Quim. Nova, Vol. 49, 2026

Supplementary Material

ADSORB-XL: AN EXCEL-BASED TOOL FOR SIMPLE NONLINEAR REGRESSION ANALYSIS IN ADSORPTION STUDIES

Joacy B. Lima[a], Cicero W. B. Bezerra[a], Francisco A. Silva[b], Izaías S. Marques[b], Victor L. F. Pinheiro[b] and Leonardo B. Cantanhede[c,*]

[a]Laboratório de Interfaces e de Materiais, Universidade Federal do Maranhão (UFMA), 65085-580 São Luís – MA, Brasil

[b]Instituto Federal de Educação, Ciência e Tecnologia do Maranhão, 65030-005 São Luís – MA, Brasil

[c]Instituto Federal de Educação, Ciência e Tecnologia do Maranhão, 65400-000 Codó – MA, Brasil

Received: 12/05/2025; accepted: 03/18/2026; published online: 04/27/2026

*e-mail: leonardo.cantanhede@ifma.edu.br

ORCID:
Joacy B. Lima: https://orcid.org/0000-0002-1826-2768
Cicero W. B. Bezerra: https://orcid.org/0000-0001-9058-9469
Francisco A. Silva: https://orcid.org/0000-0003-2767-2184
Izaías S. Marques: https://orcid.org/0000-0002-0753-5206
Victor L. F. Pinheiro: https://orcid.org/0009-0004-9638-499X
Leonardo B. Cantanhede: https://orcid.org/0000-0002-9532-5566

TEMPLATES FOR NONLINEAR REGRESSION IN ADSORPTION PROCESSES

Presentation of templates

This file shows the step-by-step code required to perform a simple nonlinear regression of adsorption processes (pseudo-first order kinetics, pseudo-second order kinetics, and Langmuir adsorption isotherm) in Jupyter Notebook using Python code, in RStudio using R code, and in Matlab using Matlab code. This file also shows procedures for using the curve-fitting tool in the Matlab applications tab and the curve fit tool in Origin software.

The experimental data used in the nonlinear regressions of this templates were obtained from studies involving activated carbons derived the shell of the monkfish fruit (*Enterolobium contortisiliquum*), SMF, and the adsorbate used was Methylene Blue in aqueous solution at pH 5.5.

JUPYTER NOTEBOOK SOFTWARE

Procedure for running a nonlinear regression in Jupyter notebook software

1. On the desktop, click the Jupyter Notebook shortcut button to open the Jupyter Notebook software;
2. Do not close or change any lines of code in the Jupyter Notebook command prompt;
3. To load the Jupyter Notebook software, click the new button in the files tab of the HTML server and then click notebook;
4. Copy all lines of code from this file (line 1 to the last);
5. Paste them into the first cell of the Jupyter Notebook software;
6. Change the experimental x and y data;
7. Click the run button or press the shortcut keys (shift + enter)

Template for nonlinear regression of pseudo-first order adsorption kinetics

```
# JUPYTER NOTEBOOK SOFTWARE_LINES OF CODE IN PYTHON     (Line 1)
# TEMPLATE FOR NONLINEAR REGRESSION_PSEUDO-FIRST ORDER ADSORPTION KINETICS
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.stats import f
from scipy.stats import t
pd.set_option('display.float_format','{:.4e}'.format)     # Table values in scientific notation with 4 decimal places
pd.set_option('display.max_columns', None)     # Code to not break the columns
```

```
pd.set_option('display.width', 200)     # Code to increase columns in table
# Input experimental data
x=np.array([3, 6, 9, 20, 40, 60, 120, 180, 300, 360, 420, 480, 540])
y=np.array([121.25, 172.91, 242.49, 257.25, 322.04, 345.65, 370.15, 370.15, 370.15, 370.15, 370.15, 370.15,
370.15])
# linearized data
u=x
v=-np.log(np.max(y)*1.008-y)
# terms for linearization
n=np.size(u)
Su=np.sum(u)
Sv=np.sum(v)
Suv=np.sum(u*v)
Suu=np.sum(u*u)
Svv=np.sum(v*v)
angular=(Su*Sv-n*Suv)/(Su**2-n*Suu)
linear=(Su*Suv-Sv*Suu)/(Su**2-n*Suu)
# Define the model function as reg
def reg(x,a,b): return a*(1-np.exp(-b*x))
initial=[np.exp(-linear), angular]     # initial is the initial estimate for the parameters
NC=95     # confidence level
# Fit the data to the model equation using the curve_fit function
params, covariance = curve_fit(reg,x,y,initial)
# Extract the parameters
a_fit=params[0]
b_fit=params[1]
std_errors=np.sqrt(np.diag(covariance))     # Calculate the standard error of the parameters
# Calculate model statistic
y_pred=reg(x,a_fit,b_fit)
SQR=np.sum((y-y_pred)**2)     # Sum of squares of residuals
R2=1-(SQR/np.sum((y-np.mean(y))**2))     # Coefficient of determination (R-squared)
ss_res=np.sum((y-reg(x,*params))**2)
ss_tot=np.sum((y-np.mean(y))**2)
ss_reg=ss_tot-ss_res
nd=len(y)     # number of observations
par=len(params)     # number of parameters
df_tot=len(y)-1
df_reg=par-1
```

```
df_res=df_tot-df_reg
ms_res=ss_res/df_res
ms_tot=ss_tot/df_tot
ms_reg=ss_reg/df_reg
f_val=ms_reg/ms_res     # F_value
alpha=1-NC/100
fc=f.ppf(1-alpha,df_reg,df_res)     # F_critical
p_val=f.sf(f_val,df_reg,df_res)     # F_significant (p-value F)
se=np.sqrt(ms_res)     # Standard error of the estimate
R2ajus=1-(1-R2)*(nd-1)/(nd-par)     # Adjusted coefficient of determination (R-squared adj)
t_stats=params/std_errors     # Calculates the t-statistics of the parameters
p_values=2*(1-t.cdf(np.abs(t_stats),df_res))     # Calculates the p-values of the parameters
t_crit=t.ppf(1-(1-NC/100)/2,df_res)     # t_critical
std_devs=t_crit*std_errors     # Standard deviations
LI=params-std_devs     # Lower limit
LS=params+std_devs     # Upper limit
# Display parameter values
print(" ")
print("RESULTS:")
print(f"Number of observations: n = {nd:.0f}")
print(f"Number of parameters: par = {par:.0f}")
print(f"Confidence level (%): CL = {NC:.1f}")
print(f"Coefficient of determination (R Square): R^2 = {R2:.5f}")
print(f"Adjusted coefficient of determination (Adjusted R Square): R^2_adj = {R2ajus:.5f}")
print(f"Residual standard error: Se = {se:.4e}")
print('_____________________________________________________________________________')
# ANOVA
parametros_anova=['Regression','Residual','Total']
gl=[df_reg,df_res,df_tot]
SQ=[ss_reg,ss_res,ss_tot]
QM=[ms_reg,ms_res,""]
Fcalc=[f_val,"",""]
Fcrit=[fc,"",""]
Fsig=[p_val,"",""]
# DataFrame ANOVA
df_anova=pd.DataFrame({'ANOVA':parametros_anova,'df':gl,'SS':SQ,'MS':QM,'F':Fcalc,'Fcrit':Fcrit,'Fsig':Fsig})
print(df_anova.to_string(index=False))
```

```
print('_____
_____')
# Regression parameters
parametros_regressao=['a','b']
par_val=[a_fit,b_fit]
erro_padrao=[std_errors[0],std_errors[1]]
t_stat=[t_stats[0],t_stats[1]]
p_value=[p_values[0],p_values[1]]
std_dev=[t_crit*std_errors[0],t_crit*std_errors[1]]
lci=[LI[0],LI[1]]
lcs=[LS[0],LS[1]]
# DataFrame regression parameters
df_pr=pd.DataFrame({'REGRESSION':parametros_regressao,'Parameters':par_val,'Standard    Error':erro_padrao,'t-
Stat':t_stat,'P-value':p_value,'Confidence int':std_dev,'Lower conf lim':lci,'Upper conf lim':lcs})
print(df_pr.to_string(index=False))
print(" ")
# Chart with the data and model
x_min=0     # np.min(x)
x_max=np.max(x)
xreg=np.linspace(x_min,x_max,200)     # Replace x-min and x-max with number to change value
yreg=a_fit*(1-np.exp(-b_fit*xreg))
y_max=np.max(yreg)
plt.figure(figsize=(5,3))     # Change the size of the chart
plt.scatter(x,y,s=10,color="black",label="Experimental data")
plt.plot(xreg,yreg,color="red",label="Regression")
plt.title("Pseudo first-order adsorption kinetics",fontsize=10)
plt.xlabel("time (min)",fontsize=10)
plt.ylabel("qt (mg/g)",fontsize=10)
plt.legend(loc='lower right',fontsize=8)
plt.tick_params(axis='both',labelsize=8)
plt.xlim(x_min,x_max*1.1)     # Customize x-axis limits  (0, 600)
plt.ylim(0,y_max*1.1)       # Customize y-axis limits  (0, 400)
plt.show()
```

Table 1S. Results obtained from nonlinear regression of pseudo-first order kinetics and statistical analysis of parameters in Jupyter Notebook

Results							
Number of observations: n = 13							
Number of parameters: par = 2							
Confidence level (%): CL = 95.0							
Coefficient of determination (R square): R^2 = 0.92658							
Adjusted coefficient of determination (Adjusted R square): R^2_adj = 0.91990							
Residual standard error: Se = 2.4300×10^{1}							
ANOVA	df	SS	MS	*F*	*F*crit	*F*sig	
Regression	1	8.1968×10^{4}	8.1968×10^{4}	1.3881×10^{2}	4.8443	1.4043×10^{-7}	
Residual	11	6.4953×10^{3}	5.9049×10^{2}				
Total	12	8.8463×10^{4}					
Regression	Parameters	Standard error	*t*-Stat	*p*-value	Confidence int	Lower conf lim	Upper conf lim
a	3.6015×10^{2}	8.1541	44.169	9.7700×10^{-14}	1.7947×10^{1}	3.4221×10^{2}	3.7810×10^{2}
b	1.0290×10^{-1}	1.2325×10^{-2}	8.3490	4.3423×10^{-6}	2.7128×10^{-2}	7.5776×10^{-2}	1.3003×10^{-1}

n: number of experimental equilibrium points; R^2: coefficient of determination; ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

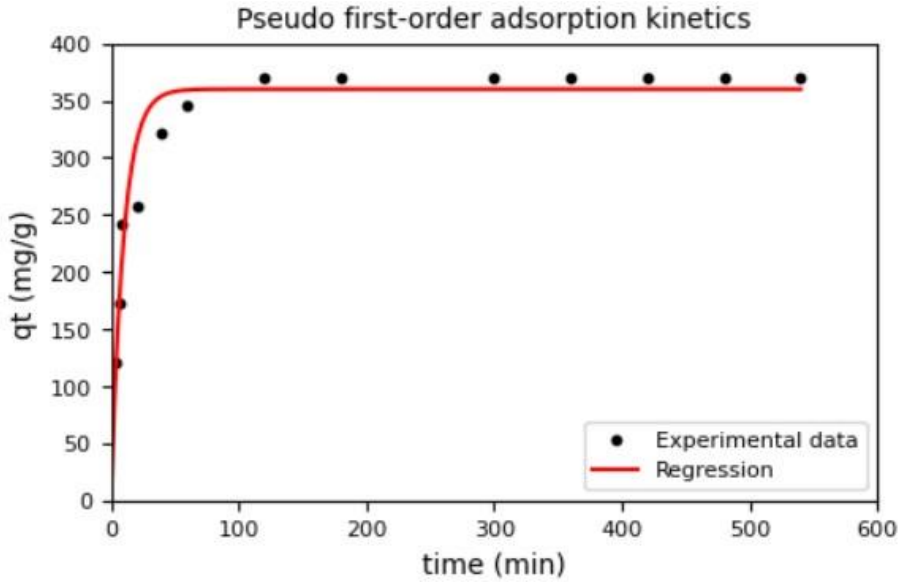

Figure 1S. *Graph obtained from nonlinear regression of pseudo-first order kinetics in Jupyter Notebook*

