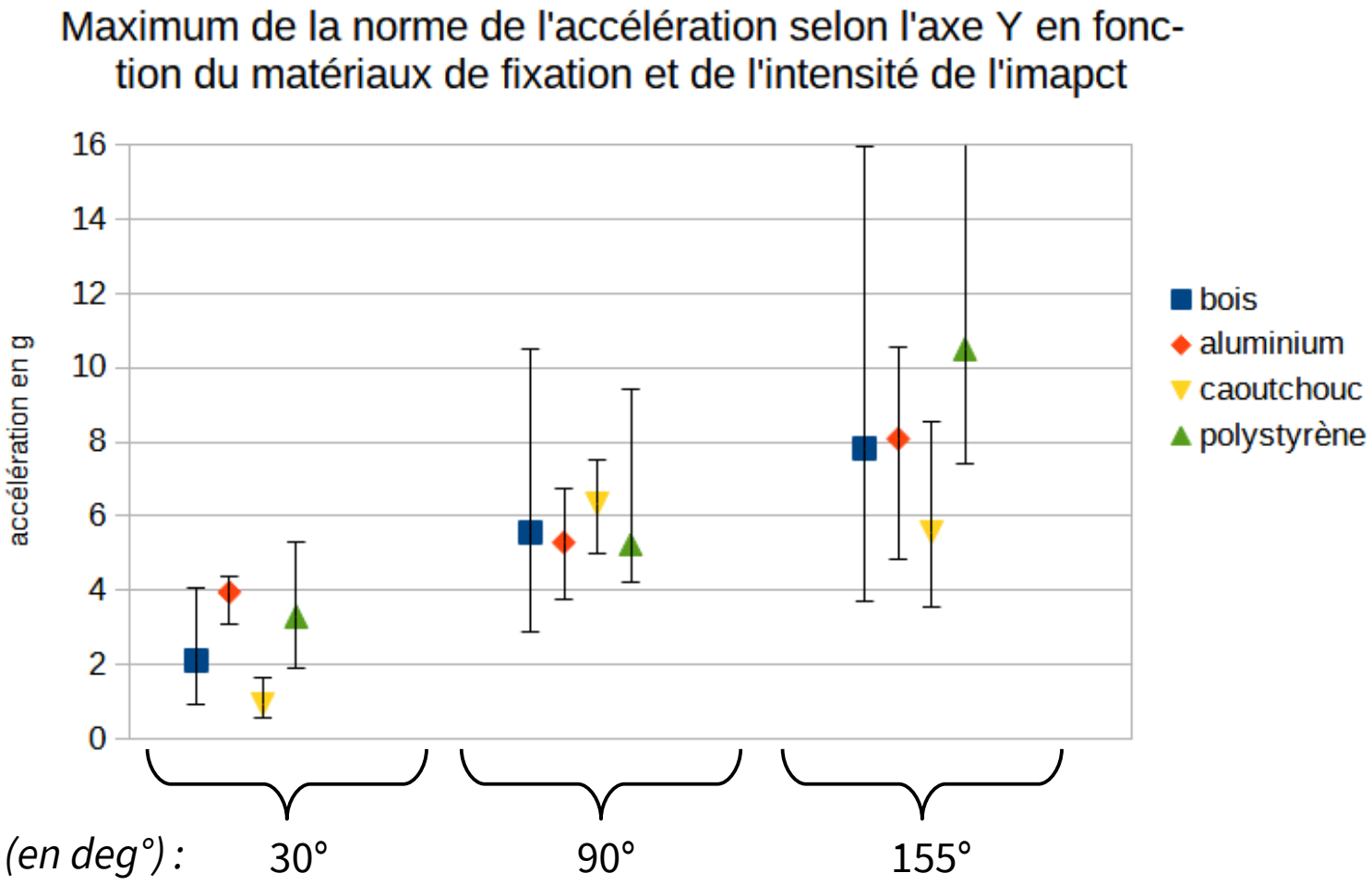
I.B) Impact des matériaux sur les mesures

Protocole expérimental :



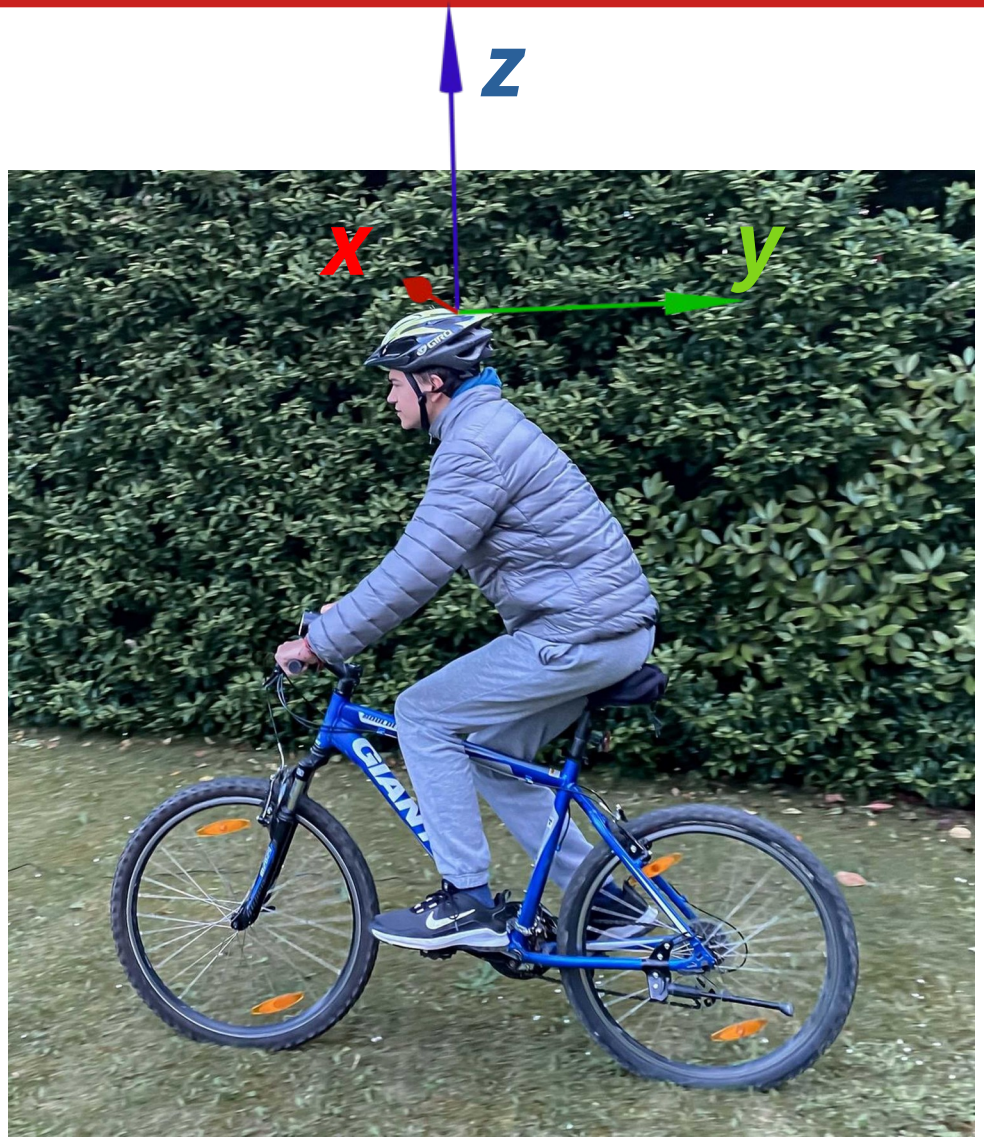
Montage expérimental :

I.B) Impact des matériaux sur les mesures



3 mesures par matériaux et par profil d'accélération (36 mesures en tout)

I.C) Protocole de mesure

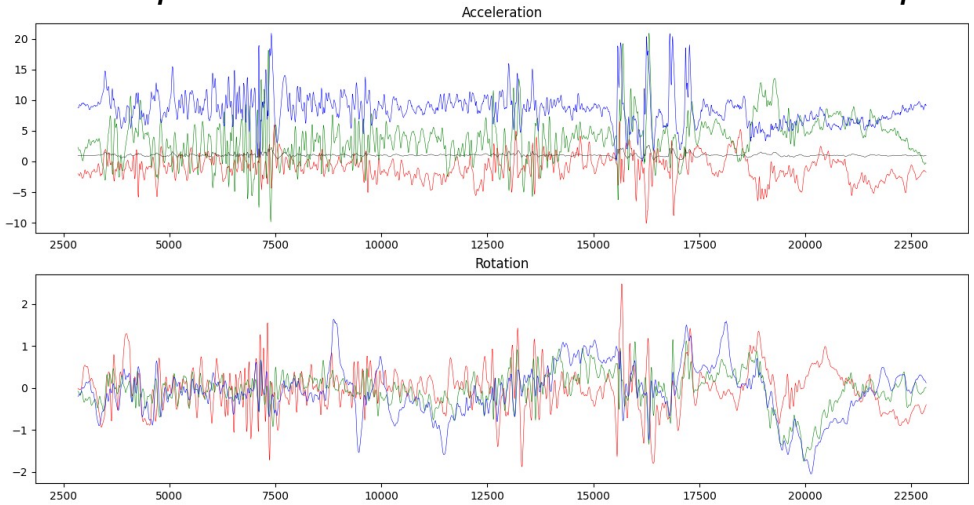


- **30 essais** sans chute :
  - → Durée de **20 secondes**
  - → Période d'échantillonnage de **7ms**
  - → **+85.000** points de donnée

- **10 essais** de chute :
  - → Période d'échantillonnage de **7ms**

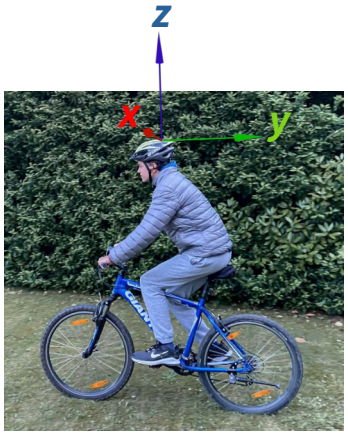
I.C) Protocole de mesure

Exemple de mesure lors d'une course classique



temps (ms)

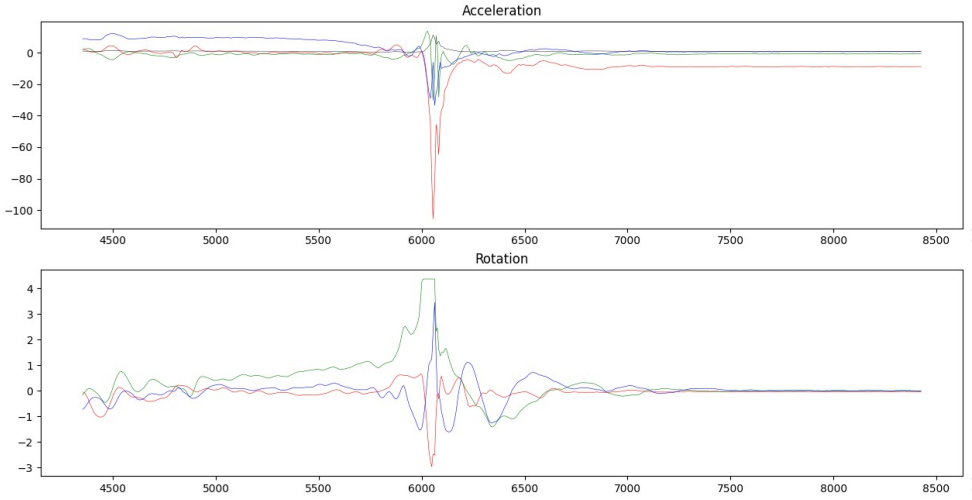
temps (ms)



Exemple de mesure lors d'une chute sur l'axe X

Accélération (en m/s^2)

Vitesse de rotation (en rad/s)



temps (ms)

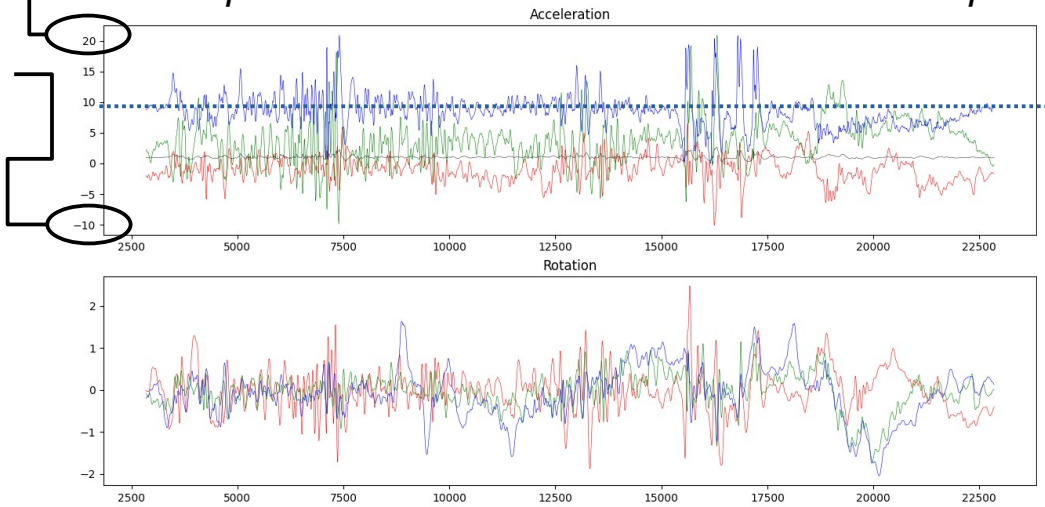
temps (ms)

