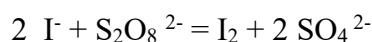


## **Annexe :**

**Protocoles et références  
Scripts Python**

### Exemple 1 : oxydation des ions iodures par les ions peroxodisulfate



#### Protocole :

Dans un bécher, on introduit 25 mL d'iodure de potassium à 0,50 mol/L et 70 mL d'eau distillée. A  $t=0\text{s}$  (déclenchement du chronomètre), on ajoute 5 mL de peroxodisulfate d'ammonium à 0,040 mol/L.

On réalise toutes les 5 min, un prélèvement de 5mL dilué immédiatement dans 45 mL d'eau distillée.

On réalise le dosage de la solution par une solution de thiosulfate de sodium à  $10^{-3}$  mol/L.

On note  $V_{eq}$ .

#### Résultats obtenus :

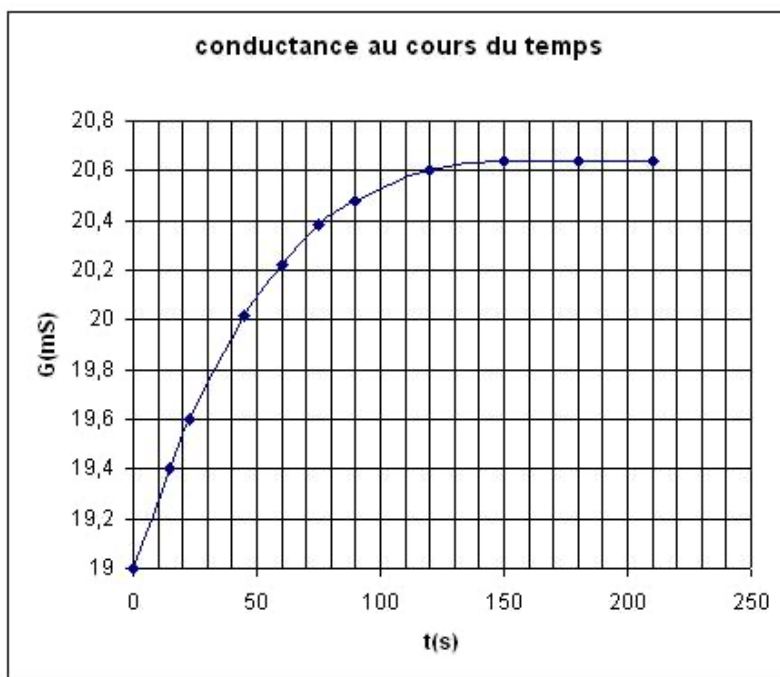
t (min)	Veq(mL)
0	0
5	2,2
10	4,7
15	6,4
20	8,4
25	9,8
30	10,7
35	12,2
40	13,5
45	14,3
50	15
1000	17,6

#### Remarques :

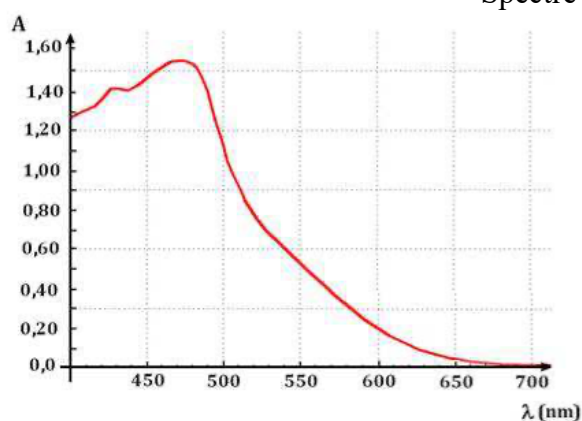
- Cette réaction peut également être suivie par conductimétrie.

Dans un bécher, on introduit un volume  $V_1 = 40 \text{ mL}$  d'une solution aqueuse de peroxodisulfate de potassium ( $2\text{K}^+ + \text{S}_2\text{O}_8^{2-}$ ) de concentration  $C_1 = 1,0 \cdot 10^{-1} \text{ mol. L}^{-1}$ . À l'instant  $t = 0 \text{ s}$ , on ajoute un volume  $V_2 = 60 \text{ mL}$  d'une solution aqueuse d'iodure de potassium ( $\text{K}^+ + \text{I}^-$ ) de concentration  $C_2 = 1,5 \cdot 10^{-1} \text{ mol. L}^{-1}$ . L'expérience est conduite à la température  $T_1$  de  $20^\circ\text{C}$ .

Un conductimètre, permet de suivre l'évolution de la conductance de la solution au cours du temps. La courbe obtenue est reproduite ci-après.



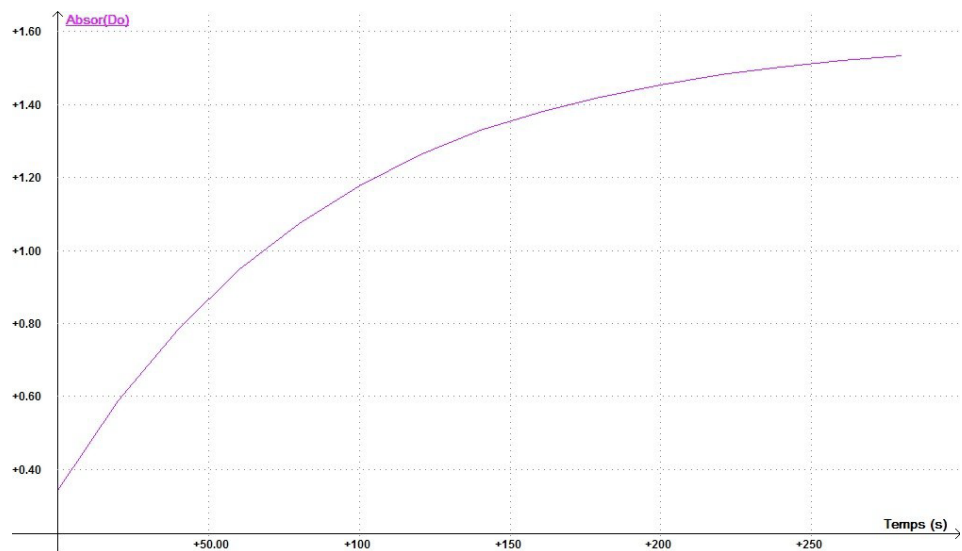
- Cette réaction peut également être suivie par spectrophotométrie.  
Spectre du diiode :



On choisit  $\lambda = 475 \text{ nm}$

Dans un becher de 100 mL, mettre 5,0 mL de peroxodisulfate de sodium de concentration  $1,2 \cdot 10^{-3} \text{ mol.L}^{-1}$ .

