éduSCOL
Informer et accompagner les professionnels de l'éducation

Liberté • Égalité • Fraternité
RÉPUBLIQUE FRANÇAISE

MINISTÈRE DE L'ÉDUCATION NATIONALE ET DE LA JEUNESSE

VOIE GÉNÉRALE

2DE | 1RE | TLE

Physique-chimie

ENSEIGNEMENT SPÉCIALITÉ

COMBIEN DE BONBONS GÉLIFIÉS BLEUS PEUT-ON MANGER PAR JOUR ? TESTER LA LOI DE BEER-LAMBERT

Cette ressource s'inscrit dans la continuité de celle de la classe de première intitulée. L'objectif est de simuler des processus aléatoires qui permettent la détermination de la valeur de différentes grandeurs avec incertitudes-types composées.

Références à la partie « Mesures et incertitudes » du programme

Notions et contenus	Capacités exigibles
Incertitude-type Incertitudes-types composées	Procéder à l'évaluation d'une incertitude-type par une autre approche que statistique (évaluation de type B). 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.

Éléments pour construire l'activité des élèves

On dispose d'une solution de bleu patenté V, notée S_0, préparée selon le protocole suivant :

On obtient tout d'abord une solution mère S_m par dissolution de m = 297 mg de bleu patenté V (M = 582,66 $g.mol^{-1}$) dans une fiole jaugée de V_{f1} = 1,0000 L. On prélève, avec une pipette jaugée, V_p = 10,00 mL de solution mère S_m, que l'on dilue dans une fiole jaugée de V_{f2} = 250,0 mL. On obtient ainsi la solution S_0.

1. Calculer la concentration en quantité de matière de la solution mère S_m et de la solution S_0.

En utilisant deux burettes graduées, introduire dans différents tubes à essai un volume V_1 mL de solution S_0 et un volume $V_2 = 10 - V_1$ mL d'eau distillée. Mesurer l'absorbance des différentes solutions ainsi préparées.

2. Remplir le tableau suivant :

Tube	1	2	3	4	5
V_1 (mL)	10,00				
$V_2 = 10 - V_1$ (mL)	0				
C ($mol.L^{-1}$)					
A_{640nm}					

3. Modéliser les données expérimentales en utilisant la loi de Beer-Lambert.

Dissoudre un bonbon gélifié bleu dans de l'eau distillée. Ajuster le volume de solution en utilisant une fiole jaugée de V_{f3} = 50,00 mL.

4. Déterminer la concentration en quantité de matière C_S de la solution de bonbon gélifié bleu en effectuant un report de point.

La DJA (Dose Journalière Admissible) de l'additif alimentaire E131 est 2,5 mg par kg de masse corporelle et par jour.

5. Combien de bonbons gélifiés bleus peut-on manger par jour ?

Éléments de correction pour le professeur

1. Solution mère :

$$C_m = \frac{m}{MV_{f1}} = \frac{297.10^{-3}}{582{,}66 \times 1{,}0000} = 5{,}097312.10^{-4}\ mol.L^{-1}$$

Solution S_0 :

$$C_0 = \frac{C_m V_p}{V_{f2}} = \frac{5{,}097312.10^{-4} \times 10{,}00.10^{-3}}{250{,}0.10^{-3}} = 2{,}0389249.10^{-5}\ mol.L^{-1}$$

On cherche à évaluer l'incertitude-type de C_m et C_0. Cependant, on ne peut pas composer des incertitudes en classe de première. La méthode de Monte-Carlo permet d'étudier la variabilité de C_m et de C_0 sans avoir besoin d'utiliser des formules de composition d'incertitudes.

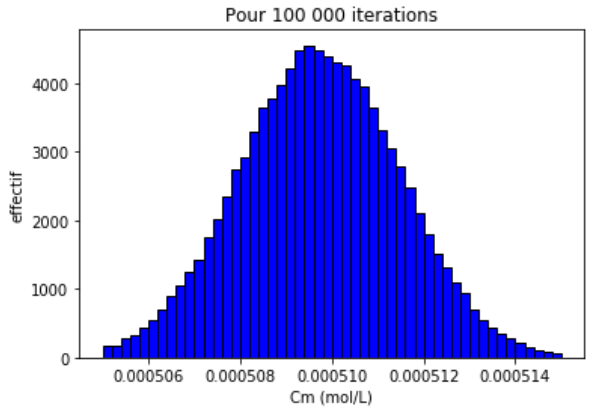
Cette variabilité est expliquée par différentes incertitudes qui s'accumulent tout au long du protocole : incertitudes de la pesée, de la masse molaire et de la fiole jaugée pour la dissolution ; incertitudes de la pipette jaugée et de la deuxième fiole jaugée pour la dilution. On prendra (évaluation des incertitudes-type par une autre approche que statistique - type B) :

- pesée : d'après la notice de la balance, on prendra u(m) = 1 mg ;
- masse molaire : u(M) = 0,01 $g.mol^{-1}$ (dernier chiffre significatif, incertitude négligeable) ;
- fiole jaugée : $u(V_{f1})$ = 0,0008 L (à lire sur la fiole jaugée) ;
- pipette jaugée : $u(V_p)$ = 0,02 mL (à lire sur la pipette jaugée) ;
- fiole jaugée : $u(V_{f2})$ = 0,3 mL (à lire sur la fiole jaugée).