I.C) Protocole de mesure

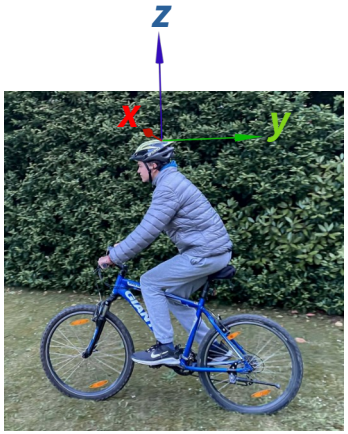
Exemple de mesure lors d'une course classique

20 m/s²

-10 m/s²

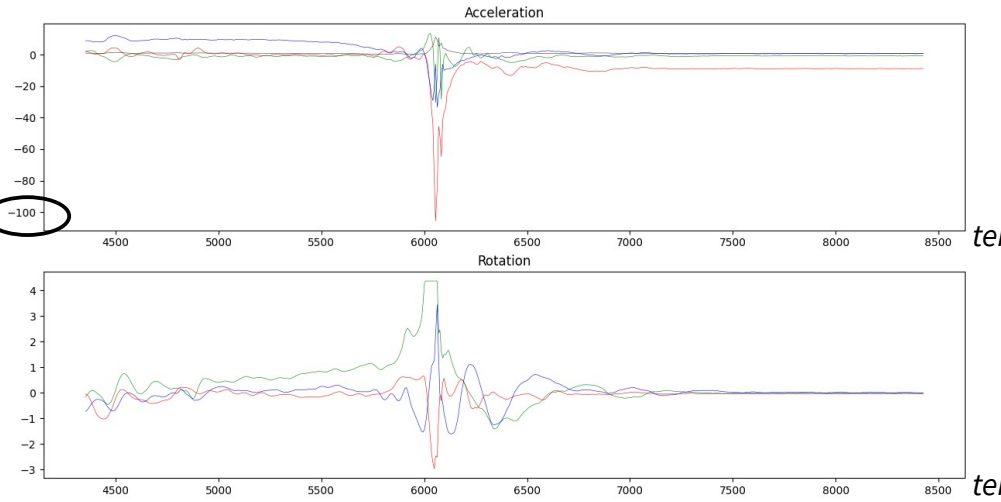


Accélération de la pesanteur



Exemple de mesure lors d'une chute sur l'axe X

-100 m/s²



Accélération (en m/s²)

Vitesse de rotation (en rad/s)

II.A) Détection d'une chute par étude des normes

Norme du vecteur accélération :

$$A_i = \sqrt{A_{x_i}^2 + A_{y_i}^2 + A_{z_i}^2}$$

```
def norme(x,y,z):
    return np.sqrt(x**2+y**2+z**2)
```

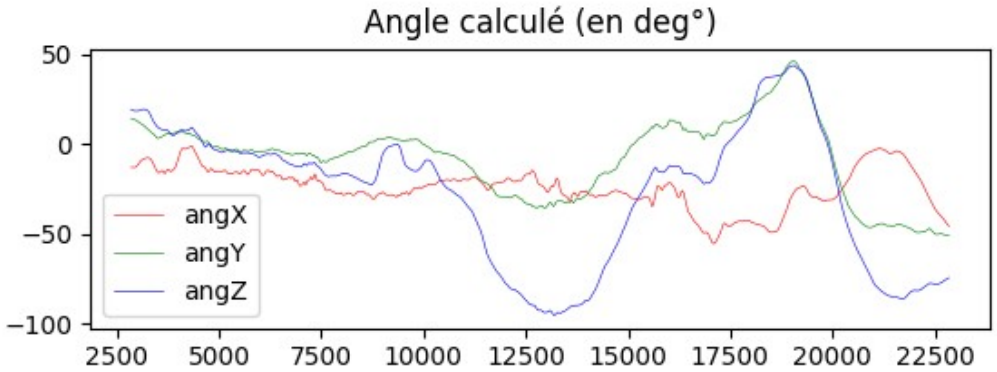
Projection du vecteur accélération sur la verticale :

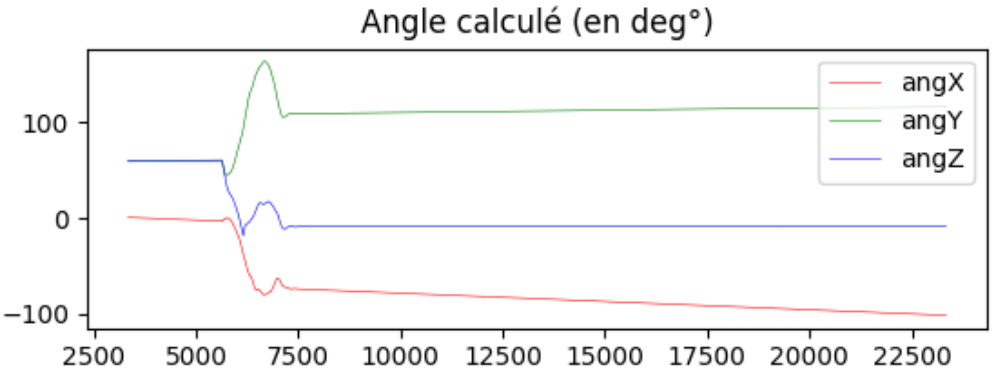
$$A_v = \left| A_x \sin(\theta_x) + A_y \sin(\theta_y) + A_z \cos(\theta_y) \cos(\theta_z) \right|$$

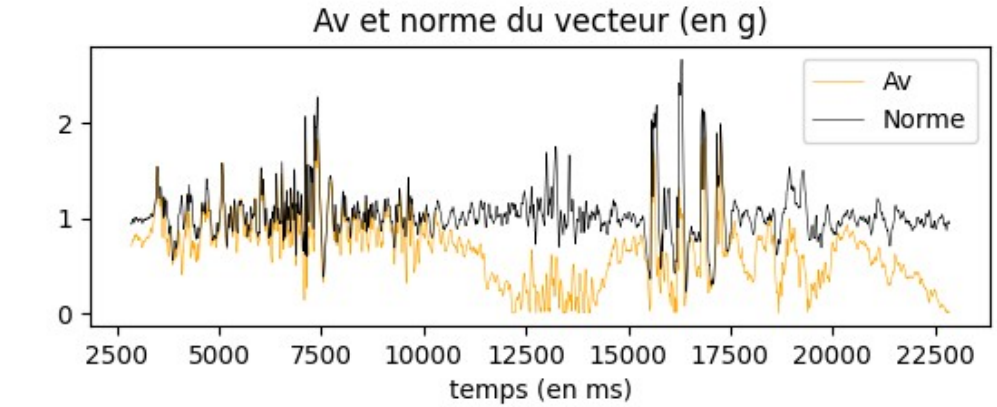
```
def rot(ax,ay,az,gx,gy,gz,time):
    #calcul des angles (en rad) initiaux grace a l acceleration initiale
    arx = np.arctan(ax[0]/(np.sqrt((ay[0]**2+az[0]**2))))
    ary = np.arctan(ay[0]/(np.sqrt((ax[0]**2+az[0]**2))))
    arz = np.arctan((np.sqrt(ay[0]**2+ax[0]**2))/az[0])
    #initialisation matrices angles
    angle_x = [arx]
    angle_y = [ary]
    angle_z = [arz]
    #calcul des angles par integration grace a vitesse rotation
    for i in range(len(time)-1):
        dt = (time[i+1]-time[i])*0.001
        angle_x.append(angle_x[i]+gx[i]*dt)
        angle_y.append(angle_y[i]+gy[i]*dt)
        angle_z.append(angle_z[i]+gz[i]*dt)
    return(np.array(angle_x),np.array(angle_y),np.array(angle_z))

def vertical_amp(ax,ay,az,rx,ry,rz):
    Av = np.abs(ax*np.sin(rx) + ay*np.sin(ry) - az*np.cos(ry)*np.cos(rz))
    return Av
```

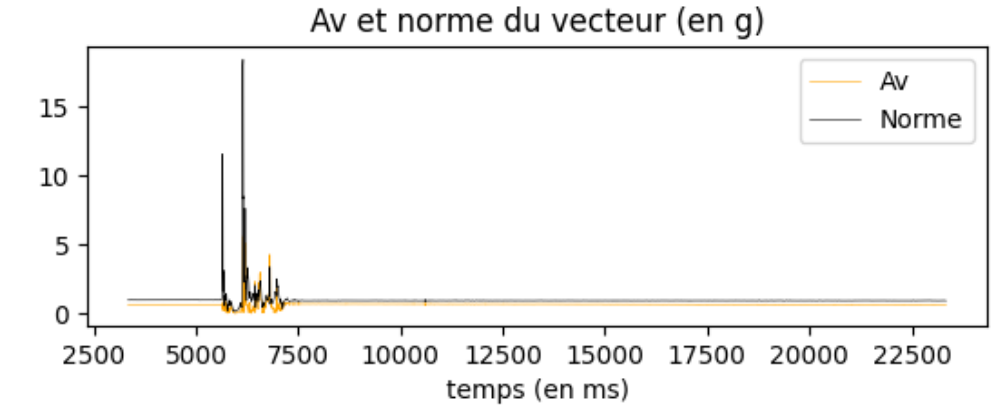
II.A) Détection d'une chute par étude des normes

Evolution des angles, de la norme du vecteur accélération et de sa projection sur la verticale en fonction du temps



