Capacités numériques

On relève dans le programme de terminale spécialité

Mesure et incertitudes

Capacité numérique : Représenter l'histogramme associé à une série de mesures à l'aide d'un tableur ou d'un langage de programmation.

Capacité numérique : Simuler, à l'aide d'un langage de programmation, un processus aléatoire illustrant la détermination de la valeur d'une grandeur avec incertitudes-types composées.

Capacité numériques

On relève dans le programme de terminale spécialité

Constitution et transformations de la matière

Capacité numérique : Représenter, à l'aide d'un langage de programmation, l'évolution des quantités de matière des espèces en fonction du volume de solution titrante versé.

Capacité numérique : Déterminer, à l'aide d'un langage de programmation, le taux d'avancement final d'une transformation, modélisée par la réaction d'un acide sur l'eau.

Capacité numérique : Tracer, à l'aide d'un langage de programmation, le diagramme de distribution des espèces d'un couple acide-base de pK_A donné.

Capacité numérique : À l'aide d'un langage de programmation et à partir de données expérimentales, tracer l'évolution temporelle d'une concentration, d'une vitesse volumique d'apparition ou de disparition et tester une relation donnée entre la vitesse volumique de disparition et la concentration d'un réactif.

Capacités numériques

On relève dans le programme de terminale spécialité

Mouvement et interactions

Capacité numérique : Représenter, à l'aide d'un langage de programmation, des vecteurs accélération d'un point lors d'un mouvement.

Capacité numérique : Représenter, à partir de données expérimentales variées, l'évolution des grandeurs énergétiques d'un système en mouvement dans un champ uniforme à l'aide d'un langage de programmation ou d'un tableur.

Capacité numérique : Exploiter, à l'aide d'un langage de programmation, des données astronomiques ou satellitaires pour tester les deuxième et troisième lois de Kepler.

Capacités numériques

On relève dans le programme de terminale spécialité

Ondes et signaux

Capacité numérique : Représenter, à l'aide d'un langage de programmation, la somme de deux signaux sinusoïdaux périodiques synchrones en faisant varier la phase à l'origine de l'un des deux.

Capacités numériques

On trouve aussi, sans l'étiquette « capacité numérique »

Établir et résoudre l'équation différentielle vérifiée par la tension aux bornes d'un condensateur dans le cas de sa charge par une source idéale de tension et dans le cas de sa décharge.

Illustrer qualitativement, par exemple à l'aide d'un microcontrôleur, d'un multimètre ou d'une carte d'acquisition, l'effet de la géométrie d'un condensateur sur la valeur de sa capacité.

Déterminer le temps caractéristique d'un dipôle RC à l'aide d'un microcontrôleur, d'une carte d'acquisition ou d'un oscilloscope.

Notions abordées avec Python ? Arduino ?

Capacités numériques

On trouve aussi, sans l'étiquette « capacité numérique »

Capacités mathématiques : Résoudre une équation différentielle, déterminer la primitive d'une fonction, utiliser la représentation paramétrique d'une courbe.

Capacité mathématique : Résoudre une équation différentielle linéaire du premier ordre à coefficients constants.

Capacité mathématique : Résoudre une équation différentielle linéaire du premier ordre à coefficients constants avec un second membre constant.

Capacité mathématique : Résoudre une équation du second degré.

Capacité mathématique : Résoudre une équation différentielle linéaire du premier ordre à coefficients constants avec un second membre constant.

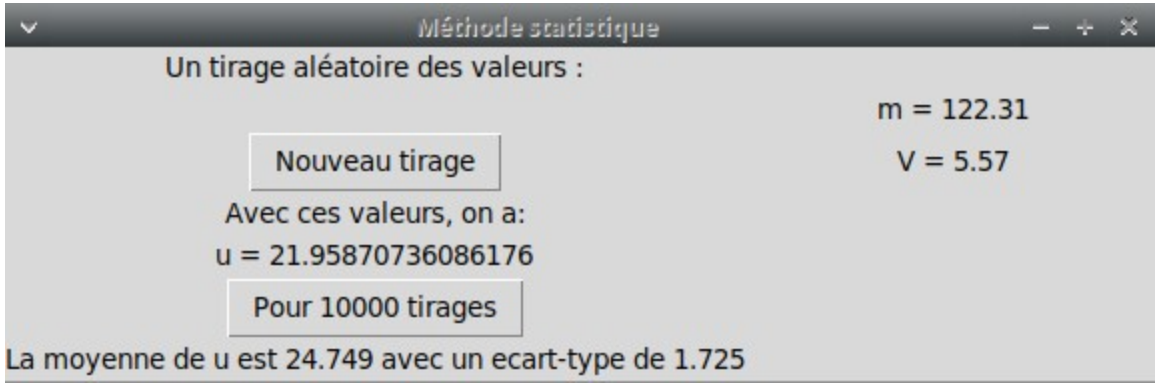
Traitement numérique des ces capacités ?
(certains élèves sans spé math depuis la première)

Capacités numériques

Mesure et Incertitudes

Capacités numériques

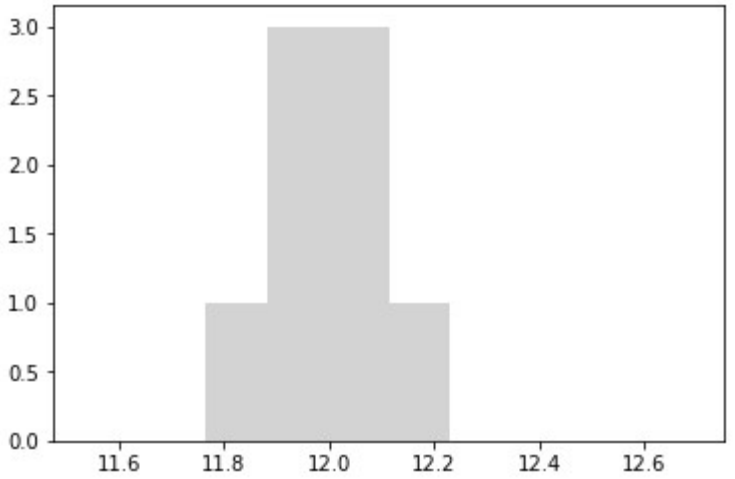
Lien vers la vidéo



Capacité numérique : Simuler, à l'aide d'un langage de programmation, un processus aléatoire illustrant la détermination de la valeur d'une grandeur avec incertitudes-types composées.