Préparer 15 mL d'iodure de potassium de concentration  $C(I^-) = 1,0 \text{ mol.L}^{-1}$ . A  $t=0\text{s}$ , mélanger les deux bécher et très rapidement faire un prélèvement dans une cuve de spectrophotométrie et introduire la cuve dans le spectrophotomètre pour suivre A en fonction du temps.



## Exemple 2 : oxydation de l'érythrosine par les ions hypochlorites

**Protocole :** (40 expériences illustrées de chimie générale et organique- La chimie, une science expérimentale, E. Martinand-Lurin, R. Grüber, Ed. De Boeck)

Préparer une solution d'érythrosine B en introduisant 15 mg dans une fiole jaugée de 100 mL. Compléter avec de l'eau distillée.  
Diluer cette solution mère d'un facteur 20 (5,0 mL dans 95 mL d'eau distillée). Cette solution est à  $8,5 \cdot 10^{-3}$  mol/L est sera notée S.  
Enregistrer le spectre de cette solution entre 400 et 650 nm.  
Solution d'hypochlorite utilisée : 4,8% CA donc à 0,78 mol/L

Expérience 1 :

Solution hypochlorite diluée : 10,0 mL dans 10,0 mL d'eau distillée

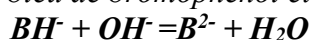
On ajoute à cette solution à  $t=0$  s, 10 mL de solution S d'érythrosine. Suivre l'absorbance au cours du temps (mesure toutes les 30 s)

Expérience 2 : solution hypochlorite 8,0 mL dans 12,0 mL d'eau

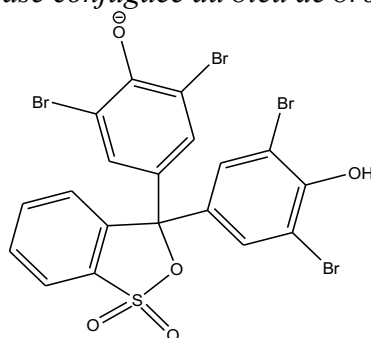
Expérience 3 : solution hypochlorite 5,0 mL dans 15,0 mL d'eau

## Exemple 3 : réaction du bleu de bromophénol

L'expérience a pour but l'étude de la détermination de l'ordre d'une réaction et de l'énergie d'activation de la réaction entre le bleu de bromophénol et les ions hydroxyde.



$BH^-$  symbolise la base conjuguée du bleu de bromophénol. Sa représentation est la suivante :

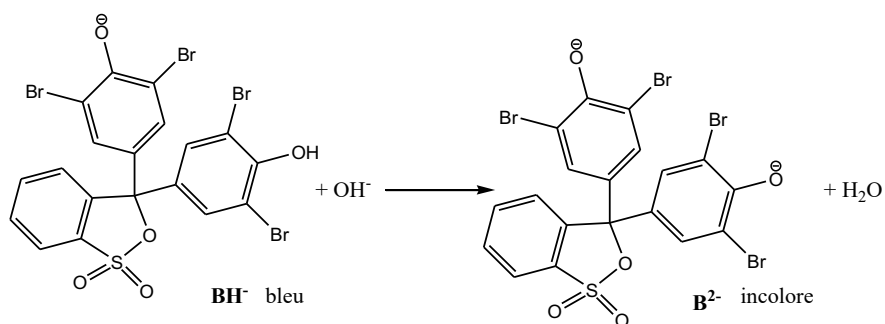


$B^{2-}$  est une espèce incolore.

Ici, nous nous intéressons au bleu de bromophénol (BBP), qui est un indicateur coloré, souvent utilisé comme indicateur de fin de réaction dans des dosages acido-basiques. Ce sont ses deux fonctions phénol qui sont responsables de ses propriétés acido-basiques.

Les solutions aqueuses de B sont donc de couleur jaune à  $pH < 3$ , de couleur bleue pour des  $pH$  compris entre 5 et 6 et incolores pour  $pH > 8$ . Dans les zones de  $pH$  entre ces limites, les solutions ont une couleur résultant du mélange jaune-bleu ou bleu-incolore.

Pour  $pH > 7$ ,  $BH^-$  réagit avec les ions hydroxydes, selon la réaction suivante :



Cette réaction est rendue irréversible par la décomposition de  $\text{B}^{2-}$  qui n'est pas stable.

$$v = k[\text{BH}^-]^a[\text{OH}^-]^b$$

### **Protocole :**

#### **Détermination de a :**

1. Préparer le spectrophotomètre pour les mesures :

- Choisir le mode suivi cinétique et régler les paramètres (longueur d'onde, temps de mesure (45min maxi), pas de mesure (30 secondes ou 1 minute))
- Effectuer le zéro d'absorbance

2. Introduire à la pipette dans un bécher muni d'un barreau aimanté et d'un système d'agitation, 2 mL de solution aqueuse de bleu de bromophénol (de concentration massique 0,4 g.L<sup>-1</sup>) et 50 mL d'une solution d'hydroxyde de sodium (de concentration CB=1,4 mol.L<sup>-1</sup>).

3. Déclencher le chronomètre.

4. Homogénéiser la solution et commencer rapidement les mesures.

Remarque : On pourra noter le temps  $t_0$  de début de mesure (il y a une option « retard » sur le spectrophotomètre) pour corriger le temps initiale.

#### **Détermination de b :**

Réaliser deux dilutions de la solution de soude afin de l'utiliser pour faire deux autres expériences avec le protocole précédent.

#### **Détermination de l'énergie d'activation :**

On pourra refaire l'une des expériences précédentes mais en plaçant le bécher dans un bain de glace ou dans un bain thermostaté. Il faudra alors faire des prélèvements afin de faire les mesures d'absorbance rapidement sur le spectrophotomètre qui n'est pas thermostaté.

