**FICHE de PRÉSENTATION d’activités**

|  |  |
| --- | --- |
| ***Niveau***  | ***1ère spécialité*** |
| ***Séquence*** | ***Ondes mécaniques*** |
| ***Titre de l’activité*** | ***La double périodicité d’une onde mécanique progressive périodique*** |
| ***Type d'activité*** | ***programmation en ½ groupe*** |
| ***Références au programme*** | Notions et contenus*Ondes sinusoïdales.**Période. Longueur d'onde.**Relation entre période, longueur**d’onde et célérité* | Capacités exigibles*Distinguer périodicité spatiale et périodicité temporelle.**Justifier et exploiter la relation entre période, longueur**d'onde et célérité.**Déterminer les caractéristiques d'une onde mécanique périodique à partir de représentations spatiales ou temporelles.****Capacité numérique :*** *Représenter un signal périodique et illustrer l’influence de ses caractéristiques (période, amplitude) sur sa représentation. Simuler à l’aide d’un langage de programmation, la propagation**d’une onde périodique.* |
| ***Compétences mobilisées***  | ❑ Analyser/raisonner ❑ Réaliser |
| ***Mise en œuvre*** | Pré-requis:Quelques notions de python (variable, exécution d’un programme sous Edupython) |
| Durée : 1h à 1h30 ***en demi-groupe***Activité traitant les notions de T, longueur d’onde, célérité, double périodicité.Les notions n’auront pas à être revues en classe entière. |
| Contraintes matérielles : salle informatique avec EduPython |
| *Liens photos* | *aucun* |
| *Auteur* | **olivier.chaumette@ac-lyon.fr** |
|  **pour le GRD groupe lycée de l’académie de LYON** |

**Fiche élève : activité**

**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…)

**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 T2, célérité v2 et amplitude A2.



source

source

**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 T1 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 λ1 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 T1 et λ1, calculer la valeur de célérité v1 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 : v2 = 4 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 v2 = 2 m/s et modifier T2 = 0,25 s. Observations ?

**15.** Sans modifier T2, choisir la célérité v2 qui permettra aux ondes 1 et 2 d’avoir la même longueur d’onde.

**16.** Modifier l’amplitude de l’onde n°2 : A2 = 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\_sinusoide\_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 créé 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 créé 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(.t) où A 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(.t)** | 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.

**Fiche professeur**

**ACTIVITÉ 1**

*Simulation de la propagation d’une onde mécanique*

*à l’aide d’un programme Python*

***Remarques pour le professeur :***

Le programme affiche deux ondes progressives (une de « référence » et une dont on modifie T et/ou v) ce qui permet de comparer les célérités quelle que soit la puissance de l’ordinateur.

Le programme est assez complexe car il affiche utilise le module « animation » de matplotlib. Faire comprendre les instructions de ce module aux élèves est totalement inutile.

Les élèves auront juste à réfléchir à la notion de période (mesure d’un aller-retour de la source sur l’axe vertical avec un chronomètre pour insister sur la notion de temps), à la notion de longueur d’onde (mesure de la distance entre 2 points en phase en mettant sur pause l’animation) et à l’influence de la célérité.

Ils modifient donc uniquement les valeurs de T2 et V2.

Les questions sont données à titre indicatif.

**1. Chargement du programme Python**

**3.** Y-a-t-il un déplacement global des points suivant l’axe horizontal ? Non, pas de déplacement global de matière. Chaque point du milieu ne fait que monter et descendre, comme la source.

**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 T1 de la source de l’onde n°1. Mesure de plusieurs T1 en observant la source osciller : T1 = 0,5s

**5.** Mesurer la période du point de l’onde n°1 situé au niveau de l’abscisse x = 0,5 m. idem : : T1 = 0,5s

**6.** Mesurer la période du point de l’onde n°1 situé au niveau de l’abscisse x = 1 m. idem : : T1 = 0,5s

**7.** Comparer les 3 périodes mesurées et conclure. Tous les points du milieu oscillent avec la même période qui est celle imposée par la source.

