

TAJJA

IMANE

Candidate 55290

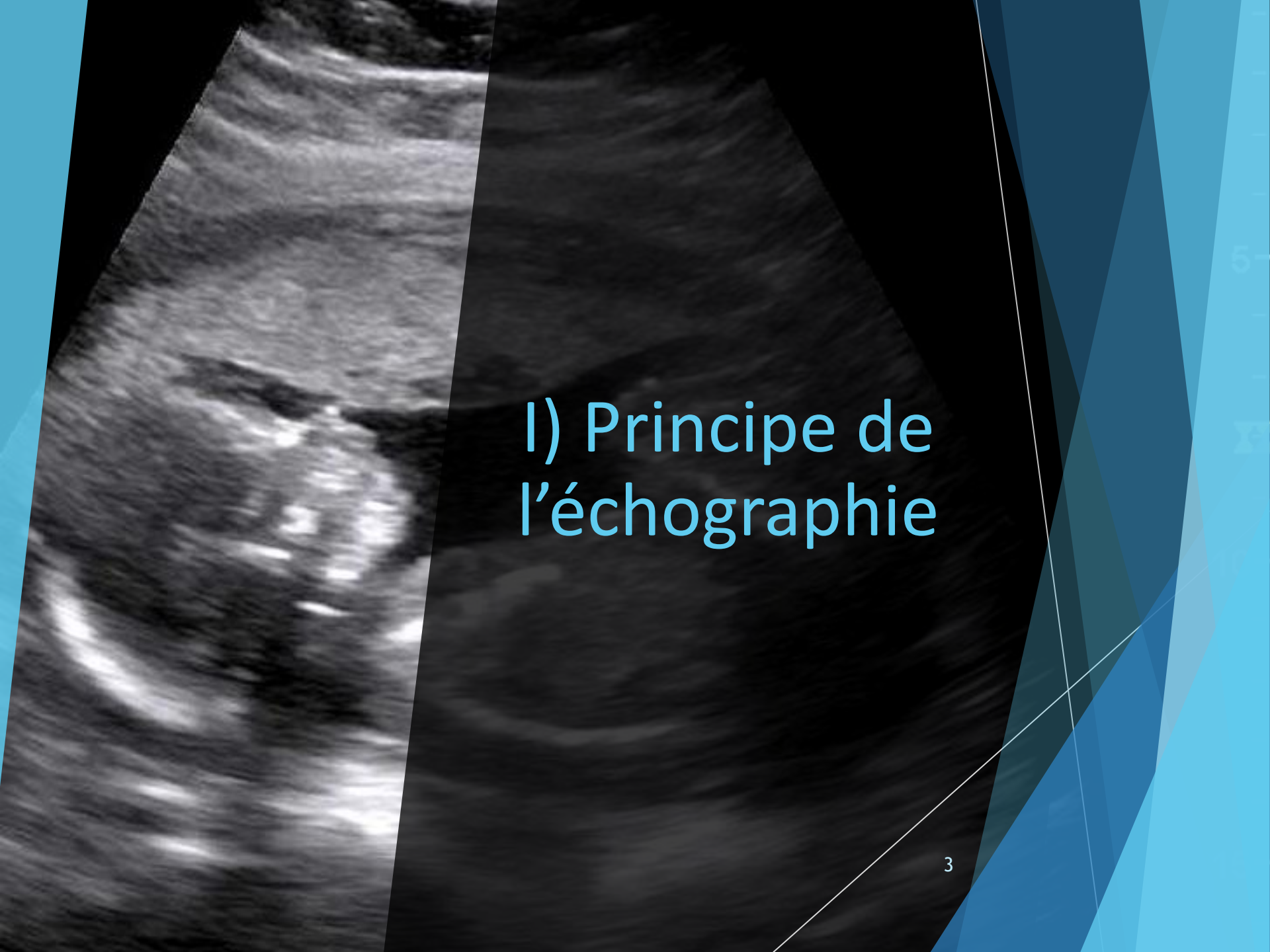
PROPAGATION DES ULTRASONS DANS LE CADRE D'ECHOGRAPHIE PELVIENNE



Thème: Santé

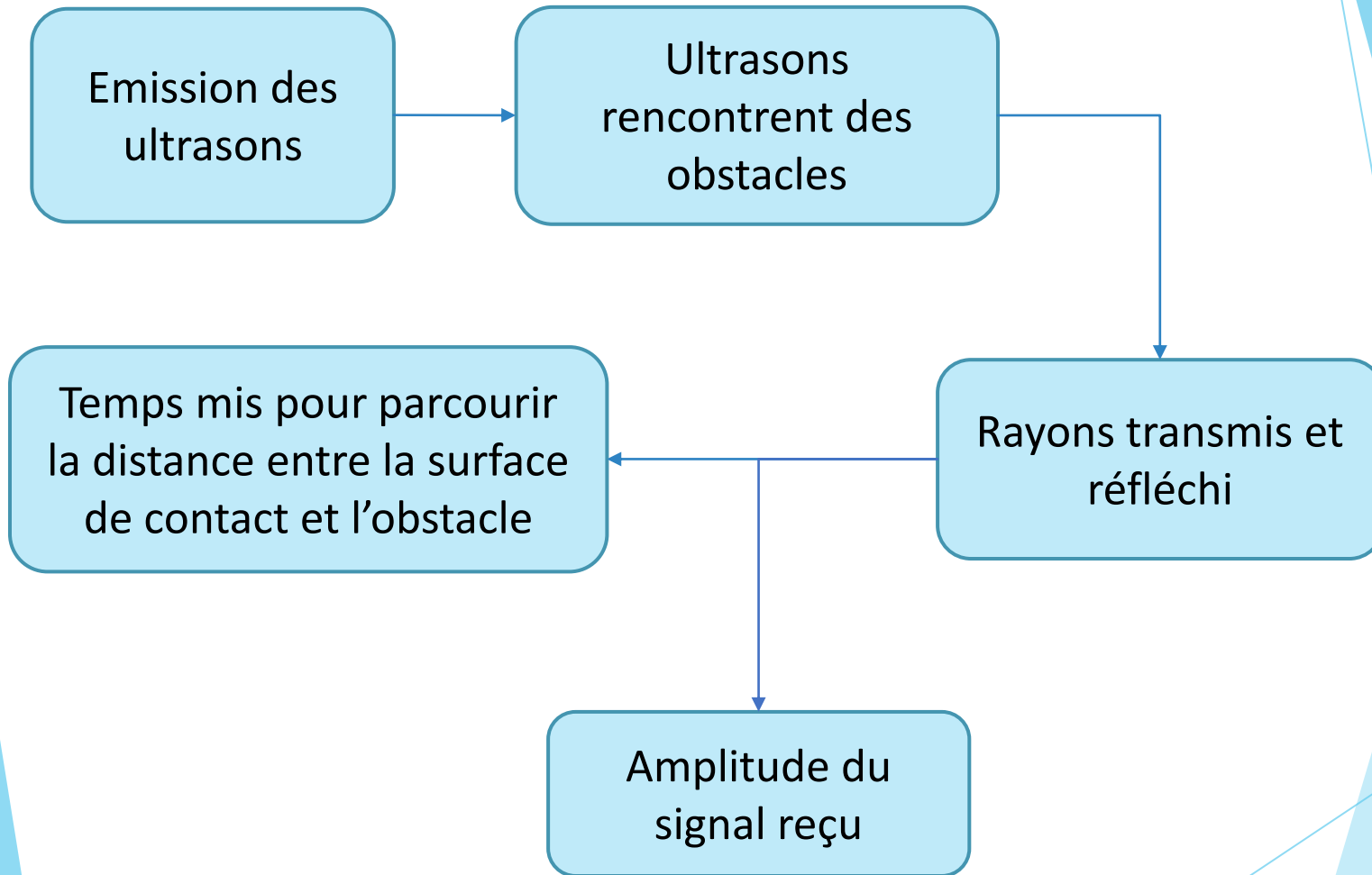
Sommaire

- I. Avant-propos: Principe de l'échographie
 - Phénomènes physiques intervenant dans l'échographie
 - II. Modélisation acoustique
 - Impédance acoustique et propagation
 - Objectifs
- III. Mise en évidence de l'importance du gel (code)
 - Zoom sur les paramètres du code
 - Résultat de l'exécution du programme
- IV. Partie expérimentale
 - Mesure de la viscosité
 - Mesure de la masse volumique
- Etude expérimentale de la propagation des ultrasons
 - V. Simulation (COMSOL)
 - Principe de la simulation
 - Résultats
 - Solution
- VI. Conclusion

