Fiche TP

Régression linéaire

Régression linéaire.	Utiliser un logiciel de régression linéaire afin d'obtenir les valeurs des paramètres du modèle. Analyser les résultats obtenus à l'aide d'une procédure de validation : analyse graphique intégrant les barres d'incertitude ou analyse des écarts normalisés. **Capacité numérique** : à l'aide d'un langage de programmation ou d'un tableur, simuler un processus aléatoire de variation des valeurs expérimentales de l'une des grandeurs – simulation Monte-Carlo – pour évaluer l'incertitude-type sur les paramètres du modèle.

Un scientifique cherche à expliquer le monde qui l'entoure, à interpréter des observations, des **faits expérimentaux**. Il établit des **lois scientifiques** qui reposent sur des expériences. La première étape pour établir une loi est de faire une **hypothèse**, c'est-à-dire une proposition intuitive d'explication, exprimée en termes de concepts plus fondamentaux. Quand une hypothèse est confirmée, parfois par la réussite d'expériences complémentaires qu'elle a suscitées ou par une formulation plus élaborée (souvent en termes mathématiques) qui la place dans un contexte scientifique plus large, elle est élevée au statut de **théorie**.

Pour développer une théorie, un scientifique modélise son système d'étude. Un **modèle** est une version simplifiée du système qui se limite aux aspects essentiels du problème. Une fois construit, le modèle est confronté aux observations connues et aux expériences qu'il inspire. **Le modèle permet-il d'expliquer les données expérimentales ?**

Il est possible de comparer des modèles. **Quel est le meilleur modèle qui permet d'expliquer des données expérimentales ?** Un modèle peut être perfectionné ou affiné, par retour d'expériences. L'objectif est de pouvoir utiliser le modèle pour prévoir des comportements.

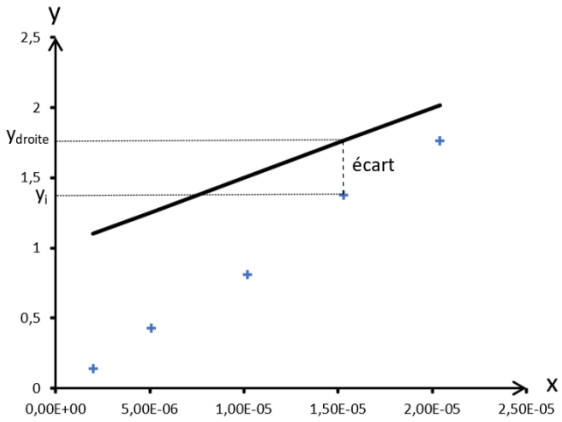
Lors d'une modélisation, on détermine une équation qui sert à l'**ajustement des données expérimentales**. Lorsqu'il s'agit d'une équation de droite $y = ax + b$, on parle de **régression linéaire**.

1. Ajustement de données expérimentales par une droite

Il existe différentes méthodes[1] pour déterminer la meilleure droite qui ajuste des données expérimentales. L'une d'entre elles consiste à *« minimiser la somme des carrés des écarts verticaux à la droite »*, c'est **la méthode des moindres carrés**. Elle présente l'avantage d'être implantée dans les calculatrices, les logiciels de traitement de données (LibreOffice Calc, Microsoft Excel, Google Sheets, Regressi…).

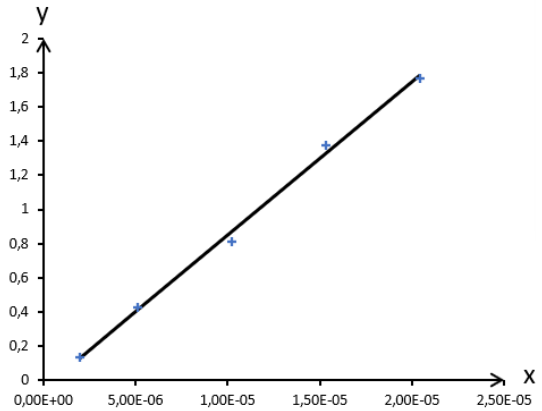
Données expérimentales

x_i	$2{,}04.10^{-5}$	$1{,}53.10^{-5}$	$1{,}02.10^{-5}$	$5{,}1.10^{-6}$	$2{,}0.10^{-6}$
y_i	1,765	1,376	0,813	0,428	0,138



L'activité présentée en *Annexe n°1* permet de comprendre la méthode des moindres carrés. On dispose d'un jeu de données expérimentales (x_i, y_i). On cherche la pente a et l'ordonnée à l'origine b de la droite de régression $y = ax + b$. Les deux paramètres a et b sont ajustés « à la main », puis optimisés par le solveur d'Excel pour minimiser la somme des carrés des écarts verticaux.

Essai n°4	xi	yi		Ydroite	Ecarts	Ecarts²
	2,04E-05	1,765		**1,78042**	0,01542	0,00024
	1,53E-05	1,376		**1,32432**	-0,05168	0,00267
	1,02E-05	0,813		**0,86823**	0,05523	0,00305
	5,10E-06	0,428		**0,41213**	-0,01587	0,00025
	2,00E-06	0,138		**0,1349**	-0,0031	9,6E-06
	pente **a** :	89430,1		Somme à minimiser :		0,00622
ordonnée à l'origine **b** :		-0,04396				



On trouve $a = 89430{,}1$ et $b = -0{,}04396$.

[1] L'article " Quelle est la « meilleure » droite ? " du B.U.P. Vol. 113, février 2019, par Thierry Pré et François Hernandez, présente des méthodes alternatives à la méthode des moindres carrés pour l'ajustement de données expérimentales : moindres carrés des écarts horizontaux, moindres carrés des distances orthogonales, moindres écarts verticaux absolus.

Il est possible de calculer la pente a et l'ordonnée à l'origine b avec les expressions suivantes :

$$a = \frac{\sum_{i=1}^{n} [(x_i - \bar{x})\,(y_i - \bar{y})]}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \qquad b = \bar{y} - a\bar{x}$$

où $\bar{x}$ est la moyenne des abscisses x_i et $\bar{y}$ la moyenne des ordonnées y_i.

	x_i	y_i	$x_i - \bar{x}$	$y_i - \bar{y}$	$(x_i - \bar{x})(y_i - \bar{y})$	$(x_i - \bar{x})^2$
	2,04E-05	1,765	9,80E-06	0,861	8,44E-06	9,60E-11
	1,53E-05	1,376	4,70E-06	0,472	2,22E-06	2,21E-11
	1,02E-05	0,813	-4,00E-07	-0,091	3,64E-08	1,60E-13
	5,10E-06	0,428	-5,50E-06	-0,476	2,62E-06	3,03E-11
	2,00E-06	0,138	-8,60E-06	-0,766	6,59E-06	7,40E-11
Moyennes :	1,06E-05	0,904		*Sommes :*	*1,99E-05*	*2,23E-10*
				a	89430,1	
				b	-0,04396	

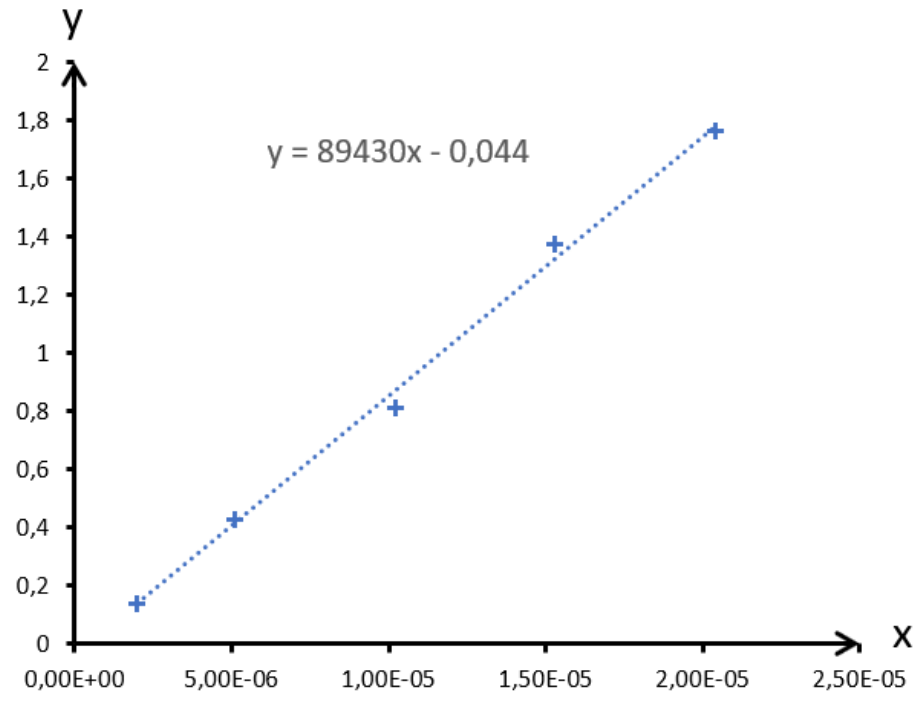
On retrouve $a = 89430{,}1$ et $b = -0{,}04396$.

