**FICHE de PRÉSENTATION d’activités**

|  |  |  |
| --- | --- | --- |
| ***Niveau*** | ***1ère spécialité*** | |
| ***Séquence*** | ***Mouvement d’un système*** | |
| ***Titre de l’activité*** | ***Étude du vecteur variation de vitesse à l’aide d’un programme Python***  ***VERSION 2***  ***Discussion de la meilleure méthode numérique pour approcher le vecteur vitesse*** | |
| ***Type d'activité*** | ***Expérimentale et programmation en ½ groupe*** | |
| ***Références au programme*** | Notions et contenus  *Vecteur variation de vitesse.*  *Lien entre la variation du vecteur*  *vitesse d’un système modélisé*  *par un point matériel entre deux*  *instants voisins et la somme des*  *forces appliquées sur celui-ci.* | Capacités exigibles  *Réaliser et/ou exploiter une vidéo ou une chronophotographie d’un système modélisé par un point matériel en mouvement pour construire les vecteurs variation de vitesse. Tester la relation approchée entre la variation du vecteur vitesse entre deux instants voisins et la somme des forces appliquées au système.*  ***Capacité numérique :*** *Utiliser un langage de programmation pour étudier la relation approchée entre la variation du vecteur vitesse d’un système modélisé par un point matériel entre deux instants voisins et la somme des forces appliquées sur celui-ci* |
| ***Compétences mobilisées*** | ❑ Analyser/raisonner ❑ Réaliser | |
| ***Mise en œuvre*** | Pré-requis:  Quelques notions de python (variable, exécution d’un programme sous Edupython)  Notion de tableau auquel on ajoute des données. | |
| Durée : 2h | |
| Contraintes matérielles :  - salle informatique avec EduPython  - vidéo lancer parabolique et ou chute libre sans Vitesse initiale  - Logiciel de pointage : Latipro ou avimeca ou avistep | |
| *Liens photos* | *aucun* | |
| *Auteur* | **olivier.chaumette@ac-lyon.fr** | |
| **pour le GRD groupe lycée de l’académie de LYON** | |

**Fiche élève : activité**

Nous allons étudier, à l’aide d’un programme python, le vecteur « variation de vitesse » lors d’un lancer parabolique sans frottement.

**1. Comment approcher au mieux le vecteur vitesse ?**

2.1. Modèle utilisé en seconde

Le vecteur vitesse au point 8 (par exemple) ne peut pas être dessiné de manière exacte.

Mais **il est possible de l’approcher en traçant le vecteur vitesse moyenne entre les points 8 et 9** :

où M8 et M9 sont les noms des points n° 8 et 9

et t8 et t9 sont les dates de passage du système aux points n°8 et n°9

L’abscisse Vx de ce vecteur vitesse approché au point n°8 s’exprime :



où x8 et x9 sont les abscisses du système au points M8 et M9

et t8 et t9 les dates de passage du système aux points M8 et M9

Il en est de même pour Vy

***En Python :***

l’abscisse du système au point xn se tape : **x[n]** La date tn au point n se tape : **t[n]**

Pour ajouter un élément « A » à un tableau « T », l’instruction est la suivante : **T = np.append(T,A)**

**1.** Écrire ci-dessous, en langage python, l’expression de Vx et de Vy au point 8.

**2.** Charger le code python « **1ere\_Spe\_Mvt\_circulaire.py** »

Ce programme contient les données x, y et t issues d’un **mouvement circulaire uniforme** et, pour l’instant, trace la trajectoire.

**3. TRAVAIL 1 :** Dans le code python chargé, lire les consignes du programme situées sous TRAVAIL 1 :

- Taper le code permettant de calculer la coordonnée du vecteur Vx à la position 8

- Faire de même pour Vy

- Taper le code permettant de tracer le vecteur vitesse au point 8 en rouge  : **draw\_Vector2(8,Vx,Vy,"r",10)**

2.2. Autre modèle : le modèle centré

Une autre approche du tracé du vecteur vitesse, dite « **centrée** », consiste à tenir compte de la position avant le point et celle après le point. Ainsi :



 ou encore :

Il en est de même pour Vy.

