Activité Python

1 Dosage par étalonnage du bleu de patenté dans les stroumpfs

1.1 Protocole

1.1.1 Préparation de la solution mère et des solutions diluées

Préparation de la solution mère avec du bleu de patenté à $4 \times 10^{-5}\,\text{mol}\,\text{L}^{-1}$. On dissout 22,4 mg dans une fiole jaugée de 1 L. On réalise ensuite les dilutions suivantes :

Volume prélevé (mL)	0	5	10	20	30	40	50
Concentration en bleu de patenté ($\times 10^{-5}\,\text{mol}\,\text{L}^{-1}$)	0	0,20	0,40	0,80	1,20	1,60	2

1.1.2 Tracé de la courbe d'étalonnage

On mesure les absorbances de chaque solution. On mesure l'absorbance au maximum d'absorption du bleu de patenté qui est $\lambda_{max} = 640\,\text{nm}$.
On rentre les mesures dans des listes python.

```
1
2  # -*- coding: utf-8 -*-
3
4  # Import de la librairie matplotlib
5  import matplotlib.pyplot as plt
6  import numpy as np
7
8  # Liste des valeurs des concentrations filles
9  C = [0, 0.20, 0.40, 0.80, 1.20, 1.60, 2]
10
11 # Liste des valeurs d'absorbance
12 A = [0, 0.182, 0.369, 0.744, 1.115, 1.449, 1.846]
```

On trace ensuite le graphique. Voir Figure 1.

```
15 # Tracé du graphe
16 plt.figure()
17 plt.plot(C,A,'.')
18 plt.xlabel('$x 10^{-5}}$ mol/L')
19 plt.ylabel("A")
20 plt.title("A = f(C)")
21 plt.grid()
22 plt.show()
```

1.1.3 Régression linéaire

Les points étant alignés, on va déterminer le coefficient d'absorption molaire du bleu patenté par régression linéaire. Listing complet du programme :

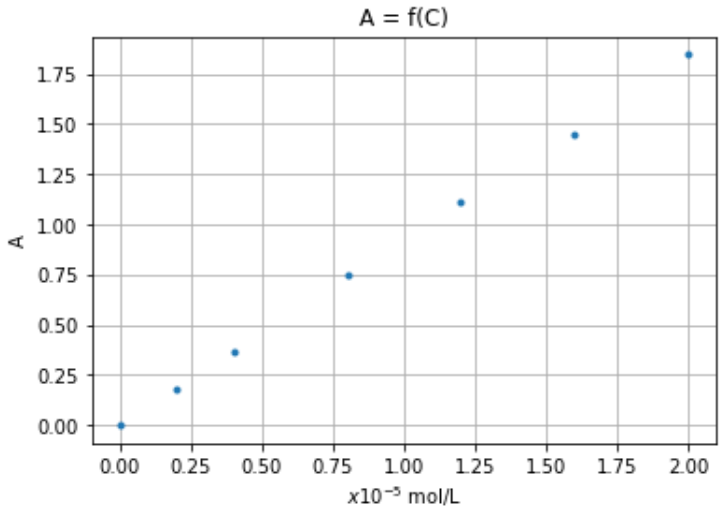


FIGURE 1 – Points expérimentaux

```python
# -*- coding: utf-8 -*-


import matplotlib.pyplot as plt
import numpy as np


# Concentrations en 10^-5 mol/L
C = np.array([0, 0.2, 0.4, 0.8, 1.2, 1.6, 2])


# Absorbance des des solutions filles
A = np.array([0, 0.182, 0.369, 0.744, 1.115, 1.449, 1.846])
# Tracé du graphe
#-----------------------------------------------------
plt.figure()
plt.scatter(C,A, label="Mesures expérimentales",color="blue")
plt.xlabel('$x 10^{-5}}$ mol/L')
plt.ylabel("A")
plt.title("A = f(C)")
plt.grid()


# Régression linéaire
#-----------------------------------------------------
P = np.polyfit(C,A,1)
poly = np.poly1d(P)
print(poly)
#print(poly(0.5))

# Tracé de la droite de régression:
#-----------------------------------------------------

plt.plot(C,P[0]*C+P[1], label="Droite de régression",color='green')
plt.legend()
plt.text(1,0.60,f'A = {P[0]:.4} C + {P[1]:.3}')

plt.show()
```

