

TIPE 2022/2023 : LA VILLE



# Etude des mouvements de foule par modélisations physique et informatique

*BOUZAOUI Asmae - N° 53881 -PC*

# Sommaire

Dans quelle mesure le guidage d'une foule dense permet sa sécurisation ?

I. ETAT DE L'ART 

II. LE DEPART D'UN MARATHON 

III. EXPERIENCES ET RESULTATS 

IV. CONCLUSION 

## I. Etat de l'art

## II. Départ d'un marathon

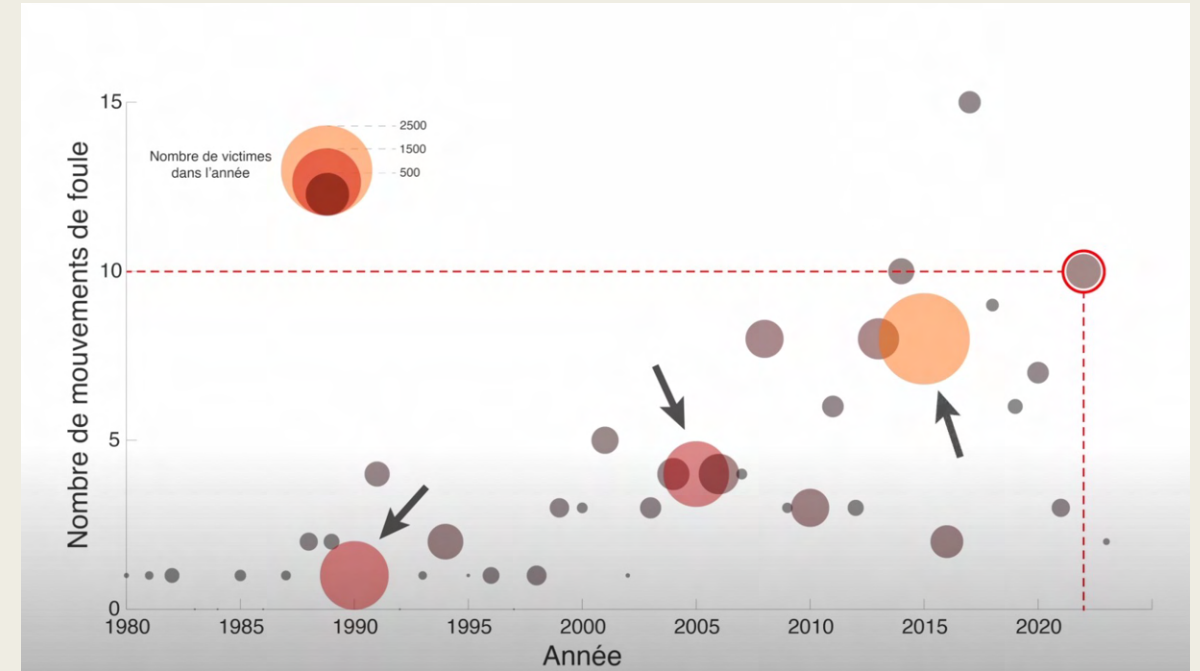
## III. Expériences et résultats

## IV. Conclusion



Foule à la Mecque lors du pèlerinage

Mehdi Moussaid Fouloscopie



Accidents de foule

Mehdi Moussaid Fouloscopie



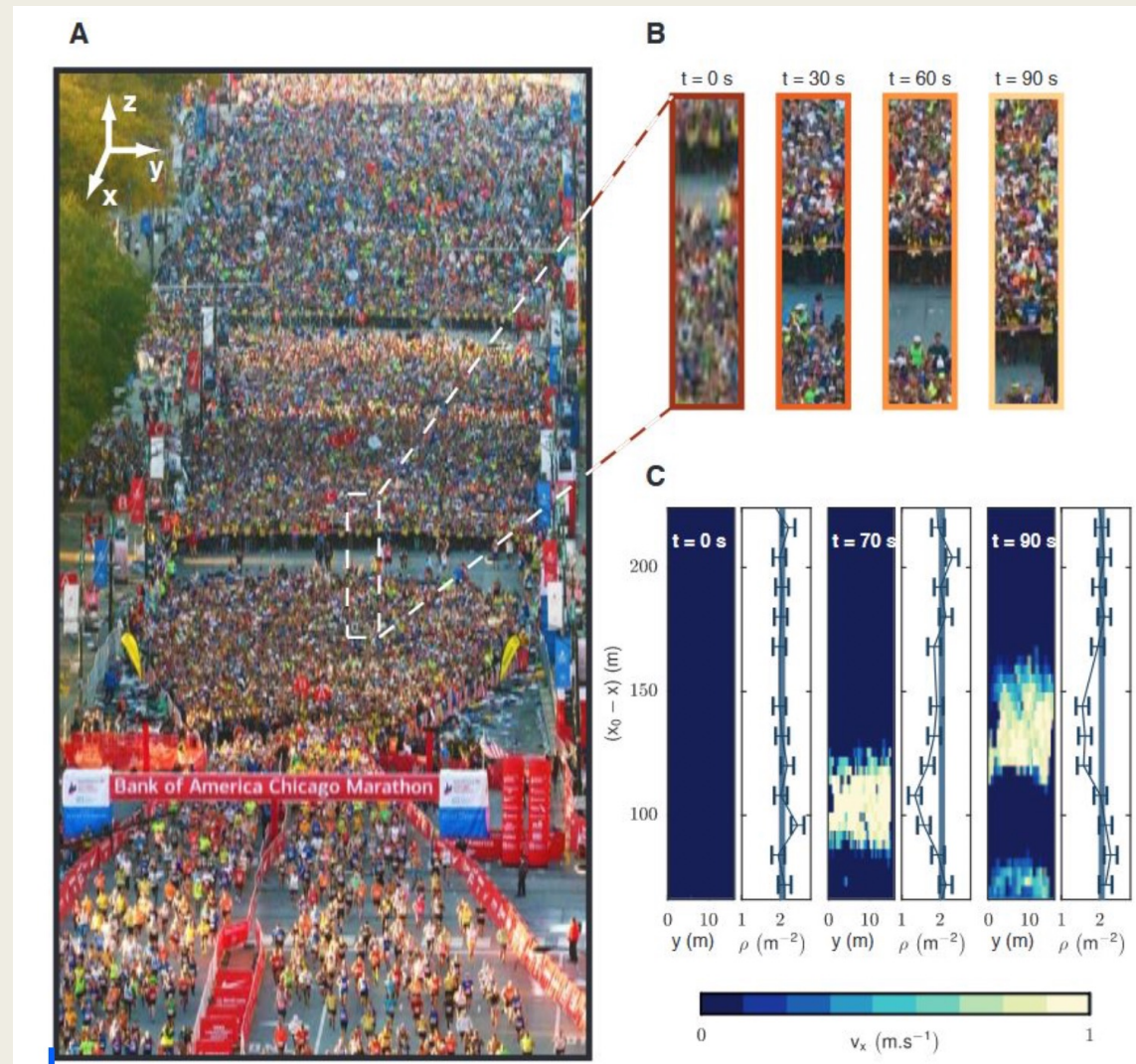


Troupeau de moutons filmé depuis le  
ciel de Nouvelle-Zélande à (0:47s)

I Genside (Groupe Prisma Media)



À (0:28s)



## Equation de propagation

Approximation acoustique :  $\rho = \rho_0 + \delta\rho$  avec  $\delta\rho \ll \rho_0$

$\rho$  : Masse volumique ( $kg.m^{-3}$ )

$v$  : Vitesse ( $m.s^{-1}$ )

$t$  : Temps (s)

$y$  : Position selon  $O_y$  (m)

Equation de conservation de la masse :  $\frac{\partial \rho}{\partial t} + \frac{\partial(\rho v)}{\partial y} = 0$  (1)

Comme  $v(\rho_0) = 0$ ,

Développement limité de  $v(\rho)$  réinjecté dans (1) :

$$\left. \frac{\partial \delta\rho}{\partial t} + \rho_0 \frac{\partial}{\partial y} \left( \delta\rho \cdot \frac{\partial v}{\partial \rho} \right) \right\} \text{HP : Hugues } v = v(\rho)$$

$$\frac{\partial \delta\rho}{\partial t} + \rho_0 \frac{\partial v}{\partial \rho} \cdot \frac{\partial \delta\rho}{\partial y} \text{ avec } c = -\rho_0 \frac{\partial v}{\partial \rho}$$

(Propagation selon les  $y$  décroissant)