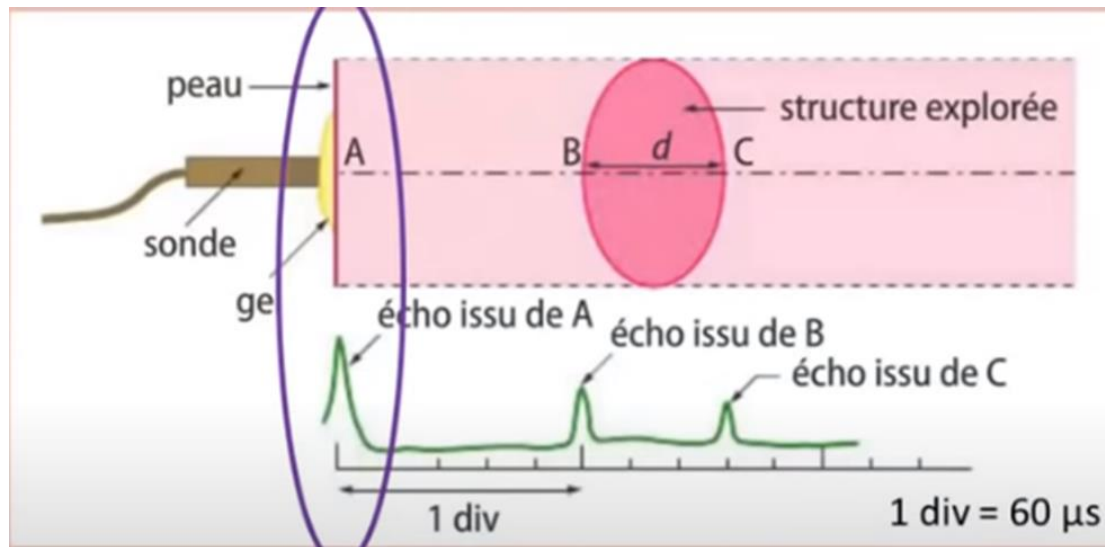
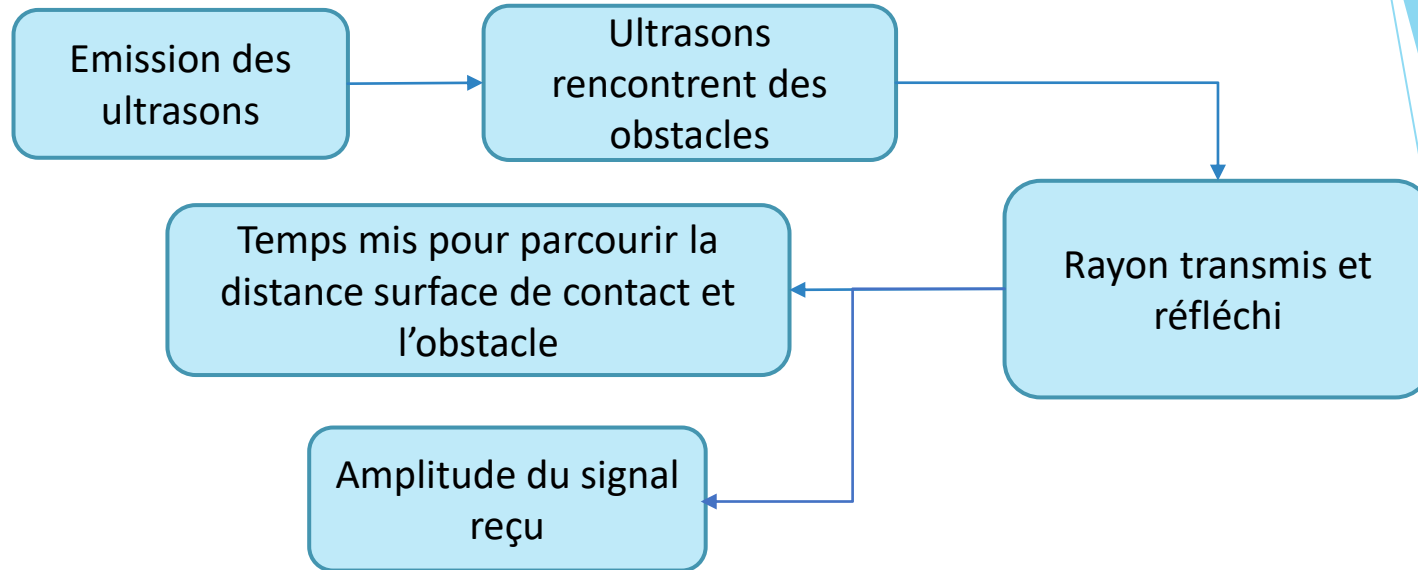
The background of the slide features a large, dark, grainy ultrasound image of a heart, showing internal structures like the ventricles and valves. This image is partially obscured by a large, dark, semi-transparent triangular shape that points towards the top left. To the right of this triangle, there are several overlapping, semi-transparent blue geometric shapes, including triangles and polygons, creating a modern, abstract design. The text 'I) Principe de l'échographie' is centered in the middle of the slide, overlaid on the dark triangular shape.

I) Principe de l'échographie

Phénomènes physiques intervenant dans échographie



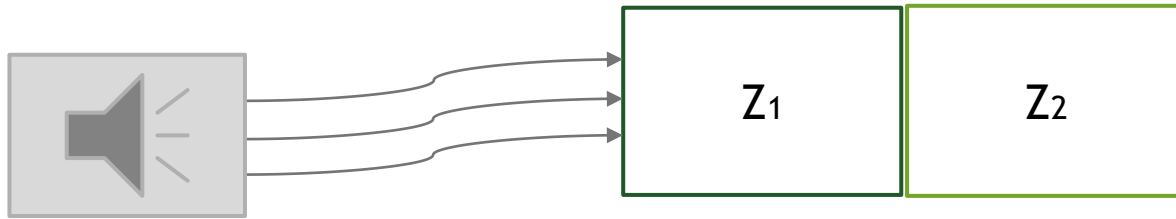
Phénomènes physiques intervenant dans échographie



Echos issues des différentes interfaces

II) Modélisation acoustique

1) Impédance acoustique et propagation



$$T = \frac{4Z_1Z_2}{(Z_1+Z_2)^2}$$

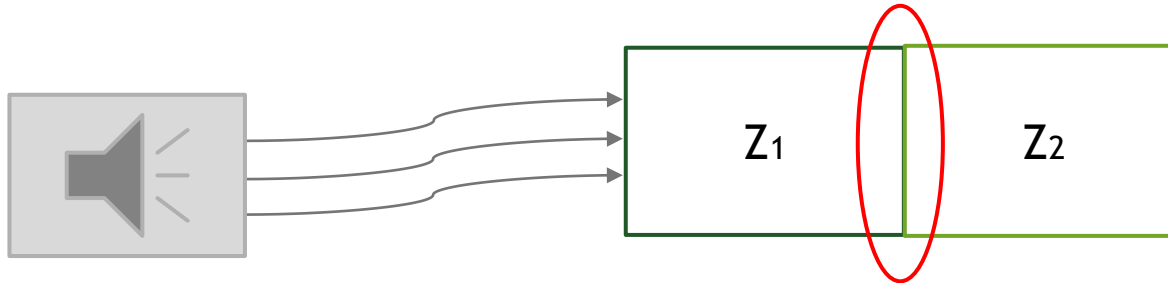
Coefficient de réflexion

$$R = \frac{(Z_2-Z_1)^2}{(Z_2+Z_1)^2}$$

Coefficient de transmission

- R augmente avec (Z_2-Z_1)
- T diminue avec (Z_2-Z_1)

1) Impédance acoustique et propagation



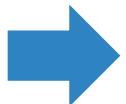
$$T = \frac{4Z_1Z_2}{(Z_1+Z_2)^2}$$

Coefficient de réflexion

$$R = \frac{(Z_2-Z_1)^2}{(Z_2+Z_1)^2}$$

Coefficient de transmission


- R augmente avec (Z_2-Z_1)
- T diminue avec (Z_2-Z_1)



- Faible différence d'impédance entre gel et peau
- Gel = adaptateur d'impédance

2) Objectifs

- ❑ Mettre en évidence importance du gel
- ❑ Paramètres du gel influençant le comportement du signal
- ❑ Exploiter les résultats par une simulation (COMSOL)