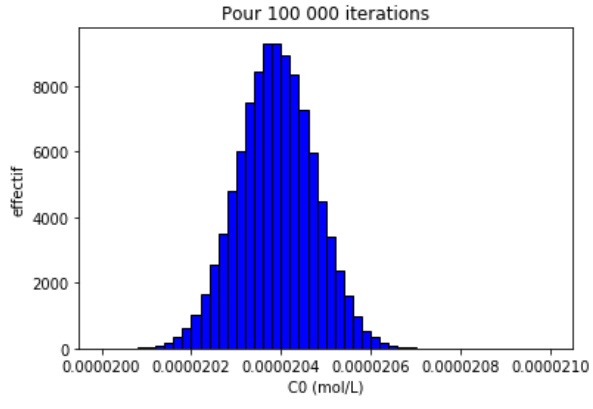
Un jeu de données (m, M, V_{f1}, V_p, V_{f2}) est tiré au sort (tirage avec écarts-types connus, loi normale) pour calculer C_m et C_0. La procédure est répétée 100 000 fois. On calcule les moyennes et les écarts-types $u(C_m)$, $u(C_0)$.

Un essai (programme Python en annexe 1) permet d'obtenir *(affichage brut)* :

```
Calcul de Cm : 0.0005097312326227989
Moyenne des Cm : 0.0005097204168597346
u(Cm) : 1.764140223368491e-06
```




```
Calcul de C0 : 2.0389249304911954e-05
Moyenne des C0 : 2.038869200493872e-05
u(C0) : 8.501743762212844e-08
```



$C_m = 5{,}10.10^{-4}\ mol.L^{-1}$ $\quad u(C_m) = 0{,}02.10^{-4}\ mol.L^{-1}$

$C_0 = 2{,}039.10^{-5}\ mol.L^{-1}$ $\quad u(C_0) = 0{,}009.10^{-5}\ mol.L^{-1}$

2. Deux élèves (groupe 1) ont choisi :

Tube	1	2	3	4	5
V_1 (mL)	10,00	7,50	5,00	2,50	1,00
$V_2 = 10 - V_1$ (en mL)	0	2,50	5,00	7,50	9,00

On cherche à évaluer l'incertitude-type de C. Sa variabilité est expliquée par toutes les incertitudes précédentes ainsi que par l'incertitude des deux burettes graduées (évaluation de l'incertitude-type par une autre approche que statistique – type B) :

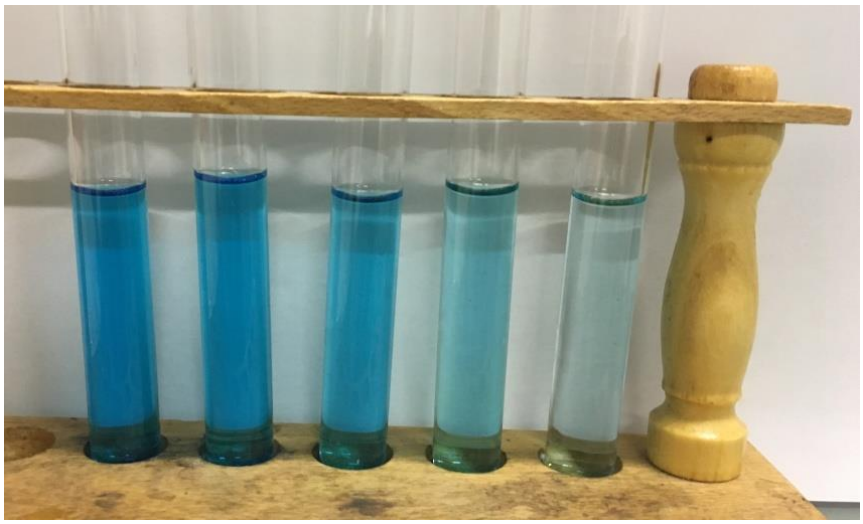
- burette graduée : $u(V_1) = u(V_2) = 0,05$ mL (à lire sur la burette graduée)

Un jeu de données (C_0, V_1, V_2) est tiré au sort (tirage avec écarts-types connus, loi normale) pour calculer C. La procédure est répétée 100 000 fois. On calcule la moyenne et l'écart-type u(C).

