

***Analyse physique approfondie des vagues  
pour une compréhension avancée dans le  
monde du surf***



DRAMÉ NOAH

# La discipline

Le surf consiste à se déplacer sur l'eau par l'intermédiaire d'une planche.

Physiquement, c'est l'opposition entre la gravité et la flottabilité des planches.



*The Encyclopedia of surfing*

Déplacement dû aux houles provoquées par les vagues.



*Répartition des différentes spots de surfs à travers le monde*

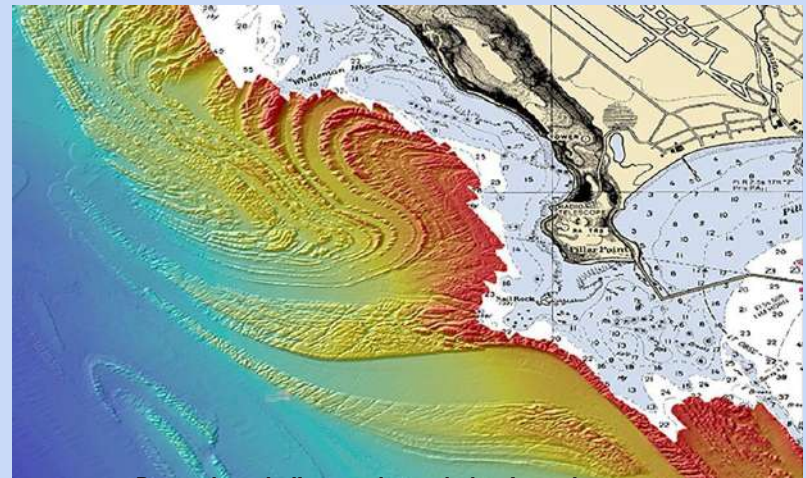
Zone de concentration de surf :

- Hawaï
- Taiwan
- Nazaré (Portugal)

La bathymétrie a un rôle important dans la formation des différentes vagues.

Influence du fond, facteurs environnementaux, etc ...

**Bathymétrie** : Sciences et technologies de mesure ou d'estimation de la profondeur



*Profondeur de l'eau au large de la côte californienne*

# Comment les caractéristiques des vagues influencent-elle sur les performances des surfeurs et comment pouvons-nous concevoir des innovations technologiques pour appréhender ces phénomènes complexes ?

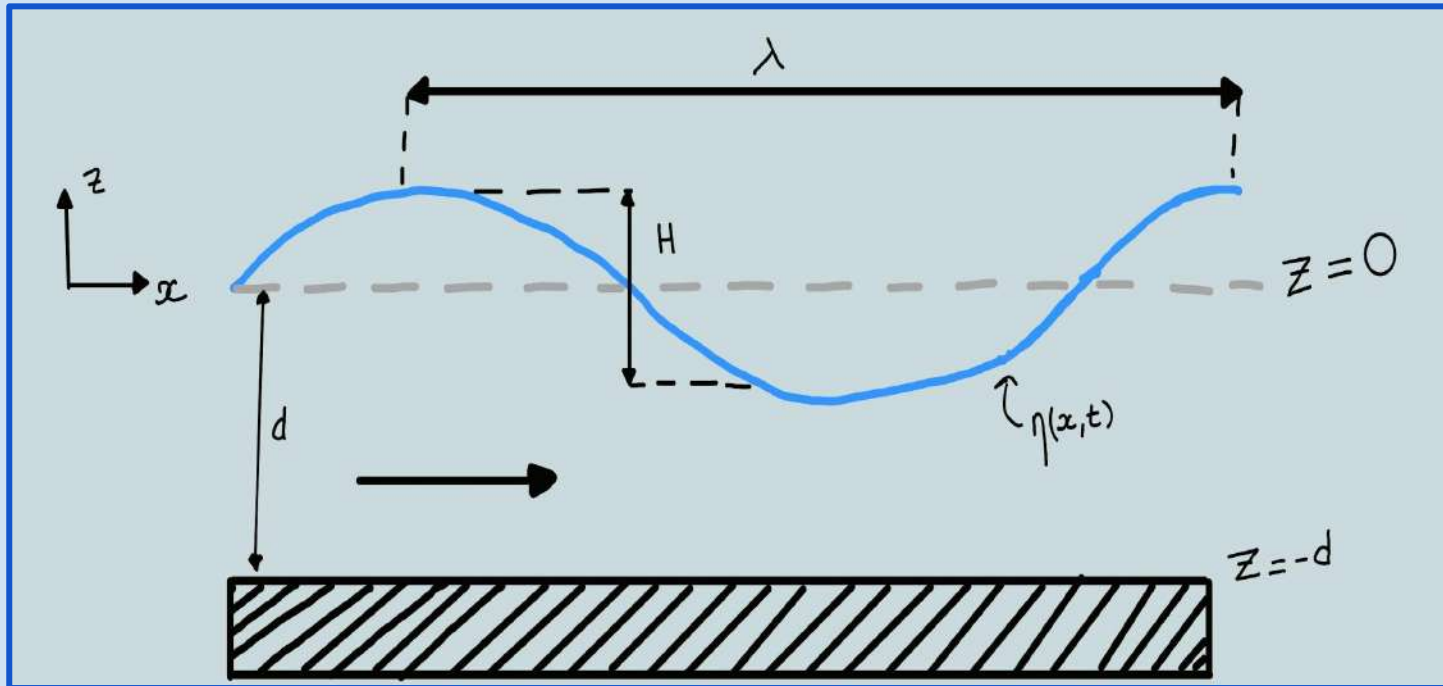
- Qu'est-qu'une vague ? Pourquoi son étude est-elle importante ?
- Modélisation physique de la vague.
- Influence du fond marin (Expérience)
- Confrontation au modèle informatique.
- Conclusion.

# Objectifs

## Les objectifs de mon binôme :

- Comprendre le système {surfeur+planche}
- L'influence de la forme de la planche
- Caractériser les conditions de surf
- Trouver les relations essentiels pour une bonne condition de surf
- Déterminer la “position idéale” (Stabilité)
- Dédire le “surf idéal”

# LA VAGUE



## Caractéristiques des vagues :

- longueur d'onde  $\lambda$
- la hauteur moyenne  $d$
- L'amplitude de la vague  $H/2$
- La hauteur de la vague  $\eta(x,t)$
- Les dimensions de propagation de la vague

# LA VAGUE

Différentes vagues, appelées déferlantes.



*Représentation des différentes formations de déferlantes*

Différentes “formes” de vague, qui dépendent des facteurs environnementaux.

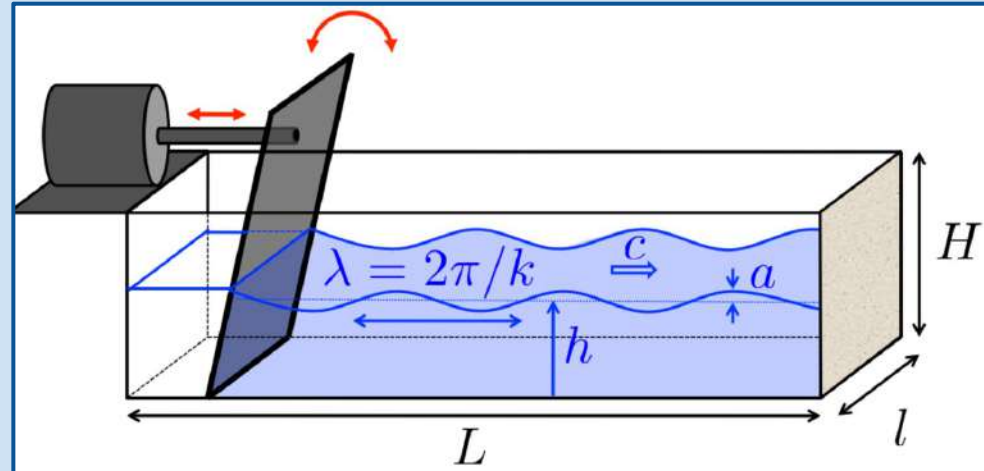


Étude de la bathymétrie sur l'influence de la formation de déferlantes.



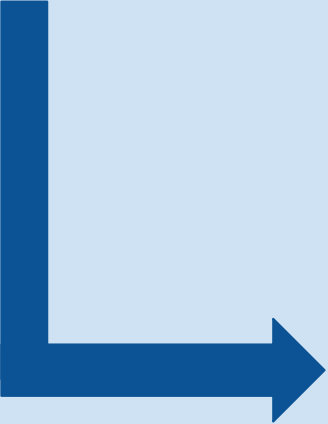
## Houle de Stokes :

- fluide parfait et incompressible
- Ecoulement irrotationnel
- Profondeur d'eau constante
- Vague linéaire (termes non linéaires négligés)



Représentation du système utilisé

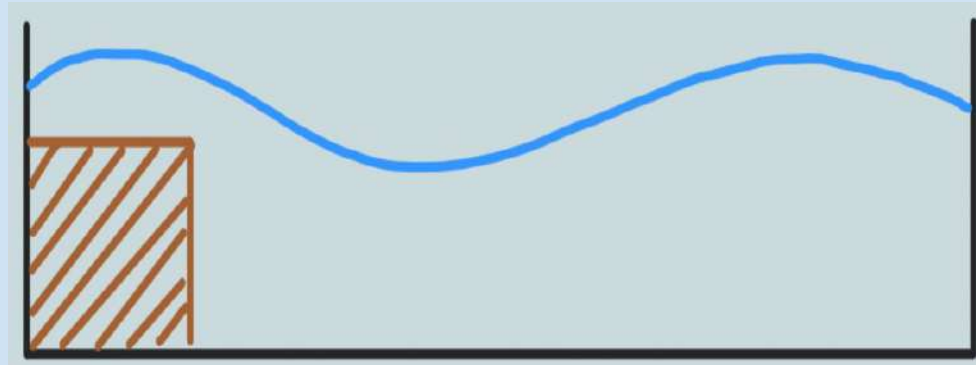
$$\omega^2 = g \times k \times \tanh(kH)$$


$$\eta(x, t) = -\Phi_m \times \frac{\omega}{g} \times \cosh(kh) \times \sin(kx - \omega t)$$

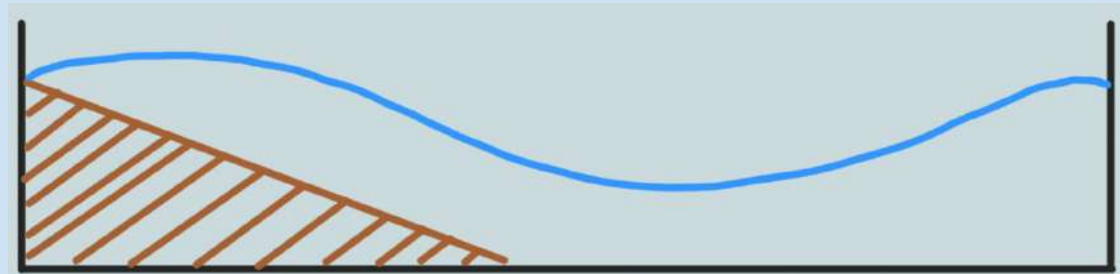


## Trois bathymétries à expérimenter :

- Le bloc "cube"



- La pente linéaire



- La pente progressive



# Matériel à disposition

- Un bassin aux dimensions 400x37x33 cm



- Différents profils bathymétriques, aux longueurs différentes

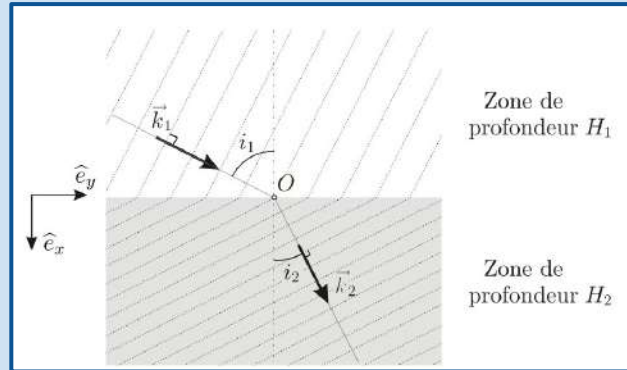


# Principe de l'expérience



## i. Bloc cube

Hypothèse :  
Absence d'onde réfléchie



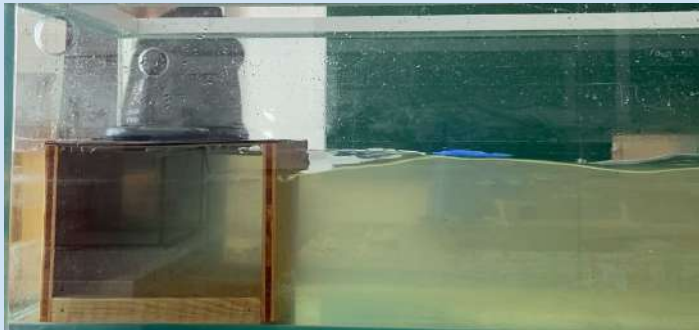
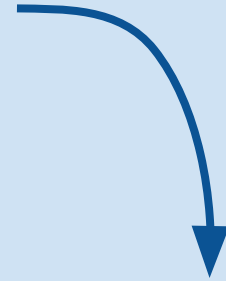
➔  $A_1 \cos(k_{1,y}x - \omega t) = A_2 \cos(k_{2,y}x - \omega t)$

(continuité du profil des vagues en  $x=0$ )