Template for nonlinear regression of pseudo-second order adsorption kinetics

```
# JUPYTER NOTEBOOK SOFTWARE_LINES OF CODE IN PYTHON     (Line 1)
# TEMPLATE FOR NONLINEAR REGRESSION_PSEUDO SECOND-ORDER ADSORPTION KINETICS
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.stats import f
from scipy.stats import t
pd.set_option('display.float_format','{:.4e}'.format)     # Table values in scientific notation with 4 decimal places
pd.set_option('display.max_columns',None)     # Code to not break the columns
pd.set_option('display.width',200)     # Code to increase columns in table
# Input experimental data
x=np.array([3, 6, 9, 20, 40, 60, 120, 180, 300, 360, 420, 480, 540])
y=np.array([121.25, 172.91, 242.49, 257.25, 322.04, 345.65, 370.15, 370.15, 370.15, 370.15, 370.15, 370.15,
370.15])
# linearized data
u=x
v=x/y
# terms for linearization
n=np.size(u)
Su=np.sum(u)
Sv=np.sum(v)
Suv=np.sum(u*v)
Suu=np.sum(u*u)
Svv=np.sum(v*v)
angular=(Su*Sv-n*Suv)/(Su**2-n*Suu)
linear=(Su*Suv-Sv*Suu)/(Su**2-n*Suu)
# Define the model function as reg
def reg(x,a,b):return(a**2*b*x)/(1+a*b*x)
initial=[1/angular,angular**2/linear]     # initial is the initial estimate for the parameters
NC=95     # confidence level
# Fit the data to the model equation using the curve_fit function
params,covariance=curve_fit(reg,x,y,initial)
# Extract the parameters
a_fit=params[0]
```

```
b_fit=params[1]
std_errors=np.sqrt(np.diag(covariance))     # Calculate the standard error of the parameters
# Calculate model statistic
y_pred=reg(x,a_fit,b_fit)
SQR=np.sum((y-y_pred)**2)     # Sum of squares of residuals
R2=1-(SQR/np.sum((y-np.mean(y))**2))     # Coefficient of determination (R-squared)
ss_res=np.sum((y-reg(x,*params))**2)
ss_tot=np.sum((y-np.mean(y))**2)
ss_reg=ss_tot-ss_res
nd=len(y)     # number of observations
par=len(params)     # number of parameters
df_tot=len(y)-1
df_reg=par-1
df_res=df_tot-df_reg
ms_res=ss_res/df_res
ms_tot=ss_tot/df_tot
ms_reg=ss_reg/df_reg
f_val=ms_reg/ms_res     # F_value
alpha=1-NC/100
fc=f.ppf(1-alpha,df_reg,df_res)     # F_critical
p_val=f.sf(f_val,df_reg,df_res)     # F_significant (p-value F)
se=np.sqrt(ms_res)     # Standard error of the estimate
R2ajus=1-(1-R2)*(nd-1)/(nd-par)     # Adjusted coefficient of determination (R-squared adj)
t_stats=params/std_errors     # Calculates the t-statistics of the parameters
p_values=2*(1-t.cdf(np.abs(t_stats),df_res))     # Calculates the p-values of the parameters
t_crit=t.ppf(1-(1-NC/100)/2,df_res)     # t_critical
std_devs=t_crit*std_errors     # Standard deviations
LI=params-std_devs     # Lower limit
LS=params+std_devs     # Upper limit
# Display parameter values
print(" ")
print("RESULTS:")
print(f"Number of observations: n = {nd:.0f}")
print(f"Number of parameters: par = {par:.0f}")
print(f"Confidence level (%): CL = {NC:.1f}")
print(f"Coefficient of determination (R Square): R^2 = {R2:.5f}")
print(f"Adjusted coefficient of determination (Adjusted R Square): R^2_adj = {R2ajus:.5f}")
print(f"Residual standard error: Se = {se:.4e}")
```

```
print('_____')
# ANOVA
parametros_anova=['Regression','Residual','Total']
gl=[df_reg,df_res,df_tot]
SQ=[ss_reg,ss_res,ss_tot]
QM=[ms_reg,ms_res,""]
Fcalc=[f_val,"",""]
Fcrit=[fc,"",""]
Fsig=[p_val,"",""]
# DataFrame ANOVA
df_anova=pd.DataFrame({'ANOVA':parametros_anova,'df':gl,'SS':SQ,'MS':QM,'F':Fcalc,'Fcrit':Fcrit,'Fsig':Fsig})
print(df_anova.to_string(index=False))
print('_____
_____')
# Regression parameters
parametros_regressao=['a', 'b']
par_val=[a_fit, b_fit]
erro_padrao=[std_errors[0],std_errors[1]]
t_stat=[t_stats[0],t_stats[1]]
p_value=[p_values[0],p_values[1]]
std_dev=[t_crit*std_errors[0],t_crit*std_errors[1]]
lci=[LI[0],LI[1]]
lcs=[LS[0],LS[1]]
# DataFrame regression parameters
df_pr=pd.DataFrame({'REGRESSION':parametros_regressao,'Parameters':par_val,'Standard  Error':erro_padrao,'t-
Stat':t_stat,'P-value':p_value,'Confidence int':std_dev,'Lower conf lim':lci,'Upper conf lim':lcs})
print(df_pr.to_string(index=False))
print(" ")
# Chart with the data and model
x_min=0    # np.min(x)
x_max=np.max(x)
xreg=np.linspace(x_min,x_max,200)    # Replace x-min and x-max with number to change value
yreg=a_fit**2*b_fit*xreg/(1+a_fit*b_fit*xreg)
y_max=np.max(yreg)
plt.figure(figsize=(5,3))    # Change the size of the chart
plt.scatter(x,y,s=10,color="black",label="Experimental data")
plt.plot(xreg,yreg,color="red",label="Regression")
plt.title("Pseudo second-order adsorption kinetics",fontsize=10)
```

```
plt.xlabel("time (min)",fontsize=10)
plt.ylabel("qt (mg/g)",fontsize=10)
plt.legend(loc='lower right',fontsize=8)
plt.tick_params(axis='both',labelsize=8)
plt.xlim(x_min,x_max*1.1)     # Customize x-axis limits  (0, 600)
plt.ylim(0,y_max*1.1)         # Customize y-axis limits  (0, 400)
plt.show()
```

Table 2S. Results obtained from nonlinear regression of pseudo-second order kinetics and statistical analysis of parameters in Jupyter Notebook

Results							
Number of observations: n = 13							
Number of parameters: par = 2							
Confidence level (%): CL = 95.0							
Coefficient of determination (R square): R^2 = 0.98183							
Adjusted coefficient of determination (Adjusted R square): R^2_adj = 0.98018							
Residual standard error: Se = 1.2088×10^{1}							
ANOVA	df	SS	MS	*F*	*F*crit	*F*sig	
Regression	1	8.6856×10^{4}	8.6856×10^{4}	5.9437×10^{2}	4.8443	6.3283×10^{-11}	
Residual	11	1.6074×10^{3}	1.4613×10^{2}				
Total	12	8.8463×10^{4}					
Regression	Parameters	Standard error	*t*-Stat	*p*-value	Confidence int	Lower conf lim	Upper conf lim
a	3.7793×10^{2}	4.7016	8.0383×10^{1}	0	1.0348×10^{1}	3.6758×10^{2}	3.8828×10^{2}
b	4.0421×10^{-4}	3.5250×10^{-5}	1.1467×10^{1}	1.8531×10^{-7}	7.7585×10^{-5}	3.2663×10^{-4}	4.8180×10^{-4}

n: number of experimental equilibrium points; R^2: coefficient of determination; ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

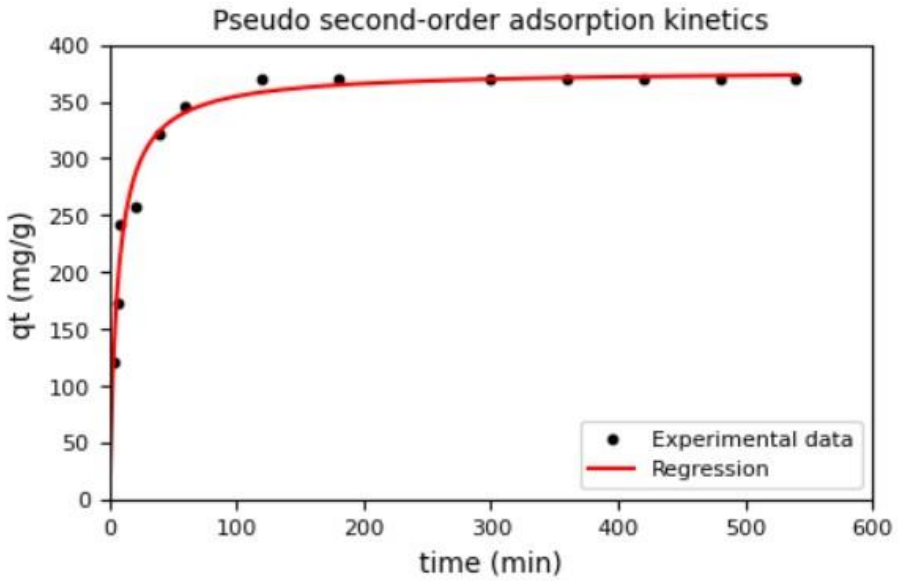

Figure 2S. *Graph obtained from nonlinear regression of pseudo-second order kinetics in Jupyter Notebook*

Template for nonlinear regression of Langmuir adsorption isotherm

```
# JUPYTER NOTEBOOK SOFTWARE_LINES OF CODE IN PYTHON     (Line 1)
# TEMPLATE FOR NONLINEAR REGRESSION_LANGMUIR ADSORPTION ISOTHERM
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy.stats import f
from scipy.stats import t
pd.set_option('display.float_format','{:.4e}'.format)     # Table values in scientific notation with 4 decimal places
pd.set_option('display.max_columns',None)     # Code to not break the columns
pd.set_option('display.width',200)     # Code to increase columns in table
# Input experimental data
x=np.array([306.14, 623.22, 892.78, 1252.3, 1540.6, 2277.8, 3128.2])
y=np.array([296.17, 370.15, 408.19, 438.5, 455.83, 477.66, 468.64])
# linearized data
u=1/x
v=1/y
# terms for linearization
n=np.size(u)
Su=np.sum(u)
Sv=np.sum(v)
Suv=np.sum(u*v)
Suu=np.sum(u*u)
Svv=np.sum(v*v)
angular=(Su*Sv-n*Suv)/(Su**2-n*Suu)
linear=(Su*Suv-Sv*Suu)/(Su**2-n*Suu)
# Define the model function as reg
def reg(x,a,b):return(a*b*x)/(1+b*x)
initial=[1/linear,linear/angular]     # initial is the initial estimate for the parameters
NC=95     # confidence level
# Fit the data to the model equation using the curve_fit function
params,covariance=curve_fit(reg,x,y,initial)
# Extract the parameters
a_fit=params[0]
b_fit=params[1]
```

```
std_errors=np.sqrt(np.diag(covariance))     # Calculate the standard error of the parameters
# Calculate model statistic
y_pred=reg(x,a_fit,b_fit)
SQR=np.sum((y-y_pred)**2)     # Sum of squares of residuals
R2=1-(SQR/np.sum((y-np.mean(y))**2))     # Coefficient of determination (R-squared)
ss_res=np.sum((y-reg(x,*params))**2)
ss_tot=np.sum((y-np.mean(y))**2)
ss_reg=ss_tot-ss_res
nd=len(y)     # number of observations
par=len(params)     # number of parameters
df_tot=len(y)-1
df_reg=par-1
df_res=df_tot-df_reg
ms_res=ss_res/df_res
ms_tot=ss_tot/df_tot
ms_reg=ss_reg/df_reg
f_val=ms_reg/ms_res     # F_value
alpha=1-NC/100
fc=f.ppf(1-alpha,df_reg,df_res)     # F_critical
p_val=f.sf(f_val,df_reg,df_res)     # F_significant (p-value F)
se=np.sqrt(ms_res)     # Standard error of the estimate
R2ajus=1-(1-R2)*(nd-1)/(nd-par)     # Adjusted coefficient of determination (R-squared adj)
t_stats=params/std_errors     # Calculates the t-statistics of the parameters
p_values=2*(1-t.cdf(np.abs(t_stats),df_res))     # Calculates the p-values of the parameters
t_crit=t.ppf(1-(1-NC/100)/2,df_res)     # t_critical
std_devs=t_crit*std_errors     # Standard deviations
LI=params-std_devs     # Lower limit
LS=params+std_devs     # Upper limit
# Display parameter values
print(" ")
print("RESULTS:")
print(f"Number of observations: n = {nd:.0f}")
print(f"Number of parameters: par = {par:.0f}")
print(f"Confidence level (%): CL = {NC:.1f}")
print(f"Coefficient of determination (R Square): R^2 = {R2:.5f}")
print(f"Adjusted coefficient of determination (Adjusted R Square): R^2_adj = {R2ajus:.5f}")
print(f"Residual standard error: Se = {se:.4e}")
print('_____')
```

```
# ANOVA
parametros_anova=['Regression','Residual','Total']
gl=[df_reg,df_res,df_tot]
SQ=[ss_reg,ss_res,ss_tot]
QM=[ms_reg,ms_res,""]
Fcalc=[f_val,"",""]
Fcrit=[fc,"",""]
Fsig=[p_val,"",""]
# DataFrame ANOVA
df_anova=pd.DataFrame({'ANOVA':parametros_anova,'df':gl,'SS':SQ,'MS':QM,'F':Fcalc,'Fcrit':Fcrit,'Fsig':Fsig})
print(df_anova.to_string(index=False))
print('_____
_____')
# Regression parameters
parametros_regressao=['a','b']
par_val=[a_fit,b_fit]
erro_padrao=[std_errors[0],std_errors[1]]
t_stat=[t_stats[0],t_stats[1]]
p_value=[p_values[0],p_values[1]]
std_dev=[t_crit*std_errors[0],t_crit*std_errors[1]]
lci=[LI[0],LI[1]]
lcs=[LS[0],LS[1]]
# DataFrame regression parameters
df_pr=pd.DataFrame({'REGRESSION':parametros_regressao,'Parameters':par_val,'Standard  Error':erro_padrao,'t-
Stat':t_stat,'P-value':p_value,'Confidence int':std_dev,'Lower conf lim':lci,'Upper conf lim':lcs})

print(df_pr.to_string(index=False))
print(" ")
# Chart with the data and model
x_min=0    # np.min(x)
x_max=np.max(x)
xreg=np.linspace(x_min,x_max,200)    # Replace x-min and x-max with number to change value
yreg=a_fit*b_fit*xreg/(1+b_fit*xreg)
y_max=np.max(yreg)
plt.figure(figsize=(5,3))    # Change the size of the chart
plt.scatter(x,y,s=10,color="black",label="Experimental data")
plt.plot(xreg,yreg,color="red",label="Regression")
plt.title("Langmuir adsorption isotherm",fontsize=10)
plt.xlabel("Ceq (mg/L)",fontsize=10)
```

```
plt.ylabel("qeq (mg/g)",fontsize=10)
plt.legend(loc='lower right',fontsize=8)
plt.tick_params(axis='both',labelsize=8)
plt.xlim(0,x_max*1.1)       # Customize x-axis limits  (0, 3200)
plt.ylim(0,y_max*1.1)       # Customize y-axis limits  (0, 500)
plt.show()
```

Table 3S. Results obtained from nonlinear regression of an adsorption isotherm according to the Langmuir model and statistical analysis of parameters in Jupyter Notebook

Results							
Number of observations: n = 7							
Number of parameters: par = 2							
Confidence level (%): CL = 95.0							
Coefficient of determination (R square): R^2 = 0.98686							
Adjusted coefficient of determination (Adjusted R square): R^2_adj = 0.98424							
Residual standard error: Se = 8.1347							
ANOVA	df	SS	MS	*F*	*F*crit	*F*sig	
Regression	1	2.4856×10^{4}	2.4856×10^{4}	3.7561×10^{2}	6.6079	6.7475×10^{-6}	
Residual	5	3.3087×10^{2}	6.6174×10^{1}				
Total	6	2.5187×10^{4}					
Regression	Parameters	Standard error	*t*-Stat	*p*-value	Confidence int	Lower conf lim	Upper conf lim
a	5.1625×10^{2}	7.4216	6.9560×10^{1}	1.1629×10^{-8}	1.9078×10^{1}	4.9717×10^{2}	5.3532×10^{2}
b	4.3280×10^{-3}	3.2833×10^{-4}	1.3182×10^{1}	4.4872×10^{-5}	8.4399×10^{-4}	3.4840×10^{-3}	5.1720×10^{-3}

n: number of experimental equilibrium points; R^2: coefficient of determination; ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

