Propagation - Paquet d'ondes
Dispersion - Vitesse de phase - Vitesse de groupe

Objectifs de l'étude

Le modèle de l'OPPH n'est pas physique car il s'agit d'une onde de durée infinie dans le temps et d'extension infinie dans l'espace (qui serait donc associée à une énergie infinie).
Cependant, l'analyse de Fourier permet d'écrire tout signal périodique comme une somme infinie de signaux sinusoïdaux, ce qui justifie l'étude approfondie des OPPH.

Qu'advient-il d'un signal périodique, associé à une somme d'OPPH, lorsque celles-ci ne se propagent pas toutes à la même vitesse dans un milieu (comme c'est le cas pour la lumière dans le verre par exemple) ?

La généralisation à un signal non périodique, appelé paquet d'ondes, sera également envisagée.

Plan

- Onde somme de deux ondes de fréquences voisines
- Onde somme de N ondes de fréquences comprises dans un intervalle
- Généralisation - Paquet d'onde

Dans chaque cas, on observera la propagation dans un milieu non dispersif (le vide par exemple pour les ondes électromagnétiques) et la propagation dans deux milieux dispersifs différents : les ondes à la surface de l'eau de relation de dispersion $\omega(k) = \sqrt{gk}$ et l'onde de matière associée à un électron libre non relativiste de relation de dispersion $\omega(k) = \dfrac{\hbar^2}{2m}k^2$.

Objectifs ✔️ : Vitesse de phase, vitesse de groupe, dispersion.
⚠️ Respecter impérativement les notations de l'énoncé.

Notations

Les grandeurs physiques ω, k, c… sont données dans des unités arbitraires. Plan

- k, k_0 : pulsations spatiales (norme du vecteur d'onde) ;
- $\omega(k)$: relation de dispersion donnant la pulsation temporelle.

Bibliothèques

[ ]:
```
from _paquetondes import *
import numpy as np
from scipy.integrate import simpson
import matplotlib.pyplot as plt
import matplotlib.animation as animation
```

Fonctions

✏️ Ecrire une fonction w_vide(k, c=1) renvoyant l'expression de ω en fonction de k et c pour une OPPH vérifiant une équation de propagation de d'Alembert.
Le paramètre c a pour valeur 1 par défaut.

[ ]:

✏️ Ecrire une fonction w_eau(k, g=1) renvoyant l'expression de ω en fonction de k et c pour une onde à la surface d'un fluide : $\omega(k) = \sqrt{gk}$.
Le paramètre g a pour valeur 1 par défaut.

[ ]:

✏️ Ecrire une fonction w_electron(k, B=1) renvoyant l'expression de ω en fonction de k et c pour un électron libre non relativiste : $\omega(k) = \dfrac{\hbar^2}{2m}k^2$ de la forme $\omega(k) = Bk^2$.
Le paramètre B a pour valeur 1 par défaut.

[ ]:

✏️ Ecrire une fonction ck(w, k, X, t) renvoyant l'expression d'un OPPH d'amplitude 1, de pulsation spatiale k, de pulsation temporelle w (w désigne une fonction quelconque w(k) et non une valeur), se propageant vers les abscisses croissantes. Utiliser la fonction cosinus de numpy.

⚠️ Dans la suite X désigne un tableau numpy d'abscisses x_i ou bien une unique abscisse x.

[ ]:

1 Animation d'un tracé

✏️ np.linspace

np.linspace linspace(start, stop, num=50)
help(np.linspace) pour davantage de détails

[ ]:
```
# Exemple
np.linspace(0, 0.1, 11)
```

1.1 Tracé sans animation

Avant d'animer un tracé, il est très fortement recommandé de tracer la fonction à différents instants afin de vérifier les bormes du domaine spatial et du domaine temporel (simplifie beaucoup la mise au point d'une animation dont le débogage n'est pas toujours simple).
✏️ Représenter la fonction ck définie ci-dessus avec k = k0 = 1, sur l'intervalle xmin = -5, xmax = 20 avec 200 valeurs (utiliser np.linspace) à $t = 0$ et $t = 1$ (unités arbitraires) et pour la relation de dispersion du vide.