III) Mise en évidence de l'importance du gel (code)

```

1 import numpy as np
2 from matplotlib import pyplot as plt
3 from matplotlib import animation
4
5 '''simulation onde plane progressive sinusoidale,
6 incidence normale sur l'interface'''
7
8
9 omega = 120
10 N = 100000
11 x_max = 40
12
13 '''impédances milieu'''
14 Z_gel = 0.0004e6 #km m^-2 s-1
15 Z_peau = 1.48e6
16
17 '''masses volumiques milieu'''
18 rho_gel = 1 #kg.m^-3
19 rho_peau = 1000 #kg.m^-3
20
21 '''Z = rho * c'''
22 c_gel = Z_gel/rho_gel
23 c_peau = Z_peau/rho_peau
24
25 '''k = w/c'''
26 k_gel = omega/c_gel
27 k_peau = omega/c_peau
28
29
30 '''coeff d'atténuation de l'onde'''
31 alpha_gel = 0.03
32 alpha_peau = 0.04
33
34 '''coefficients d'amplitude'''
35 A = 1
36 r = (Z_peau-Z_gel)/(Z_gel + Z_peau) #coeff reflexion
37 tr = 2*Z_peau/(Z_gel + Z_peau) #coeff transmission
38
39 print("Check de la relation 1 + r = t : " + str(1+r == tr))
40
41 '''interface entre deux milieux'''
42 pos_intf = 0
43
44 # Set up figure
45 fig = plt.figure()
46 ax = plt.axes(xlim=(-x_max, x_max), ylim=(-5, 5))
47
48 line_incidente, = ax.plot([], [], linewidth = 2, color = 'k', linestyle = '-')
49 line_reflechie, = ax.plot([], [], linewidth = 2, color = 'r', linestyle = '-')
50 line_transmise, = ax.plot([], [], linewidth = 2, color = 'g')
51
52 total_gauche, = ax.plot([], [], linewidth = 2, color = 'b')
53 frontiere, = ax.plot([], [], linewidth = 4, color = 'k')
54
55 # Initialisation function
56 def init():
57     line_incidente.set_data([], [])
58     line_reflechie.set_data([], [])
59     line_transmise.set_data([], [])
60
61     total_gauche.set_data([], [])
62     frontiere.set_data([], [])
63     return line_incidente, line_reflechie, line_transmise, total_gauche, frontiere
64
65 # Animation function, called sequentially
66 def animate(i):
67     t = 0.001*i
68
69     x_incident = np.linspace(-x_max, 0, int(N/2))
70     x_reflechie = np.linspace(-x_max, 0, int(N/2))
71     x_transmise = np.linspace(0, x_max, int(N/2))
72     x_gauche = x_incident
73     x_frontiere = 10*[0]
74
75     y_incident = A * np.sin(k_gel*x_incident - omega*t)*np.exp(-alpha_gel * x_incident)
76     y_reflechie = r * np.sin(-k_gel*x_reflechie - omega*t)*np.exp(-alpha_gel * x_reflechie)
77     y_transmise = tr * np.sin(k_peau*x_transmise - omega*t)*np.exp(-alpha_peau * x_transmise)
78     y_gauche = y_incident + y_reflechie
79     y_frontiere = np.linspace(-2, 2, 10)
80
81     line_incidente.set_data(x_incident, y_incident)
82     line_reflechie.set_data(x_reflechie, y_reflechie)
83     line_transmise.set_data(x_transmise, y_transmise)
84     total_gauche.set_data(x_gauche, y_gauche)
85     frontiere.set_data(x_frontiere, y_frontiere)
86
87     return line_incidente, line_reflechie, line_transmise, total_gauche, frontiere
88
89 # Call animator. blit=True means only re-draw changed parts
90 anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,
91                               interval=x_max, blit=True)
92
93 # Save animation
94 # anim.save("wave_animation", fps=30)
95
96 plt.show()
97

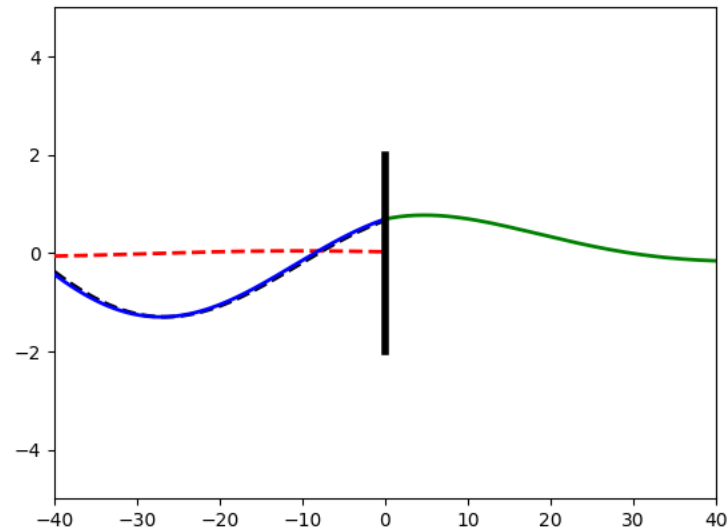
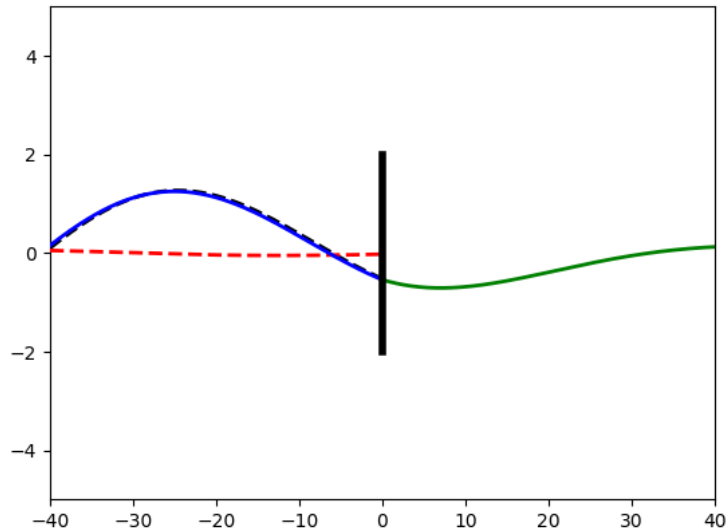
```

Simulation de la propagation d'une onde sinusoidale avec incidence normale sur l'interface air/peau

