

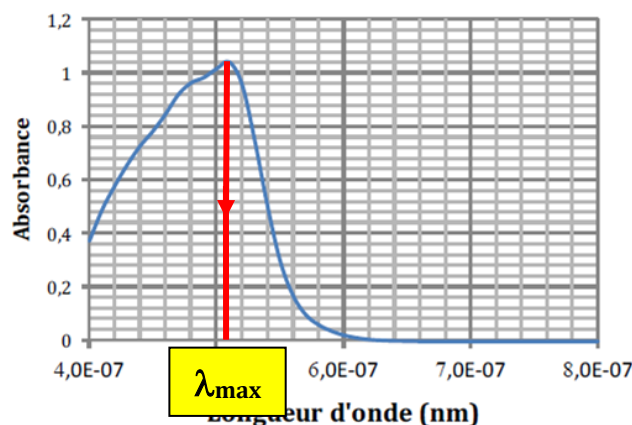
## Suivi cinétique d'une réaction par spectrophotométrie - CORRIGE

- 1- La température est un facteur cinétique, c'est-à-dire qu'elle a une influence sur la vitesse de réaction, notamment sur la valeur de la constante de vitesse.

Afin que cette constante de vitesse soit constante, on se place donc à température constante.

- 2- Une étude spectrophotométrique a toujours lieu à la longueur d'onde  $\lambda_{\max}$  qui est la plus absorbée par l'espèce chimique.

Ici, on note que la ferroïne absorbe le plus les radiations de longueur d'onde  $\lambda_{\max} = 5,1 \cdot 10^{-7} \text{ m}$  (510 nm). C'est donc à cette longueur d'onde qu'on réalisera le suivi.



- 3- Pour une température de 43 °C, les résultats sont les suivants :

t(s)	120	250	420	600	780	960	1140	1320	1500	1680	1860
A	0,981	0,840	0,680	0,556	0,448	0,338	0,301	0,231	0,162	0,140	0,112

- 4- Par définition,  $v = k \times [\text{Ferroïne}]^a \times [\text{H}_3\text{O}^+]^b$

- 5- • Concentration initiale en ferroïne dans le mélange :

$$[\text{Ferroïne}]_0 = \frac{n(\text{Ferroïne})_0}{V_{\text{mélange}}} \quad \text{soit}$$

$$[\text{Ferroïne}]_0 = \frac{C_1 \times V_1}{V_1 + V_2}$$

$$\text{AN} \rightarrow [\text{Ferroïne}]_0 = \frac{1,00 \cdot 10^{-3} \times 10,0 \cdot 10^{-3}}{10,0 \cdot 10^{-3} + 90 \cdot 10^{-3}} \quad \text{soit} \quad [\text{Ferroïne}]_0 = 1,0 \cdot 10^{-4} \text{ mol.L}^{-1}$$

- Concentration initiale en oxonium dans le mélange :

$$[\text{H}_3\text{O}^+]_0 = \frac{n(\text{H}_3\text{O}^+)_0}{V_{\text{mélange}}} \quad \text{soit}$$

$$[\text{H}_3\text{O}^+]_0 = \frac{C_2 \times V_2}{V_1 + V_2}$$

$$\text{AN} \rightarrow [\text{H}_3\text{O}^+]_0 = \frac{4,0 \times 90 \cdot 10^{-3}}{10,0 \cdot 10^{-3} + 90 \cdot 10^{-3}} \quad \text{soit} \quad [\text{H}_3\text{O}^+]_0 = 3,6 \text{ mol.L}^{-1}$$

**CONCLUSION :**  $\text{H}_3\text{O}^+$  en excès par rapport à la ferroïne. On peut donc considérer que la concentration en  $\text{H}_3\text{O}^+$  restera constante : méthode de dégénérescence de l'ordre.

- 6- A chaque instant, on peut considérer que  $[\text{H}_3\text{O}^+] = [\text{H}_3\text{O}^+]_0 = \text{Constante}$ .