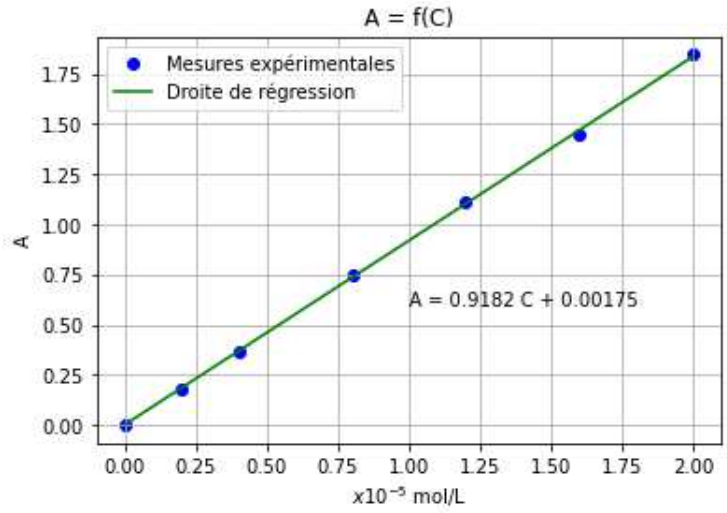


FIGURE 2 – Régression linéaire

1.1.4 Préparation de la solution en concentration inconnue

Découper 5 schtroumps en petits morceaux. Ne garder que les parties bleues. On verse les bonbons dans à peu près 50 mL d'eau et on fait chauffer jusqu'à dilution. On filtre et on récupère le filtrat dans une fiole jaugée de 100 mL que l'on complète jusqu'au trait de jauge.
On mesure l'absorption de cette solution S et on en déduit sa concentration en utilisant la loi de Beer-Lambert.

2 Cinétique chimique

On obtient les mesures suivantes :

t (min)	0	1	2	3	4	5	6	7	8	9	10
A	1.417	1.161	0.982	0.843	0.722	0.622	0.538	0.469	0.401	0.342	0.293
t (min)	11	12	13	14	15	16	17	18	19	20	
A	0.252	0.216	0.185	0.159	0.137	0.118	0.101	0.087	0.075	0.065	

3 Traitemet des données

3.1 On ouvre un interpréteur python : spyder par exemple

Dans un nouveau fichier, on entre les données expérimentales dans des listes.

```
15
16 # -*- coding: utf-8 -*-
17
18 # 'liste des valeurs du temps en minutes
19 t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
20
21 # 'liste des valeurs absorbance
22 A = [1.417, 1.161, 0.982, 0.843, 0.722, 0.622, 0.538, 0.469, 0.401, 0.342, 0.293, 0.252,
    0.216, 0.185, 0.159, 0.137, 0.118, 0.101, 0.087, 0.075, 0.065]
```

3.2 On importe la librairie matplotlib pour le graphique et numpy pour le calcul du logarithme

```
15
16 # -*- coding: utf-8 -*-
17
18 # Import de la librairie matplotlib
19 import matplotlib.pyplot as plt
20 import numpy as np
21
22 # Liste des valeurs du temps en minutes
23 t = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
24
25 # Liste des valeurs absorbance
26 A = [1.417, 1.161, 0.982, 0.843, 0.722, 0.622, 0.538, 0.469, 0.401, 0.342, 0.293, 0.252,
    0.216, 0.185, 0.159, 0.137, 0.118, 0.101, 0.087, 0.075, 0.065]
```

3.3 Tracé de A = f(t)

```
13 # Tracé du graphe
14 plt.figure()
15 plt.plot(t,A)
16 plt.show()
```

Il faut ajouter un titre, remplacer la ligne par des points, labéliser les axes et améliorer la lisibilité avec une grille. Cela donne :

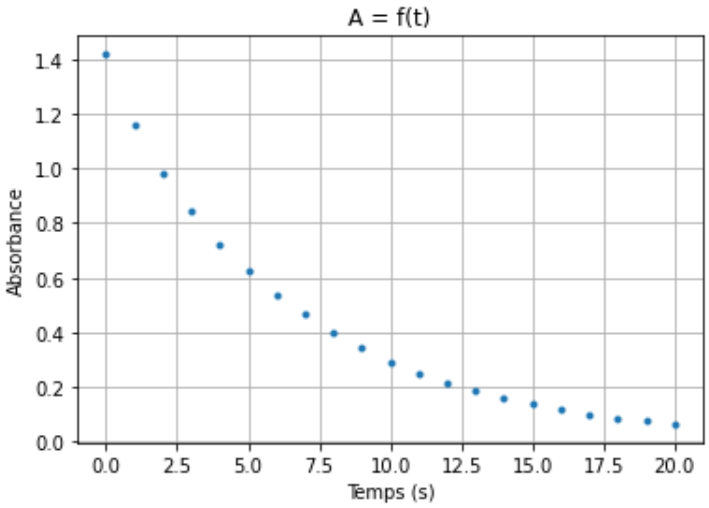
```
13 # Tracé du graphe
14 plt.figure()
15 plt.plot(t,A,'.')
16 plt.xlabel('Temps (s)')
17 plt.ylabel("Absorbance")
18 plt.title("A = f(t)")
19 plt.grid()
20 plt.show()
```

3.4 Vérification de l'ordre de la réaction

Si la réaction est d'ordre 1 alors $ln\left(\frac{A}{A_0}\right)$ est une fonction affine.

On crée donc une nouvelle liste comprenant le logarithme de chaque terme de la liste A. On appelle cette nouvelle liste par exemple lnA :

```
23 # Tracé de ln(A/A0)
24
25 lnA = [np.log(x/A[0]) for x in A]
26
27 plt.figure(2)
```




```
28 plt.plot(t,lnA,'.')
29 plt.xlabel('Temps (s)')
30 plt.ylabel("ln(A/A0")
31 plt.title("ln(A/A0) = f(t)")
32 plt.grid()
33 plt.show()
```

python_graphique.py