#### Exemple 4 : solvolysé du chlorure de tertiobutyle

---

##### **Protocole :**

*Dans une fiole jaugée de 100mL on prépare la solution S : 4mL de chlorure de tertiobutyle dans l'acétone.*

*Dans un bécher, on introduit 200 mL d'eau distillée et on déclenche le hronomètre lors de l'ajout de 5 mL de solution S. On réalise le suivi conductimétrique.*

*Autre protocole et données .csv pour traitement Python :*

<https://physique-chimie.discip.ac-caen.fr/spip.php?article1038>

## SCRIPTS PYTHON

### Exemple 1 : Oxydation des iodures

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

#déclaration des variables/paramètres
myFile = 'cinetiqueS2O8.txt'
graphShow = True # False/True
#
V0 = 5.0 #(mL)
C0_S2O3 = 1e-3 #(mol/L)

# initialisation des variables
temps = []
Vequiv = []

#lecture du fichier de données
data = open(myFile,'r')
for row in data:
    row = row.split(' ')
    temps.append(float(row[0]))
    Vequiv.append(float(row[1]))

temps = np.array(temps)
Vequiv = np.array(Vequiv)

t = temps[0:len(Vequiv)-1]
Veq = Vequiv[0:len(Vequiv)-1]

# calcul des différentes quantités
C_I2 = C0_S2O3/(2*V0)*Veq
C_S2O8 = C0_S2O3/(2*V0)*(Vequiv[len(Vequiv)-1]-Veq)
Vitesse = np.diff(C_S2O8)/np.diff(t)

#calcul des paramètres de la régression lineaire
def cost_function(coef): return np.sum((np.log(C_S2O8) - (coef[0]*t+coef[1]))**2)
res = minimize(cost_function,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(res.message)
opt_coef = res.x
residu = np.log(C_S2O8) - (opt_coef[0]*t+opt_coef[1])
r2 = np.corrcoef([np.log(C_S2O8), opt_coef[0]*t+opt_coef[1]])
txt_eqn = f'ln([S2O8]) : ({opt_coef[0]:1.4f})*t + ({opt_coef[1]:1.4f}), (R^2 =
{r2[0,1]**2:1.4f})'
print(txt_eqn)
```



```

#calcul des paramètres de la régression lineaire loi v=kA
def cost_functionV(coef): return np.sum((Vitesse - (coef[0]*C_S2O8[0:len(C_S2O8)-1]+coef[1]))**2)
resV = minimize(cost_functionV,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp': False})
print(resV.message)
opt_coefV = resV.x
residuV = Vitesse - (opt_coefV[0]*C_S2O8[0:len(C_S2O8)-1]+opt_coefV[1])
r2V = np.corrcoef([Vitesse, opt_coefV[0]*C_S2O8[0:len(C_S2O8)-1]+opt_coefV[1]])
MV=np.max(np.abs(residuV))
muV=np.mean(residuV)
sigmaV=np.std(residuV)
txt_eqnV = f'V : ({opt_coefV[0]:1.4g})*C + ({opt_coefV[1]:1.4g}), (R^2 = {r2V[0,1]**2:1.4f})'
print(txt_eqnV)

#calcul des paramètres de la régression lineaire méthode différentielle
def cost_functiondif(coef): return np.sum((np.log(-Vitesse) - (coef[0]*np.log(C_S2O8[0:len(C_S2O8)-1]+coef[1]))**2)
resdif = minimize(cost_functiondif,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp': False})
print(resdif.message)
opt_coefdif = resdif.x
residudif = np.log(-Vitesse) - (opt_coefdif[0]*np.log(C_S2O8[0:len(C_S2O8)-1]+opt_coefdif[1]))
r2dif = np.corrcoef([np.log(-Vitesse), opt_coefdif[0]*np.log(C_S2O8[0:len(C_S2O8)-1]+opt_coefdif[1]))
Mdif=np.max(np.abs(residudif))
mudif=np.mean(residudif)
sigmadif=np.std(residudif)
txt_eqndif = f'ln(V) : ({opt_coefdif[0]:1.4g})*ln(C) + ({opt_coefdif[1]:1.4g}), (R^2 = {r2dif[0,1]**2:1.4f})'
print(txt_eqndif)

#Graphes et figures
#figure1
fig1 = plt.figure(num=1, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'$V_{equiv}$ (mL)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$V_{equiv} = f(t)$', fontsize = 16)
plt.axis([0,1200,0,20])
plt.plot(temps, Vequiv, '+', label=r'$V_{equiv}$ (mL)')
plt.legend(loc='best')
#
##figure2
fig2 = plt.figure(num=2, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'$I_{S_2}$ (mol/L)', fontsize = 12)

```

```

plt.grid(linestyle='--')
plt.title(r'$[I_2] = f(t)$', fontsize = 16)
plt.axis([0,60,0,1.1*np.max(C_I2)])
plt.plot(t, C_I2, '+--', label=r'$[I_2]$ (mol/L)')
#
##figure3
fig3 = plt.figure(num=3, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'$[S_{2}O_{8}]$ (mol/L)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$[S_{2}O_{8}] = f(t)$', fontsize = 16)
plt.axis([0,60,0,1.1*np.max(C_S2O8)])
plt.plot(t, C_S2O8, '+--', label=r'$[S_{2}O_{8}]$ (mol/L)')
#
##figure4
fig4 = plt.figure(num=4, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel('temps (min)', fontsize = 12)
ax1.set_ylabel(r'$\ln([S_{2}O_{8}])$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$\ln([S_{2}O_{8}]) = f(t)$', fontsize = 16)
ax1.plot(t, np.log(C_S2O8), '+--', label=r'$\ln([S_{2}O_{8}])$')
ax1.plot(t, opt_coef[0]*t+opt_coef[1], 'r-')
ax1.text(t[1], 1.02*opt_coef[1], txt_eqn, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(t, residu, '+')
ax2.grid(linestyle='--')
ax2.set_xlabel('temps (min)', fontsize = 12)
ax2.set_ylabel('résidus', fontsize = 12)
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residu, facecolor='b', alpha=0.75)
ax3.text(0,2.7,f'moyenne = {np.mean(residu):1.4g}', fontsize=10)
ax3.text(0,2.4,f'écart-type = {np.std(residu):1.4f}', fontsize=10)