La loi de vitesse établie à la question 3- devient donc :

$$v = k \times [\text{Ferroïne}]^a \times [\text{H}_3\text{O}^+]^b \quad \leftrightarrow \quad v = k \times [\text{Ferroïne}]^a \times [\text{H}_3\text{O}^+]_0^b$$

$$\text{Soit} \quad v = k_{\text{APP}} \times [\text{Ferroïne}]^\alpha \quad \text{avec } \alpha = a \text{ et } k_{\text{APP}} = k \times [\text{H}_3\text{O}^+]_0^b$$

- 7- Si on connaissait les valeurs de  $[\text{Ferroïne}]$  au cours du temps, les graphiques à tracer seraient :

- pour valider un ordre  $\alpha = 0$ , on aurait :  $[\text{Ferroïne}] = -k_{\text{app}} \cdot t + [\text{Ferroïne}]_0$

➡ on trace donc le graphique  $[\text{Ferroïne}] = f(t)$  ;

- pour valider un ordre  $\alpha = 1$ , on aurait :  $\ln[\text{Ferroïne}] = -k_{\text{app}} \cdot t + \ln[\text{Ferroïne}]_0$

➡ on trace donc le graphique  $\ln[\text{Ferroïne}] = f(t)$  ;

- pour valider un ordre  $\alpha = 2$ , on aurait :  $1 / [\text{Ferroïne}] = k_{\text{app}} \cdot t + 1 / [\text{Ferroïne}]_0$

➡ on trace donc le graphique  $1 / [\text{Ferroïne}] = f(t)$  ;

- 8- D'après la loi de Beer-Lambert, l'absorbance mesurée au cours de l'étude est proportionnelle à la concentration molaire en ferroïne selon la relation  $A = \epsilon \times L \times [\text{Ferroïne}]$  avec :

-  $\epsilon$  = Coefficient d'absorption molaire de la ferroïne à 25 °C pour  $\lambda = 510 \text{ nm}$  ;

-  $L$  = Epaisseur de la cuve spectrophotométrique contenant le mélange réactionnel.

Ainsi, les relations précédentes deviennent :

- pour valider un ordre  $\alpha = 0$ , on aurait :  $A / (\varepsilon \times L) = -k_{app}.t + A_0 / (\varepsilon \times L)$   
Soit  $A = -\varepsilon \times L \times k_{app}.t + A_0$   
➔ on trace donc le **graphique  $A = f(t)$**  ;
- pour valider un ordre  $\alpha = 1$ , on aurait :  $\ln[A / (\varepsilon \times L)] = -k_{app}.t + \ln[A_0 / (\varepsilon \times L)]$   
Soit  $\ln(A) = -k_{app}.t + \ln(A_0)$   
➔ on trace donc le **graphique  $\ln(A) = f(t)$**  ;
- pour valider un ordre  $\alpha = 2$ , on aurait :  $\varepsilon \times L / A = k_{app}.t + \varepsilon \times L / A_0$   
Soit  $1/A = k_{app}.t / (\varepsilon \times L) + 1/A_0$   
➔ on trace donc le **graphique  $1/A = f(t)$**  ;

#### # Importation des bibliothèques utiles :


```
import numpy as np      # Pour faire des calculs, des tableaux
import matplotlib.pyplot as plt  # Pour tracer des graphiques
```

#### # Valeurs expérimentales :

```
t = np.array([120,250,420,600,780,960,1140,1320,1500,1680,1860]) # Tableau des différentes dates d'étude (en s)
A = np.array([0.981,0.840,0.680,0.556,0.448,0.338,0.281,0.231,0.186,0.140,0.112]) # Tableau des différentes absorbances (sans unité)
```

#### # Grandeurs qui seront portées sur les axes des ordonnées :

```
y0 = A          # Grandeur qui sera portée en ordonnée du graphique pour un ordre 0
y1 = np.log(A)  # Grandeur qui sera portée en ordonnée du graphique pour un ordre 1
y2 = 1/A        # Grandeur qui sera portée en ordonnée du graphique pour un ordre 2
```

