Détermination du niveau des océans par satellites

N° candidat : 49003

PLAN

1-introduction générale

2-Etude de l’orbite

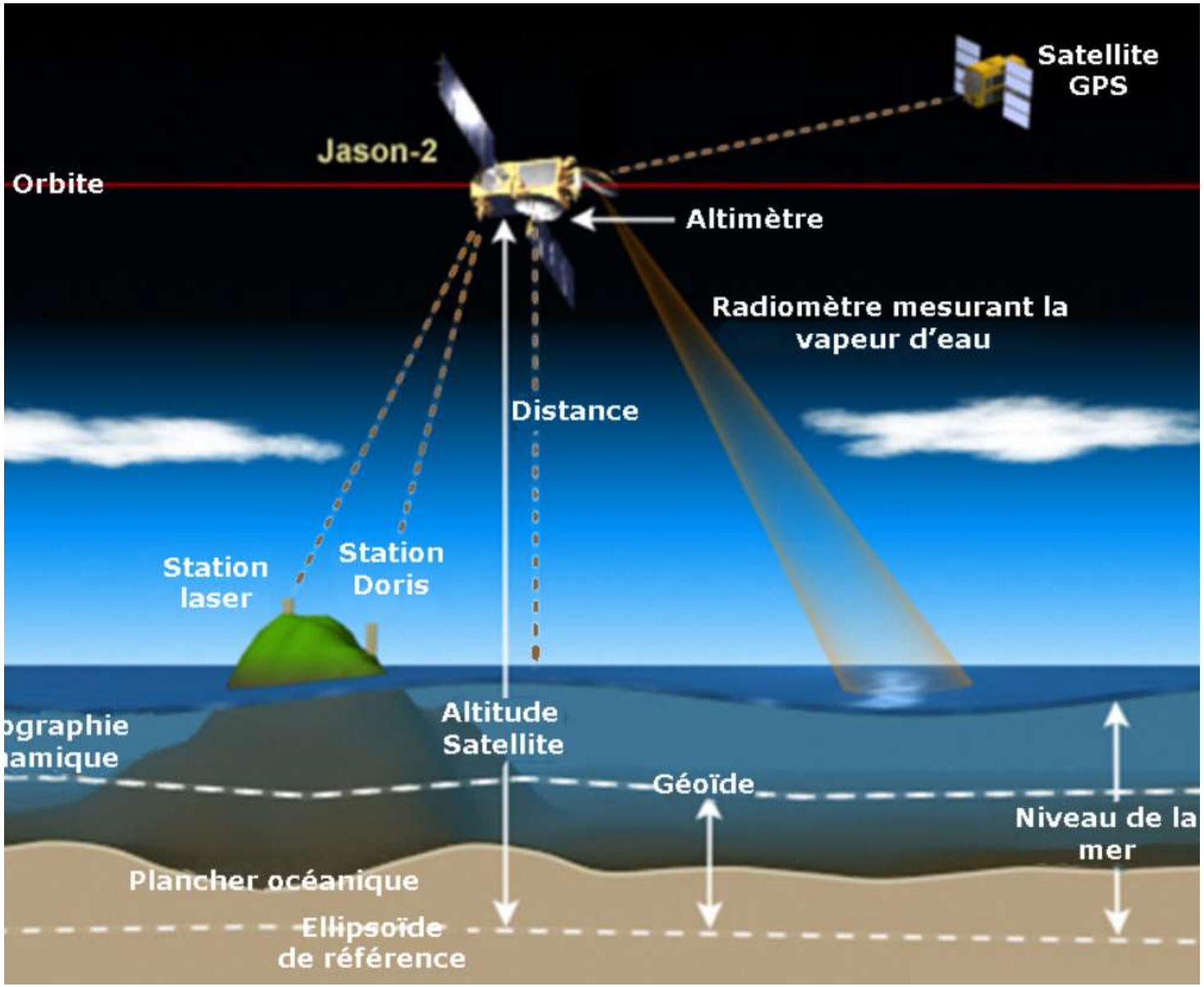
3- Propagation de l’onde

4-Réflexion de paquets d'onde sur la mer

5-Pénétration de l’onde électromagnétique dans l’eau

6-L’allure de l’océan

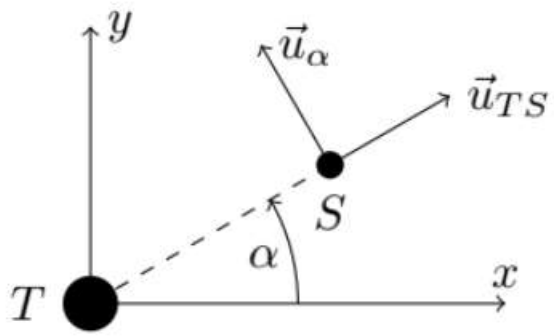
7-Conclusion



1-
Introduction sur les techniques de télédétection

2-ETUDE DE L'ORBITE :

