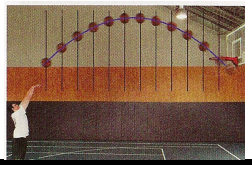
CINEMATIQUE
MOUVEMENT PARABOLIQUE
MOUVEMENT CIRCULAIRE



OBJECTIFS

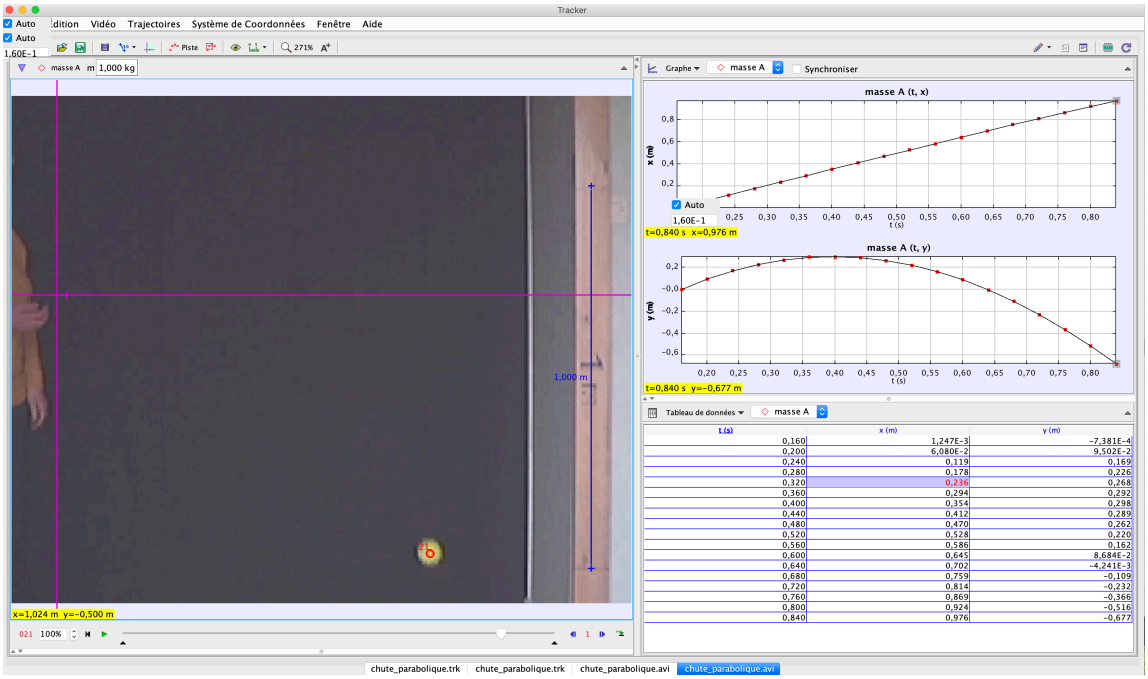
✓ Savoir utiliser un logiciel de pointage vidéo.
✓ Savoir calculer et tracer les vecteurs vitesse et accélération, comprendre leur signification physique.
✓ Comparer les résultats calculés à partir des lois de la physique et comparer aux résultats expérimentaux.
✓ Mesurer g.

MATERIEL

✓ Logiciel Tracker et environnement de programmation Python.

1. POINTAGE VIDEO AVEC LE LOGICIEL LIBRE TRACKER

Nous allons utiliser le ***logiciel libre Tracker*** pour faire un pointage vidéo de la trajectoire d'un objet, c'est-à-dire récupérer les coordonnées $x(t)$ et $y(t)$ de sa trajectoire en fonction du temps.



Vous pouvez installer ce logiciel libre sur n'importe quelle plateforme ici : ***https://physlets.org/tracker/***

Tracker Home | Help | Share | OSP Home | Discussion Group | Email Doug

Tracker
Video Analysis and Modeling Tool

Try Tracker Online

Over 1 million users in 26 languages. Completely free and open source.

Latest Tracker installers: Windows | Recent MacOS | Older MacOS | Linux

Installer Help Change Log Discussion Forum

Tip: save your work as a Tracker Project. Easy to build and share. Easy to browse in the Library Browser.