 La valeur de  $\ln(x)$  se calcule avec la fonction `np.log(x)`

#### # Tracé des 3 graphiques pour tester les ordres 0, 1 et 2 :

```
plt.figure(figsize=(20,6)) # Taille de la zone de graphique

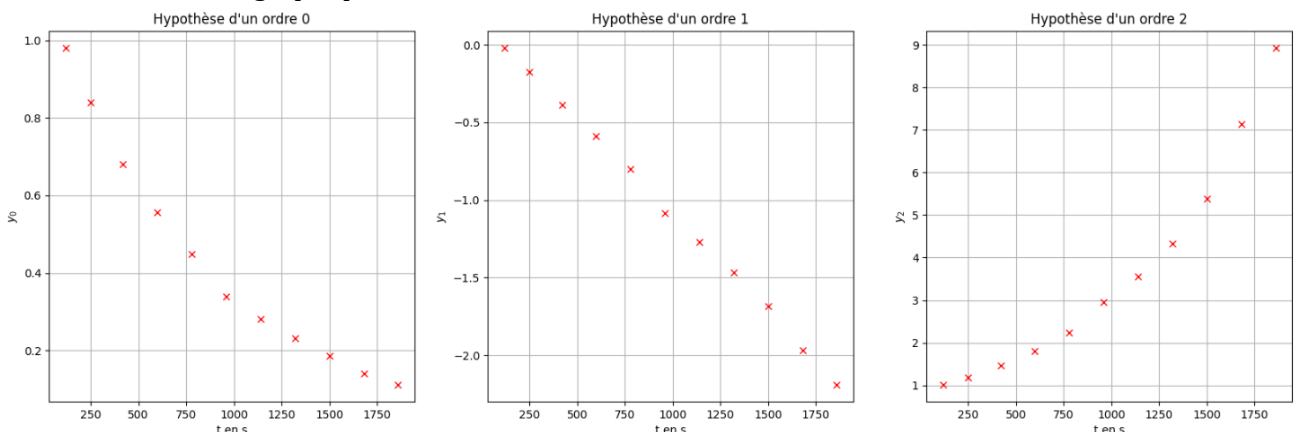
plt.subplot(1,3,1)         # Paramétrage du graphique testant l'ordre 0
plt.plot(t,y0,'rx')
plt.xlabel('t en s')
plt.ylabel('$y_0$')
plt.title("Hypothèse d'un ordre 0")
plt.grid()

plt.subplot(1,3,2)         # Paramétrage du graphique testant l'ordre 1
plt.plot(t,y1,'rx')
plt.xlabel('t en s')
plt.ylabel('$y_1$')
plt.title("Hypothèse d'un ordre 1")
plt.grid()

plt.subplot(1,3,3)         # Paramétrage du graphique testant l'ordre 2
plt.plot(t,y2,'rx')
plt.xlabel('t en s')
plt.ylabel('$y_2$')
plt.title("Hypothèse d'un ordre 2")
plt.grid()
plt.show()
```

8bis-


On obtient les graphiques suivants :



Clairement, c'est **l'hypothèse d'ordre 1** qui semble validée puisque les points expérimentaux semblent vérifier un modèle affine.

#### # Régression linéaire :

```
p = np.polyfit(t, y1, 1) # Régression linéaire de y1 en fonction de x
y_reg = p[0]*t + p[1]    # Equation de la droite de régression, avec p[0] le coefficient directeur
                           # et p[1] l'ordonnée à l'origine
```

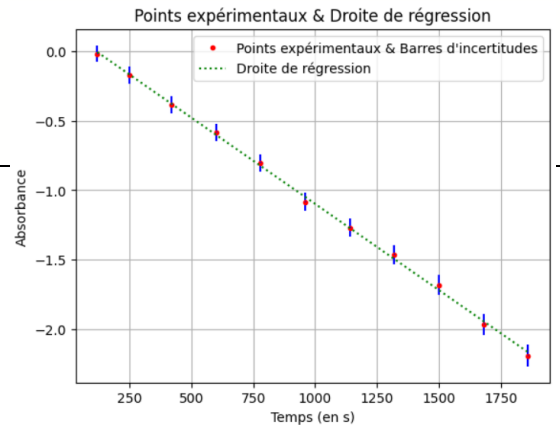
 Pour modéliser un ensemble de points expérimentaux  $(x_i, y_i)$  par une **droite de régression linéaire**, on utilise la fonction `np.polyfit(x, y, 1)`