Modélisation du problème :



*le référentiel géocentrique (Rg) est considéré comme galiléen et seule l'action de la terre est prise en compte . $\vec{F} = -GmM\dfrac{\overrightarrow{TS}}{TS^3}$

*la force de gravitation $\vec{F}$ qu'exerce la Terre (centre T, masse M) sur un satellite (point matériel S, masse m) :

*la force est centrale, le mouvement est donc à moment cinétique conservatif : $\dfrac{d\vec{\sigma}}{dt} = \overrightarrow{TS}\wedge\vec{F} = \vec{0} \Rightarrow \vec{\sigma} = m\overrightarrow{TS}\wedge\vec{v} = \overrightarrow{cste}$

Theorème de moment cinétique nous donne :

**********Le mouvement est donc plan **********

* La force est conservative, l'énergie mécanique se conserve

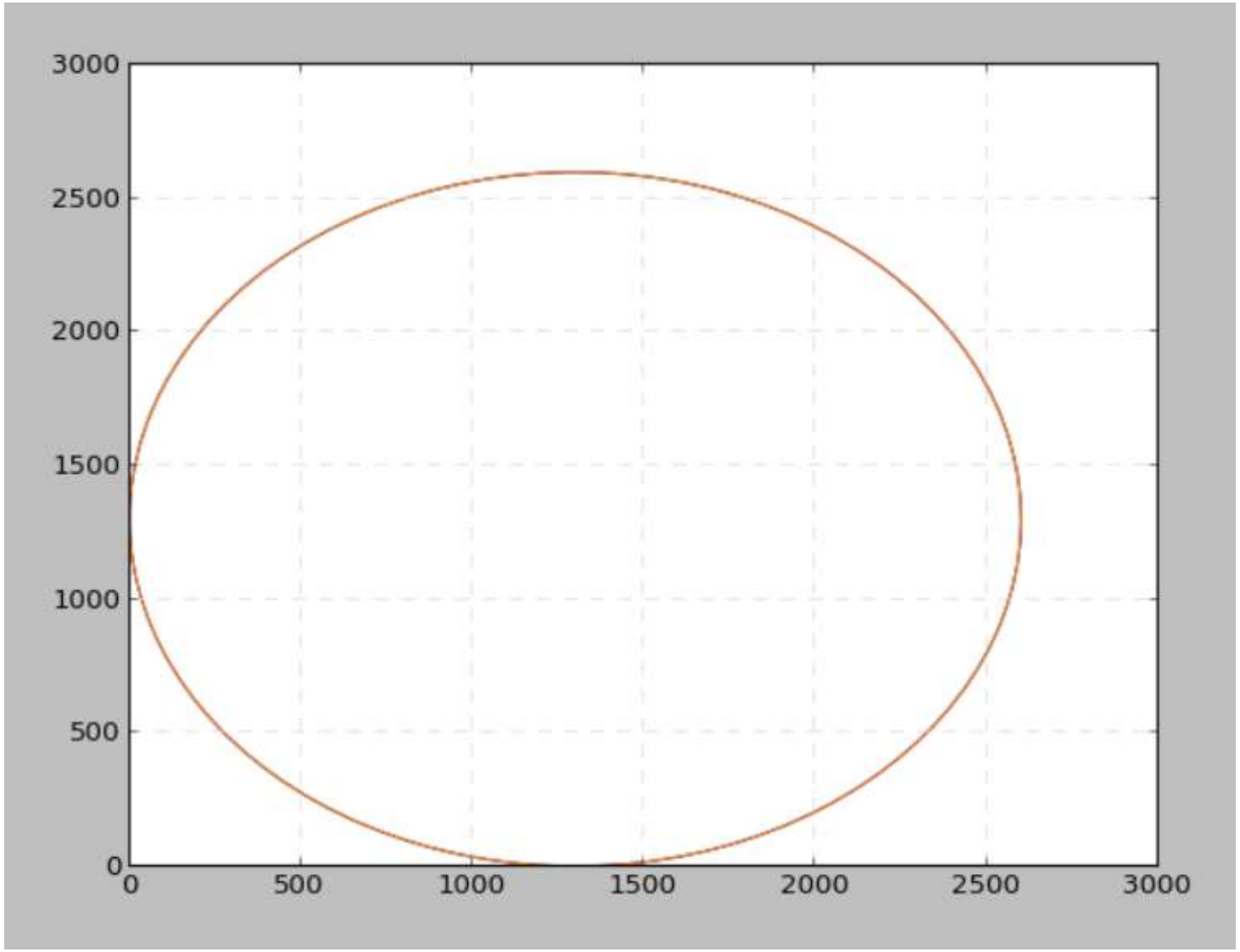
$$Em = Ec + Ep = \frac{1}{2}mv^2 - \frac{GmM}{r}$$

Le mouvement est donc plan

La deuxième loi de newton nous donne : $v = \sqrt{\frac{GM_T}{TS}}$

La première loi de Kepler nous donne : le mouvement du satellite est un ellipse

La période de révolution du satellite: $T = \frac{2\pi TS}{V} \Leftrightarrow T = 2\pi\sqrt{\frac{TS^3}{GM}}$



Trajectoire de satellite Jason 3

(annexe code 1)

3-Propagation de l'onde

*Un gaz ionisé

*dans le plasma les particules sont soumises à la force de Lorenz : $\overrightarrow{Fl} = q\left(\vec{E} + \vec{v} \wedge \vec{B}\right)$

Due au champs électriques et magnétiques de l'onde .