##figure5
fig5 = plt.figure(num=5, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'$\ln(C)$', fontsize = 12)
ax1.set_ylabel(r'$\ln(vitesse)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$\ln(v) = f(\ln(C))$', fontsize = 16)
ax1.plot(np.log(C_S2O8[0:len(C_S2O8)-1]), np.log(abs(Vitesse)), '+--', label=r'$vitesse$')
ax1.plot(np.log(C_S2O8[0:len(C_S2O8)-1]), opt_coefdif[0]*np.log(C_S2O8[0:len(C_S2O8)-1])+opt_coefdif[1], 'r-')
ax1.text(-8,-10, txt_eqndif, fontsize=12, color='red')

```

```

#
ax1.text(0.3*np.max(np.log(C)),.5*(np.min(np.log(abs(Vitesse)))+np.max(np.log(abs(Vitesse
))))/2,txt_eqndif, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(np.log(C_S2O8[0:len(C_S2O8)-1]),residudif, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('ln(C)', fontsize = 12)
ax2.set_ylabel('résidus ln(vitesse)', fontsize = 12)
#ax2.axis([0,1.1*np.max(np.log(A)),-1.1*Mdif,1.1*Mdif])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residudif,int(np rint(len(np.log(C_S2O8))/4)), density=True, facecolor='b',
alpha=0.75)
Xdif=np.linspace(-1.1*Mdif,1.1*Mdif,100)
gaussdif = 1/(sigmadif*(2*np.pi)**0.5)*np.exp(-(Xdif-mudif)**2/(2*sigmadif**2))
ax3.plot(Xdif,gaussdif)
ax3.text(-Mdif,1.3*np.max(gaussdif),f'moyenne = {mudif:1.4g}', fontsize=10)
ax3.text(-Mdif,1.2*np.max(gaussdif),f'écart-type = {sigmadif:1.4g}', fontsize=10)
ax3.axis([-1.15*Mdif,1.15*Mdif,0,1.4*np.max(gaussdif)])
#
##figure7
fig5 = plt.figure(num=7, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'$1/[S_{2}O_{8}]$ (L/mol)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$1/[S_{2}O_{8}] = f(t)$', fontsize = 16)
plt.plot(t, 1/C_S2O8, '+--', label=r'$1/[S_{2}O_{8}]$')
#
##figure8
fig6 = plt.figure(num=8, figsize=[10,7])
plt.xlabel('[S2O8] (mol/L)', fontsize = 12)
plt.ylabel(r'$v$ (L/mol/min)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$v = f([S2O8])$', fontsize = 16)
plt.plot(C_S2O8[0:len(C_S2O8)-1], Vitesse, '+--')

##Affichage des figures
if graphShow: plt.show()

```

---

### **Exemple 2 : Bromophénol**

```

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

#déclaration des variables/paramètres
myFile = 'cinetiquebromphenol.txt'
graphShow = True # False/True
#

```

```

# initialisation des variables
temps = []
A = []
C = []

#lecture du fichier de données
data = open(myFile,'r')
for row in data:
    row = row.split('\t')
    temps.append(float(row[0]))
    A.append(float(row[1]))

t = np.array(temps)
A = np.array(A)
C = A/55300

# calcul de la vitesse
VitesseA = -np.diff(A)/np.diff(t)
Vitesse = -np.diff(C)/np.diff(t)

#calcul des paramètres de la régression lineaire méthode intégrale
def cost_function(coef): return np.sum((np.log(C) - (coef[0]*t+coef[1]))**2)
res = minimize(cost_function,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(res.message)
opt_coef = res.x
residu = np.log(C) - (opt_coef[0]*t+opt_coef[1])
r2 = np.corrcoef([np.log(C), opt_coef[0]*t+opt_coef[1]])
M=np.max(np.abs(residu))
mu=np.mean(residu)
sigma=np.std(residu)
txt_eqn = f'ln(C) : ({opt_coef[0]:1.4f})*t + ({opt_coef[1]:1.4f}), (R^2 = {r2[0,1]**2:1.4f})'
print(txt_eqn)

#calcul des paramètres de la régression lineaire méthode v=kC
def cost_functionV(coef): return np.sum((Vitesse - (coef[0]*C[0:len(C)-1]+coef[1]))**2)
resV = minimize(cost_functionV,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(resV.message)
opt_coefV = resV.x
residuV = Vitesse - (opt_coefV[0]*C[0:len(C)-1]+opt_coefV[1])
r2V = np.corrcoef([Vitesse, opt_coefV[0]*C[0:len(C)-1]+opt_coefV[1]])
MV=np.max(np.abs(residuV))
muV=np.mean(residuV)
sigmaV=np.std(residuV)
txt_eqnV = f'V : ({opt_coefV[0]:1.4g})*C + ({opt_coefV[1]:1.4g}), (R^2 =
{r2V[0,1]**2:1.4f})'
print(txt_eqnV)

```

```

#calcul des paramètres de la régression lineaire méthode différentielle
def cost_functiondif(coef): return np.sum((np.log(abs(Vitesse)) - (coef[0]*np.log(C[0:len(C)-1])+coef[1]))**2)
resdif = minimize(cost_functiondif,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp': False})
print(resdif.message)
opt_coefdif = resdif.x
residudif = np.log(abs(Vitesse)) - (opt_coefdif[0]*np.log(C[0:len(C)-1])+opt_coefdif[1])
r2dif = np.corrcoef([np.log(abs(Vitesse)), opt_coefdif[0]*np.log(C[0:len(C)-1])+opt_coefdif[1]])
Mdif=np.max(np.abs(residudif))
mudif=np.mean(residudif)
sigmadif=np.std(residudif)
txt_eqndif = f'ln(V) : ({opt_coefdif[0]:1.4g})*ln(C) + ({opt_coefdif[1]:1.4g}), (R^2 = {r2dif[0,1]**2:1.4f})'
print(txt_eqndif)

```

#Graphes et figures

#figure1

```

fig1 = plt.figure(num=1, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'C (mol/L)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$C = f(t)$', fontsize = 16)
plt.axis([0,t[len(t)-1],0,1.1*np.max(C)])
plt.plot(t, C, '+', label=r'C')
plt.legend(loc='best')

```

#

##figure2

