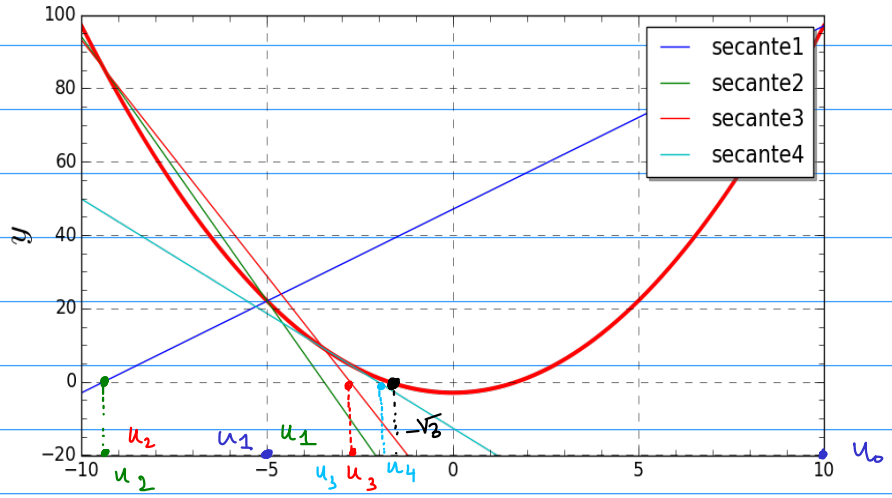
TP CN 1
Corrigé

I 1- Méthode la sécante

$f : x \to x^3 - 3$

(1)



La suite $(u_n)_{n\in\mathbb{N}}$ des zéro des sécantes successives à la courbe $\mathcal{C}_f$ semble converger vers le zéro de f (ici $-\sqrt{3}$)

(2) $u_n = g(u_{n-1}, u_{n-2})$

u_n = zéro de la sécante Δ à f en $M_{n-1}(u_{n-1}, f(u_{n-1}))$ et $M_{n-2}(u_{n-2}, f(u_{n-2}))$.

$h(u_n) = 0$ (*) avec $h : x \to h(x)$ equat° de Δ.

Equat° de Δ ?

Δ est une droite $\Leftrightarrow$ la affine soit $h(x) = ax + b$.

a et b ?

$$\begin{cases} M_{n-1} \in \Delta : f(u_{n-1}) = a u_{n-1} + b \quad (1) \\ M_{n-2} \in \Delta : f(u_{n-2}) = a u_{n-2} + b \quad (2) \end{cases}$$

(1) − (2) : $f(u_{n-1}) - f(u_{n-2}) = a(u_{n-1} - u_{n-2})$

$$\Leftrightarrow \boxed{a = \frac{f(u_{n-1}) - f(u_{n-2})}{u_{n-1} - u_{n-2}}}$$

(1) $\Rightarrow$ $\boxed{b = f(u_{n-1}) - a u_{n-1}}$

(*) $\Rightarrow$ $a u_n + b = 0 \Leftrightarrow \boxed{u_n = -\frac{b}{a}}$

③ def zero_secante(f, u0, u1, e):

```
    u, v = u0, u1                    # initialisation.
    while abs(v - u) >= e:
        a = (f(u)-f(v))/(u-v)        # u : u_{n-2}, v = u_{n-1}
        b = f(u) - a*u
        z = -b/a
        u = v
        v = z
    return z
```

```
def zero_secante(f,u0,u1,e):
    """
    Find a zero of the function f with an assumed precision of e
    performing Lagrange's method
    f : function
    u0,u1 : float, starting points for Lagrange's methode
    e: float, assumed gap between z and the real zero of f
    z: float, approximate returned value of the zero of f
    ------------Minimal example-----------------------
    f=lambda x:x**2-2
    zero_secante(f,0,10,1e-3)
    [returned value : 1.41421825734907361
    ----------------------------------------------------
    """
    u,v=u0,u1
    while abs(v-u)>e:
        a=(f(u)-f(v))/(u-v)
        b=f(u)-a*u
        z=-b/a
        u=v
        v=z
    return z
```

④ Evaluation de $\sqrt{3}$ à $2^{-52} \approx 2 \times 10^{-16}$ près :

Appel de la fonction zero_secante.

```
f=lambda x:x**2-3
print(zero_secante(f,0,3,2e-16))
```