* En négligeant l'effet de la pesanteur et les interactions entre les particules chargées, et on supposant que les particules sont non relativistes

* applicant le PFD sur les électrons : $m\frac{d\vec{v}}{dt} = \overrightarrow{Fl} \Leftrightarrow m\frac{d\vec{v}}{dt} = q\vec{E}$

D'où $\vec{v} = \frac{q}{jm\omega}\vec{E}$

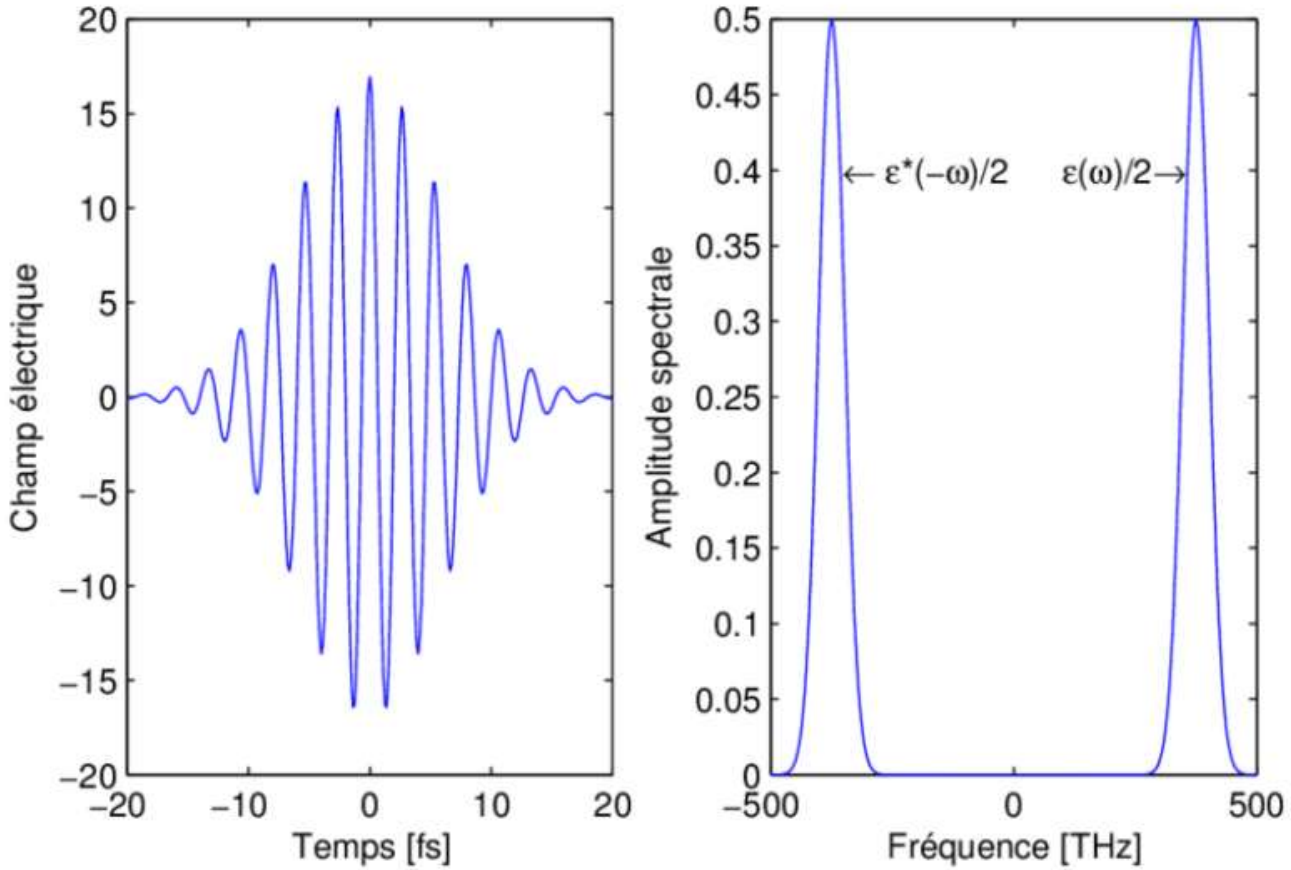
Avec q=-e pour les électrons et q=e pour les cations

