Vers un transport rapide et volumineux de l'information

Comment peut on transporter une quantité importante d'information

Dans une fibre optique?

34143

Plan:

I. Modélisation physique du phénomène.

II. Etude expérimentales

III. Simulation numérique du phénomène.

IV. Conclusion.

Contributions:

- Rencontres avec plusieurs physiciens tunisiens et français
- Prise de contact avec le monde de recherche scientifiques(Visites des laboratoires de recherches)
- Implémentation de plusieurs codes de simulation numérique
- Comparaison avec les résultats expérimentaux

Equations de Maxwell dans un milieu Matériel

1) *Forme locale du théorème de Gauss:* $div(\vec{E}) = 0$ *car le milieu est supposé localement neutre*

2) $div(\vec{B}) = 0$ car le champ $\vec{B}$ est toujours à flux conservatif

3) Forme locale de Maxwell-Faraday qui traduit le phénomène d'induction: $\overrightarrow{rot}(\vec{E}) = -\frac{\partial \vec{B}}{\partial t}$

4) Forme locale du théorème de Maxwell-Ampère

$\overrightarrow{rot}(\vec{B}) = \mu_0(\overrightarrow{J_c} + \varepsilon_0 \frac{\partial \vec{E}}{\partial t})$ c'est l'équation de Maxwell-Ampère en régime variable. Pour un milieu conducteur Ohmique $\overrightarrow{J_c} = \gamma \vec{E}$ où γ est la conductivité du milieu. Soit $\overrightarrow{rot}(\vec{B}) = \mu_0(\gamma \vec{E} + \varepsilon_0 \frac{\partial \vec{E}}{\partial t})$

Equation de la propagation

Soit $\overrightarrow{rot}\left(\overrightarrow{rot}(\vec{E})\right) = \overrightarrow{grad}(div(\overrightarrow{E})) - \Delta(\vec{E}) = -\Delta(\vec{E})$ car $div(\vec{E}) = 0$

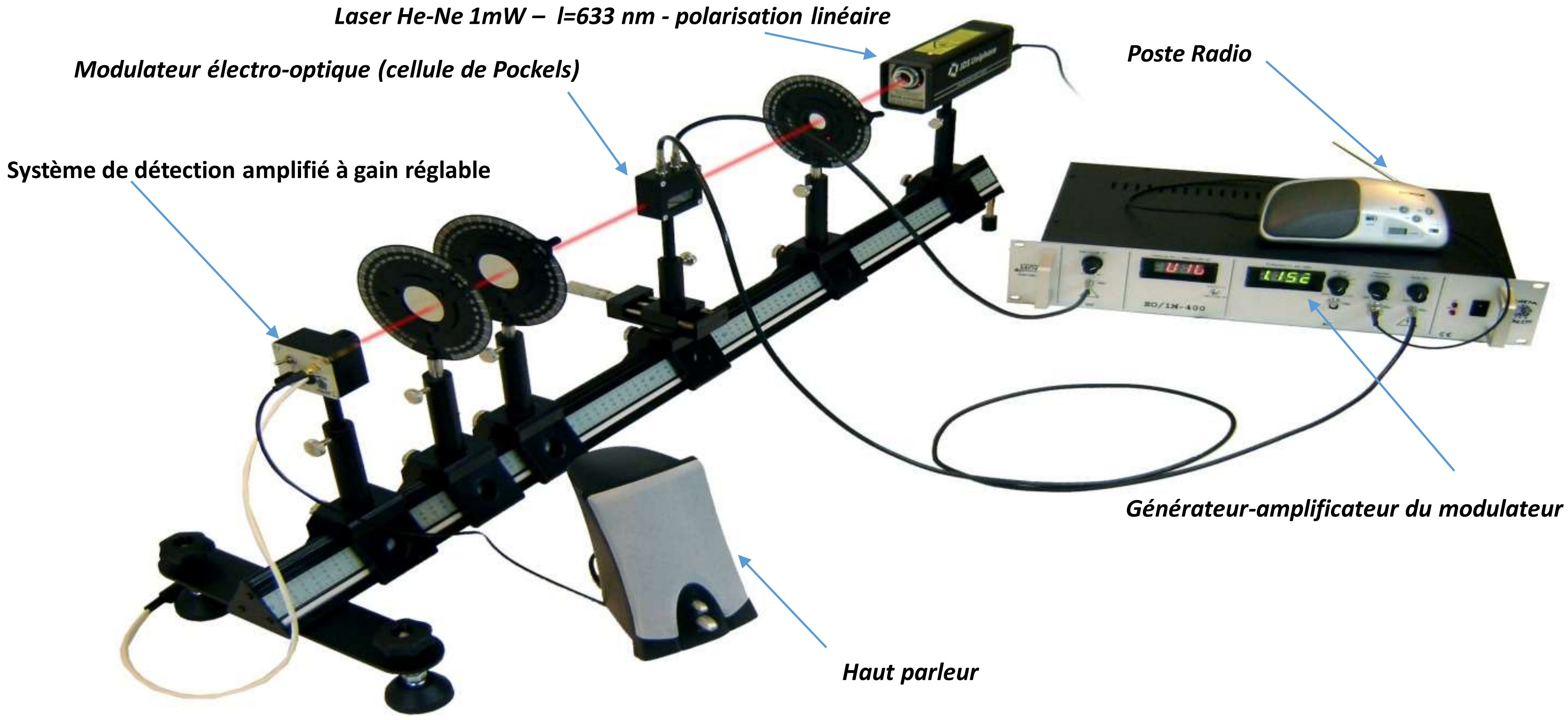
Or $\overrightarrow{rot}\left(\overrightarrow{rot}(\vec{E})\right) = \overrightarrow{rot}\left(-\frac{\partial \vec{B}}{\partial t}\right) = -\frac{\partial}{\partial t}\overrightarrow{rot}(\vec{B}) = -\frac{\partial}{\partial t}(\mu_0\left(\overrightarrow{J_c} + \varepsilon_0\frac{\partial \vec{E}}{\partial t}\right) =$

$-\frac{\partial}{\partial t}\mu_0\left(\gamma\vec{E} + \varepsilon_0\frac{\partial \vec{E}}{\partial t}\right) = -\varepsilon_r\mu_0\varepsilon_0\frac{\partial^2}{\partial t^2}(\vec{E}) - \mu_0\gamma\frac{\partial \vec{E}}{\partial t}$ avec $\mu_0\varepsilon_0 = \frac{1}{C^2}$

Soit : $$\Delta(\vec{E}) - \frac{\varepsilon_r}{C^2}\frac{\partial^2}{\partial t^2}(\vec{E}) - \mu_0\gamma\frac{\partial \vec{E}}{\partial t} = \vec{0}$$ -

Etude de la transmission d’une lumière porteuse d’une information à travers une fibre optique:

Chaine de transmission numérique

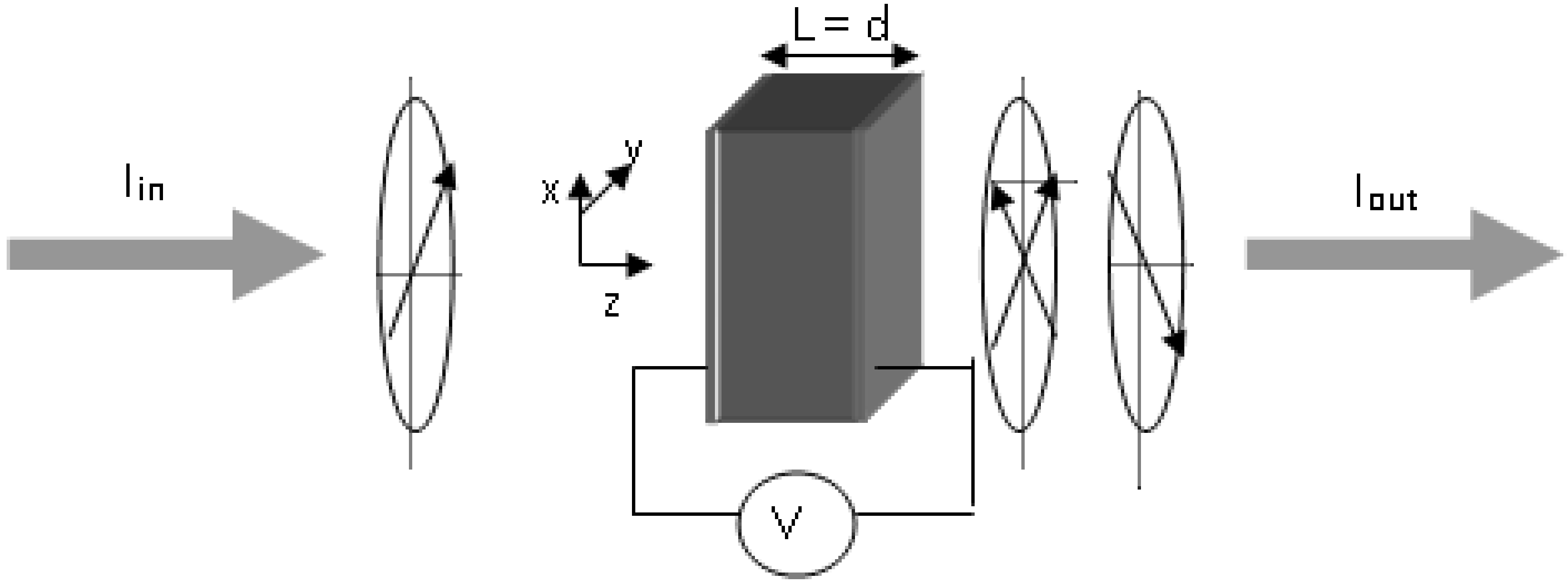


Principe de Modulation électro optique de phase (Modélisation)

- **L'application d'une tension V au borne du cristal provoque une variation d'indice dans ce milieu et le rend biréfringent ($n_x \neq n_y$)**
- **La variation d'indice de réfraction ou de biréfringence est induite par effet EO**
- **Cette variation d'indice provoque un déphasage ϕ entre les deux composantes E_x et E_y du champ électrique sortant associé à l'onde lumineuse se propageant dans le cristal.**

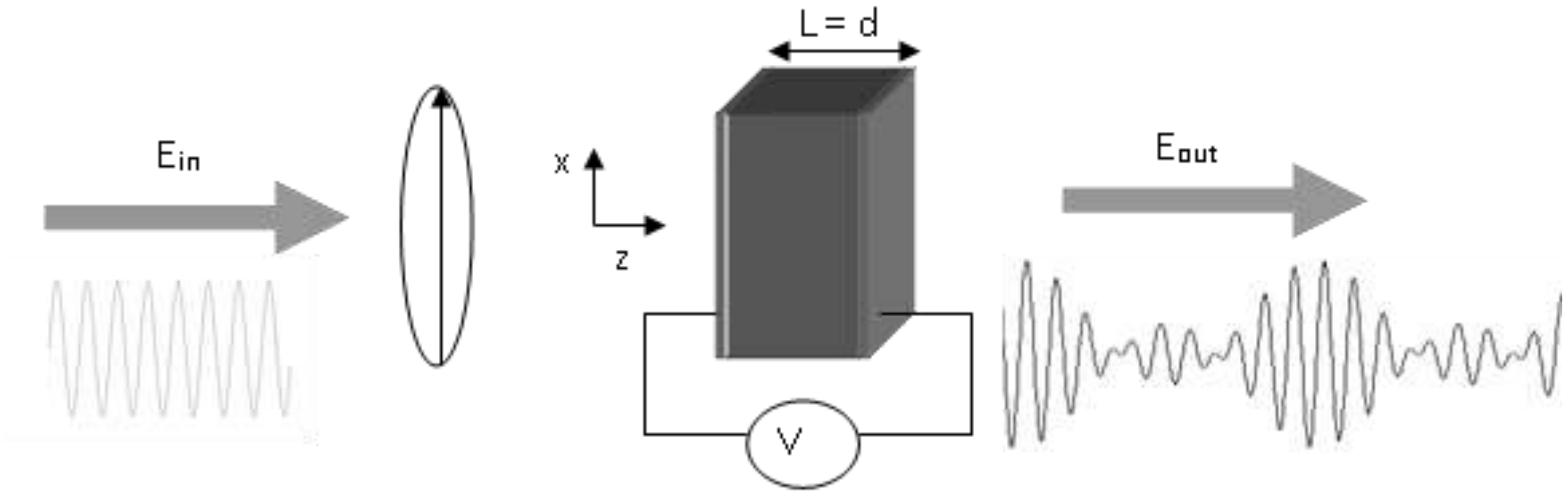
Différents types de modulation électro-optiques:

1) Modulation électro-optique d'amplitude



- Le cristal est en configuration longitudinale
- Les axes principaux du cristal sont à 45° de deux polariseurs croisés,
- Les axes principaux de la lame quart d'onde sont à 45° des axes du cristal

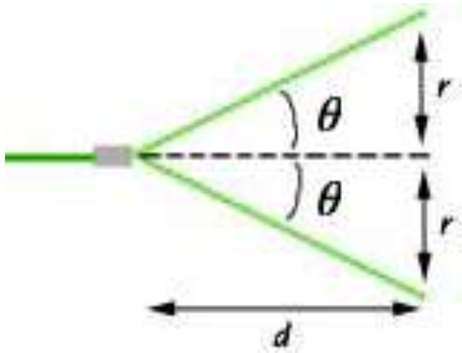
2) Modulation électro-optique de phase



- Le cristal est en configuration longitudinale
- Le faisceau lumineux est polarisé suivant l'axe principal x du cristal
- L'application de la tension ne change pas l'état de polarisation de l'onde mais uniquement sa phase

experience1

Détermination de l’ouverture numérique:



On mesure le diamètre D=2r de l’image placée sur l’écran et la distance d qui sépare l’écran de la sortie de la fibre.
Soit: (D/(2d))= tan(Θ): la pente de la courbe dessinée. ON= sin (Θ) et $ON^2 = (\sin(\Theta))^2 = 2n_0 * \Delta n$
Car on a $ON^2 = {n_0}^2 - {n_2}^2 = (n_0 + n_2) * (n_0 - n_2) = 2n_0 * \Delta n$ en utilisant l’approximation $n_0 + n_2 = 2n_0$
Soit: $\Delta n = \frac{ON^2}{2n_1}$ ce qui donne: $n_2 = n_0 - \Delta n$

d (cm)	5	7	9	11	13
D (cm)	2,8	4,8	6	7	9

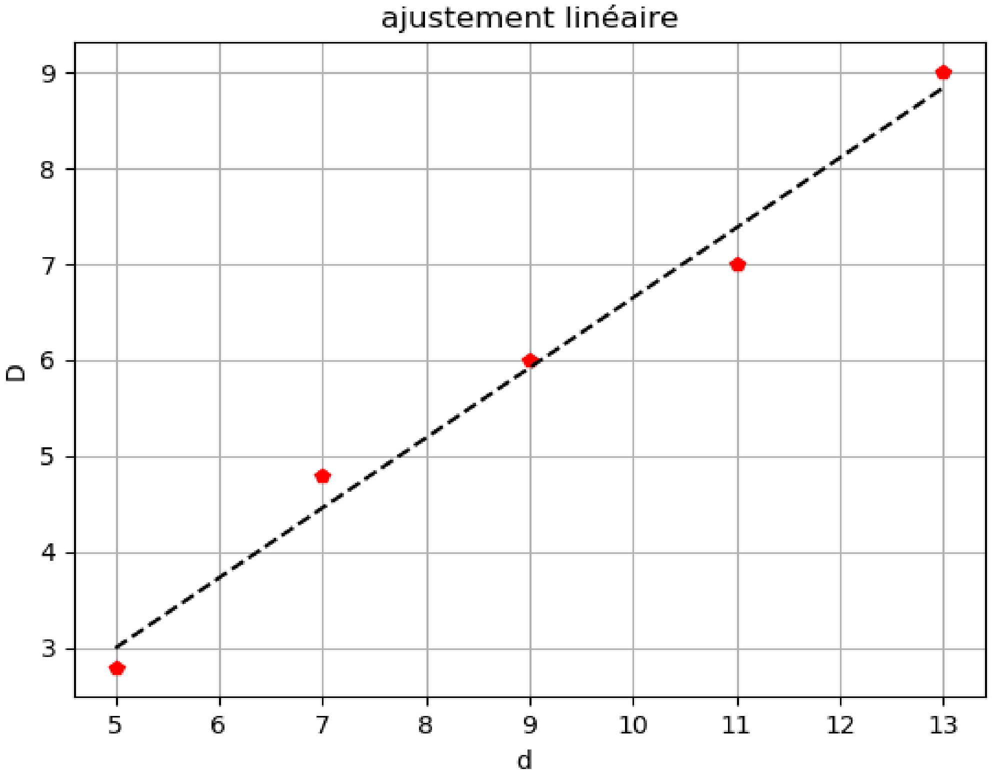
A partir du tableau précédent de mesures de d et D on déduit l'angle d'ouverture ϴ et Δn soient:

ϴ=26,565° et sin(ϴ)=0,447

Donc ON=0,447

$\rightarrow \Delta n = \frac{ON^2}{2n_0} = \frac{0{,}447^2}{2*1{,}488} = 0{,}06714$

$\rightarrow n_2 = n_0 - \Delta n = 1{,}488 - 0{,}06714 = 1{,}42086$



Experience2:

Profil d’indice d’une fibre :

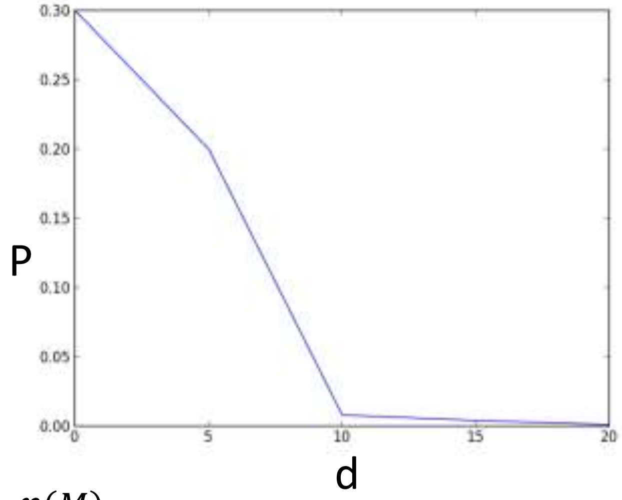
Le profil d’indice consiste à représenter la variation d’indice de la fibre en fonction de la distance r à son axe .
Dans une fibre multimode la puissance lumineuse acceptée en un point M de sa section est proportionnelle à la différence entre l’indice au point M et l’indice de la gaine.

Il a été montré que la puissance P(M) injectée au point M du cœur de la fibre peut se mettre sous la forme:
$\mathrm{P}(M) = \alpha[n(M) - n_2]$
Sur l’axe (point o) d’indice maximale la puissance est $p_{max} = \alpha[n(o) - n_2]$
$\alpha = \frac{p_{max}}{\Delta n}$ avec $\Delta n = n_0 - n_2$
Pa application numérique

$$\alpha = \frac{p_{max}}{\Delta n} = \frac{0,3}{0,06714} = 4,468mW$$

A partir du profil de la puissance P(M) en sortie de la fibre on peut déduire le profil d’indice.

P(mW)	0,3	0,02	0,008	0,004	0,001
d(µm)	0	5	10	15	20

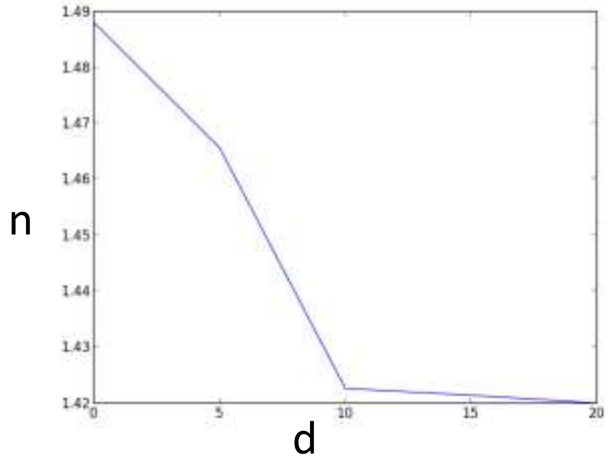


En utilisant l'équation $n(M) = \frac{p(M)}{\alpha} + n_2$ on calcule l'indice n en fonction de d.

Soit le tableau suivant:

n	1,488	1,4253	1,4226	1,4214	1,42
d(µm)	0	5	10	15	20

Courbe n=f(r)



Experience 3:

Evaluation des pertes:

on a

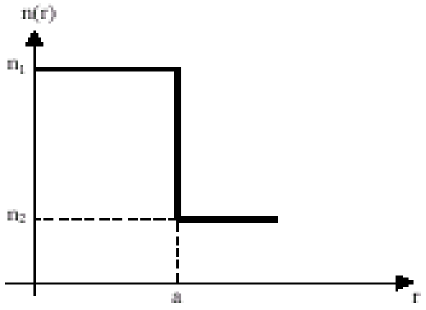
$p_e = 0{,}5\ mW$ puissance d'entrée

$p_s = 0{,}3\ mW$ puissance de sortie

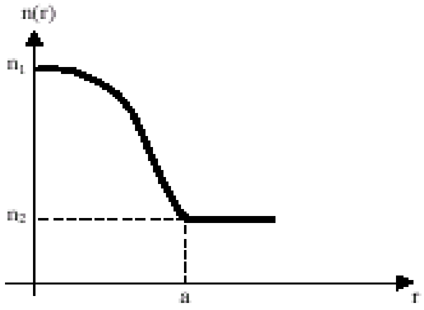
$perte_{(dB)} = 10 * \log(\frac{p_e}{p_s})$

$$perte_{(dB)} = 10 * \log\left(\frac{0{,}5}{0{,}3}\right) = 2{,}218$$

Le coefficient de perte doit être proche de 0 pour une fibre idéale et lorsqu'il augmente la fibre est moins efficace
Pour une conservation de la puissance.
Les pertes sont causées par l'absorption des électrons qui peuvent être porter à un autre niveau d'énergie,
Par diffusion et changement du diamètre du cœur générant des réflexions de certains rayons lumineux ,
Et les pertes de connexion-insertion dont le raccordement des fibres cause un gaspillage du puissance.