⟶ 1.7320508075688772

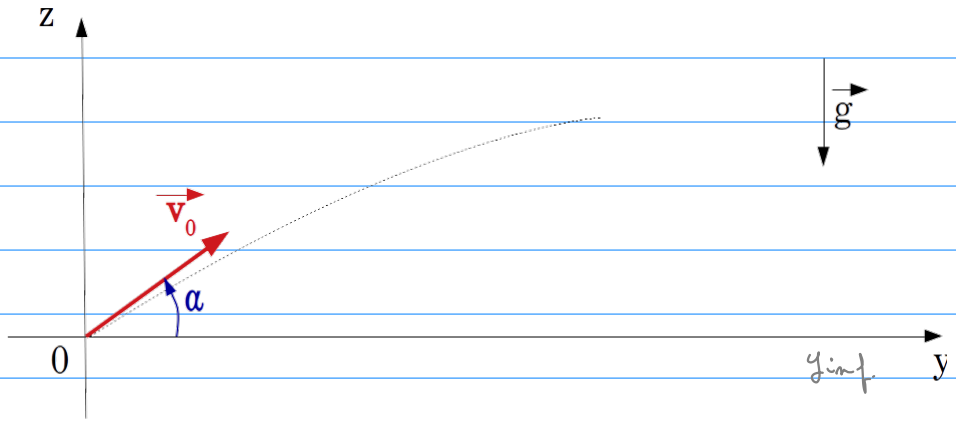
Bonus : est-ce que e est bien la précision sur le zéro de f ?

Test pour $f : x \underset{\mathbb{R}}{\longrightarrow} x^2 - 3 \underset{\mathbb{R}}{}$

e	z	sqrt(3)	en eau effective
1	1.5	1.7320508075688772	~ 0,2
0.1	1.7272727272727273	1.7320508075688772	~ 0,005
0.01	1.7319587628865979	1.7320508075688772	~ 0,0001
0.001	1.7320509347060415	1.7320508075688772	~ 10^{-7}
0.0001	1.7320509347060415	1.7320508075688772	~ 10^{-7}
1e-05	1.7320508075654999	1.7320508075688772	~ 10^{-12}
1e-06	1.7320508075654999	1.7320508075688772	~ 10^{-12}
1e-07	1.7320508075688772	1.7320508075688772	~ 10^{-16} ↑ précision machine.
1e-08	1.7320508075688772	1.7320508075688772	
1e-09	1.7320508075688772	1.7320508075688772	
1e-10	1.7320508075688772	1.7320508075688772	
1e-11	1.7320508075688772	1.7320508075688772	
1e-12	1.7320508075688774	1.7320508075688772	
1e-13	1.7320508075688774	1.7320508075688772	
1e-14	1.7320508075688774	1.7320508075688772	
1e-15	1.7320508075688774	1.7320508075688772	
1e-16	1.7320508075688772	1.7320508075688772	

Sur cet exemple, on constate que la précision effective sur la valeur de $\sqrt{3}$ est nettement meilleure que e.