Ces résultats sont obtenus directement avec Excel ou Python avec les instructions suivantes :

Avec Excel `= PENTE(y_connus;x_connus)`
`= ORDONNEE.ORIGINE(y_connus;x_connus)`

Ou

`Clic droit sur un graphique + ajout de courbe de tendance`
`Afficher l'équation sur le graphique`

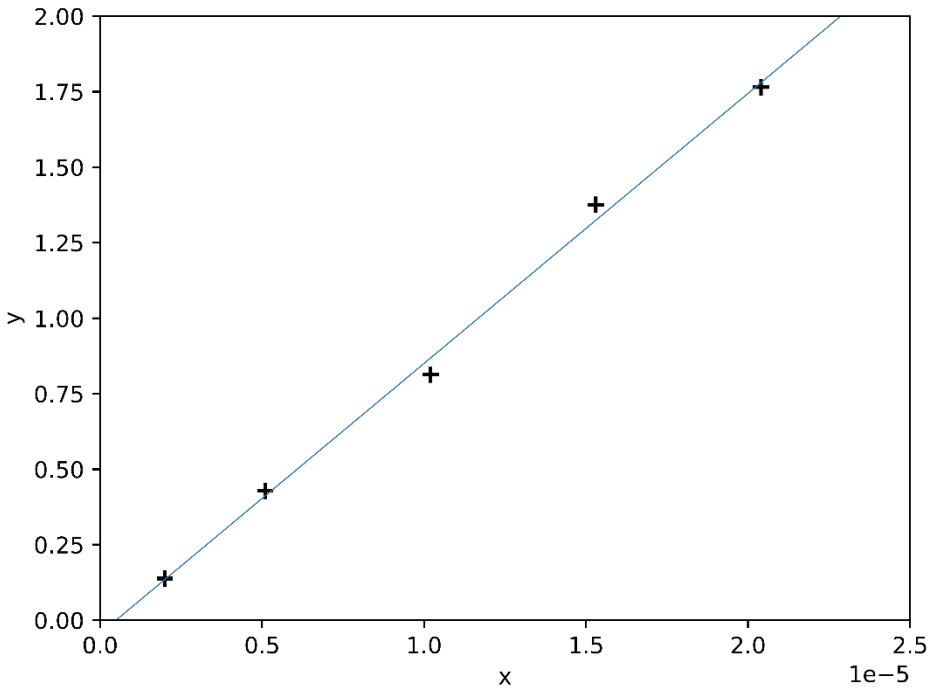


Avec Python = numpy.polyfit(xi,yi,1)

```
import numpy as np
from matplotlib import pyplot

xi=[2.040e-5,1.530e-5,1.020e-5,5.10e-6,2.00e-6]
yi=[1.765,1.376,0.813,0.428,0.138]

reglin=np.polyfit(xi,yi,1)

a=reglin[0]
b=reglin[1]

print('                Pente : ',a)
print('Ordonnée à l origine : ',b)

pyplot.scatter(xi,yi,marker='+',s=50,color='black')
pyplot.plot(np.linspace(0,2.5e-5,2),reglin[0]*np.linspace(0,2.5e-5,2)+reglin[1],linewidth=0.5)
pyplot.xlabel('x')
pyplot.ylabel('y')
pyplot.axis([0,2.5e-5, 0, 2])
pyplot.show()
```

Résultat :

```
                Pente :  89430.11235955053
Ordonnée à l origine :  -0.04395919101123579
```



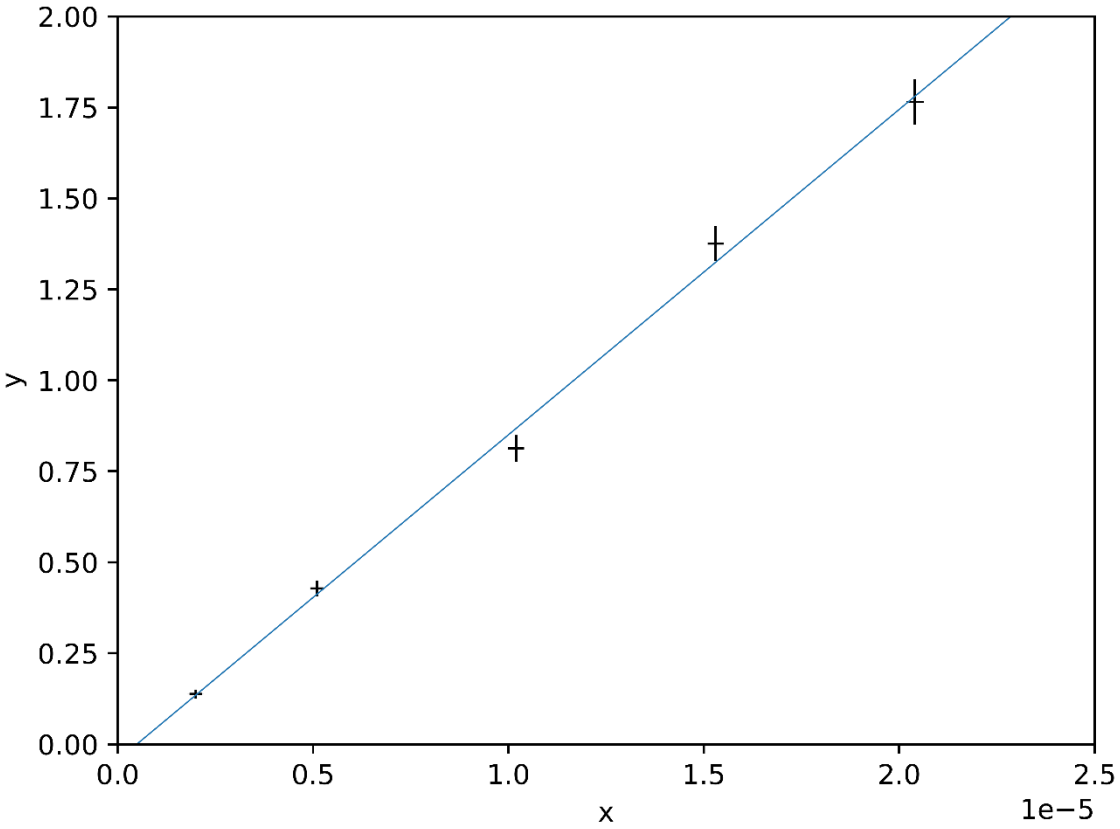
2. Incertitude-type des paramètres du modèle

Les valeurs expérimentales sur lesquelles on travaille sont entachées d'**incertitudes**.

Données expérimentales

x_i	$2{,}040.10^{-5}$	$1{,}530.10^{-5}$	$1{,}020.10^{-5}$	$5{,}10.10^{-6}$	$2{,}00.10^{-6}$
$u(x_i)$	$0{,}022.10^{-5}$	$0{,}020.10^{-5}$	$0{,}020.10^{-5}$	$0{,}17.10^{-6}$	$0{,}16.10^{-6}$
y_i	1,765	1,376	0,813	0,428	0,138
$u(y_i)$	0,062	0,048	0,036	0,021	0,012

```
import numpy as np
from matplotlib import pyplot

#########################################################################################################################################
#Procedure tracé d'une droite px+o entre les abscisses x1 et x2, n points
def Droite(p,o,x1,x2,n):
    x=np.linspace(x1,x2,n)
    y=p*x+o
    pyplot.plot(x,y,linewidth=0.5)

#########################################################################################################################################
xi=[2.040e-5,1.530e-5,1.020e-5,5.10e-6,2.00e-6]     #abscisses expérimentales x_i
uxi=[0.022e-5,0.020e-5,0.020e-5,0.17e-6,0.16e-6]    #incertitudes-types sur les x_i

yi=[1.765,1.376,0.813,0.428,0.138]                  #ordonnées expérimentales y_i
uyi=[0.062,0.048,0.036,0.021,0.012]                 #incertitudes-types sur les y_i

pyplot.errorbar(xi,yi,xerr=uxi,yerr=uyi,fmt='+',color='black',markeredgewidth=0,linewidth=0.
8)
reglin=np.polyfit(xi,yi,1)
Droite(reglin[0],reglin[1],0,2.5e-5,2)
pyplot.xlabel('x')
pyplot.ylabel('y')
pyplot.axis([0,2.5e-5, 0, 2])
pyplot.show()
```



Points expérimentaux en barres d'incertitudes, régression linaire

Les paramètres d'ajustement a et b (pente et ordonnée à l'origine de la droite de régression), ont donc une certaine **variabilité**. On cherche à déterminer leur **incertitude-type**.

