

```

# variation_vitesse.py

0001 | # Travaux pratiques P03 - Vecteur variation-de-vitesse
0002 | from matplotlib.pyplot import *
0003 | from mpl_toolkits.axes_grid1 import host_subplot
0004 | import pandas as pd
0005 | import numpy as np
0006 | # Les fonctions-----
0007 | #
0008 | def extract_donnees (fichier):
0009 | #fichier=input("Quel est le nom du fichier de pointage (sans l'extension .csv?)"+" ".csv"
0010 | # Récupération et lecture des données (sans entête) à partir des données du fichier CSV
0011 |     coordonnee = pd.read_csv(fichier, sep = ';', names = ['t', 'x', 'y'])
0012 | # Conversion avec le point en séparateur décimal
0013 |     L="txy"
0014 |     for var in L :
0015 |         coordonnee[var] = [x.replace(',', '.') for x in coordonnee[var]]
0016 |         coordonnee[var] = coordonnee[var].astype(float)
0017 | # Création des listes des variables
0018 |     date=coordonnee['t'].values.tolist()
0019 |     abscisse=coordonnee['x'].values.tolist()
0020 |     ordonnee=coordonnee['y'].values.tolist()
0021 |     print("valeurs des positions - coordonnées cartésiennes") # affichage du résultat sous forme d'un tableau
0022 |     print(coordonnee)
0023 |     return date,abscisse,ordonnee
0024 | #-----
0025 | def calcul_vitesses_aval(date,abscisses,ordonnees):
0026 |     x=np.array(abscisses) #
0027 |     y=np.array(ordonnees) #
0028 |     t=np.array(date) #
0029 |     v=[] # valeur approchée du module de la vitesse
0030 |     v_x=np.round((x[1:]-x[:-1])/(t[1:]-t[:-1]),2) # abscisse de la vitesse
0031 |     v_y=np.round((y[1:]-y[:-1])/(t[1:]-t[:-1]),2) # ordonnées de la vitesse
0032 |     v=np.round((np.sqrt(v_x**2+v_y**2),2) # norme de la vitesse
0033 |     t=t[:-1] # on enlève la dernière date pour laquelle il n'y a de valeur
0034 |     vitesse_data=pd.DataFrame({'Vx':v_x,'Vy':v_y,'V':v,index=t,dtype=float})
0035 |     print("valeurs de la vitesse - méthode par l'aval") # affichage du résultat sous forme d'un tableau
0036 |     print(vitesse_data)
0037 |     return v_x,v_y,v
0038 | #-----
0039 | def representation_graphique(x,y,v_x,v_y): # représentation des vecteurs
0040 |     fig, (ax1,ax2,ax3) = subplots(nrows=1, ncols=3,figsize=(12,6))
0041 |     #-----
0042 |     subplot(1,2,1)
0043 |     g1, = ax1.plot(x, y, "g+")
0044 |     title('Vecteur vitesse instantannée')
0045 |     xlabel('x en (m)')
0046 |     ylabel('Altitude z en (m)')
0047 |     ax1.axis("equal") # orthonormé
0048 |     xlim(min(x)-2,1.2*max(x))
0049 |     for i in range(0,len(v_x)-1):
0050 |         V=quiver(x[i],y[i],v_x[i],v_y[i],units='xy',
0051 |             scale_units='xy',angles='xy', scale=10)
0052 |     #-----
0053 |     subplot(1,2,2)
0054 |     g2, = ax2.plot(x, y, "g+")
0055 |     title('Variation du vecteur vitesse')
0056 |     xlabel('x en (m)')
0057 |     ylabel('Altitude z en (m)')
0058 |     xlim(min(x)-2,1.2*max(x))
0059 |     ax2.axis("equal") # orthonormé
0060 |     for i in range(0,len(v_x)-1):
0061 |         DV=quiver(x[i],y[i],v_x[i+1]-v_x[i],v_y[i+1]-v_y[i],units='xy',
0062 |             scale_units='xy',angles='xy', scale=10, color='red')
0063 |     #-----
0064 |     fig.tight_layout()
0065 | #-----
0066 | def graphXY(t,x,y): # Affichage des lois horaires du mouvement et de la trajectoire
0067 |     fig, (ax1,ax2) = subplots(nrows=1, ncols=2,figsize=(12,6))
0068 |     #-----
0069 |     subplot(1,2,1)
0070 |     grid()
0071 |     twin = ax1.twinx() # même axe des abscisses
0072 |     p1, = ax1.plot(t, x, "b+", label="coord. X") # graphique n°1 - axe des ordonnées =ax1
0073 |     p2, = twin.plot(t, y, "r+", label="coord. Y") # graphique n°2 - axe des ordonnées =twin
0074 |     min_axe=min(min(x),min(y)) # calcul des échelles graphiques