Capacités numériques

Lien vers la vidéo



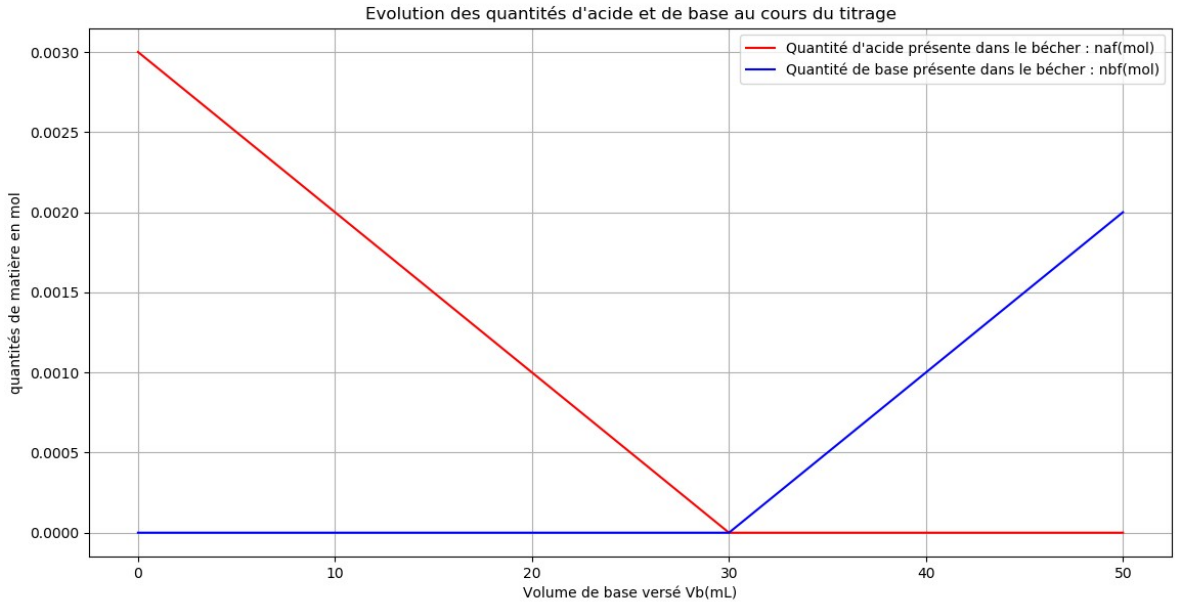
Capacité numérique : Représenter l'histogramme associé à une série de mesures à l'aide d'un tableur ou d'un langage de programmation.

Capacités numériques

Partie Chimie

capacités numériques en chimie

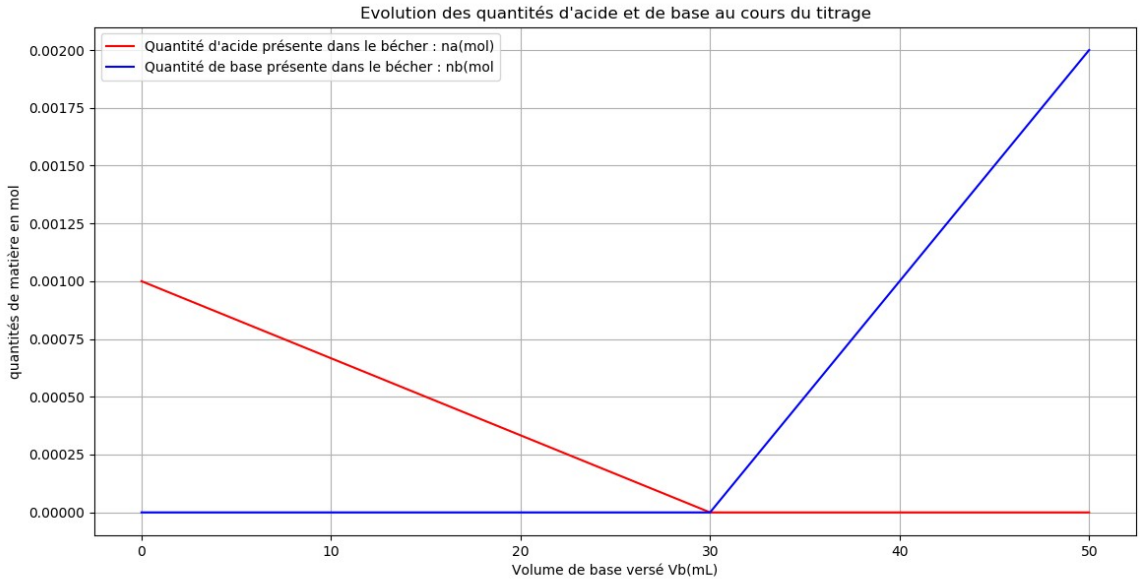
- Partie : Analyser un système par des méthodes chimiques : représenter à l'aide d'un langage de programmation, l'évolution des quantités de matière des espèces en fonction du volume de solution titrante versé.
- Équation support du titrage A + B
- Saisie des concentrations des espèces titrée et titrante, du volume de la solution titrée, du volume de la burette graduée et du nombre de points pour tracer les courbes.



```
# titrage de acide méthanoïque  par l'hydroxyde de sodium
from matplotlib import pyplot as plt
Ca = float(input(" Entrer la valeur Ca de la concentration de l'acide à titrer en mol/L : Ca = " ))
Va = float(input("Entrer le volume Va d'acide titré en mL : Va = "))
V = float(input("Entrer le volume de la burette graduée en mL : V = "))
Cb = float(input("Entrer la valeur de la concentration Cb de la base contenue dans la burette graduée en mol/L : Cb = "))
N = int(input("Nombre N de points de chaque courbe"))
Vb = []
naf = []
nbf = []
Vbe = Ca*Va/Cb
print ('Vbe',Vbe)
for i in range (0, N+1):
    Vb.append(V*i/N)
    if Vb[i]< Vbe :
        nbf.append(0)
        naf.append(Ca*Va/1000-Cb*Vb[i]/1000)
    else :
        nbf.append(Cb*Vb[i]/1000-Ca*Va/1000)
        naf.append(0)
plt.title("Evolution des quantités d'acide et de base au cours du titrage")
plt.xlabel("Volume de base versé Vb(mL)")
plt.ylabel("quantités de matière en mol")
plt.grid(True)
plt.plot(Vb,naf,c='red',label="Quantité d'acide présente dans le bécher : naf(mol)")
plt.plot(Vb,nbf,c='blue',label="Quantité de base présente dans le bécher : nbf(mol)")
plt.legend()
plt.show()
```

