

TP : Onde mécanique

ACTIVITÉ 1

Simulation de la propagation d'une onde mécanique à l'aide d'un programme Python

But de l'activité : Montrer qu'une onde mécanique progressive périodique possède une double périodicité et de voir, sur une animation programmée en python, l'influence de différents paramètres (période, célérité, amplitude...)

Pensez à consulter le Mémento PYTHON à la fin en annexe.

1. Chargement du programme Python

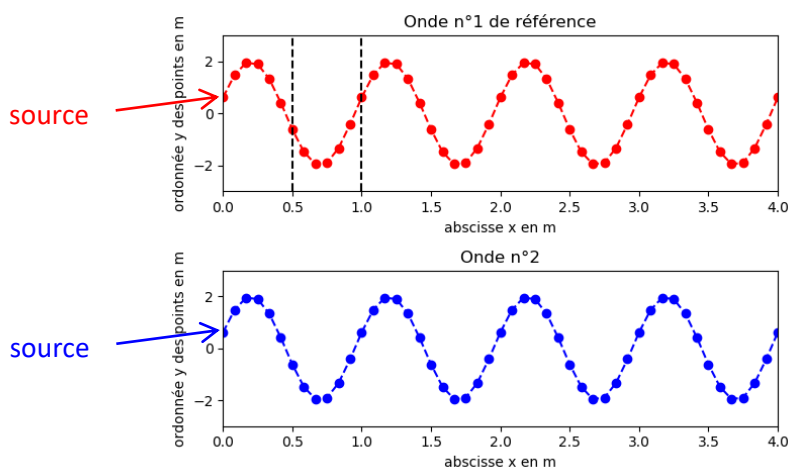
1. Ouvrir l'éditeur Python (Edupython) et ouvrir le programme « 1ere_sinusoide_Animation.py »

Ce programme simule la propagation de deux ondes : l'onde n°1 (de référence) et l'onde n°2 dont vous pourrez modifier certaines caractéristiques : période T_2 , célérité v_2 et amplitude A_2 .

```
# #####
# 2. initialisation des variables
# #####
# ONDE n° 1 de référence
# période en secondes:
T1=0.5
# célérité en m/s:
v1=2
# amplitude en mètres
A1=2

"""
----- PARTIE MODIFIABLE PAR LES ÉLÈVES -----
"""

# ONDE n° 2
# période en secondes:
T2=0.5
# célérité en m/s:
v2=2
# amplitude en mètres
A2=2
""" FIN PARTIE MODIFIABLE PAR LES ÉLÈVES """
```



2. Lancer l'animation et remarquer que les deux ondes se propagent à la même « vitesse » appelée : **célérité**.
3. Y-a-t-il un déplacement global des points suivant l'axe horizontal ?

2. Mesure de la période de l'onde n°1

La **période** (notée T) est plus petite **durée** au bout de laquelle un point du milieu se retrouve dans le même état.

4. A l'aide d'un chronomètre, mesurer la période T_1 de la source de l'onde n°1
5. Mesurer la période du point de l'onde n°1 situé au niveau de l'abscisse $x = 0,5$ m.
6. Mesurer la période du point de l'onde n°1 situé au niveau de l'abscisse $x = 1$ m.
7. Comparer les 3 périodes mesurées et conclure.

3. Périodicité spatiale de l'onde n°1

Deux points en phase sont deux points du milieu étant, à chaque instant, dans le même état.

8. Que remarque-t-on pour la source et le point situé à 1 mètre ?
9. Que remarque-t-on pour la source et le point situé à 0,5 mètre ?

La **longueur d'onde** (notée λ) est la plus petite **distance** séparant deux points du milieu vibrant en phase (c'est-à-dire étant dans le même état). On l'appelle aussi **périodicité spatiale**.

10. En mettant en pause l'animation (en cliquant sur la fenêtre), évaluer la longueur d'onde λ_1 de l'onde n°1.

4. Célérité de l'onde n°1

11. Par analyse dimensionnelle, établir la relation générale entre T , λ et la célérité de l'onde.

12. A partir de vos mesures de T_1 et λ_1 , calculer la valeur de célérité v_1 de l'onde 1 et comparer cette valeur à celle saisie dans le programme python.