I3 - Tir de projectile

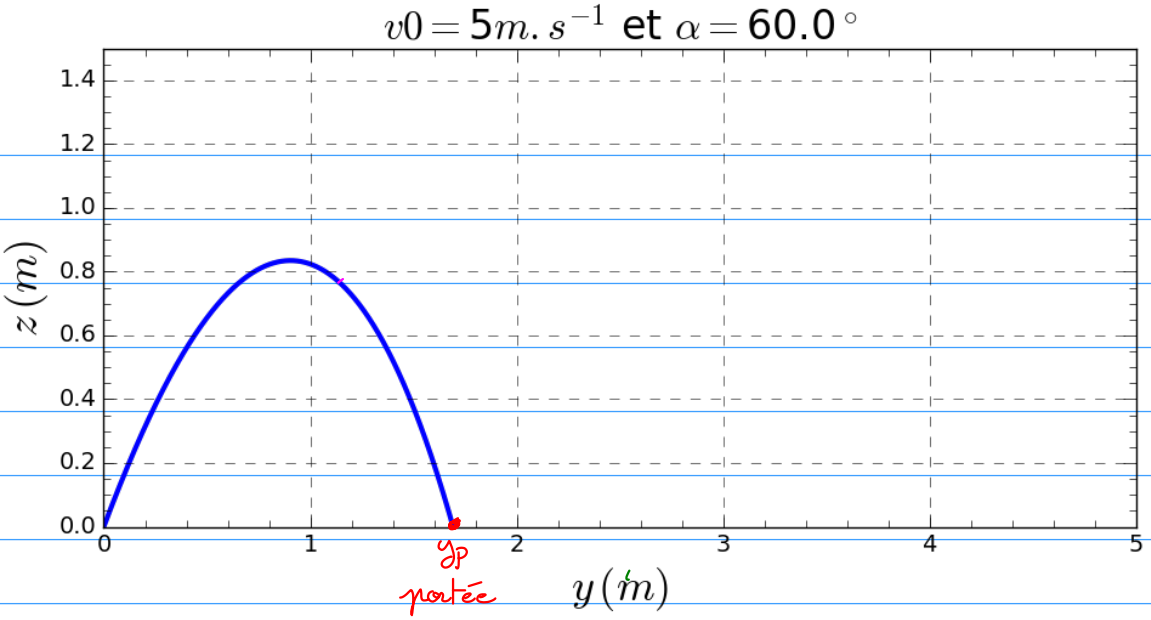


$y_{inf} = v_0 \cos\alpha \, T$

$T = m/s$

$z(y) = (gT^2 + v_0 \sin\alpha \, T)\, y/y_{inf} - gT^2 \times \ln\left(\frac{y_{inf}}{y_{inf} - y}\right)$

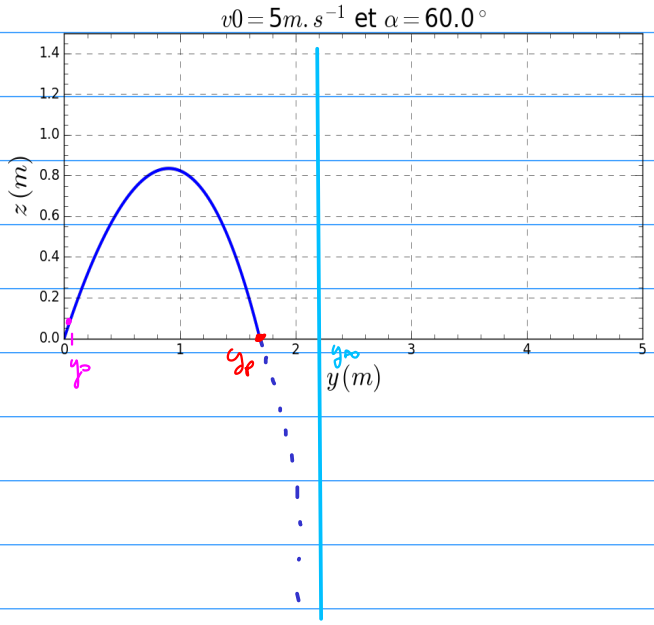
1)



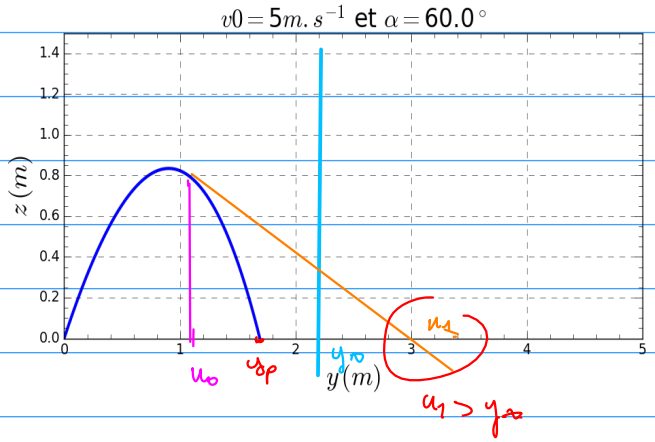

```
#Parametres constants
g=9.81#m.s-2
l=0.5#N.m-1.s
m=1#kg
tau=m/l
v0=5

def trajectoire(y):
    """
    Return z(y), trajectory of a projectile submitted to gravity and a
    linear air resistance given initial conditions y(0)=0, z=(0),
    dy/dt(0)=v0*cos(alpha), dz/dt=v0*sin(alpha)
    v0 : float, initial velocity
    alpha : float, angle between horizontal plan and initial velocity
    y: 1D-array, abscissa
    z(y) : 1D-array alitude
    Parameters v0,alpha, tau, g must be defined as global variables outside
    the function
    """
    y_inf=v0*np.cos(alpha)*tau
    return (g*tau**2+v0*np.sin(alpha)*tau)*(y/y_inf)-g*tau**2*np.log(y_inf/(y_inf-y))#z(y)

#Trajectoire
alpha=np.pi/3
plt.figure()
y_inf=v0*np.cos(alpha)*tau;#y<=y_inf
y=np.linspace(0,y_inf,100000,endpoint=False);
z=trajectoire(y);
plt.xlabel(r'$y\,(m)$')
plt.ylabel(r'$z\,(m)$')
plt.title(r'$v0=$'+'{}'.format(v0)+r'$m.s^{-1}$'+' et '+ r'$\alpha=$'+'{}'.format(round(alpha*18
plt.ylim(0,1.5)
plt.plot(y,z,lw=3)
plt.savefig('I3_trajectoire.png',transparent=True)
```