Or on a : $\vec{j} = \sum_i n_i q \ {}_i\vec{v_i} \Rightarrow \vec{j} = \frac{n_0 * e^2}{j\omega}\left(\frac{1}{m_e} + \frac{1}{m_c}\right)$

Et puisque $m_c \gg m_e$ alors $\overrightarrow{AB} = \sigma\vec{E} = \left(\frac{n_0 e^2}{j m_e \omega}\right)\vec{E}$

- D'après les équations de Maxwell et on appliquant $\overrightarrow{rot}\left(\overrightarrow{rot}(\vec{E})\right) = \overrightarrow{grad}\left(div(\vec{E})\right) - \Delta\vec{E}$
- on obtient $\Delta\vec{E} = \mu_0\varepsilon_0\frac{d^2\vec{E}}{dt^2} + \mu_0\sigma\frac{d\vec{E}}{dt}$
- Résolution :
- on cherche une solution de la forme : $\underline{\vec{E}} = \underline{\overrightarrow{E_0}}\, e^{i(\omega t - \vec{k}.\vec{r})}$
- On obtient la relation de dispersion suivante : $\underline{k}^2 = \frac{\omega^2 - \omega_p^2}{c^2}$

Avec : $\omega_p^2 = \frac{\mu_0 c^2 n e^2}{m}$



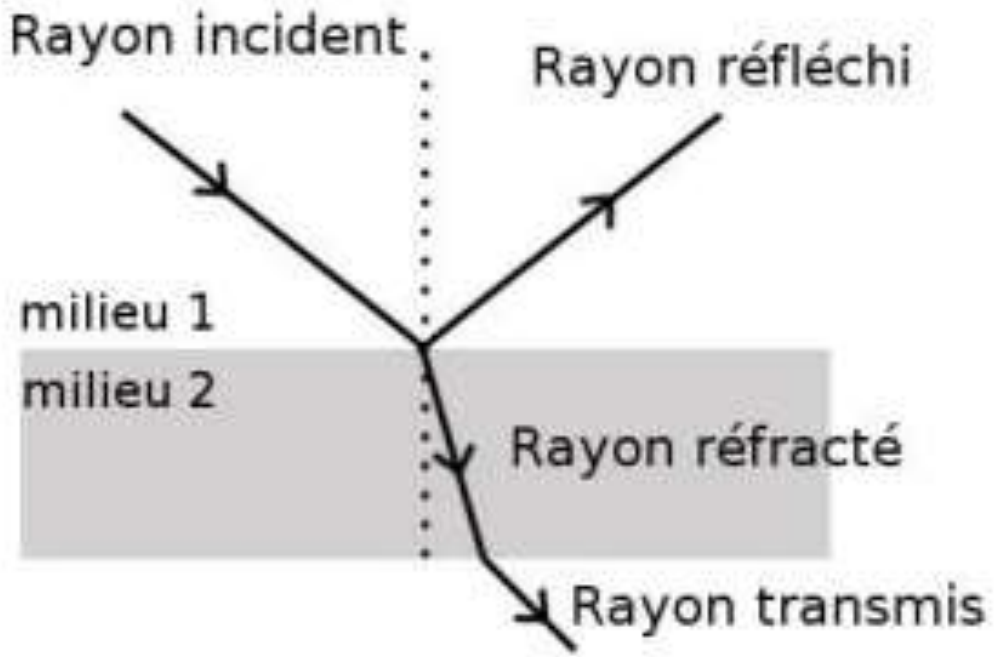
Définition d'un paquet d'onde

un **paquet d'onde**, ou **train d'onde**, est un ensemble des OPPM de pulsations voisines

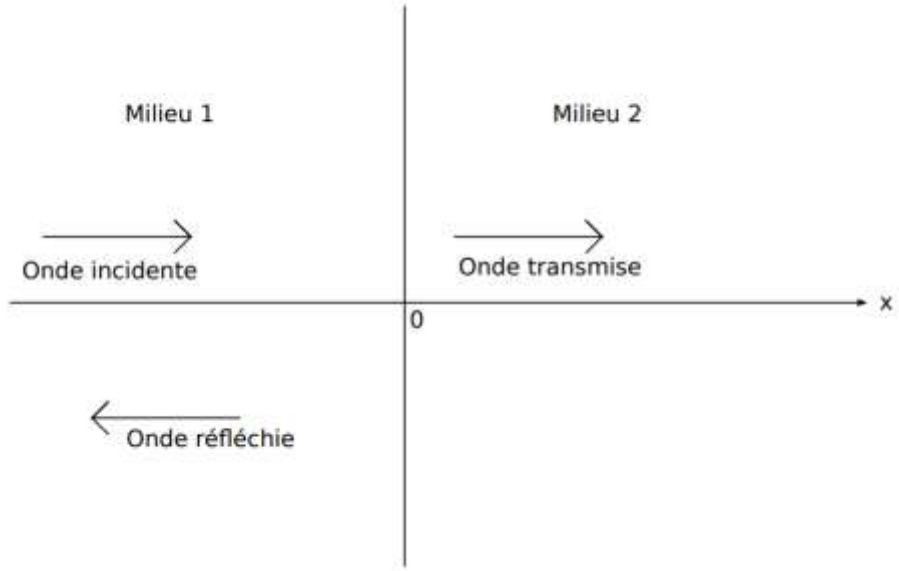
Leurs pulsations sont comprises entre

$$\omega_0 + \frac{\Delta\omega}{2} \qquad \text{et} \qquad \omega_0 - \frac{\Delta\omega}{2}$$