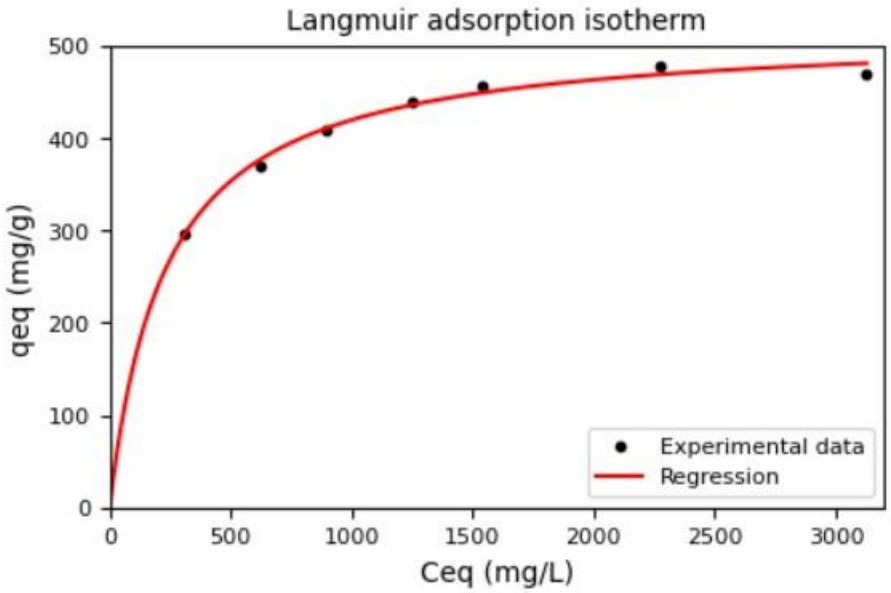

Figure 3S. *Graph obtained from nonlinear regression of an adsorption isotherm according to the Langmuir model in Jupyter Notebook*

RSTUDIO SOFTWARE

Procedure for running a nonlinear regression in RStudio software

1. Copy all the lines of code, starting with the title (line 1);
2. Paste them into the RStudio software in the workspace (numbered by lines);
3. Change the experimental data in RStudio;
4. Select all the lines of code in RStudio;
5. Click the run button or press the shortcut keys (ctrl + enter);
6. The processed result will be displayed in the console tab, the calculated values will be displayed in the environment tab, and the graph in the plots tab.

Template for nonlinear regression of pseudo-first order adsorption kinetics

```
# RSTUDIO SOFTWARE_LINES OF CODE IN R       (Line 1)
# TEMPLATE FOR NONLINEAR REGRESSION_PSEUDO FIRST-ORDER ADSORPTION KINETICS
options(digits=5)     # Defining the number of significant digits and the confidence level (%)
NC=95
# Input experimental data
x=c(3, 6, 9, 20, 40, 60, 120, 180, 300, 360, 420, 480, 540)
y=c(121.25, 172.91, 242.49, 257.25, 322.04, 345.65, 370.15, 370.15, 370.15, 370.15, 370.15, 370.15, 370.15)
# Data linearization
u<-(x)
v<-(-log(max(y)*1.008-y))
# Terms for linearization
nd<-(length(u))
Su<-(sum(u))
Sv<-(sum(v))
Suv<-(sum(u*v))
Suu<-(sum(u*u))
Svv<-(sum(v*v))
# Linear regression coefficients
angular<-((Su*Sv-nd*Suv)/(Su^2-nd*Suu))
linear<-((Su*Suv-Sv*Suu)/(Su^2-nd*Suu))
data<-(data.frame(x,y))     # Data matrix (x, y)
formula=y~a*(1-exp(-b*x))     # Equation to be adjusted to regression
start=list(a=exp(-linear),b=angular)     # Initial value of the iterative process
# Nonlinear least squares function, nls(), showing iterations
```

```
reg<-(nls(formula=formula,data=data,start=start,control=list(tol = 1E-7),trace=TRUE))
summary(reg)        # View results
coeficientes<-(summary(reg)$coefficients)     # Extract objects from the summary
# Format values in columns in scientific notation with 4 decimal places
valores_estimados<-(sprintf("%.4e",coeficientes[,1]))
erro_padrão<-(sprintf("%.4e",coeficientes[,2]))
valor_t<-(sprintf("%.4e",coeficientes[,3]))
valor_p<-(sprintf("%.4e",coeficientes[,4]))
# Anova table calculations
n<-(as.numeric(length(x)))     # number of observations
par<-(as.numeric(length(coef(reg))))     # number of parameters (coefficients of the model equation)
SQR<-(deviance(reg))
SQT<-(var(y)*(n-1))
SQreg<-(SQT-SQR)
Rsquared<-(1-SQR/SQT)
Rsquared_adj<-(1-((1-Rsquared)*(n-1)/(n-par)))
glreg<-(par-1)
glT<-(n-1)
glR<-(glT-glreg)
QMreg<-(SQreg/glreg)
QMR<-(SQR/glR)
SE<-(QMR^(1/2))
Fcalc<-(QMreg/QMR)
Fcrit<-(qf(NC/100,glreg,glR))
Fsig<-(1-pf(Fcalc,glreg,glR))
# Calculation of the parameters of the regression parameters table
par_a<-(as.numeric(valores_estimados[1]))
par_b<-(as.numeric(valores_estimados[2]))
epa<-(as.numeric(erro_padrão[1]))
epb<-(as.numeric(erro_padrão[2]))
tstat_a<-(as.numeric(valor_t[1]))
tstat_b<-(as.numeric(valor_t[2]))
valorp_a<-(as.numeric(valor_p[1]))
valorp_b<-(as.numeric(valor_p[2]))
invt<-(qt(1-(1-NC/100)/2,glR))
dpa<-(epa*invt)
dpb<-(epb*invt)
LI_a<-(par_a-dpa)
```

```
LS_a<-(par_a+dpa)
LI_b<-(par_b-dpb)
LS_b<-(par_b+dpb)
# Creating the table - Statistic
statistic_table<-(data.frame(Statistical_Analysis=c("Number of Observations, n =","Number of parameters, par =","Conficence level, CL =","Coefficient of determination (R Square), R^2 =","Adjusted coefficient of determination (Adjusted R Square), R^2_adj =","Residual standard error, Se ="),"value" = c(sprintf("%.0f",n),sprintf("%.0f",par),sprintf("%.1f",NC),sprintf("%.5f",Rsquared),sprintf("%.5f",Rsquared_adj),sprintf("%.4e",SE))))
# Creating the table - ANOVA
anova_table<-
(data.frame(ANOVA=c("Regression","Residual","Total"),df=c(glreg,glR,glT),SS=c(sprintf("%.4e",SQreg),sprintf("%.4e",SQR),sprintf("%.4e",SQT)),MS=c(sprintf("%.4e",QMreg),sprintf("%.4e",QMR),"
"),F=c(sprintf("%.4e",Fcalc)," "," "),Fcrit=c(sprintf("%.4e",Fcrit)," "," "),Fsig=c(sprintf("%.4e",Fsig)," "," ")))
# Creating the regression parameters table
regression_table<-
(data.frame(REGRESSION=c("a","b"),Parameters=c(sprintf("%.4e",par_a),sprintf("%.4e",par_b)),Standard_Error=c(sprintf("%.4e",epa),sprintf("%.4e",epb)),t_Stat=c(sprintf("%.4e",tstat_a),sprintf("%.4e",tstat_b)),p_Value=c(sprintf("%.4e",valorp_a),sprintf("%.4e",valorp_b)),Confidence_int=c(sprintf("%.4e",dpa),sprintf("%.4e",dpb)),Lower_Lim=c(sprintf("%.4e",LI_a),sprintf("%.4e",LI_b)),Upper_Lim=c(sprintf("%.4e",LS_a),sprintf("%.4e",LS_b))))
options(width=200)      # change the width of tables
# RESULTS
print(statistic_table)    # Statistic – Table
print(anova_table)    # ANOVA – Table
print(regression_table)    # Regression parameters – Table
# Getting values for regression curve
x_min<-(0)      # Calculating x_min        (can be changed by a number, for example 0)
x_max<-(max(x))     # Calculating x_max       (can be changed to any number)
xreg<-(seq(x_min,x_max,length.out=200))     # Calculating xreg
yreg<-(par_a*(1-exp(-par_b*xreg)))     # Calculating yreg
datareg=data.frame(xreg,yreg)      # Data matrix (xreg, yreg)
# Plotting the data
par(cex.main=0.95)        # Change the font size of the chart title (1 = standard, >1 increases, <1 decreases)
plot(x,y,type="p",col="black",main="Adsorption kinetic - Nonlinear pseudo first-order",xlab="time (min)",ylab="qt (mg/g)",pch=19,cex=1.0,xlim=c(0,600),ylim=c(0, 400))        # You can change the chart title, axis names and limits
lines(xreg,yreg,col="red",lwd=2.0)   # Regression line on the graph
# Inserting the legend
legend("bottomright",                # Position of the legend
```

```
legend=c("Experimental data","Regression"),  # Text for each item
col=c("black","red"),         # Colors corresponding to the items
pch=c(19,NA),                  # Point type (19 for points, NA for line)
lwd=c(NA,2.0),                # Line width (NA for points, 2.0 for line)
cex=0.8,                      # Change the font size of the legend (1 = standard, >1 increases, <1 decreases)
bty="n")                       # Remove the box around the legend
```

Table 4S. Results obtained from nonlinear regression of pseudo-first order kinetics and statistical analysis of parameters in RStudio

> # RESULTS

> print(statistic_table) # Statistic – Table

	Statistical_Analysis	value
1	Number of observations, n =	13
2	Number of parameters, par =	2
3	Conficence level, CL =	95.0
4	Coefficient of determination (R square), R^2 =	0.92658
5	Adjusted coefficient of determination (Adjusted R square), R^2_adj =	0.91990
6	Residual standard error, Se =	2.4300×10^{1}

> print(anova_table) # ANOVA – Table

	ANOVA	df	SS	MS	*F*	*F*crit	*F*sig
1	Regression	1	8.1968×10^{4}	8.1968×10^{4}	1.3881×10^{2}	4.8443	1.4043×10^{-7}
2	Residual	11	6.4953×10^{3}	5.9049×10^{2}			
3	Total	12	8.8463×10^{4}				

> print(regression_table) # Regression parameters – Table

	Regression	Parameters	Standard_Error	*t*_Stat	*p*_Value	Confidence int	Lower_Lim	Upper_Lim
1	a	3.6015×10^{2}	8.1541	44.169	9.7814×10^{-14}	1.7947×10^{1}	3.4220×10^{2}	3.7810×10^{2}
2	b	1.0290×10^{-1}	1.2325×10^{-2}	8.3489	4.3429×10^{-6}	2.7127×10^{-2}	7.5773×10^{-2}	1.3003×10^{-1}

n: number of experimental equilibrium points; R^2: coefficient of determination; ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

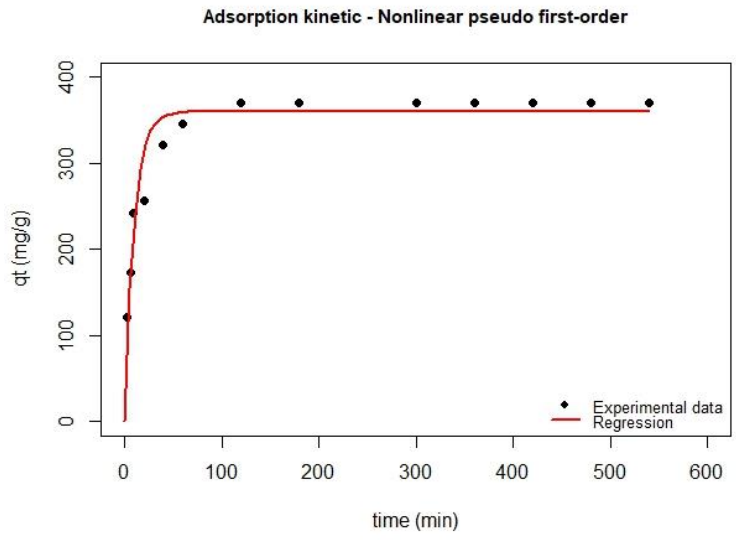

Figure 4S. *Graph obtained from nonlinear regression of pseudo-first order kinetics in RStudio*