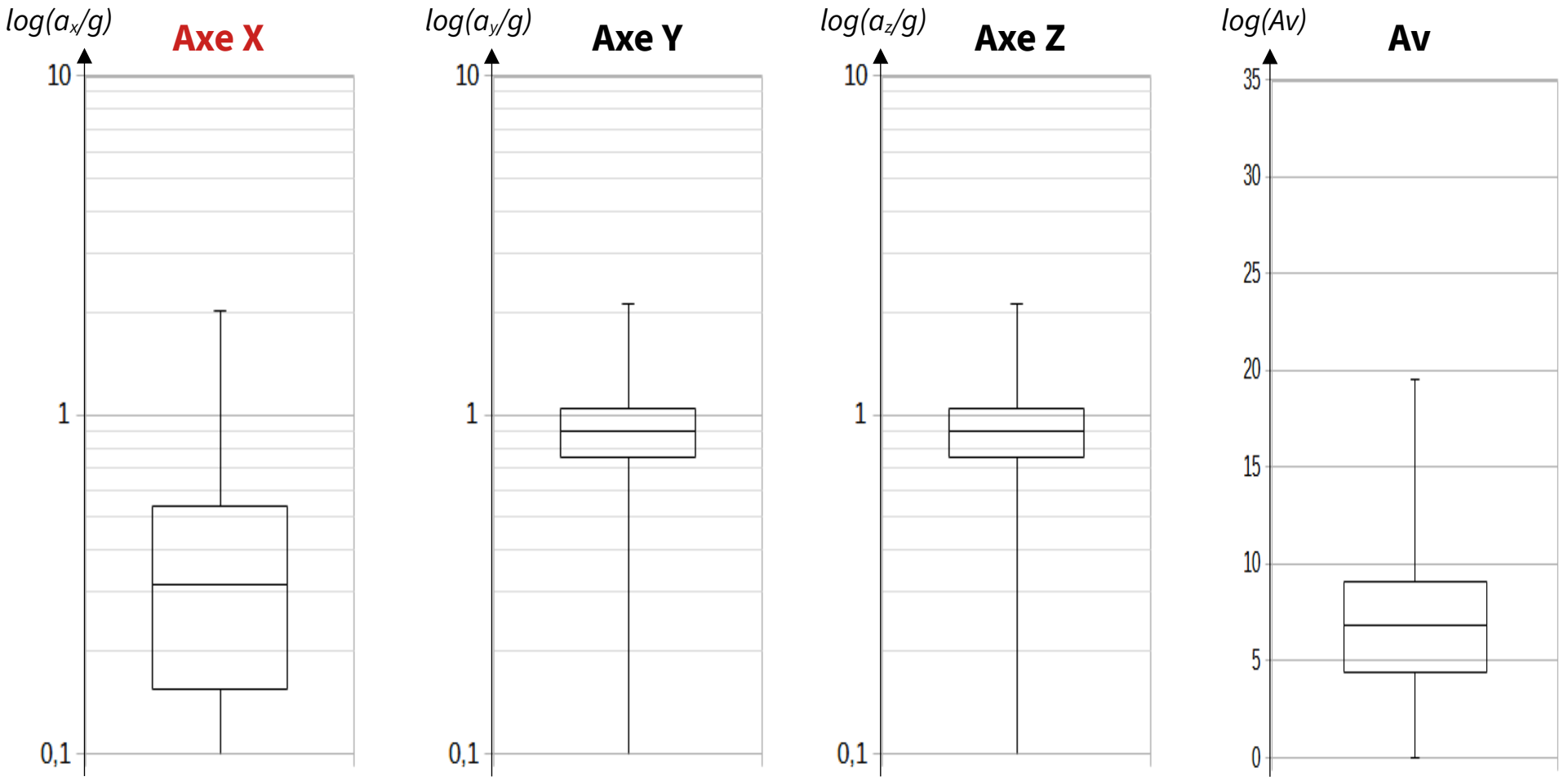






II.A) Détection d'une chute par étude des normes

Plage de valeur moyenne de la norme de l'accélération (en g) selon chaque axe

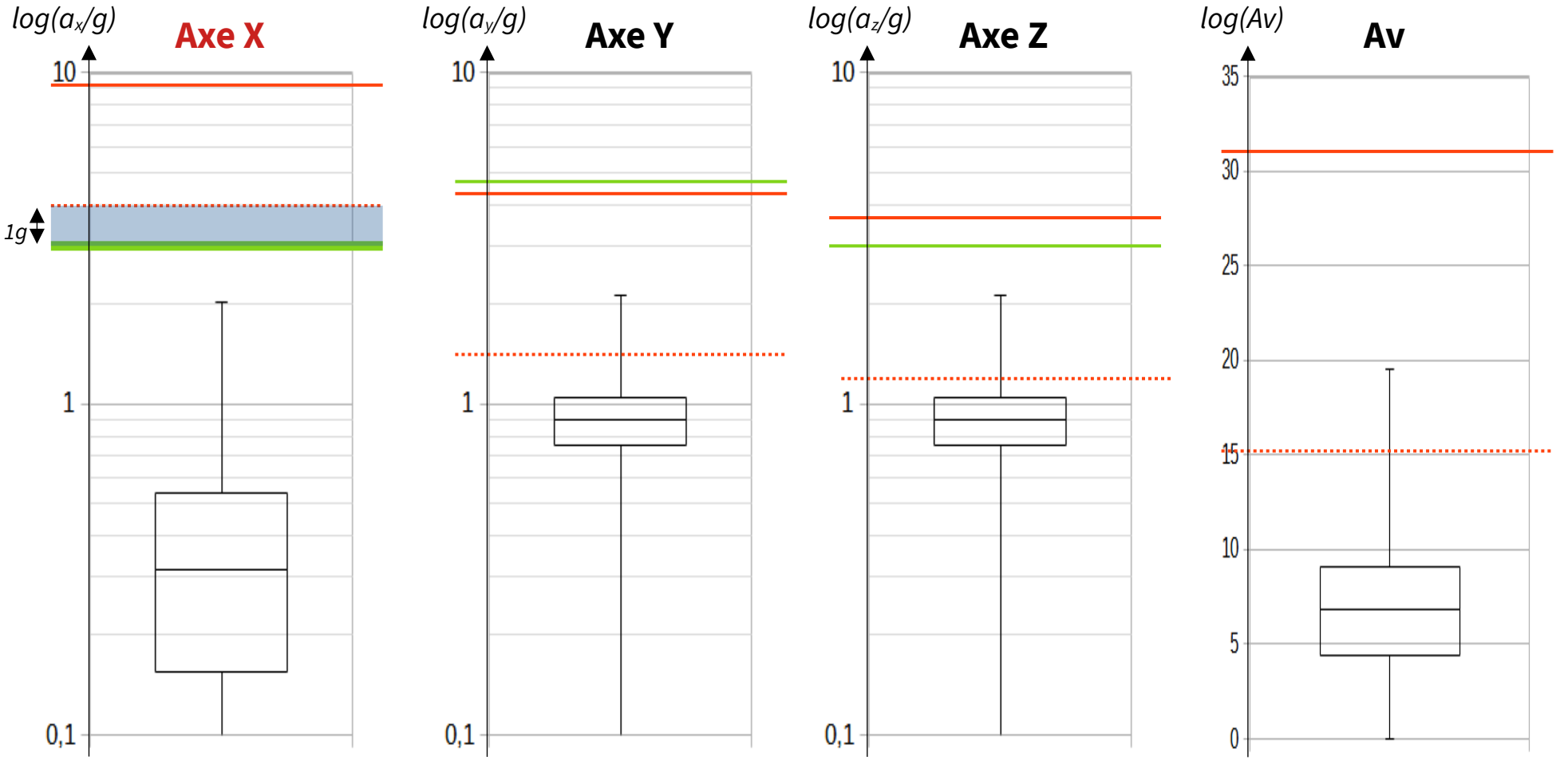


II.A) Détection d'une chute par étude des normes

Valeur moyenne des maxima lors de chutes

Plus petite valeur de ces maxima

Valeur maximale atteinte hors chutes



II.B) Étude du HIC

- Formule :

$$HIC = (t_2 - t_1) \cdot \left[\frac{1}{t_2 - t_1} \cdot \int_{t_1}^{t_2} a(t)\,dt \right]^{2,5}$$

- Algorithme:

```
def HIC(acc, time, period) :
    #acc : donnees |time : instants| period : 10-36 ms
    HIC = []
    t_dt = []   #abscisse des valeurs
    time_p = time[0]    #temps initial
    index = []

    #construction des indices des valeurs en fonction
    #de la valeur de period
    for i in range(len(acc)):
        if (time[i] - time_p)>=period :
            index.append(i)
            t_dt.append(time[i])
            time_p = time[i]
    #caclul des valeurs du HIC
    #pour les indices précédents
    for i in range(len(index)-1):
        x = np.array(acc[index[i]:index[i+1]])/g
        y = np.array(time[index[i]:index[i+1]])
        a_int = scipy.integrate.simps(x,y)
        delta = time[index[i+1]] - time[index[i]]
        HIC.append(delta*((a_int/delta)**2.5))

    HIC.append(0)
    return np.array(HIC),np.array(t_dt)
```

II.B) Étude du HIC

Origine :

- Développé entre les années 50 et 60
- Permet de déterminer une valeur maximale à ne pas dépasser, sous risque de blessure
- Utilisée depuis dans de nombreux domaines tels que l'automobile pour la conception d'amortisseurs

Interprétation physique :

$$HIC = (t_2 - t_1) \cdot \left[\frac{1}{t_2 - t_1} \cdot \int_{t_1}^{t_2} a(t)\,dt \right]^{2,5}$$

En posant :

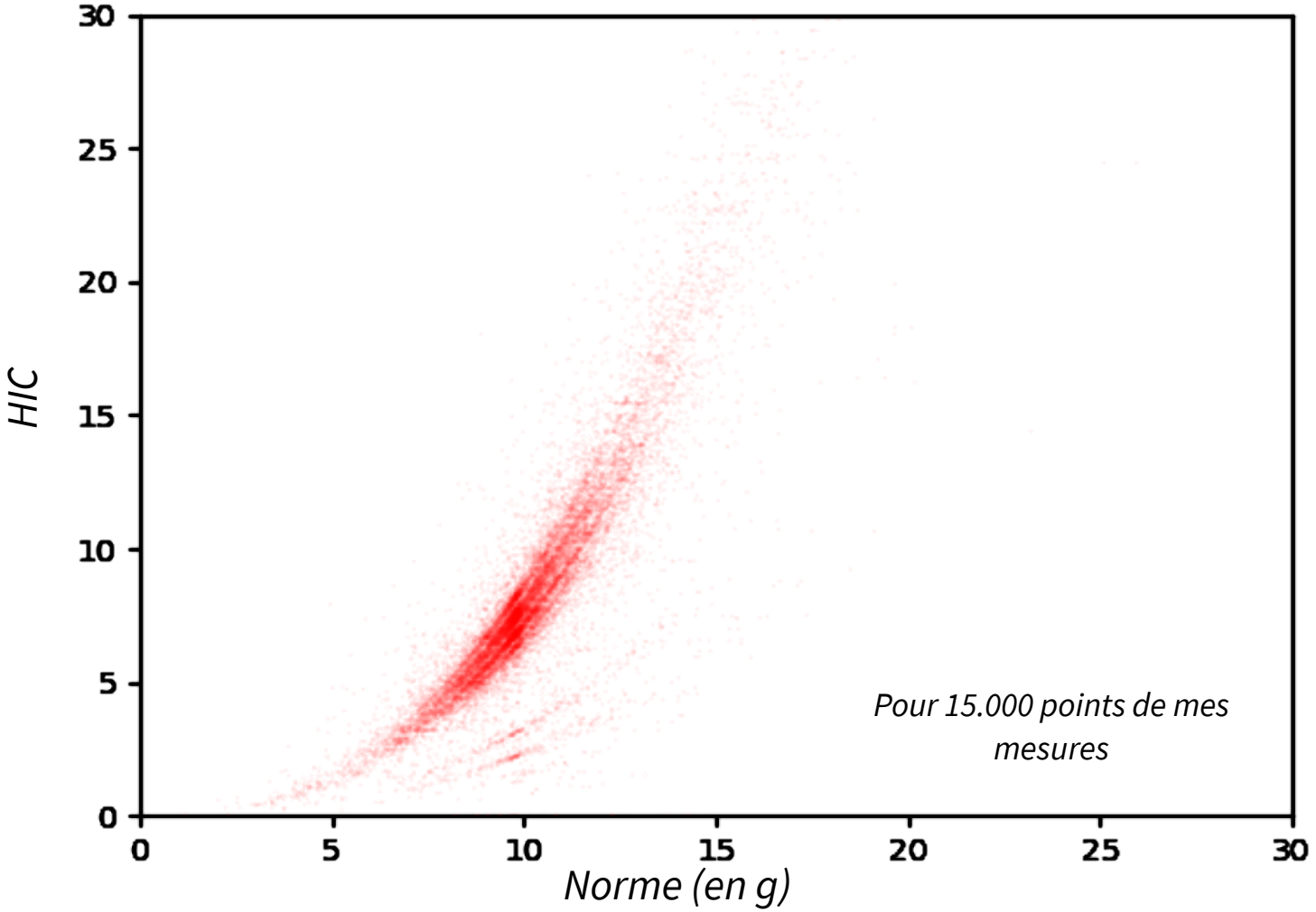
$$V = \int_{t_1}^{t_2} a(t)\,dt$$

$$\tau = t_2 - t_1$$

On obtient : $$HIC = \left(\frac{V}{\tau}\right)^{0,5} \frac{V^2}{\tau}$$

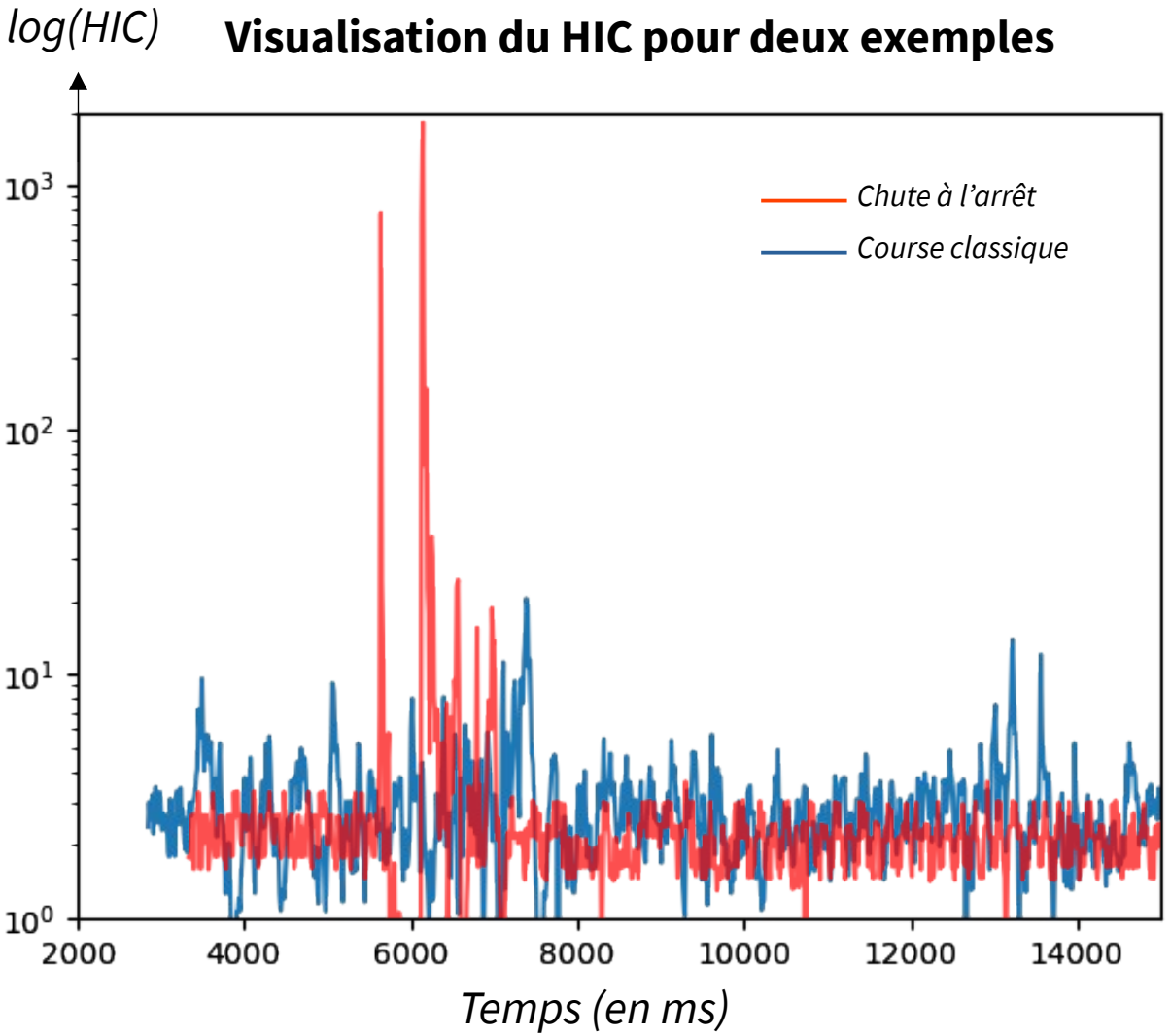
II.B) Étude du HIC

Valeur du HIC en fonction de la norme de l'accélération (en m/s²) pour un même instant