avec $\Delta\omega \ll \omega_0$



Réflexion et réfraction



4-Réflexion de paquets d'onde sur la mer

On peut considérer la mer comme un milieu semi fini

On dénit le paquet d'onde par un spectre en pulsation : $A(\omega)$

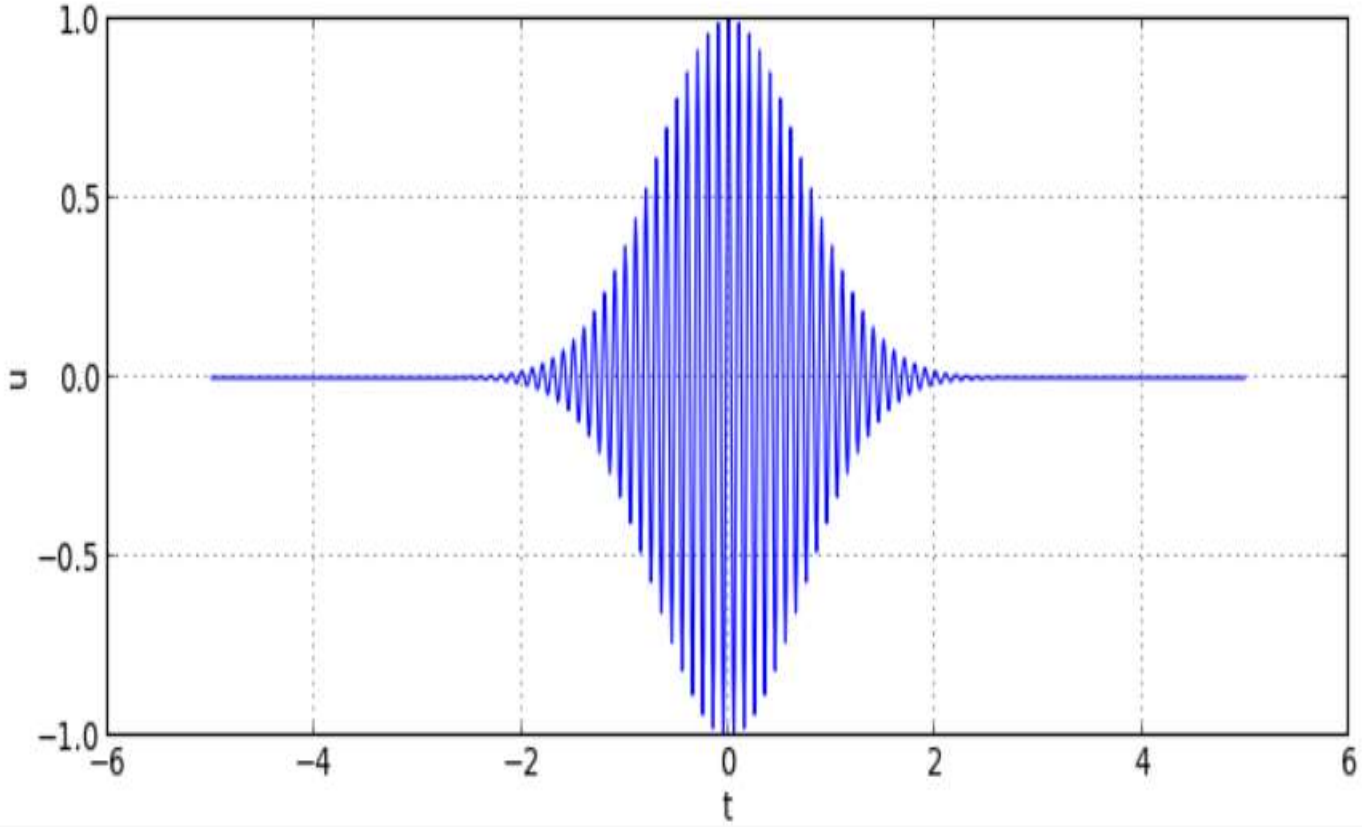
Le spectre souvent utilisé pour définir le paquet est le spectre gaussien

la fonction d'onde s'écrit : $\psi(x,t) = \sum_{n=N-P}^{N+P} A(n\omega_1) e^{i(k(n\omega_1)x - n\omega_1 t)}$