Il n'est pas facile de composer les incertitudes de toutes les abscisses et de toutes les ordonnées des points expérimentaux. On procède donc à des **simulations Monte-Carlo**. Connaissant les incertitudes-types expérimentales, on génère aléatoirement des jeux de données (x_i, y_i). On détermine par régression linéaire des valeurs possibles des paramètres d'ajustement a et b. Il s'agit de **recréer des expériences virtuelles**, d'**obtenir une collection de résultats de mesures simulées**.

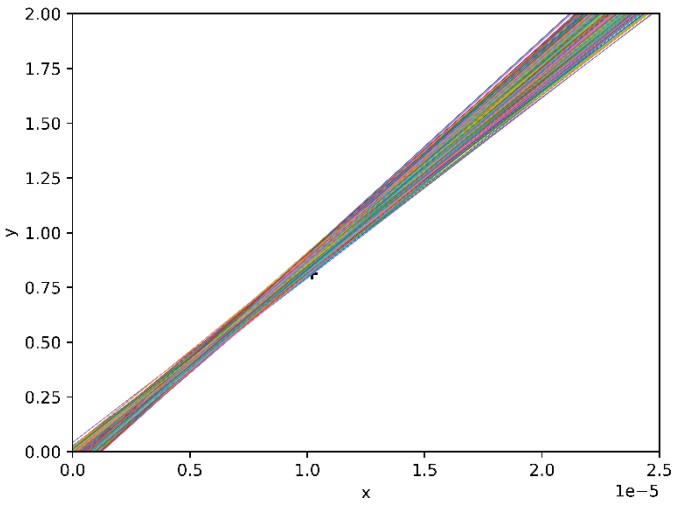
```
import numpy as np
from matplotlib import pyplot

#########################################################################################################
#Procedure tracé d'une droite px+o entre les abscisses x1 et x2, n points
def Droite(p,o,x1,x2,n):
    x=np.linspace(x1,x2,n)
    y=p*x+o
    pyplot.plot(x,y,linewidth=0.5)
#########################################################################################################

xi=[2.040e-5,1.530e-5,1.020e-5,5.10e-6,2.00e-6]     #abscisses expérimentales x_i
uxi=[0.022e-5,0.020e-5,0.020e-5,0.17e-6,0.16e-6]    #incertitudes-types sur les x_i

yi=[1.765,1.376,0.813,0.428,0.138]                  #ordonnées expérimentales y_i
uyi=[0.062,0.048,0.036,0.021,0.012]                 #incertitudes-types sur les y_i

NbSimulations=10000
La=[]                                       #initialisation de la liste des pentes
Lb=[]                                       #initialisation de la liste des ordonnées à l'origine

for k in range(NbSimulations):
    Lxi=[]                                          #initialisation du jeu de xi
    Lyi=[]                                          #initialisation du jeu de yi
    for j in range(len(xi)):
        Lxi.append(np.random.normal(xi[j],uxi[j]))  #tirage au sort d'un jeu de xi
        Lyi.append(np.random.normal(yi[j],uyi[j]))  #tirage au sort d'un jeu de yi
    reglin=np.polyfit(Lxi,Lyi,1)                    #régression linéaire sur le jeu de données
    Droite(reglin[0],reglin[1],0,2.5e-5,10)         #tracé de la droite de régression
    La.append(reglin[0])                            #stockage de la pente
    Lb.append(reglin[1])                            #stockage de l'ordonnée à l'origine

print('Pente moyenne : ',np.average(La))                            #affichage de a moyen
print('Incertitude-type sur la pente u(a) : ',np.std(La, ddof=1))   #affichage de u(a)

print('Ordonnée à l origine moyenne : ',np.average(Lb))             #affichage de b moyen
print('Incertitude-type sur la pente u(b) : ',np.std(Lb, ddof=1))   #affichage de u(b)

pyplot.scatter(xi,yi,marker='+',s=50,color='black')
pyplot.xlabel('x')
pyplot.ylabel('y')
pyplot.axis([0,2.5e-5, 0, 2])
pyplot.show()
```

Résultat :

```
Pente moyenne :  89426.72912546908
Incertitude-type sur la pente u(a) :  3162.5975017680244
Ordonnée à l origine moyenne :  -0.043702193415915774
Incertitude-type sur la pente u(b) :  0.025468435106108942
```

$\bar{a} = 89{,}4.10^3$ avec $u(a) = 3{,}2.10^3$

$\bar{b} = -0{,}044$ avec $u(b) = 0{,}025$



Variabilité des droites de régressions

3. Procédure de validation

Le modèle envisagé doit être confronté aux résultats expérimentaux**. Ce modèle est-il compatible avec les résultats expérimentaux ?**

Pour répondre à cette question, on peut tracer le **graphique des résidus**. On vérifie que les points expérimentaux sont aléatoirement répartis de part et d'autre de la droite de régression. On peut faire figurer les incertitudes-types des ordonnées sur le graphique des résidus.

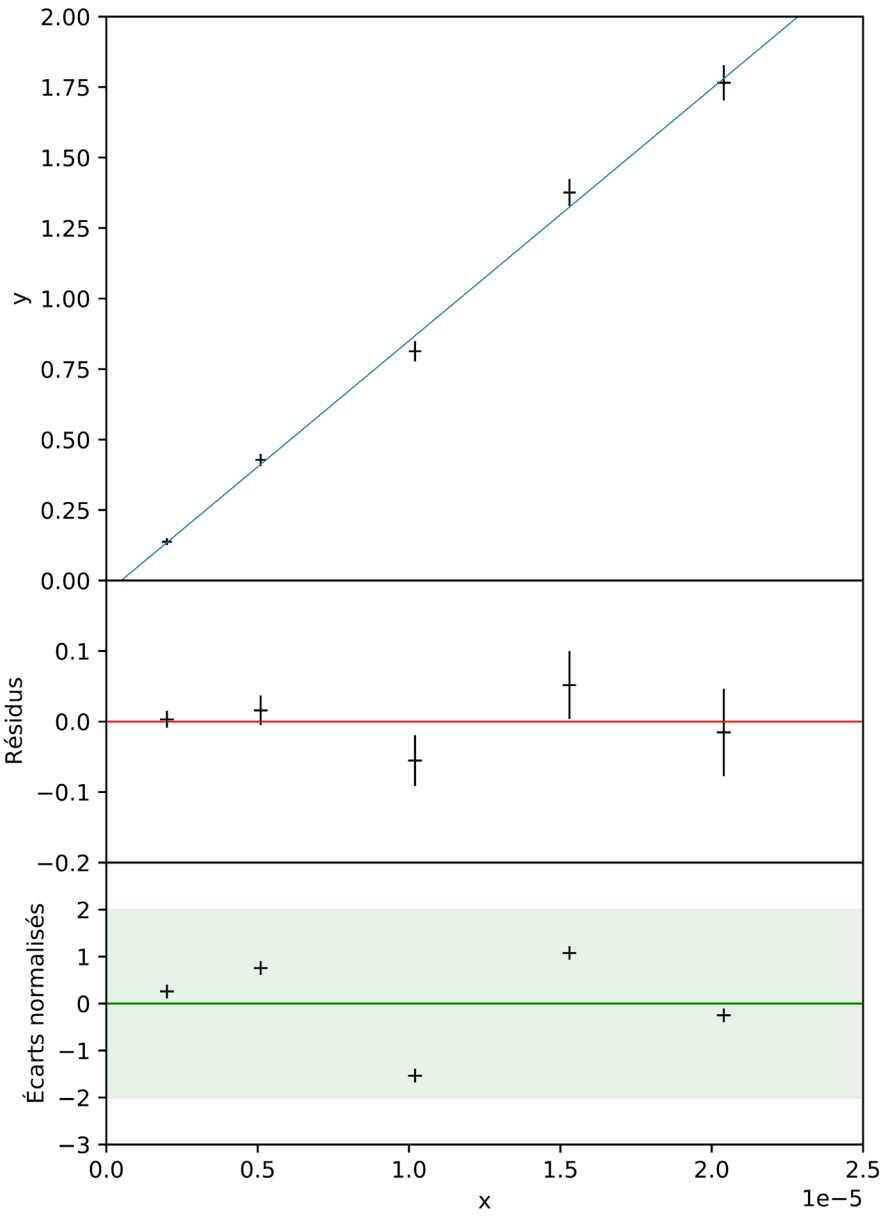
Un autre critère de validation consiste à tracer le **graphique des écarts normalisés** (z-scores z_{x_i} sur les ordonnées, pour chaque abscisse x_i). Si les écarts normalisés sont inférieurs à 2, alors on peut conclure que le modèle est compatible avec les résultats.