Fibre à saut d'indice

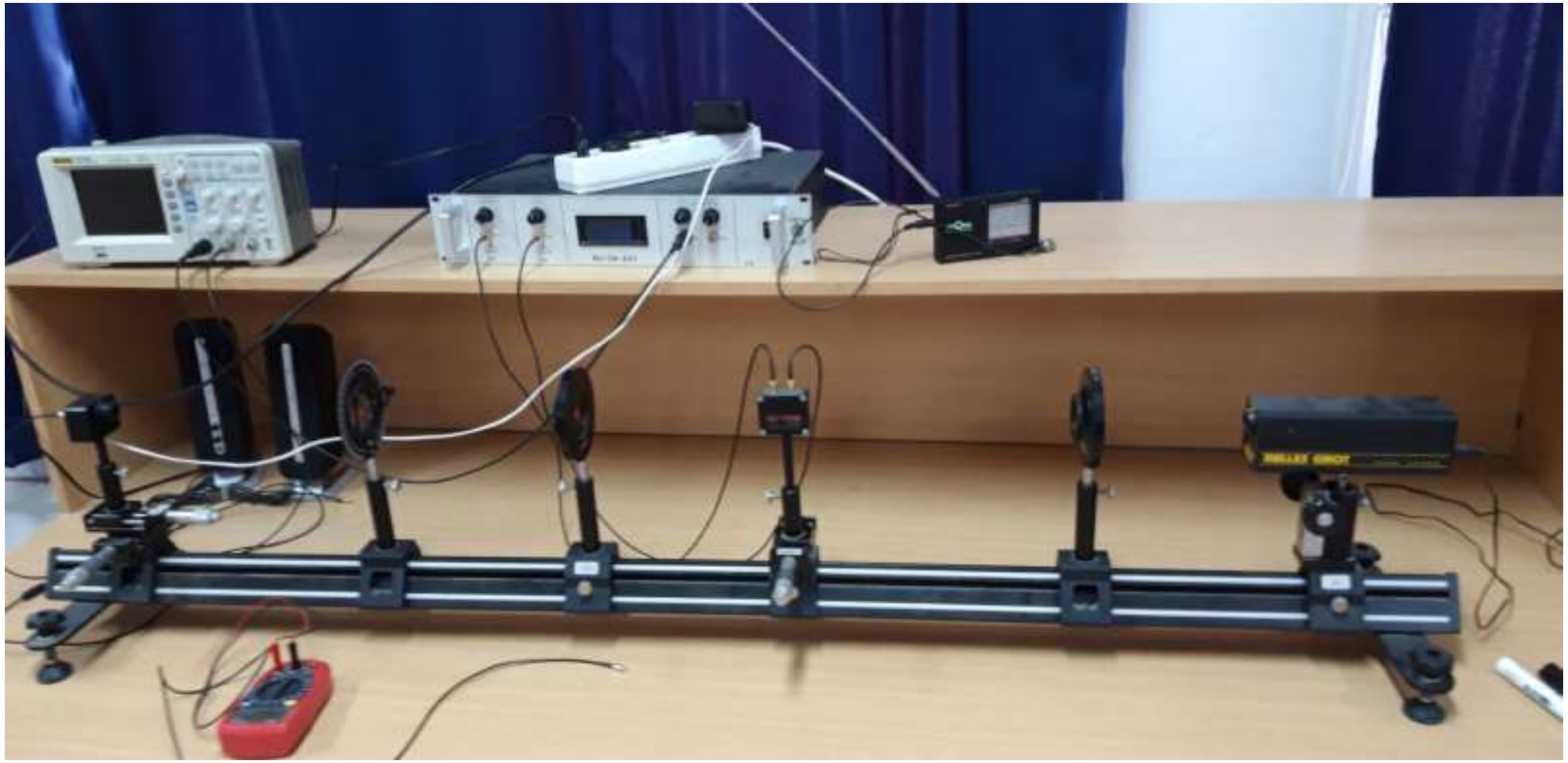


Fibre à gradient d'indice

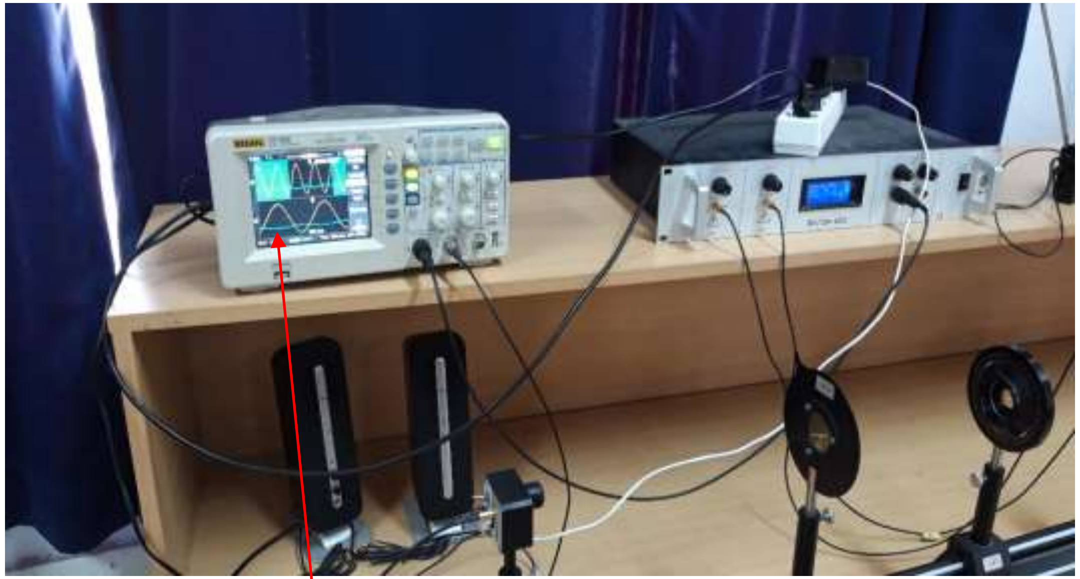
Durant tous ce travail on fait l'étude d'un fibre à gradient d'indice et on va bien la différence des courbes de n en fonction d entre ce fibre et le fibre à saut d'indice qui eux aussi diffèrent D'un point de vue matière.

Etude expérimentale de la modulation d’un signal

Dans cette étude nous avons utilisé le dispositif expérimental schématisé dans la photo suivante:



Modulateur



Visualisation des deux signaux d'entrée
et de sortie

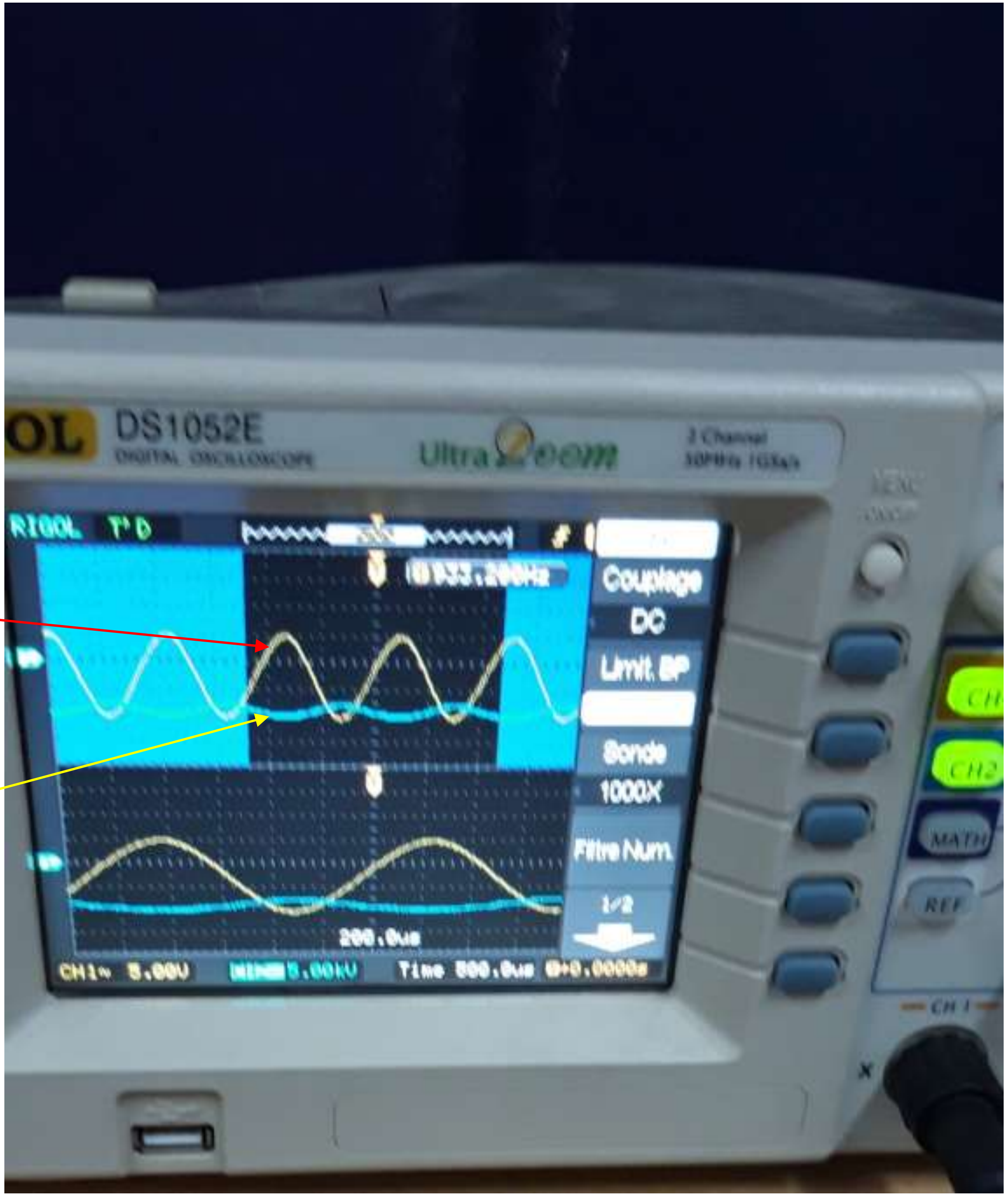
Absence du signal de sortie suite à l'introduction
D'un obstacle devant le faisceau lumineux