Template for nonlinear regression of pseudo-second order adsorption kinetics

```
# RSTUDIO SOFTWARE_LINES OF CODE IN R       (Line 1)
# TEMPLATE FOR NONLINEAR REGRESSION_PSEUDO SECOND-ORDER ADSORPTION KINETICS
options(digits=5)     # Defining the number of significant digits and the confidence level (%)
NC=95
# Input experimental data
x=c(3, 6, 9, 20, 40, 60, 120, 180, 300, 360, 420, 480, 540)
y=c(121.25, 172.91, 242.49, 257.25, 322.04, 345.65, 370.15, 370.15, 370.15, 370.15, 370.15, 370.15, 370.15)
# Data linearization
u<-(x)
v<-(x/y)
# Terms for linearization
nd<-(length(u))
Su<-(sum(u))
Sv<-(sum(v))
Suv<-(sum(u*v))
Suu<-(sum(u*u))
Svv<-(sum(v*v))
# Linear regression coefficients
angular<-((Su*Sv-nd*Suv)/(Su^2-nd*Suu))
linear<-((Su*Suv-Sv*Suu)/(Su^2-nd*Suu))
data<-(data.frame(x,y))     # Data matrix (x, y)
formula=y~(a^2*b*x)/(1+a*b*x)     # Equation to be adjusted to regression
start=list(a=1/angular,b=angular^2/linear)     # Initial value of the iterative process
# Nonlinear least squares function, nls(), showing iterations
reg<-(nls(formula=formula,data=data,start=start,control=list(tol=1E-7),trace=TRUE))
summary(reg)        # View results
coeficientes<-(summary(reg)$coefficients)     # Extract objects from the summary
# Format values in columns in scientific notation with 4 decimal places
valores_estimados<-(sprintf("%.4e",coeficientes[,1]))
erro_padrão<-(sprintf("%.4e",coeficientes[,2]))
valor_t<-(sprintf("%.4e",coeficientes[,3]))
valor_p<-(sprintf("%.4e",coeficientes[,4]))
# Anova table calculations
n<-(as.numeric(length(x)))     # number of observations
par<-(as.numeric(length(coef(reg))))     # number of parameters (coefficients of the model equation)
SQR<-(deviance(reg))
```

```
SQT<-(var(y)*(n-1))
SQreg<-(SQT-SQR)
Rsquared<-(1-SQR/SQT)
Rsquared_adj<-(1-((1-Rsquared)*(n-1)/(n-par)))
glreg<-(par-1)
glT<-(n-1)
glR<-(glT-glreg)
QMreg<-(SQreg/glreg)
QMR<-(SQR/glR)
SE<-(QMR^(1/2))
Fcalc<-(QMreg/QMR)
Fcrit<-(qf(NC/100,glreg,glR))
Fsig<-(1-pf(Fcalc,glreg,glR))
# Calculation of the parameters of the regression parameters table
par_a<-(as.numeric(valores_estimados[1]))
par_b<-(as.numeric(valores_estimados[2]))
epa<-(as.numeric(erro_padrão[1]))
epb<-(as.numeric(erro_padrão[2]))
tstat_a<-(as.numeric(valor_t[1]))
tstat_b<-(as.numeric(valor_t[2]))
valorp_a<-(as.numeric(valor_p[1]))
valorp_b<-(as.numeric(valor_p[2]))
invt<-(qt(1-(1-NC/100)/2,glR))
dpa<-(epa*invt)
dpb<-(epb*invt)
LI_a<-(par_a-dpa)
LS_a<-(par_a+dpa)
LI_b<-(par_b-dpb)
LS_b<-(par_b+dpb)
# Creating the table - Statistic
statistic_table<-(data.frame(Statistical_Analysis=c("Number of Observations, n =","Number of parameters, par =","Conficence level, CL =","Coefficient of determination (R Square), R^2 =","Adjusted coefficient of determination (Adjusted R Square), R^2_adj =","Residual standard error, Se ="),"value"=c(sprintf("%.0f",n),sprintf("%.0f",par),sprintf("%.1f",NC),sprintf("%.5f",Rsquared),sprintf("%.5f",Rsquared_adj),sprintf("%.4e",SE))))
# Creating the table – ANOVA
anova_table<-
(data.frame(ANOVA=c("Regression","Residual","Total"),df=c(glreg,glR,glT),SS=c(sprintf("%.4e",SQreg),sprintf("
```

```
%.4e",SQR),sprintf("%.4e",SQT)),MS=c(sprintf("%.4e",QMreg),sprintf("%.4e",QMR),"
"),F=c(sprintf("%.4e",Fcalc)," "," "),Fcrit=c(sprintf("%.4e",Fcrit)," "," "),Fsig=c(sprintf("%.4e",Fsig)," "," ")))
# Creating the regression parameters table
regression_table<-
(data.frame(REGRESSION=c("a","b"),Parameters=c(sprintf("%.4e",par_a),sprintf("%.4e",par_b)),Standard_Error
=c(sprintf("%.4e",epa),sprintf("%.4e",epb)),t_Stat=c(sprintf("%.4e",tstat_a),sprintf("%.4e",tstat_b)),p_Value=c(spri
ntf("%.4e",valorp_a),sprintf("%.4e",valorp_b)),Confidence_int=c(sprintf("%.4e",dpa),sprintf("%.4e",dpb)),Lower_
Lim=c(sprintf("%.4e",LI_a),sprintf("%.4e",LI_b)),Upper_Lim=c(sprintf("%.4e",LS_a),sprintf("%.4e",LS_b))))
options(width=200)      # change the width of tables
# RESULTS
print(statistic_table)    # Statistic - Table
print(anova_table)    # ANOVA - Table
print(regression_table)    # Regression parameters - Table
# Getting values for regression curve
x_min<-(0)      # Calculating x_min        (can be changed by a number, for example 0)
x_max<-(max(x))     # Calculating x_max       (can be changed to any number)
xreg<-(seq(x_min,x_max,length.out=200))     # Calculating xreg
yreg<-(par_a^2*par_b*xreg/(1+par_a*par_b*xreg))     # Calculating yreg
datareg=data.frame(xreg,yreg)      # Data matrix (xreg, yreg)
# Plotting the data
par(cex.main=0.95)        # Change the font size of the chart title (1 = standard, >1 increases, <1 decreases)
plot(x,y,type="p",col="black",main="Adsorption kinetic - Nonlinear pseudo second-order",xlab="time
(min)",ylab="qt (mg/g)",pch=19,cex=1.0,xlim=c(0,600),ylim=c(0,400))         # You can change the chart title, axis
names and limits
lines(xreg,yreg,col="red",lwd=2.0)   # Regression line on the graph
# Inserting the legend
legend("bottomright",                # Position of the legend
       legend=c("Experimental data","Regression"),  # Text for each item
       col=c("black","red"),         # Colors corresponding to the items
       pch=c(19,NA),                 # Point type (19 for points, NA for line)
       lwd=c(NA,2.0),                # Line width (NA for points, 2.0 for line)
       cex=0.8,                    # Change the font size of the legend (1 = standard, >1 increases, <1 decreases)
       bty="n")                      # Remove the box around the legend
```

Table 5S. Results obtained from nonlinear regression of pseudo-second order kinetics and statistical analysis of the parameters in RStudio

> # RESULTS

> print(statistic_table) # Statistic – Table

	Statistical_Analysis	value
1	Number of Observations, n =	13
2	Number of parameters, par =	2
3	Conficence level, CL =	95.0
4	Coefficient of determination (R square), R^2 =	0.98183
5	Adjusted coefficient of determination (Adjusted R square), R^2_adj =	0.98018
6	Residual standard error, Se =	1.2088×10^{1}

> print(anova_table) # ANOVA – Table

	ANOVA	df	SS	MS	*F*	*F*crit	*F*sig
1	Regression	1	8.6856×10^{4}	8.6856×10^{4}	5.9437×10^{2}	4.8443	6.3282×10^{-11}
2	Residual	11	1.6074×10^{3}	1.4613×10^{2}			
3	Total	12	8.8463×10^{4}				

> print(regression_table) # Regression parameters – Table

	Regression	Parameters	Standard_Error	*t*_Stat	*p*_Value	Confidence int.	Lower_Lim	Upper_Lim
1	a	3.7793×10^{2}	4.7016	8.0383×10^{1}	1.3755×10^{-16}	1.0348×10^{1}	3.6758×10^{2}	3.8828×10^{2}
2	b	4.0421×10^{-4}	3.5250×10^{-5}	1.1467×10^{1}	1.8532×10^{-7}	7.7585×10^{-5}	3.2663×10^{-4}	4.8179×10^{-4}

n: number of experimental equilibrium points; R^2: coefficient of determination; ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

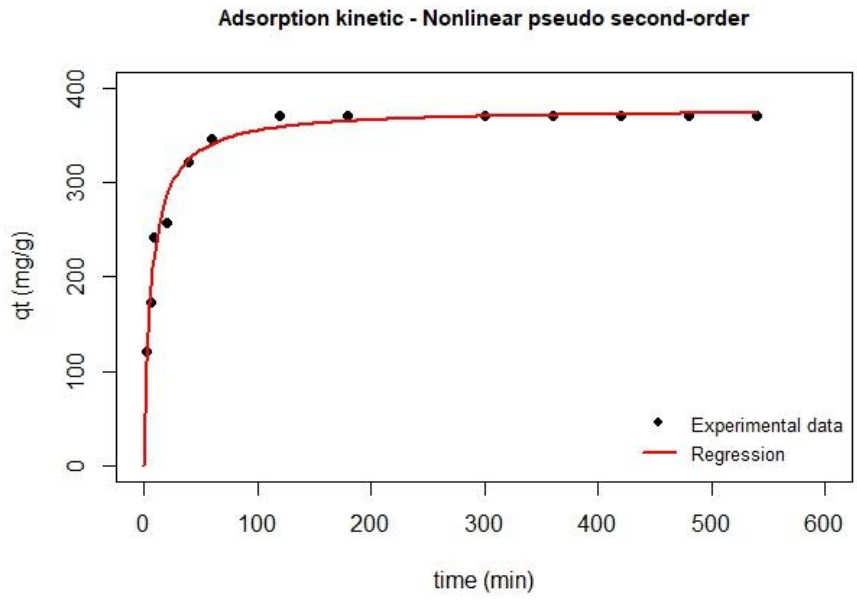

Figure 5S. *Graph obtained from nonlinear regression of pseudo-second order kinetics in RStudio*

Template for nonlinear regression of Langmuir adsorption isotherm

```
# RSTUDIO SOFTWARE_LINES OF CODE IN R       (Line 1)
# TEMPLATE FOR NONLINEAR REGRESSION_LANGMUIR ADSORPTION ISOTHERM
options(digits=5)     # Defining the number of significant digits and the confidence level (%)
NC=95
# Input experimental data
x=c(306.14, 623.22, 892.78, 1252.3, 1540.6, 2277.8, 3128.2)
y=c(296.17, 370.15, 408.19, 438.5, 455.83, 477.66, 468.64)
# Data linearization
u<-(1/x)
v<-(1/y)
# Terms for linearization
nd<-(length(u))
Su<-(sum(u))
Sv<-(sum(v))
Suv<-(sum(u*v))
Suu<-(sum(u*u))
Svv<-(sum(v*v))
# Linear regression coefficients
angular<-((Su*Sv-nd*Suv)/(Su^2-nd*Suu))
linear<-((Su*Suv-Sv*Suu)/(Su^2-nd*Suu))
data<-(data.frame(x,y))     # Data matrix (x, y)
formula=y~(a*b*x)/(1+b*x)     # Equation to be adjusted to regression
start=list(a=1/linear,b=linear/angular)     # Initial value of the iterative process
# Nonlinear least squares function, nls(), showing iterations
reg<-(nls(formula=formula,data=data,start=start,control=list(tol=1E-7),trace=TRUE))
summary(reg)        # View results
coeficientes<-(summary(reg)$coefficients)     # Extract objects from the summary
# Format values in columns in scientific notation with 4 decimal places
valores_estimados<-(sprintf("%.4e",coeficientes[,1]))
erro_padrão<-(sprintf("%.4e",coeficientes[,2]))
valor_t<-(sprintf("%.4e",coeficientes[,3]))
valor_p<-(sprintf("%.4e",coeficientes[,4]))
# Anova table calculations
n<-(as.numeric(length(x)))     # number of observations
par<-(as.numeric(length(coef(reg))))     # number of parameters (coefficients of the model equation)
SQR<-(deviance(reg))
```

```
SQT<-(var(y)*(n-1))
SQreg<-(SQT-SQR)
Rsquared<-(1-SQR/SQT)
Rsquared_adj<-(1-((1-Rsquared)*(n-1)/(n-par)))
glreg<-(par-1)
glT<-(n-1)
glR<-(glT-glreg)
QMreg<-(SQreg/glreg)
QMR<-(SQR/glR)
SE<-(QMR^(1/2))
Fcalc<-(QMreg/QMR)
Fcrit<-(qf(NC/100,glreg,glR))
Fsig<-(1-pf(Fcalc,glreg,glR))
# Calculation of the parameters of the regression parameters table
par_a<-(as.numeric(valores_estimados[1]))
par_b<-(as.numeric(valores_estimados[2]))
epa<-(as.numeric(erro_padrão[1]))
epb<-(as.numeric(erro_padrão[2]))
tstat_a<-(as.numeric(valor_t[1]))
tstat_b<-(as.numeric(valor_t[2]))
valorp_a<-(as.numeric(valor_p[1]))
valorp_b<-(as.numeric(valor_p[2]))
invt<-(qt(1-(1-NC/100)/2,glR))
dpa<-(epa*invt)
dpb<-(epb*invt)
LI_a<-(par_a-dpa)
LS_a<-(par_a+dpa)
LI_b<-(par_b-dpb)
LS_b<-(par_b+dpb)
# Creating the table - Statistic
statistic_table<-(data.frame(Statistical_Analysis=c("Number of Observations, n =","Number of parameters, par =","Conficence level, CL =","Coefficient of determination (R Square), R^2 =","Adjusted coefficient of determination (Adjusted R Square), R^2_adj =","Residual standard error, Se ="),"value"=c(sprintf("%.0f",n),sprintf("%.0f",par),sprintf("%.1f",NC),sprintf("%.5f",Rsquared),sprintf("%.5f",Rsquared_adj),sprintf("%.4e",SE))))
# Creating the table – ANOVA
anova_table<-
(data.frame(ANOVA=c("Regression","Residual","Total"),df=c(glreg,glR,glT),SS=c(sprintf("%.4e",SQreg),sprintf("
```

```
%.4e",SQR),sprintf("%.4e",SQT)),MS=c(sprintf("%.4e",QMreg),sprintf("%.4e",QMR),"
"),F=c(sprintf("%.4e",Fcalc)," "," "),Fcrit=c(sprintf("%.4e",Fcrit)," "," "),Fsig=c(sprintf("%.4e",Fsig)," "," ")))
# Creating the regression parameters table
regression_table<-
(data.frame(REGRESSION=c("a","b"),Parameters=c(sprintf("%.4e",par_a),sprintf("%.4e",par_b)),Standard_Error
=c(sprintf("%.4e",epa),sprintf("%.4e",epb)),t_Stat=c(sprintf("%.4e",tstat_a),sprintf("%.4e",tstat_b)),p_Value=c(spri
ntf("%.4e",valorp_a),sprintf("%.4e",valorp_b)),Confidence_int=c(sprintf("%.4e",dpa),sprintf("%.4e",dpb)),Lower_
Lim=c(sprintf("%.4e",LI_a),sprintf("%.4e",LI_b)),Upper_Lim=c(sprintf("%.4e",LS_a),sprintf("%.4e",LS_b))))
options(width=200)      # change the width of tables
# RESULTS
print(statistic_table)    # Statistic – Table
print(anova_table)    # ANOVA – Table
print(regression_table)    # Regression parameters – Table
# Getting values for regression curve
x_min<-(0)      # Calculating x_min        (can be changed by a number, for example 0)
x_max<-(max(x))     # Calculating x_max       (can be changed to any number)
xreg<-(seq(x_min,x_max,length.out = 200))     # Calculating xreg
yreg<-(par_a*par_b*xreg/(1+par_b*xreg))     # Calculating yreg
datareg=data.frame(xreg,yreg)      # Data matrix (xreg, yreg)
# Plotting the data
par(cex.main=0.95)        # Change the font size of the chart title (1 = standard, >1 increases, <1 decreases)
plot(x,y,type="p",col="black",main="Nonlinear Regression - Langmuir Isotherm",xlab="Ceq (mg/L)",ylab="qeq
(mg/g)",pch=19,cex=1.0,xlim=c(0,3200),ylim=c(0,500))        # You can change the chart title, axis names and limits
lines(xreg,yreg,col="red",lwd=2.0)   # Regression line on the graph
# Inserting the legend
legend("bottomright",                # Position of the legend
       legend=c("Experimental data","Regression"),  # Text for each item
       col=c("black","red"),         # Colors corresponding to the items
       pch=c(19,NA),                 # Point type (19 for points, NA for line)
       lwd=c(NA,2.0),                # Line width (NA for points, 2.0 for line)
       cex=0.8,                      # Change the font size of the legend (1 = standard, >1 increases, <1 decreases)
       bty="n")                      # Remove the box around the legend
```