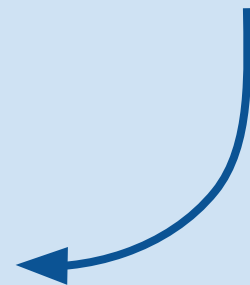
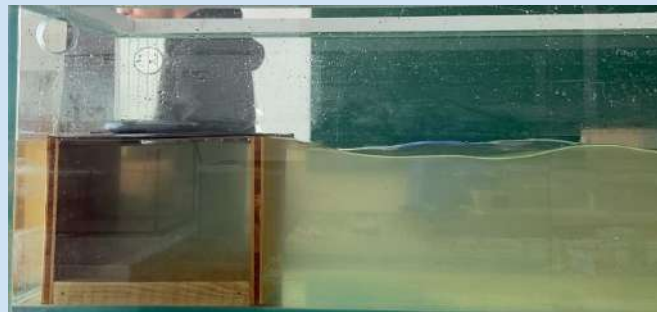
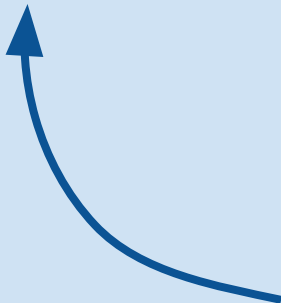
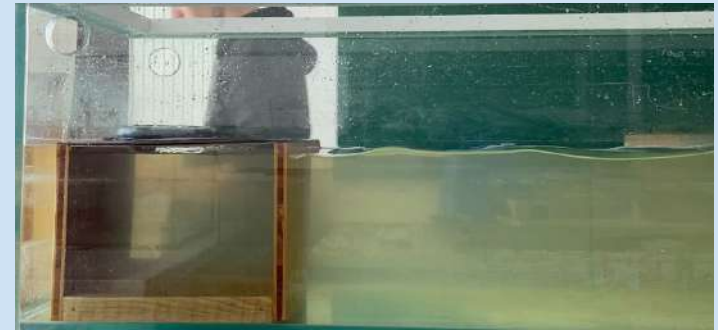
└➔  $A_1 = A_2 \quad k_{1,y} = k_{2,y}$