**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 ? Ils sont en phase

**9.** Que remarque-t-on pour la source et le point situé à 0,5 mètre ? Ils sont en opposition de phase

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 λ1 de l’onde n°1.λ1 = 1 m

**4. Célérité de l’onde n°1**

**11.** Par analyse dimensionnelle, établir la relation générale entre T, λ et la célérité de l’onde. v= λ/T

**12.** A partir de vos mesures de T1 et λ1, calculer la valeur de célérité v1 de l’onde 1 et comparer cette valeur à celle saisie dans le programme python. v1 = 1/0,5 = 2 m/s comme dans le programme : v1=2

**5. Influence des caractérsitiques 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 (en comparant avec celle de l’onde n°1)

**13.** Modifier la valeur de la célérité de l’onde n°2 : v2 = 4 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.T et amplitude inchangées. V augmente et λ aussi.

**14.** Remettre V2 = 2 m/s et modifier : T2 = 0,25 s. Observations. T diminue (la source oscille « plus vite » verticalement), mais v reste inchangé. λ diminue

**15.** Sans modifier T2, choisir la célérité v2 qui permettra aux ondes 1 et 2 d’avoir la même longueur d’onde.

Il faut prendre v2 = 4 m/s. même lambda mais l’onde 2 se propage plus rapidement.

**16.** Modifier l’amplitude de l’onde n°2 : A2 = 4 et observer l’animation. L’amplitude a-t-elle une influence sur la propagation de l’onde ? aucune influence.

**ACTIVITÉ 2**

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

***Remarques pour le professeur :***

Il est conseillé de donner aux élèves un mémento rappelant les instructions importantes pour tracer une courbe (plot, xlabel, grid etc…). Un exemple de mémento est fourni ci-dessous.

**ATTENTION :** « lambda » est un mot clé de Python (donc ne peut pas être utilisé pour un nom de variable). L’auteur a donc choisi « Lambda » comme nom de variable pour λ (avec un L majuscule)

Cette activité va permettre de montrer les analogies entre T et λ au niveau des expressions de y(t) et y(x).

L’auteur a fait le choix de ne pas donner y(x,t) aux élèves mais de « séparer » y(x,t) en deux fonctions.

L’activité permettra aussi de créer des exercices « automatisés » : mesure de T et λ puis calcul de v. Quand les élèves fermeront la fenêtre matplotlib, les valeurs s’afficheront ce qui lui permettra de se corriger.

Le programme choisit un T et V (donc λ) différents à chaque exécution du programme.

**4. Tracé de l’évolution spatiale de l’onde**

|  |  |
| --- | --- |
| **Représentation temporelle****y(t)** | **Représentation spatiale****y(x)** |
| Grandeur sur l’axe des abscisses : **t** | Grandeur sur l’axe des abscisses : **x**  |
| Période temporelle : **T** | Période spatiale: **λ**  |
| Fonction : **y(t) = A.cos(.t)** | Fonction : **y(t) = A.cos(.x)** |

**import** numpy **as** np

**import** matplotlib**.**pyplot **as** plt

**import** random

# ##############################

# initialisation des variables #

# ##############################

# période en seconde

# on tire T au hasard entre 0.02 et 1s (par pas de 0.01 s)

T**=**0.01**+(**random**.**randrange**(**1**,**100**)/**100**)**

# amplitude en m

A**=**2

# célerité en m/s

# on tire v au hasard entre 0.1 et 5 m/s (par pas de 0.1 m/s)

v**=** random**.**randrange**(**1**,**50**)/**10

""" TRAVAIL DES ÉLÈVES: """

# Exprimer la longueur d'onde à partir des données précédentes.

# attention: lambda (tout en minuscule) est un mot clé de python et ne peut pas être utilisé comme nom de variable.

# utiliser comme nom de variable: Lambda

Lambda**=**v**\***T

""" FIN TRAVAIL DES ÉLÈVES: """