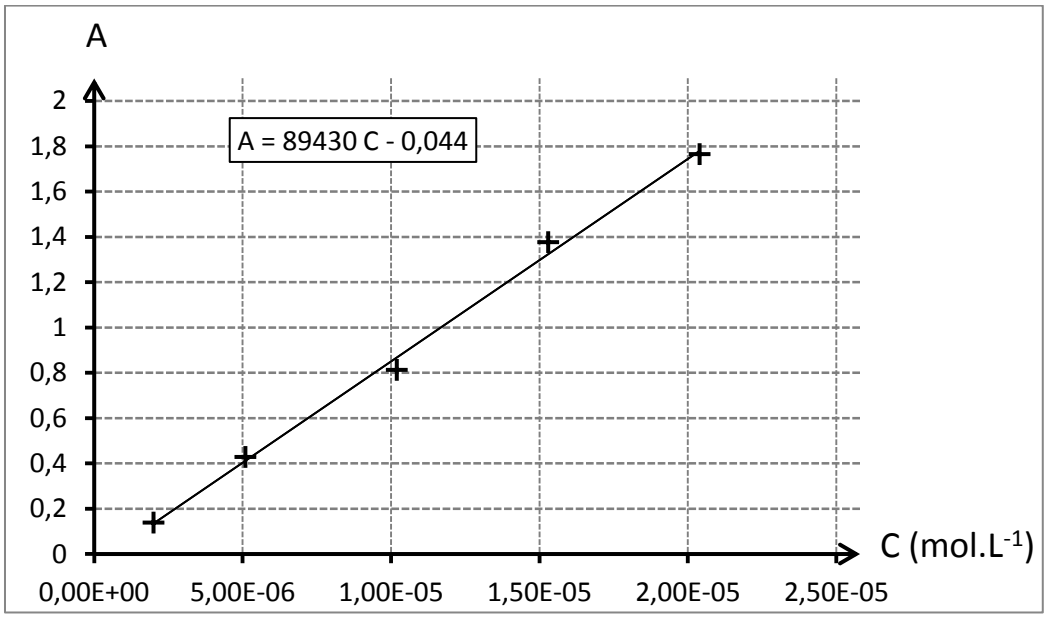
Un essai (programme Python en annexe 2) permet d'obtenir *(affichage brut)* :

```
Tube n° 1
Calcul de C : 2.0389249304911954e-05
Moyenne des C : 2.0389102116883587e-05
u(C) : 8.472527479440613e-08
Tube n° 2
Calcul de C : 1.5291936978683965e-05
Moyenne des C : 1.5292354859053284e-05
u(C) : 1.0286114694972822e-07
Tube n° 3
Calcul de C : 1.0194624652455977e-05
Moyenne des C : 1.019525874027162e-05
u(C) : 8.37431590097716e-08
Tube n° 4
Calcul de C : 5.0973123262279886e-06
Moyenne des C : 5.096695592597102e-06
u(C) : 8.3344716877658 27e-08
Tube n° 5
Calcul de C : 2.0389249304911958e-06
Moyenne des C : 2.0384217625804147e-06
u(C) : 9.245165141505295e-08
```

Tube	1	2	3	4	5
V_1 (mL)	10,00	7,50	5,00	2,50	1,00
$V_2 = 10 - V_1$ (mL)	0	2,50	5,00	7,50	9,00
C ($mol.L^{-1}$)	$2,039.10^{-5}$	$1,53.10^{-5}$	$1,019.10^{-5}$	$5,10.10^{-6}$	$2,04.10^{-6}$
u(C) ($mol.L^{-1}$)	$0,009.10^{-5}$	$0,01.10^{-5}$	$0,008.10^{-5}$	$0,08.10^{-6}$	$0,09.10^{-6}$
A_{640nm}	1,765	1,376	0,813	0,428	0,138



3. Le groupe 1 obtient par régression linéaire (méthode des moindres carrés) :



Les deux élèves du groupe 1 ont vérifié visuellement que les abscisses des points sont bien réparties (pas d'amas), que les points expérimentaux sont alignés. L'ajustement effectué permet d'obtenir la pente et l'ordonnée à l'origine de la droite de régression. Quelle incertitude-type y a-t-il sur ces deux grandeurs ? Quel nombre de chiffres significatifs faut-il conserver ? « 0 » est-il compatible avec l'ordonnée à l'origine ? Pour répondre à ces questions, on peut aussi utiliser la méthode de Monte-Carlo.

La variabilité de la pente et de l'ordonnée à l'origine est expliquée par différentes incertitudes qui s'accumulent tout au long du protocole : incertitudes de la pesée, de la masse molaire et de la fiole jaugée pour la dissolution ; incertitudes de la pipette jaugée et de la deuxième fiole jaugée pour la dilution ; incertitudes sur les burettes graduées pour la préparation de la gamme étalon ; incertitude du spectrophotomètre pour la mesure d'absorbance.

Démarche de la méthode de Monte-Carlo

1) Procéder à l'évaluation des incertitudes-type par une autre approche que statistique - type B :
- pesée : d'après la notice de la balance, on prendra u(m) = 1 mg ;
- masse molaire : u(M) = 0,01 $g.mol^{-1}$ (dernier chiffre significatif, incertitude-type négligeable) ;
- fiole jaugée : $u(V_{f1})$ = 0,0008 L (à lire sur la fiole jaugée) ;
- pipette jaugée : $u(V_p)$ = 0,02 mL (à lire sur la pipette jaugée) ;
- fiole jaugée : $u(V_{f2})$ = 0,3 mL (à lire sur la fiole jaugée) ;
- burette graduée : $u(V_1) = u(V_2)$ = 0,05 mL (à lire sur la burette graduée) ;
- spectrophotomètre : u(A)/A = 2 % (d'après la notice de l'appareil).