➔  $\frac{\sin(i)}{\sqrt{H}} = \text{constante}$

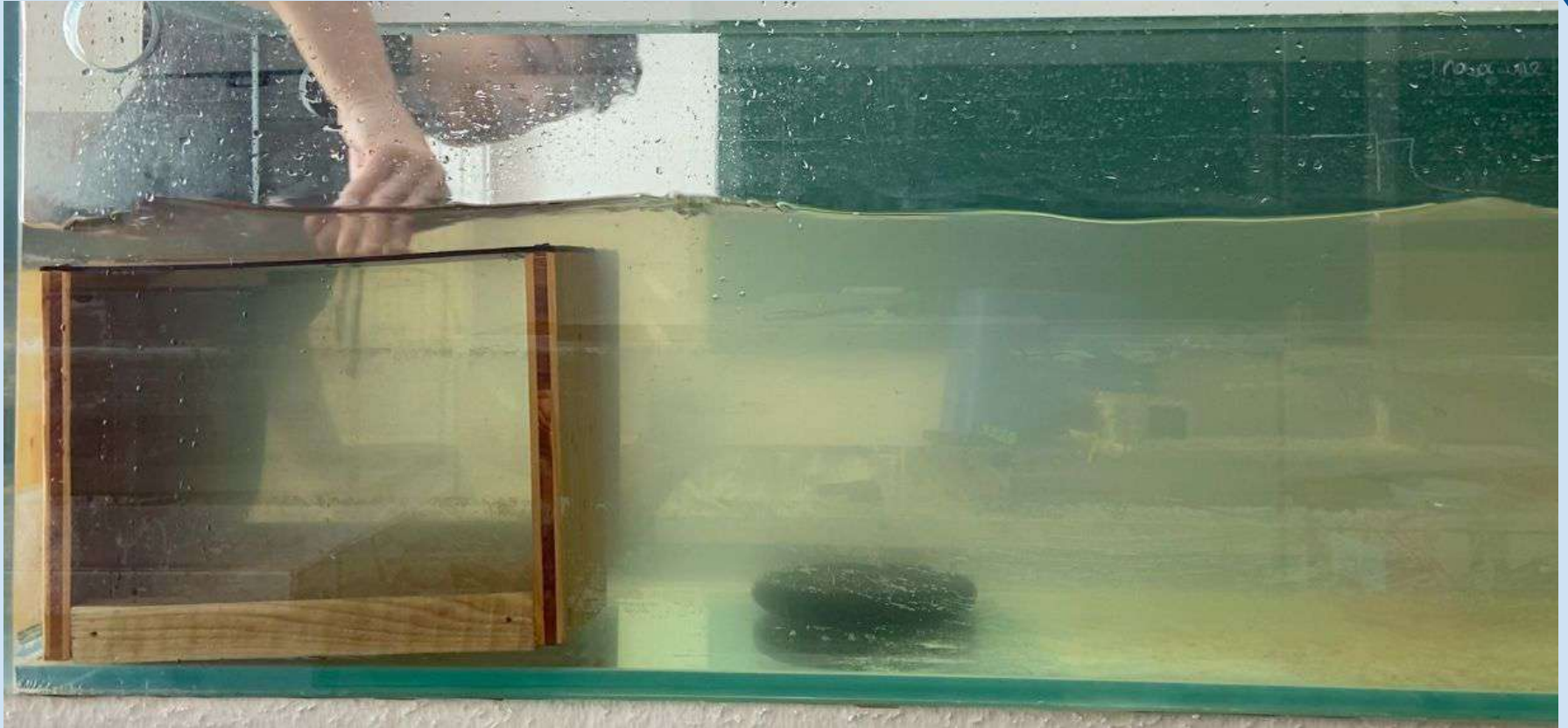




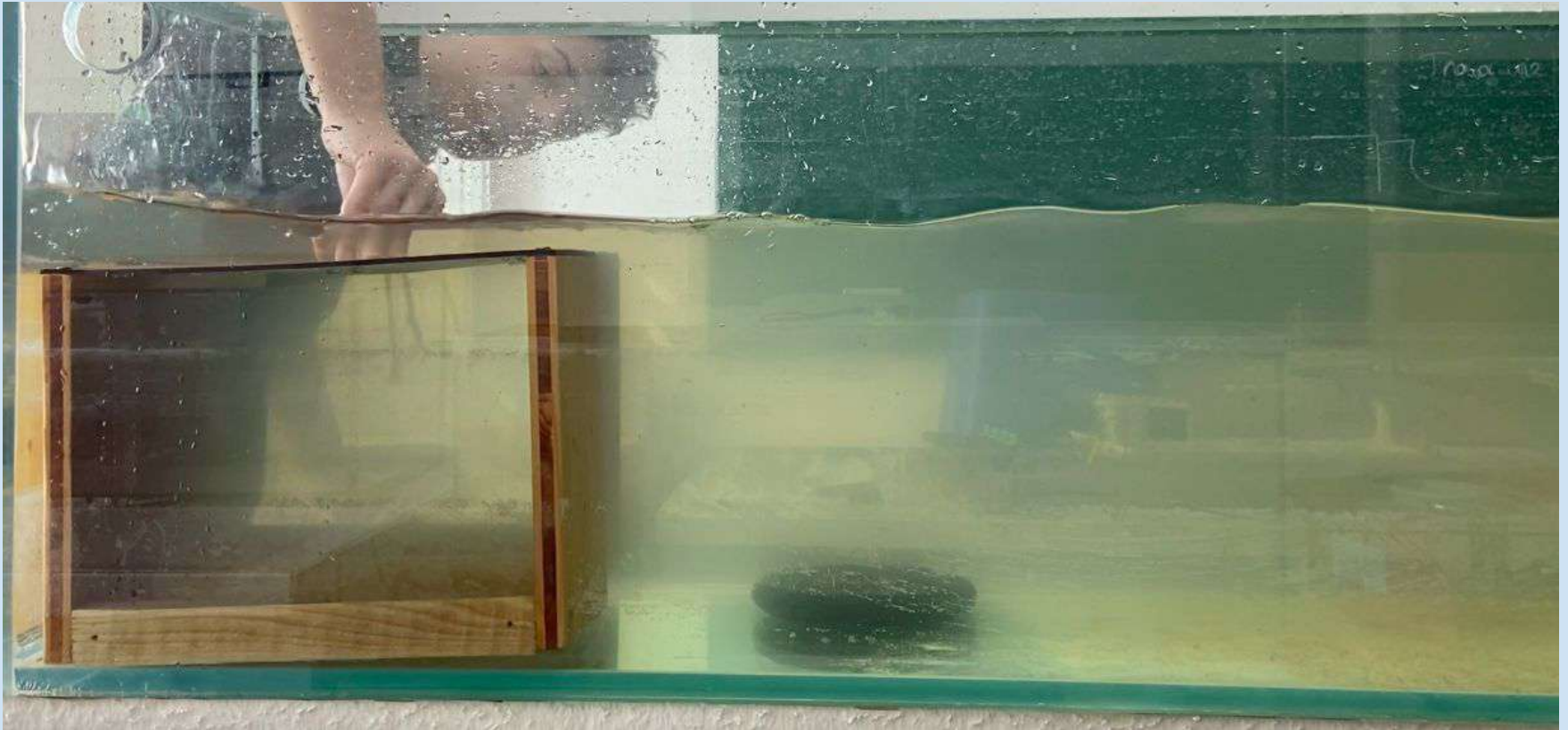
Condition :  
 $H \gg h$



Condition :  $H \ll h$

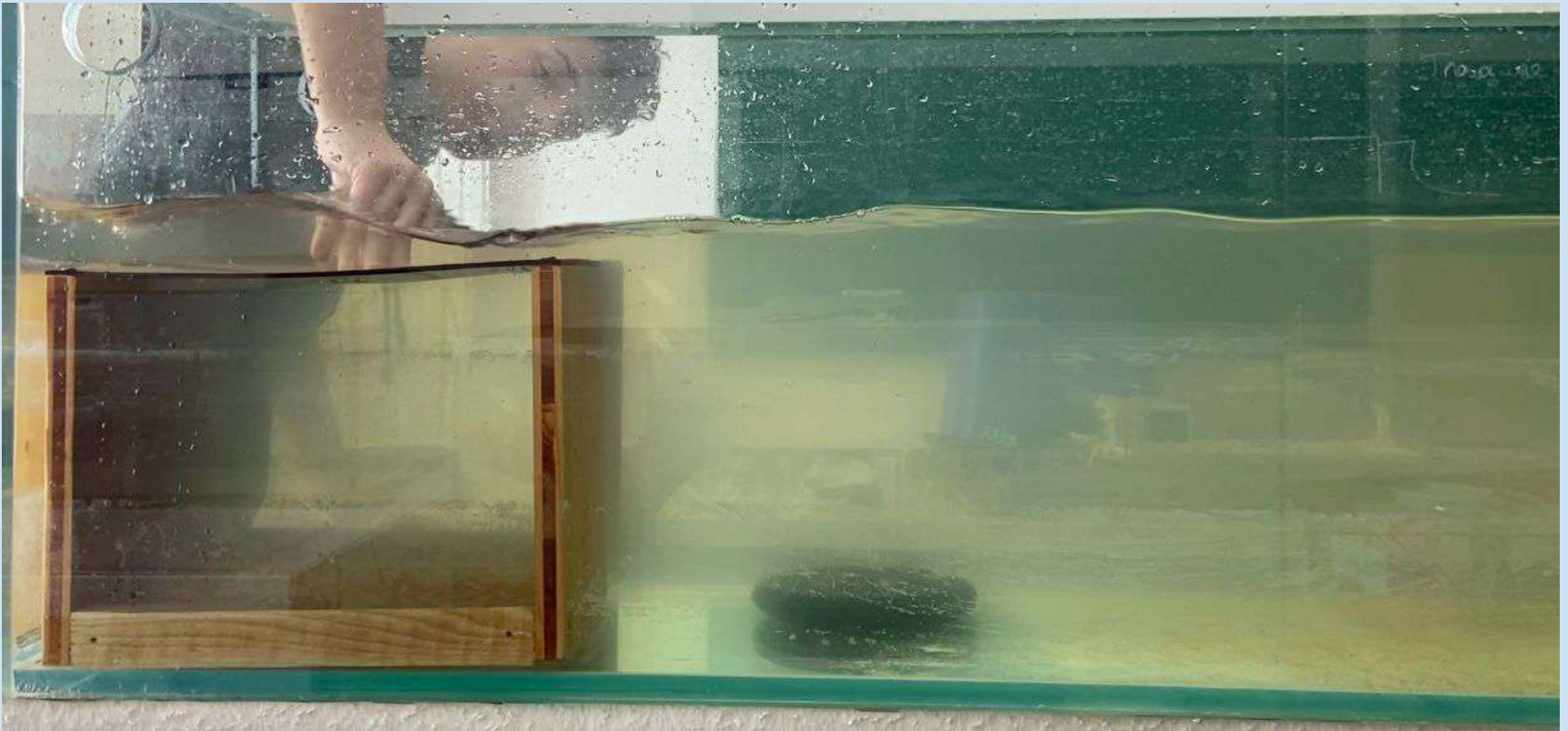


Condition :  $H \ll h$

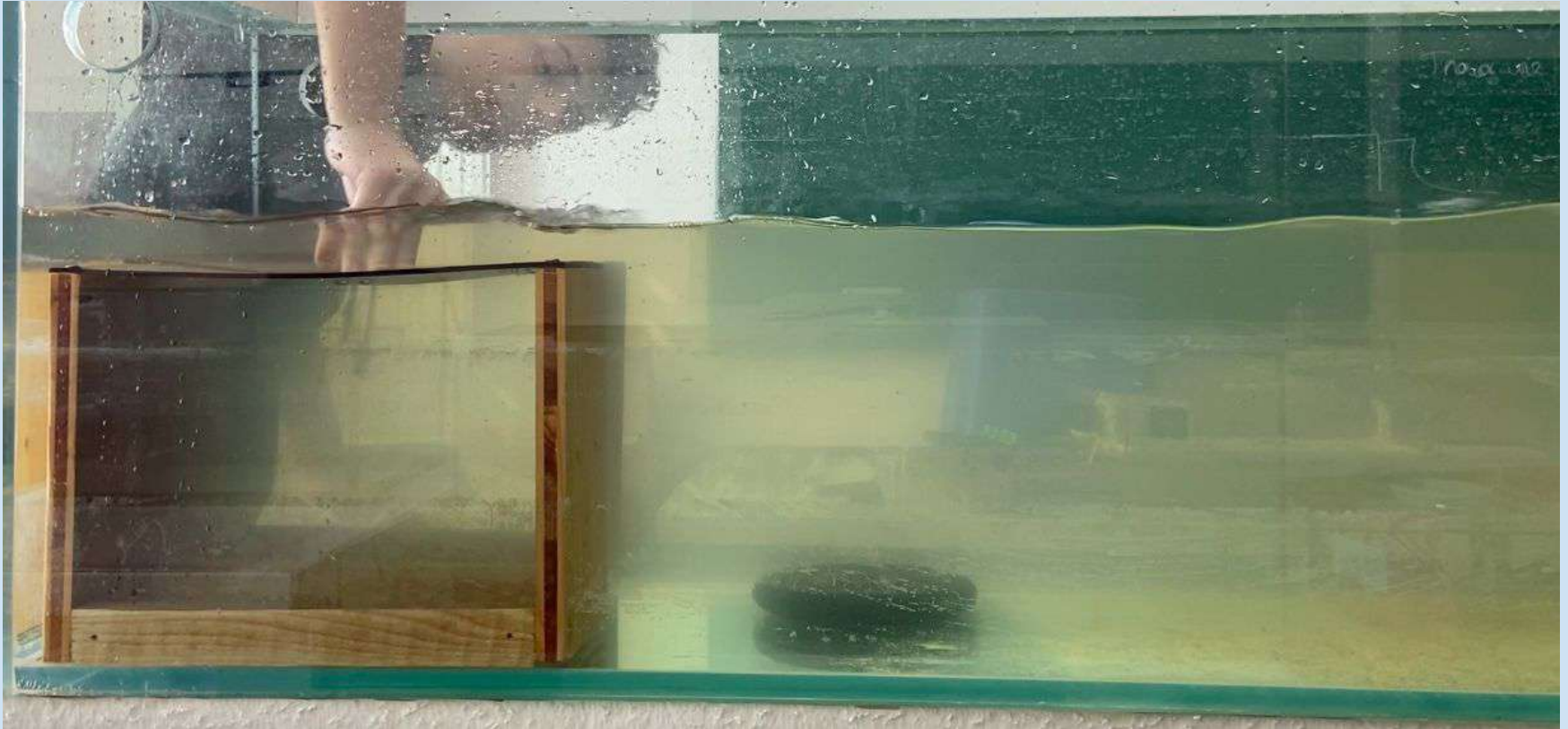




Condition :  $H \ll h$



Condition :  $H \ll h$



Condition :  $H \ll h$

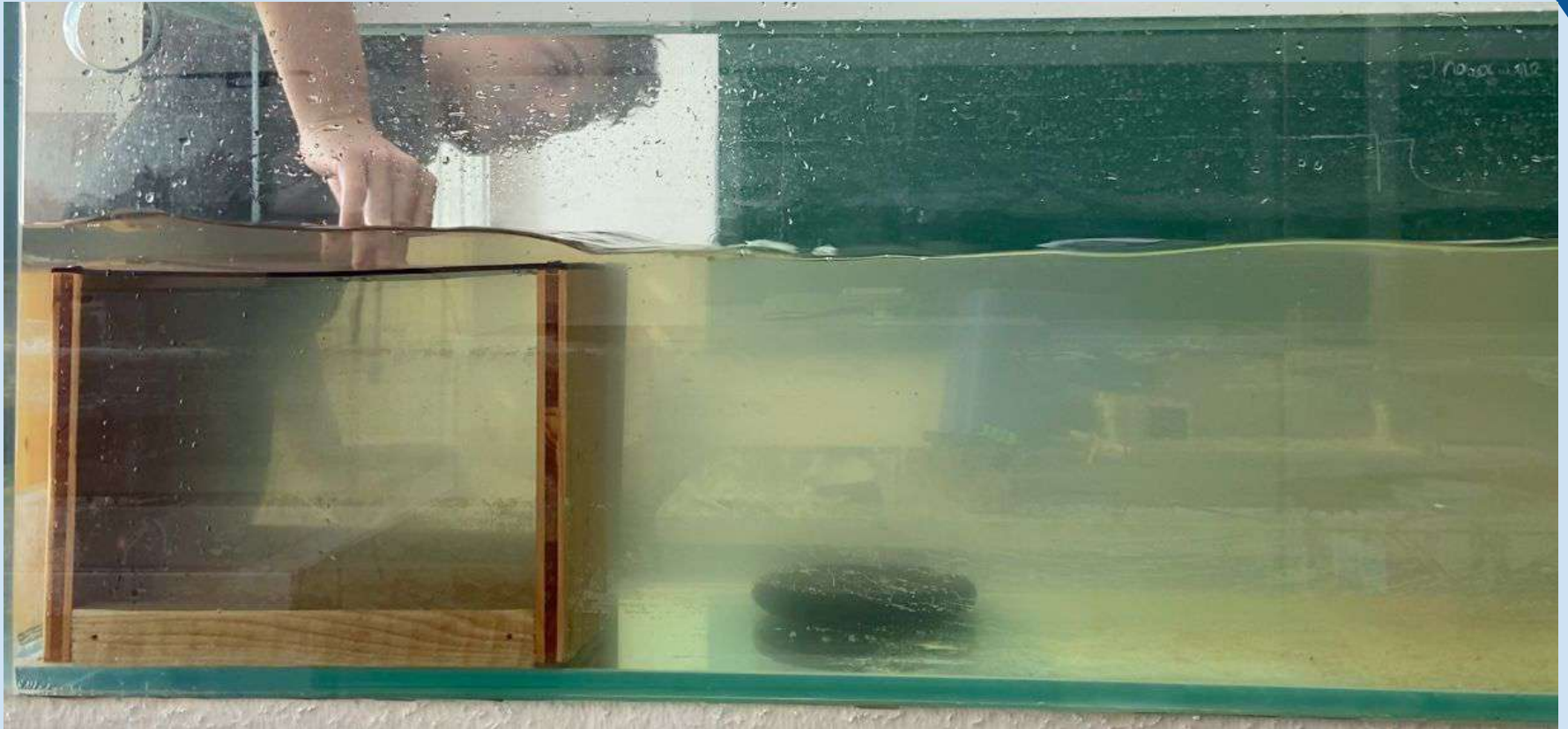




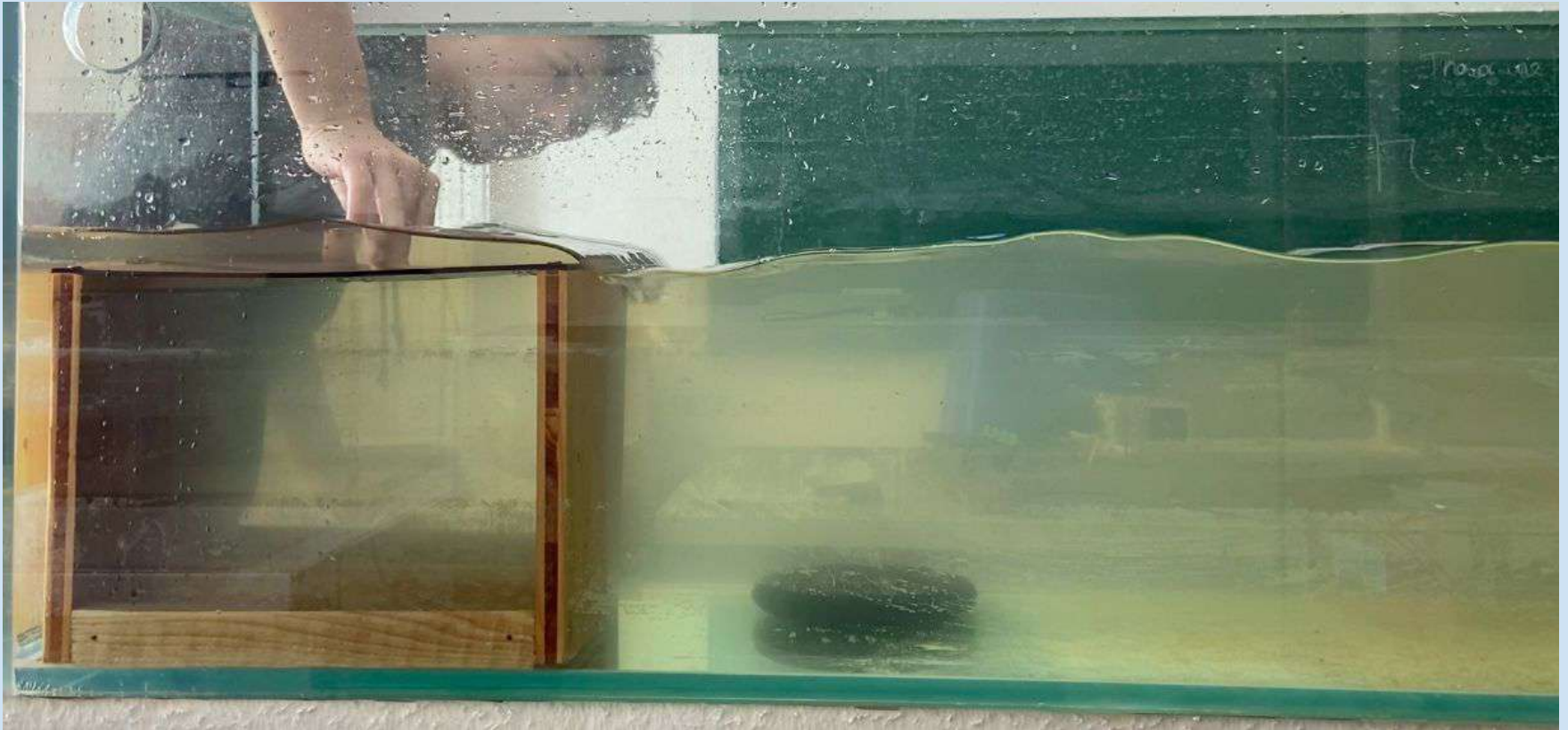
Condition :  $H \ll h$



Condition :  $H \ll h$