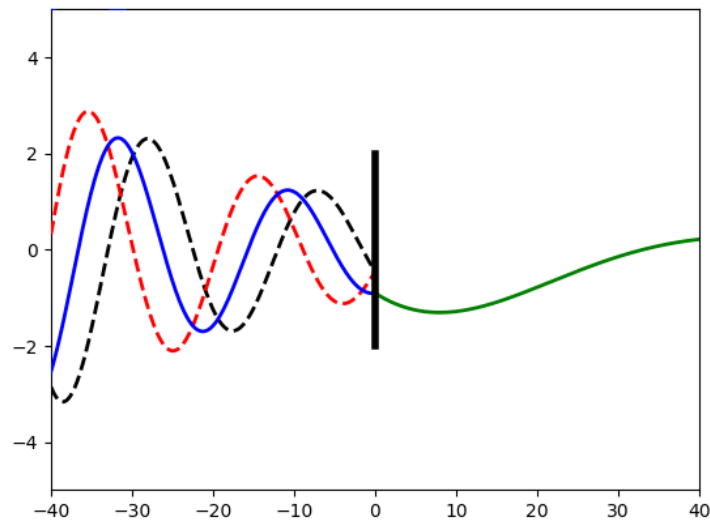
1) Zoom sur les paramètres du code

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3 from matplotlib import animation
4
5 '''simulation onde plane progressive sinusoidale,
6 incidence normale sur l'interface'''
7
8
9 omega = 120
10 N = 100000
11 x_max = 40
12
13 '''impédances milieu'''
14 Z_gel = 0.0004e6 #km m^-2 s-1
15 Z_peau = 1.48e6
16
17 '''masses volumiques milieu'''
18 rho_gel = 1 #kg.m^-3
19 rho_peau = 1000 #kg.m^-3
20
21 '''Z = rho * c'''
22 c_gel = Z_gel/rho_gel
23 c_peau = Z_peau/rho_peau
24
25 ''' k = w/c'''
26 k_gel = omega/c_gel
27 k_peau = omega/c_peau
28
29
30 '''coeff d'attenuation de l'onde'''
31 alpha_gel = 0.03
32 alpha_peau = 0.04
33
34 '''coefficients d'amplitude'''
35 A = 1
36 r = (Z_peau-Z_gel)/(Z_gel + Z_peau) #coeff relflexion
37 tr = 2*Z_peau/(Z_gel + Z_peau)#coeff transmission
38
39 print("Check de la relation 1 + r = t : " + str(1+r == tr))
40
41 '''interface entre deux milieux'''
42 pos_interf = 0
43
```

2) Résultats de l'exécution du programme



Interface gel/peau



Interface air/peau



IV) Partie expérimentale



1) Mesure de la viscosité



Gel 1 dilué à 75%



Viscosimètre



Gel 2 dilué à 75%

Résultats	Gel 1 dilué à 75% (en volume)	Gel 2 dilué à 75% (en volume)	Gel 1 non dilué	Gel 2 non dilué
Viscosité Pa/s	510 ± 5,1	460 ± 4,6	679,7 ± 6,8	612,9 ± 6,1

2) Mesure de la masse volumique



Eprouvette graduée contenant gel



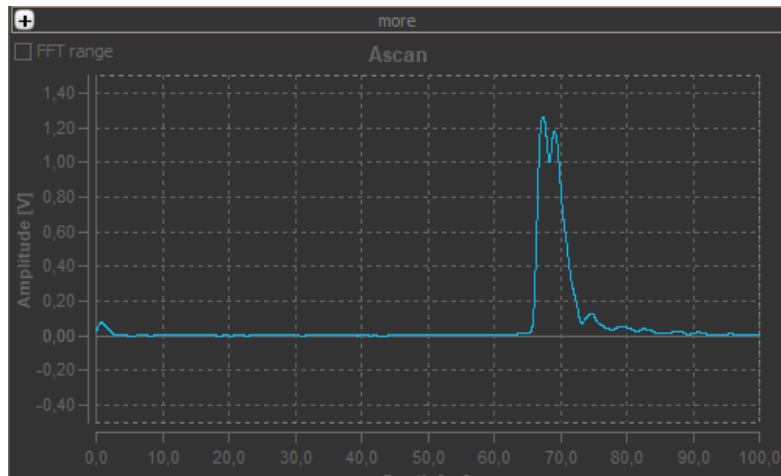
Eprouvette graduée contenant gel
sur balance

Grandeurs/ Gel	Gel 1	Gel 2
Volume	$20 \pm 1 \text{ mL}$	$20 \pm 1 \text{ mL}$
Masse	$20,5 \pm 0,01 \text{ g}$	$19,8 \pm 0,01 \text{ g}$
Masse Volumique	$1025 \pm 10 \text{ Kg/m}^3$	$990 \pm 10 \text{ Kg/m}^3$