[ ]:
```
# Pulsation spatiale
k0 =

# Paramétrage des abscisses (variable spatiale)
xmin, xmax =
Nx =     # Nbre de points
X =      # Tableau des abscisses

# Tracé du signal
plt.figure()
```

1.2 Tracé animé en fonction du temps

Principe d'une animation : succession d'images fixes séparées par un délai réglable (principe d'un film).

👁️ 💡 Observer la fonction graphe0 ci-dessous et lire les commentaires afin de comprendre le principe de l'animation.

⚠️ Ne pas supprimer les virgules qui semblent "inutiles" !

[ ]:
```
def graphe0(f):
    """ Animation de la fonction f en fonction du temps """
    global X, xmin, xmax, ymin, ymax, tmin, tmax, Nt, T

    def init():                             # Initialisation du tracé avec les
éléments fixes
        axs.set_xlim(xmin, xmax)            # Bornes du tracé en abscisses
        axs.set_ylim(ymin, ymax)            # Bormes du tracé en ordonnées
        axs.legend(loc='lower right')       # Position de la légende dans la
fenêtre
        line0.set_data(X, f(0))             # Tracé fixe (non animé), ici f(t=0)
        return line0,                       # On renvoie le(s) tracé(s) fixe(s)

    def update(nframe):                     # Mise à jour du tracé pour les
éléments mobiles
        t = T[nframe]                       # nframe = n° de l'image dans la
successions des images
```

```
        line.set_data(X, f(t))            # Mise à jour des données du tracé =
tracé de f à l'instant suivant
        nframe = nframe + 1               # Incrémentation du n° de l'image
        return line,                      # On renvoie le(s) tracé(s) modifié(s)

    fig, axs = plt.subplots(1, 1, figsize=(4,3))         # Nombre de fenêtres
graphiques et dimensions
    line0, = axs.plot([], [], 'b', label='OPPH à t=0')   # Création "variable
tracé" vide (couleur, label)
    line, = axs.plot([], [], 'r', label='OPPH à t')      # Création "variable
tracé" vide (couleur, label)

    # Création de l'animation (paramètres intuitifs comptetenu de ce qui
précède)
    ani = animation.FuncAnimation(fig, update, init_func=init, frames=Nt,
interval=10, repeat=False)
    # Le paramètre "interval" semble ne pas fonctionner dans Capytale

    plt.show()
```

⚠️ La fonction ck(w, k, X, t) est une fonction de 4 variables mais , une fois définis w (relation de dispersion), k et X (tableau des abscisses), cette fonction ne dépend plus que de t.
La ligne 10 ci-dessous permet de définir la fonction d'une seule variable $s(t)$ à partir de ck(w, k, X, t), w, k et X étant fixés.

[ ]:
```
# Pulsation spatiale
k0 = 1

# Paramétrage spatial : abscisses
xmin, xmax = -5, 20
Nx = 200                          # Nbre de points
X = np.linspace(xmin, xmax, Nx) # Tableau des abscisses
# Paramétrage spatial : ordonnées
ymax, ymin = 1, -1

# Paramétrage temporel
tmin, tmax = 0, 10
Nt = 100
T = np.linspace(tmin, tmax, Nt+1)

# Conversion de ck(w_vide, k0, X, t) en fonction de t seulement
s = lambda t: 0.5*ck(w_vide, k0, X, t)

plt.figure()
graphe0(s)
```

1.3 Tracé amélioré

Dans la suite, on utilise une fonction plus complexe nommée graphe (disponible dans l'une des biblothèques chargées) et dont l'utilisation est illustrée ci-dessous.

Cette fonction permet de réprésenter :

- une liste de signaux ;
- le spectre associé aux différents signaux.