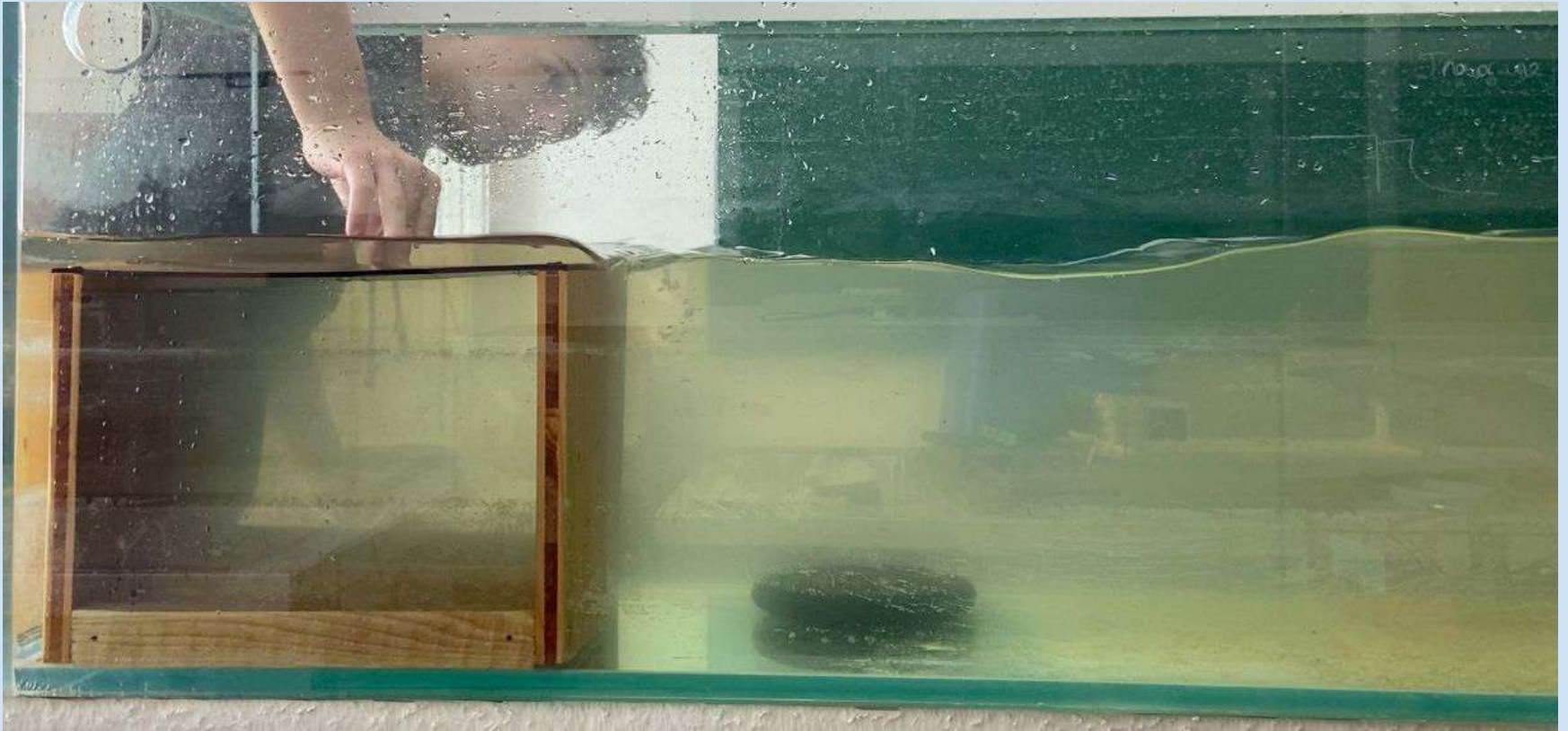


Condition :  $H \ll h$





Condition :  $H \ll h$





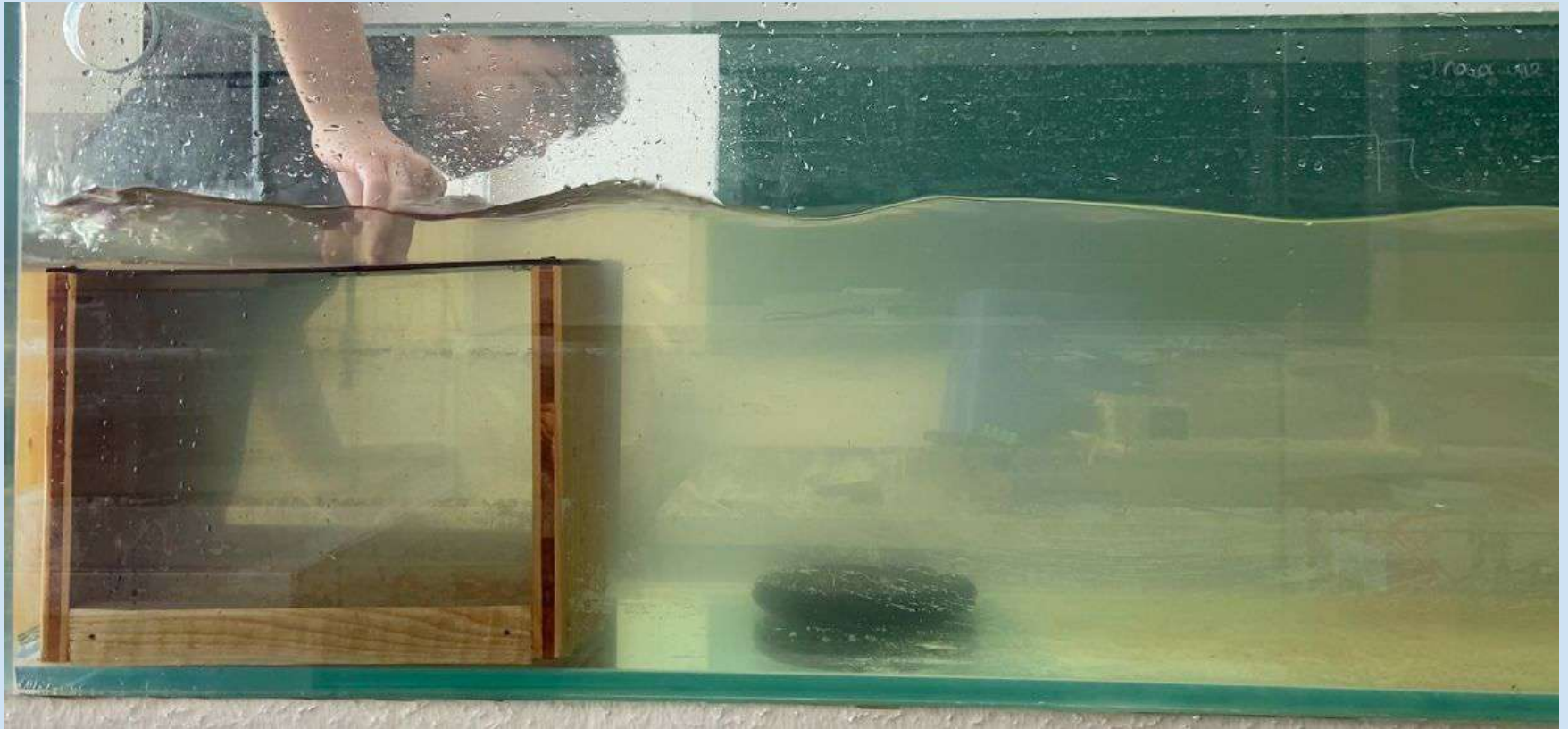
Condition :  $H \ll h$



Condition :  $H \ll h$

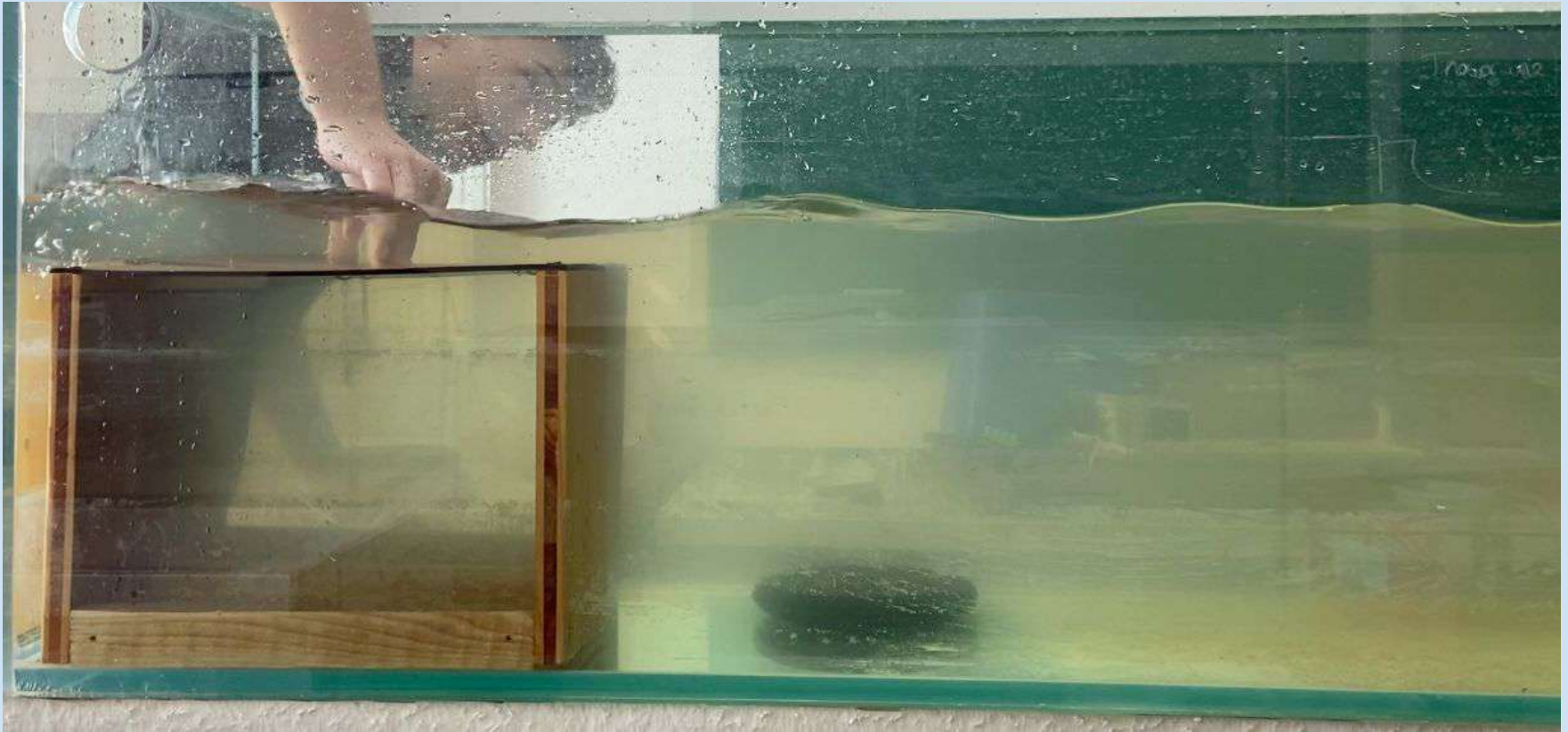


Condition :  $H \ll h$





Condition :  $H \ll h$



## ii. Pente linéaire

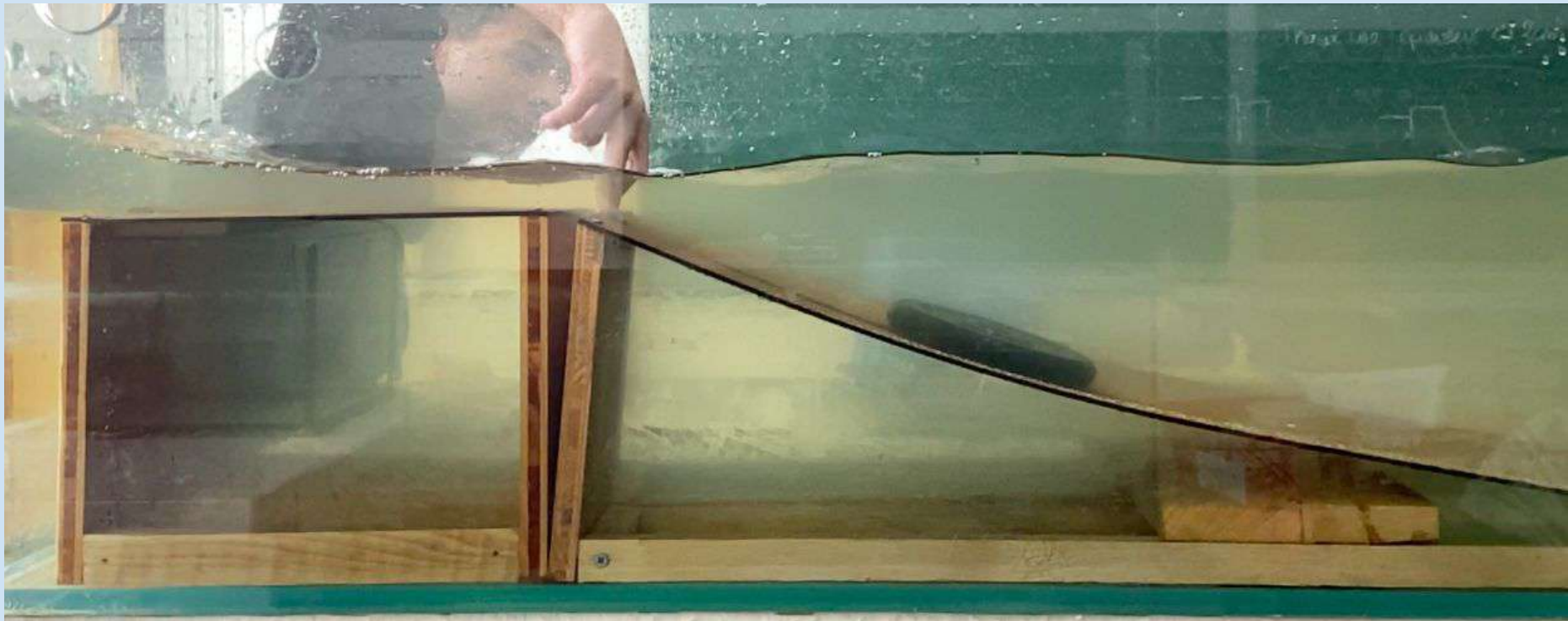
Représentation la plus réaliste  
des dénivelés que l'on peut  
observer sur les plages.



On s'attend donc à l'observation de déferlantes au  
cours du temps.