0075 |     max_axe=1.2*max(max(x),max(y))
0076 |     ax1.set_title('Coordonnées') # paramétrage de la représentation graphique
0077 |     ax1.set_xlim(min(t), max(t)+t[1])
0078 |     ax1.set_ylim(min_axe, max_axe)
0079 |     ax1.set_xlabel("$date \; t$")
0080 |     ax1.set_ylabel("$x(t)$")
0081 |     ax1.yaxis.label.set_color(p1.get_color())
0082 |     twin.set_ylim(min_axe,max_axe)
0083 |     twin.set_ylabel("$y(t)$")
0084 |     twin.yaxis.label.set_color(p2.get_color())
0085 |     tkw = dict(size=4, width=1.5)
0086 |     ax1.tick_params(axis='y', colors=p1.get_color(), **tkw)

```

```

087| ax1.tick_params(axis='x', **tkw)
088| ax1.legend(handles=[p1, p2],loc='upper left')
089| twin.tick_params(axis='y', colors=p2.get_color(), **tkw)
090| #-----
091| subplot(1,2,2)
092| grid()
093| p3, = ax2.plot(x, y, "g+") # graphique n°3 - axe des ordonnées =ax2
094| ax2.set_title('Trajectoire')
095| ax2.set_xlabel('x en (m)')
096| ax2.set_ylabel('y en (m)')
097| ax2.xaxis.label.set_color(p3.get_color())
098| ax2.yaxis.label.set_color(p3.get_color())
099| ax2.tick_params(axis='x', colors=p3.get_color(), **tkw)
100| ax2.tick_params(axis='y', colors=p3.get_color(), **tkw)
101| ax2.set_xlim(min(x)-x[1],1.2*max(x))
102| ax2.axis("equal") # orthonormé
103| #-----
104| fig.tight_layout()
105| #-----
106| def graphVxVy(t,vx,vy): # Affichage des lois horaires de la vitesse et de l'hodographe
107|     fig, (ax1,ax2) = subplots(nrows=1, ncols=2,figsize=(12,6))
108|     #-----
109|     subplot(1,2,1)
110|     grid()
111|     twin = ax1.twinx() # même axe des abscisses
112|     p1, = ax1.plot(t, vx, "b+", label="coord. Vx") # graphique n°1 - axe des ordonnées =ax1
113|     p2, = twin.plot(t, vy, "r+", label="coord. Vy") # graphique n°2 - axe des ordonnées =twin
114|     min_axe=min(min(vx),min(vy)) # calcul des échelles graphiques
115|     max_axe=1.2*max(max(vx),max(vy))
116|     ax1.set_title('Coordonnées') # paramétrage de la représentation graphique
117|     ax1.set_xlim(min(t), max(t)+t[1])
118|     ax1.set_ylim(min_axe, max_axe)
119|     ax1.set_xlabel("$date \; t$")
120|     ax1.set_ylabel("$V_x(t)$")
121|     ax1.yaxis.label.set_color(p1.get_color())
122|     twin.set_ylim(min_axe,max_axe)
123|     twin.set_ylabel("$V_y(t)$")
124|     twin.yaxis.label.set_color(p2.get_color())
125|     tkw = dict(size=4, width=1.5)
126|     ax1.tick_params(axis='y', colors=p1.get_color(), **tkw)
127|     ax1.tick_params(axis='x', **tkw)
128|     ax1.legend(handles=[p1, p2],loc='upper left')
129|     twin.tick_params(axis='y', colors=p2.get_color(), **tkw)
130|     #-----
131|     subplot(1,2,2)
132|     grid()
133|     p3, = ax2.plot(vx, vy, "g+") # graphique n°3 - axe des ordonnées =ax2
134|     for i in range(0,len(t),1):
135|         V=quiver(0,0,vx[i],vy[i],units='xy',scale_units='xy',angles='xy', scale=1)
136|     ax2.set_title('Hodographe')
137|     ax2.set_xlabel('Vx en (m)')
138|     ax2.set_ylabel('Vy en (m)')
139|     ax2.yaxis.label.set_color(p3.get_color())
140|     ax2.tick_params(axis='y', colors=p3.get_color(), **tkw)
141|     ax2.xaxis.label.set_color(p3.get_color())
142|     ax2.tick_params(axis='x', colors=p3.get_color(), **tkw)
143|     ax2.set_xlim(-1.2*abs(min(vx)),1.2*abs(max(vx)))
144|     ax2.set_ylim(-1.2*abs(min(vy)),1.2*abs(max(vy)))
145|     ax2.axis("equal") # orthonormé
146|     #-----
147|     fig.tight_layout()
148| #Le programme principal-----
149| nom_fichier=input("Quel est le nom du fichier de pointage (sans l'extension .csv)? : ")+" .csv"
150| t0,x,y=extract_donnees(nom_fichier)
151| graphXY(t0,x,y)
152| vx,vy,v=calcul_vitesses_aval(t0,x,y)
153| t1=t0[:-1]
154| graphVxVy(t1,vx,vy)
155| representation_graphique(x,y,vx,vy)
156| show()
157| sys.exit()

```