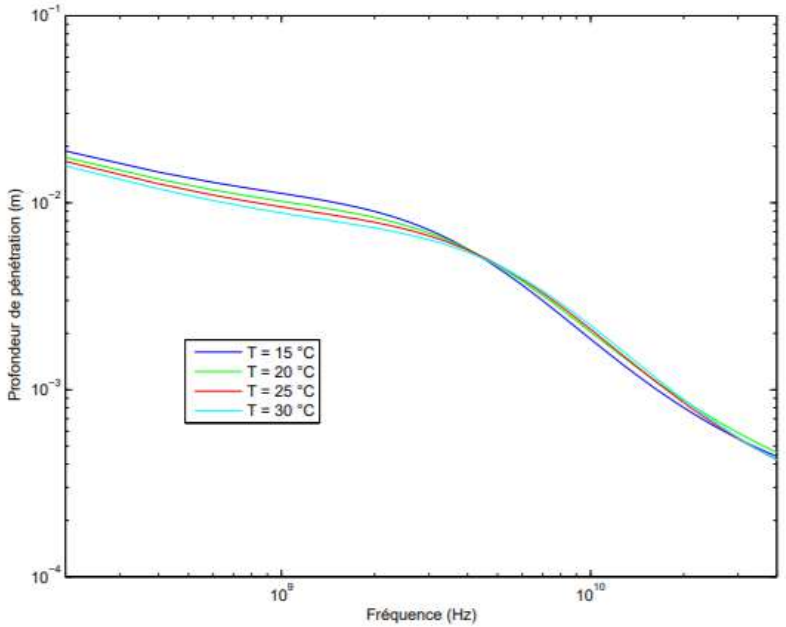
Avec ω_0 : la pulsation centrale et P le nombre de raies de part et d'autre

Dans le milieu 1: $\psi_1(x,t) = \sum_{n=N-P}^{N+P} A(n\omega_1) e^{i(k_1(n\omega_1)x - n\omega_1 t)} + \sum_{n=N-P}^{N+P} A(n\omega_1) r(n\omega_1) e^{i(-k_1(n\omega_1)x - n\omega_1 t)}$

Dans le milieu 2 : $\psi_2(x,t) = \sum_{n=N-P}^{N+P} A(n\omega_1) A(n\omega_1) e^{i(k_2(n\omega_1)x - n\omega_1 t)}$



Voir annexe code 2



(b) Salinité 35‰, température variable.

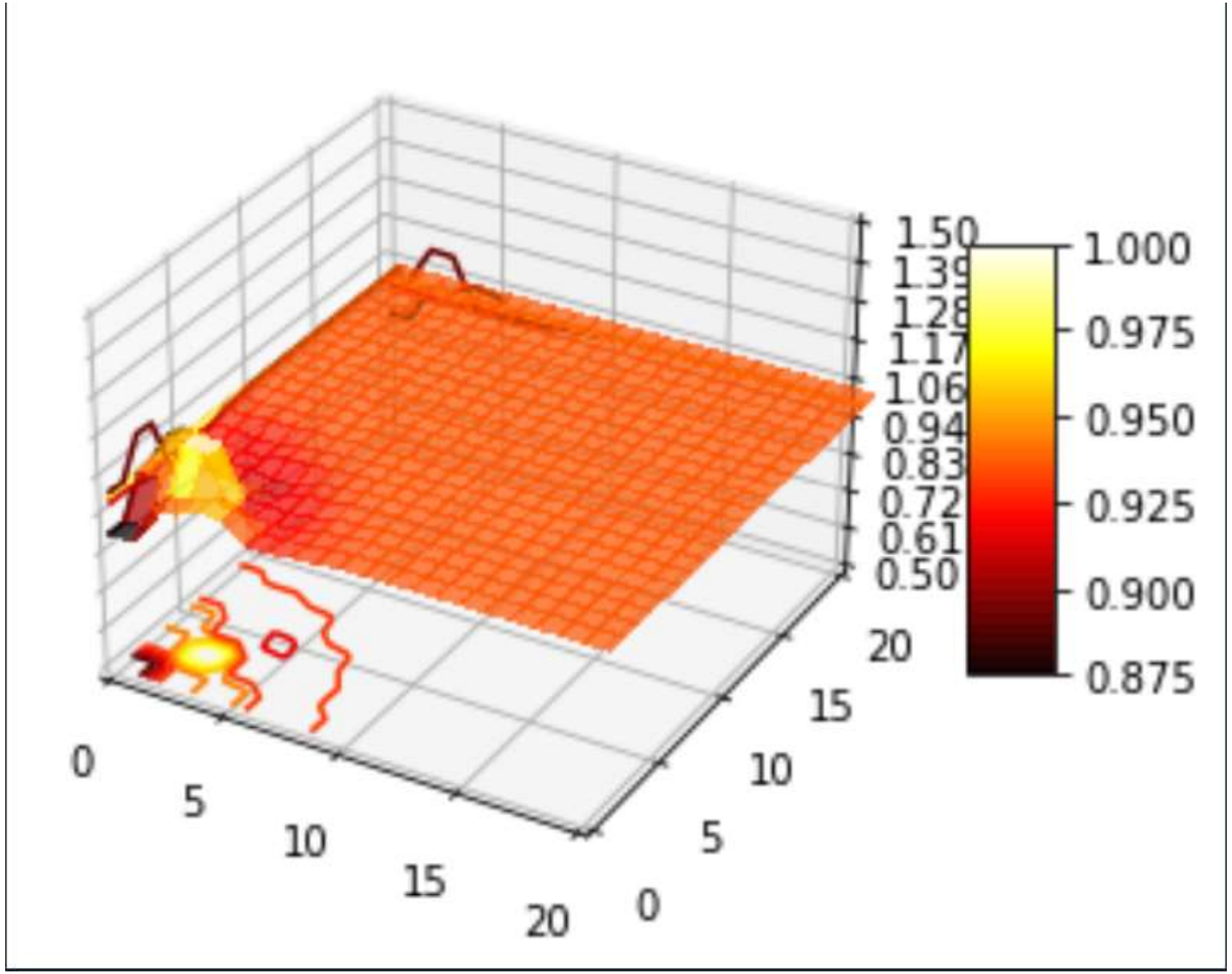
5-Pénétration de l'onde électromagnétique dans l'eau