#### # Incertitude sur la valeur de l'ordonnée de chaque point expérimental :

```
u_A = (5/100*A + 0.002)/np.sqrt(3) # Expression de l'incertitude u(A) sur l'absorbance de chaque point expérimental (voir notice du spectro)
u_y = u_A/A # Expression de l'incertitude u(y) sur la valeur de l'ordonnée de chaque point expérimental du graphique retenu
```

#### # Superposition de la Droite de régression & des Points expérimentaux avec leur barre d'incertitude verticale :

```
plt.errorbar(t, y1, yerr = 2*u_y,fmt='b,') # Représentation des barres d'erreurs verticales pour chaque point expérimental du graphique retenu
plt.plot(t,y1,'r.', label = "Points expérimentaux & Barres d'incertitudes") # Représentation des points expérimentaux y1 = f(x) avec des points rouges
plt.plot(t,y_reg,'g-', label = "Droite de régression") # Représentation de la droite de régression y_reg = f(x) en pointillés verts
plt.xlabel('Temps (en s)') # Titre de l'axe des abscisses
plt.ylabel('Absorbance') # Titre de l'axe des ordonnées
plt.title('Points expérimentaux & Droite de régression') # Titre du graphique
plt.grid() # Commande pour afficher une grille
plt.legend() # Commande pour afficher une légende
plt.show() # Commande pour afficher le graphique
```



9- On obtient le graphique représenté ci-contre sur lequel on constate que **chaque barre verticale d'incertitude coupe la droite de régression linéaire**.

➔ On en déduit donc que la régression linéaire valide le modèle affine.

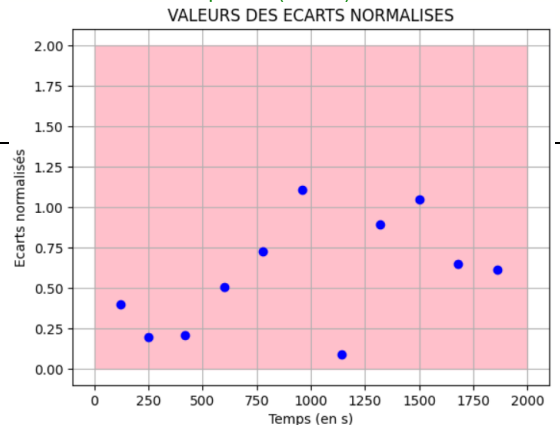
# Formule qui calcule l'écart normalisé entre y1 et y\_reg pour chaque point expérimental :

```
EN = np.abs(y1 - y_reg) / u_y
```

# Représentation graphique des écarts normalisés de chaque point expérimental :

```
plt.plot(t,EN,'bo') # Représentation par un point bleu de l'écart normalisé pour chaque point expérimental
plt.fill_between([0,2000],[2,2],color='pink') # Trame de fond rose associée au domaine des écarts normalisés acceptables (EN < 2)
plt.xlabel('Temps (en s)') # Titre de l'axe des abscisses
plt.ylabel('Ecart normalisé') # Titre de l'axe des ordonnées
plt.title('VALEURS DES ECARTS NORMALISES') # Titre du graphique
plt.grid() # Affichage d'une grille
plt.show() # Affichage du graphique
```

python La valeur absolue de x se calcule avec la fonction np.abs(x)



10- On obtient le graphique représenté ci-contre sur lequel on constate que **chaque point expérimental est caractérisé par un écart normalisé inférieur à 2**.

➔ On en déduit donc que la régression linéaire valide le modèle affine.