# Expérience

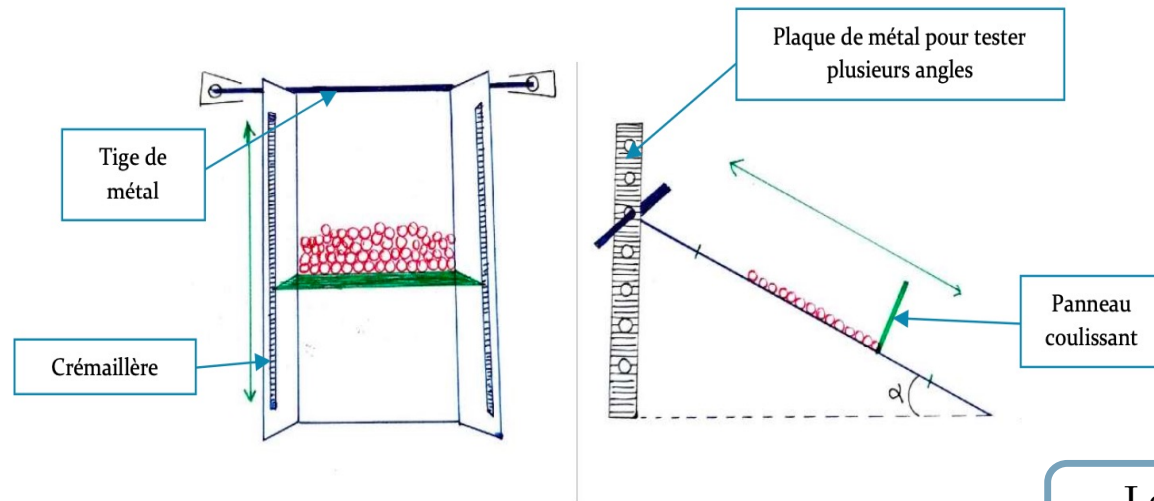
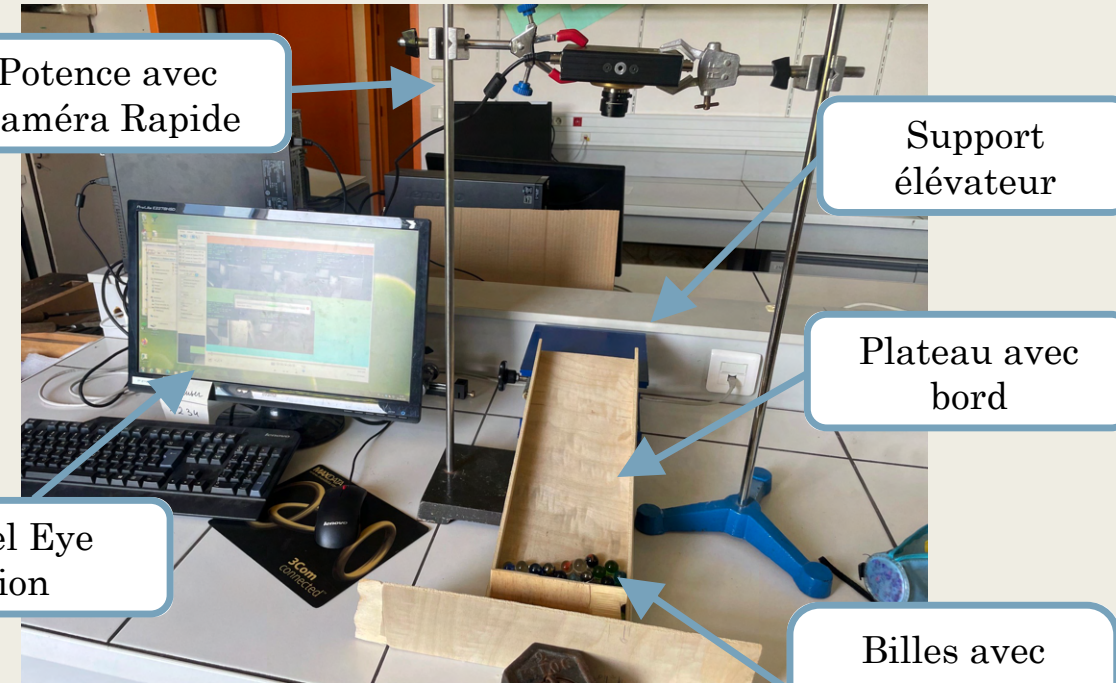
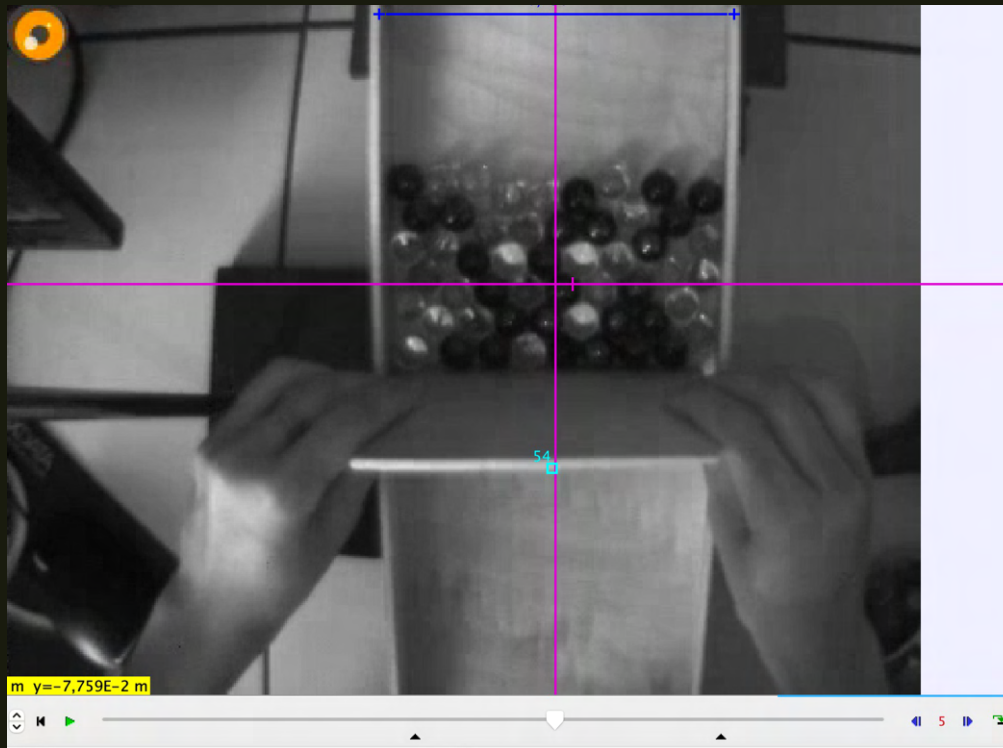