3) Etude expérimentale de la propagation des ultrasons



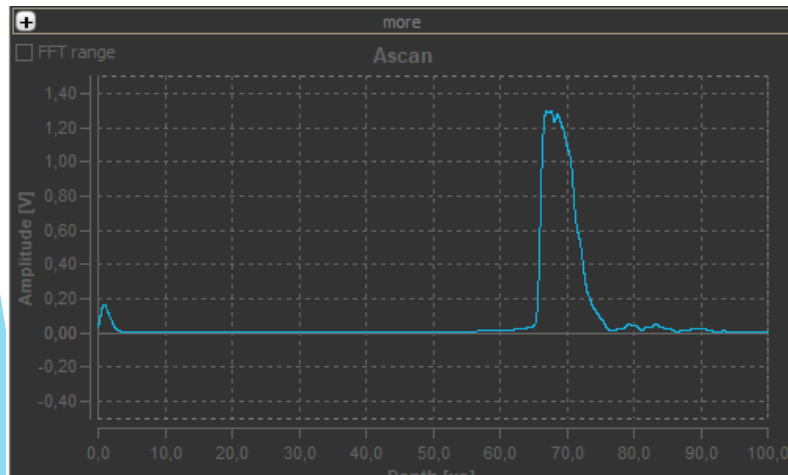
3) Etude expérimentale de la propagation des ultra-sons



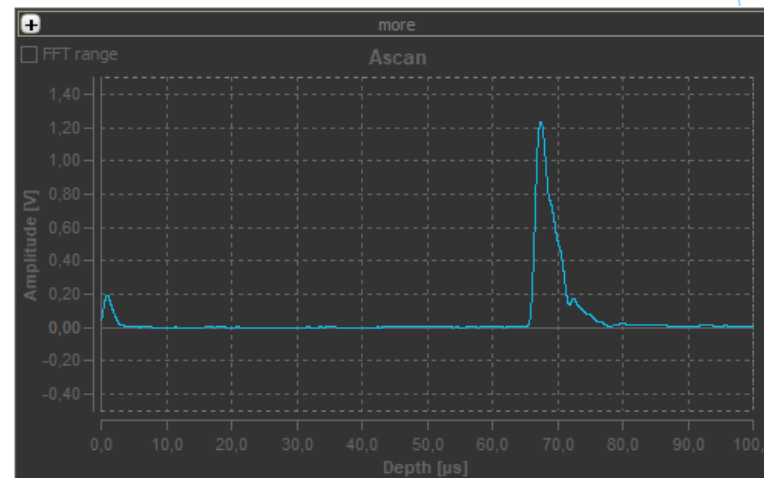
Gel 1 dilué



Gel 2 dilué



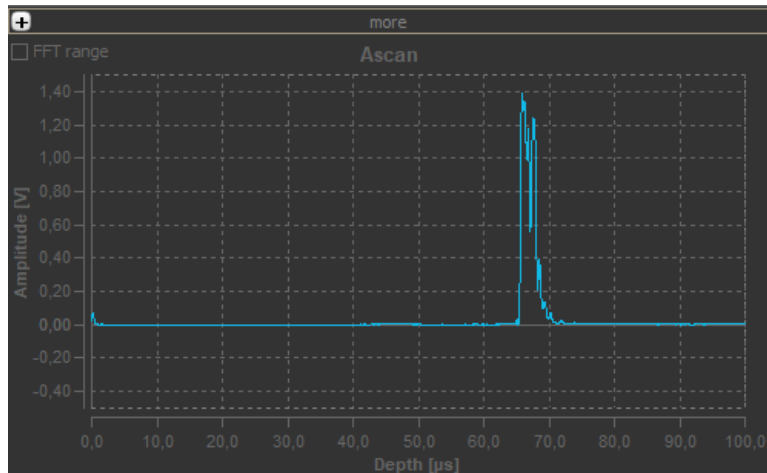
Gel 1 non dilué



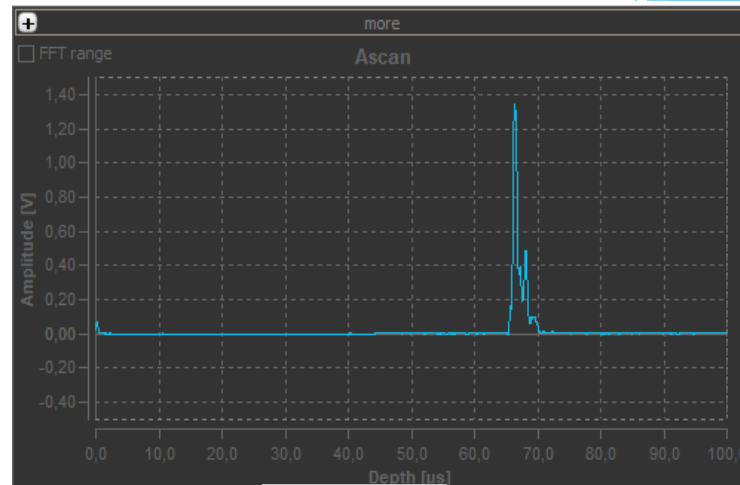
Gel 2 non dilué

Résultats pour un signal de 1 Mhz

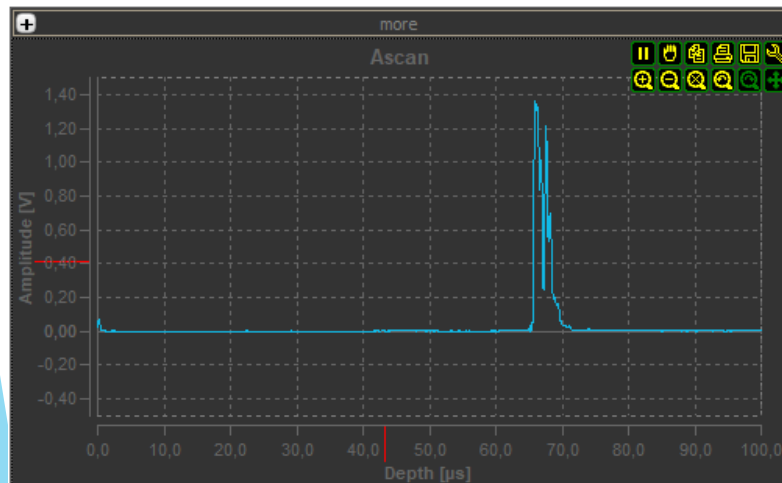
3) Etude expérimentale de la propagation des ultra-sons



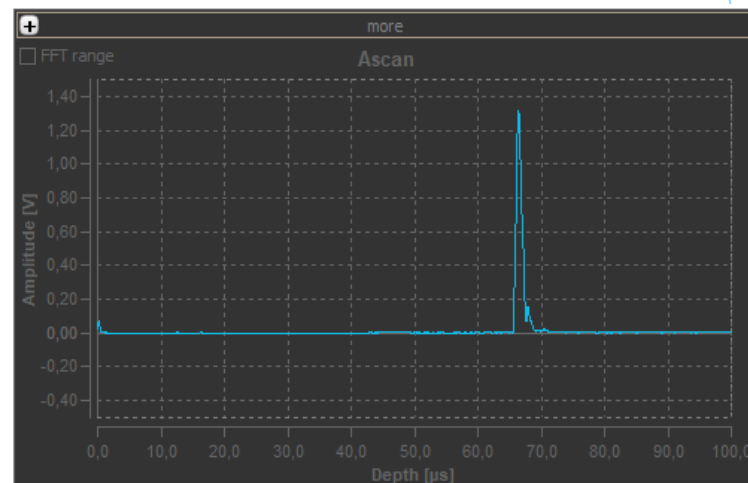
Gel 1 dilué



Gel 2 dilué



Gel 1 non dilué



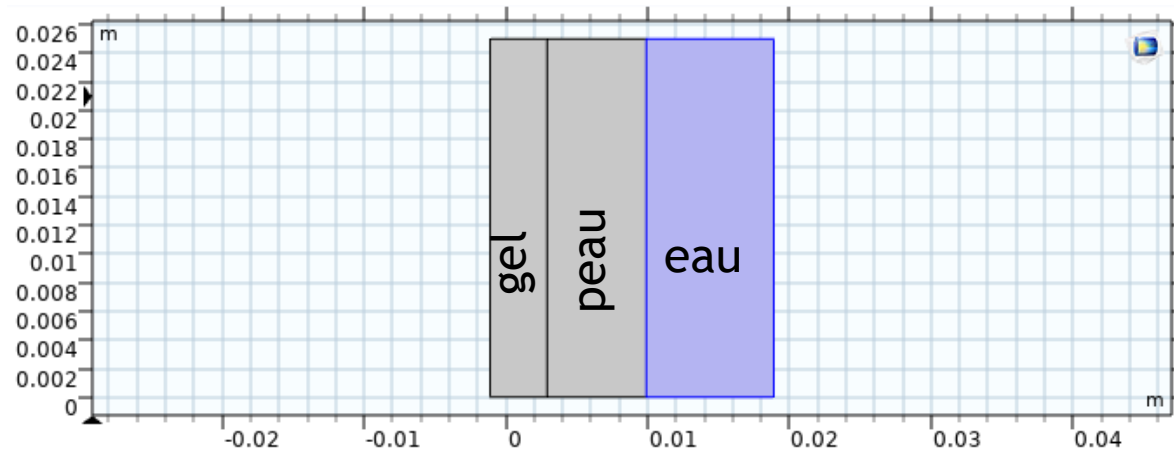
Gel 2 non dilué

Résultats pour un signal de 4 Mhz

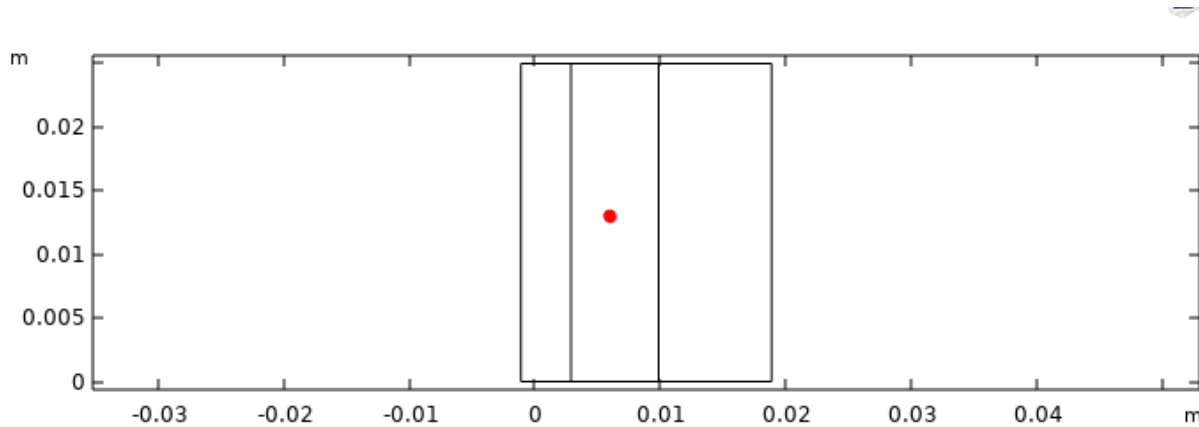


V) Simulation (COMSOL)

1) Principe de la simulation

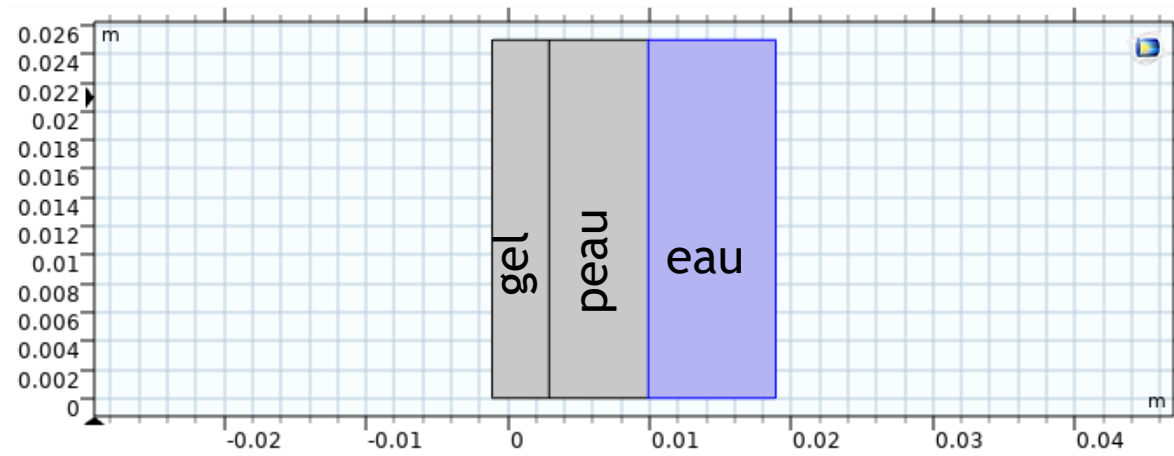


modèle COMSOL réalisé

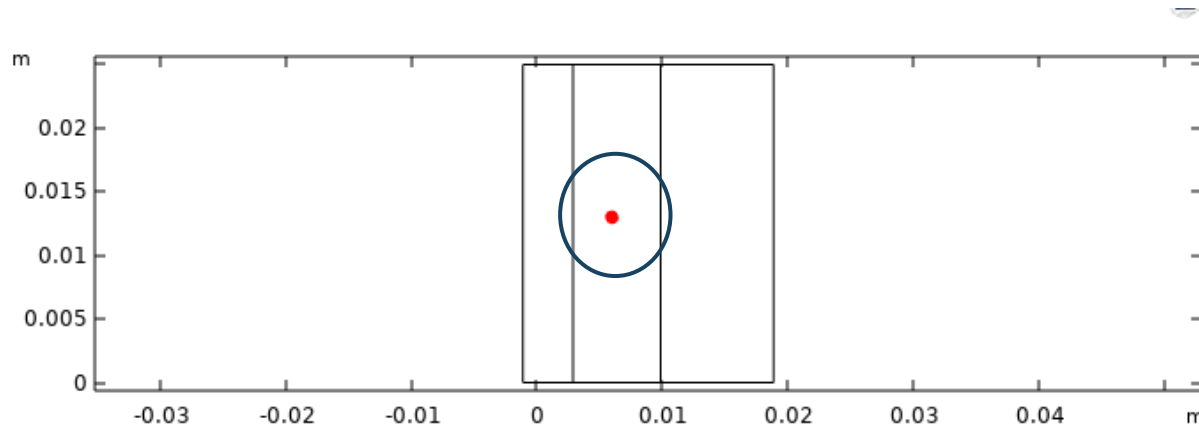


modèle COMSOL réalisé

1) Principe de la simulation



modèle COMSOL réalisé



modèle COMSOL réalisé

2) Résultats

Propriété	Variable	Valeur	Unité	Groupe de propriétés
Vitesse du son	c	1531	m/s	Basique
Masse volumique	rho	1025	kg/m ³	Basique
Viscosité dynamique	mu	510	Pa·s	Basique
Viscosité de volume	muB	510	Pa·s	Basique
Propriété locale SurfF	SurfF	SurfF_liq...	N/m	Propriétés locales

Paramètres du gel

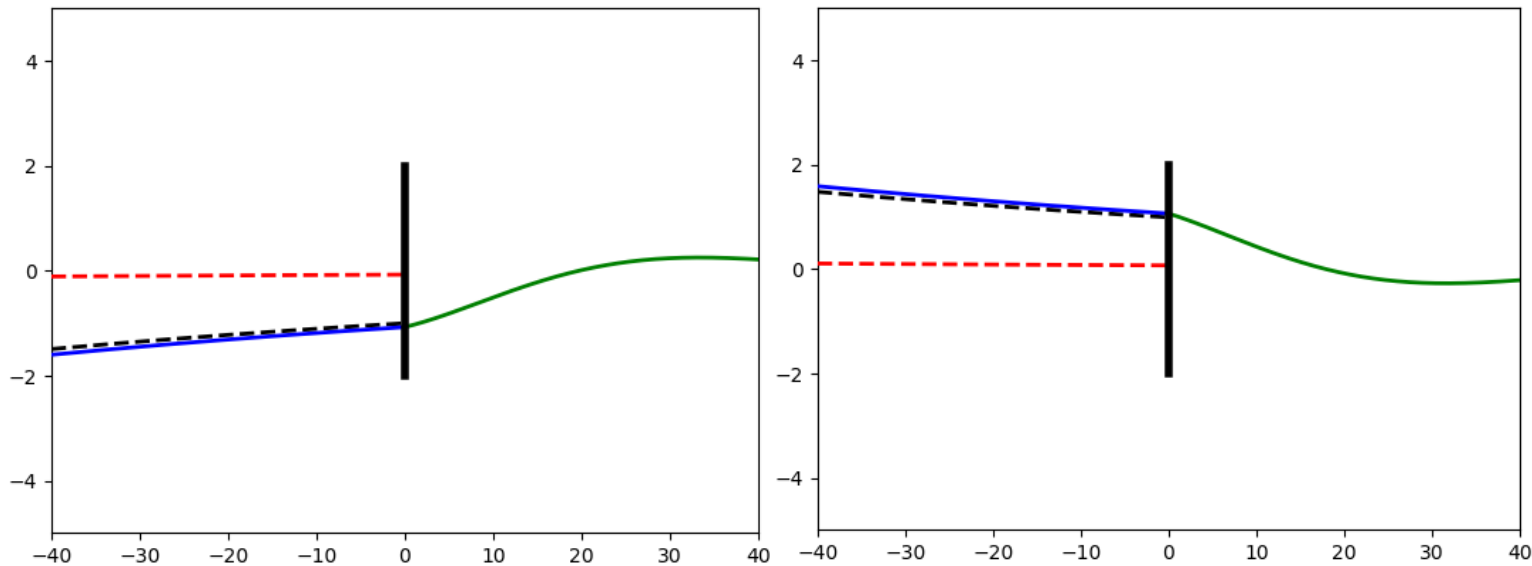
- Variation de la viscosité du gel n'influence pas la valeur de la pression mesurée
- Variation de la masse volumique du gel n'influence pas la valeur de la pression mesurée
- Echec de la simulation...

3) Solution

- Faire varier la masse volumique dans le code de la partie III
- Fixer Z, α

3) Solution

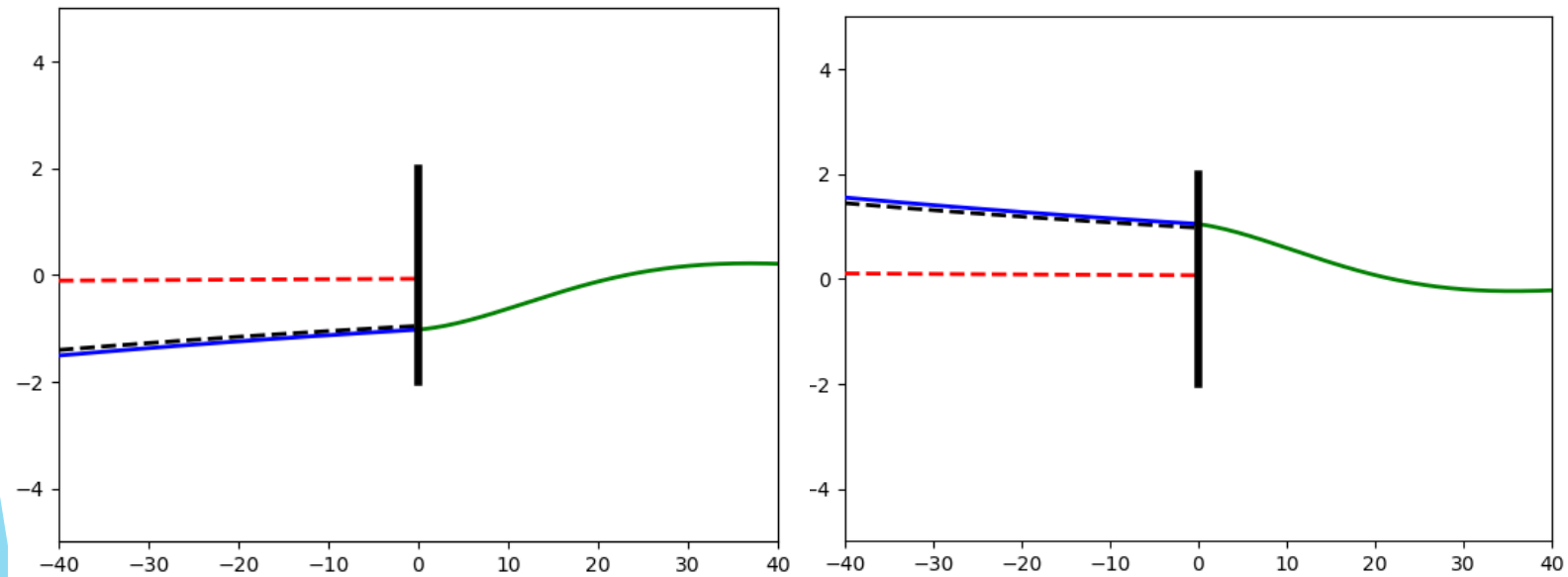