**1.** Écrire ci-dessous, en langage python, l’expression de Vx et de Vy au point 8 en utilisant la méthode « centrée » :

**2. TRAVAIL 2 :** Dans le code python chargé, lire les consignes du programme situées sous TRAVAIL 2 :

- Taper le code permettant de calculer la coordonnée du vecteur Vx à la position 8 en utilisant la méthode « centrée »

- Faire de même pour Vy

- Taper le code permettant de tracer le vecteur vitesse au point 8 en vert  : **draw\_Vector2(8,Vx,Vy,"g",10)**

2.3. Comparaison des modèles

On a vu en seconde que Vx ≃ avec Δt durée séparant le passage du système en M et M’, deux points voisins.

Plus Δt est petit, plus  se rapproche de Vx donc, à une dimension, on peut écrire : 

Vx va donc correspondre à la dérivée du déplacement et donc au coefficient directeur de la tangente à la trajectoire.

En généralisant, on admettra que  **sera toujours tangente à la trajectoire.**

En regardant le résultat des tracés de l’approche du vecteur  avec les 2 méthodes, dire laquelle nous conserverons dorénavant pour approcher nos vecteurs vitesses.

**2. Positions successives d’un système (déjà traité en classe de seconde)**

***2.1. Pointage des différentes positions du système***

***(Texte volontairement non finalisé)*** faire pointer les positions d’un objet sur une vidéo et exporter au format TXT par Latispro, Avimeca ou Regressi… **Les coordonnées des positions M0, M1 etc. sont créées.**

***2.2. Préparation et ouverture du fichier Python***

**1.** Copier les 2 fichiers «**import\_donnees\_meca2.py**» et **«1ere\_Spe\_Variation\_vect\_vitesse1.py»** dans votre dossier personnel.

**2.** Lancer l’éditeur Python (Edupython.exe) et ouvrir le programme **« 1ere\_Spe\_Variation\_vect\_vitesse1.py »** que vous venez de copier.

***2.3. Tracé des positions successives du système : sa trajectoire***

**1. TRAVAIL 1 :** **Tracé de la trajectoire.**

**Cas général en Python (à adapter à ce que l’on veut faire) :**

Pour tracer la courbe représentant une grandeur A en fonction d’une grandeur B, il faut taper:

**plt.plot(B,A,“kx”)** *“kx” signifie que les points affichés seront noirs (“k”) et représentés par des croix (“x”)*

Dans le programme, en dessous de “TRAVAIL 1”, taper (en adaptant le cas général) le code permettant de tracer **l’évolution de l’ordonnée y du système en fonction de son abscisse x** avec des points de couleur rouge représentés par une croix sans être reliés (voir annexe à la fin du TP pour le nom des couleurs).

**2. TRAVAIL 2 :** **Préparation du graphique.** Suivre les consignes en rose dans le programme au niveau de «TRAVAIL 2»:

Donner un nom pertinent au graphique, écrire les noms des grandeurs représentées sur les axes **et leur unité**.

**3.** Exécuter le programme (petit triangle vert en haut de l’écran). Une fenêtre vous permet d’aller chercher le fichier que vous avez exporté depuis LATISPRO. La courbe doit s’afficher dans une nouvelle fenêtre. Fermer cette fenêtre.

**3. Coordonnées des vecteurs vitesse**

**1.** Écrire ci-dessous, en langage python, l’expression de Vx et de Vy au point Mn en utilisant la méthode « centrée ».

**2. TRAVAIL 3 :** Dans le code python, supprimer : print(“”) (qui permettait au programme incomplet de fonctionner quand même)

Lire les consignes du programme situées sous TRAVAIL 3 :

- Dans la boucle, taper le code permettant d’ajouter au tableau Vx la coordonnée du vecteur Vx à la position Mn

- Faire de même pour Vy

**3.** Vérifier la cohérence de votre code en traçant les vecteurs vitesse en tapant, toujours dans la boucle :

**draw\_Vector2(n,Vx[n-1],Vy[n-1],"b",20)**

**4.** La première position correspond à n=0 et la dernière à n = Nbre\_mesures.

Pourquoi la boucle proposée par le professeur commence à 1 et se termine à Nbre\_mesures-1 ?