Table 6S. Results obtained from nonlinear regression of an adsorption isotherm according to the Langmuir model and statistical analysis of the parameters in RStudio

> # RESULTS

> print(statistic_table) # Statistic – Table

	Statistical_Analysis	value
1	Number of Observations, n =	7
2	Number of parameters, par =	2
3	Conficence level, CL =	95.0
4	Coefficient of determination (R square), R^2 =	0.98686
5	Adjusted coefficient of determination (Adjusted R square), R^2_adj =	0.98424
6	Residual standard error, Se =	8.1347

> print(anova_table) # ANOVA – Table

	ANOVA	df	SS	MS	*F*	*F*crit	*F*sig
1	Regression	1	2.4856×10^{4}	2.4856×10^{4}	3.7561×10^{2}	6.6079	6.7475×10^{-6}
2	Residual	5	3.3087×10^{2}	6.6174×10^{1}			
3	Total	6	2.5187×10^{4}				

> print(regression_table) # Regression parameters – Table

	Regression	Parameters	Standard_Error	*t*_Stat	*p*_Value	Confidence int	Lower_Lim	Upper_Lim
1	a	5.1625×10^{2}	7.4216	6.9560×10^{1}	1.1629×10^{-8}	1.9078×10^{1}	4.9717×10^{2}	5.3533×10^{2}
2	b	4.3280×10^{-3}	3.2833×10^{-4}	1.3182×10^{1}	4.4872×10^{-5}	8.4400×10^{-4}	3.4840×10^{-3}	5.1720×10^{-3}

n: number of experimental equilibrium points; R^2: coefficient of determination; ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

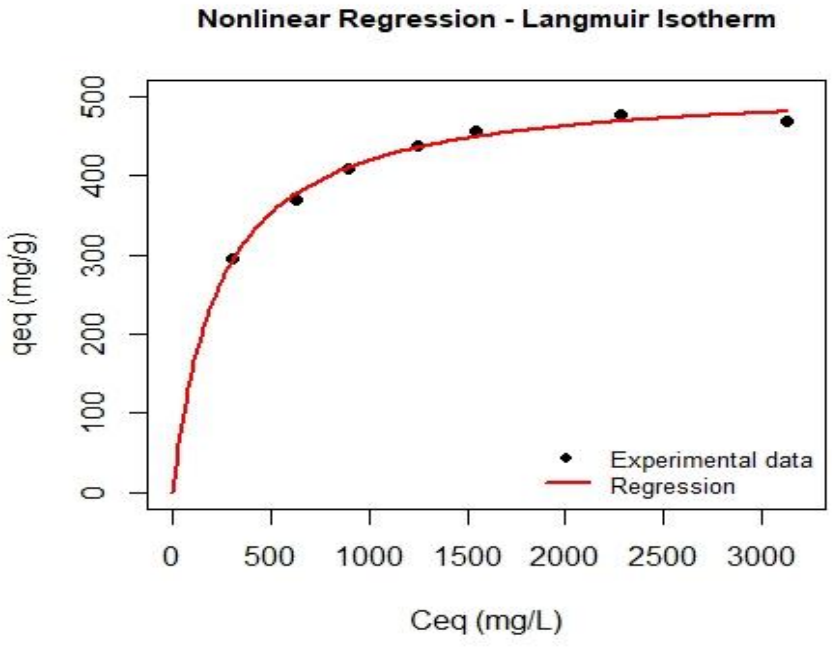

Figure 6S. *Graph obtained from nonlinear regression of an adsorption isotherm according to the Langmuir model in RStudio*

MATLAB SOFTWARE

Procedure for performing a simple nonlinear regression in Matlab software

1. On the desktop, click the Matlab shortcut button to open Matlab;
2. Copy all lines of code from this file (line 1 to the last);
3. Paste them into the Matlab command window;
4. Change the experimental x and y data;
5. To clear the workspace and command window, enter the codes below:

```
clear;
clc;
```

Template for nonlinear regression of pseudo-first order adsorption kinetics

```
% MATLAB SOFTWARE_LINES OF CODE IN MATLAB        (Line 1);
% TEMPLATE FOR NONLINEAR REGRESSION_PSEUDO FIRST-ORDER ADSORPTION KINETICS;
% Formatação em notação científica;
format short e;
NC=95;     % Confidence level (%)
% Dados
x=[3; 6; 9; 20; 40; 60; 120; 180; 300; 360; 420; 480; 540];
y=[121.25; 172.91; 242.49; 257.25; 322.04; 345.65; 370.15; 370.15; 370.15; 370.15; 370.15; 370.15; 370.15];
% Linearization of data
u=x;
v=(-log(max(y)*1.008-y));
% Terms for linearization
nd=length(u);     % number of data
Su=sum(u);
Sv=sum(v);
Suv=sum(u.*v);
Suu=sum(u.*u);
Svv=sum(v.*v);
% Linear regression coefficients
angular=(Su*Sv-nd*Suv)/(Su*Su-nd*Suu);
linear=(Su*Suv-Sv*Suu)/(Su*Su-nd*Suu);
TableData=table(x,y);     % Table with experimental data
modelFun=@(w,x)w(1)*(1-exp(-w(2)*x));       % Defining the model equation
initial_coefficients=[exp(-linear);angular];        %  initial values for the coefficients
```

```
mdl=fitnlm(TableData,modelFun,initial_coefficients);       %  Function fitnlm for nonlinear regression
% Obtenção dos coeficientes ajustados
n=mdl.NumObservations;    % Number of observations
p=mdl.NumCoefficients;    % Number of coefficients or parameters
df_reg=p-1;    % Degrees of freedom of the regression
df_err=n-p;   % Degrees of freedom of the error or residual
df_tot=n-1;    % Degrees of total freedom
SSE=mdl.SSE;    % Sum of squared errors or residuals
SST=mdl.SST;     % Sum of total squares
SSR=SST-SSE;    % Regression sum of squares
MSR=SSR/df_reg;    % Regression sum of mean squares
MSE=SSE/df_err;    % Sum of mean squares of errors or residuals;
F_statistic=MSR/MSE;    % F_statistic
F_crit=finv(1-(1-NC/100), df_reg,df_err);    % F_critic
F_sig=1-fcdf(F_statistic,df_reg,df_err);    % p_value F
w1=mdl.Coefficients{1,'Estimate'};
w2=mdl.Coefficients{2,'Estimate'};
coeff_1=mdl.Coefficients.Estimate(1);
coeff_2=mdl.Coefficients.Estimate(2);
std_err_1=mdl.Coefficients.SE(1);
std_err_2=mdl.Coefficients.SE(2);
t_stat_1=mdl.Coefficients.tStat(1);
t_stat_2=mdl.Coefficients.tStat(2);
p_value_1=mdl.Coefficients.pValue(1);
p_value_2=mdl.Coefficients.pValue(2);
conf_int=coefCI(mdl);
lower_ci=conf_int(:,1);
upper_ci=conf_int(:,2);
lower_1=lower_ci(1);
lower_2=lower_ci(2);
upper_1=upper_ci(1);
upper_2=upper_ci(2);
% Display of the values obtained with four decimal places
disp(' ');
disp('RESULTS:');
fprintf('Number of observations = %.0f\n',n);
fprintf('Number of parameters (coefficients) = %.0f\n',p);
fprintf('Confidence level = %.1f\n',NC);
```

```
fprintf('R-squared = %.5f\n',mdl.Rsquared.Ordinary);      % Visualização dos valores obtidos em notação fixa com 5 casas decimais
fprintf('Adjusted R-squared = %.5f\n',mdl.Rsquared.Adjusted);
fprintf('Standard Error (RMSE) = %.4e\n',mdl.RMSE);     % Visualização dos valores obtidos em notação científica com 4 casas decimais
disp(' ');
ANOVA=["Regression";"Residual";"Total"];
df=[df_reg;df_err;df_tot];
SS=[SSR;SSE;SST];
MS=[MSR;MSE;NaN];              % Use NaN (Not a Number) for empty cells
F=[F_statistic;NaN;NaN];
Fcrit=[F_crit;NaN;NaN];
Fsig=[F_sig;NaN;NaN];
anova_table=table(ANOVA,df,SS,MS,F,Fcrit,Fsig);
disp(anova_table);              % Display the ANOVA table
disp(' ');
REGRESSION=["w1";"w2"];
Parameters=[coeff_1;coeff_2];
Standard_error=[std_err_1;std_err_2];
t_stat=[t_stat_1;t_stat_2];
p_value=[p_value_1;p_value_2];
t_crit=tinv(1-(1-NC/100)/2,df_err);
Confidence_int=[t_crit*std_err_1;t_crit*std_err_2];
Lower_limit=[lower_1;lower_2];
Upper_limit=[upper_1;upper_2];
reg_table=table(REGRESSION,Parameters,Standard_error,t_stat,p_value,Confidence_int,Lower_limit,Upper_limit);
disp(reg_table);            % Display the regression table
% --- Viewing settings ---
% Creates a graph to compare the data and the fitted curve
fig=figure;
fig.Units='centimeters';
fig.Position=[5,5,12,7];       % 12,0 cm wide and 7,0 cm high
plot(x,y,'o','MarkerFaceColor','b','MarkerSize',5);
hold on;
title('Adsorption kinetic - Nonlinear pseudo first-order');
xlabel('time (min)');
ylabel('qt (mg/g)');
```

```
grid off;
% Plot the fitted model curve
xfit=linspace(0,max(x),200);       % If necessary, change the x values
yfit=feval(mdl,xfit);
plot(xfit,yfit,'r-','LineWidth',2);
legend('Experimental data','Regression','Location','southeast');
hold off;
% Sets the limits for the x and y axes (If you omit these command lines, the axes will be set to the data values)
% You can adjust these values as needed
xlim([0 max(x)*1.1]);    % For example from 0 to 600 for the x axis [0 600]
ylim([0 max(y)*1.1]);      % For example from 0 to 400 for the y axis [0 400]
disp(' ');
```

Table 7S. Results obtained from nonlinear regression of pseudo-first order kinetics and statistical analysis of the parameters in Matlab

Results:

Number of observations = 13

Number of parameters (coefficients) = 2

Confidence level = 95.0

R-squared = 0.92658

Adjusted R-squared = 0.91990

Standard error (RMSE) = 2.4300×10^{1}

ANOVA	df	SS	MS	*F*	*F*crit	*F*sig
"Regression"	1	8.1968×10^{4}	8.1968×10^{4}	1.3881×10^{2}	4.8443	1.4043×10^{-7}
"Residual"	11	6.4953×10^{3}	5.9049×10^{2}			
"Total"	12	8.8463×10^{4}				

Regression	Parameters	Standard_error	*t*_stat	*p*_value	Confidence int	Lower_limit	Upper_limit
"w1"	3.6015×10^{2}	8.1541	44.169	9.7815×10^{-14}	1.7947×10^{1}	3.4221×10^{2}	3.7810×10^{2}
"w2"	1.0290×10^{-1}	1.2325×10^{-2}	8.3490	4.3425×10^{-6}	2.7128×10^{-2}	7.5776×10^{-2}	1.3003×10^{-1}

ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

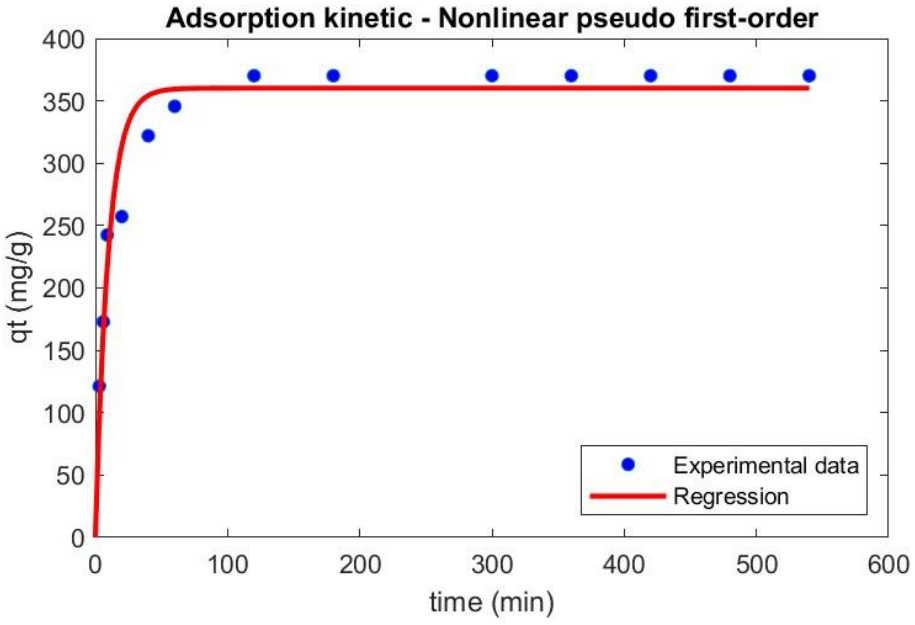

Figure 7S. *Graph obtained from nonlinear regression of pseudo-first order kinetics in Matlab*