Figure 1 : Schémas du montage expérimental

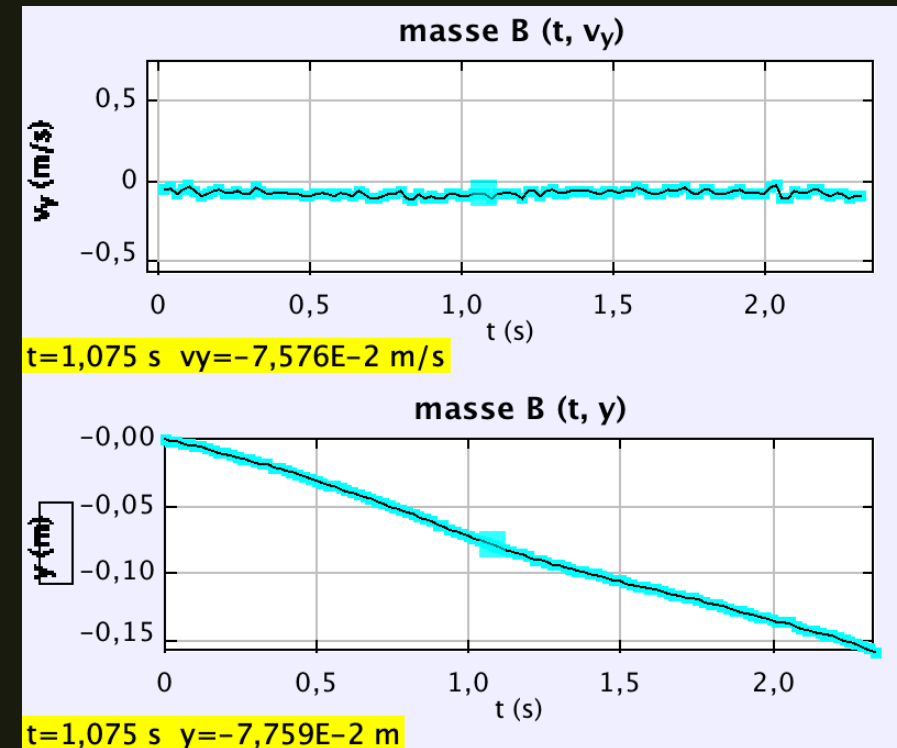


Expérience en laboratoire

## Vitesse du panneau coulissant



Pointage du panneau sur Tracker  
(Essai 1)

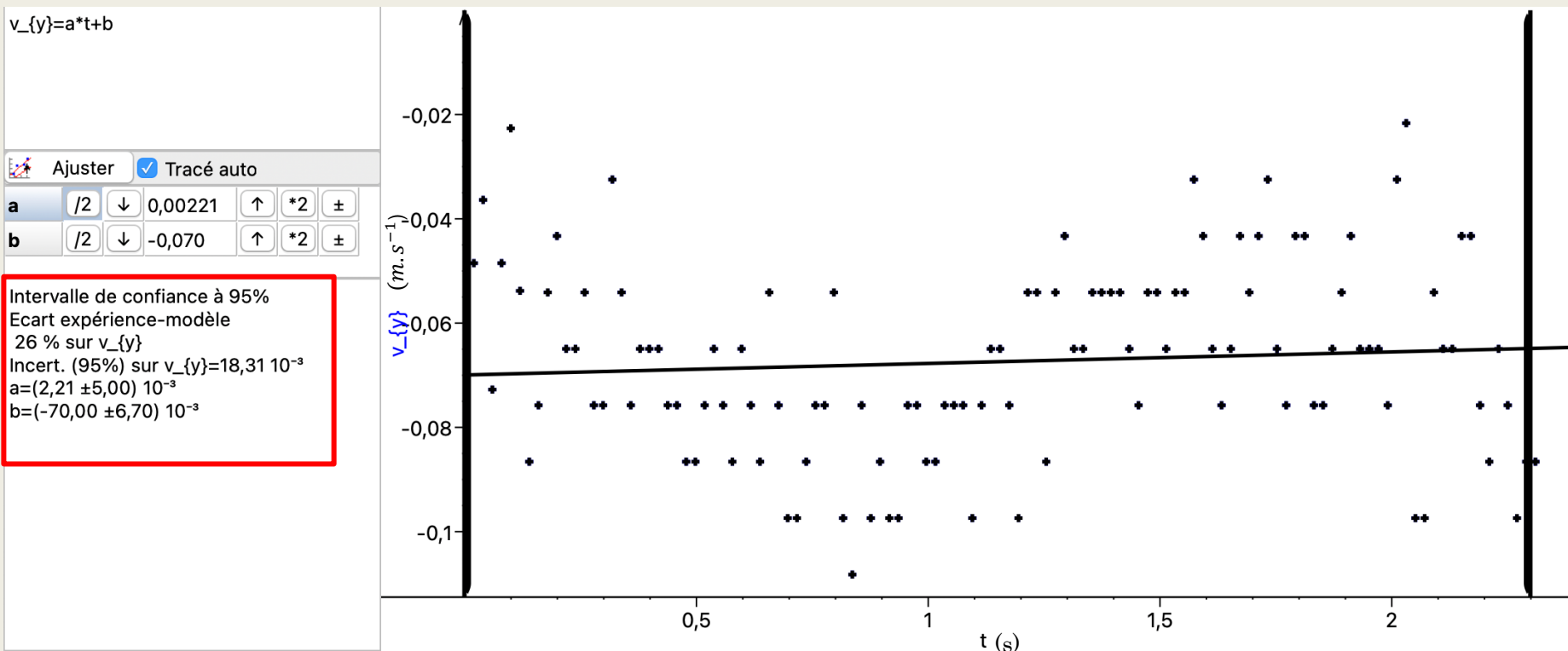


Courbes obtenues après  
pointage



# Vitesse du panneau coulissant

Essai 1:

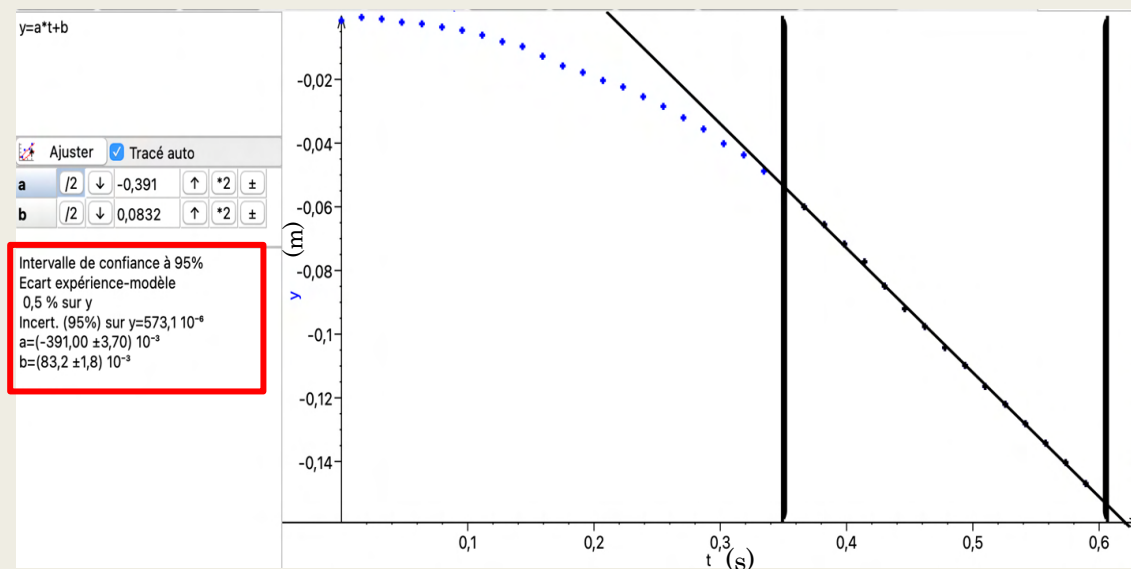


$$v_{\text{panneau},1} = f(t)$$

$$v_{\text{panneau},1} = (0,70 \pm 0,07) \text{ m.s}^{-1}$$

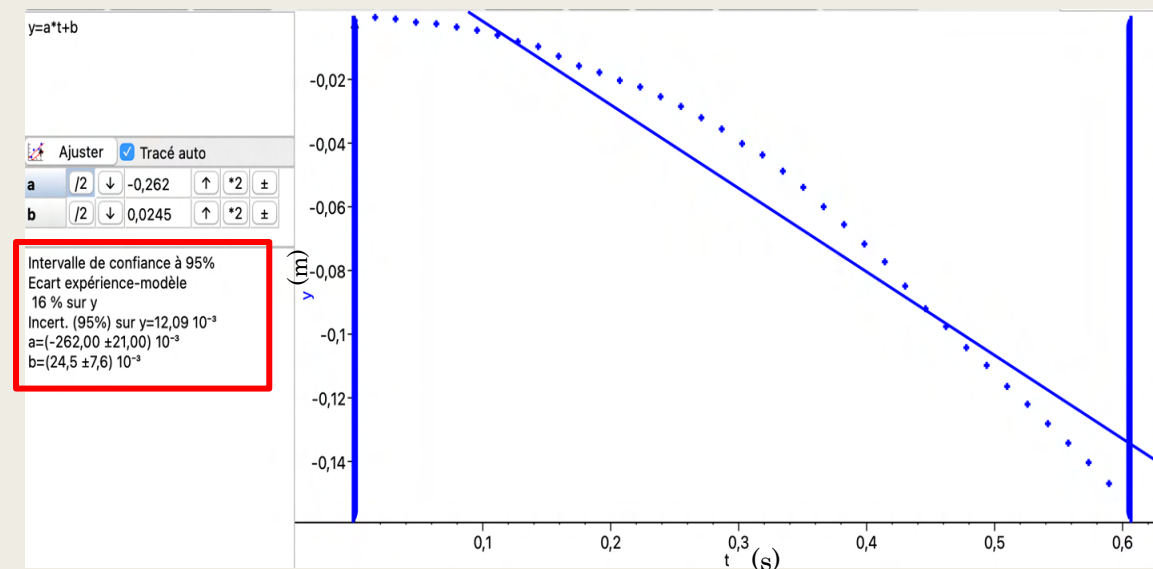
# Vitesse du panneau coulissant

## Essai 2



Modélisation 1

$$y_{\text{panneau},2} = f(t)$$



Modélisation 2

$$v_{\text{panneau},2} = (3,91 \pm 0,04) \text{ m.s}^{-1}$$

$$v_{\text{panneau},2} = (2,6 \pm 0,2) \text{ m.s}^{-1}$$

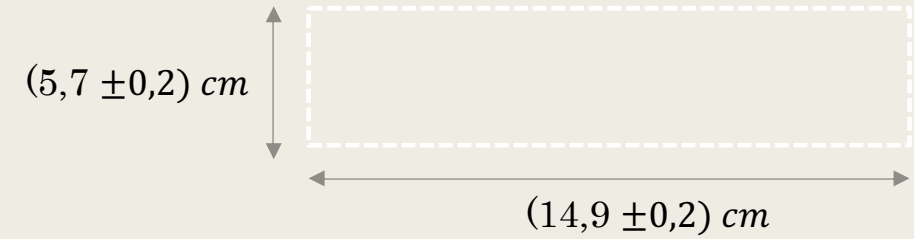
I. Etat de l'art

II. Départ d'un marathon

III. Expériences et résultats

IV. Conclusion

Densité au départ du panneau



Essai 1 :  $32 \pm 1$  billes

Essai 2 :  $25 \pm 1$  billes

Essai 3 :  $27 \pm 1$  billes

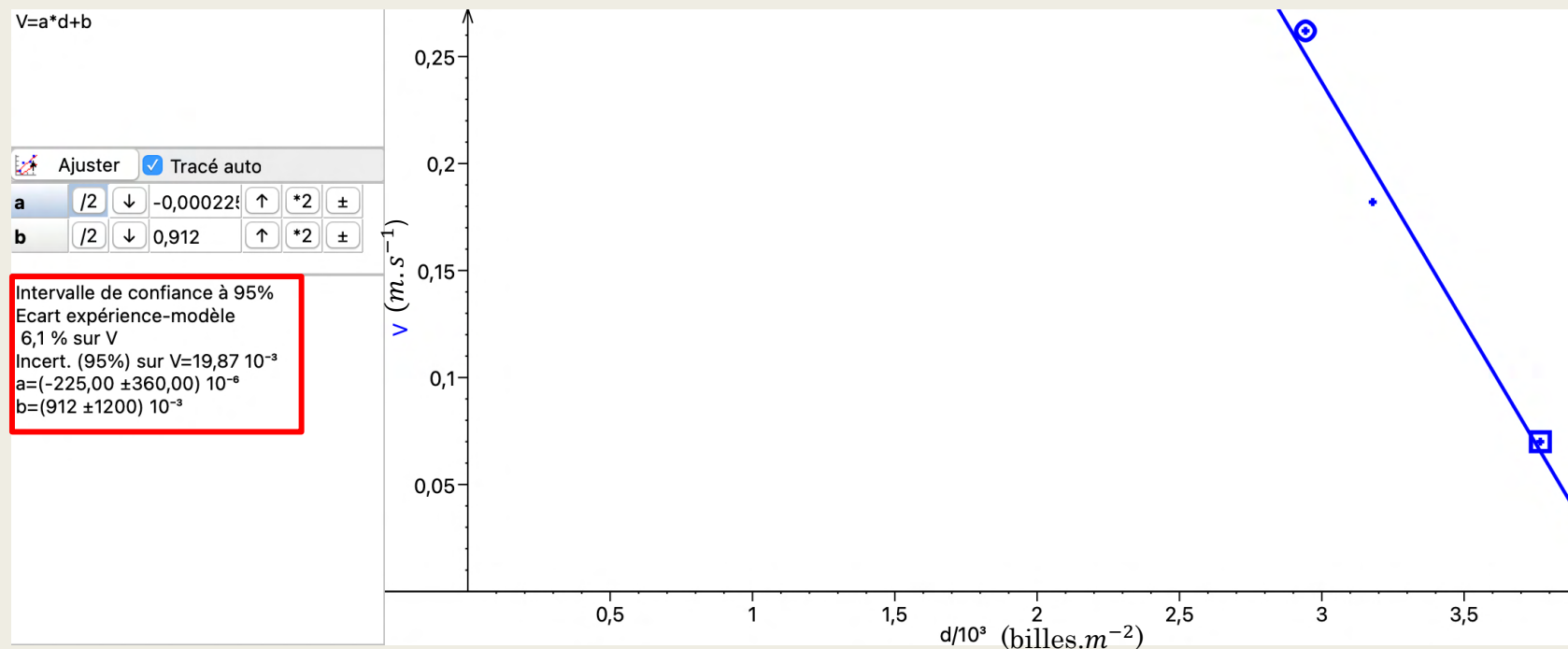
$3768 \text{ billes.m}^{-2}$

$2944 \text{ billes.m}^{-2}$

$3179 \text{ billes.m}^{-2}$



## Vitesse des billes en fonction de la densité (au départ)



$$v = f(\rho)$$

$$v(\rho) = (-2,3 \pm 3,6) \times 10^{-4} \rho + (0,9 \pm 1,2) m.s^{-1}$$

Célérité de l'onde :

$$c = -\rho_0 \cdot \frac{\partial v}{\partial \rho}$$

Avec

$$\rho_0 = \frac{nbr_{bille} \times m_{bille}}{V_{bille}}$$

$$c = 2,8 \pm 0,9 \text{ m.s}^{-1}$$