$$z_{x_i} = \frac{|y_i - y_{droite}|}{u(y_i)}$$

Il est possible de tester plusieurs modèles concurrents et de choisir celui qui est le plus compatible avec les résultats expérimentaux en **comparant les graphiques des écarts normalisés** (*voir Annexe n°2*)

```
import numpy as np
from matplotlib import pyplot

#########################################################################################################################################
#Données expérimentales et incertitudes-types
xi=[2.040e-5,1.530e-5,1.020e-5,5.10e-6,2.00e-6]
uxi=[0.022e-5,0.020e-5,0.020e-5,0.17e-6,0.16e-6]

yi=[1.765,1.376,0.813,0.428,0.138]
uyi=[0.062,0.048,0.036,0.021,0.012]

#########################################################################################################################################
#Régression linéaire
#Calcul des ordonnées sur la droite de régression pour les abscisses expérimentales
#Calcul des résidus
#Calcul des écarts normalisés
reglin=np.polyfit(xi,yi,1)
ydroite=reglin[0]*np.array(xi)+reglin[1]
residus=np.array(yi)-ydroite
ecartsnormalises=residus/np.array(uyi)

#########################################################################################################################################
#Définition d'une figure de 4 lignes (2+1+1), 1 colonne
fig3=pyplot.figure(3,figsize=(6,9))
gridsize = (4, 1)

#########################################################################################################################################
#Graphique des données expérimentales avec incertitudes-types en abscisse et en ordonnée
#Ajustement des points par régression linéaire
#Ce graphique tient sur 2 lignes de la figure
ax1=pyplot.subplot2grid(gridsize, (0, 0), colspan=1, rowspan=2)
ax1.plot(np.linspace(0,2.5e-5,2),reglin[0]*np.linspace(0,2.5e-5,2)+reglin[1],linewidth=0.5)
ax1.errorbar(xi,yi,xerr=uxi,yerr=uyi,fmt='+',color='black',markeredgewidth=0,linewidth=0.8)
ax1.axis([0,2.5e-5, 0, 2])
ax1.set_ylabel('y')

#########################################################################################################################################
#Graphique des résidus avec incertitudes-types en ordonnée
#Ce graphique tient sur 1 ligne de la figure
#Partage des abscisses avec le graphique n°1
#Tracé en rouge de la droite y=0
ax2=pyplot.subpl ot2grid(gridsize, (2, 0), sharex=ax1)
ax2.plot(np.linspace(0,2.5e-5,2),[0,0],linewidth=0.8,color='red')
ax2.errorbar(xi,residus,yerr=uyi,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax2.axis([0,2.5e-5, -0.2, 0.2])
ax2.set_ylabel('Résidus')
```

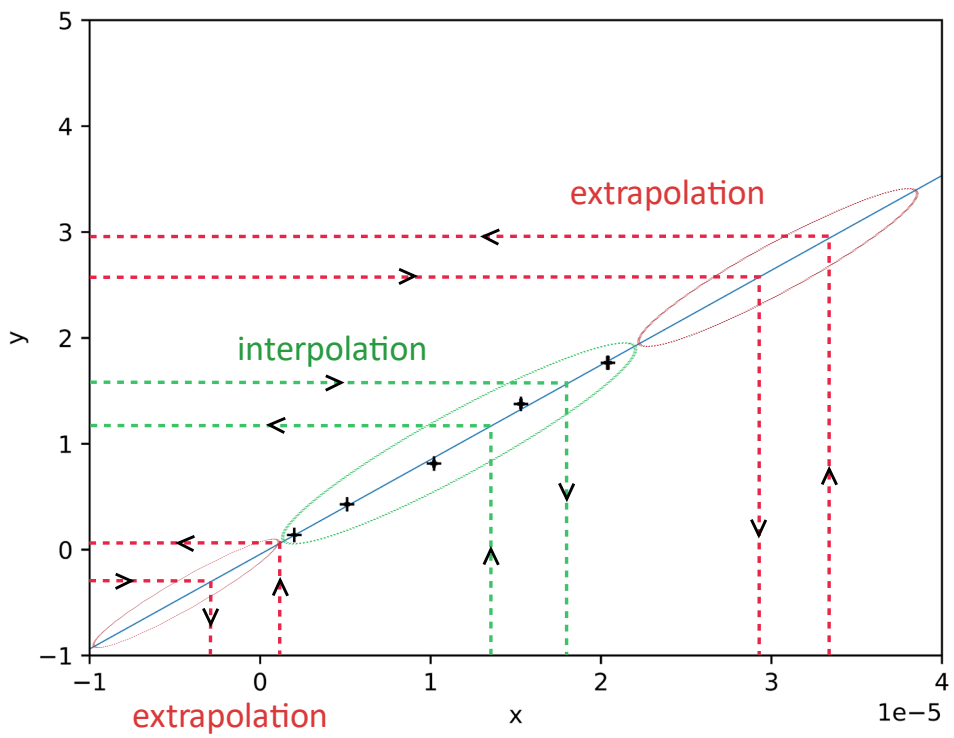
```
#########################################################################################################################################
#Graphique des écarts normalisés
#Ce graphique tient sur 1 ligne de la figure
#Partage des abscisses avec le graphique n°1
#Tracé en vert de la droite y=0
#Remplissage en vert de la zone de compatibilité
ax3=pyplot.subplot2grid(gridsize, (3, 0), sharex=ax1)
ax3.scatter(xi,ecartsnormalises,marker='+',color='black',linewidth=0.8)
ax3.axis([0,2.5e-5, -3, 3])
ax3.plot(np.linspace(0,2.5e-5,2),[0,0],linewidth=0.8,color='green')
ax3.fill_between([0,2.5e-5],y1=-2,y2=2,color='green', alpha=0.1)
ax3.set_xlabel('x')
ax3.set_ylabel('Écarts normalisés')

#########################################################################################################################################
#Réduction des espaces entre les graphiques
#On enlève les graduations des deux graphiques supérieurs
#On enlève les graduations qui se superposent
#Affichage de la figure
fig3.subplots_adjust(hspace=0)
pyplot.setp(ax1.get_xticklabels(), visible=False)
pyplot.setp(ax2.get_xticklabels(), visible=False)
ax2.set_yticks(ax2.get_yticks()[:-1])
ax3.set_yticks(ax3.get_yticks()[:-1])
pyplot.show()
```



On peut conclure que le modèle est compatible avec les données expérimentales.

4. Interpolation, extrapolation

Le modèle, s'il est validé, peut être utilisé pour prévoir les couples des grandeurs physiques étudiées. On procède à un report de point. On peut **interpoler** ou **extrapoler** une valeur numérique après un ajustement de données expérimentales.



L'incertitude-type d'une abscisse ou d'une ordonnée interpolée ou extrapolée peut être déterminée par **simulations Monte-Carlo**. Connaissant les incertitudes-types expérimentales, on génère aléatoirement des jeux de données (x_i, y_i). On détermine par régression linéaire des valeurs possibles des paramètres d'ajustement a et b. On effectue un report de point sur chaque droite de régression simulée (on peut même tenir compte de l'incertitude-type de l'abscisse ou de l'ordonnée de départ du report de point). On accède ainsi à la variabilité de l'abscisse ou de l'ordonnée interpolée ou extrapolée. Il s'agit de **recréer des expériences virtuelles**, d'**obtenir une collection de résultats de mesures simulées**.

```
import numpy as np
from matplotlib import pyplot

#########################################################################################################################
#Données expérimentales et incertitudes-types
xi=[2.040e-5,1.530e-5,1.020e-5,5.10e-6,2.00e-6]
uxi=[0.022e-5,0.020e-5,0.020e-5,0.17e-6,0.16e-6]

yi=[1.765,1.376,0.813,0.428,0.138]
uyi=[0.062,0.048,0.036,0.021,0.012]

#########################################################################################################################
#Ordonnée de départ du report de point et incertitude-type associée
Yreport=1.125
uYreport=0.042
```

```
#########################################################################################################################
#Choix du nombre de simulations Monte-Carlo
#Initialisation de la liste des abscisses après report
NbSimulations=10000
LXreport=[]

#########################################################################################################################
#Simulation Monte-Carlo
for k in range(NbSimulations):
    Lxi=[]      #initialisation du jeu de xi
    Lyi=[]      #initialisation du jeu de yi
    for j in range(len(xi)):
        Lxi.append(np.random.normal(xi[j],uxi[j]))   #tirage au sort d'un jeu de xi
        Lyi.append(np.random.normal(yi[j],uyi[j]))   #tirage au sort d'un jeu de yi
    reglin=np.polyfit(Lxi,Lyi,1)                     #régression linéaire sur le jeu de données

    #tirage au sort de l'ordonnée de départ du report de point
    TirageYreport=np.random.normal(Yreport,uYreport)

    #report de point et stockage de l'abscisse
    LXreport.append((TirageYreport-reglin[1])/reglin[0])

#########################################################################################################################
#Affichage de la moyenne des abscisses après report de point
#Affichage de l'incertitude-type de l'abscisse après report de point
print('Abscisse interpolée moyenne : ',np.average(LXreport))
print('Incertitude-type sur l abscisse interpolée u(Xreport) : ',np.std(LXreport, ddof=1))
```

Résultat :

```
Abscisse interpolée moyenne :  1.3083511309816596e-05
Incertitude-type sur l abscisse interpolée u(Xreport) :  5.536486879184971e-07
```

Pour $y = 1{,}125$ avec $u(y) = 0{,}042$, on trouve $x = 1{,}308.10^{-5}$ avec $u(x) = 0{,}055.10^{-5}$

Annexe n°1 : Méthode des moindres carrés

On cherche ici à comprendre les résultats obtenus lorsque l'on ajuste des données expérimentales avec la méthode des moindres carrés.