# Pente bathymétrique d'angle $\alpha = 14^\circ$





Pente bathymétrique d'angle  $\alpha = 14^\circ$

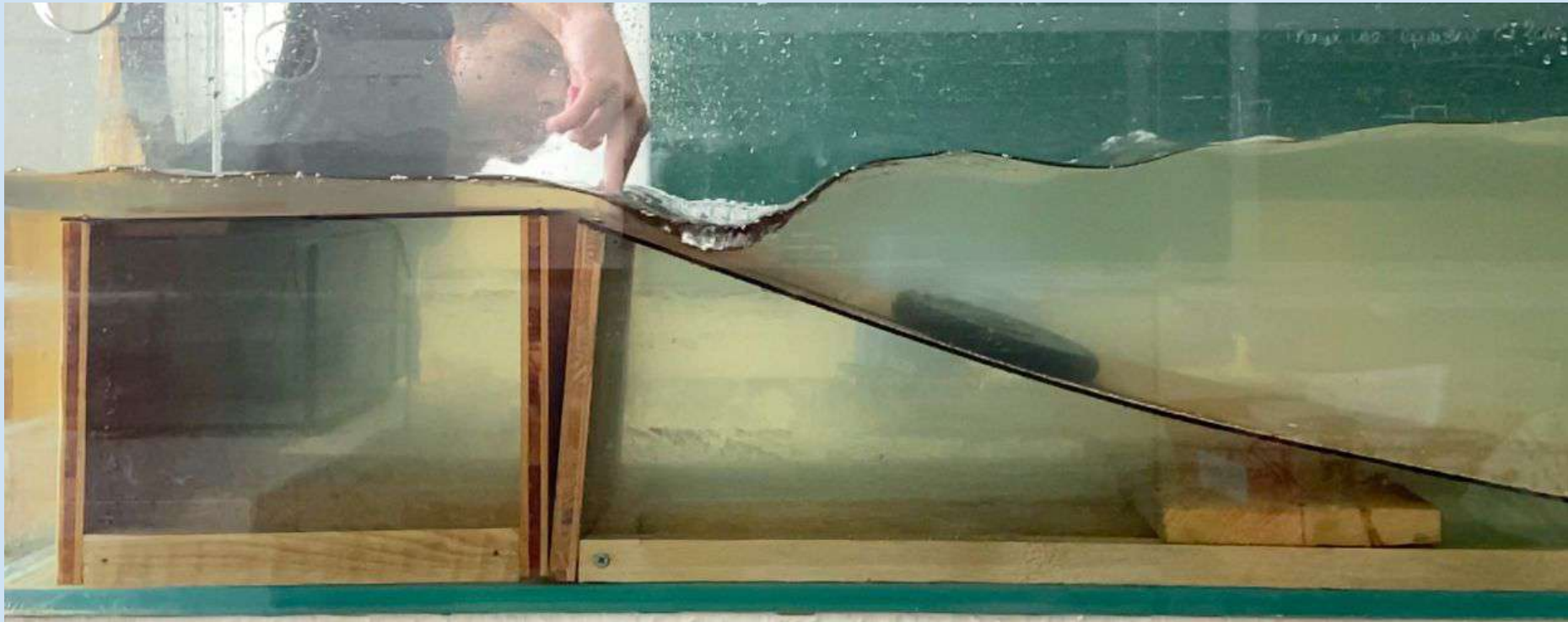




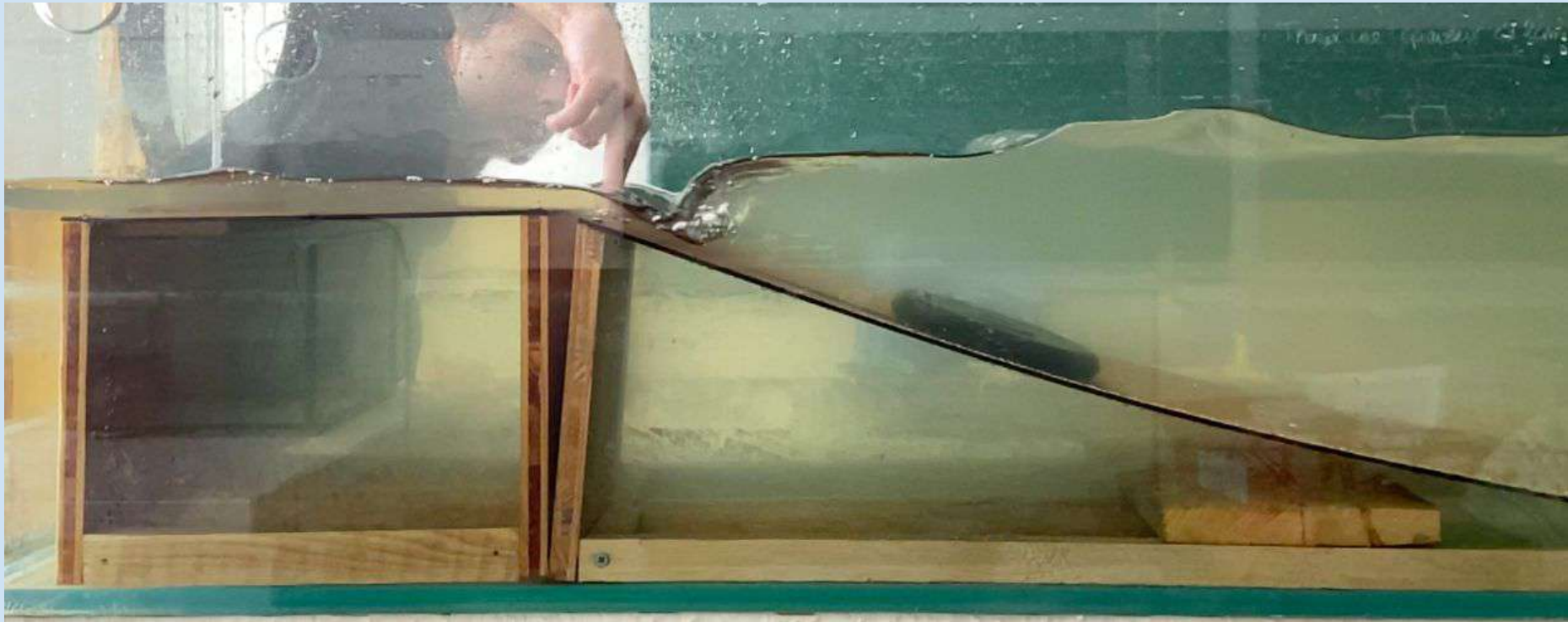
# Pente bathymétrique d'angle $\alpha = 14^\circ$



Pente bathymétrique d'angle  $\alpha = 14^\circ$

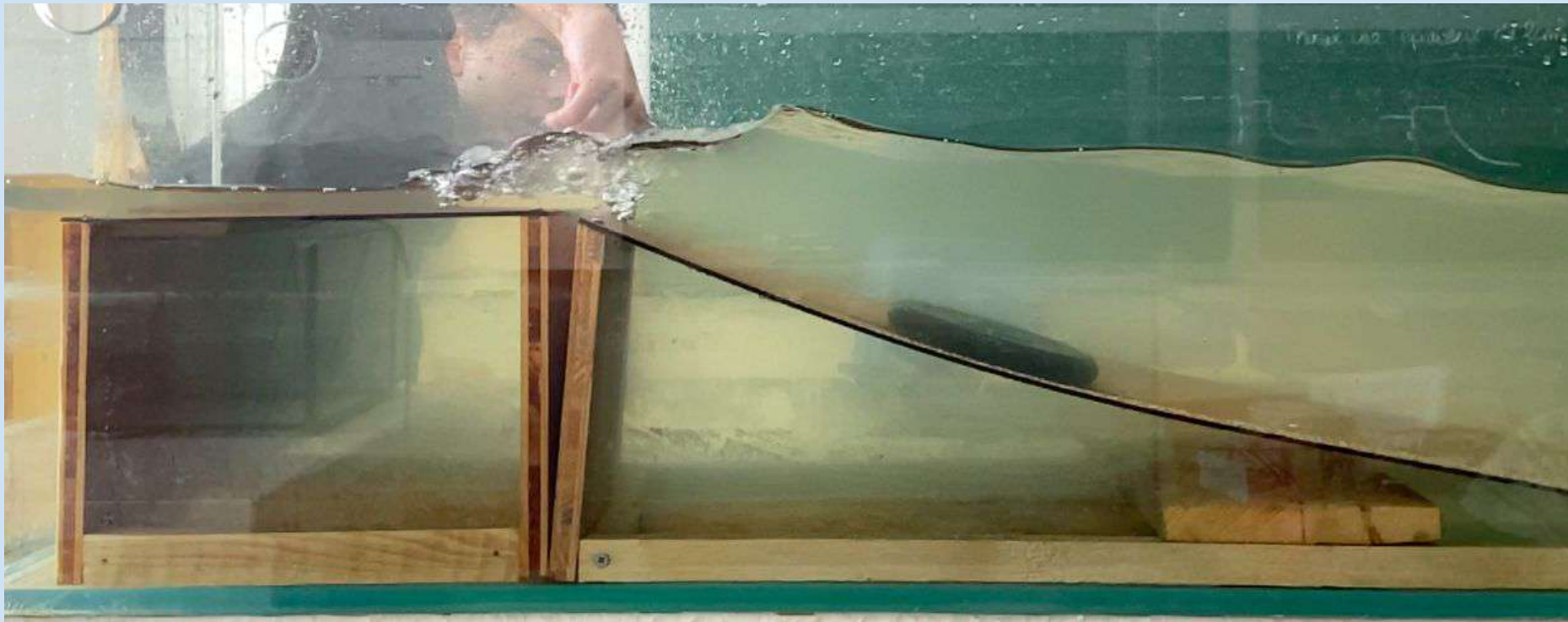


Pente bathymétrique d'angle  $\alpha = 14^\circ$

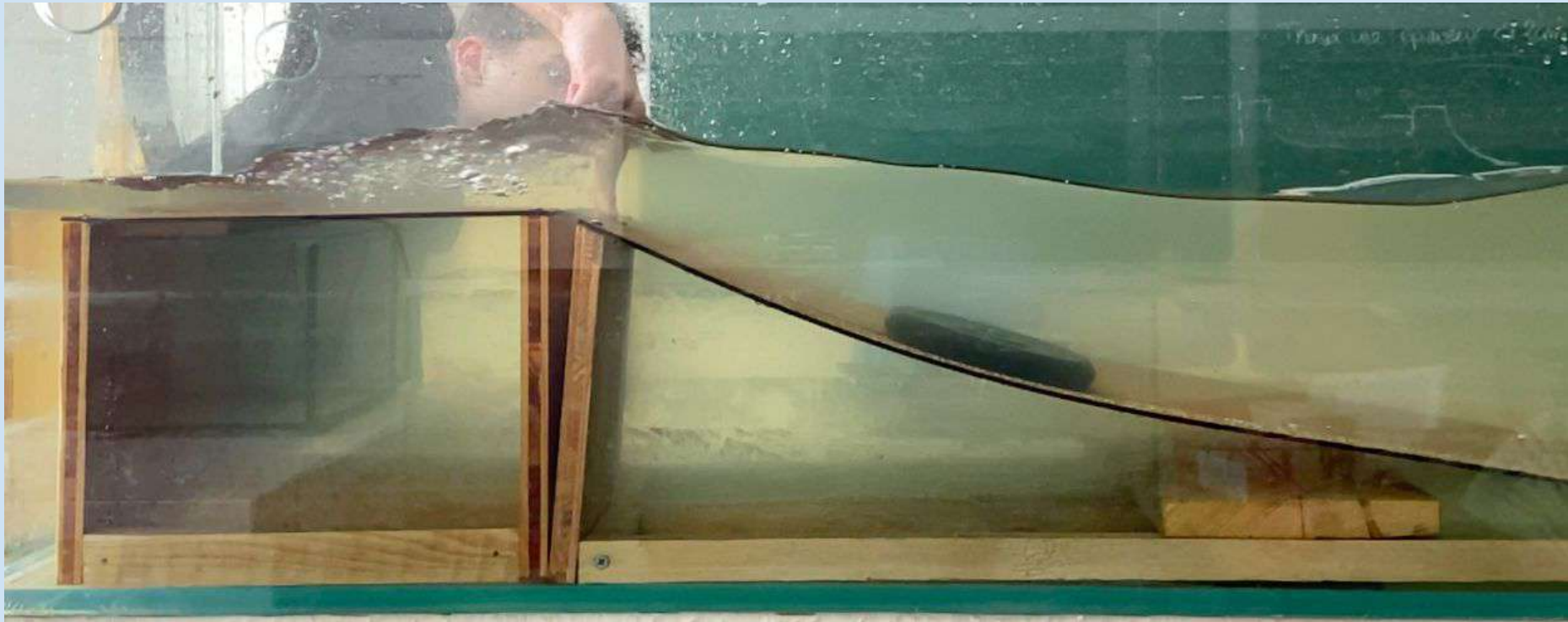




# Pente bathymétrique d'angle $\alpha = 14^\circ$



Pente bathymétrique d'angle  $\alpha = 14^\circ$



Pente bathymétrique d'angle  $\alpha = 14^\circ$





Pente bathymétrique d'angle  $\alpha = 14^\circ$





Pente bathymétrique d'angle  $\alpha = 14^\circ$



Pente bathymétrique d'angle  $\alpha = 14^\circ$



Pente bathymétrique d'angle  $\alpha = 14^\circ$





Pente bathymétrique d'angle  $\alpha = 14^\circ$





Pente bathymétrique d'angle  $\alpha = 14^\circ$



Pente bathymétrique d'angle  $\alpha = 14^\circ$



## Pente bathymétrique d'angle $\alpha = 27^\circ$

- Résultats similaires par rapport à la pente précédente.
- Les déferlantes n'ont pas de très grandes amplitudes mais il arrive qu'une grande dépression survienne  
→ Grande déferlante (Exemple ci-contre).