$$c = 1,6 \pm 0,9 \text{ m.s}^{-1}$$

# Conclusion

## Similitudes Expérience/ Théorie

- Onde de vitesse observé
- Contrôle de la vitesse et de la densité par un guidage avant

## Différences Expérience/ Théorie

- La célérité dépend de chaque perturbation



## Discussion :

- Absence d'uniformité des frottements sur la planche
- Invalidité des petites perturbations
- Hypothèse de la densité uniforme au départ
- Longueur et largeur de la planche
- Etude statistique



# ANNEXES



# Equation de propagation

$$f'(x+h) = f(x) + hf'(x)$$

*Hypothèse Simplificatrice de  
Hugues :*

$$v = v(\rho)$$



# Tableau récapitulatif des mesures (1)

	Essai 1	Essai 2	Essai 3
Vitesse du Panneau ( $m.s^{-1}$ )	$v_{panneau,1} = (70 \pm 7) \times 10^{-3} m.s^{-1}$	$v_{panneau,2} = (262 \pm 21) \times 10^{-3} m.s^{-1}$	$v_{panneau,3} = (182 \pm 12) \times 10^{-3} m.s^{-1}$
$\rho_0$ ( $kg.m^{-3}$ )	$(1,3 \pm 0,8) \times 10^4 kg.m^{-3}$	$(1,5 \pm 0,8) \times 10^4 kg.m^{-3}$	$(1,1 \pm 0,8) \times 10^4 kg.m^{-3}$
Densité au départ $\rho$ ( $billes.m^{-2}$ )	$3768 billes.m^{-2}$	$2944 billes.m^{-2}$	$3179 billes.m^{-2}$