```

fig2 = plt.figure(num=2, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'temps (s)', fontsize = 12)
ax1.set_ylabel(r'$ln(C)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$ln(C) = f(t)$', fontsize = 16)
ax1.plot(t, np.log(C), '+--', label=r'ln( $C$)')
ax1.plot(t,opt_coef[0]*t+opt_coef[1], 'r-' )
ax1.text(0,-13.5,txt_eqn, fontsize=12, color='red')

```

#

```

ax2 = plt.subplot(223)
ax2.plot(t,residu, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('temps (s)', fontsize = 12)
ax2.set_ylabel('résidus', fontsize = 12)
ax2.axis([0,t[len(t)-1],-1.1*M,1.1*M])

```

#

```

ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residu, facecolor='b', alpha=0.75)

```

```

ax3.text(-M,14,f'moyenne = {mu:1.4g}', fontsize=10)
ax3.text(-M,13,f'écart-type = {sigma:1.4f}', fontsize=10)
ax3.axis([-1.1*M,1.1*M,0,15])
#
##figure3
fig3 = plt.figure(num=3, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'$1/C$ (L/mol)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$1/C = f(t)$', fontsize = 16)
plt.plot(t, 1/C, '+--', label=r'$1/A$')
#
##figure4
fig4 = plt.figure(num=4, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'C', fontsize = 12)
ax1.set_ylabel(r'$vitesse$ (mol/L/s)', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$v = f(C)$', fontsize = 16)
ax1.plot(C[0:len(C)-1], Vitesse, '+--', label=r'vitesse')
ax1.plot(C[0:len(C)-1],opt_coefV[0]*C[0:len(C)-1]+opt_coefV[1], 'r-' )
ax1.text(0.2,.5*(np.min(Vitesse)+np.max(Vitesse))/2,txt_eqnV, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(A[0:len(A)-1],residuV, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('A', fontsize = 12)
ax2.set_ylabel('résidus vitesse', fontsize = 12)
ax2.axis([0,1.1*np.max(A),-1.1*MV,1.1*MV])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residuV,int(np rint(len(C)/4)), density=True, facecolor='b', alpha=0.75)
X=np.linspace(-1.1*MV,1.1*MV,100)
gauss = 1/(sigmaV*(2*np.pi)**0.5)*np.exp(-(X-muV)**2/(2*sigmaV**2))
ax3.plot(X,gauss)
ax3.text(-MV,1.3*np.max(gauss),f'moyenne = {muV:1.4g}', fontsize=10)
ax3.text(-MV,1.2*np.max(gauss),f'écart-type = {sigmaV:1.4g}', fontsize=10)
ax3.axis([-1.15*MV,1.15*MV,0,1.4*np.max(gauss)])

##figure5
fig5 = plt.figure(num=5, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'ln(C)', fontsize = 12)
ax1.set_ylabel(r'$ln(vitesse)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$ln(v) = f(ln(C))$', fontsize = 16)
ax1.plot(np.log(C[0:len(C)-1]), np.log(abs(Vitesse)), '+--', label=r'vitesse')
ax1.plot(np.log(C[0:len(C)-1]),opt_coefdif[0]*np.log(C[0:len(C)-1])+opt_coefdif[1], 'r-' )
ax1.text (-14.0,-18.2, txt_eqndif, fontsize=12, color='red')

```

```

#
ax1.text(0.3*np.max(np.log(A)),.5*(np.min(np.log(abs(Vitesse)))+np.max(np.log(abs(Vitesse
))))/2,txt_eqndif, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(np.log(C[0:len(C)-1]),residudif, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('ln(C)', fontsize = 12)
ax2.set_ylabel('résidus ln(vitesse)', fontsize = 12)
#ax2.axis([0,1.1*np.max(np.log(A)),-1.1*Mdif,1.1*Mdif])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residudif,int(np rint(len(np.log(A))/4)), density=True, facecolor='b', alpha=0.75)
Xdif=np.linspace(-1.1*Mdif,1.1*Mdif,100)
gaussdif = 1/(sigmadif*(2*np.pi)**0.5)*np.exp(-(Xdif-mudif)**2/(2*sigmadif**2))
ax3.plot(Xdif,gaussdif)
ax3.text(-Mdif,1.3*np.max(gaussdif),f'moyenne = {mudif:1.4g}', fontsize=10)
ax3.text(-Mdif,1.2*np.max(gaussdif),f'écart-type = {sigmadif:1.4g}', fontsize=10)
#ax3.axis([-1.15*Mdif,1.15*Mdif,0,1.4*np.max(gaussdif)])

#figure6
fig1 = plt.figure(num=6, figsize=[10,7])
plt.xlabel('C', fontsize = 12)
plt.ylabel(r'vitesse (mol/L/s)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$v = f(C)$', fontsize = 16)
#plt.axis([0,t[len(t)-1],0,1.1*np.max(A)])
plt.plot(C[0:len(C)-1], Vitesse, '+', label=r'C')

##figure7
fig7 = plt.figure(num=7, figsize=[10,7])
plt.xlabel(r'C', fontsize = 12)
plt.ylabel(r'$vitesse (mol/L/s)$', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$v = f(C)$', fontsize = 16)
plt.plot(C[0:len(C)-1], Vitesse, '+--', label=r'vitesse')
plt.plot(C[0:len(C)-1],opt_coefV[0]*C[0:len(C)-1]+opt_coefV[1], 'r-' )
plt.text(0.2,2.0,txt_eqnV, fontsize=12, color='red')

##Affichage des figures
if graphShow: plt.show()

```

---

### **Exemple 3 : erythrosine**

```

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

#déclaration des variables/paramètres

```

```

myFile = 'erythrosine1.txt'
graphShow = True # False/True
#

# initialisation des variables
temps = []
A = []

#lecture du fichier de données
data = open(myFile,'r')
for row in data:
    row = row.split('\t')
    temps.append(float(row[0]))
    A.append(float(row[1]))

t = np.array(temps)
A = np.array(A)

# calcul de la vitesse
Vitesse = -np.diff(A)/np.diff(t)
C=A/82000
CP= (A[len(A)-1]-A)/82000
VitesseC = -np.diff(C)/np.diff(t)
VitesseP = np.diff(CP)/np.diff(t)