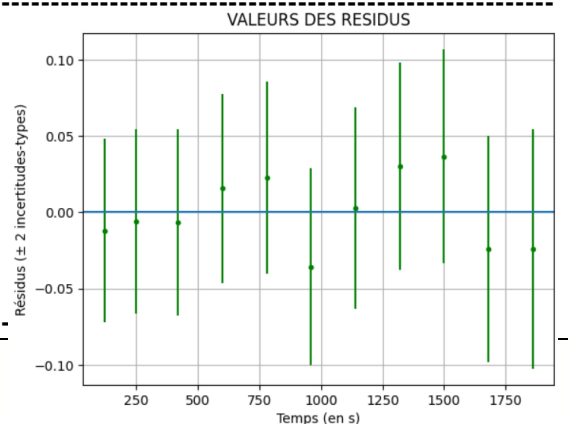
#### Méthode des RESIDUS :

On obtient le graphique représenté ci-contre sur lequel on constate que **chaque point expérimental est caractérisé par une barre verticale de résidu qui touche la droite d'équation « Résidu = 0 »**.

➔ On en déduit donc que la régression linéaire valide le modèle affine.

# Affichage des paramètres de la régression linéaire :

```
print(f"Coefficient directeur : {p[0]} (en s^-1)") # Affichage du coeff directeur de y_reg = f(t)
print(f"Ordonnée à l'origine : {p[1]} (sans unité)") # Affichage de l'ord. à l'origine de y_reg = f(t)
```



11- L'exécution du programme conduit aux résultats ci-dessous :

Coefficient directeur : -0.0012401789646437472 (en s<sup>-1</sup>)  
Ordonnée à l'origine : 0.141557204050376 (sans unité)

12- Par identification, le coefficient directeur de la droite  $\ln(A) = f(t)$  vaut  $-k_{app}$ .  
On en déduit donc que  $k_{app} = 1,2 \cdot 10^{-3} \text{ s}^{-1}$  pour une température de 43 °C.

13- Les différentes valeurs de  $\{T ; k_{app}\}$  de la classe sont les suivantes :

T (en °C)	25	30	35	40	45	50	55
k (en s <sup>-1</sup> )	$7,5 \cdot 10^{-5}$	$1,5 \cdot 10^{-4}$	$3,3 \cdot 10^{-4}$	$7,2 \cdot 10^{-4}$	$1,5 \cdot 10^{-3}$	$3,1 \cdot 10^{-3}$	$6,3 \cdot 10^{-3}$

14- Si la constante de vitesse  $k_{app}$  de cette réaction vérifie la loi d'Arrhenius, alors :

$$k_{app} = A \times \exp(-E_a / (RT))$$

On peut linéariser cette relation en la faisant passer au « ln », ce qui donne :

$$\ln(k_{app}) = \ln(A) - \frac{E_a}{RT}$$

Pour le vérifier, on trace alors le graphique  $\ln(k_{app}) = f(1/T)$ .

#### ► Tracé d'un graphique pour valider la loi d'Arrhenius

##### # Valeurs expérimentales :

```
T = np.array([25,30,35,40,45,50,55])
```

# Tableau des différentes températures d'étude (en °C)

```
kapp = np.array([7.5e-5, 1.5e-4, 3.3e-4, 7.2e-4, 1.5e-3, 3.1e-3, 6.3e-3])
```

# Tableau des différentes constantes de vitesse apparentes (en s<sup>-1</sup>)

##### # Grandeurs qui seront portées sur les deux axes du graphique pour vérifier la loi d'Arrhenius :

```
x = 1/(T+273.15)
```

# Grandeur qui sera portée en abscisse du graphique

```
y = np.log(kapp)
```

# Grandeur qui sera portée en ordonnée du graphique

##### # Tracé du graphique permettant de vérifier la loi d'Arrhenius :

```
plt.plot(x,y,'rx')
```

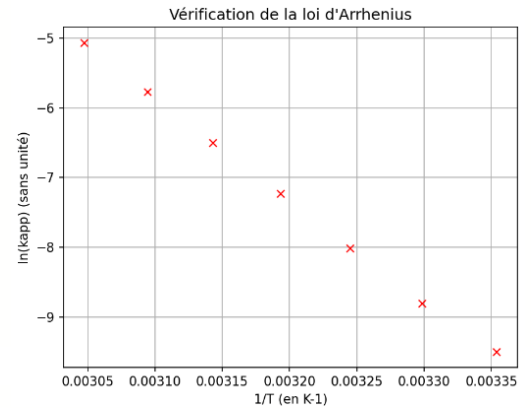
```
plt.xlabel('1/T (en K-1)')
```

```
plt.ylabel('ln(kapp) (sans unité)')
```

```
plt.title('Vérification de la loi d'Arrhenius')
```

```
plt.grid()
```

```
plt.show()
```