2. PROJECTILE : TRAJECTOIRE PARABOLIQUE

POINTAGE AVEC TRACKER

1) Ouvrez la vidéo ***mvt_parabolique.avi.*** Avec le logiciel Tracker, récupérez les coordonnées $x(t)$ et $y(t)$ de la trajectoire en fonction du temps et enregistrez-les dans un fichier mvt_parabolique.txt.

Choisissez l'origine des temps, l'instant où la balle quitte la main du lanceur, et l'origine des positions, la position de la balle quittant la main. Il faudra être très précis sur le pointage !

LECTURE DU FICHIER PAR PYTHON POUR RECUPERER `temps, x, y`

2) Le fichier que vous avez récupéré est de la forme suivante, il faudra mettre des # devant les valeurs non numériques pour ne pas les lire (cf. code ci-dessous).

mvt_parabolique.txt

```
#t      x       y\
0.16    0.0012472599646126264   -7.381002697561811E-4
0.2     0.06080262836211074     0.09501759009484856
0.24    0.11919024443808922     0.16858598635058153
0.28    0.17757786051406776     0.22580585010504045
0.32    0.23596547659004624     0.267844933679745
0.36    0.2943530926660247      0.292367732431656
0.4     0.3539084610635228      0.2982064940392538
0.44    0.4122960771395014      0.28886447546709726
0.48    0.46951594089396026     0.26200617207214716
0.52    0.5279035569699388      0.21996708849744262
0.56    0.5862911730459174      0.1615794724214641
0.6     0.6446787891218959      0.08684332384421162
0.64    0.7018986528763549      -0.0042413572343148465
0.68    0.7591185166308138      -0.10933906617107614
0.72    0.8140028757422336      -0.2319530599306311
0.76    0.8688872348536534      -0.3662445769053816
0.8     0.9237715939650731      -0.5157168740598866
0.84    0.9763204484334539      -0.6768666944295875
```

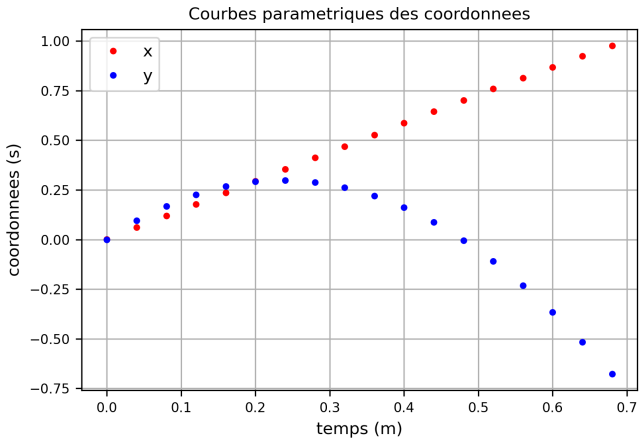
L'objectif est de récupérer trois tableaux pour pouvoir travailler dessus avec Python : `temps, x, y`. La fonction suivante permet de réaliser cette extraction (cf. cours d'informatique 13):

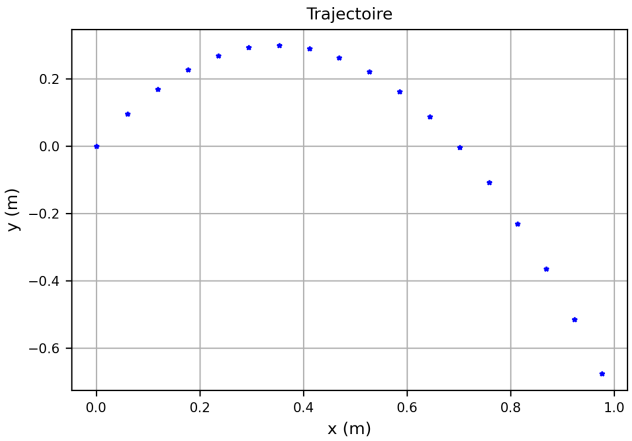
```
def extraction_data(nom_fichier):
    fichier = open(nom_fichier,'r')
    tps = []
    abcisse_x = []
    ordonnee_y = []
    Liste_lignes = fichier.readlines()
    for i in range (3,len(Liste_lignes)): # on ne lit pas les lignes avec #
            lignei = Liste_lignes[i].strip("\n")
            L_ligne_i = lignei.split("\t")
            tps.append(float(L_ligne_i[0])-0.16)
            #le premier pointage commence Ã  0.16 s
            abcisse_x.append(float(L_ligne_i[1]))
            ordonnee_y.append(float(L_ligne_i[2]))
            x = np.array(abcisse_x)
            y = np.array(ordonnee_y)
            temps = np.array(tps)
    fichier.close()
    return temps,x,y
```