5. Influence des caractéristiques d'une onde sur sa propagation

Nous allons modifier certaines caractéristiques de l'onde n°2 et voir leur influence sur la propagation de l'onde n°2 (en comparant avec celle de l'onde n°1)

13. Modifier la valeur de la célérité de l'onde n°2 : $v_2 = 4 \text{ m.s}^{-1}$. Observer l'animation et repérer les grandeurs liées à l'onde qui ont été modifiées et celles qui ne l'ont pas été.

14. Remettre $v_2 = 2 \text{ m/s}$ et modifier $T_2 = 0,25 \text{ s}$. Observations ?

15. Sans modifier T_2 , choisir la célérité v_2 qui permettra aux ondes 1 et 2 d'avoir la même longueur d'onde.

16. Modifier l'amplitude de l'onde n°2 : $A_2 = 4$ et observer l'animation. L'amplitude a-t-elle une influence sur la propagation de l'onde ?

ACTIVITÉ 2

Détermination des caractéristiques d'une onde mécanique périodique à partir de représentations spatiales et temporelles à l'aide d'un programme Python

But de l'activité : Dans un premier temps, nous allons relier les deux périodes d'une onde progressive périodique et tracer ses deux représentations (spatiale et temporelle). Le programme choisissant la période et la célérité au hasard, nous déterminerons ensuite graphiquement T et λ pour calculer la célérité (et comparer à la valeur choisie par le programme)

1. Compréhension du code Python :

Ouvrir l'éditeur Python (Edupython) et ouvrir le programme « **1ere_sinusoides_T_lambda.py** »

Ce programme comporte 3 grandes parties :

- **0. L'initialisation des variables :** A chaque grandeur liée à l'onde (période, célérité etc.), on associe une variable. Les valeurs de période et célérité sont prises au hasard par le programme.
- **1. Le tracé de l'évolution temporelle de l'onde :** on crée la fonction mathématique modélisant l'évolution de y en fonction de t et on trace la représentation graphique de cette fonction.
- **2. Le tracé de l'évolution spatiale de l'onde :** on crée la fonction mathématique modélisant l'évolution de y en fonction de x et on trace la représentation graphique de cette fonction.

Sur la paillasse, vous trouverez un mémento rappelant les instructions python les plus importantes.

2. Relation liant la période spatiale λ de l'onde et sa période temporelle T

Une onde possède une double périodicité :

- Temporelle (appelée période et notée T) : plus petite durée au bout de laquelle un point du milieu se retrouve dans le même état.
- Spatiale (appelée longueur d'onde et notée λ) : plus petite distance séparant deux points du milieu vibrant en phase (c'est-à-dire étant dans le même état)

1. Par analyse dimensionnelle, retrouver la relation entre T et λ .

2. Programmer cette relation dans le code Python au niveau de la ligne « $\lambda =$ » (Attention à la casse !)

3. Tracé de l'évolution temporelle de l'onde

L'évolution dans le temps de l'ordonnée d'un point peut-être modélisée par la fonction suivante :

$$y(t) = A.\cos\left(\frac{2\pi}{T}.t\right) \quad \text{où } A \text{ est l'amplitude du mouvement}$$

3. Au niveau de la partie 1 du programme, écrire la formule permettant de calculer les valeurs de y à partir de celles de t (t est un tableau « numpy » contenant 200 dates)

En python : π se tape : `np.pi` et la fonction cos(t) se tape : `np.cos(t)`

4. Taper ensuite le code permettant de tracer y en fonction de t en bleu continu.

5. Taper le code permettant de nommer les grandeurs sur les axes. Taper aussi le code pour dessiner une grille.

4. Tracé de l'évolution spatiale de l'onde

Compléter le tableau suivant :