Ces testes de visualisations prouvent que l'information a été transmise par un faisceau laser

Experience4:

Signal d'entrée

Signal de sortie



Effet pockels pour un angle du polariseur autour de 90°

Experience5:

Dédoublement des fréquences du signal de sortie

Effet Kerr pour un angle du polariseur autour de 30°

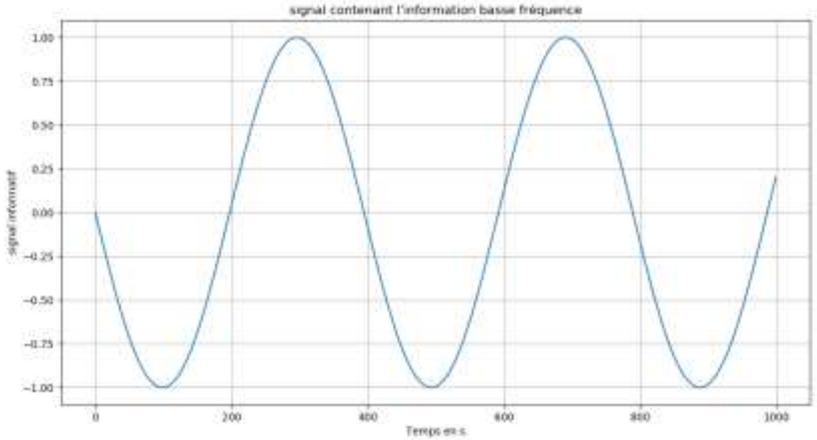
La biréfringence pour l'effet pockels qui commode la modulation est $\Delta n = \alpha E$
où n_o = 2,285 et r_{22} =6,4.10^{-12} SI sont respectivement l'indice de réfraction ordinaire et le coefficient électro-optique.
Pour l'effet kerr qui se manifeste avec un doublage de fréquence à la sortie de la manipulation
La biréfringence est $\Delta n = \alpha' E^2$ donc le champ se manifeste en E^2se qui fait:

$$E_m{}^2 cos^2(wt) = E_m{}^2 * (1 + \cos(2wt))$$

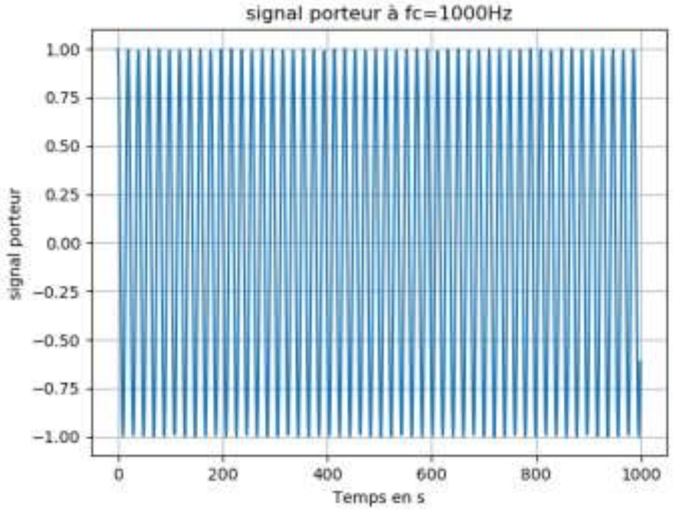
Ceci montre le dédoublement de fréquence à la sortie du Système de détection amplifié à gain réglable.

Modulation d’amplitude:

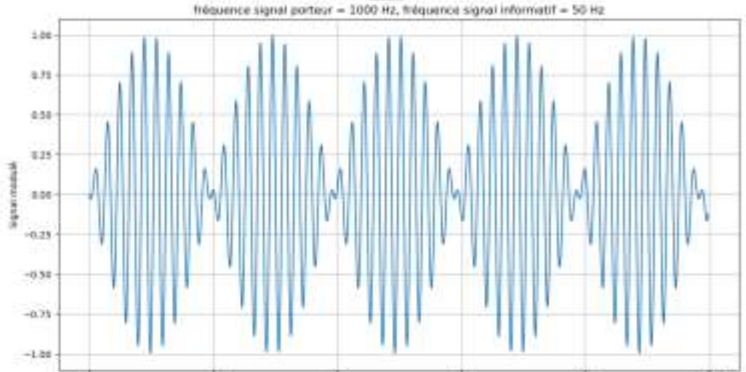
Si on fait le produit des deux
courbes des fréquences porteur
et informatif on aboutit au signal
modulé. A gauche le graphe montre un
Seul signal et à droite le graphe montre
Une superposition de quatre signaux