**4. Coordonnées et tracé des vecteurs « variation de vitesse »**

***Vecteur « variation de vitesse » ***

En conservant la méthode « centrée », le vecteur variation de vitesse à la position Mn s’approche par :

Il se construit graphiquement de la manière suivante :



Ses coordonnées s’expriment donc, à la position Mn : **ΔVxn = Vxn+1-Vxn-1** et **ΔVyn = Vyn+1-Vyn-1**

**TRAVAIL 4 :** Lire les consignes du programme situées sous TRAVAIL 4 :

- créer une boucle (compteur de boucle nommé "n") allant de 1 à Nbre\_vitesses-1

- mettre dans une variable DeltaVx l'abscisse du vecteur variation de vitesse au point n

- mettre dans une variable DeltaVy l'ordonnée du vecteur variation de vitesse au point n

- tracer le vecteur variation vitesse au point "n+1" en vert avec une échelle de 5 (n+1 car le numéro de point et l’index du vecteur dans son tableau sont décalés de 1)

En Python, pour tracer un vecteur en un point, il faut utiliser la fonction suivante :

**draw\_Vector2(*numéro du point* , *Abscisse du vecteur* , *Ordonnée du vecteur* , “k”, *échelle*)**

*où “k” représente la couleur du vecteur (k : noir, voir les couleurs dans l’annexe en fin de TP)*

*« échelle » est un nombre entier permettant de tracer le vecteur à une bonne échelle par rapport au reste du graphique*

**5. Interprétation**

***(Texte volontairement non finalisé et laissé à l’initiative du professeur)***

**1.** Que remarque-t-on pour le vecteur variation de vitesse ?

**2.** Pourquoi n’y a-t-il pas de vecteur ******au niveau des 2 premiers points et des 2 derniers ?

**Les forces extérieures appliquées sur le système sont « responsables » de cette variation de vitesse.**

**On admettra cette année que **est colinéaire (et de même sens) à la somme des forces extérieures appliquées au système.**

**3.** Montrer que, dans le cas du mouvement étudié, on peut négliger les forces de frottement.

**5. Discussion autour de la validité de la conclusion**

***(Texte volontairement non finalisé et laissé à l’initiative du professeur)***

En réalité, on voit bien que nos vecteurs ne sont pas tous parallèles et colinéaires à la résultante des forces.

**1.** Proposer une explication.

**2.** Vérifier votre réponse en utilisant, à la place de votre pointage, les équations horaires modélisant la trajectoire (la détermination des équations horaires sera vue en terminale spécialité)

Pour cela : dans la partie « choix du professeur » du code python, taper : **Logiciel\_Utilise = 'equation'**

Exécuter le programme et conclure

**ANNEXE -** *Quelques options de mise en forme du tracé avec le module MATPLOTLIB utilisé par Python*

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

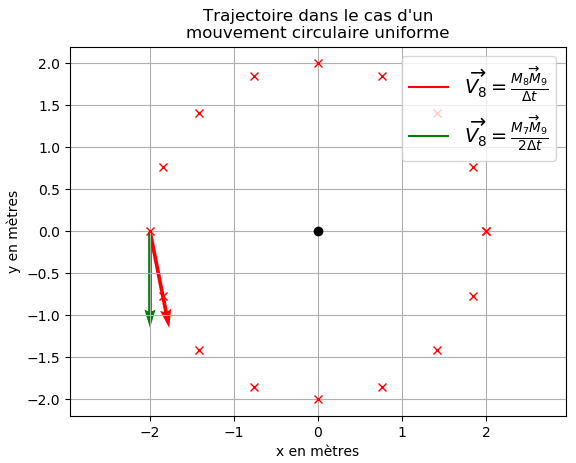
**Fiche professeur**

Une fiche expliquant comment exporter des données depuis AVIMECA ou AVISTEP est fournie à a fin de ce document. Cela permettra au professeur d’adapter la 1ère partie de l ‘énoncé au logiciel qu’il utilise.

L’auteur a défini préalablement, auprès des élèves, la vitesse Vx comme étant  (pour être cohérent par rapport à la définition mathématique). Cependant il a fait le choix de montrer aux élèves que la dérivée numérique centrée : Vx =  donne de meilleurs résultats pour approcher le vecteur vitesse car c’est elle qui se rapproche le plus de la tangente, dans le cas de points discrets (donc pour de l’expérimental).