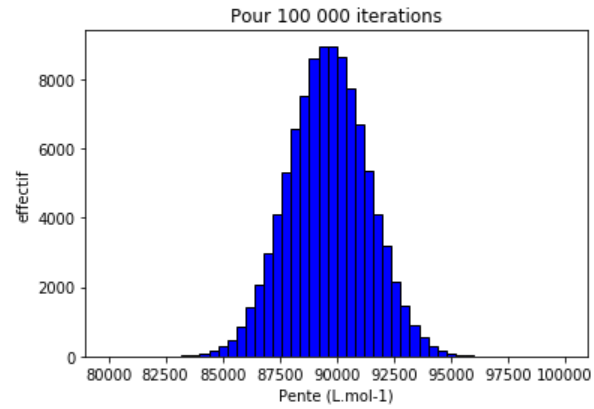
2) Prendre au hasard un jeu de données (m, M, V_{f1}, V_p, V_{f2}, V_1, V_2, A) - tirage avec écarts-types connus, loi normale.

3) Effectuer la régression linéaire liée à ce jeu de données.

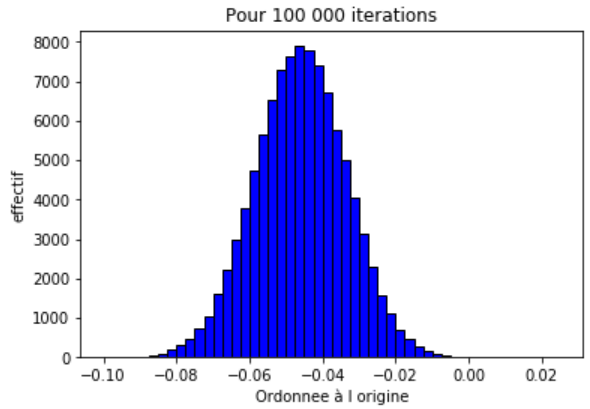
4) Répéter cette procédure 100 000 fois pour obtenir la moyenne et l'écart-type de la pente et de l'ordonnée à l'origine.

Résultats obtenus *(affichage brut)* pour le groupe 1, après 100 000 itérations (programme Python en annexe 3) :

```
Pente : 89621.51819683342
u(Pente) : 1760.3880877339418
```




```
Ordonnee à l origine : -0.046118624025373894
u(Ordonnée à l origine) 0.012589874308182326
```



Pente = 90.10^3 $L.mol^{-1}$ u(Pente) = 2.10^3 $L.mol^{-1}$

Ordonnée à l'origine = -0,05 u(Ordonnée à l'origine) = 0,01

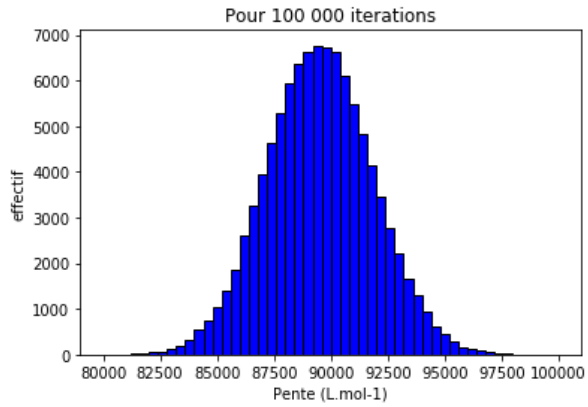
« 0 » est compatible avec l'ordonnée à l'origine à 5u près.

En considérant que l'expérimentateur est imprécis lors de la manipulation des burettes graduées

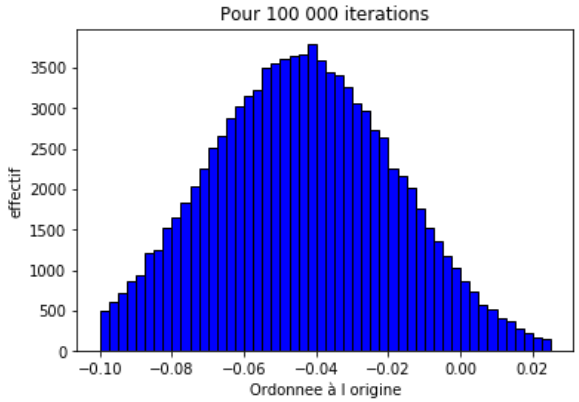
- burette graduée : $u(V_1) = u(V_2) = 0{,}2$ mL

on obtient :

```
Pente : 89501.66761153917
u(Pente) : 2334.637637968292
```




```
Ordonnee à l origine : -0.04482180604102111
u(Ordonnée à l origine) 0.027155657815468283
```