#calcul des paramètres de la régression lineaire méthode intégrale
def cost_function(coef): return np.sum((np.log(A) - (coef[0]*t+coef[1]))**2)
res = minimize(cost_function,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(res.message)
opt_coef = res.x
residu = np.log(A) - (opt_coef[0]*t+opt_coef[1])
r2 = np.corrcoef([np.log(A), opt_coef[0]*t+opt_coef[1]])
M=np.max(np.abs(residu))
mu=np.mean(residu)
sigma=np.std(residu)
txt_eqn = f'ln(A) : ({opt_coef[0]:1.4f})*t + ({opt_coef[1]:1.4f}), (R^2 = {r2[0,1]**2:1.4f})'
print(txt_eqn)

#calcul des paramètres de la régression lineaire loi v=kA
def cost_functionV(coef): return np.sum((Vitesse - (coef[0]*A[0:len(A)-1]+coef[1]))**2)
resV = minimize(cost_functionV,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(resV.message)
opt_coefV = resV.x
residuV = Vitesse - (opt_coefV[0]*A[0:len(A)-1]+opt_coefV[1])
r2V = np.corrcoef([Vitesse, opt_coefV[0]*A[0:len(A)-1]+opt_coefV[1]])
MV=np.max(np.abs(residuV))
muV=np.mean(residuV)
sigmaV=np.std(residuV)

```



```

txt_eqnV = fV : ({opt_coefV[0]:1.4g})*A + ({opt_coefV[1]:1.4g}), (R^2 =
{r2V[0,1]**2:1.4f})'
print(txt_eqnV)

#calcul des paramètres de la régression lineaire méthode différentielle
def cost_functiondif(coef): return np.sum((np.log(Vitesse) - (coef[0]*np.log(A[0:len(A)-
1])+coef[1]))**2)
resdif = minimize(cost_functiondif,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(resdif.message)
opt_coefdif = resdif.x
residudif = np.log(Vitesse) - (opt_coefdif[0]*np.log(A[0:len(A)-1])+opt_coefdif[1])
r2dif = np.corrcoef([np.log(Vitesse), opt_coefdif[0]*np.log(A[0:len(A)-1])+opt_coefdif[1]])
Mdif=np.max(np.abs(residudif))
mudif=np.mean(residudif)
sigmadif=np.std(residudif)
txt_eqndif = f'ln(V) : ({opt_coefdif[0]:1.4g})*ln(A) + ({opt_coefdif[1]:1.4g}), (R^2 =
{r2dif[0,1]**2:1.4f})'
print(txt_eqndif)

```

#Graphes et figures

#figure1

```

fig1 = plt.figure(num=1, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'A', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$A = f(t)$', fontsize = 16)
plt.axis([0,t[len(t)-1],0,1.1*np.max(A)])
plt.plot(t, A, '+', label=r'A')
#plt.legend(loc='best')

```

#

##figure2

```

fig2 = plt.figure(num=2, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'temps (s)', fontsize = 12)
ax1.set_ylabel(r'$ln(A)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$ln(A) = f(t)$', fontsize = 16)
ax1.plot(t, np.log(A), '+--', label=r'ln( $A$)')
ax1.plot(t,opt_coef[0]*t+opt_coef[1], 'r-' )
ax1.text(t[0],1.5*opt_coef[1],txt_eqn, fontsize=12, color='red')
#

```

```

ax2 = plt.subplot(223)
ax2.plot(t,residu, '+')
ax2.grid(linestyle='--')
ax2.set_xlabel('temps (s)', fontsize = 12)
ax2.set_ylabel('résidus', fontsize = 12)
ax2.axis([0,t[len(t)-1],-1.1*M,1.1*M])

```

```

#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residu, facecolor='b', alpha=0.75)
ax3.text(-M,14,f'moyenne = {mu:1.4g}', fontsize=10)
ax3.text(-M,13,f'écart-type = {sigma:1.4f}', fontsize=10)
ax3.axis([-1.1*M,1.1*M,0,15])
#
##figure3
fig3 = plt.figure(num=3, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'$1/A$ (L/mol)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$1/A = f(t)$', fontsize = 16)
plt.plot(t, 1/A, '+--', label=r'$1/A$')
#
##figure4
fig4 = plt.figure(num=4, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'A', fontsize = 12)
ax1.set_ylabel(r'$vitesse (/s)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$v = f(A)$', fontsize = 16)
ax1.plot(A[0:len(A)-1], Vitesse, '+--', label=r'vitesse')
ax1.plot(A[0:len(A)-1],opt_coefV[0]*A[0:len(A)-1]+opt_coefV[1], 'r-' )
ax1.text(0.6*np.max(A),.5*(np.min(Vitesse)+np.max(Vitesse))/2,txt_eqnV, fontsize=12,
color='red')
#
ax2 = plt.subplot(223)
ax2.plot(A[0:len(A)-1],residuV, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('A', fontsize = 12)
ax2.set_ylabel('résidus vitesse', fontsize = 12)
ax2.axis([0,1.1*np.max(A),-1.1*MV,1.1*MV])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residuV,int(np rint(len(A)/4)), density=True, facecolor='b', alpha=0.75)
X=np.linspace(-1.1*MV,1.1*MV,100)
gauss = 1/(sigmaV*(2*np.pi)**0.5)*np.exp(-(X-muV)**2/(2*sigmaV**2))
ax3.plot(X,gauss)
ax3.text(-MV,1.3*np.max(gauss),f'moyenne = {muV:1.4g}', fontsize=10)
ax3.text(-MV,1.2*np.max(gauss),f'écart-type = {sigmaV:1.4g}', fontsize=10)
ax3.axis([-1.15*MV,1.15*MV,0,1.4*np.max(gauss)])
#
##figure5
fig5 = plt.figure(num=5, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'ln(A)', fontsize = 12)
ax1.set_ylabel(r'$ln(vitesse)$', fontsize = 12)

```