***ATTENTION :*** le fichier **import\_donnees\_meca2.py** doit être présent dans le même dossier que le fichier python que complètent les élèves (car c’est lui qui permet de transformer le fichier exporté par le logiciel en tableaux numpy dans Python)

**1. Comment approcher au mieux le vecteur vitesse ?**

2.1. Modèle utilisé en seconde

Vx = (x[9]-x[8])/(t[9]-t[8])

Vy = (y[9]-y[8])/(t[9]-t[8])

2.2. Autre modèle : le modèle centré

Vx = (x[9]-x[7])/(t[9]-t[7])

Vy = (y[9]-y[7])/(t[9]-t[7])

Il n’est pas grave de réutiliser les variables Vx et Vy car le 1er vecteur aura déjà été tracé.

**IMPORTANT :**

Le but de ce paragraphe est de montrer que, de toutes façons, comme nous ne pourrons jamais tracer le « vrai » vecteur vitesse (sous-entendu instantanée), il faut l’approcher par une méthode numérique. On montre avec ce mouvement circulaire uniforme, que la méthode centrée donne de meilleurs résultats car V se rapproche « mieux » de la tangente.

Cela va permettre de l’adopter pour la suite des études expérimentales en mécanique (donc en terminale)

**3. Coordonnées des vecteurs vitesse**

**1.** Écrire ci-dessous, en langage python, l’expression de Vx et de Vy au point Mn

Vx = (x[n+1]-x[n-1])/(t[n+1]-t[n-1])

Vy = (y[n+1]-y[n-1])/(t[n+1]-t[n-1])

**4.** La première position correspond à n=0 et la dernière à n = Nbre\_mesures.

Pourquoi la boucle proposée par le professeur commence à 1 et se termine à Nbre\_mesures-1 ?

Car calcul de Vx utilise le point « d’avant » et celui « d’après » donc pas possible en n=0 et au dernier point.

**5. Interprétation**

***(Texte volontairement non finalisé et laissé à l’initiative du professeur)***

**1.** Que remarque-t-on pour le vecteur variation de vitesse ? aux erreurs de pointage près, même sens, même direction.

**2.** Pourquoi n’y a-t-il pas de vecteur ******au niveau des 2 premiers points et des 2 derniers ? car pour calculer les coordonnées de *, il faut utiliser les 2 points « avant » et les 2 points « après »*

**3.** Montrer que, dans le cas du mouvement étudié, on peut négliger les forces de frottement.

*et somme forces colinéaires et de même sens donc somme des forces verticales vers le bas.*

*Bilan de forces : poids (vertical vers le bas) et, éventuellement, forces de frottement.*

*Mais comme somme des forces verticale vers le bas, elle correspond uniquement au poids donc pas de forces de frottement. Nous venons* ***d’utiliser*** *la relation (et non la montrer)*

**5. Discussion autour de la validité de la conclusion**

En réalité, on voit bien que nos vecteurs ne sont pas tous parallèles et colinéaires à la résultante des forces.

**1.** Proposer une explication.

Pointage pas top donc calcul des variations pas exact donc logique à priori que les vecteurs ne soient pas parallèles exactement.

**2.** Vérifier votre réponse en utilisant, à la place de votre pointage, les équations horaires modélisant la trajectoire (la détermination des équations horaires sera vue en terminale spécialité)

Pour cela : dans la partie « choix du professeur » du code python, taper : **Logiciel\_Utilise = 'equation'**

Exécuter le programme et conclure

**Cette fois, OK donc cela venait bien du pointage.**

**REMARQUE POUR LE PROFESSEUR :**

L’auteur de la fiche propose aussi un code appelé « 1ere\_Spe\_Variation\_vect\_vitesse2.py » qui reprend le code précédent et qui trace également Vx et Vy en fonction de t. Cela peut permettre de montrer aux élèves qu’il y a des petites variations de Vx dûes au pointage et donc mieux expliquer le fait que les DetaV ne sont pas parfaitement parallèles.

Avec les équations horaires, on leur montre que Vx est bien constant.