#### ► Modélisation du graphique par une régression linéaire

##### # Régression linéaire :

```
p = np.polyfit(x, y, 1)
```

# Régression linéaire de ln(kapp) en fonction de 1/T

```
y_reg2 = p[0]*x + p[1]
```

# Equation de la droite de régression, avec p[0] le coefficient directeur et p[1] l'ordonnée à l'origine

##### # Superposition de la Droite de régression & des Points expérimentaux :

```
plt.plot(x,y,'r+', label = "Points expérimentaux")
```

# Représentation des points expérimentaux ln(kapp) = f(1/T) avec des croix rouges

```
plt.plot(x,y_reg2,'g-', label = "Droite de régression")
```

# Représentation de la droite de régression y\_reg2 = f(1/T) en pointillés verts

```
plt.xlabel('1/T (en K-1)')
```

# Titre de l'axe des abscisses

```
plt.ylabel('ln(kapp) (sans unité)')
```

# Titre de l'axe des ordonnées

```
plt.title('Points expérimentaux & Droite de régression')
```

# Titre du graphique

```
plt.grid()
```

# Commande pour afficher une grille

```
plt.legend()
```

# Commande pour afficher une légende

```
plt.show()
```

# Commande pour afficher le graphique

#### ► Exploitation des coefficients de modélisation

##### # Affichage des paramètres de la régression linéaire :

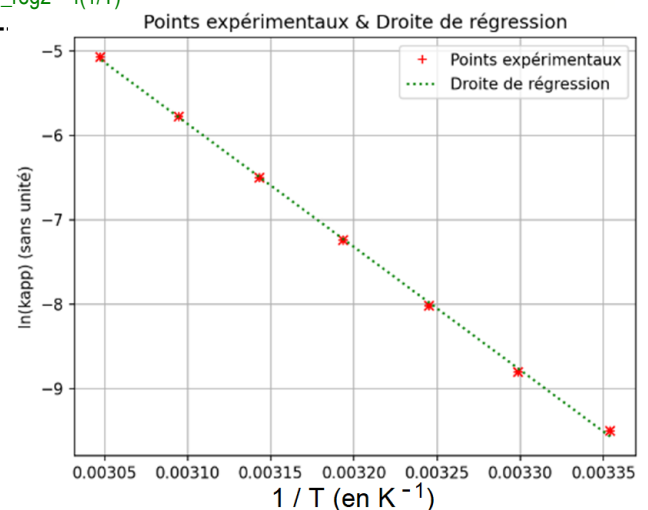
```
print("Coefficient directeur : {p[0]} (en K)")
```

# Affichage du coefficient directeur de y\_reg2 = f(1/T)

```
print("Ordonnée à l'origine : {p[1]} (sans unité)")
```

# Affichage de l'ordonnée à l'origine de y\_reg2 = f(1/T)

15- Les points expérimentaux semblent bien alignés sur la droite de régression linéaire : **la loi d'Arrhenius est donc validée.**



16- L'exécution du programme conduit aux résultats ci-dessous :

Coefficient directeur : -14580.16127647739 (en K)

Ordonnée à l'origine : 39.33560426541311 (sans unité)

17- Le coefficient directeur de la régression linéaire vaut  $-1,5 \cdot 10^4 \text{ K}$  et il s'identifie à  $-E_a / R$  donc  $E_a = 1,46 \cdot 10^4 \times R = 1,46 \cdot 10^4 \times 8,314$  soit  **$E_a = 1,2 \cdot 10^5 \text{ J.mol}^{-1}$**