Un signal est défini grâce à un dictionnaire de la forme : {'s': lambda t: expression basée sur la fonction ck, 'c': couleur, 'l': label, 'p': {'x': lambda t: vitesse*t, 'y': lambda t: valeur ou fonction}}
Signification des clés :

- 's' : signal = fonction de t définie à l'aide de la fonction ck(w, k, X, t)
- 'c' : couleur du tracé
- 'l' : label ("légende") associé à la courbe
- 'p' : dictionnaire contenant les informations pour le tracé d'un point lié au signal au cours du temps dont les clés sont :
  - 'x' : abscisse du point lié à la courbe = fonction de t liée à la vitesse du signal
  - 'y' : ordonnée du point lié

Le spectre est défini par une liste de tuples de la forme [(k, amplitude), ...].

👁️ 💡 Observer le code ci-dessous et lire les commentaires afin de comprendre comment utiliser la fonction graphe.

[ ]:
```
# Pulsation spatiale et longueur d'onde
k0 = 1
l0 = 2/k0

# Paramétrage spatial : abscisses
xmin, xmax = -5, 20
Nx = int(40 * (xmax - xmin)/l0)  # On utilise l0 pour déterminer le nbre de
périodes tracées
X = np.linspace(xmin, xmax, Nx)

# Paramétrage des ordonnées du graphe (signaux)
ymax = 2.05
ymin = -ymax

# Paramétrage temporel de l'animation
tmin, tmax = 0, 10
Nt = 20                          # Nombre d'images
T = np.linspace(tmin, tmax, Nt+1)

# Relation de dispersion utilisée
```

```
w = w_vide
titre1 = "OPPH dans le vide - Milieu non dispersif"   # Titre
titre2 = r"Relation de dispersion $\omega(k) = c k$"  # Sous-titre

# Signal défini par un dictionnaire
amplitude = 0.5  # Amplitude de l'OPPH
vp = w(k0)/k0    # Vitesse de phase de l'OPPH
onde = {'s': lambda t: amplitude*ck(w, k0, X, t),
        'c': 'b',
        'l': r'$s(x,t)=\frac{1}{2} \cos(kx-\omega(k)t)$',
        'p': {'x': lambda t: vp*t, 'y': lambda t: amplitude}}
# Le point, défini par la clé 'p', lié au signal se déplace à la vitesse de
phase

# Animation
plt.figure()
param = [k0, xmin, xmax, ymin, ymax, X, tmin, tmax, Nt, T]
graphe([onde], spectre=[(k0, amplitude)], titre=[titre1, titre2], param=param)
```

🎓 Vitesse de phase

La vitesse de déplacement du point lié à l'onde est la vitesse de phase : $v_\varphi = \dfrac{\omega(k)}{k}$

2 Somme de deux OPPH de fréquences voisines

On observe ici la propagation d'un signal défini comme la somme de deux ondes de même amplitude mais de pulsations (et donc de fréquences) voisines :
$s(x,t) = A\cos(k_1 x - \omega(k_1)t) + A\cos(k_2 x - \omega(k_)t)$.
Avec $k_1 = k_0 - \delta k$ et $k_1 = k_0 + \delta k$ (δk réglable).

2.1 Propagation dans un milieu non dispersif

👁️ 💡 Exécuter le code ci-dessous (le nombre d'images et la durée de l'animation peuvent être adaptées à la rapidité d'exécution du code...).