capacités numériques en chimie

- Idée : comment transformer le premier programme pour suivre l'évolution des quantités de matières d'un titrage A + 3B (acide citrique par la soude par exemple) ?



```
# programme titrage A + 3B de l'acide citrique par la soude
from matplotlib import pyplot as plt
Ca= float(input("Entrer la valeur Ca de la concentration de l'acide à titrer en mol/L : Ca = "))
Va= float(input("Entrer le volume Va d'acide titré en mL : Va = "))
V= float(input("Entrer le volume de la burette graduée en mL : V = "))
Cb= float(input("Entrer la valeur de la concentration Cb de la base contenue dans la burette graduée en mol/L : Cb = "))
N=int(input("Nombre N de points de la courbe"))
Vb=[]
naf=[]
nbf=[]
Vbe=3*Ca*Va/Cb
for i in range (0,N+1):
  Vb.append(V*i/N)
  if Vb[i]< Vbe :
     nbf.append(0)
     naf.append(Ca*Va/1000-Cb*Vb[i]/(3*1000))
  else :
     nbf.append(Cb*Vb[i]/1000-3*Ca*Va/1000)
     naf.append(0)
plt.title("Evolution des quantités d'acide et de base au cours du titrage")
plt.xlabel("Volume de base versé Vb(mL)")
plt.ylabel("quantités de matière en mol")
plt.grid(True)
plt.plot(Vb,naf,c='red',label="Quantité d'acide présente dans le bécher : na(mol)")
plt.plot(Vb,nbf,c='blue',label="Quantité de base présente dans le bécher : nb(mol")
plt.legend()
plt.show()
```

capacités numériques en chimie

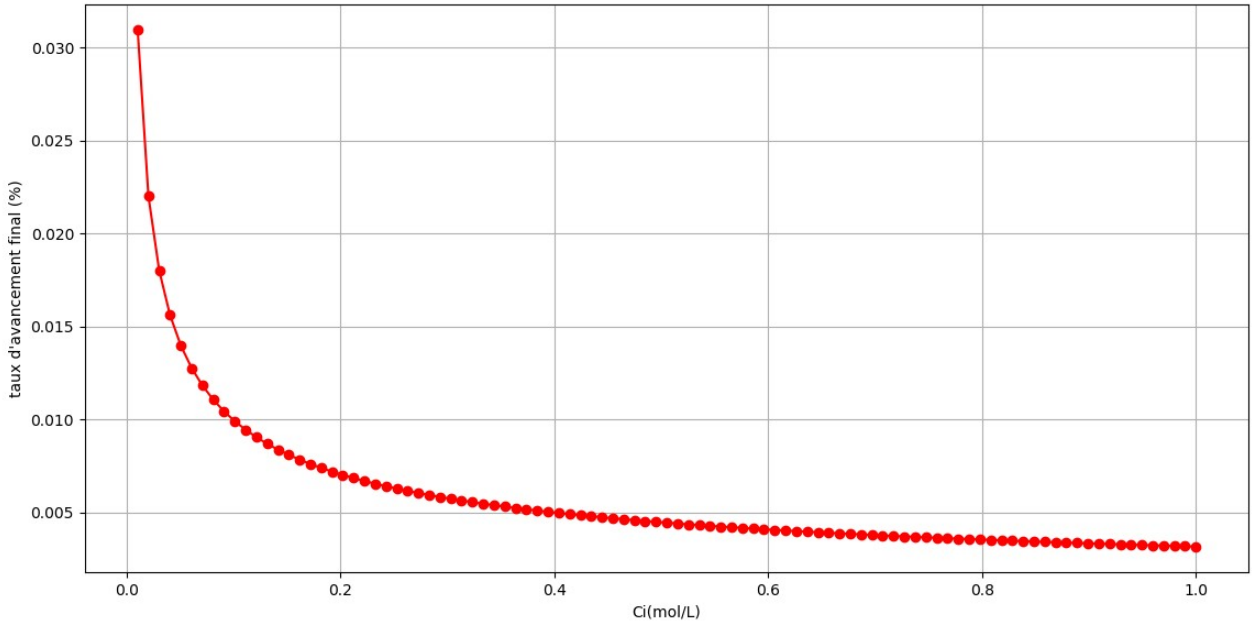
- Partie : Comparer la force des acides et des bases.
- Déterminer à l'aide d'un langage de programmation, le taux d'avancement final d'une transformation, modélisée par la réaction d'un acide dans l'eau.
- Saisie de la quantité initiale d'acide, du volume de la solution et du pH.

```
#Programme : détermination d'un taux d'avancement à partir de la mesure du pH
# AH + H2O = A- + H3O+

ni=float(input("Indiquer la quantité ni (en mol) d'acide introduite dans l'eau :"))
V=float(input("Indiquer le volume V de la solution obtenue en litre:"))
pH=float(input("Indiquer le pH de la solution obtenue :"))

h = 10**(-pH) # h est la concentration en ions oxonium
xmax=ni # calcul de l'avancement maximal
xf=V*h # calcul de l'avancement à l'équilibre
taux=xf/xmax # calcul du taux d'avancement final
print("La taux d'avancement final =", round(taux,4))# taux arrondi à 4 chiffres après 0
# affichage de la valeur du taux d'avancement final

if taux > 0.99 :
    print("L'acide est fort")
else :
    print("L'acide est faible")
# affichage de la force de l'acide
```

capacités numériques en chimie

- Partie : Comparer la force des acides et des bases.
- Déterminer à l'aide d'un langage de programmation, le taux d'avancement final d'une transformation, modélisée par la réaction d'un acide dans l'eau.
- Idée : influence de la concentration initiale sur le taux d'avancement final à partir de la saisie du pKa.
- Résolution de l'équation du second degré obtenue à partir de la relation liant K_A et τ_f :