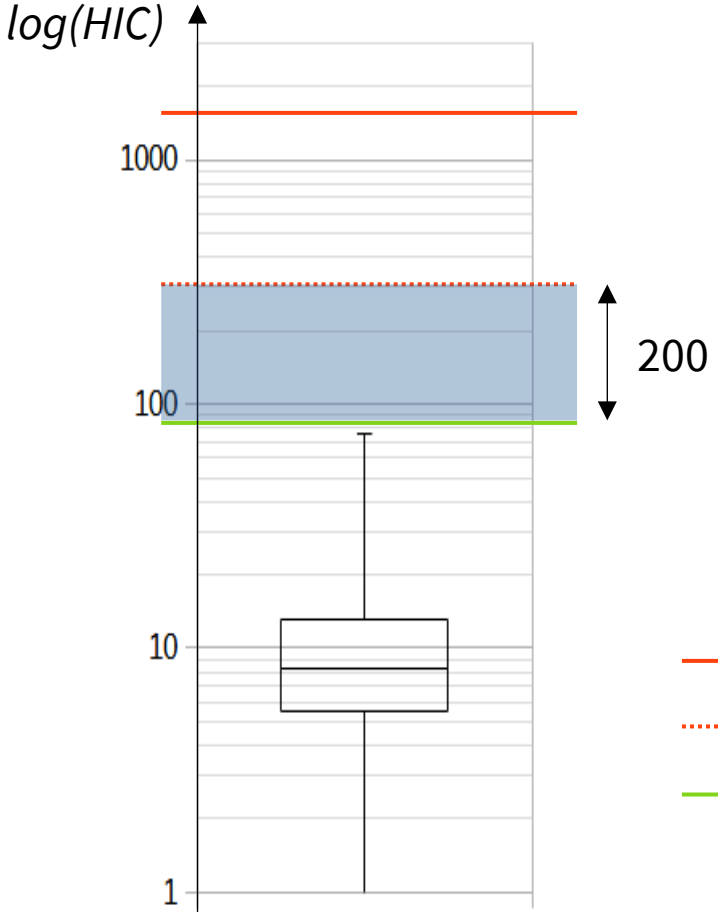
II.B) Étude du HIC

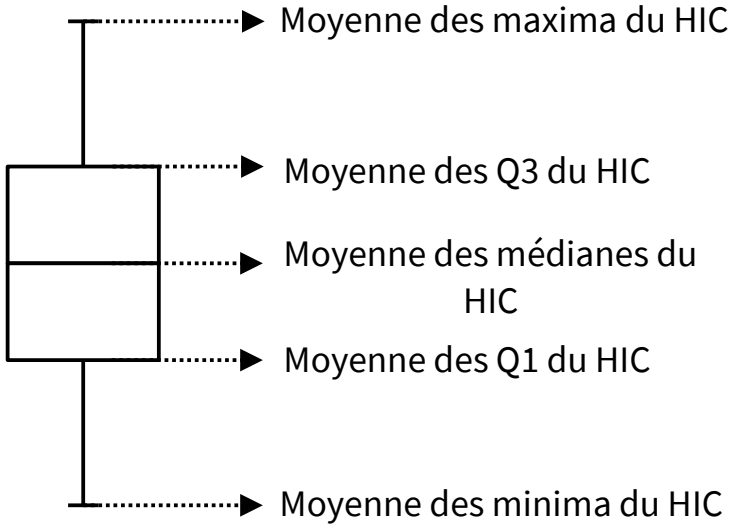
Visualisation du HIC pour deux exemples



II.B) Étude du HIC

Moyenne des plages de valeur du HIC en dehors des chutes





Valeur moyenne des maxima lors de chutes

Plus petite valeur de ces maxima

Valeur maximale atteinte hors chutes

II.C) Étude du Fall Index

- Formule :

$$FI_i = \sqrt{\sum_{k=x,y,z} \sum_{i-19}^{i} \left((A_k)_i - (A_k)_{i-1} \right)^2}$$

- Algorithme:

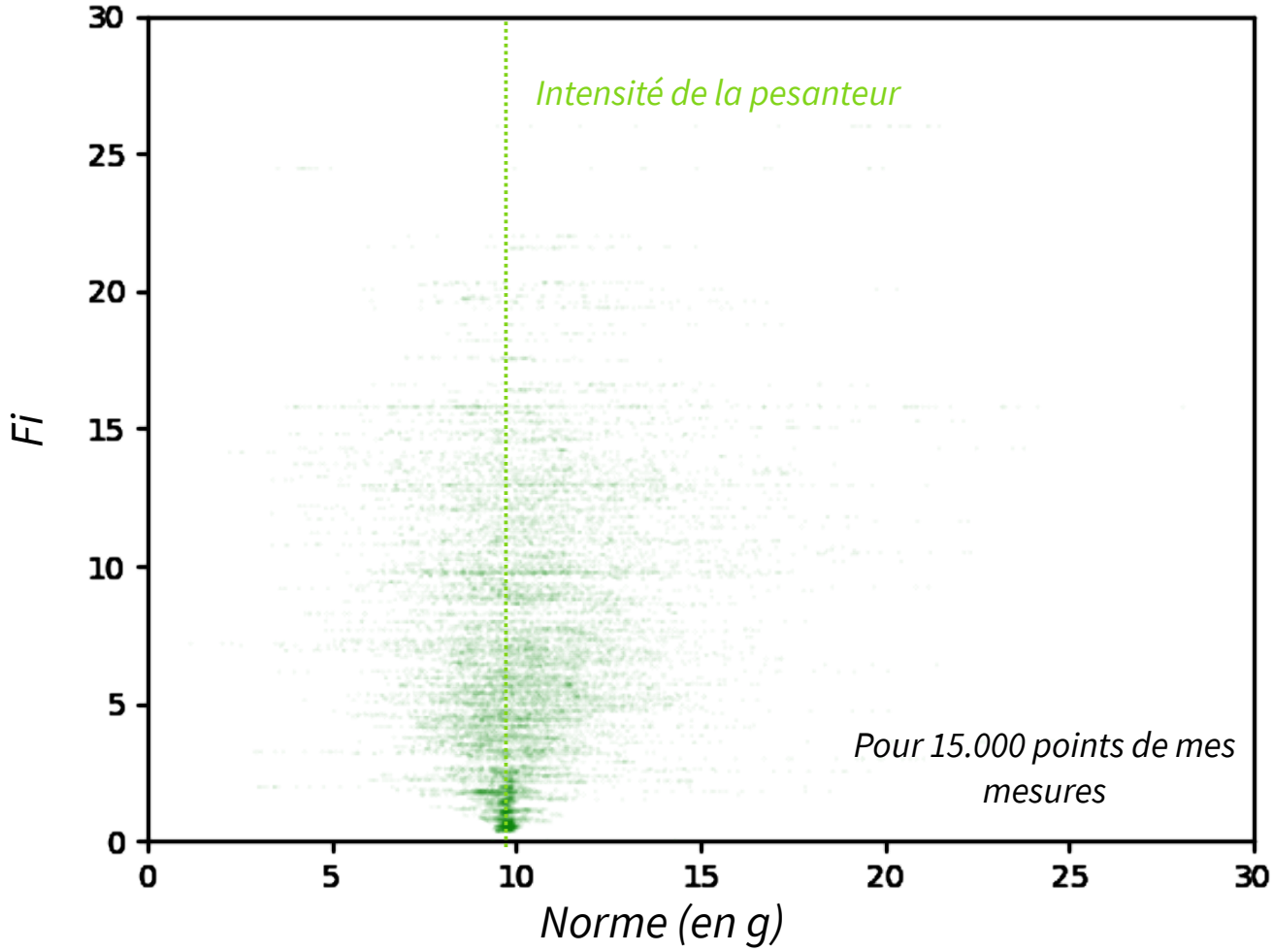
```
def Fall_Index(ax,ay,az,time):
    #calcul terme interne
    def a(x):
        a = 0
        for i in range(19):
            a += (x[i]-x[i-1])**2
        return a

    Fi = []
    tempo = []
    index = []
    #calcul total du Fi
    for j in range(len(time)//20):
        rax = a(ax[j*20:(j+1)*20])
        ray = a(ay[j*20:(j+1)*20])
        raz = a(az[j*20:(j+1)*20])
        Fi.append(np.sqrt(rax+ray+raz))
        index.append(j*19)
        tempo.append(time[j*19])

    return np.array(Fi),np.array(tempo),index
```

- Origine :

Formule empirique développée au début des années 2000, dans le but de détecter les chutes

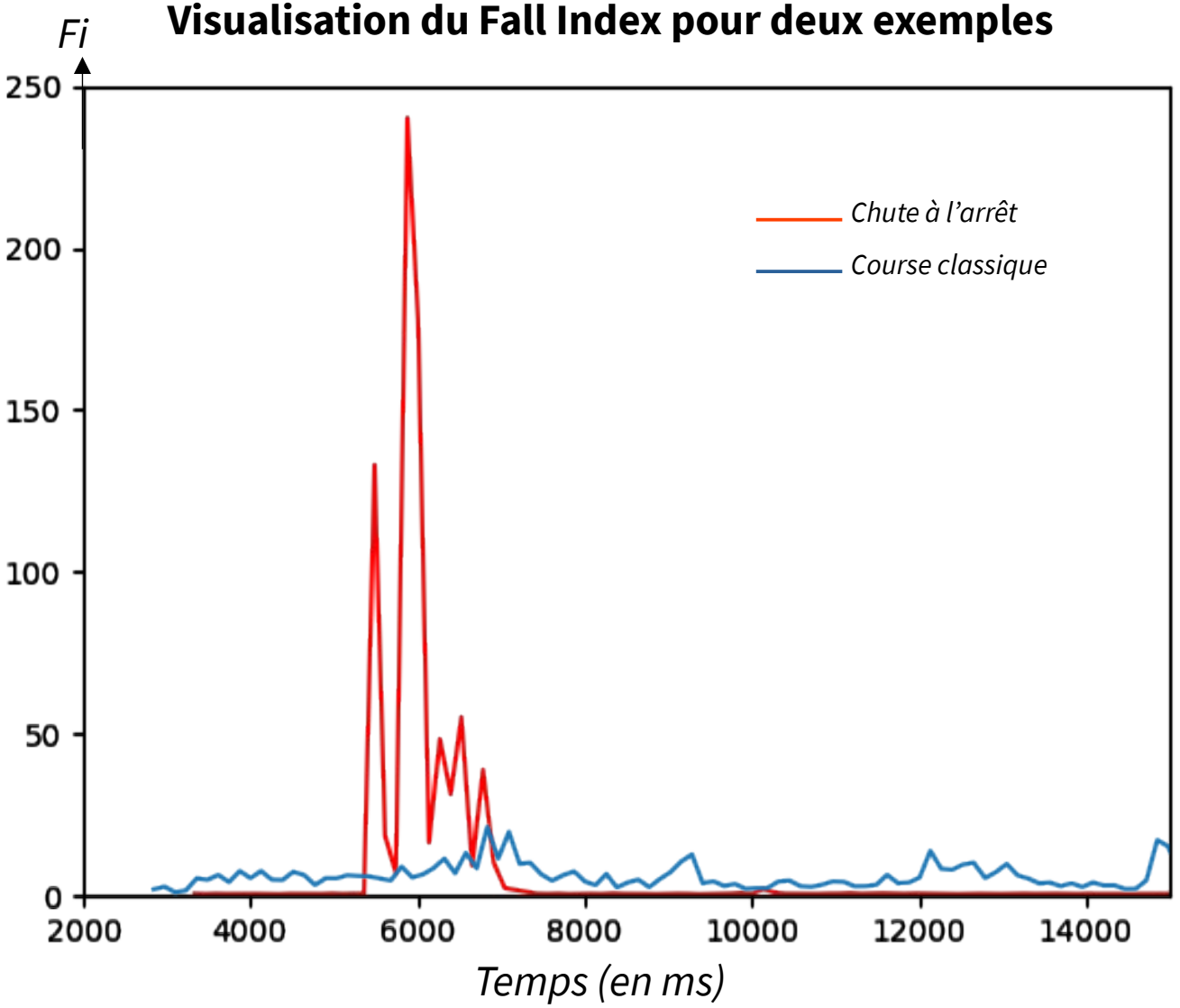
II.C) Étude du Fall Index

Valeur du Fi en fonction de la norme de L'accélération (en m/s²) pour un même instant