Régression Linéaire de ces points :

$$v(\rho) = (-2,3 \pm 3,6) \times 10^{-6} \rho + (0,9 \pm 1,2) m.s^{-1}$$

$$c = 2,8 \pm 0,9 m.s^{-1}$$

# Tableau récapitulatif des mesures (2)

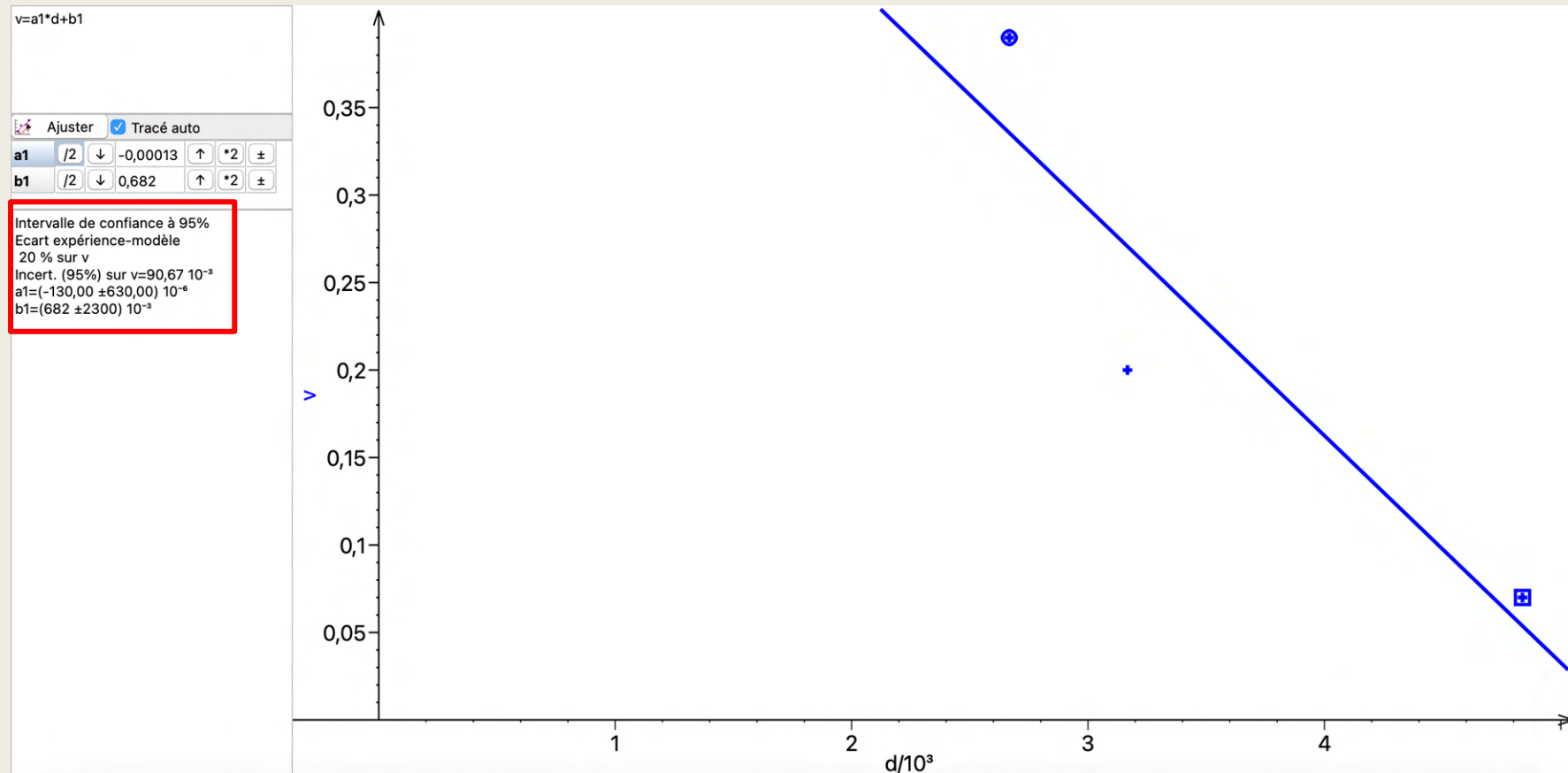
	Essai 1	Essai 2	Essai 3
Vitesse du Panneau ( $m.s^{-1}$ )	$v_{panneau,1} = (63 \pm 9) \times 10^{-3} m.s^{-1}$	$v_{panneau,2} = (329 \pm 4) \times 10^{-3} m.s^{-1}$	$v_{panneau,3} = (220 \pm 34) \times 10^{-3} m.s^{-1}$
$\rho_0$ ( $kg.m^{-3}$ )	$(1,2 \pm 0,8) \times 10^4 kg.m^{-3}$	$(1,2 \pm 0,8) \times 10^4 kg.m^{-3}$	$(1,4 \pm 0,8) \times 10^4 kg.m^{-3}$
Densité au départ $\rho$ ( $billes.m^{-2}$ )	$4883 billes.m^{-2}$	$2667 billes.m^{-2}$	$3167 billes.m^{-2}$