  $C_i \times \tau_f^2 + K_A \times \tau_f - K_A = 0$

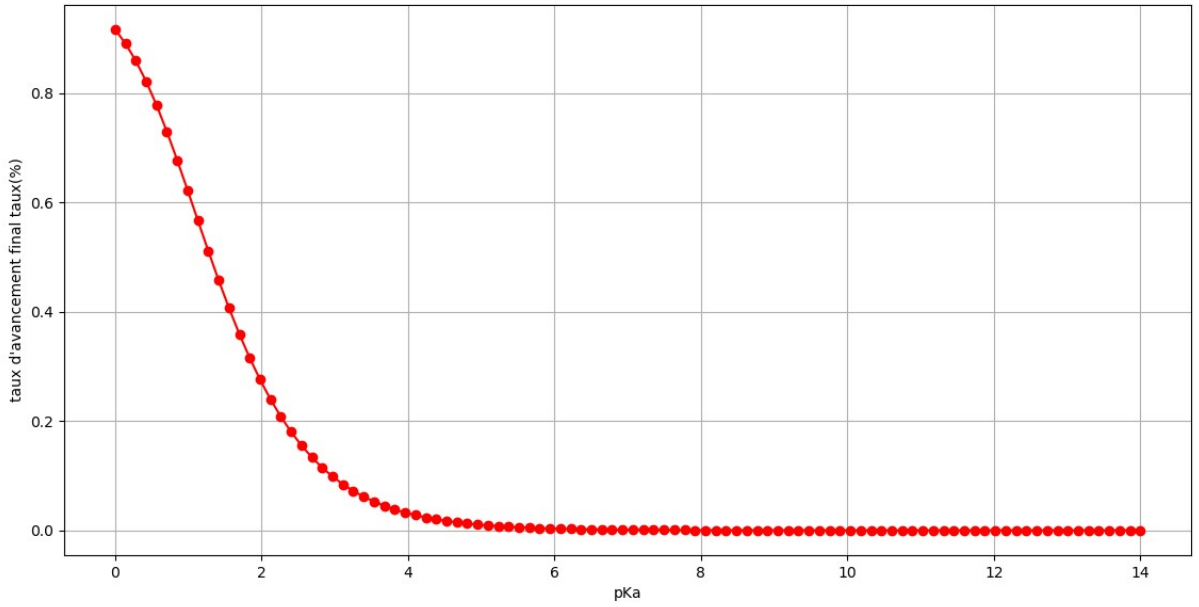


```
# Influence de la concentration Ci en quantité de soluté sur le taux d'avancement final

from matplotlib import pyplot as plt
import numpy as np
pKa=float(input("Indiquer la valeur du pKa du couple acide-base : pKa = "))
Ka = 10**(-pKa)
Ci=np.linspace(0,1,100)# fait varier Ci de 0 à 1 mol/L avec 100 valeurs
delta=Ka**2+4*Ka*Ci
tauxfinal = (-Ka + delta**0.5)/(2*Ci)
plt.xlabel("Ci(mol/L)")
plt.ylabel("taux d'avancement final (%)")
plt.plot(Ci,tauxfinal,c="red",marker="o")
plt.grid()
plt.show()
```

capacités numériques en chimie

- Partie : Comparer la force des acides et des bases.
- Déterminer à l'aide d'un langage de programmation, le taux d'avancement final d'une transformation, modélisée par la réaction d'un acide dans l'eau.
- Idée : influence du pKa sur le taux d'avancement final pour une concentration initiale constante à parir de la saisie de Ci.
- Résolution de l'équation du second degré obtenue à partir de la relation liant K_A et τ_f

$$C_i \times \tau_f^2 + K_A \times \tau_f - K_A = 0$$



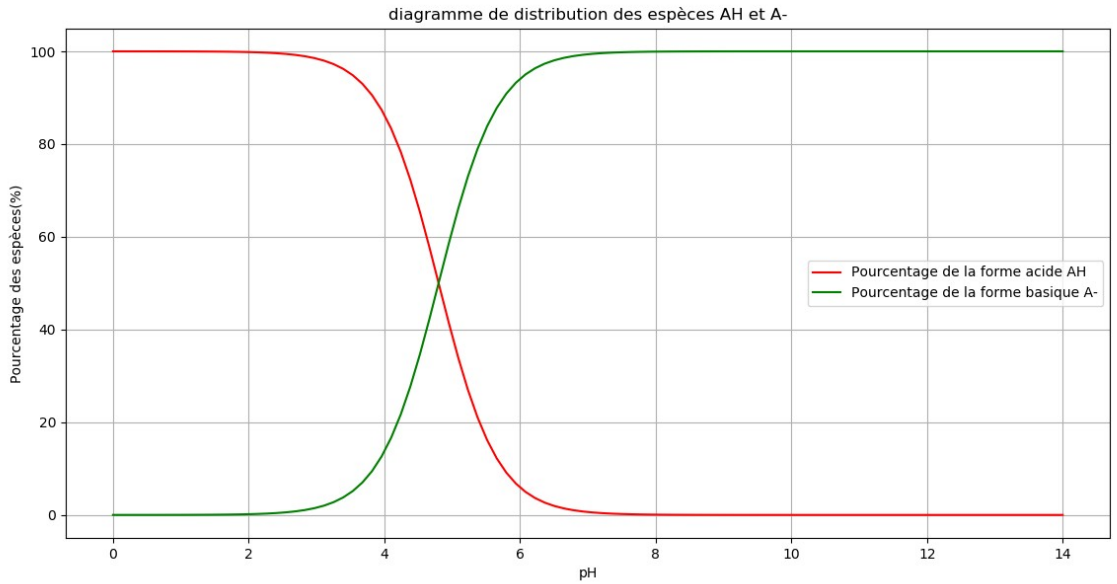
```
# Influence du pKA sur le taux d'avancement final à concentration Ci en acide égale
from matplotlib import pyplot as plt
import numpy as np

Ci=float(input("Indiquer la valeur de la concentration de l'acide en mol/L : Ci = "))

pKa=np.linspace(0,14,100)
Ka= 10**(-pKa)
Delta=Ka**2+4*Ka*Ci
tauxfinal=(-Ka + Delta**0.5)/(2*Ci)

plt.xlabel("pKa")
plt.ylabel("taux d'avancement final taux(%)")
plt.plot(pKa,tauxfinal,c="red",marker="o")
plt.grid()

plt.show()
```

capacités numériques en chimie

- Partie : Comparer la force des acides et des bases.
- Tracer, à l'aide d'un langage de programmation, le diagramme de distribution des espèces d'un couple acide-base de pKa donné.
- Saisie du pKa et de la concentration initiale Ci.
- Utilisation des formules :

$pH = pK_A + \log(CB/CA)$;