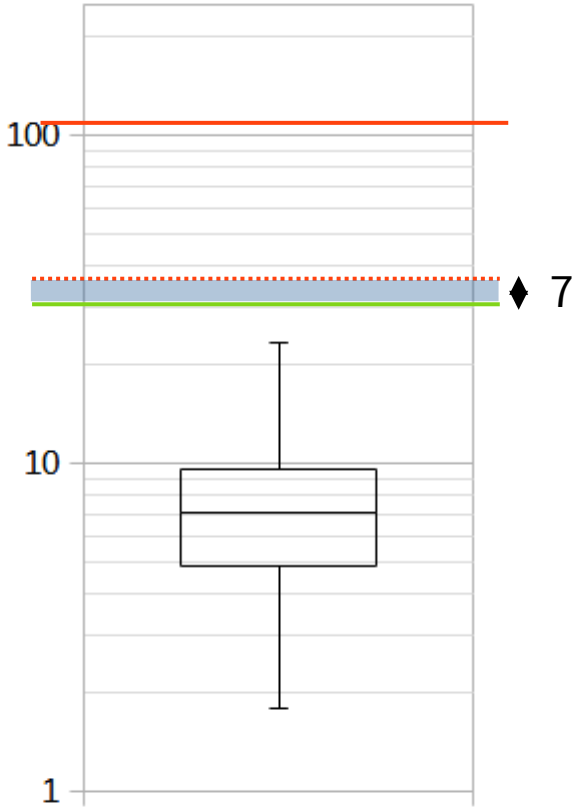
II.C) Étude du Fall Index

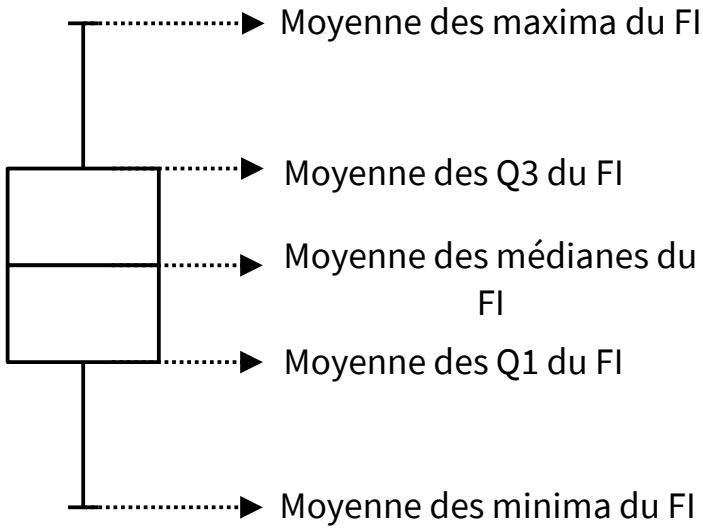
Visualisation du Fall Index pour deux exemples



II.C) Étude du Fall Index

Plage de valeurs du Fall Index pour un trajet moyen





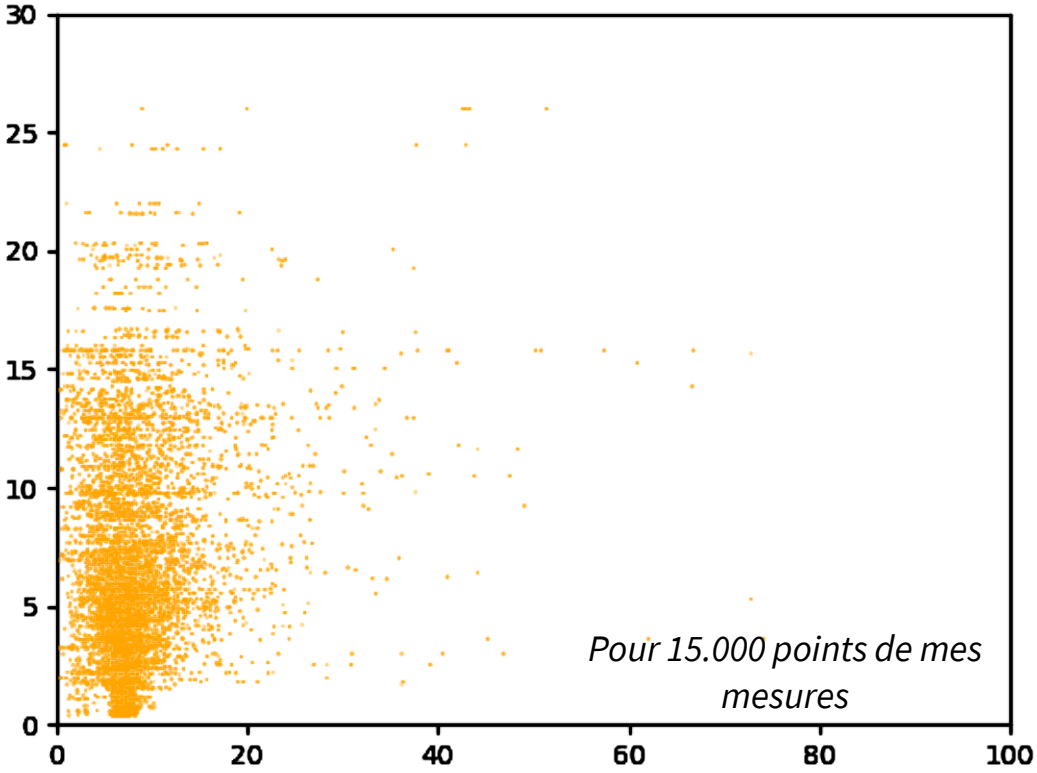
Valeur moyenne des maximum lors de chutes

Plus petite valeur de ces maximum

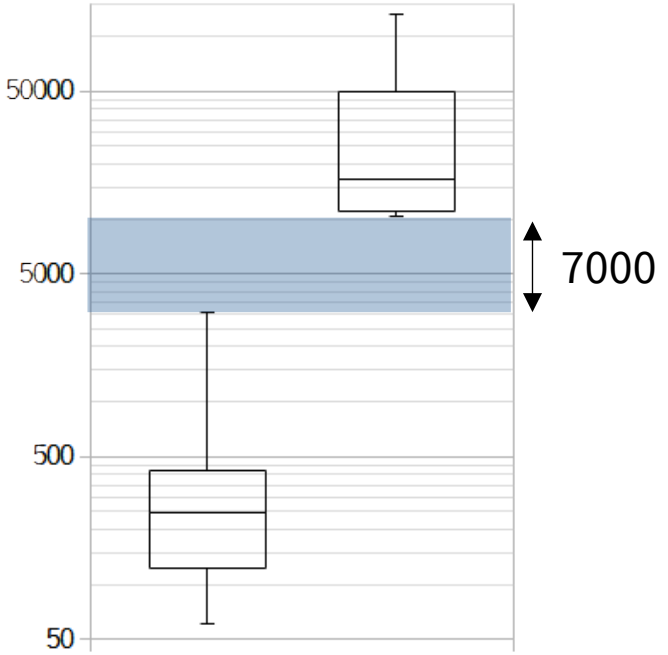
Valeur maximale atteinte hors chutes

Combinaison du HIC et du Fall Index

Valeur du Fall Index en fonction de celle du HIC

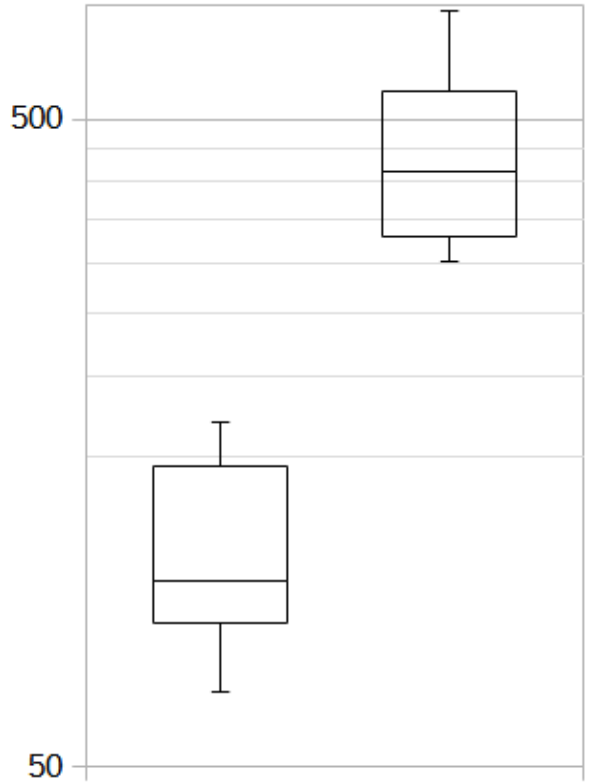


Plage de Valeurs du produit du HIC et du FI



Analyse dans des cas réels

Plage de Valeurs du produit du HIC et du FI pour une moto en conditions réelles



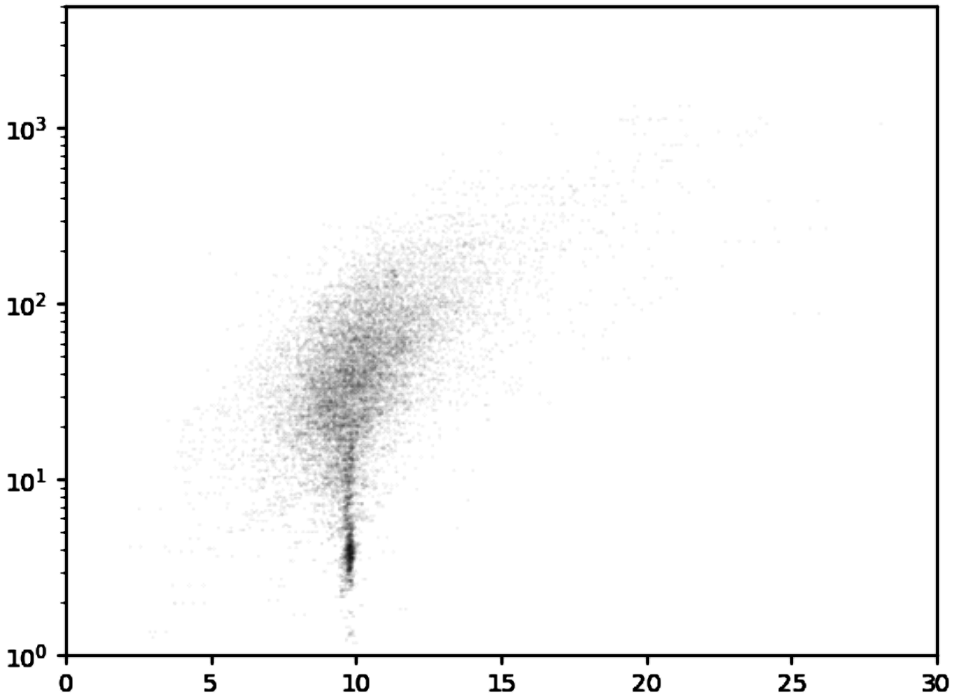
Conclusion et ouvertures

Futures améliorations :

- Mise en place de système de communication Bluetooth entre le casque et le smartphone de l'utilisateur
- Application smartphone
- Utilisation d'algorithmes de type MVS (Machines a vecteur de support)
- Utilisation d'un magnétomètre pour de meilleures mesures des angles

ANNEXE

Valeur du produit en fonction de la norme



Calcul du produit du HIC et du Fi

```
Fi_norm = []
HIC_norm = []
i_h = 0
i_f = 0

for i in range(len(time)):
    #on transfrome ici le HIC et Fi en
    #des fonctions continues par morceaux
    #afin de pouvoir facilement
    #calculer leur produit a chaque
    #instant
    if i in ind_HIC[1:] :
        HIC_norm.append(HIC[i_h])
        i_h +=1
    else :
        HIC_norm.append(HIC[i_h])

    if i in ind_Fi[1:] :
        Fi_norm.append(Fi[i_f])
        i_f += 1
    else :
        Fi_norm.append(Fi[i_f])
del HIC_norm[0]
HIC_norm.append(0)
del Fi_norm[0]
Fi_norm.append(0)

HIC_norm = np.array(HIC_norm)
Fi_norm = np.array(Fi_norm)

prod = HIC_norm*Fi_norm
```