- Faire varier la masse volumique dans le code de la partie III
- Fixer Z, α



Simulation: $\mu=1$

$Z=1,28$
 $\alpha=0,01$

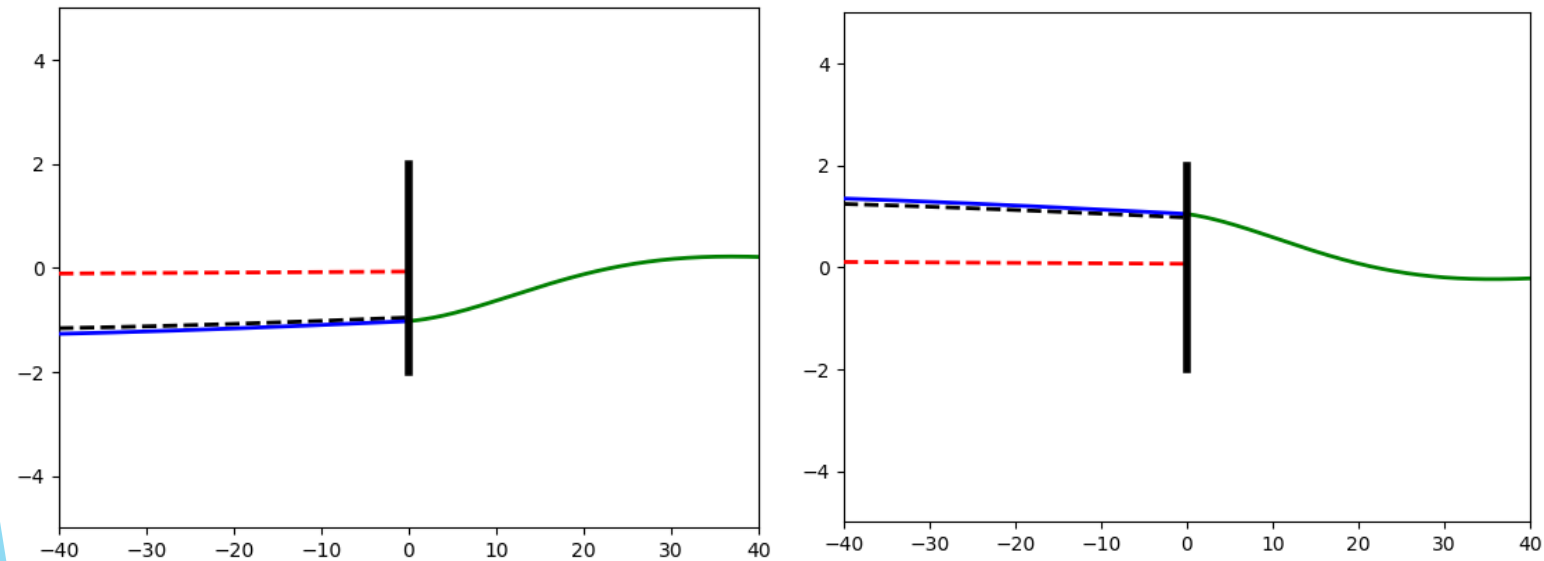
3) Solution



Simulation: $\mu=10$

$Z=1,28$
 $\alpha=0,01$

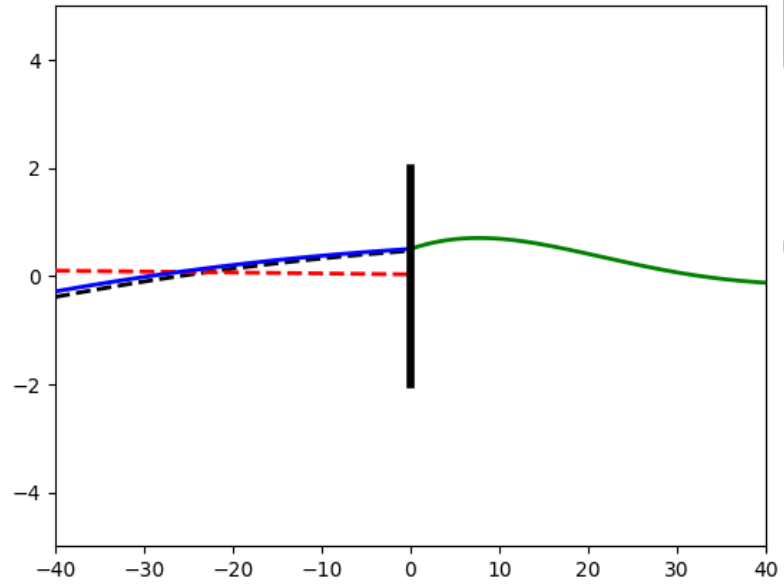
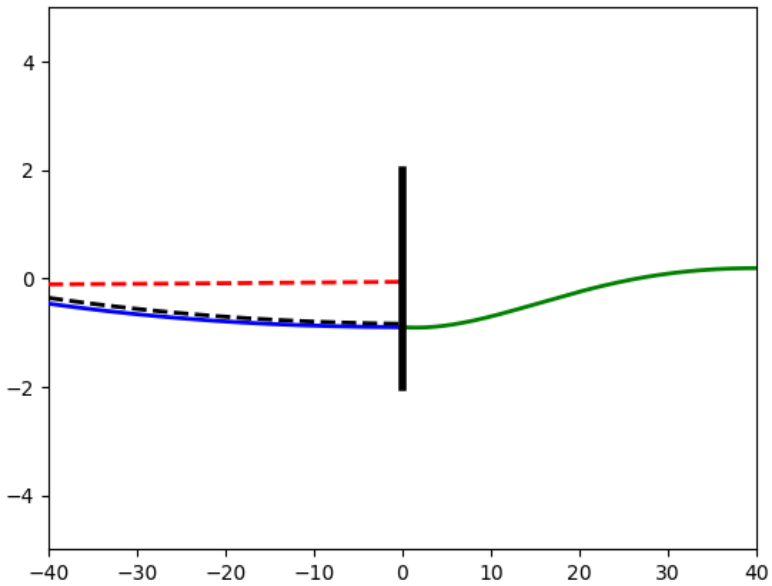
3) Solution



Simulation: $\mu=100$

$Z=1,28$
 $\alpha=0,01$

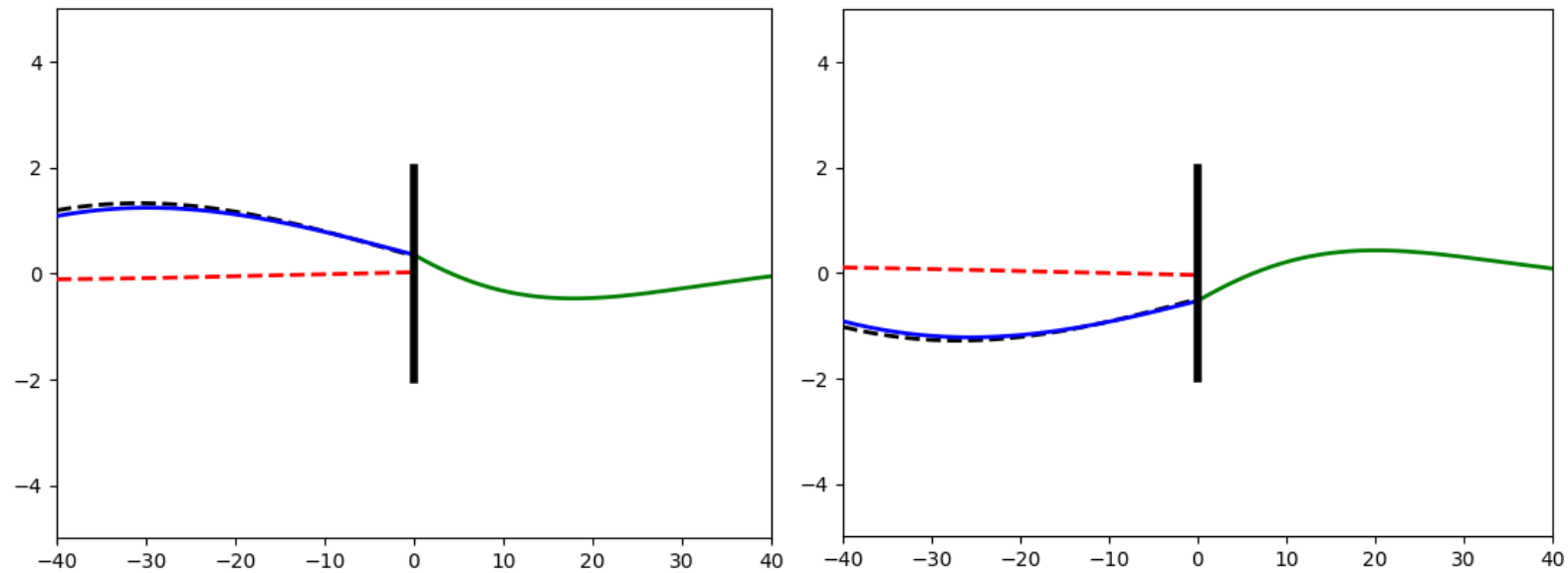
3) Solution



Simulation: $\mu=200$

$Z=1,28$
 $\alpha=0,01$

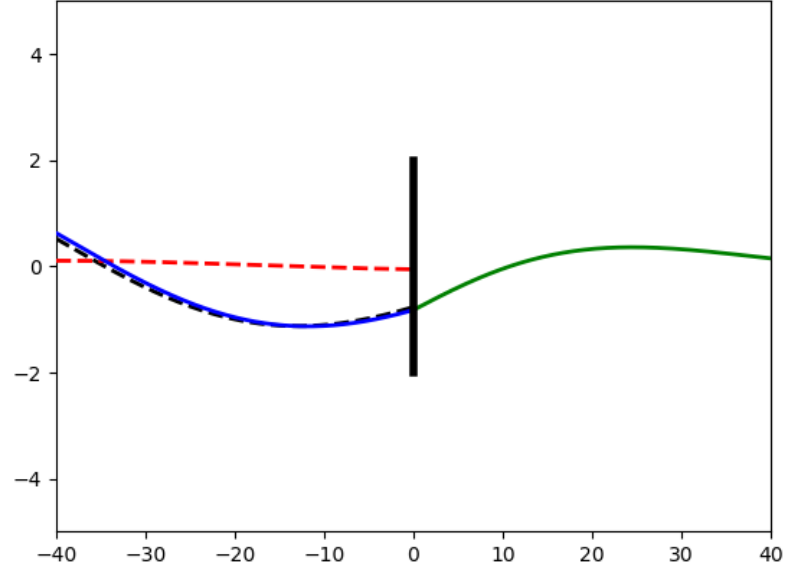
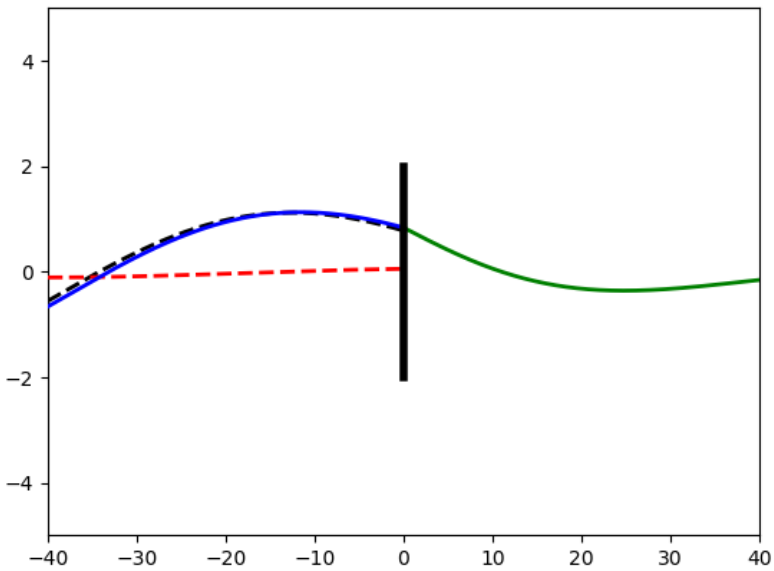
3) Solution



Simulation: $\mu=500$

$Z=1,28$
 $\alpha=0,01$

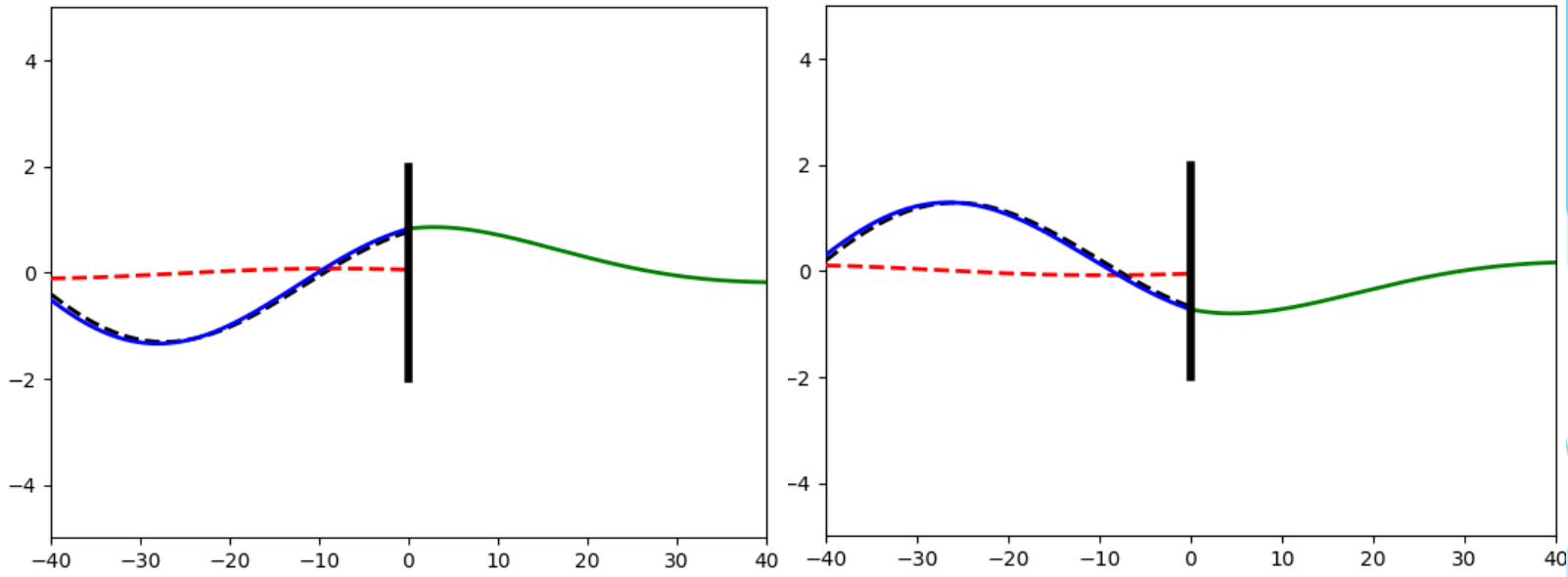
3) Solution



Simulation: $\mu=700$

$Z=1,28$
 $\alpha=0,01$

3) Solution



Simulation: $\mu=1000$

$Z=1,28$
 $\alpha=0,01$



VI) Conclusion

ANNEXE

- Détail du calcul des incertitudes de type matériel

Calcul de la viscosité:

Précision: $P = 1\%$

Gel : Valeur de viscosité	Incertitude de mesure
Gel 1 (dilué) : 510 Pa/s	$U_{c1d} = 5,1$
Gel 2 (dilué) : 460 Pa/s	$U_{c2d} = 4,6$
Gel 1: 679,7 Pa/s	$U_{c1} = 6,8$
Gel 2: 460 Pa/s	$U_{c2} = 6,1$

Calcul de la masse volumique:

- Incertitude lecture: $U_l = 1 \text{ mL}$
- Incertitude balance: $U_b = 0,01 \text{ g}$

$$U_{cr} = 0,01 \text{ g/mL}$$

$$U_{cr} = 10 \text{ Kg/m}^3$$

ANNEXE

► Mesure de la masse volumique des deux gels

- Mesure du volume



- Mesure de la masse



- Déduction de la masse volumique

ANNEXE

- ▶ Matériels utilisés pour l'expérience



Générateur à ultrason

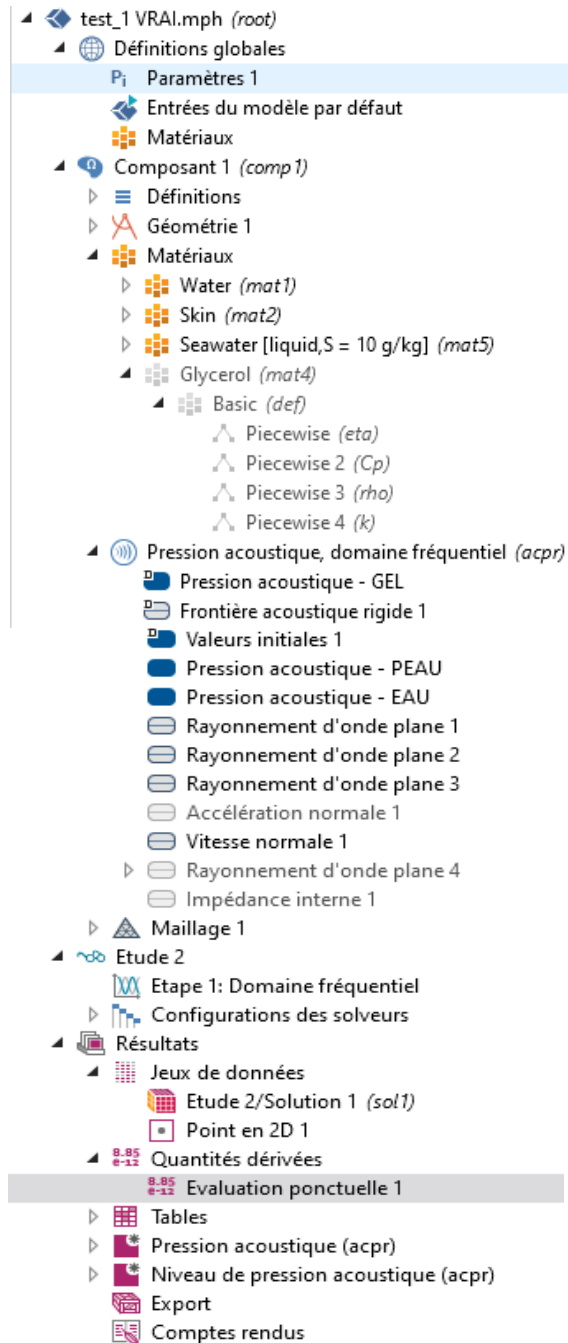


Sondes de différentes fréquences

- Logiciel: GS-EchoView

ANNEXE

► Détail de la simulation sur COMSOL



ANNEXE

► Zoom sur le code python