⌨ Tapez le code pour récupérer les trois tableaux.

⌨ Le code suivant permet de tracer les courbes paramétriques de la trajectoire, vous pouvez arranger la mise en page à votre guise. Vous pouvez aussi tracer la courbe de la trajectoire.

```
plt.plot(temps,x,'or', markersize=3)
plt.plot(temps,y,'ob', markersize=3)
plt.grid()
plt.title('Courbes parametriques des coordonnees',size=10)
plt.xlabel('temps (s)',size=10)
plt.ylabel('coordonnees (m)',size=10)
plt.legend(['x','y'],fontsize=10)
plt.xticks(size=8)
plt.yticks(size=8)
plt.show()
```





OBTENTION DES VITESSES vx *ET* vy

3) Les composantes des vitesses sont définies par :

$$v_x(t) = \lim_{\delta t \to 0} \frac{x(t+\delta t) - x(t-\delta t)}{2\delta t} \text{ et } v_y(t) = \lim_{\delta t \to 0} \frac{y(t+\delta t) - y(t-\delta t)}{2\delta t}.$$

que l'on peut approximer numériquement, si δt est « suffisamment petit », par (***dérivée centrée***, cf. cours d'informatique 10):

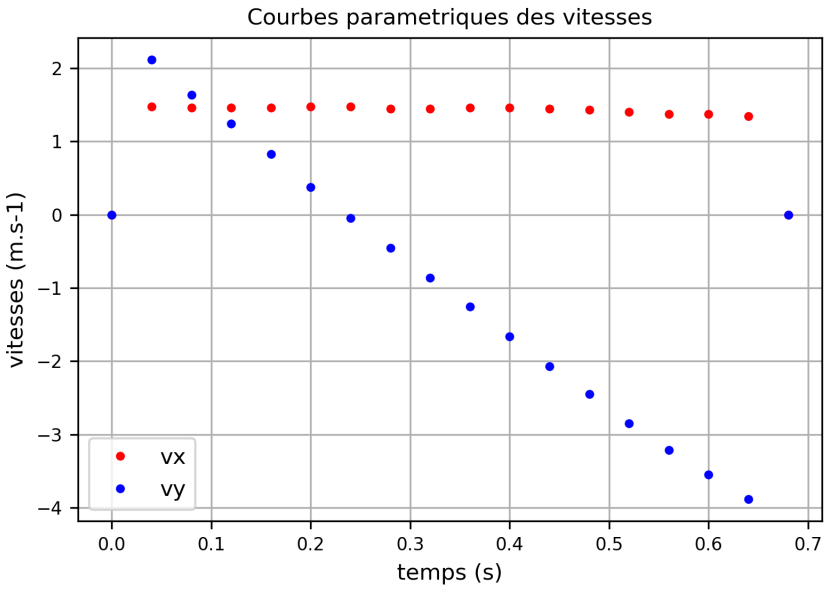
$$v_x(t) \approx \frac{x(t+\delta t) - x(t-\delta t)}{2\delta t} \text{ et } v_y(t) \approx \frac{y(t+\delta t) - y(t-\delta t)}{2\delta t}.$$

La fonction suivante génère les tableaux des vitesses (valeurs approchées) :

```
def calcul_vitesse(x,y,delta_temps):
    N = np.size(x)
    vx = np.zeros(N)
    vy = np.zeros(N)
    for i in range (1,N-1):
        vx[i] = (x[i+1]-x[i-1])/(2*delta_temps)
        vy[i] = (y[i+1]-y[i-1])/(2*delta_temps)
    return vx,vy
```

⌨ Tapez le code pour récupérer les tableaux des vitesses avec `delta_temps = temps[1]-temps[0]`.