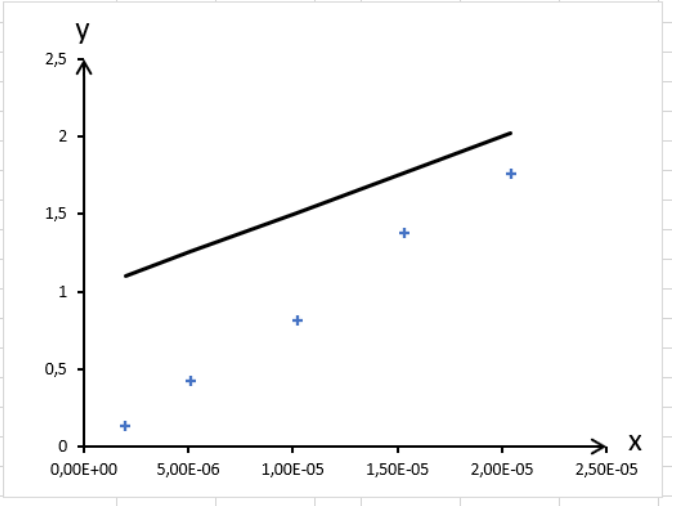
On dispose d'un jeu de données expérimentales (x_i, y_i). On cherche la pente a et l'ordonnée à l'origine b de la droite de régression $y = ax + b$. Les deux paramètres a et b sont ajustés « à la main », puis optimisés par le solveur d'Excel.

Essai n°1

xi	yi	Ydroite	Ecarts	Ecarts²
2,04E-05	1,765	**2,02**	0,255	0,06503
1,53E-05	1,376	**1,765**	0,389	0,15132
1,02E-05	0,813	**1,51**	0,697	0,48581
5,10E-06	0,428	**1,255**	0,827	0,68393
2,00E-06	0,138	**1,1**	0,962	0,92544

pente **a** : 50000 — **Somme à minimiser : 2,31153**

ordonnée à l'origine **b** : 1

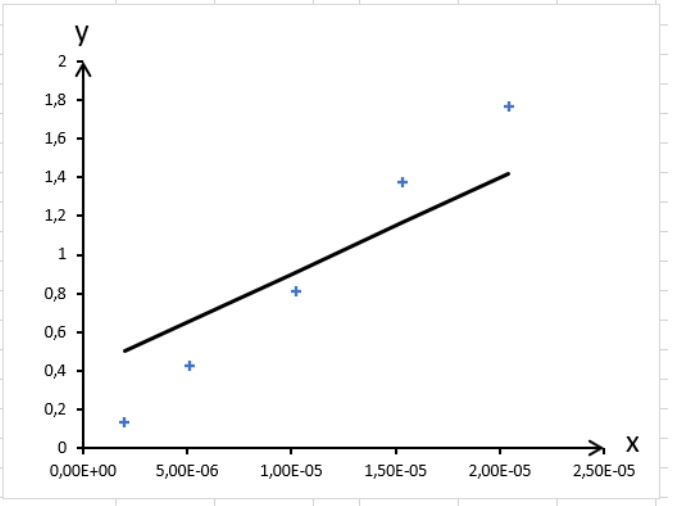


Essai n°2

xi	yi	Ydroite	Ecarts	Ecarts²
2,04E-05	1,765	**1,42**	-0,345	0,11903
1,53E-05	1,376	**1,165**	-0,211	0,04452
1,02E-05	0,813	**0,91**	0,097	0,00941
5,10E-06	0,428	**0,655**	0,227	0,05153
2,00E-06	0,138	**0,5**	0,362	0,13104

pente **a** : 50000 — **Somme à minimiser : 0,35553**

ordonnée à l'origine **b** : 0,4

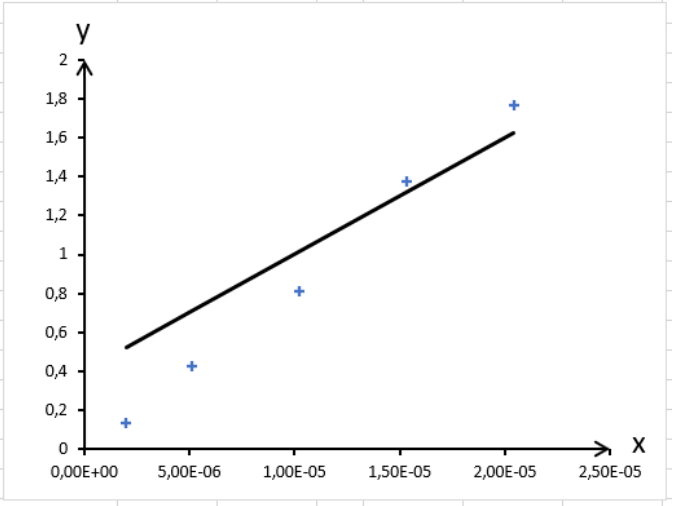


Essai n°3

xi	yi	Ydroite	Ecarts	Ecarts²
2,04E-05	1,765	**1,624**	-0,141	0,01988
1,53E-05	1,376	**1,318**	-0,058	0,00336
1,02E-05	0,813	**1,012**	0,199	0,0396
5,10E-06	0,428	**0,706**	0,278	0,07728
2,00E-06	0,138	**0,52**	0,382	0,14592

pente **a** : 60000 — **Somme à minimiser : 0,28605**

ordonnée à l'origine **b** : 0,4

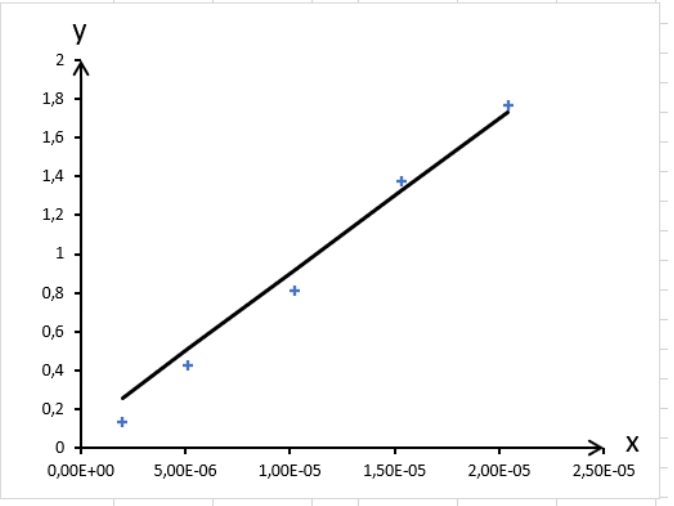


Essai n°4

xi	yi	Ydroite	Ecarts	Ecarts²
2,04E-05	1,765	**1,732**	-0,033	0,00109
1,53E-05	1,376	**1,324**	-0,052	0,0027
1,02E-05	0,813	**0,916**	0,103	0,01061
5,10E-06	0,428	**0,508**	0,08	0,0064
2,00E-06	0,138	**0,26**	0,122	0,01488