$Ci = CA + CB$; pourcentage des espèces



```
# Tracé du diagramme de distribution des espèces conjuguées A/B

import numpy as np
from matplotlib import pyplot as plt

pKa= float(input("Entrer la valeur du pKa du couple acide-base : pKa = "))
Ci= float(input("Entrer la valeur de la concentration en quantité de matière de l'acide en mol/L: Ci
="))

pH=np.linspace(0,14,100)# 100 valeurs allant de 0 à 14
CA=Ci/(1+10**(pH-pKa))
CB=Ci-CA
pA=CA/Ci*100
pB=CB/Ci*100
plt.title("diagramme de distribution des espèces AH et A-")
plt.xlabel("pH")
plt.ylabel('Pourcentage des espèces(%)')
plt.grid(True)
plt.plot(pH,pA,c='red',label="Pourcentage de la forme acide AH")
plt.plot(pH,pB,c='green',label="Pourcentage de la forme basique A-")
plt.legend()
plt.show()
```

Capacités numériques

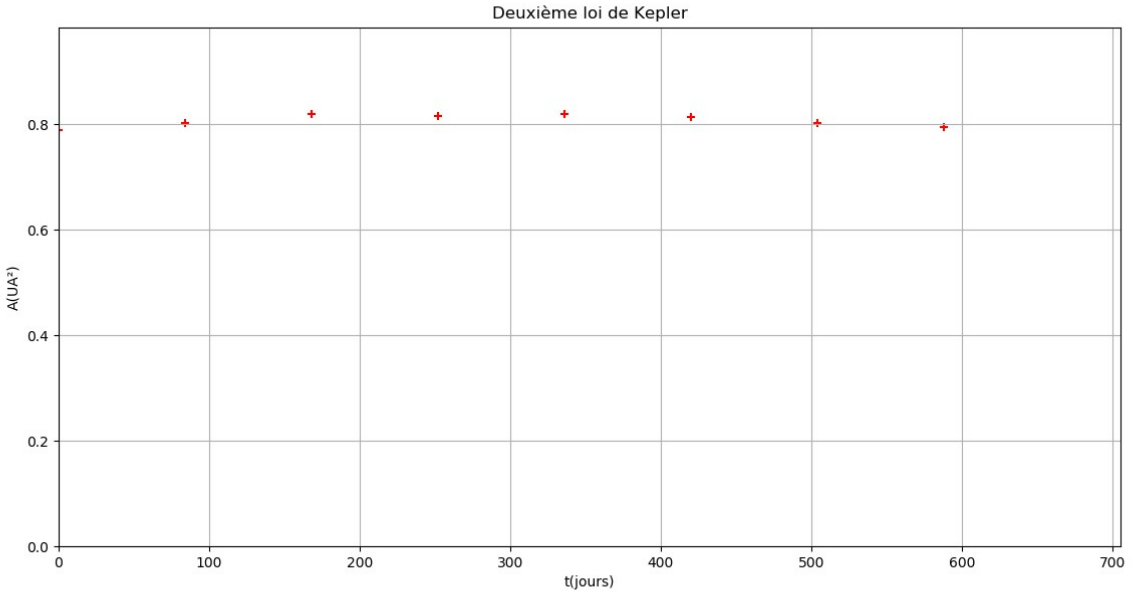
Partie Physique

capacités numériques en physique

- Partie : Relier les actions appliquées à un système à son mouvement
- Exploiter, à l'aide d'un langage de programmation, des données astronomiques ou satellitaires pour tester les deuxièmes et troisièmes lois de Kepler.
- Saisie des coordonnées polaires
- Calcul d'aire d'un triangle

$A(i) = 0.5 * (r(i+1) \times r(i) \times \sin(\theta(i+1) - \theta(i))$

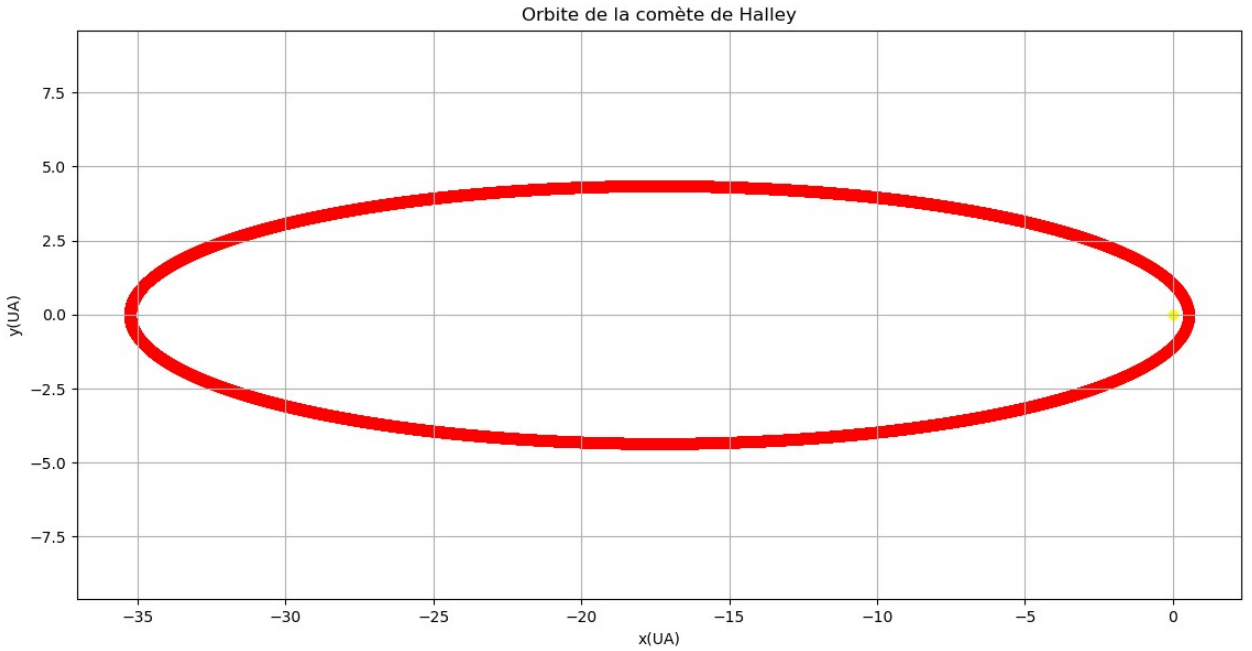
Source : M. Brumerelle Lunéville



```
# Programme permettant de vérifier la 2ème loi de Kepler à partir de données expérimentales :
import numpy as np
import matplotlib.pyplot as plt
from math import *
thêta=[0,53,100,141,178,215,255,301,354] # cette liste est à compléter : il s'agit de l'angle en degré
r=[1.381,1.430,1.535,1.629,1.666,1.636,1.548,1.441,1.382] # cette liste est à compléter : il s'agit du rayon en unité astronomique
radian=np.radians(thêta)
jours=84 # un nombre entier est attendu : c'est le nombre de jours écoulés entre deux relevés de la position de Mars
A=[] # cette liste permet de stocker les calculs d'aire balayée
t=[] # cette liste permet de stocker les dates où les mesures ont été réalisées
for i in range (len(thêta)-1) :
    A.append((r[i+1]*r[i]*sin((radian[i+1]-radian[i])))/2) # à vous de compléter !
    t.append(jours*i)
plt.title("Deuxième loi de Kepler")
plt.scatter(t,A,c="red",marker="+")
plt.xlabel("t(jours)")                    # à vous d'indiquer grandeur et unité sur cet axe
plt.ylabel("A(UA²)")                      # à vous d'indiquer grandeur et unité sur cet axe
plt.axis([0,1.2*max(t),0,1.2*max(A)])
plt.grid(True)
plt.show()
```