Régression Linéaire de ces points :

$$v(\rho) = (-1,3 \pm 0,7) \times 10^{-4} \rho + (0,7 \pm 0,2) m.s^{-1}$$

$$c = 1,6 \pm 0,9 m.s^{-1}$$

# Régression Linéaire $v = f(\rho)$ (2<sup>ème</sup> fois)



$$v(\rho) = (-1,3 \pm 0,7) \times 10^{-4} \rho + (0,7 \pm 0,2) \text{ m.s}^{-1}$$

# Programme Python (1)

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import matplotlib.image as mpimg
4 from math import *
5 import time
6 from copy import *
7 #import cv2
8 #vid=cv2.VideoCapture('Sample.mp4')
9
10 print('Chargement des images',end='\n')
11 img1=(mpimg.imread('D:/Users/LAPORTE/Desktop/images_analyse_video/1.jpg')).tolist()
12 img2=(mpimg.imread('D:/Users/LAPORTE/Desktop/images_analyse_video/2.jpg')).tolist()
13 img3=(mpimg.imread('D:/Users/LAPORTE/Desktop/images_analyse_video/3.jpg')).tolist()
14 img4=(mpimg.imread('D:/Users/LAPORTE/Desktop/images_analyse_video/4.jpg')).tolist()
15 img5=(mpimg.imread('D:/Users/LAPORTE/Desktop/images_analyse_video/5.jpg')).tolist()
16 img6=(mpimg.imread('D:/Users/LAPORTE/Desktop/images_analyse_video/6.jpg')).tolist()
17 #print(img1[0])
18
19 def negatif(img):
20     img2=deepcopy(img)
21     for i in range(len(img)):
22         for j in range(len(img[0])):
23             for t in range(len(img[0][0])):
24                 img2[i][j][t]=255-img[i][j][t]
25     return img2
26
```

```
27 def deepmoy(L):
28     if type(L[0])!=tuple and type(L[i])!=list:
29         a=0
30         for i in L:
31             a+=i
32         return a/len(L)
33     return [deepmoy(L[i]) for i in L]
34
35 def moy(L):
36     assert(type(L)==list)
37     t=0
38     for _ in L:
39         t+=_
40     return t/len(L)
41
42 def moy2(L):
43     l=[]
44     #print("liste de départ pour moy2",L)
45     for _ in range(len(L[0])):
46         l.append(moy([L[i][_] for i in range(len(L))]))
47     return l
48
49 def encommun(l1,l2):
50     for i in l1:
51         for j in l2:
52             if i==j:
53                 return True
54     return False
55
56 def coul(obj,seuil):
57     if moy(obj)<seuil:
58         return True
59     return False
60
```

# Programme Python (2)

```
61 def pepr4(L,taille,visite):
62     l=[]
63     for i in range(len(L)):
64         if L[i] not in visite:
65             visite.append(L[i])
66             l.append(pepr5(i,L,taille,visite))
67     return l
68
69 def pepr5(i,L,taille,visite):
70     li=[L[i]]
71     visite.append(L[i])
72     for j in range(i+1,len(L)):
73         if L[j] not in visite and Pythagore(L[i],L[j])<taille:
74             li+=pepr5(j,L,taille,visite)
75     return li
76
77 def memeobjet(L,taille):
78     a=pepr4(L,taille,[])
79     return [moy2(t) for t in a]
80
81 def sel_pix(img,seuil,form,taille):
82     L=[]
83     for i in range(len(img)):
84         for j in range(len(img[i])):
85             a=img[i][j]
86             if form(a,seuil):
87                 L.append((i,j))
88     return memeobjet(L,taille)
89
90 def Pythagore(a,b):
91     return ((a[0]-b[0])**2+(a[1]-b[1])**2)**(1/2)
92
```

```
93 def recordcos(L,d):
94     for t in range(len(L)):
95         d[t].append(L[t])
96     return d
97
98 def recherche2(L,l):
99     assert len(L)==len(l)
100    l2=[]
101    p=[]
102    for i in L:
103        p=[Pythagore(i,j) for j in l]
104        l2.append(l[p.index(min(p))])
105    return l2
106
107 def trilogic(L,l):
108     if len(L)==1:
109         return L
110     return recherche2(L,l)
111
112 def t(i,vid):
113     print(' ',end='\r')
114     print(f"Traitement: {i}/{len(vid)}",end='\r')
115
```

# Programme Python (3)

```
116 def analyse(vid,seuil,form,dist):
117     """
118     vid: liste d'images
119     seuil: valeur minimale
120     form: critere
121     dist: taille max des objets
122     """
123     t1=time.time()
124     i=1
125     t(i,vid)
126     pixels=sel_pix(vid[0],seuil,form,dist)
127     #print(pixels)
128     d={i:[pixels[i]] for i in range(len(pixels))}
129     for img in vid[1:]:
130         i+=1
131         t(i,vid)
132         pixelsimg=sel_pix(img,seuil,coul,taille)
133         #print(pixels,pixelsimg)
134         l2=trilogic(pixels,pixelsimg)
135         #trouver les objets dans l'image suivante
136         recordcos(l2,d)
137     d2={i:[Pythagore(d[i][j],d[i][j+1]) for j in range(len(d[0])-1)] for i in d.keys()}
138     t2=time.time()
139     #print(f"Nombre d'objets suivis: {len(d)}")
140     #print("\n", 'Temps: ',f"{t2-t1:.5f}")
141     #print("\n", 'Positions initiales: ', [d[i][0] for i in d.keys()])
142     #return f"Positions: {d} , Vitesses: {d2}"
143     return [d,d2]
144
```