### iii. Pente progressive

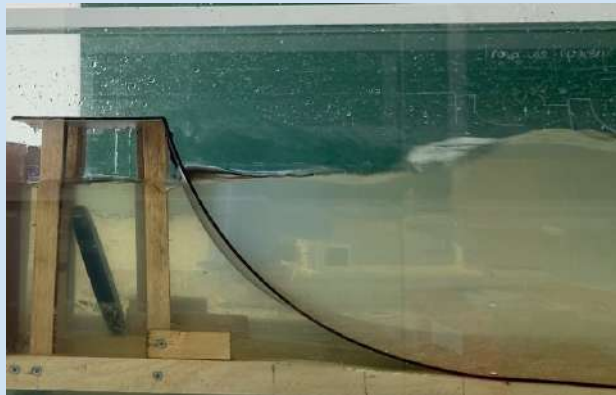
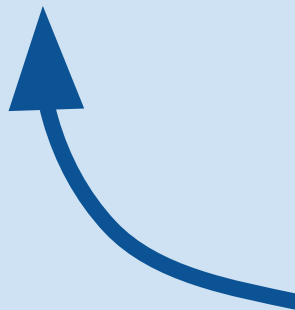
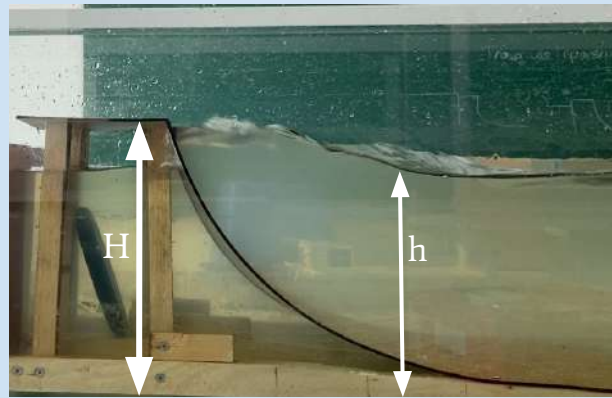
Bathymétrie où l'on possède le moins de données théoriques.

À priori, cela devrait être une pente qui devrait donner de belles déferlantes également.



La partie “bloc” nous redonne la même condition que le bloc bathymétrique.

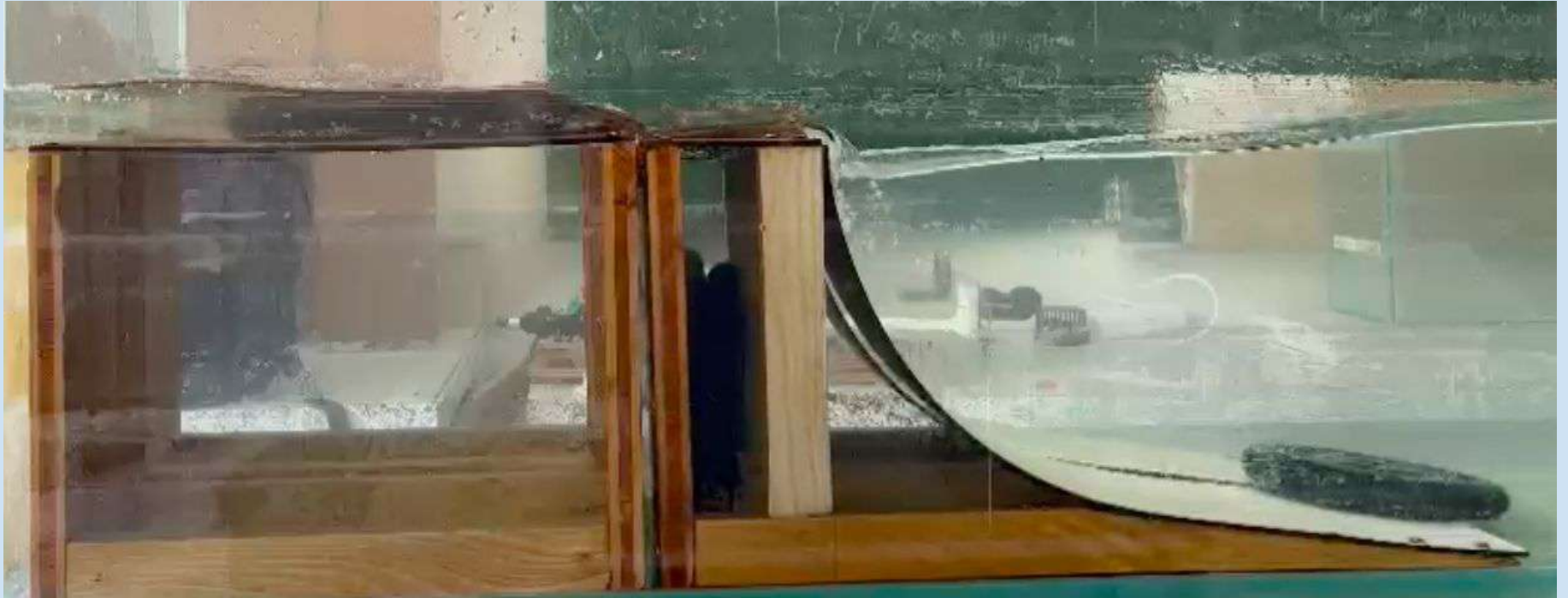




Condition :

$$H \gg h$$

## Pente bathymétrique progressive



# Pente bathymétrique progressive



# Pente bathymétrique progressive





# Pente bathymétrique progressive



# Pente bathymétrique progressive



# Pente bathymétrique progressive



# Pente bathymétrique progressive





# Pente bathymétrique progressive



# Pente bathymétrique progressive



# Pente bathymétrique progressive





# Pente bathymétrique progressive





# Résultats

<i><b>Pente</b></i>	<i><b>Hauteur</b></i>	<i><b>Amplitude</b></i>	<i><b>Longueur d'onde</b></i>	<i><b>Angle</b></i>	<i><b>Déferlement</b></i>
<i><b>Bloc</b></i>	24 cm			X	Non
<i><b>Pente linéaire</b></i>	26 cm	0,9/2,9 cm	72/59 cm	27°	Oui/Oui
<i><b>Pente linéaire</b></i>	26 cm	0,8/1,5 cm	80/68 cm	14°	Oui/Oui
<i><b>Pente progressive</b></i>	26 cm			X	Non
<i><b>Pente Progressive</b></i>	28 cm	1,1/1,6 cm	86/59 cm	X	Oui/Oui

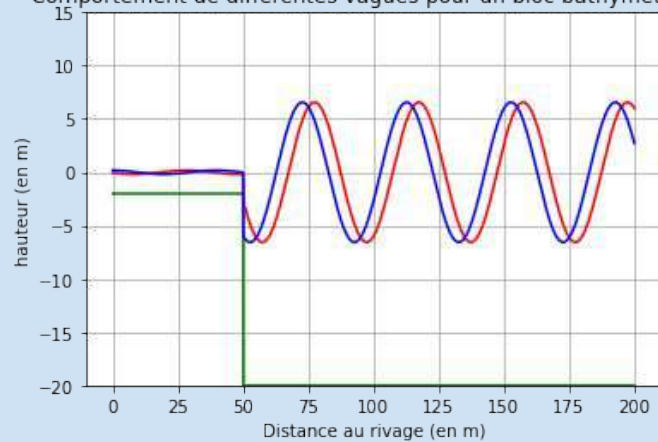
• Résultats surlignés → Meilleurs résultats

# CONFRONTATION INFO

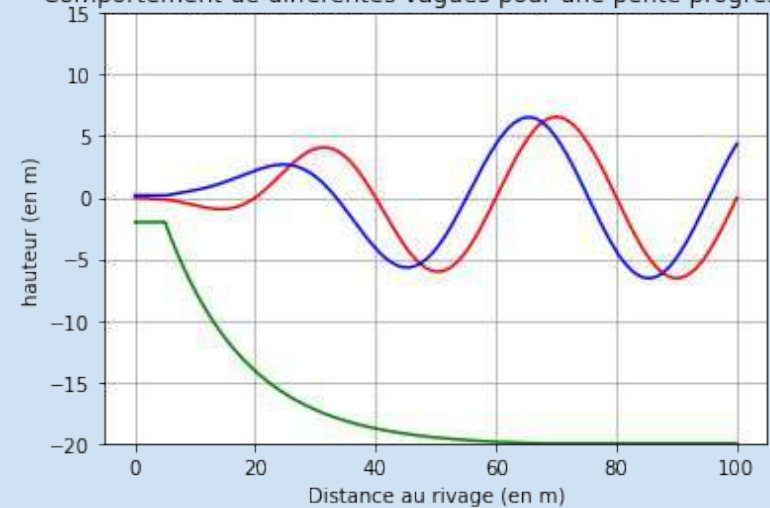
## IV

### Modélisation des vagues par python

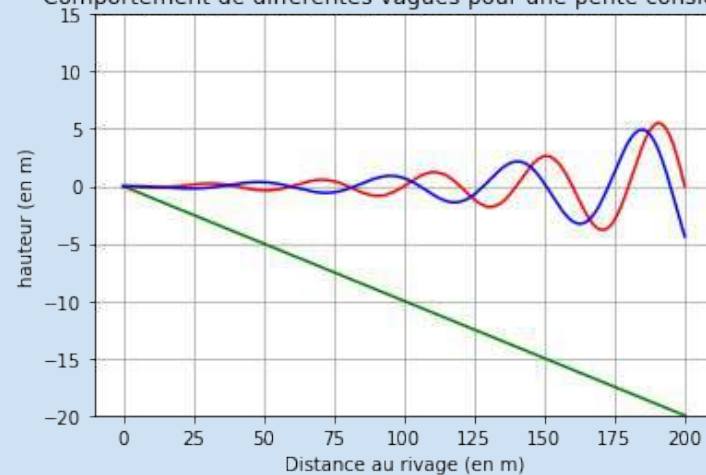
Comportement de différentes vagues pour un bloc bathymétrique



Comportement de différentes vagues pour une pente progressive

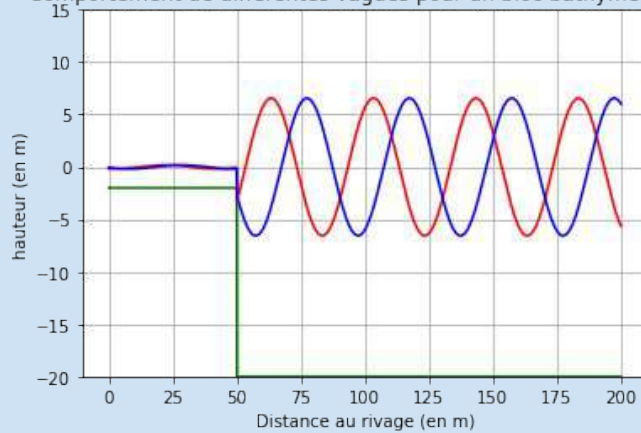


Comportement de différentes vagues pour une pente considérée

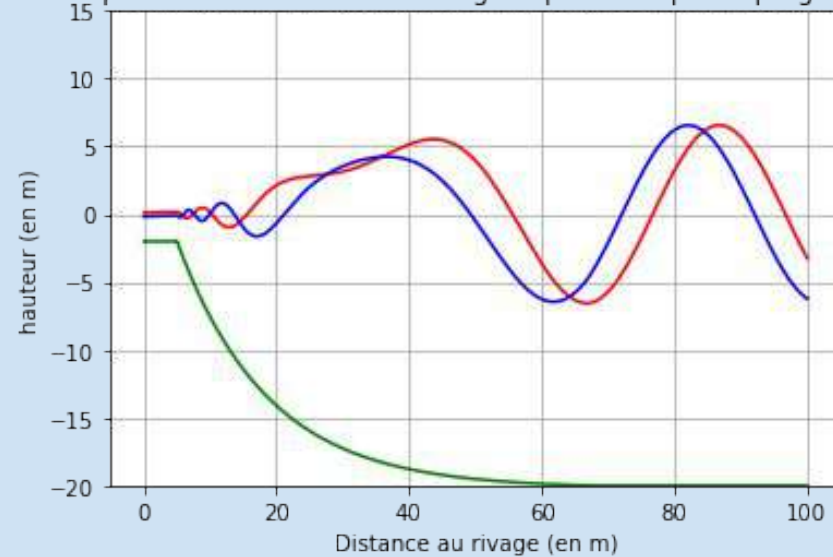


## Preuve de la présence ou de l'absence de déferlements

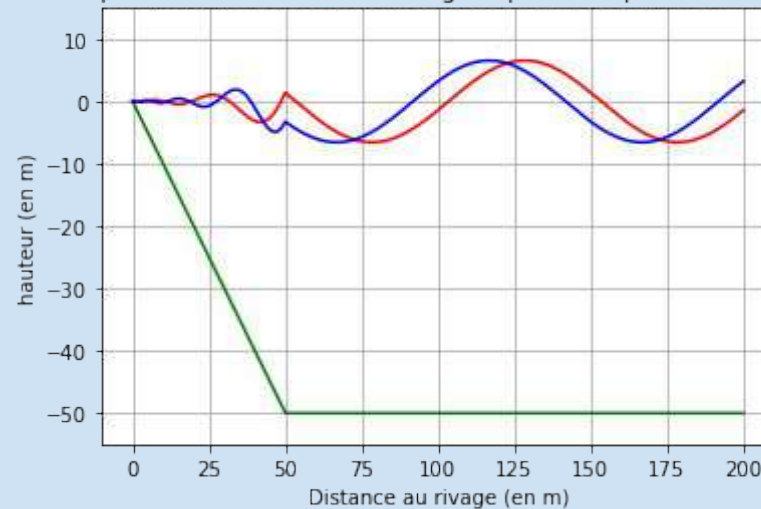
Comportement de différentes vagues pour un bloc bathymétrique