1er code corrigé :

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

# Programme Python permettant de tracer la trajectoire d'un mouvement

# circulaire uniforme dans le but de montrerque la méthode centrée pour approcher

# la vitesse donne de meilleurs résultats que la méthode de la dérivée numérique à droite.

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

# O. CHAUMETTE - Lycée JP SARTRE - 69500 BRON - olivier.chaumette@ac-lyon.fr

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

# IMPORTATION DES MODULES #

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

# importation de NUMPY (pour gestion tableaux et calculs) sous l'alias "np"

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

# importation de PYPLOT (du module MATPLOTLIB, pour le tracé de courbes) sous l'alias "plt"

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

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

# Fonction qui simplifie le tracé du vecteur vitesse

# afin que l'élève n'ait pas à définir l'origine

# du vecteur en termes de coordonnées

# (pour ne pas mélanger des positions et des coordonées de vecteurs)

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

**def** draw\_Vector2**(**numero\_du\_point**,** abscisse\_vecteur**,** ordonnee\_vecteur**,** couleur**,**echelle**):**

**global** x**,**y

plt**.**quiver**(**x**[**numero\_du\_point**],** y**[**numero\_du\_point**],**abscisse\_vecteur**,**ordonnee\_vecteur**,**color**=**couleur**,**scale**=**echelle**)**

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

# Données issues d'un pointage d'un mouvement circulaire uniforme

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

t **=** np**.**array**([**0.0**,**0.39269908169872414**,**0.7853981633974483**,**1.1780972450961724**,**1.5707963267948966**,**1.9634954084936207**,**2.356194490192345**,**2.748893571891069**,**3.141592653589793**,**3.5342917352885173**,**3.9269908169872414**,**4.319689898685965**,**4.71238898038469**,**5.105088062083414**,**5.497787143782138**,**5.890486225480862**,**6.283185307179586**])**

x**=** np**.**array**([**2.0**,**1.8477590650225735**,**1.4142135623730951**,**0.7653668647301797**,**1.2246467991473532e-16**,-**0.7653668647301795**,-**1.414213562373095**,-**1.8477590650225735**,-**2.0**,-**1.8477590650225737**,-**1.4142135623730954**,-**0.7653668647301807**,-**3.6739403974420594e-16**,**0.76536686473018**,**1.4142135623730947**,**1.847759065022573**,**2.0**])**

y**=** np**.**array**([**0.0**,**0.7653668647301796**,**1.414213562373095**,**1.8477590650225735**,**2.0**,**1.8477590650225735**,**1.4142135623730951**,**0.7653668647301798**,**2.4492935982947064e-16**,-**0.7653668647301793**,-**1.414213562373095**,-**1.847759065022573**,-**2.0**,-**1.8477590650225733**,-**1.4142135623730954**,-**0.7653668647301808**,-**4.898587196589413e-16**])**

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

# affichage de la trajectoire + axes

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

plt**.**plot **(**x**,**y**,**"rx"**)**

plt**.**xlabel**(**"x en mètres"**)**

plt**.**ylabel**(**"y en mètres"**)**

plt**.**title**(**"Trajectoire dans le cas d'un\nmouvement circulaire uniforme"**)**

""" TRAVAIL 1:

Taper,ci-dessous le code permettant

de calculer Vx et Vy au point n°8"""

Vx**=** **(**x**[**9**]-**x**[**8**])/(**t**[**9**]-**t**[**8**])**

Vy**=** **(**y**[**9**]-**y**[**8**])/(**t**[**9**]-**t**[**8**])**

draw\_Vector2**(**8**,**Vx**,**Vy**,**"r"**,**10**)**

""" TRAVAIL 2:

Taper,ci-dessous le code permettant

de calculer Vx et Vy au point n°8

en utilisant la méthode dite "centrée" """

Vx**=** **(**x**[**9**]-**x**[**7**])/(**t**[**9**]-**t**[**7**])**

Vy**=** **(**y**[**9**]-**y**[**7**])/(**t**[**9**]-**t**[**7**])**

draw\_Vector2**(**8**,**Vx**,**Vy**,**"g"**,**10**)**

""" FIN du travail des élèves

------------------------------ """