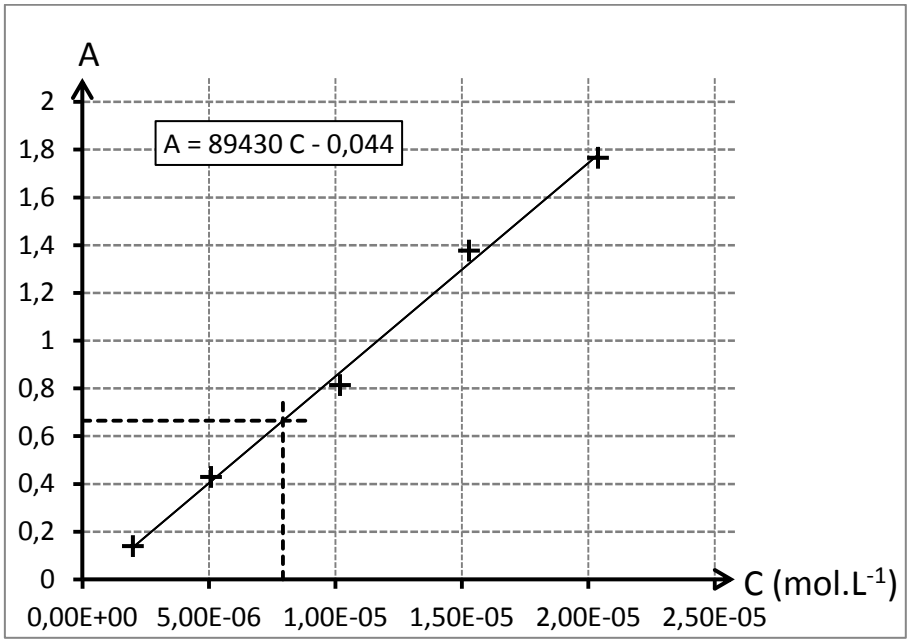


Pente = 90.10^3 L.mol^{-1} u(Pente) = 2.10^3 L.mol^{-1}
Ordonnée à l'origine = -0,04 u(Ordonnée à l'origine) = 0,03
« 0 » est alors compatible avec l'ordonnée à l'origine à 1u près.

4. L'objectif est maintenant d'effectuer un report de point pour en déduire une concentration. Le groupe 1 a trouvé une absorbance de la solution de bonbon gélifié bleu :

$$A_{640nm,S} = 0,665$$

Les élèves peuvent tout d'abord effectuer ce report de point graphiquement.



$$C_S = 7,928.10^{-6} \text{ mol.L}^{-1}$$

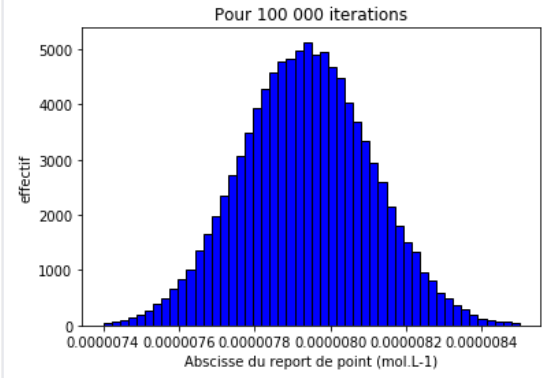
On cherche alors à évaluer l'incertitude-type de C_S en utilisant la méthode de Monte-Carlo.

Pour chacune des 100 000 droites de régression issues de jeux de données aléatoires, il est possible d'effectuer un report de point, pour lequel on prendra :

spectrophotomètre : $u(A_{640nm,S})/A_{640nm,S} = 2$ % (d'après la notice de l'appareil)

Résultats obtenus *(affichage brut)* pour le groupe 1, après 100 000 itérations (programme Python en annexe 4) :

```
Ordonnée pour le report : 0.665
Abscisse moyenne du report : 7.9359643485918l5e-06
Incertitude-type de l abscisse du report : 1.7255634801486812e-07
```



$$C_S = 7,9.10^{-6} \text{ mol.L}^{-1} \qquad u(C_S) = 0,2.10^{-6} \text{ mol.L}^{-1}$$

5. Pour conclure ce document, nous calculons la masse de colorant E131 présent dans un bonbon gélifié bleu.

$$m = M \times C_S \times V_{f3}$$

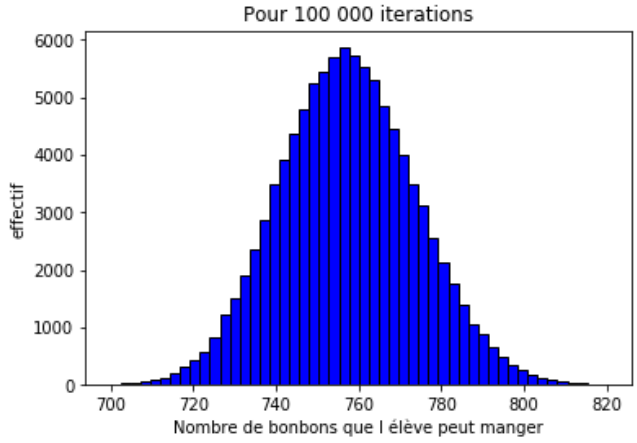
La DJA (Dose Journalière Admissible) de l'additif alimentaire E131 étant 2,5 mg par kg de masse corporelle et par jour, un élève de 70 kg peut manger :

$$N_{bonbons} = \frac{2{,}5.10^{-3} \times 70}{m}$$

En utilisant la méthode de Monte-Carlo (programme Python en annexe 5, *affichage brut*) :

- fiole jaugée : $u(V_{f3})$ = 0,05 mL (à lire sur la fiole jaugée)

```
Un élève pourrait manger : 757 bonbons...
Incertitude-type de ce nombre de bonbons : 16
```



Un élève pourrait manger maximum $7{,}6.10^2$ bonbons en une journée...

On trouve une incertitude-type de $0{,}2.10^2$ bonbons.