Représentation temporelle $y(t)$	Représentation spatiale $y(x)$
Grandeur sur l'axe des abscisses : t	Grandeur sur l'axe des abscisses :
Période temporelle : T	Période spatiale:
Fonction : $y(t) = A.\cos\left(\frac{2\pi}{T}.t\right)$	Fonction :

6. Au niveau de la partie 2 du programme, écrire la formule permettant de calculer les valeurs de y à partir de celles de x (x est un tableau « numpy » contenant 200 positions)

7. Taper ensuite le code permettant de tracer y en fonction de x en rouge continu.

8. Taper le code permettant de nommer les grandeurs sur les axes. Taper aussi le code pour dessiner une grille.

9. Exécuter le programme et « maximiser » la fenêtre pour bien voir les deux représentations.

5. Utilisation du programme pour déterminer la célérité de l'onde

10. Mesurer graphiquement T et λ . En déduire la valeur de la célérité de l'onde.

11. Fermer la fenêtre matplotlib : les valeurs de T, λ et v s'affichent. Comparer à vos valeurs mesurées.

TYPE et CONVERSIONS DE VARIABLES

MEMENTO PYTHON 3 PHYSIQUE-CHIMIE

Types courants : Entier (*int*) Réel (*float*) Texte (*str*) Booléen (*bool*)

Déclaration de variables :

T = "Bonjour" Assigne à la variable texte (string) *T* le texte « Bonjour »

N = 2 Assigne à la variable entière (int) *N* la valeur 2

Conversion de variables :

T = str(**N**) transforme un entier (int) ou réel (float) *N* en texte (string)

N = int(**T**) ou **N** = float(**T**) transforme un texte *T* (string) en entier (int) ou réel (float)

ENTREES et SORTIES : demander une valeur à l'utilisateur et afficher à l'écran.

Entrée au clavier :

Un_Texte = input("question") Pose « question » à l'utilisateur et met la réponse dans la variable « texte » :

Un_Texte

Un_Entier = int(input("question")) Réponse mise dans la variable « entière » : *un_entier*

Sortie écran :

print ("Texte", variable) écrit sur l'écran *Texte* suivi du contenu de *Variable* séparés par un espace

print (variable1 + variable2) Uniquement si *variable1* et *variable2* sont de même type

type nombre : les ajoute **type texte** : les colle (les « concatène »)

print ("% .2e"%N) Ecrit le réel (ou entier) *N* en écriture scientifique avec 2 décimales (donc 3 chiffres significatifs)

CALCULS /ARRONDIS

round (x) arrondit un "réel" *x* vers l'entier le plus proche

round (x, n) arrondit un "réel" à la décimale *n*. *n* négatif permet un arrondi à la dizaine, centaine... près.

****** marque l'exposant et la priorité sur +, -, *, /

pow (x, y) renvoie *x* à la puissance *y*, équivaut à *x**y*

abs () renvoie la valeur absolue d'un nombre (sans le signe)

TESTS et CONDITIONS :

Test simple:

if Condition:
Instructions si « Condition » est vraie

Test avec SINON (else):

if Condition:
Instructions si « Condition » est vraie
else:
Instructions si « Condition » est fausse

Test avec SINON SI (elif):

if Condition:
Instructions si « Condition » est vraie
elif Condition2:
Instructions si « Condition2 » est vraie
else:
Instructions si « Condition1 et 2 » sont fausses

Test avec conditions multiples:

if Condition1 and/or Condition2:
Instructions

and: condition 1 ET 2 respectées

or: condition 1 OU 2 respectées

Opérateurs dans les conditions: ATTENTION le signe = est réservé à l'affectation de variables

== :égal

!= :différent

not : contraire de la condition

> (ou **<**):supérieur (ou inférieur)

>= (ou **<=**):sup (ou inf) ou égal

BOUCLES :

Boucle FOR générale: Dans le cas où on connaît le nombre de répétitions

for Variable in Ensemble de valeurs:
Instructions

Variable va prendre toutes les valeurs de « ensemble de valeurs »
bien penser à l'indentation !!!