On suppose que l'atmosphère, d'où provient l'onde incidente, est quasiment équivalente au vide. Le milieu d'incidence sera caractérisé par sa permittivité électrique relative ε et sa conductivité σ: $\eta = \varepsilon - i60\sigma\lambda$

On aura alors l'angle de perte de milieu : $\alpha = \arctan\left(\frac{i60\sigma\lambda}{\varepsilon}\right)$

L'épaisseur de peau s'écrit alors : $\delta = \frac{\lambda}{2\pi\sqrt{\frac{\varepsilon}{\cos\alpha}}\sin\frac{\alpha}{2}}$

la figure dessous illustre la variation de l'épaisseur de peau de l'onde électromagnétique radar dans l'eau de mer, pour différentes températures:



6-L’allure de l’océan

Annexe code 3

7-Conclusion

Le satellite Jason a offert aux pays une possibilité de sauver la vie des milliers des gens en déclenchant une alerte au cas de détection des variations brutales de la surface de l’océan . Cette détection se fait en calculant les altitudes instantanées de la surface de l’eau.

MERCI POUR VOTRE ATTENTION

Annexe

Code 1 :

```
from matplotlib import pyplot as plt
from math import pi, cos, sin

u=1300      #x-position of the center     #radius on the y-axis
t_rot=pi*0.66 #rotation angle

t = np.linspace(0, 2*pi, 100)
Ell = np.array([u*np.cos(t) , u*np.sin(t)])
     #u,v removed to keep the same center location
R_rot = np.array([[cos(t_rot) , -sin(t_rot)],[sin(t_rot) , cos(t_rot)]])
     #2-D rotation matrix

Ell_rot = np.zeros((2,Ell.shape[1]))
for i in range(Ell.shape[1]):
    Ell_rot[:,i] = np.dot(R_rot,Ell[:,i])

plt.plot( u+Ell[0,:] , u+Ell[1,:] )     #initial ellipse
plt.plot( u+Ell_rot[0,:] , u+Ell_rot[1,:],'darkorange' )    #rotated ellipse
plt.grid(color='lightgray',linestyle='--')
plt.show()
```

Annexe

Code 2 :

```
import math
import numpy as np
from matplotlib.pyplot import *
from numpy.fft import fft
a=1.0
def signal(t):
    return math.exp(-t**2/a**2)
def tracerSpectre(fonction,T,fe):
    t = np.arange(start=-0.5*T,stop=0.5*T,step=1.0/fe)
    echantillons = t.copy()
    for k in range(t.size):
        echantillons[k] = fonction(t[k])
    N = echantillons.size
    tfd = fft(echantillons)/N
    spectre = T*np.absolute(tfd)
    freq = np.zeros(N)
    for k in range(N):
       freq[k] = k*1.0/T
    plot(freq,spectre,'r.')
    xlabel('f')
    ylabel('S')
    axis([0,fe,0,spectre.max()])
    grid()
    return tfd

a=1.0
b=0.1
def signal(t):
    return np.exp(-t**2/a**2)*np.cos(2.0*math.pi*t/b)
t = np.arange(start=-5,stop=5,step=0.01)
u = signal(t)
figure(figsize=(10,4))
plot(t,u)
xlabel('t')
ylabel('u')
grid()
```