capacités numériques en physique

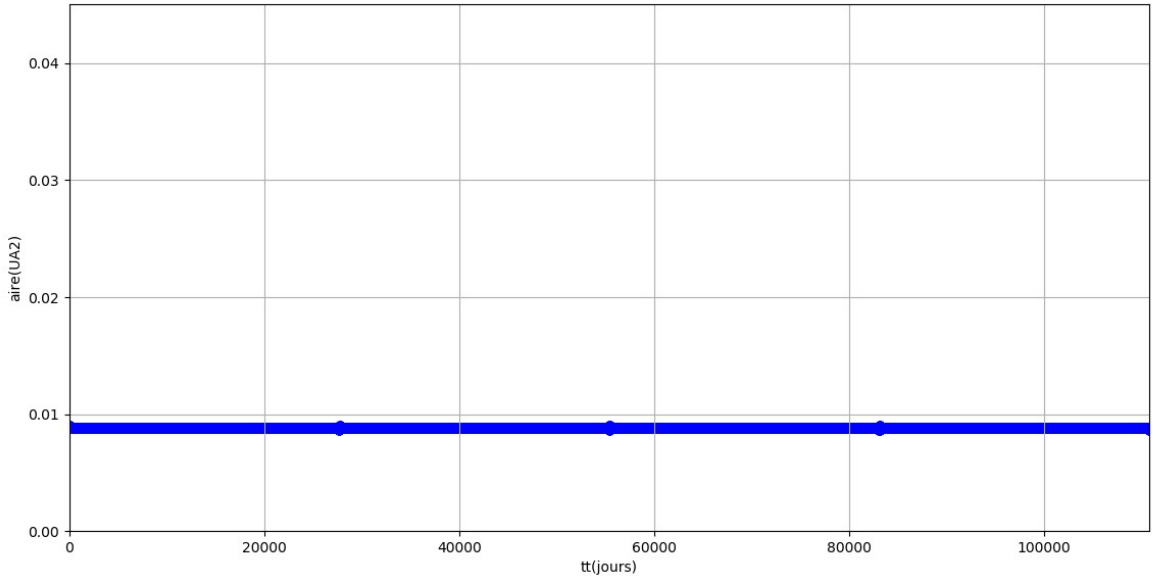
- Exploiter, à l'aide d'un langage de programmation, des données astronomiques ou satellitaires pour tester les deuxièmes et troisièmes lois de Kepler.
- Programme professeur permettant de générer les coordonnées polaires avec la méthode d'Euler saisie de l'excentricité, de la période, du demi grand axe et du pas de l'itération.
- $r(i) = a \times (1 - e^2)/(1+ e \times \cos(\theta(i))$
- $d\theta/dt = 2 \times \pi \times a^2 (1 - e)^{1/2} / (T \times r(i)^2)$
- Exemple comète de Halley



```
# Générer la trajectoire en coordonnées polaires Loi Kepler 1
from math import *
from matplotlib import pyplot as plt
dt=float(input("Entrer la valeur du pas de l'itération en jour : dt ="))
e=float(input("Entrer la valeur de l'excentricité e ="))
a=float(input("Entrer le demi grand axe de l'orbite en UA : a="))
T=int(input("Een jour Entrer la pétiode orbitale en jour : T="))
r0=a*(1-e)
t=[0]
angle=[0]
r=[r0]
x=[r0]
y=[0]
for i in range (0,4*T):
    t.append(i*dt)
    r.append(a*(1-e**2)/(1+e*cos(angle[i])))
    dangle=(2*pi*a*a*(1-e**2)**0.5)/(T*r[i]**2)*dt
    angle.append(angle[i]+dangle)
    x.append(r[i]*cos(angle[i]))
    y.append(r[i]*sin(angle[i]))
print("angle",angle)
print("r",r)
print("x",x)
print("y",y)
plt.figure(1)
plt.scatter(x,y,c="red",marker="o")
plt.scatter(0,0,c="yellow",marker="o")
plt.xlabel("x(UA)")
plt.ylabel("y(UA)")
plt.axis("equal")
plt.grid(True)
plt.show()
```

Calcul de l'aire balayée en 1 jour, on ajoute les instructions ci-dessous au programme précédent :

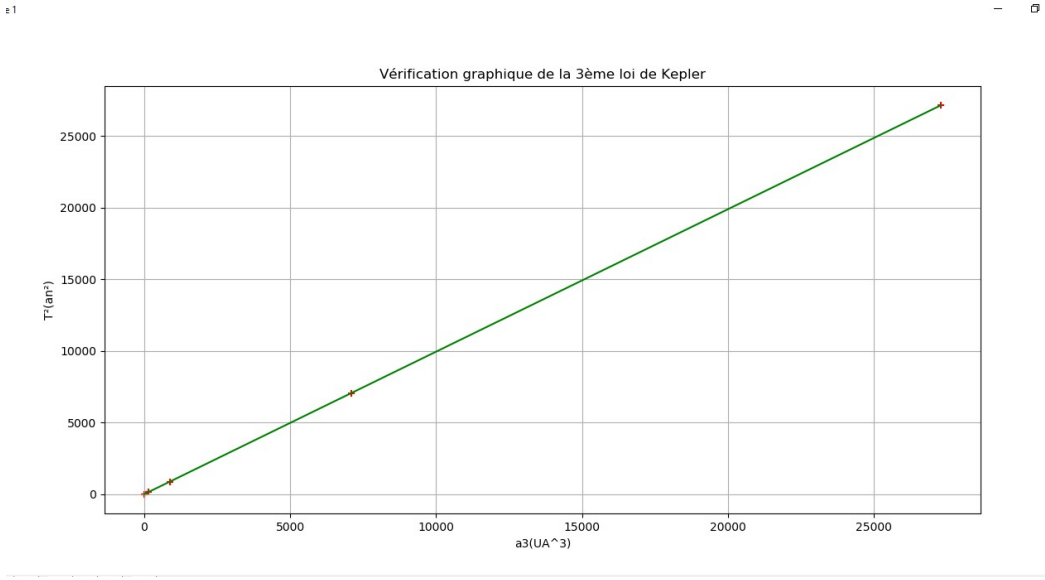
```
aire=[]
tt=[]
for i in range(0,4*T):
    aire.append((r[i+1]*r[i]*sin(angle[i+1]-angle[i])/2))
    tt.append(t[i])

plt.figure(2)
plt.scatter(tt,aire,c="blue",marker="o")
plt.xlabel("tt(jours)")
plt.ylabel("aire(UA2)")
plt.axis([0,4*T,0,5*max(aire)])
plt.grid(True)
```