Annexes

Annexe 1 : programme Python pour évaluer de l'incertitude-type de C_m et C_0

```
import numpy as np
from matplotlib import pyplot

#Renvoie une valeur aléatoire de la variable L[0] d'incertitude-type L[1]
def Alea(L):
    tirage=np.random.normal()    #Tirage entre -infini et +infini (loi
normale)
    return L[0]+L[1]*tirage
################################################################################

#Entrées
m=[297.e-3,1.e-3]
M=[582.66,0.01]
Vf1=[1.0000,0.0008]
Vp=[10.00e-3,0.02e-3]
Vf2=[250.0e-3,0.3e-3]

#Calcul de Cm et C0
Cm=m[0]/(M[0]*Vf1[0])
C0=Cm*Vp[0]/Vf2[0]
################################################################################

#Méthode de Monte Carlo pour trouver l'incertitude sur Cm et C0
#sans composition des incertitudes
LCm,LC0=[],[]
iteration=100000

for i in range(iteration):
    AleaCm=Alea(m)/(Alea(M)*Alea(Vf1))
    AleaC0=AleaCm*Alea(Vp)/Alea(Vf2)
    LCm.append(AleaCm)
    LC0.append(AleaC0)

MoyCm=sum(LCm)/iteration
MoyC0=sum(LC0)/iteration
uCm=(1/(iteration-1)*sum((np.array(LCm)-MoyCm)**2.))**0.5
uC0=(1/(iteration-1)*sum((np.array(LC0)-MoyC0)**2.))**0.5
################################################################################

#Affichage
print('Calcul de Cm :',Cm)
print('Moyenne des Cm :',MoyCm)
print('u(Cm) :',uCm)
pyplot.hist(LCm, range = (0.000505, 0.000515), bins = 50, color = 'blue',
edgecolor = 'black')
pyplot.xlabel('Cm (mol/L)')
pyplot.ylabel('effectif')
pyplot.title('Pour 100 000 iterations')
pyplot.show()

print('Calcul de C0 :',C0)
print('Moyenne des C0 :',MoyC0)
print('u(C0) :',uC0)
pyplot.hist(LC0, range = (2.0e-5, 2.1e-5), bins = 50, color = 'blue',
edgecolor = 'black')
pyplot.xlabel('C0 (mol/L)')
pyplot.ylabel('effectif')
pyplot.title('Pour 100 000 iterations')
pyplot.show()
################################################################################
```

Annexe 2 : programme Python pour l'évaluation de l'incertitude-type de C

```
import numpy as np

#Renvoie une valeur aléatoire de la variable L[0] d'incertitude-type L[1]
def Alea(L):
    tirage=np.random.normal()    #Tirage entre -infini et +infini (loi normale)
    return L[0]+L[1]*tirage
##########################################################################

#Entrées
m=[297.e-3,1.e-3]
M=[582.66,0.01]
Vf1=[1.0000,0.0008]
Vp=[10.00e-3,0.02e-3]
Vf2=[250.0e-3,0.3e-3]
Dilutions=[10.00e-3,7.50e-3,5.00e-3,2.50e-3,1.00e-3]
Ajustements=[0,2.50e-3,5.00e-3,7.50e-3,9.00e-3]
DeltaBurette=0.05e-3
##########################################################################

#Préparation de la liste des concentrations des solutions diluées
Cm=m[0]/(M[0]*Vf1[0])
C0=Cm*Vp[0]/Vf2[0]
Concentration=[]
for k in range(len(Dilutions)):
    Concentration.append(C0*Dilutions[k]/(Dilutions[k]+Ajustements[k]))
##########################################################################

#Méthode de Monte Carlo pour trouver l'incertitude sur C
#sans composition des incertitudes
for k in range(len(Dilutions)):
    Dilu=[Dilutions[k],DeltaBurette]
    if Ajustements[k]==0:
        Ajust=[Ajustements[k],0]
    else:
        Ajust=[Ajustements[k],DeltaBurette]
    LC=[]
    iteration=100000
    for i in range(iteration):
        AleaCm=Alea(m)/(Alea(M)*Alea(Vf1))
        AleaC0=AleaCm*Alea(Vp)/Alea(Vf2)
        AleaC=AleaC0*1/(1+Alea(Ajust)/Alea(Dilu))
        LC.append(AleaC)

    MoyC=sum(LC)/iteration
    uC=(1/(iteration-1)*sum((np.array(LC)-MoyC)**2.))**0.5
    print('Tube n°',k+1)
    print('Calcul de C :',Concentration[k])
    print('Moyenne des C :',MoyC)
    print('u(C) :',uC)
##########################################################################
```