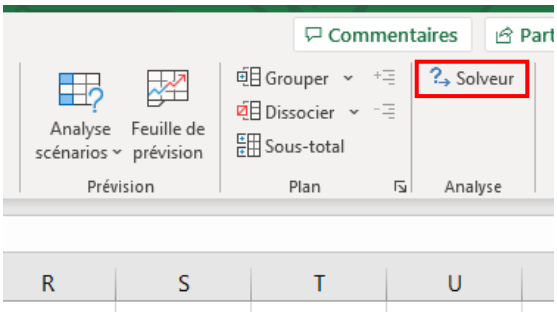
pente **a** : 80000 — **Somme à minimiser : 0,03569**

ordonnée à l'origine **b** : 0,1

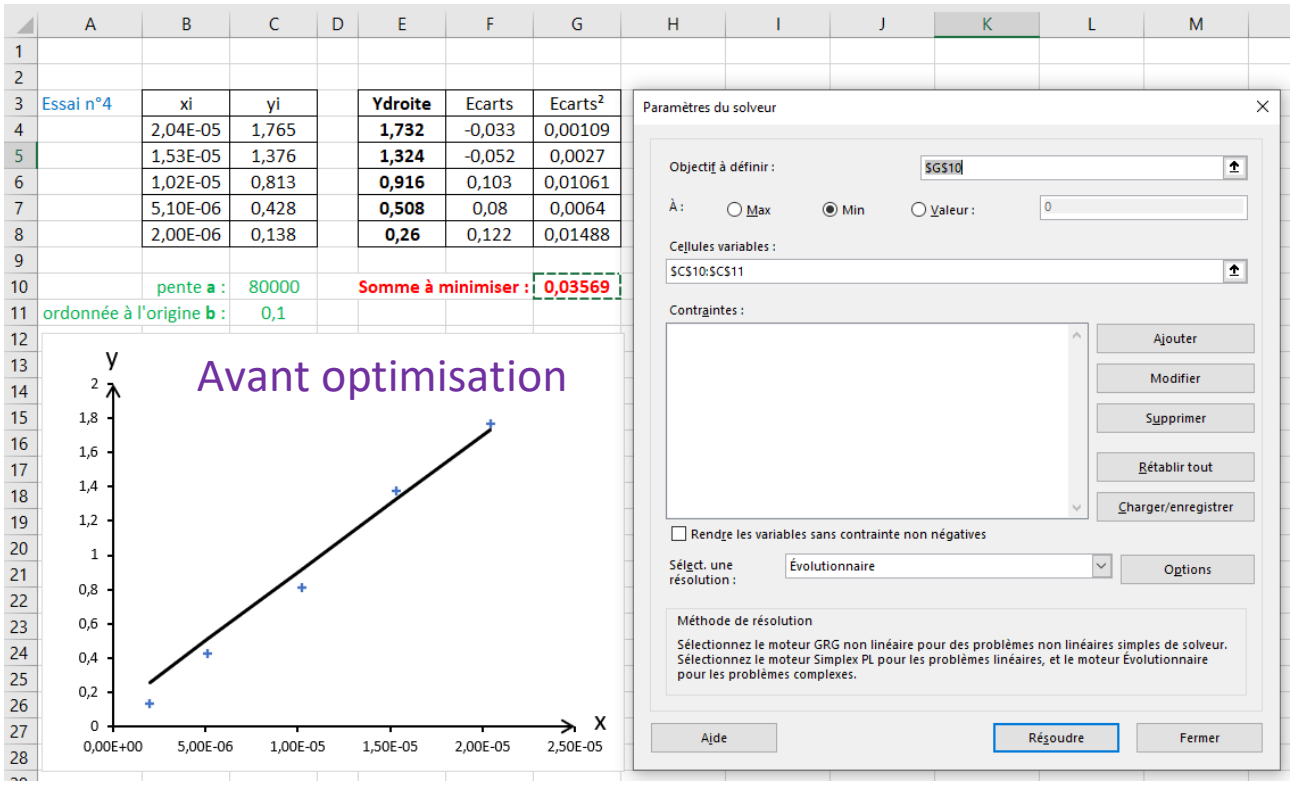


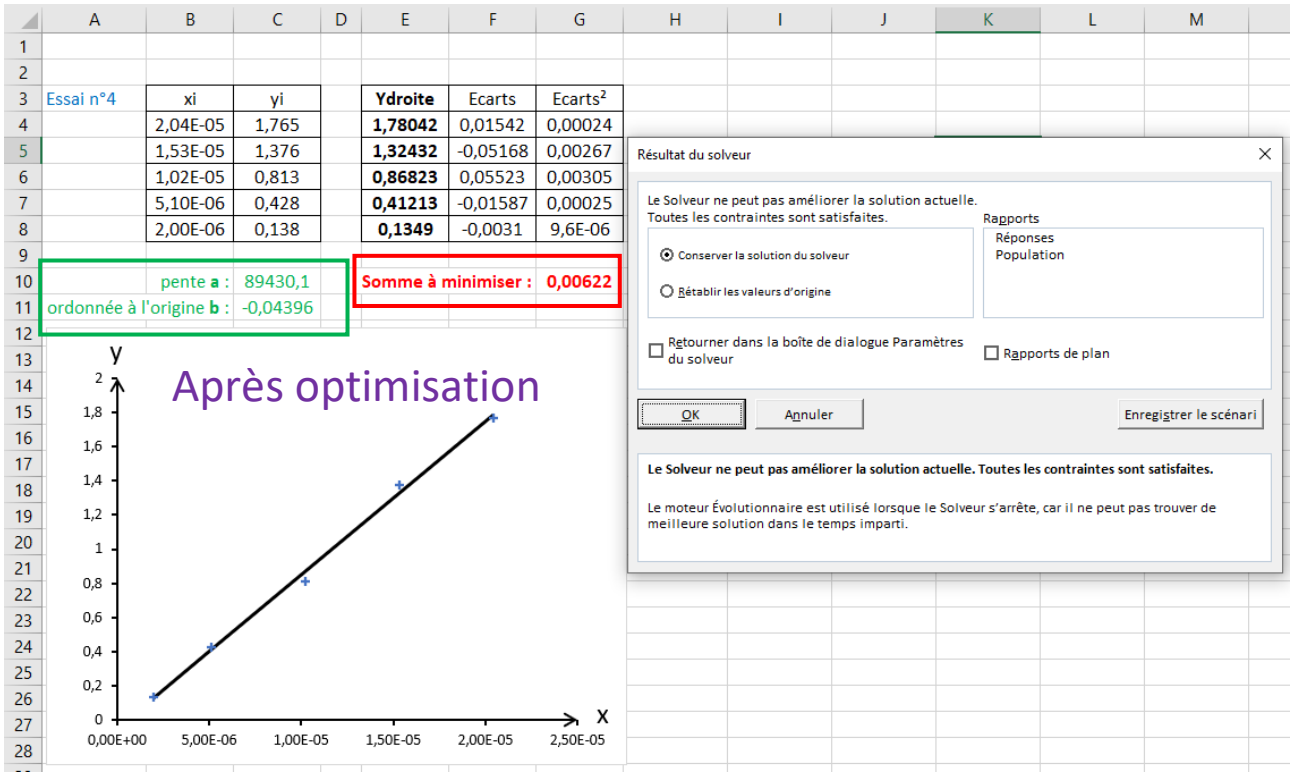
COUOT C. - CHARDON C. - VICTOR M. - BARILERO T.

Optimisation avec le solveur Excel (*Fichier, Options, Compléments, Complément Solveur, atteindre, cocher, OK*)



- Choisir la somme à minimiser comme ***Objectif***
- Cocher la case ***Min***
- Choisir la pente et l'ordonnée à l'origine comme ***Cellules variables***
- Choisir la résolution ***Évolutionnaire***





Annexe n°2 : Tester plusieurs modèles concurrents

Nous présentons ici le cas d'une étude cinétique par spectrophotométrie. On mesure l'absorbance A (incertitude-type relative 2 %) au cours du temps t (en min).