```
1 import numpy as np
2 from matplotlib import pyplot as plt
3 from matplotlib import animation
4
5 '''simulation onde plane progressive sinusoidale,
6 incidence normale sur l'interface'''
7
8
9 omega = 120
10 N = 100000
11 x_max = 40
12
13 '''impédances milieu'''
14 Z_gel = 0.0004e6 #km m^-2 s-1
15 Z_peau = 1.48e6
16
17 '''masses volumiques milieu'''
18 rho_gel = 1 #kg.m^-3
19 rho_peau = 1000 #kg.m^-3
20
21 '''Z = rho * c'''
22 c_gel = Z_gel/rho_gel
23 c_peau = Z_peau/rho_peau
24
25 ''' k = w/c'''
26 k_gel = omega/c_gel
27 k_peau = omega/c_peau
28
29
30 '''coeff d'attenuation de l'onde'''
31 alpha_gel = 0.03
32 alpha_peau = 0.04
33
34 '''coefficients d'amplitude'''
35 A = 1
36 r = (Z_peau-Z_gel)/(Z_gel + Z_peau) #coeff relfexion
37 tr = 2*Z_peau/(Z_gel + Z_peau)#coeff transmission
38
39 print("Check de la relation 1 + r = t : " + str(1+r == tr))
40
41 '''interface entre deux milieux'''
42 pos_interf = 0
43
```

ANNEXE

► Zoom sur le code python