```

ax1.grid(linestyle='--')
ax1.set_title(r'$\ln(v) = f(\ln(A))$', fontsize = 16)
ax1.plot(np.log(A[0:len(A)-1]), np.log(abs(Vitesse)), '+--', label=r'vitesse')
ax1.plot(np.log(A[0:len(A)-1]), opt_coefdif[0]*np.log(A[0:len(A)-1])+opt_coefdif[1], 'r-' )
ax1.text (-3.2,-7.5, txt_eqndif, fontsize=12, color='red')
#
ax1.text(0.3*np.max(np.log(A)),.5*(np.min(np.log(abs(Vitesse)))+np.max(np.log(abs(Vitesse))))/2,txt_eqndif, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(np.log(A[0:len(A)-1]),residudif, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('ln(A)', fontsize = 12)
ax2.set_ylabel('résidus ln(vitesse)', fontsize = 12)
#ax2.axis([0,1.1*np.max(np.log(A)), -1.1*Mdif,1.1*Mdif])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residudif,int(np rint(len(np.log(A))/4)), density=True, facecolor='b', alpha=0.75)
Xdif=np.linspace(-1.1*Mdif,1.1*Mdif,100)
gaussdif = 1/(sigmadif*(2*np.pi)**0.5)*np.exp(-(Xdif-mudif)**2/(2*sigmadif**2))
ax3.plot(Xdif,gaussdif)
ax3.text(-Mdif,1.3*np.max(gaussdif),f'moyenne = {mudif:1.4g}', fontsize=10)
ax3.text(-Mdif,1.2*np.max(gaussdif),f'écart-type = {sigmadif:1.4g}', fontsize=10)
ax3.axis([-1.15*Mdif,1.15*Mdif,0,1.4*np.max(gaussdif)])
#
#figure6
fig6 = plt.figure(num=6, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'C (mol/L)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$C(erythrosine) = f(t)$', fontsize = 16)
#plt.axis([0,t[len(t)-1],0,1.1*np.max(C)])
plt.plot(t, C, '+--', label=r'C')
#plt.legend(loc='best')

#figure7
fig7 = plt.figure(num=7, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'C(produit)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$C(produit) = f(t)$', fontsize = 16)
#plt.axis([0,t[len(t)-1],0,1.1*np.max(Vitesse)])
plt.plot(t, CP, '+--', label=r'v')
#plt.legend(loc='best')

#figure8
fig8 = plt.figure(num=8, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'vitesse (mol/L/s)', fontsize = 12)

```

```

plt.grid(linestyle='--')
plt.title(r'$v(\text{érythrosine}) = f(t)$', fontsize = 16)
#plt.axis([0,t[len(t)-1],0,1.1*np.max(Vitesse)])
plt.plot(t[0:len(A)-1], VitesseC, '+', label=r'v')
plt.legend(loc='best')

#figure9
fig9 = plt.figure(num=9, figsize=[10,7])
plt.xlabel('temps (s)', fontsize = 12)
plt.ylabel(r'$v(\text{produit}) = f(t)$', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$v(\text{produit}) = f(t)$', fontsize = 16)
#plt.axis([0,t[len(t)-1],0,1.1*np.max(Vitesse)])
plt.plot(t[0:len(A)-1], VitesseP, '+', label=r'v')

plt.legend(loc='best')

##Affichage des figures
if graphShow: plt.show()

```

---

#### **Exemple 4 : Solvolysé tertio-butyle**

```

import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize

#déclaration des variables/paramètres
myFile = 'solvolysé.txt'
graphShow = True # False/True
#

# initialisation des variables
temps = []
conductivité = []

#lecture du fichier de données
data = open(myFile,'r')
for row in data:
    row = row.split('\t')
    temps.append(float(row[0]))
    conductivité.append(float(row[1]))

t = np.array(temps)
conductivité = np.array(conductivité)

# calcul des concentrations et de la vitesse
Concentration = (conductivité[len(conductivité)-1] -
conductivité[0])/conductivité[len(conductivité)-1]
C=Concentration[0:len(Concentration)-2]

```

```
tprime=t[0:len(Concentration)-2]
Vitesse = np.diff(conductivité)/np.diff(t)
```

```
#calcul des paramètres de la régression lineaire méthode intégrale
def cost_function(coef): return np.sum((np.log(C) - (coef[0]*tprime+coef[1]))**2)
res = minimize(cost_function,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(res.message)
opt_coef = res.x
residu = np.log(C) - (opt_coef[0]*tprime+opt_coef[1])
r2 = np.corrcoef([np.log(C), opt_coef[0]*tprime+opt_coef[1]])
M=np.max(np.abs(residu))
mu=np.mean(residu)
sigma=np.std(residu)
txt_eqn = f'ln(C) : ({opt_coef[0]:1.4f})*t + ({opt_coef[1]:1.4f}), (R^2 = {r2[0,1]**2:1.4f})'
print(txt_eqn)
```

```
#calcul des paramètres de la régression lineaire méthode v=kA
def cost_functionV(coef): return np.sum((Vitesse[0:len(C)-1] - (coef[0]*C+coef[1]))**2)
resV = minimize(cost_functionV,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(resV.message)
opt_coefV = resV.x
residuV = Vitesse - (opt_coefV[0]*C[0:len(C)-1]+opt_coefV[1])
r2V = np.corrcoef([Vitesse, opt_coefV[0]*C+opt_coefV[1]])
MV=np.max(np.abs(residuV))
muV=np.mean(residuV)
sigmaV=np.std(residuV)
txt_eqnV = f'V : ({opt_coefV[0]:1.4g})*C + ({opt_coefV[1]:1.4g}), (R^2 =
{r2V[0,1]**2:1.4f})'
print(txt_eqnV)
```

```
#calcul des paramètres de la régression lineaire méthode différentielle
def cost_functiondif(coef): return np.sum((np.log(abs(Vitesse)) - (coef[0]*np.log(C[0:len(C)-
1])+coef[1]))**2)
resdif = minimize(cost_functiondif,[1,0], method='nelder-mead', options={'xatol': 1e-8, 'disp':
False})
print(resdif.message)
opt_coefdif = resdif.x
residudif = np.log(abs(Vitesse)) - (opt_coefdif[0]*np.log(C[0:len(C)-1])+opt_coefdif[1])
r2dif = np.corrcoef([np.log(abs(Vitesse)), opt_coefdif[0]*np.log(C[0:len(C)-
1])+opt_coefdif[1]])
Mdif=np.max(np.abs(residudif))
mudif=np.mean(residudif)
sigmadif=np.std(residudif)
txt_eqndif = f'ln(V) : ({opt_coefdif[0]:1.4g})*ln(C) + ({opt_coefdif[1]:1.4g}), (R^2 =
{r2dif[0,1]**2:1.4f})'
print(txt_eqndif)
```