Template for nonlinear regression of pseudo-second order adsorption kinetics

```
% MATLAB SOFTWARE_LINES OF CODE IN MATLAB        (Line 1);
% TEMPLATE FOR NONLINEAR REGRESSION_PSEUDO SECOND-ORDER ADSORPTION KINETICS;
% Formatação em notação científica;
format short e;
NC=95;     % Confidence level (%)
% Dados
x=[3; 6; 9; 20; 40; 60; 120; 180; 300; 360; 420; 480; 540];
y=[121.25; 172.91; 242.49; 257.25; 322.04; 345.65; 370.15; 370.15; 370.15; 370.15; 370.15; 370.15; 370.15];
% Linearization of data
u=x;
v=x./y;
% Terms for linearization
nd=length(u);     % number of data
Su=sum(u);
Sv=sum(v);
Suv=sum(u.*v);
Suu=sum(u.*u);
Svv=sum(v.*v);
% Linear regression coefficients
angular=(Su*Sv-nd*Suv)/(Su*Su-nd*Suu);
linear=(Su*Suv-Sv*Suu)/(Su*Su-nd*Suu);
TableData=table(x,y);     % Table with experimental data
modelFun=@(w,x)(w(1)^2*w(2)*x)./(1+w(1)*w(2)*x);       % Defining the model equation
initial_coefficients=[1./angular;angular.^2./linear];         %  initial values for the coefficients
mdl=fitnlm(TableData,modelFun,initial_coefficients);       %  Function fitnlm for nonlinear regression
% Obtenção dos coeficientes ajustados
n=mdl.NumObservations;    % Number of observations
p=mdl.NumCoefficients;    % Number of coefficients or parameters
df_reg=p-1;    % Degrees of freedom of the regression
df_err=n-p;   % Degrees of freedom of the error or residual
df_tot=n-1;    % Degrees of total freedom
SSE=mdl.SSE;    % Sum of squared errors or residuals
SST=mdl.SST;     % Sum of total squares
SSR=SST-SSE;    % Regression sum of squares
MSR=SSR/df_reg;    % Regression sum of mean squares
MSE=SSE/df_err;    % Sum of mean squares of errors or residuals;
```

```
F_statistic=MSR/MSE;    % F_statistic
F_crit=finv(1-(1-NC/100), df_reg,df_err);    % F_critic
F_sig=1-fcdf(F_statistic,df_reg,df_err);    % p_value F
w1=mdl.Coefficients{1,'Estimate'};
w2=mdl.Coefficients{2,'Estimate'};
coeff_1=mdl.Coefficients.Estimate(1);
coeff_2=mdl.Coefficients.Estimate(2);
std_err_1=mdl.Coefficients.SE(1);
std_err_2=mdl.Coefficients.SE(2);
t_stat_1=mdl.Coefficients.tStat(1);
t_stat_2=mdl.Coefficients.tStat(2);
p_value_1=mdl.Coefficients.pValue(1);
p_value_2=mdl.Coefficients.pValue(2);
conf_int=coefCI(mdl);
lower_ci=conf_int(:,1);
upper_ci=conf_int(:,2);
lower_1=lower_ci(1);
lower_2=lower_ci(2);
upper_1=upper_ci(1);
upper_2=upper_ci(2);
% Display of the values obtained with four decimal places
disp(' ');
disp('RESULTS:');
fprintf('Number of observations = %.0f\n',n);
fprintf('Number of parameters (coefficients) = %.0f\n',p);
fprintf('Confidence level = %.1f\n',NC);
fprintf('R-squared = %.5f\n',mdl.Rsquared.Ordinary);      % Visualização dos valores obtidos em notação fixa com 5 casas decimais
fprintf('Adjusted R-squared = %.5f\n',mdl.Rsquared.Adjusted);
fprintf('Standard Error (RMSE) = %.4e\n',mdl.RMSE);     % Visualização dos valores obtidos em notação científica com 4 casas decimais
disp(' ');
ANOVA=["Regression";"Residual";"Total"];
df=[df_reg;df_err;df_tot];
SS=[SSR;SSE;SST];
MS=[MSR;MSE;NaN];              % Use NaN (Not a Number) for empty cells
F=[F_statistic;NaN;NaN];
Fcrit=[F_crit;NaN;NaN];
```

```
Fsig=[F_sig;NaN;NaN];
anova_table=table(ANOVA,df,SS,MS,F,Fcrit,Fsig);
disp(anova_table);              % Display the ANOVA table
disp(' ');
REGRESSION=["w1";"w2"];
Parameters=[coeff_1;coeff_2];
Standard_error=[std_err_1;std_err_2];
t_stat=[t_stat_1;t_stat_2];
p_value=[p_value_1;p_value_2];
t_crit=tinv(1-(1-NC/100)/2,df_err);
Confidence_int=[t_crit*std_err_1;t_crit*std_err_2];
Lower_limit=[lower_1;lower_2];
Upper_limit=[upper_1;upper_2];
reg_table=table(REGRESSION,Parameters,Standard_error,t_stat,p_value,Confidence_int,Lower_limit,Upper_limit
);
disp(reg_table);            % Display the regression table
% --- Viewing settings ---
% Creates a graph to compare the data and the fitted curve
fig=figure;
fig.Units='centimeters';
fig.Position=[5,5,12,7];       % 12,0 cm wide and 7,0 cm high
plot(x,y,'o','MarkerFaceColor','b','MarkerSize',5);
hold on;
title('Adsorption kinetic - Nonlinear pseudo second-order');
xlabel('time (min)');
ylabel('qt (mg/g)');
grid off;
% Plot the fitted model curve
xfit=linspace(0,max(x),200);       % If necessary, change the x values
yfit=feval(mdl,xfit);
plot(xfit,yfit,'r-','LineWidth',2);
legend('Experimental data','Regression','Location','southeast');
hold off;
% Sets the limits for the x and y axes (If you omit these command lines, the axes will be set to the data values)
% You can adjust these values as needed
xlim([0 max(x)*1.1]);    % For example from 0 to 600 for the x axis [0 600]
ylim([0 max(y)*1.1]);      % For example from 0 to 400 for the y axis [0 400]
disp(' ');
```

Table 8S. Results obtained from nonlinear regression of pseudo-second order kinetics and statistical analysis of the parameters in Matlab

Results:

Number of observations = 13

Number of parameters (coefficients) = 2

Confidence level = 95.0

R-squared = 0.98183

Adjusted R-squared = 0.98018

Standard error (RMSE) = 1.2088×10^{1}

ANOVA	df	SS	MS	*F*	*F*crit	*F*sig
"Regression"	1	8.6856×10^{4}	8.6856×10^{4}	5.9437×10^{2}	4.8443	6.3282×10^{-11}
"Residual"	11	1.6074×10^{3}	1.4613×10^{2}			
"Total"	12	8.8463×10^{4}				

Regression	Parameters	Standard_error	*t*_stat	*p*_value	Confidence int	Lower_limit	Upper_limit
"w1"	3.7793×10^{2}	4.7016	8.0383×10^{1}	1.3756×10^{-16}	1.0348×10^{1}	3.6758×10^{2}	3.8828×10^{2}
"w2"	4.0421×10^{-4}	3.5250×10^{-5}	1.1467×10^{1}	1.8532×10^{-7}	7.7585×10^{-5}	3.2663×10^{-4}	4.8180×10^{-4}

ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

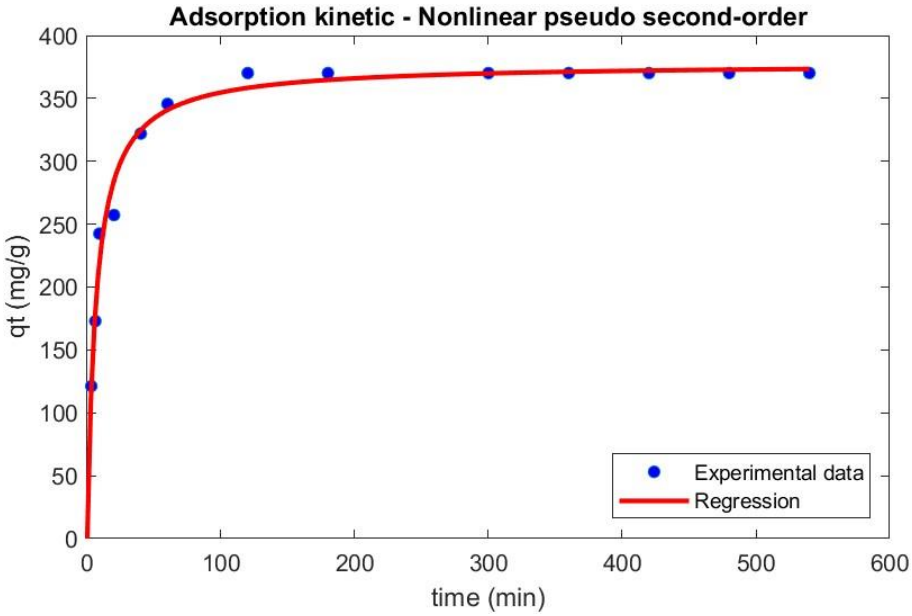

Figure 8S. *Graph obtained from the nonlinear regression of pseudo-second order kinetics in Matlab*

Template for nonlinear regression of Langmuir adsorption isotherm

```
% MATLAB SOFTWARE_LINES OF CODE IN MATLAB        (Line 1);
% TEMPLATE FOR NONLINEAR REGRESSION_LANGMUIR ADSORPTION ISOTHERM;
% Formatação em notação científica;
format short e;
NC=95;     % Confidence level (%)
% Dados
x=[306.14; 623.22; 892.78; 1252.3; 1540.6; 2277.8; 3128.2];
y=[296.17; 370.15; 408.19; 438.5; 455.83; 477.66; 468.64];
% Linearization of data
u=1./x;
v=1./y;
% Terms for linearization
nd=length(u);     % number of data
Su=sum(u);
Sv=sum(v);
Suv=sum(u.*v);
Suu=sum(u.*u);
Svv=sum(v.*v);
% Linear regression coefficients
angular=(Su*Sv-nd*Suv)/(Su*Su-nd*Suu);
linear=(Su*Suv-Sv*Suu)/(Su*Su-nd*Suu);
TableData=table(x,y);     % Table with experimental data
modelFun=@(w,x)(w(1)*w(2)*x)./(1+w(2)*x);       % Defining the model equation
initial_coefficients=[1./linear;linear./angular];         %  initial values for the coefficients
mdl=fitnlm(TableData,modelFun,initial_coefficients);       %  Function fitnlm for nonlinear regression
% Obtenção dos coeficientes ajustados
n=mdl.NumObservations;    % Number of observations
p=mdl.NumCoefficients;    % Number of coefficients or parameters
df_reg=p-1;    % Degrees of freedom of the regression
df_err=n-p;   % Degrees of freedom of the error or residual
df_tot=n-1;    % Degrees of total freedom
SSE=mdl.SSE;    % Sum of squared errors or residuals
SST=mdl.SST;     % Sum of total squares
SSR=SST-SSE;    % Regression sum of squares
MSR=SSR/df_reg;    % Regression sum of mean squares
MSE=SSE/df_err;    % Sum of mean squares of errors or residuals;
```

```
F_statistic=MSR/MSE;    % F_statistic
F_crit=finv(1-(1-NC/100), df_reg,df_err);    % F_critic
F_sig=1-fcdf(F_statistic,df_reg,df_err);    % p_value F
w1=mdl.Coefficients{1,'Estimate'};
w2=mdl.Coefficients{2,'Estimate'};
coeff_1=mdl.Coefficients.Estimate(1);
coeff_2=mdl.Coefficients.Estimate(2);
std_err_1=mdl.Coefficients.SE(1);
std_err_2=mdl.Coefficients.SE(2);
t_stat_1=mdl.Coefficients.tStat(1);
t_stat_2=mdl.Coefficients.tStat(2);
p_value_1=mdl.Coefficients.pValue(1);
p_value_2=mdl.Coefficients.pValue(2);
conf_int=coefCI(mdl);
lower_ci=conf_int(:,1);
upper_ci=conf_int(:,2);
lower_1=lower_ci(1);
lower_2=lower_ci(2);
upper_1=upper_ci(1);
upper_2=upper_ci(2);
% Display of the values obtained with four decimal places
disp(' ');
disp('RESULTS:');
fprintf('Number of observations = %.0f\n',n);
fprintf('Number of parameters (coefficients) = %.0f\n',p);
fprintf('Confidence level = %.1f\n',NC);
fprintf('R-squared = %.5f\n',mdl.Rsquared.Ordinary);      % Visualização dos valores obtidos em notação fixa com 5 casas decimais
fprintf('Adjusted R-squared = %.5f\n',mdl.Rsquared.Adjusted);
fprintf('Standard Error (RMSE) = %.4e\n',mdl.RMSE);     % Visualização dos valores obtidos em notação científica com 4 casas decimais
disp(' ');
ANOVA=["Regression";"Residual";"Total"];
df=[df_reg;df_err;df_tot];
SS=[SSR;SSE;SST];
MS=[MSR;MSE;NaN];              % Use NaN (Not a Number) for empty cells
F=[F_statistic;NaN;NaN];
Fcrit=[F_crit;NaN;NaN];
```

```
Fsig=[F_sig;NaN;NaN];
anova_table=table(ANOVA,df,SS,MS,F,Fcrit,Fsig);
disp(anova_table);              % Display the ANOVA table
disp(' ');
REGRESSION=["w1";"w2"];
Parameters=[coeff_1;coeff_2];
Standard_error=[std_err_1;std_err_2];
t_stat=[t_stat_1;t_stat_2];
p_value=[p_value_1;p_value_2];
t_crit=tinv(1-(1-NC/100)/2,df_err);
Confidence_int=[t_crit*std_err_1;t_crit*std_err_2];
Lower_limit=[lower_1;lower_2];
Upper_limit=[upper_1;upper_2];
reg_table=table(REGRESSION,Parameters,Standard_error,t_stat,p_value,Confidence_int,Lower_limit,Upper_limit
);
disp(reg_table);            % Display the regression table
% --- Viewing settings ---
% Creates a graph to compare the data and the fitted curve
fig=figure;
fig.Units='centimeters';
fig.Position=[5,5,12,7];       % 12,0 cm wide and 7,0 cm high
plot(x,y,'o','MarkerFaceColor','b','MarkerSize',5);
hold on;
title('Langmuir Adsorption isotherm');
xlabel('Ceq (mg/L)');
ylabel('qeq (mg/g)');
grid off;
% Plot the fitted model curve
xfit=linspace(0,max(x),200);       % If necessary, change the x values
yfit=feval(mdl,xfit);
plot(xfit,yfit,'r-','LineWidth',2);
legend('Experimental data','Regression','Location','southeast');
hold off;
% Sets the limits for the x and y axes (If you omit these command lines, the axes will be set to the data values)
% You can adjust these values as needed
xlim([0 max(x)*1.1]);    % For example from 0 to 3200 for the x axis [0 3200]
ylim([0 max(y)*1.1]);      % For example from 0 to 500 for the y axis [0 500]
disp(' ');
```

Table 9S. Results obtained from the nonlinear regression of an adsorption isotherm according to the Langmuir model and statistical analysis of the parameters in Matlab

Results							
Number of observations = 7							
Number of parameters (coefficients) = 2							
Confidence level = 95.0							
R-squared = 0.98686							
Adjusted R-squared = 0.98424							
Standard error (RMSE) = 8.1347							
ANOVA	df	SS	MS	F	Fcrit	Fsig	
"Regression"	1	2.4856×10^{4}	2.4856×10^{4}	3.7561×10^{2}	6.6079	6.7475×10^{-6}	
"Residual"	5	3.3087×10^{2}	6.6174×10^{1}				
"Total"	6	2.5187×10^{4}					
Regression	Parameters	Standard_error	t_stat	p_value	Confidence int	Lower_limit	Upper_limit
"w1"	5.1625×10^{2}	7.4216	6.9560×10^{1}	1.1629×10^{-8}	1.9078×10^{1}	4.9717×10^{2}	5.3532×10^{2}
"w2"	4.3280×10^{-3}	3.2833×10^{-4}	1.3182×10^{1}	4.4874×10^{-5}	8.4400×10^{-4}	3.4840×10^{-3}	5.1720×10^{-3}

ANOVA: analysis of variance; df: degrees of freedom; SS: sum of squares; MS: mean square; *F*: *F*-statistic; *F*crit: *F*-critical value; *F*sig: global significance of the model; *t*-stat: calculated *t*-value for the parameters; *p*-value: individual significance of each parameter; Confidence int: confidence interval; lower/upper conf lim: 95% confidence limit.