capacités numériques en physique

- Partie : Relier les actions appliquées à un système à son mouvement
- Exploiter, à l'aide d'un langage de programmation, des données astronomiques ou satellitaires pour tester les deuxièmes et troisièmes lois de Kepler.
- Saisie des périodes et demi-grands axes des planètes + régression linéaire

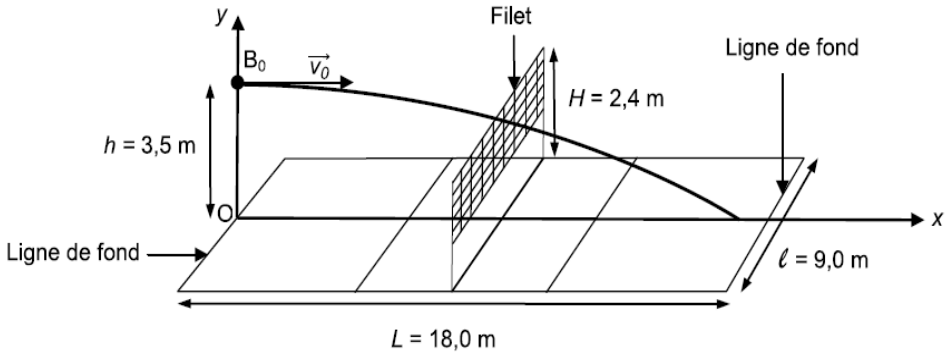
Source : M. Brumerelle Lunéville



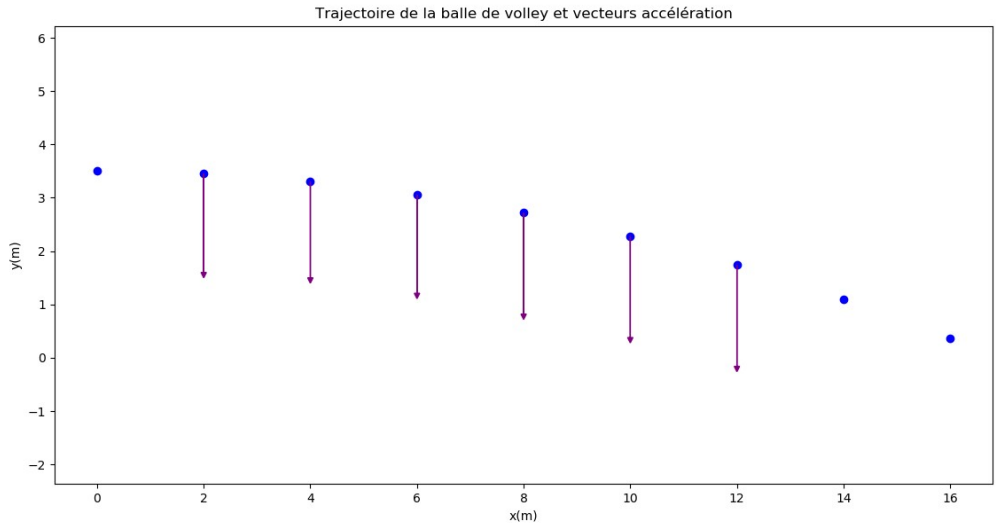
Le lien vers la vidéo explicative

capacités numériques en physique

- Partie : Décrire un mouvement
- Représenter, à l'aide d'un langage de programmation, des vecteurs accélération d'un point lors d'un mouvement