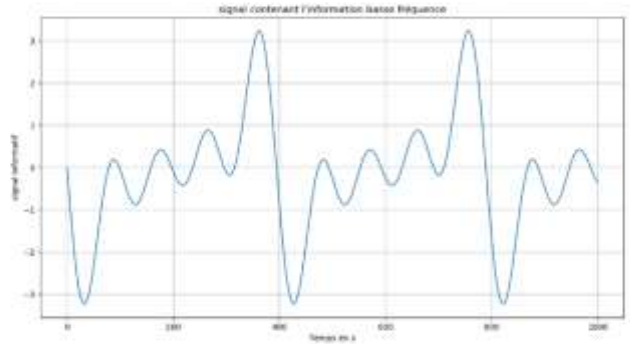


*

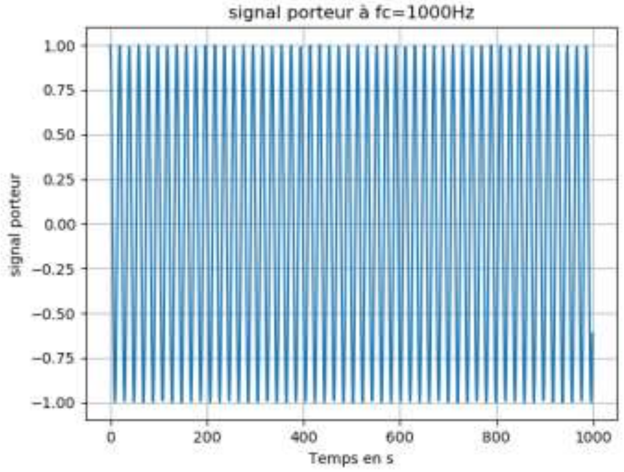


=

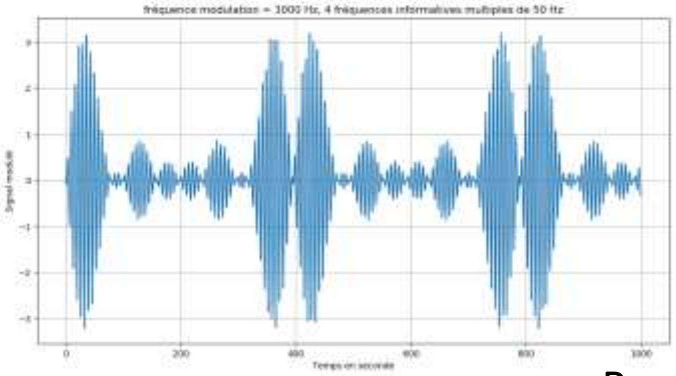




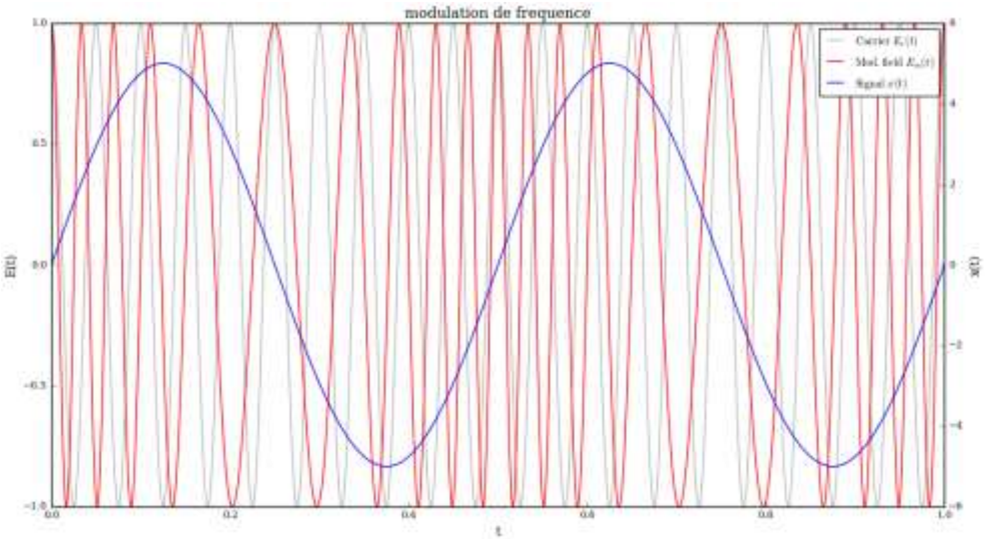
*



=



Modulation de fréquence:



Le signal de la fréquence porteuse s'écrit: $x_p(t) = A_p \cos(2\pi f_p t)$

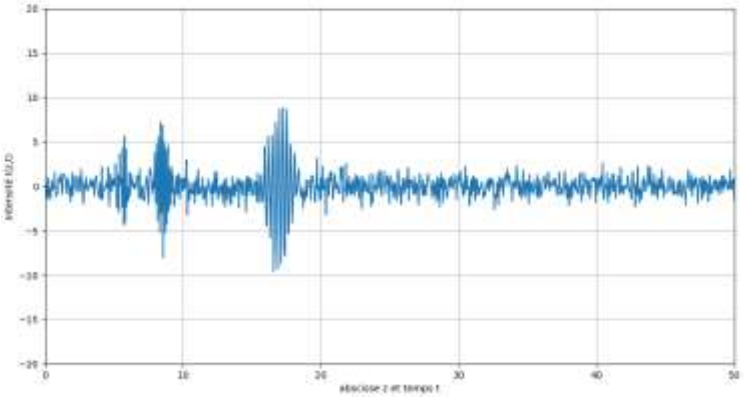
On a signal modulé s'écrit:$y(t) = A_p \cos\left(2\pi \int_0^t f(\tau)d\tau\right)$ avec $f(t) = f_p + f_{\Delta x_m}(t)$

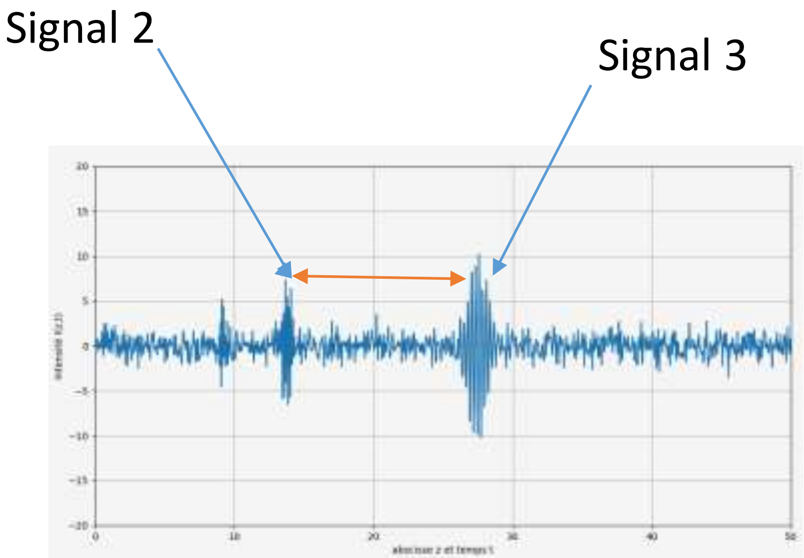
Donc $y(t) = A_p \cos\left(2\pi f_p t + 2\pi f_\Delta \int_0^t x_m(\tau)d\tau\right)$

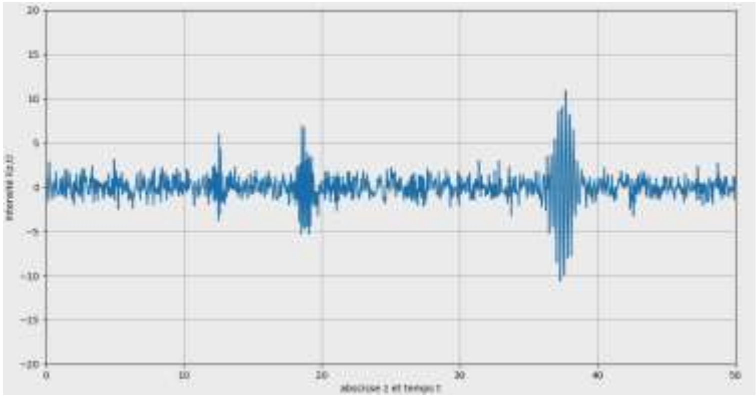
Donc $\int_0^t x_m(\tau)d\tau = \frac{A_m sinu(2\pi f_m t)}{2\pi f_m}$

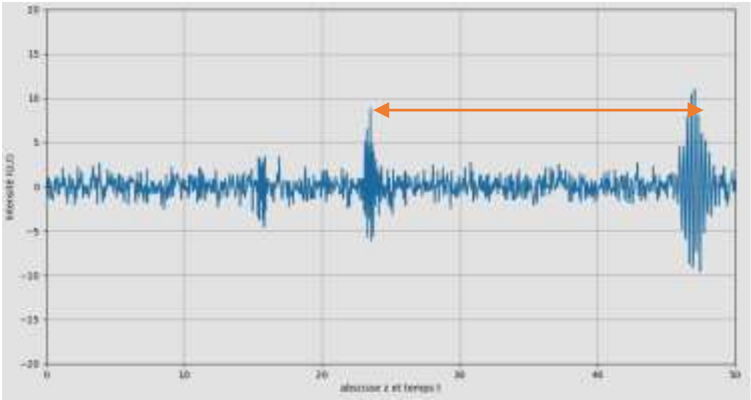
$\rightarrow y(t) = A_p \cos\left(2\pi f_p t + \frac{A_m}{f_m} f_\Delta \sin(2\pi f_m t)\right)$

Simulation numérique de la dispersion:



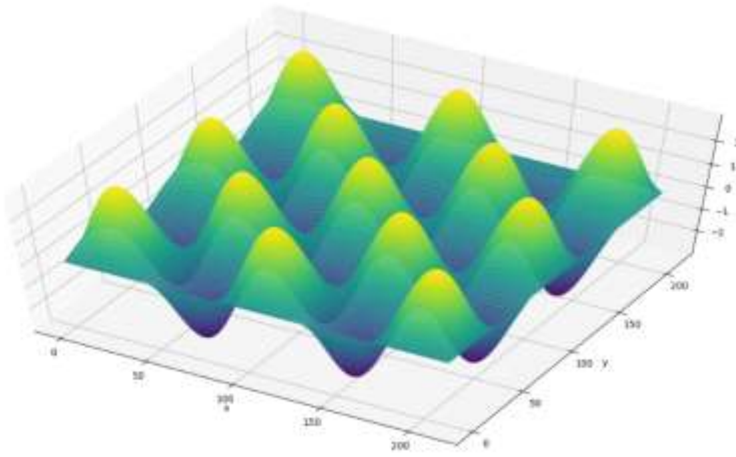


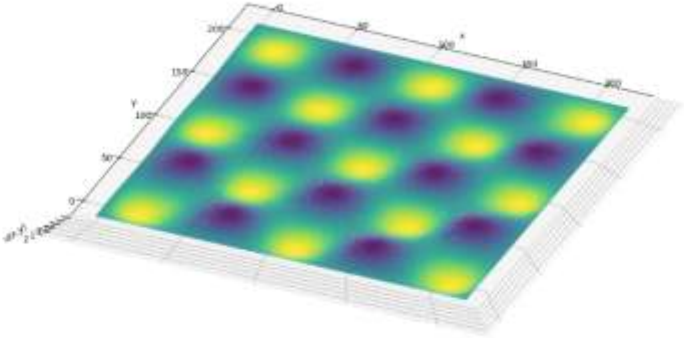


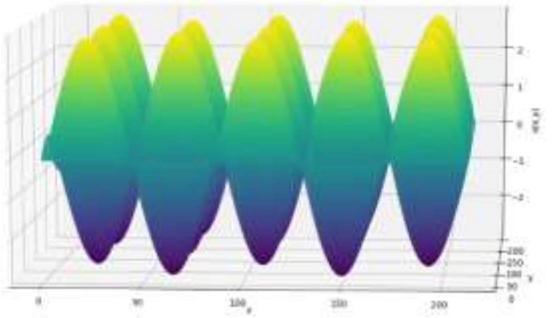


La différence entre le signal 2 et le signal 3 augmente en fonction du temps.
On voit bien le phénomène de dispersion ce qui se manifeste par une différence de vitesse de propagation lors d'un changement de fréquence dans les signaux Qui sont en mouvement par paquet. Avec présence de bruit blanc.

Propagation d' une onde en 2D sans atténuation:



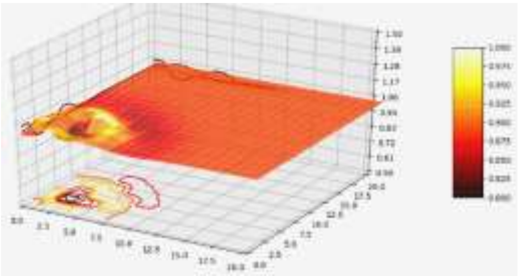
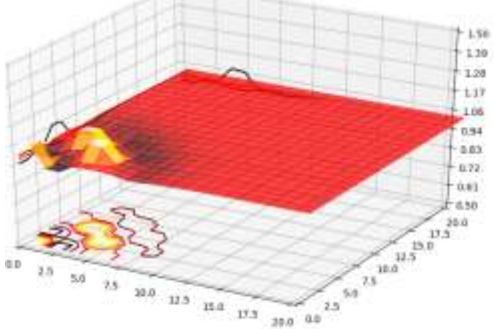
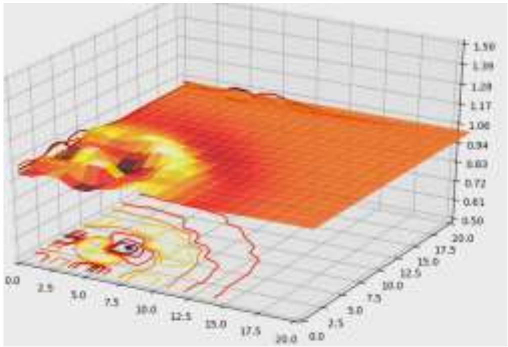
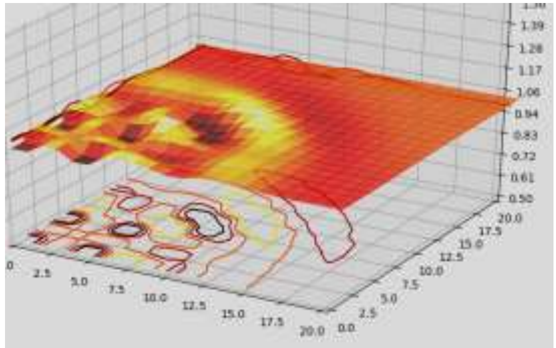
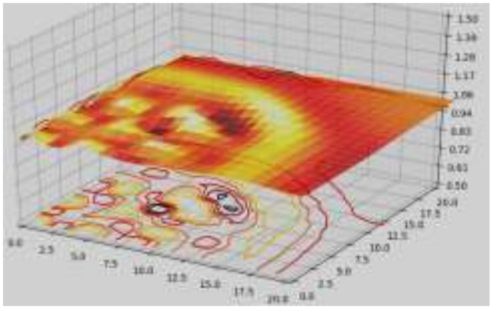
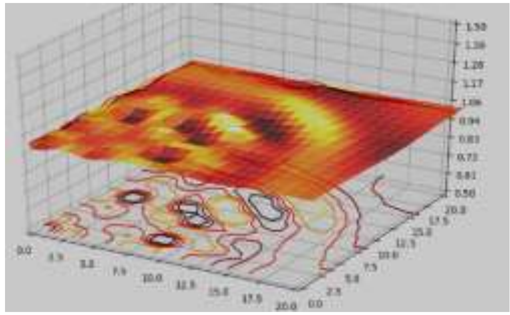
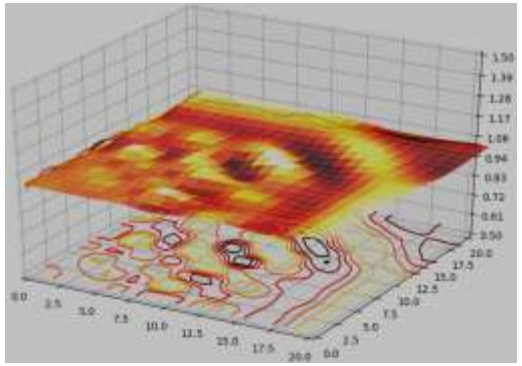




La propagation de l'onde est une propagation de l'énergie dans un milieu , dans notre cas sans atténuation
De l'amplitude en fonction du temps et la longueur du fibre. Ce signal est généré par l'équation:

$$\Delta\left(\vec{E}\right) = \frac{1}{c^2}\frac{d^2}{dt^2}\left(\vec{E}\right)$$

Propagation d’ une onde en 2D avec atténuation:

Résolution numérique de l’ équation de propagation du champ électrique avec atténuation la discrétisation est effectué a l’aide de la méthode de différence finie .

Remarques:

La simulation numérique reproduit la propagation et l’atténuation

Conclusion:

La modélisation, le phénomène de propagation de la lumière dans une fibre optique est un
Phénomène complexe
Nécessité d’un couplage entre la modélisation physique, l’étude expérimentale et la simulation
Numérique
Mon étude expérimentale a bien montré l’importance des propriétés intrinsèque de la fibre optique
Sur la d’atténuation et la dispersion de l’information pendant la
Les simulations numériques ont confirmés quelques résultats expérimentaux mais j’ai trouvé
des difficultés dans les effets non linéaires.