# affichage de la grille, légende etc...

# la legende est écrite en Latex

plt**.**plot**(**0**,**0**,**"r"**,**label**=**r"$\overrightarrow{V\_8}=\frac{\overrightarrow{{M\_8}{M\_9}}}{Δt}$"**)**

plt**.**plot**(**0**,**0**,**"g"**,**label**=**r"$\overrightarrow{V\_8}=\frac{\overrightarrow{{M\_7}{M\_9}}}{2Δt}$"**)**

plt**.**plot**(**0**,**0**,**"ko"**)**

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

plt**.**legend**(**prop**={**'size'**:**14**})**

# le repère doit être orthonormé:

plt**.**axis**(**'equal'**)**

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

***Voici ci-dessous le code corrigé des parties 2 à 4 :***

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

# Programme Python permettant de tracer les positions d'un point M

# à partir d'abscisses et d'ordonnées issues d'un pointage (ou d'équations).

# et de tracer le vecteur variation vitesse en un point

# à destination des élèves qui peuvent modifier le code sous les parties roses

# L'utilisation des listes NUMPY permet de simplifier le code

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

# O. CHAUMETTE - Lycée JP SARTRE - 69500 BRON - olivier.chaumette@ac-lyon.fr

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

# IMPORTATION DES MODULES #

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

# importation de NUMPY (pour gestion tableaux et calculs) sous l'alias "np"

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

# importation de PYPLOT (du module MATPLOTLIB, pour le tracé de courbes) sous l'alias "plt"

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

# modules permettant d'importer des données

# le fichier "import\_donnees2.py" est indispensable

# (version 2 avec bonnes équations horaires)

**from** import\_donnees\_meca2 **import** traiteDonnees

# Initialisation des tableaux numpy contenant x,y,t et les coordonnées des vitesses

x**=**np**.**array**([])**

y**=**np**.**array**([])**

t**=**np**.**array**([])**

Vx**=**np**.**array**([])**

Vy**=**np**.**array**([])**

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

# Fonction qui simplifie le tracé du vecteur vitesse

# afin que l'élève n'ait pas à définir l'origine

# du vecteur en termes de coordonnées

# (pour ne pas mélanger des positions et des coordonées de vecteurs)

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

**def** draw\_Vector2**(**numero\_du\_point**,** abscisse\_vecteur**,** ordonnee\_vecteur**,** couleur**,**echelle**):**

global x,y

plt.quiver(x[numero\_du\_point], y[numero\_du\_point],abscisse\_vecteur,ordonnee\_vecteur,color=couleur,scale=echelle)

""" ---------------------------------

CHOIX du PROFESSEUR """

# Logiciel utilisé à choisir parmi :

# 'pymecavideo' (exporter un fichier TXT)

# 'latispro' (fichier TXT créé par LATISPRO, avec décimale VIRGULE et séparation POINT VIRGULE)

# 'avimeca' (crééer un fichier TXT en export)

# 'avistep' (pointage exporté depuis AVISTEP:

# 'equation' (équation paramétrées saisies dans le fichier "import\_donnees2")

# 'entrees' (données de pointage saisies "à la main" dans le fichier "import\_donnees2.py)

Logiciel\_Utilise **=** 'latispro'

""" ---------------------------------

FIN du CHOIX du PROFESSEUR """

t**,** x**,** y **=** traiteDonnees**(**Logiciel\_Utilise**)**

Nbre\_Mesures**=**len**(**x**)**

""" ###########################################

Ici commence le travail à réaliser par les élèves """

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

# Tracé de la trajectoire

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

""" TRAVAIL 1:

Taper ci-dessous le code Python permettant de tracer l'ordonnée du point en fonction de son abscisse. """

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

""" TRAVAIL 2:

taper, à la place des points entre guillemets, les légendes pour les axes,

le titre du graphique"""

plt**.**title**(**"Trajectoire du point point M"**)** # titre du graphique

plt**.**xlabel**(**"Abscisse X du système (en m)"**)** # nom de la grandeur sur l'axe X

plt**.**ylabel**(**"Ordonnée Y du système (en m)"**)** # nom de la grandeur sur l'axe Y

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

# Vecteur vitesse

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

""" TRAVAIL 3:

Taper, DANS LA BOUCLE ci-dessous, le code python permettant:

- d'ajouter au tableau Vx l'abscisse du vecteur vitesse au point numéro "n"

- d'ajouter au tableau Vx l'abscisse du vecteur vitesse au point numéro "n" """

**for** n **in** range **(**1**,**Nbre\_Mesures**-**1**):**

Vx**=**np**.**append**(**Vx**,(**x**[**n**+**1**]-**x**[**n**-**1**])/(**t**[**n**+**1**]-**t**[**n**-**1**]))**

Vy**=**np**.**append**(**Vy**,(**y**[**n**+**1**]-**y**[**n**-**1**])/(**t**[**n**+**1**]-**t**[**n**-**1**]))**

draw\_Vector2**(**n**,**Vx**[**n**-**1**],**Vy**[**n**-**1**],**"b"**,**20**)**

""" ne pas toucher à la ligne ci-dessous """

Nbre\_vitesses**=**len**(**Vx**)** # nombre de vitesses disponibles.