Annexe 3 : programme Python pour l'évaluation de l'incertitude-type de la pente et de l'ordonnée à l'origine de la droite de régression

```
import numpy as np
from matplotlib import pyplot

#Procedure Regression Linéaire ; tableaux np X et Y
def RegLin(X,Y):
    N=len(X)
    moyX=sum(X)/N
    moyY=sum(Y)/N
    pente=sum((X-moyX)*(Y-moyY))/(sum((X-moyX)**2))
    ordor=moyY-pente*moyX
    return [pente,ordor]
#########################################################################

#Renvoie une valeur aléatoire de la variable L[0] d'incertitude-type L[1]
def Alea(L):
    tirage=np.random.normal()    #Tirage entre -infini et +infini (loi normale)
    return L[0]+L[1]*tirage
#########################################################################

#Entrées (groupe 1)
m=[297.e-3,1.e-3]
M=[582.66,0.01]
Vf1=[1.0000,0.0008]
Vp=[10.00e-3,0.02e-3]
Vf2=[250.0e-3,0.3e-3]
Dilutions=[10.00e-3,7.50e-3,5.00e-3,2.50e-3,1.00e-3]
Ajustements=[0,2.50e-3,5.00e-3,7.50e-3,9.00e-3]
DeltaBurette=0.05e-3
Absorbance=[1.765,1.376,0.813,0.428,0.138]
ErelSpectro=0.02
#########################################################################

#Préparation des listes avec incertitudes
Dilu=[]
for k in range(len(Dilutions)):
    Dilu.append([Dilutions[k],DeltaBurette])

Ajust=[]
for k in range(len(Dilutions)):
    if Ajustements[k]==0:
        Ajust.append([Ajustements[k],0])
    else:
        Ajust.append([Ajustements[k],DeltaBurette])

A=[]
for k in range(len(Dilutions)):
    A.append([Absorbance[k],Absorbance[k]*ErelSpectro])
#########################################################################
```

```
#Méthode de Monte Carlo pour trouver la pente, l'ordonnée à l'origine et
#les incertitudes-type
LPente,LOrdOr=[],[]
Iteration=100000

for j in range(Iteration):
    AleaCm=Alea(m)/(Alea(M)*Alea(Vf1))
    AleaC0=AleaCm*Alea(Vp)/Alea(Vf2)
    AleaC=[]
    AleaA=[]
    for k in range(len(Dilutions)):
        AleaC.append(AleaC0/(1+Alea(Ajust[k])/Alea(Dilu[k])))
        AleaA.append(Alea(A[k]))

    Pente=RegLin(np.array(AleaC),np.array(AleaA))[0]
    OrdOr=RegLin(np.array(AleaC),np.array(AleaA))[1]
    LPente.append(Pente)
    LOrdOr.append(OrdOr)

MoyPente=sum(LPente)/Iteration
MoyOrdOr=sum(LOrdOr)/Iteration
uPente=(1/(Iteration-1)*sum((np.array(LPente)-MoyPente)**2.))**0.5
uOrdOr=(1/(Iteration-1)*sum((np.array(LOrdOr)-MoyOrdOr)**2.))**0.5

print('Pente :',MoyPente)
print('u(Pente) :',uPente)
pyplot.hist(LPente, range = (80000, 100000), bins = 50, color = 'blue',
edgecolor = 'black')
pyplot.xlabel('Pente (L.mol-1)')
pyplot.ylabel('effectif')
pyplot.title('Pour 100 000 iterations')
pyplot.show()

print('Ordonnee à l origine :',MoyOrdOr)
print('u(Ordonnée à l origine)',uOrdOr)
pyplot.hist(LOrdOr, range = (-0.100, 0.025), bins = 50, color = 'blue',
edgecolor = 'black')
pyplot.xlabel('Ordonnee à l origine')
pyplot.ylabel('effectif')
pyplot.title('Pour 100 000 iterations')
pyplot.show()
#########################################################################
```

Annexe 4 : Programme Python pour l'évaluation de l'incertitude-type de l'abscisse après report de point

```
import numpy as np
from matplotlib import pyplot

#Procedure Regression Linéaire ; tableaux np X et Y
def RegLin(X,Y):
    N=len(X)
    moyX=sum(X)/N
    moyY=sum(Y)/N
    pente=sum((X-moyX)*(Y-moyY))/(sum((X-moyX)**2))
    ordor=moyY-pente*moyX
    return [pente,ordor]
##########################################################################

#Renvoie une valeur aléatoire de la variable L[0] d'incertitude-type L[1]
def Alea(L):
    tirage=np.random.normal()    #Tirage entre -infini et +infini (loi normale)
    return L[0]+L[1]*tirage
##########################################################################

#Entrées (groupe 1)
m=[297.e-3,1.e-3]
M=[582.66,0.01]
Vf1=[1.0000,0.0008]
Vp=[10.00e-3,0.02e-3]
Vf2=[250.0e-3,0.3e-3]
Dilutions=[10.00e-3,7.50e-3,5.00e-3,2.50e-3,1.00e-3]
Ajustements=[0,2.50e-3,5.00e-3,7.50e-3,9.00e-3]
DeltaBurette=0.05e-3
Absorbance=[1.765,1.376,0.813,0.428,0.138]
ErelSpectro=0.02
Report=[0.665,0.02*0.665]
##########################################################################

#Préparation des listes avec incertitudes
Dilu=[]
for k in range(len(Dilutions)):
    Dilu.append([Dilutions[k],DeltaBurette])

Ajust=[]
for k in range(len(Dilutions)):
    if Ajustements[k]==0:
        Ajust.append([Ajustements[k],0])
    else:
        Ajust.append([Ajustements[k],DeltaBurette])

A=[]
for k in range(len(Dilutions)):
    A.append([Absorbance[k],Absorbance[k]*ErelSpectro])
##########################################################################
```

```
##########################################################################
#Méthode de Monte Carlo pour un report de point
LPreviX=[]
Iteration=100000

for j in range(Iteration):
    AleaCm=Alea(m)/(Alea(M)*Alea(Vf1))
    AleaC0=AleaCm*Alea(Vp)/Alea(Vf2)
    AleaC=[]
    AleaA=[]
    for k in range(len(Dilutions)):
        AleaC.append(AleaC0/(1+Alea(Ajust[k])/Alea(Dilu[k])))
        AleaA.append(Alea(A[k]))
    AleaReport=Alea(Report)

    Pente=RegLin(np.array(AleaC),np.array(AleaA))[0]
    OrdOr=RegLin(np.array(AleaC),np.array(AleaA))[1]
    LPreviX.append((AleaReport-OrdOr)/Pente)

MoyPreviX=sum(LPreviX)/Iteration
uPreviX=(1/(Iteration-1)*sum((np.array(LPreviX)-MoyPreviX)**2.))**0.5

print('Ordonnée pour le report :',Report[0])
print('Abscisse moyenne du report :',MoyPreviX)
print('Incertitude-type de l abscisse du report :',uPreviX)
pyplot.hist(LPreviX, range = (7.4e-6,8.5e-6), bins = 50, color = 'blue',
edgecolor = 'black')
pyplot.xlabel('Abscisse du report de point (mol.L-1)')
pyplot.ylabel('effectif')
pyplot.title('Pour 100 000 iterations')
pyplot.show()
##########################################################################
```