```
44 # Set up figure
45 fig = plt.figure()
46 ax = plt.axes(xlim=(-x_max, x_max), ylim=(-5, 5))
47
48 line_incidente, = ax.plot([], [], linewidth = 2, color = 'k', linestyle = '--')
49 line_reflechie, = ax.plot([], [], linewidth = 2, color = 'r', linestyle = '--')
50 line_transmise, = ax.plot([], [], linewidth = 2, color = 'g')
51
52 total_gauche, = ax.plot([], [], linewidth = 2, color = 'b')
53 frontiere, = ax.plot([], [], linewidth = 4, color = 'k')
54
55 # Initialisation function
56 def init():
57     line_incidente.set_data([], [])
58     line_reflechie.set_data([], [])
59     line_transmise.set_data([], [])
60
61     total_gauche.set_data([], [])
62     frontiere.set_data([], [])
63     return line_incidente, line_reflechie, line_transmise, total_gauche, frontiere
64
65 # Animation function, called sequentially
66 def animate(i):
67     t = 0.001*i
68
69     x_incident = np.linspace(-x_max, 0, int(N/2))
70     x_reflechie = np.linspace(-x_max, 0, int(N/2))
71     x_transmise = np.linspace(0, x_max, int(N/2))
72     x_gauche = x_incident
73     x_frontiere = 10*[0]
74
75     y_incident = A * np.sin(k_gel*x_incident - omega*t)*np.exp(-alpha_gel * x_incident)
76     y_reflechie = r * np.sin(-k_gel*x_reflechie - omega*t)*np.exp(-alpha_gel * x_reflechie)
77     y_transmise = tr * np.sin(k_peau*x_transmise - omega*t)*np.exp(-alpha_peau * x_transmise)
78     y_gauche = y_incident + y_reflechie
79     y_frontiere = np.linspace(-2, 2, 10)
80
81     line_incidente.set_data(x_incident, y_incident)
82     line_reflechie.set_data(x_reflechie, y_reflechie)
83     line_transmise.set_data(x_transmise, y_transmise)
84     total_gauche.set_data(x_gauche, y_gauche)
85     frontiere.set_data(x_frontiere, y_frontiere)
86
87     return line_incidente, line_reflechie, line_transmise, total_gauche, frontiere
88
89 # Call animator. blit=True means only re-draw changed parts
90 anim = animation.FuncAnimation(fig, animate, init_func=init, frames=200,\
91                               interval=x_max, blit=True)
92
93 # Save animation
94 # anim.save("wave_animation", fps=30)
95
96 plt.show()
97
```