2. Portée du tir.

2.1. Détermination de la portée du tir. = trouver le zéro y_p de la trajectoire. Newton ou dichotomie ?



On sait que $\forall t,\ y < y_\infty$ et $\forall t > 0\ \ y > 0$
donc la portée peut-être recherchée par dichotomie sur l'intervalle $[y_0, y_\infty]$ avec y_0 aussi petit que l'on souhaite.



La méthode de Newton risque de conduire à une sortie du domaine de définition de $z : y \to z(y)$ si u_0 est trop éloigné de y_p.
On évite donc cette méthode.

2.2.)

```
def portee(m,l,v0,alpha,e):
    """
    Calculate the impact point coordonate given the intial velocity's module v0 and
    its angle relative to the ground level.
    """
    g=9.81
    tau=m/l
    y_inf=v0*np.cos(alpha)*tau
    traj=lambda y:(g*tau**2+v0*np.sin(alpha)*tau)*(y/y_inf)-g*tau**2*np.log(y_inf/(y_inf-y))
    yp=op.bisect(traj,e,y_inf-e,xtol=e)
    return yp
```

— $z(y)$ diverge en y_{inf} donc on doit pas être évaluée en ce point.
— $z(0) = 0$ donc la condition $f(a)\,f(b) < 0$ n'est pas respectée.

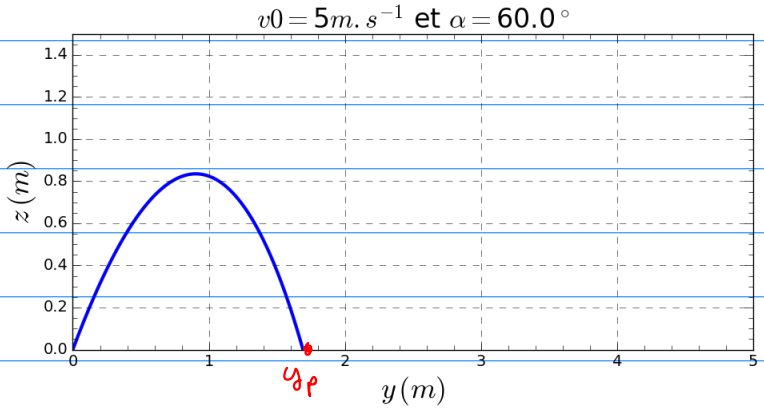
) recherche de y_p sur $[e, y_\infty - e]$

2.3) Essai de la fonction portee.

```
#application
g=9.81#m.s-2
l=0.5#N.m-1.s
m=1#kg
v0=5#m.s-1
alpha=np.pi/3
e=1e-3
y_inf=v0*np.cos(alpha)*tau
yp=portee(m,l,v0,alpha,e)
print('yp={}+/-{} m'.format(round(yp,int(-np.log10(e))),e))
```

Output : yp=1.692+/-0.001 m

Cohérent avec le graphe



3. Angle de portée maximale

3.1.
```
def maxi(l):
    n=len(l)
    m=l[0];im=0
    i=1
    while i<n:
        if l[i]>m:
            m=l[i];im=i
        i+=1
    return m,im
```

Renvoie le max m et l'indice im du max d'une liste.