Annexe 5 : Programme Python pour l'évaluation de l'incertitude-type du nombre de bonbons

```
import numpy as np
from matplotlib import pyplot

#############################################################################
#Procedure Regression Linéaire ; tableaux np X et Y
def RegLin(X,Y):
    N=len(X)
    moyX=sum(X)/N
    moyY=sum(Y)/N
    pente=sum((X-moyX)*(Y-moyY))/(sum((X-moyX)**2))
    ordor=moyY-pente*moyX
    return [pente,ordor]

#############################################################################
#Renvoie une valeur aléatoire de la variable L[0] d'incertitude-type L[1]
def Alea(L):
    tirage=np.random.normal()   #Tirage entre -infini et +infini (loi normale)
    return L[0]+L[1]*tirage
#############################################################################

#############################################################################
#Entrées (groupe 1)
m=[297.e-3,1.e-3]
M=[582.66,0.01]
Vf1=[1.0000,0.0008]
Vp=[10.00e-3,0.02e-3]
Vf2=[250.0e-3,0.3e-3]
Dilutions=[10.00e-3,7.50e-3,5.00e-3,2.50e-3,1.00e-3]
Ajustements=[0,2.50e-3,5.00e-3,7.50e-3,9.00e-3]
DeltaBurette=0.05e-3
Absorbance=[1.765,1.376,0.813,0.428,0.138]
ErelSpectro=0.02
Report=[0.665,0.02*0.665]
Vf3=[50.00e-3,0.05e-3]
#############################################################################

#############################################################################
#Préparation des listes avec incertitudes
Dilu=[]
for k in range(len(Dilutions)):
    Dilu.append([Dilutions[k],DeltaBurette])

Ajust=[]
for k in range(len(Dilutions)):
    if Ajustements[k]==0:
        Ajust.append([Ajustements[k],0])
    else:
        Ajust.append([Ajustements[k],DeltaBurette])

A=[]
for k in range(len(Dilutions)):
    A.append([Absorbance[k],Absorbance[k]*ErelSpectro])
#############################################################################
```

```
#########################################################################
#Méthode de Monte Carlo pour le nombre de bonbons
LNbonbon=[]
Iteration=100000

for j in range(Iteration):
    AleaM=Alea(M)
    AleaCm=Alea(m)/(AleaM*Alea(Vf1))
    AleaC0=AleaCm*Alea(Vp)/Alea(Vf2)
    AleaC=[]
    AleaA=[]
    for k in range(len(Dilutions)):
        AleaC.append(AleaC0/(1+Alea(Ajust[k])/Alea(Dilu[k])))
        AleaA.append(Alea(A[k]))
    AleaReport=Alea(Report)

    Pente=RegLin(np.array(AleaC),np.array(AleaA))[0]
    OrdOr=RegLin(np.array(AleaC),np.array(AleaA))[1]
    Cs=(AleaReport-OrdOr)/Pente
    LNbonbon.append(2.5e-3*70/(AleaM*Cs*Alea(Vf3)))

MoyNbonbon=sum(LNbonbon)/Iteration
uNbonbon=(1/(Iteration-1)*sum((np.array(LNbonbon)-MoyNbonbon)**2.))**0.5

print('Un élève pourrait manger :', int(MoyNbonbon), 'bonbons...')
print('Incertitude-type de ce nombre de bonbons :',int(uNbonbon))
pyplot.hist(LNbonbon, range = (700,820), bins = 50, color = 'blue', edgecolor
= 'black')
pyplot.xlabel('Nombre de bonbons que l élève peut manger')
pyplot.ylabel('effectif')
pyplot.title('Pour 100 000 iterations')
pyplot.show()
#########################################################################
```

Ces programmes écrits en langage Python sont disponibles dans l'annexe disponible sur la page éduscol du GRIESP.