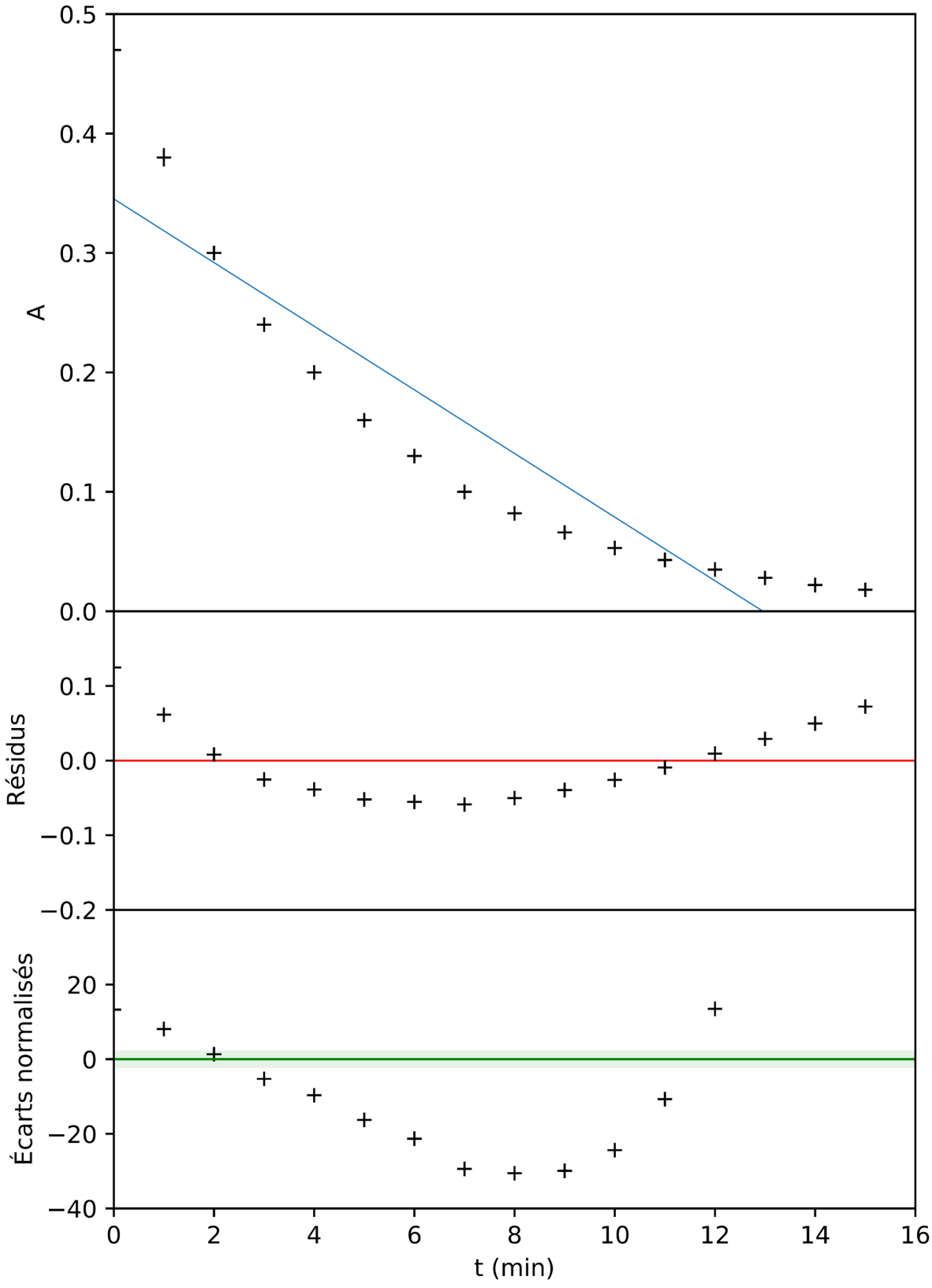
t (min)	0	1	2	3	4	5	6	7	8
A	0,47	0,38	0,30	0,24	0,20	0,16	0,13	0,10	0,082

t (min)	9	10	11	12	13	14	15
A	0,066	0,053	0,043	0,035	0,028	0,022	0,018

On montre que si la réaction est d'ordre 0, alors A(t) est une droite.
Si la réaction est d'ordre 1, alors ln A (t) est une droite.
Si la réaction est d'ordre 2, alors 1/A (t) est une droite.

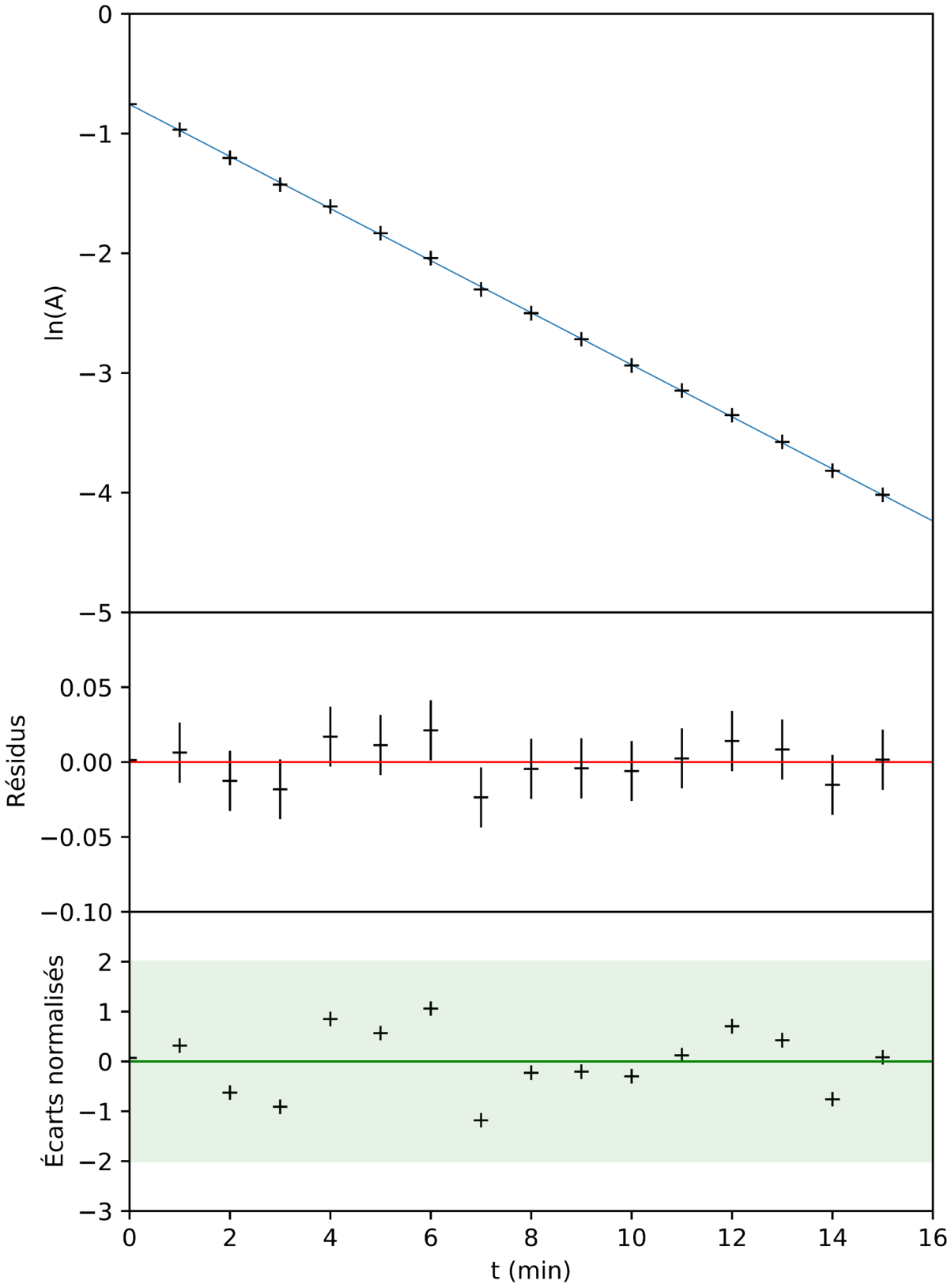
Quel est l'ordre, parmi 0, 1 ou 2, le plus compatible avec les données expérimentales ?

```
import numpy as np
from matplotlib import pyplot

#########################################################################################################################
#Données expérimentales et incertitude-type sur A
t=np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15])

A=np.array([0.47,0.38,0.3,0.24,0.2,0.16,0.13,0.1,0.082,0.066,0.053,0.043,0.035,0.028,0.022,0.018])
uArel=0.02
uA=uArel*A

#########################################################################################################################
#Régression linéaire A(t), calcul des résidus et des écarts normalisés
reglin=np.polyfit(t,A,1)
ydroite=reglin[0]*t+reglin[1]
residus=A-ydroite
ecartsnormalises=residus/uA

#########################################################################################################################
#Tracé des 3 graphiques
fig1=pyplot.figure(1,figsize=(6,9))
gridsize = (4, 1)

ax1=pyplot.subplot2grid(gridsize, (0, 0), colspan=1, rowspan=2)
ax1.plot(np.linspace(0,16,2),reglin[0]*np.linspace(0,16,2)+reglin[1],linewidth=0.5)
ax1.errorbar(t,A,yerr=uA,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax1.axis([0,16, 0, 0.5])
ax1.set_ylabel('A')

ax2=pyplot.subplot2grid(gridsize, (2, 0), sharex=ax1)
ax2.plot(np.linspace(0,16,2),[0,0],linewidth=0.8,color='red')
ax2.errorbar(t,residus,yerr=uA,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax2.axis([0,16, -0.6, 0.6])
ax2.set_ylabel('Résidus')

ax3=pyplot.subplot2grid(gridsize, (3, 0), sharex=ax1)
ax3.scatter(t,ecartsnormalises,marker='+',color='black',linewidth=0.8)
ax3.axis([0,16, -40, 40])
ax3.plot(np.linspace(0,16,2),[0,0],linewidth=0.8,color='green')
ax3.fill_between([0,16],y1=-2,y2=2,color='green', alpha=0.1)
ax3.set_xlabel('t')
ax3.set_ylabel('Écarts normalisés')

fig1.subplots_adjust(hspace=0)
pyplot.setp(ax1.get_xticklabels(), visible=False)
pyplot.setp(ax2.get_xticklabels(), visible=False)
ax2.set_yticks(ax2.get_yticks()[:-1])
ax3.set_yticks(ax3.get_yticks()[:-1])
pyplot.show()
```