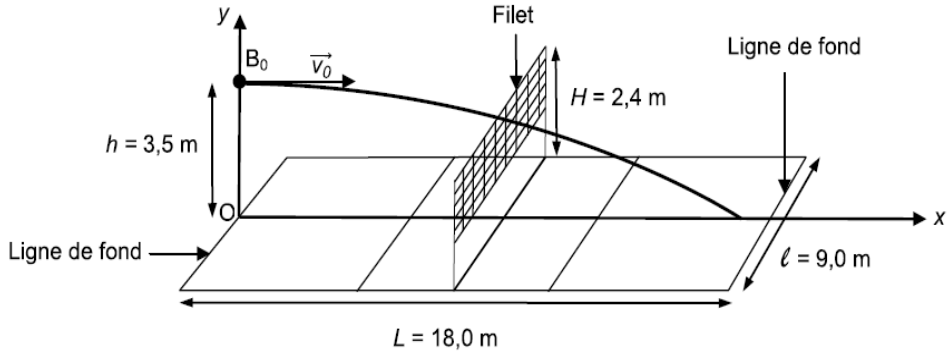


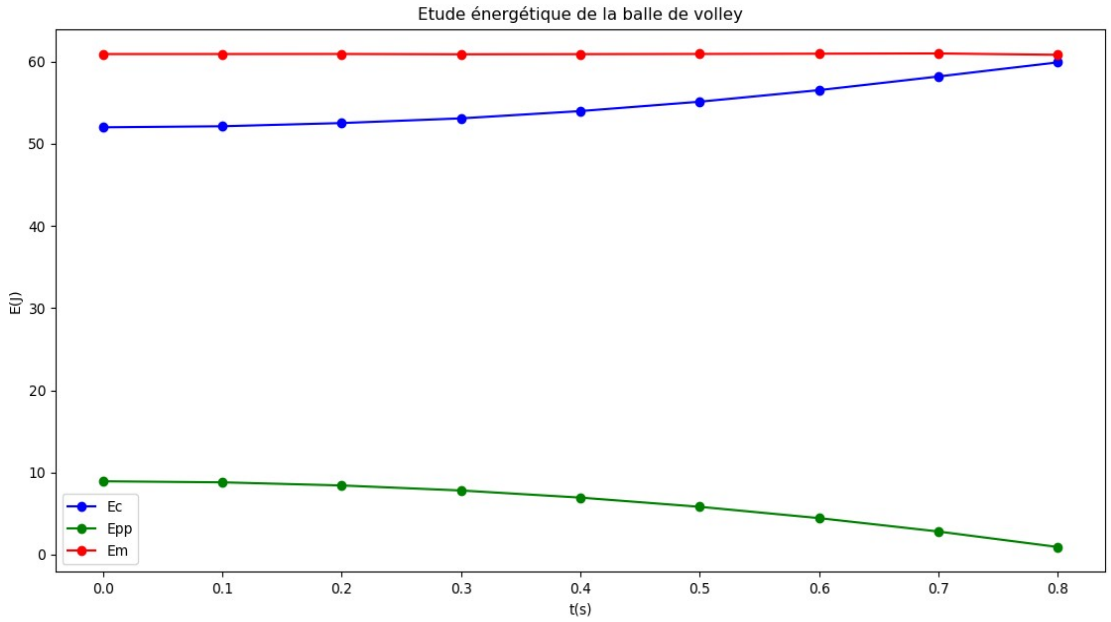
t(s)	Vx(m/s)	Vy(m/s)	V(m/s)	x(m)	y(m)
0,0	20	*0,0*	20,0	0,0	3,50
0,1	20	*-1,0*	20,0	2,0	3,45
0,2	20	*-2,0*	20,1	4,0	3,30
0,3	20	*-2,9*	20,2	6,0	3,06
0,4	20	*-3,9*	20,4	8,0	2,72
0,5	20	*-4,9*	20,6	10,0	2,28
0,6	20	*-5,9*	20,8	12,0	1,74
					,10
					,36



capacités numériques en physique

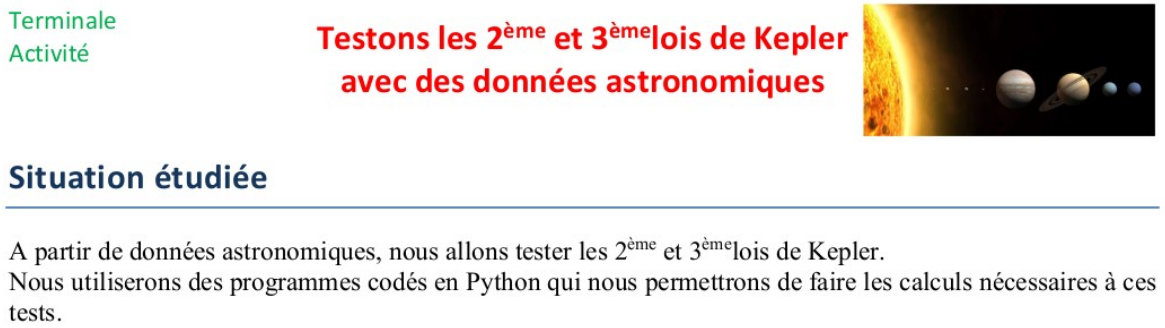
- Partie : Relier les actions appliquées à un système à son mouvement
- Représenter, à l'aide d'un langage de programmation ou d'un tableur, l'évolution des grandeurs énergétiques d'un système en mouvement dans un champ uniforme.





capacités numériques en physique

- **2 activités « clé en mains »**
- Lois de Kepler (Eric Garnier)
- Le lien vers l'activité
- Trajectoire d'une balle (une même expérience, traitée de la seconde à la terminale) (Laurent Meyer)
- Le lien vers l'activité



Terminale
Activité

Testons les 2ème et 3èmelois de Kepler avec des données astronomiques

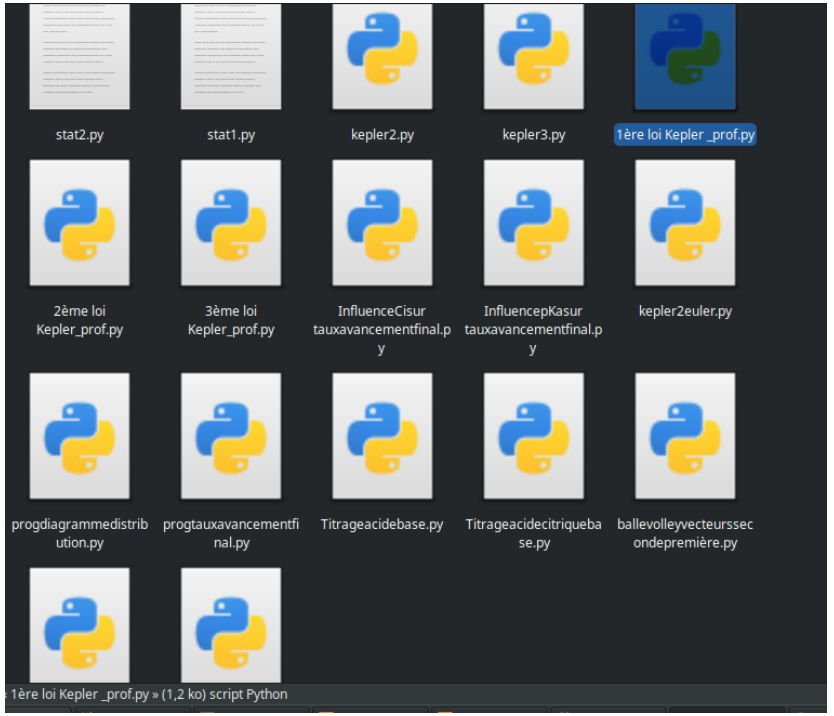
Situation étudiée

A partir de données astronomiques, nous allons tester les 2ème et 3èmelois de Kepler.
Nous utiliserons des programmes codés en Python qui nous permettrons de faire les calculs nécessaires à ces tests.

capacités numériques en physique

- **Les programmes python**
- Dans un dossier zippé, disponible à cette adresse.

- Une formation PAF reconduite « Programmation Python pour les Sciences Physiques » (2 niveaux?)



capacités numériques en physique

Nous contacter

Eric Garnier	Eric.Garnier@ac-nancy-metz.fr
Étienne Hilger	Etienne.Hilger@ac-nancy-metz.fr
Karine Ladmiral	Karine-Anne.Ladmiral@ac-nancy-metz.fr
Didier Lommelé	Didier.Lommele@ac-nancy-metz.fr
Laurent Meyer	Laurent.Meyer@ac-nancy-metz.fr