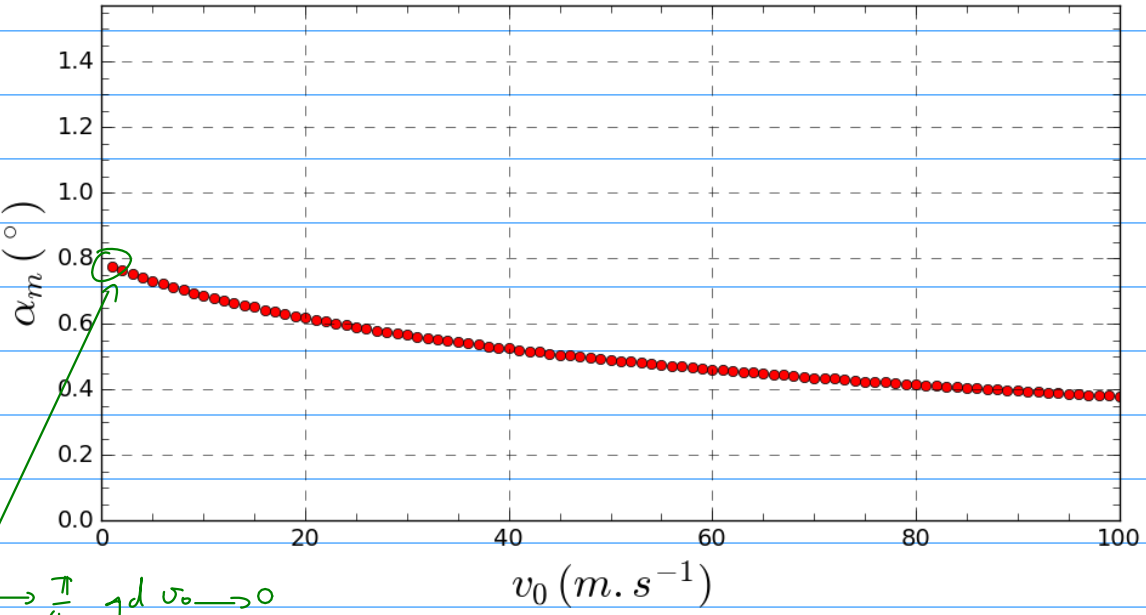
3.3)
```
def portee_max(m,l,v0,N):
    """
    Calculate the angle alpham for which, given initial velocity module,
    a projectile of mass m, submitted to air resistance of coefficient l, reach the furthest impact
    Also return the impact point coordonnate ypm.
    """

    yps=[]
    alphas=np.linspace(np.pi/(2*N),np.pi/2,N,endpoint=False)#bisect non applicable si alpha=0
    for alpha in alphas:
        yp=portee(m,l,v0,alpha,e=1e-6)
        yps.append(yp)
    ypm,index=maxi(yps);alpham=alphas[index]
    return ypm,alpham
```

4. Influence de v_0 sur l'angle α_m assurant une portée maximale du tir.

```
"""--------------------------------------------------------------------------
----------INFLUENCE DE V0 SUR L'ANGLE DE PORTEE MAXIMALE--------------------
--------------------------------------------------------------------------"""
v0s=np.linspace(1,100,100)
N=1000#number of tested angles
l_alpham=[]
for v0 in v0s:
  ypm,alpham= portee_max(m,l,v0,N)
  l_alpham.append(alpham)

plt.figure()
plt.xlabel(r'$v_0\,(m.s^{-1})$')
plt.ylabel(r'$\alpha_m\,(^\circ)$')
plt.ylim(0,np.pi/2)
plt.plot(v0s,l_alpham,'ro')
plt.savefig('I3_portee_max=f(v0).png',transparent=True)
```

On calcule l'angle de portée maximale pour différentes valeurs de v_0.

Output :



$\alpha_m \to \frac{\pi}{4}$ qd $v_0 \to 0$

cohérent avec l'influence des frottements $\propto$ à la vitesse.

D'après la simulation, pour une force de frottement linéaire, plus v_0 est élevée plus l'angle de tir donnant une portée maxi est faible.

Rem : pour des vitesses trop élevées, le modèle $\vec{f} = -\lambda \vec{v}$ est à revoir.