⌨ De façon analogue aux courbes paramétriques des positions, tracez les courbes paramétriques des vitesses. Vous pouvez arranger la mise en page à votre guise.



OBTENTION DES ACCELERATIONS ax *ET* ay

4) Les composantes des accélérations sont définies par :

$$a_x(t) = \lim_{\delta t \to 0} \frac{v_x(t+\delta t) - v_x(t-\delta t)}{2\delta t} \text{ et } a_y(t) = \lim_{\delta t \to 0} \frac{v_y(t+\delta t) - v_y(t-\delta t)}{2\delta t}.$$

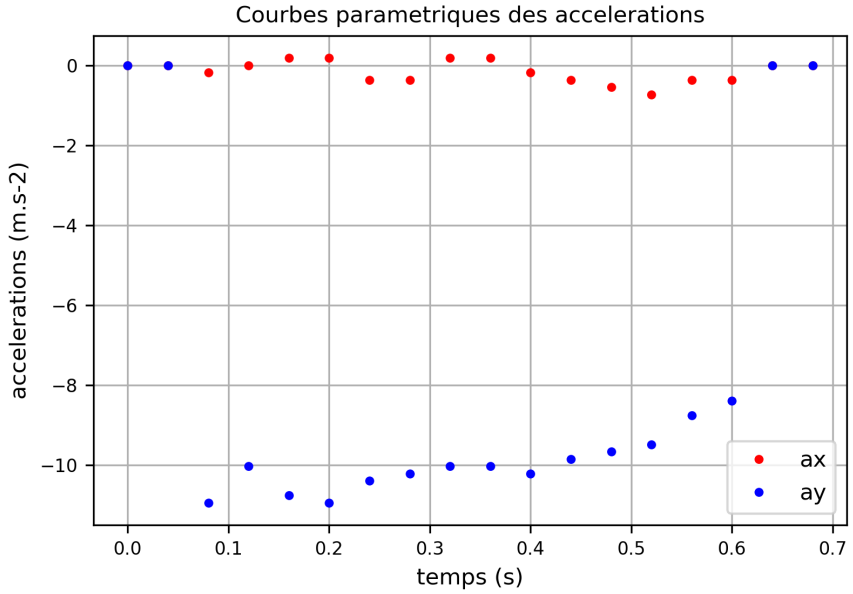
que l'on peut approximer numériquement, si δt est « suffisamment petit », par (***dérivée centrée***, cf. cours d'informatique 10):

:

$$a_x(t) \approx \frac{v_x(t+\delta t) - v_x(t-\delta t)}{2\delta t} \text{ et } a_y(t) \approx \frac{v_y(t+\delta t) - v_y(t-\delta t)}{2\delta t}.$$

⌨ De façon analogue aux cas des vitesses, définissez une fonction `calcul_acceleration(vx,vy,delta_temps)` qui, à partir des tableaux des vitesses, retournera les tableaux des accélérations `ax` et `ay`.

⌨ Tracez les courbes paramétriques des accélérations. Vous pouvez arranger la mise en page à votre guise.