Programmes annexes

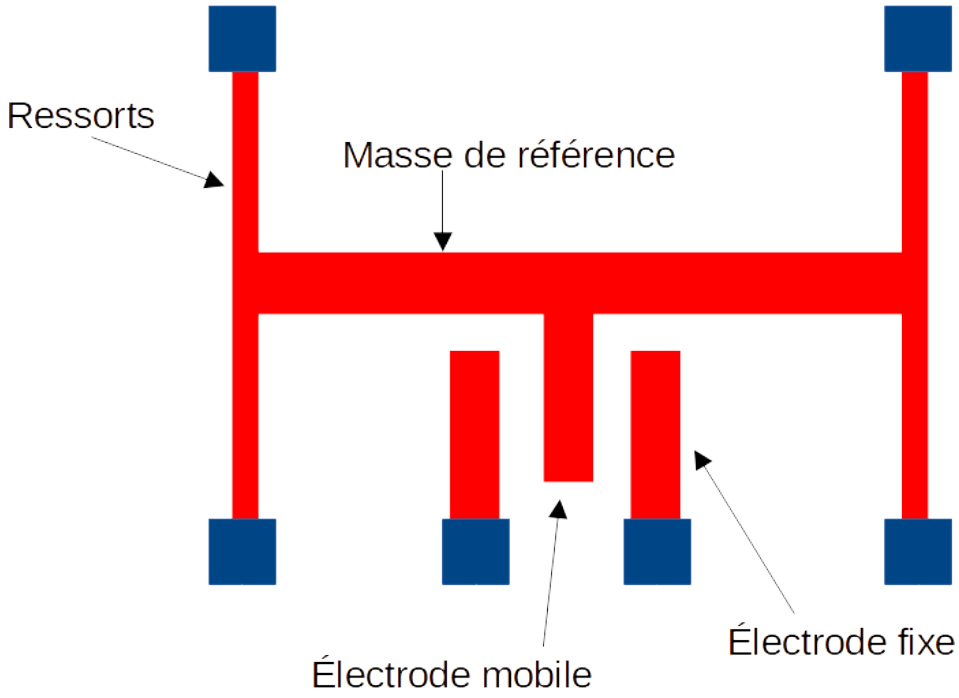
Initialisation du programme

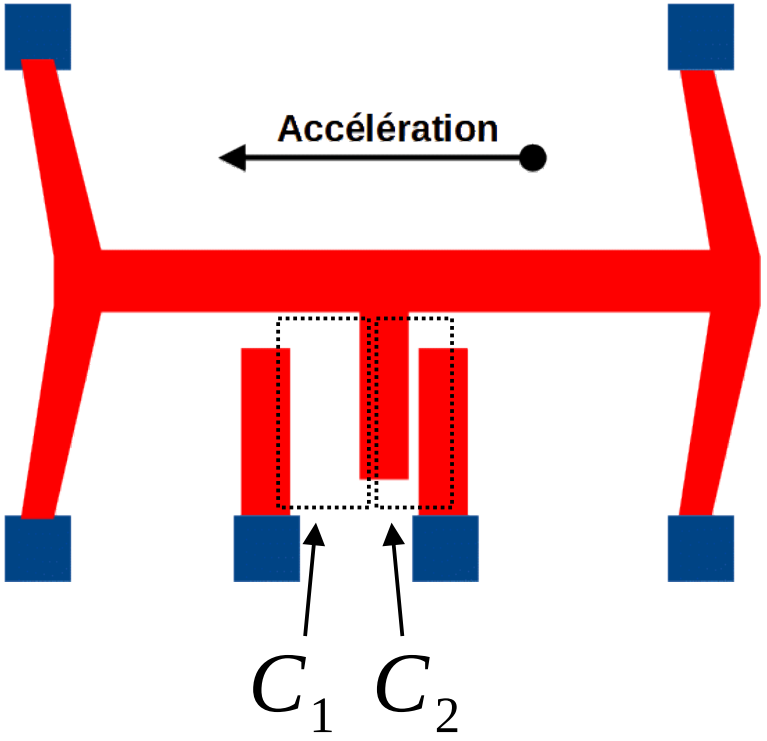
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.integrate
from scipy.stats import pearsonr

g = 9.81

data = open('C:\\                    /TIPE/mesure_TIPE/Data18.txt')

plt.clf()
lignes = []

for ligne in data :
    cl_ligne = ligne.split('|')
    f_ligne = [float(x) for x in cl_ligne]
    lignes.append(f_ligne)

table = np.array(lignes)

time = table[:,0]#ms
ax = table[:,1]#m/s2
ay = table[:,2]
az = table[:,3]
gx = table[:,4]#radians
gy = table[:,5]
gz = table[:,6]

temp = table[:,7]#degre
```

Affichage

```
def values(X, name):
    print(name + " min :",round(np.min(Av),2))
    print(name + " Q1 :", round(np.quantile(Av,0.25),2))
    print(name + " med :",round(np.median(Av),2))
    print(name + " Q2 :", round(np.quantile(Av,0.75),2))
    print(name + " max :",round(np.max(Av),2))
    print(name + " moy :", round(np.mean(Av),2))
    print("")

values(HIC, 'HIC')
values(Fi, 'Fi')

fig,axs = plt.subplots(3)

axs[0].set_title("Acceleration lineaire")
axs[0].plot(time,ax,'red',linewidth=0.4)
axs[0].plot(time,ay,'green',linewidth=0.4)
axs[0].plot(time,az,'blue',linewidth=0.4)
axs[0].plot(time,norme(ax,ay,az)/g,'black',linewidth = 0.3)

axs[1].set_title("Vitesse de rotation")
axs[1].plot(time,gx,'red',linewidth=0.4)
axs[1].plot(time,gy,'green',linewidth=0.4)
axs[1].plot(time,gz,'blue',linewidth=0.4)

axs[2].set_title("Indices")
axs[2].plot(time, HIC_norm,'orange')
axs[2].plot(time,Fi_norm,'brown')