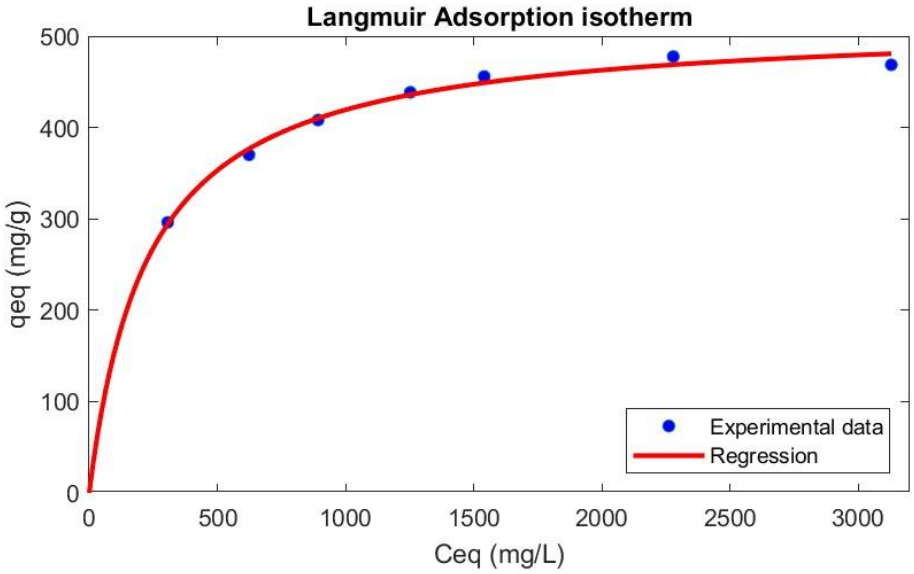

Figure 9S. *Graph obtained from the nonlinear regression of an adsorption isotherm according to the Langmuir model in Matlab*

Procedure for performing a nonlinear regression using the curve fitter tool in the apps tab of the Matlab software

The Curve Fitter app (formerly known as the cftool tool) in Matlab is a powerful and intuitive interactive graphical tool for fitting curves and surfaces to your data. Located in the apps tab of the Matlab ribbon, it simplifies the process of finding the best mathematical equation to represent the relationship between variables in a dataset.

The main function of the Curve Fitter app is to visualize and analyze the relationship between variables. You can use the tool to: fit data to models, compare models, view and analyze graphs which has statistical tools such as: R-squared (R^2), sum of squared errors (SSE), root mean square error (RMSE), and confidence intervals to help assess the goodness of fit.

Nonlinear regression of pseudo-second order adsorption kinetics in Matlab software using the curve fitter tool in the apps tab

1. Enter the experimental data in square brackets, separated by semicolons in the command window and press enter.
2. To open the curve fitter tool, simply type cftool in the command window and press enter or click the curve fitter button in the apps tab (Figure 10S).

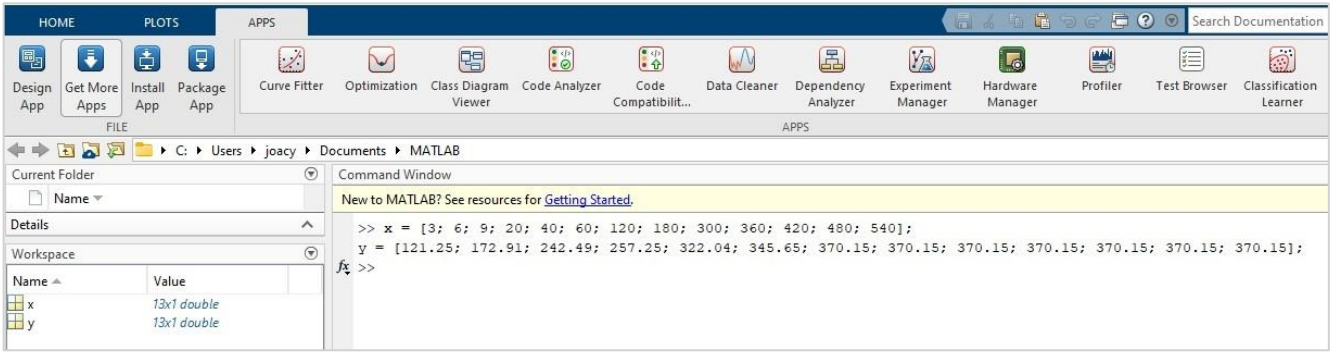

Figure 10S. *Matlab R2023b interface layout displaying the curve fitter tool in the apps tab*

3. After opening the curve fitter tool, select x data and y data for the independent and dependent variables in the dialog box that appears (Figure 11S). Experimental data can be imported from files or manually entered the command window (ideal for small datasets) or the workspace, which allows for manual editing.

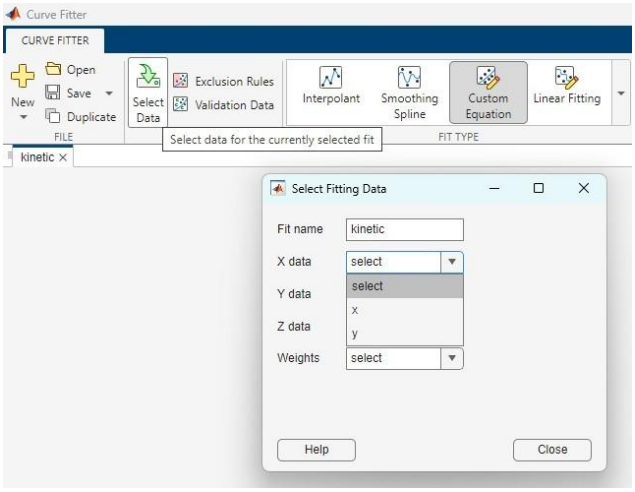

Figure 11S. *Selecting the data in Matlab*

4. Click the custom equation button in the FIT TYPE toolbox. Enter the model equation in the custom equation field within the fit options window, in this case the equation for pseudo-second order kinetics was entered. Click advanced options and enter the initial parameter values under coefficient constraints. If the iterations do not converge, upper and lower parameter constraints can be set (Figure 12S).

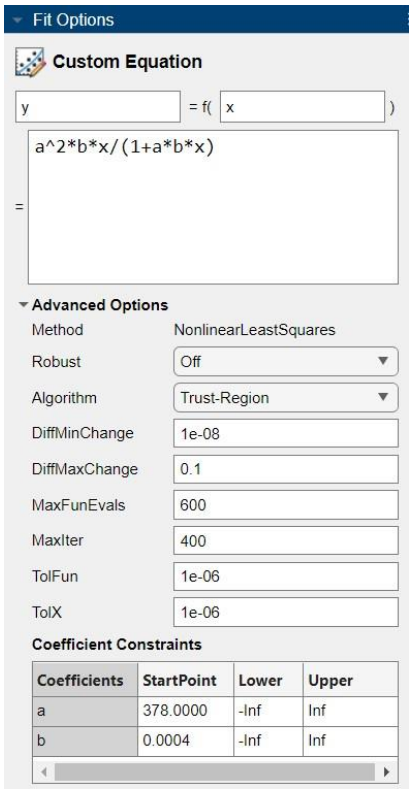

Figure 12S. *Entering the model equation and initial parameter values*

5. The adjusted parameter values and statistical data appear under results and table of fits, located on the right side and below the graph, respectively (Figure 13S).

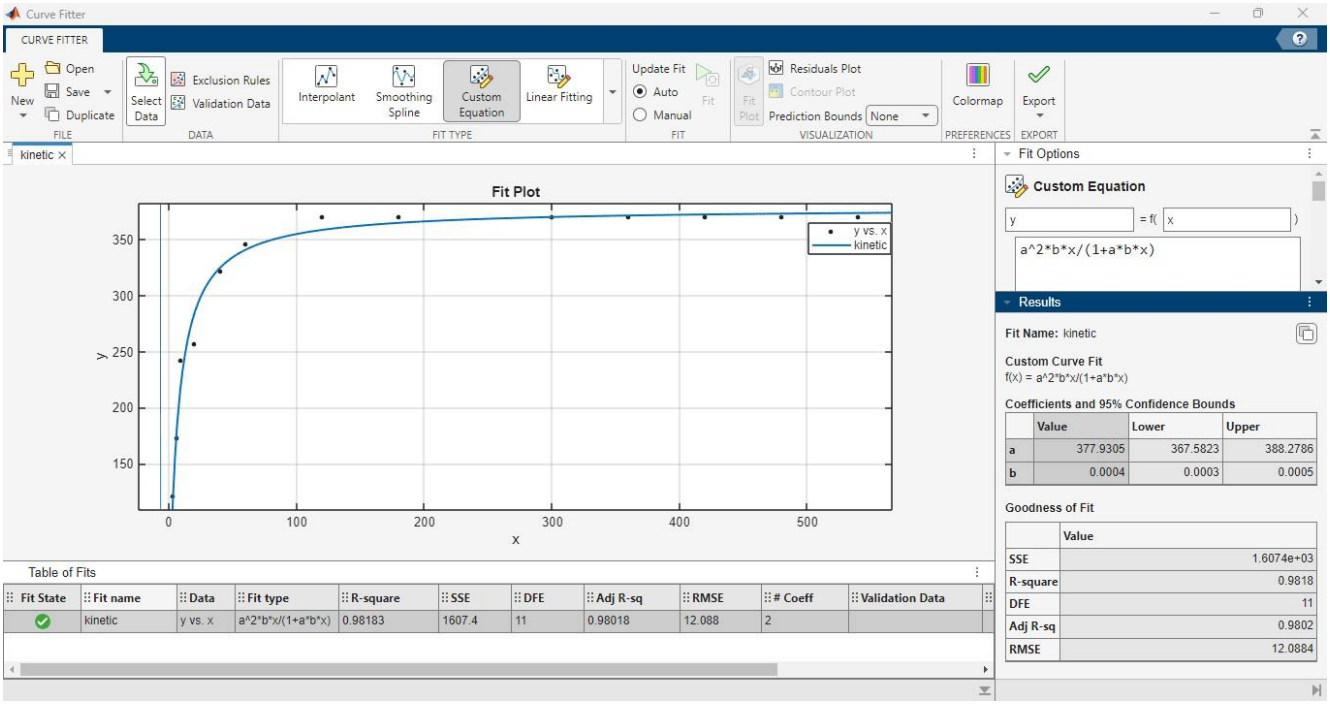

Figure 13S. *Graph and parameters from nonlinear regression of pseudo-second order kinetics in Matlab using the curve fit tool*

The values obtained of the parameters and statistical analysis are exactly the same as those obtained through the lines of code using the fitnlm function.

Nonlinear regression of Langmuir adsorption isotherm in Matlab software using the curve fitter tool in the apps tab

The procedure used to perform the nonlinear regression of the adsorption isotherm according to the Langmuir model is exactly the same as for the nonlinear regression of pseudo-second order adsorption kinetics as described in the previous item, differing only in the experimental data and the model equation. Figure 14S shows the graph and parameters from nonlinear regression of the Langmuir adsorption isotherm in Matlab software using the curve fitter tool in the apps tab.

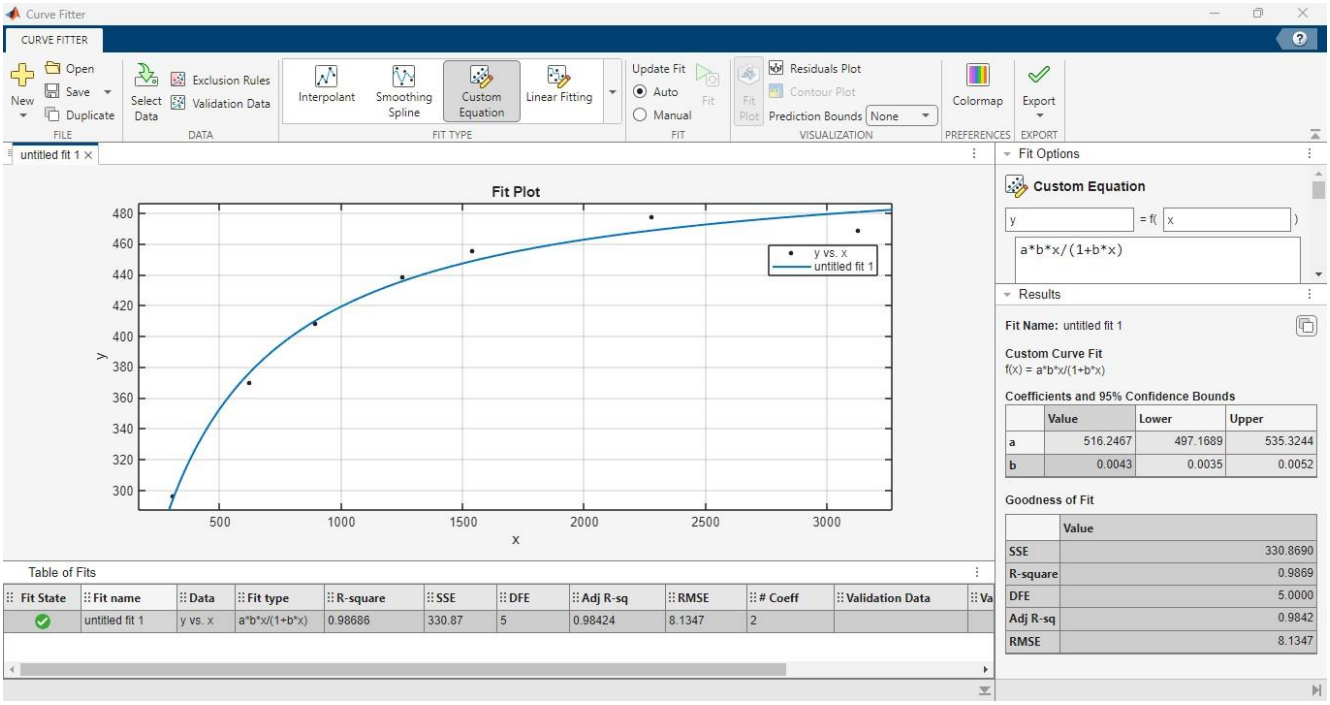

Figure 14S. *Graph and parameters from nonlinear regression of the Langmuir isotherm in Matlab using the curve fit tool*

The values obtained of the parameters and statistical analysis are exactly the same as those obtained through the lines of code using the fitnlm function.