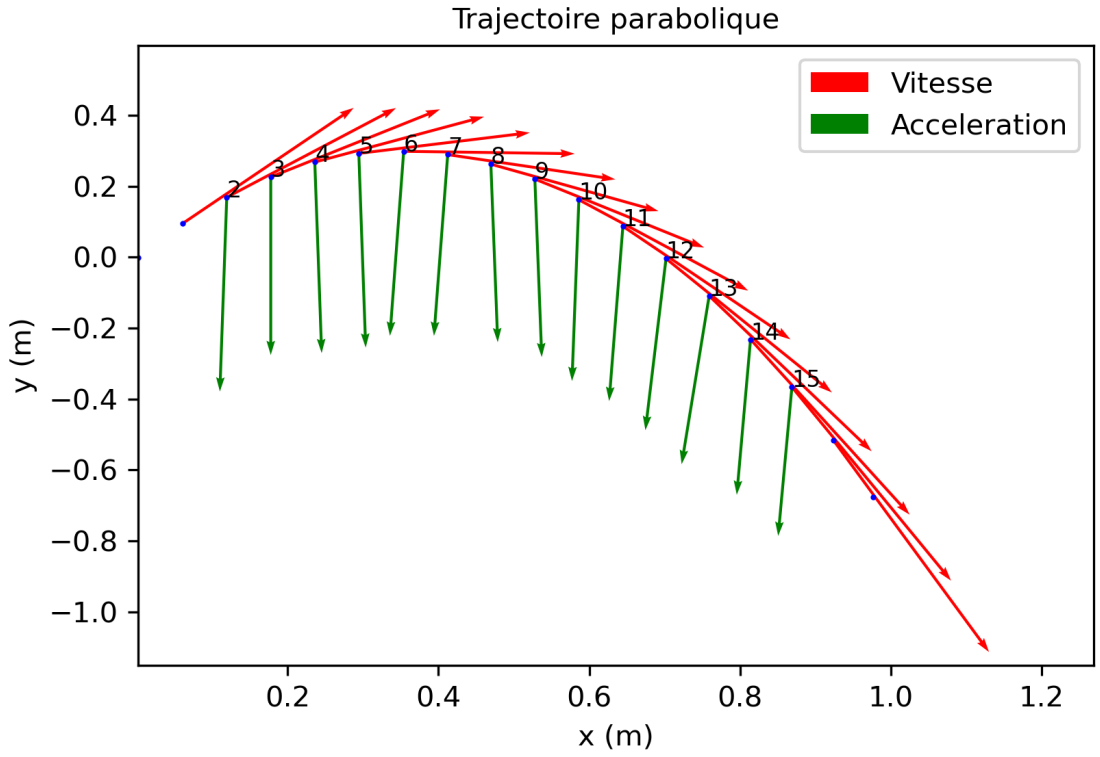
Remarque : le calcul des vitesses puis des accélérations est approché. On constate que les erreurs se propagent et s'amplifient, c'est pourquoi il faut faire un pointage précis dès le départ.

TRACE DES VECTEURS vx, vy,ax *ET* ay *SUR LA TRAJECTOIRE*

5) En utilisant la fonction `plt.quiver`, il est possible de tracer sur la trajectoire de la balle les vecteurs vitesse et accélération.

```
# Trace des vecteurs vitesse et acceleration sur la trajectoire
def trace_vecteurs_vitesse_acceleration(x,y,vx,vy,ax,ay):
    plt.plot(x,y,'o',color='blue', markersize=1)
    plt.xlabel('x (m)',size=10)
    plt.xlim(1.3*np.min(x),1.3*np.max(x))
    plt.ylabel('y (m)',size=10)
    plt.ylim(1.7*np.min(y),2.0*np.max(y))
    plt.xticks(size=10)
    plt.yticks(size=10)
    plt.title("Trajectoire parabolique",size=10)
    plt.quiver(x,y,vx,vy,angles='xy',scale_units='xy',scale=6.5,color='red',width=0.003,label='Vitesse')
    plt.quiver(x,y,ax,ay,angles='xy',scale_units='xy',scale=20,color='green',width=0.003,label='Acceleration')
    plt.legend(fontsize=10)
    for i in range (2,len(x)-2):
        plt.annotate(i,(x[i],y[i]),size=8)
    plt.savefig("parabolique",dpi=300)
    plt.show()
```

⌨ Tapez le code ci-dessus pour afficher les vecteurs sur votre trajectoire. Vous pouvez arranger la mise en page à votre guise.



EXPLOITATION

6) Grâce à votre programme (que vous allez compléter, modifier …) et un crayon+papier, réfléchissez et répondez aux points suivants :

• ***Vitesse initiale*** : Trouvez à l'aide des graphes les coordonnées v_{ox} et v_{oy} du vecteur $\overrightarrow{v_0}$ et déduisez-en sa norme v_0.