plt.show()
```

Fonctionnement de l'accéléromètre





ε : *Permittivité*

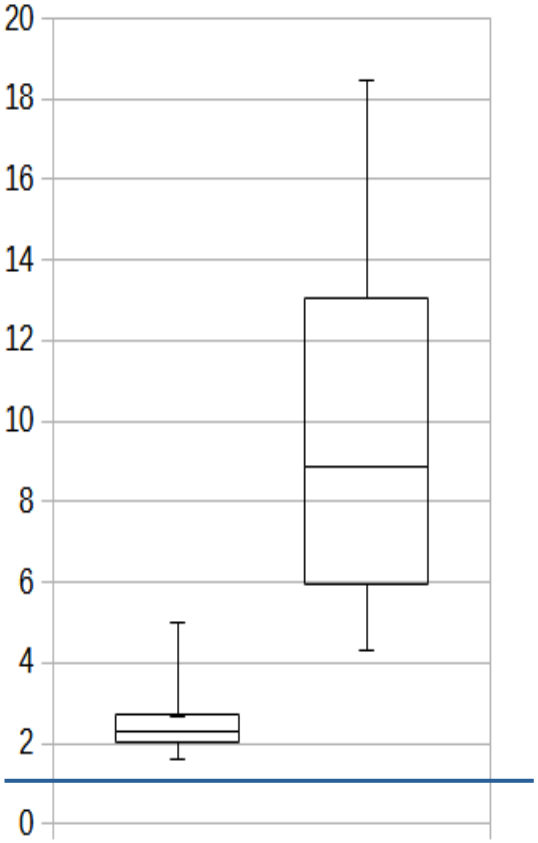
S : *Surface des élctrodes en correspondance*

e : *Distance entre les plaques*

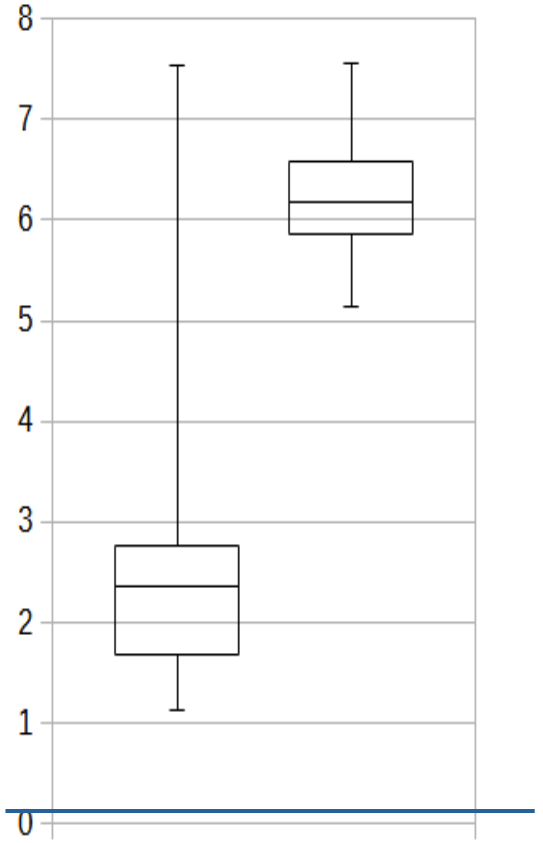
$$C \simeq \frac{\varepsilon S}{e} \Rightarrow C_1 < C_2$$

Seuil

Maximum de la norme de l'accélération linéaire (en g)

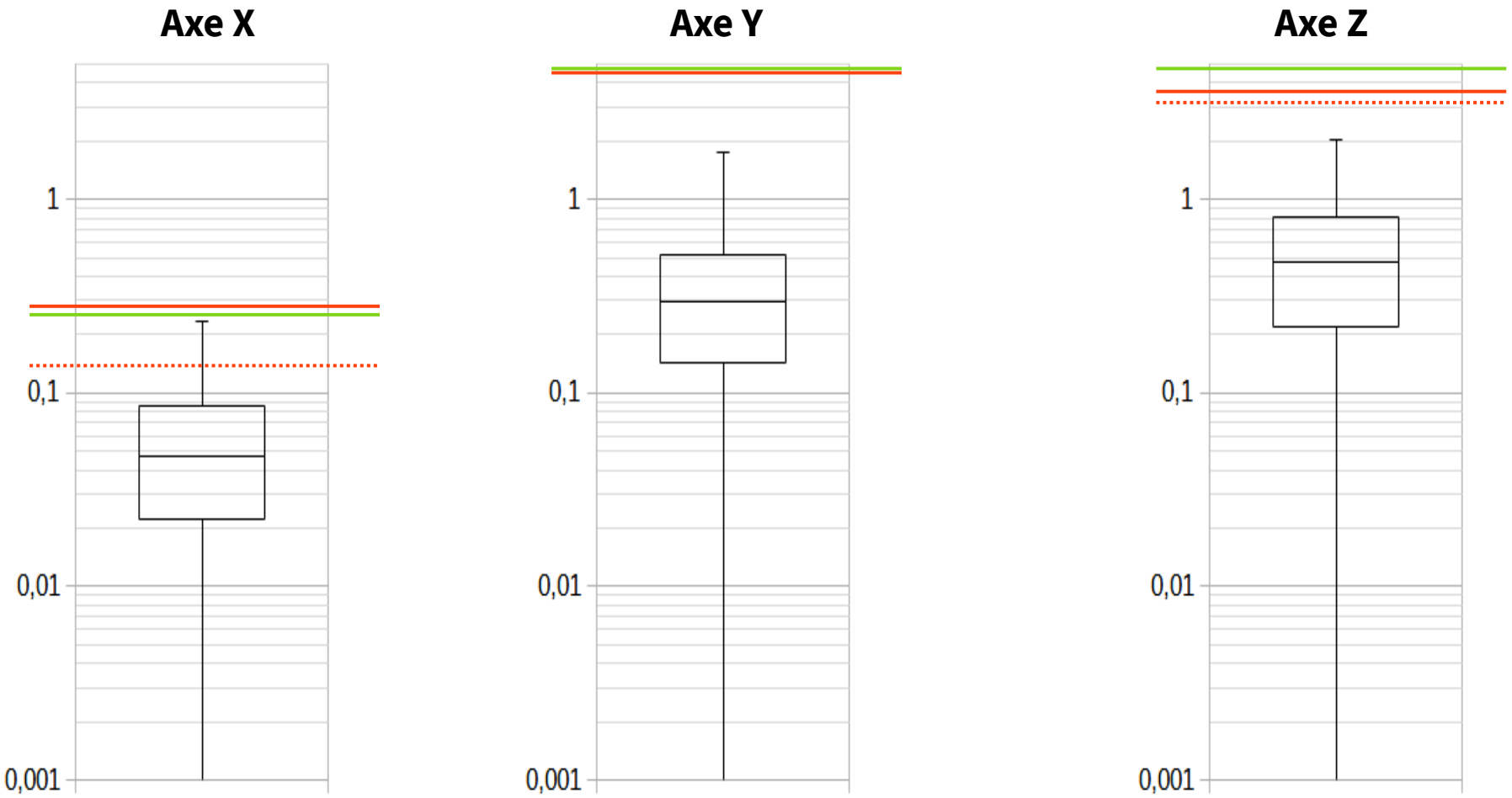


Maximum de la norme de la vitesse angulaire (en rad/s)



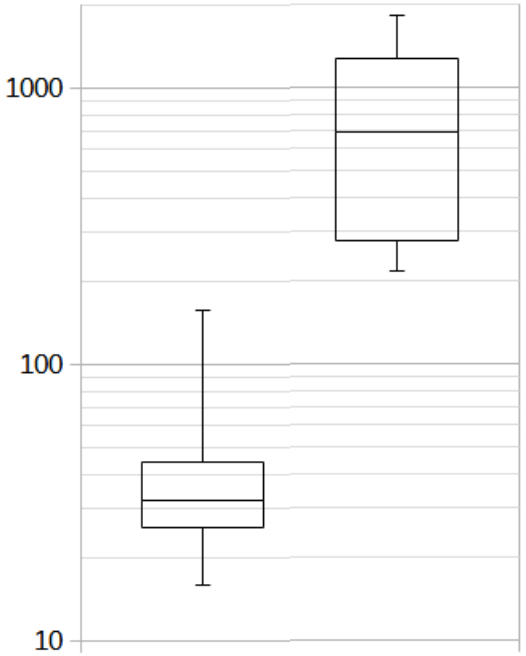
Seuil de vitesse de rotation

Comportement moyen de la vitesse de rotation (en rad/s) en fonction de l'axe

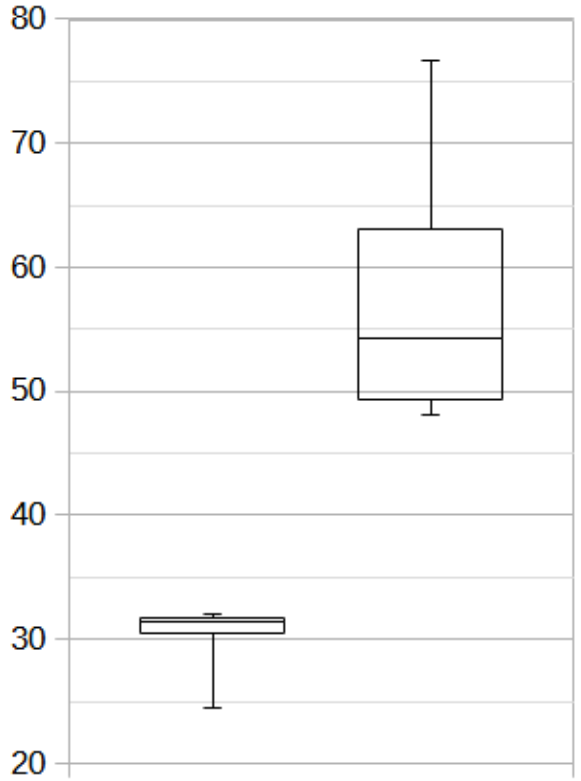


Somme HIC et Fi

Essais vélo



Moto sur circuit



Période du HIC

Plage de valeurs du HIC pour un trajet moyen

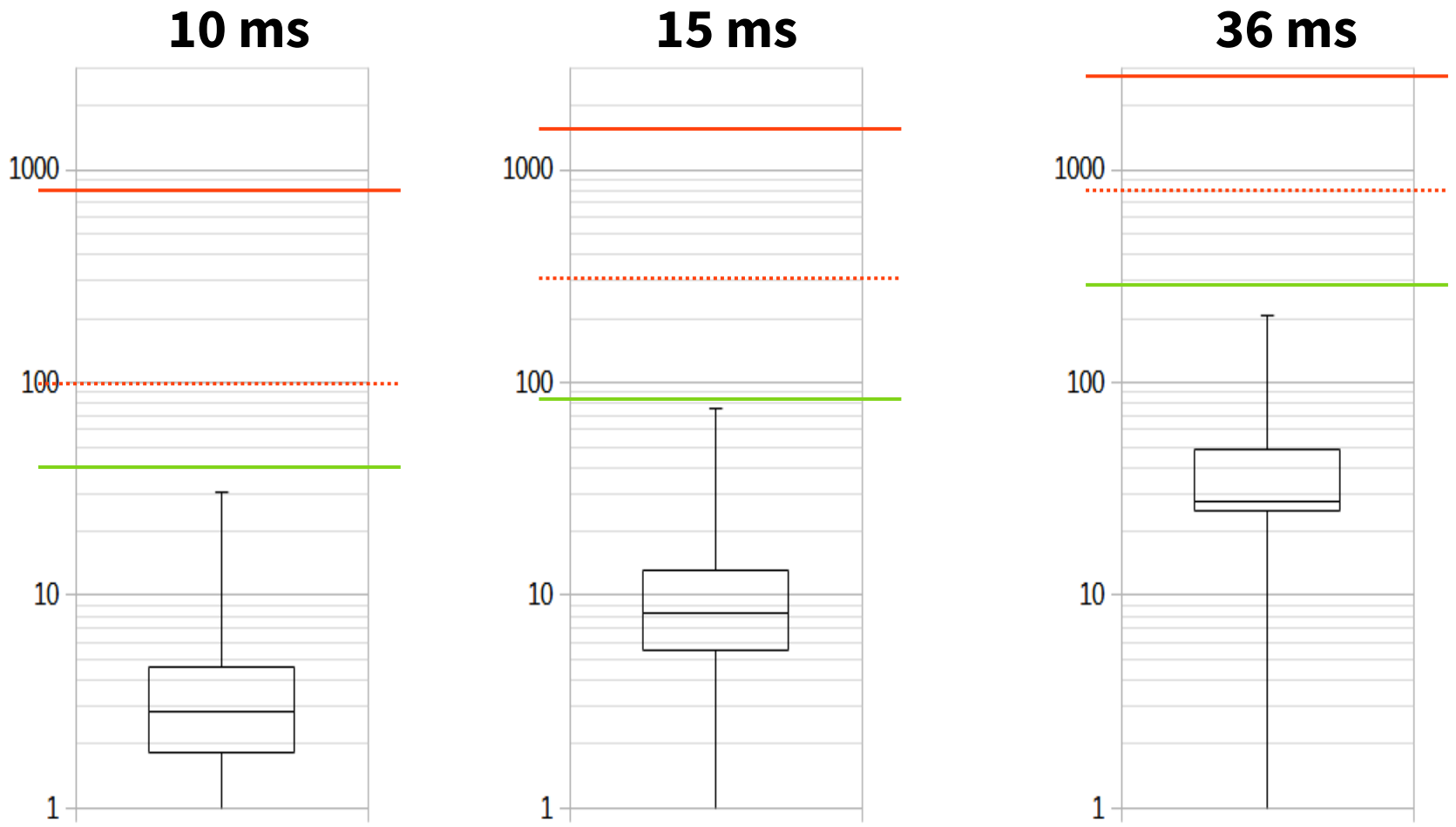
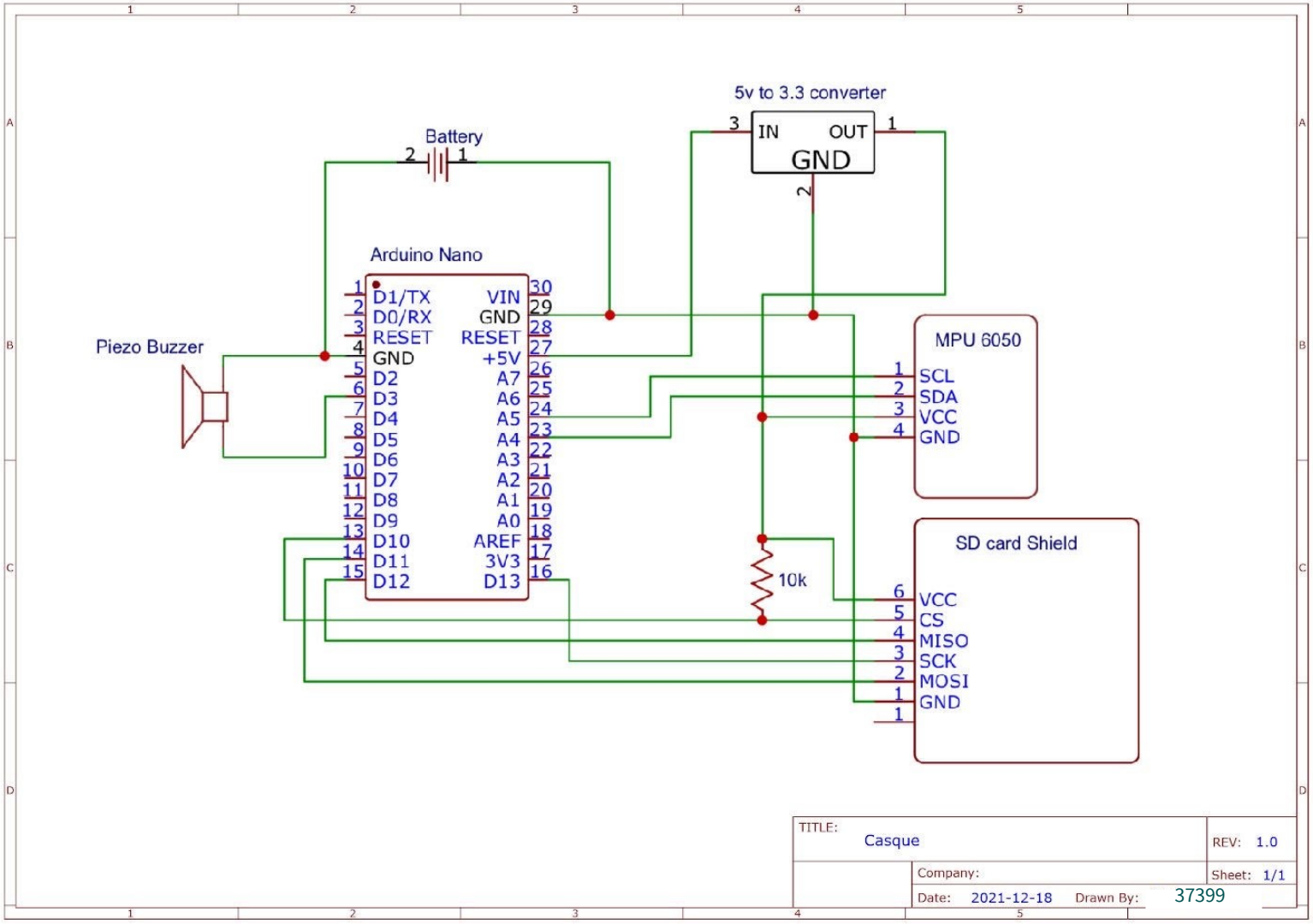


Schéma électrique du casque



Code Arduino

```
#include "SdFat.h"
#include <Adafruit_MPU6050.h>
#include <Wire.h>
#define FILE_BASE_NAME "Data"

SdFat sd;
File file;

const uint8_t BASE_NAME_SIZE = sizeof(FILE_BASE_NAME) - 1;
char fileName[] = FILE_BASE_NAME "00.txt";
const int pinBranchementsCS = 10; //pin broche CS
const int delta = 20000;   //duree echantillonage
long time;
long deb;
bool flag;
bool fin;

Adafruit_MPU6050 mpu;
```

Code Arduino

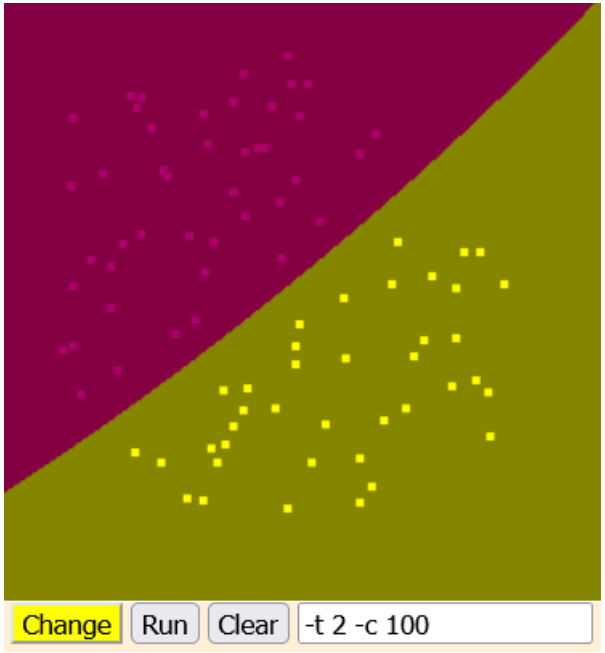
```
void setup() {
  Serial.begin(9600);

  //Init accelerometre
  Serial.println(F("Initialize Accel System"));
  if (!mpu.begin(0x68)) { // Changer addresse si necessaire 0x69
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }

  Serial.println("Starting Accel");
  mpu.setAccelerometerRange(MPU6050_RANGE_16_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_260_HZ);
  delay(1000);

  // Initialisation de la carte SD
  delay(10);
  Serial.println(F("Initialisation de la carte"));
  delay(10);
  if(!sd.begin(pinBranchementsCS)){
    Serial.println(F("Echec de l'initialisation"));
  }
  delay(10);
  sd.begin(pinBranchementsCS);
  Serial.println(F("Initialisation terminée."));