```
#Graphes et figures
```

```

#figure1
fig1 = plt.figure(num=1, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'C/C_0', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$C/C_0 = f(t)$', fontsize = 16)
plt.axis([0,t[len(t)-1],0,1.1*np.max(C)])
plt.plot(t, C, '+', label=r'C/C_0')
#plt.legend(loc='best')

#
##figure2
fig2 = plt.figure(num=2, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'temps (min)', fontsize = 12)
ax1.set_ylabel(r'$\ln(C)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$\ln(C) = f(t)$', fontsize = 16)
ax1.plot(tpime[0:len(C)-1], np.log(C[0:len(C)-1]), '+--', label=r'$\ln(C)$')
ax1.plot(tpime,opt_coef[0]*tpime+opt_coef[1], 'r-' )
ax1.text(tpime[7],1.5*opt_coef[1],txt_eqn, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(t,residu, '+')
ax2.grid(linestyle='--')
ax2.set_xlabel('temps (min)', fontsize = 12)
ax2.set_ylabel('résidus', fontsize = 12)
ax2.axis([0,t[len(t)-1],-1.1*M,1.1*M])
#
#ax3 = plt.subplot(224)
#ax3.grid(linestyle='--')
#ax3.hist(residu, facecolor='b', alpha=0.75)
#ax3.text(-M,14,f'moyenne = {mu:1.4g}', fontsize=10)
#ax3.text(-M,13,f'écart-type = {sigma:1.4f}', fontsize=10)
#ax3.axis([-1.1*M,1.1*M,0,15])
#
##figure3
fig3 = plt.figure(num=3, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'$C_0/C$ (L/mol)', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$C_0/C = f(t)$', fontsize = 16)
plt.plot(t[0:len(C)-2], 1/C[0:len(C)-2], '+--', label=r'$C_0/C$')
#
##figure4
fig4 = plt.figure(num=4, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'C', fontsize = 12)
ax1.set_ylabel(r'$vitesse$', fontsize = 12)
ax1.grid(linestyle='--')

```

```

ax1.set_title(r'$v = f(C)$', fontsize = 16)
ax1.plot(C[0:len(C)-1], Vitesse, '+--', label=r'vitesse')
ax1.plot(C[0:len(C)-1], opt_coefV[0]*C[0:len(C)-1]+opt_coefV[1], 'r-' )
ax1.text(0.3*np.max(C),.5*(np.min(Vitesse)+np.max(Vitesse))/2,txt_eqnV, fontsize=12,
color='red')
#
ax2 = plt.subplot(223)
ax2.plot(C[0:len(C)-1],residuV, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('C', fontsize = 12)
ax2.set_ylabel('résidus vitesse', fontsize = 12)
ax2.axis([0,1.1*np.max(C),-1.1*MV,1.1*MV])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residuV,int(np rint(len(C)/4)), density=True, facecolor='b', alpha=0.75)
X=np.linspace(-1.1*MV,1.1*MV,100)
gauss = 1/(sigmaV*(2*np.pi)**0.5)*np.exp(-(X-muV)**2/(2*sigmaV**2))
ax3.plot(X,gauss)
ax3.text(-MV,1.3*np.max(gauss),f'moyenne = {muV:1.4g}', fontsize=10)
ax3.text(-MV,1.2*np.max(gauss),f'écart-type = {sigmaV:1.4g}', fontsize=10)
ax3.axis([-1.15*MV,1.15*MV,0,1.4*np.max(gauss)])

###figure5
fig5 = plt.figure(num=5, figsize=[10,7])
ax1 = plt.subplot(211)
ax1.set_xlabel(r'ln(C)', fontsize = 12)
ax1.set_ylabel(r'$ln(vitesse)$', fontsize = 12)
ax1.grid(linestyle='--')
ax1.set_title(r'$ln(v) = f(ln(C))$', fontsize = 16)
ax1.plot(np.log(C[0:len(C)-1]), np.log(abs(Vitesse)), '+--', label=r'vitesse')
ax1.plot(np.log(C[0:len(C)-1]),opt_coefdif[0]*np.log(C[0:len(C)-1])+opt_coefdif[1], 'r-' )
ax1.text (-3.2,-7.5, txt_eqndif, fontsize=12, color='red')
#
ax1.text(0.3*np.max(np.log(A)),.5*(np.min(np.log(abs(Vitesse)))+np.max(np.log(abs(Vitesse
))))/2,txt_eqndif, fontsize=12, color='red')
#
ax2 = plt.subplot(223)
ax2.plot(np.log(C[0:len(C)-1]),residudif, '+' )
ax2.grid(linestyle='--')
ax2.set_xlabel('ln(C)', fontsize = 12)
ax2.set_ylabel('résidus ln(vitesse)', fontsize = 12)
#ax2.axis([0,1.1*np.max(np.log(A)),-1.1*Mdif,1.1*Mdif])
#
ax3 = plt.subplot(224)
ax3.grid(linestyle='--')
ax3.hist(residudif,int(np rint(len(np.log(C))/4)), density=True, facecolor='b', alpha=0.75)
Xdif=np.linspace(-1.1*Mdif,1.1*Mdif,100)
gaussdif = 1/(sigmadif*(2*np.pi)**0.5)*np.exp(-(Xdif-mudif)**2/(2*sigmadif**2))
ax3.plot(Xdif,gaussdif)

```

```

ax3.text(-Mdif,1.3*np.max(gaussdif),f'moyenne = {mudif:1.4g}', fontsize=10)
ax3.text(-Mdif,1.2*np.max(gaussdif),f'écart-type = {sigmadif:1.4g}', fontsize=10)
#ax3.axis([-1.15*Mdif,1.15*Mdif,0,1.4*np.max(gaussdif)])
#
#figure6
fig6 = plt.figure(num=6, figsize=[10,7])
plt.xlabel('temps (min)', fontsize = 12)
plt.ylabel(r'conductivité', fontsize = 12)
plt.grid(linestyle='--')
plt.title(r'$Conductivité = f(t)$', fontsize = 16)
#plt.axis([0,t[len(t)-1],0,1.1*np.max(C)])
plt.plot(t, conductivité, '+', label=r'C')

##Affichage des figures
if graphShow: plt.show()

```