Procedure for performing a nonlinear regression using the curve fit tool in the Origin software

The Curve Fit tool is a powerful and easy-to-use feature in Origin software designed for data analysis. Located in the analysis tab, it allows you to fit curves to data sets, which is essential in various scientific and engineering fields for modeling and predicting phenomena. The Curve Fit tool offers a variety of built-in fit models, such as: linear, polynomial, exponential, logarithmic, Gaussian, sigmoid, and Lorentzian functions among others and it also allows the creation of mathematical models applied in exact and natural sciences.

In Origin version 6.0, mathematical models of both kinetics and isotherm adsorption processes must be created.

Procedure for performing a nonlinear regression of pseudo-second order adsorption kinetics in Origin software

1. The initial step is to create and save the model equation using the Nonlinear Curve Fit tool in the analysis tab according to the desired nonlinear regression. Figure 15S shows the model equation for the pseudo-second order adsorption kinetics created in nonlinear curve fitting.

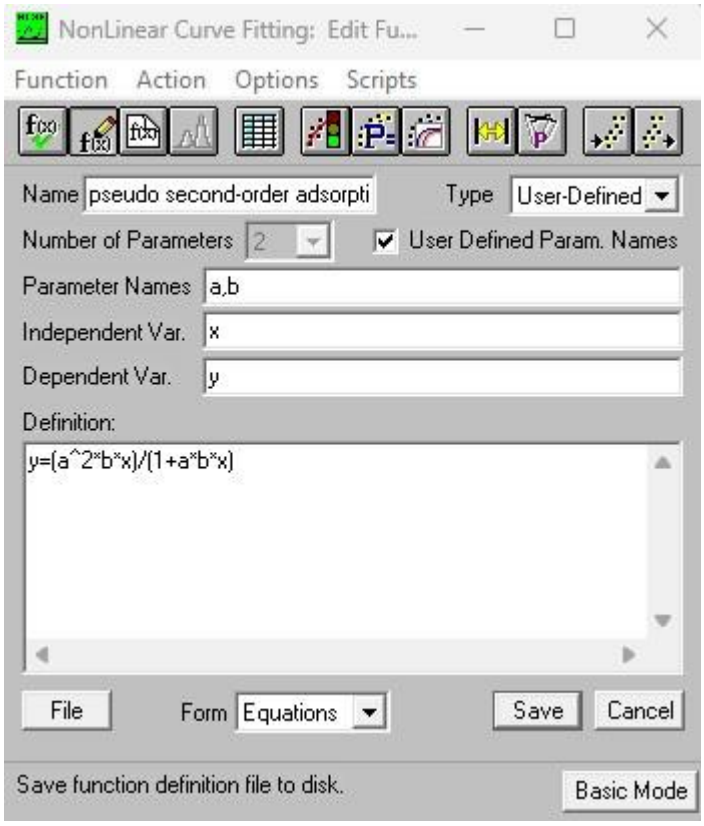

Figure 15S. *Nonlinear Curve Fitting tool in Origin version 6.0*

2. Select the experimental data in the spreadsheet and access the tool: Go to the Analysis tab in the top menu, then navigate to the fitting option and select non-linear curve fit (Figure 16S).

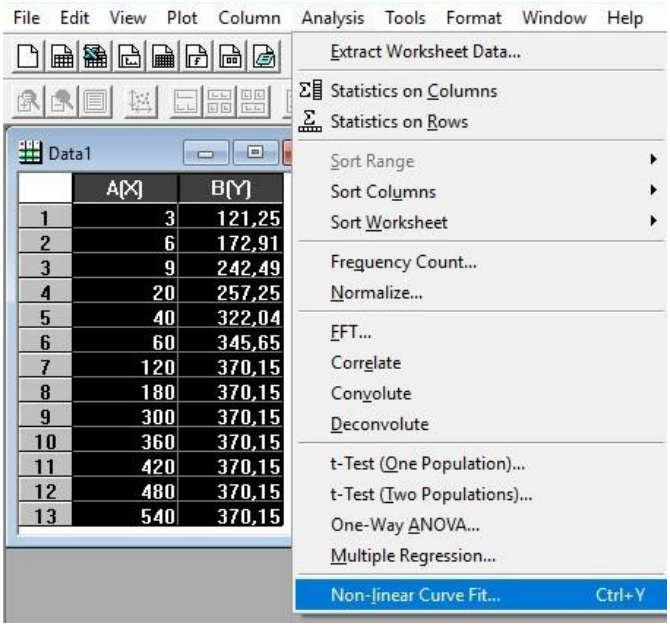

Figure 16S. *Selecting the experimental data and Non-linear Curve fit in the analysis tab*

3. Choose the model: a dialog window will open, where you can choose the appropriate fitting model (Figure 17S).

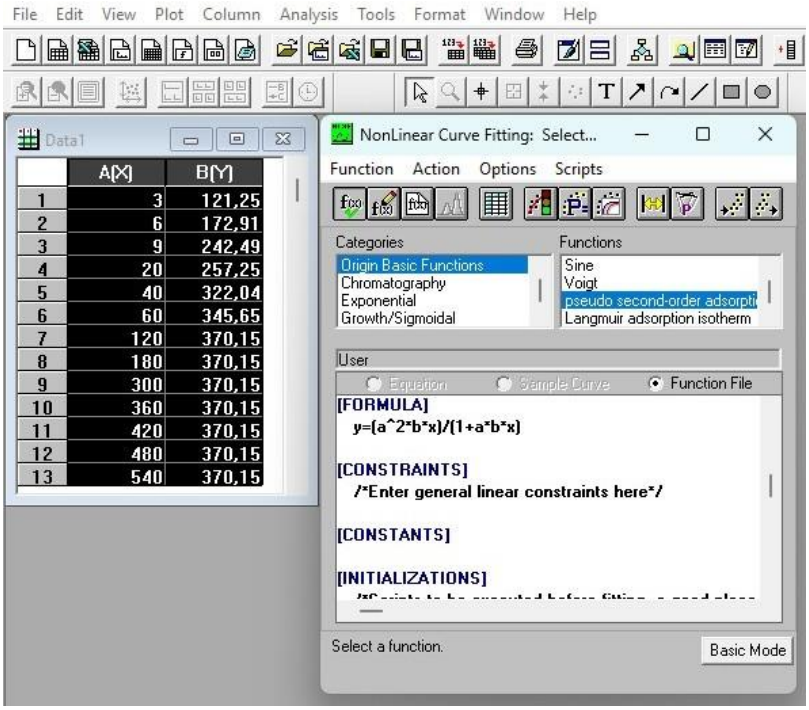

Figure 17S. *Choosing the model function*

4. In the initialization and control buttons, enter the initial parameter values, confidence level, tolerance, number of iterations, and number of significant digits.

5. Click in the fit button and then click in the appropriate iteration button. To record the parameters on the graph, click done button. The parameters and the variance-covariance matrix can also be viewed in the form of a spreadsheet (Figure 18S).

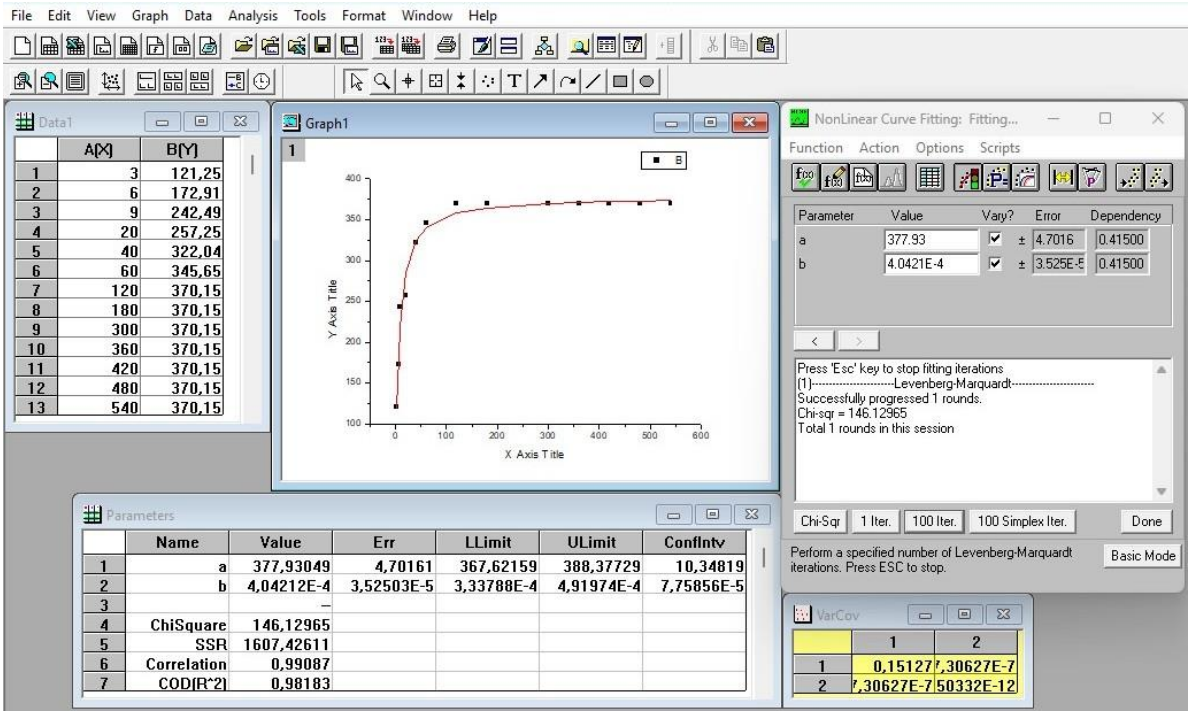

Figure 18S. *Carrying out the iteraction process*

Procedure for performing a nonlinear regression of Langmuir adsorption isotherm in Origin software

The procedure used to perform the nonlinear regression of the adsorption isotherm according to the Langmuir model is exactly the same as for the nonlinear regression of pseudo-second order adsorption kinetics as described in the previous item, differing only in the experimental data and the model equation. Figure 19S shows the graph and parameters from nonlinear regression of the Langmuir adsorption isotherm in Origin software using the non-linear curve fit tool in analysis tab.

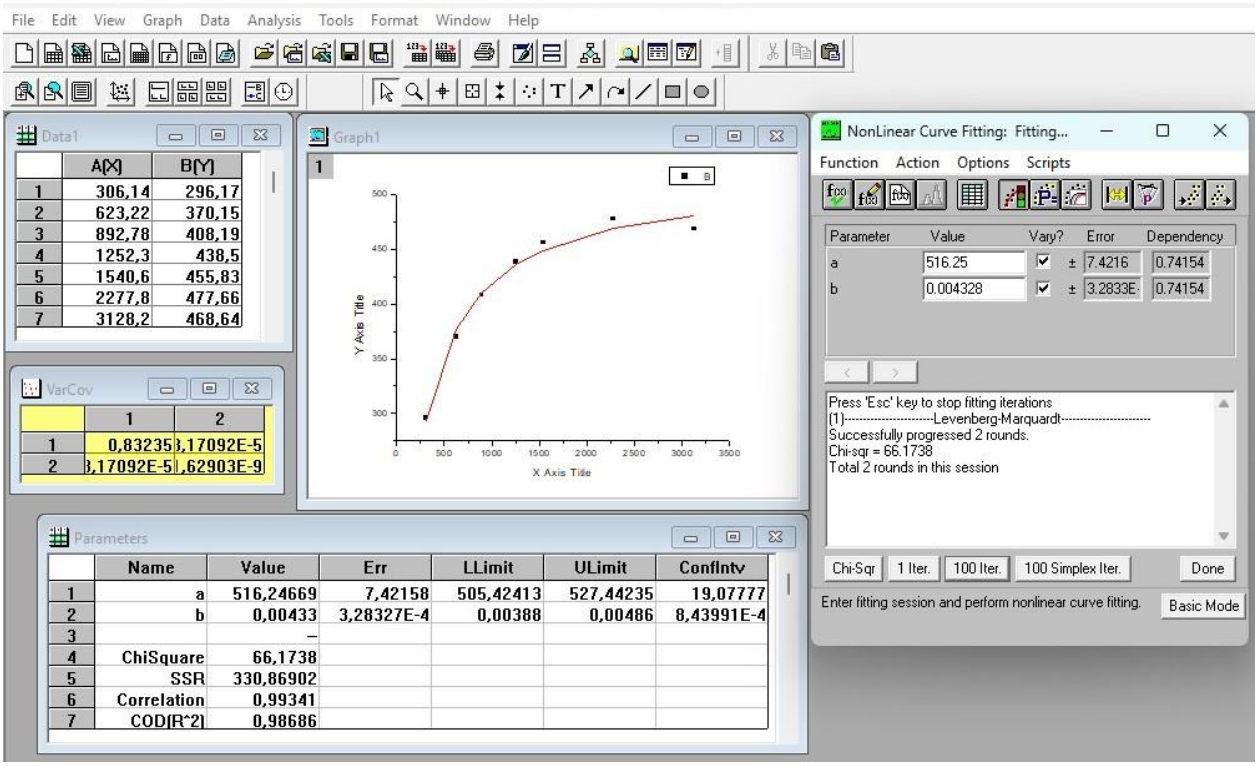

Figure 19S. *Experimental data, graph and parameters obtained from nonlinear regression of the Langmuir isotherm in Origin*