hw6

March 13, 2023

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

0.1 1.)

A downgoing P wave in a medium with a P velocity of 6 km/s travels through this "corner" shaped structure. If the incident ray is at an angle of 60○ from the horizontal and the final ray is at an angle of 75○ from the vertical, what is the P velocity within the corner-shaped medium?

Start with Snell's Law to find α_2 and write two equations for the two unkowns.

$\frac{\sin\theta 1}{\alpha_1} = \frac{\sin\theta_2}{\alpha_2}$ and $\frac{(\sin 90-\theta_2)}{\alpha_2} = \frac{\sin\theta_3}{\alpha 1}$ where $\theta_1 = 30°$ and $\theta_3 = 15°$ and $\alpha_1 = 6$.

$\alpha_2 = \frac{\alpha_1 \sin(90-\theta_2)}{\sin\theta_3} = \frac{\alpha_1\cos\theta_2}{\sin\theta_3}$

and

$\alpha_2 = \frac{\alpha_1\sin\theta_2}{\sin\theta_1} \rightarrow \frac{\cos\theta_2}{\sin\theta_3} = \frac{\sin\theta_2}{\sin\theta_1} \Rightarrow \frac{\sin\theta_1}{\sin\theta_3} = \tan\theta_2 \Rightarrow \theta_2 = \tan^{-1}\frac{\sin\theta_1}{\sin\theta_3}$

$\alpha_2 = \alpha_1\frac{\cos(\theta_2)}{\sin\theta_3}$

[2]:
```
#question 1
alpha_1 = 6
theta_1 = 30 * (np.pi / 180)
theta_3 = 15 * (np.pi / 180)
theta_2 = np.arctan(
    np.sin(theta_1) / np.sin(theta_3)
)
alpha_2 = np.round(alpha_1 * (np.cos(theta_2) / np.sin(theta_3)),2)
print(f'Wavespeed in the object is: {alpha_2} km/s')
```

```
Wavespeed in the object is: 10.66 km/s
```

0.2 2.)

Model the Antarctic ice sheet as a simple layer over half-space with h = 2 km, ρ1 = 900 kg/m3, ρ2 = 2.6 g/cm3, β1 = 1950 m/s, and β2 = 3250 m/s.

We can re-write μ in terms of β and ρ as $\mu = \beta^2\rho$. Now:

$$h\omega\sqrt{\frac{1}{\beta_1^2}-\frac{1}{c^2}} = \tan^{-1\}\left[\frac{\beta_2^2\rho_2\sqrt{\frac{1}{c^2}-\frac{1}{\beta_2^2}}}{\beta_1^2\rho_1\sqrt{\frac{1}{\beta_1^2}-\frac{1}{c^2}}}\right]\Rightarrow\omega=\frac{\tan^{-1}\left[\frac{\beta_2^2\rho_2\sqrt{\frac{1}{c^2}-\frac{1}{\beta_2^2}}}{\beta_1^2\rho_1\sqrt{\frac{1}{\beta_1^2}-\frac{1}{c^2}}}\right]}{h\sqrt{\frac{1}{\beta_1^2}-\frac{1}{c^2}}}}$$

0.3 A.)

[4]:
```
import pandas as pd
```

[5]:
```
df = pd.DataFrame()

c = [2.000, 2.400, 2.800, 3.200]
# put everything in kilometers and grams
h = 2
rho_1 = 900  * (1/1e-9) * (1e3)
rho_2 = 2.6 * (1/1e-15)
beta_1 = 1.950
beta_2 = 3.250
for speed in c:
    c = speed
    theta = (beta_2**2 * rho_2 * np.sqrt((1/c**2) - (1/beta_2**2)
        ))/ beta_1**2 * rho_1 * np.sqrt((1/beta_1**2) - (1/c**2))
    omega = np.arctan(theta) / h * np.sqrt((1/beta_1**2) - (1/c**2))
    T = np.round(1 / omega, 2)
    df = pd.concat([df, pd.Series(c)], ignore_index = True)
    df.loc[df.index[-1], 'omega (s^-1)'] = omega
    df.loc[df.index[-1], 'omega (s^-1)'] = omega
    df.loc[df.index[-1], 'period (s)'] = T

df = df.rename(columns = {0:'speed (km/s)'})
df
```

[5]:

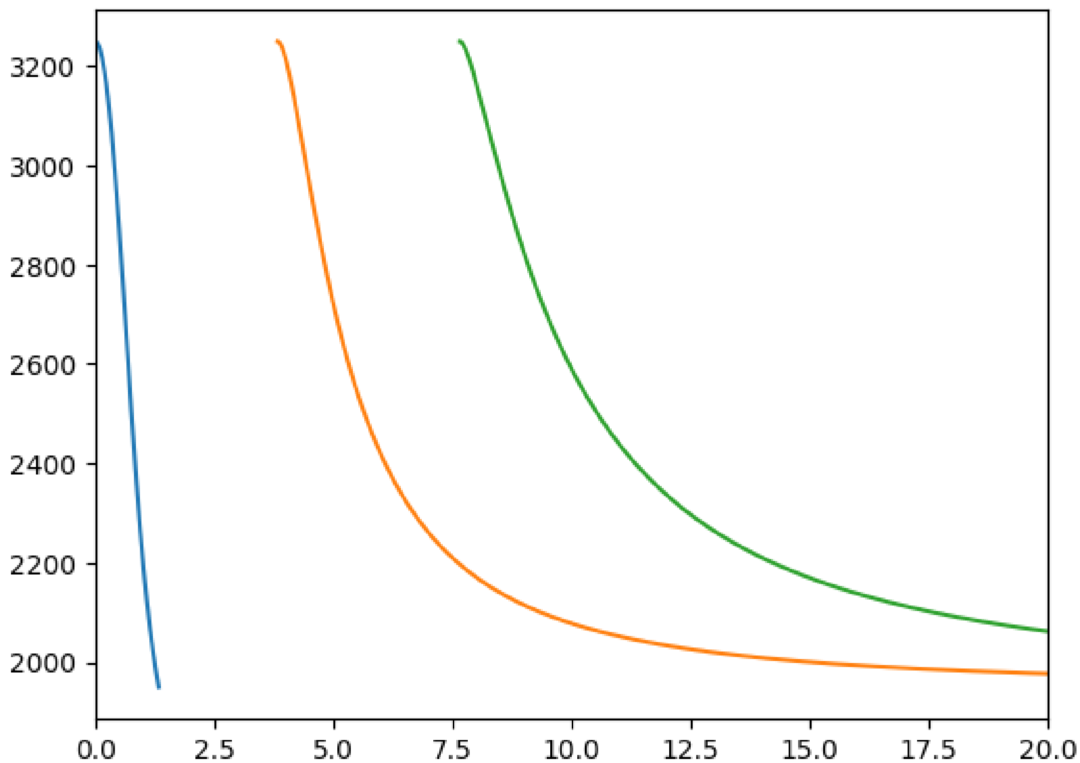
	speed (km/s)	omega (s^-1)	period (s)
0	2.0	0.089497	11.17
1	2.4	0.234798	4.26
2	2.8	0.289037	3.46
3	3.2	0.319348	3.13

[8]:
```
nbeta = 100001
cc = np.zeros((len(range(0,3,1)),nbeta - 2))
ww = np.zeros((len(range(0,3,1)),nbeta - 2))
h = 2000
beta_1 = 1950
beta_2 = 3250
rho_1 = 900
rho_2 = 2600
```

```
carray = np.linspace(beta_1 - 2, beta_2 + 2, nbeta - 2)


for i in range(0,3,1):
    for j in range(len(carray)):
        c = carray[j]
        cc[i,j] = c
        first = 1/c**2

        theta = (beta_2**2 * rho_2 * np.sqrt((1/c**2) - (1/beta_2**2)
            ))/ beta_1**2 * rho_1 * np.sqrt((1/beta_1**2) - (1/c**2))
        omega = (np.arctan(theta) + i * np.pi) / (h * np.sqrt((1/beta_1**2) - (1/
 ↪c**2)))

        ww[i,j] = omega

plt.plot(ww[0], cc[0])
plt.plot(ww[1], cc[1])
plt.plot(ww[2], cc[2])
plt.xlim(0,20)
```

```
/tmp/ipykernel_318330/3278707206.py:20: RuntimeWarning: invalid value
encountered in sqrt
  ))/ beta_1**2 * rho_1 * np.sqrt((1/beta_1**2) - (1/c**2))
/tmp/ipykernel_318330/3278707206.py:21: RuntimeWarning: invalid value
encountered in sqrt
  omega = (np.arctan(theta) + i * np.pi) / (h * np.sqrt((1/beta_1**2) -
(1/c**2)))
/tmp/ipykernel_318330/3278707206.py:19: RuntimeWarning: invalid value
encountered in sqrt
  theta = (beta_2**2 * rho_2 * np.sqrt((1/c**2) - (1/beta_2**2)
```

[8]: (0.0, 20.0)


3200
3000
2800
2600
2400
2200
2000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0