Modulation d’amplitude

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
@author: Mohamed salah Chefi
"""
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0, 100, 0.1)

message_freq = 50
message = np.sin(6.28*message_freq*t)

#message = np.sin(6.28*message_freq*t) + np.sin(6.28*message_freq*2*t)+np.sin(6.28*message_freq*3*t)+np.sin(6.28*message_freq*4*t)


carrier_freq = 1000

class Filters():

    def LowPassFilter(self, cutoff):
        f = np.sin(6.28*cutoff*t)/(3.14*t)
        return f


class AmplitudeModem:
    def modulate(self, message, fc):
        c = np.cos(6.28*fc*t)

        mod = message*c # ici on multiplie les deux signaux
        return mod

    def demodulate(self, received_array, fc, fm):
        c = np.cos(6.28*fc*t)
        demod = 2*c*received_array
        baseband = demod
        #self.modulate(4c, received_array, 2*fc)
        return np.convolve(Filters().LowPassFilter(message_freq), baseband)
```

```python
modulated = AmplitudeModem().modulate(message, carrier_freq)
demodulated = AmplitudeModem().demodulate(modulated, carrier_freq, message_freq)

plt.plot(modulated)
#plt.plot(demodulated)
plt.xlabel("Temps en seconde")
plt.ylabel("Signal modulé")
#plt.ylabel("Signal démodulé")
plt.grid()
plt.show()




"""
message_freq = 50
message = np.sin(6.28*message_freq*t)

message = np.sin(6.28*message_freq*t) + np.sin(6.28*message_freq*2*t)+np.sin(6.28*message_freq*3*t)+np.sin(6.28*message_freq*4*t)

plt.plot(message)
plt.xlabel("Temps en s")
plt.ylabel("signal informatif")
plt.grid()
plt.title("signal contenant l'information basse fréquence")
"""
```

Modulation de frequence

```python
import numpy as np

import matplotlib                      # pour tracer
import matplotlib.pyplot as plt
from pykat import finesse              # Importer le packet pykat.finesse .
from pykat.commands import *           # importer tout les packets dans pykat.commands.
from IPython.display import display, HTML  # nous permer de d'afficher HTML.

pykat.init_pykat_plotting(dpi=90)

## Code pour montrer un champ modulé en phase  ##

# Parametres
# -----------------------------------------------------
t = np.linspace(0,1,2048)   # temps de tableau
E0 = 1                      # Amplitude of the field.
fc = 20                     # frequence porteur
phi_c = 0                   # phase du champ
fm = 2                      # Phase de la frequence de modulation
m = 5                       #  index de la modulation
phi_m = 0                   # phase de la modulation

# calculer les tableaux des champs electriques et des signaux.
# -----------------------------------------------------
# champ de transport
E = E0*np.cos(fc*2*np.pi*t + phi_c)
# Signal
x = m*np.sin(fm*2*np.pi*t + phi_m)
# champ modulé en phase
E_m = E0*np.cos(fc*2*np.pi*t + phi_c + m*np.sin(fm*2*np.pi*t+phi_m))

# tracage
# -----------------------------------------------------
fig = plt.figure(figsize=(10,3))

# les axes du champ porteur et modulé
ax = plt.subplot(1,1,1)
p1 = ax.plot(t,E,'0.7', label='$\mathrm{Carrier}\ E_c(t)$')
p2 = ax.plot(t,E_m,'r',label='$\mathrm{Mod. field}\ E_m(t)$')
ax.set_xlabel('t')
ax.set_ylabel('E(t)')
ax.set_title('modulation de frequence')
ax.set_xlim(-0.01,1.01)
```

```python
# Deuxième axe des ordonnées du signal
ax2 = ax.twinx()
p3 = ax2.plot(t,x,'b', label='$\mathrm{Signal}\ x(t)$')
ax2.set_ylabel('x(t)')
# la legende
plots = p1+p2+p3
labs = [lab.get_label() for lab in plots]
ax.legend(plots, labs, loc=1, fontsize=10)
# montrer la figure
plt.show(fig)
```

dispersion

```python
from random import gauss
from random import seed
from pandas import Series
from pandas.plotting import autocorrelation_plot
from matplotlib import pyplot

#  générateur de nombres aléatoires
seed(1)
#  créer une série de bruit blanc
series = [gauss(0.0, 1.0) for i in range(1000)]
series = Series(series)
# statistiques sommaires
print(series.describe())
# graphique linéaire
series.plot()
pyplot.show()
import numpy as np
import matplotlib.pyplot as plt
A= 10.
w=400
c=1
N = 50
erreur = 1

x=np.linspace(0,N,1000)
t = np.arange(0,N,0.5)
#pure = A*np.exp(-(t-x/c)**2)*np.cos(w*(t-x/c))
# noise = np.random.normal(0, 1, pure.shape)
# signal = pure + noise
"""
plt.plot(pure)
plt.plot(noise)
plt.plot(signal)
"""
```

```python
for t in np.arange(0,N,0.5):
    plt.clf()
    plt.axis([0,N,-20,20])
    plt.xlabel('abscisse z et temps t')
    plt.ylabel("Intensité I(z,t)")
    # plt.plot(x,onde(x,t))
    plt.plot(x,A*np.exp(-(t-x/c)**2)*np.cos(w*(t-x/c)) +
             0.75*A*np.exp(-(t-2*x/c)**2)*np.cos(w*(t-2*x/c)) +
             0.5 * A*np.exp(-(t-3*x/c)**2)*np.cos(w*(t-3*x/c)) +
             np.random.normal(0, erreur, x.shape))
    plt.grid()
    plt.pause(0.01)
```

Propagation sans attenuation

```
import matplotlib.pylab as p; from numpy import *
from mpl_toolkits.mplot3d import Axes3D

tim = 15;     N = 3*71
#tim = 30;    N = 142
c = sqrt(180./390)                          # Speed = sqrt(ten[]/den[kg/m2;])
u = zeros((N,N,N),float);      v = zeros((N,N),float)
incrx = pi/N;                incry = pi/N
cprime = c;
covercp = c/cprime;     ratio = 0.5*covercp*covercp  # c/c' 0.5 for stable

def vibration(tim):
   y = 0.0
   for j in range(0,N):                              # Initial position
      x = 0.0
      for i in range(0,N):
         u[i][j][0] = 3*sin(5*x)*sin(5*y)                 # Initial shape
         x += incrx
      y += incry

   for j in range(1,N-1):                            # First time step
      for i in range(1,N-1):
         u[i][j][1] = u[i][j][0] + 0.5*ratio*(u[i+1][j][0]+u[i-1][j][0]
              + u[i][j+1][0]+u[i][j-1][0]-4.*u[i][j][0])

   for k in range(1,tim):                           # Later time steps
      for j in range(1,N-1):
         for i in range(1,N-1):
          u[i][j][2] = 2.*u[i][j][1] - u[i][j][0] + ratio*(u[i+1][j][1]
           + u[i-1][j][1] +u[i][j+1][1]+u[i][j-1][1] - 4.*u[i][j][1])
      u[:][:][0] = u[:][:][1]                                # Reset past
      u[:][:][1] = u[:][:][2]                             # Reset present
      for j in range(0,N):
         for i in range(0,N):
            v[i][j] = u[i][j][2]          # Convert to 2D for matplotlib
   return v
```

```
v = vibration(tim)
x1 = range(0, N)
y1 = range(0, N)
X, Y = p.meshgrid(x1,y1)

def functz(v):
    z = v[X,Y]; return z

Z = functz(v)
fig = p.figure()
ax = Axes3D(fig)
#ax.plot_wireframe(X, Y, Z, color = 'r')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,cmap='viridis', edgecolor='none')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('u(x,y)')
p.show()
```

Propagation avec attenuation

```
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))- (pow(u[i,j+1],2)/h[i,j+1]+ k/2*pow(h[i,
            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+1]/h[i,j+1]))

    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,j]/h[i+1,j]))
            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)) - (pow(v[i+1,j],2)/h[i+1,j] + k/2*pow(

    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)/hx[i-1,j-1] + k/2*pow(hx[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])) - (dt/dy)*((pow(vy[i-1,j],2)/hy[i-1,j] +

#dh = dt/dt*(ux[1:,:]-ux[:-1,:])+ dt/dy*(vy[:,1:]-vy[:,:-1])
    reflective()
    return h,u,v
```

```
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():
    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,linewidth=0,antialiased=False, alpha=0.7)

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

```
from matplotlib import animation


def data(k,h,surf):
    proses()
    ax.clear()
    plotset()
    surf = ax.plot_surface(x, y, h,rstride=1, cstride=1,cmap=cm.hot,linewidth=0,antialiased=False, alpha=0.7)
    return surf,

ani = animation.FuncAnimation(fig, data, fargs=(h,surf), interval=100, blit=False)
ani.save('laser.mp4', bitrate=512)
plt.show()
```