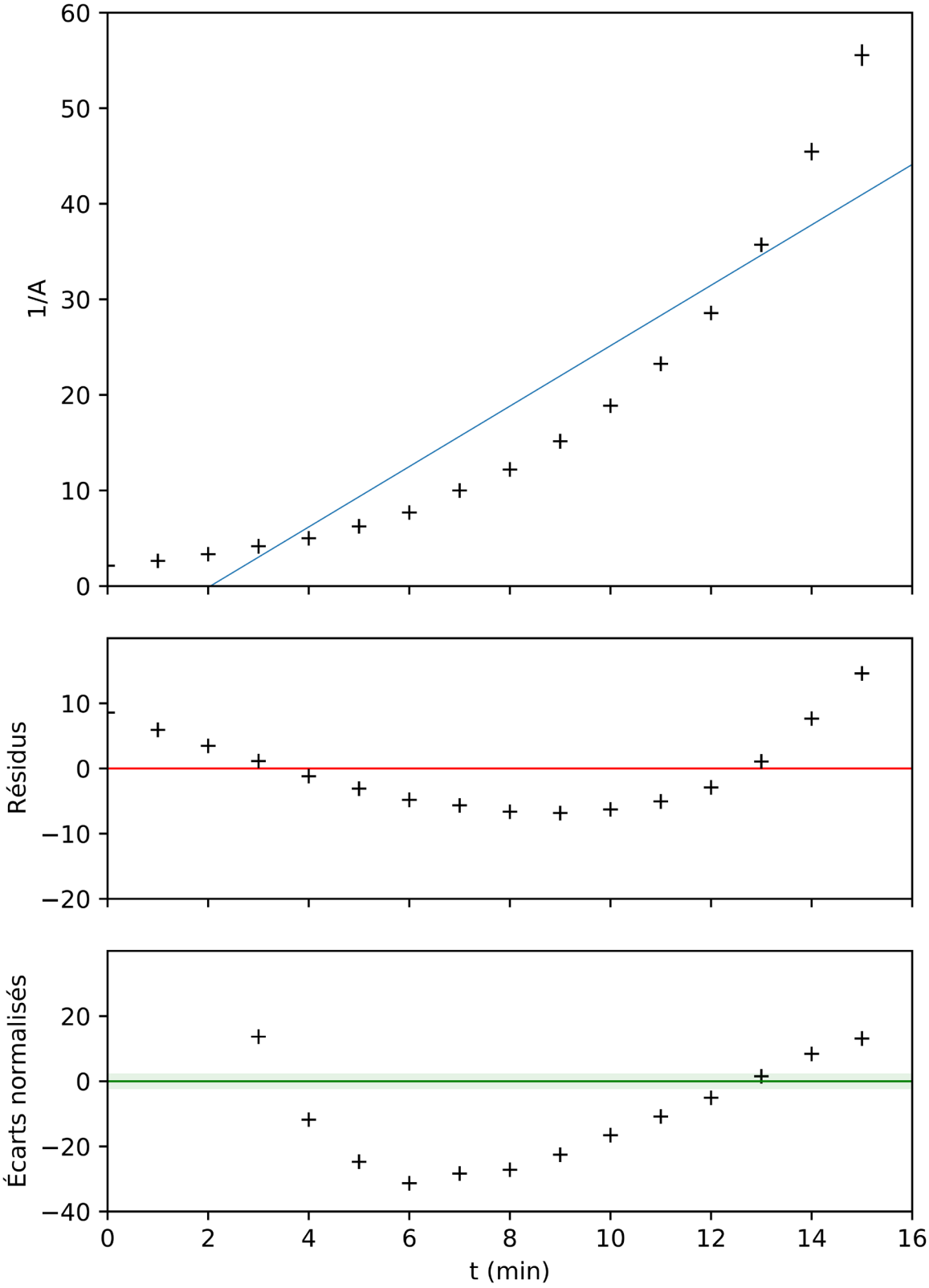
L’ordre 0 n’est pas compatible avec les données expérimentales.

```
#########################################################################################################################
#Calcul de ln(A), Détermination des incertitudes-types de ln(A) par simulations de Monte-Carlo
lnA=np.log(A)
NbSim=100000
ulnA=[]
for j in range(len(A)):
    ulnA.append(np.std(np.log(np.random.normal(A[j],uA[j],NbSim)), ddof=1))

#########################################################################################################################
#Régression linéaire lnA(t), calcul des résidus et des écarts normalisés
reglin=np.polyfit(t,lnA,1)
ydroite=reglin[0]*t+reglin[1]
residus=lnA-ydroite
ecartsnormalises=residus/ulnA

#########################################################################################################################
#Tracé des 3 graphiques
fig2=pyplot.figure(2,figsize=(6,9))
gridsize = (4, 1)

ax1=pyplot.subplot2grid(gridsize, (0, 0), colspan=1, rowspan=2)
ax1.plot(np.linspace(0,16,2),reglin[0]*np.linspace(0,16,2)+reglin[1],linewidth=0.5)
ax1.errorbar(t,lnA,yerr=ulnA,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax1.axis([0,16, -5, 0])
ax1.set_ylabel('ln(A)')

ax2=pyplot.subplot2grid(gridsize, (2, 0), sharex=ax1)
ax2.plot(np.linspace(0,16,2),[0,0],linewidth=0.8,color='red')
ax2.errorbar(t,residus,yerr=ulnA,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax2.axis([0,16, -0.1, 0.1])
ax2.set_ylabel('Résidus')

ax3=pyplot.subplot2grid(gridsize, (3, 0), sharex=ax1)
ax3.scatter(t,ecartsnormalises,marker='+',color='black',linewidth=0.8)
ax3.axis([0,16, -3, 3])
ax3.plot(np.linspace(0,16,2),[0,0],linewidth=0.8,color='green')
ax3.fill_between([0,16],y1=-2,y2=2,color='green', alpha=0.1)
ax3.set_xlabel('t')
ax3.set_ylabel('Écarts normalisés')

fig2.subplots_adjust(hspace=0)
pyplot.setp(ax1.get_xticklabels(), visible=False)
pyplot.setp(ax2.get_xticklabels(), visible=False)
ax2.set_yticks(ax2.get_yticks()[:-1])
ax3.set_yticks(ax3.get_yticks()[:-1])
pyplot.show()
```



L’ordre 1 est compatible avec les données expérimentales.

```
#########################################################################################################################
#Calcul de 1/A, Détermination des incertitudes-types de 1/A par simulations de Monte-Carlo
InvA=1/A
NbSim=100000
uInvA=[]
for j in range(len(A)):
    uInvA.append(np.std(1/(np.random.normal(A[j],uA[j],NbSim)), ddof=1))

#########################################################################################################################
#Régression linéaire lnA(t), calcul des résidus et des écarts normalisés
reglin=np.polyfit(t,InvA,1)
ydroite=reglin[0]*t+reglin[1]
residus=InvA-ydroite
ecartsnormalises=residus/uInvA

#########################################################################################################################
#Tracé des 3 graphiques
fig3=pyplot.figure(3,figsize=(6,9))
gridsize = (4, 1)

ax1=pyplot.subplot2grid(gridsize, (0, 0), colspan=1, rowspan=2)
ax1.plot(np.linspace(0,16,2),reglin[0]*np.linspace(0,16,2)+reglin[1],linewidth=0.5)
ax1.errorbar(t,InvA,yerr=uInvA,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax1.axis([0,16, 0, 60])
ax1.set_ylabel('1/A')

ax2=pyplot.subplot2grid(gridsize, (2, 0), sharex=ax1)
ax2.plot(np.linspace(0,16,2),[0,0],linewidth=0.8,color='red')
ax2.errorbar(t,residus,yerr=uInvA,fmt='+',color='black',markeredgewidth=0.8,linewidth=0.8)
ax2.axis([0,16, -20, 20])
ax2.set_ylabel('Résidus')

ax3=pyplot.subplot2grid(gridsize, (3, 0), sharex=ax1)
ax3.scatter(t,ecartsnormalises,marker='+',color='black',linewidth=0.8)
ax3.axis([0,16, -40, 40])
ax3.plot(np.linspace(0,16,2),[0,0],linewidth=0.8,color='green')
ax3.fill_between([0,16],y1=-2,y2=2,color='green', alpha=0.1)
ax3.set_xlabel('t')
ax3.set_ylabel('Écarts normalisés')

fig2.subplots_adjust(hspace=0)
pyplot.setp(ax1.get_xticklabels(), visible=False)
pyplot.setp(ax2.get_xticklabels(), visible=False)
ax2.set_yticks(ax2.get_yticks()[:-1])
ax3.set_yticks(ax3.get_yticks()[:-1])
pyplot.show()
```



L'ordre 2 n'est pas compatible avec les données expérimentales.

Conclusion : l'ordre le plus compatible avec les données expérimentales est l'ordre 1.