Comportement de différentes vagues pour une pente progressive



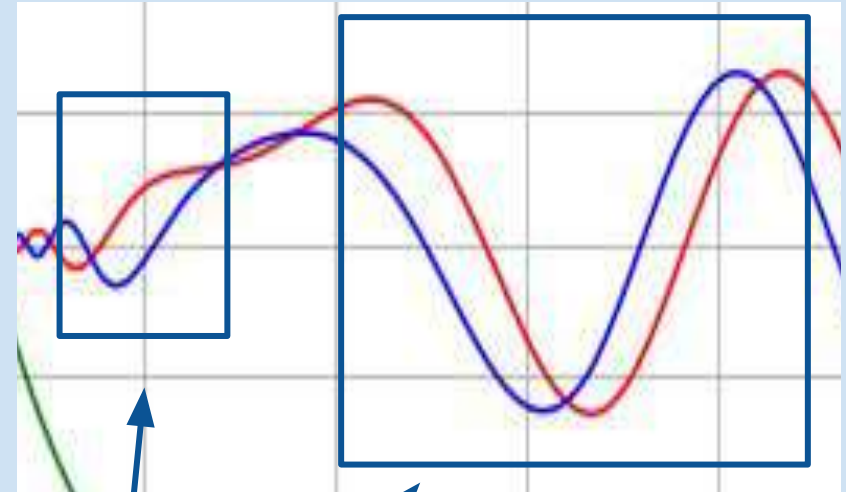
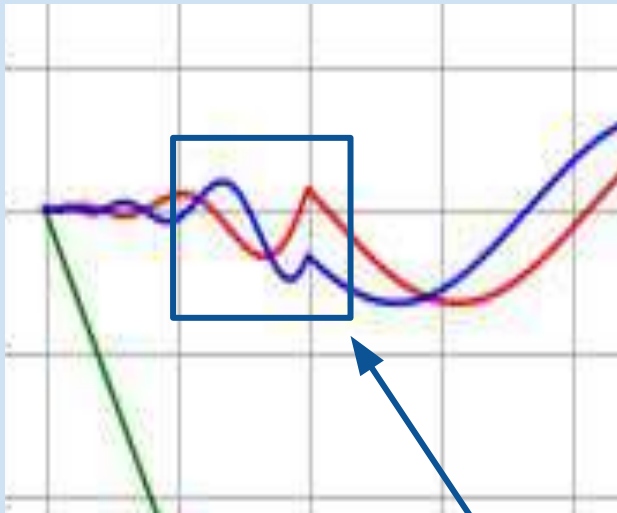
Comportement de différentes vagues pour une pente considérée



# CONFRONTATION INFO

# IV

Preuve de la présence ou de l'absence de déferlements



Présence d'une dépression  
→ Déferlement



# CONCLUSION

V

- Bien que les expériences ne soient pas réalisées dans des conditions optimales, les résultats obtenus sont ceux attendus.
- Les déferlantes influencent les performances des surfeurs, c'est un fait : Les conditions d'obtentions de ces dernières dépendent majoritairement du développement bathymétrique des côtes.
- Les pentes progressives ainsi que les pentes linéaires à fort dénivelés sont celles qui permettent d'avoir une meilleure déferlante, ce qui explique l'attraction de site comme Taïwan ou Nazaret.

# POSSIBLES AMÉLIORATIONS

V

## Pour les expériences déjà réalisées :

Mettre une mousse à la fin du bassin.

Réaliser un moteur performant qui aurait permis la réalisation de vagues plus harmoniques.

Pointage pour observer les variations de hauteur.

Réalisation d'un grand nombre de maquette pour comprendre l'influence de l'angle (pour la pente linéaire).

## Expériences non réalisées

Modéliser les vagues mécaniquement ou électriquement (par paquet d'onde).

Essayer de découvrir la théorie des vagues non linéaires.

Comparaison eau profonde/eau peu profonde.

# ANNEXES

Modélisation sur python de  
la propagation de la vague  
pour une pente  
bathymétrique linéaire  
continue

```
1 import matplotlib.pyplot as plt
2 import math as mp
3
4 #Définition de grandeurs importantes
5 g=9.81
6 a=-0.1
7 lbda=40
8 A=1
9
10 #élaboration des fonctions qui vont nous permettre de faire les différents tracés
11 def hauteur(A,H,lbda,x,t):
12     k=(2*mp.pi)/lbda
13     w=(g*k*(-H))**0.5
14     eta=-A*(w/g)*mp.cosh(k*H)*mp.sin(x*k-w*t)
15     return eta
16
17 def pente_bathymétrique(x):
18     if a*x>20:
19         return a*x
20     else:
21         return -20
22
23 #élaboration des listes
24 j=0
25 hauteur_vagues1=[]
26 hauteur_vagues2=[]
27 pente=[]
28 x=[]
29 while j<=200:
30     hauteur_vagues1.append(hauteur(A,pente_bathymétrique(j),lbda,j,0))
31     hauteur_vagues2.append(hauteur(A,pente_bathymétrique(j),lbda,j,1))
32     pente.append(pente_bathymétrique(j))
33     x.append(j)
34     j+=0.01
35
36 #Tracé des courbes
37 plt.plot(x,pente,'g-', label='pente bathymétrique')
38 plt.plot(x,hauteur_vagues1,'r-', label='t=0')
39 plt.plot(x,hauteur_vagues2,'b-', label='t=1')
40 plt.title('Comportement de différentes vagues pour une pente considérée')
41 plt.xlabel('Distance au rivage (en m)')
42 plt.ylabel('hauteur (en m)')
43 plt.ylim(-20,15)
44 plt.grid(True)
45 plt.show()
```

# ANNEXES

Modélisation sur python  
de la propagation d'une  
vague pour un bloc  
bathymétrique  
et une hauteur d'eau  
suffisante

```
1 import matplotlib.pyplot as plt
2 import math as mp
3
4 #Définition de grandeurs importantes
5 g=9.81
6 a=-0.1
7 lbda=40
8 A=1
9
10 #élaboration des fonctions qui vont nous permettre de faire les différents tracés
11 def hauteur(A,H,lbda,x,t):
12     k=(2*mp.pi)/lbda
13     w=(g*k*(-H))**0.5
14     eta=-A*(w/g)*mp.cosh(k*H)*mp.sin(x*k-w*t)
15     return eta
16
17 def bloc_bathymétrique(x):
18     if x<50:
19         return -2
20     else:
21         return -20
22
23 #élaboration des listes
24 j=0
25 hauteur_vagues1=[]
26 hauteur_vagues2=[]
27 bloc=[]
28 x=[]
29 while j<=200:
30     hauteur_vagues1.append(hauteur(A,bloc_bathymétrique(j),lbda,j,7))
31     hauteur_vagues2.append(hauteur(A,bloc_bathymétrique(j),lbda,j,8))
32     bloc.append(bloc_bathymétrique(j))
33     x.append(j)
34     j+=0.01
35
36 #Tracé des courbes
37 plt.plot(x,bloc,'g-', label='bloc bathymétrique')
38 plt.plot(x,hauteur_vagues1,'r-', label='t=0')
39 plt.plot(x,hauteur_vagues2,'b-', label='t=1')
40 plt.title('Comportement de différentes vagues pour un bloc bathymétrique')
41 plt.xlabel('Distance au rivage (en m)')
42 plt.ylabel('hauteur (en m)')
43 plt.ylim(-20,15)
44 plt.grid(True)
45 plt.show()
```

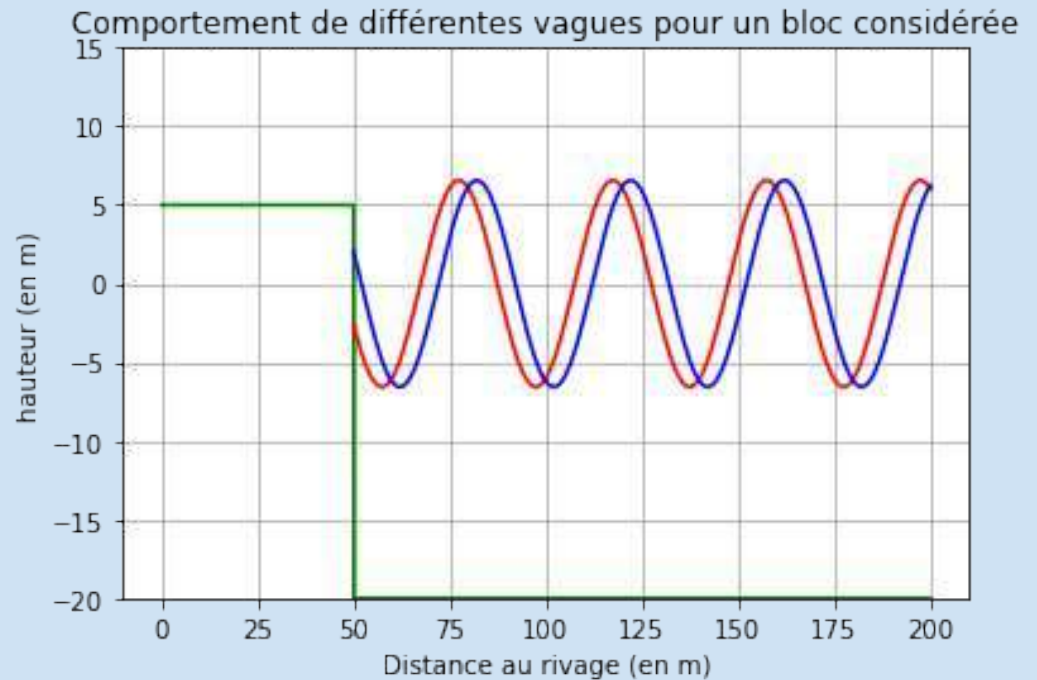
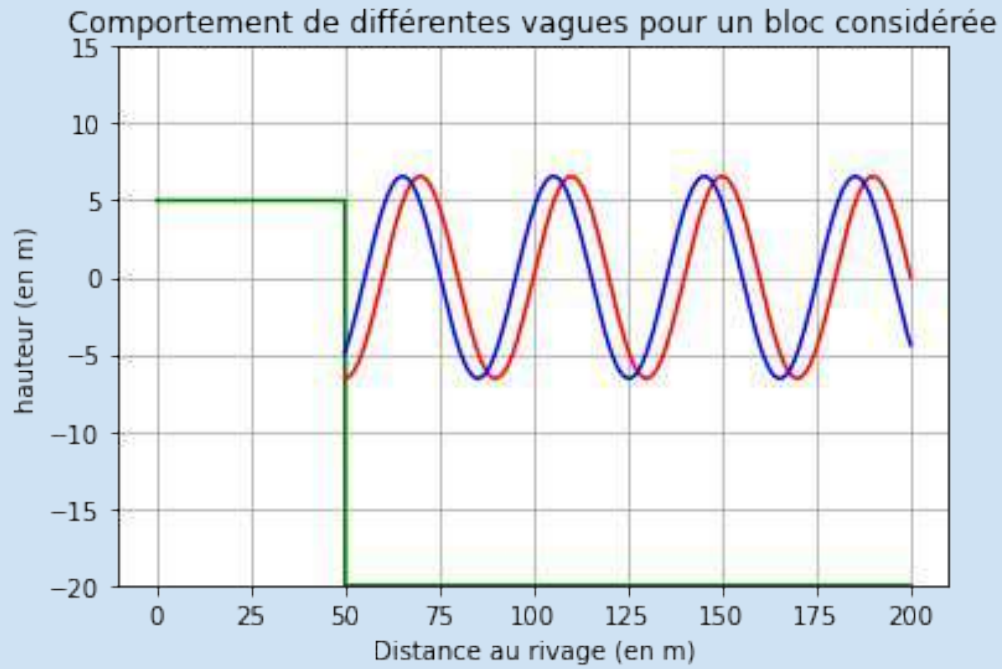


# ANNEXES

Modélisation sur python  
de la propagation d'une  
vague pour une pente  
bathymétrique  
progressive

```
1 import matplotlib.pyplot as plt
2 import math as mp
3
4 #Définition de grandeurs importantes
5 g=9.81
6 a=-0.1
7 lbda=40
8 A=1
9
10 #élaboration des fonctions qui vont nous permettre de faire les différents tracés
11 def hauteur(A,H,lbda,x,t):
12     k=(2*mp.pi)/lbda
13     w=(g*k*(-H))*0.5
14     eta=-A*(w/g)*mp.cosh(k*H)*mp.sin(x*k-w*t)
15     return eta
16
17 def pentep_bathymétrique(x):
18     if x<5:
19         return -2
20     elif 5<=x<66.8:
21         return (3.4*mp.exp((x-28)/-13.7)-20.2)
22     else:
23         return -20
24
25 #élaboration des listes
26 j=0
27 hauteur_vagues1=[]
28 hauteur_vagues2=[]
29 pentep=[]
30 x=[]
31 while j<=100:
32     hauteur_vagues1.append(hauteur(A,pentep_bathymétrique(j),lbda,j,0))
33     hauteur_vagues2.append(hauteur(A,pentep_bathymétrique(j),lbda,j,1))
34     pentep.append(pentep_bathymétrique(j))
35     x.append(j)
36     j+=0.001
37
38 #Tracé des courbes
39 plt.plot(x,pentep,'g-', label='pente bathymétrique progressive')
40 plt.plot(x,hauteur_vagues1,'r-', label='t=0')
41 plt.plot(x,hauteur_vagues2,'b-', label='t=1')
42 plt.title('Comportement de différentes vagues pour une pente progressive')
43 plt.xlabel('Distance au rivage (en m)')
44 plt.ylabel('hauteur (en m)')
45 plt.ylim(-20,15)
46 plt.grid(True)
47 plt.show()
```