[ ]:
```
# Pulsation spatiale moyenne et longueur d'onde moyenne
k0 = 1
l0 = 2/k0
# Pulsations spatiales des ondes à sommer
dk = k0/10
k1, k2 = k0 - dk, k0 + dk

# Paramétrage des abscisses (variable spatiale)
xmin, xmax = 0, 70
Nx = int(40 * (xmax - xmin)/l0)
X = np.linspace(xmin, xmax, Nx)
```

```
# Paramétrage des ordonnées du graphe (signaux)
ymax = 2.05
ymin = -ymax

# Relation de dispersion utilisée
# ___ Ondes dans le vide ___
w = w_vide
vmax = 1
titre1 = "Somme de deux OPPH - Milieu non dispersif"
titre2 = "Ondes dans le vide"

# Paramétrage temporel de l'animation
tmin, tmax = 0, 20
Nt = 2*tmax
T = np.linspace(tmin, tmax, Nt+1)

# Signaux définis via des dictionnaires
onde1 = {'s': lambda t: 0.5*ck(w, k1, X, t),
         'c': 'b',
         'l': r'OPPH $k_0-\delta k$',
         'p': {'x': lambda t: w(k1)/k1*t, 'y': lambda t: 0.5}}

onde2 = {'s': lambda t: 0.5*ck(w, k2, X, t),
         'c': 'g',
         'l': r'OPPH $k_0+\delta k$',
         'p': {'x': lambda t: w(k2)/k2*t, 'y': lambda t: 0.5}}

y_suivi = lambda t: 0.5*(ck(w,k1,w(k0)/k0*t,t) + ck(w,k2,w(k0)/k0*t,t))
somme = {'s': lambda t: onde1['s'](t) + onde2['s'](t),
         'c': 'r',
         'l': 'Somme',
         'p': {'x': lambda t: w(k0)/k0*t, 'y': y_suivi}}

envl1 = {'s': lambda t: np.cos(dk*X - (w(k2) - w(k1))/2*t),
         'c': 'gold',
         'l': 'Enveloppe',
         'p': {'x': lambda t: (w(k2) - w(k1))/(k2-k1)*t, 'y': lambda t: 1}}

envl2 = {'s': lambda t: -envl1['s'](t),
         'c': 'gold',
         'l': ''}

# Animation
ondes = [onde1, onde2, somme, envl1, envl2]
spectre = [(k1, 0.5), (k2, 0.5)]
plt.figure()
param = [k0, xmin, xmax, ymin, ymax, X, tmin, tmax, Nt, T]
```

```
graphe(ondes, spectre=spectre, titre=[titre1, titre2], param=param)
```

🎓 Vitesse de groupe

La vitesse de déplacement du point lié à l'enveloppe de l'onde résultante est la vitesse de groupe : $v_g = \dfrac{d\omega(k)}{dk}$

❓ Conclusion milieu non dispersif

En observant la simulation, que peut-on dire des vitesses de phase des trois ondes ?
De la vitesse de groupe ? Justifier.

[ ]:
```
# Réponses milieu non dispersif
```

2.2 Propagation dans un milieu dispersif - Ondes de surface

👁️ 💡 Copier/coller puis adapter le code précédent au cas des ondes de surface.

Remarques

- la modification essentielle concerne la relation de dispersion ;
- vmax permet d'ajuster automatiquement l'échelle de temps via tmax (compte tenu des expressions connues de v_φ et v_g, il est aisé de déterminer la vitesse la plus grande, de plus la vitesse de groupe est visible dans le code) ;
- ne pas oublier de modifier les titres.

[ ]:
```

```

❓ Conclusion ondes de surface

En observant la simulation, que peut-on dire des vitesses de phase des trois ondes ?
De la vitesse de groupe ? Justifier.

[ ]:
```
# Réponses ondes de surface
```

2.3 Propagation dans un milieu dispersif - Electron libre

👁️ 💡 Copier/coller puis adapter le code précédent au cas d'un électron libre.

[ ]:
```

```

❓ Conclusion onde électron libre

En observant la simulation, que peut-on dire des vitesses de phase des trois ondes ?
De la vitesse de groupe ? Justifier.

[ ]:
```
# Réponses onde électron libre
```

3 Somme de N OPPH

On observe ici la propagation d'un signal défini comme la somme de N ondes de même amplitude mais de pulsations (et donc de fréquences) comprises dans un intervalle de largeur δk :
$s(x,t) = \sum_{k0-\delta k}^{k0+\delta k} A\cos(kx - \omega(k)t)$.

3.1 Propagation dans un milieu non dispersif

✏️ Ecrire une fonction vg_vide(k, c=1) renvoyant l'expression de la vitesse de groupe v_g en fonction de k et c dans le vide.
Le paramètre c a pour valeur 1 par défaut.

[ ]:

👁️ 💡 Exécuter le code ci-dessous.