# ################################

# Tracé

# ################################

# 1. Répresentation temporelle

# ################################

# on sépare la figure subplot en colonnes (subplot 12)

# et on se place sur celle de gauche : le dernier "1" de subplot(121)

plt**.**subplot**(**121**)**

# on crée un tableau avec 200 dates réparties entre 0 et 4 périodes

t**=**np**.**linspace**(**0**,**4**\***T**,**200**)**

""" TRAVAIL DES ÉLÈVES """

# on crée l'expression de l'ordonnée d'un point du milieu en fonction de la date t

y**=** A**\***np**.**cos**(**2**\***np**.**pi**\*(**t**/**T**))**

# tracer y en fonction de t en bleu continu

plt**.**plot**(**t**,**y**,**"b-"**)**

# nommer les axes et le graphique. mettre une grille

plt**.**xlabel**(**"date t"**)**

plt**.**ylabel**(**"ordonnée d'un point (abscisse fixée)"**)**

plt**.**title**(**"Rerpésentation temporelle d'un signal périodique"**)**

plt**.**grid**()**

""" FIN TRAVAIL DES ÉLÈVES: """

# ################################

# 2. Répresentation spatiale

# ################################

# on sépare la figure subplot en colonnes (subplot 12)

# et on se place sur celle de droite : le dernier "2" de subplot(122)

plt**.**subplot**(**122**)**

# on crée un tableau avec 200 positions réparties entre 0 et 4 LONGUEURS D'ONDES

x**=**np**.**linspace**(**0**,**3**\***Lambda**,**200**)**

""" TRAVAIL DES ÉLÈVES """

# Créer l'expression de l'ordonnée d'un point du milieu en fonction de son abscisse x

# pour cela, ADAPTER l'expression donnée pour la représentation temporelle: y= A\*np.cos(2\*np.pi\*(t/T))

y**=** A**\***np**.**cos**(**2**\***np**.**pi**\*(**x**/**Lambda**))**

# tracer y en fonction de x en rouge continu

plt**.**plot**(**x**,**y**,**"r-"**)**

# nommer les axes et le graphique. mettre une grille

plt**.**xlabel**(**"abscisse x"**)**

plt**.**ylabel**(**"ordonnée d'un point (date fixée)"**)**

plt**.**title**(**"Représentation spatiale d'un signal périodique"**)**

plt**.**grid**()**

""" FIN TRAVAIL DES ÉLÈVES: """

# on affiche la fenêtre

plt**.**show**()**

# et quand la fenêtre Matplotlib se ferme, on affiche les réponses

**print(**"La période vaut T="**,**T**,**"s"**)**

**print(**"La longueur d'onde vaut"**,**chr**(**955**),**"="**,**Lambda**,**"m"**)** # chr(955) affiche la lettre grecque lambda

**print(**"La célérité vaut v="**,**v**,**"m/s"**)**

**TYPE et CONVERSIONS DE VARIABLES**

**MEMENTO**

**PYTHON 3**

**PHYSIQUE-CHIMIE**

***O. Chaumette***

**Lycée JP SARTRE – 69500 BRON**

*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 a 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: Test avec SINON SI (elif):***

**if***Condition***: if***Condition***:**

*Instructions si « Condition » est vraie Instructions si « Condition » est vraie*

***Test avec SINON (else):* elif** *Condition2***:**

**if***Condition***:** *Instructions si « Condition2 » est vraie*

*Instructions si « Condition » est vraie* **else:**

**else:** *Instructions si « Condition1 et 2 » sont fausses*

 *Instructions si « Condition » est fausse*

***Test avec conditions multiples:***

**if** *Condition1* **and/or** *Condition2***: and**: condition 1 **ET** 2 respectées

*Instructions* **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***:** *Variable* va prendre toutes les valeurs de « ensemble de valeurs »

 *Instructions* 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**



**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**



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