# Remarque: Nbre\_vitesses=Nre\_mesures-2

# puisque pour le calcul de Vx, Vy, on ne prend pas le

# 1er et le dernier point

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

# Vecteur variation de vitesse

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

""" TRAVAIL 4:

Taper ci-dessous le code python permettant:

- de créer une boucle (compteur de boucle nommé "n") allant de 1 à Nbre\_vitesses-1

- de mettre dans une variable DeltaVx l'abscisse du vecteur variation de vitesse au point n

- de mettre dans une variable DeltaVy l'ordonnée du vecteur variation de vitesse au point n

- de tracer le vecteur variation vitesse au point "n+1" en vert avec une échelle de 5"""

**for** n **in** range**(**1**,**Nbre\_vitesses**-**1**):**

DeltaVx**=**Vx**[**n**+**1**]-**Vx**[**n**-**1**]**

DeltaVy**=**Vy**[**n**+**1**]-**Vy**[**n**-**1**]**

draw\_Vector2**(**n**+**1**,**DeltaVx**,**DeltaVy**,**"g"**,**5**)**

""" fin du travail """

# On affiche une grille

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

# le repère doit être orthonormé:

plt**.**axis**(**'equal'**)**

# on affiche la fenêtre

#plt.get\_current\_fig\_manager().window.state('zoomed') # <-- Affiche la fenêtre maximisée sous windows uniquement

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

**MODE D’EMPLOI des programmes Python de mécanique (formation 2019)**

*L’objectif de cette fiche est d’expliquer comment utiliser les programmes Python de mécanique proposés lors de la formation Académique de 2019.*

**1. Pourquoi cette notice ?**

Les programmes py de mécanique proposés nécessitent l’importation de données de pointage (abscisse, ordonnée et date). **Une ligne de ces programmes dit être modifiée par le professeur** en tenant compte du logiciel de pointage utilisé par les élèves.

**Attention, pour tous les programmes Python de mécanique, le fichier “import\_donnees\_meca.py” doit être présent dans le même dossier que le programme (car c’est lui qui permet d’importer les données)**

**2. Choix, par le professeur, de la manière d’importer des données dans le programme**

**La ligne à modifier se situe, dans le programme, au niveau du commentaire “CHOIX DU PROFESSEUR” :**

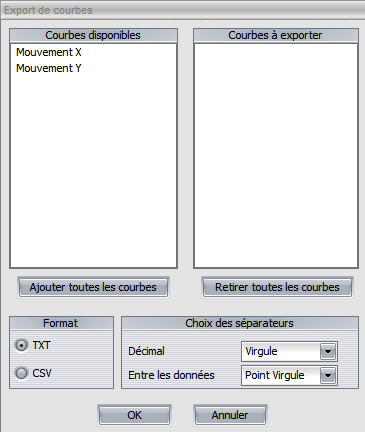
Dans la version par défaut, la ligne est la suivante : **Logiciel\_Utilise = 'latispro'**

S’il le souhaite, le professeur peut remplacer **‘latispro’** par un des choix proposés : **‘pymecavideo’** ou **‘avimeca’** ou **‘avistep’** ou **‘equations’** ou **‘entree’** (ATTENTION : bien respecter la casse et ne pas mettre d’accents)

**‘latispro’** ou **‘pymecavideo’** ou **‘avimeca’** ou **‘avistep’**: nom du logiciel utilisé.

**‘equations’** : le professeur saisit les équations horaires dans le fichier “import\_donnees\_meca.py” (voir explication § 7)

**‘entree’** : le professeur saisit les données « à la main » dans le fichier “import\_donnees\_meca.py” (voir explication § 8)



**3. Export des données depuis LATISPRO**

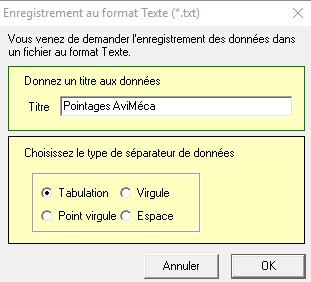
**1.** Une fois le pointage réalisé (Mouvement X, Mouvement Y et Temps seulement), choisir Menu **Fichier** >> **Exportation**.