[ ]:
```
# Pulsation spatiale moyenne et longueur d'onde moyenne
k0 = 1
l0 = 2/k0
# Pulsations spatiales des ondes à sommer
rk = 5
dk = k0/rk
# Nombre de pulsations à sommer
Nk = 200
K = np.linspace(k0-dk, k0+dk,Nk)

# Paramétrage des abscisses (variable spatiale)
xmin, xmax = 0, 200
Nx = int(40 * (xmax - xmin)/l0)
X = np.linspace(xmin, xmax, Nx)
# Paramétrage des ordonnées du graphe (signaux)
ymax = 1.5 * Nk + 0.5
ymin = -ymax

# Relation de dispersion utilisée
# ___ Ondes dans le vide ___
w = w_vide
vg = vg_vide
vmax = 1
titre1 = "Somme de deux OPPH - Milieu non dispersif"
titre2 = "Ondes dans le vide"

# Paramétrage temporel de l'animation
tmin, tmax = 0, 100
Nt = 50
T = np.linspace(tmin, tmax, Nt+1)

# Signaux définis via des dictionnaires
```

```python
def sommeOPPH(w, K, X, t):
    """ Somme des OPPH pour toutes les valeurs ki dans K """
    return np.sum([ck(w, k, X, t) for k in K], axis=0)

y_suivi = lambda t: Nk
somme = {'s': lambda t: sommeOPPH(w, K, X, t),
         'c': 'r',
         'l': 'Somme',
         'p': {'x': lambda t: w(k0)/k0*t, 'y': lambda t: y_suivi(t)}}

envl1 = {'s': lambda t: Nk*np.sinc((k0*X - vg(k0)*t)/rk/np.pi),
        'c': 'gold',
        'l': 'Enveloppe sans dispersion',
        'p': {'x': lambda t: vg(k0)*t, 'y': lambda t: Nk}}

envl2 = {'s': lambda t: -envl1['s'](t),
         'c': 'gold',
         'l': ''}

ondes = [somme, envl1, envl2]
spectre = [(k, 1) for k in K]
plt.figure()
param = [k0, xmin, xmax, ymin, ymax, X, tmin, tmax, Nt, T]
graphe(ondes, spectre=spectre, titre=[titre1, titre2], param=param)
```

🎓 Notion de paquet d'ondes

On constate que la somme de N ondes conduit à un signal mieux localisé dans l'espace donc plus réaliste physiquement.

3.2 Propagation dans un milieu dispersif - Ondes de surface

✏️ Ecrire une fonction vg_eau(k, g=1) renvoyant l'expression de la vitesse de groupe v_g en fonction de k et g pour les ondes à la surface de l'eau.
Le paramètre g a pour valeur 1 par défaut.

[ ]:

👁️ 💡 Copier/coller puis adapter le code précédent au cas d'ondes de surface.

On admettra que la forme du paquet d'onde à l'instant initial est donnée par la fonction :

```python
y_suivi = lambda t: Nk*np.sinc((vg(k0)*t)/rk/np.pi)
```

[ ]:

3.3 Propagation dans un milieu dispersif - Electron libre

✏️ Ecrire une fonction vg_electron(k, B=1) renvoyant l'expression de la vitesse de groupe v_g en

fonction de k et B pour un électron libre.
Le paramètre B a pour valeur 1 par défaut.

[ ]:

👁️ 💡 Copier/coller puis adapter le code précédent au cas d'un électron libre.

On admettra que la forme du paquet d'onde à l'instant initial est encore donnée par la fonction :

```
y_suivi = lambda t: Nk*np.sinc((vg(k0)*t)/rk/np.pi)
```

[ ]:

❓ Conclusion

Quel phénomène suplémentaire observe-t-on dans les deux derniers cas (milieu dispersif) ?

[ ]:
```
# Réponse milieux dispersifs
```

🎓 Généralisation

La forme du paquet d'ondes obtenu n'est pas encore très réaliste, on va donc généraliser cette notion en passant d'une somme discrète à une somme continue (intégrale).
Mathématiquement, on passe d'une somme de Fourier à une tranformée de Fourier.