Boucle FOR avec des nombres:

for Compteur in range (Nombre) :

Compteur varie de 0 à *Nombre-1*

for Compteur in range (début, fin) :

Compteur varie de *début* à *fin-1*

for Compteur in range (début, fin, pas) :

Compteur varie de *début* à *fin-1* par sauts de *pas*

Boucle WHILE (tant que) : Dans le cas où on ne connaît **pas** le nombre de répétitions
while *Condition* : bien penser aux « : » et à l'indentation
Instructions tant que « Condition » est vraie
Modifier la variable intervenant dans la condition sinon la boucle serait infinie !

MODULE RANDOM : hasard

Déclaration :

import random

Fonctions :

random.choice (*Ma_Liste*) choisit un élément de la liste *Ma_Liste*

random.randrange (*borne1*, *borne2*) renvoie un entier au hasard entre *borne1* (incluse) et *borne2* (exclue)

MODULE NUMPY : FONCTIONS COURANTES

import numpy as np

importe Numpy sous l'alias **np**

Fonctions trigonométriques			Fonctions trigonométriques inverses			π	\sqrt{x}
np.sin(x)	np.cos(x)	np.tan(x)	np.arcsin(x)	np.arccos(x)	np.arctan(x)	np.pi	np.sqrt(x)
ln(x) et e^x	log(x) et 10^x	Val absolue	Signe (renvoie 1 si positif et -1 si négatif)		arrondi à ndécimales		y^x
np.log(x)	np.log10(x)	np.abs(x)	np.sign(x)		np.around(x, n)		y**x
np.exp(x)	10**x						

AFFICHAGE avec MATPLOTLIB

import matplotlib.pyplot as plt importe pyplot sous l'alias **plt**

Affichage d'un point ou des valeurs d'un tableau Numpy

plt.plot (*X*, *Y*, "*style*") Place un point en *X,Y* avec un certain *style*. *X* et *Y* peuvent être des tableaux Numpy

plt.plot ([*X1,X2*], [*Y1,Y2*], "*style*") Trace un segment entre les points de coordonnées (*X1,Y1*) et (*X2,Y2*)

plt.plot (*X*, *Y*, "*style*", *options*) Point en *X,Y* avec *options* :

linewidth= *valeur*

épaisseur du trait

label= "*texte*"

texte à lier à la courbe (s'utilise avec

legend())

markersize= *valeur* taille des marques (points)

Styles disponibles : à placer entre guillemets dans l'ordre : **1. Couleur** **2. Type de point** **3. Type de tracé**

Couleurs						Type de points tracés					Tracé	
r	b	g	k	m	c	o	.	x	+	v	-	--
Rouge	Bleu	vert	noir	magenta	cyan	Gros point	Petit point	Croix	Croix +	Triangle	Points reliés	Points reliés en pointillé

Exemple de style : "**rx-**"

Affichage d'un texte

plt.text (*X*, *Y*, "*Texte à afficher*", **color**=*C*)

Affiche le texte aux coordonnées *X,Y* de couleur

C

plt.legend() Affiche le texte de l'option **label** de **plot** (utile si plusieurs courbes)

Affichage d'un vecteur

plt.quiver (*X*, *Y*, *Vx*, *Vy*, **color**=*C*, **scale**=20) vecteur en (*X*, *Y*) de coordonnées (*Vx*, *Vy*) de couleur *C*

Affichage de la fenêtre MATPLOTLIB (à placer tout à la fin)

plt.show()

GESTION DE LA FENÊTRE ET DES AXES MATPLOTLIB

plt.figure ("*NOM de la Fenetre*")

Donne une nom à la fenêtre **à placer au début**

plt.title ("*TITRE du graphique*")

Donne un titre au graphique

plt.xlabel ("*NOM de l'axe des X*")

Donne un nom à l'axe des abscisses (**ylabel** pour les

ordonnées)

plt.invert_xaxis()

Change le sens de l'axe X (**invert_yaxis()** pour l'axe Y)

plt.axis ([*Xmin*, *Xmax*, *Ymin*, *Ymax*])

Définit les valeurs min et max des abscisses et ordonnées

plt.grid()

Affichage de la grille