• ***Angle de tir*** : Trouvez la valeur de l'angle initial du lancé θ_0 (angle avec l'horizontale

• ***Graphique*** : A quel instant la balle passe-t-elle au sommet de sa trajectoire ? Comment est orienté le vecteur $\overrightarrow{v_S}$? Trouvez les coordonnées x_S et y_S du sommet de la trajectoire.

• ***Equation horaire*** : Retrouvez, par le calcul, l'instant de passage au sommet de la trajectoire. Calculez les coordonnées x_S et y_S du sommet de la trajectoire et comparez aux résultats précédents.

• ***Accélération*** : La trajectoire vérifie-t-elle $\vec{a} = \vec{g}$?

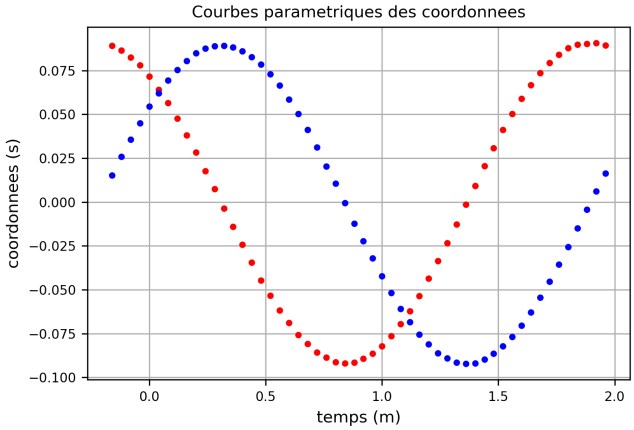
3. MOUVEMENT CIRCULAIRE

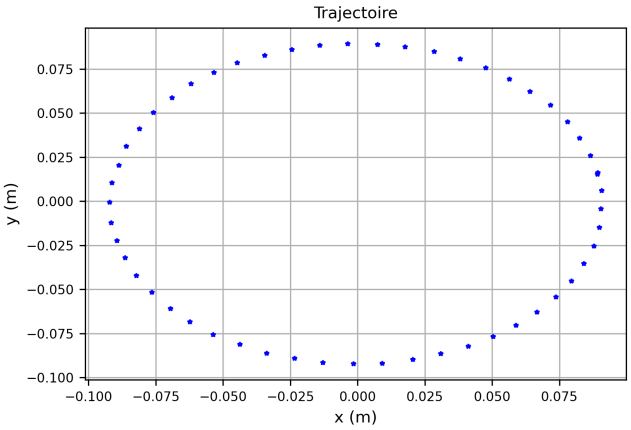
POINTAGE DANS AVIMECA ET CALCULS DANS REGRESSI

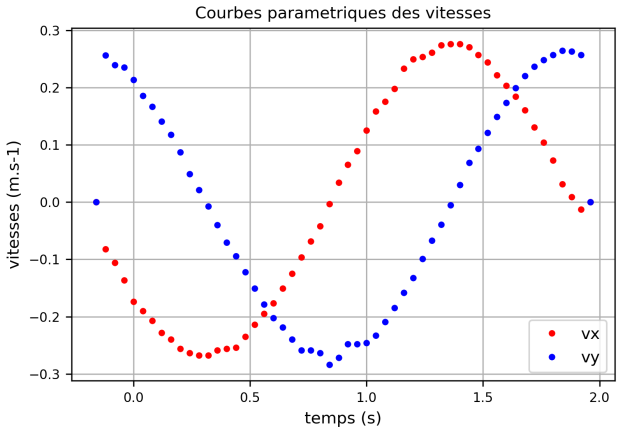
1) Ouvrez la vidéo ***mvt_circulaire.avi.*** Avec le logiciel Tracker, récupérez les coordonnées $x(t)$ et $y(t)$ de la trajectoire en fonction du temps et enregistrez-les dans un fichier mvt_circulaire.txt. On pointera le centre de la plaque dont la diagonale vaut 10 cm.
Choisissez l'origine des temps, l'instant où la balle quitte la main du lanceur, et l'origine des positions, la position de la balle quittant la main. Il faudra être très précis sur le pointage !