Annexe

Code 3 (partie 1)

```
import numpy as np

n = 20;

k = 10;
dt = 0.02;
dx = 1.0;
dy = 1.0;

h = np.ones((n+2,n+2))
u = np.zeros((n+2,n+2))
v = np.zeros((n+2,n+2))

hx = np.zeros((n+1,n+1))
ux = np.zeros((n+1,n+1))
vx = np.zeros((n+1,n+1))

hy = np.zeros((n+1,n+1))
uy = np.zeros((n+1,n+1))
vy = np.zeros((n+1,n+1))

nsteps = 0
h[1,1] = .5;

def reflective():
    h[:,0] = h[:,1]
    h[:,n+1] = h[:,n]
    h[0,:] = h[1,:]
    h[n+1,:] = h[n,:]
    u[:,0] = u[:,1]
    u[:,n+1] = u[:,n]
    u[0,:] = -u[1,:]
    u[n+1,:] = -u[n,:]
    v[:,0] = -v[:,1]
    v[:,n+1] = -v[:,n]
    v[0,:] = v[1,:]
    v[n+1,:] = v[n,:]
```

```
def proses():
    #hx = (h[1:,:]+h[:-1,:])/2-dt/(2*dx)*(u[1:,:]-u[:-1,:])
    for i in range (n+1):
        for j in range(n):
            hx[i,j] = (h[i+1,j+1]+h[i,j+1])/2 - dt/(2*dx)*(u[i+1,j+1]-u[i,j+1])
            ux[i,j] = (u[i+1,j+1]+u[i,j+1])/2- dt/(2*dx)*((pow(u[i+1,j+1],2)/h[i+1,j+1]+ k/2*pow(h[i+1,j+1],2))
            vx[i,j] = (v[i+1,j+1]+v[i,j+1])/2 - dt/(2*dx)*((u[i+1,j+1]*v[i+1,j+1]/h[i+1,j+1]) - (u[i,j+1]*v[i,j

    for i in range (n):
        for j in range(n+1):
            hy[i,j] = (h[i+1,j+1]+h[i+1,j])/2 - dt/(2*dy)*(v[i+1,j+1]-v[i+1,j])
            uy[i,j] = (u[i+1,j+1]+u[i+1,j])/2 - dt/(2*dy)*((v[i+1,j+1]*u[i+1,j+1]/h[i+1,j+1]) - (v[i+1,j]*u[i+1
            vy[i,j] = (v[i+1,j+1]+v[i+1,j])/2 - dt/(2*dy)*((pow(v[i+1,j+1],2)/h[i+1,j+1] + k/2*pow(h[i+1,j+1],2

    for i in range (1,n+1):
        for j in range(1,n+1):
            h[i,j] = h[i,j] - (dt/dx)*(ux[i,j-1]-ux[i-1,j-1]) - (dt/dy)*(vy[i-1,j]-vy[i-1,j-1])
            u[i,j] = u[i,j] - (dt/dx)*((pow(ux[i,j-1],2)/hx[i,j-1] + k/2*pow(hx[i,j-1],2)) - (pow(ux[i-1,j-1],2
            v[i,j] = v[i,j] - (dt/dx)*((ux[i,j-1]*vx[i,j-1]/hx[i,j-1]) - (ux[i-1,j-1]*vx[i-1,j-1]/hx[i-1,j-1]))

#dh = dt/dt*(ux[1:,:]-ux[:-1,:])+ dt/dy*(vy[:,1:]-vy[:,:-1])
    reflective()
    return h,u,v
'''
for i in range (17):
    #print h
    proses(1)
'''

import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
from mpl_toolkits.mplot3d import Axes3D
a = n
x = np.arange(n+2)
y = np.arange(n+2)
x,y = np.meshgrid(x,y)

fig = plt.figure()

ax = fig.add_subplot(111, projection='3d')

def plotset():
```

Annexe

Code 3 (partie 2)

```
def plotset():
    ax.set_xlim3d(0, a)
    ax.set_ylim3d(0, a)
    ax.set_zlim3d(0.5,1.5)
    ax.set_autoscalez_on(False)
    ax.zaxis.set_major_locator(LinearLocator(10))
    ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
    cset = ax.contour(x, y, h, zdir='x', offset=0 , cmap=cm.hot)
    cset = ax.contour(x, y, h, zdir='y', offset=n , cmap=cm.hot)
    cset = ax.contour(x, y, h, zdir='z', offset=.5, cmap=cm.hot)

plotset()

surf = ax.plot_surface(x, y, h,rstride=1, cstride=1,cmap=cm.hot,l

fig.colorbar(surf, shrink=0.5, aspect=5)


from matplotlib import animation
```

```
from matplotlib import animation


def data(k,h,surf):
    proses()
    ax.clear()
    plotset()
    surf = ax.plot_surface(x, y, h,rstride=1, c
    return surf,

ani = animation.FuncAnimation(fig, data, fargs=
ani.save('laser.mp4', bitrate=512)
plt.show()
```

Annexe

Code 3 (partie4)