# ANNEXES



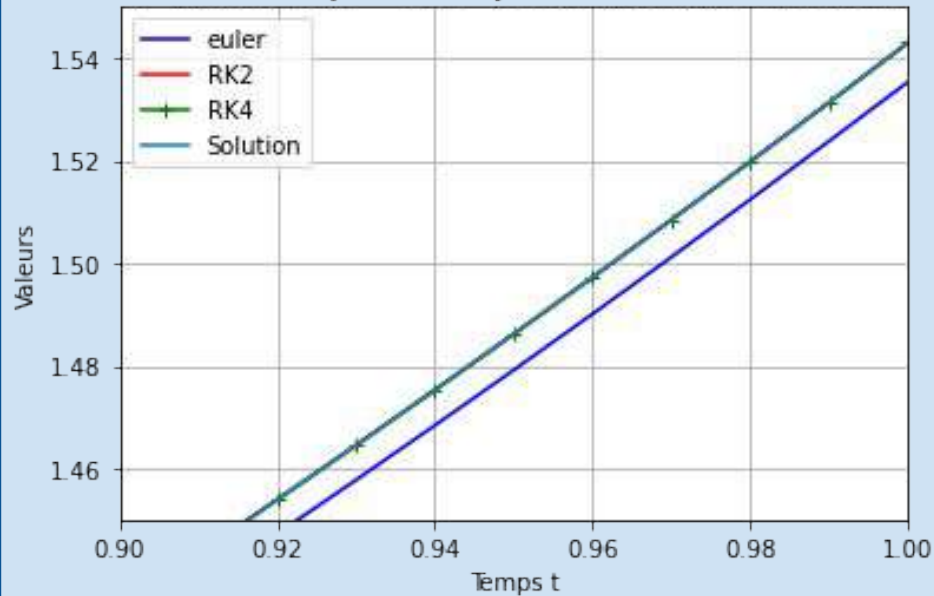
# ANNEXES

Modélisation de la propagation des vagues pour un bloc bathymétrique qui ne respecte pas la condition de déferlement pour une vague (on a  $H > h$ )

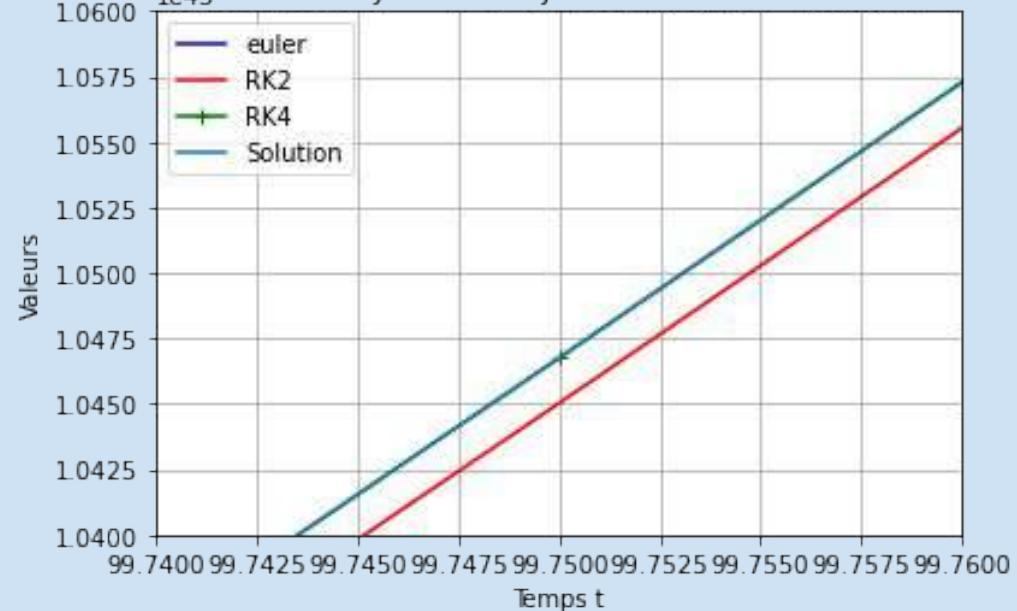
```
1 import matplotlib.pyplot as plt
2 import math as mp
3
4 #Définition de grandeurs importantes
5 g=9.81
6 a=-0.1
7 lbda=40
8 A=1
9
10 #élaboration des fonctions qui vont nous permettre de faire les différents tracés
11 def hauteur(A,H,lbda,x,t):
12     k=(2*mp.pi)/lbda
13     w=(g*k*(-H))**0.5
14     eta=-A*(w/g)*mp.cosh(k*H)*mp.sin(x*k-w*t)
15     return eta
16
17 def bloc_bathymétrique(x):
18     if x<50:
19         return 5
20     else:
21         return -20
22
23 #élaboration des listes
24 j=50
25 i=0
26 hauteur_vagues1=[]
27 hauteur_vagues2=[]
28 bloc=[]
29 x=[]
30 x1=[]
31 while j<=200:
32     hauteur_vagues1.append(hauteur(A,bloc_bathymétrique(j),lbda,j,0))
33     hauteur_vagues2.append(hauteur(A,bloc_bathymétrique(j),lbda,j,1))
34     x.append(j)
35     j+=0.01
36 while i<=200:
37     bloc.append(bloc_bathymétrique(i))
38     x1.append(i)
39     i+=0.01
40
41 #Tracé des courbes
42 plt.plot(x1,bloc,'g-', label='bloc bathymétrique')
43 plt.plot(x,hauteur_vagues1,'r-', label='t=0')
44 plt.plot(x,hauteur_vagues2,'b-', label='t=1')
45 plt.title('Comportement de différentes vagues pour un bloc considérée')
46 plt.xlabel('Distance au rivage (en m)')
47 plt.ylabel('hauteur (en m)')
48 plt.ylim(-20,15)
49 plt.grid(True)
50 plt.show()
```

# Résolution par python d'une équation du second degré (Euler et Runge-Kutta)

Résolution de  $y'' = k^2 * y$  avec différentes méthodes



Résolution de  $y'' = k^2 * y$  avec différentes méthodes





# ANNEXES

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import math as mp
4
5 def euler(y1_0, y2_0, k, t0, t_end, dt):
6     """
7     Résout l'équation différentielle  $y'' = k^2 y$  en utilisant la méthode d'Euler.
8
9     Paramètres :
10    y1_0 : valeur initiale de y
11    y2_0 : valeur initiale de y'
12    k : constante de l'équation différentielle
13    t0 : temps initial
14    t_end : temps final
15    dt : pas de temps
16
17    Retourne :
18    t_values : valeurs du temps
19    y1_values : valeurs de y
20    y2_values : valeurs de y'
21    """
22    n = int((t_end - t0) / dt) + 1
23    t_values = np.linspace(t0, t_end, n)
24    y1_values = np.zeros(n)
25    y2_values = np.zeros(n)
26    y1_values[0] = y1_0
27    y2_values[0] = y2_0
28    for i in range(1, n):
29        y1_values[i] = y1_values[i-1] + dt * y2_values[i-1]
30        y2_values[i] = y2_values[i-1] + dt * k**2 * y1_values[i-1]
31
32    return t_values, y1_values, y2_values
```

```
122 # Paramètres de l'équation
123 k = 1 # Constante k
124 y1_0 = 1.0 # Condition initiale pour y
125 y2_0 = 0.0 # Condition initiale pour y'
126 t0 = 0.0 # Temps initial
127 t_end = 1.0 # Temps final
128 dt = 0.01 # Pas de temps
129
130 # Résolution de l'équation différentielle
131 t_values, y1_values, y2_values = euler(y1_0, y2_0, k, t0, t_end, dt)
132 t_valuesRK2, y1_valuesRK2, y2_valuesRK2 = rk2(y1_0, y2_0, k, t0, t_end, dt)
133 t_valuesRK4, y1_valuesRK4, y2_valuesRK4 = rk4(y1_0, y2_0, k, t0, t_end, dt)
134
135 # Définition de la fonction qui est solution
136 y3 = []
137 for i in range(len(t_values)):
138     y3.append(mp.cosh(t_values[i] * (k**2)))
139
```

# ANNEXE



```
34 def rk2(y1_0, y2_0, k, t0, t_end, dt):
35     """
36     Résout l'équation différentielle  $y'' = k^2 y$ 
37     en utilisant la méthode de Runge-Kutta d'ordre 2.
38
39     Paramètres :
40     y1_0 : valeur initiale de y
41     y2_0 : valeur initiale de y'
42     k : constante de l'équation différentielle
43     t0 : temps initial
44     t_end : temps final
45     dt : pas de temps
46
47     Retourne :
48     t_values : valeurs du temps
49     y1_values : valeurs de y
50     y2_values : valeurs de y'
51     """
52     n = int((t_end - t0) / dt) + 1
53     t_values = np.linspace(t0, t_end, n)
54     y1_values = np.zeros(n)
55     y2_values = np.zeros(n)
56     y1_values[0] = y1_0
57     y2_values[0] = y2_0
58     for i in range(1, n):
59         t = t_values[i-1]
60         y1 = y1_values[i-1]
61         y2 = y2_values[i-1]
62
63         k1_y1 = y2 * dt
64         k1_y2 = (k**2 * y1) * dt
65
66         k2_y1 = (y2 + k1_y2 / 2) * dt
67         k2_y2 = (k**2 * (y1 + k1_y1 / 2)) * dt
68
69         y1_values[i] = y1 + k2_y1
70         y2_values[i] = y2 + k2_y2
71
72     return t_values, y1_values, y2_values
```

```
74 def rk4(y1_0, y2_0, k, t0, t_end, dt):
75     """
76     Résout l'équation différentielle  $y'' = k^2 y$ 
77     en utilisant la méthode de Runge-Kutta d'ordre 4.
78
79     Paramètres :
80     y1_0 : valeur initiale de y
81     y2_0 : valeur initiale de y'
82     k : constante de l'équation différentielle
83     t0 : temps initial
84     t_end : temps final
85     dt : pas de temps
86
87     Retourne :
88     t_values : valeurs du temps
89     y1_values : valeurs de y
90     y2_values : valeurs de y'
91     """
92     n = int((t_end - t0) / dt) + 1
93     t_values = np.linspace(t0, t_end, n)
94     y1_values = np.zeros(n)
95     y2_values = np.zeros(n)
96
97     y1_values[0] = y1_0
98     y2_values[0] = y2_0
99
100     for i in range(1, n):
101         t = t_values[i-1]
102         y1 = y1_values[i-1]
103         y2 = y2_values[i-1]
104
105         k1_y1 = y2 * dt
106         k1_y2 = (k**2 * y1) * dt
107
108         k2_y1 = (y2 + k1_y2 / 2) * dt
109         k2_y2 = (k**2 * (y1 + k1_y1 / 2)) * dt
110
111         k3_y1 = (y2 + k2_y2 / 2) * dt
112         k3_y2 = (k**2 * (y1 + k2_y1 / 2)) * dt
113
114         k4_y1 = (y2 + k3_y2) * dt
115         k4_y2 = (k**2 * (y1 + k3_y1)) * dt
116
117         y1_values[i] = y1 + (k1_y1 + 2*k2_y1 + 2*k3_y1 + k4_y1) / 6
118         y2_values[i] = y2 + (k1_y2 + 2*k2_y2 + 2*k3_y2 + k4_y2) / 6
119
120     return t_values, y1_values, y2_values
121
```

## Détermination de la relation de dispersion

$$\Phi(x, z, t) = X(x, t) \times Z(z)$$

$$\longrightarrow \begin{cases} \frac{\partial^2 X}{\partial x^2} = -\mu \times X \\ \frac{\partial^2 Z}{\partial z^2} = \mu \times Z \end{cases} \quad \text{car le Laplacien est nul}$$

$$\left(\frac{\partial \Phi}{\partial z}\right)_{z=0} = \frac{\partial \eta}{\partial t} \Rightarrow \begin{cases} X(x, t) = \cos(kx - \omega t) \\ \left(\frac{\partial Z}{\partial z}\right)_{z=0} = A\omega \end{cases}$$

$$\longrightarrow \mu = k^2 \Rightarrow Z(z) = \alpha \cosh(kz) + \beta \sinh(kz)$$

$$\Rightarrow Z(z) = \frac{A\omega}{k} \left( \frac{\cosh(k(z + H))}{\sinh(kH)} \right)$$

$$\eta(x, t) = A \sin(kx - \omega t) = -\frac{1}{g} \times \frac{\partial \Phi}{\partial t} \Big|_{z=0} = \frac{\omega}{g} \sin(kx - \omega t) Z(0)$$

$$A = \frac{A\omega^2}{gk \times \tanh(kH)} \Rightarrow \omega^2 = gk \tanh(kH)$$

$$k_{1,y} = k_1 \sin(i_1)$$

$$k_{2,y} = k_2 \sin(i_2)$$

- En supposant  $kH \ll 1$ , on a :

$$v_\varphi = \frac{\omega}{k} = \sqrt{g \times H}$$

$$\sqrt{g \times H_1} k_1 = \omega = k_2 \times \sqrt{g \times H_2}$$

$$\frac{\sin(i_1)}{\sin(i_2)} = \sqrt{\frac{H_1}{H_2}} \longrightarrow \frac{\sin(i)}{\sqrt{H}} = \text{constante}$$