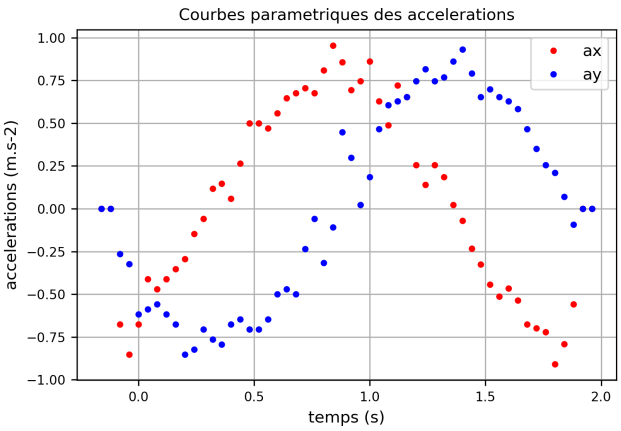
EXPLOITATION AVEC PYTHON

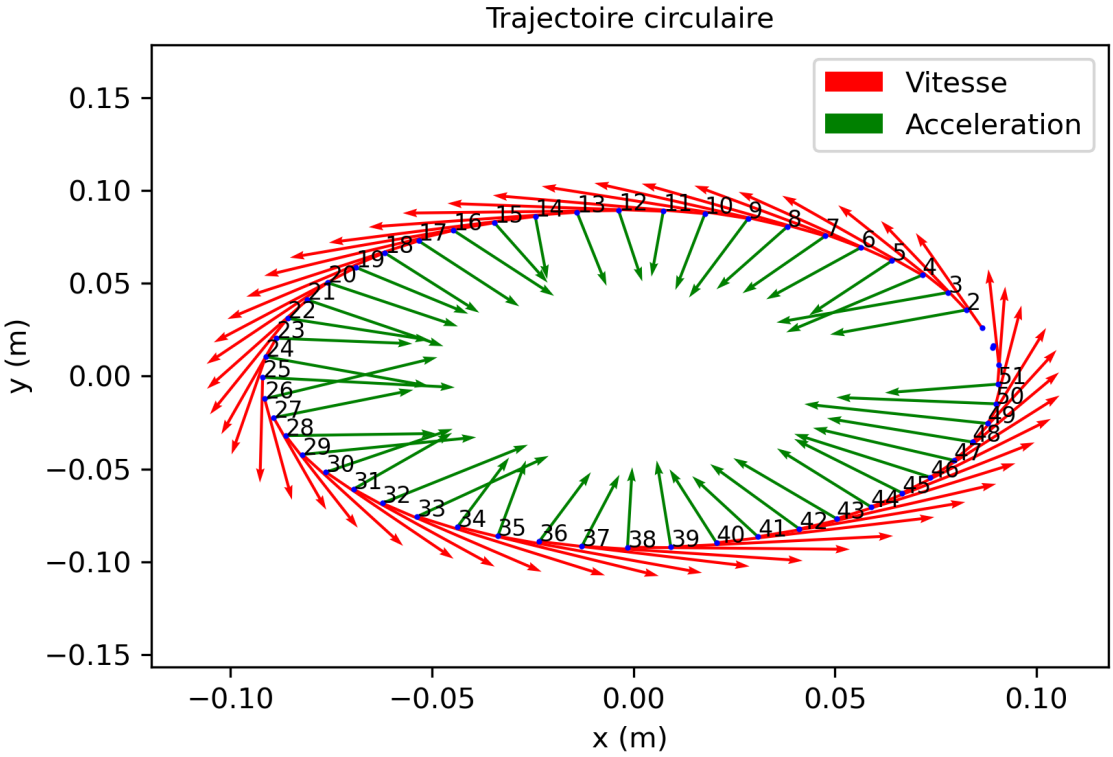
2) ⌨ Adaptez le code précédent pour exploiter et analyser le fichier mvt_circulaire.txt. Vous devriez obtenir les courbes suivantes:











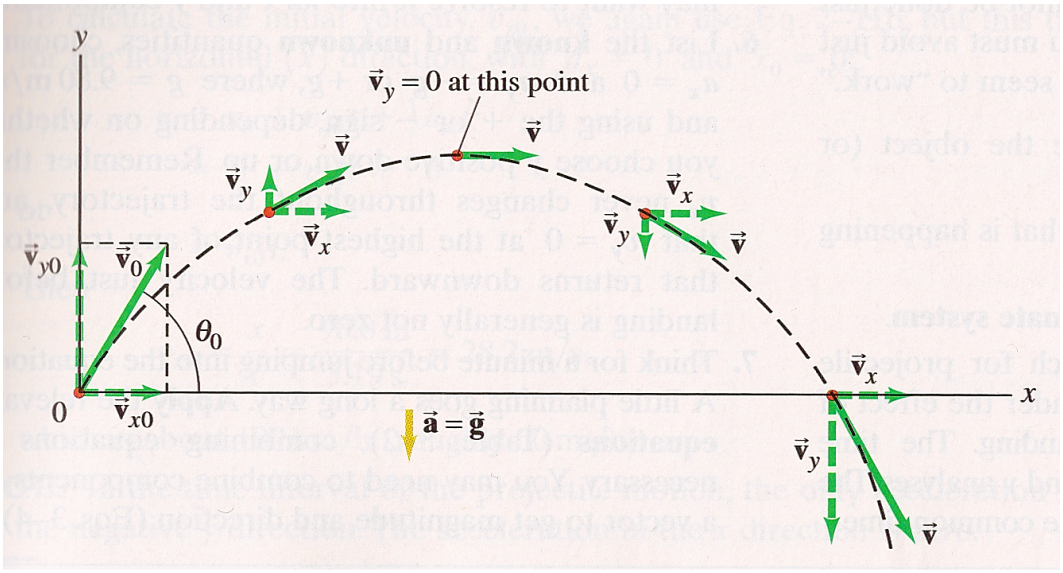
EXPLOITATION

3)  Grâce à votre programme (que vous allez compléter, modifier …) et un crayon+papier, réfléchissez et répondez aux points suivants :

• ***Vitesse*** : Déterminez la vitesse linéaire et la vitesse angulaire de la trajectoire. Il faudra déterminer le rayon de la trajectoire R.

• ***Accélération*** : Déterminez l'accélération radiale (centripète) et vérifiez que

ANNEXE 1 : EQUATION DU MOUVEMENT DANS LE CHAMP DE PESANTEUR



Ici tout frottement est négligé. Avec les conditions initiales de la figure, on obtient :

$$\vec{a}:\begin{cases} a_x = 0 \\ a_y = -g \end{cases} \quad \vec{v}:\begin{cases} v_x = v_0\cos\theta_0 \\ v_y = -gt + v_0\sin\theta_0 \end{cases} \quad \overrightarrow{OP}:\begin{cases} x = \left(v_0\cos\theta_0\right)t \\ y = -\frac{1}{2}gt^2 + \left(v_0\sin\theta_0\right)t \end{cases}$$

Equation cartésienne de la trajectoire $y = f\left(x\right)$:

$$y = -\left(\frac{g}{2v_0^2\left(\cos\theta_0\right)^2}\right)x^2 + \left(\tan\theta_0\right)x$$

ANNEXE 2 : CINEMATIQUE DU MOUVEMENT CIRCULAIRE

Caractéristiques : $r =$ constante et $\omega \equiv \dot{\theta}$ variable dans le cas général.

$$\overrightarrow{OP} = r\vec{u}_r \quad \vec{v} = r\dot{\theta}\vec{u}_r = r\omega\vec{u}_r \quad \vec{a} = -r\dot{\theta}^2\vec{u}_r + r\ddot{\theta}\vec{u}_\theta = -r\omega^2\vec{u}_r + r\dot{\omega}\vec{u}_\theta = -\frac{v^2}{r}\vec{u}_r + r\dot{\omega}\vec{u}_\theta$$