4 Paquet d'onde

On observe ici la propagation d'un signal défini comme la somme d'un continuum d'ondes de pulsations (et donc de fréquences) variant continûment mais avec une amplitude variable selon k : $s(x,t) = \int_{-\infty}^{\infty} A(k)\cos(kx - \omega(k)t)\,dk$.

Le spectre $A(k)$ d'un tel signal est continu.

Dans la suite, on envisage un paquet d'onde Gaussien tel que $A(k) = \dfrac{1}{2\pi\delta k} e^{-\dfrac{(k-k_0)^2}{2\delta k^2}}$.

✏️ Ecrire une fonction Ak(k, k0, Dk) renvoyant l'expression de $A(k)$.

[ ]:

✏️ Tracer $A(k)$ (i.e. le spectre de $s(x,t)$) avec k0=1, Dk = k0/10 et 100 valeurs comprises entre k0-4Dk et k0+4Dk (utiliser linspace).

[ ]:

👁️ 💡 Lire le code ci-dessous qui permet de créer $s(x,t) = \int_{-\infty}^{\infty} A(k)\cos(kx - \omega(k)t)\,dk$ et de définir les grandeurs utiles puis l'exécuter.

[ ]:
```
# Composante de Fourier de pulsation k
def sk(w, k, x, t, k0, Dk):
    return Ak(k, k0, Dk) * ck(w, k, x, t)

# Onde somme (intégration par la méthode de Simpson) pour toutes les valeurs ki
dans le tableau K
def s(w, K, X, t, k0, Dk):
    return np.array([simpson(sk(w, K, x, t, k0, Dk), K) for x in X])

# Paramétrage de l'onde
k0 = 1
Dk = k0/4
# Domaine d'intégration sur k
kmin, kmax = k0-3*Dk, k0+3*Dk
Nk = 120
K = np.linspace(kmin, kmax, Nk)
# REMARQUE : en réalité le domaine d'intégration n'est pas infini
# L'intervalle choisi correspond à des valeurs telles que A(k) >= Amax/100

# Paramétrage des abscisses du graphe
xmin, xmax = -10, 70
Nx = 10 * xmax
X = np.linspace(xmin, xmax, Nx)
# Paramétrage des ordonnées du graphe
ymax = 1.5 * Nk + 0.5
ymin = -ymax
```

Une fonction grapheS (disponible dans l'une des bibliothèques chargées) permet de visualiser l'évolution du paquet d'onde et de deux composantes de pulsations $k_0 - Dk$ et $k_0 + Dk$.

4.1 Propagation dans un milieu non dispersif

👁️ 💡 Exécuter le code ci-dessous.

[ ]:
```
w = w_vide
vg = 1

plt.figure()
param = [k0, Dk, K, w, vg, xmin, xmax, X]
grapheS(s, ck, tmin=0, tmax=30, Nt=40, titre=r'$\omega(k) \propto k$',
param=param)
```

4.2 Propagation dans un milieu dispersif - Ondes de surface

👁️ 💡 Exécuter le code ci-dessous.

[ ]:
```
w = w_eau
vg = vg_eau(k0)
```

```
plt.figure()
param = [k0, Dk, K, w, vg, xmin, xmax, X]
grapheS(s, ck, tmin=0, tmax=90, Nt=100, titre=r'$\omega(k) \propto \sqrt{k}$',
param=param)
```

4.3 Propagation dans un milieu dispersif - Electron libre

👁️ 💡 Exécuter le code ci-dessous.

[ ]:
```
w = w_electron
vg = vg_electron(k0)

plt.figure()
param = [k0, Dk, K, w, vg, xmin, xmax, X]
grapheS(s, ck, tmin=0, tmax=20, Nt=50, titre=r'$\omega(k) \propto k^2$',
param=param)
```

❓ Conclusion

Quels phénomènes observe-t-on dans les deux derniers cas (milieu dispersif) ? Comment la dispersion se manifeste-t-elle ?

📌 Remarque

Le signal $s(x,t)$ a été programmé de façon naïve (donc peu efficace) mais les outils adaptés pour effectuer ces calculs (transformée de Fourier) dépassent le cadre du programme.