  // construction nouveau fichier

  while(sd.exists(fileName)){
    if (fileName[BASE_NAME_SIZE + 1] != '9'){
      fileName[BASE_NAME_SIZE +1]++;
    } else if (fileName[BASE_NAME_SIZE] != '9'){
      fileName[BASE_NAME_SIZE + 1] = '0';
      fileName[BASE_NAME_SIZE]++;
    } else {
      Serial.println(F("Creation impossible"));
      return;
    }
  }
  file = sd.open(fileName, FILE_WRITE); //ouverture du fichier
  delay(100);
  //Début
  file.println();
  fin = false;
  flag = true;
  tone(3, 220, 200);
  delay(500);
  tone(3, 220, 200);
  delay(500);
  tone(3, 196, 200);
  delay(300);
  deb = millis();
}
```

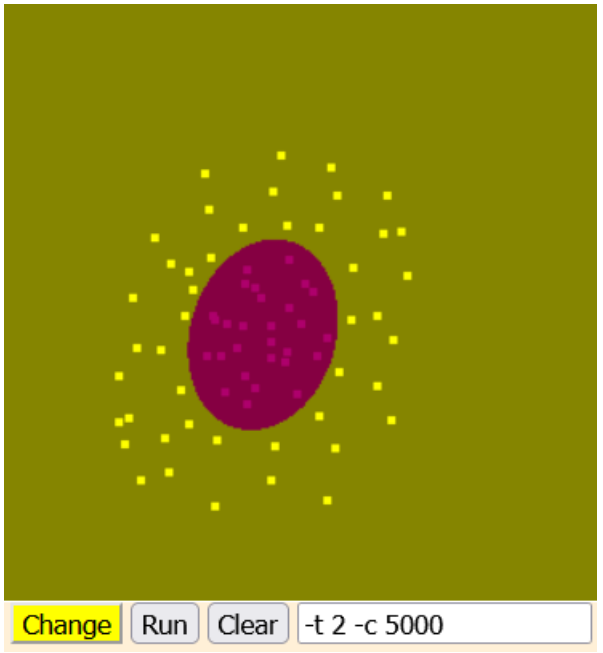
Code Arduino

```
void loop() {
  while(!fin){
    while(flag){
      time = millis();

      sensors_event_t a, g, temp;
      mpu.getEvent(&a, &g, &temp);

      if (file){
        file.print(time);
        file.print("|");
        file.print(a.acceleration.x);
        file.print("|");
        file.print(a.acceleration.y);
        file.print("|");
        file.print(a.acceleration.z);
        file.print("|");
        file.print(g.gyro.x);
        file.print("|");
        file.print(g.gyro.y);
        file.print("|");
        file.print(g.gyro.z);
        file.print("|");
        file.print(temp.temperature);
        file.println();
        Serial.print(".");
      }
      else{
        Serial.println(F("Echec d'ouverture du fichier"));
      }
      if ((time-deb)>delta){
        flag = false;
      }
    }
    Serial.print("Close");
    file.close();
    tone(3, 131, 200);
    delay(200);
    tone(3, 147, 200);
    delay(200);
    tone(3, 175, 200);
    delay(200);
    tone(3, 247, 200);
    delay(200);
    fin  = true;
    delay(10);
  }
}
```

MVS

Utilisation de la bibliothèque libsvm





Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 1–27.

MSV

Objectif d'optimisation :

- ***θ*** représente le vecteur des paramètres à optimiser
- Soit ***x*** le vecteur colonne représentant les données liées à une course
- Le vecteur ***y*** représente la classe de x
  On a donc y = 1 si il s'agit d'une chute, et y = 0 sinon

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

- Si y = 1, on veut donc :

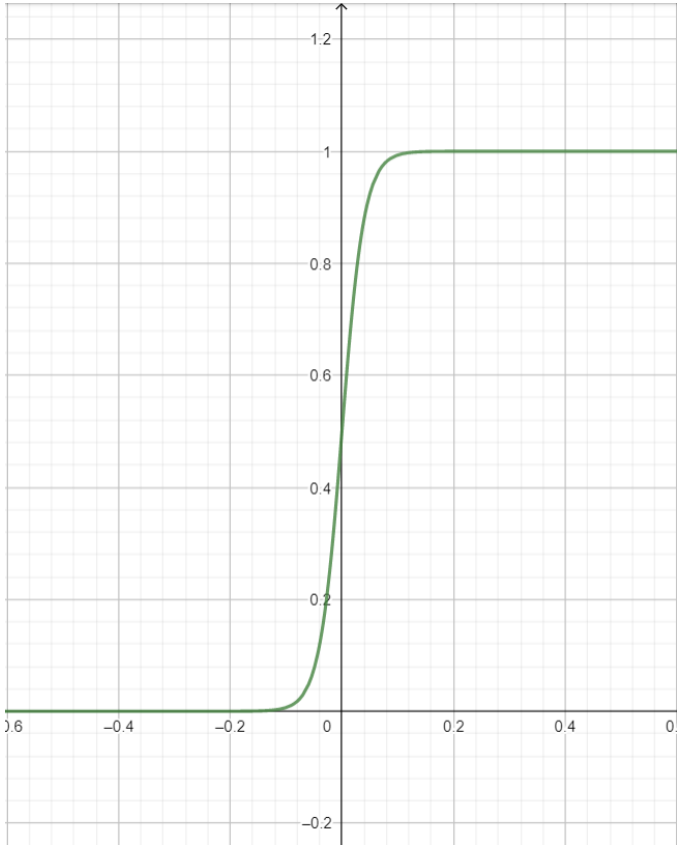
$$h_\theta(x) \approx 1$$

Soit $\theta^T x \gg 0$

- Si y = 0, on veut donc :

$$h_\theta(x) \approx 0$$

Soit $\theta^T x \ll 0$



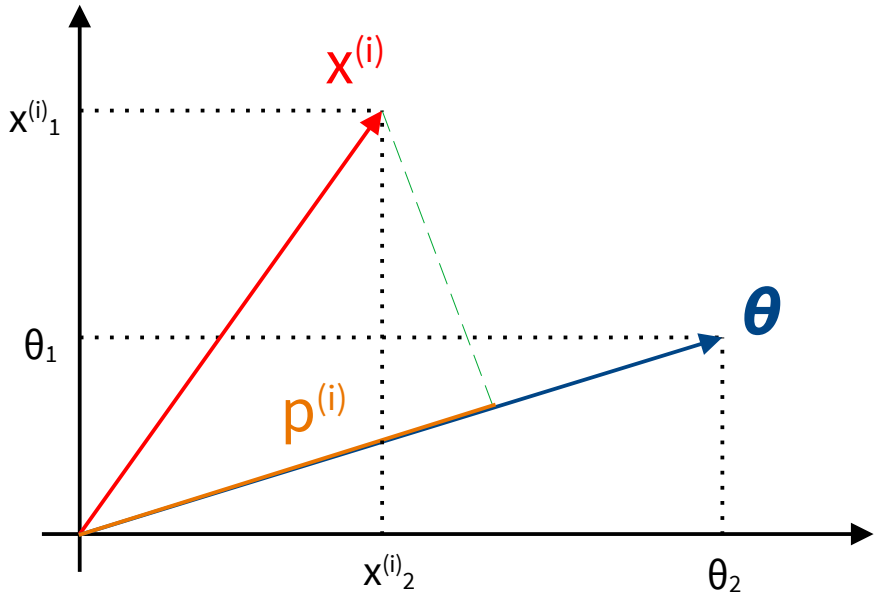
MSV

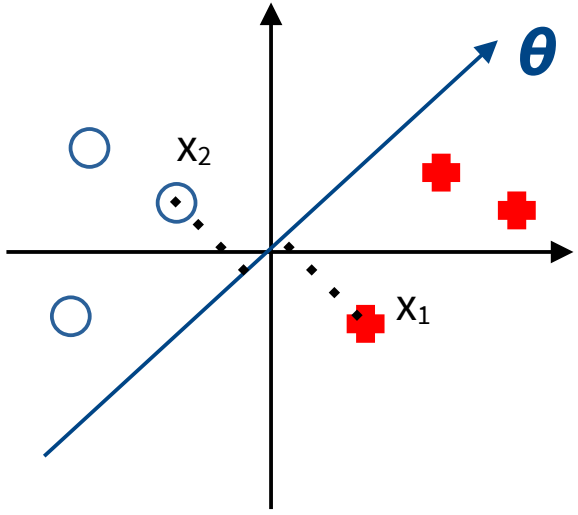
On a pour un exemple $x^{(i)}$, $\theta^T x^{(i)} = p^{(i)}.\|\theta\|$

D'où $\theta^T x^{(i)} = \theta_1 x^{(i)}_1 + \theta_2 x^{(i)}_2$

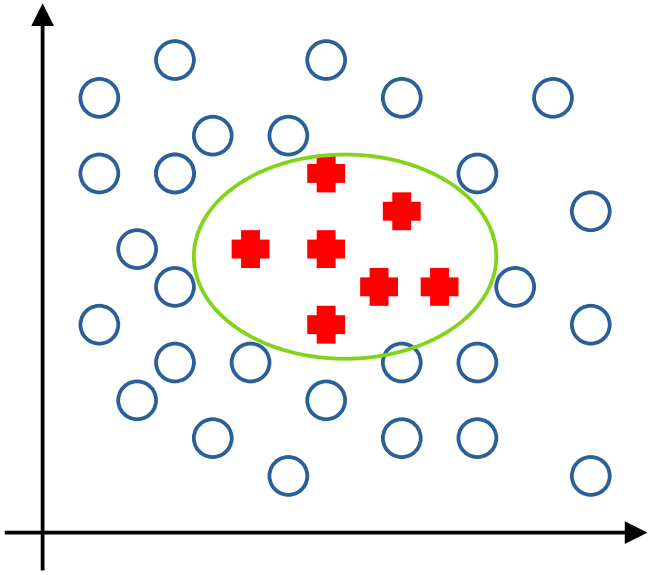
On veut : $p^{(i)}.\|\theta\| \geq 1$ si $y^{(i)} = 1$

$p^{(i)}.\|\theta\| \leq -1$ si $y^{(i)} = 0$

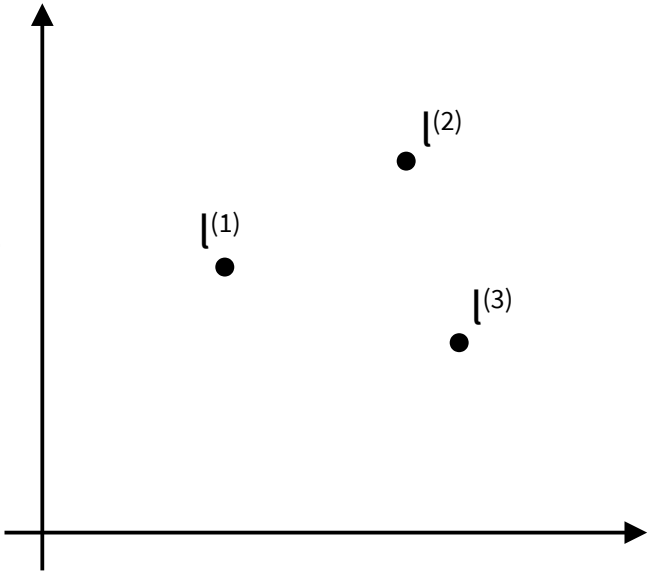




MSV



On extrait certaines valeurs



On veut :

$h_\theta(x)=1$ Si $\theta_0+\theta_1x_1+\theta_2x_+\ldots\geq 0$

$h_\theta(x)=0$ sinon

On peut définir une fonction f_i qui prend x et en variable, et qui nous donne un valeur caractérisant la proximité ou « similarité » entre x et $l^{(i)}$. f_i est ce qu'on appelle un noyau.

MSV

On prend généralement un noyau de type « Gaussien » de la forme :

$$f_i(x) = \exp\left(\frac{-\|x - l^{(i)}\|^2}{2\sigma^2}\right)$$

- Si $x \sim l^{(i)}$:

$$f_i(x) \simeq \exp\left(\frac{-0^2}{2\sigma^2}\right) \simeq 1$$

- Si x loin de $l^{(i)}$:

$$f_i(x) \simeq \exp\left(\frac{-inf^2}{2\sigma^2}\right) \simeq 0$$

On veut à présent : $h_\theta(x) = 1$ Si $\theta_0+\theta_1 f_1(x)+\theta_2 f_2(x)+\ldots \geq 0$

$h_\theta(x) = 0$ sinon

Bibliographie

- Kangas, « Development of Accelerometry-Based Fall Detection : From Laboratory Environment to Real Life »
- Evaluation of Algorithm for the Fall and Fall Direction Detection during Bike Riding. International Journal of Control and Automation, 6(6), 209–218 | 10.14257/ijca.2013.6.6.20
- Hutchinson, Kaiser, et Lankarani, « The Head Injury Criterion (HIC) Functional »
- Schnee, Stegmaier, et Li, « A Probabilistic Approach to Online Classification of Bicycle Crashes »
- Tabei, Askarian, et Chong, « Accident Detection System for Bicycle Riders »
- Schepers et al., « An International Review of the Frequency of Single-Bicycle Crashes (SBCs) and Their Relation to Bicycle Modal Share »

Bibliographie

- Markiewicz et al. « Biomecanique Du Choc Critères De Blessure Et Tolérance Humaine A L'impact Tests Et Procédures D'essais Automobile »
- Niska et Wenäll, « Simulated single-bicycle crashes in the VTI crash safety laboratory »
- Schepers et Klein Wolt, « Single-Bicycle Crash Types and Characteristics »
- Williams, « bicycle Crash Detection: Using A Voice-Assistant For More Accurate Reporting »
- Cripton et al., « Bicycle Helmets Are Highly Effective at Preventing Head Injury during Head Impact »
- Wang et al., « Evaluation of Head Injury Criteria for Injury Prediction Effectiveness »
- Dima et Covaciu, « Solutions for Acceleration Measurement in Vehicle Crash Tests »

Bibliographie

- Prati, Pietrantoni, et Fraboni, « Using Data Mining Techniques to Predict the Severity of Bicycle Crashes »