**2. “Ajouter toutes les courbes”**. Vérifier que TXT est coché (dans “Format”) et que le **Choix des séparateurs** est **Virgule** (pour “Décimal”) et **Point Virgule** (pour “Entre les données”).

**4. Export des données depuis AVISTEP**

**1.** Une fois le pointage réalisé, choisir Menu **Résultats** >> **Tableau des valeurs**.

**2.** Dans la fenêtre Tableau des valeurs, choisir Menu **Fichier** >> **Enregistrer**.



**5. Export des données depuis AVIMECA**

**1.** Une fois le pointage réalisé, choisir Menu **Fichier** >> **Mesures >> Enregistrer dans un Fichier >> Format Texte (\*.txt)**.

**2.** Dans la fenêtre, vérifier que « **Tabulation** » est bien coché. Puis **OK**.

**6. Export des données depuis PYMECAVIDEO**

PyMecaVideo est un logiciel gratuit, libre et multiplateforme disponible à l’adresse : <http://outilsphysiques.tuxfamily.org/wiki/index.php?title=Pymecavideo>

Une fois le pointage réalisé, choisir Menu **Fichier >> Enregistrer les données…..**

**7. Choix EQUATIONS**

**1.** Charger le programme “**import\_donnees\_meca.py**” dans l’éditeur Python (EduPython ou Spyder par exemple)

**2.** Modifier la durée et les équations horaires dans le code suivant :

**elif origine=='equation':**

**""" CAS 2:**

**X ET Y ISSUES D'EQUATIONS HORAIRES DONNEES PAR LE PROFESSEUR**

**(méthode rapide)"""**

**tmax = 2 Saisir ici la durée maximale du temps en seconde**

**Nbre\_Mesures = 20 Saisir ici le nombre de points voulus. L’intervalle de temps sera tmax/Nombre\_Mesures**

**t = np.linspace(0, tmax,Nbre\_Mesures)**

**x=2\*t Saisir ici l’équation horaire x(t)**

**y=-4.9\*t\*t+2\*t+1 Saisir ici l’équation horaire y(t)**

**3.** Enregistrer le programme “**import\_donnees\_meca.py**” (sans changer son nom)

**8. Choix ENTREE**

**1.** Charger le programme “**import\_donnees\_meca.py**” dans l’éditeur Python (EduPython ou Spyder par exemple)

**2.** Modifier le contenu des tableaux x, y et t dans le code suivant :

**if origine=='entree':**

**""" CAS 1:**

**X ET Y ISSUES DU POINTAGE et entrées à la main"""**

**x=np.array([0,0,0,0,0,0,0,0,0]) Saisir ici les valeurs de x (séparées par une virgule)**

**y=np.array([9.86E-3,-5.92e-2,-1.48E-1,-2.76e-1,-4.24E-1,-6.02E-1,-8.48E-1,-1.07,-1.39])**

**Saisir ici les valeurs de y (séparées par une virgule)**

**t=np.array([0,0.033,0.067,0.1,0.133,0.167,0.2,0.233,0.267]) Saisir les valeurs de t**

**3.** Enregistrer le programme “**import\_donnees\_meca.py**” (sans changer son nom)

**9. En cas d’erreurs d’importation**

Il peut arriver que l’exportation par les logiciels enregistre des données absurdes ou fausses. Ces données font que Python renvoie une erreur.

Dans ce cas, le professeur peut ouvrir le fichier exporté avec le bloc-note Windows et supprimer « à la main »les lignes à l’origine de l’erreur.

***Exemple d’un fichier Latispro incorrect :***

Depuis le bloc-note, supprimer cette ligne « à la main » et enregistrer le fichier

**Temps;Mouvement X;Temps;Mouvement Y**

**0;0,00902378999179662;0;1,18229227704207**

**NAN ;NAN ;NAN,NAN**

**0,04;0,0986757295206844;0,04;1,32995429508965**

**0,08;0,20414859955467;0,08;1,47761631313723**